



2024

Web Almanac

HTTP Archive's annual
state of the web report



Table of Contents

Introduction

Foreword ii

Part I. Page Content

Chapter 1: JavaScript 1
Chapter 2: Markup 37
Chapter 3: Structured Data 67
Chapter 4: Fonts 117
Chapter 5: Media 167
Chapter 6: Third Parties 201

Part II. User Experience

Chapter 7: SEO 213
Chapter 8: Accessibility 277
Chapter 9: Performance 329
Chapter 10: Privacy 369
Chapter 11: Security 403

Part III. Content Publishing

Chapter 12: CMS 459
Chapter 13: Ecommerce 499
Chapter 14: Jamstack 519
Chapter 15: Sustainability 537

Part IV. Content Distribution

Chapter 16: Page Weight 609
Chapter 17: CDN 649
Chapter 18: HTTP 673
Chapter 19: Cookies 703

Appendices

Methodology..... 733

Contributors..... 743

Foreword

It feels as if a lot has changed in the world in just the last two years. Some changes are negative, such as the increase in regional conflicts and the broader tension they raise globally. Some changes are positive though, such as putting the COVID-19 pandemic mostly behind us and important growth in sustainable energy generation.

In just about every corner of the earth however, it's likely that the web contributes to your quality of life in some way. Has the web gotten better over the last couple of years? How should we feel about its health?

The 2024 Web Almanac is here to provide you data to help answer that question. After a hiatus in 2023, a team of passionate researchers have stepped forward this year to investigate important aspects of the modern web using data. Like most technologies, the web grows more complex over time and the Web Almanac help us make sense of things, including a new chapter on Cookies this year.

This year marks a significant step in evolving the Web Almanac towards a more community-driven project, inspired by the organizational models of academic conferences. Previously maintained by the dedicated HTTP Archive team, the 2024 Almanac introduces a formal organizing committee to foster broader collaboration and inclusivity. Key roles, such as General Chair, Program Committee Chair, and Event Chair, have been established to streamline responsibilities and encourage participation from diverse contributors. This distributed leadership structure is designed to make the Web Almanac more community-driven, which in turn aims to enhance its sustainability by inviting a wider range of voices and perspectives, making the Web Almanac a more collaborative resource for the web community. We hope that this shift towards a community-driven model will strengthen the Web Almanac's foundation and lead to many future editions.

It's been great to experience the motivation and inclusion from the community—spanning young doctoral researchers to senior experts worldwide. We sincerely thank each of our contributors; it is their hard work that makes this open-source project possible.

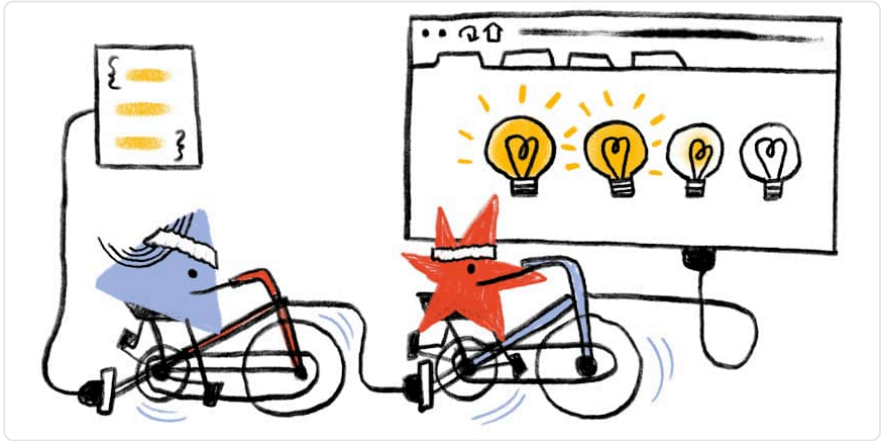
As an engine of prosperity, it's important that we understand what's working well on the web and where it needs more support. We encourage you to explore, debate, and share the details in the 2024 Web Almanac so it can better serve us all. We are all in this together.

—*Nurullah Demir, 2024 Web Almanac General Co-Chair*

—*Caleb Queern, 2024 Web Almanac General Co-Chair*

Part I Chapter 1

JavaScript



Written by Abdul Haddi Amjad and Nishu Goel

Reviewed by Barry Pollard

Analyzed by Onur Güler and Nurullah Demir

Edited by Barry Pollard

Introduction

JavaScript is essential for creating interactive web experiences, driving everything from basic animations to advanced functionalities. Its development has significantly enhanced the web's dynamic capabilities.

However, this heavy dependence on JavaScript involves compromises. Each stage—from downloading and parsing to execution—demands substantial browser resources. Using too little can compromise user experience and business objectives while overusing it can lead to sluggish load times, unresponsive pages, and poor user engagement.

In this chapter, we will re-evaluate JavaScript's role on the web and offer recommendations for designing smooth, efficient user experiences.

How much JavaScript do we load?

We will analyze the volume of JavaScript being deployed by web developers. Gaining a clear picture of the current state is crucial for driving any impactful improvements.

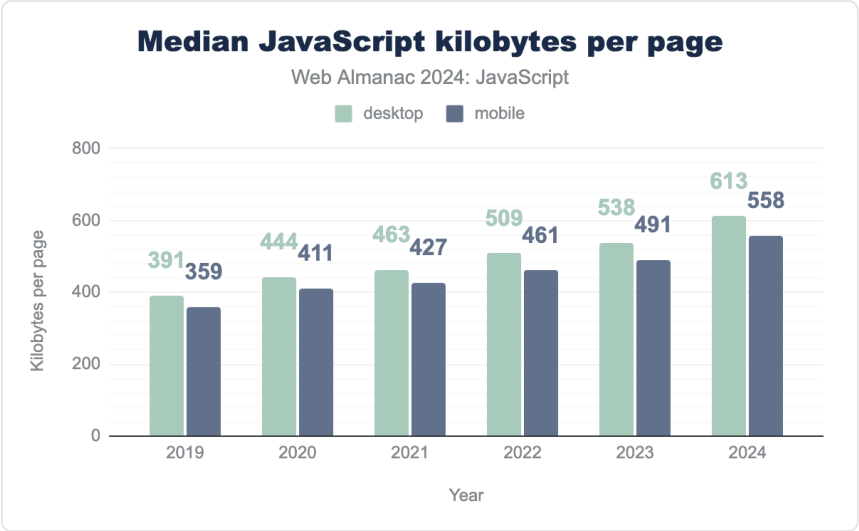


Figure 1.1. Median JavaScript kilobytes per page.

There has been a continuous increase in the volume of JavaScript. In 2024, this upward trend resumed, with the median JavaScript payload rising by 14%, reaching 558 kilobytes on mobile and 613 kilobytes on desktop. This ongoing trend is concerning. While device capabilities are improving, not everyone has access to the latest technology. Larger JavaScript bundles place additional strain on device resources, impacting performance, especially for users with older or less powerful hardware.

How many JavaScript requests per page?

Every resource on a web page sparks at least one request, but it can snowball quickly if that resource starts pulling in others.

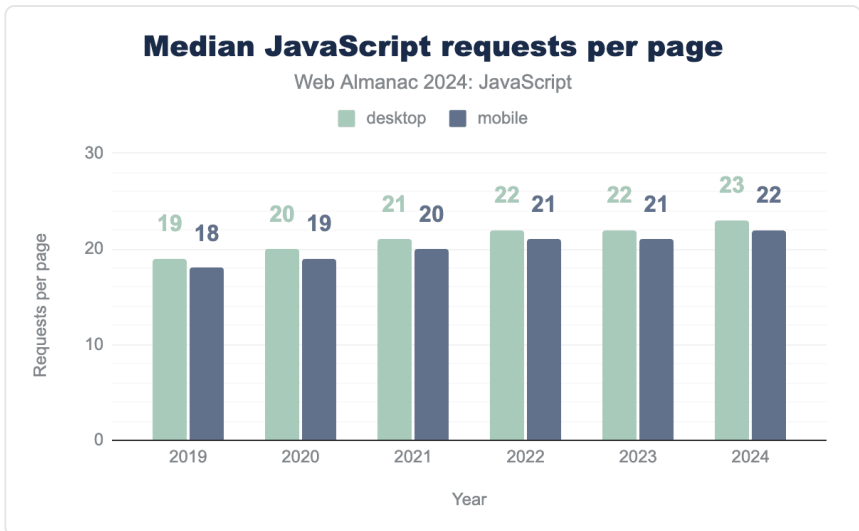


Figure 1.2. Median JavaScript requests per page.

With script requests, the stakes are higher. The more requests you send, the more JavaScript you load—and the more likely you'll trigger a bottleneck as these scripts compete for attention on the main thread. This battle for resources can grind performance to a crawl, delaying startup and leaving users waiting.

In 2024, the median mobile page is making 22 JavaScript requests, while those in the 90th percentile soar to 68. This represents a subtle yet notable increase of one request at the median and four at the 90th percentile compared to last year.

On the desktop front, the story is similar: the median jumps to 23 JavaScript requests, with the 90th percentile climbing to 70 requests. Again, we see an increase of one request at the median and five at the 90th percentile—echoing the trends we observe in mobile.

While these increases might appear modest at first glance, they signal a continuing evolution in web-behavior. Since the Web Almanac's inception in 2019, we've witnessed a steady rise in JavaScript requests, hinting at a future where growth may substantially outpace performance improvements to counteract this. What will this mean for developers and users alike as we navigate the complexities of an increasingly JavaScript-heavy web?

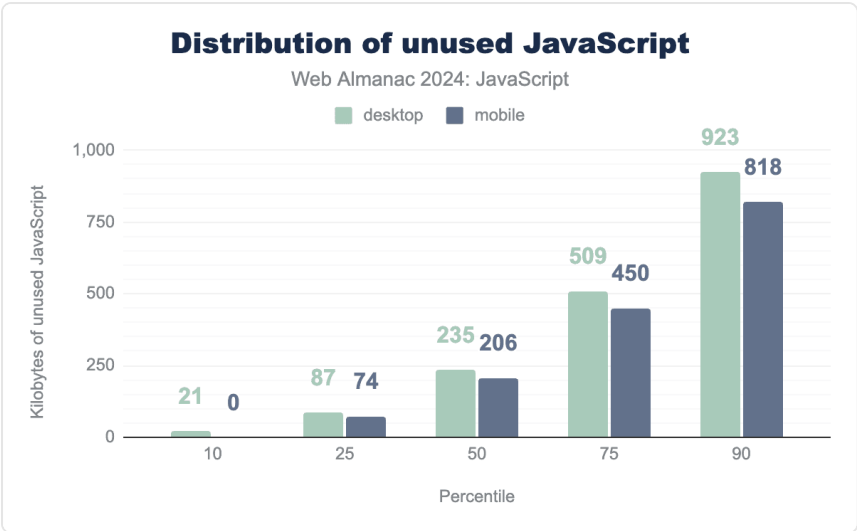


Figure 1.3. Distribution of unused JavaScript.

Along with this increase in JavaScript, we see an increase in unused bytes of JavaScript with approximately half the bytes downloaded being unused during page load (206 kilobytes—or 44% of bytes delivered—at the media on mobile).

How is JavaScript packaged processed?

JavaScript bundling and transpiling have transformed web development by optimizing how applications are built and delivered. Bundlers like webpack and Parcel package multiple files into a single bundle, reducing HTTP requests and improving loading times. Transpilers like Babel allow developers to use modern JavaScript features while ensuring compatibility across various browsers. However, managing these processes is crucial to avoid larger payloads that can hinder performance. Ultimately, they strike a balance between innovation and user experience, ensuring powerful yet efficient applications.

Bundlers

JavaScript bundlers, like webpack and Parcel, package multiple JavaScript files into a single bundle to streamline delivery to users. They analyze the code and its dependencies, optimizing the final output to reduce the number of HTTP requests. By combining files, bundlers can improve loading times and performance. However, these tools can sometimes unintentionally tangle functional code with user tracking scripts, complicating performance and privacy

considerations.

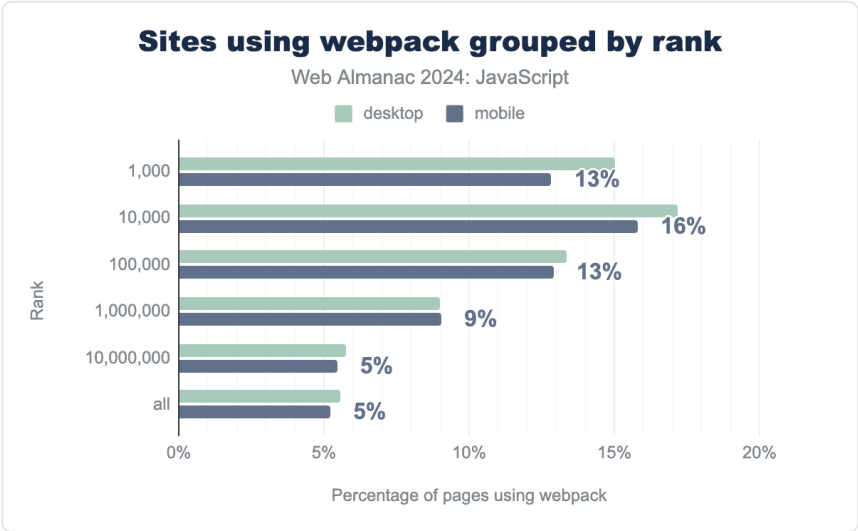


Figure 1.4. Sites using webpack grouped by rank.

While webpack usage remains at a steady 5% across websites, recent trends reveal a decline in its adoption among the top 1,000 sites, both on mobile and desktop. Although 5% may seem modest, it still represents a substantial number of Web Almanac’s sites.

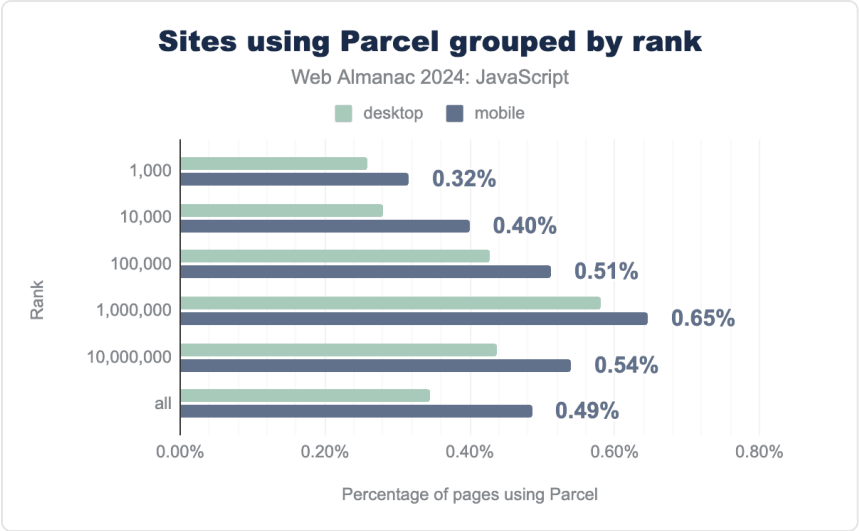


Figure 1.5. Sites using Parcel grouped by rank.

Parcel stands as the second most popular alternative to webpack, boasting a notable adoption rate among developers. However, recent trends indicate a decline in its usage, dropping from 1.3% of mobile websites last year to just 0.3% this year. A similar pattern emerges on desktop platforms, reflecting a shift in the landscape of JavaScript bundlers.

Transpilers

In the 2022 Web Almanac, year we looked at transpilers as a percentage of those sites with available source maps¹. This year, we've changed to the percentage of overall sites. This methodology change means we're moving from a likely over counting of sites (sites using sourcemaps are, by definition, more likely to more complex web applications than need transpilation), to an under counting (as not all sites publish public source maps).

1. <https://almanac.httparchive.org/en/2022/javascript#transpilers>

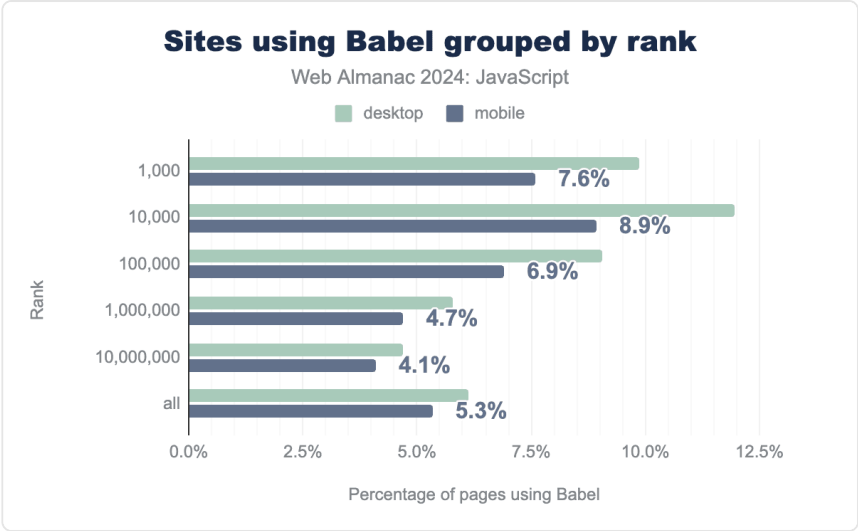


Figure 1.6. Sites using Babel grouped by rank.

Babel is particularly prevalent among higher-ranked websites, with 12% of the top 10,000 mobile sites using Babel (and this is 23% - 38% of sites using source maps, so similar to last year’s Babel results²). Mobile sites consistently show higher adoption rates than desktop sites, regardless of rank. These trends highlight Babel’s prominence, especially among top-tier and mobile-optimized sites, indicating its growing importance in modern web development.

How often is TypeScript used?

TypeScript is a superset of JavaScript that adds static types, which help catch errors during development and make the code more maintainable. These tools streamline the development process and ensure cross-browser compatibility.

2. <https://almanac.httparchive.org/en/2022/javascript#fig-6>

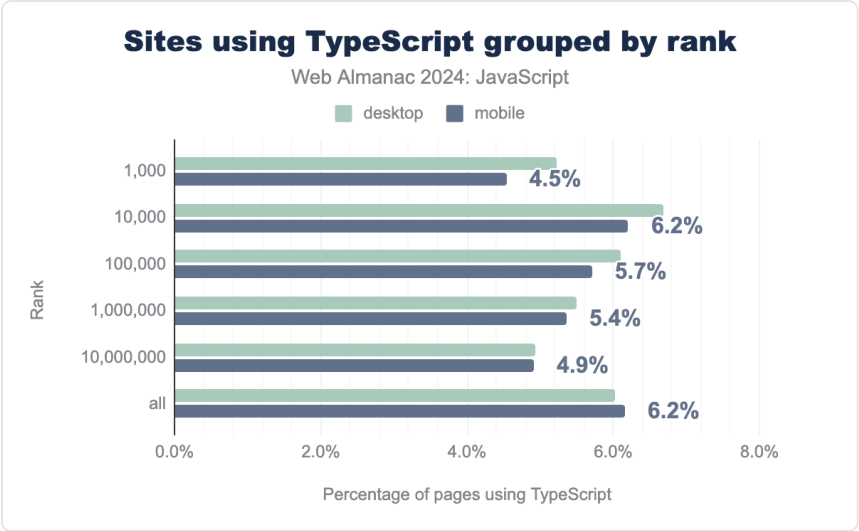


Figure 1.7. Sites using TypeScript grouped by rank.

Across all ranks, around 6% of pages use TypeScript, with mobile generally having slightly higher adoption. Again, this only counts sites publishing sourcemaps. The usage is fairly consistent, though top-ranked pages (1,000) show slightly lower TypeScript adoption compared to lower-ranked ones.

How is JavaScript requested?

In the fast-paced world of web development, the way JavaScript is loaded can make or break a site's performance. From synchronous loading that can slow page rendering to asynchronous techniques that boost speed, developers have a range of options at their disposal. The challenge lies in balancing the power of JavaScript's interactivity with the need for swift, seamless user experiences. By mastering optimal loading strategies, web creators can significantly enhance their sites' responsiveness and user satisfaction.

`async`, `defer`, `module`, and `nomodule`

When optimizing JavaScript loading, developers have several powerful attributes at their disposal.

The `async` attribute allows scripts to load asynchronously while HTML parsing continues, executing them as soon as they're available. In contrast, `defer` postpones script execution

until after HTML parsing is complete, maintaining the order of deferred scripts.

For modern web applications, the `module` attribute indicates that a script is a JavaScript module, enabling ES6 import/export syntax and strict mode by default. Complementing this, the `nomodule` attribute specifies fallback scripts for browsers that don't support ES6 modules, ensuring broader compatibility while allowing modern browsers to ignore these fallbacks. By strategically employing these attributes, developers can fine-tune script loading behavior to optimize page performance and user experience.

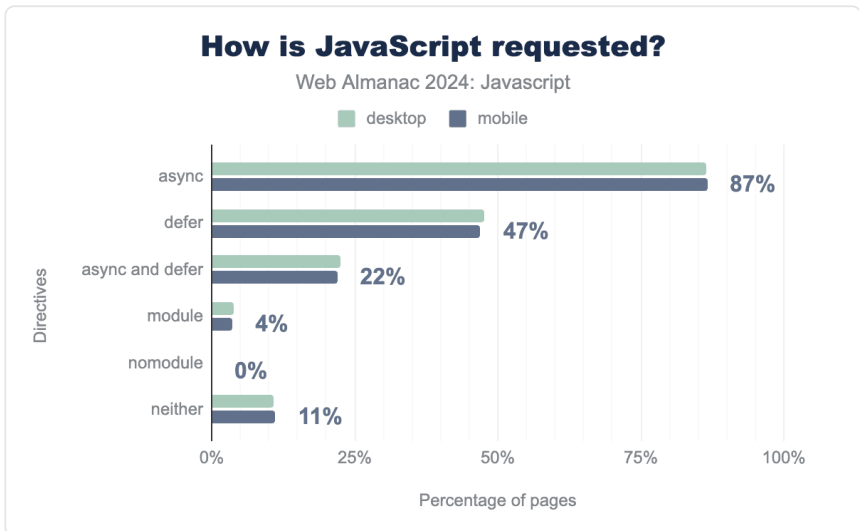


Figure 1.8. How is JavaScript requested?

Comparing JavaScript loading trends from the last Web Almanac 2022³ to 2024 reveals notable shifts in developer practices. The use of the `async` attribute has increased significantly, from 76% to 87% of pages on both desktop and mobile. The `defer` attribute usage has seen a modest increase from 42% to 47%. The combination of `async` and `defer` attributes has decreased slightly from 28-29% to 22%, possibly due to developers choosing one method over the other.

`module` usage remains low at 4%, while `nomodule` shows close to zero adoption, indicating that modern JavaScript module systems are still not widely implemented across the web.

3. <https://almanac.httparchive.org/en/javascript/#async-defer-module-and-nomodule>

Attribute	2022	2024	% change
async	47.2%	49.5%	4.8%
defer	9.1%	13.0%	43.3%
async and defer	3.1%	3.0%	-3.0%
module	0.4%	1.2%	208.8%
nomodule	0.0%	0.0%	N/A

Figure 1.9. JavaScript requests by script (mobile).

Comparing the trends on `<script>` tags from 2022 to 2024, the overall use of `async` in scripts saw a 4.8% increase, maintaining its dominance. The `defer` attribute usage, however, experienced a notable increase, where it climbed from 9.1% to 13.0% on mobile in 2024. The combination of `async` and `defer` saw a slight decrease. The `module` attribute saw a tripling in used but continues to have very low adoption across both desktop and mobile platforms.

preload, prefetch, and modulepreload

Resource hints play a crucial role in optimizing browser performance by indicating which resources should be fetched early. Preload is used to fetch resources required for the current navigation, ensuring that critical assets are available as soon as they are needed. Modulepreload serves a similar purpose but specifically for preloading JavaScript modules, helping to load modular scripts efficiently. Prefetch, on the other hand, is designed for resources that will be needed in the next navigation, allowing the browser to anticipate and prepare for future page transitions.

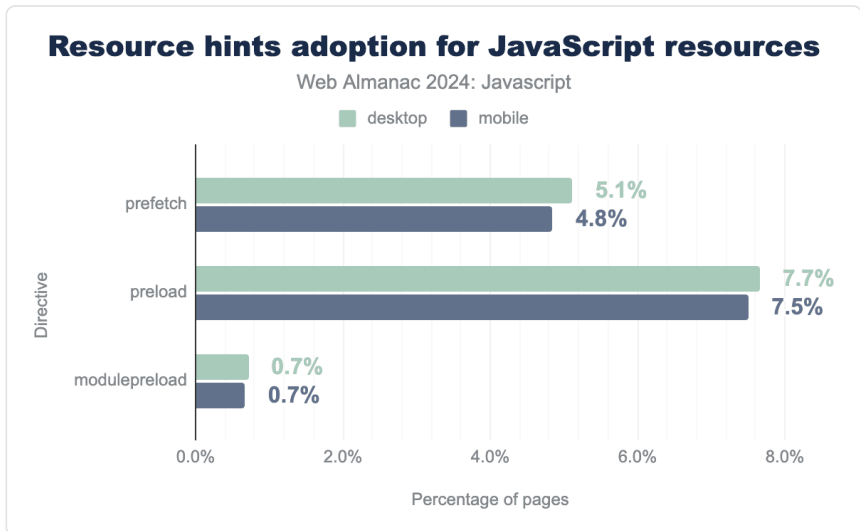


Figure 1.10. Resource hints adoption for JavaScript resources.

In comparing the trends between 2022 and 2024 for resource hints adoption, `preload` usage dropped significantly from 16.4% on desktop in 2022⁴ to 7.5% overall in 2024. `prefetch` adoption increased considerably from around 1.0% in 2022 to 4.8% overall in 2024. `modulepreload` usage stayed very minimal across both years, hovering around 0.1% in 2022 and showing a similar low percentage of 0.7% in 2024.

4. <https://almanac.httparchive.org/en/2022/javascript#preload-prefetch-and-modulepreload>

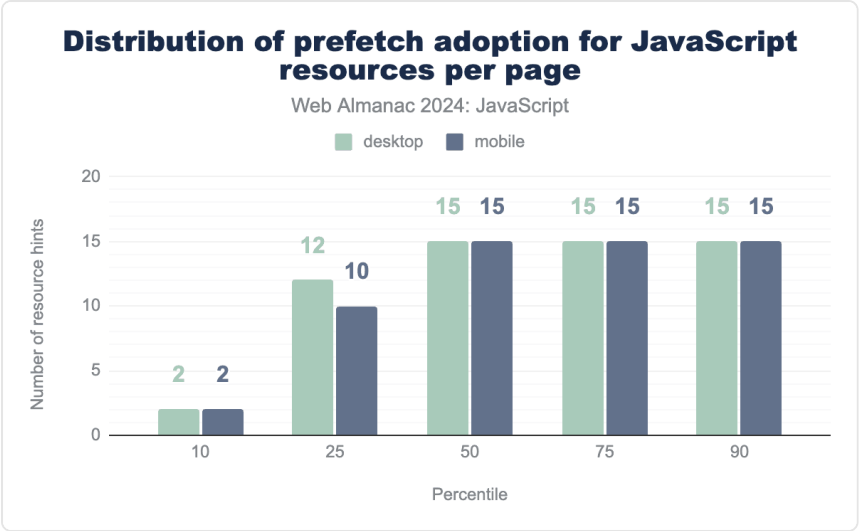


Figure 1.11. Distribution of `prefetch` adoption for JavaScript resources per page.

The distribution of `prefetch` adoption for JavaScript resources (among pages that use `prefetch`) shows that the number of prefetch hints peaks at 15 prefetch hints from the median to 90th percentile.

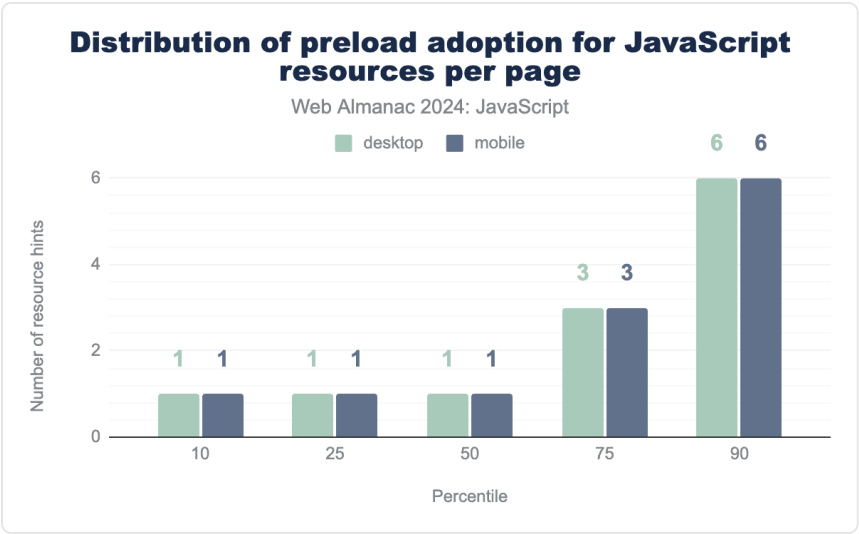


Figure 1.12. Distribution of `preLoad` adoption for JavaScript resources per page.

Among pages that use `preload`, there is less usage with only one `preload` at the median, increasing to six at the 90th percentile.

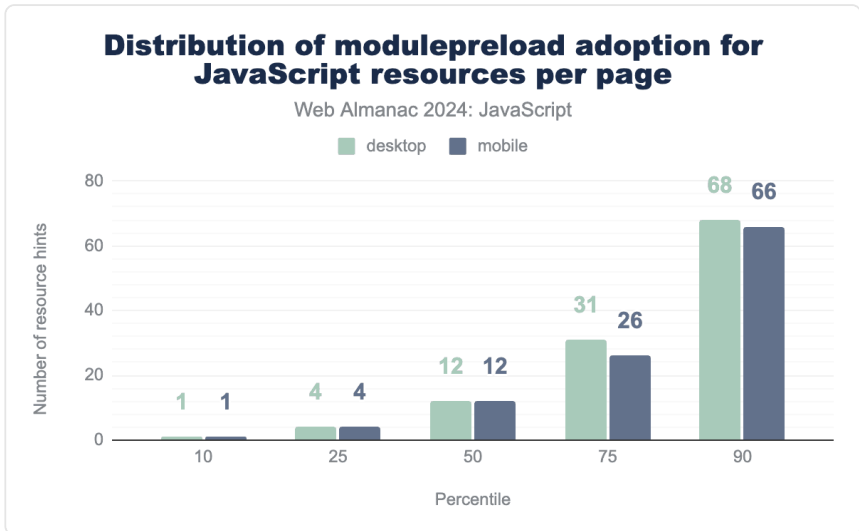


Figure 1.13. Distribution of `modulepreload` adoption for JavaScript resources per page.

`modulepreload` usage is much more varied, but with low usage this is easily skewed by a few sites.

Injected scripts

Script injection involves creating an `HTMLScriptElement` using `document.createElement` and adding it to the DOM via a DOM insertion method, or injecting `<script>` markup as a string using `innerHTML`. While common in many use cases, this practice bypasses the browser's preload scanner, making the script undetectable during the initial HTML parsing. This can negatively impact performance metrics like Largest Contentful Paint (LCP)⁵, especially if the injected script triggers long tasks or parses large amounts of markup dynamically.

5. <https://web.dev/articles/lcp>

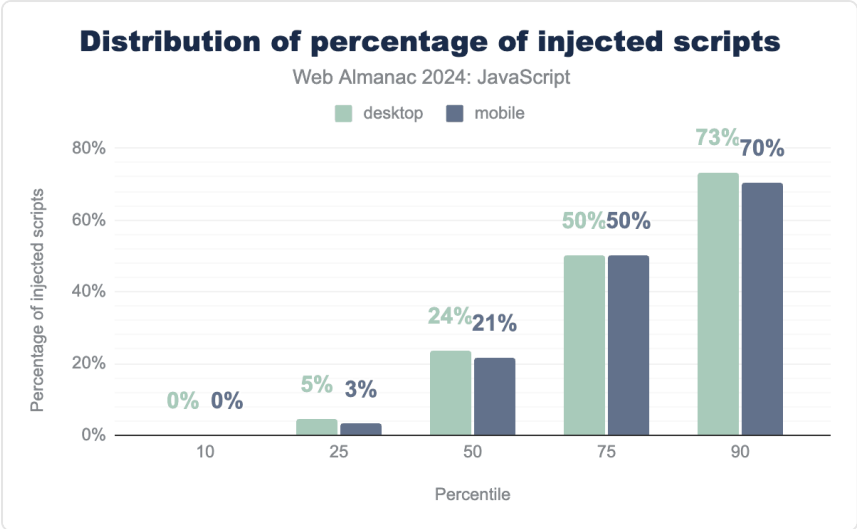


Figure 1.14. Distribution of percentage of injected scripts.

Comparing the two distributions of injected scripts, the 2024 graph shows a notable decrease in the percentage of injected scripts at the 50th percentile, rising from 25% in 2022 to 21% in 2024. At higher percentiles, the trend remains consistent between the two years, with 70% of scripts being injected at the 90th percentile in both 2022 and 2024. The early percentiles saw a slight uptick in injection, but overall, the pattern of script injection has remained steady at higher resource levels.

First-party versus third-party JavaScript

First-party JavaScript is code that is directly served by and belongs to the website's domain, playing a key role in the site's functionality and user experience. In contrast, third-party JavaScript comes from external domains and is typically used for services like analytics, ads, or social media integrations. While first-party scripts have direct control and transparency, third-party scripts can introduce performance, security, and privacy risks. Managing the balance between these two types is crucial for optimizing site performance and safeguarding user data. In this section, we'll explore the distribution of first-party and third-party code and assess how modern websites are splitting their JavaScript loads across different sources.

Requests

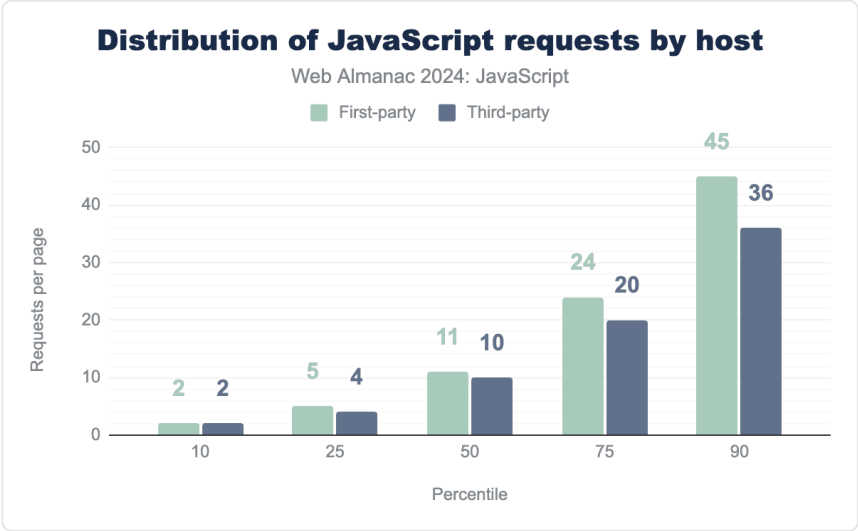


Figure 1.15. Distribution of JavaScript requests by host.

Comparing the two trends, the 2024 graph shows an increase in third-party JavaScript requests compared to 2022⁶, particularly at the 90th percentile, where third-party requests grew from 34 in 2022 to 36 in 2024. First-party requests also increased slightly, but the rise in third-party scripts is more pronounced. This increase in third-party JavaScript is concerning, as it can negatively impact performance, introduce security vulnerabilities, and pose privacy risks for users due to the lack of direct control over external scripts

6. <https://almanac.httparchive.org/en/2022/javascript#first-party-versus-third-party-javascript>

Bytes

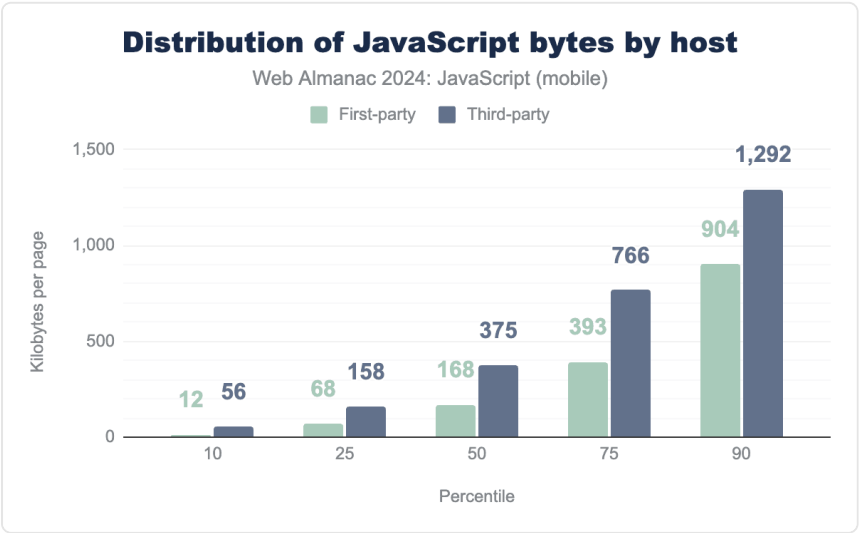


Figure 1.16. Distribution of JavaScript bytes by host.

The amount of bytes downloaded, shows simialr increases to the number of requests.

Dynamic Imports

Dynamic `import()` offers a more flexible alternative to the traditional static import syntax, allowing it to be called from anywhere within a script, unlike static imports that are restricted to the top of a JavaScript file.

By deferring the loading of non-essential code until it's actually needed, dynamic imports can significantly enhance startup performance, reducing the initial load and boosting overall efficiency.

3.70%

Figure 1.17. The percentage of mobile pages using dynamic `import()`.

The jump in the usage of dynamic imports on mobile pages from 0.34% to 3.70% signifies a growing adoption of this technique for performance optimization. This sharp increase highlights how developers are increasingly leveraging dynamic imports to improve load times

and reduce the upfront JavaScript payload on mobile devices. By deferring non-critical scripts, websites can enhance both performance and user experience, particularly on resource-constrained mobile environments. The rise reflects an industry shift towards more efficient and on-demand loading strategies for better mobile performance.

Web workers

Web workers are a powerful web platform feature designed to alleviate the load on the main thread by running JavaScript in the background on a separate thread. Unlike traditional scripts, web workers operate independently without direct access to the DOM, allowing them to handle intensive tasks—such as data processing or complex calculations—without affecting the UI's responsiveness. By offloading these resource-heavy operations, web workers ensure smoother performance and prevent the main thread from becoming overwhelmed, making them essential for delivering faster, more efficient web experiences.



Figure 1.18. The number of mobile pages using web workers.

The increase in mobile pages utilizing web workers from 12% to 30% marks a significant shift in the adoption of this technology. This nearly threefold rise highlights how developers are increasingly leveraging web workers to offload intensive tasks, improving performance and ensuring smoother user experiences on mobile devices. With nearly a third of mobile pages now incorporating web workers, it reflects a growing recognition of the importance of keeping the main thread free for critical UI updates, ultimately driving more responsive and efficient mobile interactions.

Worklets

Worklets are a specialized class of web workers designed to provide low-level access to rendering pipelines for tasks like painting and audio processing. A key performance advantage of worklets is their ability to run independently on separate threads, offloading resource-intensive tasks from the main thread. This not only boosts efficiency but also enhances performance by keeping the main thread free for essential operations, leading to smoother visuals and seamless audio experiences.

0.0016%

Figure 1.19. The percentage of mobile pages that register at least one paint worklet.

0.0004%

Figure 1.20. The percentage of mobile pages that register at least one audio worklet worklet.

The adoption of worklets on mobile devices remains extremely low, with paint worklets slightly higher at 0.0016%, a slight increase from the previous Web Almanac, and audio worklets used on just 0.0004% of mobile pages. These figures suggest that despite the potential benefits worklets offer for offloading tasks like audio processing and rendering, their usage is still far from mainstream in the mobile web development space. This could be due to their specialized functionality and the need for more widespread browser support and developer familiarity.

How is JavaScript delivered?

The way JavaScript resources are delivered to browsers continues to be a critical aspect of web performance, with compression playing a significant role in reducing payload sizes and improving load times. Let's examine how JavaScript resources are being compressed and delivered across the web in 2024.

Compression Methods

When JavaScript resources are delivered over the network, they can be compressed to reduce their transfer size. Compression is a crucial optimization technique that helps reduce bandwidth usage and improve page load times. Several compression algorithms used for JavaScript delivery include brotli (br), gzip, zstd.

While brotli offers better compression ratios than gzip, especially for text resources like JavaScript, gzip has been a web standard for many years with its broader browser support and fast compression speeds. Zstandard (zstd) is a newer compression algorithm developed by Facebook that aims to provide high compression ratios with fast compression and decompression speeds. Despite its promising capabilities, our data shows minimal adoption at just 1% of requests in 2024.

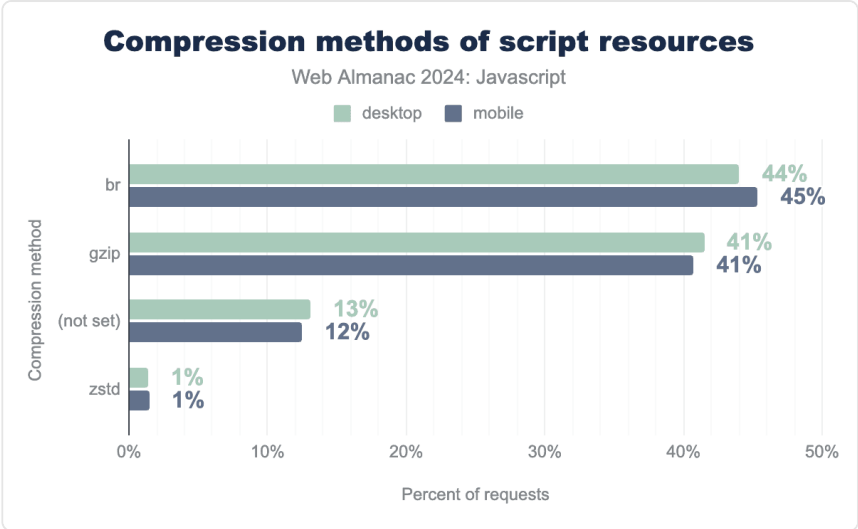


Figure 1.21. Compression methods of script resources.

Turns out 2024 marks a pivotal shift in JavaScript compression trends, with Brotli (`br`) finally overtaking gzip as the most prevalent compression method. Brotli now commands 45% of mobile and 44% of desktop JavaScript requests, compared to gzip's 41% across both platforms. This is a remarkable transformation from 2022, when gzip led with 52% compared to Brotli's 34%, and an even more dramatic change from 2021's numbers (gzip: 55%, Brotli: 30.8%).

This Brotli ascendancy represents a major win for web performance, as Brotli typically achieves better compression ratios than gzip, particularly for JavaScript resources. The steady year-over-year growth in Brotli adoption (30.8% → 34% → 45%) suggests a growing recognition of its benefits among web developers and hosting providers.

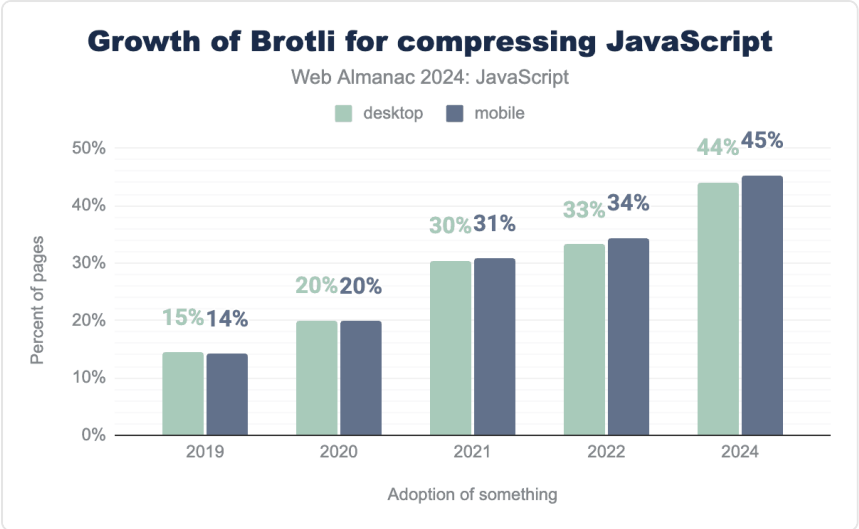


Figure 1.22. Growth of Brotli for compressing JavaScript.

The remaining landscape shows that about 12-13% of requests still arrive uncompressed, while the newer zstd compression method maintains a minimal 1% adoption rate across both platforms. A negligible amount of websites also seem to use a combination of compression techniques, like gzip + deflate, or br + gzip. This doesn't necessarily mean that both are being used at the same time, because using both anyway doesn't have any additional effect. What could explain the usage is either that different assets on the same page might be using different compression methods, or supporting different kind of browsers when brotli for example was not supported in all browsers.

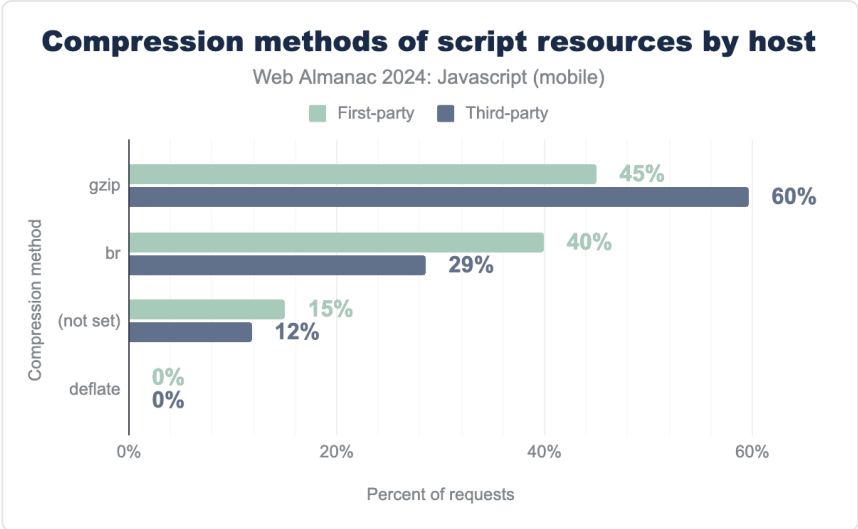


Figure 1.23. Compression methods of script resources by host.

An interesting contrast is seen when looking at third-party scripts making gzip a clear winner again. Looking at the trends, gzip is still the primary compression method used with a 60% vs 29% comparison. This shows the missed performance gains due to a lot of third-party JavaScript still being deployed without brotli compression.

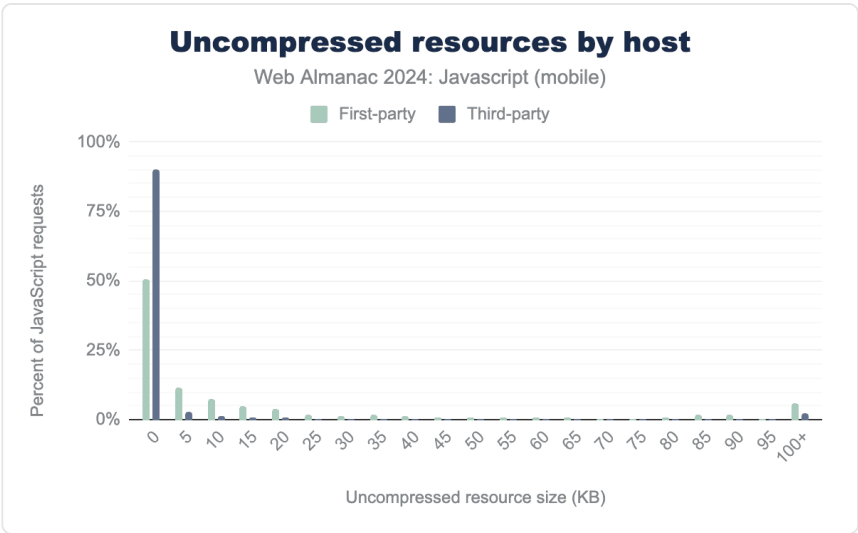


Figure 1.24. Uncompressed resources by host.

Looking at uncompressed resources across the spectrum, we see the same smaller resources of less than 5 KB that are not sent over with any compression which makes sense as applying compression on such small resources does not add much value and in fact could add the compression overhead. However, some larger first party resources are still not enjoying the benefits of any kind of compression methods, wiht 6% of scripts of 100 kilobytes or mote not being compressed at all.

Minification

JavaScript minification is a crucial optimization technique that reduces the size of JavaScript code by eliminating unnecessary characters without changing its functionality. Think of it like taking a lengthy novel and removing all the whitespace and making character names shorter - the story remains the same, but it takes up less space. The part about making function names, variable names, class names etc. shorter is also called uglification.

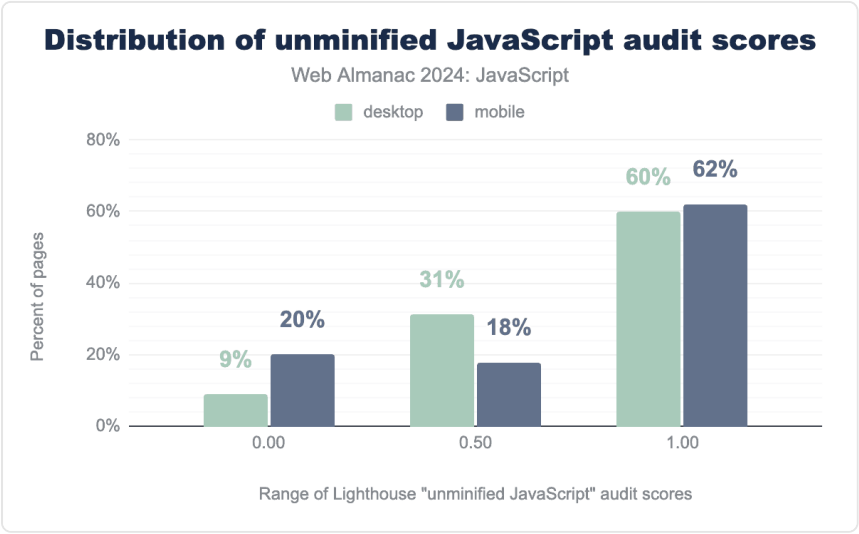


Figure 1.25. Distribution of unminified JavaScript audit scores.

Here, 0.00 represents the worst score whereas 1.00 represents the best score. 62% of mobile pages are scoring between 0.9 and 1.0 on Lighthouse's minified JavaScript audit, whereas the figure for desktop pages is 60%. This means that on mobile, 38% of pages have opportunities to ship minified JavaScript, whereas that figure for desktop pages is 40%.

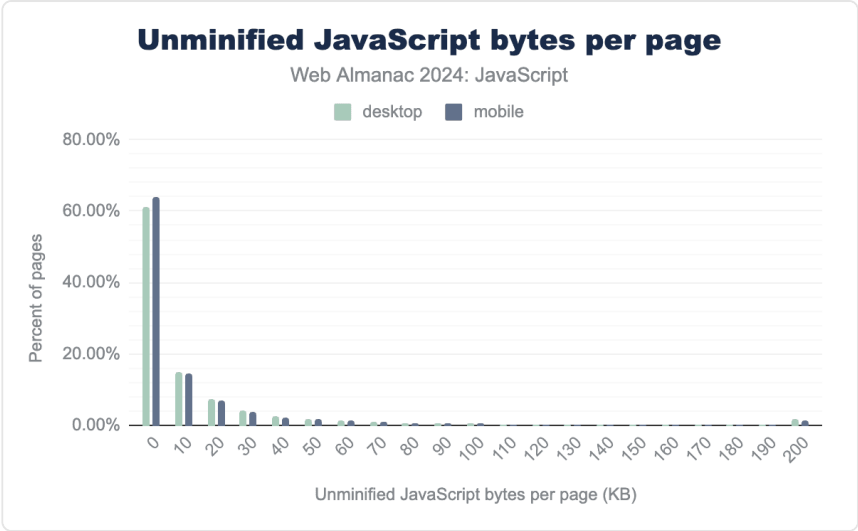


Figure 1.26. Unminified JavaScript bytes per page.

At the median, we see that pages are shipping around 12 KB of JavaScript that can be minified. By the time we get to the 75th and 90th percentiles, however, that number jumps quite a bit, from 34 KB to about 76 KB. Third-parties are pretty good throughout, up until we get to the 90th percentile, however, where they’re shipping around 19 KB of unminified JavaScript.

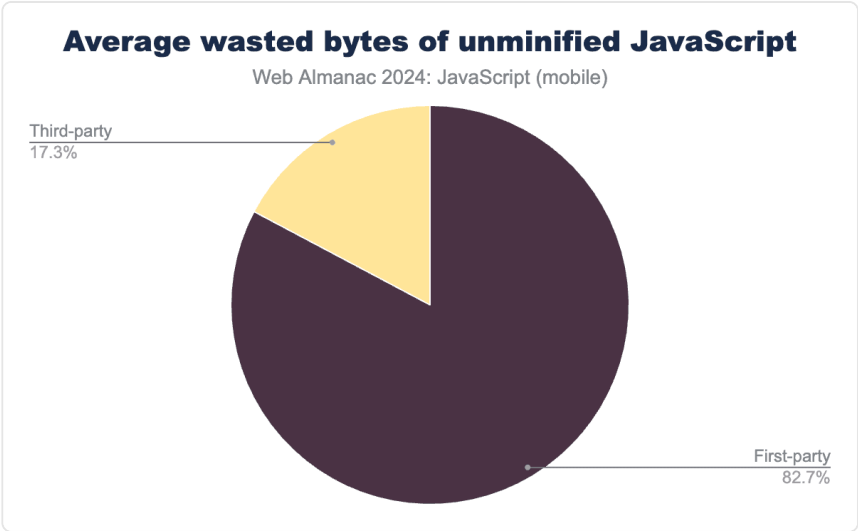


Figure 1.27. Average wasted bytes of unminified JavaScript.

When it comes to unminified JavaScript wasting bandwidth in 2024, first-party code is still the main offender. The numbers don't lie—82.7% of these wasted bytes come from an organization's own scripts, while third-party code chips in a smaller but still notable 17.3%. It's a pattern we've seen before, but it's surprising how little has changed. We often point fingers at third parties for performance issues, but the real low-hanging fruit seems to be the first party code.

Minification isn't just a checkbox—it's a must. Stripping out comments, whitespace, and unused code shrinks file sizes, which speeds up how fast a page loads. And first-party scripts are our responsibility, which if not minified leaves obvious performance gains on the table. Third parties aren't off the hook, though. That 17.3% might seem small, but unoptimized vendor scripts can drag down performance, especially on slower networks or older devices. If a third-party script isn't minified, the question really should be to ask ourselves if the tool is worth it or could we swap it for something leaner?

Every kilobyte shaved off a script means faster load times, happier users, and better SEO.

Source Maps

Source maps remain a critical tool for developers, bridging the gap between minified production code and its original, human-readable form. They're essential for debugging but often underutilized—or misused—in practice. Let's break down the 2024 data.



Figure 1.28. The percentage of mobile pages specifying source map comments to publicly accessible source maps.

This marks a slight uptick from previous years. In 2022, only 14% of mobile pages used source map comments, and just 0.12% leveraged headers. While adoption is inching upward, progress feels glacial. The HTTP header method remains stubbornly rare, likely due to its reliance on server configuration and developer awareness.



Figure 1.29. The number of mobile pages specifying source map headers.

Source maps themselves aren't a performance issue—browsers ignore them unless explicitly

requested (e.g., via DevTools). However, their misuse can backfire:

Inline source maps (base64-encoded within production files) bloat JavaScript payloads, slowing downloads and processing. Publicly exposed source maps risk revealing sensitive code logic or credentials if not properly scoped.

The data suggests most teams still opt for source map comments over headers. While comments are easier to implement (often automated by build tools like webpack or Rollup), headers offer better control. For instance, headers can be conditionally served only to internal tools or authenticated users, reducing exposure of raw source code.

Responsiveness

We rely on JavaScript to provide interactivity, but the use of JavaScript can result in poor input responsiveness. More information can be found in the dedicated Performance chapter.

Metrics

We look at both field data from the Chrome UX Report (CrUX)⁷ and Lighthouse lab data to measure responsiveness.

7. <https://developer.chrome.com/docs/crux>

Interaction to Next Paint (INP)

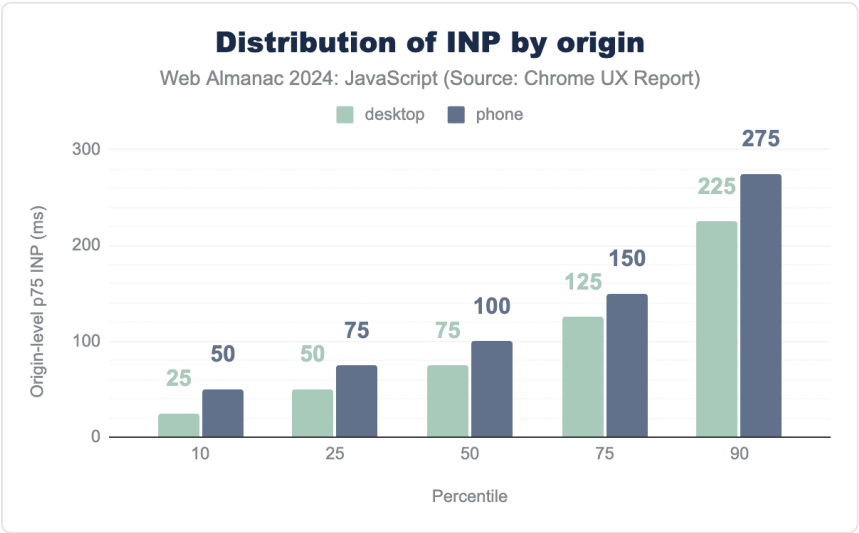


Figure 1.30. Distribution of INP by origin.

Interaction to Next Paint (INP)⁸ became a Core Web Vital in 2024⁹. It measures all keyboard, mouse, and touch interactions on a page and selects a high percentile of interaction latency to represent overall page responsiveness.

A “good” INP score is 200 milliseconds or less. At the median (50th percentile), both mobile (100 ms) and desktop (75 ms) score well within this threshold. However, at the 75th percentile, desktop (125 ms) and mobile (150 ms) approach the “needs improvement” range. By the 90th percentile, desktop (225 ms) and mobile (275 ms) exceed the “good” threshold, indicating responsiveness issues for a significant portion of websites.

8. <https://web.dev/articles/inp>
9. <https://web.dev/blog/inp-cwv-launch>

Total Blocking Time (TBT)

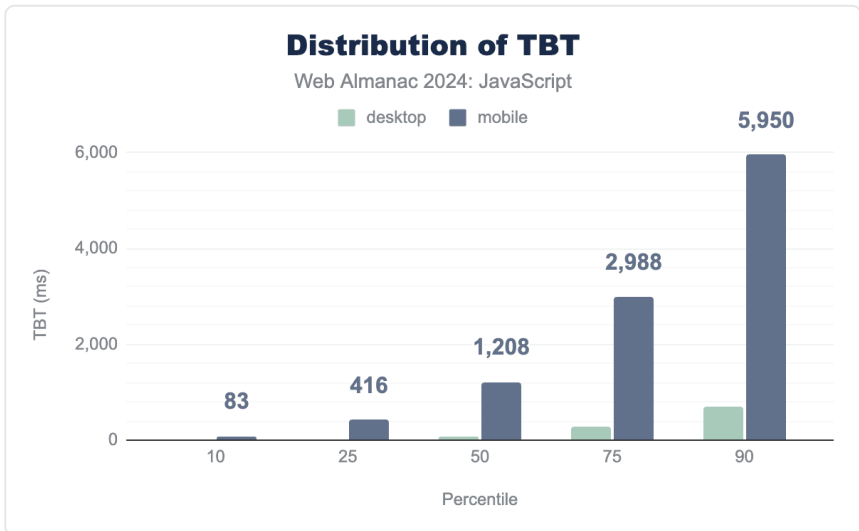


Figure 1.31. Distribution of TBT.

The Total Blocking Time (TBT)¹⁰ metric is a lab metric which calculates the total blocking time of long tasks during startup.

TBT is sourced from Lighthouse rather than real-user data. This measures synthetic performance in simulated desktop and mobile environments with device-appropriate CPU and network throttling enabled.

At the 75th percentile, mobile pages have nearly 3.0 seconds (2,988 ms) of blocking time, indicating a poor user experience. By the 90th percentile, mobile blocking time surges to 5.95 seconds (5,950 ms), whereas desktop remains substantially lower, reinforcing the performance gap between device types.

Long Tasks / blocking time

A long task is any task that runs on the main thread for longer than 50 milliseconds. The length of the task beyond 50 milliseconds is that task's blocking time, which can be calculated by subtracting 50 milliseconds from the task's total time.

10. <https://web.dev/articles/tbt>

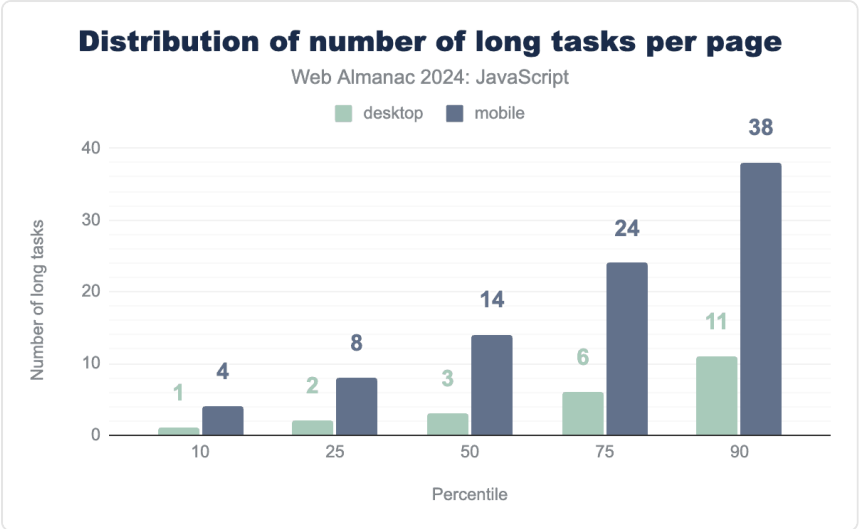


Figure 1.32. Distribution of number of long tasks per page.

The median page encounters 14 long tasks on mobile and 3 long tasks on desktop devices. This aligns with expectations, as desktop devices typically have greater processing power and memory resources than mobile devices.

At the 75th percentile, mobile pages experience 24 long tasks, while desktop pages have 6 long tasks. By the 90th percentile, mobile pages encounter 38 long tasks, whereas desktop pages still manage significantly fewer at 11 long tasks. This highlights the challenge of optimizing JavaScript execution, particularly for mobile users who face a much higher burden of blocking tasks.

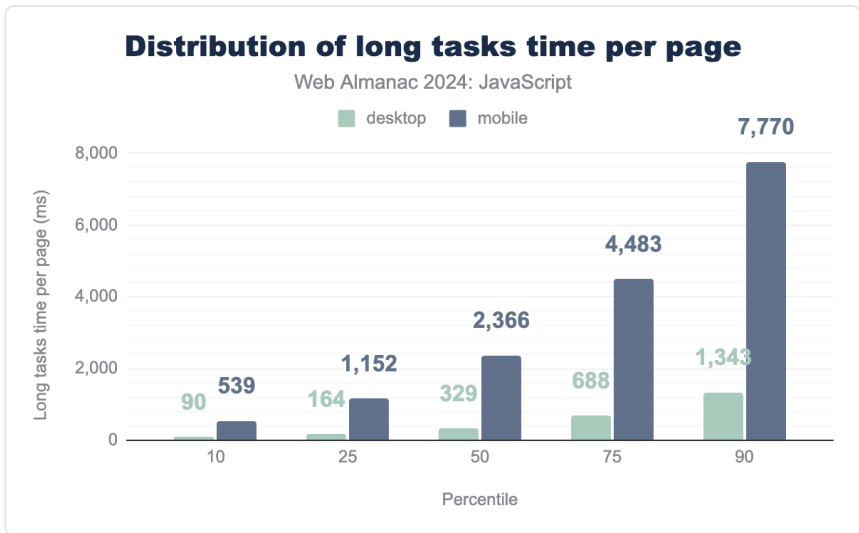


Figure 1.33. Distribution of long tasks time per page.

The median mobile page has 2.37 seconds (2,366 ms) dedicated to long tasks, whereas desktop pages experience significantly less, at just a fraction of that time.

At the 75th percentile pages spend 4.48 seconds (4,483 ms) processing long tasks, while desktop pages remain much lower. By the 90th percentile, mobile long task time soars to 7.77 seconds (7,770 ms), highlighting a major responsiveness issue. This excessive processing time suggests a strong need for JavaScript optimizations, such as breaking up long tasks or leveraging web workers to handle intensive computations off the main thread. These results underscore the challenges mobile users face when dealing with heavy JavaScript execution.

Scheduler API

The Scheduler API¹¹ has recently been expanded with the `yield` method, which provides an easier method to break up long tasks¹².

0.65%

Figure 1.34. The percentage of mobile pages using the Scheduler API.

11. <https://developer.mozilla.org/docs/Web/API/Scheduler>

12. <https://web.dev/articles/optimize-long-tasks>

Currently, only 0.65% desktop pages are shipping JavaScript that uses the Scheduler API, a significant increase from 0.002% last time we looked¹³. On mobile, 0.81% of pages now utilize this feature. This growth from 2022 suggests that as documentation improves and support expands, developers are increasingly integrating the Scheduler API into their applications. The rising adoption, especially in frameworks, indicates a shift towards more efficient scheduling and performance optimization in JavaScript execution. We expect this trend to continue, ultimately contributing to better user experience outcomes.

Synchronous XHR

AJAX—or usage of the XMLHttpRequest (XHR) has a flag that allows you to make synchronous requests. Synchronous XHR is harmful for performance because the event loop and main thread is blocked until the request is finished, resulting in the page hanging until the data becomes available. `fetch` is a much more effective and efficient alternative with a simpler API, and has no support for synchronous fetching of data.



Figure 1.35. The percentage of mobile pages using synchronous XHR.

Synchronous XHR is now used on 2.15% of mobile pages and 2.22% of desktop pages, marking a small decline from 2022 (2.5% and 2.8% respectively). While its usage is decreasing, its continued presence—even at these levels—indicates that some legacy applications still rely on this outdated method, which negatively impacts user experience.

`document.write`

The only `document.write` API is very problematic for a number of reasons that the HTML spec itself warns against its use¹⁴.



Figure 1.36. The number of mobile pages using `document.write`.

A notable 12% of mobile pages observed are still using `document.write` to add content to

13. <https://almanac.httparchive.org/en/2022/javascript#scheduler-api>

14. [https://html.spec.whatwg.org/multipage/dynamic-markup-insertion.html#document.write\(\)](https://html.spec.whatwg.org/multipage/dynamic-markup-insertion.html#document.write())

the DOM instead of proper insertion methods. On desktop, 13% of pages continue to rely on this approach. This marks a decline from 2022 (18% and 17%), suggesting that more developers are moving away from this inefficient method. However, legacy applications and third-party scripts still contribute to its usage.

Legacy JavaScript

Lighthouse currently checks for Babel transforms that may be unnecessary on the modern web, such as transforming use of `async` and `await`, JavaScript classes, and other newer, yet widely supported language features.



Figure 1.37. The percentage of mobile pages that ship legacy JavaScript.

Just over two-thirds of mobile pages are still shipping JavaScript resources that are being transformed or contain unnecessary legacy JavaScript. This figure remains unchanged from 2022, indicating that despite awareness of performance issues, many pages continue to rely on these transformations. On desktop, 70% of pages are still shipping these transforms.

Despite a negligible change in this statistic from 2022, we hope to see a decline over time as JavaScript's evolution stabilizes and developers adopt more efficient practices.

How is JavaScript used?

JavaScript can be used directly, or via abstractions such as libraries and frameworks.

Library usage

To understand the usage of libraries and frameworks, HTTP Archive uses Wappalyzer to detect the technologies used on a page.

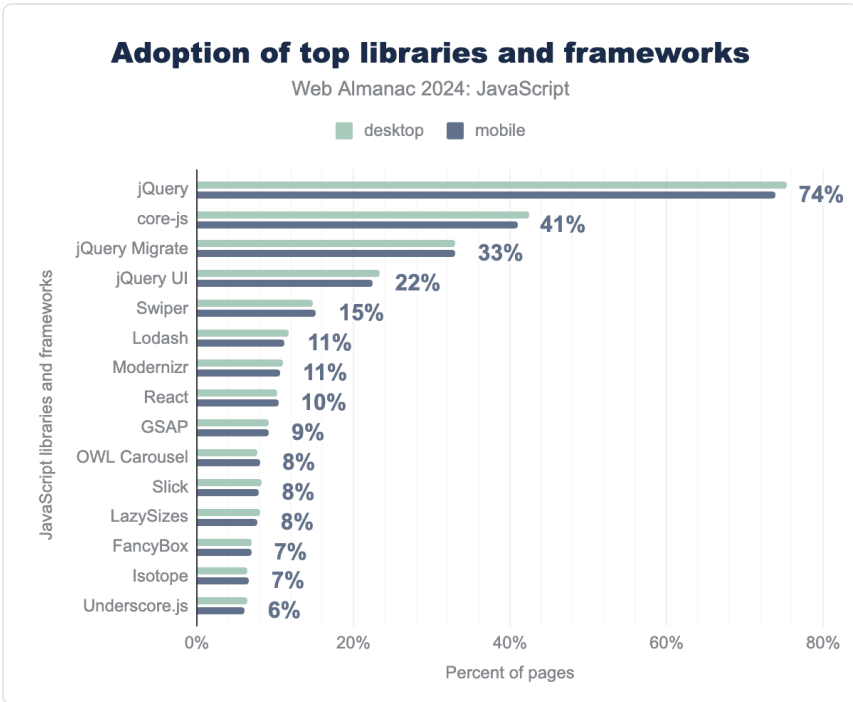


Figure 1.38. Adoption of top libraries and frameworks.

Readers of previous editions of the Web Almanac will not be surprised to see that jQuery still remains the most widely used library on the web, appearing on 74% of pages. A significant portion of this is due to WordPress, but even outside of WordPress, jQuery continues to be a dominant choice for many websites.

The widespread adoption of core-js (41%) is also expected, as many web applications rely on Babel, which often uses core-js to provide polyfills for missing JavaScript features. As browsers continue to evolve and support more modern features natively, this number should decline, reducing unnecessary bytes in web applications.

jQuery Migrate is present on 33% of pages, indicating that a large number of websites are still relying on older jQuery versions. Similarly, jQuery UI is still in use on 22% of pages, despite being mostly deprecated.

Other notable libraries include Swiper (15%), Lodash (11%), and Modernizr (11%), all of which play roles in handling UI elements and feature detection.

React usage has grown slightly to 10%, compared to 8% last year, but this still suggests a relatively stable adoption rate. This may indicate that while React remains popular, its growth

has plateaued due to increasing competition in the JavaScript ecosystem.

Meanwhile, libraries like GSAP (9%), OWL Carousel (8%), Slick (8%), LazySizes (8%), and FancyBox (7%) continue to be widely used, especially in performance and animation-heavy applications.

As the web ecosystem continues to modernize, we expect some of these legacy libraries, especially jQuery-based ones, to decline over time in favor of more native solutions and modern frameworks.

Libraries used together

Frameworks and libraries are often used together on the same page. As with last year, we'll examine this phenomenon to gain insight into how many libraries and frameworks have been used together in 2024.

apps	desktop	mobile
jQuery	7.57%	7.61%
jQuery, jQuery Migrate	3.71%	3.91%
core-js, jQuery	1.71%	1.67%
GSAP, Lodash, React	1.25%	1.50%
jQuery, jQuery UI	1.71%	1.48%
core-js, jQuery, jQuery Migrate	1.33%	1.29%
Swiper, core-js, jQuery, jQuery Migrate, jQuery UI	1.10%	1.22%
core-js, jQuery, jQuery Migrate, jQuery UI	1.02%	1.12%
Lodash, Modernizr, Stimulus, YUI, core-js	1.02%	0.88%
core-js	0.73%	0.75%

Figure 1.39. Common library combinations.

It's clear though that jQuery has some serious staying power, with some combination of it, its UI framework, and its migration plugin occurring in the top seven spots, with core-js having a prominent role in library usage as well.

Web Components and Shadow DOM

Web Components allows encapsulation of logic and styling through web components and the shadow DOM. To kick off this year's analysis, we'll begin with custom elements.

7.8%

Figure 1.40. The percentage of desktop pages that used custom elements.

This figure has increased from the 2022 analysis of custom element usage on desktop pages, which was 2.0%. With the advantages that custom elements provide and their reasonably broad support in modern browsers, we're encouraged to see the growing adoption of the web component model. This trend suggests that developers are increasingly leveraging web platform built-ins to create faster user experiences.

2.5%

Figure 1.41. The percentage of mobile pages that used shadow DOM.

Shadow DOM allows you to create dedicated nodes in a document that contain their own scope for sub-elements and styling, isolating a component from the main DOM tree. Compared to the 2022 figure of 0.39% of mobile pages using shadow DOM, adoption of the feature has significantly increased, reaching 2.51% in 2024. This growth indicates a rising trend in developers leveraging shadow DOM for better component encapsulation and styling consistency.

0.28%

Figure 1.42. The percentage of mobile pages that use templates.

The template element helps developers reuse markup patterns. Their contents render only when referenced by JavaScript. Templates work well with web components, as the content that is not yet referenced by JavaScript is then appended to a shadow root using the shadow DOM.

Roughly 0.28% of web pages on mobile are currently using the template element, a notable increase from 0.05% in 2022. Though templates are well-supported in browsers, their adoption

remains relatively low but is showing signs of growth.

0.29%

Figure 1.43. The percentage of mobile pages that used the `is` attribute.

The HTML `is` attribute is an alternate way of inserting custom elements into the page. Rather than using the custom element's name as the HTML tag, the name is passed to any standard HTML element, which implements the web component logic. The `is` attribute is a way to use web components that can still fall back to standard HTML element behavior if web components fail to be registered on the page.

In 2024, the adoption of the `is` attribute has increased to 0.29% from 0.08% in 2022. Despite this growth, its usage remains lower than custom elements themselves. Due to the lack of support in Safari, browsers on iOS and macOS cannot utilize the attribute, possibly contributing to its limited adoption.

Conclusion

The state of JavaScript continues to follow expected trends—its usage keeps growing, but so do efforts to mitigate its impact. Developers are increasingly leveraging minification, resource hints, compression, and smarter dependency management to balance performance and functionality.

However, our growing reliance on JavaScript raises concerns for web performance and user experience. Reducing unnecessary script execution and optimizing delivery remain crucial challenges. As the web platform evolves, we hope to see greater adoption of native APIs where feasible, while frameworks continue to improve their efficiency and embrace performance-conscious best practices.

Looking ahead, a meaningful shift in the trend would require a collective push toward better tooling, best practices, and awareness. Until then, we must remain diligent in optimizing JavaScript delivery—ensuring a fast, resilient web for all users.

Authors



Abdul Haddi Amjad

✉ @haddiamjad 📧 hadiamjad 🌐 <https://haddiamjad.github.io/>

Amjad is a fourth-year Ph.D. candidate at Virginia Tech, advised by Dr. Muhammad Ali Gulzar from Virginia Tech. He is also member of ProperData research group¹⁵ and co-host at NothingToHide?¹⁶ podcast. His research focuses on using program analysis techniques to solve internet security and privacy problems.



Nishu Goel

✉ @TheNishuGoel 📧 NishuGoel 🌐 <https://unravelweb.dev/>

Nishu Goel is an engineer at Epilot¹⁷. She is a Google Developer Expert for Web Technologies and Angular, Microsoft MVP for Developer Technologies, and the author of Step by Step Guide Angular Routing (BPB, 2019) and A Hands-on Guide to Angular (Educative, 2021). Find her writings at unravelweb.dev¹⁸.

15. <https://properdata.eng.uci.edu/researchers/>

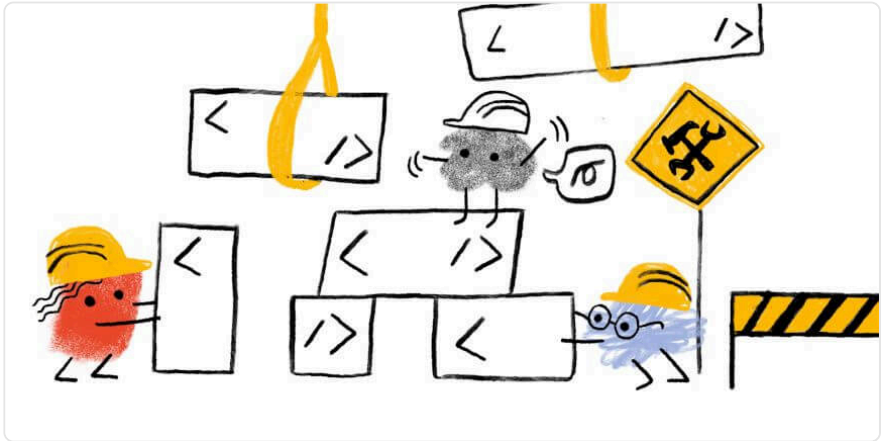
16. <https://nothingtohide.online/>

17. <https://www.epilot.cloud/en>

18. <https://unravelweb.dev/>

Part I Chapter 2

Markup



Written by Estela Franco

Reviewed by Brian Kardell, Jens Oliver Meiert, and Simon Pieters

Analyzed by Estela Franco

Introduction

The web as we know it is built on the foundation of HTML. Every website, every web application, and every online interaction starts with HTML at its core, making it one of the most essential web standards. It's the language that structures content, defines relationships, and communicates with browsers, ensuring that what we create can be viewed, interacted with, and understood by users worldwide. This chapter is dedicated to understanding how HTML continues to shape the web in 2024, exploring trends in its use, the rise of custom elements, and how developers are leveraging new features to build more accessible, efficient, and future-proof websites.

This year's edition brings a broader perspective, as our dataset now includes not only home pages but also a wide variety of secondary pages. By analyzing pages beyond just the front doors of websites, we're able to capture a richer, more accurate snapshot of how HTML is used across different types of content and contexts. From blog posts and product pages to login screens and article archives, this expanded scope gives us deeper insights into the real-world application of HTML.

We encourage readers to dive deeper into the data, explore their own insights, and join the conversation about the future of the web’s foundational language.

General

Let’s start with some of the more general aspects of a markup document. In this section we’re covering the document types, the size of the documents, language and compression. ‘

Doctypes

Doctype	Rendering Mode”	Desktop	Mobile
<!doctype html>	standards mode	91.7%	92.8%
html public "-//w3c//dtd xhtml 1.0 transitional//en" "http://www.w3.org/tr/xhtml1/dtd/xhtml1-transitional.dtd"	almost standards mode	3.4%	2.7%
No doctype	quirks mode	2.1%	2.2%
html public "-//w3c//dtd xhtml 1.0 strict//en" "http://www.w3.org/tr/xhtml1/dtd/xhtml1-strict.dtd"	standards mode	0.8%	0.7%
html public "-//w3c//dtd html 4.01 transitional//en" "http://www.w3.org/tr/html4/loose.dtd"	almost standards mode	0.6%	0.4%
html public "-//w3c//dtd html 4.01 transitional//en"	quirks mode	0.3%	0.3%

Figure 2.1. Doctype usage.

92.8%

Figure 2.2. Mobile pages using the standard HTML doctype.

19. <https://hsivonen.fi/doctype/>

93% of all mobile pages use the standard HTML doctype. That is, `<!DOCTYPE html>`.

This is 3 percentage points higher than the 2022 data²⁰. The surprising part is the next most popular: `XHTML 1.1 Transitional`—but slowly disappearing (2.7%, down from 3.9% in 2022).

Document size

A page's document size is the amount of HTML bytes transferred over the network, including compression.

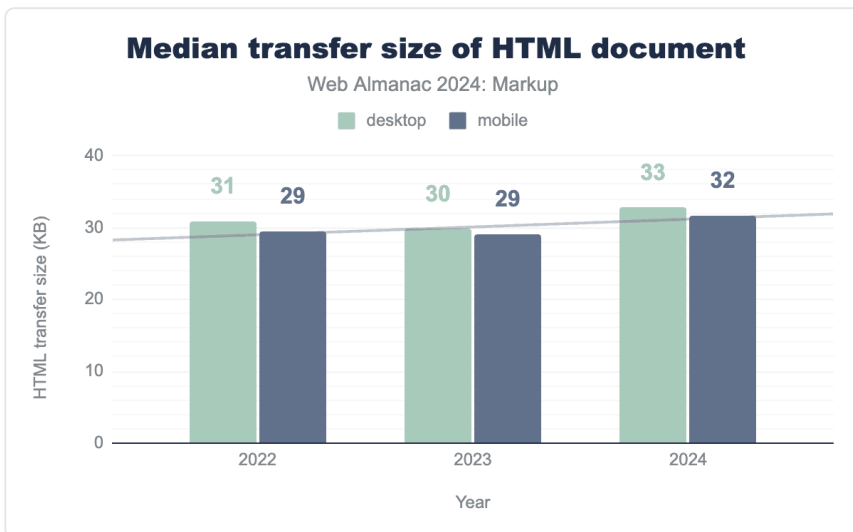


Figure 2.3. Median transfer size of HTML document.

After a slight decrease in 2023, the HTML transfer size increased this year compared to 2022 and 2023.

Although the median looks like something reasonable, let's take a closer look at the other percentiles.

20. <https://almanac.httparchive.org/en/2022/markup#doctype>

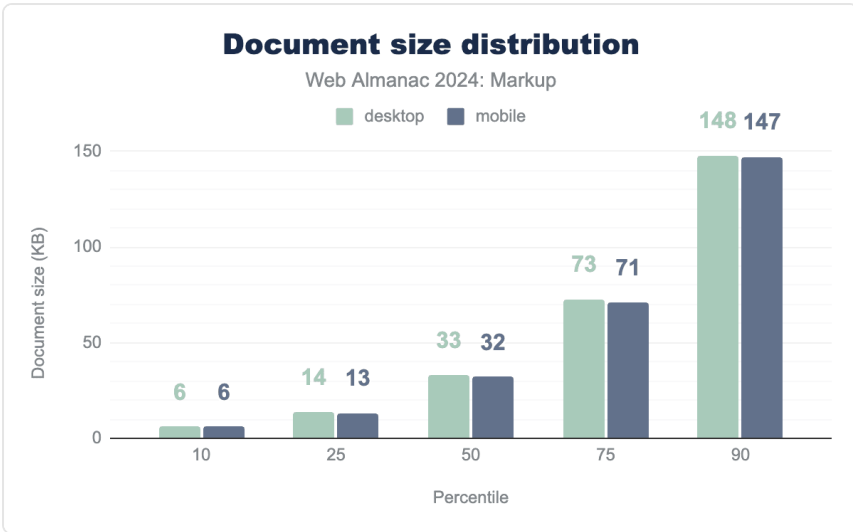


Figure 2.4. Distribution of the transfer size of HTML document.

The percentile distribution reveals that at the 10th percentile, HTML files are as small as 6 KB, while at the 90th percentile, they reach up to 147 KB. These extremes highlight a significant variation in how developers structure their pages.

Compression

In the context of analyzing HTML document files, compression continues to play a crucial role in improving load times and overall performance.

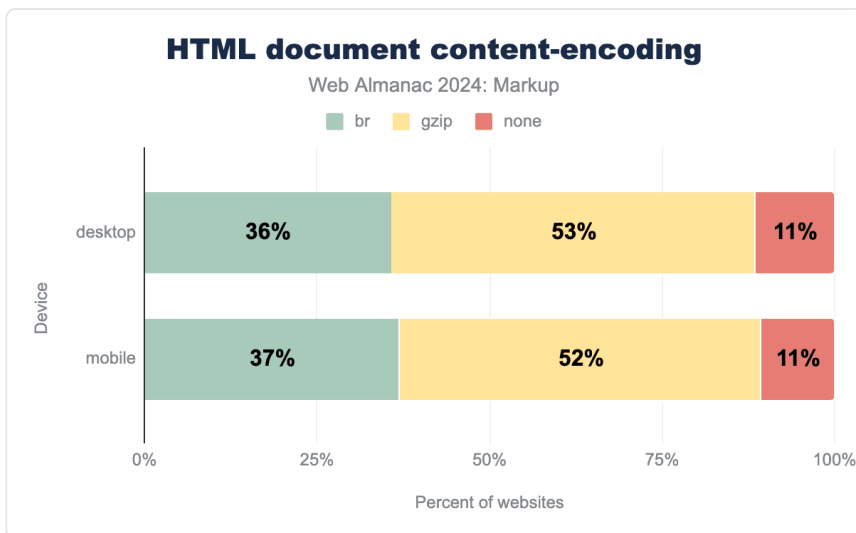


Figure 2.5. HTML document content-encoding.

One notable trend is the increasing popularity of the Brotli (`br`) compression format. In 2024, Brotli is used on 37% of mobile pages, a steady increase from 28% in 2023.

While `gzip` remains the most widely used compression method (52% on mobile), its usage has slightly declined from previous year as `br` gains traction (58% in 2022).

Despite these improvements, a small percentage of HTML files (10.5% on mobile) are still served without any compression, presenting missed opportunities for optimization.

Document language

5,625

Figure 2.6. Unique `lang` attribute codes on mobile.

In our analysis, we've encountered 5,625 unique instances of the `lang` attribute on the `html` element on mobile.

The HTML `lang` attribute plays an important role in helping screen readers and search engines understand the language of a webpage's content. However, interestingly, Google Search ignores the `lang` attribute when determining the language of a page because they've

identified that “it is almost always wrong”²¹. This may explain why `en` remains dominant in the dataset, with 44.2% of desktop and 40.5% of mobile pages using it as the primary language attribute, even though the actual language of the content might differ.

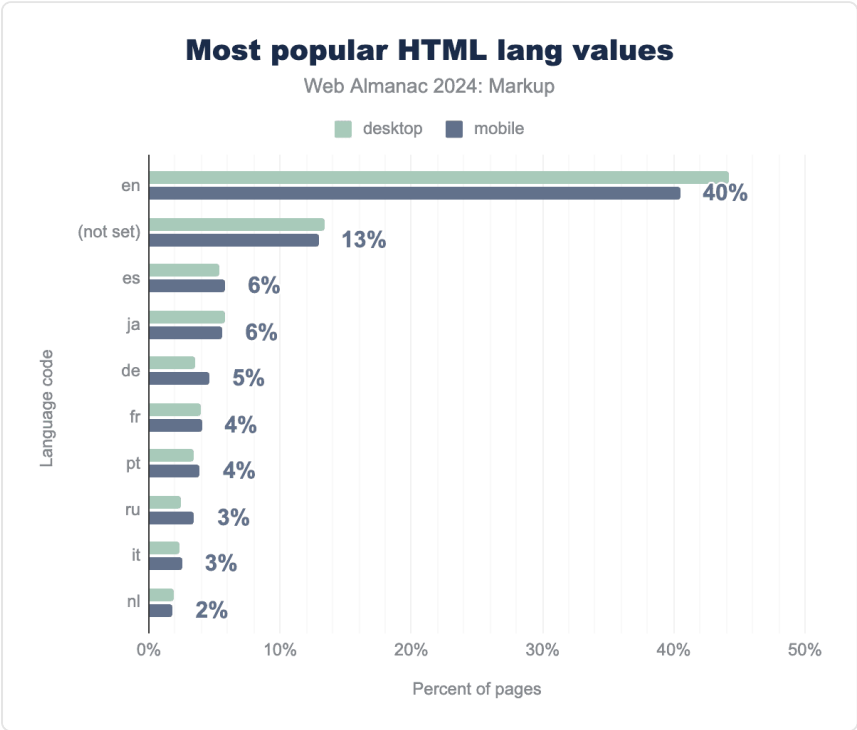


Figure 2.7. Most popular HTML language codes, not including region.

Additionally, 13% of pages have no `lang` attribute set at all, showing that many websites fail to provide this indicator.

If we aggregate the percentages of non-English and non-“not set” `lang` values, we still capture around 46% of the total pages, reflecting the truly global nature of web content. However, as mentioned above, it’s important to remember that the high proportion of `en` values doesn’t always mean the content is in English, given the frequent misconfiguration of the `lang` attribute.

21. <https://www.youtube.com/watch?v=isW-Ke-AJJU&t=3354s>

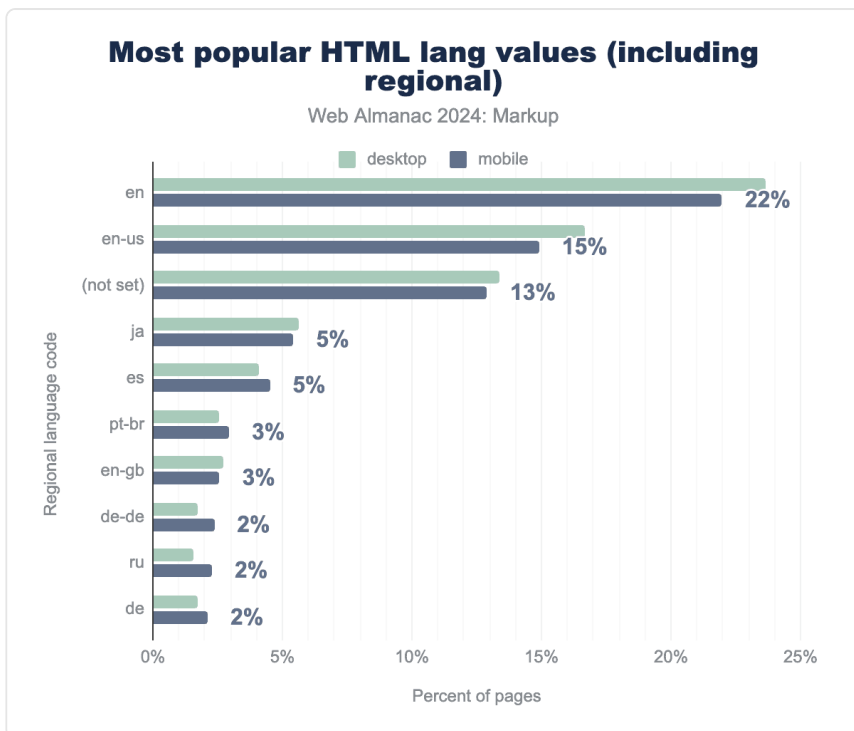


Figure 2.8. Most popular HTML language codes, including region.

In terms of non-English languages, `ja` (Japanese) and `es` (Spanish) stand out as some of the most popular choices, used on approximately 5-6% of pages.

The most common regional variant, `en-us`, appears on 16.7% of desktop and 15% of mobile pages.

Despite the issues with incorrect lang attribute values, the attribute still plays a vital role in improving accessibility. For users with screen readers, setting the `lang` attribute correctly remains an essential practice in modern web development.

Comments

HTML comments are snippets of text that developers include within their code to leave notes or explanations without affecting the visual display of the webpage. These comments are enclosed in `<!-- -->` tags and are not rendered by browsers, meaning users will never see them. While useful during the development process, HTML comments are not necessary in production code, as they can slightly increase the file size without any benefit to end users.

86%

Figure 2.9. Mobile pages with at least one comment.

According to our analysis, 86% of mobile pages still contain at least one comment.

In addition to regular comments, there's a specific type known as **conditional comments**. These were once used extensively to target specific versions of Internet Explorer (IE), allowing developers to provide custom styles or scripts that only older IE browsers would process.

```
<!--[if IE]>
<link rel="stylesheet" href="ie-only-styles.css">
<![endif]-->
```

With modern browsers and the retirement of Internet Explorer, conditional comments have become obsolete. Despite this, **26%** of mobile pages still contain conditional comments, likely due to legacy code that was never cleaned up, or because some sites continue to support older versions of Internet Explorer for compatibility reasons.

Elements

In this section, we'll explore HTML elements—what elements are commonly used, how often they appear, and which ones you're likely to find on a typical page. We'll also look into custom and outdated elements. And just to clarify: is “divitis” still around? Yes, it is.

Element diversity

For both desktop and mobile pages, the data shows that the 10th percentile has 22 distinct elements, while the 90th percentile reaches 44 elements on desktop and 43 on mobile. The median number of distinct elements for mobile pages has remained consistent at 32 this year, the same as in 2022²², and only slightly higher than the 31 observed in 2021²³.

22. <https://almanac.httparchive.org/en/2022/markup#element-diversity>

23. <https://almanac.httparchive.org/en/2021/markup#element-diversity>

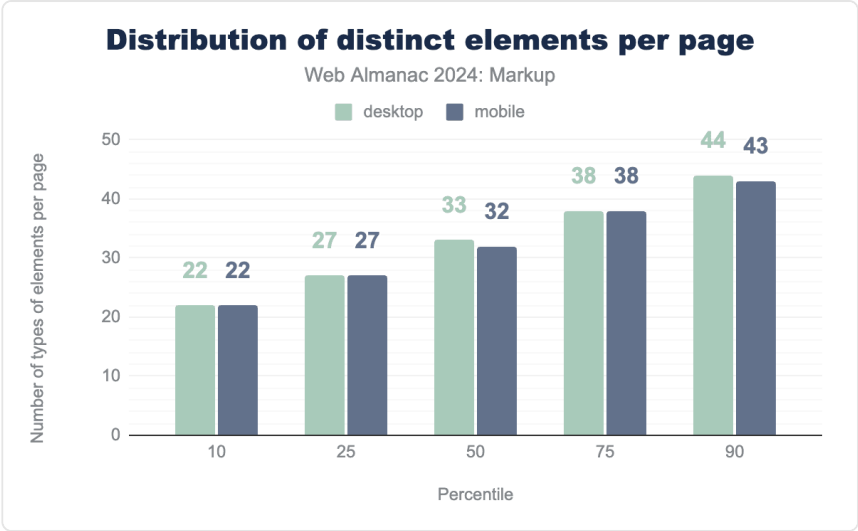


Figure 2.10. Distribution of the number of distinct types of elements per page.

However, there are some differences when checking the distribution of elements per page. The data shows a slight decrease compared to 2022²⁴. For mobile, the median number of elements has dropped from 653 in 2022 to 594 in 2024. At the lower end, the 10th percentile for mobile shows a small drop from 192 to 180. The 90th percentile also shows a modest decrease, with mobile pages dropping from 1,832 to 1,716. This overall reduction suggests that pages are becoming slightly leaner in terms of the number of HTML elements used.

24. <https://almanac.httparchive.org/en/2022/markup#element-diversity>

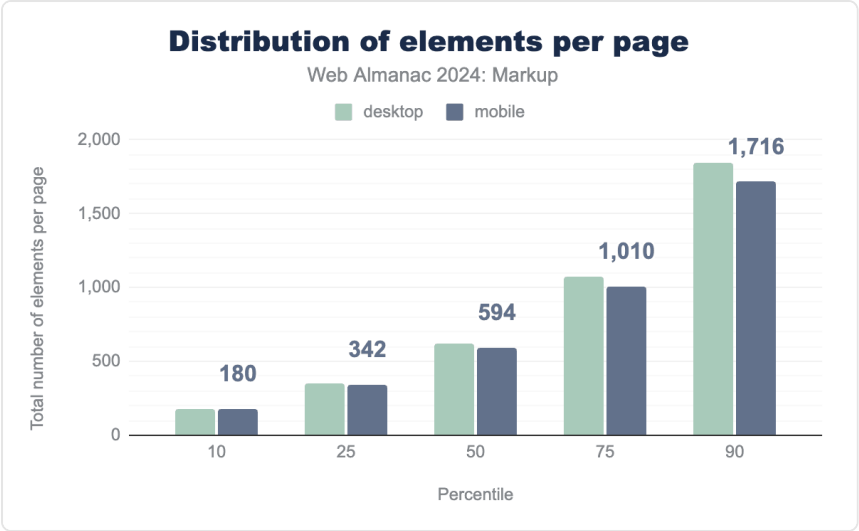


Figure 2.11. Distribution of the number of elements per page.

Top elements

The following elements are used most frequently:

2021	2022	2023	2024
div	div	div	div
a	a	a	a
span	span	span	span
li	li	li	li
img	img	script	script
script	script	img	img
p	p	p	p
link	link	link	link
meta	i	meta	path
i	meta	path	meta

Figure 2.12. Most used elements.

The list remains largely consistent with previous years, but some shifts have occurred.

29%

Figure 2.13. Percentage of elements which are div elements.

`<div>` remains by far the most dominant element. So “divitis” is still a thing, and it doesn’t look like it’s going to change in the next few years.

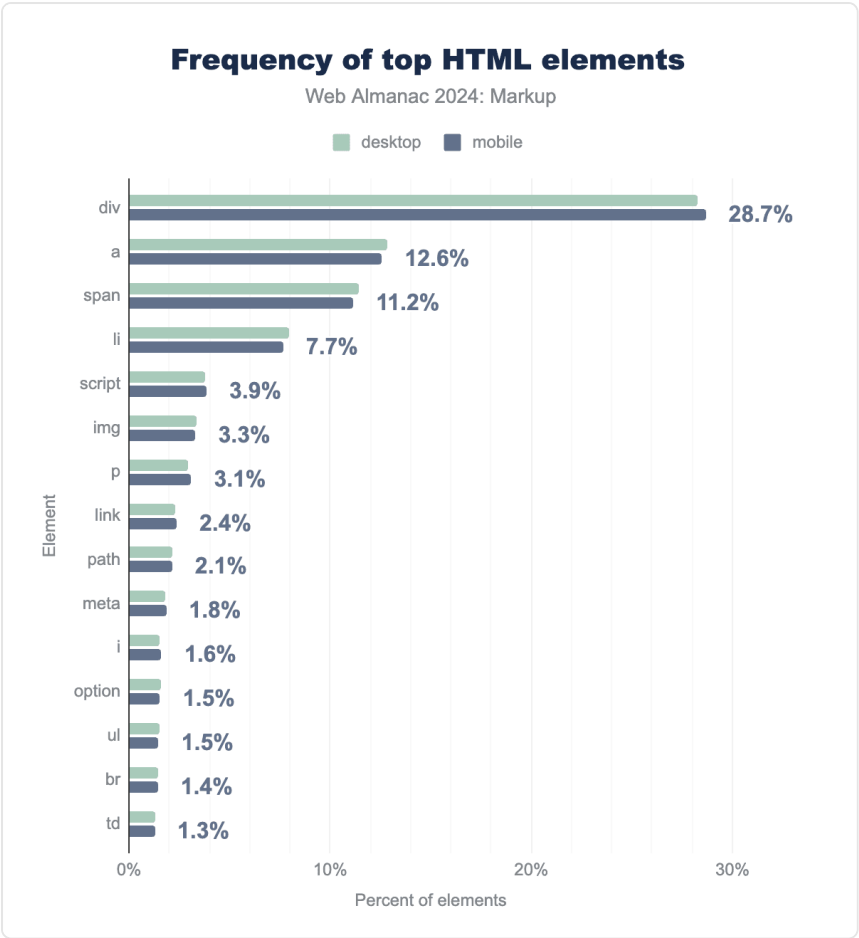


Figure 2.14. Frequency of top HTML elements.

Following `<div>`, the `<a>` element remains a key player, consistently in second place. As the backbone of hyperlinking, it plays a critical role in navigation, anchoring user journeys across sites.

One of the notable shifts in recent years has been the increased usage of `<script>`. In 2023, it surpassed `<img>` in popularity, reflecting the growing reliance on JavaScript for dynamic content, interactivity, front-end logic, and trace marketing campaigns. The trend has continued in 2024, solidifying `<script>` as the fifth most-used element.

Another notable shift is the emergence of `<path>`, which entered the top 10 in 2023. In 2024, it has surpassed `<meta>`, reflecting the increasing use of Scalable Vector Graphics (SVG) for

icons, illustrations, and graphical elements.

The adoption of top HTML elements across both desktop and mobile platforms remains consistently high, reflecting their foundational role in modern web development. The `<html>`, `<head>`, and `<body>` elements are nearly ubiquitous, appearing on over 99.7% of both desktop and mobile pages.

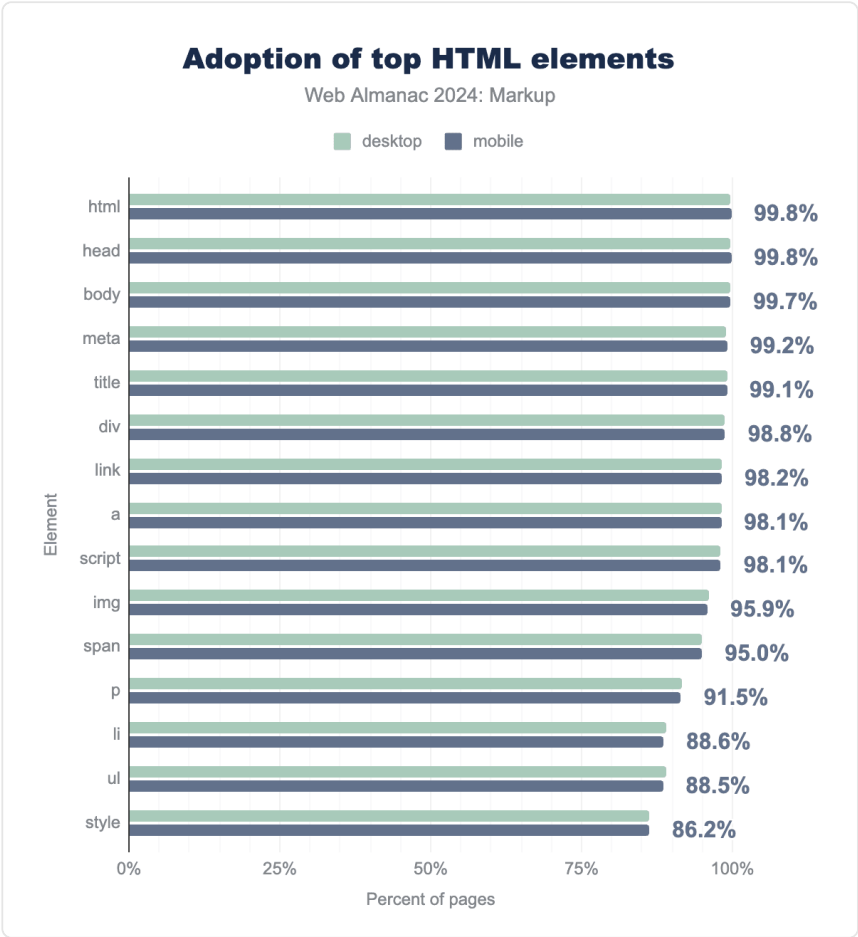


Figure 2.15. Popularity of top HTML elements.

A notable observation is that 0.9% of mobile pages are missing the `<title>` tag, similar to the 2022 data²⁵ (1%).

25. <https://almanac.httparchive.org/en/2022/markup#top-elements>

The next elements, `<link>`, `<a>`, `<script>`, and `<img>`, also have strong adoption rates. It's also interesting to see the increasing use of SVG (Scalable Vector Graphics), even though this tag is not part of the top 15 elements. `<svg>` adoption on mobile has grown from 45.5% in 2022 to 51.6% in 2024, marking a significant shift towards more scalable, resolution-independent graphics on the web.

Custom elements

Custom elements, easily recognized by their hyphenated names, have once again made their mark in our analysis this year, showcasing their continued importance in extending HTML's native functionality.

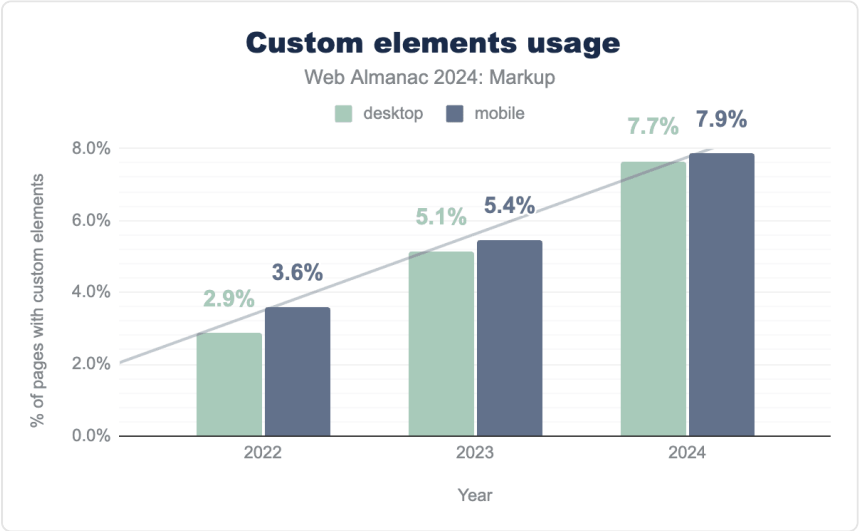


Figure 2.16. Custom elements usage by year.

The usage of custom elements has seen a significant increase in recent years, with adoption rates rising from 3.6% on mobile in 2022 to 7.9% in 2024. This increase highlights a growing trend among developers and technologies to leverage custom elements for building richer, more interactive web experiences.

However, custom elements typically need extra JavaScript to enable their functionality and interactivity. This requirement is particularly evident when examining the JavaScript payloads of web pages.

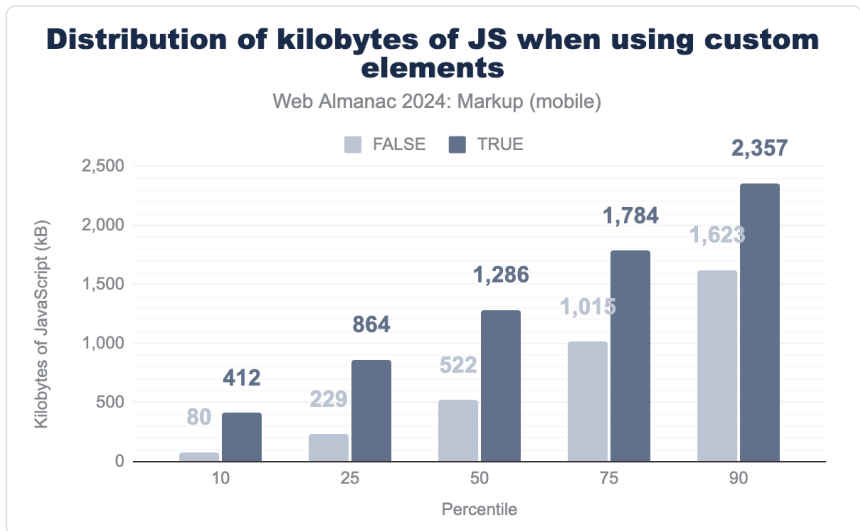


Figure 2.17. Distribution of kB of JS when using custom elements.

In this chart, we can see that at the median, pages with custom elements use 1,286 kB of JavaScript while pages without custom elements only require 522 kB. Hence, while the rise of custom elements represents a valuable evolution in web development—enabling developers to create modular and reusable components—it’s essential to consider the implications of their use.

Let’s now take a closer look at the top 10 custom elements:

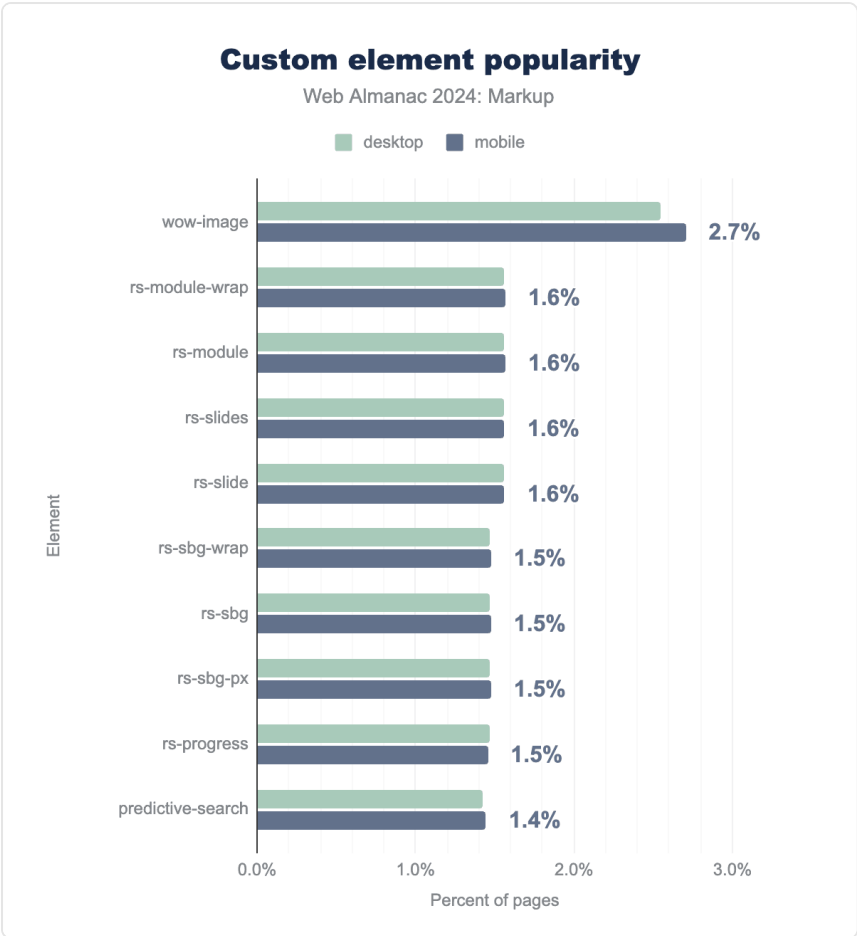


Figure 2.18. Custom element popularity.

As in the 2022 edition²⁶, most of the top 10 custom elements are dominated by `rs-*` elements from Slider Revolution²⁷. However, this year we see a new (and surprising) winner: `wow-image` element, which is used by the `@wix/image` package on Wix sites.

The last of this year's top 10 list is also a newcomer: `predictive-search`, a Shopify component that shows suggested results as you type.

26. <https://almanac.httparchive.org/en/2022/markup#custom-elements>

27. <https://www.sliderrevolution.com/>

Obsolete elements

There are currently 29 obsolete and deprecated elements²⁸ according to HTML specification. And except from keygen, all of them still appear in some (or many) pages of this year's dataset.

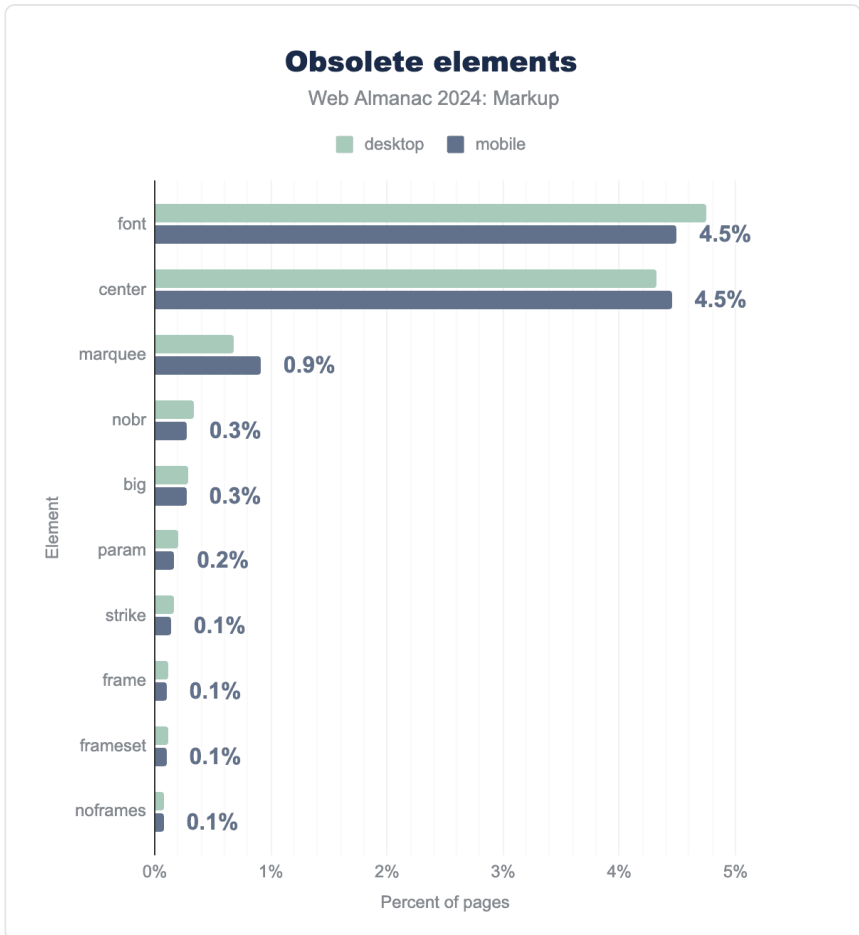


Figure 2.19. Obsolete element popularity.

If we compare these results to the 2022 ones²⁹, we see a slow but steady decline in their usage. One notable improvement is the drop in the use of the `<center>` element, which has fallen from 6.1% in mobile sites last year to 4.5% this year. This marks a significant decrease and has led to `<center>` being surpassed by the `<font>` element as the most commonly used

28. <https://html.spec.whatwg.org/multipage/obsolete.html#non-conforming-features>

29. <https://almanac.httparchive.org/en/2022/markup#obsolete-elements>

obsolete tag, now present on 4.5% of mobile pages. Interestingly, despite this positive trend, some high-profile sites, like Google's home page, still rely on the `<center>` element in their markup.

Attributes

This section focuses on how attributes are used in documents and explores patterns in `data-`* usage and social markup.

Top attributes

In HTML, attributes are key-value pairs attached to elements that provide additional information or modify the behavior of the element. These attributes are fundamental in defining characteristics such as styles, classes, links, and behavior within the web page. They often influence how elements are displayed or interacted with by users and scripts. For example, the `src` attribute in an `<img>` tag defines the image source, while the `href` attribute in an `<a>` tag specifies the link's destination.

For another year, the most used attribute by far is `class`, with 48 billion occurrences in our mobile dataset, representing 33% of all attributes used.

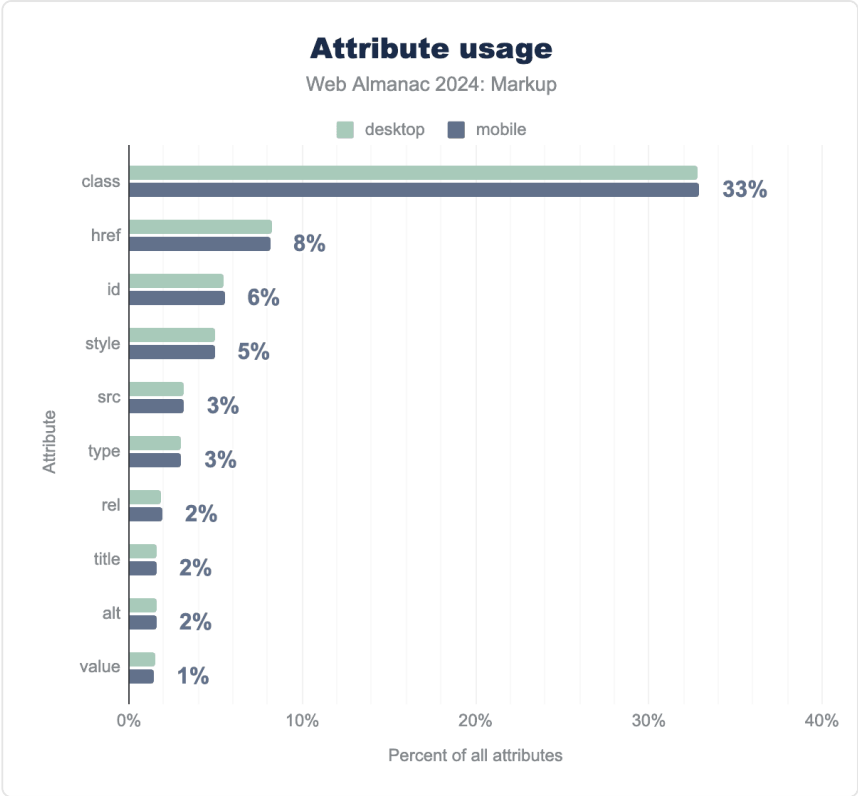


Figure 2.20. Frequency of top attributes.

And when we look at the attributes used per page, we find the following used on almost all of them:

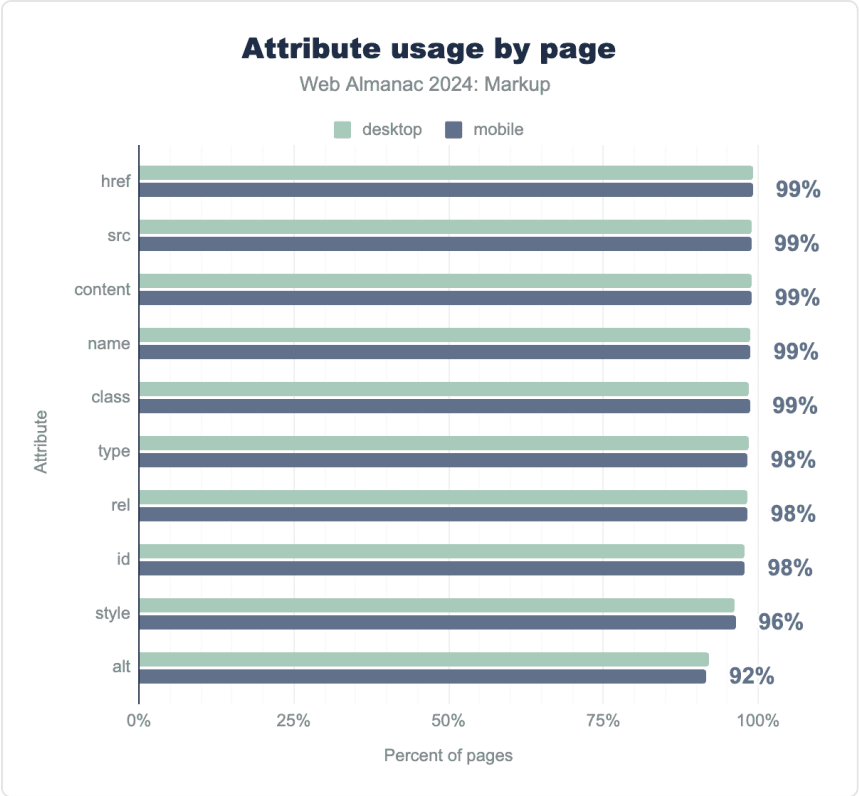


Figure 2.21. Popularity of top attributes.

data-* attributes

Let's now take a closer look to a subset of attributes: the `data-*` attributes. HTML allows developers to define custom attributes that begin with `data-`. These attributes are designed to store additional information specific to the page or application, such as custom data, annotations, or state information. They offer a way to embed extra, non-standard metadata that doesn't fit into any predefined HTML attributes, making them particularly useful when there's no existing attribute or tag to handle that specific information. The `data-` attributes are private to the application and can be easily accessed or manipulated via JavaScript, providing a flexible method to manage dynamic content or data states.

90%

Figure 2.22. Pages with at least one `data-*` attribute.

The overall data shows that 90% of the analyzed documents use at least one `data-*` attribute. Let's deep dive into the data.

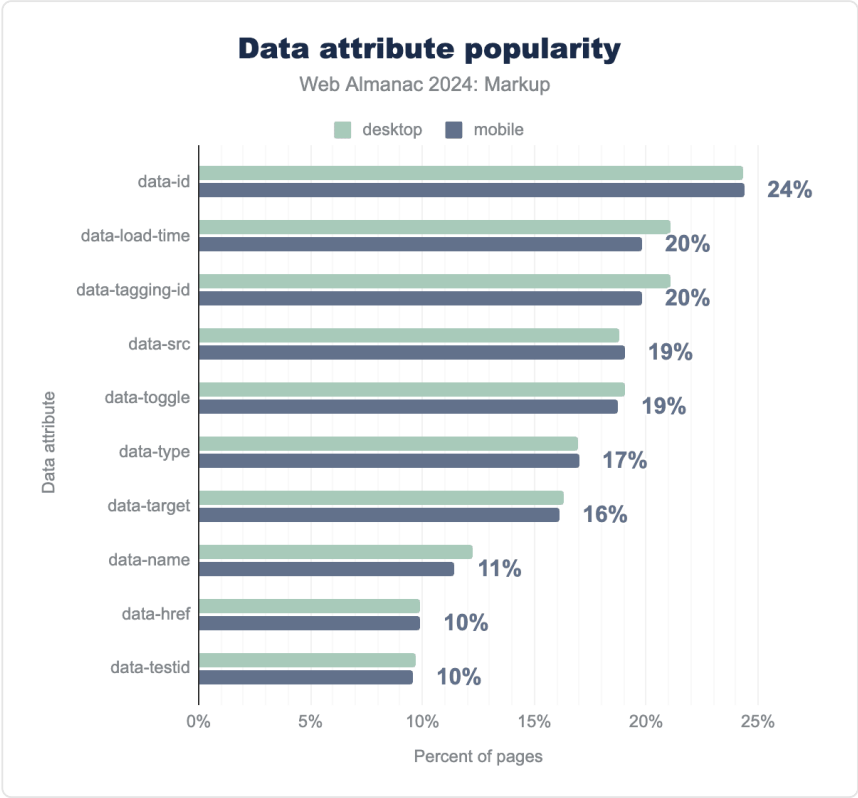


Figure 2.23. Popularity of top data attributes.

Analyzing the popularity of `data-*` attributes from 2022 to 2024 reveals some interesting shifts in their usage. This year, `data-id` is the most popular, used on 24% of mobile pages, a significant increase from 19% in 2022. This increase also marked a significant jump from fifth place in 2022 to first place this year.

Another notable change is the appearance of new elements in the list: `data-load-time` and

`data-tagging-id` appear on 20% of pages in 2024, occupying the second and third position of the ranking. These attributes were not part of the `data-*` attributes identified in 2022, indicating that performance tracking and tagging have become more important in modern web development.

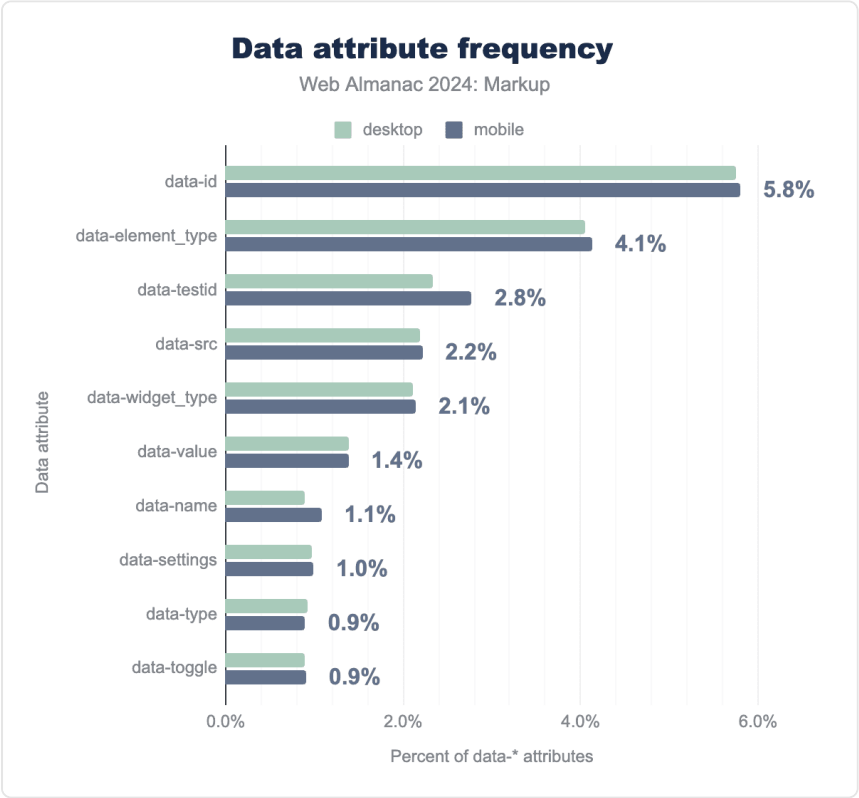


Figure 2.24. Frequency of top data attributes.

Social markup

Social markup refers to the set of meta tags embedded within HTML documents that enhance how web content is shared and displayed across social media platforms. These tags provide essential metadata, such as titles, descriptions, images, and URLs, ensuring that when users share a webpage, platforms like Facebook, X (formerly Twitter), and LinkedIn can pull the correct information. The most common social markup standards include Open Graph (`og:`) and Twitter Cards (`twitter:`), both of which offer a richer, more controlled sharing experience by defining how content appears in previews.

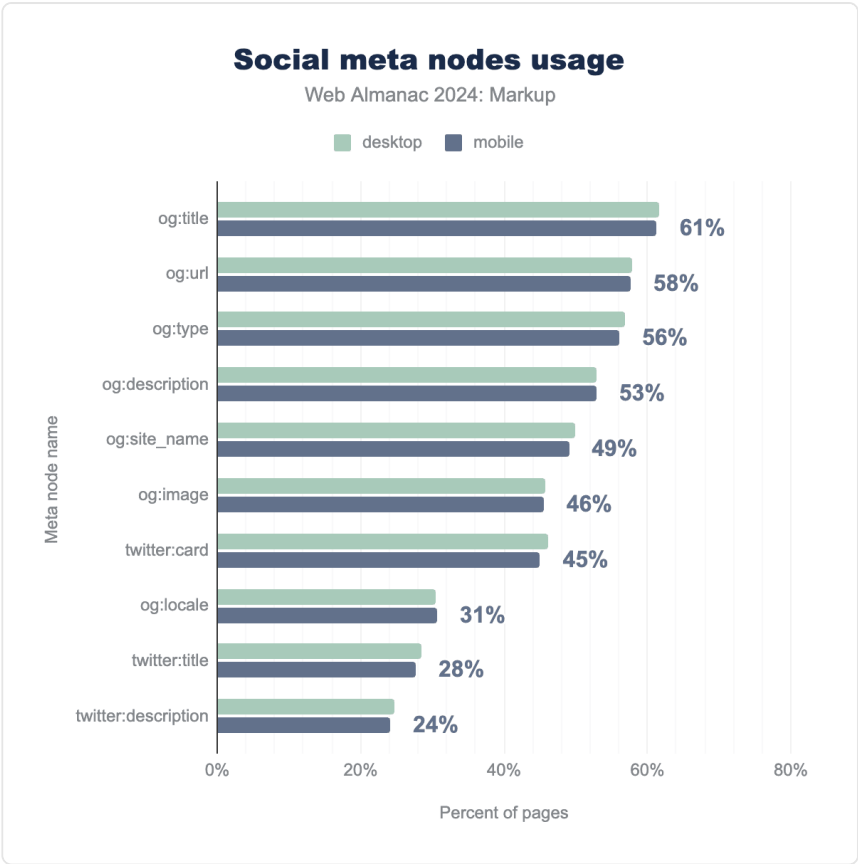


Figure 2.25. Popularity of top social meta nodes.

According to 2024 data, the most frequently used Open Graph meta tags are `og:title` (used by 61% of mobile pages) and `og:url` (58%). These tags define the title and canonical URL of the shared content, followed closely by `og:type` (56%) and `og:description` (53%), which offer insights into the content type and a brief summary. Twitter-specific meta tags like `twitter:card` (45%) and `twitter:description` (24%) are also still widely used, even though the platform is now branded as “X,” illustrating a lag in terminology updates across the platform.

Miscellaneous

In the preceding sections, we have provided an overview of HTML in general, as well as the

adoption of the most commonly used elements and attributes. In this section, we will undertake a deeper analysis of some special cases, including viewports, favicons, buttons, inputs, and links.

viewport specifications

The `viewport` meta tag specifies how the content should be scaled on various devices by setting properties like `width` and `initial-scale`. A common configuration, `width=device-width,initial-scale=1`, ensures that the page takes the full width of the screen and loads at the correct zoom level for mobile devices.

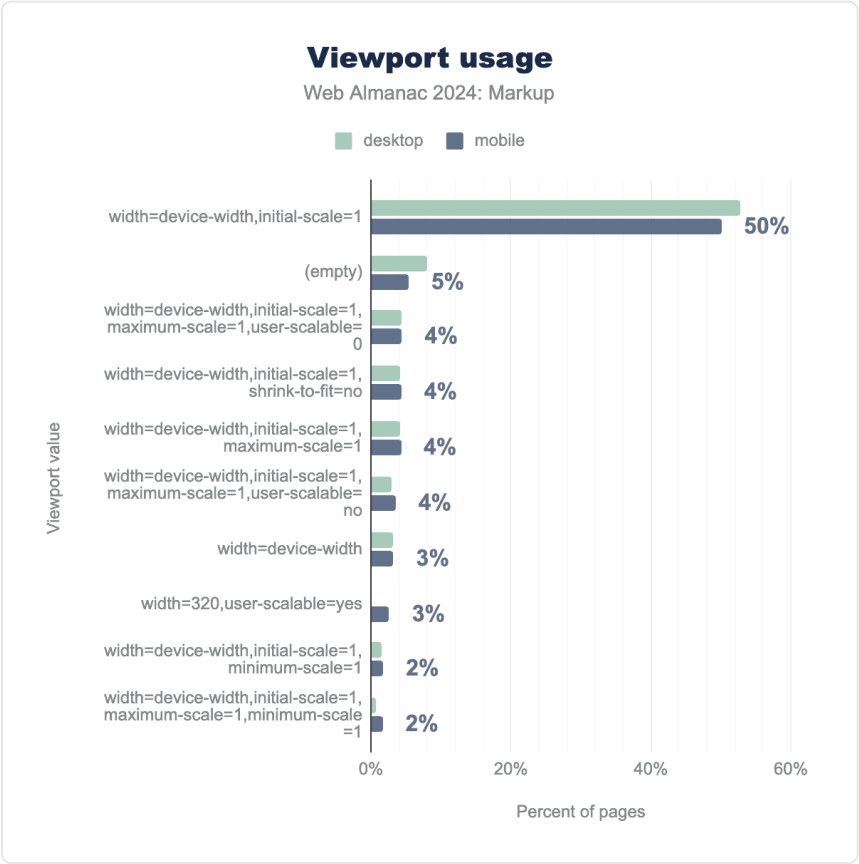


Figure 2.26. Meta viewport specifications.

In terms of current usage, the most common configuration is `width=device-width,initial-scale=1`, present on 50% of mobile pages. Interestingly, 5.4% of the pages

analyzed on mobile have no viewport tag. So, these pages are not designed for mobile experiences. Other configurations include variations like `width=device-width,initial-scale=1,maximum-scale=1,user-scalable=0`, which disables user scaling, found on 4.4% of mobile pages.

Favicons

Favicons, those small icons associated with websites, play an important role in enhancing the user experience and brand recognition. These icons are displayed in browser tabs, bookmarks, and even on mobile home screens when users save websites. One of the most interesting aspects of favicons is that they can work even without explicit HTML markup. Favicons support various image formats, including `.png`, `.ico`, `.jpg`, and `.svg`.

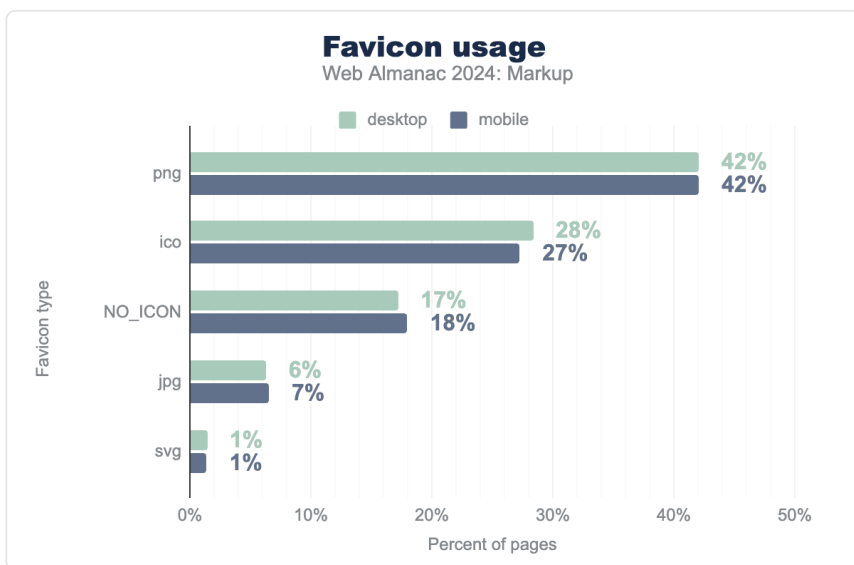


Figure 2.27. Popularity of favicon types.

As of 2024, `.png` is the most commonly used format for favicons informed by `<link rel="icon">` tags, appearing on 42% of mobile pages, up from 35% in 2021. In contrast, the use of `.ico` files has decreased from 33% in 2021 to 27%, likely due to developers moving away from this format in favor of other options like `.png` and `.svg`. However, it's interesting to highlight that `.svg` favicons are not supported on Safari³⁰.

Interestingly, about 18% of pages still lack a favicon, showing a slight improvement from the

30. <https://caniuse.com/link-icon-svg>

22% that had no favicon in 2021.

Buttons and input types

Buttons in web development have been a source of frequent debate due to their dual functionality and various use cases. The controversy typically revolves around when to use the native `<button>` element versus anchor (`<a>`) links or even custom-styled `div` elements acting as buttons. We won't get into that debate, but we will look at the data to review its usage.

73%

Figure 2.28. Mobile pages using at least one button element.

73% of mobile pages use at least one `<button>` element on them, a significant increase from 65.5% in 2021³¹. Like in 2021, we didn't run a query for input-typed buttons, but the Accessibility chapter has more very interesting information about buttons. You should read it too!

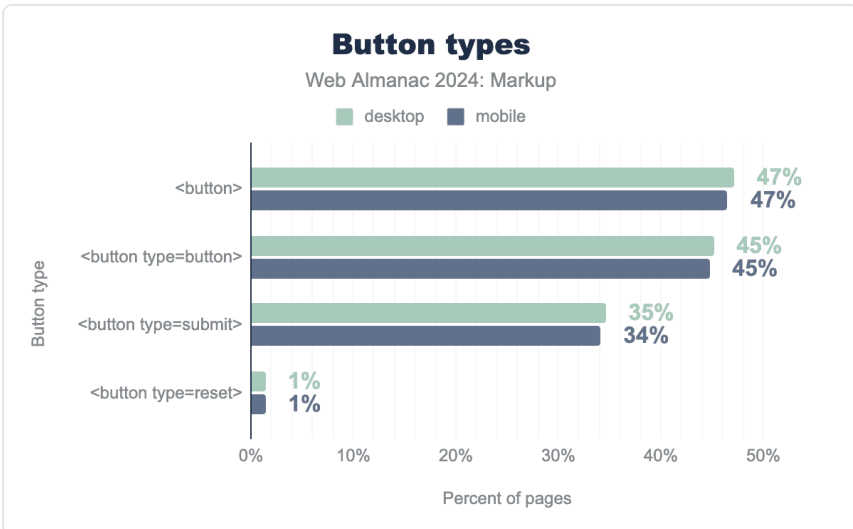


Figure 2.29. Popularity of button types.

Here's a closer look at the breakdown:

31. <https://almanac.httparchive.org/en/2021/markup#button-and-input-types>

- The generic `<button>` element appears on 46.5% of mobile pages. The button has no default behavior so it can have client-side scripts listen to the element's events.
- 44.7% of mobile pages use `<button type="button">`, which is typically employed for actions not associated with form submissions (e.g., triggering JavaScript functions).
- The `<button type="submit">` variant, used specifically for form submission, is present on 34.1% of mobile pages.
- `<button type="reset">` is relatively rare, seen on just 1.4% of mobile pages, indicating that resetting forms is less common or developers opt for custom solutions.

Apart from buttons, certain `input` elements are also rendered and used as buttons.

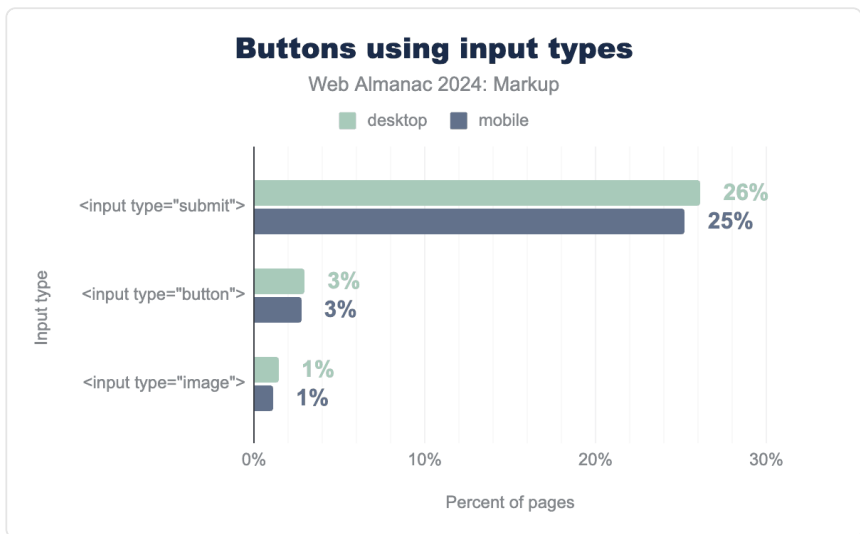


Figure 2.30. Popularity of buttons using input types.

Data shows that 25.2% of mobile pages in our data set have at least one `<input type="submit">` element, 2.8% have at least one `<input type="button">` element, and 1.1% have at least one `<input type="image">` element.

Link targets

In the past, if you linked to a page with a `target="_blank"` attribute to open it in a new tab,

the target page could access your page via `window.opener`, which could be exploited to perform malicious actions. To prevent this, developers had to add a `rel="noopener"` attribute to `target="_blank"` links. The `noopener` value ensures that the new tab doesn't have access to the `window.opener` object. In addition, `noreferrer` was often used in conjunction with `noopener` to prevent the referrer information from being passed to the new tab.

In modern browsers, this security issue has been resolved: now, when `target="_blank"` is used, browsers automatically apply `rel="noopener"` behind the scenes. This means that, in most cases, developers no longer need to manually include `noopener` in their link attributes to avoid the security vulnerability. Despite this, we still see a widespread use of `noopener` and `noreferrer` on many web pages, likely due to legacy code or developers being cautious about cross-browser compatibility.

Link	Desktop	Mobile
Has <code>target="_blank"</code>	81%	81%
Sometimes uses <code>target="_blank"</code> with <code>noopener</code> and <code>noreferrer</code>	77%	76%
Has <code>target="_blank"</code> without <code>noopener</code> and <code>noreferrer</code>	68%	67%
Has <code>target="_blank"</code> with <code>noopener</code>	25%	24%
Always uses <code>target="_blank"</code> with <code>noopener</code> and <code>noreferrer</code>	23%	24%
Has <code>target="_blank"</code> with <code>noopener</code> and <code>noreferrer</code>	20%	19%
Has <code>target="_blank"</code> with <code>noreferrer</code>	3%	3%

Figure 2.31. Adoption of various combinations of link attributes.

Looking at the data, 81% of pages use `target="_blank"`. Interestingly, 76% of pages include at least one `target="_blank"` link with `noopener` and `noreferrer` while 67% have `target="_blank"` without `noopener` and `noreferrer`. Additionally, 24% of mobile pages always use `target="_blank"` links with `noopener` and `noreferrer`.

Conclusion

The analysis of HTML usage in 2024 reveals significant trends and insights that underscore its evolution and the ongoing relevance of this foundational language in web development.

One of the most notable findings is the increasing standardization around the HTML doctype,

with 93% of mobile pages now using the standard `<!DOCTYPE html>`. This reflects a positive shift towards compliance with web standards, though XHTML remains present.

Document size has seen a slight increase, indicating a trend towards more complex pages, yet the use of compression—especially Brotli—has become more prevalent, which enhances load performance. However, the continued absence of compression in about 10% of HTML files suggests that there are still optimization opportunities for many developers.

The rise of custom elements usage, which has increased from 3.6% to 7.9%, indicates a growing trend for building richer, more interactive web experiences. The presence of obsolete items, while decreasing, still indicates the need for ongoing code maintenance and adoption of modern standards.

Surprisingly, the top `data-*` attribute list shows significant changes, with a completely different top 3 attributes in it. `data-id`, `data-load-time`, and `data-tagging-id` usage suggests that performance tracking and tagging have become more important in current web development.

However, some things remain stable from year to year. *Divitis* is still a thing, and `class` continues to be the sovereign of the attribute world.

Author



Estela Franco

X @guaca

📧 @https://toot.cafe/@guaca

🐦 @guaca.bsky.social

🎧 guaca

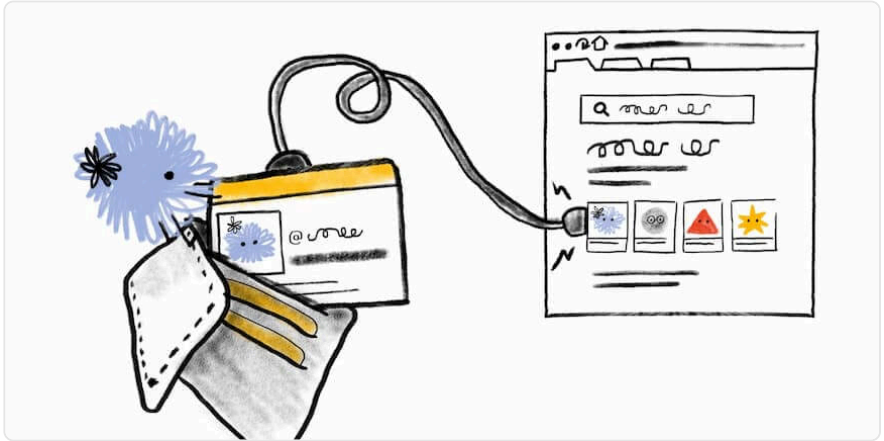
📄 estelafranco

🌐 <https://estelafranco.com/>

Estela Franco is a web performance and technical SEO specialist at Schneider Electric. But apart from that, she loves being connected to the community. That is why she is an international conference speaker, a Google Developer Expert in Web Technologies, a Storyblok ambassador, co-organizer of the Barcelona Web Performance Meetup, and co-founder of the Mujeres en SEO community.

Part I Chapter 3

Structured data



Written by Andrea Volpini

Reviewed by Jarno van Driel and Ryan Levering

Analyzed by Nurullah Demir

Edited by James Gallagher

Introduction

We find ourselves at a critical moment in the evolution of the semantic web. AI is becoming widely accessible and integrated into many of our day-to-day applications. Now, in its third year, this chapter provides a unique opportunity to analyze the past year's trends and examine the rapid developments occurring over time. Looking at the previous editions, we can offer a comprehensive view of where structured data stands today and where it's headed.

The expanding landscape of structured data

Over the past 18 months, there have been significant changes in the structured data landscape. In 2023, Google deprecated rich results for `FAQs` and `HowTos` from its search engine results pages (SERP) (source³²). In November 2024, Google will also remove the sitelinks search box

32. <https://developers.google.com/search/blog/2023/08/howto-faq-changes>

from search results (source³³). However, in parallel, there has been a new wave of innovation and expansion in using structured data from both Google and Bing.

Key developments in 2023–2024:

1. **New structured data types:** Google introduced several new types, including Vehicle listings, Course info, Vacation Rentals, and 3D Models for products. Also, in the ecommerce space, Google has integrated loyalty programs into its structured data offerings, particularly through the Merchant Center and Schema.org.
2. **Enhanced existing types:** Improvements to organization data, product variants, and the introduction of discount-rich results.
3. **Structured data carousels:** The beta launch of structured data carousels, combining `ItemList` with other types, opens new content presentation possibilities on Google's SERP (source³⁴).
4. **GS1 integrations:** There has been increased support for GS1 standards such as the GS1 Digital Link³⁵, which aims to bridge the gap between physical and digital product information. This technology enables manufacturers and retailers to connect physical products to their digital identities through QR codes. When scanned, these codes provide access to comprehensive product information, enhancing transparency and customer engagement. Also, the `gs1:CertificationDetails` property has been officially adopted by Google as `schema:Certification`, demonstrating how industry-specific extensions can successfully influence and integrate with Schema.org standards.
5. **Semantic data beyond search applications:** Structured data is now being leveraged beyond traditional search engines, playing a pivotal role in social web applications. For instance:
 - **Identity verification:** Platforms like Mastodon use `rel=me` links for two-way identity verification (source³⁶).
 - **Federated social networks:** The use of `rel=me` allows Mastodon users to verify their accounts with third-party websites (e.g., Ghost), strengthening cross-platform identity (discussion on `rel=me` with Ghost³⁷).

33. <https://developers.google.com/search/docs/appearance/structured-data/sitelinks-searchbox>

34. <https://developers.google.com/search/docs/appearance/structured-data/carousels-beta>

35. <https://www.gs1.org/standards/gs1-digital-link>

36. <https://docs.joinmastodon.org/user/profile/#verification>

37. <https://forum.ghost.org/t/verifying-mastodon-account-with-rel-me/34227>

- **New journalism features:** Mastodon recently introduced the `fediverse:creator` attribute to support content verification for journalists and publishers (source³⁸).

Beyond traditional implementation

As the structured data ecosystem matures, we're witnessing a diversification in implementation strategies. While search engines remain a primary consumer of structured data, its applications are expanding significantly:

1. **Schema.org as markup:** The traditional method of embedding structured data directly into webpages continues to be a cornerstone of modern SEO practices.
2. **Schema.org as a data standard:** Beyond its use in HTML, Schema.org is increasingly employed to standardize data shared via APIs or feeds. For example, Google's Data Commons³⁹ initiative uses an extended Schema.org vocabulary to integrate datasets from hundreds of organizations globally. This standardization supports tasks like dataset discovery and relationship mapping, crucial for understanding provenance, subsets, and derivations of datasets in AI-driven environments (source⁴⁰).
3. **Semantic data in social web applications:**
 - Platforms like Mastodon leverage structured data for identity verification. The `rel=me` attribute allows users to verify accounts across federated networks (source⁴¹).
 - Features like `fediverse:creator` are being used to validate content and authorship, enhancing trust in the decentralized social web (source⁴²).
4. **Digital Product Passports (DPPs):**

Structured data plays a key role in emerging regulatory requirements like the EU's Digital Product Passports⁴³, designed to enhance transparency and sustainability in ecommerce. These passports leverage GS1 Digital Links to provide comprehensive product information through QR codes.

38. <https://blog.joinmastodon.org/2024/07/highlighting-journalism-on-mastodon/>

39. <https://datacommons.org/>

40. <https://research.google/blog/relationships-are-complicated-an-analysis-of-relationships-between-datasets-on-the-web/>

41. <https://docs.joinmastodon.org/user/profile/#verification>

42. <https://blog.joinmastodon.org/2024/07/highlighting-journalism-on-mastodon/>

43. <https://wordlift.io/blog/en/digital-product-passport-implementation/>

5. Structured data for AI-powered Discovery:

As AI-powered search engines, chatbots, and conversational assistants continue to expand their reach, structured data plays a pivotal role in enhancing content discoverability and contextual understanding across these platforms. Key examples include:

- **AI Search Engines:** Platforms like Bing Chat and Google AI Overview utilize structured data not only to train their language models but also to deliver contextually rich and accurate responses. By leveraging structured data, these systems can interpret complex relationships between datasets, improve search relevance, and enable users to seamlessly navigate interconnected datasets (source⁴⁴).

These capabilities demonstrate structured data's evolving role in not only improving discoverability but also in enhancing AI systems' ability to interpret and act on relationships between data, thereby creating richer and more useful user experiences.

This diversification highlights structured data's growing role in facilitating data interoperability, social trust, regulatory compliance, and AI-driven content discovery. By enabling systems to understand and act on complex relationships between data, structured data lays the foundation for richer, more intelligent digital experiences.

Structured data in the age of AI and machine learning

The rise of generative AI and advanced machine learning has further underscored the importance of structured data:

- **Fact validation:** Structured data provides a parsable source for AI systems, enabling them to efficiently extract, interpret, and validate information. This helps:
 - **Combat misinformation:** AI can cross-reference structured data with other trusted sources to validate facts.
 - **Improve content understanding:** By offering clear entity definitions and relationships, structured data supports nuanced interpretation of complex topics.
 - **Enhance user experiences:** Structured data allows AI systems, such as chatbots and voice assistants, to deliver accurate and context-rich

44. <https://research.google/blog/relationships-are-complicated-an-analysis-of-relationships-between-datasets-on-the-web/>

responses to user queries.

- **Enhanced search understanding:** It enables a more nuanced interpretation of content by search engines and AI-powered systems.
- **Training data:** Well-structured data is high-quality training material for machine learning models.

What this chapter provides

This chapter offers a data-driven analysis of structured data trends in 2023-2024, highlighting key developments and best practices:

1. Evolution of the landscape:

- Key shifts in structured data, especially with the rise of AI-powered search like Google AI Overview and Bing Chat.
- Changes in Google and Bing structured data policies, and their impact on SEO.

2. Prevalence and growth:

- Trends in popular formats like JSON-LD, Microdata, and RDFa.
- Adoption rates by schema types such as `Product`, `Organization`, and `Article`.

3. Implementation and best practices:

- Best practices for structured data, including JSON-LD usage.
- Common mistakes and how to avoid them.

4. Rich results & SERP features:

- Effects of deprecated features like `FAQ` and `HowTo`.
- Introduction of carousels and product knowledge panels.

5. AI-Powered search:

- The role of structured data in AI-driven search and voice assistants.
- Trends in AI-powered content discovery.

6. Ecommerce innovations:

- Growth of Digital Product Passports and GS1 Digital Links.
- Structured data's role in ecommerce and new rich result types.

7. Knowledge graphs & Graph RAG:

- The rising importance of knowledge graphs and Graph RAG for enhancing AI outputs.

8. Quality & data integrity:

- Best practices for maintaining high-quality structured data.

9. Emerging schemas & use cases:

- Innovations in schema types and their application in SEO and ecommerce.

10. Future outlook:

- The evolving role of structured data in AI, semantic SEO, and content discovery.

This chapter provides a comprehensive view of structured data's impact on SEO, AI, and ecommerce, with actionable insights for developers and marketers.

Key concepts

As structured data evolves in complexity, exploring and explaining key concepts is crucial before diving into a deeper analysis. This section outlines fundamental ideas and recent developments in the field.

Linked data and the semantic web

Linked data remains a cornerstone of structured data. By adding structured data to web pages and providing URI links to referenced entities, we create an interconnected web of information. This contributes to the semantic web, where data is linked through the Resource Description Framework (RDF), enabling machines to treat web pages as databases.

The concept of semantic triples (subject-predicate-object) continues to be fundamental in expressing relationships between entities. While SPARQL⁴⁵ is a query language specifically designed for querying graph data and RDF triples, GraphQL⁴⁶ serves as a flexible API query language for retrieving structured data from diverse backends, including databases and microservices. These tools complement each other: SPARQL excels in querying RDF datasets for semantic web applications, while GraphQL simplifies access to structured data for web and mobile applications.

Open data and the 5 stars model

Tim Berners-Lee's 5 stars of the open data model remain relevant. It emphasizes the importance of web-available, structured, non-proprietary, URI-identified, and interlinked data. Structured data plays a crucial role in achieving higher levels of this model, contributing to a more open and interconnected web ecosystem.

AI-Powered search, voice assistants, and digital assistants

The landscape of search and digital assistance has dramatically evolved with the integration of AI, LLMs, and advanced natural language processing. This convergence has blurred the lines between traditional search engines, voice-activated systems, and AI-powered digital assistants.

Semantic search engines and AI-powered search

Semantic search has progressed beyond traditional keyword matching to include sophisticated AI-powered experiences. These systems leverage structured data to provide more accurate, contextual, and often conversational search results. Key developments include:

- Google AI Overview: A feature that provides comprehensive (sometimes misleading) AI-generated summaries on complex topics.
- Microsoft Bing Chat: Integrates chat-based AI interactions directly into Bing search

45. <https://wikipedia.org/wiki/SPARQL>

46. <https://wikipedia.org/wiki/GraphQL>

results.

- **Meta AI:** Meta's AI assistant is integrated across platforms like Facebook, Messenger, Instagram, and WhatsApp.
- **SearchGPT (and ChatGPT):** OpenAI's AI search engine that integrates search results into conversational responses.
- **Perplexity.ai:** An AI-powered search engine that provides detailed, sourced answers to queries.
- **You.com:** Offers AI-summarized search results and a chat interface for more interactive searching.

These platforms demonstrate an enhanced ability to understand user intent and context, significantly improving search accuracy and user experience. They often combine traditional web indexing with real-time information retrieval and natural language generation.

The role of structured data

Structured data plays a crucial role in these AI-powered systems by:

1. **Enhancing entity recognition:** Helping systems accurately identify and disambiguate entities mentioned in queries.
2. **Providing context:** Offering additional information about entities and their relationships, improving response accuracy.
3. **Facilitating knowledge graph integration:** Allowing these systems to tap into vast, interconnected information databases.
4. **Enabling rich responses:** Supporting the generation of detailed, multi-faceted answers that often include visual elements or interactive features.
5. **Improving voice query interpretation:** Assisting in understanding the intent behind spoken queries, which can be more ambiguous than text-based searches.

While it is still challenging to assess the impact of structured data on Generative AI and AI search engines, in some cases, such as geo-referencing queries, we can observe the early emergence of entities in the user experience of Perplexity.ai and You.com.

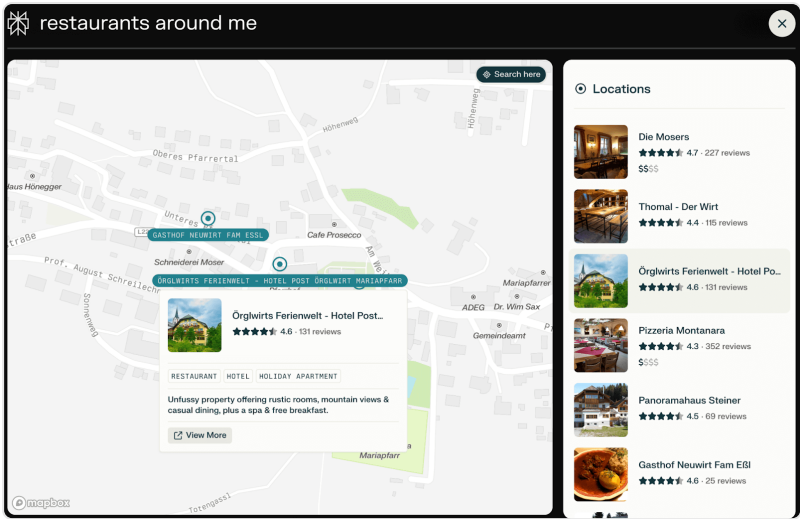


Figure 3.1. Search results in Perplexity, showcasing a map view with listings of local restaurants.

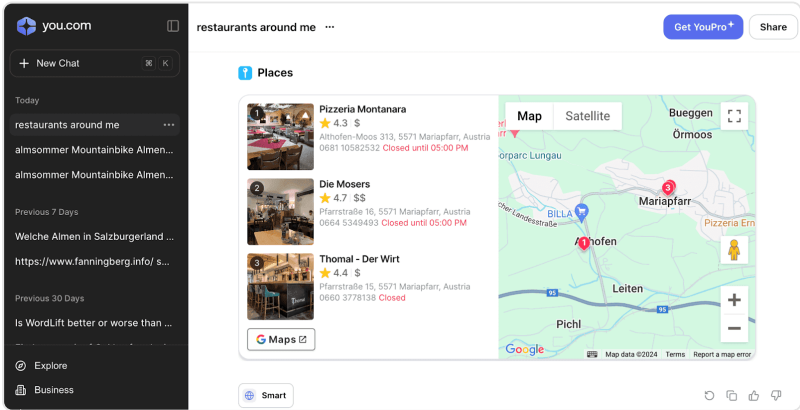


Figure 3.2. Search results in You.com with a map and restaurant listings.

This is way more consistent when interacting with Bing Copilot or Gemini by Google.

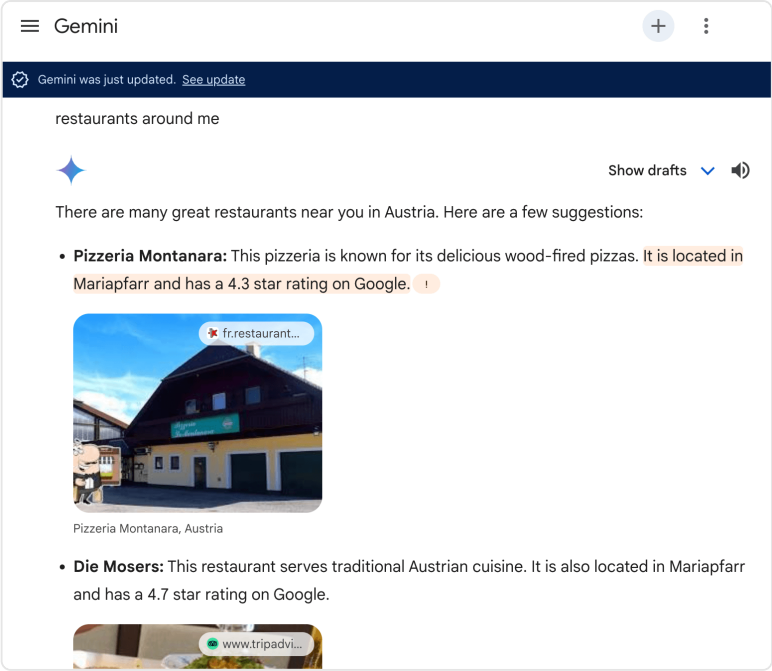


Figure 3.3. Google Gemini.

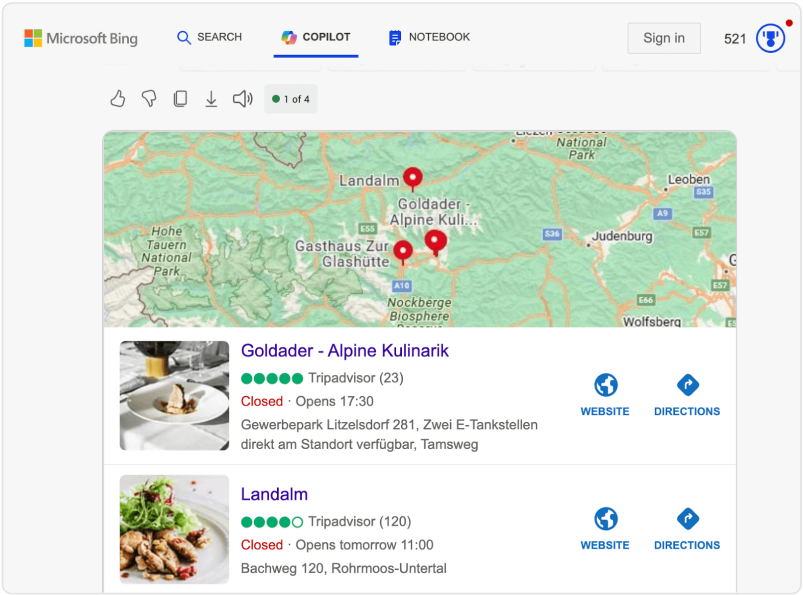


Figure 3.4. Bing Copilot.

Empirically, AI-powered search systems, as seen above, are sourcing data from a variety of established knowledge bases and authoritative platforms:

- Map services: Google Maps and Bing Maps serve as crucial data sources for location-based information.
- Authoritative websites: Platforms rich in structured data markup, such as TripAdvisor, contribute significantly to the knowledge base of AI search systems.
- Vertical-specific databases: Industry-specific databases and platforms provide specialized information for AI-powered search in various sectors.

The shift to AI-powered search and its implications

This transition from traditional search to AI-powered search demands a broader, more nuanced approach to optimization:

1. **Multi-platform visibility:** SEO strategies must now account for visibility across a diverse array of AI surfaces and platforms, including:
 - Traditional search engines (Google, Bing)
 - AI chatbots (ChatGPT, Google's Gemini, Perplexity, Anthropic's Claude)
 - Integrated assistants (Microsoft Copilot, potential Apple-ChatGPT integration)
 - Ecosystem-specific tools (Google Workspace, Microsoft 365)
 - Browser and device-level integrations
2. **Beyond conventional optimization:** Success in this landscape goes beyond optimizing for specific features like Google's AI Overview. It requires a holistic approach to making content discoverable and comprehensible across all emerging search interfaces.
3. **Leveraging structured data strategically:** The key to improved visibility lies not just in publishing structured data using schema markup but in facilitating access to structured information about entities that matter to your business or content. This involves:
 - Ensuring clear, structured information is available and easily interpretable by various AI systems.

- Ensuring that the metadata used to describe the webpage for bots is consistent with the content presented to human readers.
- Directly feeding accurate information to relevant platforms and marketplaces (e.g., Google Merchant, Amazon) for products and services.

Rich results and knowledge panels

Rich results and knowledge panels, powered by structured data, are essential features of search engine results pages (SERPs). These enhanced displays offer users immediate and relevant information, significantly boosting click-through rates and user engagement. As rich results become more diverse and sophisticated, they present new opportunities for content visibility. A recent example from Google is the introduction of a structured data carousel for listicle pages related to local businesses (including subtypes like restaurants, hotels, vacation rentals), products, and events.

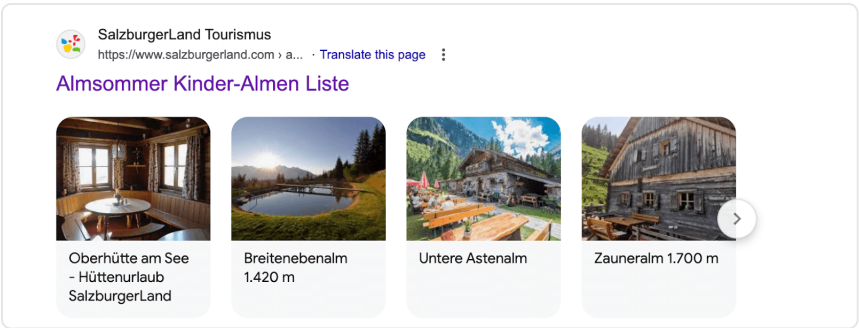


Figure 3.5. An example of the new beta carousel rich result.

This carousel format enhances the display of structured data for listicle pages, offering users quick access to multiple options, such as local businesses or products, directly on the SERP.

Another notable example, while not directly influenced by structured data, is the new Google Merchant knowledge panel, which extends the functionality of the product knowledge graph panel. Structured data acts as a signal that contributes to entity disambiguation, helping search engines accurately identify businesses and their attributes, which can lead to the appearance of these panels. This feature helps businesses, both small and large, build trust with users by displaying key information about the merchant directly on Google's search results page.

Knowledge graphs and Graph RAG

Knowledge graphs have become increasingly central to structured data applications, encapsulating factual information through precise, explicit triple representations (source⁴⁷). They provide a powerful way to represent and query complex relationships between entities while offering transparent symbolic reasoning capabilities. The emergence of Graph RAG⁴⁸ (Retrieval-Augmented Generation) represents a significant advancement, combining knowledge graphs with large language models to enhance AI-generated responses with verifiable, structured information while addressing the challenges of factual inconsistencies and opacity inherent in LLMs.

Difference between Labeled Property Graphs and RDF graphs

Labeled Property Graphs (LPGs) and Resource Description Framework (RDF) graphs are two distinct approaches to organizing and representing data. LPGs, commonly used in databases like Neo4j, structure data with nodes and relationships, each carrying labels and properties. This allows for a flexible and intuitive way to model complex data relationships. On the other hand, RDF graphs, which are foundational to the semantic web, use a triple-based structure (subject-predicate-object) to represent data. RDF emphasizes interoperability and standardization, making it ideal for linking data across different systems and domains. While LPGs offer ease of use and performance for certain applications, RDF provides a robust framework for semantic data integration and reasoning.

The importance of structured data in creating knowledge graphs cannot be overstated. Structured data enables the precise definition of entities and their relationships, which is crucial for the development of accurate and reliable knowledge graphs. By leveraging structured data, organizations can build comprehensive knowledge graphs that enhance data discoverability, interoperability, and the overall quality of AI-generated insights.

Data Commons

Data Commons⁴⁹ is an open-source and open-data initiative by Google that organizes public datasets from various global sources, such as the United Nations and national census bureaus, to make them universally accessible. The platform provides over 250 billion data points and 2.5 trillion triples, encompassing a wide range of statistical variables. Schema.org is utilized to encode structured data in Data Commons, creating a unified knowledge graph that standardizes and normalizes diverse datasets, enabling easier access and exploration through a

47. <https://arxiv.org/pdf/2407.18470>

48. <https://microsoft.github.io/graphrag/>

49. <https://datacommons.org/>

common framework. This structured approach helps to integrate vast amounts of data into a coherent, searchable system.

Digital Product Passports and GS1 Digital Link

In the ecommerce and supply chain sectors, Digital Product Passports⁵⁰ (DPPs) and the GS1 Digital Link standard⁵¹ are revolutionizing how product information is shared and accessed. These technologies leverage structured data to create comprehensive, easily accessible digital representations of physical products, enhancing traceability, sustainability efforts, and consumer information access.

AI, machine learning, and structured data

The synergy between structured data and AI/ML has deepened. Structured data is crucial in training machine learning models, providing consistent, machine-readable labels. It's particularly important in areas such as:

- **Large Language Models (LLMs):** Fine-tuning with structured data for improved performance in specific domains.
- **Explainable AI:** Using knowledge graphs to trace and explain AI decision-making processes.
- **Multimodal AI:** Linking different data types (text, images, video) in AI systems.

Semantic SEO and data quality

SEO has evolved beyond simple keyword matching into what we now call Semantic SEO⁵². This modern approach leverages structured data and contextual understanding to help search engines provide more accurate results. By implementing structured metadata and focusing on topical relationships, websites can build deeper meaning into their content. This allows search engines like Google and Bing to better understand user intent, rather than just counting keyword frequency.

By implementing semantic SEO, businesses can create content clusters based on topics, not just

50. <https://wordlift.io/blog/en/digital-product-passport-implementation/>

51. <https://www.gs1.org/standards/gs1-digital-link>

52. <https://wordlift.io/blog/en/entity/semantic-seo/>

individual keywords, making their content more discoverable and contextually relevant across various search platforms, including voice search assistants. This approach significantly boosts search engine rankings and user engagement, as structured data allows search engines to understand the content at a more granular level, making it easier to match user intent.

Data quality plays a key role here as well. High-quality structured data ensures consistency and accuracy, which is crucial not only for search engines but also in combating misinformation. It helps maintain trustworthiness across the web, especially as structured data is increasingly used in AI-powered systems like knowledge graphs for fact validation and enhancing large language model (LLM) training.

For example, organizations such as EssilorLuxottica, L'Oréal, Walmart, Shiseido and others are using semantic technologies like knowledge graphs to link content and provide users with more detailed, contextually relevant results. This practice also aids in AI-powered content discovery and makes content easier to retrieve through Generative Search like Perplexity or You.com.

Investing in semantic SEO and maintaining high-quality structured data not only enhances search visibility but also lays a foundation for future-proofing content for AI-driven discovery.

A year in review

The landscape of structured data implementation continues to evolve. To better understand this landscape, it's essential to distinguish between *syntax/encoding* and *vocabularies*:

- **Syntax/encodings:** These define how structured data is embedded into webpages:
 - **RDFa:** Maintains a strong presence, used on 66% of pages.
 - **JSON-LD:** Growing in popularity, implemented on 41% of pages.
 - **Microdata:** Steady usage, appearing on 26% of pages.
 - **HEAD data:** Includes non-RDFa meta tags like Twitter Cards.
- **Vocabularies:** These define the meaning and semantics of the data:
 - **Open Graph Protocol (OGP):** Widely used vocabulary, often encoded as RDFa (64% of pages).
 - **Twitter meta tags:** Expanding rapidly, appearing on 45% of pages.

- **Schema.org**: Flexible vocabulary used across multiple syntaxes.
- **Dublin Core**: Niche use cases, typically encoded as RDFa.
- **Microformats**: Primarily implemented using class-based metadata.

Structured data usage trends (2022–2024)

The data reveals notable trends in both syntax and vocabulary usage:

- **RDFa and Open Graph**: Dominant, with adoption on 66% and 64% of pages, respectively.
- **JSON-LD**: Continues its upward trajectory, increasing from 34% in 2022 to 41% in 2024.
- **Twitter meta tags (HEAD data)**: Significant growth, now at 45%.
- **Microdata**: Steady at 26%, primarily used in legacy contexts.
- **Facebook meta tags**: Declined to 7%, reflecting a shift to Open Graph.
- **Dublin Core and Microformats**: Minimal usage, each below 1%.

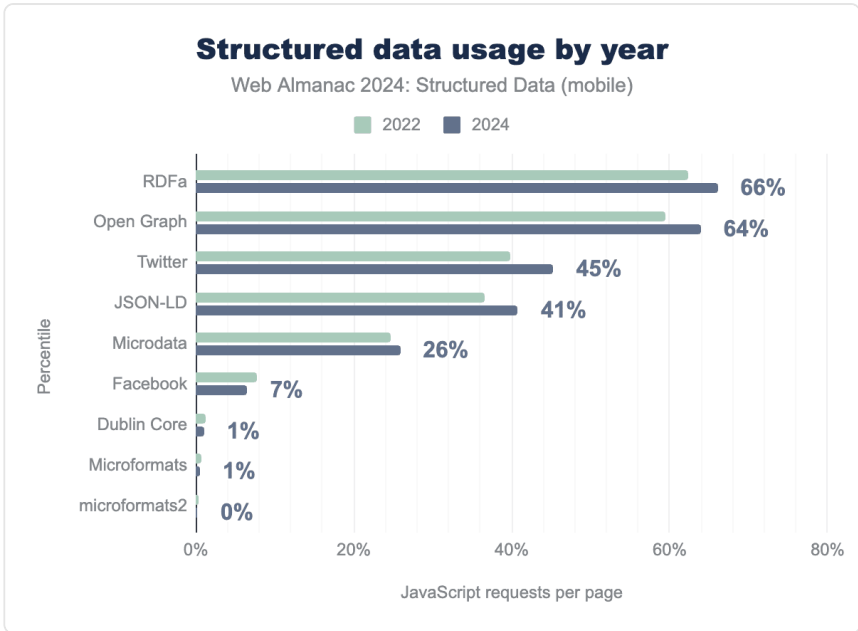


Figure 3.6. Structured data usage by year on mobile.

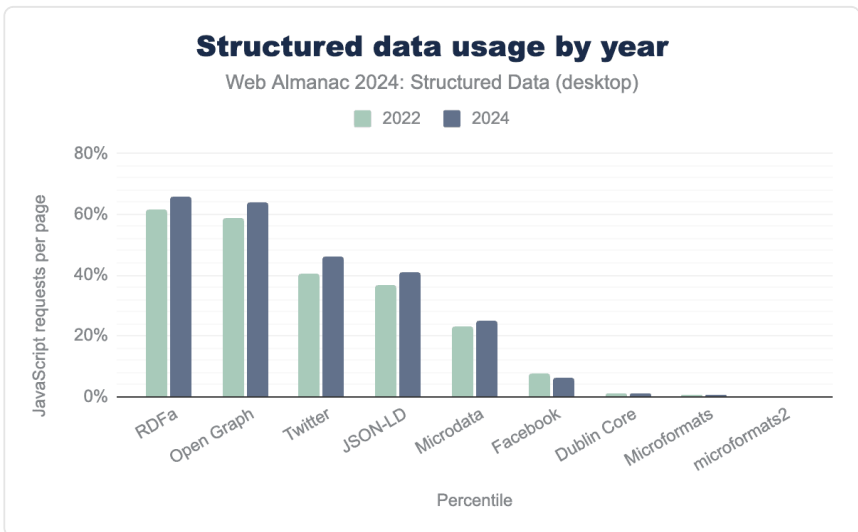


Figure 3.7. Structured data usage by year on desktop.

Platform differences between desktop and mobile implementations are becoming less pronounced, suggesting a shift toward standardized structured data strategies across devices.

This trend aligns with the growing reliance of search engines and AI systems on structured data for better content understanding and presentation.

Comparison of JSON-LD, Microdata, and RDFa usage

The three main structured data formats show distinctly different adoption patterns:

- **RDFa:** Highest adoption at 66% of pages. Most prevalent on legacy CMS platforms. Common implementations: Navigation elements (breadcrumbs), Basic page structure, Image and document metadata, List items.
- **JSON-LD:** Present on 41% of pages (up from 34% in 2022). Growing fastest among the three formats, preferred by Google and gaining wider developer adoption. Most commonly used for: organization data, local business information, product listings, articles and creative works.
- **Microdata:** Present on 26% of pages. Showing steady but slower growth. Primarily used for webpage structure (8.34% of pages), site navigation (6.42%), headers and footers (5.97% and 5.33%), organization information (4.87%) |

Let's analyze now more in detail each type.

RDFa

RDFa continues to play a significant role in structured data, particularly within legacy CMS platforms. However, there has been a noticeable shift towards using RDFa for navigation elements, such as `listitem` and `breadcrumblist`, which are now prevalent on a significant portion of web pages. This reflects an industry-wide emphasis on enhancing structured navigation data for better user experience, particularly on mobile platforms.

In contrast, traditional RDFa types like `foaf:image` and `foaf:document` have seen declining usage, as newer formats like JSON-LD and Open Graph offer more flexible solutions for image and document metadata. The adoption of Schema.org types within RDFa, such as `schema:webpage`, has shown modest but stable growth, further indicating a shift towards Schema.org vocabularies.

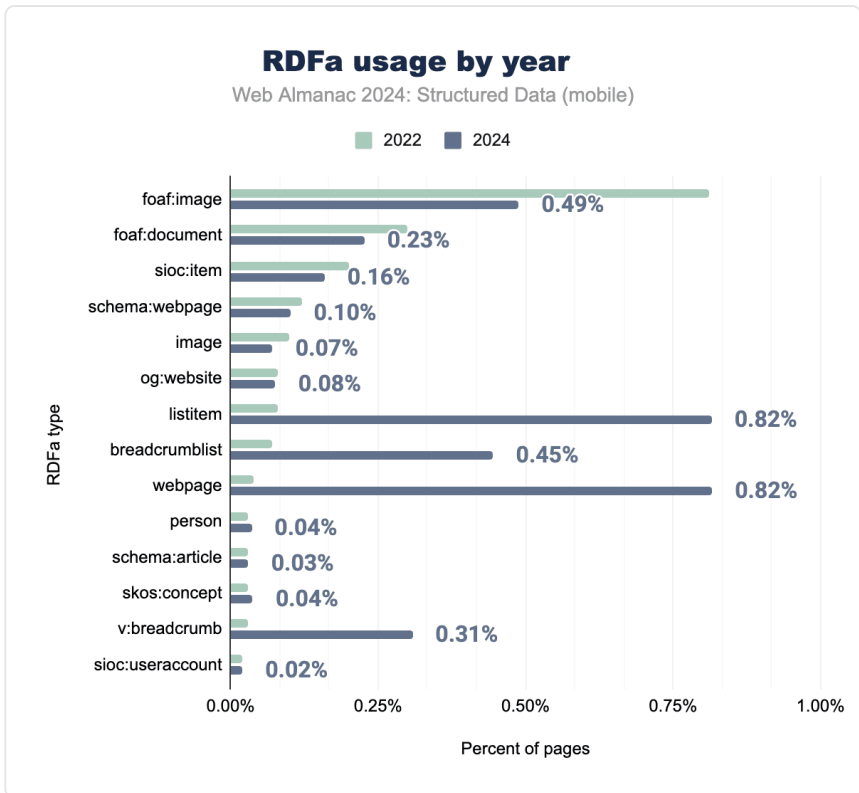


Figure 3.8. RDFa usage by year on mobile.

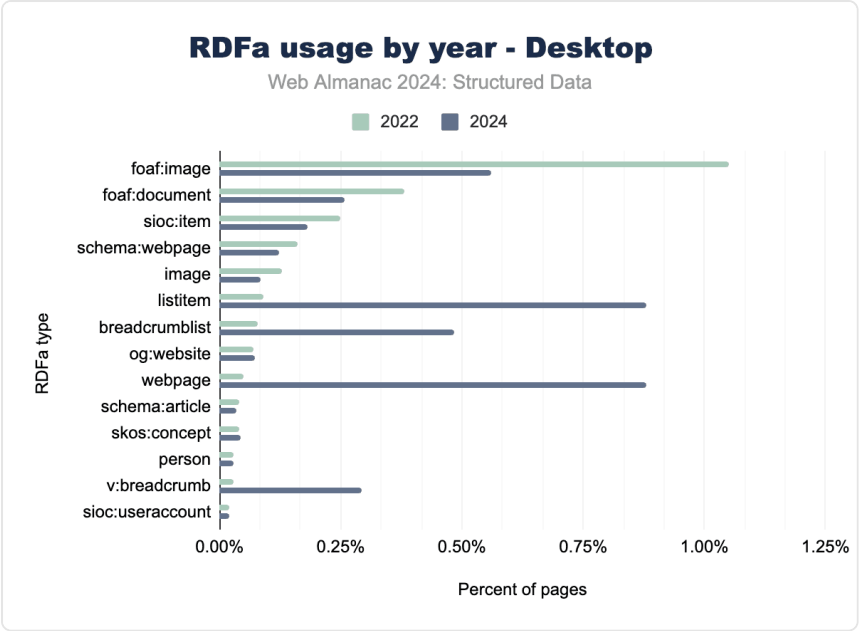


Figure 3.9. RDFa usage by year on desktop.

The data suggests that while RDFa remains a valuable tool, its dominance is gradually being overtaken by modern structured data formats like JSON-LD, particularly in dynamic content applications.

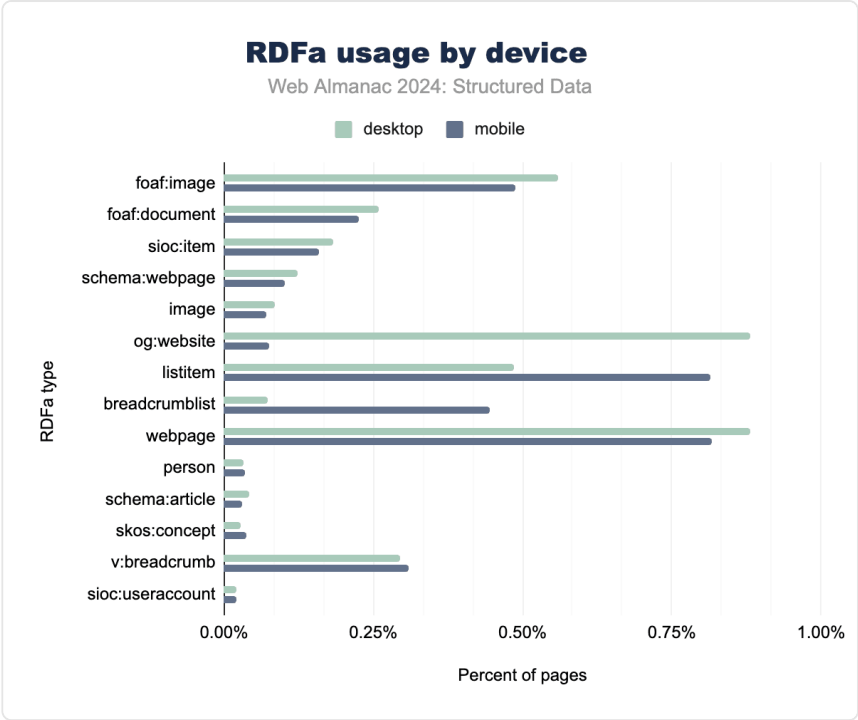


Figure 3.10. RDFa usage by device (desktop vs mobile).

Dublin Core

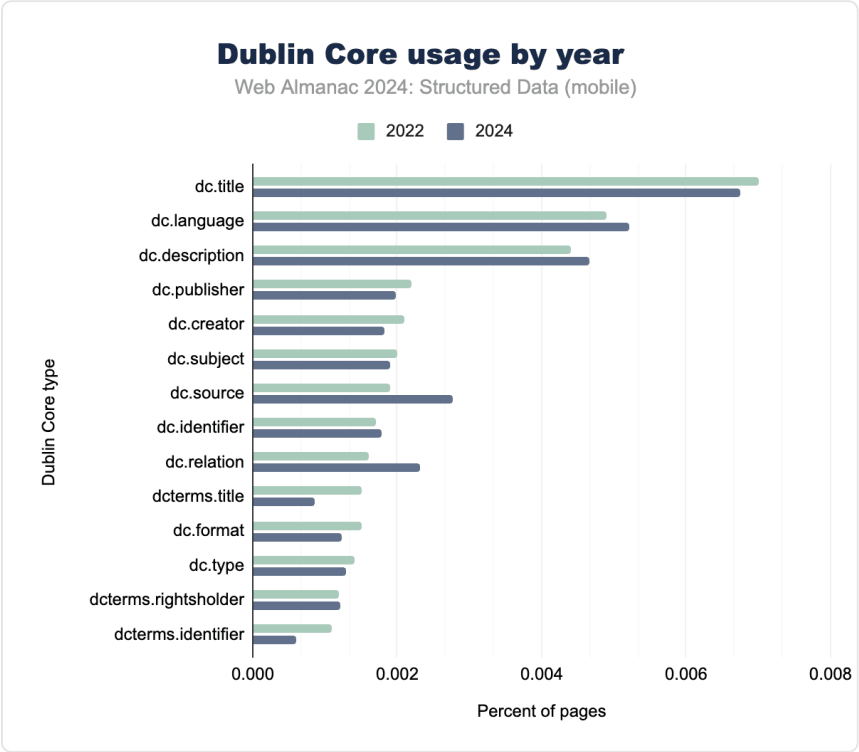


Figure 3.11. Dublin Core usage by year on mobile.

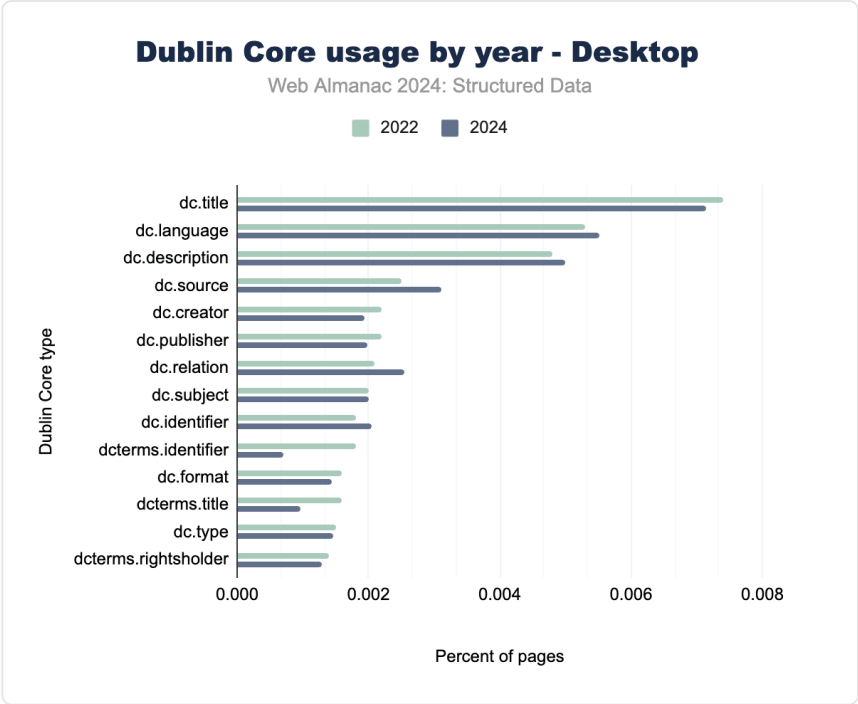


Figure 3.12. Dublin Core usage by year on desktop.

Dublin Core remains a stable but less frequently used format for metadata, especially when compared to modern formats like JSON-LD and Open Graph.

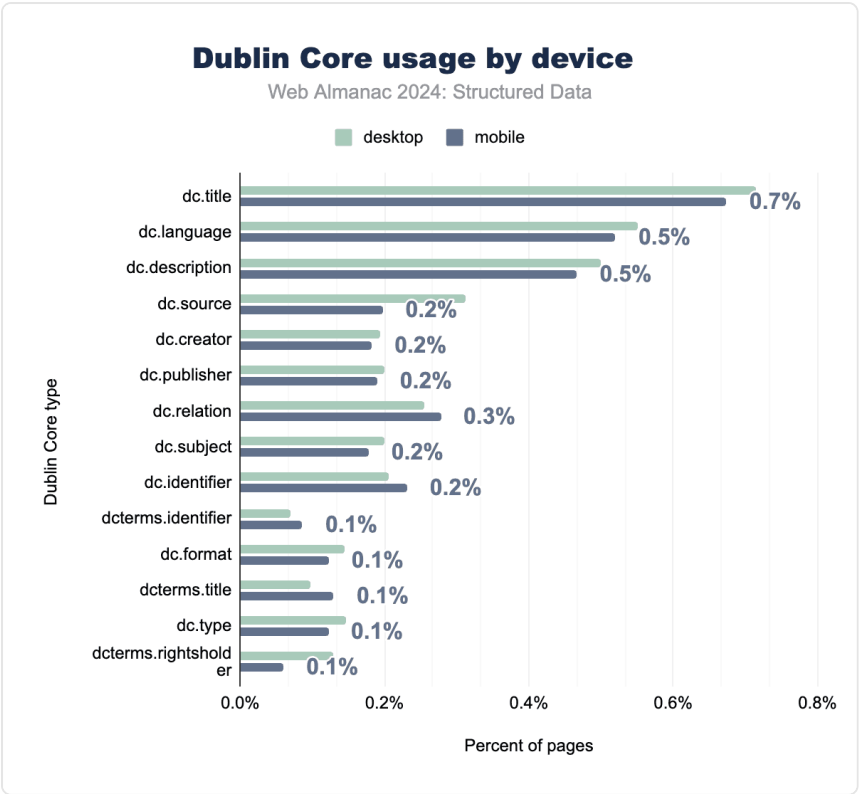


Figure 3.13. Dublin Core usage by device (desktop vs mobile).

Its key fields, such as `dc.title` and `dc.language`, show minimal year-on-year changes, maintaining a consistent presence primarily in academic and legacy web projects.

An increase in the use of `dc.source` reflects a growing emphasis on citing original sources, while fields like `dc.identifier` continue to be crucial for resource identification. However, specialized fields such as `dcterms.identifier` have seen declining adoption, signaling that Dublin Core is less central in today’s web environments.

Interestingly, Dublin Core retains relevance in multilingual document management, particularly through the `dc.language` field, which is essential for managing and categorizing content in multiple languages. This makes it a valuable tool in contexts where document metadata needs to support internationalization and localization efforts.

Overall, while Dublin Core is being gradually outpaced by more versatile formats like JSON-LD, it continues to serve niche needs where structured document metadata and multilingual support are critical.

Open Graph

Open Graph continues to be one of the most widely implemented structured data formats, particularly in the context of **social media sharing**. The `og:image` tag remains the most frequently used property, reflecting the growing emphasis on visual content optimization. Other image-related tags, such as `og:image:width` and `og:image:height`, have also seen a steady increase in adoption as websites strive to enhance the presentation of shared content across platforms.

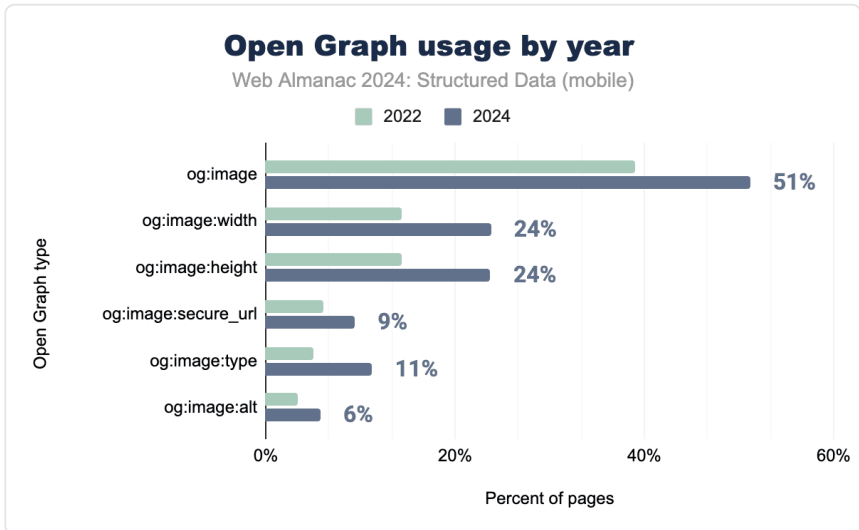


Figure 3.14. Open Graph usage by year on mobile.

A key development in 2024 is Google's update to its search documentation, now including the **`og:title` meta tag** as a source for generating title links in search results. This update allows Google to consider the `og:title` tag alongside traditional sources, such as the HTML `<title>` tag, when determining how clickable titles are displayed in search results. As a result, the `og:title` tag has gained renewed significance, not only for social media visibility but also for SEO.

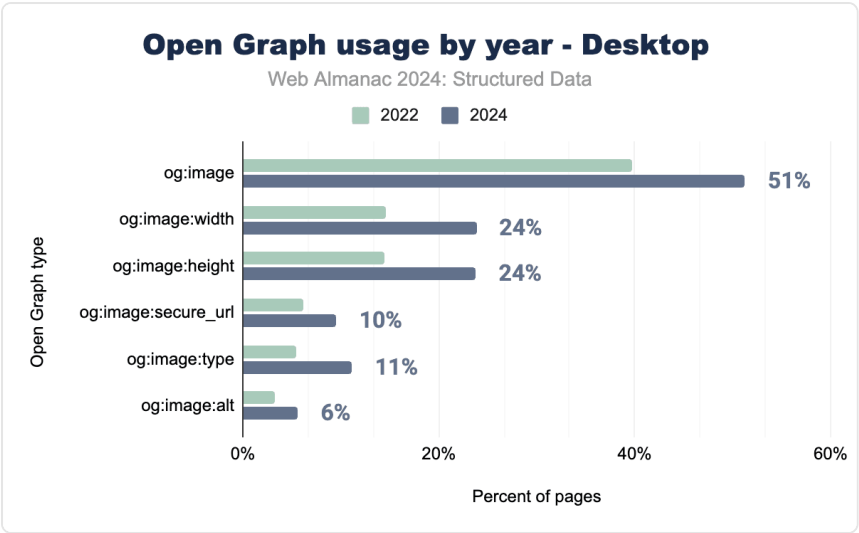


Figure 3.15. Open Graph usage by year (desktop).

This dual role of Open Graph in social sharing and search engine optimization makes it a critical tool for webmasters looking to improve both user engagement on social platforms and visibility in search results.

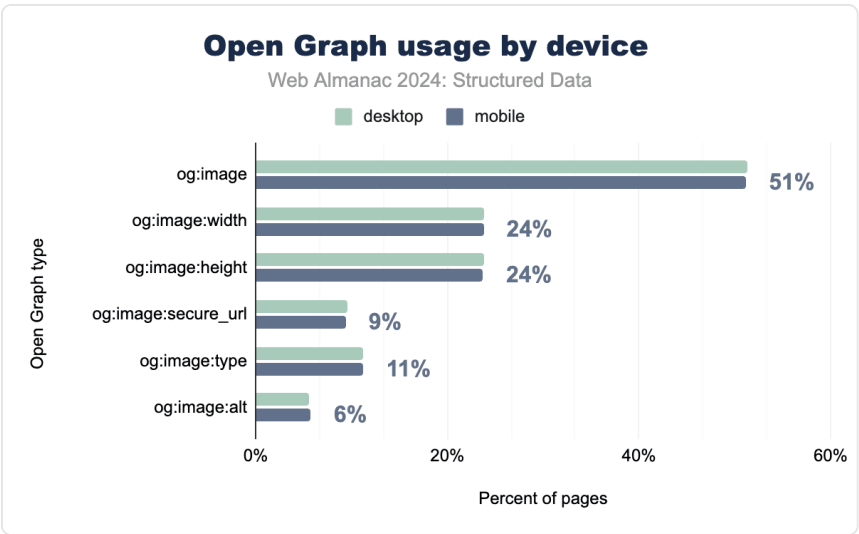


Figure 3.16. Open Graph usage by device.

Security and type-related properties have also gained traction. The `og:image:secure_url` property, which ensures image URLs are served over secure HTTPS connections, has increased to 9.41% on mobile and 9.56% on desktop. Similarly, `og:image:type`, which specifies the MIME type of the image, has grown to 11.26% on mobile and 11.17% on desktop. These properties help ensure consistent and secure media delivery across devices and platforms.

Twitter

Despite the platform's transition to new ownership and its rebranding as X, Twitter's meta tags remain a vital part of the structured data landscape, particularly in the realm of social media optimization. The `twitter:card` tag continues to dominate, showing significant growth across mobile and desktop pages, as it plays a key role in defining how content is displayed when shared on the platform.

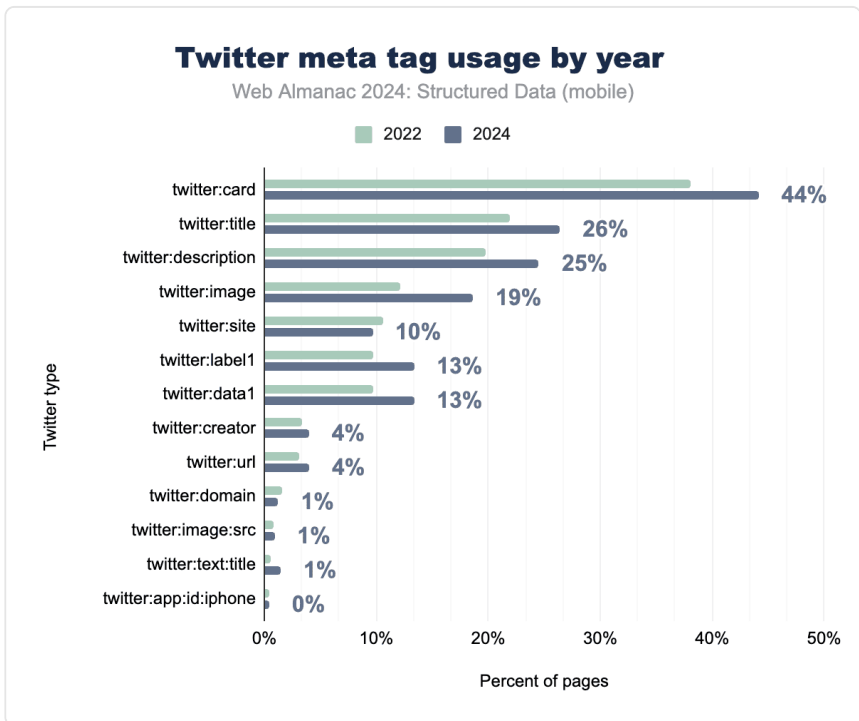


Figure 3.17. Twitter meta tag usage by year (mobile).

Core descriptive tags like `twitter:title` and `twitter:description` have also seen widespread adoption, appearing on approximately 26% of mobile pages and 24% of desktop

pages. These tags are essential for content previews, enhancing how web pages appear when shared on social media, and ensuring key information is highlighted.

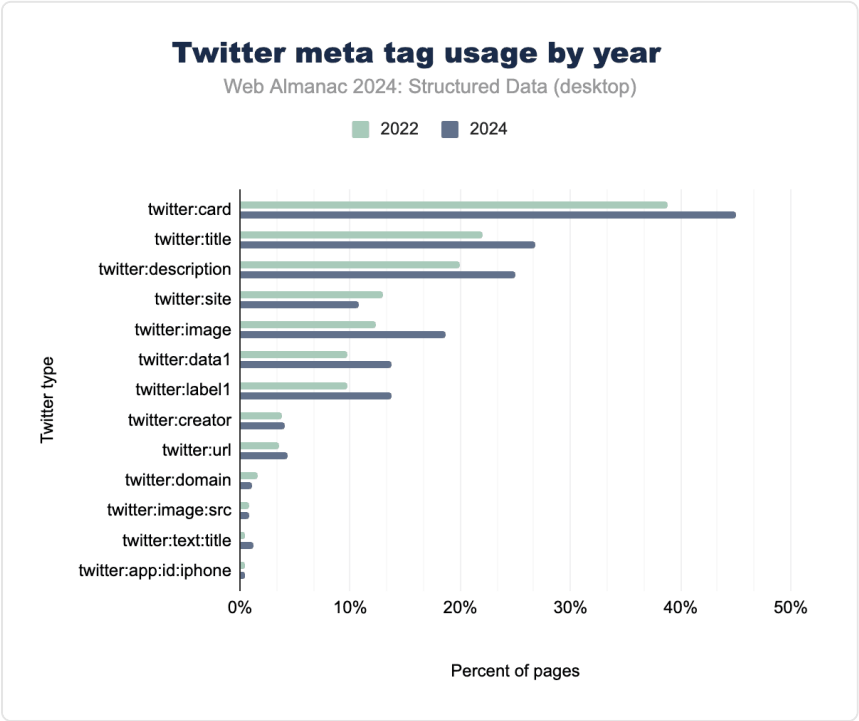


Figure 3.18. Twitter meta tag usage by year (desktop).

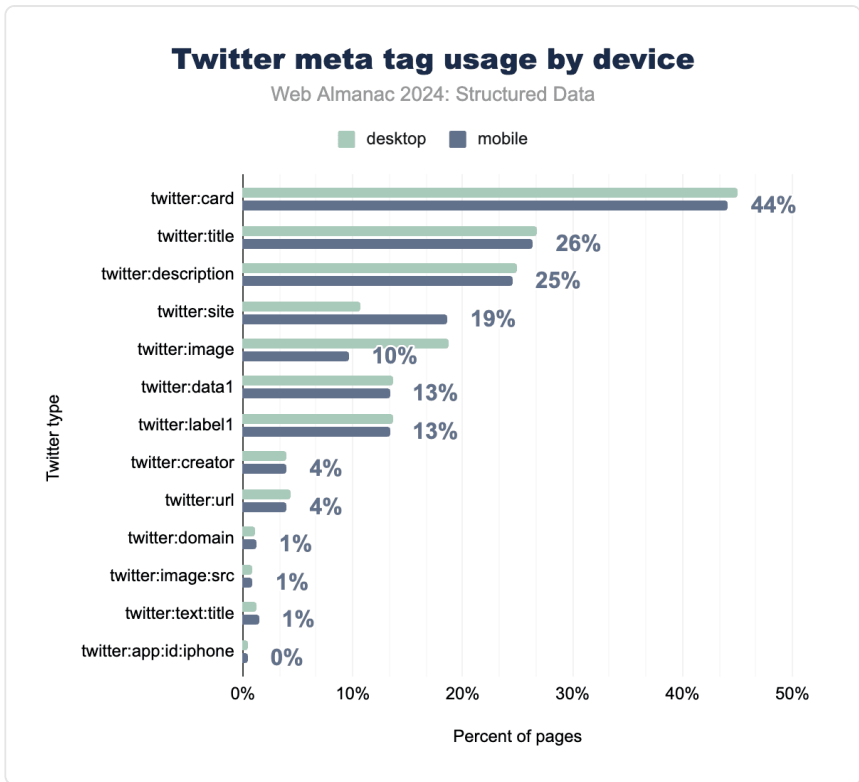


Figure 3.19. Twitter meta tag usage by device.

The platform's enhanced metadata properties, such as `twitter:data1` and `twitter:label1`, which support rich card features, have seen coordinated growth, now appearing on 13.36% of mobile pages. This indicates the increasing use of Twitter Cards for more detailed content representations, such as for product listings or event details.

While X has undergone major branding changes, the metadata architecture it introduced remains critical for webmasters and SEO professionals, ensuring content shared on social media is engaging, informative, and optimized for interaction. This highlights the platform's enduring importance in the social media and metadata ecosystem.

Facebook

Facebook-specific meta tags have seen a marked decline in usage between 2022 and 2024, reflecting the broader industry shift toward Open Graph as the preferred format for social

sharing metadata. The `fb:app_id` tag, once widely used to integrate apps with the Facebook platform, now appears on only 4.9% of mobile pages, down from previous years. Similarly, administrative tags like `fb:admins` have dropped to just 2.4%, serving primarily for backend management rather than enhancing content visibility.

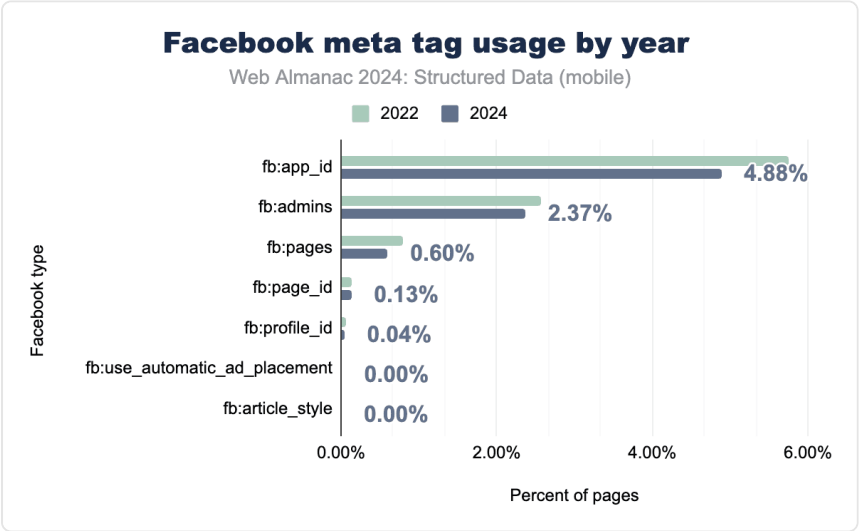


Figure 3.20. Facebook meta tag usage by year (mobile).

This decline underscores a strategic move by developers and webmasters to adopt Open Graph, which originated with Facebook but has since become the standard for social media sharing across platforms. The Open Graph format offers greater flexibility and interoperability, making it the go-to choice for content optimization on Facebook as well as other social networks.

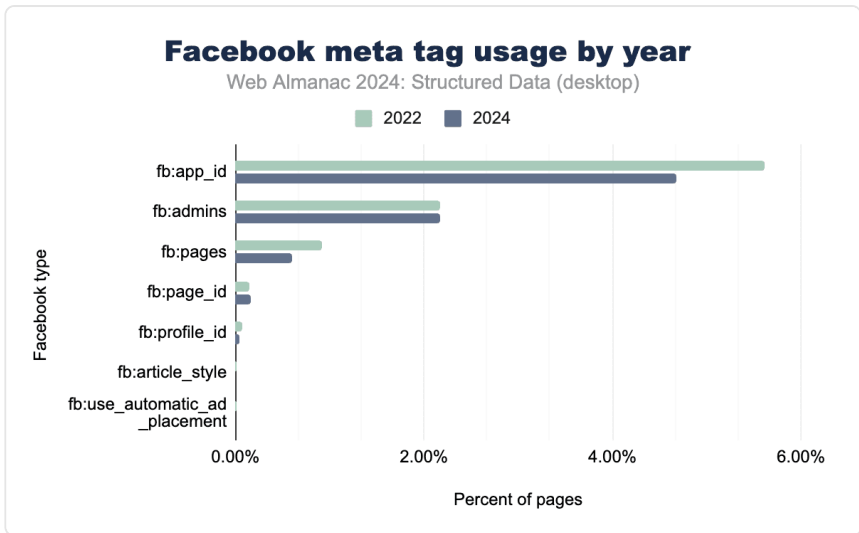


Figure 3.21. Facebook meta tag usage by year (desktop).

Despite the decreasing adoption of Facebook-specific tags, Facebook itself remains a key player in the social media landscape, with Open Graph handling most of its metadata needs. This trend reflects the consolidation of social sharing standards, where platform-agnostic tags provide greater reach and functionality.

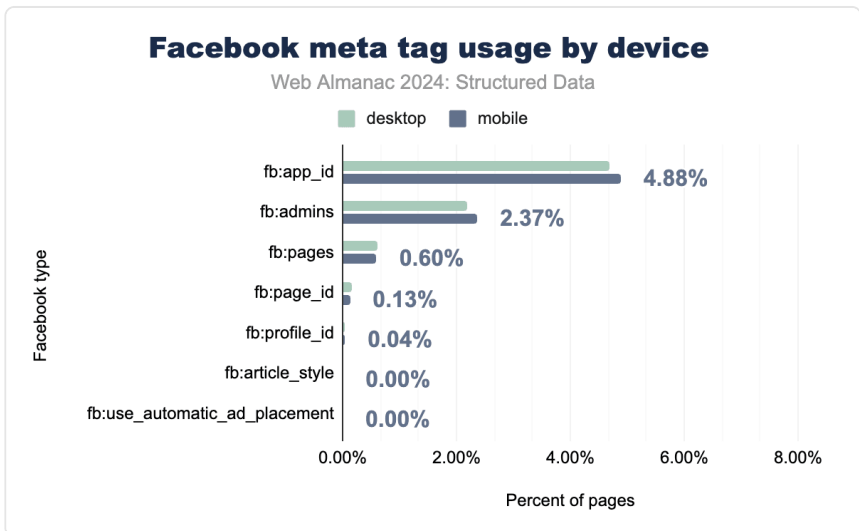


Figure 3.22. Facebook meta tag usage by device.

Microformats and Microformats2

Microformats continue to show limited adoption, primarily in niche use cases where simple, semantic data is required. The `adr` tag, used for address-related data, remains the most widely adopted Microformats type, appearing on approximately 0.4% of pages across both mobile and desktop platforms. Other tags, such as `geo` and `hReview`, have minimal usage, as more sophisticated formats like JSON-LD and Open Graph have become more prevalent.

Microformats2, while still relatively niche, has seen slightly higher adoption than its predecessor. Tags like `h-entry` and `h-card`, which are used for blogging and personal identity data, now appear on 0.22% of mobile pages and 0.15% of desktop pages. These tags continue to serve specific needs, particularly for address data and simple content structures.

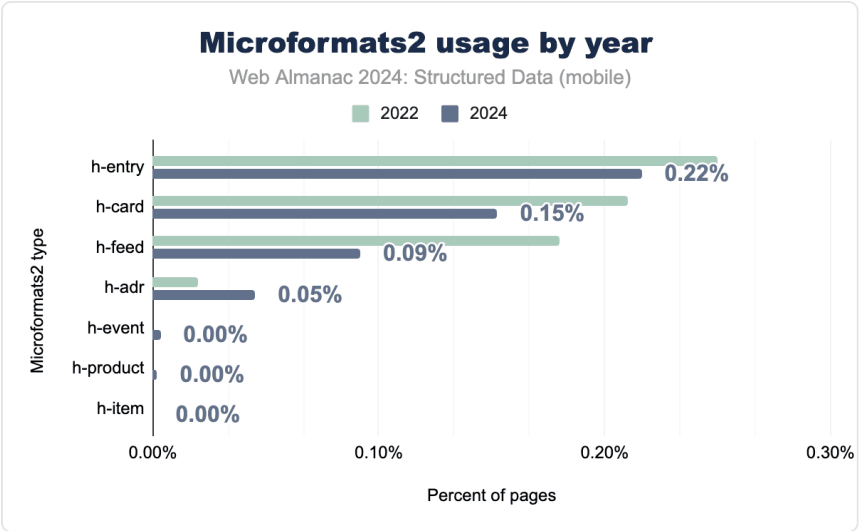


Figure 3.23. Year-on-year comparison of Microformats2 usage on mobile pages in 2022 and 2024.

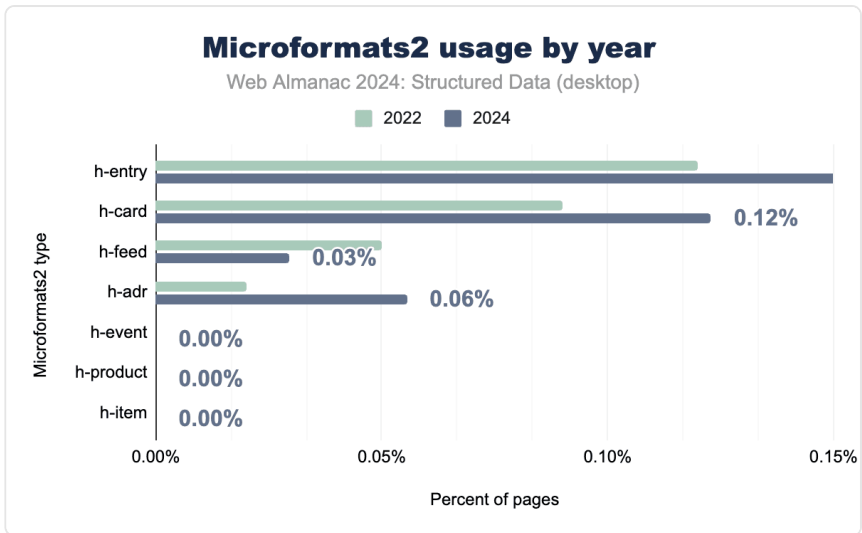


Figure 3.24. Year-on-year comparison of Microformats2 usage on desktop pages in 2022 and 2024.

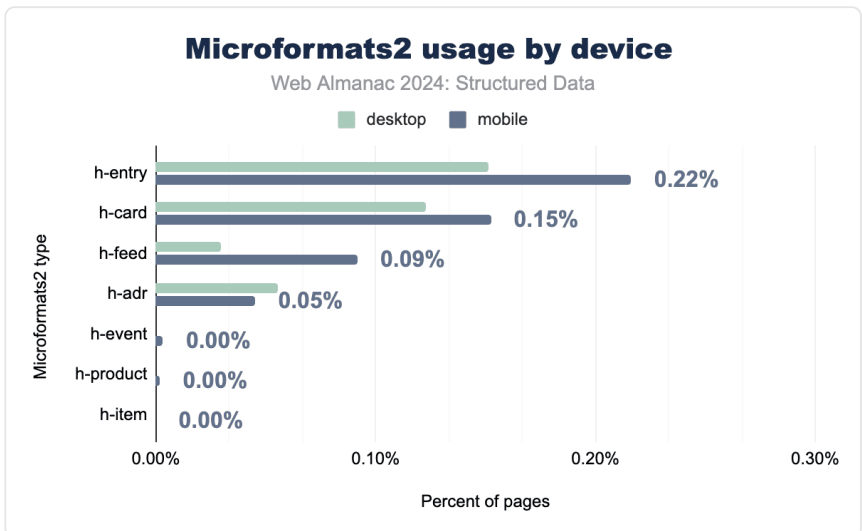


Figure 3.25. Microformats2 usage by device in 2024, comparing desktop and mobile implementations.

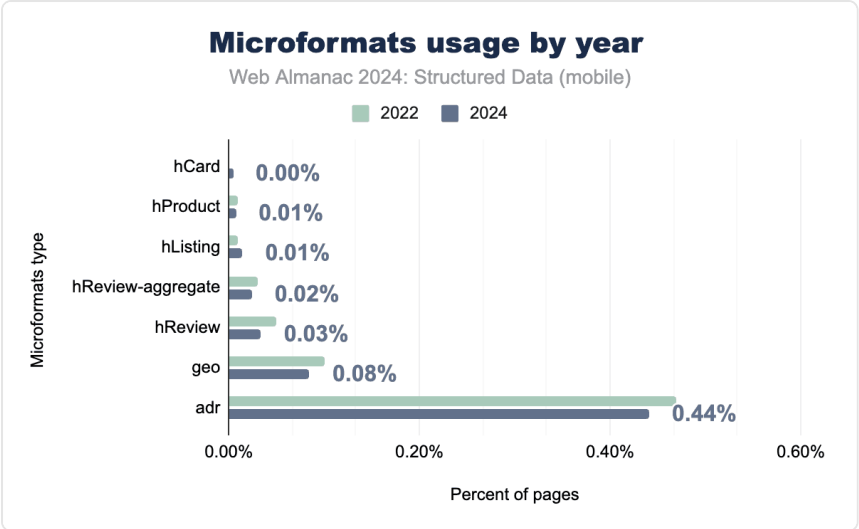


Figure 3.26. A year-on-year comparison of Microformats usage on mobile pages in 2022 and 2024.

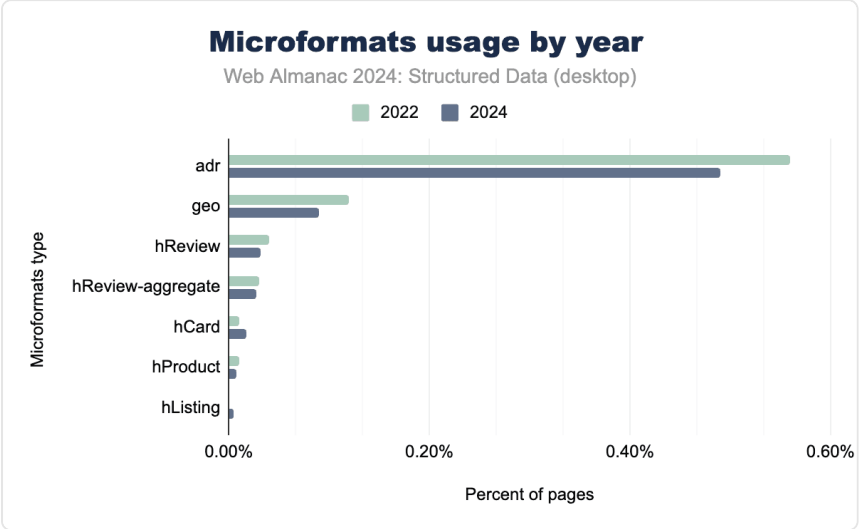


Figure 3.27. Microformats usage by year (desktop).

Cross-device implementation remains relatively consistent, though with some variation between mobile and desktop. The data shows a general decline in traditional Microformats usage from 2022 to 2024, particularly in review-related properties like `hReview` and `hReview-aggregate`. This decline reflects the industry’s shift toward more modern structured data formats like JSON-LD and RDFa, which offer broader functionality and better

integration with current web standards.

Despite this decline, Microformats and Microformats2 remain useful in specific contexts where lightweight, human-readable semantic data is needed. However, their overall presence continues to be eclipsed by more versatile formats like JSON-LD, which dominate the structured data landscape.

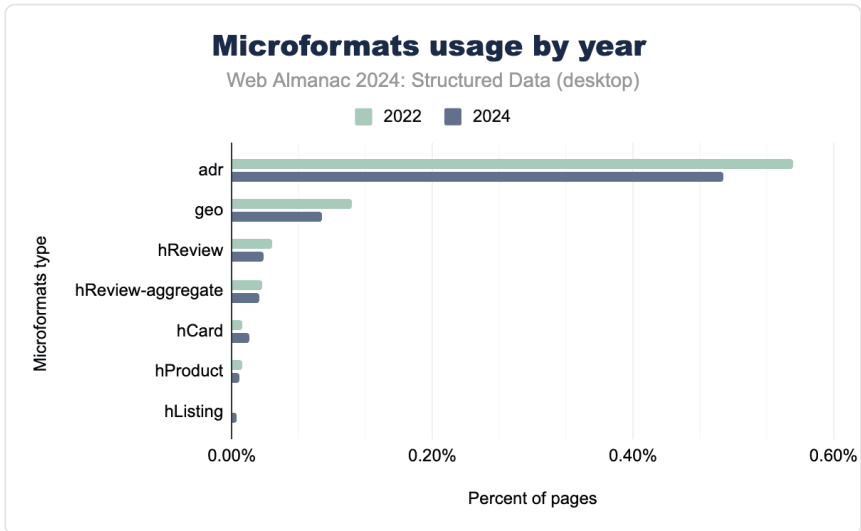


Figure 3.28. Microformats usage by year on desktop pages.

Microdata

Microdata continues to be widely used for structural elements and navigation data, particularly within legacy platforms and sites where simpler, static page structures are required. The most frequently implemented types include `schema.org/webpage` (appearing on 8.34% of mobile pages) and `schema.org/sitenavigationelement` (used on 6.42% of mobile pages), indicating the format's enduring relevance for webpage structure and site navigation.

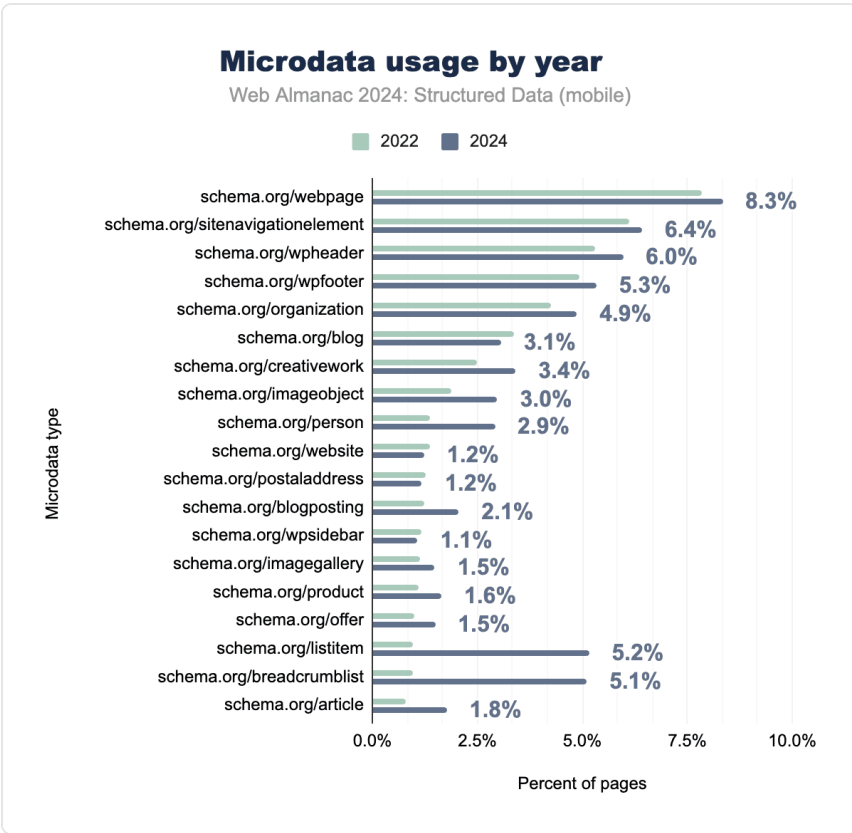


Figure 3.29. A year-on-year comparison of Microdata usage on mobile pages.

Navigation-related types like `listitem` and `breadcrumblist` have also seen steady growth, reflecting the need for more organized and structured navigation data, particularly on mobile devices. However, content-specific types such as `schema.org/article` and `schema.org/product` remain less common, with adoption rates of 1.77% and 1.50% respectively, as developers increasingly turn to JSON-LD for more flexible and scalable implementations.

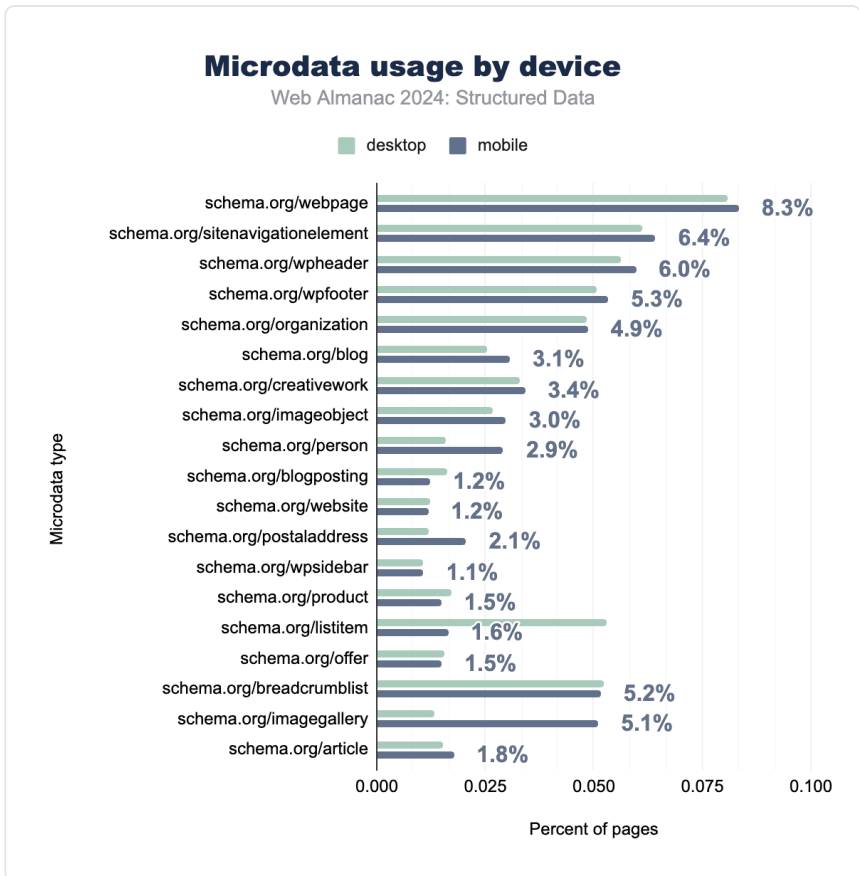


Figure 3.30. A comparison of Microdata usage by device.

While Microdata remains a significant format for fundamental webpage structure, its use in dynamic content and ecommerce applications has been gradually overtaken by more modern formats like JSON-LD, which offer broader support for content enrichment and structured data scaling across large websites.

JSON-LD

JSON-LD types continue to be widely implemented across websites, with varied types of data used depending on the purpose of the site. The `WebSite` schema leads adoption, appearing on 12.73% of mobile pages, followed by `Organization` and `LocalBusiness` types at 7.16% and 3.97%, respectively. These types are crucial for establishing entity identity and providing

contextual information to search engines.

The diversity in implementation patterns reflects how different industries and website types prioritize specific structured data. For instance:

- Ecommerce sites frequently implement `Product`, `Offer`, and `Review` schemas.
- Local businesses prioritize `LocalBusiness`, `GeoCoordinates`, and `OpeningHoursSpecification` to enhance local search visibility.
- Content publishers often utilize `Article` and `BlogPosting` schemas to structure written content effectively.

`BreadcrumbList` implementation has seen notable growth, appearing on 5.66% of pages, suggesting an increased focus on structured navigation data. The `WebPage` schema shows steady adoption at 1.49%, while the `Product` schema appears on 0.77% of pages. Content-specific types like `BlogPosting` (1.40%) and `Article` (0.18%) maintain consistent presence, though at lower levels.

Specialized business types such as `Restaurant` (0.19%), `AutoDealer` (1.09%), and `Store` (0.17%) demonstrate the growing adoption of industry-specific markup, corresponding to Google's increased support for these schemas. Supporting content types including `VideoObject`, `FAQPage`, and `Event` each appear on approximately 0.34% of pages, indicating steady but modest implementation of specialized content markup.

`ItemList` schema shows healthy adoption at 2.44%, suggesting increased use of structured listing data. The overall distribution of JSON-LD types reflects a maturing ecosystem where fundamental entity types dominate, while specialized schemas serve specific business and content needs.

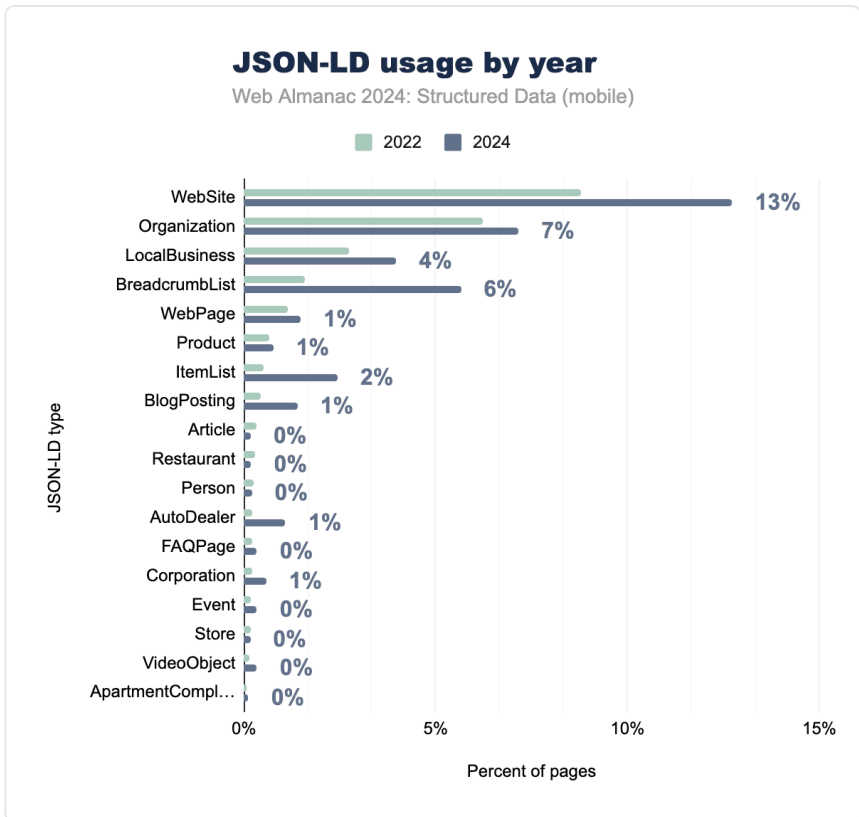


Figure 3.31. A year-on-year comparison of JSON-LD usage on mobile pages.

The consistency in implementation across devices indicates a mature approach to structured data deployment, where developers are ensuring uniform markup regardless of the target platform. This alignment between mobile and desktop implementations suggests that organizations are following best practices for responsive design while maintaining consistent structured data across all user experiences.

Despite Google's deprecation of `FAQ` and `HowTo` rich results in August 2023 (source⁵³), we see limited impact on their adoption rates. `HowTo` schema adoption has historically been low due to its complexity, with implementation rates below 1% for both desktop and mobile. `FAQPage`, on the other hand, has not only maintained its adoption but even shows a slight increase on desktop, rising from 0.2% in 2022 to 0.6% in 2024. This trend suggests that webmasters may still find value in implementing `FAQPage` for additional search engine visibility besides rich results.

53. <https://developers.google.com/search/blog/2023/08/howto-faq-changes>

These observations highlight the *resilience of certain structured data types* despite changes in Google’s support. It also points to the importance of monitoring how structured data evolves across various platforms, as its utility often extends beyond immediate search result enhancements.

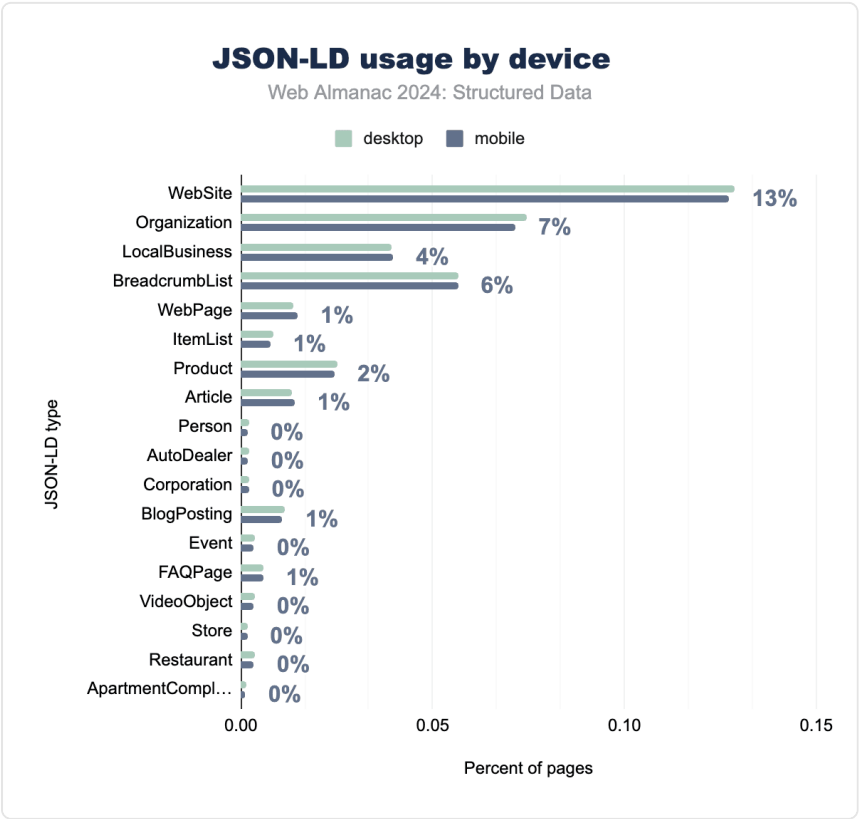


Figure 3.32. A comparison of JSON-LD usage by device.

JSON-LD relationships

When evaluating JSON-LD relationships in structured data implementations, several key patterns emerge in how entities are connected in a graph. These relationship patterns reflect how structured data is used to create comprehensive, interconnected entity descriptions that help search engines better understand content context and relationships. The most successful implementations leverage these connections to provide rich, detailed information while maintaining logical content relationships.

Let's review the most critical patterns from the JSON-LD relationship analysis:

1. **Local business ecosystem.** The most sophisticated structured data implementations are occurring in the local business sector, where we see rich interconnections between `LocalBusiness`, `OpeningHoursSpecification`, `PostalAddress`, and `GeoCoordinates`. This suggests businesses are creating comprehensive digital identities that go beyond basic location information to include detailed operational data. This aligns with Google's increasing focus on local search and the growing importance of location-based services.
2. **Content organization.** Maturity There's a clear pattern of publishers implementing more sophisticated content structures. The relationships between `Article`, `BlogPosting`, and `WebPage` entities consistently link to `ImageObject`, author attributes, and publishing details. This isn't just about marking up individual pieces of content – it's about creating proper content graphs that establish clear relationships between content, creators, and organizational entities.
3. **Ecommerce integration.** The product-related relationships show an interesting evolution. Beyond basic product markup, we're seeing more connections to `ReviewRating`, `AggregateOffer`, and `PriceSpecification` entities. This suggests ecommerce sites are building more comprehensive product knowledge graphs that can support advanced features like price tracking and inventory status.

Most notably, these patterns indicate that structured data implementation is moving beyond simple SEO markup toward creating true knowledge graphs that can support AI-powered search experiences and rich data integrations.

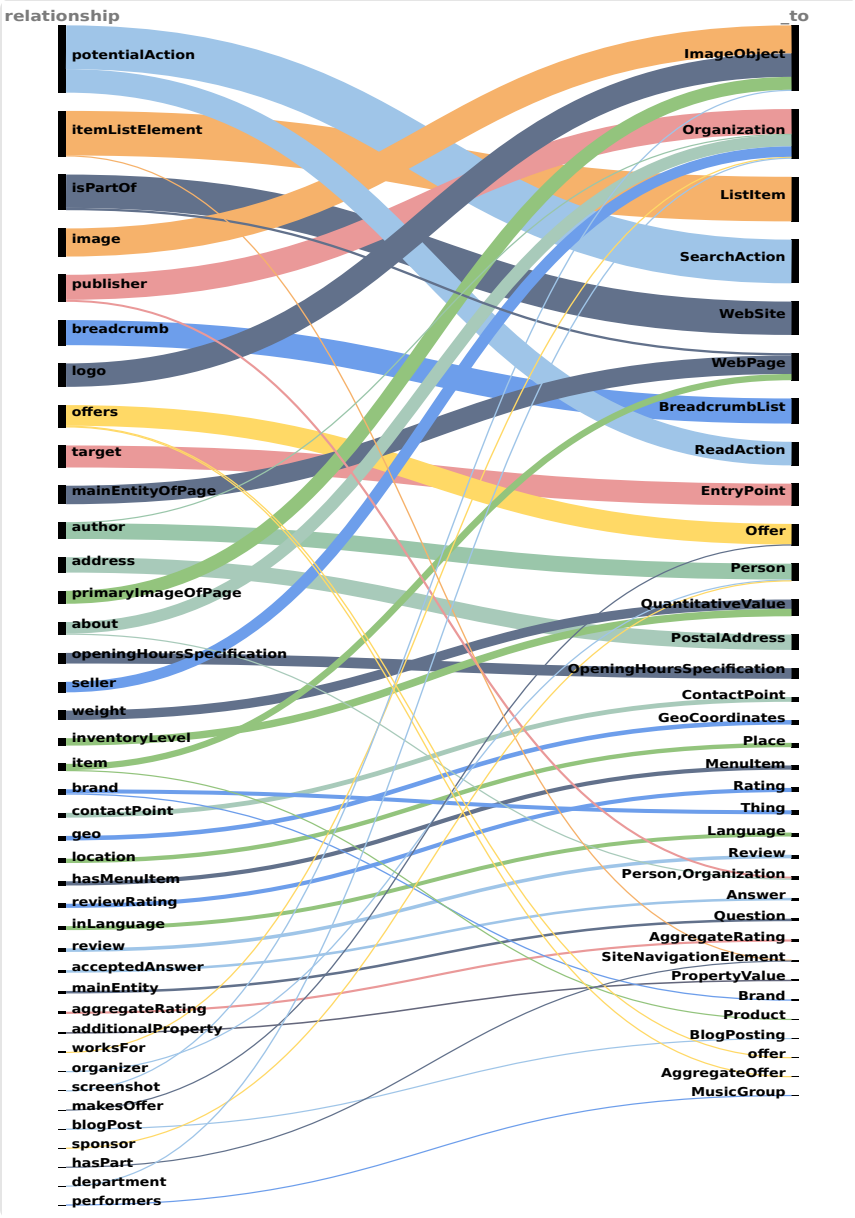


Figure 3.33. Sankey diagram showing relationships between structured data types and their connections.

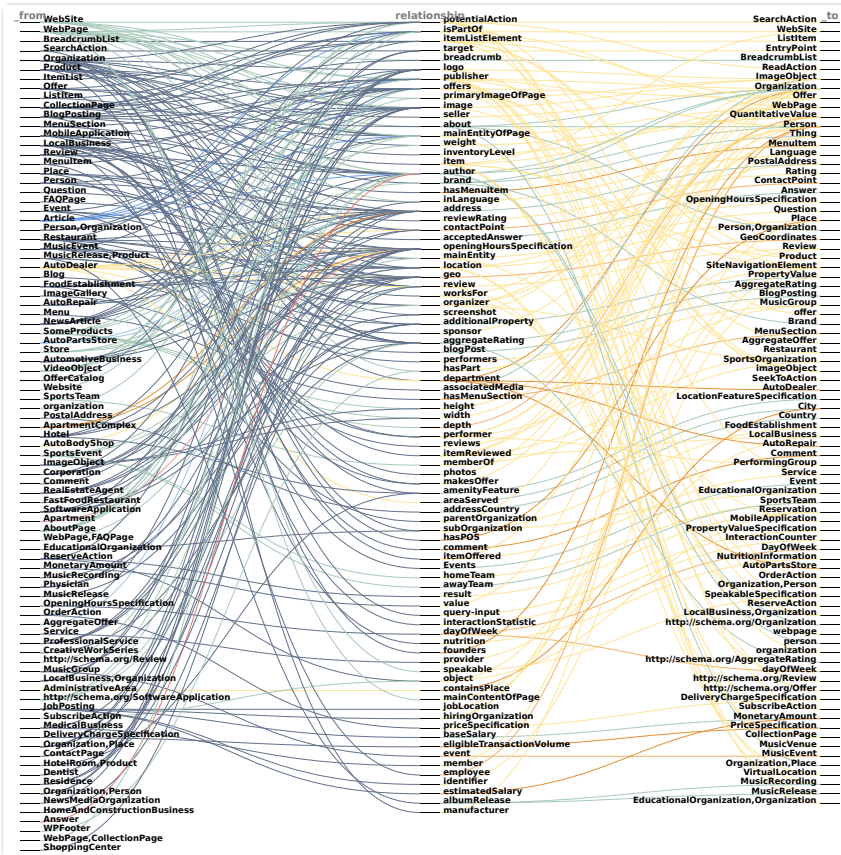


Figure 3.34. Detailed Sankey diagram of dataset relationships in JSON-LD.

As seen also in the previous chart the most frequent JSON-LD property relationships reveal several critical implementation patterns across websites. `PotentialAction` emerges as a dominant property, showing strong connections to `SearchAction` and `WebSite`, indicating widespread implementation of site search functionality (we expect this to decrease as Google is removing support for this feature snippet). Image-related properties form another major cluster, with `ImageObject` frequently connected to `Organization` and `WebPage` entities, demonstrating the importance of visual content attribution. The `publisher` and `logo` properties frequently link to `Organization` entities, establishing clear brand identity.

Navigation structures show clear patterns through `BreadcrumbList` and `ItemListElement` properties, typically connecting to `WebPage` entities. Content relationships are evidenced by `mainEntityOfPage` connections, while business-specific information flows through address, `openingHoursSpecification`, and geo properties.

Particularly noteworthy is the consistent implementation of contact and location information, with `PostalAddress`, `ContactPoint`, and `GeoCoordinates` forming a well-defined cluster. This suggests businesses are prioritizing local presence markup. The presence of review-related properties (`reviewRating`, `rating`) connected to various entities indicates strong focus on reputation management through structured data.

`sameAs`

The `sameAs` property plays a crucial role in entity disambiguation and knowledge graph development, extending far beyond simple social media profile linking. While our data shows strong implementation for major platforms (Facebook at 4.53%, Instagram at 3.67%), the true strategic value lies in how `sameAs` helps search engines understand and validate entity relationships.

When properly implemented, `sameAs` serves as a powerful tool for entity disambiguation, particularly for organizations and persons. By linking to authoritative sources like Wikidata (0.17%) and Wikipedia (0.13%), brands can establish unambiguous entity identification. This creates what we might call a “*entity fingerprint*” that helps search engines confidently associate various online presences with the correct entity.

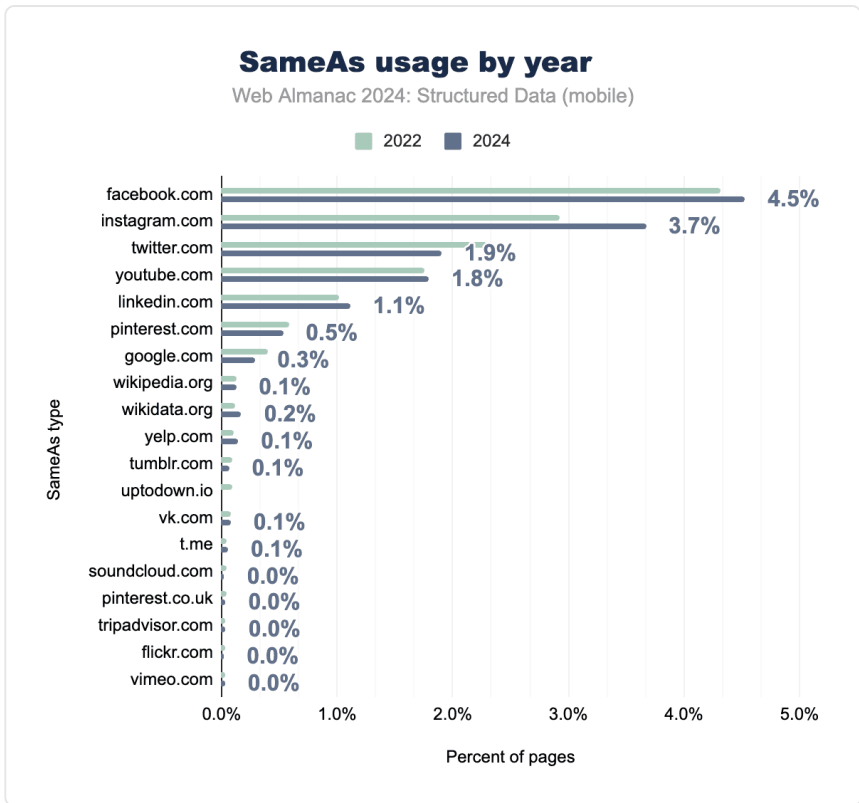


Figure 3.35. Year-on-year comparison of `sameAs` usage on mobile pages in 2022 and 2024.

For personal entities (executives, authors, experts), `sameAs` similarly helps establish authority and credibility by connecting professional profiles (LinkedIn at 1.11%) with other authentic entity markers. This becomes particularly valuable for E-E-A-T signals and knowledge panel generation.

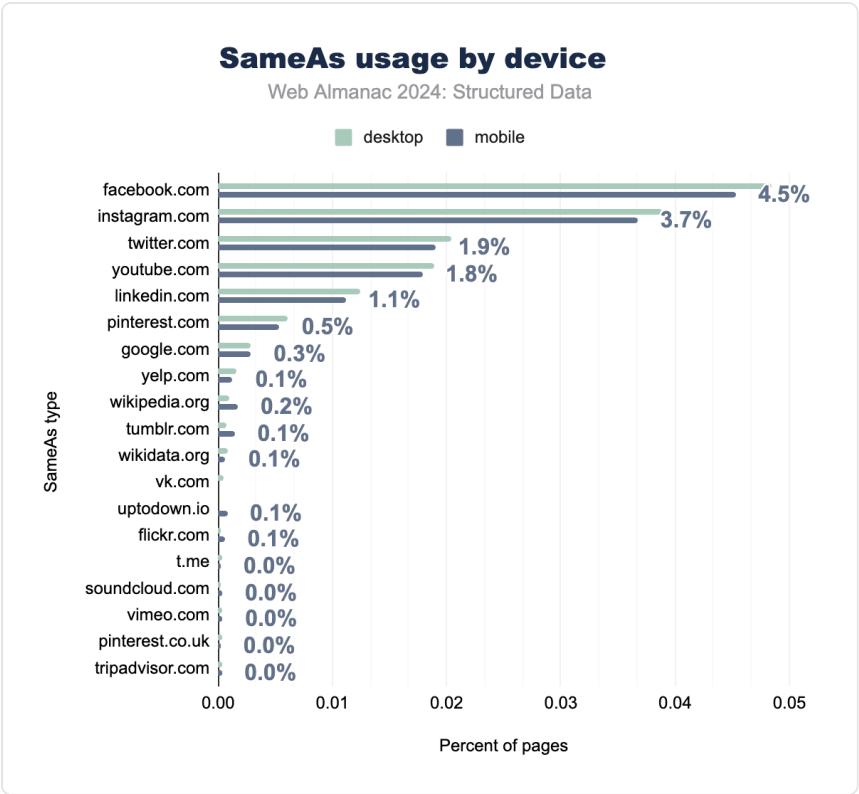


Figure 3.36. `sameAs` usage by device on mobile and desktop for various platforms.

This parity between mobile and desktop implementations represents a significant advancement in structured data deployment. It suggests that organizations are increasingly using consistent templating systems or automated solutions to manage their structured data, rather than maintaining separate implementations for different devices.

JSON-LD context

Schema.org remains the dominant force in JSON-LD context implementation with over 20 million instances, far exceeding all other contexts. This dominance (20,960,693 implementations versus the next highest at 11,973) reflects its position as the industry standard for structured data markup.

Among secondary implementations, contao.org leads with 11,973 instances, primarily within its CMS ecosystem, followed by googleapis.com (3,743) and baidu.com (1,409). Educational

institutions show consistent adoption patterns around 25-50 implementations each, while regional variations appear through implementations like Schema.org.cn and Schema.gov.sg, indicating global adoption of structured data standards.

The vast gap between Schema.org and other contexts underscores its critical role in structured data standardization and reflects strong alignment with search engine requirements.

Emerging trends and future outlook

The structured data landscape is rapidly evolving, marked by Google's introduction of specialized schemas for vehicles, courses, and 3D product models, alongside increased support for Digital Product Passports through GS1 Digital Link. The growing adoption of JSON-LD (now at 41% of pages) and sophisticated entity relationships through `sameAs` properties indicates a maturing ecosystem focused on comprehensive knowledge graph development.

The data shows a clear shift toward more specialized implementation patterns, particularly in ecommerce and local business contexts. For instance, structured data types like `Product`, `Offer`, and `Review` have become more prevalent in ecommerce, while `LocalBusiness` and `GeoCoordinates` are increasingly used to improve local search visibility.

This shift can be partially attributed to Google's policy changes, which encouraged webmasters to focus on more domain-specific schemas. Entity disambiguation has also become increasingly critical, with organizations leveraging structured data like `sameAs` and `Organization` to establish clear digital identities across platforms and knowledge bases.

Looking ahead: the future of structured data

As we analyze current trends, we also cast our gaze forward to emerging developments:

- **AI and structured data symbiosis**

The growing interdependence between AI systems and structured data is becoming crucial for delivering grounded, hallucination-free content generation and enhancing conversational search experiences. As AI relies increasingly on structured data for accurate and context-rich information, this symbiosis is redefining how AI-powered tools interact with content across the web.

- **Data Commons and knowledge graph integration**

The expansion of open data initiatives, such as Google's Data Commons, which leverages Schema.org for structuring and linking public datasets, is further fueling the evolution of knowledge graph-based systems. These initiatives provide a rich, unified foundation for AI-driven data enrichment and exploration, creating new possibilities for scalable and reliable data integration across platforms.

- **SEOntology and specialized vocabularies**

In parallel, the development of SEOntology⁵⁴ and other specialized vocabularies is addressing the need for SEO-specific structured data that can improve content discoverability and search engine optimization. By creating vocabularies tailored to the unique requirements of SEO, we can further enhance the alignment between structured data and AI, driving more targeted and efficient search experiences.

- **Regulatory impacts**

Finally, regulations such as the EU's Digital Product Passport are poised to reshape future structured data standards. These initiatives will likely influence how structured data is applied, especially in domains like ecommerce and product traceability, encouraging more structured and transparent data practices.

By examining these aspects, we aim to provide a comprehensive overview of the state of structured data in 2024, its recent evolution, and its future trajectory. Whether you're a seasoned SEO professional, a web developer, an ecommerce strategist, or simply interested in the evolution of the web, this chapter offers valuable insights into how structured data is reshaping our digital world and paving the way for a more connected, transparent, and intelligent online experience.

Conclusion

The analysis of structured data in 2024 highlights a clear shift from its SEO roots toward a broader, more strategic role in AI and semantic metadata. The dominance of RDFa and Open Graph on over 60% of pages, combined with JSON-LD's growth (now on 41% of pages,

54. <https://www.searchenginejournal.com/introducing-seontology-the-future-of-seo-in-the-age-of-ai/524773/>

particularly in ecommerce), points to a maturing technology. But the true impact lies in how structured data is transforming AI discovery and enhancing machine understanding.

This year, we've seen significant changes in how search engines handle structured data. While Google has deprecated certain rich results, such as `FAQs`, `HowTo`s, and `SiteLinks`, they've simultaneously introduced new types for vehicles, courses, 3D product models, loyalty cards, and certifications, expanding the scope of structured data. Even more importantly, structured data is now essential for AI systems, supporting tasks from fact-checking to improved search capabilities and training large language models (LLMs).

The advent of Digital Product Passports and increased adoption of GS1 standards underlines the growing importance of structured data in commerce and regulatory compliance. As AI-driven search becomes the norm, businesses are realizing that structured data is no longer just about search visibility—it's key to ensuring content is machine-readable and future-proof.

For businesses developing their structured data strategy, the way forward is clear: implement it comprehensively, maintain it rigorously, and adapt continuously. New projects should focus on JSON-LD, while legacy formats should be preserved where appropriate. Systems must be built to scale and evolve alongside emerging technologies and standards.

In conclusion, the future of the web is structured, semantic, and increasingly intelligent. Organizations that invest on structured data today won't just improve their search visibility – they are building the foundation for success in AI Discovery.

Author



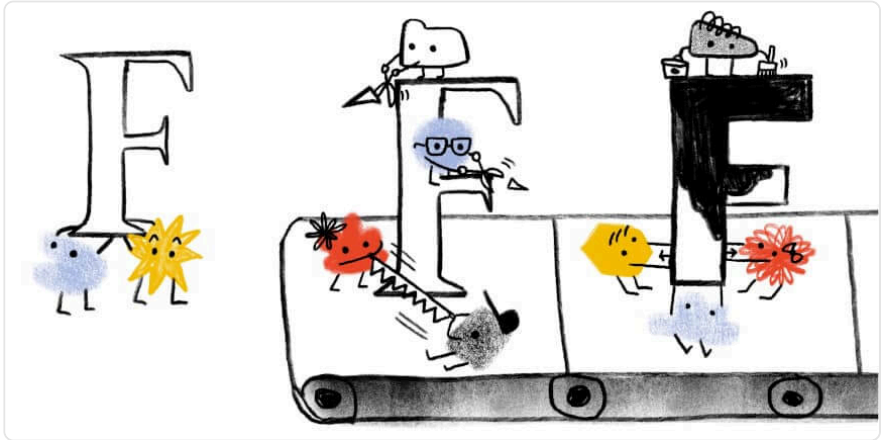
Andrea Volpini

✉ @cyberandy 📱 cyberandy 🌐 <https://wordlift.io/blog/en/entity/andrea-volpini>

Andrea Volpini, CEO of WordLift, is pioneering advancements at the intersection of SEO and neuro-symbolic artificial intelligence, driving innovation in semantic technologies and AI-driven content discovery.

Part I Chapter 4

Fonts



Written by Bram Stein and Charles Berret

Reviewed by Dominik Röttsches, Chris Lilley, Ivan Ukhov, José Solé, Liam Quin, Mandy Michael, and Raph Levien

Analyzed by Ivan Ukhov

Edited by Charles Berret

Introduction

Typography plays a major role in user experience on the web, from legibility and readability to accessibility and emotional impact. And whereas web developers used to be limited to a small number of web-safe fonts, we now have vast libraries offering both expressive range and increasingly comprehensive script support for the world's many writing systems.

This year's HTTP Almanac web crawl found that web font usage continues to grow, though at a slower pace than what was observed in previous years. Web fonts are now used on around 87% of all websites, whether alone or in combination with self-hosted fonts. At the same time, an increasing number of websites are now self-hosting as their exclusive means of delivering fonts. This trend coincides with a slight decline in websites using a combination of self-hosting and a font service. Still, the Google Fonts service continues to deliver the majority of fonts seen on the web. Around 57% of websites observed on the HTTP Archive's desktop crawl and 48% on its mobile crawl use Google Fonts, whether alone or alongside another hosting option.

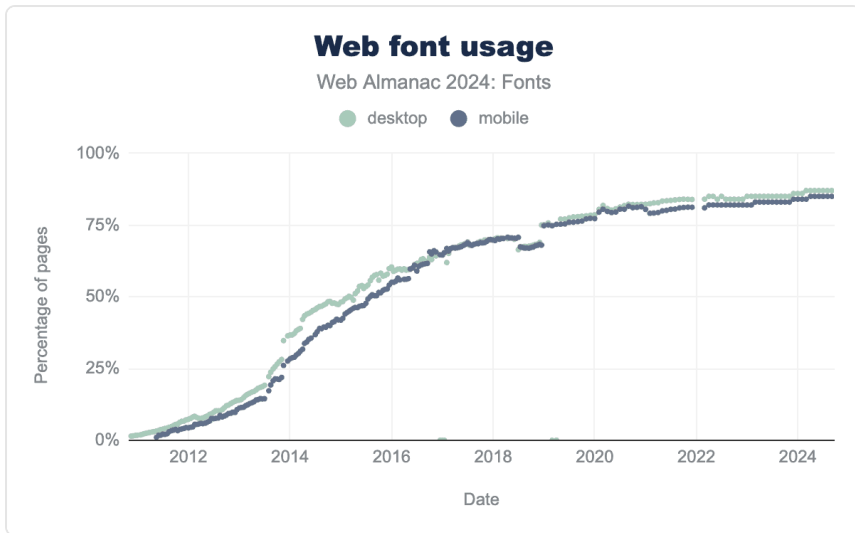


Figure 4.1. Web font usage.

OpenType feature support continues to rise, following a trend seen in the past editions of the HTTP Almanac. This means an increasing share of the fonts currently used on the web have been designed with at least one OpenType feature, such as ligatures, kerning, or fractions. While this trend reflects an increasing tendency for designers to include OpenType features in the fonts themselves, this year's data shows that more web developers are also taking advantage of these OpenType features in CSS.

Alongside broader implementation and use of longstanding OpenType features, there has also been a significant uptake of newer OpenType capabilities, such as color and variable fonts. In the case of color fonts, adoption is still at fairly low levels (just a few thousand websites across the entire internet) but rising at a considerable rate each year. Meanwhile, variable font usage has picked up even more dramatically, and a large factor driving adoption appears to be the popularity of variable fonts for several writing systems used by large populations of web users. Chinese, Japanese, and Korean (CJK) fonts, especially in the Noto superfamily, accounted for an especially large share of the variable fonts currently in use: about 42% on desktop and 34% on mobile of all the variable fonts on the web come from Noto's CJK families.

More broadly, there has been a general rise in the use and support of various global scripts and languages on the web, reducing the once-overwhelming presence of Latin fonts. This displays the fruit of recent efforts to support the design and development of quality typefaces for languages that were long neglected in type catalogs, historically focused almost exclusively on Western characters.

The remainder of this chapter explores these subjects and more in detail, using data from the HTTP Archive web crawl as a means of depicting the current state of fonts on the web. The chapter is divided into sections touching upon various subjects related to the design and use of web fonts. We start with practical decisions surrounding how fonts are delivered to users, including the hosting, format, and size of font files. We then examine the most popular font families, the foundries that designed these fonts, and their level of support for different writing systems. We close by discussing emerging technologies, such as color and variable fonts, as well as technical choices concerning how fonts are built and used on the web.

Before proceeding into the chapter, however, we would like to make a few technical notes. Our overall approach to analyzing this year's font data is heavily focused on trends. To examine these trends, we compare this year's data to previous editions of the HTTP Almanac. Because there was no Almanac published in 2023, many of our comparisons point to data from the 2022 edition⁵⁵. In several cases, we also include 2023 data when it's available and relevant.

When we present percentages throughout the chapter, it is important to pay close attention to *what* is actually counted in each specific case and *how* the corresponding count is normalized to arrive at the percentage in question. Without keeping this in mind, it would be easy to erroneously compare apples with oranges when considering any two percentages. We use three counting methods:

- **Web Pages:** This method follows the Web Almanac's methodology and counts the number of root pages.
- **Font Requests:** This method counts font requests on root pages, then divides by the total number of font requests in the crawl. If a certain font happens to be requested several times by the browser when loading a page, it will be counted equally many times.
- **Font Files:** This method counts the number of distinct font URLs, then divides by the total number of font URLs in the crawl. If the same URL happens to be used on multiple websites, it will be counted only once. This mode of counting aims to observe the total set of font files accessible online.

Hosting and services

There are basically just two methods for delivering fonts to website visitors. One way is to provide web fonts through a service, whether a free one like Google Fonts⁵⁶ or a paid one like

55. <https://almanac.httparchive.org/en/2022/fonts>

56. <https://fonts.google.com/>

Adobe Fonts⁵⁷. The other way is to self-host the font files from the website's own domain, keeping full control over the files with no external dependencies.

To understand the font hosting choices made by web developers, we follow the method from past Almanacs and look at several overlapping categories. The “Self-hosted (non-exclusive)” category points to all websites using self-hosted fonts, even if they also use a hosting service. The “Self-hosted (exclusive)” category counts websites that only use self-hosted fonts. Likewise, the “Services (non-exclusive)” category points to all websites that use a hosting service, even if they also use a self-hosted font. Sites in the “Services (exclusive)” category use only a hosting service. We have also added a new category this year, “Self-hosted plus service,” referring to sites that use both self-hosted fonts and a service (e.g. the non-exclusive self-hosted sites minus exclusive self-hosted sites).

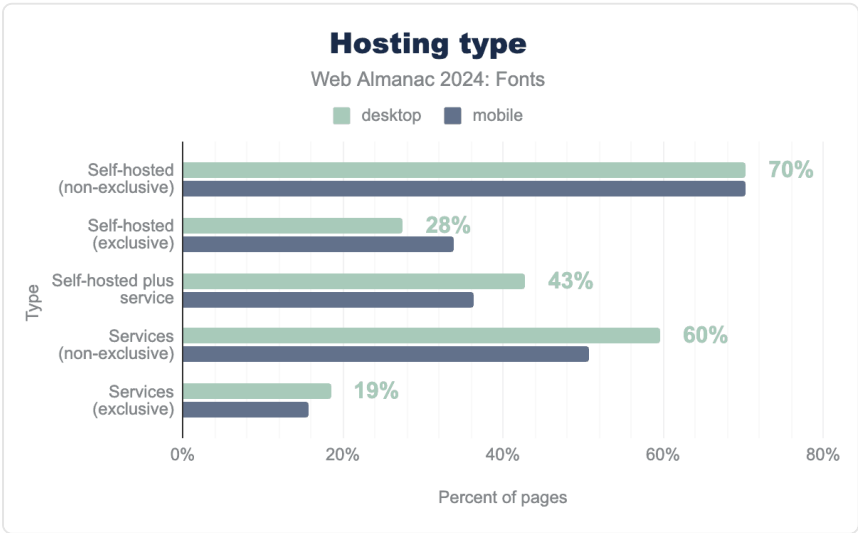


Figure 4.2. Hosting Type.

This year, there has been a significant increase in exclusive self-hosting (desktop: from 22% in 2022 to 28%; mobile: from 28% in 2022 to 34%). At the same time, there has been a coinciding decrease in non-exclusive use of services (desktop: from 63% in 2022 to 60%; mobile: from 55% in 2022 to 51%). These interconnected trends were first spotted in 2022, when more people began to self-host their fonts because it often yields better performance and privacy (since the introduction of cache partitioning⁵⁸, using a shared font CDN is no longer beneficial). This suggests a sizable number of websites that once used both a web service and their own self-hosted fonts are now using self-hosted fonts alone.

57. <https://fonts.adobe.com/>

58. <https://developer.chrome.com/blog/http-cache-partitioning>

Meanwhile, the number of websites exclusively using a web font service has actually remained fairly consistent over the last two years, amounting to roughly 19% of desktop and 16% of mobile websites. A full 70% of websites now use some form of self-hosted font, whether alone or with a service. This means the overall share of websites with self-hosted fonts has risen about two percentage points since 2022.

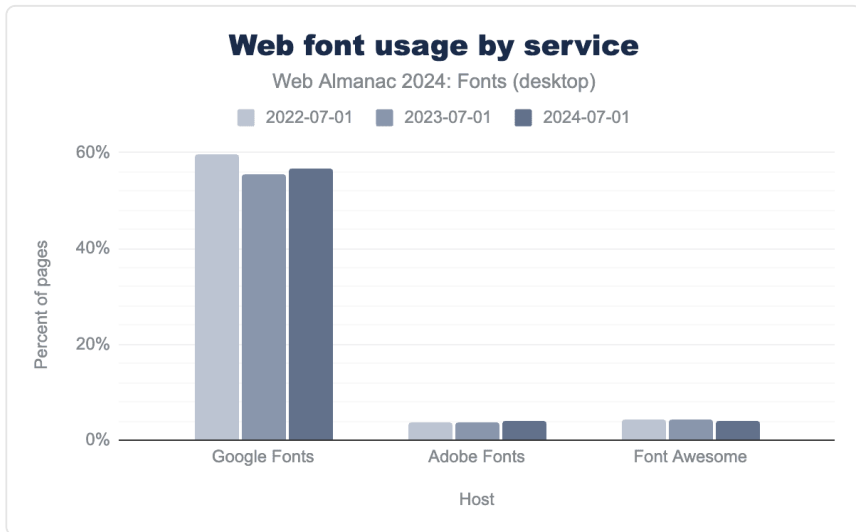


Figure 4.3. Web font usage by service.

The service market for web fonts is highly consolidated, and increasingly so. The only major font services are now Google Fonts, Adobe Fonts, and Font Awesome. Fonts.com and cloud.typography dropped to very low levels of usage as of two years ago. Even Google Fonts, which is something of a giant in font hosting, registered a few points of declining usage in the 2023 crawl—dropping from 60% to 57% of total font hosting—albeit with a recovery showing now-stable usage levels. Meanwhile, fonts served by Adobe and Font Awesome were each found on about 4% of webpages this year.

With their 4% share of the service market, Adobe Fonts presents the one case of a font service whose proportion of the web fonts market grew in this year's data. Adobe registered an increase of 11 percentage points over the last two years. The most likely explanation is that Adobe Fonts bundles many high quality and popular commercial typefaces with its Creative Cloud subscription. As Adobe web font usage is not charged by pageviews, it can be a cheap option for high traffic websites compared to buying a more expensive web license from a distributor or foundry.

Because a website can pull fonts from multiple sources, the popularity of different font hosting

options is not a zero-sum game and the most popular options are often found in combinations on a single website.

Services	desktop	mobile
Google Fonts, Self-hosted	39%	33%
Self-hosted	28%	34%
Google Fonts	13%	11%
Google Fonts, Font Awesome, Self-hosted	2%	1%
Google Fonts, Font Awesome	1%	1%

Figure 4.4. The top 5 most popular web font hosting combinations.

About 39% of desktop and mobile websites crawled by the HTTP Archive this year used both Google and self-hosted fonts, whereas 28% used only self-hosted fonts and 13% used only Google. Altogether, these two sources together provided the vast majority of fonts seen on the web: 79% of websites crawled in 2024 used either self-hosted fonts, Google web fonts, or both. Still, there was a noticeable drop in the number of websites combining Google Fonts with self-hosting. Between 2022 and 2023, the combination of Google and self-hosted fonts dropped from 41% of websites to 38%. That number has begun to rebound slightly this year, as the figure started climbing again to reach 39%.

Overall, the trend is clear: more and more people prefer to self-host their web fonts. This is a great choice in many cases because self-hosting avoids external dependencies for something as critical to rendering as fonts are. Plus, when self-hosted fonts are well optimized, they give you the best performance (but more on that later).

NB: The numbers presented in this section are slightly different from those in the 2022 chapter. The 2022 chapter attempted to include base64 encoded fonts embedded in CSS files for some (somewhat) popular web font service. Fortunately, encoding base64 fonts in CSS is no longer a popular method of serving fonts. For this reason the 2024 chapter switched to counting fonts served as separate files only. The spreadsheet for this year includes the recalculated figures for 2022 and 2023 as well (and they are mentioned where appropriate in this section).

File formats

Which font formats are found most often on the web? WOFF2 is by far the most popular format for web fonts, being used on 81% of desktop and 78% of mobile websites. This marks an increase of three percentage points in WOFF2 usage since 2022. It is also an encouraging trend because WOFF2 offers smaller file sizes, and thus increased loading performance, among other

benefits. This format's predecessor, WOFF, is also found on 8% of desktop and 10% of mobile websites, though these numbers represent a two percentage point drop since 2022.

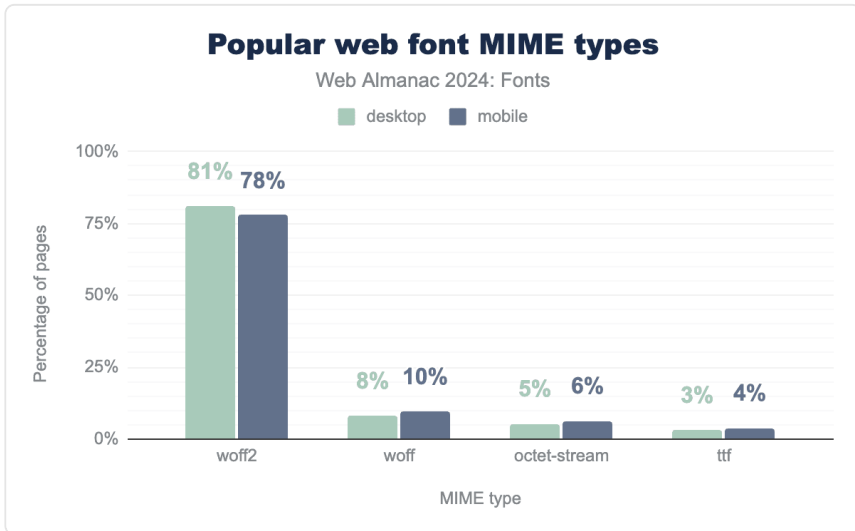


Figure 4.5. Popular web font MIME types.

Altogether, WOFF and WOFF2 make up the vast majority of web fonts at nearly 90% of the combined total for desktop and mobile websites. TrueType files also accounted for a small but noteworthy share (3%–4%) of the non-WOFF web fonts found in this year's data. It's also worth pointing out that 5%–6% of websites are serving fonts as `application/octet-stream`, an incorrect MIME type. Looking at the data, the main "self-hosted" hosts serving incorrect mime types for fonts are two incorrectly configured CDNs: [cdnjs](https://cdnjs.com/)⁵⁹ and [Wix](https://www.wix.com/)⁶⁰.

While these are useful insights into the global state of web font formats, the global data paints a slightly-too-positive picture of trends because the market is skewed so heavily toward web services like Google Fonts, Font Awesome, and Adobe Fonts. These services have a vested interest in reducing the amount of data they serve, and because of their large footprint on the web, the decisions made by these few major players will tend to skew the overall picture. To understand decisions made by web developers, it's much more interesting to exclude web services and look at the dataset for self-hosted fonts alone.

59. <https://cdnjs.com/>
60. <https://www.wix.com/>

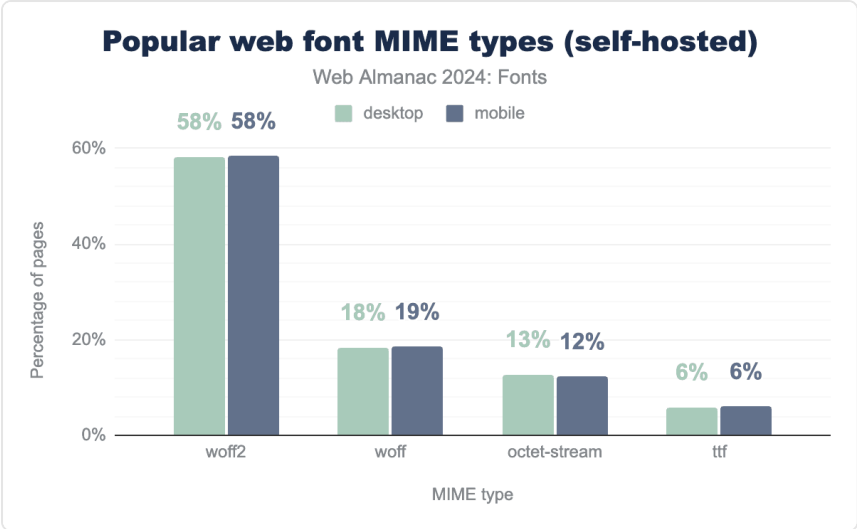


Figure 4.6. Popular web font MIME types (self-hosted).

Luckily, WOFF2 also takes the lead here, but surprisingly, the older WOFF format is still very popular on websites that self-host their fonts. In general, uncompressed font formats still make up a significant portion of self-hosted fonts. Developers who are still holding out have a lot to gain by switching to WOFF2, and making the switch should be viewed as low-hanging fruit. There are many online and command line tools to convert OTF or TTF files to WOFF2. It's also possible to decompress WOFF and recompress the files as WOFF2 (though one must be cautious that conversion is in compliance with a font's license).

File sizes

The average size of web fonts has risen for most websites on desktop and mobile since 2022. This general trend is especially striking in the 50th, 75th, and 90th percentiles, where the average size across desktop and mobile websites shows major increases. Delivering these larger fonts in compressed format offers a valuable means of keeping manageable load times.

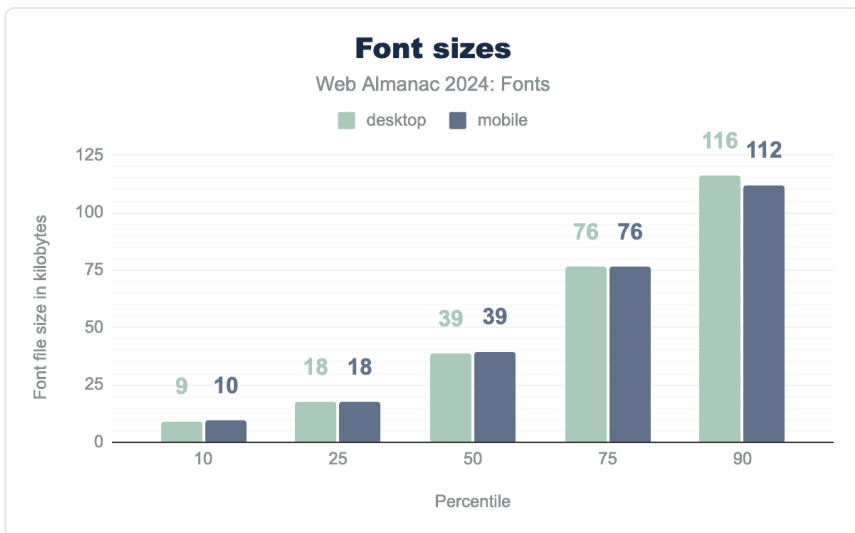


Figure 4.7. Font file sizes.

By observing font formats individually, we can compare the relative impact of compression by looking at WOFF2 file sizes and plain TTF file sizes side by side. The size of a font file depends a lot on its format. The more highly compressed formats like WOFF and WOFF2 should have smaller file sizes, on average, than non-compressed font formats like plain TrueType and OpenType files (which you shouldn't be using). The steady increase in WOFF2 usage is good news: with average font file size going up, WOFF2 can help manage the performance impact of these larger files.

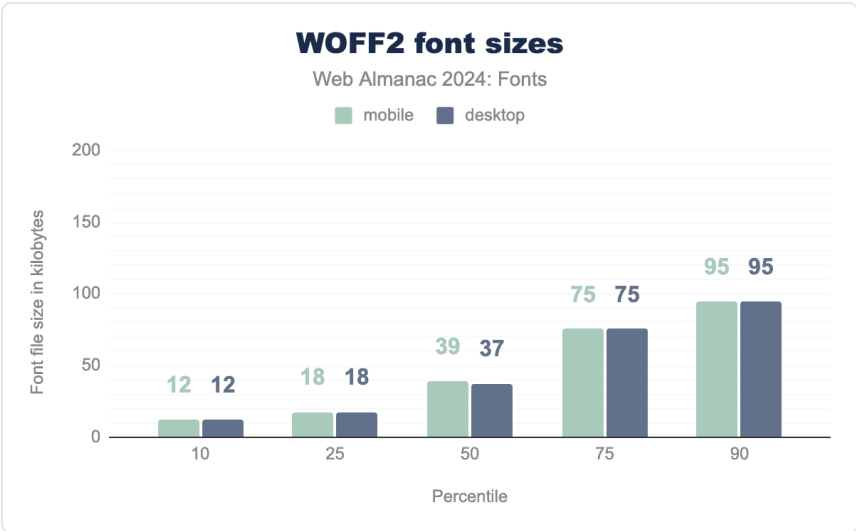


Figure 4.8. WOFF2 font file sizes.

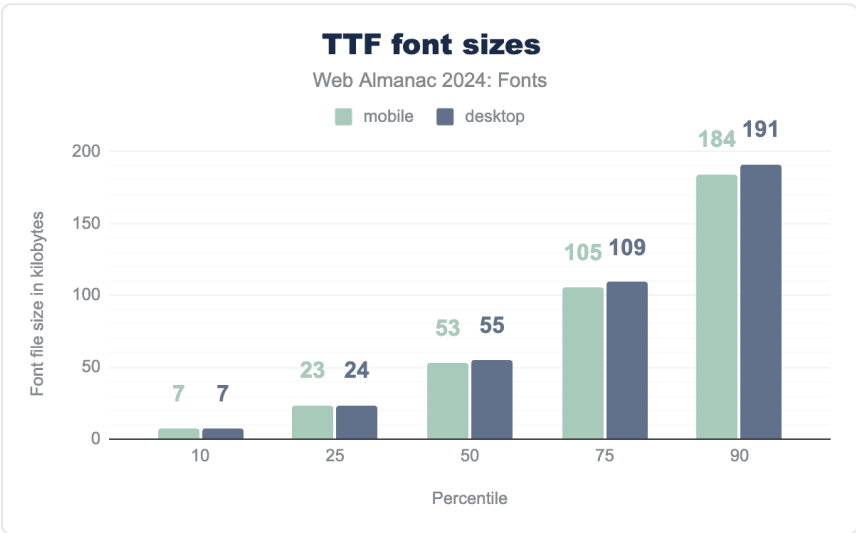


Figure 4.9. TTF font file sizes.

The difference in file size for WOFF2 compared to TTF is greatest in the higher percentiles, representing the largest files of each type used on the web this year. At the 90th percentile, the TTF files used this year were roughly twice as large as WOFF2 files. Moving toward lower percentiles, this difference converges and finally reverses. At the 10th percentile, representing the smallest bracket of fonts found for each format, WOFF2 files are nearly twice the size of

TTF files. This relationship in file sizes is most likely due to the overhead of the WOFF2 compression dictionary. Because your choice of font format can have such a dramatic effect on website performance, we repeat our call to action from 2022 and urge developers to use WOFF2 fonts.

Looking at the difference between WOFF2 file sizes for self-hosted sites versus Google Fonts (we chose Google for the comparison as they are the most performance focused service), the difference is staggering. In the 50th percentile and up, self-hosted WOFF2 file sizes are on average double that of what is served by Google Fonts.

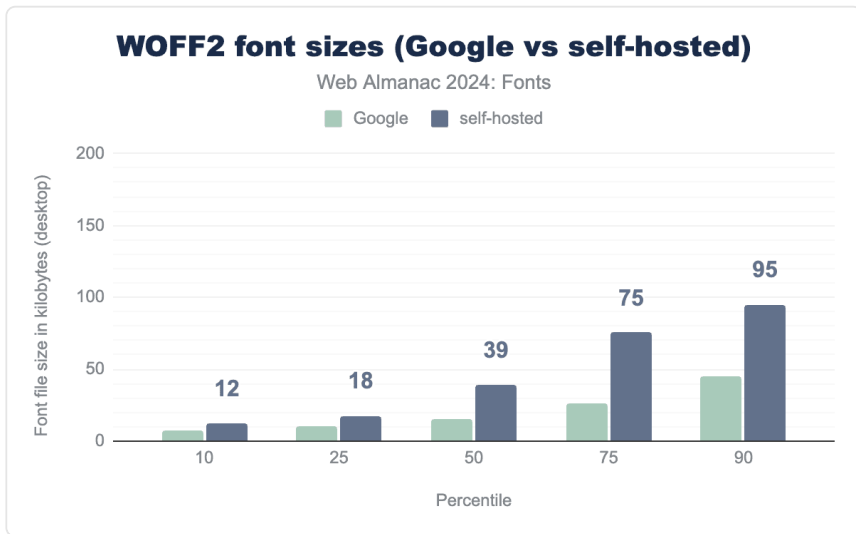


Figure 4.10. WOFF2 font file sizes (Google vs. Self-hosted).

We can only speculate on the reasons for this discrepancy. Some of it can be explained by differences in the set of fonts used by each of these groups. But as we'll see later, a lot of self-hosted fonts are downloaded from Google Fonts and should have similar compression rates. Another explanation could be that a lot of Google Fonts users are using the subsetting offered by Google, while the self-hosted users are serving the entire font. Whatever causes this difference, it'll be interesting to explore in future editions of the almanac.

Digging down even further, we can take a look at the individual table sizes used in fonts. As noted in 2022, a reasonable approach to measuring the impact of a particular OpenType table on overall file size is to multiply its median size by the number of fonts that include that table.

OpenType table	desktop	mobile
glyf	77%	78%
GPOS	6%	6%
CFF	5%	4%
hmtx	3%	3%
post	2%	2%
name	1%	1%
cmap	1%	1%
gvar	1%	1%
fpgm	1%	1%
GSUB	1%	1%

Figure 4.11. The top 10 OpenType tables measured by "impact".

Because it contains the actual glyph outlines, `glyf` remains the table with the highest impact. However, there have been some noticeable changes in the order of the tables compared to 2022. `GPOS` (Glyph Positioning, which controls the placement of glyphs) has overtaken `CFF` (Compact Font Format, which is an alternative to `glyf`). This trend is most likely due to declining usage of CFF fonts (on which more in the next section). It's also good to see that the `kern` table has dropped out of the top 10 as it is replaced by the more modern kerning implementation in the `GPOS` table.

The `post` and `name` tables are still in the top 10, which (as pointed out in the 2022 chapter) means the fonts have not been properly optimized. We would still love to see a tool that helps with this optimization process, as `post` and `name` mostly contain unnecessary data for web fonts (unless a web app allows users to add web fonts to its font menu).

Outline formats

The most common outline format continues to be TrueType (`glyf`), which accounted for 92% of both desktop and mobile fonts. This number has slowly ticked upward in recent years, suggesting that the `glyf` format has a solid hold over its nearest competitor, `CFF`, which held a declining share of 8%. Compared to 2022, the slight increase in `glyf` usage (2 percentage points for desktop, 1 point for mobile) corresponds almost exactly to the drop in `CFF` outlines.

Other outline formats, such as SVG and CFF2, registered a fairly minuscule presence well under 1% of web fonts (not pictured).

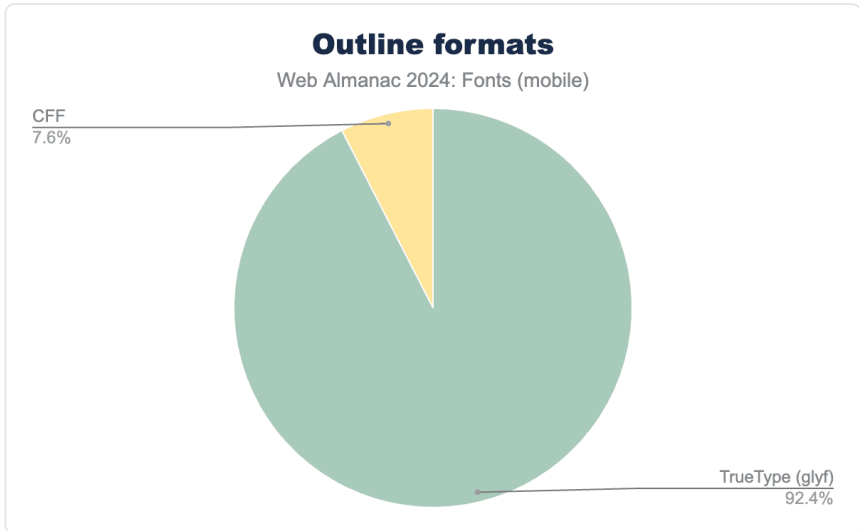


Figure 4.12. Outline formats.

There has also been a small increase in font outline sizes, consistent with general increases in the size of font files. Interestingly, this increase seems to disproportionately affect CFF. We think this is caused by the decrease in overall CFF usage combined with the fact that the most used CFF-based fonts are CJK fonts, which tend to be on the larger side.

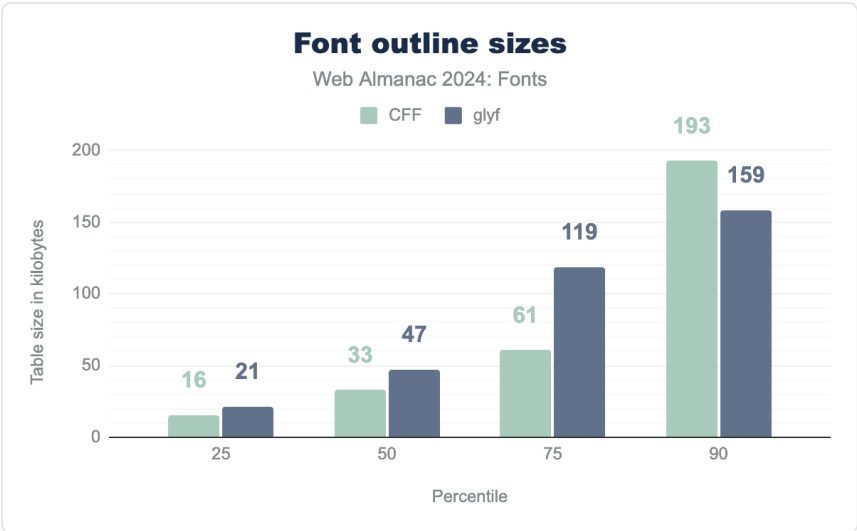


Figure 4.13. Font outlines sizes comparing `CFF` and `glyf`.

As noted in 2022, it is not a good idea to take raw table size comparisons at face value. Web fonts should always be compressed, so a fairer comparison would be to look at the compressed table sizes. For this we used the same approach as in 2022, by applying the median compression rates in the WOFF2 evaluation report⁶¹. Approximating compression paints a very clear picture: large fonts are better served using `glyf` (TrueType) outlines rather than `CFF`.

61. <https://www.w3.org/TR/2016/NOTE-WOFF20ER-20160315/#brotli-adobe-cff>

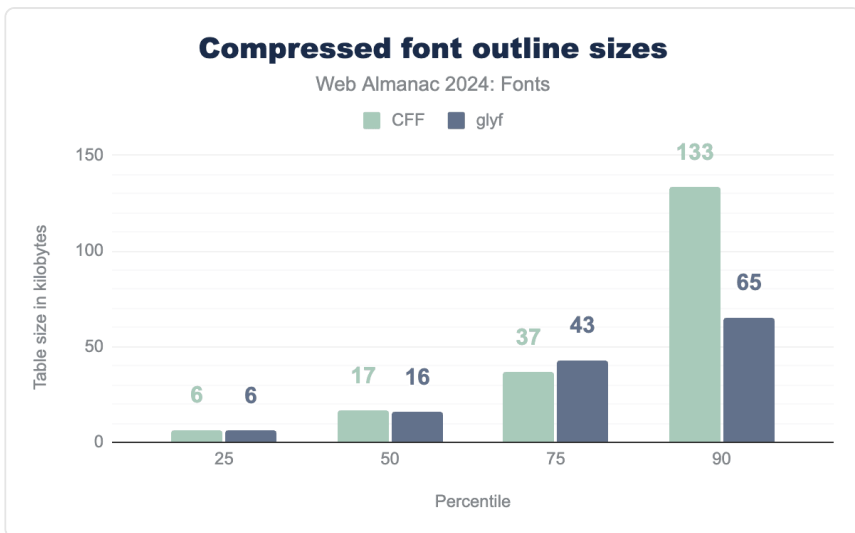


Figure 4.14. Compressed font outlines sizes comparing `CFF` and `glyf`.

There are some efforts underway to add cubic Bézier curves to `glyf`, so it will be interesting to see whether the difference in size between `glyf` and `CFF` is due to the different curve type (cubic Bézier curves have more control points), or inefficiencies in the `CFF` format. Our money is on a combination of both factors, but time will tell.

Resource hints

To decrease page loading times, web developers can instruct browsers to load essential resources, such as web fonts, before they're actually needed. This is done using resource hints, which guide the browser to load or render specific resources ahead of schedule. By leveraging resource hints, you can inform the browser to download and load critical fonts even if they haven't been explicitly referenced in the code yet, thereby improving page performance. The browser can then display content faster and provide a smoother user experience.

There are three types of resource hints relevant for web fonts, each with its own level of impact. The `preload` hint is the most impactful type of resource hint, as it directly instructs the browser to load a resource (such as a web font) before it's actually needed. The `preconnect` hint tells the browser to establish a connection with a server, preparing it for future requests, including font loading, and this has a medium impact on performance. The `dns-prefetch` hint signals the browser to prefetch DNS information for a specific domain, but doesn't initiate a connection nor font loading. This has a relatively low impact on

performance.

We made some changes in the data gathered on resource hints for this year's Almanac, as we realized the 2022 analysis was capturing too much. We are now measuring two different aspects. For `dns-prefetch` and `preconnect` we are only measuring resource hint usage against known font services (the same ones used throughout the chapter). This excludes pre-connecting and DNS-prefetching to one's own web host or CDN that self-hosts fonts, so actual usage is probably much higher. For `preload`, we are measuring when the hint has an `as` attribute with value `font`.

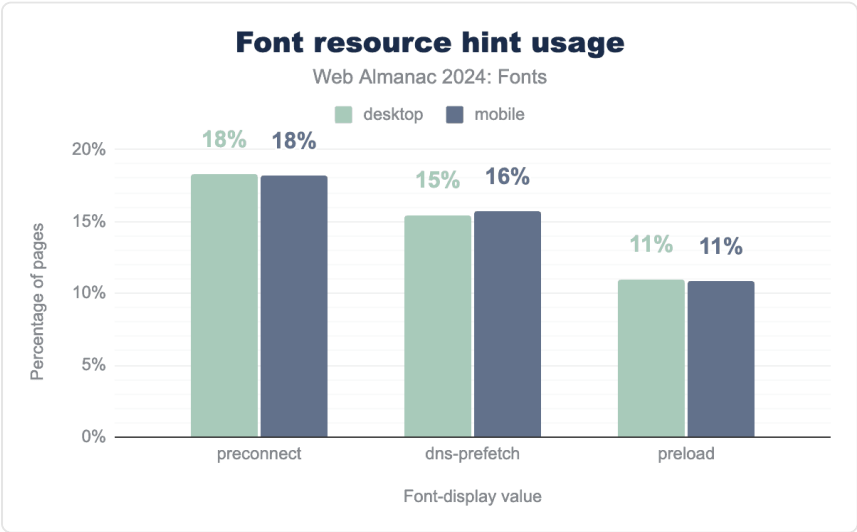


Figure 4.15. Font resource hint usage.

In this year's data, `preconnect` and `dns-prefetch` are used to speed up connecting to web font services at 18% and 16% respectively. The most effective resource hint, `preload`, is only used on 11% of pages. We'd like to see this number go up by a lot, as using the `preload` resource hint is the single most effective thing you can do to speed up your font loading! With that said, it is not always possible to use `preload`, for example if you use a service that does not provide stable font URLs. In those cases, it is best to use `preconnect` or `dns-prefetch` hints.

Unfortunately, the usage of resource hints for fonts hasn't changed much in the last two years either, so this is an underused (but very effective!) feature that we would love to see adopted more broadly by web developers.

Font display

The `font-display` descriptor for the `@font-face` CSS directive allows developers to choose when and how their website renders its text depending on the time it takes to fetch its fonts. Depending on the value of the `font-display` descriptor, the browser will either wait until web fonts are downloaded or swap to a fallback font after timing out.

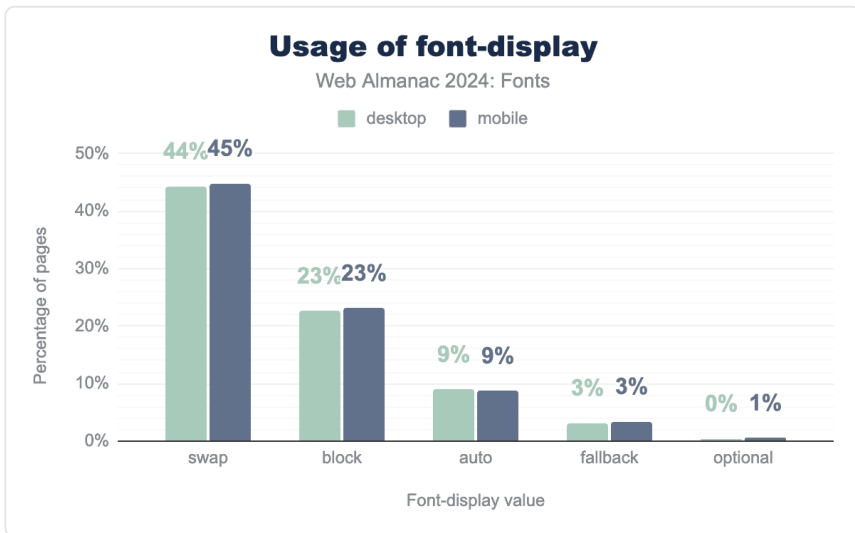


Figure 4.16. Usage of `font-display` values.

The use of `swap` for `font-display` has grown considerably in recent years, from 11% in 2020, to 30% in 2022, to about 45% on desktop and mobile this year. This is a good sign because `swap` offers earlier text rendering. The only tradeoff is a potential layout shift when the web font loads. This is preferable for users on slower connections as they'll see content much earlier, while users on faster connections might not even notice the layout shift (which can be further reduced using resource hints and font metrics overrides).

Meanwhile, the bad news is that this year's crawl also shows increasing use of `block` for `font-display`, which literally blocks text from rendering until the intended font is available or the timeout period has expired. While there are legitimate use cases for `block`, most websites should use `swap`, `fallback`, or `optional`. The rising use of `block` (24% desktop, 23% mobile) continues a trend from 2022, when it overtook `auto` as the second most common value chosen when using the `font-display` descriptor. The `auto` value itself is now used on 9% of websites, while `fallback` is used on 3%. The `optional` and `normal` values were used with `font-display` on less than 1% of websites.

We think the rise of `block` usage is concerning so we decided to investigate a bit further. Looking at the top 10 `@font-face` rules using `font-display: block` reveals an interesting discovery: all of the top 10 fonts are icon fonts!

Family	desktop	mobile
Font Awesome	15.5%	16.1%
ETmodules	1.7%	1.7%
TablePress	1.1%	1.1%
icomoon	1.0%	1.1%
vcpb-plugin-icons	0.8%	0.7%
fl-icons	0.6%	0.6%
dm-social-font	0.5%	0.5%
dm-font	0.5%	0.5%
dm-social-icons	0%	0.5%
dm-common-icons	0%	0.5%

Figure 4.17. The top 10 most commonly used fonts with `font-display: block`.

Font Awesome leads at a whopping 15% usage, while other icon fonts make up the remainder of the top 10. This makes sense, as icon fonts are usually encoded in the Private Use Area (or worse, override ASCII), so showing a fallback while an icon font is loading does not provide a good user experience. This is one of the major downsides of using fonts to display icons, but it does explain the rising use of `font-display: block`. We remain highly skeptical of the usage of icon fonts and feel that in most cases it will be better to provide icons as (embedded) SVG files.

Emoji fonts do not suffer from this problem as emoji are encoded in Unicode so they'll fall back to a system font properly when the font loading is delayed or fails. So it is safe to use emoji fonts on the web (using `font-display: swap` of course).

Families and foundries

Which font families were most popular this year, and which foundries made them? Taking a look at the top 20 there are few surprises in the first ten entries compared to 2022. Roboto still leads the pack, with a small increase in use*. Font Awesome use has remained fairly stable, and

so has Noto usage (unlike 2022, Noto is now split into script specific items). The only surprise is the decline of Lato, which has been overtaken by Poppins and Montserrat.

** As noted in 2022 the discrepancy between desktop and mobile use of Roboto is mostly likely due to the use of `local()` which loads a locally installed version of Roboto; due to it being the system font on Android, usage on mobile is low.*

Looking at the remainder of the top 20, Proxima Nova usage has increased slightly to reach about 1% of websites. As the only commercial, non-icon font in the top 20, this level of popularity is extremely impressive. Like in 2022, icon fonts make up about 18% of web fonts in 2024. The meteoric rise of Inter should also not go unnoticed, as it also stands around 1%. Due to its prominence in frameworks and libraries, we expect Inter to rise into the top 10 within the next few years.

Family	desktop	mobile
Roboto	15.2%	2.7%
Font Awesome	10.4%	12.4%
Noto Sans JP	6.1%	5.7%
Open Sans	5.6%	6.8%
Poppins	4.7%	5.8%
Montserrat	3.3%	3.9%
Lato	3.2%	3.8%
Noto Sans KR	1.6%	0.8%
Source Sans Pro	1.4%	1.7%
Noto Serif JP	1.2%	1.4%
Proxima Nova	1.2%	1.2%
Raleway	1.2%	1.4%
Inter	1.0%	1.1%
icomoon	0.9%	1.1%
Oswald	0.7%	0.8%
Ubuntu	0.6%	0.8%
eicons	0.6%	0.8%
Barlow	0.6%	0.7%
Rubik	0.6%	0.6%
NanumGothic	0.6%	0.3%

Figure 4.18. Top 20 family list for desktop and mobile.

Looking at foundries, the main surprise is the rise of Adobe Fonts as a foundry. There are two reasons for this change. This year we combined the two vendor identifiers that Adobe uses for their commercial and open source fonts. The other reason is that the Noto Sans CJK (Chinese, Japanese, Korean) fonts were a collaboration between Google, Adobe, and several other foundries. In 2022 these fonts were served with a Google vendor identifier and thus attributed to Google, but these fonts are now served with an Adobe vendor identifier and thus the very

popular Noto CJK superfamily is attributed to Adobe.

Foundry	desktop	mobile
Google Fonts	34%	19%
Adobe Fonts	14%	15%
Font Awesome	14%	19%
Indian Type Foundry	7%	10%
Łukasz Dziedzic	5%	6%
Julieta Ulanovsky	5%	6%
Mark Simonson Studio	2%	2%
Ascender Corporation	2%	2%
Paratype	2%	2%
Linotype	1%	2%

Figure 4.19. Top 10 foundries for desktop and mobile.

For this year, we thought it would also be interesting to take a look at the top 10 self-hosted fonts and the top 10 fonts for Google Fonts and Adobe Fonts (we excluded Font Awesome as they only serve a single font).

The top 10 for self-hosted fonts contains very few surprises. As we saw earlier, a lot of people switched from using hosted Google Fonts to self-hosting files from the Google Fonts library, and this is reflected in this list as well. Apart from Font Awesome, icomoon, and eicons, the most popular self-hosted families are all open source fonts.

Rank	Family
1	Font Awesome
2	Open Sans
3	Roboto
4	Montserrat
5	Poppins
6	icomoon
7	Lato
8	eicons
9	Inter
10	Source Sans Pro

Figure 4.20. Top 10 self-hosted families.

There is a close match between the top 10 most popular families from Google Fonts and the global top list. It should be noted that the Google Fonts top 10 contains quite a few CJK families (Noto Sans JP, Noto Sans KR, and Noto Serif JP) that are not present in the self-hosted list. It's great that CJK languages are seeing more use and that Google is actively supporting the development of global scripts (more on that later in the Writing systems section).

Rank	Family
1	Roboto
2	Noto Sans JP
3	Open Sans
4	Poppins
5	Lato
6	Montserrat
7	Noto Sans KR
8	Noto Serif JP
9	Source Sans Pro
10	Raleway

Figure 4.21. Top 10 families from Google Fonts.

The Adobe Fonts top 10 list is quite distinct from the other top lists because it contains primarily commercial fonts that Adobe licenses from foundries. As such, it offers an interesting insight into the world of commercial type (well, at least those foundries that licensed their fonts to Adobe). The most popular font at Adobe Fonts is Proxima Nova, which is no surprise as it also holds a high position on the global list. It's noteworthy that Adobe themselves only have two of their own fonts on the top list, with Adobe Garamond Pro at 4th and Acumin Pro at 7th place. The rest of the Adobe top 10 list is dominated by other foundries like Mark Simonson Studio⁶² (Proxima Nova), Paratype⁶³ (Futura PT), HVD fonts⁶⁴ (Brandon Grotesque), MoTyFo⁶⁵ (Sofia Pro), Dalton Maag⁶⁶ (Aktiv Grotesk), EuropaType⁶⁷ (Europa), The Freight Collection⁶⁸ (Freight Sans), and exljbris⁶⁹ (Museo Sans).

62. <https://www.marksimonson.com/fonts/view/proxima-nova>

63. <https://www.paratype.com/fonts/pt/futura-pt>

64. <https://www.hvdfonts.com/fonts/brandon-grotesque>

65. <https://www.motyfo.com/font-family/sofia-pro/>

66. <https://www.daltonmaag.com/font-library/aktiv-grotesk.html>

67. <https://europatype.com/>

68. <https://freightcollection.com/>

69. <https://www.exljbris.com/museosans.html>

Rank	Family
1	Proxima Nova
2	Futura PT Web
3	Brandon Grotesque
4	Adobe Garamond Pro
5	Sofia Pro
6	Aktiv Grotesk
7	Acumin Pro
8	Europa
9	FreightSans Pro
10	Museo Sans

Figure 4.22. Top 10 families from Adobe Fonts.

As noted, there has been an uptick in the support of various global scripts over the last two years, so let's take a look at that next!

Writing systems

There are thousands of languages in the world, and these languages are represented in at least 150 distinct character sets, known as writing systems or simply scripts. This presents type designers and developers with the daunting task of making and supporting fonts for so many different scripts with their own unique features, idiosyncrasies, and technical demands. Among the world's many character sets, the Latin script holds a position of somewhat dubious privilege as the longtime epicenter of digital type design. Because the Latin alphabet is the basis of digital character encodings and also the most commonly supported character set, other scripts tend to be commonly lumped into the unfortunate catch-all category of 'non-Latin' fonts. This term is very Eurocentric and should no longer be used. While this change of terminology may not happen any time soon, the overall balance has begun to shift in recent years as the overall level of support for various scripts has expanded along with access to free, high-quality web fonts handling these character sets.

The trend toward increasing support for multi-script fonts can be seen directly in this year's data. The overall proportion of fonts supporting Latin stands at roughly 46% this year, declining

by 8% for desktop and mobile websites since 2022. Meanwhile, there has been a corresponding increase in the number of fonts supporting multi-script text, amounting to many-fold increases essentially across the board. In other words, it's not that fewer fonts are being made in languages like English, French, Swedish, and Polish that use the Latin script, but rather more fonts are now available to support scripts like Arabic, Cyrillic, Hangul, Devanagari, and many others that now represent a growing share of text across the web.

To give a sense of where these increases have been most consequential, it helps to break down the level of script support in fonts by their overall presence. Cyrillic is the second most common script on the web, and rising. This year's crawl found fonts supporting Cyrillic on 13% of websites, a rise of about seven percentage points from 2022. Meanwhile, Greek character support has also risen about five percentage points to total about 8% of all websites.

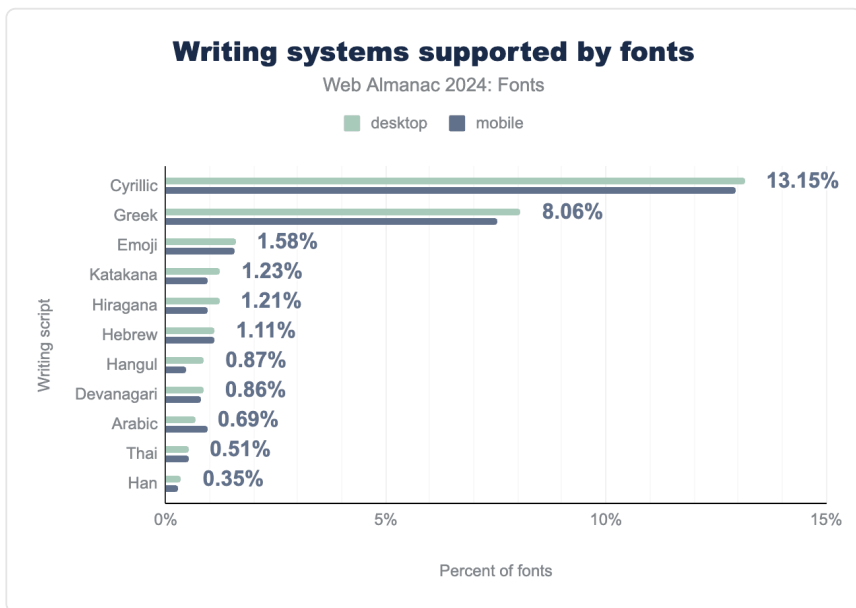


Figure 4.23. Writing systems supported by fonts.

In terms of the world's most widely used fonts, notable increases also registered in languages of the Indian subcontinent, East Asia, and the Middle East. Support for the Devanagari script, which is used for languages spoken by roughly 700 million people in Northern India and Nepal, has increased roughly three-fold. Likewise, support for Arabic has tripled, which is good news for over 400 million native speakers. Meanwhile, Thai support has nearly tripled for its 25-30 million speakers.

Use of web fonts, however, remains near-zero for Chinese. This is most likely a matter of file

size: these fonts are simply too large to serve as a simple WOFF2 file. Even compressed, a Chinese language font file will be several megabytes, which is far too large.

A recent technical development called Incremental Font Transfer⁷⁰ (IFT) offers a promising solution to this problem. Languages like Chinese usually have an extraordinarily large possible character set, but not every document will use every character. The IFT specification attempts to solve this problem by splitting a font file into “chunks” of characters that are loaded on demand when needed. It essentially streams the required portions of a large font to the browser depending on the content of a particular page. We look forward to seeing more “large” scripts use IFT to load web fonts efficiently.

In terms of the global scripts that are benefiting most from recent increases in web font support, some of the most dramatic numerical increases have registered for smaller language groups (found in less than 1% of fonts on this year’s crawl). Armenian support is up by roughly 500%, Cherokee by nearly 400%, and Tamil by about 300%. Writing systems found on fewer than 10,000 websites show even more dramatic increases from 2022. The Tibetan script showed a sixteen-fold increase this year, Syriac nine-fold, Samaritan roughly thirty-fold, and Balinese by about seven-fold. In other words, it’s not just the writing systems for huge populations like Hindi and Arabic that have benefited from the growing diversity of scripts supported by new type designs.

So, which font families are most used for different scripts? The expansion of certain families into “superfamilies” supporting several different scripts has made this question somewhat more complicated than it once was. Families like Roboto, Open Sans, Montserrat, and Lato are not only among the most popular Latin fonts, but also register on the top list for Cyrillic and Greek. So just because it is included in the top list for a certain script doesn’t necessarily mean it is *used* for that script, it just means it *supports* that script.

Noto is an outlier in this regard. Noto’s goal is to offer a single superfamily supporting every script encoded in the Unicode standard, which includes both living and extinct scripts. While Noto does not make the top 10 for Latin font families, where the competition is most intense, it is near the top for a large number of other scripts, particularly in East Asian countries using Noto’s CJK variants. Overall, Noto Sans and Serif together reached the top 10 for over 30 different scripts.

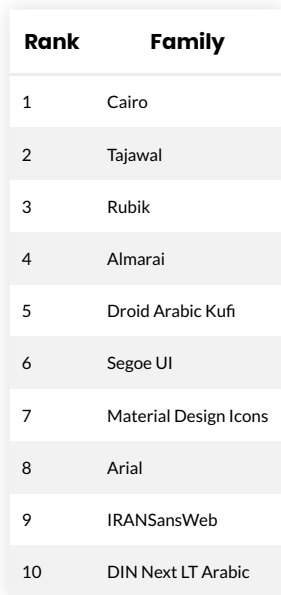
Another complexity arises when gathering data to measure “support” of a script. Let’s say a given script, as defined by Unicode, has 100 characters. If a font includes 50 of those characters, does it support the script? Depending on your needs, the answer might be different. For the purpose of this chapter we have (rather arbitrarily) defined “support” as having more than 5% of that script’s characters. That’s a low threshold. The reason for this low threshold is that most scripts are complex, and very few fonts have 100% coverage of any given script. This low-

70. <https://www.w3.org/TR/2024/WD-IFT-20240709/>

threshold measurement is meant to capture the intent to support a script, and we feel that covering even 5% of a given script shows that the type designer did intend to support it. Needless to say, this approach will generate some false positives, and so we must take these results with a grain of salt.

With those caveats out of the way, let's take a look at the top lists for Arabic, Devanagari, Korean, Japanese, and Chinese. While there is some overlap between these scripts, there is usually less overlap with popular Latin fonts (apart from the exceptions noted above).

The top 10 list for Arabic contains primarily fonts that were either designed explicitly for the Arabic script (for example, Cairo, Tajawal, and Almarai), or fonts created for other scripts that were extended to support Arabic (Droid Arabic Kufi, Segoe UI, Arial, DIN Next).



Rank	Family
1	Cairo
2	Tajawal
3	Rubik
4	Almarai
5	Droid Arabic Kufi
6	Segoe UI
7	Material Design Icons
8	Arial
9	IRANSansWeb
10	DIN Next LT Arabic

Figure 4.24. Top 10 families supporting Arabic.

The one oddity in the list of top Arabic web fonts is the inclusion of Material Design Icons. One possible reason for this font's apparent popularity in Arabic is that it maps some icons to code points used in Arabic's Unicode range, but we were unable to verify this conjecture through the actual crawl data. Regardless, what stands out most in the top 10 list for Arabic fonts is the variety of styles represented, which is great news because Arabic is a script with many distinctly expressive forms of writing.

Like Arabic, the top 10 Devanagari family list contains a mix of fonts specifically designed for the Devanagari script and existing families extended to support Devanagari. The top font is Poppins, followed by Noto Sans. We're not sure if these fonts are popular because they are used

for Devanagari or if they're in the list just because they're popular and happen to support Devanagari. However, there's no question about Hind, Mukta, Rajdhani, and Baloo 2, which are fonts explicitly designed for Devanagari.

Rank	Family
1	Poppins
2	Noto Sans
3	Hind
4	Mukta
5	Segoe UI
6	Rajdhani
7	Teko
8	FiraGO
9	SVN-Poppins
10	Baloo 2

Figure 4.25. Top 10 families supporting Devanagari.

For Korean, Pretendard is listed as the most commonly used font, with various Noto versions in the top 4 (adding up the totals for the various Noto versions would put it in the number one spot). And while open-source fonts have a strong presence in many writing systems, Korea had an especially strong showing: their ten most popular fonts are all open source!

Rank	Family
1	Pretendard
2	Noto Sans KR
3	NotoKR
4	Noto Sans CJK KR
5	나눔고딕
6	Spoqa Han Sans Neo
7	SpoqaHanSans
8	NanumGothic
9	나눔스퀘어
10	SUIT

Figure 4.26. Top 10 families supporting Korean.

The top 10 Japanese font list is surprisingly similar to the Korean list. Noto and Pretendard take the top spots. It's also interesting to see three Korean fonts in the Japanese top 10 list: Noto Sans KR, 나눔고딕 (Nanum Gothic), and 나눔스퀘어 (Nanum Square). While these are Korean fonts, they also have support for a significant number of Japanese characters and, by our standard of measurement, have made the list. Having so much crossover with the Korean top 10 list, it's fitting that the Japanese list also consists entirely of open-source fonts. Nice!

Rank	Family
1	Noto Sans JP
2	Pretendard
3	Noto Serif JP
4	Noto Sans TC
5	Noto Sans KR
6	나눔고딕
7	Rounded Mplus 1c
8	Zen Kaku Gothic New
9	나눔스퀘어
10	Noto Sans CJK JP

Figure 4.27. Top 10 families supporting Japanese.

Unfortunately, the data for top Chinese language fonts seems to be unreliable. The Chinese list only contains a single (Traditional) Chinese font. The rest of the families are all false positives, most likely due to the inclusion of Kanji in Japanese and Korean fonts. Kanji characters are adapted from the Chinese writing system, and due to the Han unification in Unicode⁷¹ share the same code points as the Chinese writing systems.

71. https://wikipedia.org/wiki/Han_unification

Rank	Family
1	Noto Sans JP
2	Noto Sans TC
3	나눔스퀘어
4	나눔바른고딕
5	나눔고딕
6	Noto Sans KR
7	源ノ角ゴシック JP
8	카카오OTF
9	Noto Sans CJK JP
10	Noto Serif JP

Figure 4.28. Top 10 families supporting Chinese.

It's fair to say there are very few Chinese (Traditional or Simplified) fonts being used. The reason is that Chinese has a very large possible set of characters (100,000 or more) and as noted above, Chinese fonts are especially large. This is an area where the new Incremental Font Transfer standard⁷² will definitely help and we hope to see more Chinese fonts in the coming years (and many other writing systems!).

OpenType features

OpenType features are one of the “hidden” gems in the OpenType format. Some OpenType features are required to render text correctly (quite common in various scripts) while others offer different stylistic options (for example, alternate versions of the ampersand). Browsers (and other applications) often enable some features by default in case they are required to render the text correctly, while others work on an opt-in basis. It's up to the type designer to decide what OpenType features they'll include in their fonts, so not all fonts have the same features. In this section we'll take a look at the prevalence of OpenType features and how they're used most often on the web.

72. <https://www.w3.org/TR/2024/WD-IFT-20240709/>

55%

Figure 4.29. Fonts including OpenType features.

The prevalence of OpenType features has risen steadily in recent years, reaching about 55% of fonts. This marks an increase of about seven percentage points from 2022. Looking at individual features we see a similar increase. This year’s data shows ligature (`liga`) support up from 10% to 40%, kerning (`kern`) from 13% to 38%, localized forms (`locl`) from 10% to 27%, fractions (`frac`) from 8% to 26%, numerator (`numr`) from 7% to 19%, and denominator (`dnom`) from 7% to 19%.

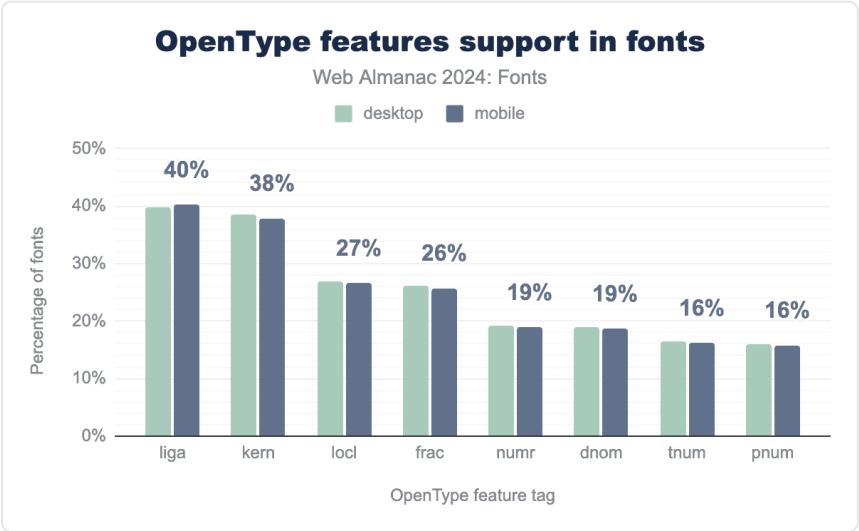


Figure 4.30. OpenType features support in fonts.

Some of OpenType’s lesser used features have also shown increased uptake in font files used on the web this year versus 2022: access to alternates (`aalt` , which provides multiple versions of a given character), ordinals (`ordn` , which provides ordinal numbers), and character composition (`ccmp` , which provides special character combinations) have each risen from about 1% to 3% support in fonts used for both desktop and mobile websites.

It’s great that fonts now support more features, but are they actually being used on websites? There are two different CSS properties available to control the behavior of fonts on a website: `font-variant` (and its various longhand properties that make it up) and the lower-level `font-feature-settings`. The `font-variant` property is used to select from a set of

predefined font variants, such as small caps (`small-caps`). The `font-feature-settings` property should primarily be used when there is no `font-variant` equivalent.

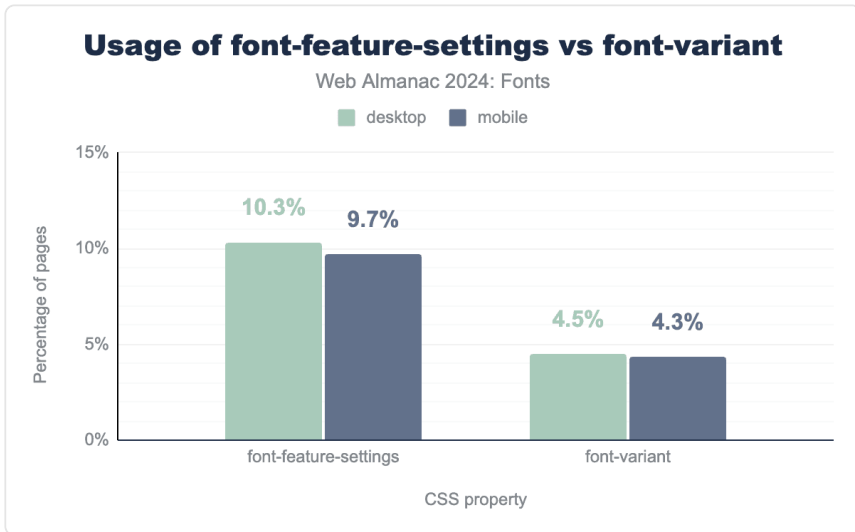


Figure 4.31. Usage of `font-feature-settings` vs. `font-variant`.

Overall use of `font-feature-settings` is down by about 3 percentage points this year, from 13.3% to 10.3% on desktop and 12.6% to 9.7% on mobile websites. Meanwhile, use of `font-variant` has risen slightly, up from 3.9% to 4.5% on desktop and 3.5% to 4.3% on mobile websites. The most likely explanation is that more and more sites are using the newer (and better!) `font-variant` properties now that they are better supported. We hope to see this trend continue and `font-variant` eventually overtake `font-feature-settings` as the primary method to enable or disable OpenType features.

This point is driven home if we take a look at the features used in combination with the `font-feature-settings` property. All of the top features used with `font-feature-settings` have `font-variant` equivalents!

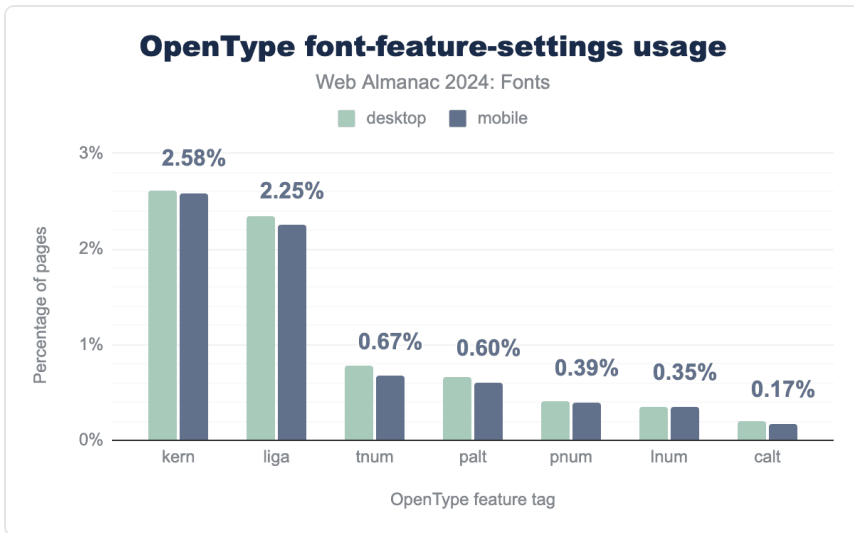


Figure 4.32. Most popular `font-feature-settings` values.

Worse, many of the features are enabled by default in all browsers (kerning, common ligatures, and contextual alternates), so there is no need to include it in `font-feature-settings`. The only reason would be to disable these features, but that would be an odd thing to do. The good news is that the use of these features hasn't really grown since 2022. Aside from custom and non-standard OpenType features, there isn't any need to use `font-feature-settings`. You can achieve expert-level typesetting with the `font-variant` properties alone.

Variable fonts

Variable fonts represent a major advancement in expressive possibilities for digital typography. Officially known as OpenType Variable Fonts (OTVF), this format allows for the continuous variation of letterforms along a set of axes that fine-tune the font's appearance. In other words, a single variable font file contains the full range of instances in a font family, as well as every granular adjustment and combination of adjustments the designer has defined along the specific axes included in the font. So, how does this work?

Whereas a conventional font family may offer bold or thin weights, a variable font allows the user to make the letters exactly as bold or thin as they like using the weight (`wght`) axis. Likewise, tweaking the width (`wdth`) axis can push or pull the letterform into condensed and extended variants. And because the size of type often calls for fine adjustments (such as x-height) that affect legibility and typographic color, variable fonts permit subtle refinement of

optical size until the letters feel just right for the space they occupy and role they play.

In addition to giving designers and end-users more typographic expressiveness, variable fonts can also be a performance improvement if you use multiple styles from the same family. Internally variable fonts do not store outlines for each style, but a more efficient set of deltas and offsets, so a variable font can be many times smaller than their corresponding “static” styles.

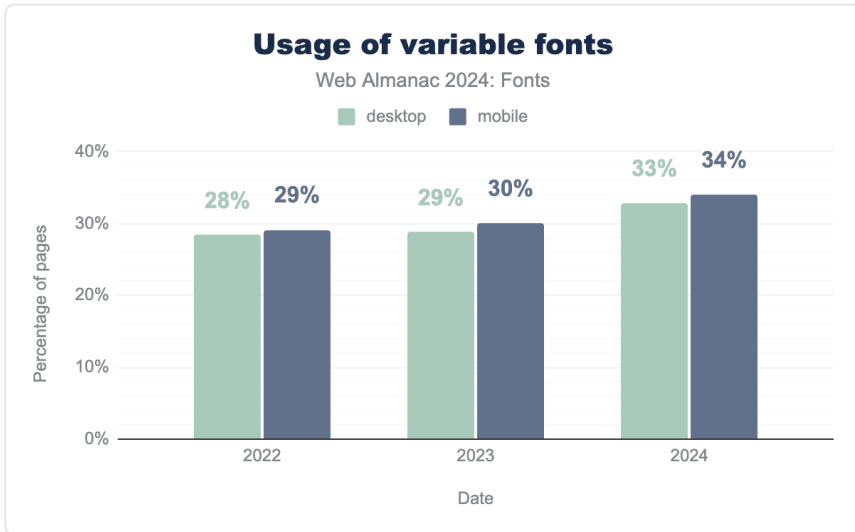


Figure 4.33. Usage of variable fonts on websites over time.

While variable fonts may sound like an exotic new technology, they are already supported by all major browsers and are used on an increasing number and share of websites across the web. In all, about 33% of websites are now using variable fonts. This marks a 4–5 percentage point increase in variable fonts across the web since 2022. Even so, the jump in variable font adoption was much larger between 2020 and 2022, when the presence of variable fonts nearly tripled. It will be interesting to continue tracking this rate of adoption in the coming years to see whether variable font use keeps growing or has begun to reach a plateau.

As an aside, usage doesn’t necessarily imply that a web developer chose to use a variable font over regular fonts. It’s very likely that quite a large percentage of web pages are using variable fonts because the service(s) they use chose to serve a variable font instead of regular font styles. Variable fonts usually contain instances that correspond to the individual styles of a font family, so a service can easily serve a variable font without web developers needing to modify their CSS styles. In fact, as noted in 2022, this is likely the cause of the sudden jump in variable font usage, as Google Fonts have been rapidly replacing their fonts with variable equivalents.

Family	desktop	mobile
Noto Sans JP	27%	23%
Open Sans	16%	18%
Montserrat	9%	10%
Noto Sans KR	7%	4%
Noto Serif JP	5%	5%
Raleway	3%	4%
Inter	3%	3%
Noto Sans TC	2%	2%
Google Sans 18pt	2%	2%
Oswald	2%	2%

Figure 4.34. Top 10 most used variable fonts.

The most popular variable font family this year was Noto Sans JP, which was found on about 27% of desktop websites and 23% of mobile websites. Its serif variety, Noto Serif JP, accounted for a further 5% of websites. The Hangul version of the same font, Noto Sans KR, also pulled a considerable share, found on a little over 7% of sites using variable fonts. And while Noto Sans TC (Traditional Chinese) trailed at just about 2% and Noto Serif TC had a little under half a percent, altogether Noto’s CJK offerings have an impressive footprint in the current adoption of variable fonts: about 42% of all the sites using variable fonts come from the Noto super family.

Open Sans was the second most popular variable font this year, being found on 16% of the websites using variable fonts. Montserrat was also found on 9%–10% of these websites. Given that both Open Sans and Montserrat support several of the most widely used scripts, including Greek and Cyrillic, the adaptability of these fonts to many use cases may help explain why these are some of the most used typefaces on the web, both as variable fonts and in general popularity.

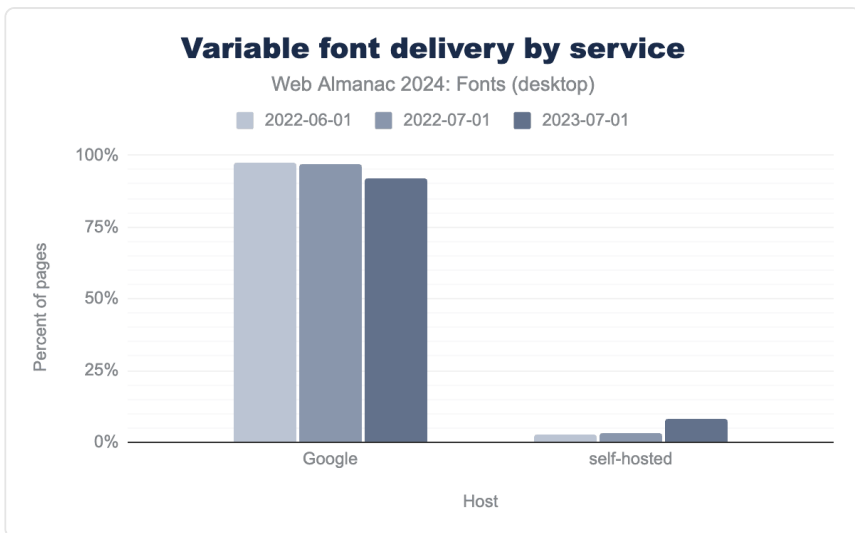


Figure 4.35. Popular hosts for serving variable fonts over time.

In terms of serving variable fonts, the vast majority of users choose between two options: Google and self-hosting. No other variable font source registered more than a fraction of a percent of the websites using this technology. And while Google Fonts serves the vast majority of variable fonts (92% desktop, 91% mobile), their share of this market has actually declined slightly in recent years. In 2022, Google Fonts served 97% of variable fonts for desktop and mobile, meaning their share is down about 5- 6% for desktop and mobile this year. This change corresponds to an equivalent rise in self-hosting as a website's primary means of delivering variable fonts, amounting to 8% of sites crawled this year. Given that the share of non-Google variable fonts served in 2022 was just 3%, the corresponding rise of 5 percentage points in the share of websites that self-host variable fonts appears to reflect a direct drift in Google's share of the variable font market. As noted above, other services like Adobe Fonts and Type Network served below 1% of the variable fonts found on the web this year.

Like color fonts, variable fonts also have two competing outline formats: the variable extensions of the `glyf` format and Compact Font Format 2 (`CFF2`). The main technical differences between the `glyf` format and `CFF2` are the types of Bézier curves, more automated hinting, and (perhaps most importantly) the companies backing them. `CFF2` is backed by Adobe and `glyf` by all other major contributors to the OpenType specification (Google, Microsoft, and Apple).

99%

Figure 4.36. Variable fonts using the `glyf` outline format.

Unfortunately for Adobe, over 99% of the outlines used for variable fonts this year were in the `glyf` format. This overwhelming share for `glyf` outlines has been fairly consistent since 2022, dropping only 0.2 percentage points in the intervening two years. Since variable fonts introduction in 2016, `CFF2` has only accumulated a miniscule 0.6% of usage. Usage of `CFF2` just doesn't seem to be picking up and it's worth wondering if efforts are not better spent elsewhere.

For now, our recommendation is the same as in 2022: avoid `CFF2` based variable fonts because browser and operating system support is still patchy (and there is a perfectly viable alternative in `glyf`).

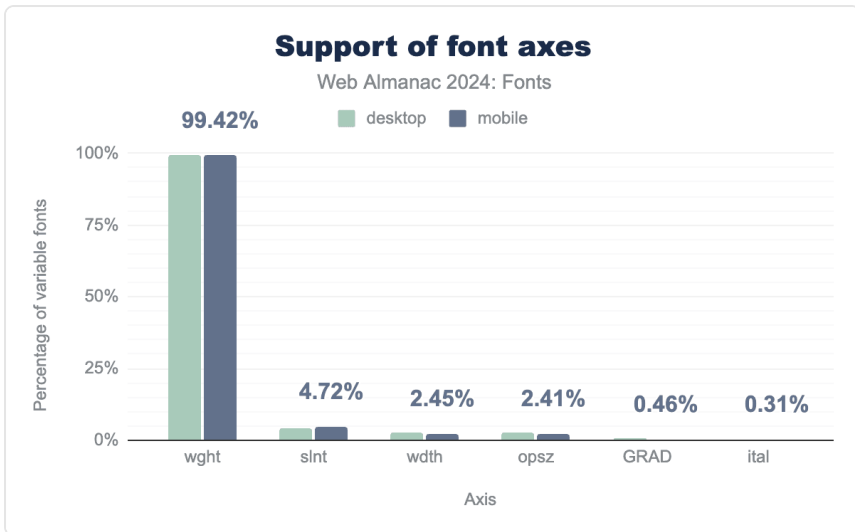


Figure 4.37. Support of font axes in variable fonts.

In order to get a sense of how type designers are approaching the technical affordances of variable fonts, this is the first year the HTTP Almanac has gathered data about the axes supported in variable fonts used on the web. At over 99%, no other variable font axis comes close to the near ubiquity of `wght`. Trailing in the single digits were slant (`slnt`) at 5%, width (`width`) with 2%, optical size (`opsz`) with 2%, and the grade axis (`GRAD`) with about half a percent on desktop and mobile websites. It will be interesting in the coming years to continue

tracking which axes are found in variable fonts on the web because this will signal trends in the design and availability of different affordances for using these fonts.

Beyond these trends in the design of variable fonts, how are these variable font axes actually being used on websites? For that, we turn to CSS data. There are generally two ways to use variable fonts in CSS, through the low-level `font-variation-settings` property or through the good old `font-weight`, `font-stretch`, and `font-style` properties that have been updated to support variable fonts.

It's difficult to detect usage of variable fonts with the standard font properties in CSS because it requires reconstructing the CSS object model and tracing the usage of each family and associated property for each crawled site. Instead we take a look at the usage of `font-variation-settings` as an approximation (so take these results with a grain of salt).

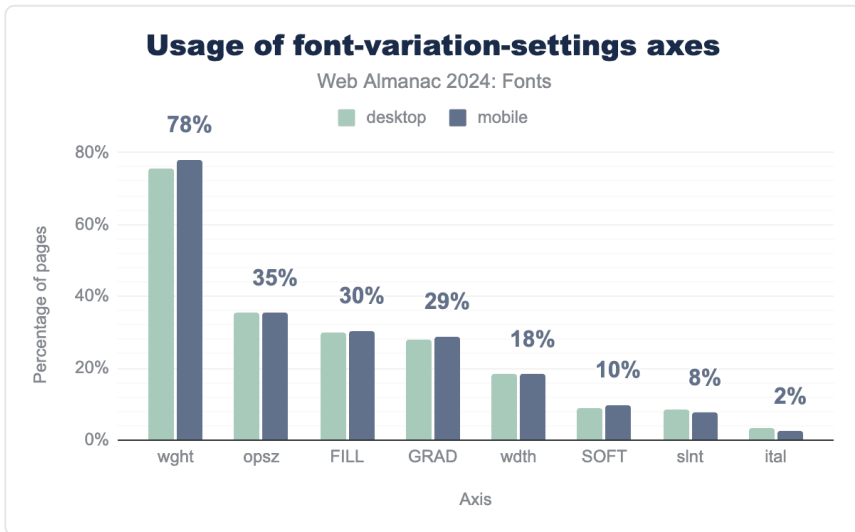


Figure 4.38. Use of `font-variation-settings` axes on pages using variable fonts.

The weight axis (`wght`) remains the most commonly used variable font axis by a wide margin. This axis, which affects the thinness or boldness of glyphs, was found on over 78% of sites using variable fonts this year. This is a slight decline from two years ago when the weight axis was used on 82% of sites. This decline is most likely due to an increase in people switching to the more common `font-weight` property for setting variable font weight axis values. At the same time, we see an increase in most other “standard” axes: `opsz` (optical size), `width` (width), `slnt` (slant), and `ital` (italic). While there are standard CSS properties for each of these axes, people are either not aware of how to use them or are following old out-dated advice. We hope to see the use of these values drop over the coming years as more people

become accustomed to using variable fonts.

It's also good to see growing use of several non-standard axes (`FILL` , `GRAD` or grade, and `SOFT`) which do not have CSS properties to control them. The only way to use these axes is through the `font-variation-settings` property. If their popularity grows, it would be a clear indicator to the authors of the CSS specification (and the OpenType standard) that these axes deserve their own CSS properties (if applicable; some axes are highly specific to a single typeface and can not be standardized).

Other interesting trends appear in the use of more niche variable font features this year. Variable font animation is on the rise but remains quite low. In 2022, a total of just 163 desktop and 147 mobile sites were found to use animation by the HTTP Archive crawl, but this year that figure grew to 35,000 desktop and 46,000 mobile sites. On the scale of the internet, this is still just a tiny fraction of websites (0.28%), but suggests that more advanced variable font features are slowly being adopted.

Color fonts

Color fonts, also known as chromatic fonts, offer the capacity to display letters in multiple colors or gradients.



Figure 4.39. Nabla Color⁷³ a COLR v1 font using gradients by Arthur Reinders Folmer (Typearture⁷⁴) and engineered by Just van Rossum⁷⁵.

73. <https://nabla.typearture.com/>

74. <https://www.typearture.com/>

75. https://wikipedia.org/wiki/Just_van_Rossum

The number of websites using color fonts is still quite small on the scale of the entire internet, but has risen considerably over the last two years. The 2022 crawl found color fonts on between 1,000–1,400 websites, amounting to 0.02% of the total websites surveyed in that year’s data. This year’s crawl found color fonts on between 5,800–6,200 websites, amounting to 0.04%. Going by these figures, it would appear that the number of websites using color fonts has tripled over the last two years, despite remaining a small fraction of all websites.

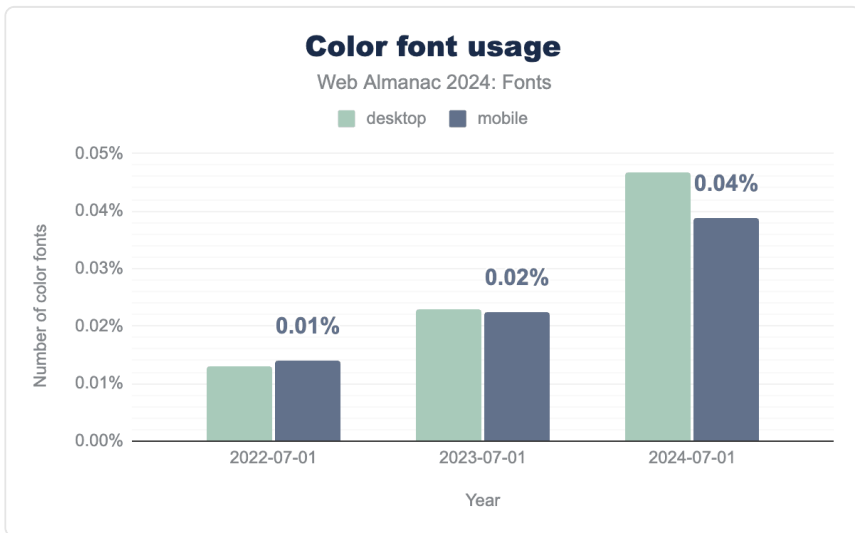


Figure 4.40. Color font usage over time.

As a relatively new technology, color font use is split between several competing formats. The most popular color font formats, `SVG` (also called OT-SVG, not to be confused with SVG images) and `COLR`, are based on vector outlines. The difference between these two is that `COLR` reuses the outline format in the font itself (i.e. either `glyf` or `CFF`) while `SVG` embeds an SVG document for each glyph. `COLR` also provides integration with OpenType Variations, a feature that OT-SVG lacks. So it is possible to make a color variable font using `COLR` (see Nabra⁷⁶ for a great example of a color variable font), but not with `SVG`. Still, `SVG` was the most commonly used format this year, being found on 53% of desktop sites with color fonts. The `COLR` format has two different versions: `v0` and `v1` (a newer version of `COLR` with more features, such as gradients). We'll refer to these as `COLRv0` and `COLRv1` from here. `COLRv0` fonts also carried a considerable share with 28% of desktop sites. In contrast, `COLRv1` fonts were found on just 8% of desktop sites using color fonts.

76. <https://nabra.typearture.com/>

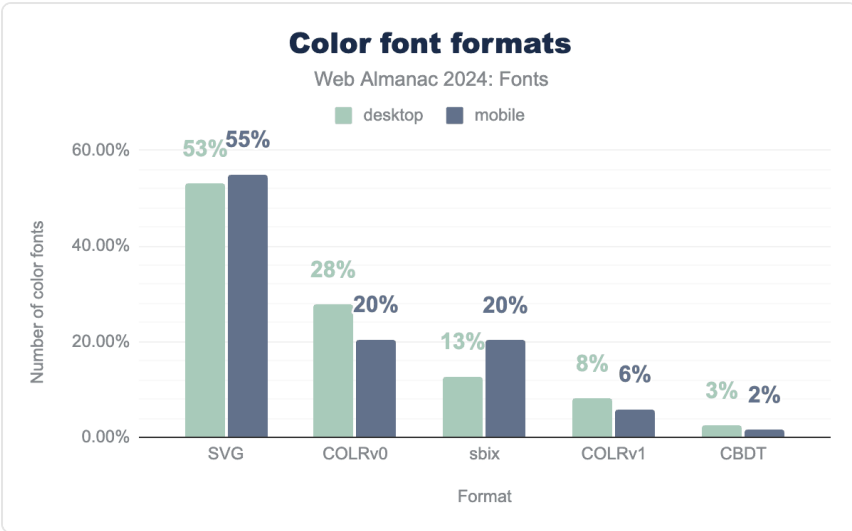


Figure 4.41. Color font format usage.

The two raster-based color font formats, SBIX (Sbit Binary Image eXtension, lowercased as an OpenType table) and `CBDT`, were found on a small but notable share of websites using this technology. SBIX accounted for 13% of desktop, while `CBDT` files made up 3% of desktop color fonts.

The most popular color font families this year were Noto Color Emoji (25% desktop, 28% mobile) and Joy Pixels SVG (23% desktop, 11% mobile). Two Japanese fonts also registered a strong share of color fonts in this year's data: 貂明朝 (Ten Mincho, 11% desktop, 13% mobile) and 貂明朝テキスト (Ten Mincho Text, 7% desktop, 9% mobile). However, as we'll see later, it is questionable to call these true color fonts as they contain only a handful of color glyphs.

Family	desktop	mobile
Noto Color Emoji	25%	28%
JoyPixelsSVG	23%	11%
韶明朝	11%	13%
韶明朝テキスト	7%	9%
Toss Face Font Mac	4%	7%
noto-glyf_colr_1	3%	2%
Material Icons Two Tone	3%	5%
Twemoji Mozilla	1%	0%
Aref Ruqaa Ink	1%	1%
Cairo Play	1%	2%

Figure 4.42. Top 10 most used color fonts.

We wanted to draw special attention to the most used color font: Noto Color Emoji. It is somewhat unique in that it comes in multiple color formats and thus occupies several spots on the list of most popular color fonts. Noto Color Emoji has a COLR, CBDT (raster), and an SVG version. The COLR version is the most popular, followed by CBDT, and finally SVG.

Given the relatively small number of websites with color fonts, usage trends over the last two years have led to some significant changes in the most popular formats. There was a large decrease in the use of SVG from 82% in 2022 to 53% in 2024. Both COLRv0 and COLRv1 saw a steady increase in use in 2023 and 2024. The combined use of COLRv0 and COLRv1 formats was 26% of total color font usage in 2023 and 36% in 2024. That's a significant increase, and is expected to continue as SVG declines in popularity.

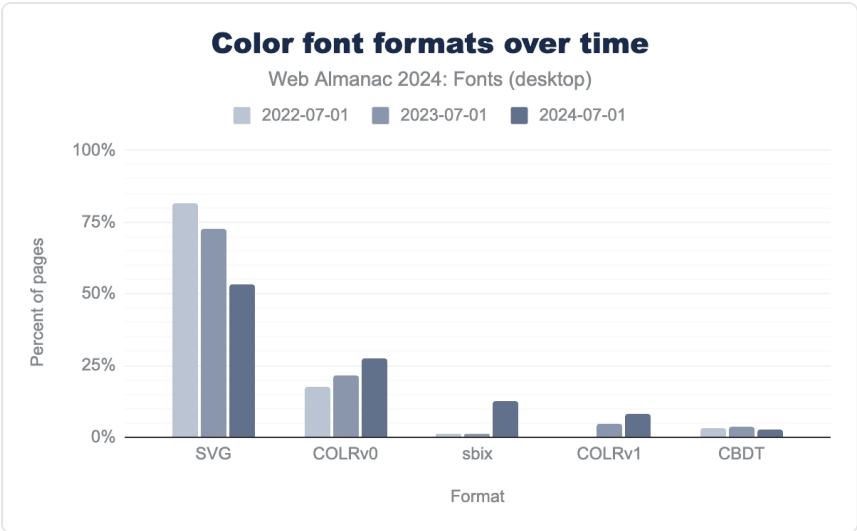


Figure 4.43. Color font format usage over time.

The most unexpected change in this year’s color font data is the sudden rise in SBIX format. SBIX was introduced by Apple to support the storage of bitmap images within font files. However, the SBIX format has some limitations when it comes to efficiency: binary overhead and lack of compression. This year’s rise in SBIX usage is almost entirely due to the popularity of the Toss Face Font⁷⁷, a Korean font with thousands of emojis and a very large file size. At 12.7 MB uncompressed and 11.7 MB compressed (WOFF2), this font would surely be better delivered in the COLR format. In fact, the home page for Toss Face Font is using a simplified COLRV0 font, which, sadly, does not appear to be publicly available. The COLRV0 version weighs in at 6 MB uncompressed and 1.1 MB compressed (WOFF2). While it may be hard to create a COLRV1 version with the same fidelity as the SBIX version, the COLRV0 version demonstrates the benefits of conversion to a vector format in terms of file size reduction. We hope the COLRV0 version is soon made available to the public, and this sudden rise in SBIX usage is a temporary blip.

Although emoji offered a major impetus for the initial development and support of color fonts, the last edition of the HTTP Almanac found a surprising lack of emojis on websites where color fonts had been used (just 4% of the desktop crawl and 2% of mobile). This year’s data shows a strong upward shift in the use of color fonts for emoji: 42% of desktop and 31% percent of mobile results found a color font with at least some emoji characters. While this signals significant growth in the use of color fonts for emoji, it is also worth noting (once again) that the total number of websites using color fonts is still quite small. This year’s crawl found color fonts

77. <https://toss.im/tossface>

on about 5,800 desktop and 6,200 mobile sites across the web.

It's worth noting a point of complexity in the data gathered on color fonts. Our analysis considers a font with a single color glyph a color font. That is a rather broad definition and also includes many fonts which are primarily used for their non-color glyphs. If we remove fonts with only a small percentage of color glyphs from the equation, the use of color font technology looks very different. For example, the top three fonts using `SVG`, 韶明朝 (Ten Mincho), 韶明朝 テキスト (Ten Mincho Text), and Source Code Pro make up a combined 41% of `SVG` total usage, while having only a handful of color glyphs. Removing all “color” fonts with less than 5% of their codepoints mapped to color glyphs from the calculation would reduce the share of `SVG` to roughly 5%! While the `COLR` formats also have some fonts with a low percentage of color glyphs, that percentage is much lower at 3-4%, reducing the `COLRv0` share to 25.1% and 7.2% for `COLRv1`. This would make `COLR` the most popular format by far!

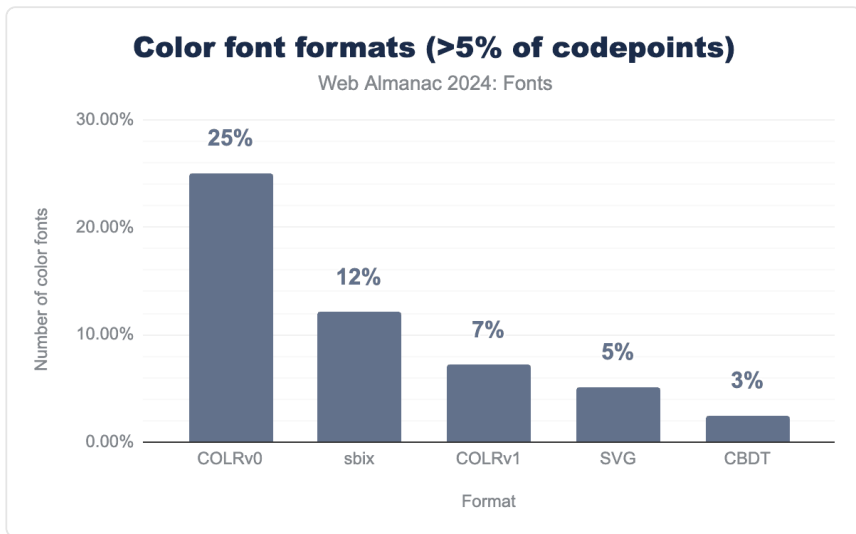


Figure 4.44. Most popular color formats after removing fonts with only a handful of color glyphs.

We plan to refine our analysis of color fonts in future editions of the Almanac, but our expectation is that `COLR` usage will continue to grow and it will soon become the dominant color font format (if it isn't already).

The observant reader may have noticed that the color font usage percentages between the 2022 fonts chapter and 2024 font chapter are slightly different. On closer inspection of the color font data we noticed that some popular font tools include an empty `SVG` table, even though the font does not contain any color glyphs. This erroneously caused the popularity of `SVG` to be inflated. We have corrected this in 2024 and included the 2022 and 2023 data for comparison.

Font smoothing

The non-standard CSS font smoothing properties⁷⁸ claim to allow developers to choose between different ways of rendering a font. As these are non-standard properties, they are prefixed with a dash and a vendor prefix: `-webkit-font-smoothing` and `-moz-osx-font-smoothing`. Theoretically they allow web developers to switch between grayscale and subpixel antialiasing on Apple's MacOS only. That's interesting because in 2018, Apple removed subpixel antialiasing from MacOS⁷⁹ with the release of Mojave. Even more interesting, more than about 70-77% of all websites set this property to `antialiased` or `grayscale`.

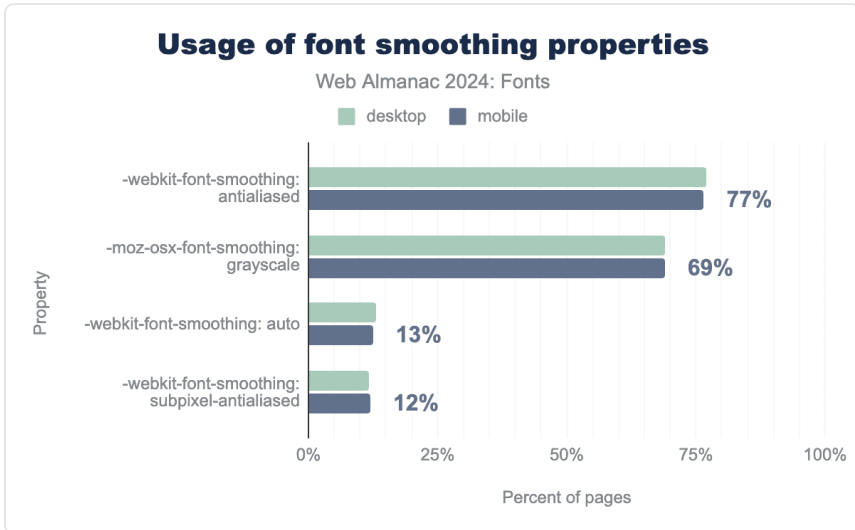


Figure 4.45. Font smoothing property usage.

So what is going on here? Why is everyone setting this property if MacOS uses grayscale antialiasing regardless of the value of the property? Is it legacy cruft set by old versions of CSS libraries and frameworks, or does it actually do *something*?

To clear this up: the property does nothing on other operating systems and it does not change the antialiasing settings on MacOS. However, it disables the MacOS-specific stem darkening that makes fonts look a little bolder than they were drawn. Apple implemented this to improve the legibility of small text by making the stems (the vertical parts of a letter) a bit thicker. Setting `-webkit-font-smoothing: antialiased` (and its `-moz` equivalent) disables this stem darkening and makes the font appear a little lighter, especially on darker backgrounds.

78. <https://developer.mozilla.org/docs/Web/CSS/font-smooth>

79. https://wikipedia.org/wiki/MacOS_Mojave#Removed_features

Our hunch is that a lot of designers and developers do not like this MacOS-specific behavior and use these properties to disable it. We're not clear what the best solution might be. It's clearly a MacOS-only issue (which would generally disqualify it from standardization in CSS). Standardizing the `font-smoothing` property does not make sense either. The stem darkening is only tangentially related to antialiasing, plus browsers and operating systems generally don't allow you to switch antialiasing methods anyway. However, judging by the popularity of this font smoothing choice in the crawl data, it certainly appears to be a problem for web developers and a better solution deserves some careful consideration.

Generic family names

While system font families are not ideal from a design perspective, as universally available options they offer a useful fallback in case self-hosted or web fonts fail to load for some reason. They're also a good alternative if you want to achieve a "native" look and feel for your web application, or if you have performance budget constraints and can't use web fonts at all.

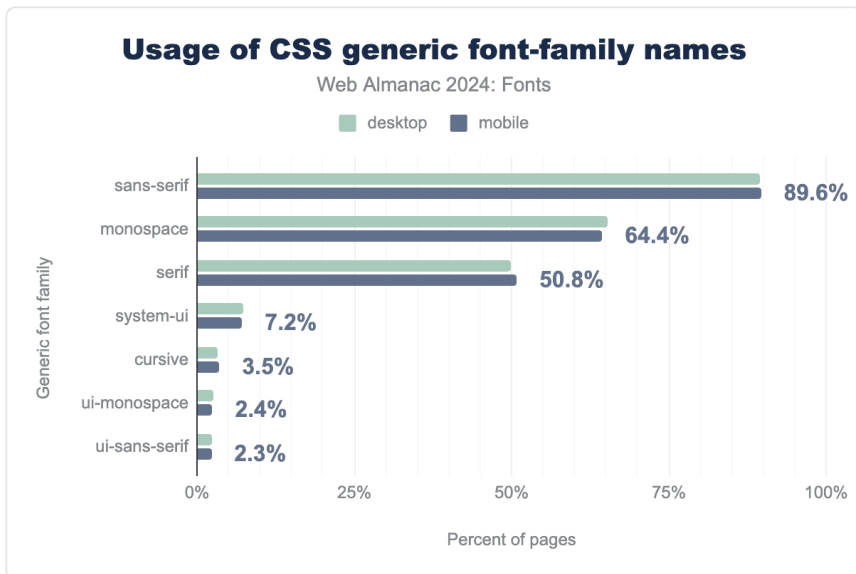


Figure 4.46. Usage of generic `font-family` names.

So, what are the most used generic family names on the web? Most used in 2024 were `sans-serif` at about 90%, `monospace` at 65%, and `serif` at 50%. These figures have been relatively consistent since 2022 for `sans-serif` and `monospace`, but mark a 7% decline for `serif` system fonts. While `system-ui` fonts were only represented on about 3.6% of

desktop and mobile websites, it is now at 7.2%, roughly double what was seen in 2022. This is a clear indication that more and more people are using it to build apps whose typography matches the system.

When it comes to system fonts, the greatest increase over the last two years registered for several generic families totaling less than 1% of cases (not pictured). Emoji system fonts, in particular, grew with something on the order of a thirteen-fold increase between 2022 and 2024. System fonts for math also showed a major increase despite a small overall footprint, growing from a little over 100 websites in the 2022 crawl to reach roughly 1,300 websites on desktop and mobile this year.

Conclusion

Where are things going in the world of web typography? This year's data highlights several promising trends in matters of performance. More developers are using the WOFF2 format, whose efficient compression offers better load times to handle font files that are growing in size. That's great, but we would like to see more self-hosted fonts in the TrueType, OpenType, and WOFF formats converted to WOFF2. In most cases, this is a simple process that will significantly reduce file sizes and thus load times. Another area where there's significant potential for performance gain is the use of resource hints. Currently only 11% of pages use resource hints to preload web fonts. We'd love to see that number go up significantly in the next couple of years as it requires very little effort (adding a simple `<link rel="preload" href="..." as="font">` to your pages) and has a huge impact on web font performance.

This year's web font data also leads to several key insights. The rise in self-hosting rates and declining use of web font services seen in the 2022 data has largely stabilized for now. In the future, it will be interesting to observe whether these rates continue to plateau, or else begin to shift again in either direction. A major factor in future hosting decisions will be privacy regulations surrounding the use of web font services. Other factors may involve a tradeoff between performance and convenience for self-hosted fonts versus services.

We also expect the use of new technologies like color fonts and variable fonts to continue rising in the coming years. As color fonts go, the COLR format is the clear winner due to its overall versatility, integration with variable fonts, and support for palettes in CSS. As for variable fonts, the trend toward designing for more axes, and implementing these in practice, is likely to continue proliferating as more designers and developers avail themselves of these technical affordances. Although variable fonts appear to have gained initial and impressive traction largely through CJK fonts, we expect the coming years to show increasing spread of variable fonts through other writing systems. Moreover, the continued development of new multi-script fonts is likely to continue increasing the typographic variety of global web design.

What does the future hold for (web) fonts? We expect many incremental technical advancements, such as the addition of cubic Beziér curves to `glyph` outlines and breaking the 65k glyph limit in fonts. These and other changes to fonts are outlined in the so-called “Boring Expansion” specification⁸⁰, whose aim it is to add support for various features to the OpenType font standard while mostly staying backwards compatible (hence the “boring” part). The recent development of Incremental Font Transfer (IFT)⁸¹ also promises considerable performance increases for web fonts, particularly ones with large character sets, as the user will only need to download the part of the font file that’s used on a particular website.

Overall, observing the state of web fonts in 2024, we’re excited to see more and more support for various writing systems, and an increasing adoption of variable and color font technology. While there is still a lot of low-hanging fruit when it comes to web font performance, it is clear that developers are adopting technologies like WOFF2, resource hints, and `font-display` to improve the performance of their sites. We hope to see these trends continue in 2025!

Authors



Bram Stein

@https://typo.social/@bram
 bramstein
 bramstein
 http://www.bramstein.com/

Bram Stein is CTO at The Type Founders⁸². Before that he was Head of Webfonts at Adobe Fonts⁸³. He is the author of the Webfont Handbook⁸⁴ and speaks about typography and web performance at conferences around the world.



Charles Berret

charlesberret
 https://charlesberret.net/

Charles Berret is a writer, interdisciplinary researcher, and data journalist who studies the history and philosophy of media and information technologies.

80. <https://github.com/harfbuzz/boring-expansion-spec>

81. <https://w3c.github.io/IFT/Overview.html>

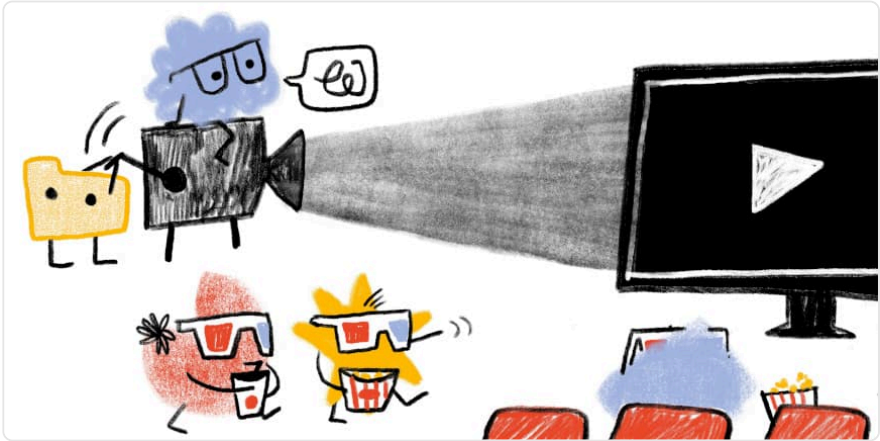
82. <https://thetypefounders.com>

83. <https://fonts.adobe.com>

84. <https://abookapart.com/products/webfont-handbook>

Part I Chapter 5

Media



Written by Stefan Judis and Eric Portis

Reviewed by Chris Lilley

Analyzed by Stefan Judis and Eric Portis

Edited by Michael Lewittes

Introduction

Images and videos are everywhere on the web. However, the ways they're encoded and embedded on web pages are surprisingly varied and complex, and best practices are always evolving. The Web Almanac gives us a chance to take stock of that complexity and how well we're managing it, giving us a zoomed-out, panoramic view of where media on the web is, how far it has come, and—just maybe—where it is going. So, let's go!

Images

We'll kick off with the most common media type—images. How often do you look at a web page without images? For us, it is extremely rare and if there are no images, we're most likely looking at a nerdy developer blog.

It comes at no surprise that of the more than 10 million scanned and parsed pages, 99.9%

requested at least one image.

99.9%

Figure 5.1. Pages that requested at least one image resource.

Almost every page serves up some kind of an image, even if it's just a background or favicon.

How many `<img>` elements did we find, per page?

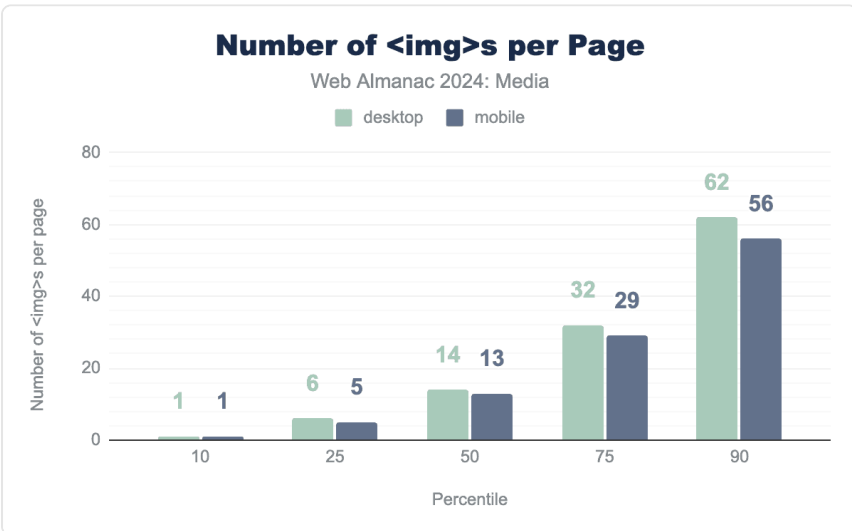


Figure 5.2. Count of `<img>` elements per page.

The median mobile page contains 13 `<img>` elements. And even at the 90th percentile, pages “only” contain 56 `<img>` s. Considering the visual nature of today’s web, this seems reasonable.

If you think that 56 `<img>` s per page is a lot, we probably shouldn’t tell you that the mobile crawler found a page with more than two thousand `<img>` elements.

2,174

Figure 5.3. Most `<img>` elements on a single page (mobile).

Images aren't just pervasive and plentiful. Most of the time they are also a central part of users' experiences. One way to measure that is to see how often images are responsible for pages' Largest Contentful Paint.

68%

Figure 5.4. Mobile pages whose LCP responsible element has an image.

It's hard to overstate the importance of images on the web. So, let's find out what we're dealing with!

Image resources

We'll start with the resources themselves. Most images are made of pixels (let's ignore vector images for a moment). How many pixels do the web's images typically have?

Perhaps surprisingly, many images contain just a single pixel!

Single-pixel images

Client	1×1 images
Mobile	6.4%
Desktop	6.0%

Figure 5.5. Resources loaded by `<img>` elements that contain just a single pixel.

One-by-one pixel images make up roughly 6% of all of the captured image requests. These are most likely tracking beacons and spacer GIFs as discovered in last year's Media chapter⁸⁵. And looking back, we're happy to report some good news: the percentage of single-pixel images has declined a full point since 2022. So maybe old habits are slowly being replaced with newer and better alternatives⁸⁶.

Image dimensions

Let's now turn to images that were larger than 1×1. How big were they?

85. <https://almanac.httparchive.org/en/2022/media#a-note-on-single-pixel-images>

86. https://developer.mozilla.org/docs/Web/API/Beacon_API

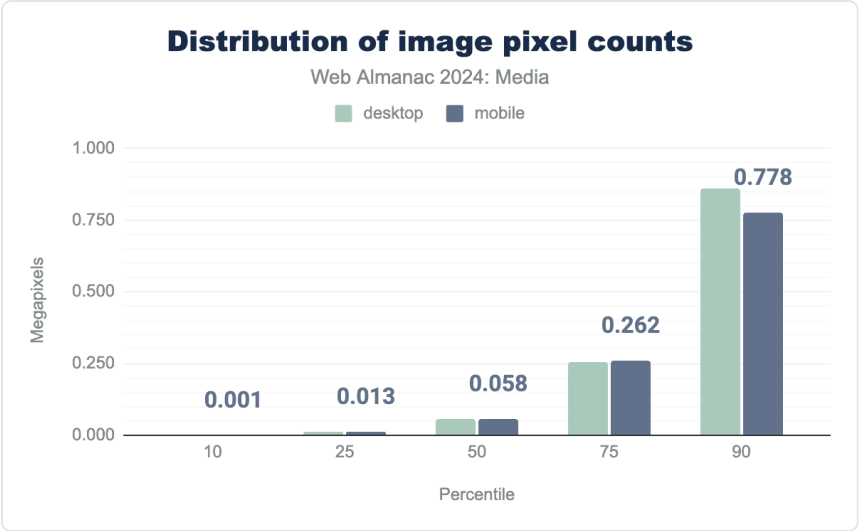


Figure 5.6. Distribution of image pixel counts.

Despite the fact that the Web Almanac’s mobile crawler hasn’t grown at all (rendering pages to a 360px-wide viewport, at a device pixel ratio of 3x), the median image—weighing in at 0.058 megapixels—is about 25% larger than it was the last time we looked⁸⁷. For reference, at a square aspect ratio, 0.058 megapixels works out to about 240×240.

Most pages have one image that has almost 10 times as many pixels as the median image:

87. <https://almanac.httparchive.org/en/2022/media#image-dimensions>

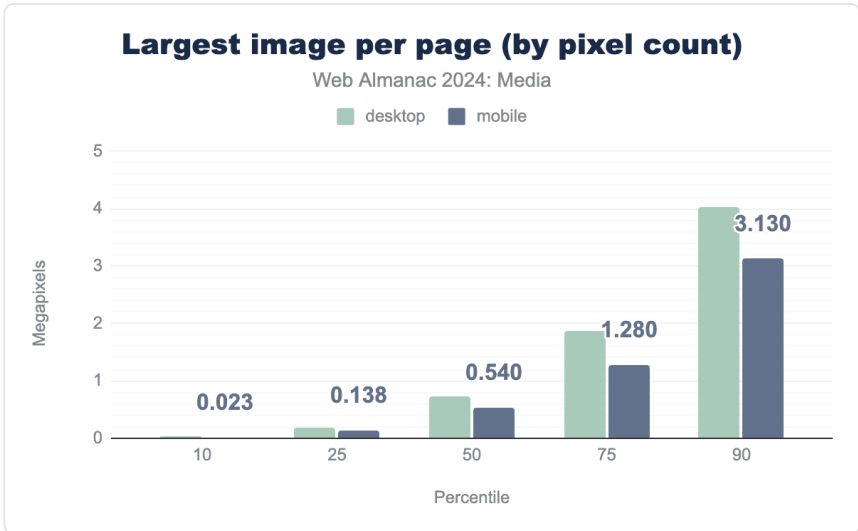


Figure 5.7. Largest image per page (by pixel count).

At a square aspect ratio, 0.54 megapixels works out to 735×735. Given the mobile crawler's viewport and density, it is quite likely that many pages have one “hero” image that is being displayed full-width at high density.

As for the 50% of pages that sent images even larger than that, they are almost certainly sending the mobile crawler more pixels than it can actually display, and could have prevented that waste with some well-written responsive image markup. But more on that later.

Image aspect ratios

Now that we have some sense of how images on the web are sized, how are they shaped?

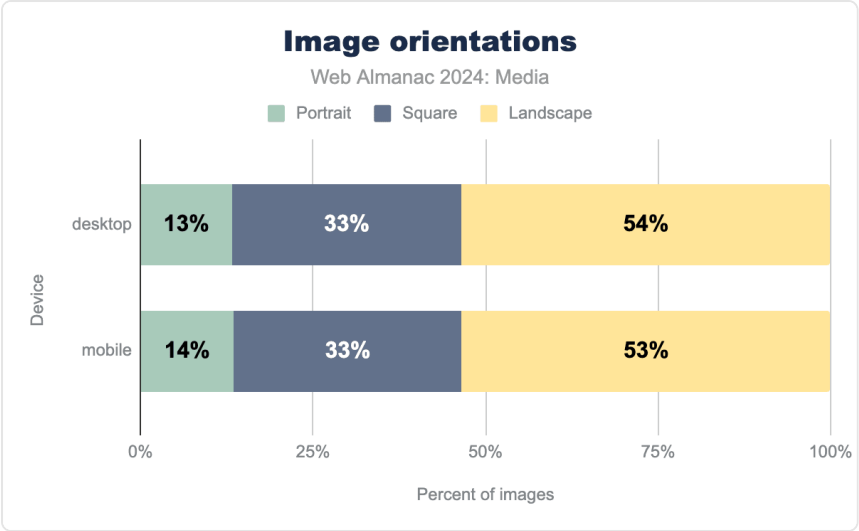


Figure 5.8. Image orientations.

Most images are wider than they are tall—only 1 in 8 are taller than they are wide and a full one-third are exactly square. Square is by far the most popular exact aspect ratio:

Aspect ratio (width / height)	% of images
1:1	33.2%
4:3	3.5%
3:2	2.9%
2:1	1.8%
16:9	1.6%
3:4	1.1%
2:3	0.8%
5:3	0.5%

Figure 5.9. Top image aspect ratios (mobile).

This data is essentially unchanged from two years ago. It still seems to indicate a bias towards desktop-based browsing—creators are missing opportunities to fill portrait-oriented mobile screens with big, beautiful, portrait-oriented imagery.

Image color spaces

The range of colors that are possible within a given image is determined by that image’s color space⁸⁸. The default color space on the web is sRGB⁸⁹. Unless images signal that their color data uses a different color space, browsers will use sRGB⁹⁰.

The traditional way to explicitly assign a color space to an image is to embed an ICC profile⁹¹ within it. We looked at all of the ICC profiles embedded in all of the images crawled in the dataset.

Here are the top ten:

ICC profile description	sRGB-ish	Wide-gamut	% of images
No ICC profile	✓		87.7%
sRGB IEC61966-2.1	✓		3.8%
c2ci	✓		3.2%
sRGB	✓		1.6%
uRGB	✓		0.9%
Adobe RGB (1998)		✓	0.7%
Display P3		✓	0.4%
c2	✓		0.3%
GIMP built-in sRGB	✓		0.3%
Display			0.3%

Figure 5.10. Top ICC profiles (mobile).

The vast majority of the web’s images rely on the sRGB default for correct rendering and don’t contain any ICC profile at all.

The most common ICC profile is the full, official sRGB color profile. This profile is relatively heavy—it weighs 3 KB. Thus, most of the rest of the top 10 ICC profiles are “sRGB-ish” profiles like Clinton Ingrahm’s 424-byte c2ci⁹², which unambiguously specify that an image uses sRGB but with a minimum of overhead.

88. https://wikipedia.org/wiki/Color_space
89. <https://wikipedia.org/wiki/sRGB>
90. <https://imageoptim.com/color-profiles.html>
91. https://wikipedia.org/wiki/ICC_profile
92. <https://photosauce.net/blog/post/making-a-minimal-srgb-icc-profile-part-1-trim-the-fat-abuse-the-spec>

Over the last decade, hardware and software are increasingly able to capture and present colors that are outside of the range of colors that are possible with sRGB (aka the sRGB gamut⁹³). Adobe RGB (1998) and Display P3 are the only two “wide-gamut” profiles in the top 10. While Adobe RGB (1998)’s usage ticked down slightly from 2022, Display P3’s has ticked up⁹⁴, and total wide-gamut ICC profile adoption is up about 10% in relative terms⁹⁵. In absolute terms, wide-gamut ICC profiles are still relatively rare. We found them in 1 in 80 images on the web and 1 in 10 images that have an ICC profile.

One very important caveat here is that in our analysis we were only able to look at ICC profiles. As mentioned, these profiles can be relatively heavy. Modern image formats like AVIF (and recently modernized ones like PNG⁹⁶) allow images to signal their color space much more efficiently using a standard called CICC—which allows common color spaces to be signaled in just four bytes⁹⁷. It stands to reason that modern PNG encoders and any AVIF encoder worth its salt would use CICC instead of ICC to signal a wide gamut color space.

However, in our analysis, images containing CICC are categorized under “No ICC profile.” So, our accounting of wide-gamut usage on the web should be seen as a floor, rather than as an estimate of total adoption. In other words, we found that at least 1 in 80 images on the web is wide-gamut.

Encoding

Now that we’ve gleaned a bit about the web’s image content, what can we say about how that content is encoded for delivery?

Format adoption

For decades there were just three bitmap formats in common use on the web: JPG, PNG, and GIF. They are still the three most common formats:

93. <https://wikipedia.org/wiki/Gamut>

94. <https://almanac.httparchive.org/en/2022/media#fig-8>

95. <https://docs.google.com/spreadsheets/d/1QZ1TOe6ZMIXGKHtXqK9XmUA1eQBX9CLQkxarQJFck/edit?gid=644447618#gid=644447618>

96. https://github.com/w3c/png/blob/main/Third_Edition_Explainer.md#labelling-hdr-content

97. <https://www.w3.org/TR/png-3/#cicc-chunk>

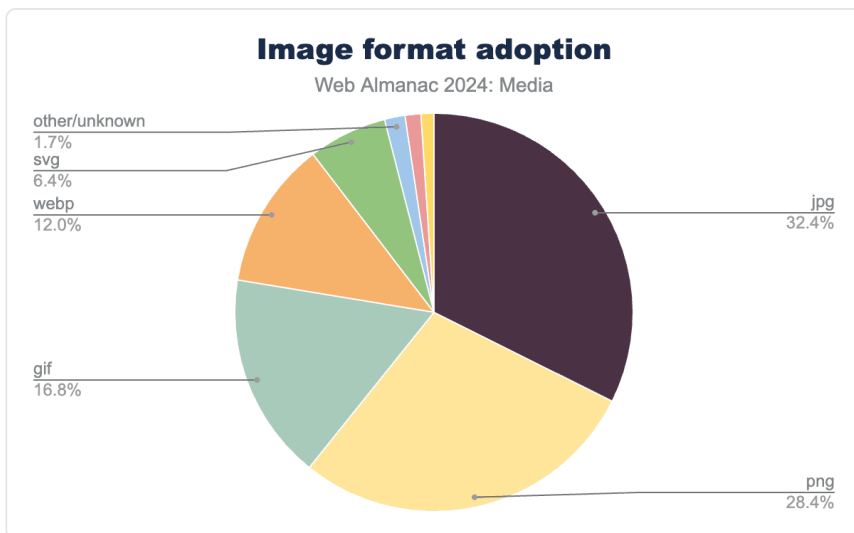


Figure 5.11. Image format adoption (mobile).

But we are happy to report that change is happening. The largest single absolute change in usage since 2022 was from JPEG, which fell from 40% of all images in 2022 down eight full percentage points to 32% in 2024. That’s a huge loss over two years.

Which formats saw more usage to make up the difference? WebP picked up three percentage points, SVG picked up a bit under two percentage points, and AVIF picked up almost a full point. Most surprisingly, the oldest and least efficient format of them all, GIF, picked up a percentage point, too.

And in relative terms, AVIF usage is taking off—we found almost four times more AVIFs served up by the crawled pages than we did two years ago.

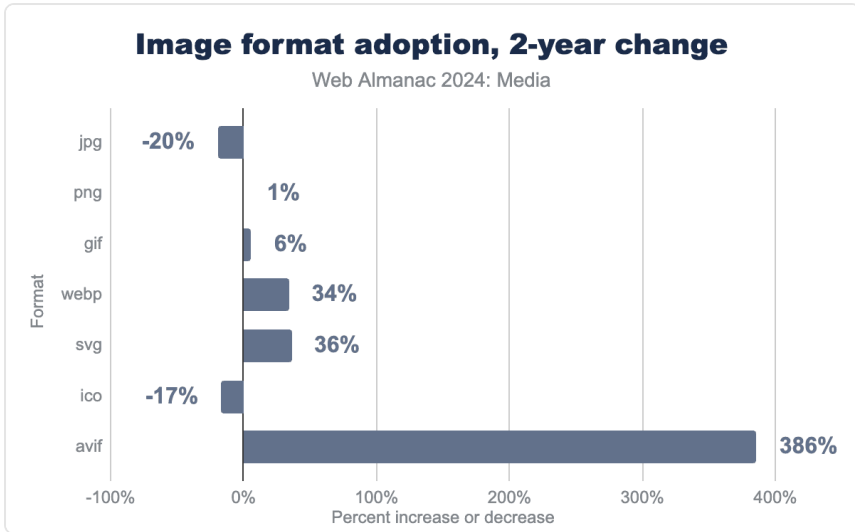


Figure 5.12. Image format adoption, 2-year change (mobile).

If the crawler had accepted JPEG XL⁹⁸s, we probably would have seen a fair number of them as well. Alas, Chromium-based browsers don't support the format⁹⁹.

Almost all of the JPEGs, PNGs, and GIFs on the web would be better-served using a modern format. WebPs are good, but AVIFs and JPEG XLs are even better. It is nice to see the massive ship that is all-of-the-images-on-the-web slowly but surely turning towards these more efficient formats. And it's nice to see SVG usage tick upwards, too!

Lastly, a few words for the oldest format of the bunch: “Burn All GIFs¹⁰⁰” was good advice in 1999, and it is even better advice today. Developers should take Tyler Sticka’s advice¹⁰¹ about how to replace the 37-year-old format.

Byte sizes

How heavy is the typical image on the web?

98. <https://jpeg.org/jpegxl/>

99. <https://caniuse.com/jpegxl>

100. <https://www.theatlantic.com/past/docs/unbound/citation/wc991103.htm>

101. <https://cloudfour.com/thinks/video-gifs-are-forever-lets-make-them-better/>

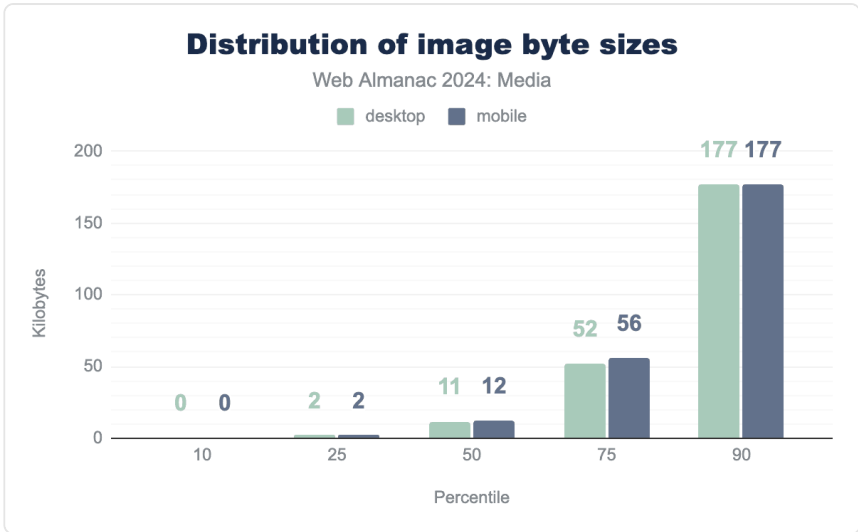


Figure 5.13. Distribution of image byte sizes.

A median of 12 KB might lead one to think, “Eh, that’s not that heavy!” But, just as when we looked at pixel counts, we found that most pages contain many small images, and at least one large one.

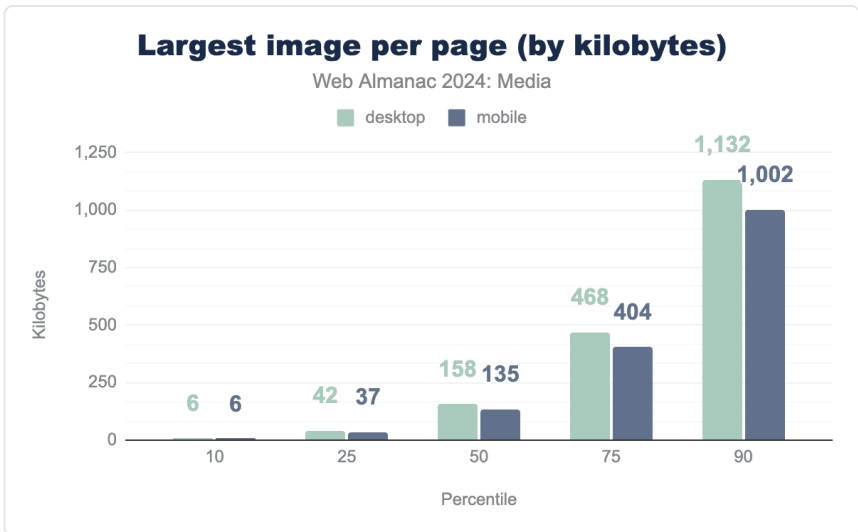


Figure 5.14. Largest image per page (by kilobytes).

Most mobile pages have one image that's 135 KB or more. That's an 8% increase since 2022. And the further up we go in the distribution, the more things are accelerating: the 75th percentile is up 10% and the 90th percentile is up 13% (to almost exactly a megabyte).

Images are getting heavier, and the heaviest images are getting heavier faster.

Bits per pixel

Bytes and pixel counts are interesting on their own, but to get a sense of how compressed the web's image data is we need to put bytes and pixels together to calculate bits per pixel. Doing this allows us to make apples-to-apples comparisons of the information density of images, even if those images have different resolutions.

In general, bitmaps on the web decode to eight bits of uncompressed information per channel (per pixel). So, if we have an RGB image with no transparency, we can expect a decoded, uncompressed image to weigh in at 24 bits per pixel.

A good rule of thumb for lossless compression is that it should reduce file sizes by a 2:1 ratio (which would work out to 12 bits per pixel for our 8-bit RGB image). The rule of thumb for 1990s-era lossy compression schemes—JPEG and MP3—was a 10:1 ratio (2.4 bits per pixel).

It should be noted that, depending on image content and encoding settings, these ratios vary widely and modern JPEG encoders like MozJPEG¹⁰² and Jpegli¹⁰³ typically outperform this 10:1 target at their default settings.

To summarize:

Type of bitmap data	Expected compression ratio	Bits per pixel
Uncompressed RGB	1:1	24 bits/pixel
Losslessly compressed RGB	~2:1	12 bits/pixel
1990s-era lossy RGB	~10:1	2.4 bits/pixel

Figure 5.15. Typical compression ratios and resulting bits per pixel numbers for bitmap RGB data.

So, with all of that as context, here's how the web's images stack up:

102. <https://github.com/mozilla/mozjpeg>

103. <https://opensource.googleblog.com/2024/04/introducing-jpegli-new-jpeg-coding-library.html>

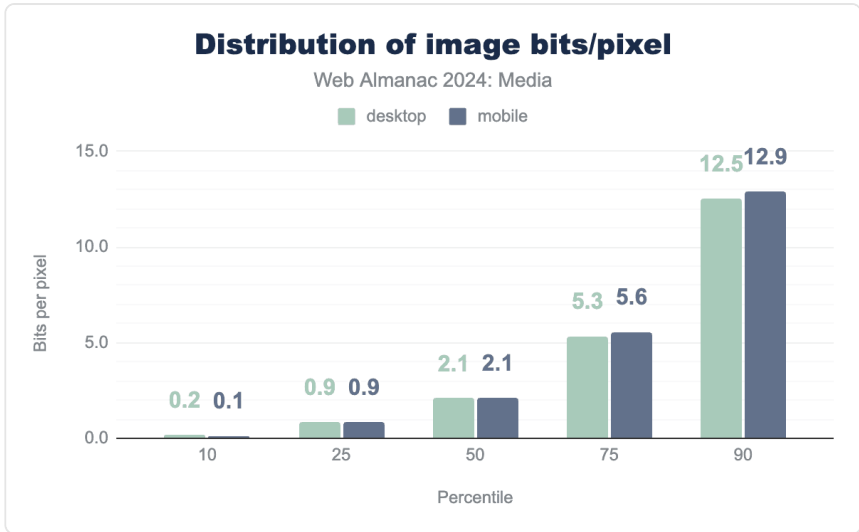


Figure 5.16. Distribution of image bits per pixel.

The median image is compressed to 2.1 bits per pixel, representing a tad more compression than that 1990s rule of thumb. This is also 8–10% more compression than we saw when we last surveyed the web’s images in 2022¹⁰⁴.

When we break compression down by format, we can see that every format saw fewer bits spent per pixel in 2024 than they did in 2022—except for one.

104. <https://almanac.httparchive.org/en/2022/media#fig-14>

Bits per pixel by format

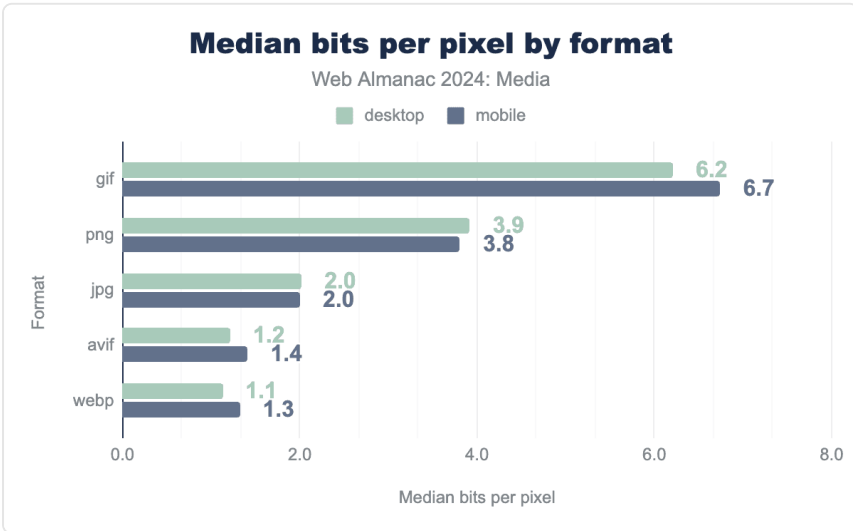


Figure 5.17. Median bits per pixel by format.

Compared to 2022¹⁰⁵, on mobile, the median PNG is compressed about 10% more, the median WebP is compressed about 7% more, and the median JPEG is compressed around 3% more. It is hard to know exactly what the causes are here, but we hypothesize that an increase in compression is the result of wider adoption of two things: modern encoders, which provide more bang for the buck, and automated image-processing pipelines, which ensure that every image that makes its way to a user has been well-compressed.

The one format that bucked this trend was AVIF. The median AVIF’s bits-per-pixel went up from approximately 1.0 in 2022 to around 1.4 bits per pixel in 2024—an increase of 47%. Funnily enough, we hypothesize the same root cause. The current, diverse crop of AVIF encoders is likely making different quality/filesize tradeoffs, sacrificing less quality at default settings than AOM’s official libavif encoder was two years ago.

We have no idea why GIFs got significantly more efficient, but we do know why they are so much less-compressed than all of the other formats. Our query is per pixel, and it does not take animated images into account, though many GIFs are animated!

105. <https://almanac.httparchive.org/en/2022/media#fig-15>

GIFs, animated and not

How many GIFs are animated?

32%

Figure 5.18. Percentage of GIFs that were animated on mobile.

When we separate the animated GIFs out from the animated ones, we can see that the median non-animated GIF is much more reasonably compressed:

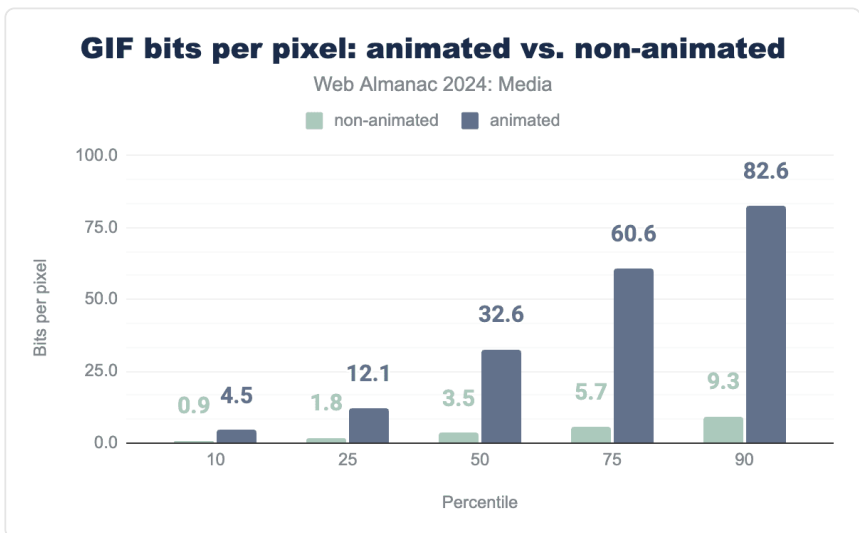


Figure 5.19. GIF bits per pixel: animated vs. non-animated.

3.5 bits per pixel is even less than the median PNG!

Turning to look at the animated GIFs specifically: How many frames do they have?

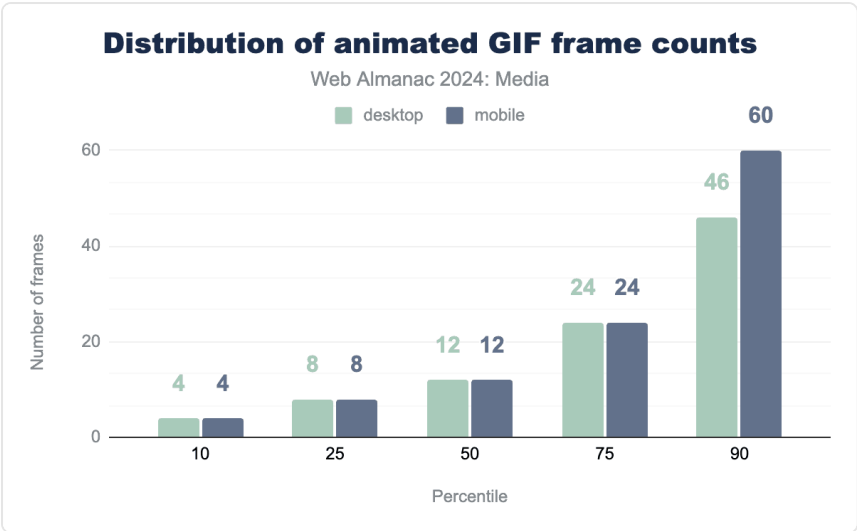


Figure 5.20. Distribution of animated GIF frame counts.

Generally: 10 to 20, which is more or less unchanged since 2022¹⁰⁶, although the longest GIFs have gotten longer, especially on mobile.

Just for fun, we also looked at the GIF with the most frames:

54,028

Figure 5.21. The highest GIF frame count in the data set.

At 24-frames-per-second, that would take more than 37 minutes to play once through. Every animated GIF should probably be a video these days, but this one definitely should.

Embedding

Now that we have a sense of how the web's image resources have been encoded, what can we say about how they are embedded on websites?

106. <https://almanac.httparchive.org/en/2022/media#fig-18>

Lazy-loading

The biggest recent change in how images are embedded on websites has been the rapid adoption of lazy-loading. Lazy-loading was introduced in 2020, and just two years later it was adopted on almost a quarter of websites¹⁰⁷. Its climb continues, and it is now used on a full one-third of all websites:

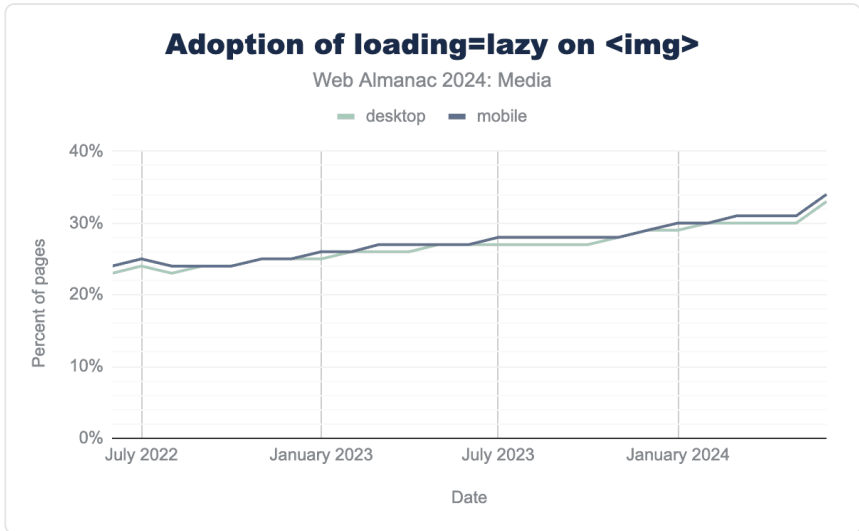


Figure 5.22. Adoption of `<img loading=lazy>`.

And, just like last year, it seems pages are using lazy-loading a bit too much:

9.5%

Figure 5.23. Percentage of LCP-responsible `<img>` elements that use native lazy-loading on mobile.

Lazy-loading the LCP element is an anti-pattern that makes pages much slower¹⁰⁸. While it is disheartening that nearly one-in-ten LCP-responsible `<img>`s are lazy-loaded, we are happy to report that things have improved ever-so-slightly over the last two years. The percentage of offending sites has decreased by 0.3 percentage points since 2022.

107. <https://almanac.httparchive.org/en/2022/media#lazy-loading>

108. <https://web.dev/articles/lcp-lazy-loading>

alt text

Images embedded with `<img>` elements are supposed to be contentful. That is to say: They're not just decorative¹⁰⁹, and they should contain something meaningful. According to both WCAG requirements¹¹⁰ and the HTML spec¹¹¹, most of the time, `<img>` elements should have alternative text, and that alternative text should be supplied by the `alt` attribute.

55%

Figure 5.24. Percentage of images that had a non-blank `alt` attribute.

Unfortunately, 45 percent of `<img>` elements don't have any `alt` text. Worse, the in-depth analysis from this year's accessibility chapter indicates that many of the `<img>`s that do have `alt` text aren't all that accessible either since their attributes only contain filenames or other meaningless, short strings.

There has been a one percentage point increase in `alt` text deployment since 2022, but still we can—and must—do better.

srcset

Prior to lazy-loading, the biggest thing to happen to `<img>` elements on the web was a suite of features for “responsive images,” which allowed images to tailor themselves to fit within responsive designs. First shipped in 2014, the `srcset` attribute, the `sizes` attribute, and the `<picture>` element are now a decade old. How often and how well are we using them?

Let's start by looking at the `srcset` attribute, which allows authors to give the browser a menu of resources to choose from, depending on the context.

42%

Figure 5.25. Percentage of pages using the `srcset` attribute on mobile.

The last time we checked, this number was 34%¹¹²—an eight percentage point increase over two

109. <https://html.spec.whatwg.org/multipage/images.html#a-purely-decorative-image-that-doesn't-add-any-information>

110. <https://www.w3.org/WAI/WCAG22/Understanding/non-text-content>

111. <https://html.spec.whatwg.org/multipage/images.html#alt>

112. <https://olmanac.httparchive.org/en/2022/media/srcset>

years is significant and encouraging.

The `srcset` attribute allows authors to describe resources using one of two descriptors. `x` descriptors specify the resource's density, allowing browsers to select different resources depending on users' screen densities. `w` descriptors give the browser the resource's width in pixels. When used in conjunction with the `sizes` attribute, `w` descriptors allow browsers to select a resource appropriate for both variable layout widths and variable screen densities.

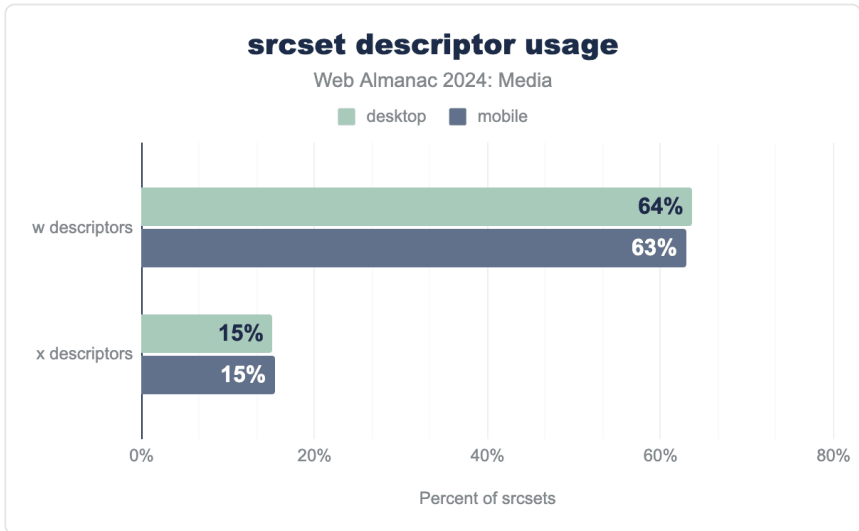


Figure 5.26. `srcset` descriptor usage.

`x` descriptors shipped first and are simpler to reason about, but `w` descriptors are more powerful. It is encouraging to see that `w` descriptors are more common. And while `x` descriptor adoption hasn't increased much since 2022, `w` descriptor usage is still growing—`w` descriptor adoption is up four percentage points on mobile and six percentage points on desktop.

sizes

We mentioned earlier that `w` descriptors should be used in conjunction with `sizes` attributes. So, how well are we using `sizes`? Not very well!

The `sizes` attribute is supposed to be a hint to the browser about the eventual layout width of the image, usually relative to the viewport width. The `sizes` attribute is explicitly supposed to be a hint, and so a little inaccuracy is OK and even expected.

But if the `sizes` attribute is more-than-a-little inaccurate, it can affect resource selection, causing the browser to load an image to fit the `sizes` width when the actual layout width of the image is significantly different.

So, how accurate are our `sizes`?

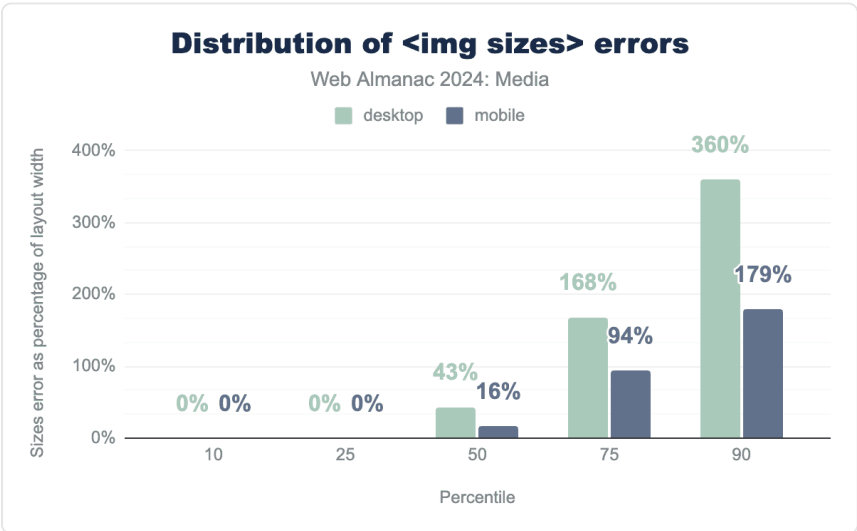


Figure 5.27. Distribution of `<img sizes>` errors.

While many `sizes` attributes are entirely accurate, the median `sizes` attribute is 16% too large on mobile and 43% too large on desktop. That might be OK, given the hint-like nature of the feature, but as you can see, the 75th and 90th percentiles aren't pretty. Most worryingly, all of these numbers have gotten significantly worse over the past two years¹¹³—the median desktop sizes is more than twice as inaccurate as it was two years ago.

What's the impact of all of this inaccuracy?

20%

Figure 5.28. `sizes` attributes that were inaccurate enough to affect `srcset` selection on desktop. On mobile, it is 14%.

On desktop, where the difference between the default `sizes` value (`100vw`) and the actual

113. <https://almanac.httparchive.org/en/2022/media#fig-24>

layout width of the image is likely to be larger than on mobile, 1 in 5 `sizes` attributes is inaccurate enough to cause browsers to pick a suboptimal resource from the `srcset`. These errors add up.

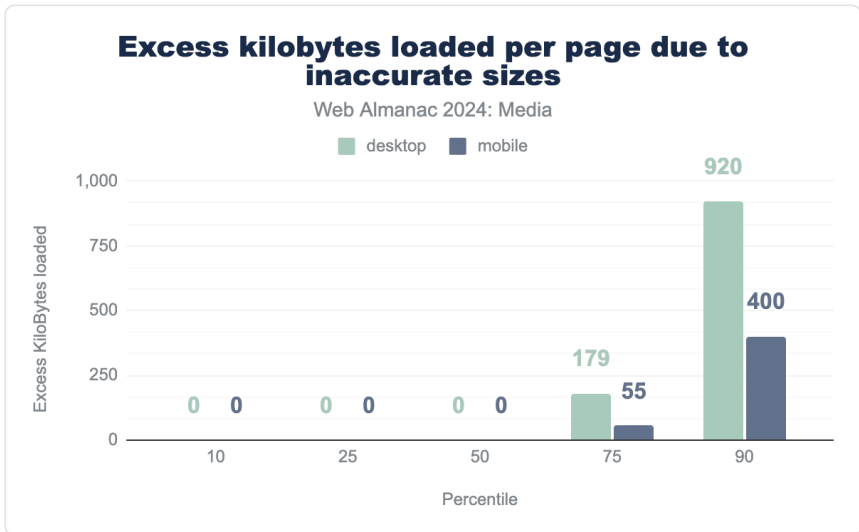


Figure 5.29. Excess kilobytes loaded per page due to inaccurate `sizes`.

We estimate that a quarter of all desktop pages that use `w` descriptors are loading 180 KB or more of wasted image data, because of their inaccurate `sizes` attributes. That is to say, a better, smaller resource is there for the picking in the `srcset`, but because the `sizes` attribute is so erroneous, the browser doesn't pick it. The worst 10% of desktop pages that use `w` descriptors load close to a megabyte of excess image data because of bad `sizes` attributes.

This is quite troubling, but what's worse is that all of these numbers are almost twice as bad as they were just two years ago. Things are bad and getting worse.

Note: Our crawlers didn't actually load the correct resources, so the numbers here are estimates, based on the compression densities and aspect ratios of the incorrect resources, which the crawlers actually did load.

There are two solutions here that developers should pursue.

For LCP-responsible and other critical images, developers need to fix their `sizes` attributes. The best tool to audit and repair `sizes` is RespImageLint¹¹⁴, which can help fix a host of other responsive image problems, too.

114. <https://ausi.github.io/respimagelint/>

For below-the-fold and non-critical images, authors should start to adopt `sizes="auto"`. This value can only be used in conjunction with lazy-loading, but it tells the browser to use the actual layout size of the `<img>` as the `sizes` value, ensuring that the used value is perfectly accurate.

Auto-`sizes` for lazy-loaded images is currently only implemented in Chrome, but Safari and Firefox have both expressed support for it. We hope they implement it soon and that developers start rolling it out now (with fallback values).

`<picture>`

The last responsive image feature to land in 2014 was the `<picture>` element. While `srcset` hands browsers a menu of resources to choose from, the `<picture>` element allows authors to take charge, giving browsers an explicit set of context-adaptive instructions about which child `<source>` element to load a resource from.

The `<picture>` element is used far less than `srcset`:

9.3%

Figure 5.30. Percentage of mobile pages that use the `<picture>` element.

This is up more than a percentage point and a half from 2022, but the fact that there are more than four pages that use `srcset` for every one page that uses `<picture>` suggests that either `<picture>` use cases are more niche or that `<picture>` is more difficult to deploy—or both.

What are people using `<picture>` for?

The `<picture>` element gives authors two ways to switch between resources. Type-switching allows authors to provide cutting-edge image formats to browsers that support them and fallback formats for everyone else. Media-switching facilitates art direction¹¹⁵, allowing authors to switch between `<source>`s based on media conditions¹¹⁶.

115. <https://www.w3.org/TR/resping-usecases/#art-direction>

116. <https://www.w3.org/TR/mediaqueries-5/#media-condition>

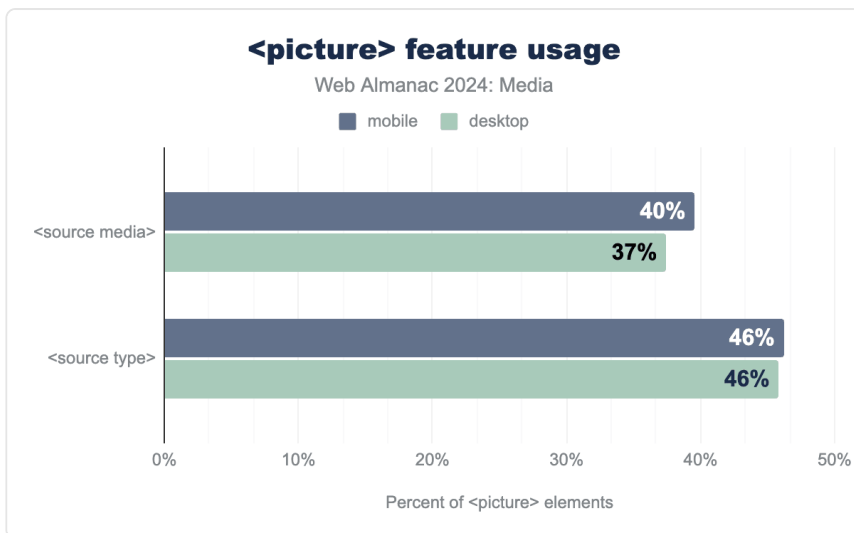


Figure 5.31. `<picture>` feature usage.

While usage of the `media` attribute is down three percentage points from 2022, `type`-switching usage is up three percentage points. This increase is likely related to the increasing popularity of next-generation image formats, especially JPEG XL which does not yet enjoy universal browser support.

Layout

We already saw how the web's image resources size up. But before they can be shown to a user, embedded images must be placed within a layout and potentially squished or stretched to fit it.

Note: It will be useful to keep in mind the crawlers' viewports throughout this analysis. The desktop crawler was 1376px wide, with a DPR of 1x, the mobile crawler was 360px wide, with a DPR of 3x.

Layout widths

Let's start by asking: How wide do the web's images end up when painted to the page?

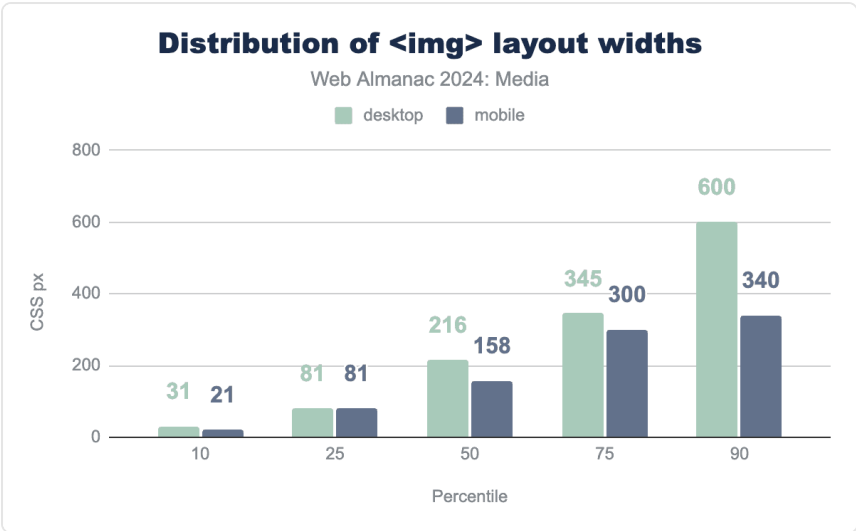


Figure 5.32. Distribution of `<img>` layout widths.

Most of the web's images end up pretty small within layouts. Interestingly, while most of the mobile layout sizes are essentially unchanged since 2022, the top half of desktop layout sizes have all increased by around 8%¹¹⁷.

But while the majority of layout sizes are small, most pages have at least one fairly large `<img>`.

117. <https://almanac.httparchive.org/en/2022/media#sizes>

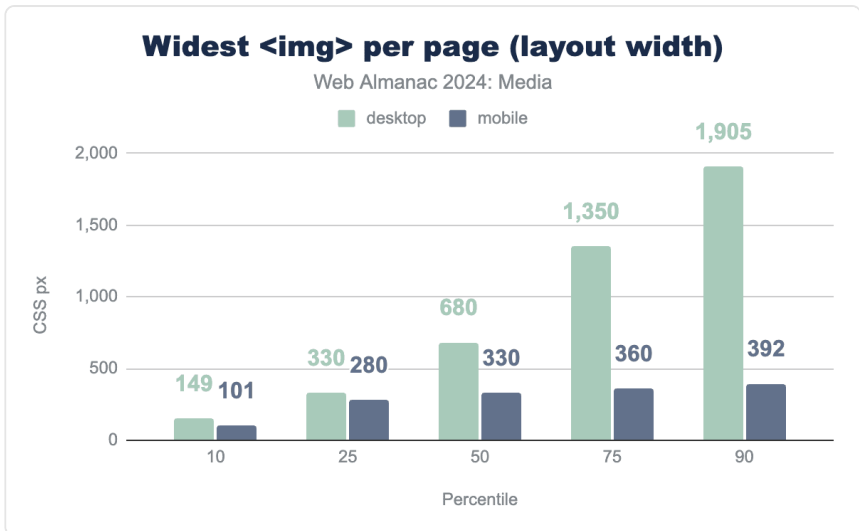


Figure 5.33. Widest `<img>` per page (layout width).

Half of all mobile pages have at least one image that takes up approximately the full viewport. At the top end, mobile layouts are doing a good job of containing images so that they don't take up much more than that. You can see the distribution quickly approach the mobile crawlers' viewport width (360px) and then only barely exceed it.

Contrast this with the desktop layout widths, which don't top out at all. They just keep growing, hitting full-viewport-width (1360px) at the 75th percentile and blowing right past it at the 90th percentile. Equally interesting, the 25th, 50th and 75th percentile layout sizes on desktop have gotten larger than they were two years ago¹¹⁸, while the ends of the distribution are essentially unchanged. Large hero images are getting larger.

Intrinsic vs extrinsic sizing

How do the web's images end up at these layout sizes? There are many ways to scale an image with CSS. But how many images are being scaled with any CSS at all?

Images, like all “replaced elements,”¹¹⁹ have an intrinsic size¹²⁰. By default—in the absence of a `srcset` controlling their density or any CSS rules controlling their layout width—images on the web display at a density of 1x. Plop a 640×480 image into an `<img src>` and, by default, that `<img>` will be laid out with a width of 640 CSS pixels.

118. <https://almanac.httparchive.org/en/2022/media#fig-30>

119. https://developer.mozilla.org/docs/Web/CSS/Replaced_element

120. https://developer.mozilla.org/docs/Glossary/Intrinsic_Size

Authors may apply extrinsic sizing to an image's height, width, or both. If an image has been extrinsically sized in one dimension (for instance, with a `width: 100%;` rule), but left to its intrinsic size in the other (`height: auto;` or no rule at all), it will scale proportionally, using its intrinsic aspect ratio.

Complicating things further, some CSS rules size `<img>` elements based on their intrinsic dimensions, unless those intrinsic dimensions violate some constraint. For instance, an `<img>` element with a `max-width: 100%;` rule will be intrinsically sized, unless that intrinsic size is larger than the size of the `<img>` element's container, in which case it will be extrinsically scaled down to fit.

With all of that explanation out of the way, here's how the web's `<img>` elements are sized for layout:

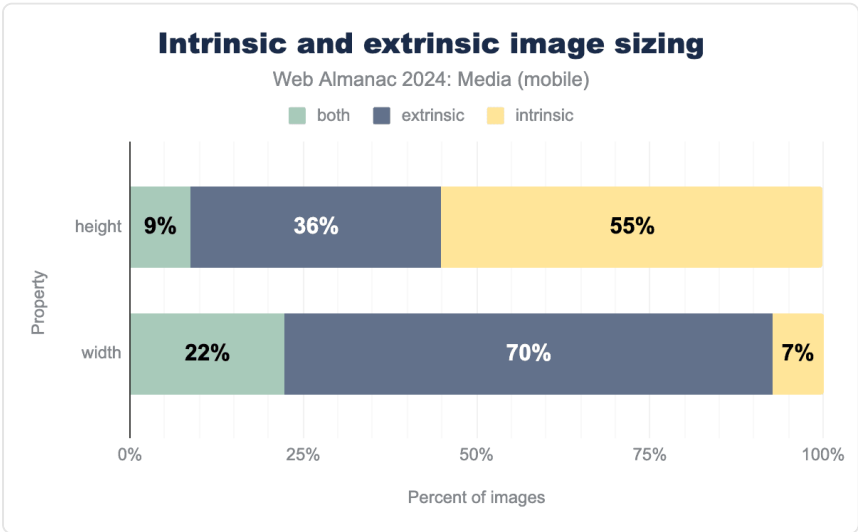


Figure 5.34. Intrinsic and extrinsic image sizing (mobile).

The majority of images have extrinsic widths and intrinsic heights. The “both” category for width—representing images with either a `max-width` or `min-width` sizing constraint—is also fairly popular. Leaving images to their intrinsic widths is far less popular and slightly less popular than it was in 2022¹²¹.

121. <https://almanac.httparchive.org/en/2022/media#fig-31>

height, width and Cumulative Layout Shifts

Any `<img>` whose layout size is dependent on its intrinsic dimensions risks triggering a Cumulative Layout Shift (CLS)¹²². In essence, such images risk being laid out twice—once when the page’s DOM and CSS have been processed, and then a second time when they finally finish loading and their intrinsic dimensions are known.

As we’ve just seen, extrinsically scaling an image to fit a certain width while leaving the height (and aspect ratio) intrinsic is very common. To prevent the resulting plague of layout shifts, authors should set the `width` and `height` attributes on the `<img>` element so that browsers can reserve layout space before the embedded resource loads.



Figure 5.35. Percentage of `<img>` elements on mobile that have both `height` and `width` attributes set.

Usage of `height` and `width` is up four percentage points from 2022¹²³, which is good. But the attributes are still only used on a one-third of images, meaning we have a long way to go.

Delivery

Finally, let’s take a look at how images are delivered over the network.

Cross-domain image hosts

How many images are being delivered from a different domain than the document they’re embedded on? A growing majority:

122. <https://web.dev/articles/cls>

123. <https://almanac.httparchive.org/en/2022/media#fig-32>

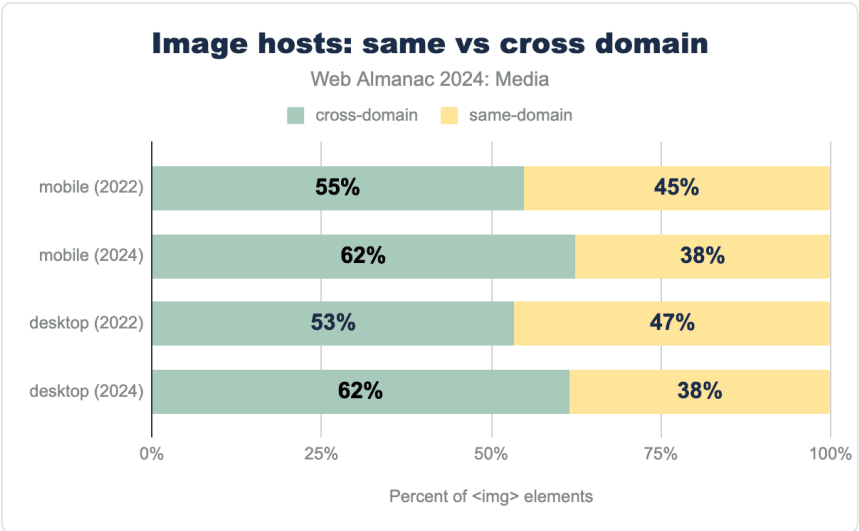


Figure 5.36. Image hosts: same vs cross domain.

It is hard to disentangle the various potential causes here, but we hypothesize that one factor is just how hard images are to get right. This leads teams to adopt image CDNs¹²⁴, which provide image optimization and delivery as a service.

So, there you have it: a panoramic view of the current state of images on the web. Now let’s take a look at video on the web in 2024.

Video

The `<video>` element shipped in 2010, and has been the best and—since the demise of plugins like Flash and Silverlight—only way to embed video content on websites ever since. How are we using it?

`<video>` element adoption

Let’s start by answering the first and most basic question: How many pages include `<video>` elements at all?

124. <https://web.dev/image-cdns/>

6.7%

Figure 5.37. Percentage of mobile pages that include at least one `<video>` element. On desktop, it's 7.7%.

This is a small fraction of the pages that include `<img>` s. But even though `<video>` was introduced 14 years ago, adoption is currently growing fast. The mobile number is up 32% (in relative terms) from 2022¹²⁵.

Video durations

How long are those videos? Not very long!

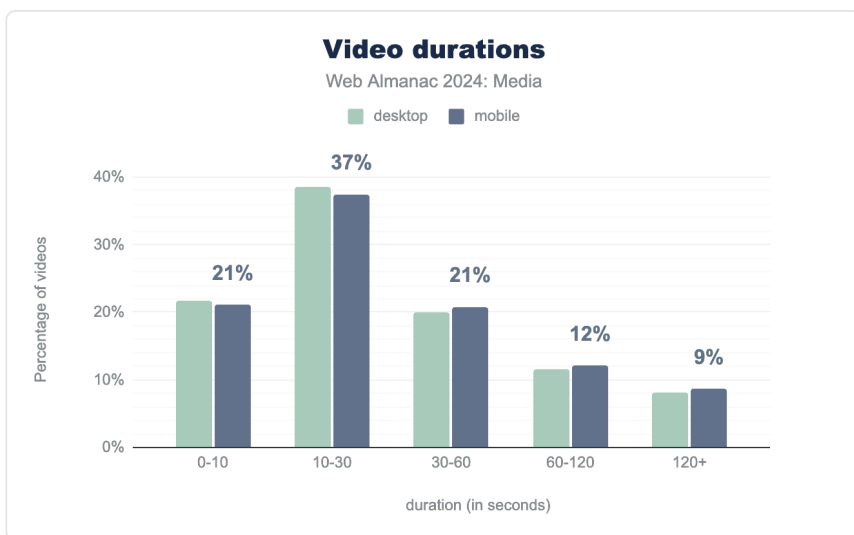


Figure 5.38. Video durations.

Nine out of ten videos are less than two minutes long. More than half are under 30 seconds. And almost one-quarter of videos are under ten seconds.

125. <https://almanac.httparchive.org/en/2022/media#fig-34>

Format adoption

What formats are sites delivering in 2024? MP4, which enjoys universal support¹²⁶, is king:

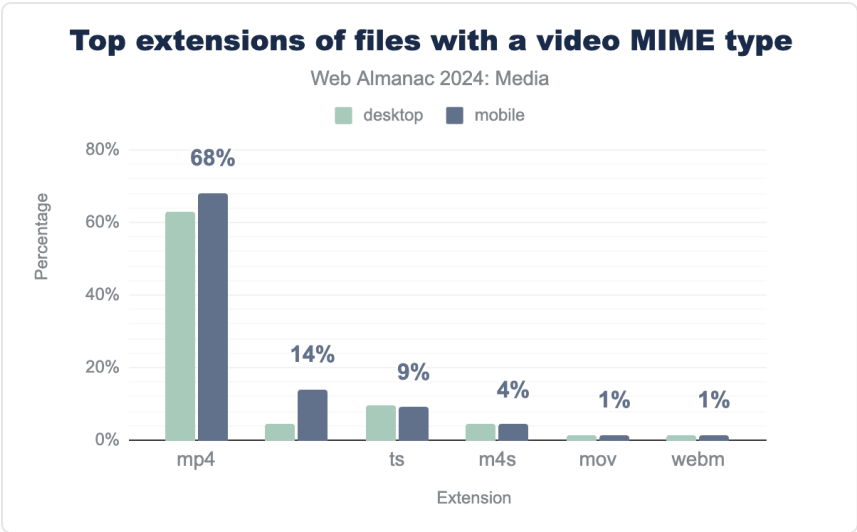


Figure 5.39. Top extensions of files with a video MIME type.

After `.mp4`, the three most common extensions are no-extension, `.ts`, and `.m4s`. This trio is delivered when a `<video>` element employs adaptive bitrate streaming using either HLS or MPEG-DASH. Video elements that deliver anything besides `.mp4` or adaptive bitrate streaming are rare, accounting for only 4% of the extensions we found.

Embedding

The `<video>` element offers a number of attributes that allow authors to control how the video will be loaded and presented on the page. Here they are, ranked by usage:

126. <https://caniuse.com/mpeg4>

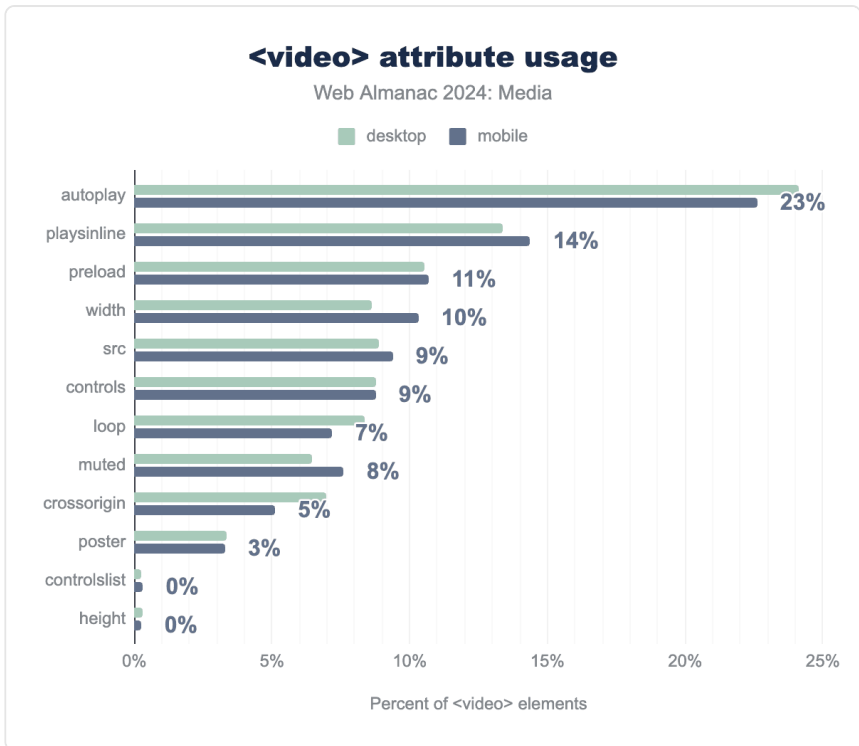


Figure 5.40. Video attribute usage.

While both `playsinline` and `autoplay` are up three percentage points from 2022—likely representing increased adoption of short inline videos that play the same role as GIFs—the biggest mover over the past few years has been `preload`, whose usage has decreased six percentage points.

This continues a trend we have seen throughout the 2020s, and our hypothesis about why remains the same as it was in 2022¹²⁷. Browsers know more than authors do about end users' contexts. By not including the `preload` attribute, authors are increasingly getting out of the browser's way.

src and source

The `src` attribute is only present on 9% of `<video>` elements. Many of the rest of the `<video>` elements on the web use `<source>` children, allowing authors to—in

127. <https://almanac.httparchive.org/en/2022/media#preload>

theory—supply multiple, alternate video resources for use in different contexts.

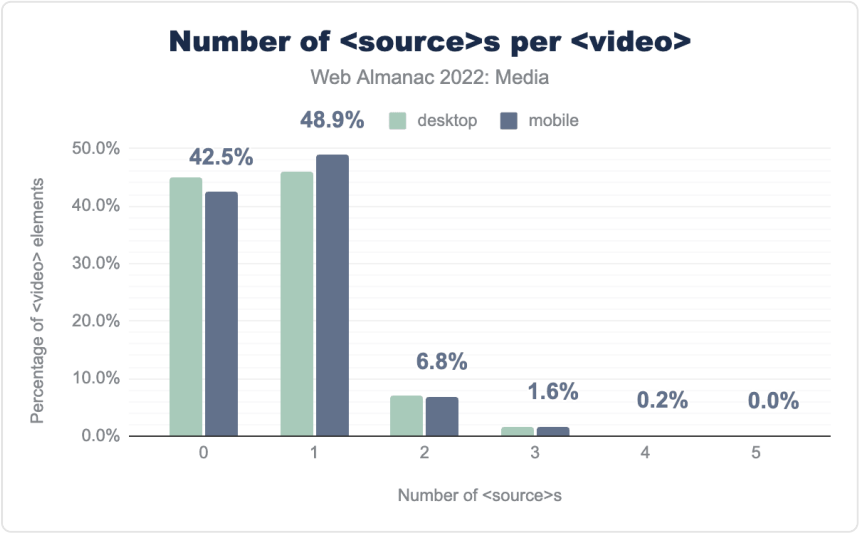


Figure 5.41. Number of `<source>`s per `<video>`.

However most of the `<video>` elements with `<source>` children only have one—only one in ten `<video>` elements have multiple `<source>`s.

Conclusion

So there you have it, a snapshot of the state of media on the web in 2024, along with a look at how things have changed over the last couple of years. We’ve seen just how ubiquitous and important media is to the user experience of the web, and taken stock of how sites are—and aren’t—delivering it effectively.

Perhaps unsurprisingly, we found that images on the web are getting bigger. Whether you’re counting image pixels or layout dimensions, the numbers are going up. So even though we also saw an increase in compression ratios—driven in part by increased adoption of modern image formats—total image byte sizes are going up, too.

In a similar vein, the most notable change we saw on the video side was simply that there were a lot more of them than there were two years ago. The web continues to get more and more visual.

Aside from those top-line findings, notable encouraging things we found in this year’s analysis

included the first sparks of adoption of wide-gamut color spaces, the continued rapid adoption of lazy-loading, and the continued steady rise of responsive image markup.

On the discouraging side, we saw a huge number of `<img>` elements with no or meaningless `alt` text, over-usage of lazy-loading leading to needlessly slow LCP times, a mysterious (if small) increase in GIF usage, and `sizes` accuracy getting even worse than it already was.

Here's to more effective visual communication on the web in 2025!

Authors



Stefan Judis

[X @stefanjudis](#) [@stefanjudis.com](#) [stefanjudis](#) <https://www.stefanjudis.com>

Stefan Judis fell in love with Frontend development ten years ago, and learns in public on his [blog](#)¹²⁸ and newsletter¹²⁹.



Eric Portis

[X @etportis](#) [@ericportis.com](#) [eeeps](#) <https://ericportis.com>

Eric Portis¹³⁰ is a Web Platform Advocate at Cloudinary¹³¹.

128. <https://www.stefanjudis.com/blog/>

129. <https://webweekly.email/>

130. <https://ericportis.com>

131. <https://cloudinary.com/>

Part I Chapter 6

Third Parties



Written by Tobias Urban, Yash Vekaria, Zubair Shafiq, and Chris Böttger

Reviewed by Barry Pollard

Analyzed by Yash Vekaria and Chris Böttger

Edited by Barry Pollard

Introduction

Website developers can use third parties to implement certain features such as advertising, analytics, social media integration, payment processing, and content delivery. A web page typically comprises resources served by the first party and various third parties. Using third parties to compose a web page allows for modular development, which enables efficient and rapid deployment of rich features but can also pose potential privacy, security, and performance issues.

In this chapter, we conduct an empirical analysis to shed light on the practice of using third parties on the web. We find that nearly all websites contain one or more third parties. We provide a breakdown of the types of resources served by these third parties, such as images, JavaScript, fonts, etc. We provide a breakdown of different categories of third parties on the Web, such as ad, analytics, CDN, video, tag manager, etc. We also provide a breakdown of how different third parties are included—directly or indirectly—on web pages.

Definitions

Before we start on our analysis, it helps to have some common definitions of what we will cover in this chapter.

Sites and pages

In this chapter, we use the term **site** to depict the registerable part of a given domain—often referred to as *extended Top Level Domain plus one* (eTLD+1). For example, given the URL `https://www.bar.com/` the eTLD+1 is `bar.com` and for the URL `https://foo.co.uk` the eTLD+1 is `foo.co.uk`. By **page** (or web page), we mean a unique URL or, more specifically, the document (for example HTML or JavaScript) located at the particular URL.

What is a third party?

We stick to the aforementioned definition of a third party used in previous editions of the Web Almanac to allow for comparison between this and the previous editions.

A **third party** is an entity different from the site owner (aka first party). It involves the aspects of the site not directly implemented and served by the site owner. More precisely, third-party content is loaded from a different site (i.e., the third party) rather than the one originally visited by the user. Assume that the user visits `example.com` (the first party) and `example.com` includes silly cat images from `awesome-cats.edu` (for example using an `<img>` tag). In that scenario, `awesome-cats.edu` is the third party, as it was not originally visited by the user. However, if the user directly visits `awesome-cats.edu`, `awesome-cats.edu` is the first party.

Only third parties originating from a domain whose resources can be found on at least 50 unique pages in the HTTP Archive dataset were included to match the definition. When third-party content is directly served from a first-party domain, it is counted as first-party content. For example, self-hosted CSS or fonts are counted as first-party content. Similarly, first-party content served from a third-party domain is counted as third-party content—assuming it passes the “more than 50 pages criteria.” Some third parties serve content from different subdomains. However, regardless of the number of subdomains, they are counted as a single third party. Further, it is becoming increasingly common for third parties to be masqueraded as a first party, for example, through techniques like CNAME cloaking¹³². We consider them a first party in this analysis. Thus, our results present a lower bound on the prevalence of third parties on the web.

132. <https://ldklab.github.io/assets/papers/madweb21-cloaking.pdf>

Categories

As previously indicated, third parties can be used for various use cases—for example, to include videos, to serve ads, or to include content from social media sites. To categorize the observed third parties in our dataset, we rely on the third-party Web¹³³ repository from Patrick Hulce¹³⁴. The repository breaks down third parties along the following categories:

- **Ad:** These scripts are part of advertising networks, either serving or measuring.
- **Analytics:** These scripts measure or track users and their actions. There's a wide range of impact here, depending on what's being tracked.
- **CDN:** These are a mixture of publicly hosted open source libraries (for example jQuery) served over different public CDNs and private CDN usage.
- **Content:** These scripts are from content providers or publishing-specific affiliate tracking.
- **Customer Success:** These scripts are from customer support/marketing providers that offer chat and contact solutions. These scripts are generally heavier in weight.
- **Hosting*:** These scripts are from web hosting platforms (WordPress, Wix, Squarespace, etc.).
- **Marketing:** These scripts are from marketing tools that add popups/newsletters/etc.
- **Social:** These scripts enable social features.
- **Tag Manager:** These scripts tend to load many other scripts and initiate many tasks.
- **Utility:** These scripts are developer utilities (API clients, site monitoring, fraud detection, etc.).
- **Video:** These scripts enable video player and streaming functionality.
- **Consent provider:** These scripts allow sites to manage the user consent (eg. for the General Data Protection Regulation¹³⁵ compliance). They are also known as the 'Cookie Consent' popups and are usually loaded on the critical path.
- **Other:** These are miscellaneous scripts delivered via a shared origin with no precise category or attribution.

133. <https://github.com/patrickhulce/third-party-web/#third-parties-by-category>

134. <https://x.com/patrickhulce>

135. https://wikipedia.org/wiki/General_Data_Protection_Regulation

Note: The CDN category here includes providers that provide resources on public CDN domains (for example `bootstrapcdn.com`, `cdnjs.cloudflare.com`, etc.) and does not include resources that are simply served over a CDN. For example, putting Cloudflare in front of a page would not influence its first-party designation according to our criteria.

Similar to previous years, the Hosting category is removed from our analysis. For example, if you happen to use WordPress.com for your blog, or Shopify for your e-commerce platform, then we're going to ignore other requests for those domains by that site as not truly "third-party" as they are, in many ways, part of hosting on those platforms.

Content Type

We use the `Content-Type` HTTP header to determine the type of the third party resources. The values of Content-Type include `text/javascript` or `application/javascript` (for scripts), `text/html` (for HTML content), `application/json` (for JSON data), `text/plain` (for plain text), `image/png` (for PNG images), `image/jpeg` (for JPEG images), `image/gif` (for GIF images), etc.

Prevalence

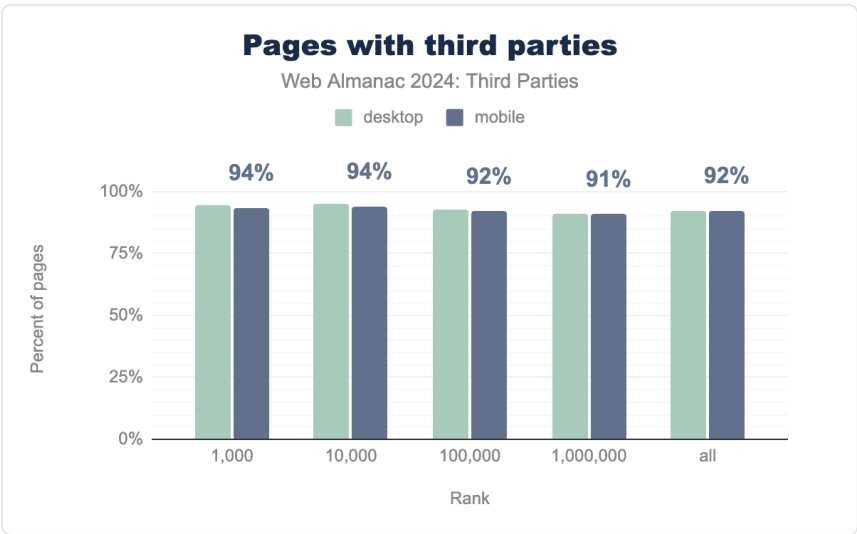


Figure 6.1. Percentage of pages that use one or more third parties.

There is a slight decrease in the percentage of pages that use one or more third parties for low-ranked websites. Similar to 2021 and 2022, the percentage of pages with one or more third

parties remains high at 92%.

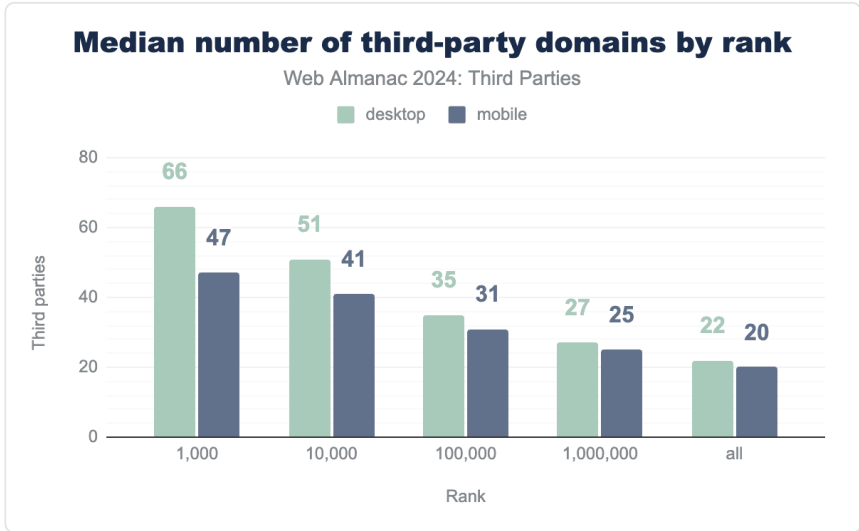


Figure 6.2. Distribution of the number of third parties by rank.

We note a considerable decrease in the number of third parties for lower-ranked websites. The median number of third-parties is 66 for the top thousand websites and 27 for the top million websites. The number of third parties on the desktop is higher than that for mobile pages. The contrast between desktop and mobile is greater for higher-ranked websites.

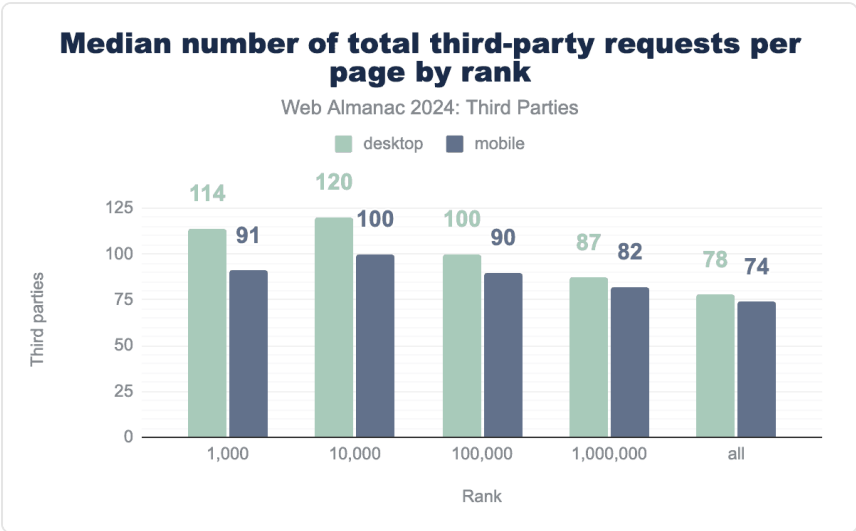


Figure 6.3. Distribution of the number of third party requests per page by rank.

We note that the number of third-party requests is higher for higher-ranked websites than lower-ranked websites. When looking at requests, the difference between higher- and lower-ranked websites is less skewed than when looking at the number of third-parties in figure 2.

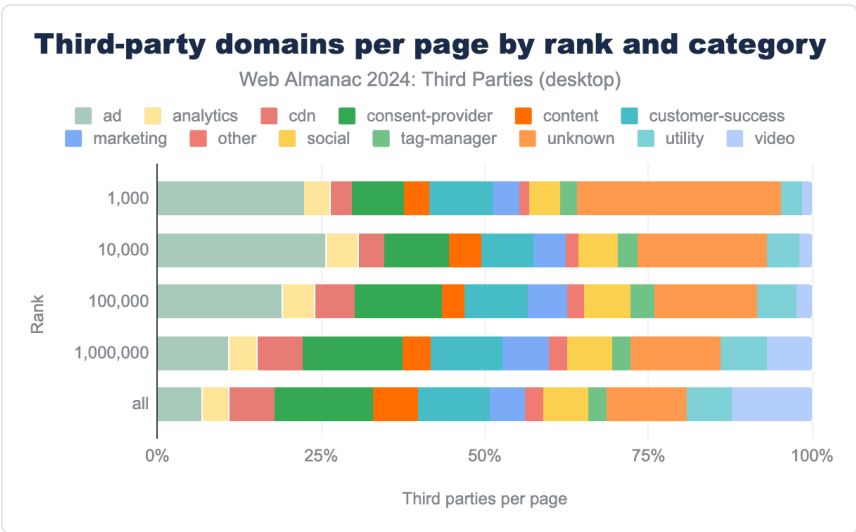


Figure 6.4. Distribution of the third party request categories by rank.

Excluding unknown, the top categories include consent provider, video, and customer success. The most popular consent provider domain is `fundingchoicesmessages.google.com`, the most popular video domain is `www.youtube.com`, and the most customer-success domain is `embed.tawk.to`.

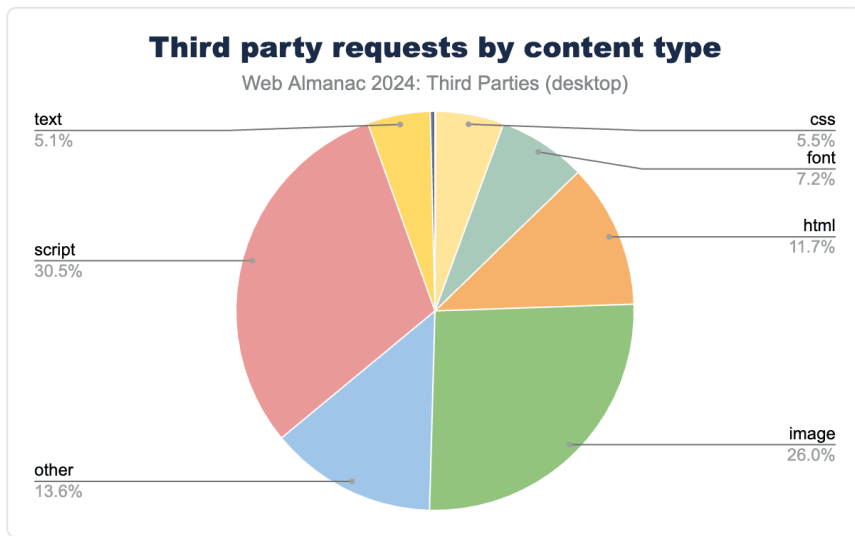


Figure 6.5. Distribution of the third party request types by rank.

The top 3 types include `script`, `image`, and `other`. The most popular domain under these content-types is `fonts.googleapis.com`.

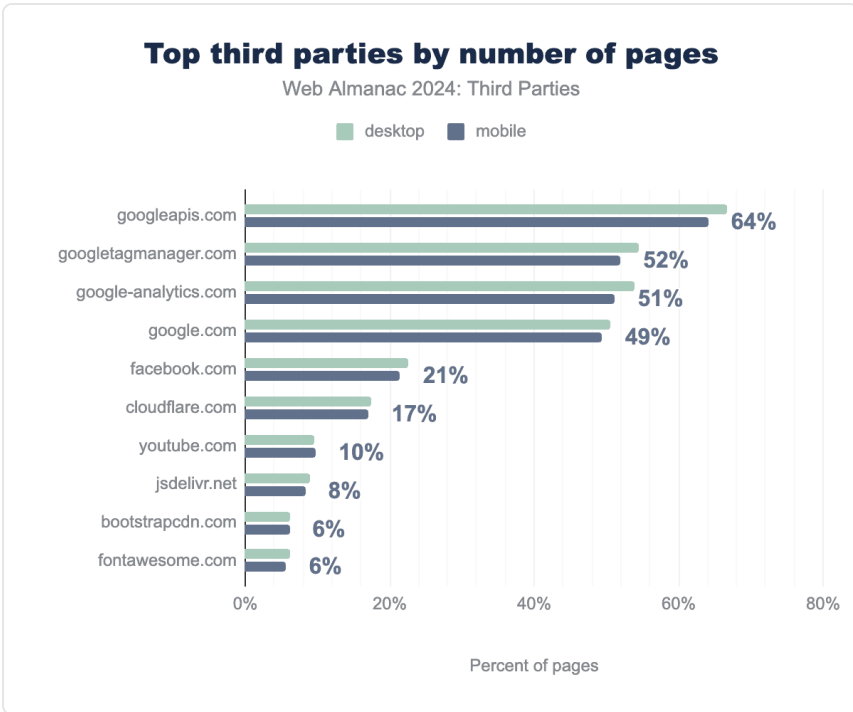


Figure 6.6. Top third parties by the number of pages.

The top 10 third-party domains include several Google-owned domains such as `googleapis.com`, `googletagmanager.com`, `google-analytics.com`, `google.com`, and `youtube.com`. Meta's `facebook.com` is the only non-Google domain in the top 5.

Inclusion

Recall from our earlier example that `example.com` (a first party) can include an image from `awesome-cats.edu` (a third party via an `<img>` tag). This inclusion of an image would be considered direct inclusion. However, if the image was loaded by a third-party script on the site via the `XMLHttpRequest`, then the inclusion of the image would be considered indirect inclusion. The indirectly included third parties can further include additional third parties. For example, a third-party script that is directly included on the site may further include another third-party script.

Such indirect inclusion of third parties on a page can be represented as a third-party inclusion chain. The inclusion chain can be constructed using the initiator information, identifying what

triggered a particular request. We use the eTLD+1 of a third party as the node identifier in the inclusion chain. An inclusion chain might include multiple domains operated by the same company (for example: `example.com` → `googletagmanager.com` → `google-analytics.com` → `doubleclick.net`) or different companies (for example: `example.com` → `googletagmanager.com` → `facebook.com`).

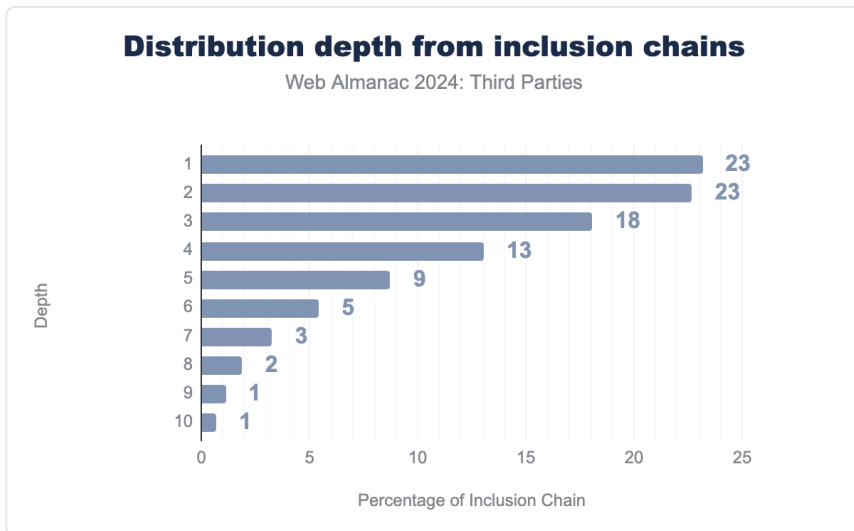


Figure 6.7. Median depth of third-party inclusion chains.

The median depth of the inclusion chains is 3.4 of the inclusion chains are of length > 1, which means that they indirectly include at least one third party on the page. Notably, 14% of the inclusion chains are of length > 5. The inclusion chain with the highest depth has a length of 2,930.

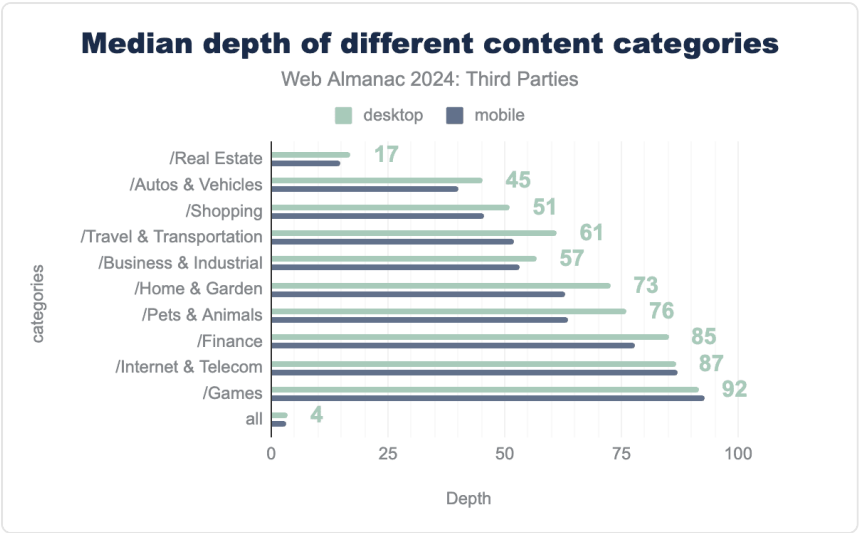


Figure 6.8. Median depth of different categories of websites.

Across all categories, desktop pages have longer inclusion chains than mobile pages. We observe substantial differences across different website categories. The website category with the longest inclusion chains is `/Games`.

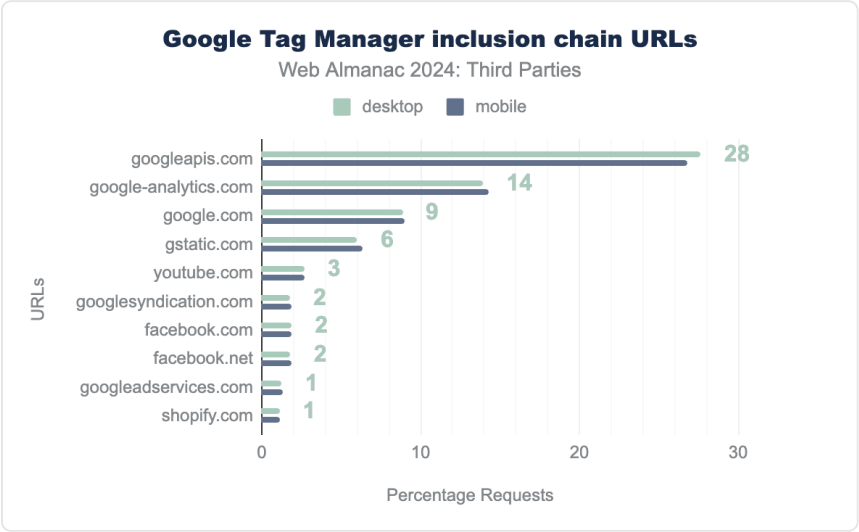


Figure 6.9. Google Tag Manager inclusion chain URLs.

When we look specifically at `googletagmanager.com`, one of the top third-party domains. Note that it includes a number of other Google domains such `googleapis.com`, `google-analytics.com`, `google.com`, `gstatic.com`, `youtube.com`, `googlesyndication.com`, and `googleadservices.com`. Only three of the top 10 third-party domains included by `googletagmanager.com` are non-Google domains, which are `facebook.com` and `facebook.net` for Meta and `shopify.com` for Shopify.

Conclusion

Our findings show the ubiquitous and complex nature of third-parties on the web. We find that the use of third parties on the web is more common than ever before. More than nine-in-ten web pages include one or more third-parties, often indirectly.

We find that third parties are often not directly included by the first party. Nearly one-third of third parties on all web pages are used for advertising, analytics, and consent management. Google is the most popular third party on the web, with five of the top ten third-party domains being Google domains: `googleapis.com`, `googletagmanager.com`, `google.com`, `google-analytics.com`, and `youtube.com`.

The inclusion of third-parties presents privacy, security, and performance implications that should be considered by web developers.

Authors



Tobias Urban

 [turban1988](#)  <https://internet-sicherheit.de/ueber-uns/team/alle-mitarbeiter/urban-tobias-2/>

Doctor of Engineering Tobias Urban is a Professor of Computer Science, specializing in Cybersecurity, at the Westphalian University of Applied Sciences¹³⁶. His research focuses on IT security, data protection, and the impact of the GDPR on the Internet.

136. <https://www.en.w-hs.de/>



Yash Vekaria

✉ @vekariayash 🌐 Yash-Vekaria 🌐 <https://www.yashvekaria.com/>

Yash Vekaria is a PhD candidate in Computer Science at University of California, Davis¹³⁷. He carries out web-based large-scale Internet measurements to study and improve the dynamics of web. Specifically, his research is focused at studying and bringing transparency to online tracking practices and user privacy issues.



Zubair Shafiq

✉ @zubair_shafiq 🌐 zubairshafiq 🌐 <http://www.cs.ucdavis.edu/~zubair>

Zubair Shafiq is a computer science professor at University of California, Davis¹³⁸. His research aims to make the Internet private, secure, and safe using empirically grounded measurement and modeling methods.



Chris Böttger

🌐 ChrisBeeti

Chris Böttger is a PhD candidate in Computer Science at the Westphalian University of Applied Science¹³⁹. His research focuses on web and network security, primarily focusing on user privacy and tracking technologies.

137. <https://www.ucdavis.edu/>

138. <https://www.ucdavis.edu/>

139. <https://www.en.w-hs.de/>

Part II Chapter 7

SEO



Written by Jamie Indigo, Dave Smart, Mikael Araújo, and Michael Lewittes

Reviewed by Barry Pollard

Analyzed by Henry Price and Chris Nichols

Edited by Michael Lewittes

Introduction

Search Engine Optimization (SEO) is the practice of improving a website's technical setup, content, and authority to boost its visibility in search results. It helps businesses attract organic traffic by aligning website content with user search intent.

The Web Almanac's SEO chapter focuses on the critical elements and configurations influencing a website's organic search visibility. The primary goal is to provide actionable insights that empower website owners to enhance their sites' crawlability, indexability, and overall search engine rankings. By conducting a comprehensive analysis of prevalent SEO factors, we hope that websites can uncover the most impactful strategies and techniques for optimizing a website's presence in search results.

This chapter combines data from HTTP Archive¹⁴⁰, Lighthouse¹⁴¹, Chrome User Experience

140. <https://httparchive.org/>

141. <https://developer.chrome.com/docs/lighthouse/overview/>

Report¹⁴², and custom metrics to document the state of SEO and its context in the digital landscape.

This year, we analyzed one inner page per site crawled in addition to the home pages, which is all this chapter has historically analyzed. Since home pages are often quite different from inner pages, this has unlocked new insights and allowed us to compare the behaviors of home pages verses inner pages.

Crawlability & indexability

For a page to appear in a search engine results page (SERP), the page must first be crawled and indexed. This process is a critical foundation of SEO.

Search engines may discover pages in several ways, including external links, sitemaps, or by being directly submitted to the search engine using webmaster tools. In 2022, Bing shared that its crawler discovered nearly 70 billion new pages *daily*. During this year's antitrust suit against Google, it was revealed that its *index* is only around 400 billion documents¹⁴³. That means far more pages are crawled than indexed.

This section addresses the state of the web, as it relates to how search engines crawl and index content, as well as the directives and signals SEOs can provide so that crawlers interact and retain versions of their content.

robots.txt

As search engines explore the web, they stop at the `robots.txt` file, which one can think of as a visitors' center for each site. The `robots.txt` file sits at the root of an origin and allows site owners to implement the Robots Exclusion Protocol¹⁴⁴. It's designed to signal or instruct bots which URLs a search engine can or cannot crawl.

It is not a hard, technical directive. Rather, it's up to the bot to honor these instructions. Since the major search engines respect this protocol, it's relevant for our SEO analysis.

While the `robot.txt` file has been used since 1994 to control how a site is crawled, it only became formally standardized with the Internet Engineering Task Force in September 2022. The formalization of the RFC 9390¹⁴⁵ protocol in 2022 occurred after the previous year's edition of the Web Almanac was published and led to stricter enforcement of technical standards.

142. <https://developers.google.com/web/tools/chrome-user-experience-report>

143. <https://zyppy.com/seo/google-index-size/#:-:text=But%20recently%2C%20during%20testimony%20in,Google's%20index%20size%20in%202020.>

144. <https://wikipedia.org/wiki/Robots.txt>

145. <https://datatracker.ietf.org/doc/html/rfc9309>

For the measurements for this year's Web Almanac, we ran Lighthouse, an open-source, automated tool for improving the quality of web pages in tandem with our own data collection. These audits showed that 8.43% of desktop pages and 7.40% of mobile pages failed the tool's check for valid `robots.txt` files.

`robots.txt` status codes

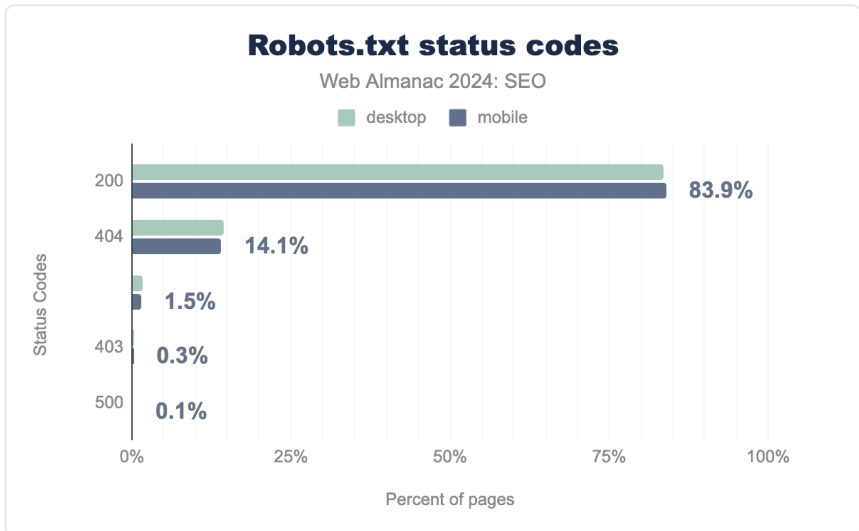


Figure 7.1. `robots.txt` status codes.

Since 2022, there has been a nominal increase in the percentage of sites whose `robots.txt` files return a 200 status code. In 2024, 83.9% of `robots.txt` files for mobile sites returned a 200 status code, while 83.5% of desktop sites returned a 200 status code. That's up from 2022 when mobile and desktop sites returned 200 status codes of 82.4% and desktop 81.5%, respectively.

This small increase signals that, despite the standard's relatively recent formalization, its previous three-decade history had already led to wide-scale adoption.

Additionally, the difference between mobile and desktop sites returning a 200 status code has now narrowed to just a 0.4% difference, compared to the 1.1% gap in 2022. While one cannot draw a definitive conclusion for the percentage decrease, one possible explanation can be the greater adoption of mobile responsive design principles versus the previous prevalence of separate mobile sites.

HTTP status codes significantly impact how a `robots.txt` file functions. When the file

returns a 4XX status code, it is assumed there are no crawling restrictions. Awareness of this behavior has continued to increase as we see a continuing trend of fewer 404 responses to `robots.txt` files.

In 2022, 15.8% of mobile sites' `robots.txt` files returned a 404 status code and 16.5% of desktop sites. Now in 2024, it's down to 14.1% for mobile site visits and 14.3% for desktop. The drops are fairly consistent with the growth of `robots.txt` returning 200 status codes, suggesting more sites have decided to implement a `robots.txt` file.

Only 1.7% of mobile and 1.5% of desktop crawls in 2024 received no response. Google interprets these as an error caused by the server.

For 0.1% of both mobile and desktop requests tested, we received an error code in the 5xx range. While these error codes represent a tiny percentage, it's worth noting that for Google this would mean the search engine would consider the whole site blocked from crawling for 30 days. After 30 days, the search engine would revert to using a previously fetched version of the file. If a prior cached version isn't available, it is assumed the search engine crawled all of the content hosted on the site.

The nominal error rate suggests that simple text files in most cases—or handled automatically by many popular CMS systems that provide a `robots.txt` directive—are not a large challenge.

Note: The above data does not indicate whether the `robots.txt` files returning a 200 status code are beneficial for a site or if they allow or block aspects that they should not.

robots.txt size

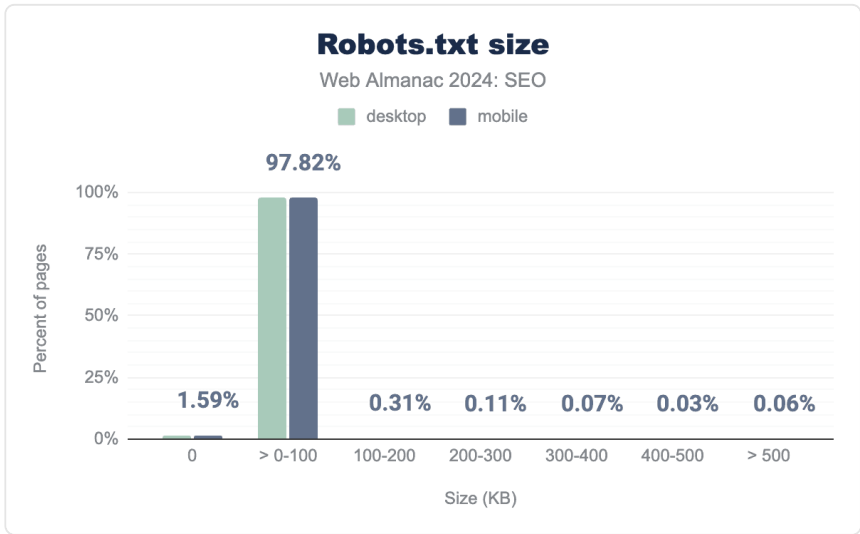


Figure 7.2. robots.txt size.

The vast majority of robots.txt files—97.82% of mobile crawls and 97.80% of desktop crawls—were no larger than 100KB.

According to RFC 9309 standards, crawlers should limit the size of robots.txt files they look at, and the parsing limit must be at least 500 kiB¹⁴⁶. A robots.txt file under that size should be fully parsed. Google, for example, enforces the max limit at 500 kiB¹⁴⁷. Only a tiny number of sites (just 0.06%) had robots.txt files over this limit. Directives found beyond that limit are ignored by the search engine.

Interestingly, 1.59% of mobile crawls and 1.66% of desktop crawls returned a 0-sized robots.txt file. This is likely a configuration issue. Since it is not documented by the RFC 9303 specification or support documentation for popular search engine crawlers, it is unclear how this would be handled. If a site returns an empty response for robots.txt, a sensible approach would be to return a robots.txt file with appropriate rules or, if one does not wish to restrict crawling, return a 404 status code for the URL.

146. <https://www.rfc-editor.org/rfc/rfc9309.html#name-limit>

147. https://developers.google.com/search/docs/crawling-indexing/robots/robots_txt#file-format

robots . txt user agent usage

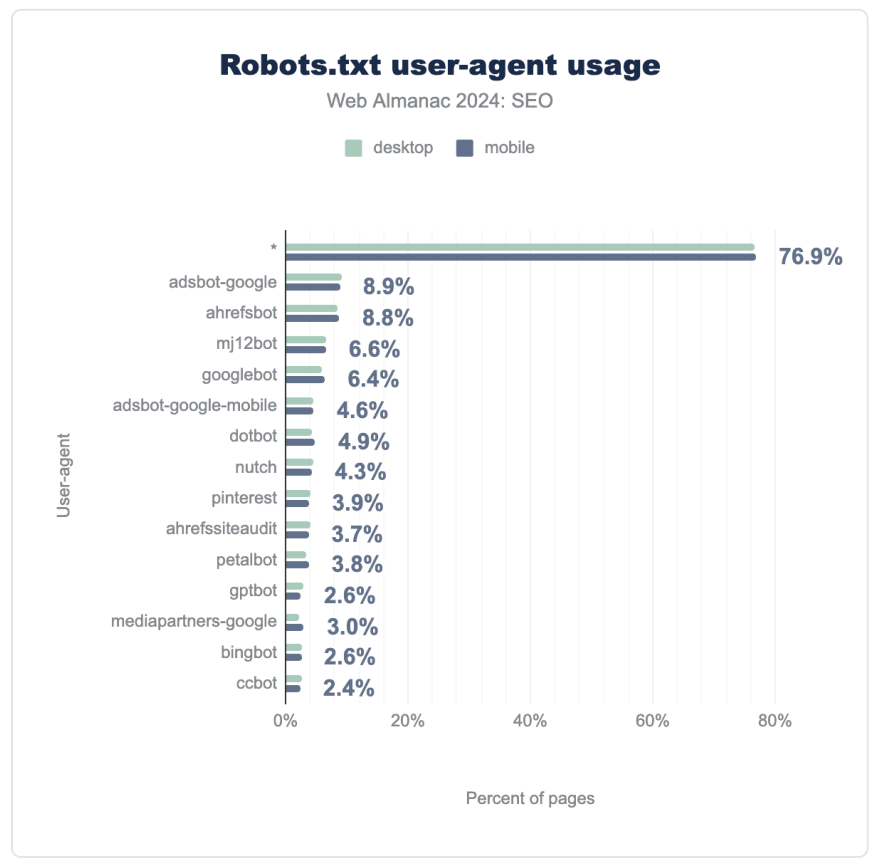


Figure 7.3. robots . txt user agent usage

The * user agent

A full 76.9% of robots . txt files encountered by the mobile crawl and 76.6% of those found in the desktop crawl specify rules for the catch-all user agent of *. This represents a small uptick over the data from 2022 in which it was 74.9% for desktop and 76.1% for mobile crawls. A possible explanation could be the slight overall increase in robots . txt availability.

The * user agent denotes the rules a crawler should follow unless there's another set of rules that specifically target the crawler's user - agent . There are notable exceptions that don't

respect the `*` user agent, including Google's Adsbob crawler¹⁴⁸. Other crawlers will try another common `user-agent` before falling back to `*`, such as Apple's `Applebot`, which uses `Googlebot`'s rules if they are specified and `Applebot` if not specified. We recommend checking the support documentation for any crawlers you are targeting to ensure that behavior is as expected when relying on fallback.

Bingbot

Much like in 2022, `Bingbot` again was not in the top 10 most specified `user-agent`s. Only 2.7% of mobile and 2.6% of desktop `robots.txt` files specified that `user-agent`, relegating it down to 14th place.

SEO tools

The data shows there has been an increase in sites specifying rules for the popular SEO tools. `AhrefsBot`, for instance, has now been detected in 8.8% of mobile crawls, up from 5.3% in 2022. It has overtaken Majestic's `MJ12Bot`, which itself increased to 6.6% from 6.0% in 2022 and had previously been the second most popular specifically targeted `user-agent`.

148. <https://developers.google.com/search/docs/crawling-indexing/google-special-case-crawlers#adsbot-mobile-web>

AI crawlers

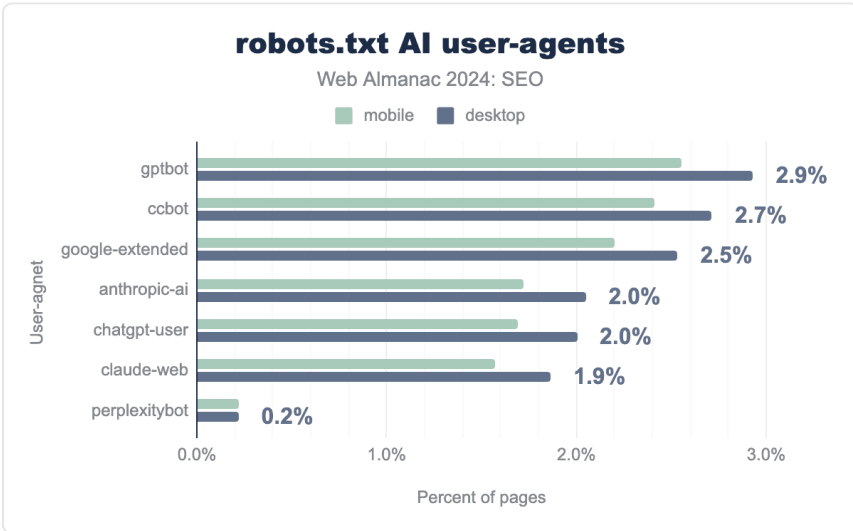


Figure 7.4. `robots.txt` AI user-agent usage.

Over the past two years, large language models (LLMs) and other generative systems have gained traction in both awareness and usage. It appears people are increasingly specifying rules for the crawlers they use in order to gather data for training and other purposes.

Of these, `GPTBot` is the most commonly specified and found in 2.7% of mobile crawls. The next most common is `CCBot`, which is Common Crawl's agent¹⁴⁹. While `CCBot` isn't related only to AI, a number of popular vendors use or have used data gathered from this crawler to train their models.

In summary:

- The formalization of the `robots.txt` protocol in RFC 9309 has led to better adherence to technical standards.
- Analysis shows an increase in successful `robots.txt` responses and a decrease in errors, indicating improved implementation.
- Most `robots.txt` files are within the recommended size limit.
- The `* user-agent` remains dominant, but AI crawlers like `GPTBot` are on the

149. <https://commoncrawl.org/ccbot>

rise.

- These insights are valuable for understanding the current state of `robots.txt` usage and its implications for SEO.

Robots directives

A robots directive¹⁵⁰ is a granular, page-specific approach to controlling how an individual HTML page should be indexed and served. Robots directives are similar to, but distinct from, `robots.txt` files since the former affect indexing and serving while `robots.txt` affects crawling. In order for directives to be followed, the crawler must be allowed to access the page. Directives on pages that are disallowed in a `robots.txt` file may not be read and followed.

Robots directive implementation

Robots directives tags are critical for curating which pages are available to return in search results and how they should be displayed. Robots directives may be implemented in two ways:

1. Placing a robots meta tag in the `<head>` of a page (for example, `<meta name="robots" content="noindex">`).
2. Placing the X-Robots-Tag HTTP header in the HTTP header response.

150. <https://developers.google.com/search/docs/crawling-indexing/robots-meta-tag>

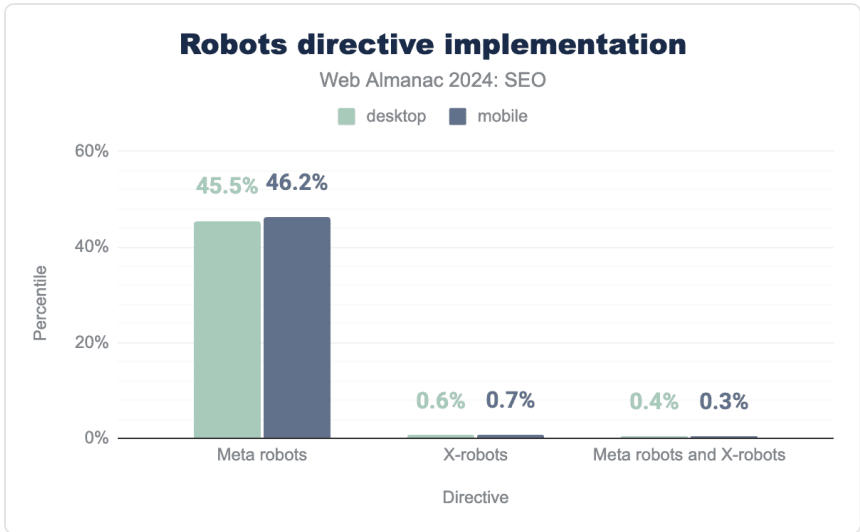


Figure 7.5. Robots directive implementation

Either implementation method is valid and can be used in tandem. The meta tag implementation is the most widely adopted, representing 45.5% of desktop and 46.2% of mobile pages. The X-robots HTTP header¹⁵¹ is applied to fewer than 1% of pages. A small number of sites used both tags in tandem. They represented 0.4% of desktop pages and 0.3% of mobile pages.

In 2024:

- 0.4% of desktop and 0.3% of mobile pages saw the directives' values changed by rendering.
- Inner pages were more likely to have robots directives. And 48% of inner pages contained a meta robots tag compared to 43.9% of home pages.
- Rendering was more likely to change the robots directive of a home page (0.4%) than that of an inner page (0.3%).

Robots directive rules

In 2024, there were 24 valid values¹⁵²—known as rules—that could be asserted in a directive to control the indexing and serving of a snippet. Multiple rules can be combined via separate meta

151. <https://developers.google.com/search/docs/crawling-indexing/robots-meta-tag#xrobotstag>

152. <https://developers.google.com/search/docs/crawling-indexing/robots-meta-tag#directives>

tags or combined in a comma separated-list for both meta tags and `X-robots` HTTP headers.

For our study of directive rules, we relied on the rendered HTML.

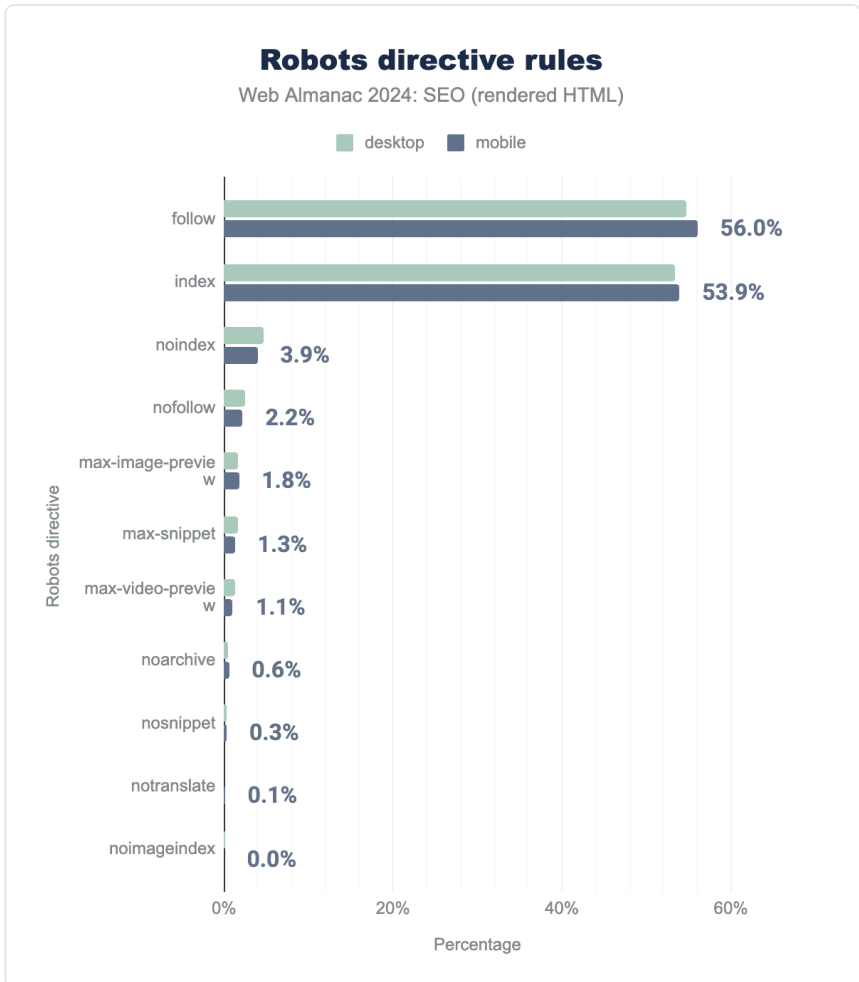


Figure 7.6. Robots directive rules.

The most prominent rules in 2024 were `follow` (54.7% desktop; 56.0% mobile), `index` (53.4% desktop; 53.9% mobile), `noindex` (4.7% desktop; 3.9% mobile), and `nofollow` (2.5% desktop; 2.2% mobile). This is noteworthy since neither “index” nor “follow” directives have any function and are ignored by `Googlebot`. Google’s documentation on robots tags¹⁵³ advises that

153. <https://developers.google.com/search/docs/crawling-indexing/special-tags>

“The default values are index, follow and don't need to be specified.”

The `name` value of the robots `meta` tag specifies to which crawler(s) the rule applies. For example, `meta name="robots"` applies to all bots whereas `meta name="googlebot"` applies to only to Google. To analyze the application of name attributes, we looked at rates where values were stated for the `follow` tag since it is the most prevalent robots `meta` rule.

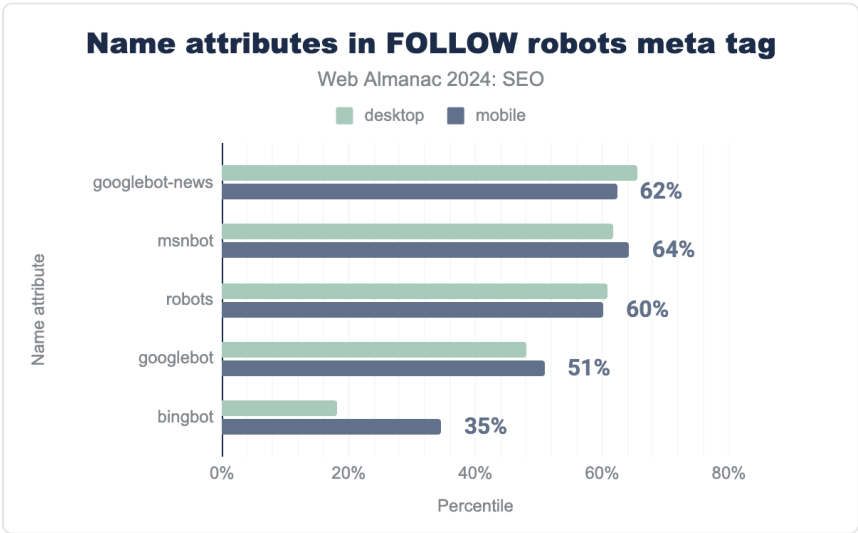


Figure 7.7. Name attributes in `follow` robots meta tag.

The five most named crawlers in robots directives were the generic robots value, Googlebot, Googlebot-News, MSNBot, and Bingbot. The name attributes used in the `follow` robots `meta` tag show that sites with the tags are apt to tailor their rules to specific bots. While there was generally a slight variance by device, there was a large difference for Bingbot which saw significantly more follow directives on mobile pages (35%) compared to desktop (18%).

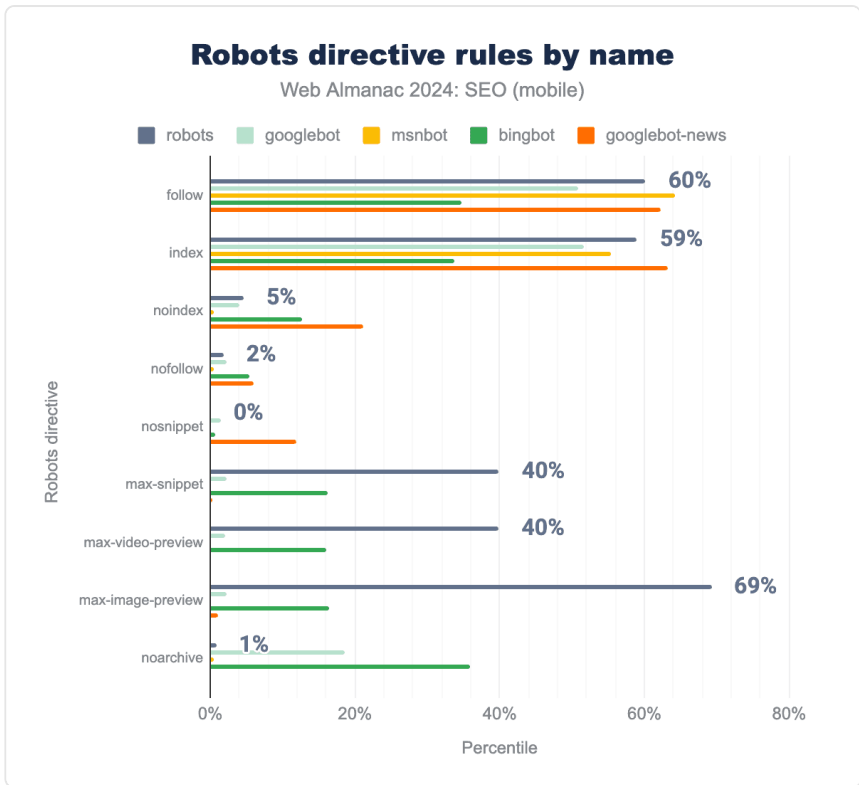


Figure 7.8. Robots rules by name attribute values.

When robots directive rules are viewed by their name attributes, we can see varied application rates. This implies that SEOs are adopting directives by specific bot names to curate indexing and serving for individual search engines.

Noteworthy takeaways from our analysis of the rules by bot name include:

- The `noarchive` rule was applied overwhelmingly to `Bingbot` at 36%. This is likely due to the tag's ability to keep content out of Bing chat answers¹⁵⁴.
- `max-snippet`, `max-video-preview`, and `max-image-preview` rules are widely implemented for all robots at the rate of 40%, 40%, and 69%, respectively.
- `Googlebot-News` was the most named for `index` (63%) and `nosnippet` (12%).
- `MSNBOT` was the least likely to be given a `noindex` directive (1%). In comparison,

154. <https://blogs.bing.com/webmaster/september-2023/Announcing-new-options-for-webmasters-to-control-usage-of-their-content-in-Bing-Chat>

the most likely was `Googlebot-News` at 21%.

- 0.01% of sites provided a `noindex` rule, using the invalid crawler name: Google. Google has two valid crawler names for recognized robots `meta` tags: `Googlebot` and `Googlebot-News`.

IndexIfEmbedded tag

In January 2022, Google introduced a new robots tag¹⁵⁵, `indexifembedded`. The tag is placed in the HTTP header and offers indexing control for resources used to build a page. A common use case for this tag is for controlling indexation when content is in an `iframe` on a page, even when a `noindex` tag has been applied.

The presence of an `<iframe>` provides a baseline for cases where the `indexifembedded` robots directive might be applicable. In 2024, 7.6% of mobile pages contained an `<iframe>` element. This is a noteworthy increase of 85% from 2022's rate of 4.1%.

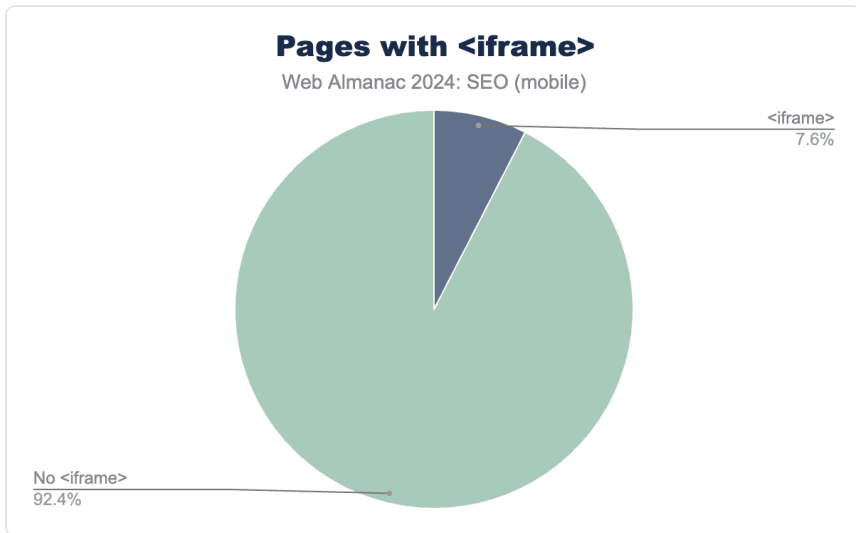


Figure 7.9. Mobile pages with `<iframe>`.

Nearly all sites employing iframes also use the `indexifembedded` directive. When `iframe` headers from mobile pages were examined, 99.9% used `noindex` directives and 97.8% used `indexifembedded`.

155. <https://developers.google.com/search/blog/2022/01/robots-meta-tag-indexifembedded>

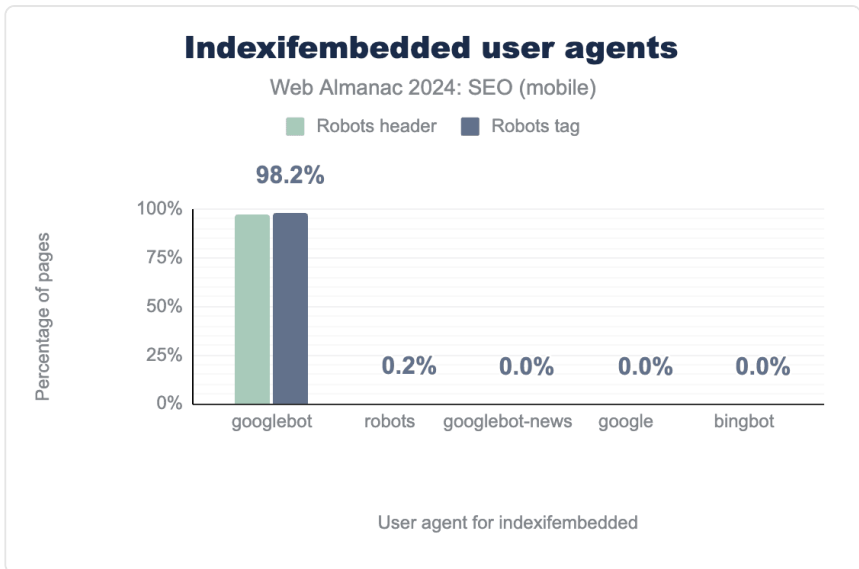


Figure 7.10. Indexifembedded user agents.

As we saw in 2022, `indexifembedded` directives continued to be almost exclusively used for `Googlebot`. While robots header use decreased slightly to 97.2% in 2024 from 98.3% in 2022, adoption of the robots tag increased significantly to 98.2% in 2024 from 66.3% in 2022.

Invalid `<head>` elements

Search engine crawlers follow HTML standard and when they encounter an invalid element in the `<head>`, it ends the `<head>` and assumes the `<body>` has started. This can prevent important metadata from being discovered or incomplete renderings.

The impact of a prematurely closed `<head>` is often difficult to catch since the problematic tag's position may change on every page load. Broken `<head>` tags are frequently identified by reports on missing elements such as `canonical`, `hreflang`, and `title` tags.

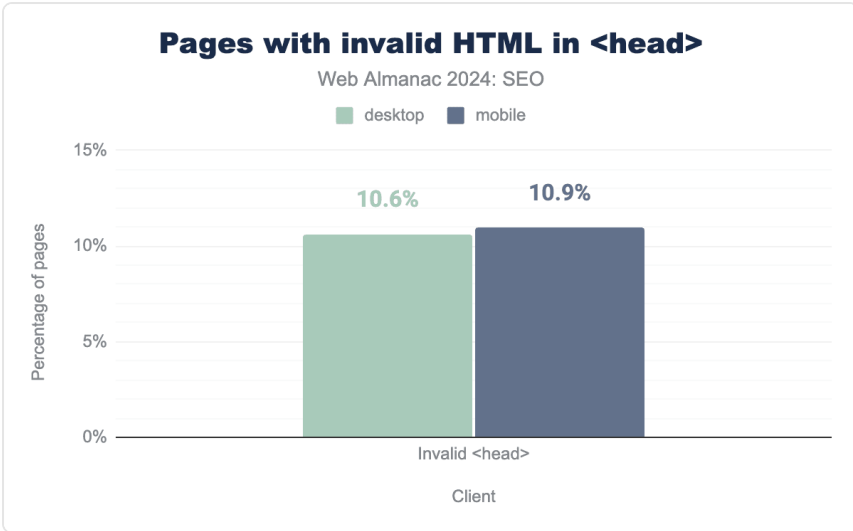


Figure 7.11. Invalid HTML in `<head>`.

In 2024, 10.9% of mobile pages had `<head>` breaking invalid HTML elements. That represented a 12% decrease from 2022's rate of 12.6%. Meanwhile, desktop pages with invalid HTML in the `<head>` decreased from 12.7% in 2022 to 10.6% in 2024.

10.9%

Figure 7.12. mobiles pages contained invalid HTML elements in the `<head>`

There are eight valid elements that may be used in the `<head>` according to Google Search documentation¹⁵⁶. These are:

1. `title`
2. `meta`
3. `link`
4. `script`
5. `style`
6. `base`
7. `noscript`
8. `template`

156. <https://developers.google.com/search/docs/crawling-indexing/valid-page-metadata>

No element other than the aforementioned is permitted by the HTML standard in the `<head>` element. Documentation¹⁵⁷ further states, “Once Google detects one of these invalid elements, it assumes the end of the `<head>` element and stops reading any further elements in the `<head>` element.”

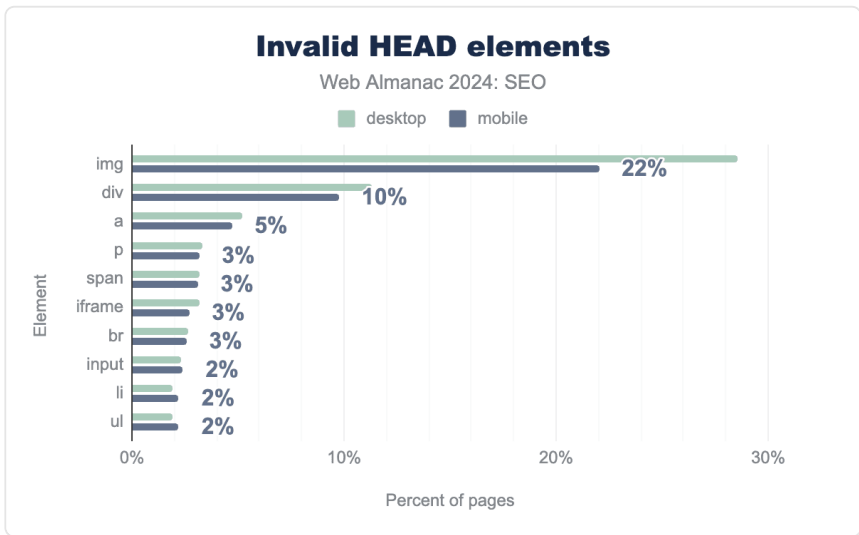


Figure 7.13. Invalid `<head>` elements.

The most prevalent `<head>` breaking tag was the `<img>` element, affecting 29% of desktop and 22% of mobile instances of the issue. Comparatively, the 2022 chapter found `<img>` tags misapplied on 10% of mobile pages and 10% of desktop pages. The likely difference comes from deprecated implementation methods for third-party tools.

Misapplied `<div>` tags also substantially increased from 2022. In 2024, 11% of desktop and 10% of mobile pages had the `<div>` element in the `<head>`. That’s more than a three times increase from 2022 when the invalid `<head>` occurred on 4% of desktop pages and 4% of mobile pages.

Canonicalization

Canonicalization is the process of identifying the “preferred” version of a document when multiple versions are available. This is often necessary when a website has the same content accessible through different URLs, such as with HTTP/HTTPS, www/non-www, trailing slashes,

157. <https://developers.google.com/search/docs/crawling-indexing/valid-page-metadata#dont-use-invalid-elements-in-the-head-element>

query parameters, and other variations.

`canonical` tags are signals for search engines about which version of the content to return in search results. While they are not directives like meta robots, they do serve as “strong hints.” They benefit SEO by mitigating duplicate content, consolidating signals such as links to page variations, and allowing webmasters to better manage content syndication.

`canonical` tag usage was up slightly in 2024. In 2022, 61% of mobile and 59% of desktop pages used `canonical` tags. In 2024, it was up to 65% of mobile and 69% of desktop pages.

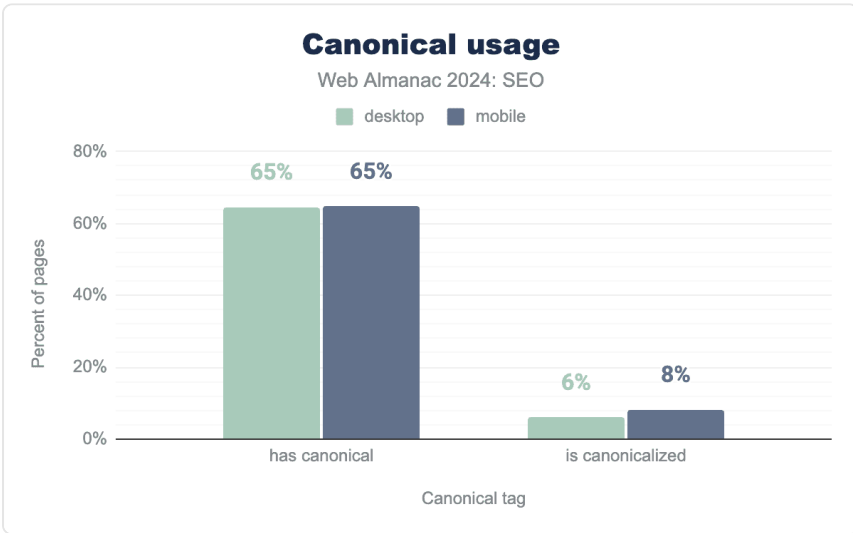


Figure 7.14. Canonical usage.

Canonical implementation

`canonical` tags have three implementation methods¹⁵⁸:

1. In the HTTP headers (via the Link HTTP header.
2. In the HTML's `<head>` section of a pag.
3. Via sitemap

HTML `<head>` tag implementation can occur in two specific points:

1. As a link in the raw HTML received in response from the serve.
2. As a link In the rendered HTML after scripts have been executed

158. <https://developers.google.com/search/docs/crawling-indexing/consolidate-duplicate-urls>

Each implementation has its own nuance. HTTP headers can be used on non-HTML documents like PDFs, whereas `rel="canonical"` cannot. Additionally, canonicals via sitemap may be easier to maintain but are a weaker signal¹⁵⁹ than on-page declarations.

Analyzing `canonical` sitemap implementation requires determining the associated duplicate for any declared `canonical` values. As excited researchers, we adjusted our analysis accordingly to report on the three places where search crawlers would encounter `canonical` tags. A `canonical` could first be found in the HTTP header, next in the raw HTML, and finally in the rendered DOM.

Only 1% of mobile pages currently use the HTTP header, down from 1% in 2022. This is likely due to its implementation requiring server configuration. Instead, 65% of mobile pages use a `rel="canonical"` nested in the `<head>`.

Most sites using `<head>` `canonical` tags implement them in the raw and rendered HTML. Fewer than 1% of mobile and desktop pages had a `canonical` element present in the raw HTML (but not in the rendered HTML).

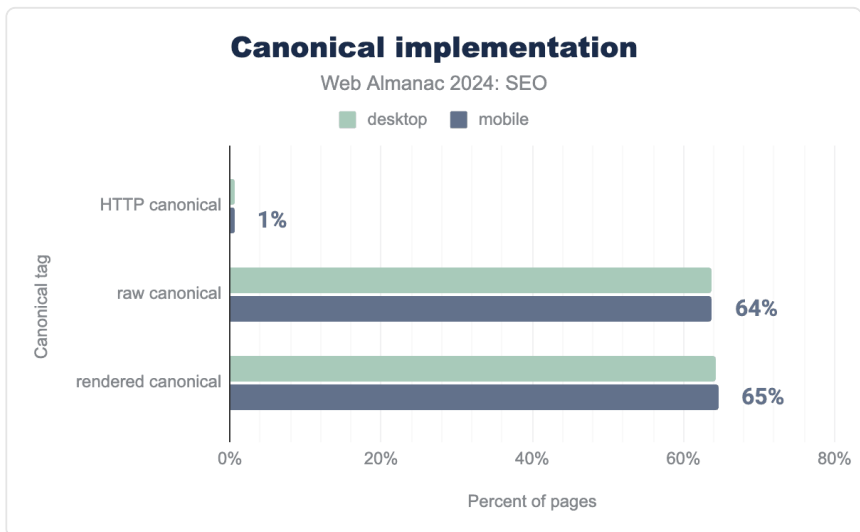


Figure 7.15. Canonical implementation methods.

Rendered `canonical` implementation was up from 60% of mobile in 2022 to 65% of mobile usage in 2024. Desktop usage increased from 58% in 2022 to 65% in 2024, making adoption rates near identical between device types.

159. <https://developers.google.com/search/docs/crawling-indexing/consolidate-duplicate-urls#---text=Less%20powerful%20signal%20to%20Google%20than%20the%20rel%3D%22canonical%22%20mapping%20technique.>

Canonical conflicts

There are three opportunities for a page to send a `canonical`, but sending the signal more than once can result in conflicts. For example, if the `canonical` URL in the HTTP doesn't match that in the rendered HTML, then the search engine is signaled that the primary version of content exists in multiple places. This undermines the purpose of the element and causes undefined behavior, according to Google¹⁶⁰. In 2022, this affected 0.4% of pages. Now, the rate has doubled to 0.8% in 2024.

Similarly, rendering can change the `canonical` found in the raw HTML. This is more prevalent, and it affects 2.1% of mobile pages. HTTP header `canonical` tags were less likely to be changed in the rendering process. In 2024, only 0.6% of desktop pages and 0.5% of mobile pages saw a `canonical` value passed in the HTTP header and then changed.

Of pages where a `canonical` element was detected, 98% passed the Lighthouse audit for valid `rel=canonical`.

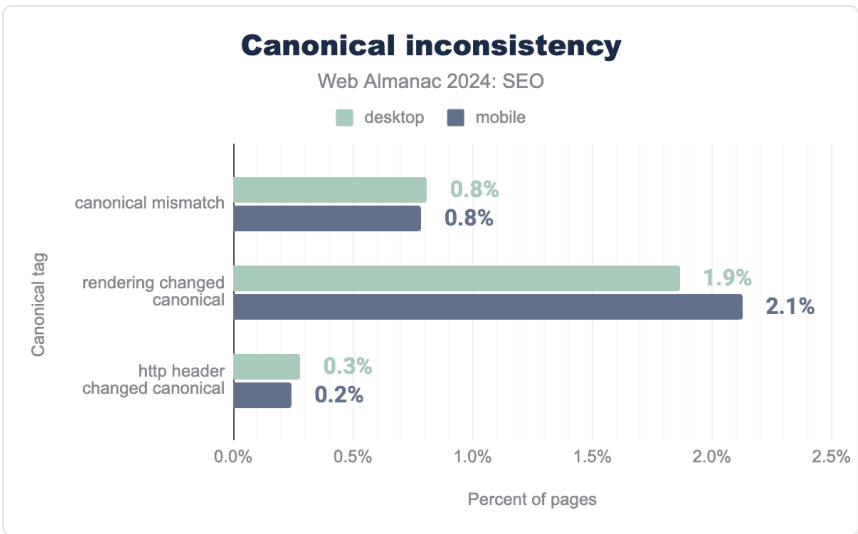


Figure 7.16. Canonical inconsistency.

Mismatched `canonical` values occurred on 0.8% of mobile and desktop pages. Rendering was more likely to change `canonical` elements on desktop (1.9%) than on mobile (2.1%). HTTP `canonical` elements, while seldom used, were changed during the rendering process in 0.3% of desktop pages and 0.2% of mobile pages.

¹⁶⁰. <https://www.youtube.com/watch?v=bAE3L1E1Fmk&t=772s>

Page experience

Simply, users want good experiences on the web. In 2020, Google introduced page experience to its algorithm. This section looks at how page experience has evolved.

HTTPS

Google adopted HTTP as a ranking signal in 2014¹⁶¹. HTTPS¹⁶² uses a protocol to encrypt communications. It's established via the presence of a secure certificate issued by a third party at the time of crawl. Adoption has continued to steadily increase over the years. In 2024, 89% of desktop pages and 88.9% of mobile pages used the HTTPS protocol.

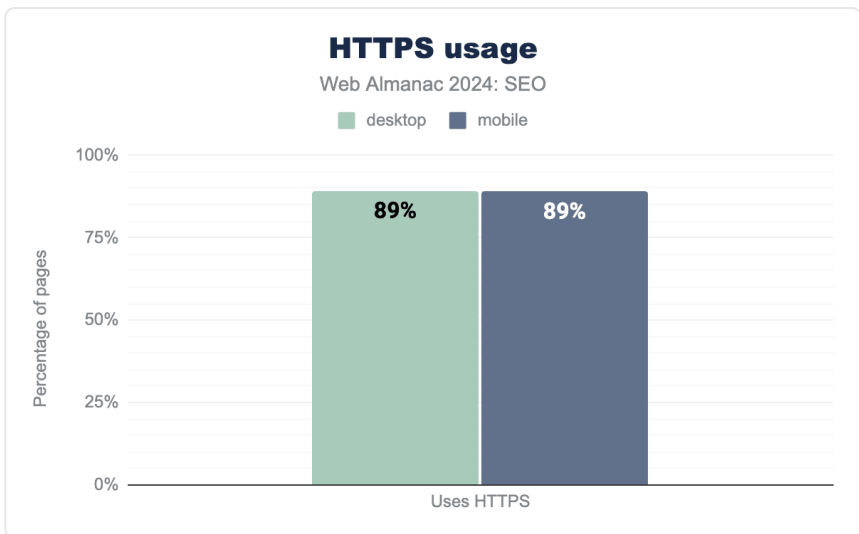


Figure 7.17. Percentage of websites supporting HTTPS.

For a more in-depth analysis of this topic, see the Security chapter.

Mobile-friendliness

Search engines and websites have a common goal—meeting users where they are. There are 6.61 billion mobile users worldwide, and 69.4% of the world's total population uses a mobile device.

161. <https://developers.google.com/search/blog/2014/08/https-as-ranking-signal>

162. <https://www.cloudflare.com/learning/ssl/what-is-https/>

Google search has considered mobile-friendliness to be a requirement¹⁶³ since 2015. The search engine completed its seven-year migration to a mobile-first index¹⁶⁴ in 2023.

Mobile-friendliness can be determined by the presence of two tags:

1. **Viewport** meta tag, which is commonly used in responsive design.
2. **Vary: User-Agent** header, which is used in dynamic serving and is based on the request header

Viewport meta tag

A `<meta name="viewport">` optimizes for mobile screen sizes and can reduce delays to user input¹⁶⁵.

Usage of the Viewport meta tag¹⁶⁶ continued to increase in 2024 with 92% of desktop pages and 94% of mobile pages passing the Lighthouse check for a 'viewport tag' with `width` or `initial-scale` set. This was up by 1% from 2022's adoption rates when 91% of mobile pages used the tag.

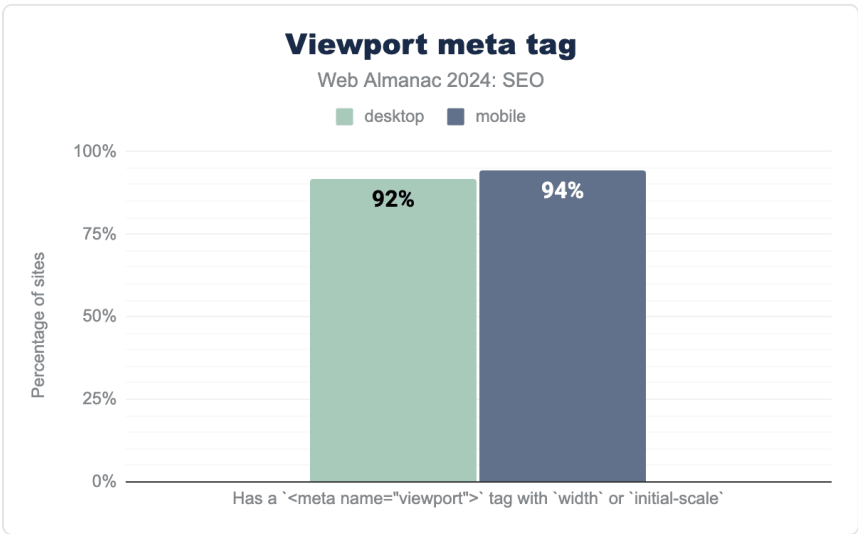


Figure 7.18. Viewport meta tag usage.

Viewport meta tag use on mobile increased to 94% in 2024 from 92% in 2022.

163. <https://developers.google.com/search/blog/2015/02/finding-more-mobile-friendly-search>

164. <https://developers.google.com/search/blog/2023/10/mobile-first-is-here>

165. <https://developer.chrome.com/docs/lighthouse/pwa/viewport/>

166. <https://dequeuniversity.com/rules/axe/4.9/meta-viewport>

Vary header usage

The `vary` HTTP response header enables different content to be served based on the requesting user agent. Also known as dynamic serving, this header allows the page to return content best suited for the requesting device.

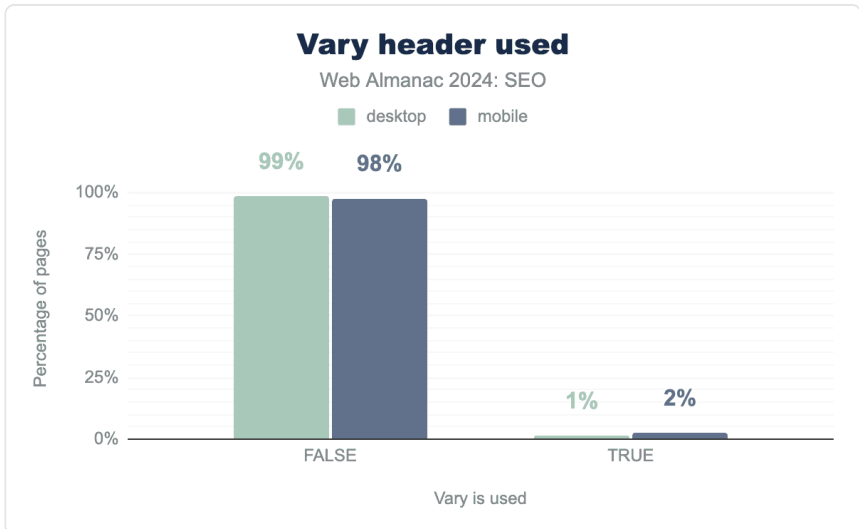


Figure 7.19. Vary header used.

Vary header usage fell significantly in 2024. The header appeared on 12% of desktop and 13% of mobile pages in 2022. It is now down to 1% of desktop and 2% of mobile. The decrease is likely due to Google specifically stating how dynamic rendering is not a sustainable long-term solution¹⁶⁷ for problems related to JavaScript-generated content. Instead, the search engine recommends single solution rendering strategies such as server-side rendering¹⁶⁸, static rendering¹⁶⁹, or hydration¹⁷⁰ as solutions.

Legible font sizes

One of the basics of a good mobile experience is being able to easily read the on-page content. Font sizes under 12 pixels require mobile visitors to “pinch to zoom” when reading content. This is considered too small to be legible.

167. <https://developers.google.com/search/docs/crawling-indexing/javascript/dynamic-rendering>

168. <https://web.dev/articles/rendering-on-the-web#server-side>

169. <https://web.dev/articles/rendering-on-the-web#static>

170. <https://web.dev/articles/rendering-on-the-web#rehydration>

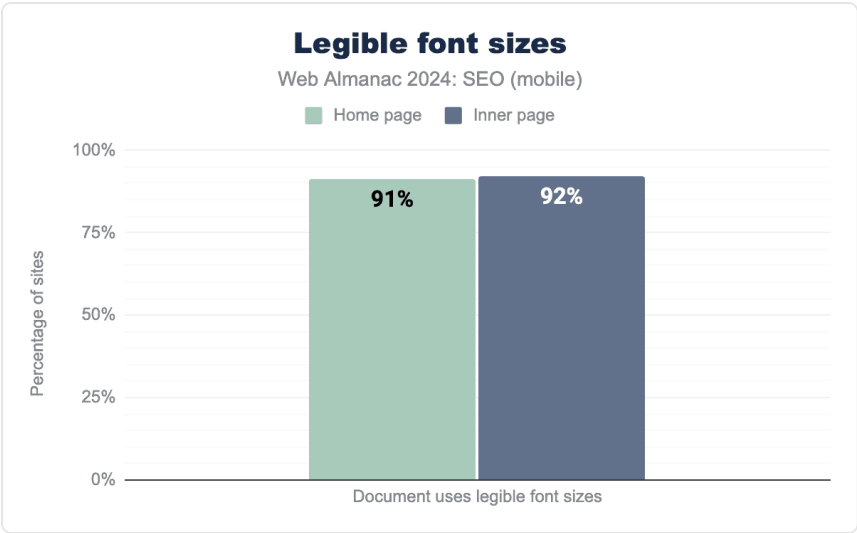


Figure 7.20. Legible font sizes.

Lighthouse has a legible font size audit¹⁷¹ that is run as part of with the HTTP Archive crawl. The audit checks for pages that had 60% or more of its content using fonts greater than 12 pixels. The test, specific to mobile pages, saw 92% of eligible pages passing. This percentage was consistent for both home pages and inner or inner pages.

Core Web Vitals

Core Web Vitals (CWV) are a series of standardized metrics to help measure how a user experiences a page. Google first introduced them as a ranking factor with the page experience ranking signal¹⁷² This independent signal was deprecated when it was absorbed into the Helpful content system¹⁷³, which has since been folded into the core algorithm.

Core Web Vitals are designed to answer three human-centric questions related to performance:

1. *Is the page loading?* Largest Contentful Paint¹⁷⁴ (LCP).
2. *Is the page interactive?* Interaction to Next Paint¹⁷⁵ (INP).
3. *Is the page visually stable?* Cumulative Layout Shift¹⁷⁶ (CLS)

171. <https://developer.chrome.com/docs/lighthouse/seo/font-size/>
172. <https://developers.google.com/search/blog/2020/05/evaluating-page-experience>
173. <https://developers.google.com/search/docs/appearance/ranking-systems-guide#helpful-content>
174. <https://web.dev/articles/lcp>
175. <https://web.dev/articles/inp>
176. <https://web.dev/articles/cls>

Core Web Vitals is measured via the page loads of real Chrome users across millions of websites and available via a public dataset, the Chrome User Experience Report¹⁷⁷ (CrUX).

These metrics are designed to evolve. In March 2024¹⁷⁸, Interaction to Next Paint¹⁷⁹ (INP) took over as the main measurement for interactivity from the previous metric, First Input Delay (FID), which only measured the input delay of the first interaction on a page. FID was an inaccurate measurement for a number of reasons, and many sites (particularly JavaScript-heavy sites) often falsely represented that they provided good interactivity to users. As a result, many JavaScript frameworks have seen their pass rate drop in 2024 because of this change. It should be noted, however, that currently SPAs are not accurately measured by Core Web Vitals¹⁸⁰.

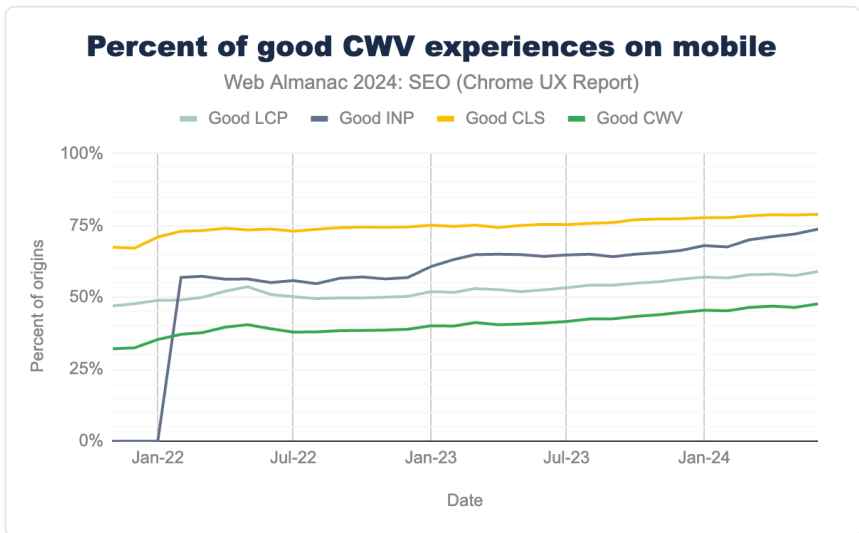


Figure 7.21. Percent of good CWV experiences on mobile.

CWV assessment is divided into mobile and desktop. In 2024, 48% of mobile sites passed. The percentage of passing sites has increased each year, with 39% in 2022, 29% in 2021, and just 20% in 2020.

Looking at the metrics individually, 59% pass Largest Contentful Paint, 74% pass Interaction to Next Paint, and 79% of mobile sites pass Cumulative Layout Shift.

177. <https://developer.chrome.com/docs/crux>

178. <https://web.dev/blog/inp-cwv-march-12>

179. <https://web.dev/articles/inp>

180. <https://web.dev/articles/vitals-spa-faq>

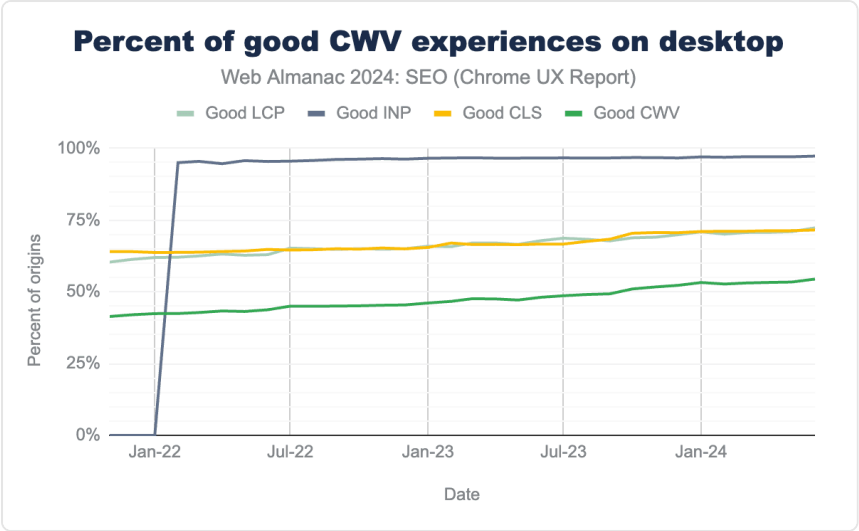


Figure 7.22. Percent of good CWV experiences on desktop.

Desktop devices offer more leniency, as they often have higher bandwidth connections. To be sure, 54% of sites pass the desktop CWV assessment, which is 8% more than mobile. The individual metrics show much higher pass rates as well, with 72% passing LCP, 97% passing INP, and 72% passing CLS.

You can explore the Performance chapter to learn more about Core Web Vitals.

Image loading property usage

Images are a critical component when it comes to page load. Image loading properties help browsers prioritize fetching resources as they build a web page. Implementing image loading properties can benefit user experience and performance. Downstream improvements may also include improved conversions and SEO success.

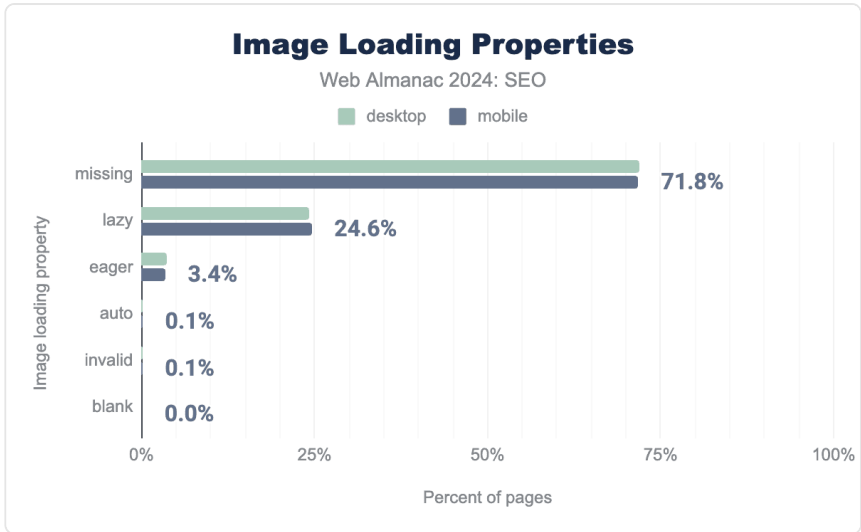


Figure 7.23. Image loading properties.

Most sites do not use these valuable signals, with 71.9% of desktop pages and 71.8% of mobile pages missing image-loading properties. The most adopted attribute was `loading="lazy"`. Lazy loading¹⁸¹ is a technique that defers the loading of non-critical elements on a web page until they are needed. This helps reduce the page weight and conserves bandwidth and system resources. This tag was used for 24.6% of mobile pages and 24.3% of desktop pages in 2024. The increased adoption can likely be attributed to `loading` attributes becoming a web standard.

The counterpart to `lazy` loading is `eager` loading. A browser `eager` loads images by default. Therefore, an image with the `eager` attribute and an image without any loading attribute will behave the same. In 2024, `eager` loading was the second most used property, but only appeared on 3.4% of mobile pages and 3.6% of desktop pages.

181. <https://web.dev/articles/browser-level-image-lazy-loading>

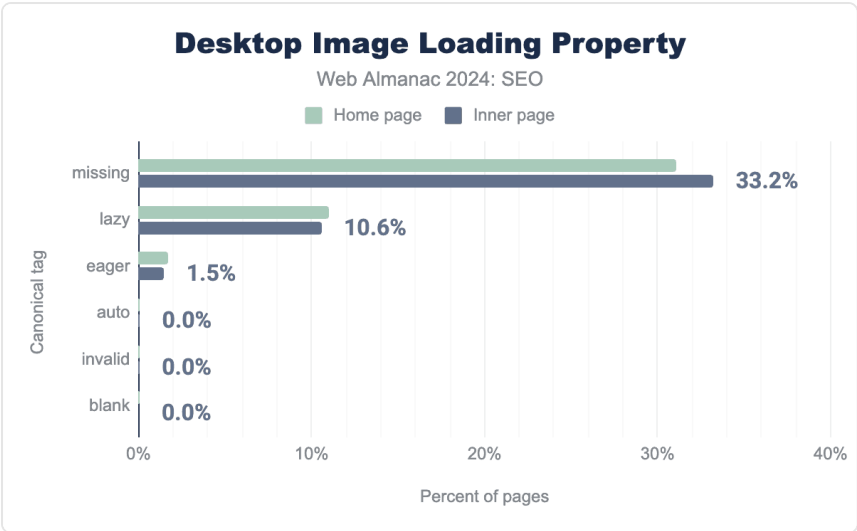


Figure 7.24. Image loading properties on home vs. inner pages.

A third deprecated value, `auto`, was never standardized and has since been removed from Chrome support. It is now considered an invalid value and ignored.

Lazy loading vs. eager loading iframes

Like images, iframes can also lazy-loaded through the `loading` attribute. Similarly to `img` loading attributes, `auto` is invalid and ignored.

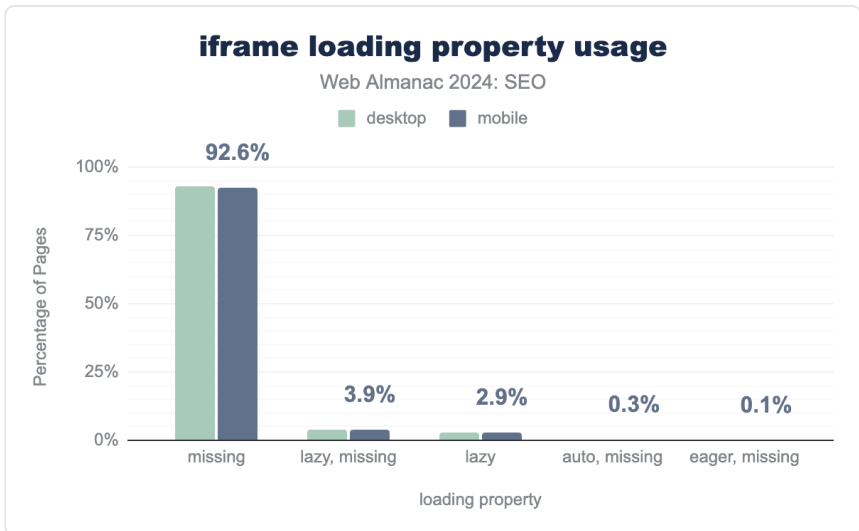


Figure 7.25. `iframe` loading property usage.

Of the sites containing one or more `<iframe>` elements, 92.8% of desktop and 92.6% did not declare a loading property. `lazy` was the most prominent declaration and most often occurred when there were multiple `<iframe>` elements on the page. We found 4.0% of desktop and 3.9% of mobile pages had a mixed `lazy` loaded and `<iframe>` elements without a declaration. Additionally, 2.6% of desktop and 2.9% of mobile pages used the `lazy` attribute on all `<iframe>` elements discovered during the crawl.

In 2022, 3.7% of desktop and 4.1% of mobile pages used the `lazy` loading attribute. The attribute increased to 6.6% of desktop and 6.9% of mobile pages in 2024.

Since `<iframe>` elements can be controlled by either the site on which the page is hosted or a third-party service, the prevalence of loading attribute combinations suggests that sites are adopting loading attributes wherever possible. It is reasonable to assume that third-party controlled `<iframe>` elements are less likely to have attributes.

On-page

When determining which pages to return in a search engine results page (SERP), search engines consider the on-page content as the primary factor. There are various SEO on-page elements that impact a search engine's understanding of content and its relevance to a user query.

Metadata

On-page content is the main measurement of a page’s relevance to a particular query. Certain HTML elements, such as `<title>` elements and `<meta>` descriptions, may appear in a Search Engine Results Page (SERP), but they are often just used as signals about the page’s content.

In 2021, Google started rewriting more websites’ `<title>` tags in their search results. As search engines have become less likely to use the direct content from these tags, their adoption rates have decreased.

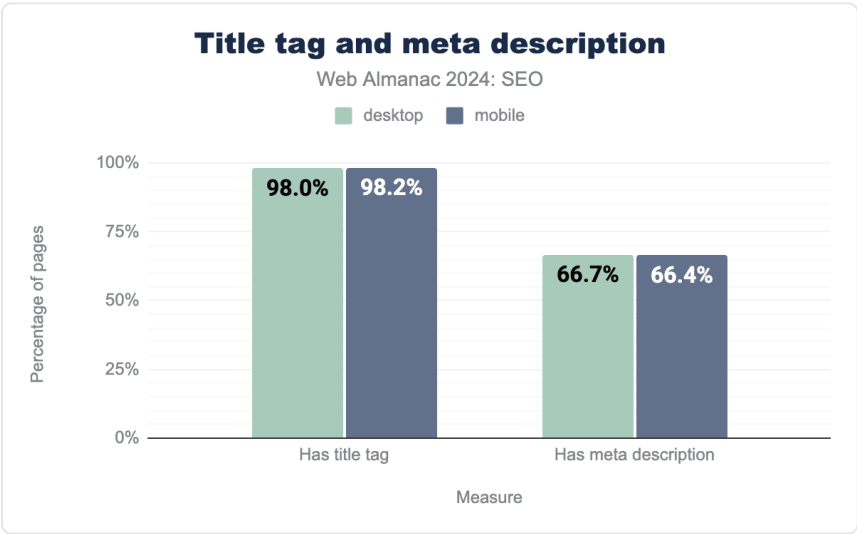


Figure 7.26. Title elements and meta descriptions.

In 2022, 98.8% of desktop and mobile pages used the `<title>` tag. Now in 2024, 98.0% of desktop pages have a `<title>` tag and 98.2% of mobile pages have one. Similarly, `<meta description>` usage dipped from 71% of desktop and mobile home pages in 2022 to 66.7% of desktop and 66.4% of mobile home pages in 2024.

`<title>` element

The `<title>` element populates the name displayed in a browser tab, and is one of the strongest on-page elements related to the page’s content and a query’s relevance.

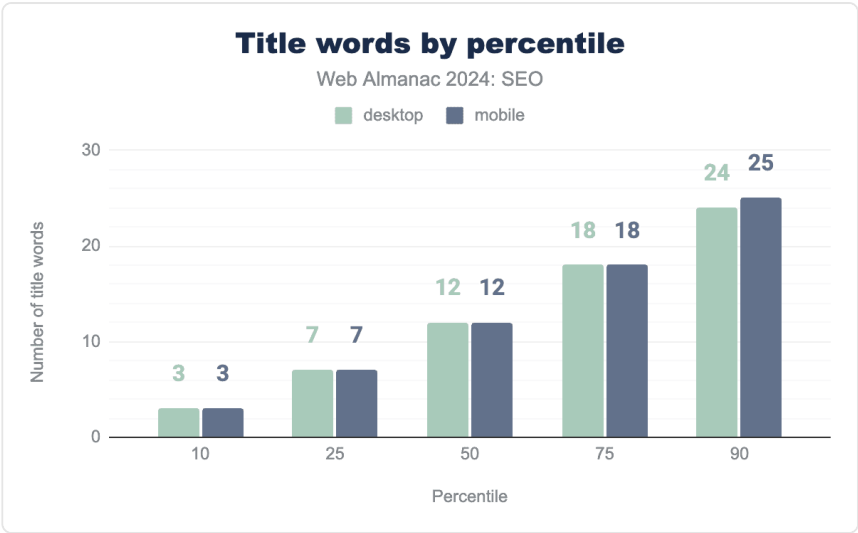


Figure 7.27. Title words by percentile.

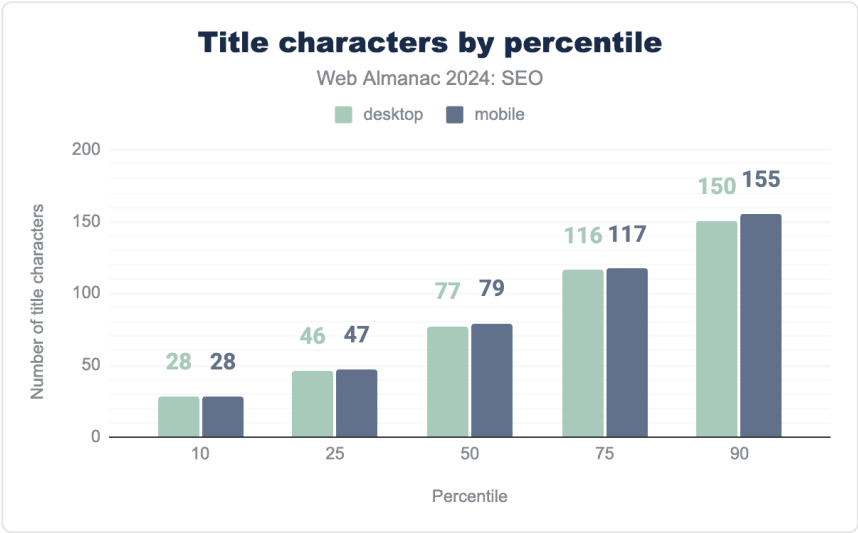


Figure 7.28. Title characters by percentile.

The word count in `title` elements was consistent between mobile and desktop experiences, though the character count was slightly higher for mobile with 79 characters compared to 77 words for desktop at the median.

Meta description tag

While the `<meta name="description">` tag is not a ranking factor, for some search engines and queries, the content within this tag may appear in SERPs and influence click-through rate.

Today, search engines like Google primarily create snippets to display in the SERPs from on-page content, based on the query. One study showed that 71% of meta descriptions are rewritten for the first page of results.

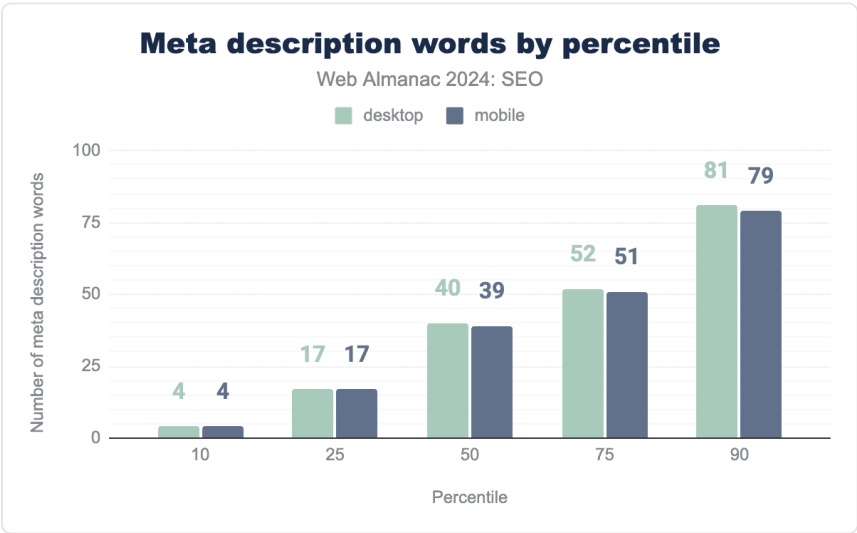


Figure 7.29. Meta description words by percentile.

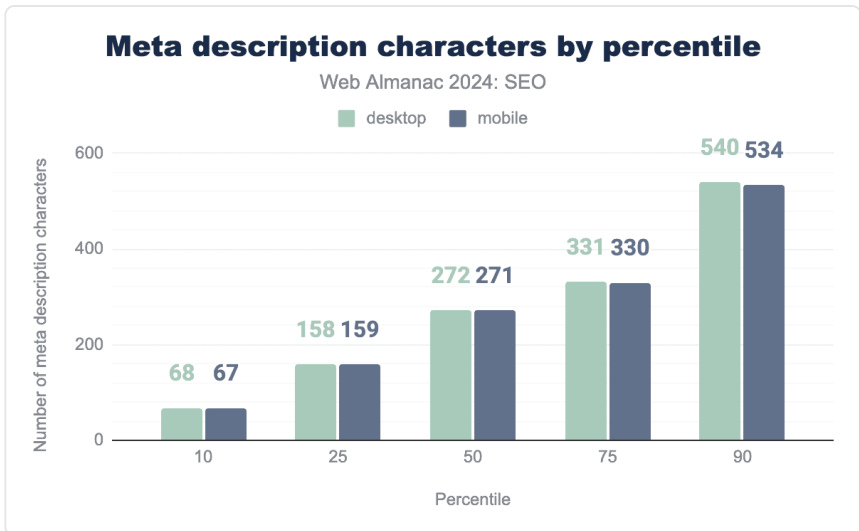


Figure 7.30. Meta description characters by percentile.

In 2024:

- The median desktop and mobile page `<meta name="description">` tag contained 40 and 39 words, respectively. That represents a 110% increase in word count for mobile and 105% increase for desktop since 2022. Two years ago, the median for both desktop and mobile was just 19 words.
- The median desktop and mobile page `<meta name="description">` tag contained 272 characters and 271 characters, respectively. That's a 99% increase for both device types compared to 2022.
- At the 10th percentile, the mobile and desktop `<meta name="description">` tag contained 4 words.
- At the 90th percentile, the `<meta name="description">` tag contained 81 words on desktop and 79 words on mobile.

Header elements

Header elements are used to establish the semantic structure of a page. They are important to a search engine's understanding of a page since they help organize the page's content. They follow a hierarchical order with `<h1>` used to describe the overall on-page content and subheaders such as `<h2>`, `<h3>` and so on to describe sections and subsections.

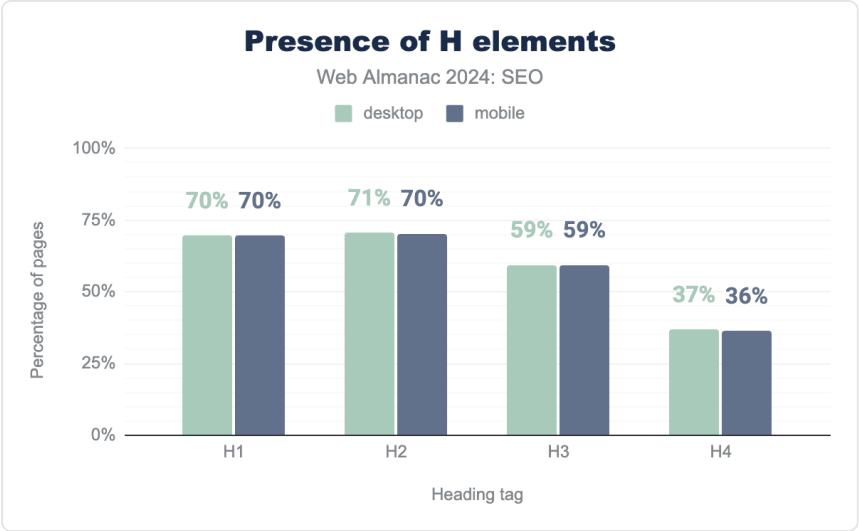


Figure 7.31. Presence of H elements.

Header elements have been widely adopted because they're easy to implement and help improve understanding for users and bots. For desktop pages in 2024, 70% had `<h1>` tags, 71% had `<h2>` tags , 59% had `<h3>` tags, and 37% had `<h4>` tags. Mobile pages were similar at 70%, 70%, 59%, and 36%, respectively.

These numbers vary slightly from 2022. Notable shifts in 2024 include increased adoption of `<h1>` tags, which in 2022 was at 66%. The subsequent headers, however, saw decreased usage. The `<h2>`, which was 71% in 2024 , was previously at 73% in 2022 for both device types. Meanwhile, the `<h3>` and `<h4>` tags, which were 59% and 37% in 2024, were higher in 2022 at 62% and 38%, respectively.

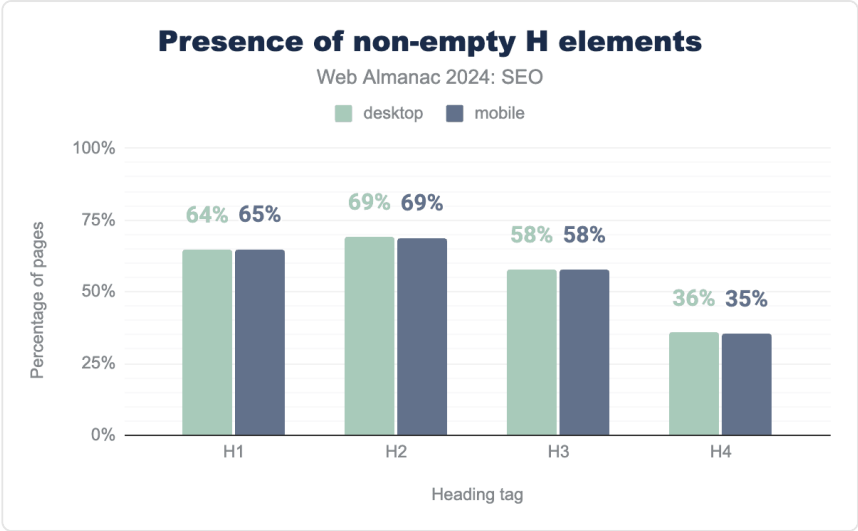


Figure 7.32. Presence of non-empty H elements.

In a continued trend from prior years, relatively few header elements are left empty. The most common empty header elements is the desktop `<h1>` at 6%.

Images

Images make the web a richer experience. The median page features 14.5 images (with marginal differences between device types). Image use is notably higher for home pages, which have an average of 18.5 images compared to 10.5 images on inner pages.

19

Figure 7.33. images on the median desktop home page

For more information on image use and weight, see the Page Weight chapter.

`alt` attributes

The image `alt` attribute provides information about the image for those who, for whatever reason, cannot view it. Its primary purpose is accessibility. Search engines also use the tag for indexing images since they can provide useful content. Therefore, alt tags are considered when

serving and ranking images in search engine result pages.

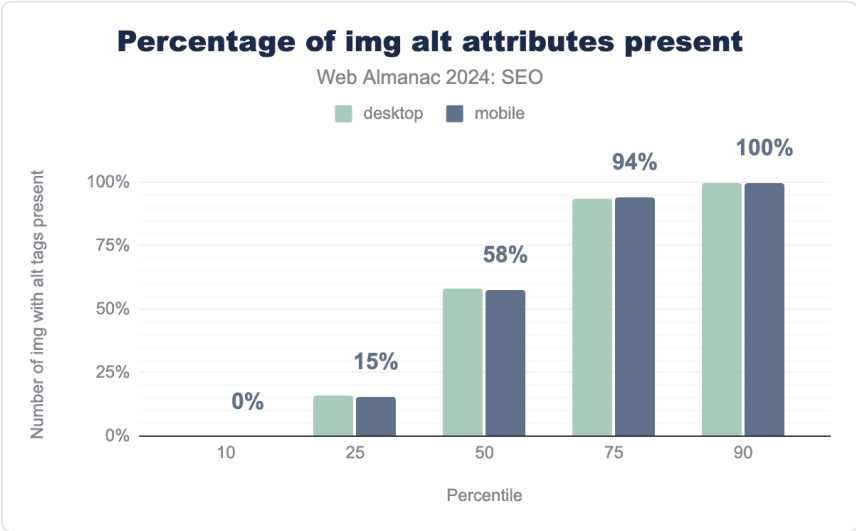


Figure 7.34. Percentage of `<img>` with `alt` tags present.

The median mobile site in 2024 had `alt` attributes on 58% of its images. That’s slightly up from 2022 when 54% of mobile pages used alt tags.

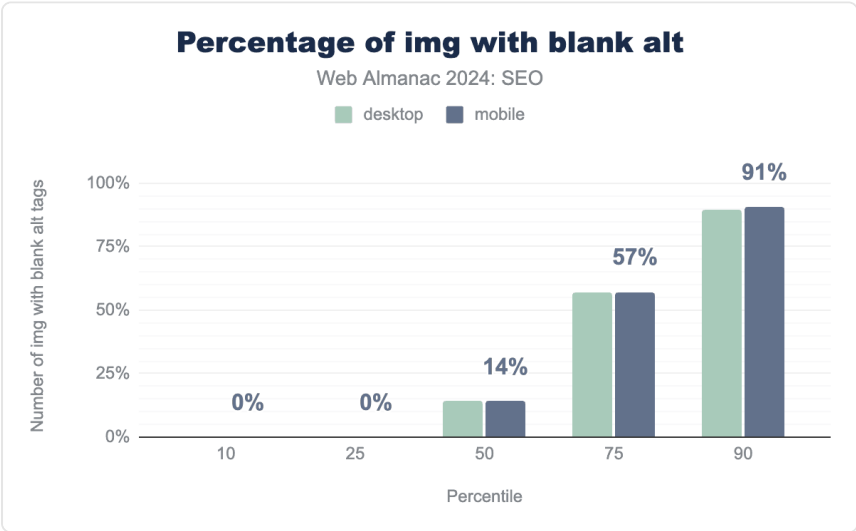


Figure 7.35. Percentage of `<img>` with blank `alt` tags.

There was an increase in blank `alt` attributes in 2024. Two years ago, the median page had 0% blank `alt` attributes. Now the median is 14% of desktop pages and 14% of mobile that are left blank. At the 75th percentile, 57% of desktop and 57% of mobile pages saw blank `alt` attributes.

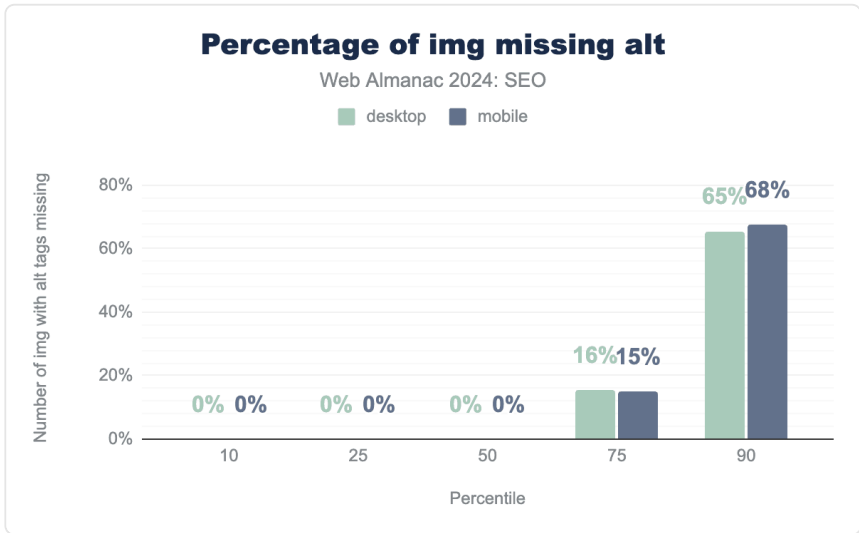


Figure 7.36. Percentage of `img` with missing `alt` tags.

In 2024, the median page saw no missing `alt` attributes on either mobile or desktop pages. This is a noteworthy drop from 2022 when 12% of desktop pages and 13% of mobile pages were missing the attributes. At the 75th percentile, 16% of desktop pages and 15% of mobile pages did not include `alt` attributes. In 2022, this was 51% and 53%, respectively.

The decrease in missing `<img> alt` attributes combined with the increase in blank attributes suggests that more CMS instances may be including an `alt` attribute for each image.

Video

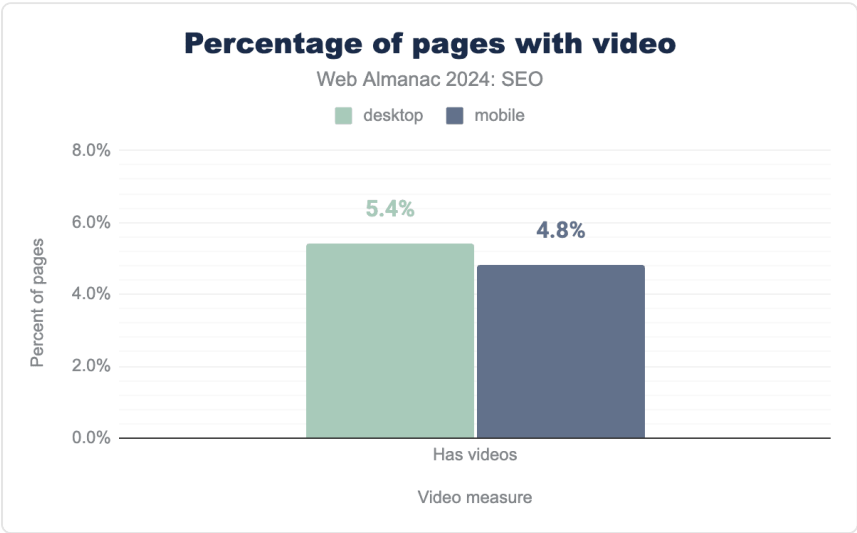


Figure 7.37. Percentage of pages with video.

Only 0.9% of pages had the `VideoObject` structured data markup in 2024. While this is more than double the 0.4% rate of 2022, it means there is still a significant gap between the percentage of pages that have video and those that have video and schema for it.

Links

Links on a page are used by search engines in a number of important ways.

One of the methods that search engines employ to discover new URLs for crawling, for example, is by finding a link targeting it from a page that they're already crawling and parsing.

Search engines also use links for ranking. Links serve as a proxy for how important and relevant a particular URL might be, based on the links targeting it. This is the basis of PageRank¹⁸², an algorithm on which Google was built.

When it comes to links, it is not a simple case of more links equals better ranking. There's a lot more nuance to it. These days, links are less of a factor when it comes to ranking. Search engines have evolved to better detect and rank great content, irrespective of links and, at the

182. <https://wikipedia.org/wiki/PageRank>

same time, to combat manipulation and link spam¹⁸³.

Non-descriptive links

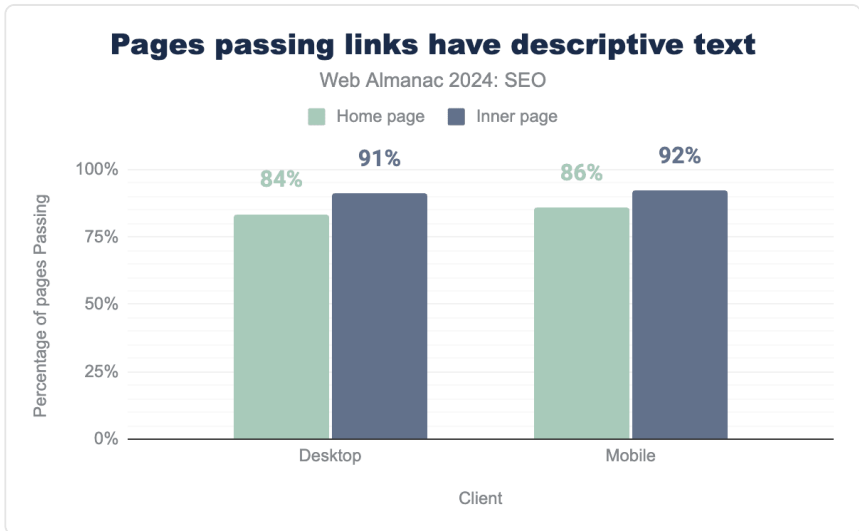


Figure 7.38. Pages passing links have descriptive text lighthouse test

The anchor text of a link, which is the hyperlinked text you click on, helps both users and search engines understand the content of the targeted linked page. Non-descriptive anchor text, such as 'More Info' or 'Click Here', doesn't have any inherent or contextual meaning, and is a lost opportunity from an SEO perspective. It is also bad for accessibility and those who use assistive technologies.

Lighthouse can detect whether there are non-descriptive links on a page. In 2024, 84% of desktop and 91% of mobile pages passed this test. And for inner pages, it was 86% of desktop pages and 92% of mobile pages that passed.

There was a small discrepancy between desktop and mobile, indicating perhaps that on mobile pages there may generally be less poorly indicated call to action links and 'click here' or 'read more' additional links to content, as they might be supplemental to other links on the page to the same target.

183. <https://developers.google.com/search/docs/essentials/spam-policies#link-spam>

Outgoing links

Outgoing links are `<a>` anchor elements that have an `href` attribute linking to a different page.

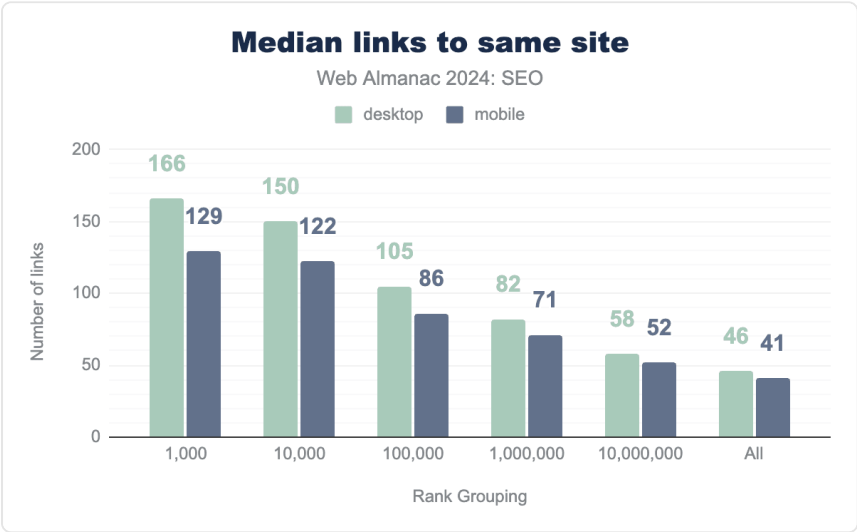


Figure 7.39. Median number of links to same site.

Internal links are links to other pages on the same website. The trend continues from 2022 in which desktop pages have more internal links than mobile pages do. This is most likely attributed to developers minimizing the navigation menus and footers on mobile for ease and to accommodate the smaller screens.

Overall, the number of internal links on a page have grown, with pages in the top 1,000 sites now having 129 internal links on mobile compared to 106 internal links in 2022. There has been a similar level of growth across all rank groupings.

According to CrUX ranking data, it's clear that the more popular sites have more outgoing internal links. This might simply be because the more visited sites are bigger entities with more useful internal links, as well as their investment in developing 'mega-menu' type navigation to help them handle more pages.

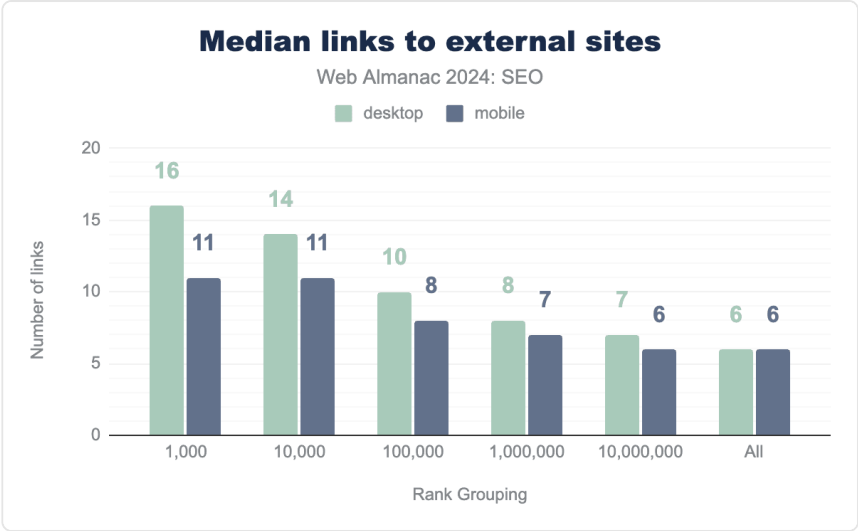


Figure 7.40. Median number of links to external sites.

External links are links that point to other websites. The link count has remained remarkably similar to that in 2022’s chapter¹⁸⁴. There is very slight growth, consisting of one or two links.

Similarly, the more popular sites tend to have more external links, but again the difference is very slight.

184. <https://almanac.httparchive.org/en/2022/seo#fig-35>

Anchor `rel` attribute use

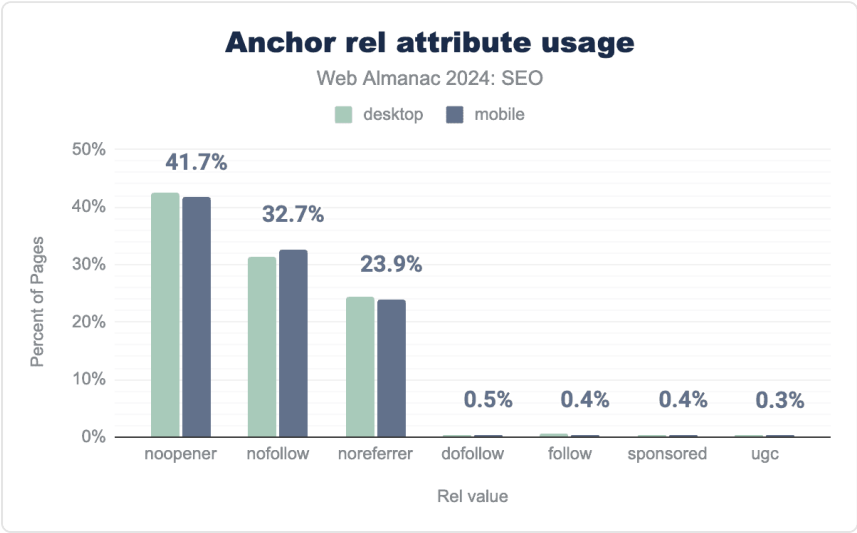


Figure 7.41. Anchor `rel` attribute usage.

The `rel` attribute dictates the relationship between the page and its linked target. For SEO, the primary use of the `rel` attribute is to inform search engines of its relationship with the page. Google terms this as qualifying outbound links¹⁸⁵.

The `nofollow` attribute, first introduced in 2005, is intended to inform search engines that you don't want to be associated with the targeted site nor wish them to crawl it based on links on your page. In 2024, the attribute was present in 32.7% of pages, up from 29.5% of pages in 2022.

Some more specific attributes were introduced in 2019, including `sponsored`, which denotes a link to sponsored content and `ugc`, which denotes links to user-generated content added by users (rather than publishers). Adoption of these attributes remains low. In 2024, it was just 0.4% for `sponsored` and 0.3% for `ugc`. Both were less popular than or equal to `dofollow` and `follow`, which actually aren't even real attributes and are ignored by search engines.

Interestingly, the most popular attribute is `noopener`, which is not related to SEO, and is just to prevent a page opened in a browser tab or window from accessing or having control over the original page. Additionally, `noreferrer`, which also has no effect on SEO, prevents the passing of the `Referer` HTTP header, so the target site doesn't know where the visitor came

185. <https://developers.google.com/search/docs/crawling-indexing/qualify-outbound-links>

from, unless unique tracking parameters are present in the link.

Word count

Search engines do not rank sites based on word count; however, it is a useful metric for tracking how much text sites contain, as well as a proxy for seeing how much site owners are leaning on client-side rendering to display the content for which they want to be found.

Home pages rendered word count

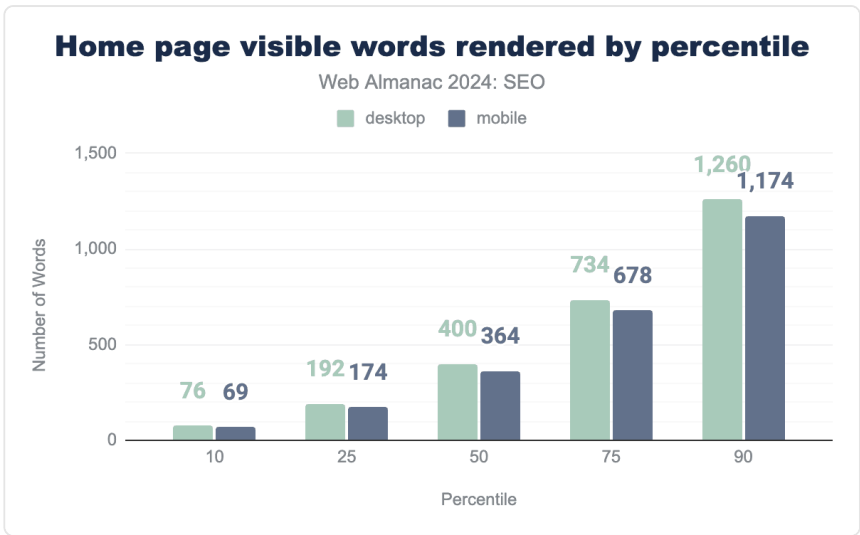


Figure 7.42. Home page visible words rendered by percentile.

Rendered word count is the amount of visible words on a page after JavaScript has been executed. The median mobile home page in 2024 contained 364 words, while the median desktop page had slightly more at 400 words. This was a small drop from the data in 2022 when the median was 366 words for mobile home pages and 421 words for desktop.

Of note, the gap between mobile and desktop home page word counts narrowed to just 36 words in 2024, compared to that of 55 words in 2022. This suggests a marginally closer parity to the content served to mobile users. Google has completed the process of moving to a mobile-first indexing strategy, in which it primarily crawls and indexes pages with a mobile user agent. It's reasonable to conclude that this helped push a few remaining sites to offer their full content to mobile visitors.

Inner pages rendered word count

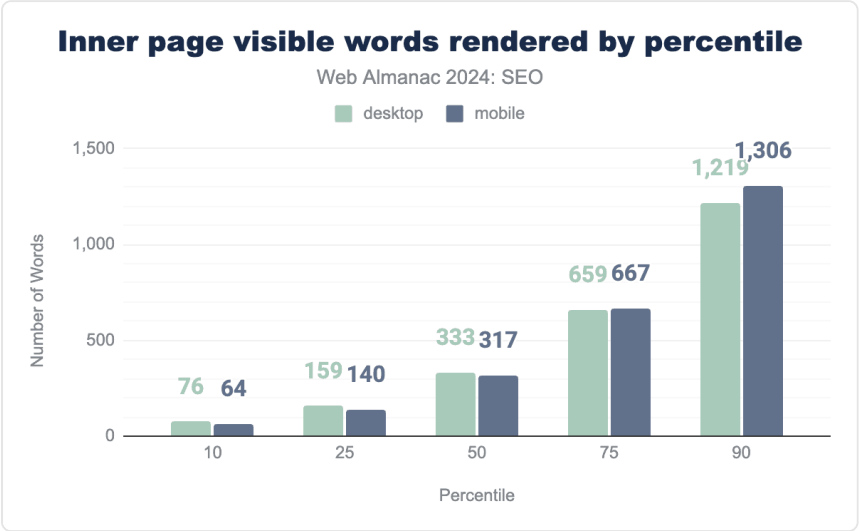


Figure 7.43. Inner page visible words rendered by percentile.

Inner pages contain slightly fewer words overall. The median mobile inner page in 2024 had 317 visible words after rendering, while desktop inner pages had 333 words.

One noticeable difference between home pages and inner pages is that while desktop pages generally have more words than mobile pages at the lower word counts, that gap narrows as the percentiles get higher. By the 75th percentile, for instance, mobile pages have more visible words on their inner pages than desktop's inner pages.

Home pages raw word count

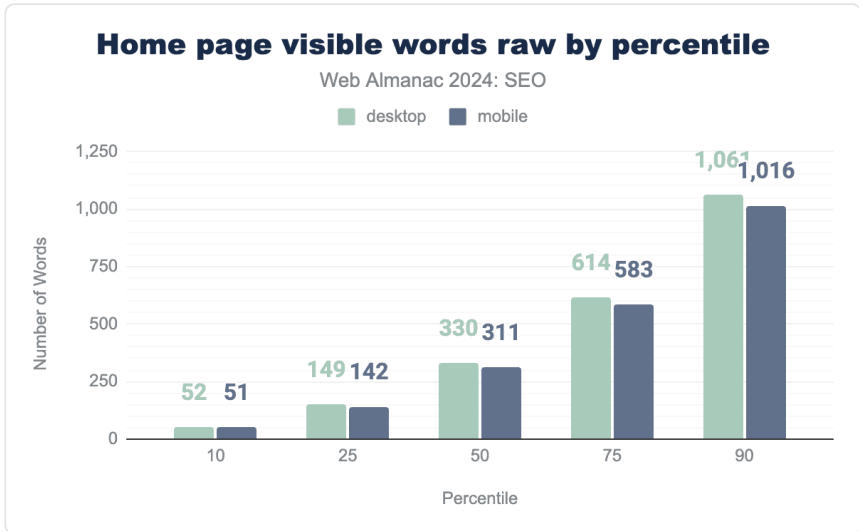


Figure 7.44. Home page visible words raw by percentile.

The raw word count represents the words contained in the initial HTML response from the server before JavaScript is executed and no other modifications have been made in the DOM or CSSOM.

Much like the rendered word count, there's similarly a small change in 2024 from 2022. The median page's raw word count in 2024 was 311 words for mobile user agents and 330 words for desktops. That's a tiny drop from 2022 when the median page's raw word count was 318 for mobile and 363 for desktop.

Inner pages raw word count

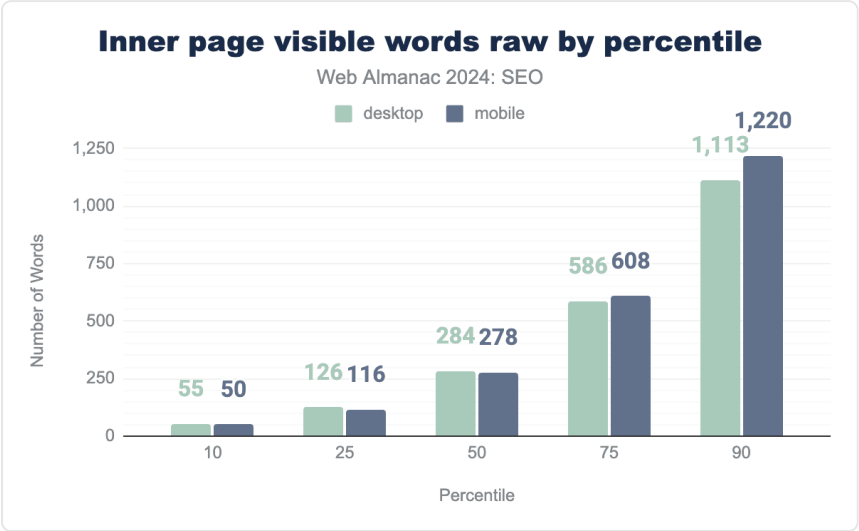


Figure 7.45. Inner page visible words raw by percentile.

Like home pages, the inner pages' visible words raw count very much follows the rendered word count figures, including mobile pages containing more words than desktop pages at the 75th percentile and above.

This pattern in both the raw word count and rendered word count pages suggests the trend is unrelated to infinite scrolling, which is a more popular choice for publishers on mobile layouts.

Rendered vs. raw home pages

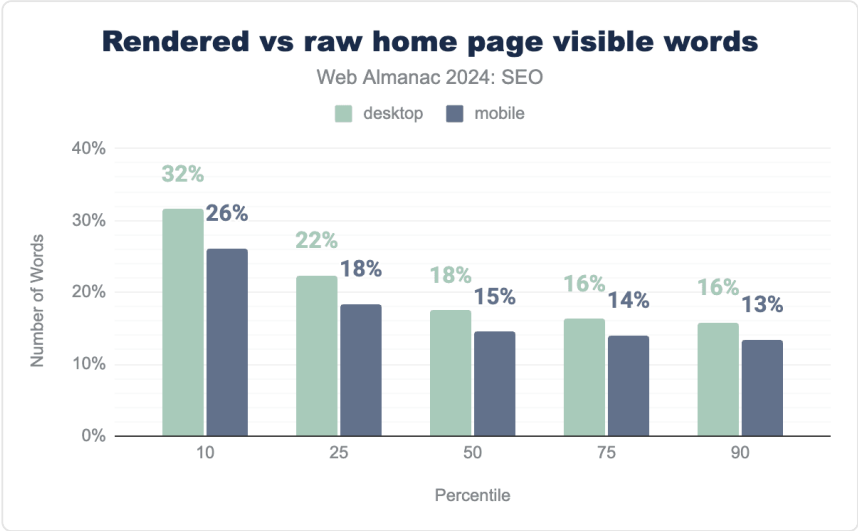


Figure 7.46. Rendered vs raw home page visible words.

When the rendered visible and raw word counts are compared on home pages, there's a surprisingly small discrepancy, with the median showing a difference of 13.6% for mobile and 17.5% for desktop.

Home pages served to desktop user agents have a slightly higher percentage of words visible after rendering versus mobile. One possible factor is that mobile layouts often employ tabs or accordions where, even if the content is in the DOM, it's visually hidden, so it wouldn't show up as visible.

There is an interesting trend in which the higher the word count there is, the smaller the difference between rendered and raw word count. This suggests perhaps that server-side rendered technologies are relatively more popular than client-side rendered ones for publishers of longer-form content.

Rendered vs. raw inner pages

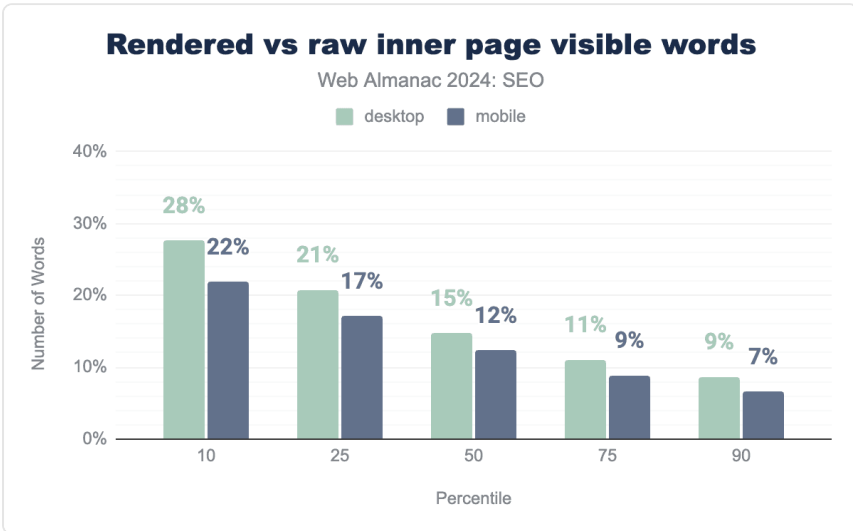


Figure 7.47. Rendered vs raw home page visible words.

Somewhat surprisingly, there is an even narrower gap between rendered and raw word counts on inner pages, which suggests they're less likely to contain significant amounts of client-side rendered content.

Although the gap is narrower, it does follow the same pattern of the more words, the less they rely on client-side rendering.

Structured data

Structured data remains important for optimizing many sites. While it is not a ranking factor, *per se*, it often powers rich results, especially on Google.

These enhanced listings often make a site or elements of one stand out. Additionally, correctly implemented structured data can, for example, surface in other search engines.

The addition of inner pages in this year's crawl is particularly relevant for structured data, since many types are only applicable to specific pages.

Home pages

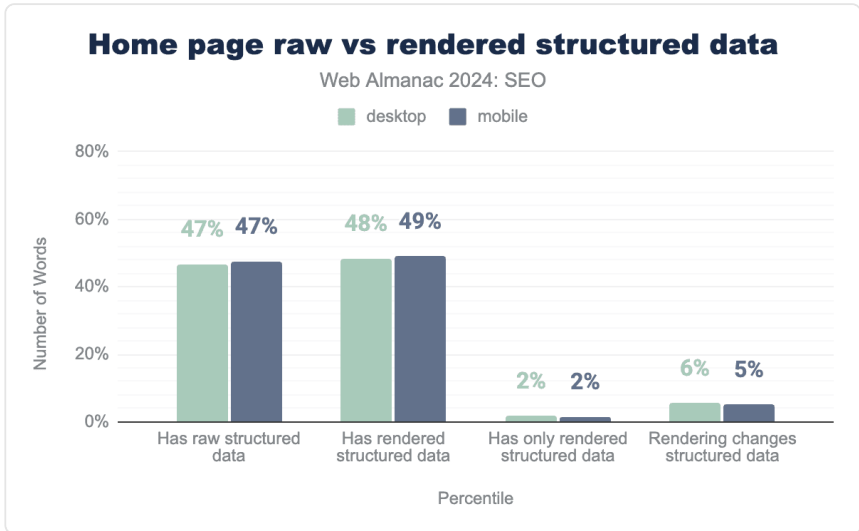


Figure 7.48. Home page raw vs rendered structured data.

Overall usage of structured data grew in 2024 to 49% of mobile home pages and 48% of desktop home pages. This was a slight increase from 2022 when 47% of crawled mobile home pages and 46% of crawled desktop home pages had structured data.

The majority of sites provide structured data in the raw HTML, while only 2% of mobile and desktop crawls to home pages have structured data added via JavaScript.

A few more home pages, 5% of mobile and 6% of desktop crawls, contained some structured data that had been altered or augmented by JavaScript.

The trend appears to be that of providing structured data markup in the raw HTML, something Google itself highlights¹⁸⁶ as, if not best practice, perhaps the simplest and most reliable way of implementing structured data.

186. <https://developers.google.com/search/updates#clarifying-dynamically-generated-product-markup>

Inner pages

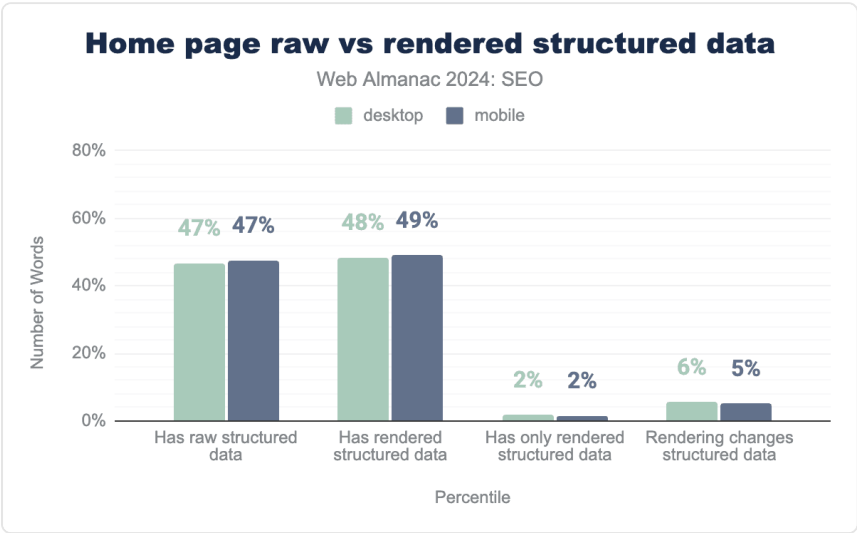


Figure 7.49. Inner page raw vs rendered structured data.

Inner pages, such as product or event pages, are more likely to have structured data. In 2024, 53% of mobile and 51% of desktop inner pages had some structured data markup. And it dovetails with the fact that there are a number of Google developer documents that detail eligibility for rich results, based on structured data.

Overall, the trend of providing the markup in the raw HTML carries across from what was seen on home page crawls.

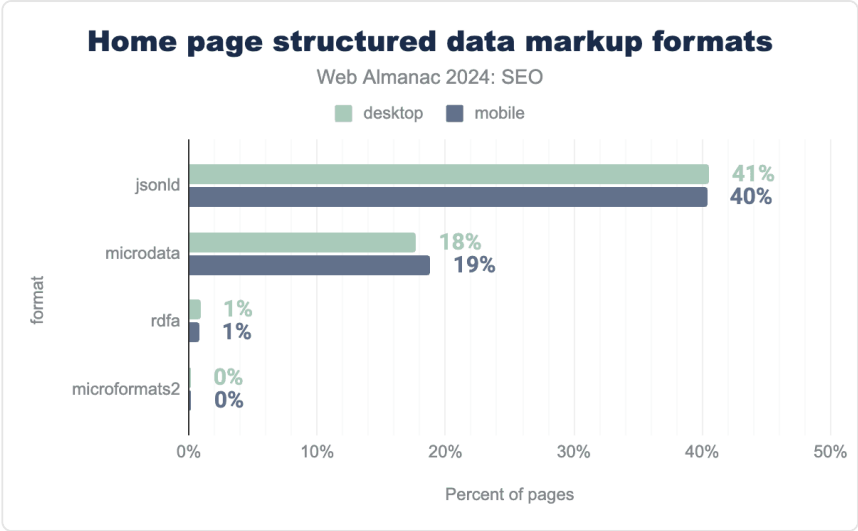


Figure 7.50. Home page structured data markup formats.

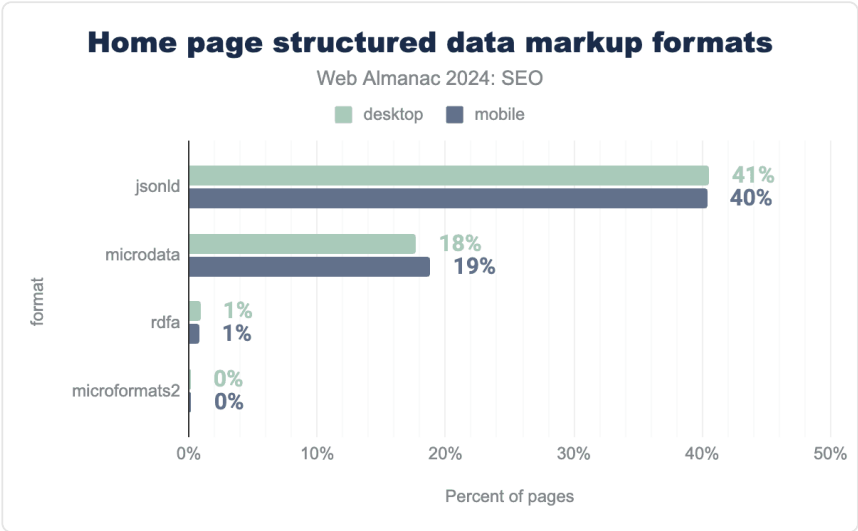


Figure 7.51. Inner page structured data markup formats.

There are a number of different ways structured data can be implemented on a page, but JSON-LD is by far the most popular format for home pages. It accounts for 40% of mobile and 41% of desktop home pages crawled.

It's simply the easiest format to implement, and is done by adding `<script>` elements that are

independent of the HTML structure. Other formats, such as Microdata, involve adding attributes to the HTML elements of the page. Since Google advises using JSON-LD as a preferred format, it is not surprising that this is the most popular implementation.

For the most part, inner pages similarly utilize JSON-LD, but there is a slight increase in the use of structure data with Microdata for those pages.

Most popular home pages structured data types

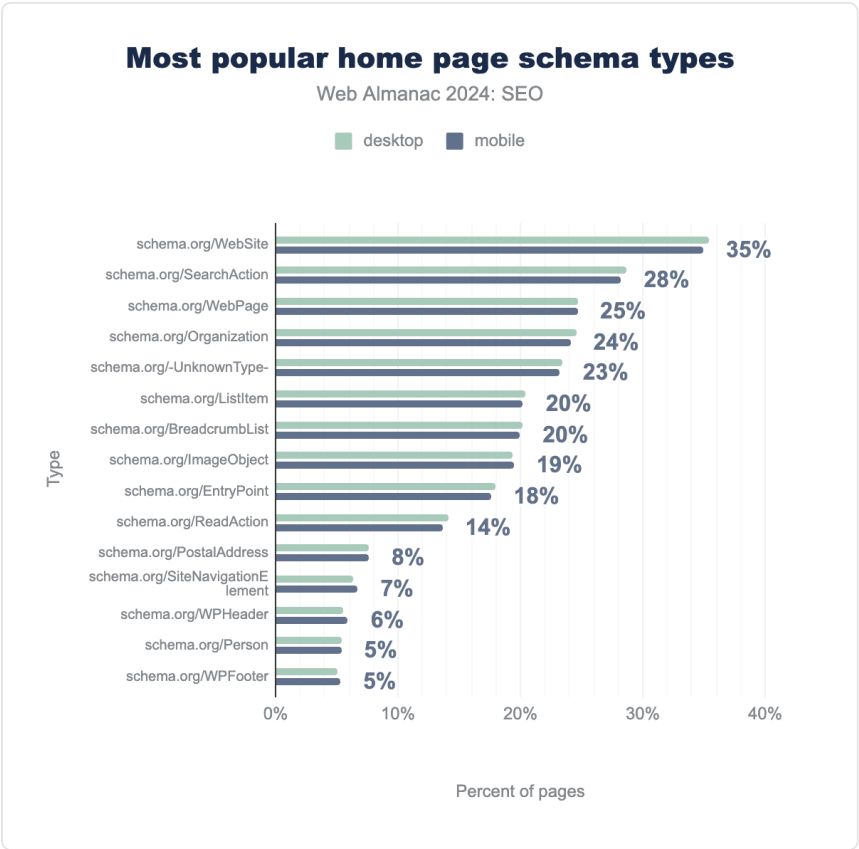


Figure 7.52. Most popular home page schema types.

Compared to 2022, there wasn't a big shift in 2024 in terms of the popularity of structured data types found on home pages. `WebSite`, `SearchAction`, and `WebPage` remained the three most popular schema types since they power the Sitelinks Search Box on Google.

Interestingly, `WebSite` grew a little more in 2024 to 35% of mobile home pages from 30% in 2022 since Google recommends this as a way to influence a site name¹⁸⁷ in the SERPs.

As for implementing the most popular schema types, there were minor differences between the percentages of mobile and desktop structured data usage.

Most popular inner pages structured data types

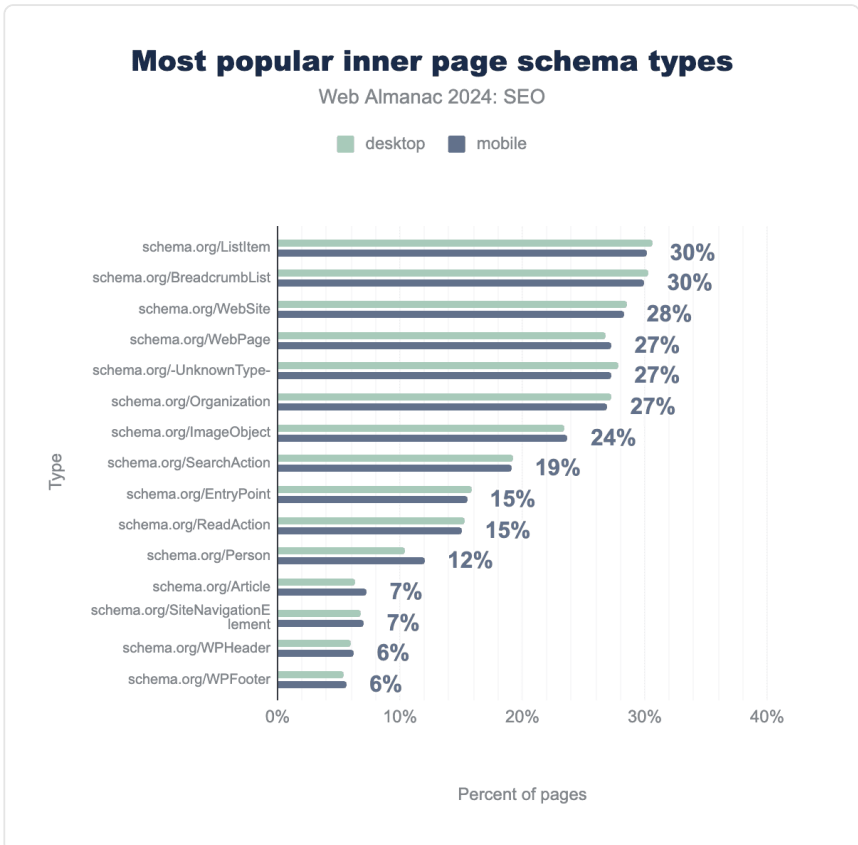


Figure 7.53. Most popular inner page schema types.

In terms of the inner pages, `ListItem` was the most popular schema type in 2024, representing 30% of mobile and 31% of desktop URLs. It stands to reason there would be more listing pages than “leaf” pages, such as product, event, or article pages (although `Article`

187. <https://developers.google.com/search/docs/appearance/site-names>

schema was the 12th most popular type).

`BreadcrumbList` was the second most popular schema type. That was to be expected since one would be more likely to show a breadcrumb on an inner page.

What is surprising is that `WebSite` structured data, which is, at least for Google, home page specific, was the third most popular schema type on inner pages. A possible explanation is that particular structured data type is often implemented at a site template level and carried across the entire site.

Outside of the more popular schema types, `product` structured data was found on 4% of mobile pages and 5% of desktop pages.

For a deeper dive into structured data on the web, visit the [Structured Data](#) chapter.

AMP

Accelerated Mobile Pages, known mostly by the acronym AMP, is a framework for building pages, particularly mobile pages, that offer solid performance. Though designed for mobile pages, it is entirely possible to build a website for all devices using AMP.

It has been, however, a somewhat divisive technology, with many feeling the burden of maintaining a separate version of a page. Additionally, there are some limitations to AMP in performance with which publishers and site owners have grappled.

While it is not a direct ranking factor, in the past certain features, including Top Stories in Google, were reliant on, or at least influenced by, having an AMP version.

Home page usage

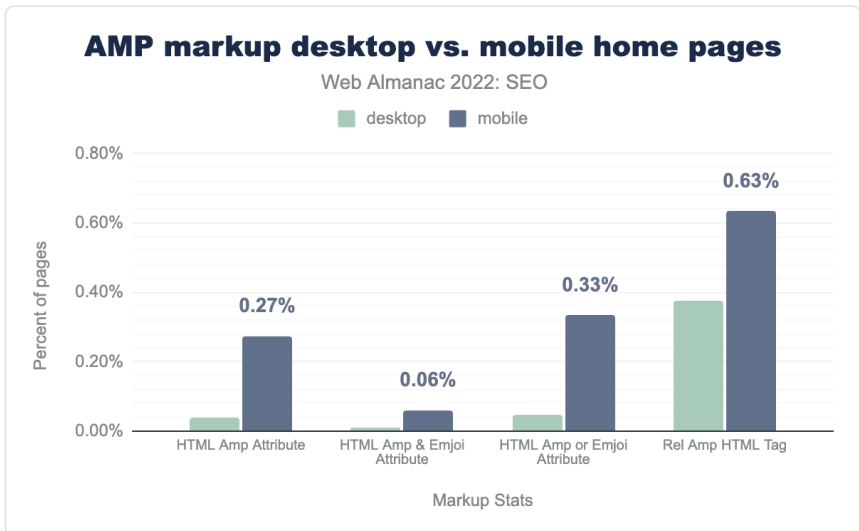


Figure 7.54. AMP markup on desktop vs. mobile home pages

With the advent of Core Web Vitals¹⁸⁸ (CWV), allowing the ability to quantify performance of non-AMP pages, the requirement for AMP to gain valuable real estate in search results, like Top Stories, has gone, as has much of the upside.

That's why it's a little surprising there was a slight uptick in the percentage of pages containing the `amp` html attribute. In 2024, it went up to 0.27% for mobile crawls compared to 0.19% in 2022. The desktop crawls, however, dropped to just 0.04%, down from 0.07% in 2022.

It's worth noting these figures are relatively tiny, so the changes might not be statistically relevant, but they do point to low adoption of the technology.

188. <https://web.dev/articles/vitals>

Home pages vs. inner pages

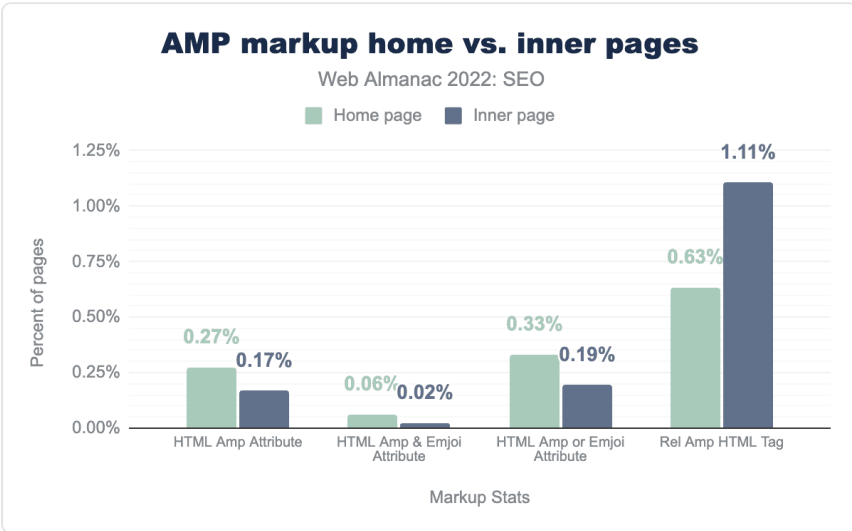


Figure 7.55. AMP markup home pages vs. inner pages.

Home pages are more likely when crawled to be AMP pages than inner pages, with 0.31% of home pages across mobile and desktop, and only 0.21% of inner pages having the HTML AMP attribute.

Internationalization

Internationalization is the process of optimizing a website to target multiple countries, languages, or regions, ensuring proper crawling and indexing by search engines. This involves employing best practices to deliver content to the correct audience.

Modern search engines like Google can determine a page's language from its visible content¹⁸⁹. Additionally, they can detect the language used in navigation elements.

Still, it can be confusing for search engines to identify the appropriate language, such as when an English course is targeted at a German-speaking audience. In that case, while the page content would be in English, the target audience would be German speakers in different countries.

189. <https://developers.google.com/search/docs/specialty/international/managing-multi-regional-sites>

Therefore, the main purpose of internationalization mechanisms (via HTTP headers, HTML, or sitemaps), such as `hreflang` tags or content-language attributes, is to avoid confusion and help search engines deliver content to the correct audience.

`hreflang` implementation

`hreflang` tags help search engines understand what the main language is on a particular page. Its SEO application is that different countries or regions can be targeted using the appropriate language across different (though related) websites.

The analysis of `hreflang` tag implementation reveals that 0.1% of websites still use the HTTP protocol within their `hreflang` tags both on desktop and mobile. This indicates that a small portion of internationalized websites have not yet adopted the HTTPS standard.

As a result, the use of HTTP can cause an inconsistency that may confuse search engines in their correct interpretation of page content.

Furthermore, there's a notable discrepancy between the raw and rendered versions of the tag. A difference of 0.1% exists on desktop (9.5% raw vs. 9.6% rendered) and 0.2% on mobile (9.1% raw vs. 9.2% rendered).

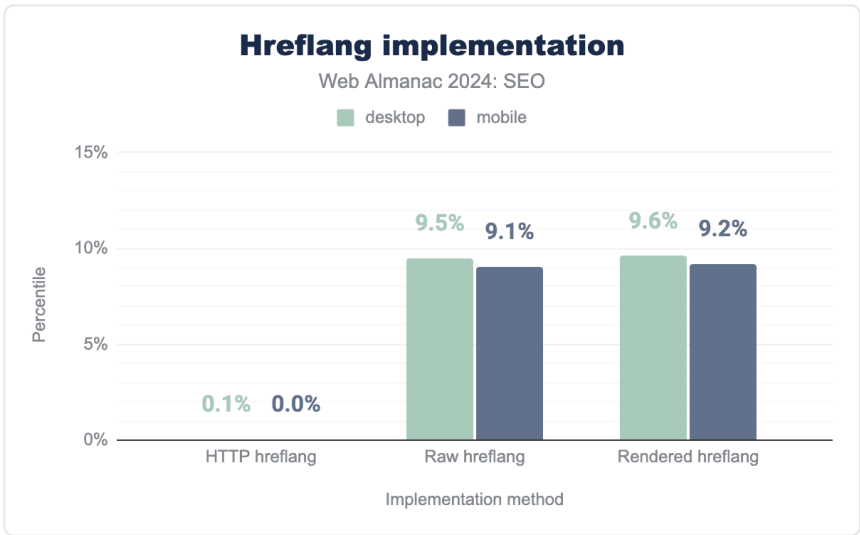


Figure 7.56. `hreflang` implementation.

The discrepancy between the “raw” and “rendered” versions of the `hreflang` tag indicates there are technical issues that are preventing the proper rendering of content, which affects

how search engines interpret it.

Even when these discrepancies are considered minor, highly trafficked websites and/or those containing essential information for the public (such as from international institutions, research institutes, universities...) may experience significant losses in visibility with their intended audiences.

Home page `hreflang` Usage

While search engines can often detect a page's language on its own, `hreflang` tags provide explicit signals to ensure content reaches its intended audience. These tags are typically used when a website has multiple language versions targeting different locales or regions.

Currently, 10% of desktop websites and 9% of mobile websites utilize `hreflang`. This represents a slight increase from 2022 when usage was 10% and 9% for desktop and mobile, respectively.

The most popular `hreflang` value in 2024 remained "en" (English), with 8% usage on desktop and 8% on mobile. That particular tag experienced considerable growth from 2022 when usage was 5% on desktop and 5% on mobile.

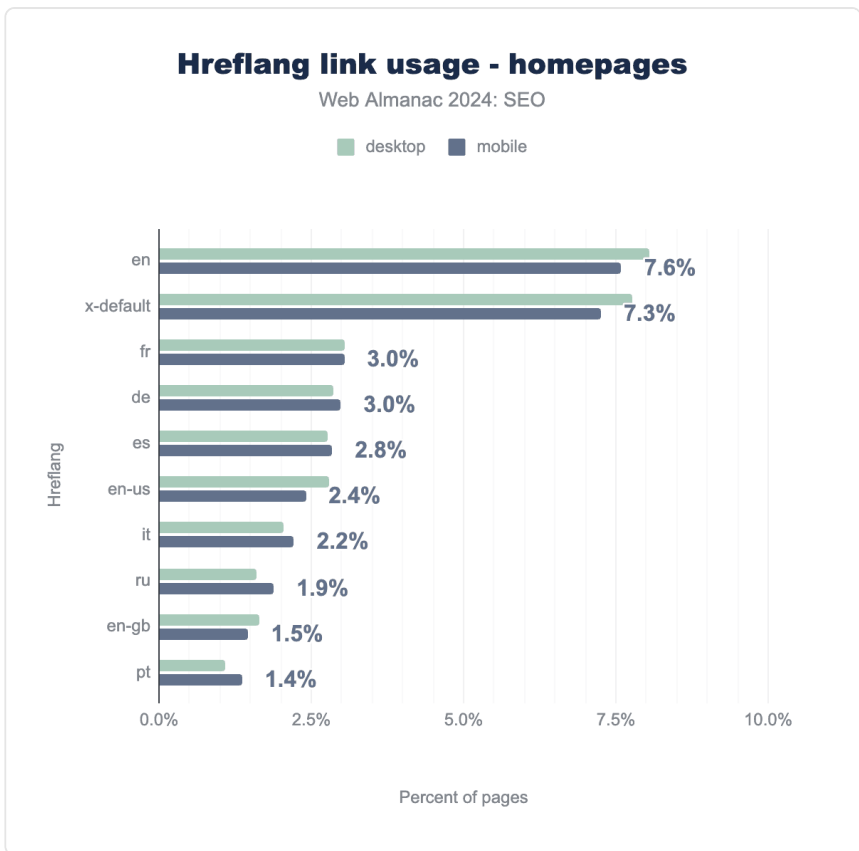


Figure 7.57. `hreflang` link usage for home pages.

The most common variations of en are en-us (American English) at 2.8% (desktop) and 2.4% (mobile) and en-gb (British English) at 1.7% (desktop) and 1.5% (mobile).

Following en, the x-default tag, which specifies the default language version, is the next most popular tag. After that, fr (French), de (German), and es (Spanish) are the most frequently used `hreflang` values, which is similar to the findings in 2022.

Inner page hreflang usage

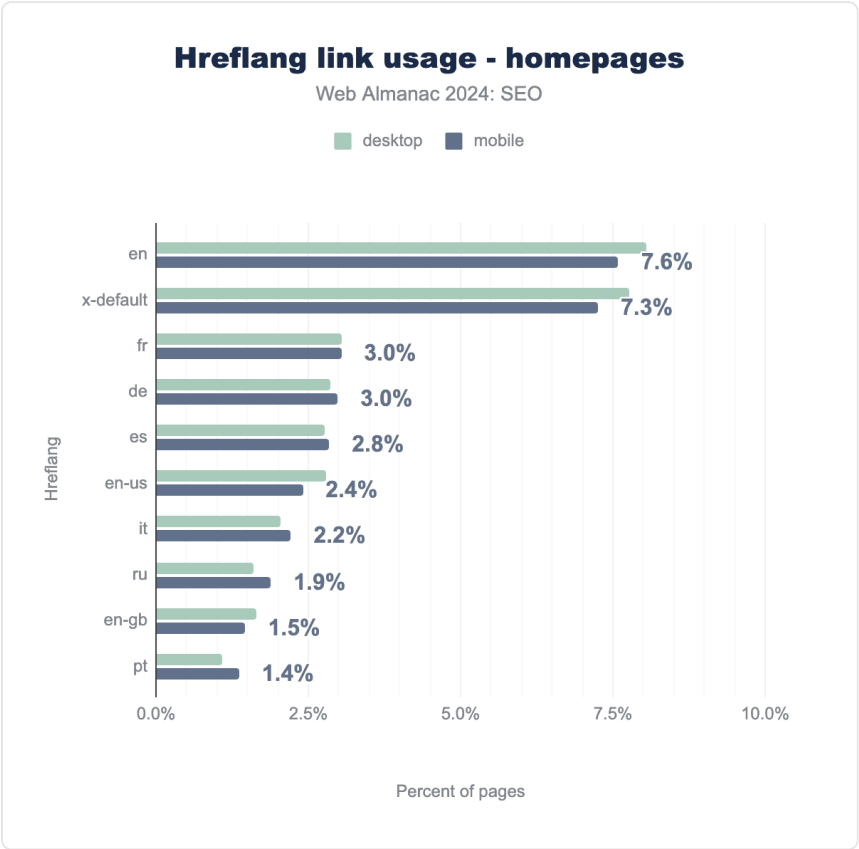


Figure 7.58. hreflang link usage on inner pages

The use of hreflang tags on inner pages has x-default (7.3%) and en (English, 7.1%) as the most common values. When the values are broken down between mobile and desktop, we get 8.0% for desktop and 7.3% for mobile for x-default, and 8.0% for desktop and 7.1% for mobile for en.

Desktop usage is slightly higher than mobile for most hreflang values on inner pages. The differentials are quite small. With the exception of fr, the other hreflang values (de, es, en-us, it, ru, en-gb, pt) have usage below 3.0% and show a degree of concentration in the most common values.

As for distribution, the use of hreflang tags on inner pages is similar to that found on home

pages. The x-default and en lead in adoption in both categories and underscores their global reach. Their percentages are lower on inner pages, which implies that `hreflang` implementation is generally prioritized on home pages.

Content language usage (HTML and HTTP header)

While search engines like Google¹⁹⁰ and >Yandex¹⁹¹ only employ `hreflang` tags, others also use the content-language attribute¹⁹², which can be implemented in two ways:

- HTM.
- HTTP Header

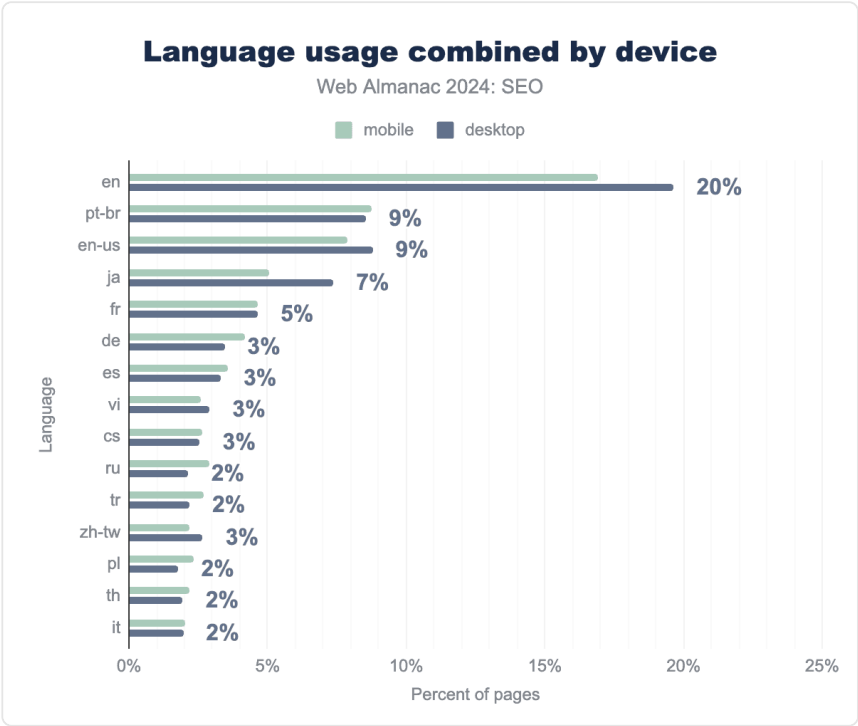


Figure 7.59. Language usage (html and http header) for mobile and desktop.

When examining language usage data for home pages and inner pages (the latter of which were

190. <https://developers.google.com/search/docs/specialty/international>
191. <https://webmasters.yandex.ru/blog/15326>
192. <https://developer.mozilla.org/docs/Web/HTTP/Headers/Content-Language>

not discussed in 2022), English (en) appeared as the main element (home page: 18% and inner page: 18%), followed by pt-br (home page: 9% and inner page: 9%), en-us (home page: 8% and inner page: 8%), and ja (home page: 6% and inner page: 6%).

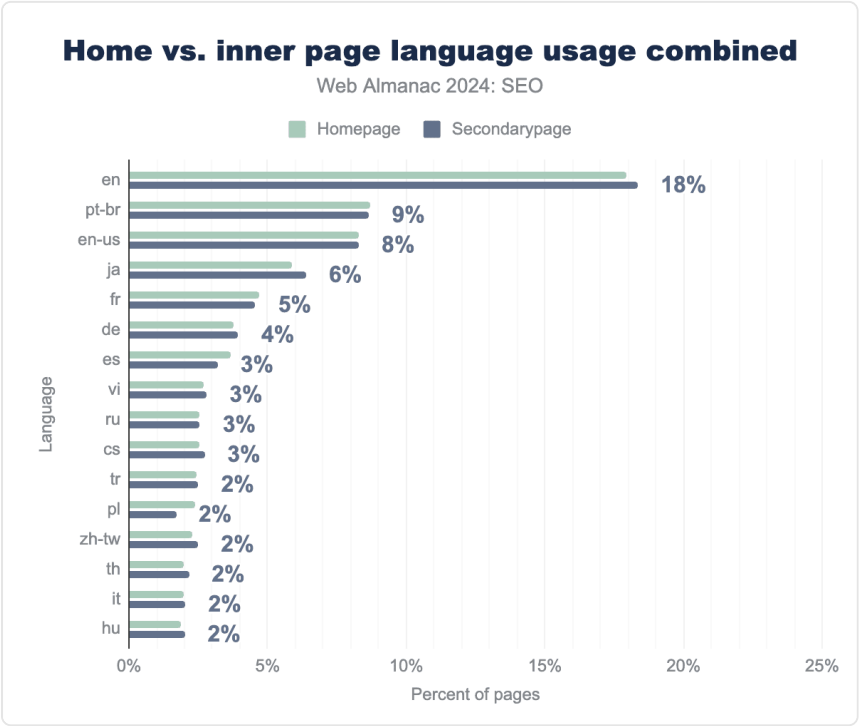


Figure 7.60. Language usage (html and http header) for home page and inner page.

Regarding other elements, the order followed almost the same pattern as the mobile and desktop comparison shown above.

When analyzing the resulting data for both graphs, the dominance of `en` suggests that a large proportion of content is still tailored to English speakers. The correlation appears to be a result of English not only being the most spoken language¹⁹³, but also widely used throughout global markets and a requisite for entrance to the powerful United States market (en-us).

Even though Mandarin is the second most spoken language in the world, the dominant search engine for this language, Baidu, does not require specific tags for locating Chinese websites. As a result, it presents a challenge when collecting data for the language. Still, `zh-tw` (the Chinese spoken in Taiwan) appears in the 13th position for language usage.

193. <https://www.statista.com/statistics/266808/the-most-spoken-languages-worldwide/>

Additionally, the growth of `pt-br` from 6th position in the 2022 mobile versus desktop comparison to second position is quite significant and may indicate a pursuit of audience gains in this language.

Conclusions

The two years between the last Web Almanac SEO chapter in 2022 and this year's edition may seem like a long time in SEO, which is often a fast-moving field. However, the data shows incremental changes to the fundamentals have been slow-moving.

The recent growth of the `IndexIfEmbedded` tag, for example, signals that perhaps certain practices and protocols need some time before there's mass adoption within the SEO industry.

That said, it has not been business-as-usual. The amount of sites passing Core Web Vitals (CWV) has been tremendous, despite Interaction to Next Paint (INP) replacing the arguably much easier to pass metric of First Input Delay (FID). That positive news signals how performance, in general, is being taken more seriously in the SEO industry.

Most notably, AI and LLMs are presenting some of the biggest changes search engines have encountered in a long time, and they have the potential to be hugely disruptive. As a result, adoption of `robots.txt`, related to the associated crawlers, has already grown.

The ever-changing search landscape and the new opportunities afforded by AI and LLMs have the potential for SEO to quickly move into new areas. At the same time, the slow but steady improvements to fundamentals underscore that the state of SEO remains one where long-standing best practices, despite large sea changes, are both prized and ultimately rewarded.

Authors



Jamie Indigo

@not-a-robot.com
 fellowhuman1101
 jamie-indigo
 <https://not-a-robot.com>

Jamie Indigo isn't a robot, but speaks bot. As director of technical SEO at Cox Automotive¹⁹⁴, they study how search engines crawl, render, and index the web. Jamie loves to tame wild JavaScript and optimize rendering strategies. When not working, they like horror movies, graphic novels, and terrorizing lawful good paladins in Dungeons & Dragons.

194. <https://www.coxautoinc.com/>



Dave Smart

🐦 @tamethebots.com 🔄 dwsmart 🔗 davevsmart 🌐 <https://tamethebots.com>

Dave Smart is a developer and technical search engine consultant at Tame the Bots¹⁹⁵. They love building tools and experimenting with the modern web, and can often be found at the front in a gig or two.



Mikael Araújo

✂ @miknaraujo 🦋 @mikaelaraujo.bsky.social 🔄 mikaelaraujo 🔗 mikael-araujo

🌐 <https://www.mikaelaraujo.com>

Mikael Araújo is an international SEO consultant¹⁹⁶, speaker and marketing strategist. He has worked and works remotely for several companies based in Europe, China, Russia, the United States and Brazil. He is currently a Data Science student and loves spending his free time with his family.



Michael Lewittes

✂ @MichaelLewittes 🌐 @https://seocommunity.social/@MichaelLewittes

🦋 @michaellewittes.bsky.social 🔄 MichaelLewittes 🔗 michael-lewittes-a22b831

🌐 <https://www.ranktify.com/team>

Michael Lewittes is the founder of Ranktify¹⁹⁷, an SEO software company that improves the quality and trustworthiness of content so that it rises higher in search results. Michael previously owned and sold a content company, as well as wrote for and edited several major U.S. publications. This is the second time he's worked on the Web Almanac.

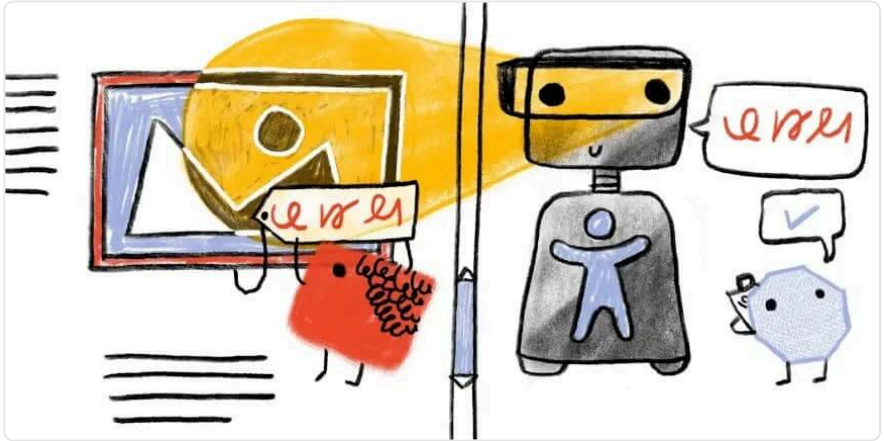
195. <https://tamethebots.com>

196. <https://www.mikaelaraujo.com>

197. <https://www.ranktify.com>

Part II Chapter 8

Accessibility



Written by Mike Gifford

Reviewed by Hidde de Vries, Beatriz González Mellídez, and Kate Kalcevic

Analyzed by Mike Gifford

Edited by Jonathan Pagel, Jonathan Avila, and Shaina Hantsis

Introduction

The web is continuing to change. Voice assistants like Siri, Alexa, and Cortana often provide responses by reading from web pages using screen reader technology. Similar methods have been around since the early days of personal computing¹⁹⁸. Captions (sometimes referred to as subtitles) were created for people who are hard of hearing, but are increasingly used for convenience¹⁹⁹ by everyone, and the vibration mode of the smartphone is now a standard feature. Other groups that enjoy using captions include individuals with ADHD, who use them to maintain focus, non-native speakers, who rely on them to enhance language comprehension, and people in noisy environments, where spoken content might be easily missed.

Modern devices and platforms offer many options for accessibility. These help to personalize the user experience for both people with disabilities as well as the general public. But not many people open the accessibility menu to try them out.

¹⁹⁸. <https://www.theverge.com/23203911/screen-readers-history-blind-henter-curran-teh-mvda>

¹⁹⁹. https://wikipedia.org/wiki/Subtitles#Use_by_hearing_people_for_convenience

Good accessibility is beneficial to everyone, not just those with disabilities. This is a fundamental principle of Universal Design²⁰⁰. As Tim Berners-Lee stated: “The power of the Web is in its universality. Access by everyone regardless of disability is an essential aspect.”²⁰¹ The COVID-19 pandemic made it clear that improving accessibility for digital interfaces can no longer be perceived as optional. It is increasingly hard to navigate the real world without reliable access to the virtual world.

Microsoft’s Inclusive Design Guidelines²⁰² go beyond permanent disability scenarios and extend them to temporary or situational limitations. Human abilities vary. No matter if a person has lost an arm (permanent), or is wearing a cast because of an accident (temporary) or holding a baby (situational limitation), being able to use the computer or phone with one hand or voice interaction benefits them.

200. <https://universaldesign.ca/principles-and-goals/>
201. <https://www.w3.org/WAI/fundamentals/accessibility-intro/>
202. <https://inclusive.microsoft.design/>

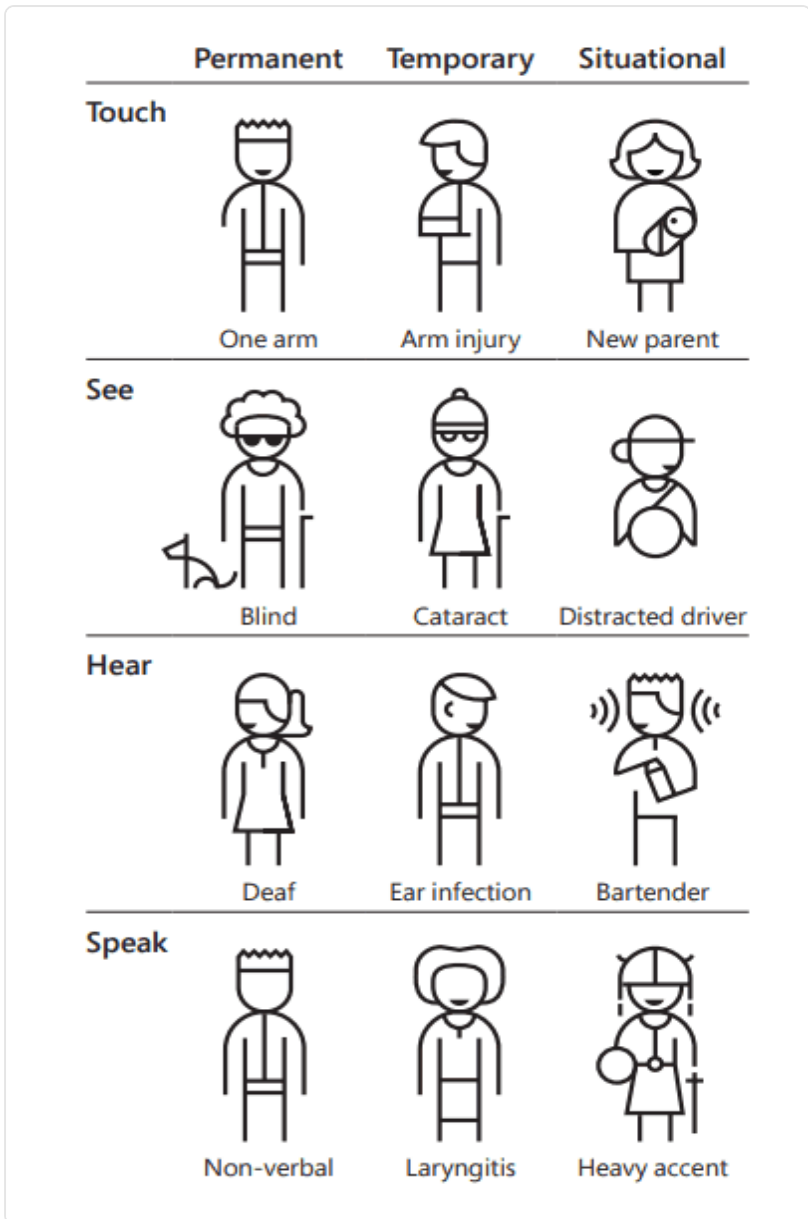


Figure 8.1. Microsoft's Inclusive Design illustration.
This image and approach is courtesy of their Inclusive 101 Guidebook²⁰³.

203. <https://inclusive.microsoft.design/tools-and-activities/Inclusive101Guidebook.pdf>

Governments worldwide recognize that digital accessibility is not only a moral obligation but also in many instances legally required. Accessibility is also great for commerce and democracy. For example, the European Union (EU) mandates that by June 2025, websites in a wide range of sectors must adhere to the Web Content Accessibility Guidelines (WCAG)²⁰⁴ principles (via the EN 301 549²⁰⁵ standard). This will ultimately allow more people to buy and sell services in the EU. Other countries passed similar laws, which increase the pressure on organizations to make their virtual offerings more accessible.

Standards referenced in legislation like EN 301 549 and Section 508²⁰⁶ are based on the Web Content Accessibility Guidelines, and the automated accessibility tests used in this report can only test against some parts of the guidelines. Our tests leverage the open source tool, Google Lighthouse²⁰⁷, which in turn uses Deque's open source axe-core²⁰⁸.

There is a lot of updated information on our report from previous years²⁰⁹. It is useful to track changes over time, and note where change occurs. We also wanted to introduce a new section about different sectors and web accessibility. Through our analysis we can track CMS's, JavaScript frameworks, and evaluate the average accessibility of different technologies. We can also compare the accessibility of different countries and governments and track how it changes over time.

Despite ongoing challenges, there has been noticeable improvement in web accessibility. The median score for Lighthouse Accessibility audits rose to 84% over the past two years. In WCAG 2.2, the 4.1.1 Success criteria was removed. Deque therefore removed `duplicate-id` and `duplicate-id-active` audits from axe, and so this was no longer included in our scan. These deprecated axe rules impacted millions of sites surveyed in our 2022 report. There were also new Success Criteria added in 2.2²¹⁰ added with corresponding tests being added to axe-core²¹¹.

Accessibility scores are an important tool, but people familiar with Goodhart's Law²¹² will know the danger of a measure becoming a target. We must also acknowledge that automated tests can only address a portion²¹³ of the WCAG Success Criteria, and that a perfect score does not guarantee an accessible site²¹⁴.

204. <https://www.w3.org/WAI/standards-guidelines/wcag/>

205. https://wikipedia.org/wiki/EN_301_549

206. <https://www.section508.gov/manage/laws-and-policies/>

207. <https://developer.chrome.com/docs/lighthouse/accessibility/scoring>

208. <https://github.com/dequelabs/axe-core>

209. <https://almanac.httparchive.org/en/2022/accessibility>

210. <https://www.w3.org/WAI/standards-guidelines/wcag/new-in-22/>

211. <https://www.deque.com/blog/axe-core-4-5-first-wcag-2-2-support-and-more/>

212. https://wikipedia.org/wiki/Goodhart%27s_law

213. <https://www.smashingmagazine.com/2022/11/automated-test-results-improve-accessibility/#automate-it>

214. <https://www.matuza.at/blog/building-the-most-inaccessible-site-possible-with-a-perfect-lighthouse-score>

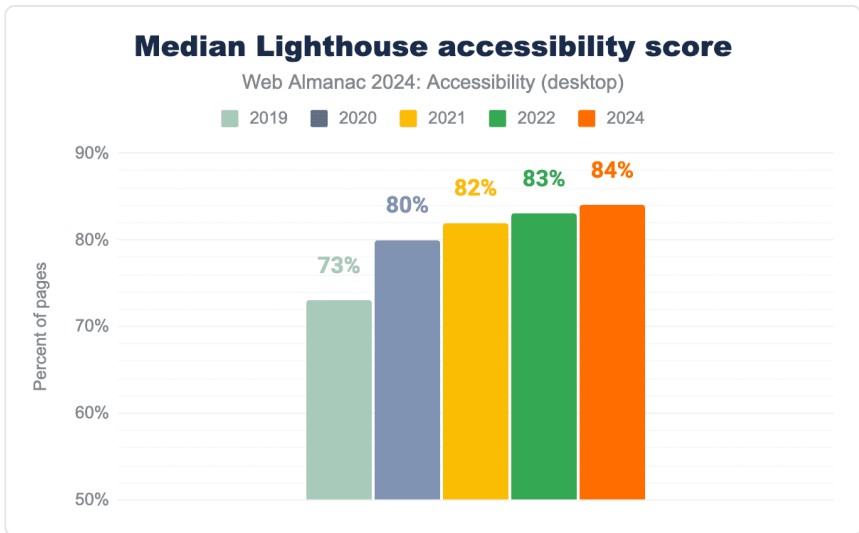


Figure 8.2. Lighthouse audit improvements year-over-year.

We can similarly see an increase in the median Lighthouse score by page rank with an increase in the percentage of pages evaluated. It is a smaller improvement than it was between 2020 and 2021 and the 2023 Web Almanac wasn't produced, so there is a 1% increase for two years. However, it is also worth noting that the Lighthouse score is leveraging axe, which has increased its tests to align more with WCAG 2.2²¹⁵.

Looking at the most common errors with most improved Lighthouse tests, it is possible to see which parts of the Lighthouse audit improved the most. Although far from perfect, we are seeing advancements.

215. <https://www.w3.org/WAI/standards-guidelines/wcag/new-in-22/>

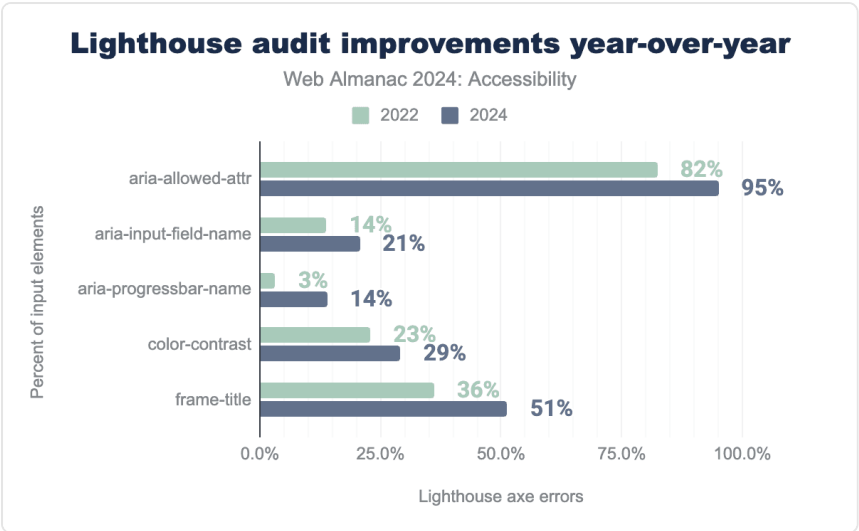


Figure 8.3. Most improved Lighthouse accessibility tests (axe).

Google Lighthouse now contains 57 different audit tests²¹⁶ which they use for their scoring. These are all based on Deque’s open source axe-core²¹⁷, which is widely adopted in a range of accessibility products and services. With the exception of 7 audits (aria-meter-name, aria-toggle-field-name, aria-tooltip-name, document-title, duplicate-id-active, html-lang-valid, and object-alt) there have been improvements across the board. Both object-alt and aria-tooltip-name were called out for their improvements in 2022, but sadly this improvement was not repeated in 2024.

Throughout this chapter, we have included actionable links and solutions that readers can apply and follow in their own accessibility initiatives. For the sake of consistency, we opted to use the person-first language “people with disabilities” throughout, though we recognize that the identity-first term “disabled people” is also widely used. Our terminology choice is not intended to suggest which term is more appropriate.

Ease of Reading

Readability of information and content on the web is crucial. There are different factors in a website that contribute to the content’s readability. Taking these aspects into account ensures that everyone on the internet can easily consume the content. This report covers those things

216. <https://developer.chrome.com/docs/lighthouse/accessibility/scoring>
217. <https://github.com/dequelabs/axe-core/blob/develop/doc/rule-descriptions.md>

which can be measured, and although plain language is critical to readability, it is not easy to measure. There are fairly straightforward mathematical readability scores, like Flesch–Kincaid²¹⁸. Some people use it to determine readability²¹⁹ in English, but the web is global. Language is difficult, and there is no agreed-to standard for automated plain language testing that can be applied even across the most popular languages.

Color Contrast

Color contrast refers to the difference between the foreground and background colors of elements on a page that enables users to see the content. It is very common for websites to have insufficient contrast on key elements like text and icons, despite WCAG making it quite clear how to comply with contrast guidelines.

The minimum contrast requirement²²⁰ defined by the WCAG for normal sized text (up to 24px or if bolded 18px) is 4.5:1 for AA conformance and 7:1 for AAA conformance. However, for larger font sizes, the contrast requirement is only 3:1 as larger text has increased legibility even at a lower contrast.

Google Lighthouse can easily test for most but not all color contrast issues. There are a wide range of open source tools for checking color combinations²²¹ which can be easily incorporated into anyone's workflow. It is important to note that the use of these tools is necessary, as you can't rely on your own perception of sufficient contrast when you have typical contrast sensitivity.

The Lighthouse test determined that 29% of mobile sites and 28% of desktop sites have sufficient text color contrast. This is a moderate improvement over previous years, but it is still far below what is required for basic readability.

218. https://wikipedia.org/wiki/Flesch%E2%80%93Kincaid_readability_tests

219. <https://wikipedia.org/wiki/Readability>

220. <https://www.w3.org/WAI/WCAG22/Understanding/contrast-minimum>

221. <https://accessibility.civicactions.com/guide/tools#color>

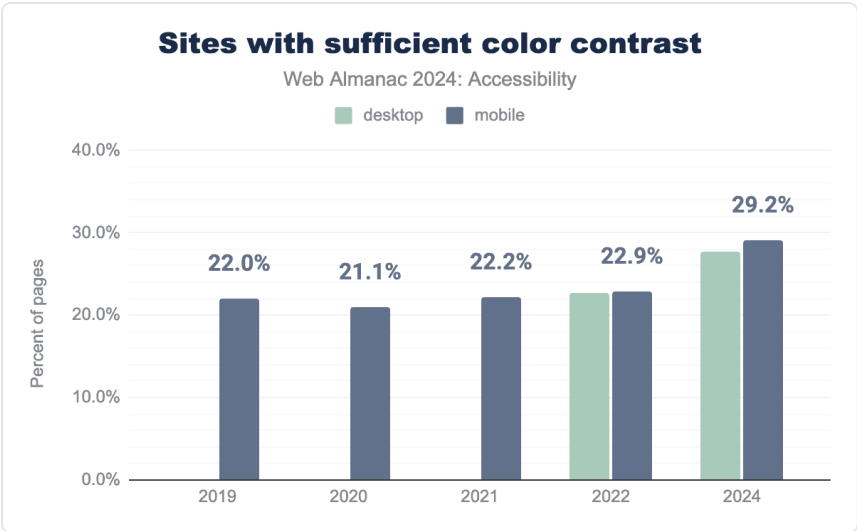


Figure 8.4. Sites with sufficient color contrast.

Color contrast becomes more important as we age²²². It is also something which regularly is an issue for temporary disabilities and situational limitations²²³, such as when people don't have their reading glasses or need to read content outside. Achieving appropriate contrast is becoming more challenging as browsers and operating systems have implemented support for light, dark, and high-contrast modes. These are well supported by browsers and operating systems, but not yet well supported by most websites. There is a growing demand for sites to follow a user's set preference, and multiple types of disabilities can benefit from giving users this control. See the User preferences section below for more information.

Zooming and scaling

More than ever, users are engaging with websites with a variety of technologies from super-wide curved screens to mobile phones and even watches. Disabling scaling takes away user agency to define what works best for them. WCAG requires²²⁴ that text in a website must be resizable up to at least 200% without any loss in content or functionality.

We're revisiting Adrian Roselli's post²²⁵, which we previously highlighted in 2022, to emphasize the importance of not disabling zoom functionality, as many still don't fully understand the reasons behind it.

222. <https://www.nia.nih.gov/health/vision-and-vision-loss/aging-and-your-eyes>

223. <https://inclusive.microsoft.design/>

224. <https://www.w3.org/WAI/WCAG22/Understanding/resize-text>

225. <https://adrianroselli.com/2015/10/dont-disable-zoom.html>

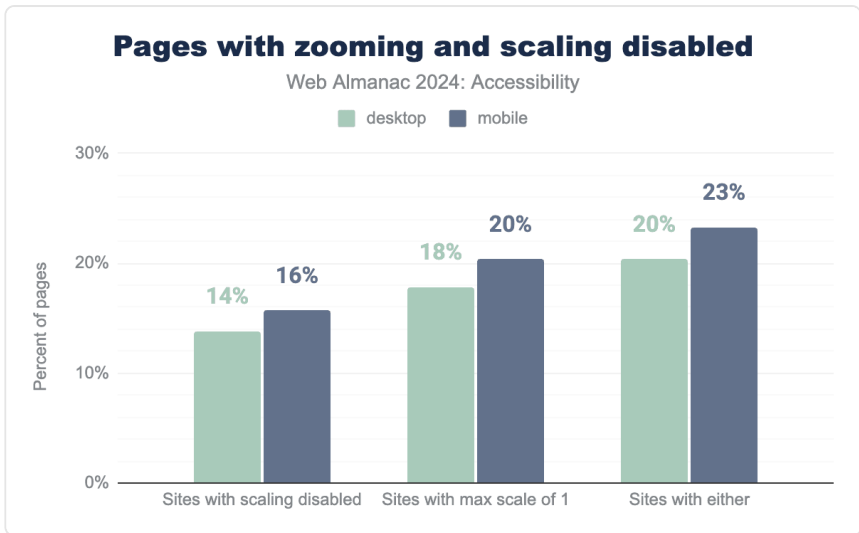


Figure 8.5. Pages with zooming and scaling disabled.

We are happy to say that we are seeing a reduction in sites which are disabling zooming and scaling. Compared with 2022's data for mobile users there are 2% less sites who have disabled scaling and 4% that have disabled a max scale of 1. The desktop average both went down by 2% compared to the last report. Users can configure their browsers to override this setting, but some defaults still respect the author's preferences.

To check if your site has disabled or limited zoom look at the source of the page and search for `<meta name="viewport">` if it is tagged with a maximum-scale, minimum-scale, user-scalable=no, or user-scalable=0. Ideally, there wouldn't be any restrictions for content resizing, but WCAG only specifies the need for 200% magnification²²⁶.

226. <https://www.w3.org/TR/UNDERSTANDING-WCAG20/visual-audio-contrast-scale.html#:~:text=Content%20satisfies%20the%20Success%20Criterion%20if%20it%20can,more%20extreme%2C%20adaptive%20layouts%20may%20introduce%20usability%20p>

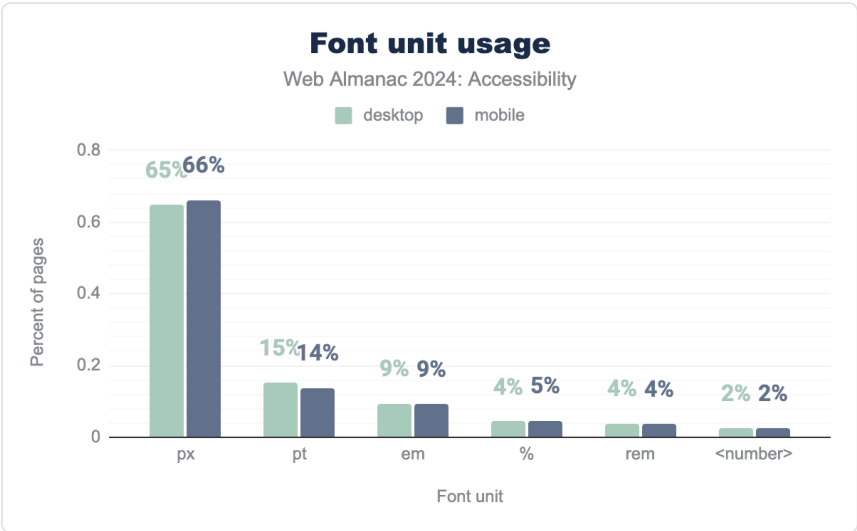


Figure 8.6. Font unit usage.

How font sizes are defined also affects the readability, as pixels are not as flexible as other units. Pixel (px) use is 65% in desktop and 66% in mobile. The use of em has increased to 9% from 6% in 2022. rem was at 6% in 2022, and has reduced to 4% now. There is not a significant increase in the use of em or rem since 2022, even though it often gives the user a better experience²²⁷ when they increase or decrease their font size in their browser settings.

Language identification

Identifying language with the lang attribute enhances screen reader support and facilitates automatic browser translations. This feature benefits all users. For instance, without the lang attribute, Chrome's automatic translation feature may produce inaccurate translations. Manuel Matuzović provides an example of how missing lang attributes can lead to translation errors²²⁸. The lang attribute is also helpful when styling web pages for different languages and reading directions²²⁹, as Chen Hui-Jing points out.

86%

Figure 8.7. Mobile sites have a valid lang attribute.

227. <https://www.freecodecamp.org/news/css-units-when-to-use-each-one/>

228. <https://www.matuzo.at/blog/lang-attribute/>

229. <https://chenhuijing.com/blog/css-for-i18n/>

It's promising to note that in 2022, 83% of mobile websites included a lang attribute, and two years later it is at 86%. However, since this is a Level A conformance issue under WCAG, there is still room for improvement. To meet this criterion, the lang attribute should be added to the `<html>` tag with a recognized primary language tag²³⁰. Properly defining the language is crucial. Sometimes, when a template is copied to create a new website, discrepancies can arise between the language of the content and the language attribute (`lang="en"`) in the code.

Also, keep in mind that there is the page language, but pages often contain multiple languages within them. The lang attribute can also be applied to other tags if the page contains multiple languages. The W3C has good documentation on how to address the Language of Parts²³¹.

User preference

Modern CSS includes Level 5 Media Queries²³², which include User Preference Media Queries²³³. User Preference Media Queries enhance accessibility by allowing users to select configurations that work for them. This includes choices like color schemes or contrast modes that suit individual preferences, such as dark mode. Users can also choose to minimize animations on a page, which is beneficial for users with vestibular disorders.

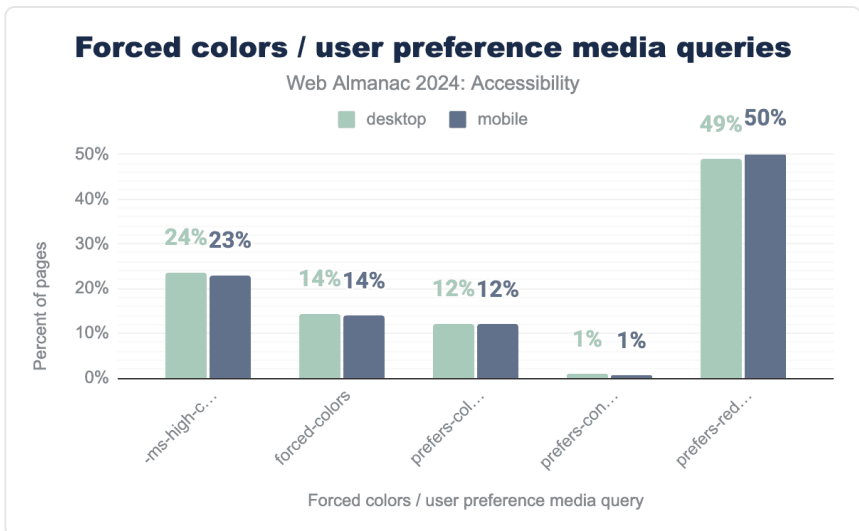


Figure 8.8. User preference media queries.

We discovered that over 50% of mobile sites include the `prefers-reduced-motion` media

230. <https://www.w3.org/WAI/standards-guidelines/act/rules/bf051a/#known-primary-language-tag>

231. <https://www.w3.org/WAI/WCAG21/Understanding/language-of-parts.html>

232. <https://www.w3.org/TR/mediaqueries-5>

233. <https://12daysofweb.dev/2021/preference-queries/>

query, up from 34% in 2022. This is important because digital animations can harm individuals with vestibular disorders²³⁴; using this query allows for adapting or removing such animations to improve accessibility. Mozilla's Developer community has some good resources on building motion-sensitive sites²³⁵.

The Sustainability chapter has some great statistics on the rise in the use of animations, and the impact on digital sustainability.

For contrast, only 12% of desktop and mobile websites utilize the `prefers-color-scheme` media query, which is up from 8% in 2022. To enhance content readability it is a good practice to allow your users to adjust display modes. The `prefers-color-scheme` query enables browsers to detect the user's preferred color scheme, supporting light and dark modes. The `prefers-contrast` query is valuable for users with low vision or photosensitivity by enabling high contrast modes.

Support for `forced-contrast` increased about 4% in 2024 to 14% for desktop. Forced Colors Mode is an accessibility feature designed to enhance text readability through improved color contrast. When activated, the user's operating system takes control of most color-related styles, overriding website color settings. This mode disables common patterns like background images to ensure more predictable text-to-background contrast.

The most well-known implementation is Windows' High Contrast Mode, now called Contrast Themes²³⁶. These themes offer alternative color palettes with low and high contrast options, and allow users to customize the system colors to their preference. Use of `-ms-high-contrast` has decreased slightly in 2024 to about 23%. This can be emulated in Edge²³⁷ and Chrome²³⁸, so it is now easier to test.

Navigation

When discussing website navigation, it's crucial to recognize that users may employ a range of methods and input devices. Some might use a mouse to scroll, while others may rely on a keyboard, switch control device, or screen reader to navigate through headings. When designing a website, it's essential to ensure it functions effectively for all users, regardless of the device or assistive technology they use. Wide-screen TV's and voice interfaces (like Siri and Amazon Alexa) both place challenges on how we design our navigation. Building a good semantic structure into a site helps screen reader users navigate a site, but also helps users of many other types of technology.

234. <https://kb.iu.edu/d/bizw>

235. <https://developer.mozilla.org/docs/Web/CSS/@media/prefers-reduced-motion>

236. <https://support.microsoft.com/en-us/topic/fedc744c-90ac-69df-aed5-c8a90125e696>

237. <https://blogs.windows.com/msedgedev/2024/04/29/deprecating-ms-high-contrast/>

238. <https://developer.chrome.com/docs/devtools/rendering/emulate-css/>

Focus indication

Focus indication is crucial for users who navigate websites primarily using a keyboard and some alternative navigation devices. WebAim has some great resources on assistive technology for individuals with limited motor abilities²³⁹. These devices are also customized by the user to maximize what they can control. There are a lot of similarities between how visible focus styles and focus order are managed with these devices, but it may be different than for keyboard only users.

Most automated tests do not test for focus order or keyboard traps. Lighthouse can not tell you that your site is keyboard navigable. All it can do is tell you that it isn't if your site fails some basic tests which are essential. Lighthouse uses Deques' axe rules²⁴⁰ to evaluate best practices. Even with a perfect score for focus indication, you need to test your pages manually. Lighthouse recommends testing for:

- Focus traps²⁴¹
- Interactive controls are keyboard focusable²⁴²
- Logical tab order²⁴³
- Focus directed to new content on a page²⁴⁴

Accessibility Insights²⁴⁵ is a great open source tool that leverages Deque's axe. This Chrome/Edge extension²⁴⁶ can help with keyboard-only testing and guide developers through other tests. The Tab Stops feature is a great visual indicator of a keyboard-only user's progress through a website.

Focus styles

WCAG mandates a visible focus indicator for all interactive content to ensure users can identify which element is currently focused as they move through a page.

239. <https://webaim.org/articles/motor/assistive>

240. <https://github.com/dequelabs/axe-core/blob/develop/doc/rule-descriptions.md>

241. <https://developer.chrome.com/docs/lighthouse/accessibility/focus-traps>

242. <https://developer.chrome.com/docs/lighthouse/accessibility/focusable-controls>

243. <https://developer.chrome.com/docs/lighthouse/accessibility/logical-tab-order>

244. <https://developer.chrome.com/docs/lighthouse/accessibility/managed-focus>

245. <https://accessibilityinsights.io/docs/web/overview/>

246. <https://accessibilityinsights.io/downloads/>

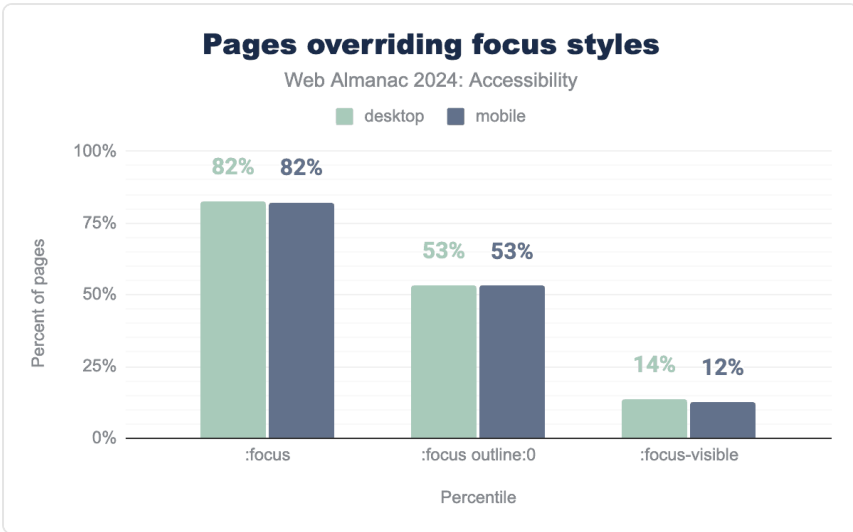


Figure 8.9. Pages overriding browser focus styles.

We discovered that 53% of websites apply `:focus {outline: 0}` (in 2022 it was 86%), which removes the default outline provided by browsers for focused interactive elements. Although some websites implement custom styles to override this, it's not always the case, making it challenging for users to identify the currently focused element and impeding navigation. Sara Soueidan offers valuable guidance on designing WCAG-compliant focus indicators²⁴⁷. On a positive note, 12% of websites now use `:focus-visible` (up from 9% in 2022 and 0.6% in 2021), which is a pseudo class that uses browser heuristics to determine when to show the focus indicator. This is a significant improvement in accessibility practices.

tabindex

Generally, HTML will have focus order set without having to set the `tabindex`. CSS and JavaScript often cause changes to how it is presented in the DOM. The `tabindex` attribute controls whether an element can receive focus and determines its position in the keyboard focus or “tab” order.

Our analysis shows that 63% of mobile websites and 64% of desktop websites utilize `tabindex` (up from 60% and 62% respectively). This attribute serves several purposes, which can affect accessibility:

247. <https://www.sarasoueidan.com/blog/focus-indicators/>

- `tabindex="0"` places an element in the sequential keyboard focus order. Custom interactive elements and widgets should have `tabindex="0"` to ensure they are included in the focus sequence.
- `tabindex="-1"` removes the element from the keyboard focus order but allows it to be focused programmatically via JavaScript.
- A positive `tabindex` value overrides the default keyboard focus order, often leading to issues with WCAG 2.4.3 - Focus Order²⁴⁸.

It's important to avoid placing non-interactive elements in the keyboard focus order, as this can be confusing for screen reader users who can see and alternative navigation users.

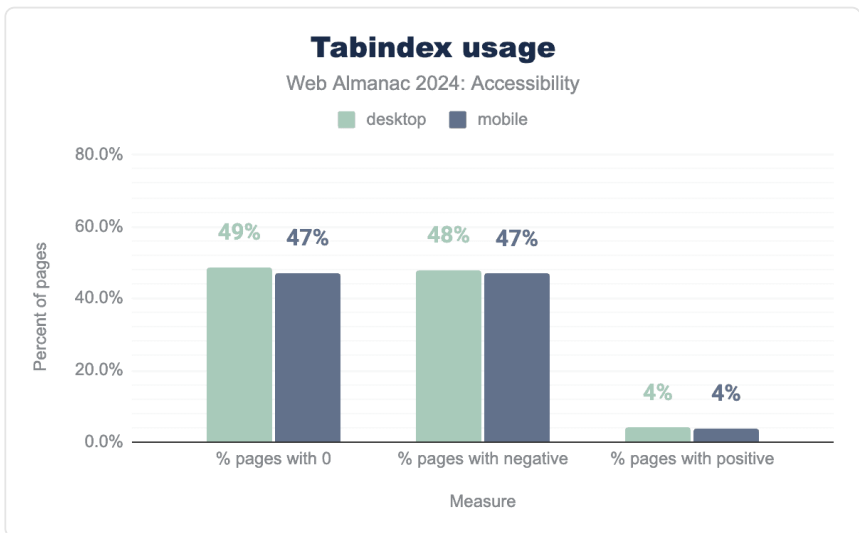


Figure 8.10. *tabindex* usage.

Of all websites using the `tabindex` attribute, 4% employ positive values (down from 7% in 2022). Using positive `tabindex` values is generally considered poor practice as it disrupts the natural tab order. Karl Groves provides an insightful article on this topic²⁴⁹.

Landmarks

Landmarks help structure a web page into distinct thematic regions, facilitating easier

²⁴⁸. <https://www.w3.org/WAI/WCAG21/Understanding/focus-order.html>

²⁴⁹. <https://karlgroves.com/2018/11/13/why-using-tabindex-values-greater-than-0-is-bad>

navigation for users of assistive technologies. For instance, a rotor menu²⁵⁰ can help navigate between different page landmarks, while skip links²⁵¹ can direct users to specific landmarks, such as `<main>`. Landmarks can be defined using various HTML5 elements. This semantic structure can also be applied with Web Accessibility Initiative – Accessible Rich Internet Applications (ARIA)'s landmark roles²⁵². However, it's best to use native HTML5 elements whenever possible, in line with ARIA's first rule²⁵³.

Although ARIA landmarks have traditionally been only visible to screen reader users, some sites are beginning to use tools like this open source SkipTo script²⁵⁴ which aggregates headings and landmarks into a page-specific table of contents. Exposing the document structure to the user helps everyone's comprehension. SkipTo delivers what really should be basic browser functionality. This goes beyond the skip links that are discussed in a later section.

Element type	% of element	% of role	% of both
main	37%	17%	44%
header	65%	12%	66%
nav	66%	19%	70%
footer	65%	10%	67%

Figure 8.11. Landmark element and `role` usage (desktop).

250. <https://www.dfb.org/blindness-and-low-vision/using-technology/cell-phones-tablets-mobile/apple-ios-iphone-and-ipad-2>
251. <https://webaim.org/techniques/skipnav/>
252. <https://www.w3.org/TR/WCAG20-TECHS/ARIA11.html>
253. <https://www.a11y-collective.com/blog/the-first-rule-for-using-aria/>
254. <https://github.com/paypal/skipto>

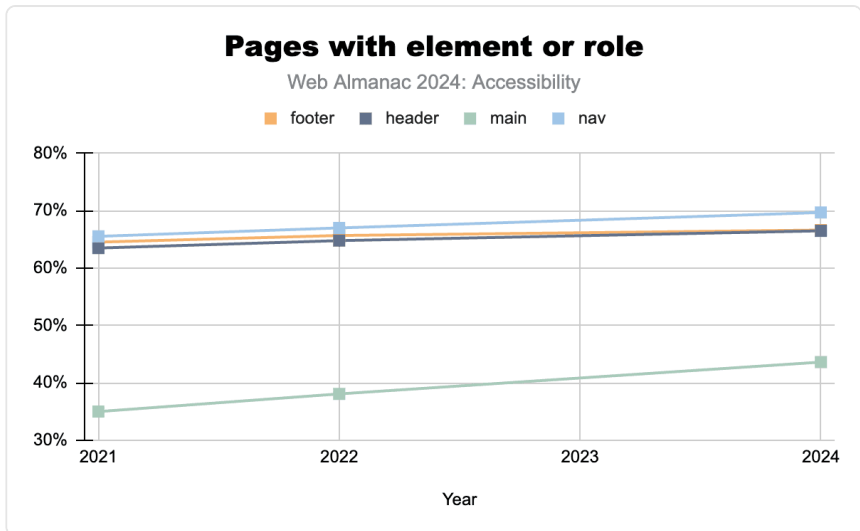


Figure 8.12. Yearly growth in pages with element role.

The most commonly expected landmarks for most web pages include `<main>`, `<header>`, `<nav>`, and `<footer>`. Our findings reveal that:

- only 37% of all pages use the native HTML `<main>` element,
- 17% of all pages have an element with `role="main"`, and
- 43% are using either one of them (up from 35% in 2021).

Scott O'Hara's article on landmarks²⁵⁵ provides valuable insights for enhancing accessibility.

Heading hierarchy

Headings are crucial for all users, including those with assistive technologies, as they aid in navigating a website. Assistive technologies enable users to jump to specific sections of interest. As highlighted in Marcy Sutton's article on headings and semantic structure²⁵⁶, headings function like a table of contents, allowing users and search engines to navigate to particular content areas efficiently.

Sadly the heading hierarchy has gotten worse over the last two years. Lighthouse tracks properly ordered headings and it has dropped just over 1% to:

255. <https://www.scotthara.me/blog/2018/03/03/landmarks.html>

256. <https://marcsutton.com/how-i-audit-a-website-for-accessibility#Headings-and-Semantic-Structure>

57%

Figure 8.13. Mobile sites passing the Lighthouse audit for properly ordered heading.

Heading levels are associated with different font sizes and are frequently misused to visually style a website or mark specific sections rather than structure the content. This misuse negatively impacts both user experience and accessibility tools as well as search engines. CSS should be used to style elements, not heading tags.

WCAG mandates that websites offer multiple navigation options beyond the primary menu in the header, as outlined in Success Criterion 2.4.5: Multiple Ways²⁵⁷. For instance, many users, including those with cognitive disabilities, prefer search features to locate pages on large websites.

Currently, 21% of mobile websites and 22% of desktop websites include a search input (down from 23% and 24% respectively in 2022). This is not a good trend.

Skip links

Skip links enable users who rely on the keyboard, switch control devices, or other alternative navigation tools to bypass various sections of a webpage without having to navigate through every focusable element. A common use is to skip over the primary navigation and move directly to the `<main>` content, particularly on sites with extensive interactive navigation menus. This can dramatically improve the user experience for some users.

24%

Figure 8.14. Mobile and desktop pages likely featuring a skip link.

We discovered that 24% of both desktop and mobile pages likely include a skip link, helping users to avoid unnecessary parts of the page. This percentage may actually be higher, as our analysis only detects skip links positioned near the top of the page (such as those for bypassing navigation). Skip links can also be used to skip other sections of the page, as we described above with the SkipTo script.

²⁵⁷. <https://www.w3.org/WAI/WCAG21/Understanding/multiple-ways.html>

Document titles

Descriptive page titles are important for navigating between pages, tabs, and windows. Assistive technologies, such as screen readers, read these titles aloud, helping users keep track of their location.

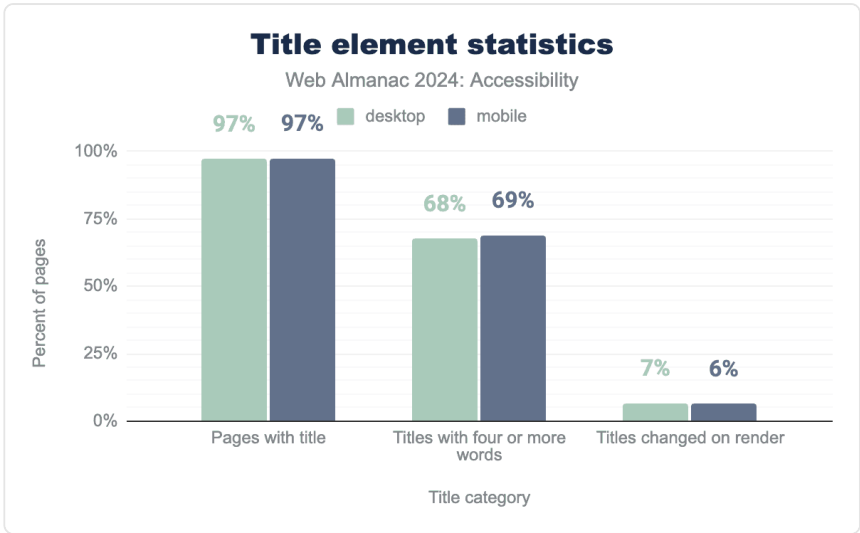


Figure 8.15. Title element statistics.

While 97% of mobile websites include a document title, only 69% have titles that are more than four words long. Since our analysis is limited to the home page and a secondary, we have limited insights about the inner pages. We did find that secondary pages were 8% more likely to have descriptive titles of more than four words (78% on average in 2024). Ideally, a title should include both a brief description of the page's content to enhance navigation and the website's name.

The titles changed on render value is derived from a comparison of the initial HTML title and the final value of the page after JavaScript has loaded. The data indicates that 7% of desktops sites scanned are dynamically changing the content of the title. Secondary pages are slightly less likely to change the title than the home page.

Tables

Tables present data and relationships using two dimensions. For accessibility, tables need a well-structured format with elements like captions and header cells to help users with assistive technologies understand and navigate the data. A caption, using the `<caption>` element, is

especially important as it provides context for screen readers. Currently, 1.6% of desktop sites use a `<caption>` (slightly up from 1.3 in 2022), but this is a crucial aspect for making tables more accessible.

	Table Sites		All Sites	
	desktop	mobile	desktop	mobile
Captioned tables	5.5%	4.8%	1.6%	1.5%
Presentational table	4.4%	5.0%	3.1%	4.2%

Figure 8.16. Table usage.

Tables don't need to be used for page layout, thanks to CSS Flexbox and Grid. If necessary, tables can use `role="presentation"` to explicitly remove semantics and thereby avoid confusion when they are used for layout purposes. We see that 4% of mobile tables use this workaround (versus 1% in 2022).

Forms

Forms are essential for user interactions, such as logging in or making purchases. For users with disabilities, accessible forms are crucial for completing tasks and achieving equal functionality. Forms are also often much more complicated for developers to build than static HTML pages.

`<label>` element

The `<label>` element is the preferred way for linking input fields with their accessible names²⁵⁸. Using the `for` attribute to match the `id` of an input ensures proper programmatic association, improving form usability. Furthermore, when the label element is used properly it allows users to click or tap on the label to focus the form field.

For example:

```
<label for="emailaddress">Email</label>
<input type="email" id="emailaddress">
```

258. https://developer.mozilla.org/docs/Learn/Forms/Basic_native_form_controls

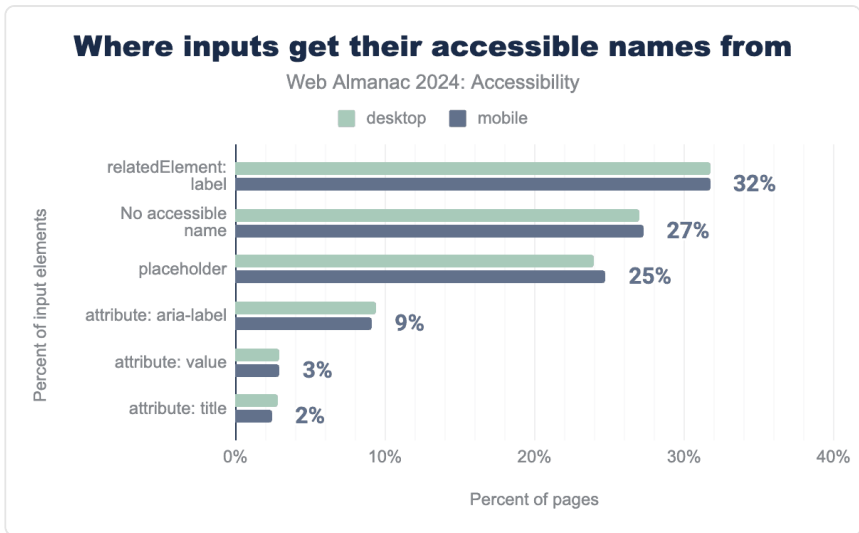


Figure 8.17. Where inputs get their accessible names from.

Unfortunately, 13% of mobile inputs lack accessible names (a significant improvement from 38% in 2022). Only 15% of mobile sites use `<label>` (down from 19% in 2022), which can hinder users relying on screen readers or voice-to-text tools. An accessible name always needs to be used and sites should support assistive technology beyond just screen readers by ensuring the accessible name matches the visible label on the input. WCAG 2.1 added 2.5.3 Label in Name (Level A) to help ensure that technologies like Voice Control would be better supported. Use of `aria-label` and `aria-labelledby` should only be used if an HTML `<label>` cannot be used.

placeholder attribute

The `placeholder` attribute provides example input formats. It should not replace a `<label>` as a way to provide an accessible name. When placeholders are the only way of providing a visible label, that reference point disappears when the user starts typing. It is not a new concern that browsers by default do not give placeholder text sufficient contrast²⁵⁹ to meet WCAG. Furthermore, they are not always supported by screen readers²⁶⁰. A better solution is to show example input formats below or beside the input and connect them to the input programmatically using `aria-describedby`.

259. https://www.w3.org/WAI/GL/low-vision-a11y-tf/wiki/Placeholder_Research

260. <https://www.digital11y.com/anatomy-of-accessible-forms-placeholder-is-a-mirage/>

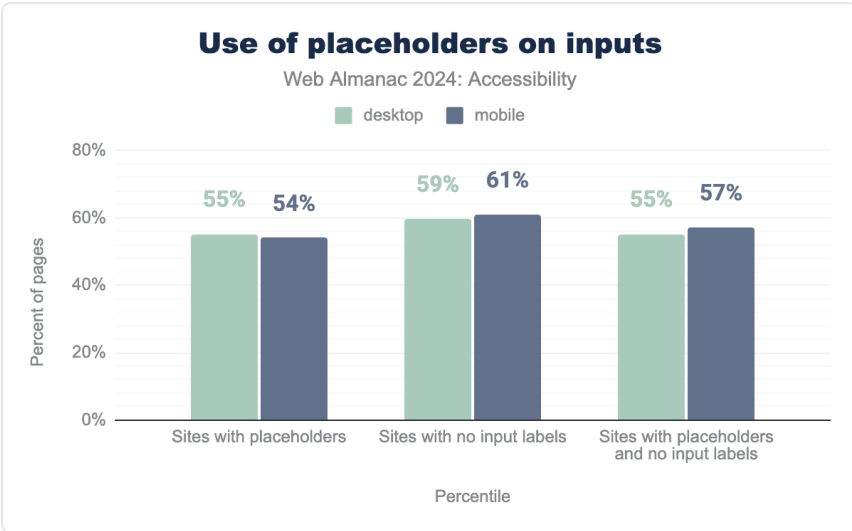


Figure 8.18. Use of placeholders on inputs.

57% of mobile sites and 55% of desktop sites use placeholders alone, which can lead to accessibility issues. As per HTML5 guidelines, placeholders should not replace labels for accessibility.

Use of the placeholder attribute as a replacement for a label can reduce the accessibility and usability of the control for a range of users including older users and users with cognitive, mobility, fine motor skill or vision impairments.

— The W3C's Placeholder Research²⁶¹

Requiring information

Indicating required fields is crucial for forms. Before HTML5, an asterisk (*) was commonly used, but it's only a visual cue and doesn't provide error validation. In addition, the required attribute in HTML5 and `aria-required` attribute can improve the semantics for indicating mandatory fields.

261. https://www.w3.org/WAI/GL/low-vision-a11y-tf/wiki/Placeholder_Research

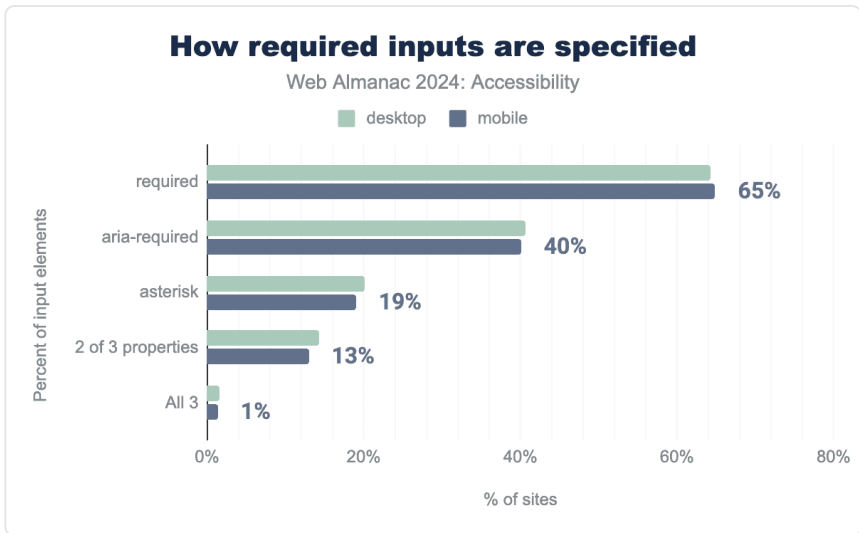


Figure 8.19. How required inputs are specified.

Currently,

- 65% of mobile sites use the required attribute (down from 67% in 2022), and
- 40% use `aria-required` (up from 32% in 2022), but
- 19% still rely only on an asterisk (which is down from 22% in 2022).

This should be avoided unless supplemented by required and `aria-required`.

Captchas

Websites often use CAPTCHAs to verify that a visitor is human and not a bot. CAPTCHAs, which stands for “Completely Automated Public Turing Test to Tell Computers and Humans Apart,” are commonly used to prevent malicious software.

16%

Figure 8.20. Mobile sites implementing one of the two detectable CAPTCHA types.

These tests can be challenging for everyone, especially for those with low vision or reading

disabilities. The W3C has suggested alternatives to visual CAPTCHAs, which are worth exploring.

Media on the web

Accessibility of media is crucial. People with disabilities need alternative methods to understand and interact with media content. For example, blind users require audio descriptions for images or videos, while those who are deaf or hard of hearing need sign language or captions.

Transcripts are needed for audio only and video only content. Non-text content such as images need equivalent alternatives or if they are just decorative they need to be semantically marked as such.

A media player is often embedded in the page to allow a user to play the audio or video content directly inline. If this is the case, it is important that an accessible player, such as the open source Able Player²⁶², is used.

Images

Images can have an `alt` attribute that provides a text description for screen readers. 69% of images passed the Google Lighthouse audit for images with alt text²⁶³ (up from 59% in 2022). Which is a notable increase, especially because the number only increased about 1% from 2021 to 2022.



Figure 8.21. Pass the Lighthouse audit for images with alt text.

The alt text should reflect the image's context. For decorative images, `alt=""` is appropriate, while meaningful images require informative descriptions. It's also important to avoid using file names as alt text, as it almost never provides relevant information. Currently 7.5% of mobile and 7.2% of desktop sites currently do.

262. <https://ableplayer.github.io/ableplayer/>

263. <https://dequeuniversity.com/rules/axe/4.7/image-alt>

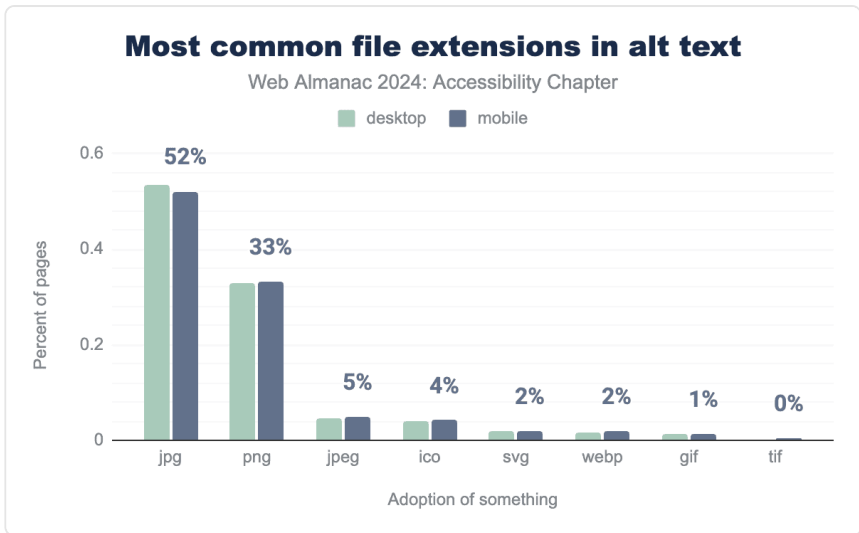


Figure 8.22. Most common file extensions in alt text.

The most common file extensions found in alt text values (for sites with non-empty `alt` attributes) are `jpg` (and `jpeg`), `png`, `ico`, and `svg`. This likely indicates that CMS or other content management systems either automatically generate alt text or require content editors to provide it. However, if the CMS merely includes the image filename in the `alt` attribute, it typically offers no benefit to users. Therefore, it's crucial to use meaningful text descriptions.

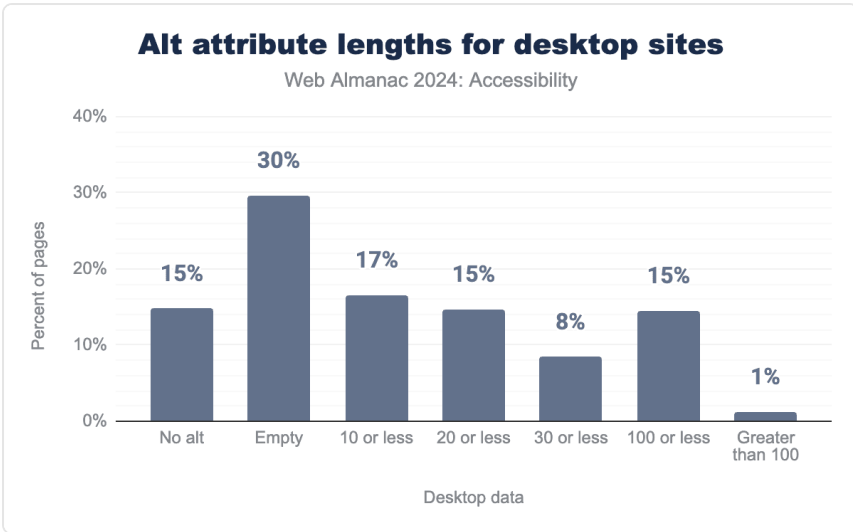


Figure 8.23. `alt` attribute lengths.

There is a slight decline in images with no alt text currently at 15%, down 3% from the 2022 values. We discovered that 30% of alt text attributes on both desktop and mobile sites are empty, up from 27% in 2022. An empty `alt` attribute should be used only for images that are purely decorative and do not need to be described by screen readers or other assistive technologies. Most images contribute to the page content, so they should generally include a meaningful description²⁶⁴.

Unfortunately, 17% of `alt` attributes contain 10 or fewer characters, this is down one percent from 2022. These unusually brief descriptions suggest inadequate information to describe the image appropriately. While some of these may be used to label links, which is acceptable, many lack sufficient descriptive content²⁶⁵.

There is certainly more that can be done to change these stats. Very few content tools support authors through better documentation and validation as suggested in Authoring Tools Accessibility Guidelines (ATAG) 2.0²⁶⁶. Increasingly people are looking at Artificial Intelligence (AI) to create the alt text, usually on the client side. Brian Teeman wrote an interesting critique of the AI generation of Alt Text²⁶⁷.

One promising approach is from Mike Feranda in Drupal who has incorporated AI into CKEditor with AIDmi²⁶⁸. By showing authors an example of what the alt-text could be, they may

264. <https://www.craigabbott.co.uk/blog/how-to-write-good-alt-text-for-screen-readers/>

265. <https://adrianroselli.com/2024/05/my-approach-to-alt-text.html>

266. <https://www.w3.org/TR/ATAG20/>

267. <https://magazine.joomla.org/all-issues/june-2024/ai-generated-alt-text>

268. <https://www.drupal.org/project/aidmi>

be more likely to edit it to have it reflect what they are trying to say. This approach could be applied to other editing tools.

Audio and video

The `<track>` element is used to provide timed text for `<audio>` and `<video>` elements, such as captions and descriptions. This helps users with hearing loss or visual impairments understand the content.

0.1%

Figure 8.24. Sites with `<audio>` elements include a `<track>` element.

For `<video>` elements, the figure is slightly higher at 0.5% for both desktops and 0.65% for mobile sites. These statistics do not cover audio or video embedded via `<iframe>`, where third-party services are less likely to offer text alternatives. Our industry can do a lot better.

The methodology to collect the data for this report did not include modern HLS (HTTP Live Streaming) which would include a manifest file to include subtitles for the deaf and hard-of-hearing (SDH). In the future, scanning for this would allow better understanding of what languages are supported with closed captions and also collect information about the use of audio description.

Assistive technology with ARIA

Accessible Rich Internet Applications (ARIA)²⁶⁹ provides a set of attributes for HTML5 elements designed to enhance web accessibility for individuals with disabilities. However, excessive use of ARIA attributes can sometimes create more problems than it solves. ARIA should be employed only when native HTML5 elements are inadequate for ensuring a fully accessible experience and should not replace or be used beyond what is necessary.

ARIA roles

When assistive technologies interact with an element, the element's role helps convey how users might engage with its content.

For instance, tabbed interfaces²⁷⁰ often require specific ARIA roles to accurately represent their

269. <https://www.w3.org/TR/using-aria/>

270. <https://inclusive-components.design/tabbed-interfaces/>

structure. The WAI-ARIA Authoring Practices Design Patterns²⁷¹ outline how to create an accessible tabbed interface, suggesting that a `tablist` role be assigned to the container element due to the absence of a native HTML equivalent.

HTML5 introduced numerous native elements with built-in semantics and roles. For example, the `<nav>` element inherently has a `role="navigation"`, making it unnecessary to explicitly add this role with ARIA.

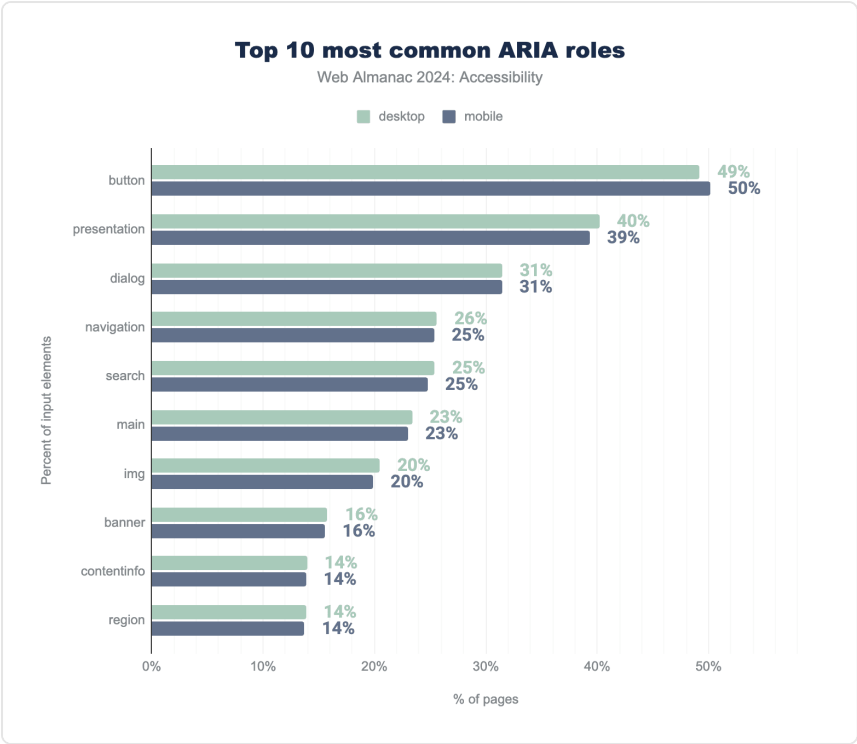


Figure 8.25. Top 10 most common ARIA roles.

We observed that over 50% of mobile sites had home pages with at least one element assigned the `role="button"` (up from 33% in 2022, and 29% in 2021 and 25% in 2020). This increase is concerning, as it suggests websites may be using `<div>` or `<span>` elements as custom buttons or redundantly applying roles to `<button>` elements. Both practices are problematic and violate the fundamental ARIA principle of using native HTML elements—such as `<button>`—whenever possible.

271. <https://www.w3.org/TR/wai-aria-practices-1.1/#tabpanel>

18%

Figure 8.26. Websites with at least one anchor with an href with a `role="button"`.

18% of websites have at least one link with `role="button"` (slightly down from 21% in 2022). While adding an ARIA role can inform assistive technologies about an element's purpose, it doesn't make the element function like its native counterpart. This discrepancy can lead to issues with keyboard navigation since links and buttons have different behaviors. For example, links are not activated by the space key, whereas buttons are.

Using the presentation role

When an element is assigned the `role="presentation"`, it loses its inherent semantics, along with those of its required child elements (e.g., list items within a `<ul>`, or rows and cells within a table). For instance, applying `role="presentation"` to a parent `<table>` or `<ul>` element will propagate this role to its child elements, causing them to lose their table or list semantics.

Removing semantics with `role="presentation"` means the element only has visual presence and its structure is not recognized by assistive technologies. The element's content will be read by a screen reader, but no information about the semantics will be provided.

40%

Figure 8.27. Of desktop sites and 39% of mobile sites have at least one `role="presentation"`.

This is concerning as in 2022 it was already high at 25% of desktop sites and 24% of mobile sites.

Similarly, using `role="none"` also removes the element's semantics. This year, 5% of sites used `role="none"`, down from 11% in 2022. While it may be useful in rare cases, such as when a `<table>` is used purely for layout, it generally should be used cautiously as it can be detrimental to accessibility.

Most browsers disregard `role="presentation"` and `role="none"` when exposing a role in the accessibility tree for focusable elements, including links and inputs, or elements with a `tabindex` attribute. Similarly, if an element with these roles includes global ARIA states or properties (such as `aria-describedby`), the `presentation` and `none` roles may be

ignored.

Labeling elements with ARIA

In addition to the DOM, browsers have an accessibility tree²⁷², containing details about HTML elements such as accessible names, descriptions, roles, and states. This information is communicated to assistive technologies through accessibility APIs.

An element's accessible name can come from its content (e.g., button text), attributes (e.g., image `alt` attribute), or associated elements (e.g., a label linked to a form control). There is a hierarchy used to determine the source of the accessible name when multiple sources are available. For further reading on accessible names, Léonie Watson's article, "What is an accessible name?"²⁷³ is a valuable resource.

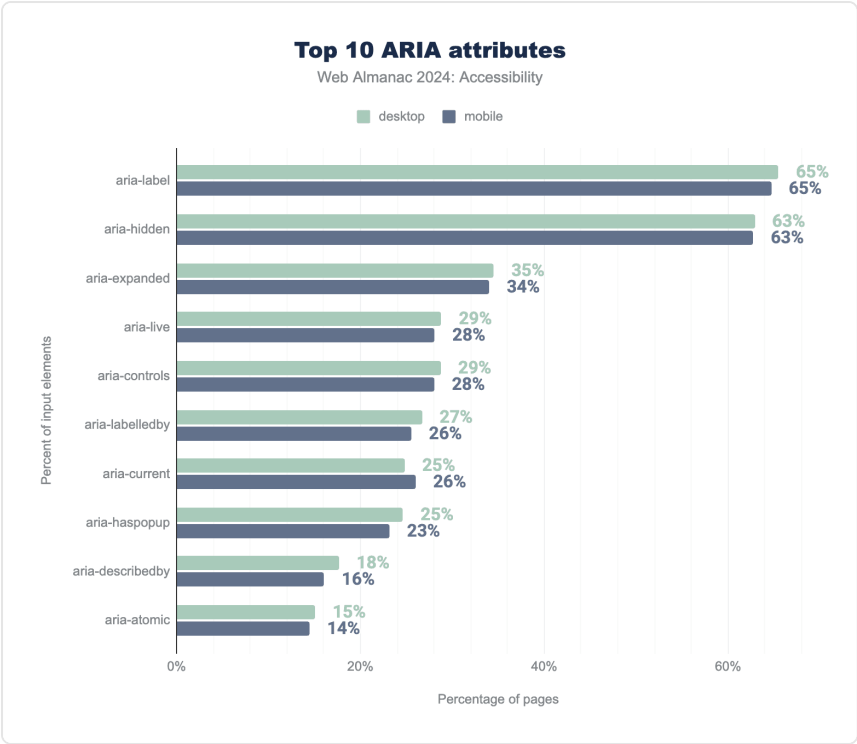


Figure 8.28. Top 10 ARIA attributes.

272. https://developer.mozilla.org/docs/Glossary/Accessibility_tree

273. <https://developer.paciellogroup.com/blog/2017/04/what-is-an-accessible-name/>

Two ARIA attributes that aid in assigning accessible names are `aria-label` and `aria-labelledby`. These attributes are given precedence over natively derived accessible names and should be used sparingly and only when necessary. Testing accessible names with screen readers and involving individuals with disabilities is crucial to ensure that the names are helpful and do not hinder accessibility.

We observed that almost 66% of pages evaluated featured at least one element with the `aria-label` attribute (up from 58% for desktop and 57% on mobile in 2022), making it the most frequently used ARIA attribute for accessible names. Additionally, 27% of desktop pages and 25% of mobile pages had at least one element with the `aria-labelledby` attribute (both are up 2% from 2022 data). This trend suggests that while more elements are being assigned accessible names, it might also indicate a rise in elements lacking visual labels. This can be challenging for users with cognitive disabilities and voice input users.

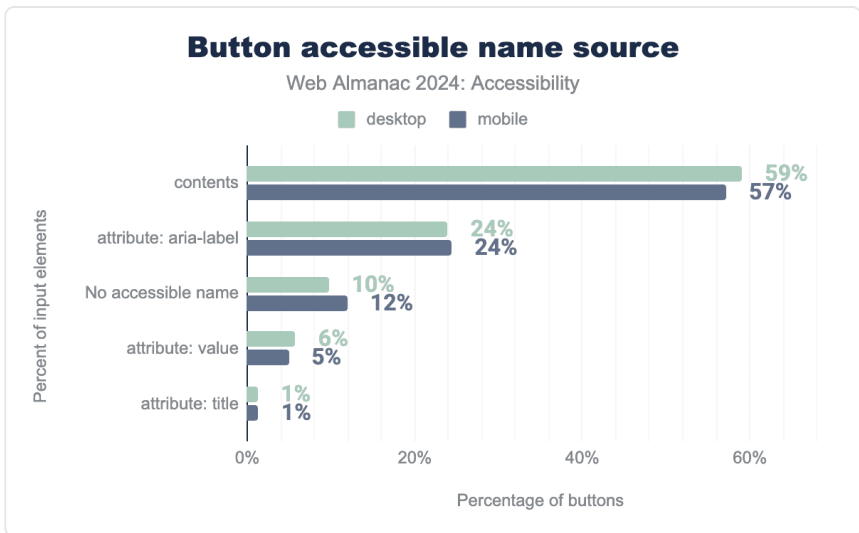


Figure 8.29. Button accessible name source.

Buttons typically receive their accessible names from their content or ARIA attributes. According to ARIA guidelines²⁷⁴, it's preferable for an element to derive its accessible name from its content rather than an ARIA attribute if possible. We found that 59% of buttons on desktop obtain their accessible names from their text content, a slight drop from 2022 when it was 61%. Use of the `aria-label` attribute is up slightly to 23.9% on desktop (from 20% in 2022) meaning more sites are using the `aria-label` attribute for their accessible names.

In some cases, `aria-label` is useful, such as when multiple buttons have the same content

274. <https://www.w3.org/WAI/ARIA/apg/patterns/button/>

but different functions, or when a button contains only an image or icon.

Hiding content

Sometimes, visual interfaces include redundant elements that aren't beneficial for users of assistive technologies. In these cases, `aria-hidden="true"` can be used to hide elements from screen readers²⁷⁵. However, this approach should not be used if removing the element would result in less information for screen reader users compared to what is presented visually. Hiding content from assistive technologies should not be a way to bypass content that is difficult to make accessible.



Figure 8.30. Had at least one element with the `aria-hidden` attribute.

Using this attribute to hide and show semantic content is a common practice in modern interfaces in order to indicate when content is hidden to the accessibility API. . An example of this is a accordion component where content under a list of headings is hidden until a user selects one of the headings to show the related content.

ARIA can have a huge impact on accessibility and needs to be used cautiously. It's crucial to apply ARIA correctly²⁷⁶ to convey the right message.

For instance, disclosure widgets should use the `aria-expanded` attribute to signal to assistive technologies when an element is revealed or hidden by expanding or collapsing. We observed that 34% of mobile pages had at least one element with the `aria-expanded` attribute, which is up almost 5% from 2022.

Screen reader-only text

A common approach developers use to provide extra information for screen reader users involves hiding text visually with CSS while keeping it accessible to screen readers. This CSS technique ensures that the text is included in the accessibility tree but remains hidden from sight.

275. <https://niquette.ca/articles/hiding-elements/>

276. <https://blog.pope.tech/2022/07/12/what-you-need-to-know-about-aria-and-how-to-fix-common-mistakes/>

16%

Figure 8.31. Desktop websites with a `sr-only` or visually-hidden class.

The `sr-only` and visually-hidden class names are frequently used by developers and UI frameworks to create text that is only accessible to screen readers. For instance, Bootstrap and Tailwind include `sr-only` classes for this purpose. We found that 16% of desktop pages and 15% of mobile pages used one or both of these CSS classes (each up a percentage point from 2022). It's important to note that not all screen reader users are visually impaired, so relying too heavily on screen reader-only solutions should be avoided. When this technique is used with an interactive element's accessible name, it can make it difficult for people who use their voice to control their computer to know what command to give to interact with the element.

Dynamically-rendered content

Sometimes, it's necessary to inform screen readers about new or updated content in the DOM. For example, form validation errors should be communicated, while a lazy-loaded image might not need to be announced. Updates to the DOM should be made in a non-disruptive manner.

29%

Figure 8.32. Desktop pages with live regions using `aria-live`.

ARIA live regions enable screen readers to announce changes in the DOM. We found that 29% of desktop pages use live regions with the `aria-live` attribute (up from 23% in 2022) and 28% of mobile pages use `aria-live` (up from 22% in 2022). Additionally, pages use ARIA live region roles with implicit `aria-live` values:

role	desktop	mobile	Implicit aria-live value
status	9.2%	8.7%	polite
alert	6.9%	6.7%	assertive
timer	0.8%	0.8%	off
log	0.6%	0.6%	polite
marquee	0.1%	0.1%	off

Figure 8.33. Pages with live region ARIA roles, and their implicit `aria-live` value.

For more details on live region variants and their usage, check the MDN live region documentation²⁷⁷ or explore this live demo by Deque²⁷⁸.

User Personalization Widgets and Overlay Remediation

Users are increasingly used to seeing accessibility widgets on websites. These allow them to access accessibility features that improve their experience. Accessibility Overlays are one type of these and usually include two types of technology: a personalization widget and a JavaScript overlay. Overlays can be either generic or custom:

- **User personalization:** tools that enable the site visitor to make changes to the appearance of the site via an on-site menu — changes like font or color contrast adjustments, and
- **Automated overlay remediation:** a generic technology that automatically scans for and attempts to remediate many common WCAG issues which affect the user interface, with complex algorithms and/or Artificial Intelligence.
- **Custom overlay remediation:** site specific code written by expert developer(s) to address specific conformance needs, and verified by accessibility experts in context, to avoid conflict with assistive technology.

Browsers have great built-in tools for personalization, but many users do not know about them. Some sites add **personalization widgets** that often provide a range of accessibility features to make customization easier. Often this includes font size, spacing, and contrast, which is

277. https://developer.mozilla.org/docs/Web/Accessibility/ARIA/ARIA_Live_Regions

278. <https://dequeuniversity.com/library/aria/liveregion-playground>

included in the browser”²⁷⁹. This may also include tools like text to speech²⁸⁰, which is included in Edge²⁸¹. This can be useful for a range of users, but especially for those that do not have their own assistive technology available in that environment. These widgets can be helpful for users who are not actively using assistive technology or already maximizing their browser’s built-in accessibility features.

If used, it is important that these tools do not interfere with the user experience (UX) including that of assistive technology users. For that reason, the European Disability Forum (EDF) published a report clearly stating that Accessibility overlays don’t guarantee compliance with European legislation²⁸²:

“Users of assistive technology already have their devices and browsers configured to their preferred settings. The overlay technology can interfere with the user’s assistive technology and override user settings, forcing people to use the overlay instead. This makes the website less accessible to some user groups and may prevent access to content.”

Overlay remediations are the second type of technology often found in an overlay product. Automated overlay remediation continuously tries to find and address common WCAG issues as the page is being rendered in the browser. Custom overlay remediations can also be written in JavaScript to overcome accessibility barriers, especially when there is legacy code which can no longer be updated. With good manual testing, especially with users with disabilities, a custom overlay can be an effective solution.

There are many documented reports of popular automated overlays making a product less accessible for some users.

This technology can address some common barriers for some users, making the site more accessible. Automated overlays can also advance an organization’s accessibility progress and path to compliance by freeing development teams to focus on more complex issues that can only be resolved by addressing the design or source code.

Unfortunately, many teams simply stop investing in accessibility after investing in an overlay.

This technology does not replace the need for good accessibility practices. Accessibility needs to be included in all stages of the product life cycle. Overlays are always going to have more usability, security and performance problems than simply fixing the errors at the source. It is important to remember that no automated tool can make a website fully accessible or WCAG compliant.

279. <https://mcmw.abilitynet.org.uk/>

280. https://wikipedia.org/wiki/Speech_synthesis

281. <https://www.microsoft.com/en-us/edge/features/read-aloud?form=MA13FJ>

282. <https://www.edf-efph.org/accessibility-overlays-dont-guarantee-compliance-with-european-legislation>

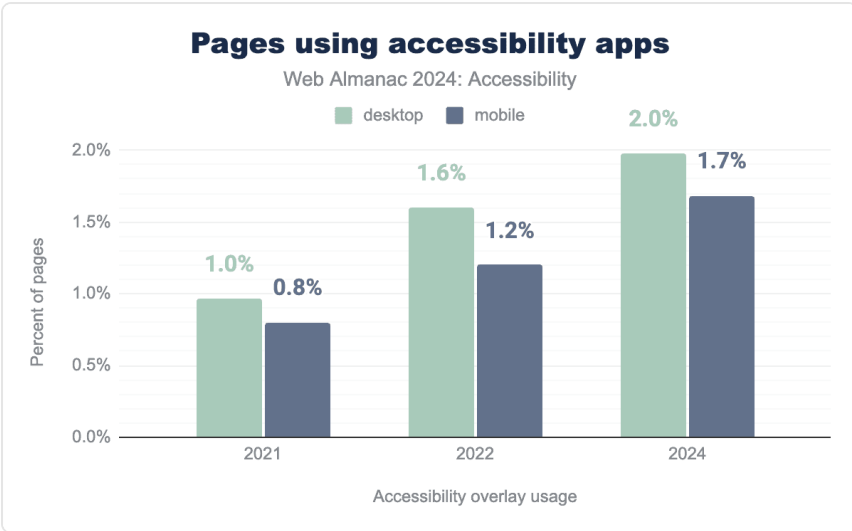


Figure 8.34. Pages using accessibility apps (overlays).

In 2024, we observed that almost 2% of desktop websites utilize known accessibility apps. While not all of these products are accessibility overlays, the detectable overlays show a similar growth trend.

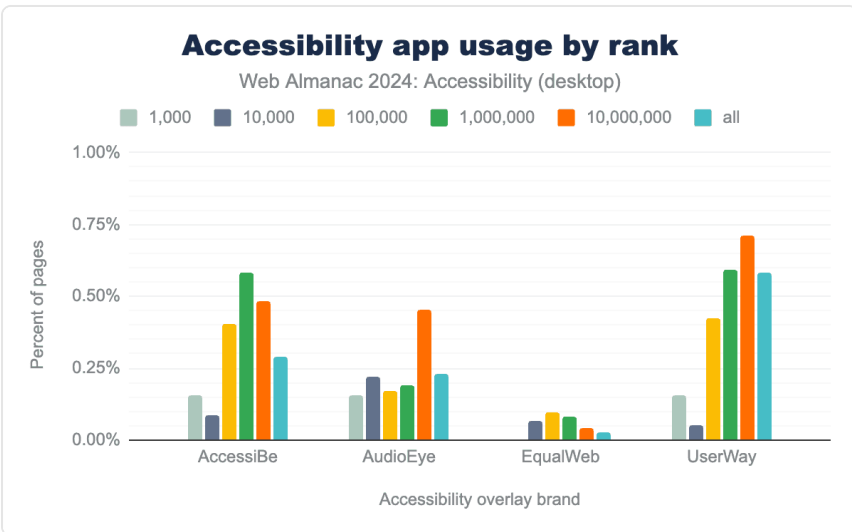


Figure 8.35. Accessibility app usage by rank.

UserWay is the most widely used overlay in our dataset.

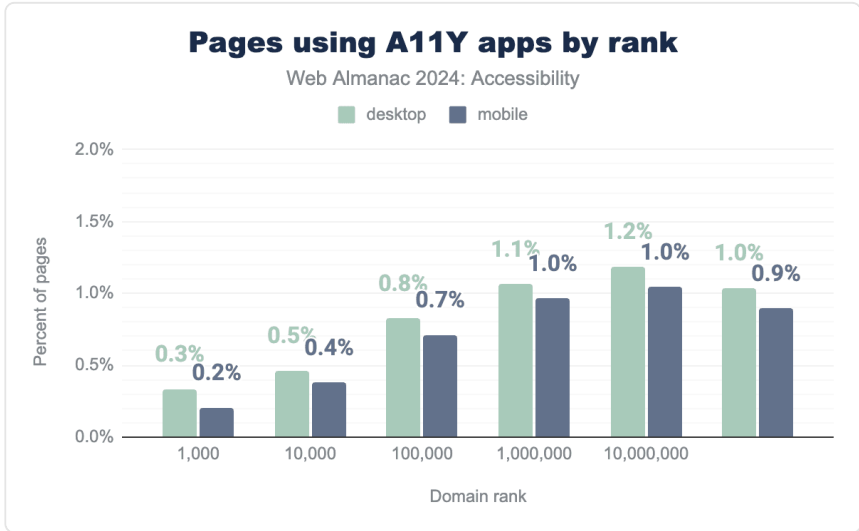


Figure 8.36. Pages using accessibility apps by rank.

These solutions are generally used less for high-traffic websites. For sites ranked in the top 1,000 by visits, only 0.2% use an overlay.

Confusion on Overlays

The International Association of Accessibility Professionals (IAAP)²⁸³ published a paper outlining its Accessibility Overlay Position and Recommendations²⁸⁴. In it, IAAP stresses that overlay technology must never impede a user's access. Furthermore, it states that IAAP members must not support claims that imply a website or application can be made fully accessible with overlay technology.

False advertising claims made by many overlay providers have prompted outcry from accessibility advocates: Adrian Roselli's #accessiBe Will Get You Sued²⁸⁵ initially published in 2020 but actively updated as the case evolved; Lainey Feingold's American legal perspective²⁸⁶.

It is important to clearly understand the capabilities and limitations of any tool or technology used to advance accessibility. False claims by companies about their abilities have confused many clients. Organizations are responsible for doing appropriate research to ensure they

283. <https://www.accessibilityassociation.org/>

284. <https://www.accessibilityassociation.org/s/overlay-position-and-recommendations>

285. <https://adrianroselli.com/2020/06/accessiBe-will-get-you-sued.html>

286. <https://www.lflegal.com/2023/07/adrian-roselli-slopp-lawsuit/>

meet applicable accessibility requirements and provide the best experience to visitors.

Neither the EU Commission or the US Department of Justice (DOJ) state how web accessibility standards have to be met—just that they must be met. From the DOJ ADA Title II rulemaking²⁸⁷ the rule “does not address the internal policies or procedures that public entities might implement to conform to the technical standard under this rule.”

In some instances, a combination of overlays and manual expertise has the potential to accelerate accessibility improvements.

Sectors and accessibility

This year we are providing a series of new data comparisons. We want to highlight that there are discernible differences in how different communities have handled accessibility. Whether it is based on good governance, or good defaults, it is possible to see differences in accessibility that are significant. It is the hope of the authors of this section that this will prompt a review of how the various communities treat accessibility.

We also assessed the accessibility of websites in this section using the open source tool, Google Lighthouse²⁸⁸.

Country

There are two means by which we can identify country information, first by the GeoID of the server, and the second by the Top Level Domain. Because of the price of hosting in different countries, some are much better represented by GeoID than others. Likewise, given that many domains can operate independently of the country like the `.ai` or `.io` domain, we can't assume that all `.ca`, `.es`, or `.fi` domains are located in Canada, Spain or Finland.

287. <https://www.federalregister.gov/documents/2024/04/24/2024-07758/nondiscrimination-on-the-basis-of-disability-accessibility-of-web-information-and-services-of-state>

288. <https://developer.chrome.com/docs/lighthouse/accessibility/scoring>

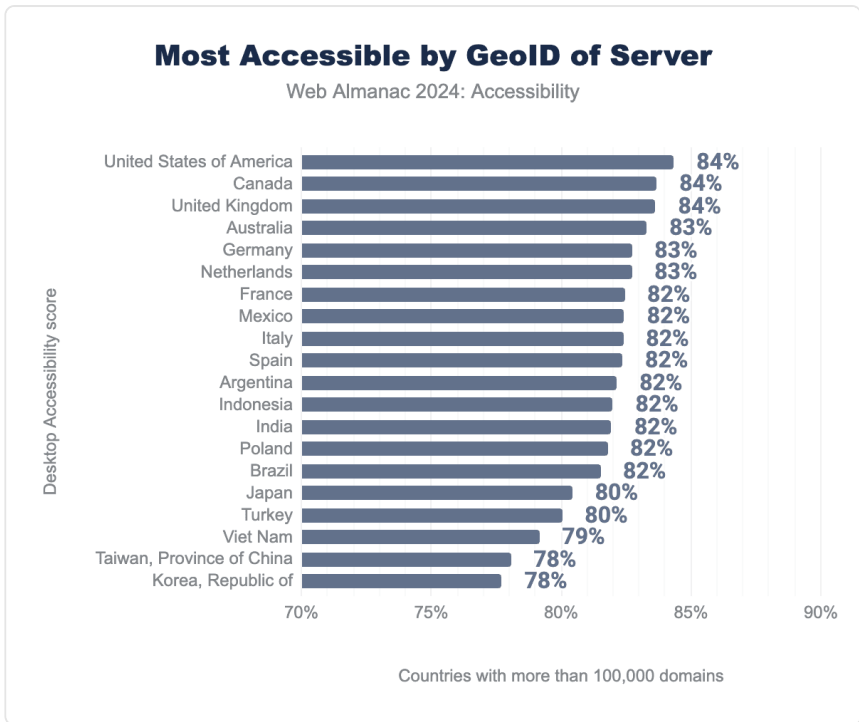


Figure 8.37. Most accessible countries by GeoID.

It is worth noting that many sites that operate in the USA are subject to the Section 508 guidelines on accessibility. Organizations are being sued in the USA, under ADA Title III, for not having accessible websites. It is not surprising that the USA is the most accessible country. Other jurisdictions are beginning to penalize companies that sell inside their geography or to their citizens. Increasingly people are looking at the European Accessibility Act²⁸⁹ and preparing for the new requirements that will be introduced in 2025.

The following map shows the average desktop accessibility score by country top level domain (TLD).

289. https://wikipedia.org/wiki/European_Accessibility_Act

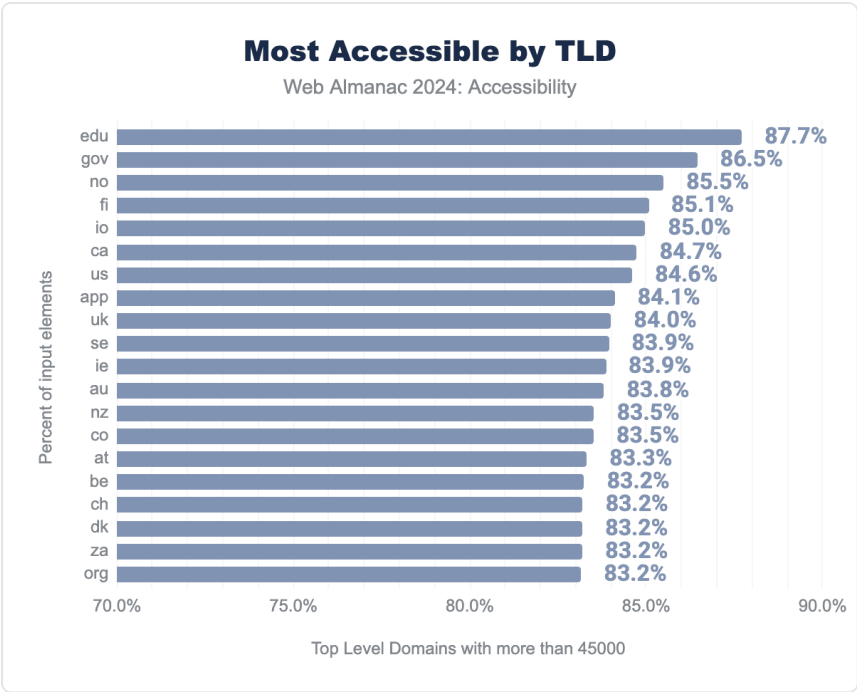


Figure 8.38. Accessible countries by Top Level Domain (TLD).

But it is a bit easier to see the TLD ranked and including the non-country codes as well.

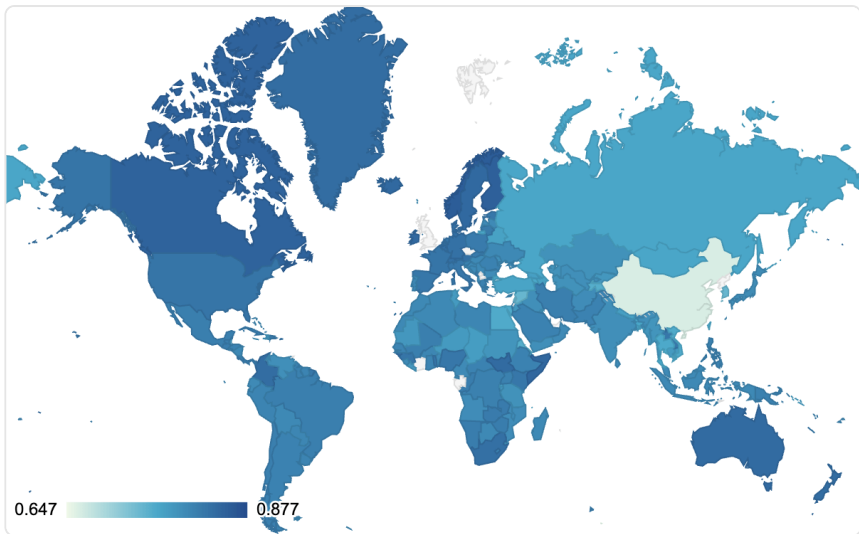


Figure 8.39. Map of the accessible countries by Top Level Domain (TLD).

As with the prior chart `.edu` and `.gov` domains are the most accessible. The US Government under Section 508 and Section 504²⁹⁰, have had this as part of their mandate for more than two decades. Early accessibility legislation and active lawsuits²⁹¹ have driven accessibility adoption in the United States. Countries outside the USA started providing legislation and enforcement measures for WCAG conformance later. Lainey Feingold maintains a great list of global law and policy²⁹².

Government

Not all government domains follow consistent accessibility rules, however we were able to isolate many countries' government sites. Some countries are inconsistent about naming government sites, so there will be exceptions which are not covered. We have collected averages for most government agencies around the world.

290. <https://www.levelaccess.com/compliance-overview/section-504-compliance/>

291. <https://www.accessibility.com/digital-lawsuits>

292. <https://www.lflegal.com/global-law-and-policy/>

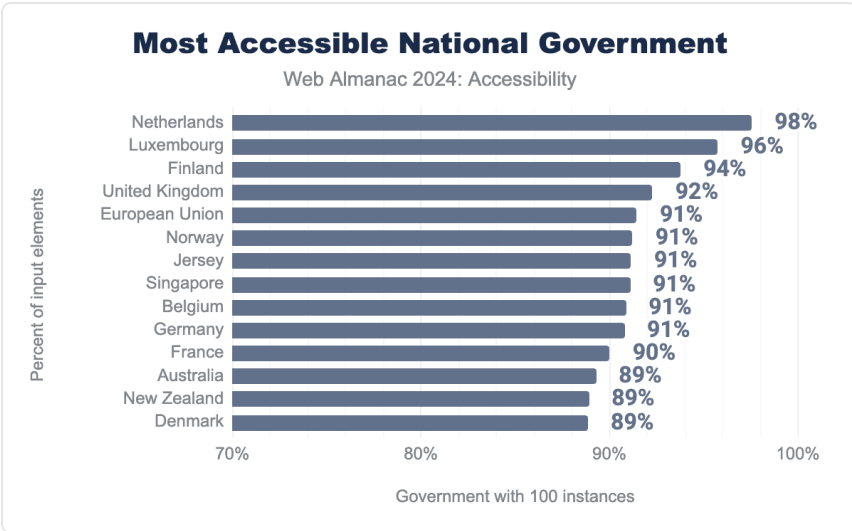


Figure 8.40. Most accessible government websites.

Most modern governments have committed to either WCAG 2.0 AA or WCAG 2.1 AA. It is clear that the implementation of these policies isn't being equally delivered. This is particularly important when looking into accessibility within the European Union where each member state needs to implement legislation based on the Web Accessibility Directive²⁹³. It should be possible to compare the 3-year EU member state reports²⁹⁴ with the values provided here and in future Web Almanacs. It is worth noting that the average for the United States is 87%.

293. <https://digital-strategy.ec.europa.eu/en/policies/web-accessibility-directive-standards-and-harmonisation>

294. <https://digital-strategy.ec.europa.eu/en/library/web-accessibility-directive-monitoring-reports>

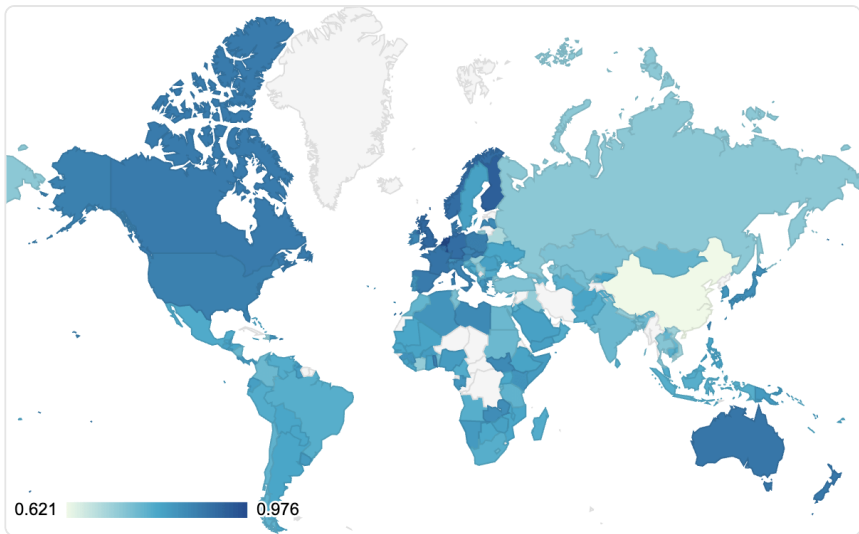


Figure 8.41. Map of the accessibility of global government websites.

The Netherlands (98%) are firmly in the lead, followed by Luxembourg (96%) and Finland (94%). The United Kingdom²⁹⁵ and the Netherlands²⁹⁶ both have a standardized design system which is prioritizes accessibility. What contributes to Luxembourg and Finland's success? Considering that most accessibility content is available only in English, has this reduced adoption by some governments?

Government domains were largely found based on domain name pattern matching. There are a lot of inconsistencies in how governments use domain names, but there is enough information here to provide comparisons. It is worth noting that `.gov` covers all levels of the US government, so we have tried to filter out those state specific sub domains except in the state specific reporting. In this report, we could not filter out municipal or regional `.gov` sites. When looking at the TLD `.gov` domain chart above the average was 87%.

We can also review the accessibility of various states.

295. <https://design-system.service.gov.uk>

296. <https://github.com/nl-design-system>

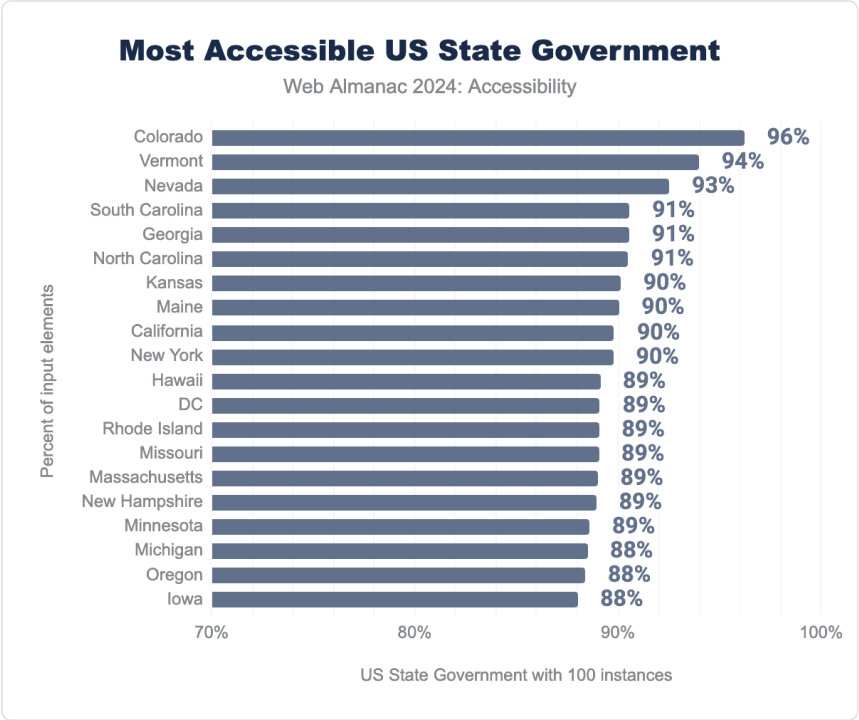


Figure 8.42. The most accessible US state governments.

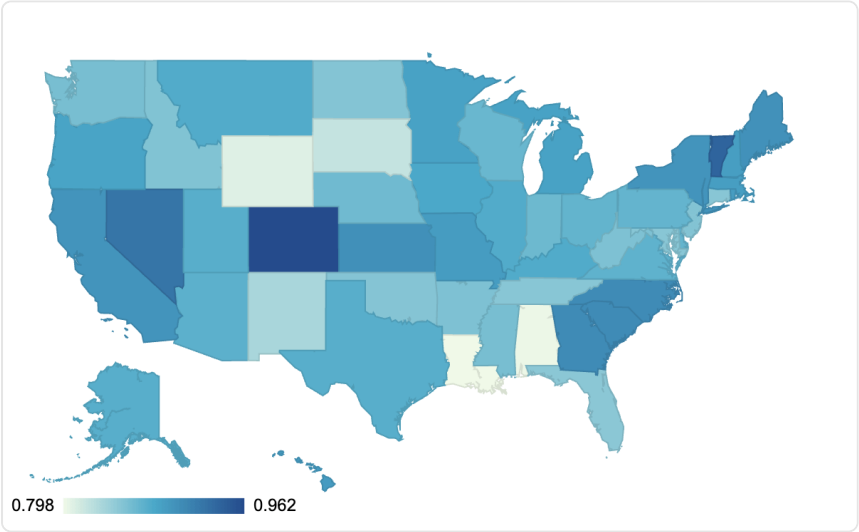


Figure 8.43. Map of the most accessible US state governments.

Again, Colorado and Vermont are much further ahead than other states. Colorado has established a centralized Statewide Internet Portal Authority (SIPA)²⁹⁷, along with new accessibility legislation²⁹⁸ and now has an average of 96%. The state of Georgia has a central Drupal installation²⁹⁹ managed through a central agency, does this explain why it is in the top 5? Pennsylvania's state average is much lower at 82% but they also have a new digital experience team³⁰⁰ established in 2023.

Earlier this year, the US Department of Justice updated its regulations for Title II of the Americans with Disabilities Act (ADA)³⁰¹. US state and local governments will now all be required to be fully WCAG 2.1 AA compliant. The compliance date depends on the size of their population but will be either April 2026 or April 2027. It will be important to measure how US states comply with this new regulation. We should see improvements in these numbers.

Content Management Systems (CMS)

The WebAim Million³⁰² study reviewed CMS data, and we are able to provide comparable results through the Web Almanac. The Web Almanac uses a customized version of a fork of Wappalyzer, from when it was open source. With this the report can identify which CMS is used and compare results. It is clear that Typo3 had better results in WebAim than when using Google Lighthouse data. Both studies clearly indicated that the choice of CMS had an impact on accessibility.

When most folks think about CMS, they think about the ones that you can download and install yourself. This is predominantly made up of open source tools, but not exclusively. Adobe Experience Manager (AEM), Contentful and Sitecore were the most accessible three in this list of top 10. A possible explanation for this is that closed-source software like AEM is more likely to be used by larger corporations, which have more resources to address accessibility issues. Additionally, open-source software gives website owners a lot of freedom, which in some cases can lead to worse accessibility.

297. <https://sipa.colorado.gov/>

298. <https://oit.colorado.gov/accessibility-law>

299. <https://digital.georgia.gov/services/govhub>

300. <https://code.oh.pa.gov/>

301. <https://www.ada.gov/resources/2024-03-08-web-rule/>

302. <https://webaim.org/projects/million/>

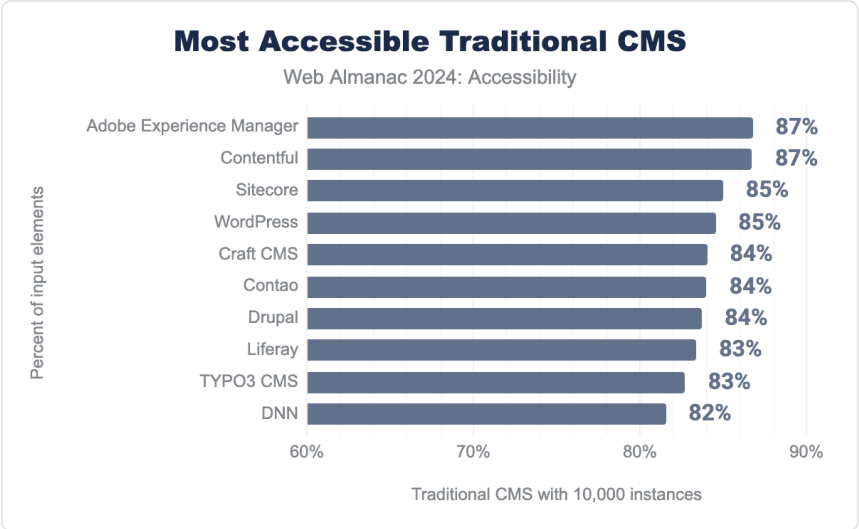


Figure 8.44. A bar chart with the accessible traditional Content Management Systems (CMS).

Looking at audits of these Traditional CMS, the top four Lighthouse issues have a great deal of consistency. Color contrast, link name, heading order and alt text are regular problems across these CMS, those issues are mainly related to the user since a CMS can not be responsible for the chosen colors or the naming of the links.

Traditional CMS	Most popular	2nd most	3rd most	4th most
Adobe Experience Manager	color-contrast	link-name	heading-order	label-content-name-mismatch
Contentful	color-contrast	link-name	heading-order	image-alt
Sitecore	color-contrast	link-name	heading-order	image-alt
WordPress	color-contrast	link-name	heading-order	target-size
Craft CMS	color-contrast	link-name	heading-order	image-alt

Figure 8.45. Top accessibility audit issues for popular CMS'.

The different CMS do have a lot of commonalities in the top errors that they have. They mostly have to do with content issues, which is something that ATAG 2.0 was written to support. It is hoped that the best practices of ATAG will be brought into WCAG 3.0. This scan is only for publicly available websites, so authoring interfaces are not evaluated. It is worth noting that authors have disabilities, and authors should be able to expect an accessible interface. Authors also need support in creating accessible content. To help facilitate greater focus on authoring tools, the W3C produced a ATAG Report Tool³⁰³.

There are many tools which can be used to help authors evaluate the accessibility of a page. Institutions that control the browser configurations of their staff, could choose to simply install the open source Accessibility Insights³⁰⁴ browser plugin for all of their browsers. This would make errors much more visible to administrators. For many of the CMS above though, the best solution might be to install a tool like Sa11y³⁰⁵ or Editoria11y³⁰⁶ which is geared to help authors. From Joomla version 4.1 onwards Sa11y is included by default³⁰⁷, so all authors benefit.

Website platforms in general performed better than the Traditional CMS with Wix, Squarespace and Google Sites being significantly better.

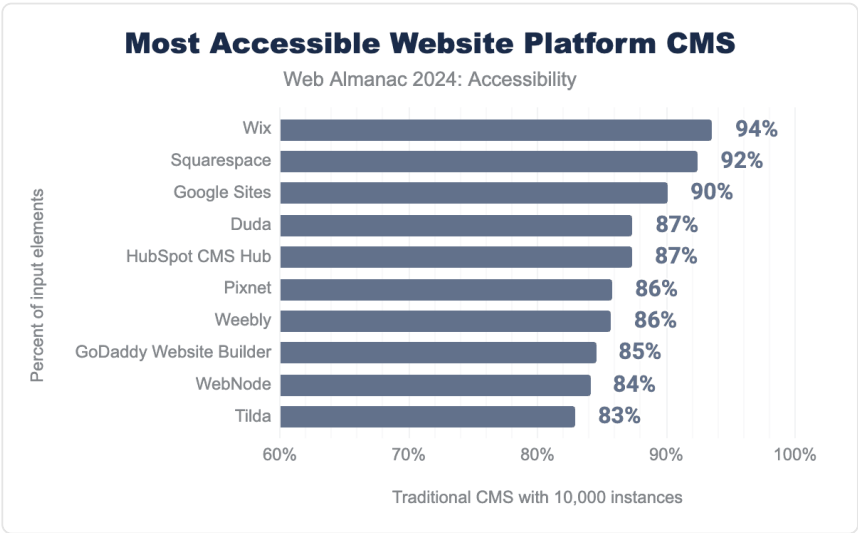


Figure 8.46. A bar chart with the most accessible Website Platform Content Management Systems (CMS).

Looking at audits of these CMS Platforms, the top four Lighthouse issues have less consistency

303. <https://www.w3.org/WAI/atag/report-tool/>
304. <https://accessibilityinsights.io/docs/web/getstarted/assessment/>
305. <https://sa11y.netlify.app/>
306. <https://editoria11y.princeton.edu/>
307. <https://sa11y.netlify.app/joomla/>

in the frequency of issues but still have lots of similarities. Alternative text, link name, heading order and color contrast are all still issues, but just with different rates of occurrence.

Platform CMS	Most popular	2nd most	3rd most	4th most
Wix	heading-order	link-name	button-name	color-contrast
Google Sites	image-alt	link-name	aria-allowed-attr	heading-order
Duda	link-name	color-contrast	image-alt	heading-order
HubSpot CMS	color-contrast	heading-order	link-name	target-size
Pixnet	heading-order	link-name	color-contrast	frame-title

Figure 8.47. Top accessibility audit issues for popular CMS platforms.

Different CMS platforms have varying strengths and weaknesses. For example, it’s clear that ARIA components must have accessible names, yet 36% of websites built with GoDaddy Website Builder fail this test, while the median failure rate for all CMS platforms with more than 100,000 occurrences in our dataset is just 1%. GoDaddy is also an outlier in the area of dialog names, with 14% of tests failing compared to a mean failure rate of 1.3%.

On the positive side, Duda stands out for button names, where only 3% of its websites fail the test, compared to a median of 13%. Even more impressive is Wix only 20% of Wix websites fail the Lighthouse test for color contrast, while the median failure rate among the most-used CMS platforms is 70%. Similarly, Wix performs exceptionally well regarding alternative text for images, with only 1% failing, compared to a median of 34%.

The differences show that it is possible for CMS to make an impact on accessibility even when the author needs to take the last step to make content accessible.

JavaScript Frontend Frameworks

WebAim Million³⁰⁸ also looks at the impact of JavaScript frameworks and libraries. Again it is

308. <https://webaim.org/projects/million/#frameworks>

possible to see patterns in the data based on the libraries used. We have worked with the definitions from the State of JavaScript 2023³⁰⁹.

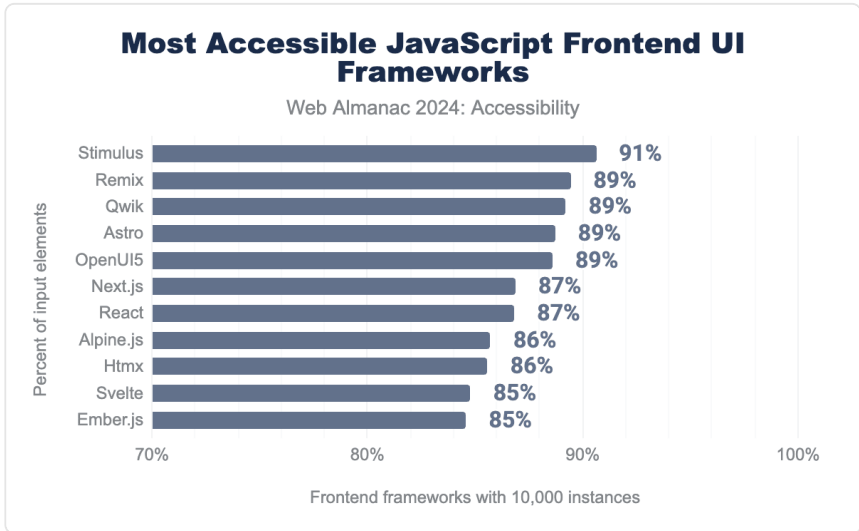


Figure 8.48. Most Accessible JavaScript Frontend UI Frameworks.

Stimulus, Remix and Qwik are several percent more accessible on average than React, Svelte or Ember.js.

309. <https://2023.stateofjs.com/>

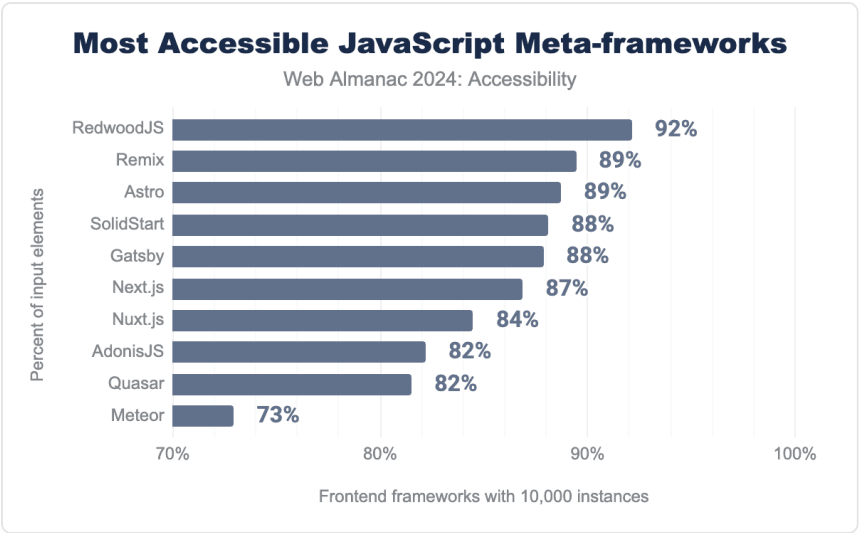


Figure 8.49. Most Accessible JavaScript Meta-frameworks.

RedwoodJS is clearly the most accessible, followed by Remix and Astro.

Conclusion

Our analysis indicates that there hasn't been a significant change in web accessibility. While there have been some improvements, many straightforward issues remain unresolved. Improving color contrast and use of image `alt` attributes could have a substantial impact if addressed. CMS systems and JavaScript frameworks have a huge responsibility and examples prove that they can have real positive impact on accessibility.

We often observe that features intended to enhance accessibility can sometimes create a false sense of improvement, while actually degrading the user experience. Many of these accessibility problems could be avoided if designers and developers integrated accessibility considerations from the start rather than treating them as an afterthought. Organizations must prioritize accessibility training, operations, and budgets to enable the development of more accessible user experiences. Some governments have demonstrated how effective that approach is.

The web community must understand that a website only offers an excellent customer experience when it accommodates everyone. In 2024 we should not be discriminating against people based on the device, browser or assistive technology used. We have focused on key metrics that are straightforward to address and are hoping to see more improvements in 2025.

Author



Mike Gifford

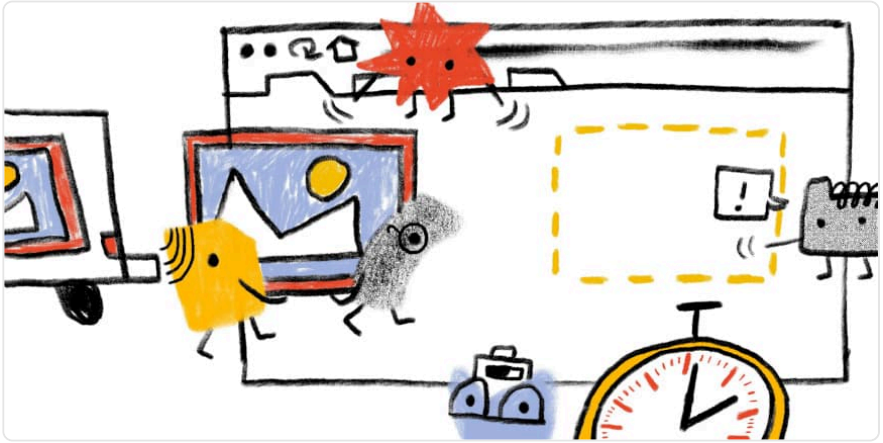
 [@https://mastodon.social/@mgifford](https://mastodon.social/@mgifford)  [@mgifford.bsky.social](https://twitter.com/mgifford.bsky.social)  [mgifford](https://github.com/mgifford)  [mgifford](https://www.linkedin.com/in/mgifford)

 <https://accessibility.civicactions.com/>

Mike Gifford is CivicActions' Open Standards & Practices Lead. He is also a thought leader on open government, digital accessibility and sustainability. He has served as a Drupal Core Accessibility Maintainer and also a W3C Invited Expert. He is a recognized authoring tool accessibility expert and contributor to the W3C's Draft Web Sustainability Guidelines (WSG) 1.0.

Part II Chapter 9

Performance



Written by Jevgenija Zigisova and Ines Akrap

Reviewed by Rick Viscomi and Sia Karamalegos

Analyzed by Kevin Farrugia and Estela Franco

Edited by James Ross

Introduction

No one ever complained about a fast website, but a slow-loading and sluggish website quickly frustrates users. Website speed and overall performance directly impact user experience and the success of a website. Moreover, if a website is slow, it becomes less accessible to users, which is against the fundamental goal of the web—to provide universal access to the universe of information.

In recent years, Core Web Vitals³¹⁰ performance metrics have improved, showing positive trends across many performance metrics. However, some inconsistencies can be observed. For example, the gap between high-end and low-end devices is widening, especially in mobile web performance, as highlighted in Alex Russell's research in The Performance Inequality Gap³¹¹. Web performance is tied to what devices and networks people can afford. Fortunately, more developers are aware of these challenges and are actively working to improve performance.

310. <https://web.dev/articles/vitals>

311. <https://infrequently.org/2024/01/performance-inequality-gap-2024/>

In the performance chapter, we focus on Core Web Vitals, as they are key user-centric metrics³¹² for assessing web performance. However, we also analyze the web performance from a broader perspective: loading, interactivity, and visual stability, adding supportive metrics like First Contentful Paint. This allows us to explore other performance and user experience-related metrics to get a more comprehensive picture of how websites performed in 2024.

What's new this year?

- Interaction to Next Paint (INP) has officially replaced First Input Delay (FID)³¹³ as part of Core Web Vitals. INP helps to evaluate overall interactivity performance more accurately.
- Long Animation Frames (LoAF)³¹⁴ data is available for the first time, providing new insights into the reasons for poor INP.
- As of this year, the Performance chapter also includes an analysis of the data for secondary pages in addition to home pages. This allows us to compare the home page with the secondary page performance.

Notes on data sources

The HTTP Archive contains only lab performance data. In other words, it is data from a single website load event. This is useful but limited if we want to understand how users experience performance.

Thus, in addition to the HTTP Archive data, most of this report is based on real user data from the Chrome User Experience Report (CrUX)³¹⁵. Note that while Chrome is the most widely used browser worldwide, it doesn't reflect performance across all browsers and all regions of the world.

CrUX is a great source of data, but it doesn't contain certain metrics like LCP and INP sub-parts, as well as Long Animation Frames. Luckily, the performance monitoring platform RUMvision³¹⁶ has provided us with this data for the period from 1st January to 6th October 2024. Compared to The HTTP Archive, RUMvision tests a smaller amount of websites, which is why the results for the same metrics might be slightly different.

312. <https://web.dev/articles/user-centric-performance-metrics>

313. <https://web.dev/blog/inp-cww-march-12>

314. <https://developer.chrome.com/docs/web-platform/long-animation-frames>

315. <https://developer.chrome.com/docs/crux>

316. <https://www.rumvision.com/>

Core Web Vitals

Core Web Vitals (CWV) are user-centric metrics designed to measure the different aspects of web performance. These include the Largest Contentful Paint (LCP)³¹⁷, which tracks loading performance, Interaction to Next Paint (INP)³¹⁸, which measures interactivity, and Cumulative Layout Shift (CLS)³¹⁹, which assesses visual stability.

Starting this year, INP has officially replaced First Input Delay (FID)³²⁰ and became a part of the CWV. While INP measures the full delay of all interactions experienced by a user, FID only focuses on the input delay of the first interaction. This wider scope makes INP a better reflection of the full user experience.

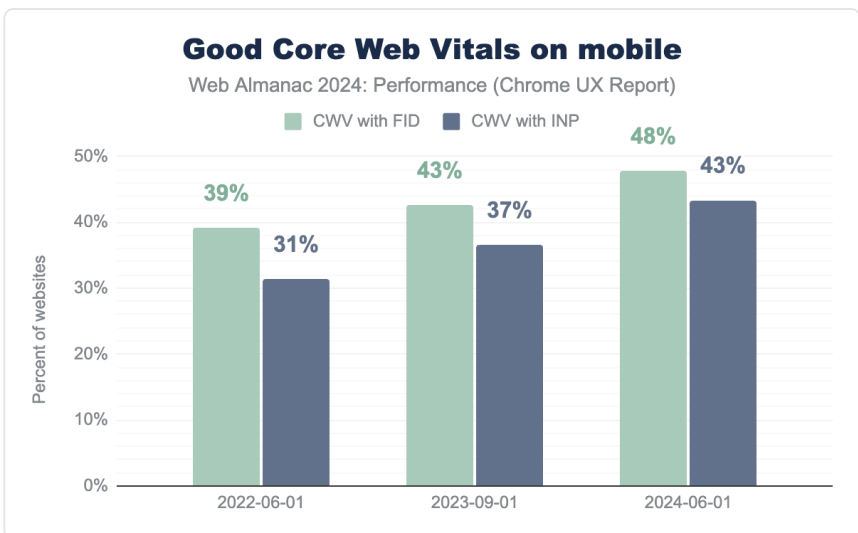


Figure 9.1. The percent of websites having good CWV using FID and INP, segmented by year.

The replacement of the FID with the INP metric significantly impacted the percentage of websites with good CWV on mobile. This doesn't mean the user experience has worsened, just that is now reflected more accurately due to the metric update. If we still used FID as a measure of interactivity, 48% of the websites would have good CWV on mobile devices. However, with the INP metric, this figure drops to 43%. Interestingly, performance on desktop devices stays the same regardless of which responsiveness metric we use at 54%.

In the period from 2020 to 2022, we saw that mobile web performance measured by CWV with

317. <https://web.dev/articles/lcp>

318. <https://web.dev/articles/inp>

319. <https://web.dev/articles/cls>

320. <https://web.dev/articles/fid>

FID was improving faster than desktop one, and the gap between them was closing, reaching just 5% in 2022. As CWV with INP chart shows, in 2024, the websites on the desktop performed 11% better than on mobile, so the introduction of the INP shows that the gap is much bigger.

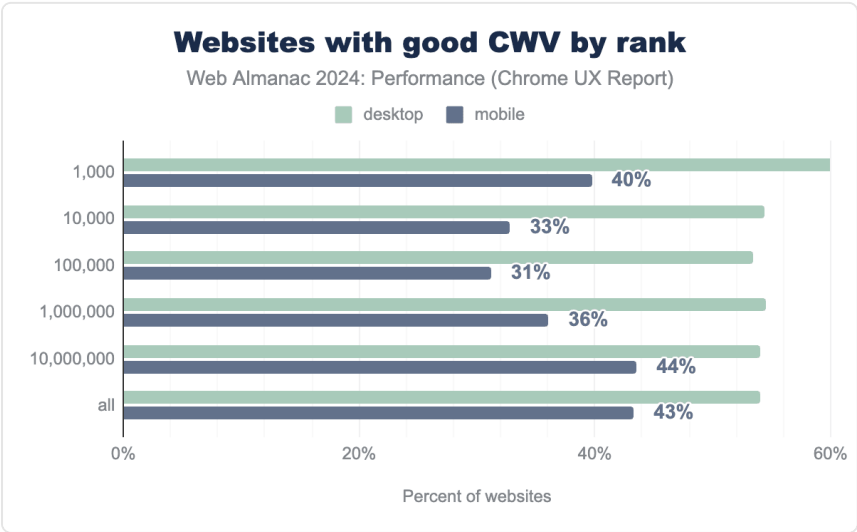


Figure 9.2. The percent of websites having good CWV, segmented by rank and desktop vs mobile.

CWV with INP shows a new tendency when analyzing websites by rank. Previously, the most popular websites tended to have the best CWV experience³²¹, however, this year’s statistics show the opposite: 40% of 1000 most popular websites on mobile have good CWV which is lower than total website CWV of 43%.

321. <https://almanac.httparchive.org/en/2022/performance#fig-2>

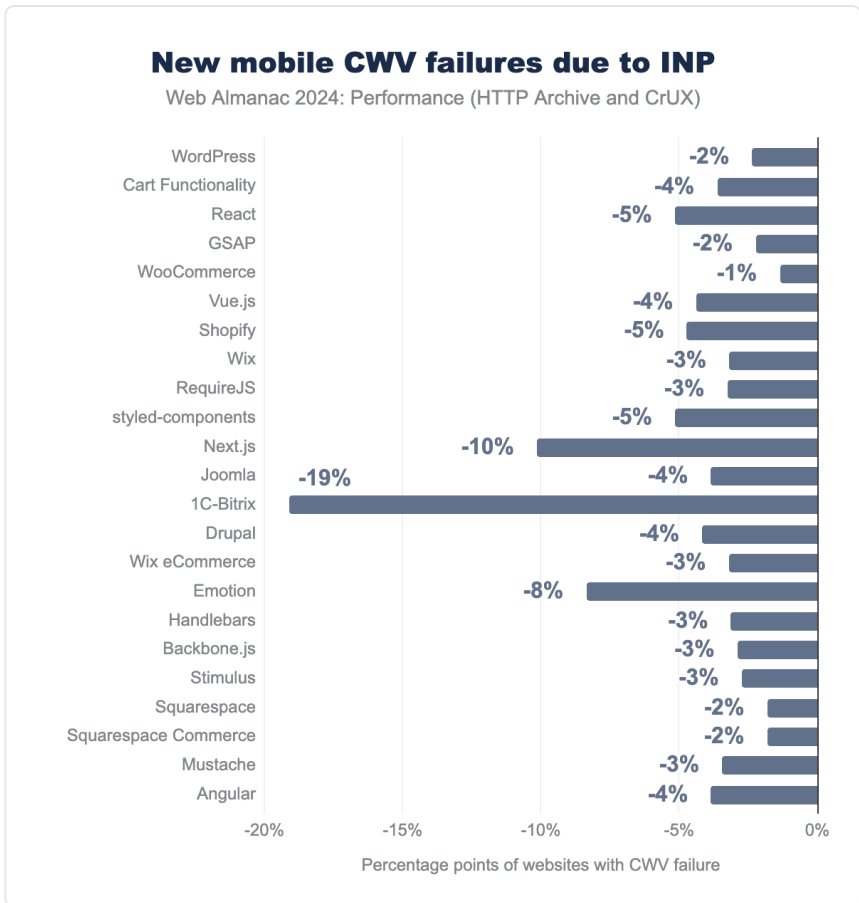


Figure 9.3. Percent point change of websites having good CWV from FID to INP, by technology.

As mentioned earlier, the CWV scores have decreased due to the switch of the INP metric. We investigated how different technologies have been affected by this shift. The diagram above illustrates the percent point drop in the percentage of websites with good CWV across various technologies after the INP was introduced.

Several technologies were significantly impacted, including a 19% drop for 1C-Bitrix (a popular CMS in Central Asia), a 10% drop for Next.js (a React-based framework), and an 8% drop for Emotion (a CSS-in-JS tool). We can't be entirely certain that the decline in CWV scores is solely due to the technology used. Next.js has server-side rendering (SSR) and static site generation (SSG) features, which should theoretically enhance INP, but it has still seen a significant decline. As Next.js is based on React, many websites rely on client-side rendering, which can negatively impact INP. This could serve as a reminder for developers to leverage the SSR and SSG

capabilities of the framework they use.

As of this year, secondary pages are available to compare with home page data.

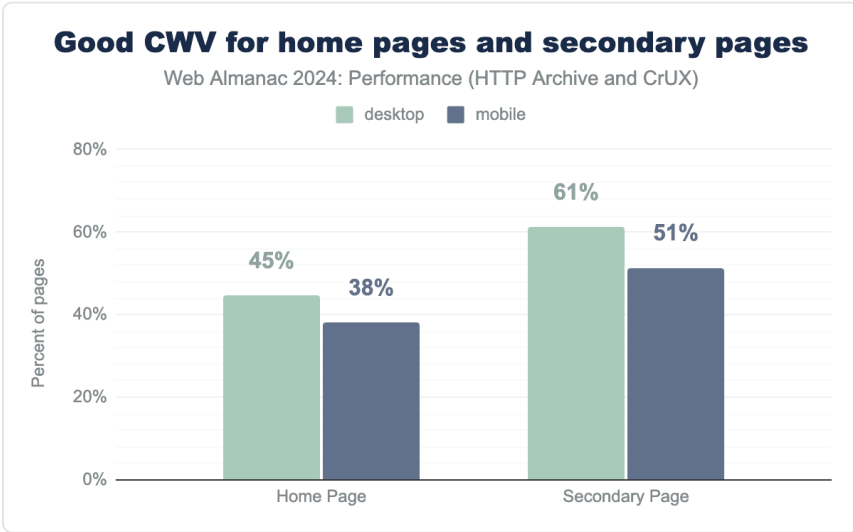


Figure 9.4. The percent of websites having good CWV, segmented by page type.

Secondary pages demonstrate significantly better CWV results than home pages. The percentage of the desktop secondary pages with good CWV is by 14 percentage points better than for home pages. For mobile websites, the difference is 13 percentage points. By looking at CWV data only, it is hard to identify what kind of performance experience is better. We will explore these aspects—layout shift, loading, and interactivity—in the corresponding sections.

Loading speed

People often refer to website loading speed as a single metric, but in fact, the loading experience is a multi-stage process. No single metric fully captures all aspects of what makes up loading speed. Every stage has an impact on the speed of a website.

Time to First Byte (TTFB)

Time to First Byte³²² (TTFB) measures the time from when a user initiates loading a page until the browser receives the first byte of the response. It includes phases like redirect time, DNS

322. <https://web.dev/articles/ttfb>

lookup, connection and TLS negotiation, and request processing. Reducing latency in connection and server response time can improve TTFB. 800 milliseconds is considered the threshold for good TTFB—with some caveats!³²³

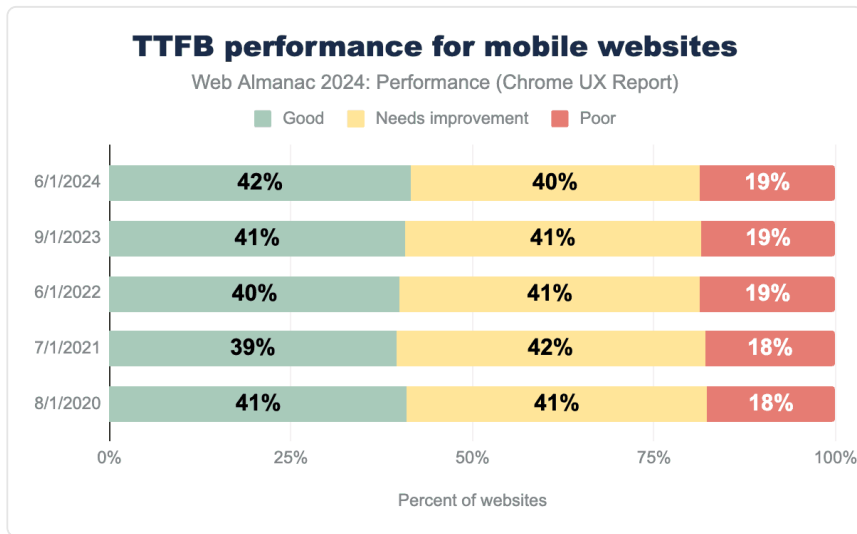


Figure 9.5. The percent of websites having good TTFB, segmented by device and year.

Over the past five years, the percentage of mobile web pages with good TTFB has remained stable, from 41% in 2021 to 42% in 2024. The percentage of pages that need TTFB improvements has decreased by 1%, and unfortunately, the percentage of pages with poor TTFB remains the same. Since this metric has not changed significantly, we can conclude that there have been no major improvements in connection speed or backend latency.

First Contentful Paint (FCP)

First Contentful Paint (FCP)³²⁴ is a performance metric that helps indicate how quickly users can start seeing content. It measures the time from when a user first requests a page until the first piece of content is rendered on the screen. A good FCP should be under 1.8 seconds.

323. <https://web.dev/articles/ttfb#good-ttfb-score>

324. <https://web.dev/articles/fcp>

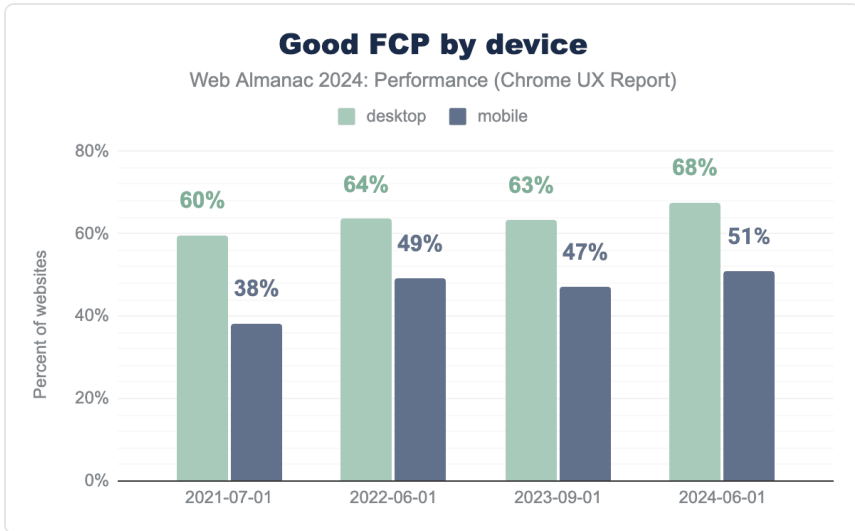


Figure 9.6. The percent of websites having good FCP, segmented by device and year.

FCP has shown improvements over the past few years. Although there was a slight decline in 2023, the metric recovered in 2024, reaching 68% for desktop and 51% for mobile websites. Overall, this reflects a positive trend in how fast the first content is loaded. Taking into account that the TTFB metric remained mostly unchanged, FCP improvements might be driven by client-side rendering rather than server-side optimizations.

Interestingly, website performance is not the only factor that influences FCP. In the research *How Do Chrome Extensions Impact Browser Performance?*³²⁵ Matt Zeunert found that browser extensions can significantly affect page loading times. Many extensions start running their code as soon as a page starts loading, delaying the first contentful paint. For instance, some extensions can increase FCP from 100 milliseconds to 250 milliseconds.

Largest Contentful Paint (LCP)

Largest Contentful Paint (LCP)³²⁶ is an important metric as it indicates how quickly the largest element in the viewport is loaded. A best practice is to ensure the LCP resource starts loading as early as possible. A good LCP should be under 2.5 seconds.

325. <https://www.debugbear.com/blog/chrome-extension-performance-2021#impact-on-page-rendering-times>

326. <https://web.dev/articles/lcp>

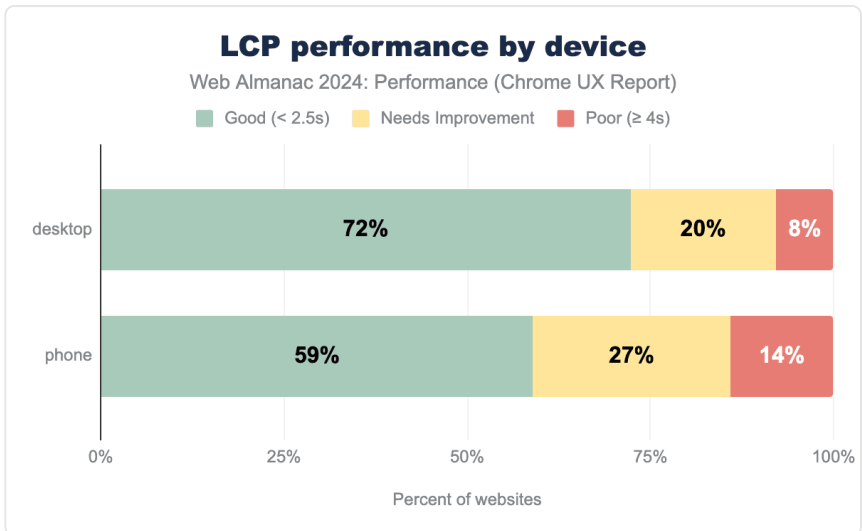


Figure 9.7. The percent of websites having good, need improvements and poor LCP, segmented by device.

LCP has also improved in recent years (from 44% of pages with good LCP in 2022 to 54% in 2024) following the overall positive tendency in CWV. In 2024, 59% of mobile pages achieved a good LCP score. However, there is still a significant gap compared to desktop sites, where 74% have good LCP. This firmly established trend is explained by differences in device processing power and network quality. However, it also highlights that many web pages are still not optimized for mobile use.

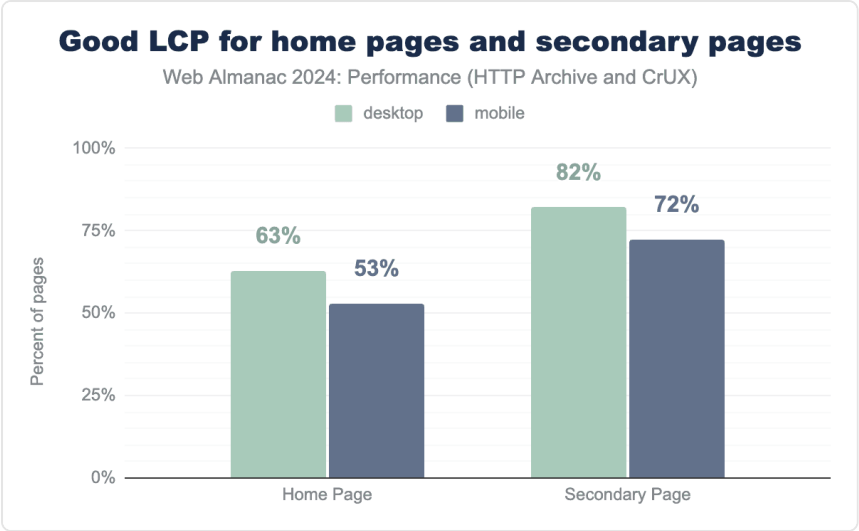


Figure 9.8. The percent of websites having good LCP, segmented by device and page type.

The comparison between home pages and secondary pages reveals an interesting trend: 72% of all secondary pages have good LCP, which is 20% higher than the result for home pages. This is likely because users typically navigate on the home page first, causing the initial load to happen on the home page. After they navigate to secondary pages, many of the resources are already loaded and cached, speeding up the LCP element to render. Another possible reason is that the home page often contains more media-rich content such as video and images, compared to secondary pages.

LCP content types

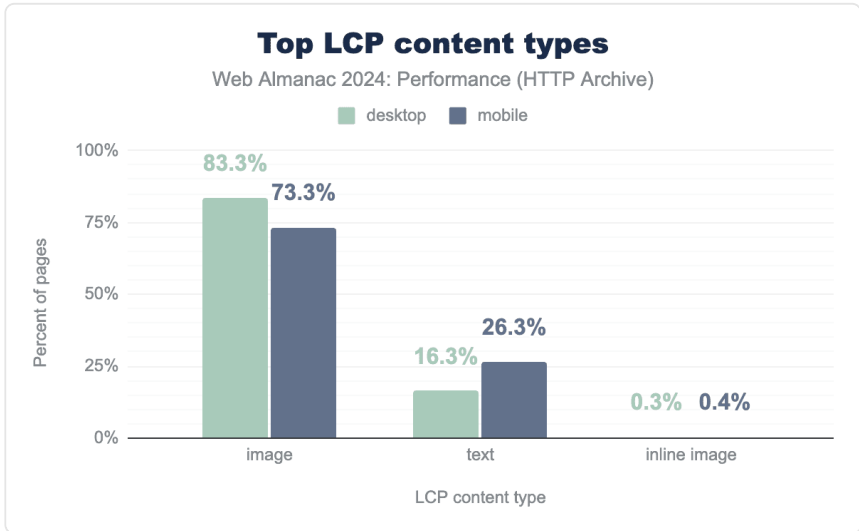


Figure 9.9. Top three LCP content types segmented by device.

Most LCP elements, or 73% of mobile pages, are images. Interestingly, this percentage is 10% higher on desktop pages. The situation is reversed for text content. Compared to desktop, 10% more mobile webpages use text as their LCP element. This difference is likely because desktop websites can accommodate more visual content due to larger viewport sizes and generally higher performance.

LCP sub-parts

Several stages of processing must occur before the LCP element can be fully rendered:

- **Time to First Byte (TTFB)**, which is the time it takes the server to begin responding to the initial request.
- **Resource Load Delay**, which is how long after TTFB the browser begins loading the LCP resource. The LCP elements that originate as inline resources, such as text-based elements or inline images (data URIs), will have a 0 millisecond load delay. Those that require another asset to be downloaded, like an external image, might experience a load delay.
- **Resource Load Duration** which measures how long it takes to load the LCP

resource; this stage is also 0 millisecond if no resource is needed.

- **Element Render Delay** which is the time between when the resource finished loading and the LCP element finished rendering.

In the article [Common Misconceptions About How to Optimize LCP³²⁷](#), Brendan Kenny analyzed a breakdown of LCP sub-parts using recent CrUX data.

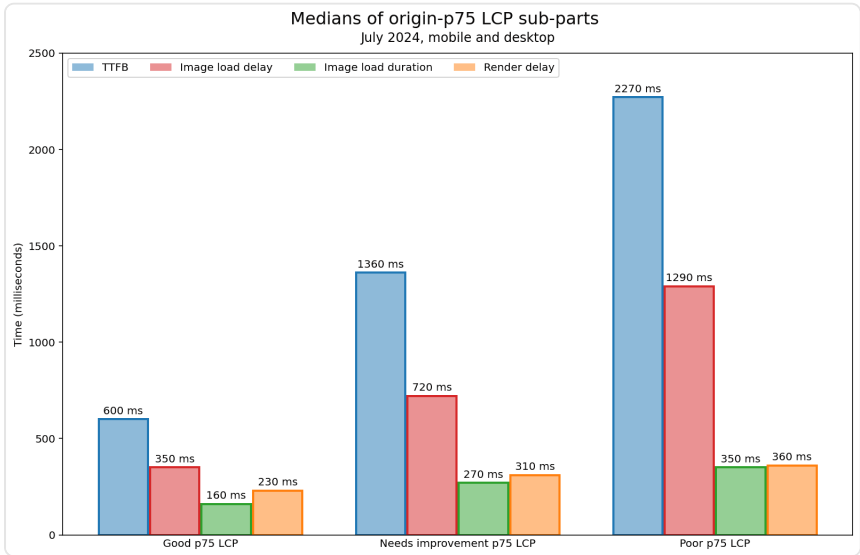


Figure 9.10. Time spent in each LCP subpart, grouped into LCP buckets of good, needs improvement, and poor.

The study showed that image load duration has the least impact on LCP time, taking only 350 milliseconds at the 75th percentile for websites with poor LCP. Although resource load duration optimization techniques like image size reduction are often recommended, they don't offer as much time savings as other LCP sub-parts, even for sites with poor LCP.

TTFB is the largest part among all LCP sub-parts due to the network requests for external resources. Websites with poor LCP spend 2.27 seconds on TTFB alone, which is almost as long as the threshold for a good LCP (2.5 seconds). As we saw in the TTFB section, there hasn't been much improvement in the percentage of websites with good TTFB, indicating that this metric offers significant opportunities for LCP optimization.

Surprisingly, websites spend more time on resource load delay than on load duration, regardless of their LCP status. This makes load delay a good candidate for optimization efforts.

327. https://web.dev/blog/common-misconceptions-lcp#lcp_sub-part_breakdown

One way to improve load delay is by ensuring that the LCP element starts loading as early as possible, which will be explored in detail in the section on LCP static discoverability.

This year, we analyzed LCP sub-part data from another real user monitoring source: RUMvision. Although RUMvision has a different population of websites, it's interesting to compare it with the larger CrUX website population. We assume that websites using performance monitoring tools like RUMvision should have more insights into performance optimization opportunities than the average website represented in CrUX. Naturally, the LCP sub-part results from two different datasets show some differences.

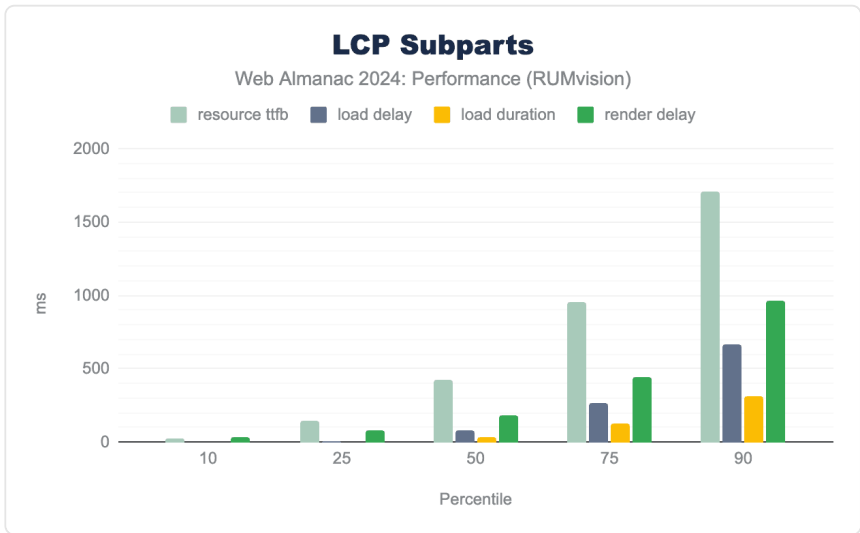


Figure 9.11. Time spent in each LCP subpart by percentile.

According to RUMvision data, TTFB is also the largest contributor to the LCP time in comparison to the other LCP sub-parts. However, the results of other sub-parts vary. Render delay is the second largest contributor to LCP, taking 184 milliseconds. At the 75th percentile, render delay grows to 443 milliseconds. This reflects a tendency that is different from the CrUX dataset, where LCP load delay is the second largest sub-part.

Typically, LCP element rendering takes a long time if the LCP element hasn't been added to the DOM yet—a common issue with client-side generated content that we explore in the next section. Also, the main thread blocked by long tasks can contribute to the delay. In addition, render-blocking resources like stylesheets or synchronous scripts in the `<head>` can delay rendering.

It's interesting to observe the different LCP challenges that websites across various datasets face. While an average website from the CrUX dataset struggles with image load delay,

websites from the RUMvision dataset often face rendering delay issues. Nevertheless, all websites can benefit from using performance monitoring tools with Real User Monitoring (RUM), as these tools provide deeper insights into the performance issues experienced by real users.

LCP static discoverability

One of the most effective ways to optimize the LCP resource load delay is to ensure the resource can be discovered as early as possible. If you make the resource discoverable in the initial HTML document, it enables the LCP resource to begin downloading sooner.



35%

Figure 9.12. The percent of mobile pages on which the LCP element was not statically discoverable.

Unfortunately, 35% of mobile websites do not have an LCP element that is statically discoverable in the document. While this is a slight improvement over the 39% we saw in 2022, it's still a significant blocker of LCP performance.

As we'll explore in the following sections, there are three primary ways that websites prevent their LCP resources from being statically discoverable: lazy loading, CSS background images, and client-side rendering.

LCP lazy-loading

A major obstacle to LCP resource discoverability is lazy-loading of the LCP resource. Overall, lazy-loading images is a helpful performance technique that should be used to postpone loading of non-critical resources until they are near the viewport. However, using lazy-loading on the LCP image will delay the browser from loading it quickly. That is why lazy-loading should not be used on LCP elements.



16%

Figure 9.13. The percent of mobile pages having image-based LCP that use native or custom lazy-loading on it.

The good news is that in 2024, fewer websites are using this performance anti-pattern. In 2022, 18% of mobile websites were lazy-loading their LCP images. By 2024, this decreased to 16%.

In terms of the specific lazy-loading technique used, 9.5% of mobile websites natively lazy-load their LCP images with the `loading=lazy` attribute. This is very similar to the 9.8% of sites we saw in 2022. However, the biggest improvement came from custom approaches. This year we see 6.7% of mobile websites using a custom approach, for example hiding the LCP image source behind the `data-src` attribute, which is down from 8.8% in 2022.

Note that the `src` attribute of an LCP image with `loading=lazy` is technically set and therefore discoverable in the static HTML, so we don't count it towards the static discoverability figure in the previous section. However, natively lazy-loaded images absolutely do contribute to resource load delays, albeit in a slightly different way than an image whose source is set by CSS or JavaScript, as we'll explore next.

CSS background images

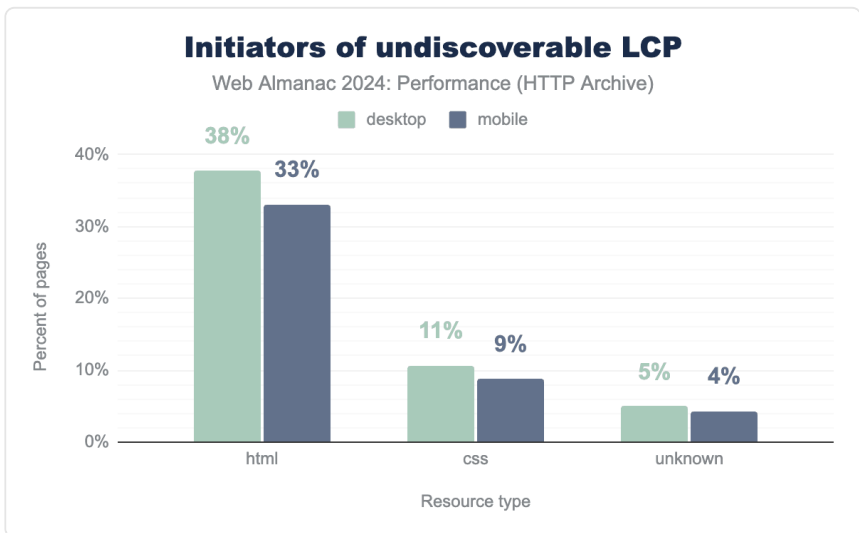


Figure 9.14. The percent of pages whose LCP is not statically discoverable and initiated from a given resource.

Also, websites that initiate LCP elements as CSS background images delay LCP static discovery until the CSS file is processed. The data shows that 9% of mobile pages initialize the LCP resource from CSS. Compared to 2022, this metric has remained unchanged.

Dynamically added images

One more common reason for non-discoverable LCP elements is dynamically added images.

These images are added to the page through JavaScript after the initial HTML is loaded, making them undiscoverable during the HTML document scan.

The chart below illustrates the distribution of client-side generated content. It compares the initial HTML with the final HTML (after JavaScript runs) and measures the difference. It displays how the percentage of websites with good LCP changes as the percentage of client-side generated content increases.

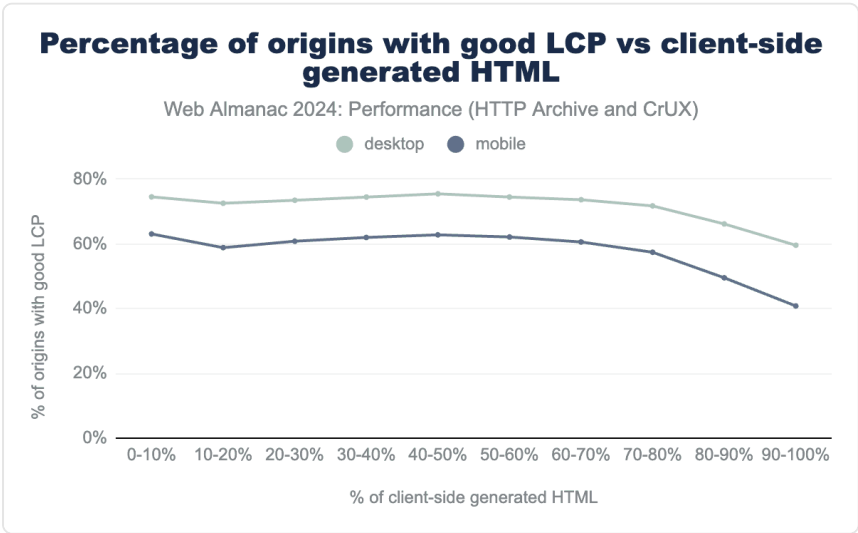


Figure 9.15. The percent of websites with good LCP vs percentage of client-side generated content on a page.

The percentage of pages with good LCP stays at approximately 60% for mobile devices until the amount of client-side generated content reaches 70%. After this threshold, the percentage of websites with good LCP starts to drop at a faster rate until ending at 40%. This suggests that a combination of server- and client-side generated content doesn't significantly impact how fast the LCP element gets rendered. However, fully rendering a website on the client side has a significantly negative impact on LCP.

LCP prioritization

Another one of the most effective ways to optimize the loading delay of LCP images is to declaratively prioritize them, using the `fetchpriority=high` attribute. Even if the LCP resource is statically discoverable by the browser's preload scanner, it might still not start loading immediately if there are other higher priority resources in line. Images are typically not considered high priority resources, so by providing this hint to the browser, it can adjust the

LCP resource's priority accordingly, loading it sooner and reducing its load delay phase.

15%

Figure 9.16. The percent of mobile pages that use `fetchpriority=high` on their LCP image.

Adoption of LCP image prioritization skyrocketed to 15% of mobile websites in 2024, up from just 0.03% in 2022! This massive leap is thanks in large part to WordPress implementing core support³²⁸ for `fetchpriority` in 2023.

As amazing as it is to see such rapid growth, there is still significant room for more sites to take advantage of this impactful one-line optimization.

LCP size

The CrUX and RUMvision data on LCP sub-parts showed that resource load duration is rarely the main bottleneck for a slow LCP. However, it is still valuable to analyze the key optimization factors, such as the size and format of the LCP resource.

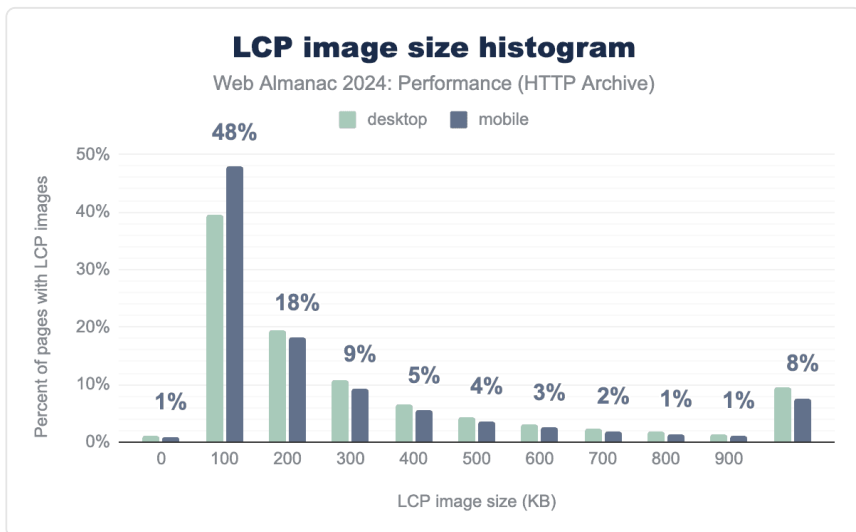


Figure 9.17. Distribution of LCP image sizes, segmented by device type.

328. <https://make.wordpress.org/core/2023/07/13/image-performance-enhancements-in-wordpress-6-3/>

In 2024, 48% of mobile websites used an LCP image that was 100KB or less. Though, for 8% of the mobile pages the LCP element size is more than 1000KB.

This aligns with the Lighthouse audit on unoptimized images³²⁹, which also reports the amount of wasted kilobytes that could be saved by image optimization.

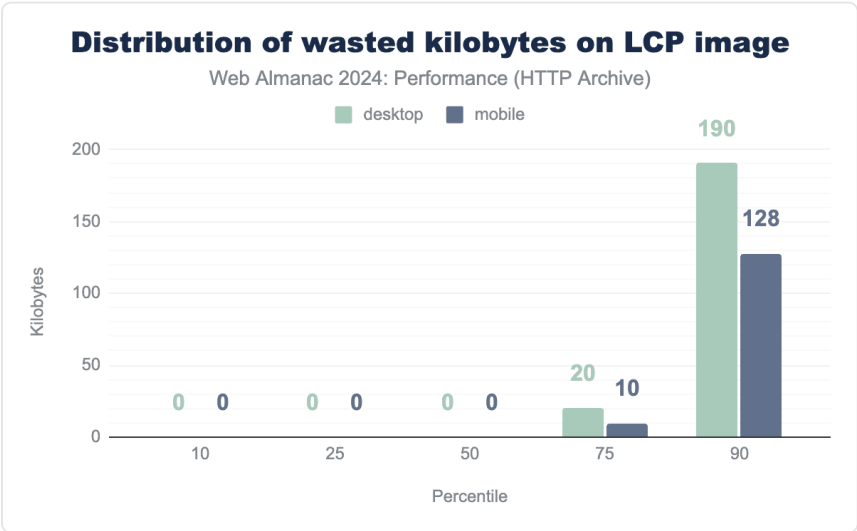


Figure 9.18. Distribution of wasted kilobytes on LCP image.

The audit results indicate that the median website wastes 0 KB on LCP images, i.e. serves optimized images. This leads to the conclusion that many sites are optimizing their LCP resources effectively, although some still need to improve.

You can reduce image sizes through resizing dimensions and increasing compression. Another way to reduce image sizes is by using new image formats like WebP and AVIF, which have better compression algorithms.

329. <https://github.com/GoogleChrome/lighthouse/blob/main/core/audits/byte-efficiency/uses-optimized-images.js>

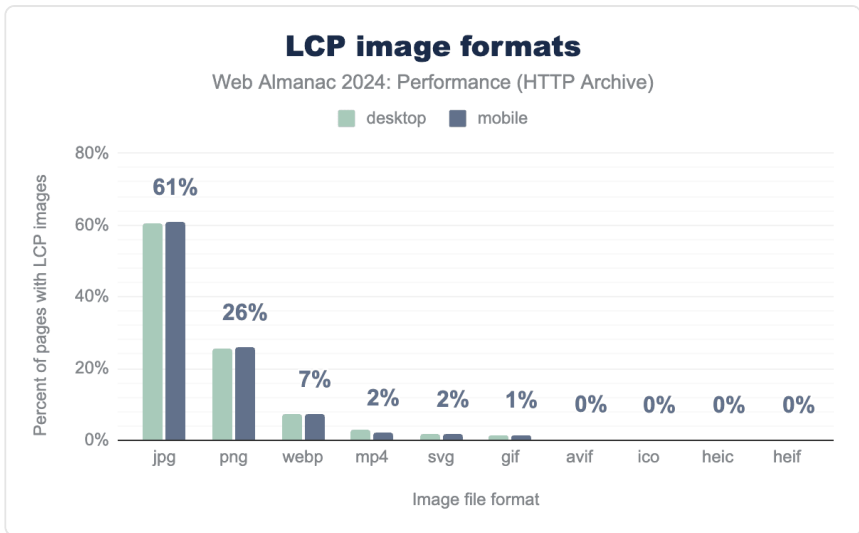


Figure 9.19. The percent of pages that use a given image file format for their LCP images.

JPG and PNG still have the highest proportion of adoption at 87% combined, however WebP and AVIF formats are both increasing in adoption. In comparison to 2022, WebP image format usage increased from 4%³³⁰ to 7%. Also, AVIF usage increased slightly from 0.1% to 0.3%. According to Baseline³³¹, AVIF format is newly available across major browsers, so we expect to see higher adoption in the future.

Loading speed conclusions

- The percentage of websites with good FCP and LCP has improved, though TTFB showed no significant change.
- One cause for slow LCP is lazy-loading the LCP element. Usage of this antipattern has decreased, but 15% of websites still fail this test and could benefit from removing lazy-loading for their LCP elements.
- The adoption of modern image formats like AVIF and WebP is growing for LCP elements.

330. <https://almanac.httparchive.org/en/2022/performance#lcp-format>

331. <https://webstatus.dev/?q=avif>

Interactivity

Interactivity on a website refers to the degree to which users can engage with and respond to content, features, or elements on the page. Measuring interactivity involves assessing the performance for a range of user interactions, such as clicks, taps, and scrolls, as well as more complex actions like form submissions, video plays, or drag-and-drop functions.

Interaction to Next Paint (INP)

Interaction to Next Paint (INP)³³² is calculated by observing all the interactions made with a page during the session and reporting the worse latency (for most sites). An interaction's latency consists of the single longest duration of a group of event handlers that drive the interaction, from the time the user begins the interaction to the moment the browser is next able to paint a frame.

For an origin to receive a “good” INP score, at least 75% of all sessions need an INP score of 200 milliseconds or less. The INP score is the slowest or near-slowest interaction time for all interactions on the page. See Details on how INP is calculated³³³ for more information.

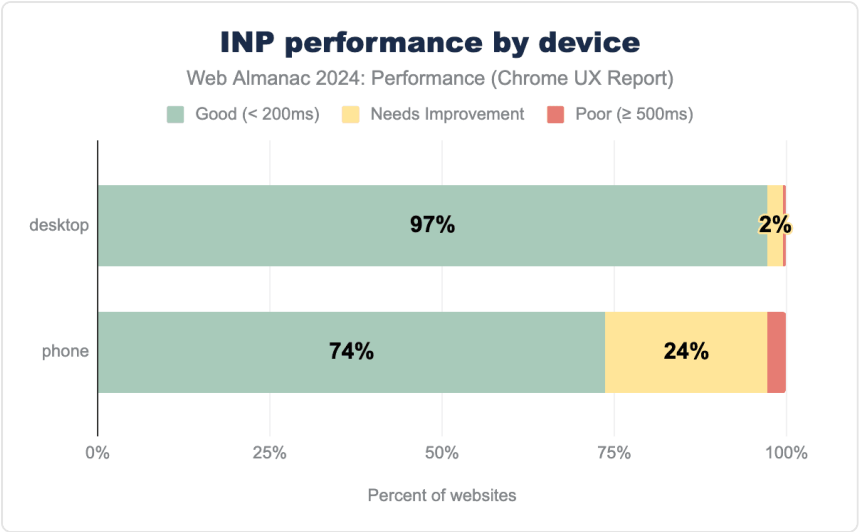


Figure 9.20. Distribution of INP performance by device.

In 2024, 74% of mobile and 97% of desktop websites had good INP. Interestingly, the gap

332. <https://web.dev/articles/inp>

333. <https://web.dev/articles/inp#good-score>

between mobile and desktop is huge, i.e. more than 20%.

The primary reason for weaker performance on mobile is its lower processing power and frequently poor network connections. Alex Russell's article "The Performance Inequality Gap"³³⁴ (2023) raises the issue of the growing performance inequality gap caused by the affordance of high-end vs low-end devices. As the prices of high-end devices rise, fewer users can afford them, widening the inequality gap.

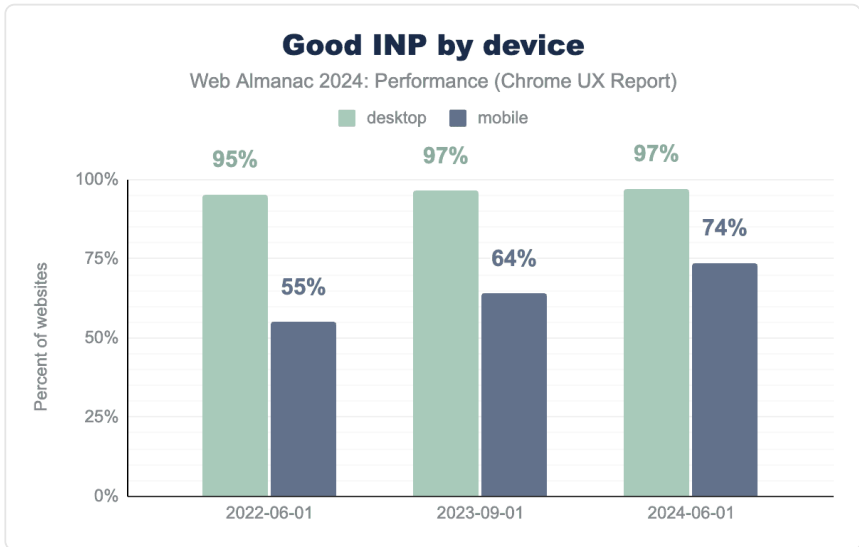


Figure 9.21. Good INP score by device.

Although the INP metric displays worse results than the FID, there has been a positive tendency over the past three years. The percentage of mobile pages having good INP increased from 55% in 2022 to 74% in 2024. This is a significant increase, and even though we can't be exactly sure what to attribute it to, we can think of a few potential drivers for this change.

The first one could be awareness. With the introduction of the INP and the announcement that it will replace FID, many teams realized the impact that could have on their overall CWV score and search ranking. That could have encouraged them to actively work towards fixing parts of the sites that contributed to low INP scores. The second driver could be just a regular advancement in technology. With the above-displayed INP data coming from real users, we can also assume that users' devices and network connections could have slightly improved over the years, providing them with better site interactivity. The third (and perhaps biggest?) driver is improvements to browsers themselves (and in particular to Chrome, given that powers out

334. <https://infrequently.org/2022/12/performance-baseline-2023/>

insights). The Chrome team have made a number of improvements that impact INP³³⁵ over the last two years.

Mobile INP metric by rank reveals an interesting trend. In the 2022 chapter³³⁶, we assumed that the more popular a website is, the more performance optimizations it would have, leading to better performance. However, when it comes to INP, the opposite seems to be true.

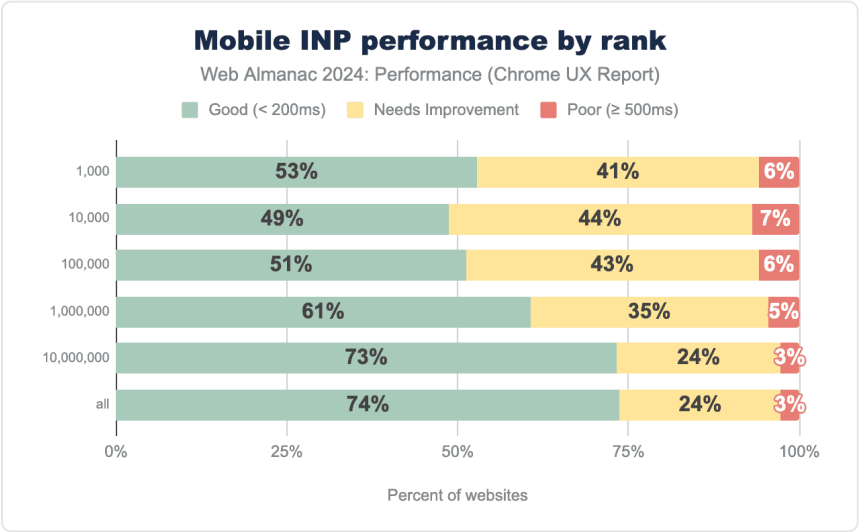


Figure 9.22. INP performance on mobile devices segmented by rank.

Fewer websites in the top 1,000 rank have good INP compared to the results for all websites. For example, 53% of the top 1,000 websites have a good INP score, while a much bigger percentage of all websites, i.e. 74%, meet this threshold.

This could be because the most visited websites often have more user interactions and complex functionality. Logically, the INP for an interactive e-commerce site would differ from a simple, static blog.

335. https://chromium.googlesource.com/chromium/src/+refs/heads/main/docs/speed/metrics_changelog/inp.md

336. <https://almanac.httparchive.org/en/2022/performance#inp-by-rank>

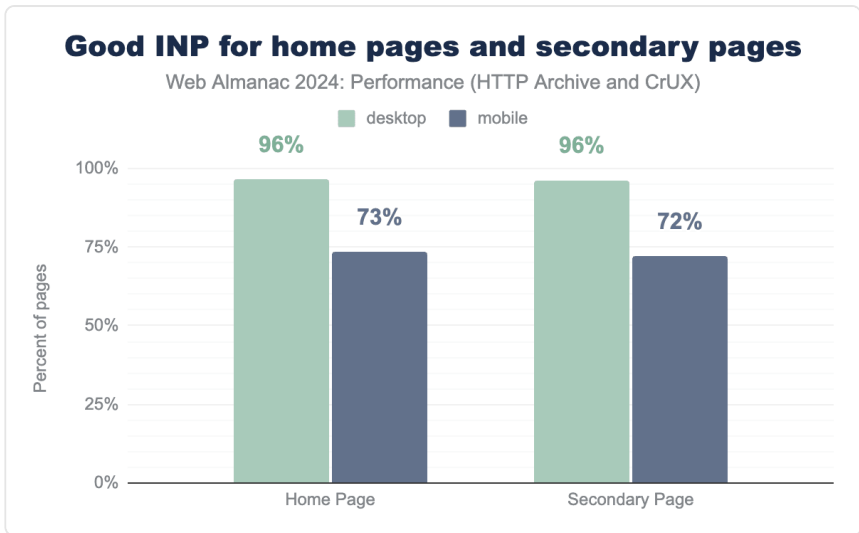


Figure 9.23. Good INP performance on Home and Secondary page by device.

Unlike other performance metrics like FCP and LCP, the percentage of secondary pages with good INP does not differ from the home page results. This is likely because INP isn't as impacted by caching as loading speed is.

INP sub-parts

Interaction to Next Paint metric can be broken down into three key sub-parts:

- **Input Delay:** the time spent to finish processing the tasks that were already in the queue at the moment of the interaction
- **Processing Time:** the time spent processing the event handlers attached to the element which the user interacted with
- **Presentation Delay:** the time spent figuring out the new layout, if changed, and painting the new pixels on the screen

To optimize your website's interactivity, it's important to identify the duration of every sub-part.

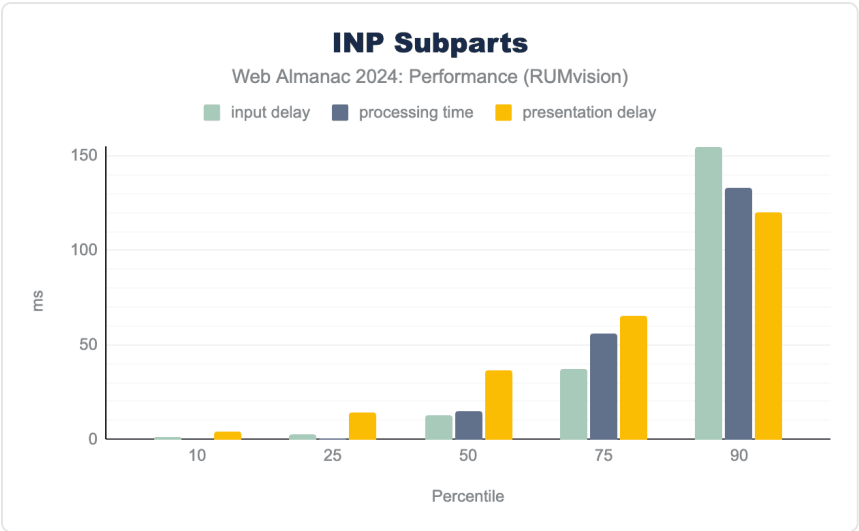


Figure 9.24. INP sub-parts by percentile.

The INP sub-part duration distribution data from RUMvision shows that presentation delay (36 milliseconds) contributes the most to the median INP. As percentiles increase, input delay and processing time become longer. At the 75th percentile, input delay reaches 37 milliseconds and processing delay 56 milliseconds. By the 90th percentile, input delay jumps to 155 milliseconds, which makes it the biggest contributor to poor INP. One way to optimize input delay is by avoiding long tasks, which we explore in the Long Tasks section.

Long tasks

One of the sub-parts of INP is input delay, which can be longer than it should be due to various factors, including long tasks. A task³³⁷ is a discrete unit of work that the browser executes, and JavaScript is often the largest source of tasks. When a task exceeds 50 milliseconds, it is considered a long task. These long tasks can cause delays in responding to user interactions, directly affecting interactivity performance.

Due to the lack of same-source data for long tasks and INP, we decided not to correlate them. We will, however, explore the average Long Task duration using data from RUMvision.

337. <https://web.dev/articles/optimize-long-tasks#what-is-task>

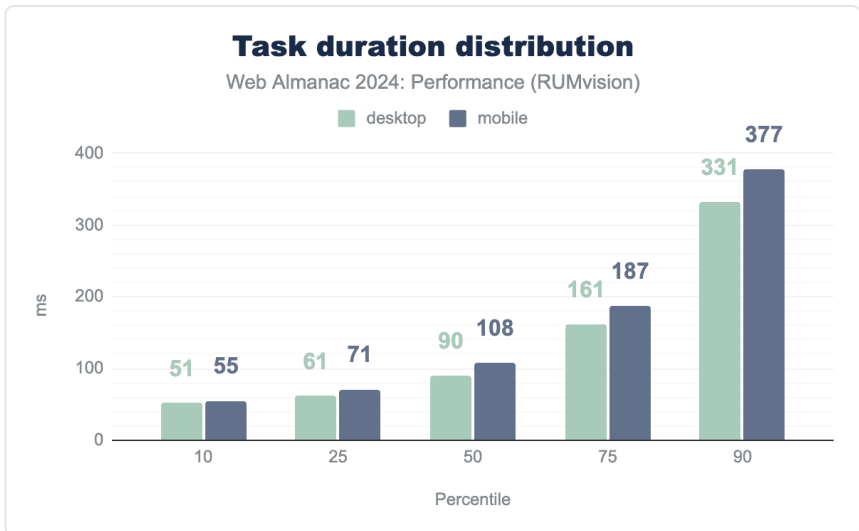


Figure 9.25. Task duration, segmented by device.

The task duration distribution shows a median task duration of 90 milliseconds for desktop and 108 milliseconds for mobile, which is twice more than the best practice recommendation of under 50 milliseconds. Less than 25% of websites have an optimal task duration below 50 milliseconds. We can also see that in every percentile, task duration on mobile sites is longer than on desktop sites, with the gap increasing as the percentile increases. On the 90th percentile, there is a 46 millisecond difference between the average task duration between device types. This correlates well with INP scores that show better results on desktop compared to mobile.

Task duration data was retrieved using the Long Tasks API³³⁸, which provides some useful data about performance issues, but it has limitations when it comes to accurately measuring sluggishness. It only identifies when a long task occurs and how long it lasts. It might overlook essential tasks such as rendering. Due to these limitations, we will explore the Long Animation Frames API in the next section, which offers more detailed insights.

Long animation frames

Long Animation Frames (LoAF)³³⁹ are a performance timeline entry for identifying sluggishness and poor INP by tracking when work and rendering block the main thread. LoAF tracks animation frames instead of individual tasks like the Long Tasks API. A long animation frame is

338. <https://www.w3.org/TR/longtasks-1/>

339. <https://developer.chrome.com/docs/web-platform/long-animation-frames>

when a rendering update is delayed beyond 50 milliseconds (the same as the threshold for the Long Tasks API). It helps to find scripts that cause INP performance bottlenecks. This data allows us to analyze INP performance based on the categories of scripts responsible for LoAF.

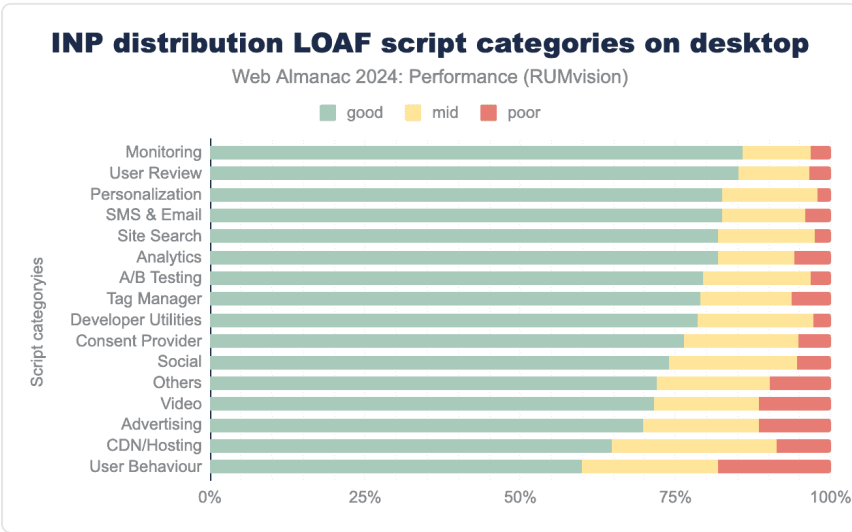


Figure 9.26. Distribution of INP performance segmented by script categories on desktop.

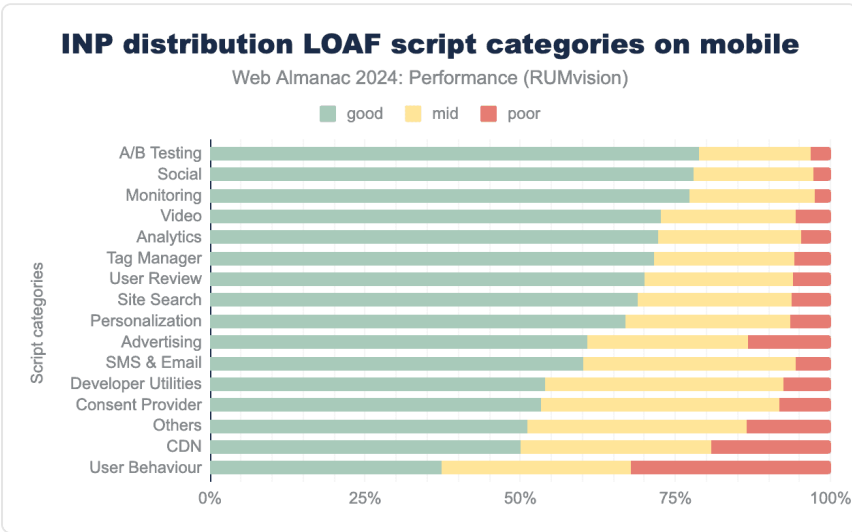


Figure 9.27. Distribution of INP performance segmented by script categories on mobile.

The top two categories contributing the most to slow INP scores on mobile and desktop devices

are User Behavior scripts (37% of mobile and 60% of desktop pages with good INP) and CDN/Hosting (50% of mobile and 65% of desktop pages with good INP).

User Behavior scripts include scripts from hosts like `script.hotjar.com`, `smartlook.com`, `newrelic.com`, etc. While these tools provide valuable insights about users, our data shows that they can significantly degrade user experience by slowing down website interactions.

CDN and Hosting script category examples come from domains like `cdn.jsdelivr.net`, `ajax.cloudflare.com`, `cdnjs.cloudflare.com`, `cdn.shopify.com`, `sdk.aws.waf.com`, `cloudfront.net`, `s3.amazonaws.com` and others. Having CDNs among the categories with the poorest INP results seems controversial because CDNs are usually recommended as a performance optimization technique that reduces server load and delivers content faster to users. However, the CDNs included in this category usually deliver first- or third-party JavaScript resources, which contribute to LoAF and negatively impact interactivity.

On mobile devices, Consent Providers seem to have a significant impact on INP, resulting in only 53% of mobile pages having good INP when using one. This category consists of providers like `consentframework.com`, `cookiepro.com`, `cookiebot.com`, `privacy-mgmt.com`, `usercentrics.eu`, and many others. On desktop devices, Consent Provider scripts show much better results, i.e. 76% of pages with good INP. This difference is likely due to the more powerful processors on desktop devices.

It is worth noting that the monitoring category, which also includes performance monitoring tools, has one of the least impacts on poor INP results. This is a good argument in favor of using web performance monitoring tools, as they help with valuable web performance insights without significantly affecting interactivity performance.

Total Blocking Time (TBT)

Total Blocking Time (TBT)³⁴⁰ measures the total amount of time after First Contentful Paint (FCP) where the main thread was blocked for long enough to prevent input responsiveness.

TBT is a lab metric and is often used as a proxy for field-based responsiveness metrics, such as INP, which can only be collected using real user monitoring, such as CrUX and RUMvision. Lab-based TBT and field-based INP³⁴¹ are correlated, meaning TBT results generally reflect INP trends. A TBT below 200 milliseconds is considered good, but most mobile websites exceed this target significantly.

340. <https://web.dev/articles/tbt>

341. <https://colab.research.google.com/drive/12UjmaABgyVjaUbmWvrbzj9BkkTxw6ay2>

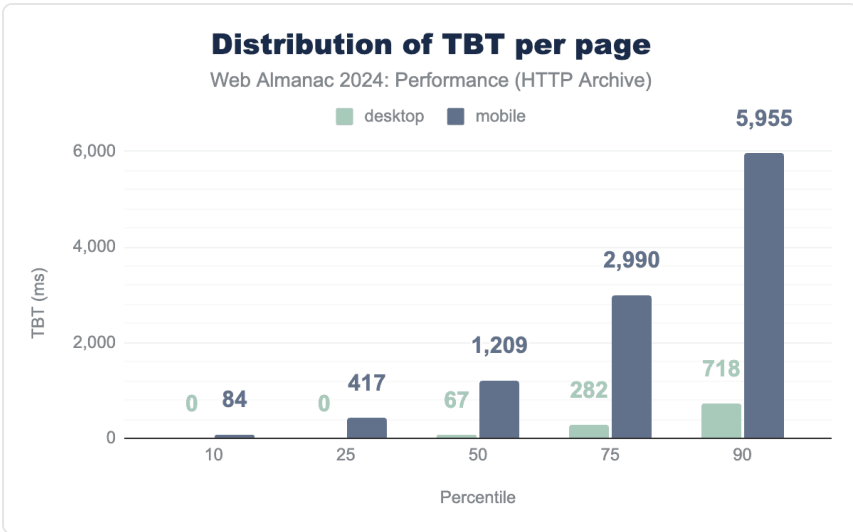


Figure 9.28. TBT per page by percentile.

The median TBT on mobile is 1,209 milliseconds, which is 6 times higher than the best practice. In contrast, desktop websites show much better performance, with a median TBT of just 67 milliseconds. It is important to emphasize that the lab results use an emulated low-power device and a slow network, which may not reflect the real user data, as actual device and network conditions can vary. However, even with that in mind, these results still show that in the 90th percentile, user on mobile device will need to wait almost 6 seconds before the site becomes interactive.

With TBT being caused by long tasks it is not surprising to notice the same trend per percentiles as well as similar trend in gap between mobile and desktop in the two metrics results. It is also important to note that high TBT can be contributing to the input delay part of the INP, negatively impacting the overall INP score.

Interactivity conclusion

The main takeaways of the interactivity results are:

- Despite the improvement in INP each year, a significant gap between desktop (97% good INP) and mobile (74% good INP) performance still exists.
- The top visited websites show poorer INP results compared to less popular ones.
- INP can be divided into three sub-parts: Input Delay, Processing Time, and

Presentation Delay. Presentation Delay has the biggest share of the median INP in RUMvisions's data.

- Scripts from user behavior tracking, consent provider, and CDN categories are the main contributors to poor INP scores.

Visual stability

Visual stability on a website refers to the consistency and predictability of visual elements as the page loads and users interact with it. A visually stable website ensures that content does not unexpectedly shift, move, or change layout, which can disrupt the user experience. These shifts often happen due to assets without specified dimensions (images and videos), third-party ads, heavy fonts, etc. The primary metric for measuring visual stability is Cumulative Layout Shift (CLS)³⁴².

Cumulative Layout Shift (CLS)

CLS measures the biggest burst of layout shift scores for any unexpected layout shifts that happen while a page is open. Layout shifts occur when a visible element changes its position from one place to another.

A CLS score of 0.1 or less is considered good, meaning the page offers a visually stable experience, while scores between 0.1 and 0.25 indicate the need for improvement, and scores above 0.25 are considered poor, indicating that users may experience disruptive, unexpected layout shifts.

³⁴². <https://web.dev/articles/cls>

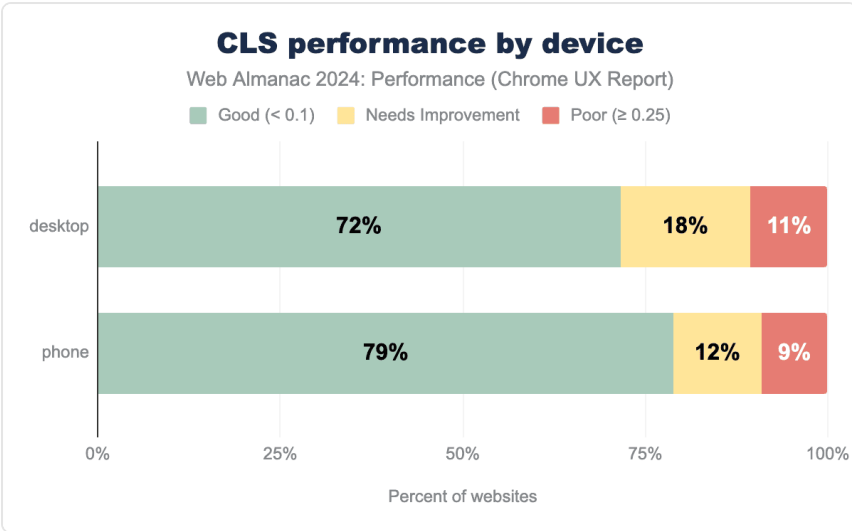


Figure 9.29. CLS performance by device for 2024.

In 2024, 72% of websites achieved good CLS scores, while 11% had poor ones. We can also see that websites on mobile devices provide a better user experience when it comes to site stability than desktop sites.

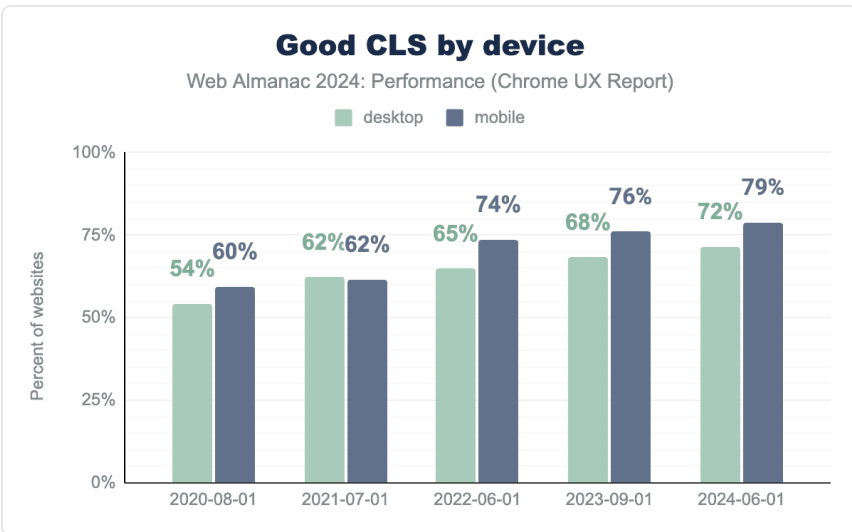


Figure 9.30. The percent of websites having good CLS, segmented by device and year.

Looking at the metrics over time, we can see a nice upward trend. There is an increase from 60% of websites with good visual stability in 2020 to almost 80% in 2024. A visible jump in mobile data is already addressed in detail and attributed to the introduction of bfcache in the 2022 chapter³⁴³. There is still a visible difference from 2022, so we will look in detail at some of the aspects that possibly contributed to it.

Back/forward cache (bfcache)

The back/forward cache (bfcache)³⁴⁴ is a browser optimization feature that improves the speed and efficiency of navigating between web pages by caching a fully interactive snapshot of a page in memory when a user navigates away from it. However, not all sites are eligible for bfcache. With an extensive eligibility criteria³⁴⁵, the easiest way to check if the site is eligible is to test it in Chrome DevTools³⁴⁶.

Let's look deeper by checking a few eligibility criteria that are quite a common cause and easily measurable using lab data.

One of the “usual suspects” is the `unload` event that is triggered when a user navigates away from a page. Due to how bfcache preserves a page's state, `unload` event makes the page ineligible for bfcache. Important to note here is that this feature is specific for browsers on desktops. Mobile browsers ignore the `unload` event when deciding bfcache eligibility, since it is already unreliable on those devices given how background pages are discarded more often there. This behavior could explain CLS improvement over the years and the gap between mobile and desktop numbers:

343. <https://almanac.httparchive.org/en/2022/performance#cumulative-layout-shift-cls>

344. <https://web.dev/articles/bfcache>

345. <https://html.spec.whatwg.org/multipage/nav-history-apis.html#nrr-details-reason>

346. <https://developer.chrome.com/docs/devtools/application/back-forward-cache>

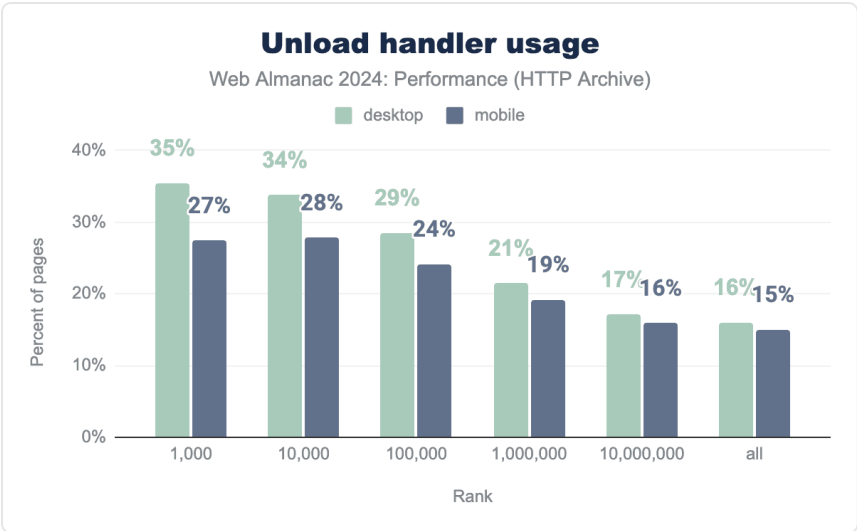


Figure 9.31. Usage of unload by site rank.

From the above chart showing `unload` events from pages, we can see a few interesting things. Overall event usage is quite low, 15-16%. However, it increases drastically for the top 1.000 sites, to 35% on desktop and 27% on mobile, indicating that more popular sites probably use quite some more third-party services that often use this specific event. The gap between mobile and desktop is significant as, while mobile sites using `unload` events are still eligible for the bfcache, they are still unreliable.

It is expected to see this decrease in the use of unload events with major browsers like Google Chrome and Firefox moving towards its deprecation since around 2020 and encouraging the use of alternative events like `pagehide` and `visibilitychange`. These events are more reliable, do not block the browser’s navigation, and are compatible with bfcache, allowing pages to be preserved in memory and restored instantly when users navigate back or forward.

Another common reason for websites to fall in the bfcache ineligibility category is the use of the `cache-control: no-store` directive. This cache control header instructs the browser (and any intermediate caches) not to store a copy of the resource, ensuring that the content is fetched from the server on every request.

21%

Figure 9.32. Percentage of sites using `Cache-Control: no-store`.

21% of sites are using `Cache-Control: no-store`. That is a slight decrease from the 2022 report when this measure was about 22%.

When bfcache was first introduced, it brought noticeable improvements to Core Web Vitals. Based on that, Chrome is gradually bringing bfcache to more sites³⁴⁷ that were previously ineligible due to the use of the `Cache-Control: no-store` header. This change aims to further improve site performance.

Unload event, as well as `Cache-Control: no-store`, do not directly affect the page's visual stability. As already mentioned, the concept of bfcache load as a side-effect has this positive impact by eliminating some potential issues affecting metrics directly, such as unsized images or dynamic content. To continue exploring the visual stability aspect of the web, let's check some of the practices that directly impact the CLS.

CLS best practices

The following best practices allow you to reduce, or even completely avoid CLS.

Explicit dimensions

One of the most common reasons for unexpected layout shifts is not preserving space for assets or incoming dynamic content. For example, adding `width` and `height` attributes on images is one of the easiest ways to preserve space and avoid shifts.



Figure 9.33. The percent of mobile pages that fail to set explicit dimensions on at least one image.

66% of mobile pages have at least one unsized image, which is an improvement from 72% in 2022.

³⁴⁷. <https://developer.chrome.com/docs/web-platform/bfcache-ccns>

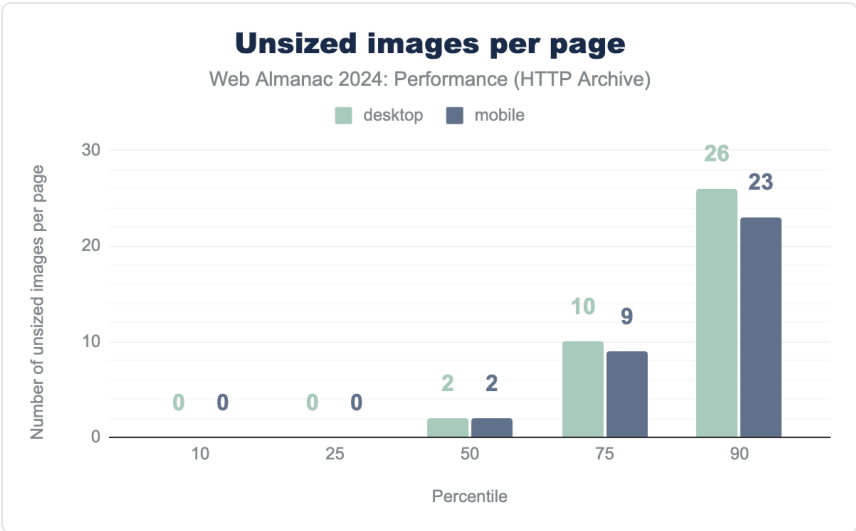


Figure 9.34. The number of unsize images per page.

The median number of unsize images per web page is two. When we shift to the 90th percentile, that number jumps to 26 for desktop sites and 23 for mobile. Having unsize images on the page can be a risk for layout shift; however, an important aspect to look at is if images are affecting the viewport and if yes, how much.

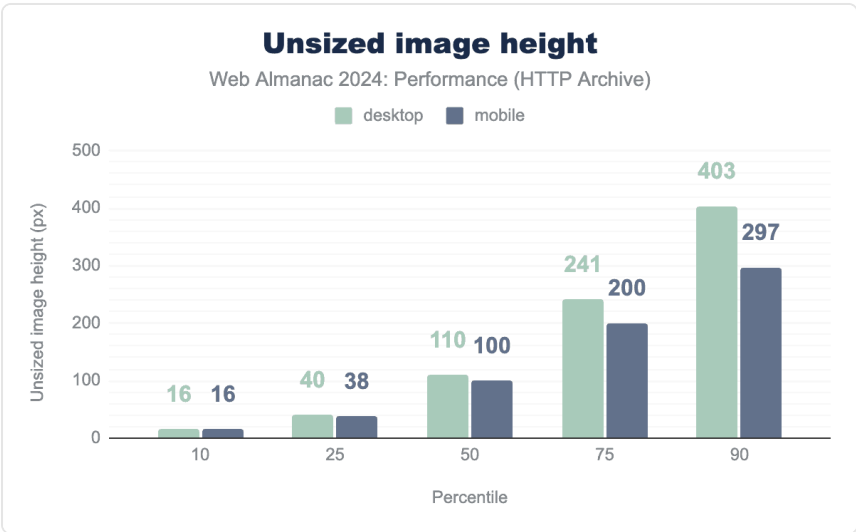


Figure 9.35. Distribution of the heights of unsize images.

The median mobile site has unsized images of about 100 pixels in height. Our test devices have a mobile viewport height of 512 pixels, representing almost 20% of the screen width. This can potentially be shifted down when an unsized (full-width) image loads, which is not an insignificant shift.

As expected, image heights on desktop pages are larger, with the size on the median being 110px and on the 90th percentile 403 pixels.

Fonts

Fonts can directly impact CLS. When web fonts are loaded asynchronously, a delay occurs between the initial rendering of the page and the time when the custom fonts are applied. During this delay, browsers often display text using a fallback font, which can have different dimensions (width, height, letter spacing) compared to the web font. When the web font finally loads, the text may shift to accommodate the new dimensions, causing a visible layout shift and contributing to a higher CLS score.



Figure 9.36. The percent of mobile pages that use web fonts.

Using system fonts is one way to fix this issue. However, with 85% of mobile pages using web fonts it is not very likely that they will stop being used any time soon. A way to control the visual stability of a site that uses web fonts is to use the `font-display` property in CSS to control how fonts are loaded and displayed. Different `font-display` strategies can be used depending on the team's decision about the tradeoff between performance and aesthetics.

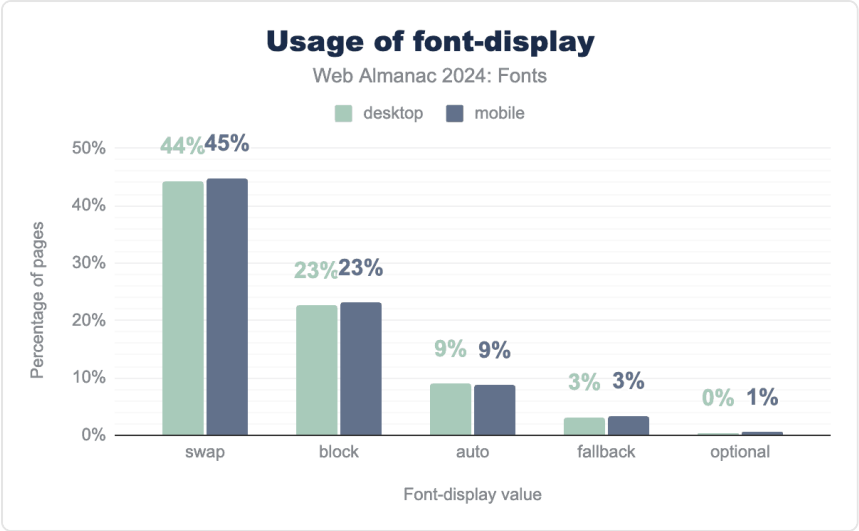


Figure 9.37. Usage of font-display.

From the data displayed above, we can see that around 44% of both mobile and desktop sites use `font-display: swap` while 23% of sites use `font-display: block`. 9% of sites set the `font-display` property to `auto` and 3% use the `fallback` property. Only around 1% of sites use the `optional` strategy.

Compared to the 2022 data, there is a visible increase in the use of all `font-display` strategies, the biggest one being on `swap`, whose usage on both mobile and desktop pages jumped from around 30% in 2022 to over 44%.

Since most `font-display` strategies can contribute to CLS, we need to look at other strategies for minimizing potential issues. One of those is using resource hints to ensure third-party fonts are discovered and loaded as soon as possible.

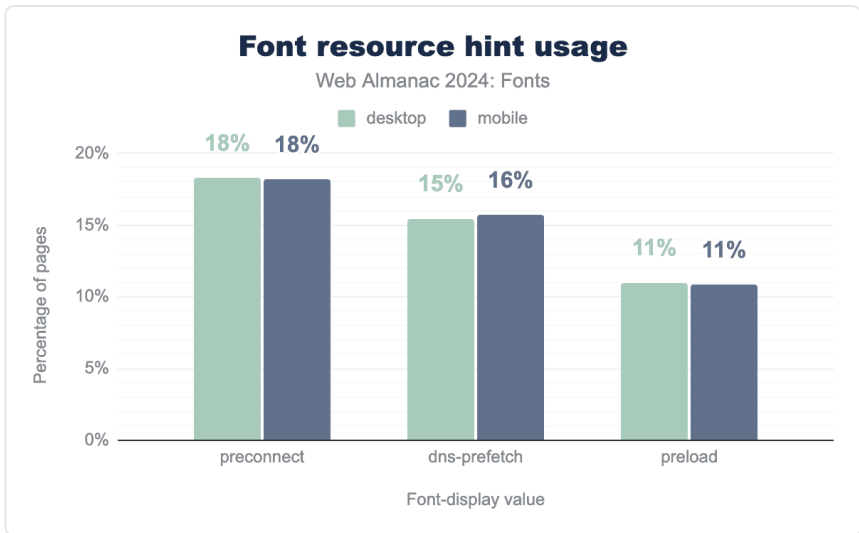


Figure 9.38. Adoption of resource hints for font resources.

Around 11% of all tested mobile and desktop pages are preloading their web fonts, indicating to the browser that they should download these files, hopefully early enough to avoid shifts due to late font arrival. Note that using preload incorrectly can harm performance instead of helping it. To avoid this, we need to make sure that the preloaded font will be used and that we don't preload too many assets. Preloading too many assets can end up delaying other, more important resources.

18% of sites are using `preconnect` to establish an early connection to a third-party origin. Like with `preload` it is important to use this resource hint carefully and not to overdo it.

Animations

Another cause of unexpected shifts can be non-composited³⁴⁸ CSS animations. These animations involve changes to properties that impact the layout or appearance of multiple elements, which forces the browser to go through more performance-intensive steps like recalculating styles, reflowing the document, and repainting pixels on the screen. The best practice is to use CSS properties such as `transform` and `opacity` instead.

348. <https://developer.chrome.com/docs/lighthouse/performance/non-composited-animations>

39%

Figure 9.39. The percent of mobile pages that have non-composited animations.

39% of mobile pages and 42% of desktop pages still use non-composited animations, which is a very slight increase from 38% for mobile and 41% for desktop in the analysis from 2022.

Visual stability conclusion

Visual stability of the site can have a big influence on the user experience of the page. Having text shifting around while reading or a button we were just about to click disappear from the viewport can lead to user frustration. The good news is that Cumulative Layout Shift (CLS) continued to improve in 2024. That indicates that more and more website owners are adopting good practices such as sizing images and preserving space for dynamic content, as well as optimizing for bfcache eligibility to benefit from this browser feature.

Conclusion

Web performance continued to improve in 2024, with positive trends across many key metrics. We now have a more comprehensive metric to assess website interactivity—INP—which hopefully should lead to even greater performance optimizations.

However, challenges remain. For example, there is still a significant gap in INP performance between desktop and mobile. Presentation Delay is the main contributor to poor INP, mostly caused by third-party scripts for behavior tracking, consent providers, and CDNs.

Visual stability continues to improve by the adoption of best practices like proper image sizing and preserving space for dynamic content. Additionally, with recent changes in Chrome's bfcache eligibility, more sites will benefit from faster back and forward navigation.

Overall, web performance is on a promising track, making loading times faster, interactivity smoother, and visual stability more reliable. However, the difference between mobile and desktop experiences remains large. In future Web Almanac reports, we hope to see this gap decreasing, making the web experience consistent across all devices.

Authors



Jevgenija Zigisova

✕ @jevgeniazi 🌐 imeugenia in imeugenia

Jevgenija is a frontend engineer and tech event organizer who is passionate about web performance and developer experience. She works at Limehome³⁴⁹—a digital first hotel brand. She ran a Google Developer Group in Latvia and Berlin for several years.



Ines Akrap

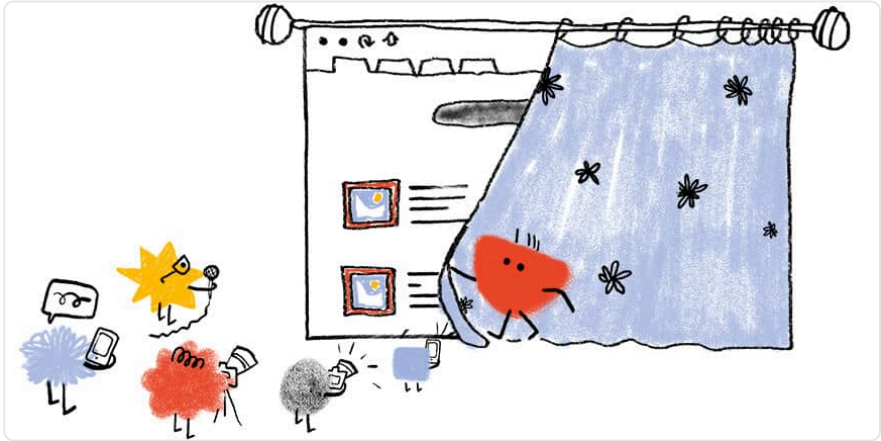
✕ @InesAkrap 🌐 ines-akrap 🌐 <https://inesakrap.com/>

Ines Akrap is a Frontend Software Engineer passionate about optimizing websites to be fast, sustainable, and provide the best user experience for every user. She works in Storyblok as a Solutions Engineer. She enjoys sharing her knowledge through talks, podcasts, workshops, and courses.

349. <https://www.limehome.com/en/>

Part II Chapter 10

Privacy



Written by Yash Vekaria, Benjamin Standaert, Max Ostapenko, Abdul Haddi Amjad, Yana Dimova, Shaoor Munir, Chris Böttger, and Umar Iqbal

Reviewed by Alberto Fernandez-de-Retana

Analyzed by Max Ostapenko, Yash Vekaria, Benjamin Standaert, and Abdul Haddi Amjad

Edited by Barry Pollard

Introduction

Users face significant privacy issues when they browse the web. For example, most websites visited by the user contain *trackers*, which observe user activities and profile them. Profiled user activities are then used for various privacy-invasive purposes, such as targeted personalized online advertising and direct selling of user data.

Trackers deploy a wide range of techniques to track users on the web, such as cookies (both first and third-party), browser fingerprinting, and use of personally identifiable information (such as email addresses).

To protect their privacy, users rely on privacy-enhancing tools—such as ad and tracker blockers—which stop online trackers from loading on web pages. Similarly, browsers are deploying privacy-protections in the browser that aim to eliminate several privacy issues by design, such as blocking of third-party cookies.

Unfortunately, trackers engage in an arms-race with privacy-enhancing technologies and continuously explore mechanisms to bypass privacy protections in the browser—recently with bounce tracking and CNAME tracking. Over the last few years, governments have stepped in with data protection regulations—such as CCPA in California and GDPR in EU—which provide mechanisms for users to exercise their rights, such as not consenting to data collection.

In this chapter, we provide an overview of online tracking (including the mechanism used for online tracking, such as cookies and browser fingerprinting), privacy protections provided by the browsers to eliminate some privacy issues by design (for example, User-Agent Client Hints), techniques used by online trackers to bypass privacy-protections (for example, bounce tracking), and adoption of protections offered to users under data protection regulations (for example, CCPA adoption by websites).

Online tracking

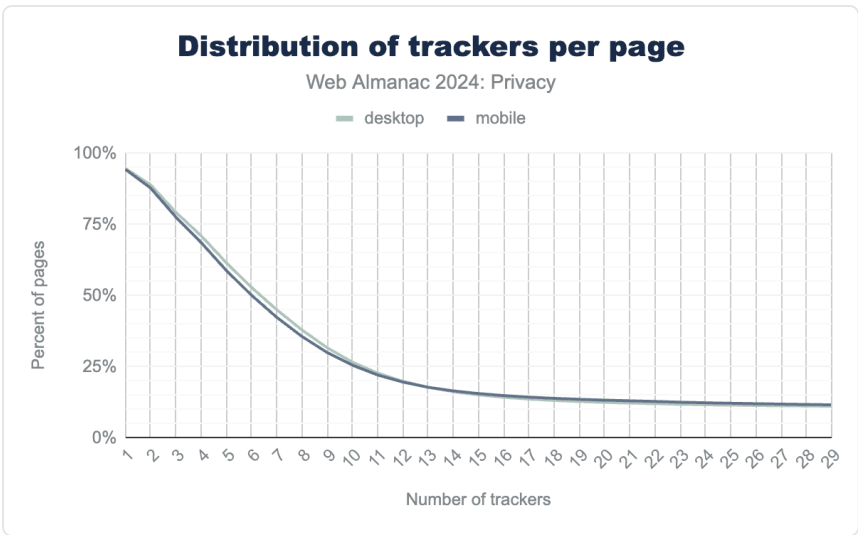


Figure 10.1. Distribution of trackers per page.

We mostly leverage data from WhoTracks.Me³⁵⁰, a publicly available list that catalogs third-party trackers present across a wide range of websites. By utilizing this resource, we identify the most prevalent trackers on the websites. This helped us assess the dominance of certain tracking companies and better understand the overall landscape of third-party tracking. It's important to note that WhoTracks.Me identifies several trackers at the domain level. While a

350. <http://WhoTracks.Me>

significant number of URLs associated with these domains engage in tracking, not all URLs from those domains necessarily do.

Online tracking is a routine practice on the internet. A significant number of websites include specialized online services that record user activities within their website and across sites. Our findings reveal that 95% of the desktop and 94% of the mobile websites include at least one tracker.

We also note that more than a quarter of both desktop (27%) and mobile (26%) sites contain more than 10 trackers. These trackers enable companies to build detailed user profiles based on online behavior, which are regularly used for personalized advertising and to provide insights to website owners. In the following sections, we explore the various techniques trackers use to monitor user activity and examine how they attempt to bypass the privacy protections introduced by modern browsers.

Stateful tracking

Online tracking is broadly classified into two categories: stateful and stateless tracking. Stateful tracking involves storing information about a user directly on their device, typically through cookies and also through other storage mechanisms such as the local storage, that persists across sessions.

When users visit the websites where such trackers are embedded, the cookies associated with these trackers are automatically included in the network requests. Thus tracking services that are embedded on several websites are able to observe all the websites which the user has visited.

Third-party tracking services

The following figure provides the distribution of prevalence of online tracking domains.

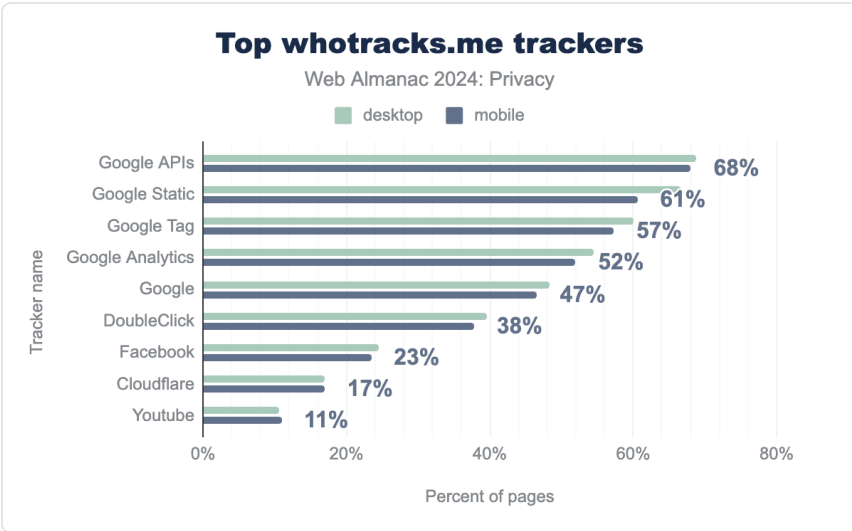


Figure 10.2. Top WhoTracksMe trackers.

We note that Google-owned domains dominate the tracking landscape, with `googleapis.com` and Google's `gstatic.com` appearing on the highest percentage of pages—68% and 61%, respectively. Other prominent trackers include Google Tag and Google Analytics, each seen on over 50% of pages, highlighting the significant reach of Google's tracking services. In addition to Google and its associated services, we also observe a notable presence of Facebook and Cloudflare.

Third-party cookies

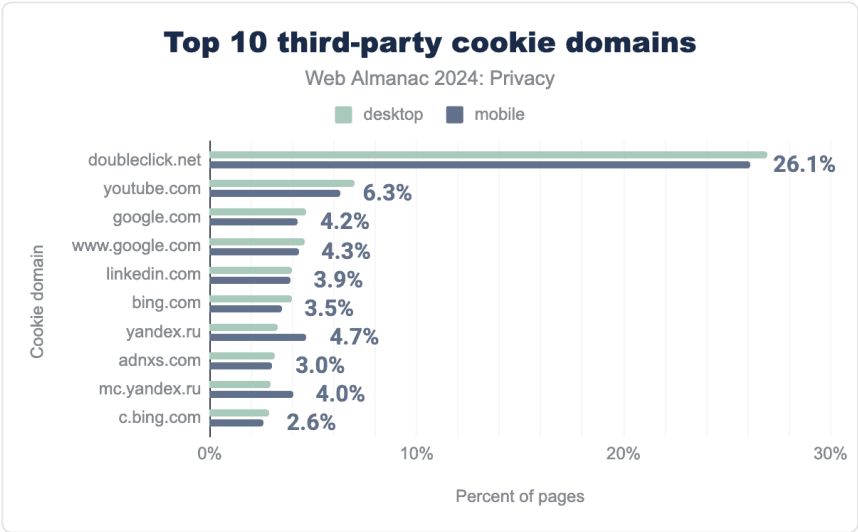


Figure 10.3. Top third-party cookie origins.

Third-party cookies are the main mechanism used to track users on the web. Our measurements reveal that Google's `doubleclick.com` is the largest source of third-party cookies, with presence on more than a quarter of the crawled web pages. Compared to the 2022 analysis³⁵¹, the top sources of third-party cookies have remained largely static, with the notable absence of Facebook, previously the second-largest source of third-party cookies. However, as shown in the next section, we instead see a significant number of cookies set by Facebook in the first-party context.

351. <https://almanac.httparchive.org/en/2022/privacy>

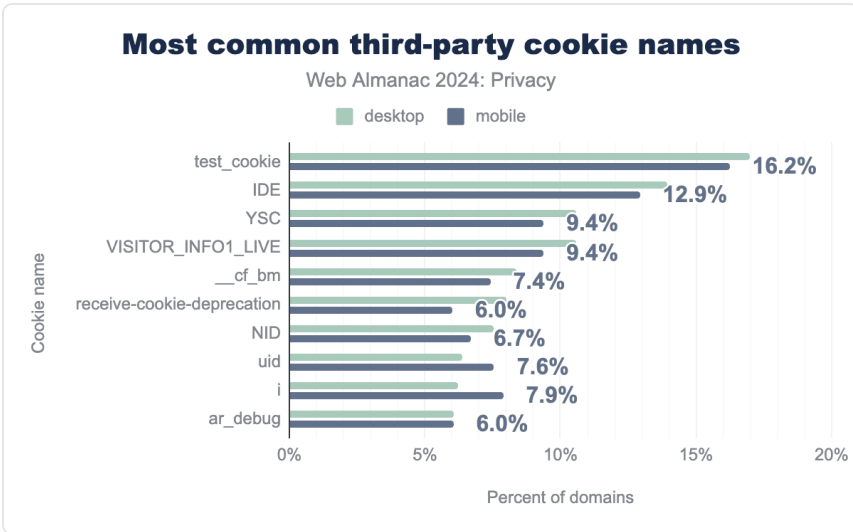


Figure 10.4. Top third-party cookie names.

In order to identify trackers that share cookies across many domains, we also examine the most common names for third-party cookies. We note that the top four cookie names correspond to cookies set by Google’s advertising products and Youtube, as described in their documentation³⁵², and the fifth most-common name corresponds to a cookie set by Cloudflare. Cloudflare’s cookie, `__cf_bm`, is used to “identify and mitigate automated traffic”³⁵³. As this cookie is set on the domains of Cloudflare’s individual customers, it is not captured in a per-domain ranking of cookies.

352. <https://business.safety.google/adscookies/>

353. https://developers.cloudflare.com/fundamentals/reference/policies-compliances/cloudflare-cookies/#__cf_bm-cookie-for-cloudflare-bot-products

First-party cookies

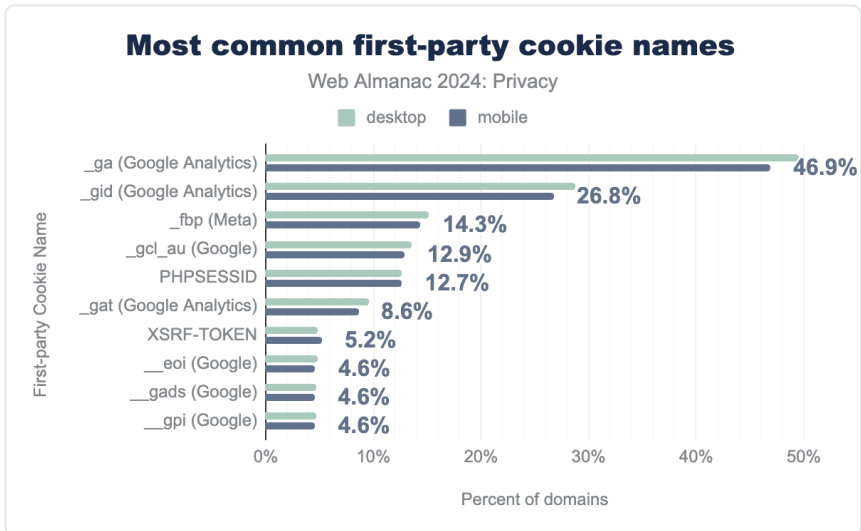


Figure 10.5. Top first-party cookie names.

When measuring common first-party cookies, we see extensive evidence of analytics and advertising services setting cookies in the first-party context. The top two cookies, `_ga` and `_gid`, are both part of Google Analytics.

The next cookie, `_fbp`, is a tracking cookie set by Meta. Since 2022, Meta’s cookie tracking appears to have moved primarily from the third-party context into the first-party context; the default setting for the Meta Pixel now sets first-party cookies³⁵⁴.

The majority of the remaining cookies match cookies set by Google, as described in their documentation³⁵⁵. First-party cookies can be used either to track activity on a single site or for cross-site tracking; we do not attempt to determine the exact purpose of these cookies. Only two of the top cookie names, `PHPSESSID` and `XSRF-TOKEN`, have a clear non-tracking purpose. `PHPSESSID` is the default cookie name used by the PHP framework³⁵⁶ to store the user’s session ID, and `XSRF-TOKEN` is a default name used by the Angular framework³⁵⁷.

The Cookies chapter further describes the details and usage trends of cookies.

354. <https://www.facebook.com/business/help/471978536642445>

355. <https://business.safety.google/adscookies/>

356. <https://www.php.net/manual/en/session.configuration.php#ini.session.name>

357. <https://v17.angular.io/api/common/http/HttpClientXsrfModule>

Stateless tracking

In contrast to stateful tracking, where identifiers are stored in the browser, in stateless tracking, identifiers are generated at runtime. These identifiers often depend on unique characteristics of the user's device or browser. Although this method may be less reliable than stateful tracking, it is typically more difficult to identify and block.

Browser fingerprinting

Browser fingerprinting is one of the most common stateless tracking techniques. To conduct browser fingerprinting, trackers use device configuration information exposed by the browser through JavaScript APIs (for example, Canvas) and HTTP headers (for example, `User-Agent`).

As browsers continue to expand the restrictions placed on cookies, fingerprinting has become an attractive alternative. Prior studies³⁵⁸ have found that fingerprinting is now common and is increasing in prevalence. Here, we attempt to determine the most common sources of fingerprinting across the web.

In our analysis, we first looked for the presence of well-known fingerprinting libraries. We found that, among the libraries tested, the most prevalent library used on the web to perform fingerprinting is FingerprintJS (FingerprintJS³⁵⁹), which we found on 0.57% of all websites. Most likely this is because the library is open source, and has a free version. Compared to our measurements from 2022³⁶⁰, we find that the use of these fingerprinting libraries has slightly decreased; however, it is important to note that this year we crawl roughly ~4 million extra web pages.

358. <https://ieeexplore.ieee.org/abstract/document/9519502>

359. <https://github.com/fingerprintjs/fingerprintjs>

360. <https://olmanac.httparchive.org/en/2022/privacy#evasion-technique-fingerprinting>

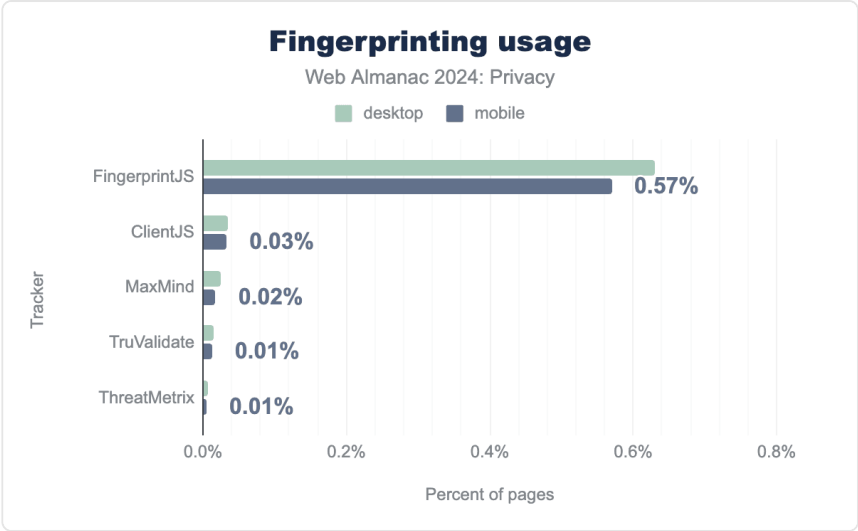


Figure 10.6. Fingerprinting usage.

While we detect the prevalence of well-known fingerprinting vendors, there are several other services (including first-part scripts) that may engage in fingerprinting. To identify such potential sources of fingerprinting, we start by examining the source code of the commonly-used fingerprinting library FingerprintJS. We then compile a list of APIs used by the library, and search for the occurrences of these APIs in all of the crawled scripts. We mark any script with 5 or more usages as a *potential* fingerprinting script. We then rank the scripts by the number of pages on which they are loaded.

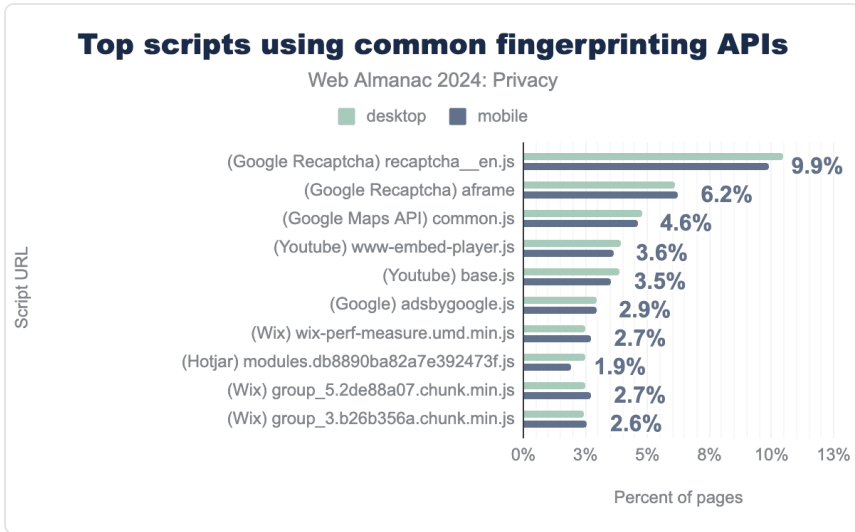


Figure 10.7. Top Scripts with usages of fingerprinting APIs.

We find a mixture of scripts that are primarily used for tracking, and scripts that also have a non-tracking purpose. Recaptcha, the most common script, is known to use fingerprinting³⁶¹ to separate humans from bots. Google Ads and Yandex Metrika are also primarily tracking scripts. However, other scripts, such as the Google Maps API and Youtube embed API, also have a non-tracking purpose, and so their usage of these APIs may have purposes other than fingerprinting. Further manual analysis is required to confirm whether these scripts are actually performing fingerprinting.

Evading tracking protections

As tracking protections such as third-party blocking by cookies are becoming common on web browsers trackers are increasingly exploring mechanisms to bypass them. These methods exploit browser functionality and DNS configurations, enabling persistent tracking even as privacy measures become more stringent.

We examined two prominent tracking protection evasion practices: CNAME tracking and bounce tracking, and looked into how prevalent these are on the web and how browsers are trying to reduce these and maintain user privacy by default.

361. <https://media.kasperskycontenthub.com/wp-content/uploads/sites/63/2017/11/21031220/asia-16-Sivakorn-Im-Not-a-Human-Breaking-the-Google-reCAPTCHA-wp.pdf>

CNAME cloaking

CNAME cloaking leverages the DNS CNAME record to mask third-party trackers as first-party entities. A CNAME record allows a subdomain to point to another domain. Trackers utilize this by setting up a CNAME record on a subdomain of the website they are embedded within. For example, `tracker.example.com` could point to `tracker.trackingcompany.com`. When the tracker sets a cookie, it appears to originate from `example.com`, effectively becoming a first-party cookie and bypassing many third-party cookie blocking mechanisms. This tactic is particularly effective because most tracking protection measures concentrate on restricting third-party access, while first-party cookies are generally allowed for essential website functionality.

Our analysis of DNS data identifies CNAME records used by requests originating from the website's primary domain and pointing to third-party domains. While CNAME records can legitimately be used by hosting services like CDNs, they can also be exploited for tracking. To focus on tracking-specific usage, we cross-referenced identified domains with AdGuard tracker list³⁶² and used data from WhoTracks.Me³⁶³ to filter out primarily hosting-related domains.

In 2022, our analysis of CNAME cloaking relied on mapping first-party hostnames with the AdGuard's disguised CNAME hostnames list³⁶⁴. This year's analysis incorporates a significant enhancement: the collection of actual DNS records for each requested hostname originating from a given page. This direct DNS resolution allows for precise identification of hostnames with CNAME records that redirect to third-party domains, providing a more accurate and comprehensive view of CNAME cloaking activity. This improved methodology has enabled us to identify previously undocumented trackers associated with `utiq.com`, `truedata.co`, `actioniq.com` and others, and contribute these back to the AdGuard list.

362. <https://github.com/AdguardTeam/cname-trackers/blob/master/script/src/cloaked-trackers.json>

363. <http://WhoTracks.Me>

364. <https://github.com/AdguardTeam/cname-trackers/tree/master/data>

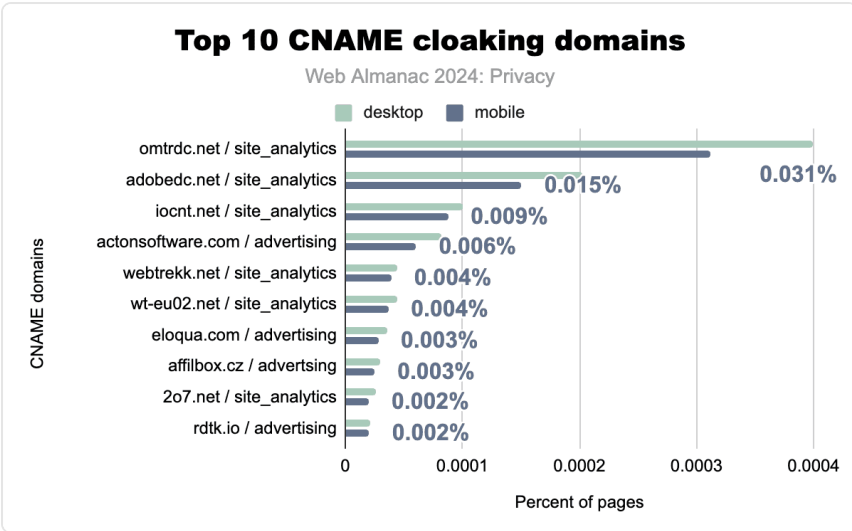


Figure 10.8. Top 10 CNAME cloaking domains.

The 2024 Web Almanac data reveals a continuing trend of CNAME cloaking, with `omtrdc.net` and `adobedc.net`, both associated with Adobe Analytics, leading the site analytics category with appearances on over 0.031% (~9,000) and 0.015% (~4,500) mobile pages respectively.

The prevalence of analytics-related domains suggests that CNAME cloaking is not solely confined to advertising, but is also utilized for broader data collection purposes. The presence of `actionsoftware.com` and other advertising-related domains further solidifies the use of this technique for targeted advertising.

The data highlights that while overall CNAME usage remains relatively low compared to traditional tracking methods, its concentration on high-traffic websites presents a significant privacy concern for a large number of users.

Bounce tracking

Bounce tracking represents another sophisticated evasion technique that allows trackers to read cookies from their first-party context. More specifically, bounce tracking tricks the browser into visiting the tracking domain as a first-party site, allowing it to read and write cookies from its first-party storage. Instead of directly communicating with a tracking server, the browser is first redirected to an intermediary domain—the “bounce” domain (`demo365`). Thus

365. <https://bounce-tracking-demo.glitch.me/>

in case third-party cookies are blocked, trackers can read persistent identifiers from their first-party storage. This intermediary then redirects to the actual website.

This navigation pattern is similar to functional patterns, such as federated authentication (for example, OAuth), which makes it challenging to block bounce tracking. However, web browsers, such as Chrome³⁶⁶ (when opted-in to blocking third-party cookies), Safari³⁶⁷, Brave³⁶⁸ and Firefox³⁶⁹ have deployed or are in the process of deploying mitigation against bounce tracking.

Given the constrained nature of the crawl, limited to the loading of a specific set of pages, our analysis of redirections encompassed only those returning to the originating page after navigating to another page. We identify bounce tracking by detecting instantaneous redirects to a third-party tracker that sets a first-party cookie before returning the user to the original page.

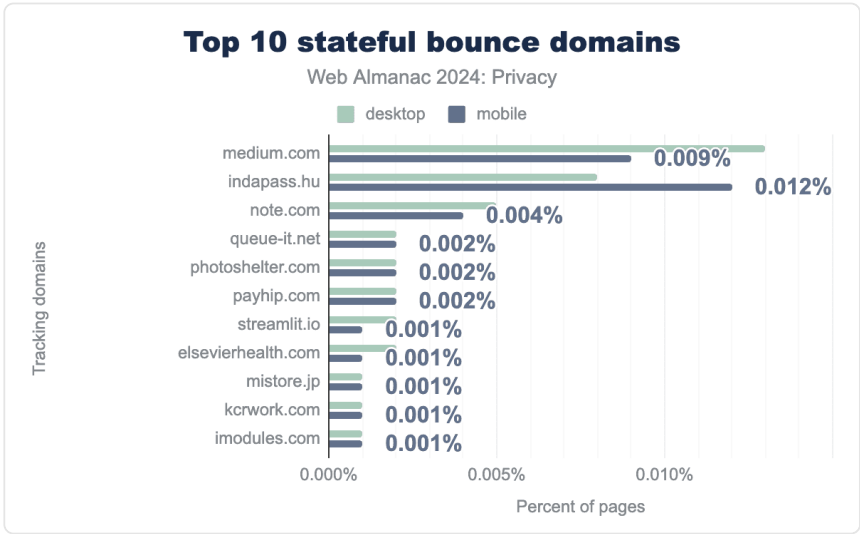


Figure 10.9. Top 10 stateful bounce domains.

We note that `medium.com` (available on 0.009% or 1,515 mobile and 0.013% or 1,641 desktop pages) and `indapass.hu` (IndaMedia³⁷⁰) (0.012% or 1,991 mobile pages) appear the most in bounce tracking like navigations. These companies use bounce tracking to manage a global identity of the visitors to count visits and improve services³⁷¹ (both Medium and IndaMedia are publishing companies)

366. <https://developers.google.com/privacy-sandbox/protections/bounce-tracking-mitigations>
367. <https://webkit.org/blog/11338/cname-cloaking-and-bounce-tracking-defense/#--:text=SameSite%3DS Strict%20Cookie%20Jail%20for%20Bounce%20Trackers>
368. <https://brave.com/privacy-updates/16-unlinkable-bouncing/>
369. <https://firefox-source-docs.mozilla.org/toolkit/components/antitracking/anti-tracking/bounce-tracking-protection/index.html>
370. <https://indamedia.hu/>
371. <https://policy.medium.com/medium-privacy-policy-f03bf92035c9>

Our analysis, limited to crawlable pages, is not exhaustive, and not all identified domains necessarily exhibit privacy-intrusive behavior. Legitimate uses, like SSO (for example `login.taobao.com`) and payment solutions can often be distinguished from tracking by the presence of user interaction on the bounce domain.

Browser policies to improve privacy

It is a common practice for websites to include content from third-party services, such as the advertising and social media platforms. Unfortunately, third-party services cannot be implicitly trusted as, more often than not, they directly harm user privacy. For example by including third-party tracking services. See the Third Parties chapter for a more detailed analysis.

Recently, web standards bodies and browser vendors have tried to step in and provide many controls to website developers that they can use to mitigate privacy threats posed by third-party services. We analyze the prevalence of such prominent browser-provided controls. Note that some of the browser policies, such as Permissions Policy, have both security and privacy implications; we discuss such policies in the Security chapter.

User-Agent Client Hints

In an effort to minimize the amount of information exposed about the browsing environment, particularly through the User-Agent string, the User-Agent Client Hints mechanism is introduced by browsers and standards bodies.

The key idea is that the websites that want to access certain high entropy information about the users' browsing environment have to set a header (Accept-CH) in the first response.

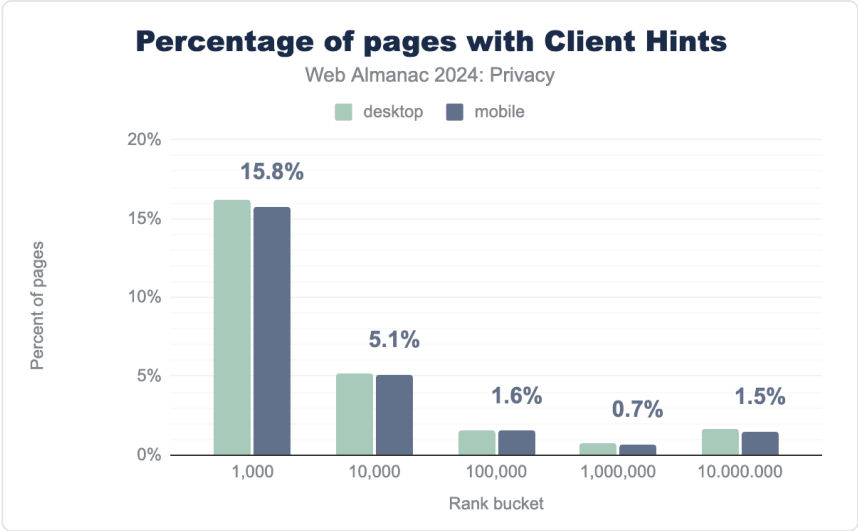


Figure 10.10. Percentage of pages with Client Hints.

We note that it is deployed by 15.8% of the top 1,000 and 5.1% of the top 10,000 mobile websites. When we look at the adoption of sites that respond with the `Accept-CH` header in comparison with the results from 2022 chapter³⁷² (top 1,000: 9.11%, top 10,000: 3.12%), we see an increase in adoption by 6.69% for the 1,000 popular sites. We surmise that this increase in adoption is related to the fact that Chromium has been reducing the information that is shared in the User-Agent string (through the User-Agent Reduction plan³⁷³). For all websites, `Accept-CH` is deployed in 0.4% and 0.5% of all the crawled websites for desktop and mobile, respectively.

Referrer Policy

By default, most user agents include a `Referer` header, which discloses to third parties the website—or even the specific page—from which a request originated. This occurs for any resource embedded within a web page, as well as for requests triggered by a user clicking on a link. Consequently, third parties may gain insight into which website or page a particular user was visiting, leading to potential privacy concerns.

372. <https://almanac.httparchive.org/en/2022/privacy#user-agent-client-hints>

373. <https://www.chromium.org/updates/ua-reduction/>

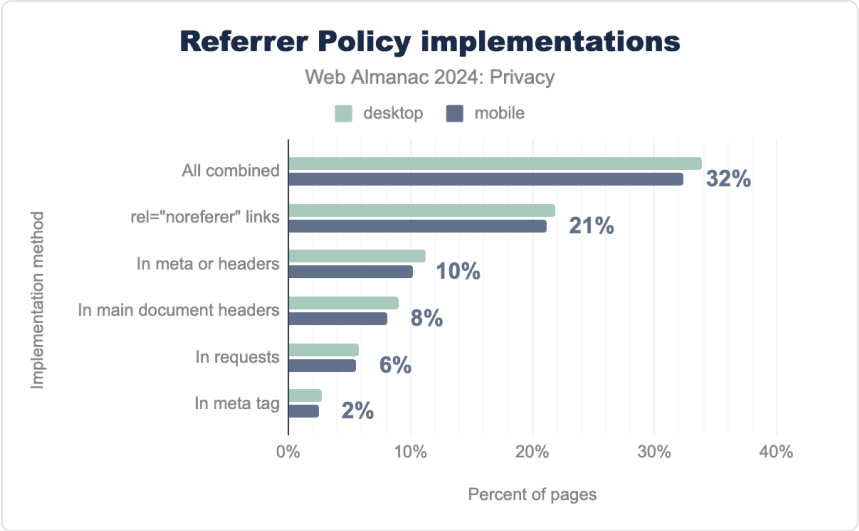


Figure 10.11. Referrer Policy implementations.

By making use of the Referrer Policy³⁷⁴, websites can limit the instances in which the Referrer header is included in requests and thus improve user privacy.

Referrer policy can be included both at the document-level and also at the request-level. We find that referrer policy is deployed on 33.87% of the desktop web pages and 32% of the mobile web pages, overall. On 21.82% of such pages, Referrer Policy is deployed at the request-level with the `ref=noreferrer` HTML tag, and in 11.31% of the instances, the referrer policy is deployed at the document level.

374. <https://developer.mozilla.org/docs/Web/HTTP/Headers/Referrer-Policy>

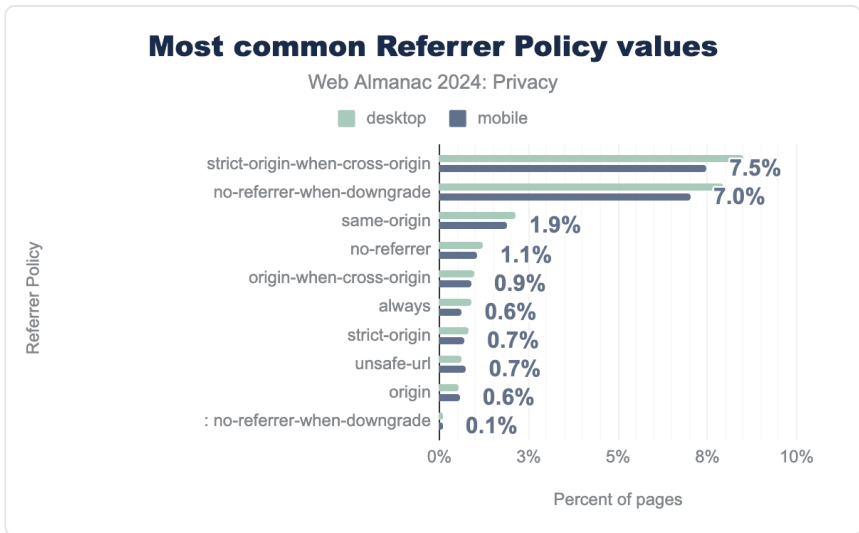


Figure 10.12. Most common Referrer Policy values.

While referrer policy allows to mitigate some tracking, not all of its options have the same effect. Thus we next measure the deployment of individual referrer policy options.

Our analysis reveals that `strict-origin-when-cross-origin` is the most commonly used option of the Referrer Policy, with a deployment on 8% of the crawled web pages. We also note that its deployment has increased by nearly 3X as compared to 2022 chapter³⁷⁵ when it was deployed on only 2.68% of the crawled web pages.

`strict-origin-when-cross-origin` is also the default option if no policy is specified and only shares the full url in the `referrer` header to the same-origin requests. For cross-origin requests, the path and the query string parameters are stripped out.

The next most commonly deployed option is `no-referrer-when-downgrade`, which does not include the Referrer header on downgrade requests, that is, HTTP requests initiated on an HTTPS-enabled page. Unfortunately, this still leaks the page that the user is visiting in most scenarios—in HTTPS-enabled requests.

Privacy-related origins trials

Origin trials allow website developers to test new features released by web browsers (Chrome or FireFox) such as new browser APIs. Once website developers register in origin trials, the new

375. <https://almanac.httparchive.org/en/2022/privacy#user-agent-client-hints>

browser features are made available to all their users. Since web browsers are increasingly deploying privacy-enhancing features, such as eliminating third-party cookies, we next analyze whether website developers are participating in privacy-related origin trials to assess their readiness for the upcoming privacy-enhancing features in browsers.

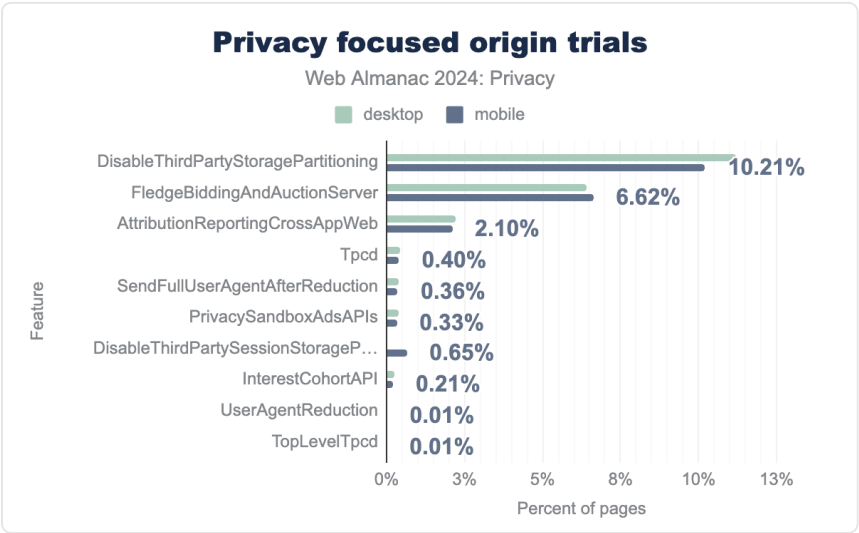


Figure 10.13. Privacy-focused origin trials.

Among the privacy-enhancing features, we note that `disableThirdPartyStoragePartitioning` is the most widely used control with deployment on 10.21% of the mobile websites. `disableThirdPartyStoragePartitioning` allows a top-level site to un-partition (temporarily remove isolation by top-level site) in storage, service workers, and communication APIs in third-party content embedded on its pages.

It means that more than 10% of the websites are testing a feature that disables the benefits provided by the partitioning of third-party storage. Note that the storage partitioning³⁷⁶ applies to select storage related APIs that do not include cookies. The second most prevalent trial is `FledgeBiddingAndAuctionServer` with deployment over 6.62% of the mobile websites.

Privacy Sandbox proposals

Privacy Sandbox, introduced by Google in 2019, contains several proposals that are aimed at

376. <https://developers.google.com/privacy-sandbox/cookies/storage-partitioning>

curbing privacy-invasive practices on the web by aiming to strike a balance between user privacy and the continued viability of online advertising, which supports free content and services on the web. Among privacy sandbox proposals, Topics, Protected Audience, and Attribution Reporting have garnered significant attention because of their implications on targeted advertising, interest-based ad auctions, and privacy-preserving conversion tracking, respectively. In this section, we measure the adoption of these proposals to assess the readiness of websites and ad-tech (for example, advertising platforms, tracking entities), in incorporating these proposals. Note that some of these proposals are not solely limited to Chrome, they are tested by other browsers such as Microsoft Edge.

We first provide the prevalence of these APIs. We note that Topics API, Protected Audience API (previously known as FLEDGE), and Attribution Reporting API have the highest presence across different advertising publishing technologies. These are respectively present on 33%, 63%, and 27% of top 1,000 websites. Amongst top 10 million websites, the presence drops to 7%, 63%, and 24%, respectively. Note that the presence does not imply the adoption of these APIs by websites.

Topics API

Google's Topics proposal works by assigning a small set of high-level topics to a user based on their recent browsing activity, such as "sports" or "technology". These topics are stored locally on the user's devices and shared with websites and advertisers to serve relevant ads. Users also have the ability to see and control the topics that are shared with advertisers.

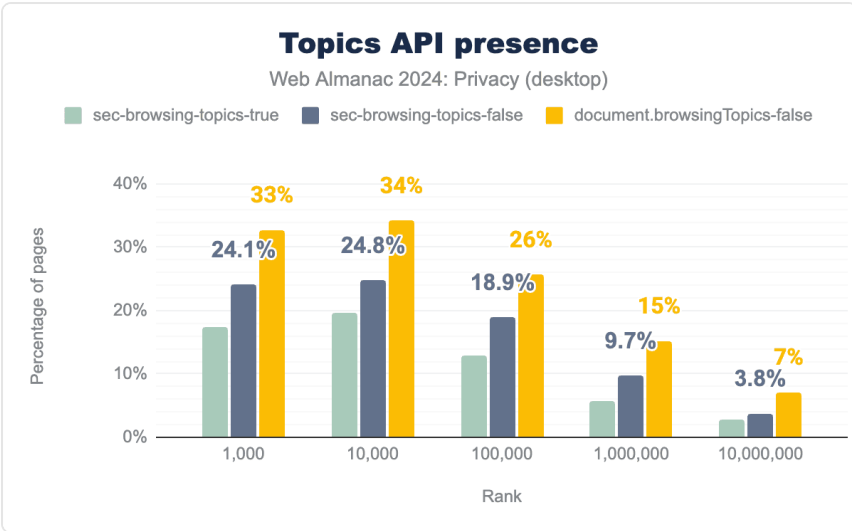


Figure 10.14. Topics API presence.

Since this API can be deployed both through the HTTP headers and JavaScript, we measure the adoption of the Topics API across both of these axes. We observe JavaScript-based presence (`document.browsingTopics`) at 7% of pages, to be more widespread than header-based presence (`sec-browsing-topics`) at ~4% pages.

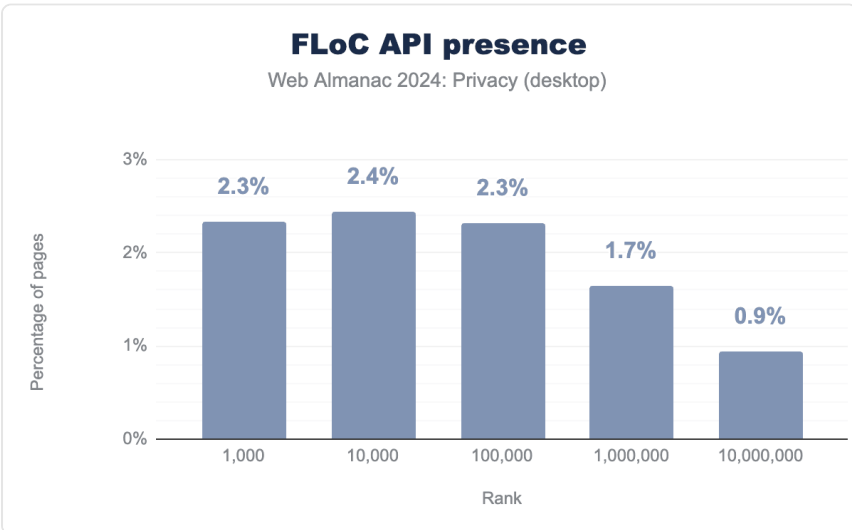


Figure 10.15. FLoC API presence.

Surprisingly, we also note that the Federated Learning of Cohorts API (FLoC)—a precursor proposal to Topics API, despite being deprecated due to several privacy issues, is still present on a considerable amount of pages. While Topics API improves the status quo, prior research³⁷⁷ has shown that monitoring the topics returned by the user’s browser over a period of time can aid in reidentification of users.

Protected Audience API

The Protected Audience API enables on-device auctions by the browser, to choose relevant ads from websites the user has previously visited. It eliminates the need for privacy-invasive data collection and pervasive tracking practices that are otherwise employed for remarketing and targeted advertising. This ensures that advertisers can serve relevant ads without needing to track users across sites.

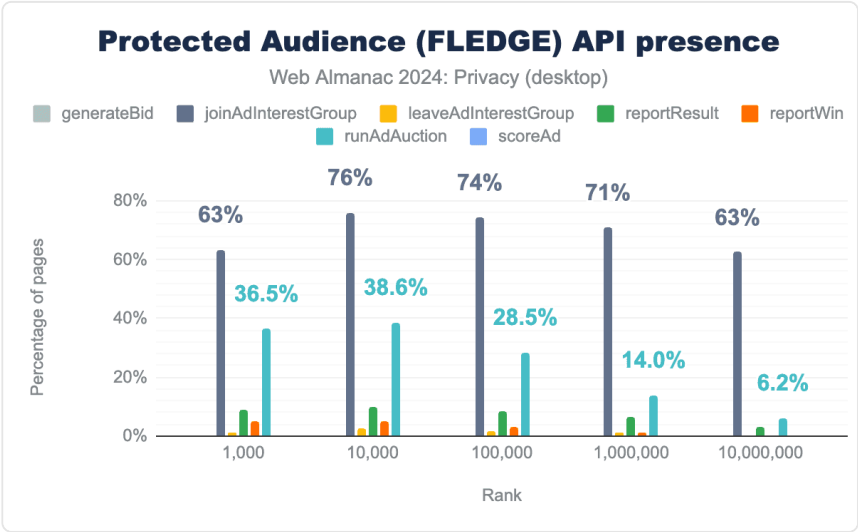


Figure 10.16. Protected Audience (FLEDGE) API presence.

Amongst different method calls available for the Protected Audience API, we note that `navigator.joinAdInterestGroup()` is used the most by third-party services—63% of top 10 million websites.

This API provides an ability to a third-party service to direct the browser to add an interest group to the browser’s membership list for the visiting user. Recent research (Calderonio et

377. <https://petsymposium.org/popets/2024/popets-2024-0004.pdf>

al.³⁷⁸, Long and Evans³⁷⁹) has discovered various privacy flaws with respect to the Protected Audience API. For example, third-party trackers can potentially link the interest groups of the users to an actual user using side-channels and track them across sites. Possibility of colluding entities further alleviate the associated privacy risk.

Attribution Reporting API

Attribution Reporting API³⁸⁰ (ARA) introduces a privacy-preserving mechanism for measuring ad conversions in Google Chrome. Its purpose is to enable attribution measurement by providing a capability to register attribution source and trigger on publisher and advertiser websites, respectively. Chrome records every conversion, and generates a differentially private report that is sent to authorized sources with a delay, preventing cross-site linking of the users. This mechanism works through the use of specific HTTP headers:

1. `attribution-reporting-eligible`: This header signals that a particular request's response is eligible for attribution reporting.
2. `attribution-reporting-register-source`: Used to register attribution sources when displaying an advertiser's ad on publishers.
3. `attribution-reporting-register-trigger`: Used on the advertiser's website to register triggers that measure conversions when users interact with ads.

From our analysis, we observe that twice as many third parties are registering triggers compared to those registering sources. This trend indicates a higher focus on measuring conversions as compared to tracking the initial ad display events.

378. <https://www.usenix.org/system/files/usenixsecurity24-calderonio.pdf>

379. <https://arxiv.org/pdf/2405.08102>

380. <https://developers.google.com/privacy-sandbox/private-advertising/attribution-reporting>

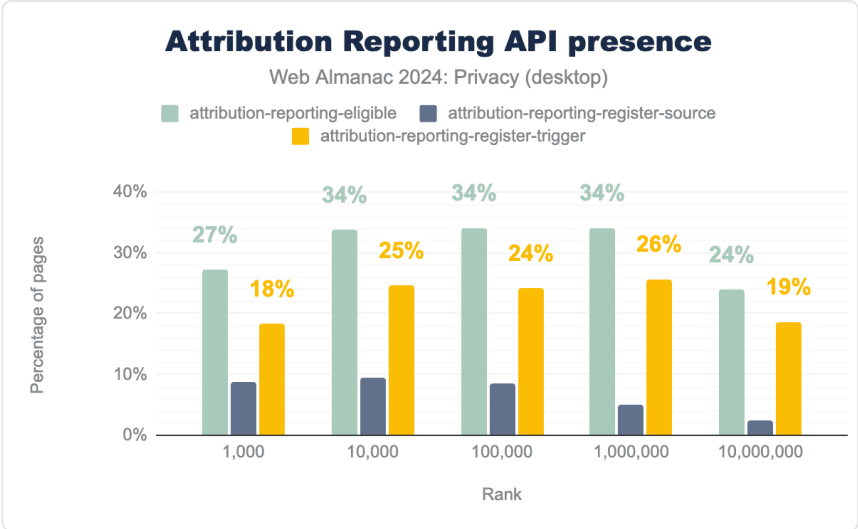


Figure 10.17. Attribution Reporting API presence.

Since most of the popular browsers are competing with each other in the space of privacy preserving attribution with proposals like ARA for Chrome, Private Click Measurement³⁸¹ (PCM) for Safari, and Interoperable Private Attribution³⁸² (IPA) by Mozilla and Meta, we analyze ARA in more detail. We look at registrations of advertising destinations on different websites.

381. <https://webkit.org/blog/11529/introducing-private-click-measurement-pcm/>
382. <https://github.com/patcg-individual-drafts/ipa>

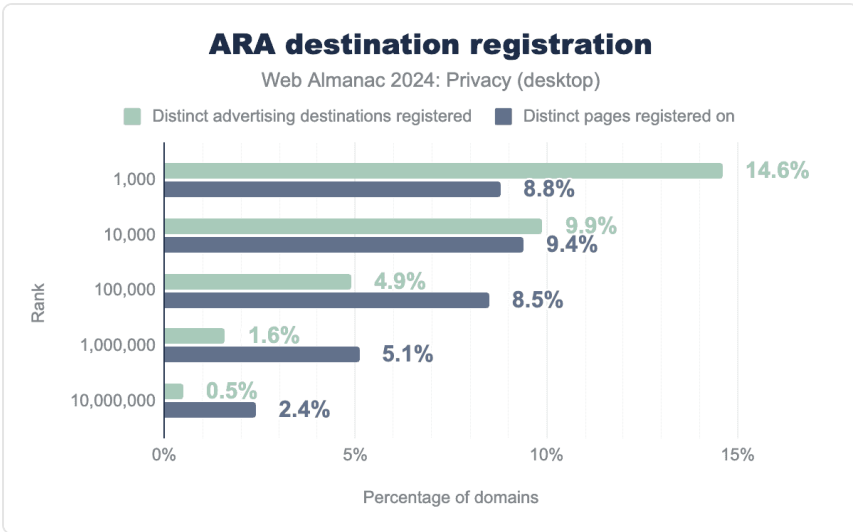


Figure 10.18. ARA destination registration.

We observe that 14.6% distinct advertisers are using ARA to register themselves using attribution-reporting-register-trigger header on 8.8% distinct publishers on the top 1,000 websites. In total, we observe 1.6% (0.5%) distinct advertisers have adopted ARA across 5.1% (2.4%) publishers in top million (top 10 million) websites. This shows that not many publishers have adopted ARA as compared to the advertiser adoption—preparing themselves for the post-cookie world where they will need to rely on ARA to attribute user conversions to ad clicks.

Limitation: Note that by “presence”, we refer to the mere presence of privacy sandbox API calls in the JavaScript in this analysis. This does not mean that the APIs are guaranteed to be executed or used during the runtime.

Related Websites Sets

Related Website Sets³⁸³ allow websites from the same owner to share cookies among themselves. The creation and submission of a Related Website Set is done at the moment through opening a pull request on a GitHub repository³⁸⁴ that the Google project contributors check and merge if deemed valid. Websites that belong to the same related website set must also indicate it by placing a corresponding file at the well-known URI³⁸⁵ `.well-known/related-website-set.json`.

383. <https://developers.google.com/privacy-sandbox/cookies/related-website-sets>

384. <https://github.com/GoogleChrome/related-website-sets>

385. <https://www.iana.org/assignments/well-known-uris/well-known-uris.xhtml>

Chrome ships with a pre-loaded file containing related website sets validated by the Chrome team. At the moment of writing (version 2024.8.10.0), there are 64 distinct related website sets. Each related website set contains a primary domain and a list of other domains related to the primary one under one of the following attributes: `associatedSites`, `servicesSites`, and/or `ccTLDs`. These 64 primary domains are each associated with secondary domains as part of their set: 60 sets contain `associatedSites`, 11 `servicesSites`, and 7 `ccTLDs`—see the Cookies chapter for more results.

Law and policy

With increasing scrutiny against online tracking, there have been numerous new laws and regulations passed to make online advertisers and trackers more accountable. In this section we look at the impact these regulations have had on privacy.

Consent dialogs

With the introduction of privacy regulations like the General Data Protection Regulation³⁸⁶ (GDPR) in the European Union and California Consumer Privacy Act³⁸⁷ (CCPA), websites require user consent to collect, share, and process user data—for example, collection and usage of third-party tracking cookies. This has led to the widespread use of cookie consent dialogs, which notify users about the data collection practices and allow them to accept, reject, or customize their consent.

These consent dialogs have become a ubiquitous feature across the web, but their effectiveness in truly protecting user privacy is debated. Many websites use “dark patterns” to nudge users into accepting tracking, while others present complex options that can overwhelm non-technical users. The Interactive Advertising Bureau (IAB) Europe introduced the Transparency and Consent Framework³⁸⁸ (TCF) to standardize the process of obtaining consent for targeted advertising. The IAB consent dialog is used by many websites and ad tech companies to comply with GDPR and other privacy laws while continuing to serve personalized ads. The framework is designed to provide transparency into how user data is processed and to give users the ability to grant or withhold consent for different purposes, such as personalized ads, analytics, or content delivery.

Our findings show that the TCF, along with other privacy frameworks, is widely implemented as publishers seek to comply with data protection laws like GDPR and CCPA. We would like to note here that our measurement is USA-based, and according to TCF, no consent banner is

386. <https://gdpr-info.eu/>

387. https://leginfo.ca.gov/faces/codes_displayText.xhtml?division=3.&part=4.6&lawCode=CIV&title=1.81.5

388. <https://iab europe.eu/transparency-consent-framework/>

required for non-EU visits. Therefore, this can result in smaller than actual measurements of TCF usage.

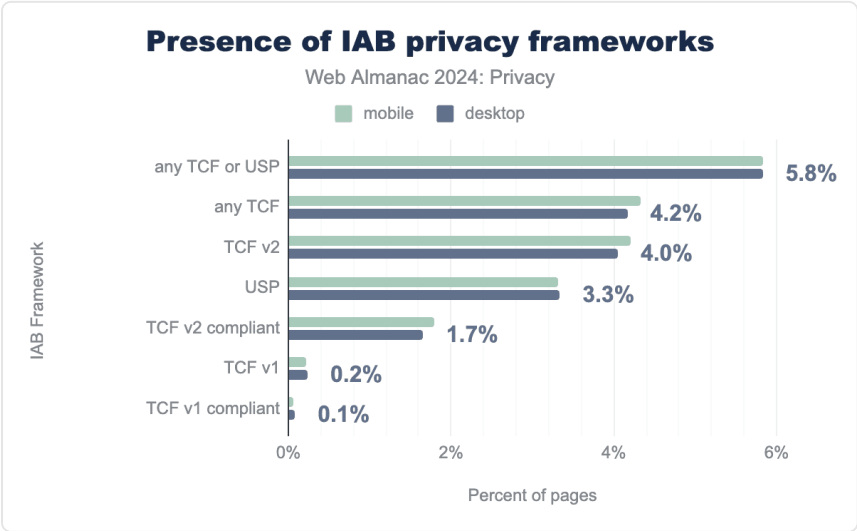


Figure 10.19. Presence of IAB privacy frameworks.

The 2024 data reveals a shift in the privacy landscape compared to 2022³⁸⁹. Firstly, the overall prevalence of IAB frameworks has increased. In 2022, the broadest IAB presence (“IAB all”) was found on 4.4% of pages (desktop and mobile combined). In 2024, any TCF or USP framework appears on 5.8% of home pages. This suggests wider adoption of privacy standards, potentially driven by increased regulatory scrutiny and user awareness.

Examining individual frameworks, TCF usage (any version) appears on 4.2% of pages, while USP stands alone on 3.3% of pages. 4.0% of pages use the latest version of TCF (v2), which also makes it the most prevalent version. TCF v2 with compliant setup (presence of vendor consent configuration) appears on a smaller subset, 1.7% of pages. The older TCF v1, which predates GDPR enforcement, is negligible at 0.2%.

Interestingly, while overall IAB framework usage is up, USP adoption has remained relatively stable, hovering around 3.4% in 2022 and 3.3% in 2024. This suggests that the overall growth in privacy framework adoption is primarily driven by increased TCF usage, specifically TCF v2.

Finally, the shift from TCF v1 to TCF v2 is evident. While TCF v1 in 2022 had some measurable presence (0.3% on mobile), it is nearly obsolete in 2024 at 0.2%. TCF v2 adoption has grown considerably (1.9% to 4%), further indicating a movement toward newer, GDPR-aligned

389. <https://almanac.httparchive.org/en/2022/privacy#iab-consent-frameworks>

consent mechanisms. However, full TCF v2 compliance remains relatively low, highlighting the ongoing challenge of implementing its complex requirements fully.

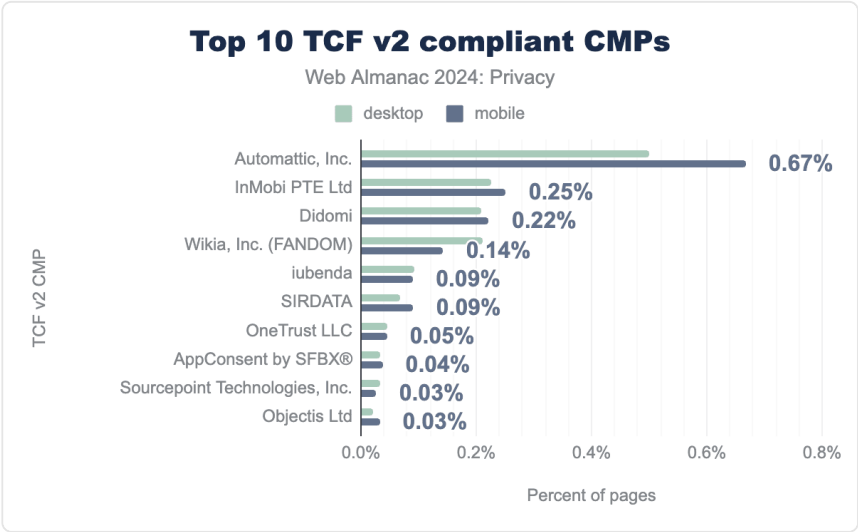


Figure 10.20. Top 10 TCF v2 compliant CMPs.

Our analysis of Consent Management Platform (CMP) usage within the TCF v2 ecosystem shows that Automattic, Inc. leads in adoption, appearing on 0.67% of pages, followed by InMobi PTE Ltd at 0.25% and Didomi at 0.22%. This suggests that certain CMPs have become trusted for managing consent effectively, though the relatively low adoption rates imply that many sites may still depend on in-house solutions or less widely recognized CMPs.

Do Not Track

Do Not Track (DNT) was a browser-based privacy initiative introduced in the early 2010s. It allowed users to set a browser preference indicating that they did not wish to be tracked by websites. However, DNT failed to gain widespread adoption, largely because it was voluntary and lacked enforcement mechanisms.

19.8%

Figure 10.21. Desktop pages still using the DNT (Do Not Track) HTTP header

While DNT was a pioneering idea in user privacy, it ultimately became obsolete as major

advertisers and trackers chose to ignore DNT requests, and it was not enshrined in any legal frameworks. Despite being obsolete, our analysis shows that 19.8% of desktop websites, and 18.4% of mobile websites still support a DNT signal. It's crucial to point out here that while these sites may check for the DNT signal, how well these sites adhere to and comply with the signal is unclear.

Global Privacy Control

Global Privacy Control (GPC) is a more recent initiative designed to give users a simple, browser-based mechanism to communicate their privacy preferences to websites, similar to DNT. However, unlike DNT, GPC is backed by legal regulations like the CCPA (California Consumer Privacy Act).

GPC allows users to signal that they do not want their data to be sold or shared with third parties, and companies are legally obligated to respect this signal under certain laws. Major browsers and privacy-focused extensions support GPC, and it is gaining traction as a more effective tool for user privacy.

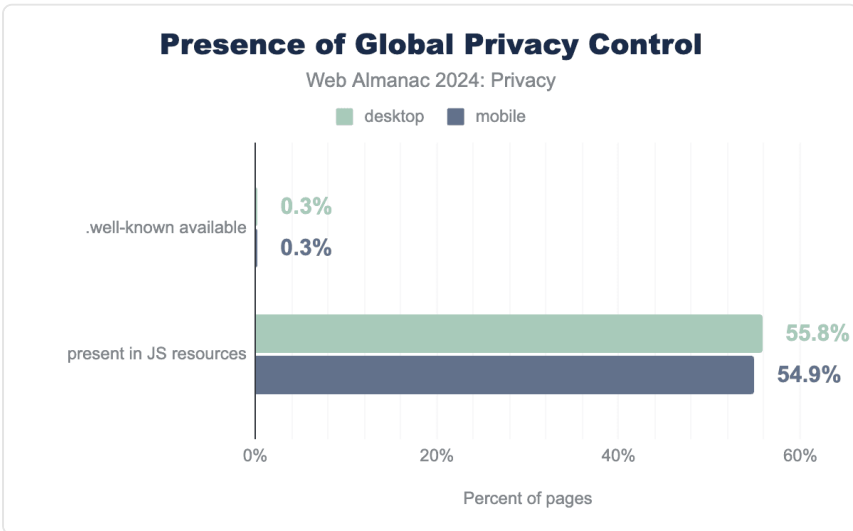


Figure 10.22. Presence of Global Privacy Control.

Analysis shows that on 55.8% of desktop sites and 54.9% of mobile sites the GPC signal can be accessed through JavaScript, which is significantly higher than the DNT signal. Another optional requirement of GPC is a well-known URL which resides at the `/ .well-known/gpc . json` endpoint (relative to the website's origin server URL). This resource is meant to indicate the website's awareness and support of GPC, but at the same time it doesn't guarantee

that it abides by GPC. In our measurements, we find that only 0.2% of mobile and desktop sites have an accessible well-known endpoint.

California Consumer Privacy Act

The California Consumer Privacy Act (CCPA), enacted in 2018, is one of the most significant privacy laws passed in the United States. It grants California residents rights over their personal data, including the right to know what data is being collected, the right to request deletion of their data, and the right to opt out of the sale of their data. CCPA has had a profound impact on the web, as companies across the globe must comply if they collect or process data from California residents. This has led to the introduction of “Do Not Sell My Info” links on many websites and increased awareness around data privacy in the U.S.

Under the law³⁹⁰, any business that does business in California and meets certain size thresholds must provide a way for users to opt-out of the selling or sharing of their personal information. To comply with the law, the California Attorney General’s office recommends³⁹¹ placing a link on the business’ home page with the text “Do Not Sell My Personal Information” and a standardized icon. Building on prior work³⁹² that identified a common set of CCPA link phrases, we conducted an analysis of the prevalence of these links across sites according to their popularity level.

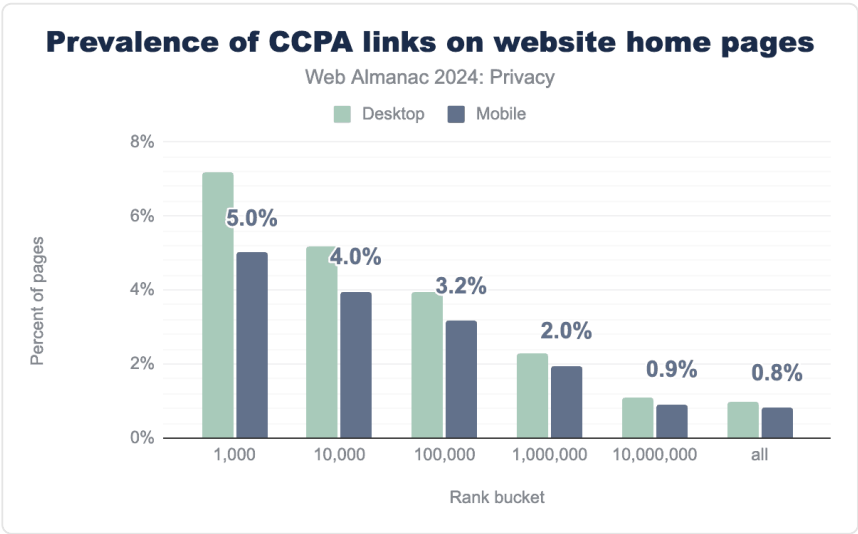


Figure 10.23. Prevalence of CCPA links on website home pages.

390. <https://www.oag.ca.gov/privacy/ccpa>
391. <https://www.oag.ca.gov/privacy/ccpa/icons-download>
392. <https://petsymposium.org/popets/2022/popets-2022-0030.pdf>

Overall, we note that only 0.96% of websites CCPA links. We also note that more websites with higher ranks include CCPA links, as compared to websites with lower ranks. Higher rank websites are more likely to have a CCPA link, either because they are more likely to meet the thresholds to be covered under the CCPA, or simply because they are more aware of the requirements.

However, the rate of links among the top 1,000 websites is only 7.19%, which is quite low. While it is impossible to know how many of these sites meet the requirements to be covered under the CCPA, it is likely that many top ranked websites at least meet the revenue threshold, so unless they take steps to actively block California users, they would appear to be covered.

One limitation of our crawl is that it is geographically distributed, and as such we cannot accurately account for websites that dynamically show a CCPA link only to visitors in California. Therefore, our results likely underestimate the prevalence of these links. However, it is important to note that, as per prior research³⁹³ conducted in 2022, only 17% of CCPA links were dynamically hidden.

Finally, we examine which phrasing is most commonly used in CCPA links.

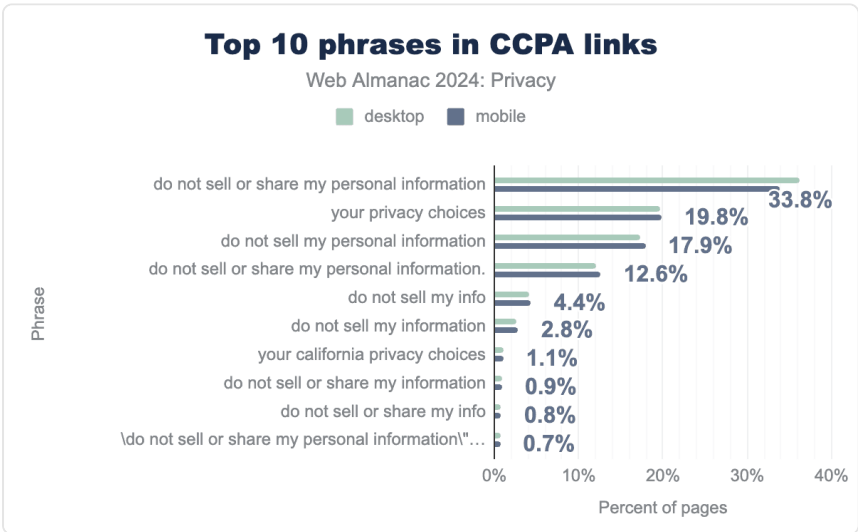


Figure 10.24. Top 10 phrases in CCPA links.

The majority of sites use variants of the phrase recommended under CCPA, “do not sell my personal information”. However, a significant number of sites also contain links titled “your privacy choices”, whose implication is less clear. This may make it more difficult for users to opt-

393. <https://petsymposium.org/popets/2022/popets-2022-0030.pdf>

out on these sites.

Conclusion

We find that online tracking is not just prevalent but almost ubiquitous, with 95% of desktop and 94% of mobile websites containing at least one tracker. Major companies like Google and Facebook dominate this landscape with presence on 68% and 23% of the web pages, respectively. We also observe that trackers utilize both stateful methods (like cookies and local storage) and stateless methods (like browser fingerprinting) to track users across the internet.

Trackers are continually developing sophisticated techniques to bypass privacy-enhancing efforts. Notably, methods such as CNAME cloaking and bounce tracking have emerged, allowing trackers to disguise themselves as first-party entities and exploit browser functionalities to persist in their tracking efforts.

The chapter also assesses the adoption of browser policies designed to enhance user privacy, such as User-Agent Client Hints and Referrer Policy. While there is a gradual increase in the implementation of these features from 2022 (when this analysis was last conducted), their adoption remains uneven across the web. Additionally, the introduction of Privacy Sandbox proposals like the Topics API, Protected Audience API, and Attribution Reporting API signifies a positive shift towards privacy-preserving technologies in online advertising.

On the legal front, regulations like the GDPR and CCPA have prompted a surge in consent dialogues and frameworks like the IAB's Transparency and Consent Framework. Yet, the effectiveness of these measures is questionable due to limited adoption.

Authors



Yash Vekaria

✉ @vekariayash 🌐 Yash-Vekaria 🌐 <https://www.yashvekaria.com/>

Yash Vekaria is a PhD candidate in Computer Science at University of California, Davis³⁹⁴. He carries out web-based large-scale Internet measurements to study and improve the dynamics of web. Specifically, his research is focused at studying and bringing transparency to online tracking practices and user privacy issues.

³⁹⁴. <https://www.ucdavis.edu/>



Benjamin Standaert

 [bstandaert-wustl](#)  [ben-standaert-15580a1b8](#)

Benjamin Standaert is an M.Sc student in the Computer Science department at Washington University in St. Louis³⁹⁵. He has prior experience developing ad-blocking software and privacy-enhancing technologies.



Max Ostapenko

 [max-ostapenko](#)  [max-ostapenko](#)  <https://maxostapenko.com>

Max Ostapenko is a product manager in data and advertising with expertise in tracking, privacy, and web monetization. As a co-maintainer of the HTTP Archive, Max contributes to the accessibility of web insights.



Abdul Haddi Amjad

 [@haddiamjad](#)  [haddiamjad](#)  <https://haddiamjad.github.io/>

Hadi Amjad³⁹⁶ is a Ph.D. student in Computer Science at Virginia Tech³⁹⁷. His research lies at the intersection of internet security, privacy, and program analysis. He specializes in JavaScript program analysis to identify code regions responsible for privacy and security violations on the internet. His work enhances privacy-focused tools, contributing to a safer and more secure web environment.



Yana Dimova

 [ydimova](#)

Yana Dimova is a PhD student at DistriNet, KU Leuven, focusing on the user's perspective of privacy and how they can protect it on the web. Her research interests are online tracking, personal data leaks and privacy and data protection law.

^{395.} <https://wustl.edu/>
^{396.} <https://haddiamjad.github.io>
^{397.} <https://cs.vt.edu/>



Shaoor Munir

✉ @Shaoor_Munir 🔗 shaoormunir 🌐 <https://shaoormunir.com>

Shaoor Munir is a Ph.D. student in Computer Science at UC Davis, specializing in the analysis and development of privacy-enhancing technologies (PETs) and the policies that govern online data collection. His work seeks to advance privacy safeguards for netizens by balancing technological innovation with responsible data governance on the internet.



Chris Böttger

🔗 ChrisBeeti

Chris Böttger is a PhD candidate in Computer Science at the Westphalian University of Applied Science³⁹⁸. His research focuses on web and network security, primarily focusing on user privacy and tracking technologies.



Umar Iqbal

✉ @umaarr6 🔗 UmarIqbal 🌐 <https://www.umariqbal.com>

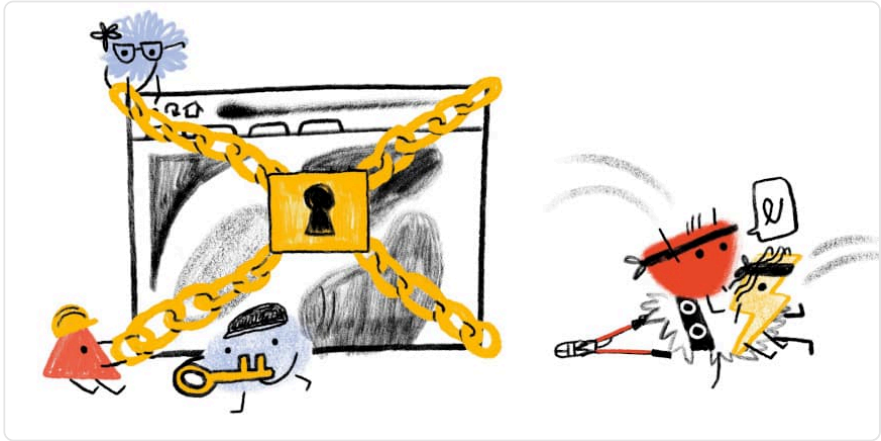
Umar Iqbal is an assistant professor in the department of Computer Science and Engineering at the Washington University in St. Louis³⁹⁹. His research focus on bringing transparency and control in computing systems to empower users, platforms, and regulators to improve user privacy and security.

³⁹⁸. <https://www.en.w-hs.de/>

³⁹⁹. <https://wustl.edu/>

Part II Chapter 11

Security



Written by Gertjan Franken and Vik Vanderlinden

Reviewed by Matteo Große-Kampmann, Alberto Fernandez-de-Retana, and Brian Clark

Analyzed by Jannis Rautenstrauch

Edited by Caleb Queern

Introduction

With how much of our lives happen online these days - whether it's staying in touch, following the news, buying, or even selling products online - web security has never been more important. Unfortunately, the more we rely on these online services, the more appealing they become to malicious actors. As we've seen time and time again, even a single weak spot in the systems we depend on can lead to disrupted services, stolen personal data, or worse. The past two years have been no exception, with a rise in Denial-of-Service (DoS) attacks⁴⁰⁰, bad bots⁴⁰¹, and supply-chain attacks targeting the Web⁴⁰² like never before.

In this chapter, we take a closer look at the current state of web security by analyzing the protections and security practices used by websites today. We explore key areas like Transport Layer Security (TLS), cookie protection mechanisms, and safeguards against third-party

^{400.} <https://blog.cloudflare.com/ddos-threat-report-for-2024-q2/>

^{401.} <https://www.imperva.com/resources/resource-library/reports/2024-bad-bot-report/>

^{402.} <https://www.darkreading.com/vulnerabilities-threats/risin-tide-of-software-supply-chain-attacks>

content inclusion. We'll discuss how security measures like these help prevent attacks, as well as highlight misconfigurations that can undermine them. Additionally, we examine the prevalence of harmful cryptominers and the usage of `security.txt`.

We also investigate the factors driving security practices, analyzing whether elements like country, website category, or technology stack influence the security measures in place. By comparing this year's findings with those from the 2022 Web Almanac⁴⁰³, we highlight key changes and assess long-term trends. This allows us to provide a broader perspective on the evolution of web security practices and the progress made over the years.

Transport security

HTTPS⁴⁰⁴ uses Transport Layer Security (TLS⁴⁰⁵) to secure the connection between client and server. Over the past years, the number of sites using TLS has increased tremendously. As in previous years, adoption of TLS continued to increase, but that increase is slowing down as it closes in to 100%.



Figure 11.1. The percentage of requests that use HTTPS.

The number of requests served using TLS climbed another 4% to 98% on mobile since the last Almanac in 2022.

403. <https://almanac.httparchive.org/en/2022/security>

404. <https://developer.mozilla.org/docs/Glossary/https>

405. <https://www.cloudflare.com/en-gb/learning/ssl/transport-layer-security-tls/>

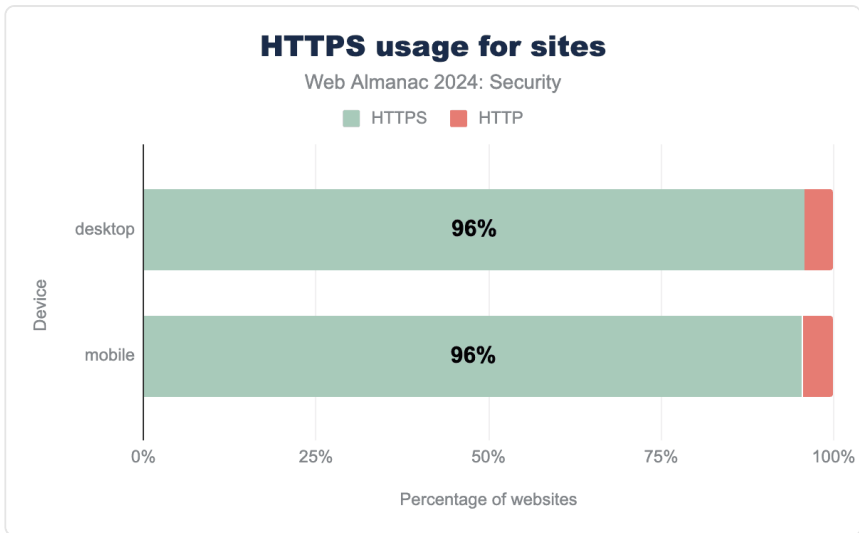


Figure 11.2. The percentage of hosts that use HTTPS.

The number of home pages served over HTTPS on mobile increased from 89% to 95.6%. This percentage is lower than the number of requests served over HTTPS due to the high number of third-party resources websites load, which are more likely to be served over HTTPS.

Protocol versions

Over the years, multiple new versions of TLS have been created. In order to remain secure, it is important to use an up to date version of TLS. The latest version is TLS1.3⁴⁰⁶, which has been the preferred version for a while. Compared to TLS1.2, version 1.3 deprecates some cryptographic protocols still included in 1.2 that were found to have certain flaws and it enforces perfect forward secrecy. Support for older versions of TLS have long been removed by major browser vendors. QUIC (Quick UDP Internet Connections), the protocol underlying HTTP/3 also uses TLS, providing similar security guarantees as TLS1.3.

406. <https://www.cloudflare.com/en-in/learning/ssl/why-use-tls-1.3/>

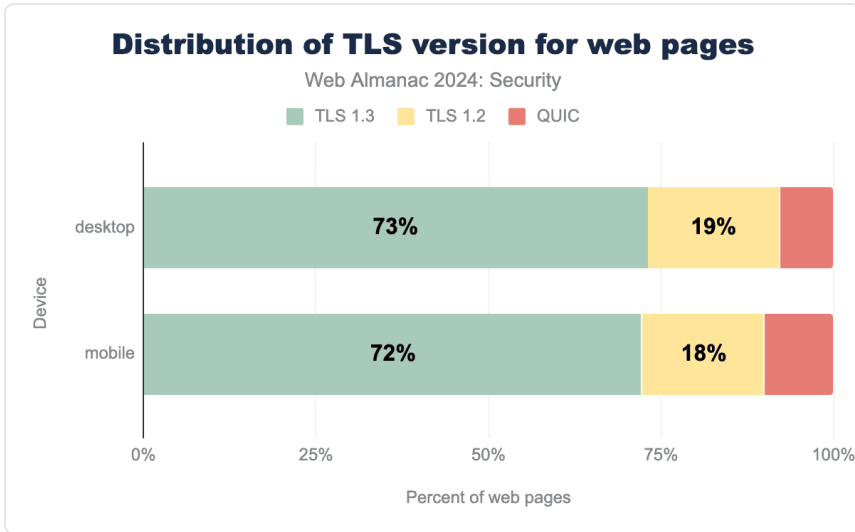


Figure 11.3. The distribution of TLS versions in use.

We find that TLS1.3 is supported and used by 73% of web pages. The use of TLS1.3 overall has grown, even though QUIC has gained significant use compared to 2022, moving from 0% to almost 10% of mobile pages. The use of TLS1.2 continues to decrease as expected. Compared to the last Almanac it decreased by more than 12% for mobile pages, while TLS1.3 has increased by a bit over 2%. It is expected that the adoption of QUIC will continue to rise, as the use of TLS1.2 will continue to decrease.

We assume most websites don't move from TLS1.2 directly to QUIC, but rather that most sites using QUIC migrated from TLS1.3 and others moved from TLS1.2 to TLS1.3, thereby giving the appearance of limited growth of TLS1.3.

Cipher suites

Before client and server can communicate, they have to agree upon the cryptographic algorithms, known as cipher suites⁴⁰⁷, to use. Like last time, over 98% of requests are served using a Galois/Counter Mode (GCM)⁴⁰⁸ cipher, which is considered the most secure option, due to them not being vulnerable to padding attacks⁴⁰⁹. Also unchanged is the almost 79% of requests using a 128-bit key, which is still considered a secure key-length for AES in GCM mode. There are only a handful of suites used on the visited pages. TLS1.3 only supports GCM

407. <https://learn.microsoft.com/en-au/windows/win32/secauthn/cipher-suites-in-channel>

408. https://wikipedia.org/wiki/Galois/Counter_Mode

409. <https://blog.qualys.com/product-tech/2019/04/22/zombie-poodle-and-goldendoodle-vulnerabilities>

and other modern block cipher modes⁴¹⁰, which also simplifies its cipher suite ordering⁴¹¹.

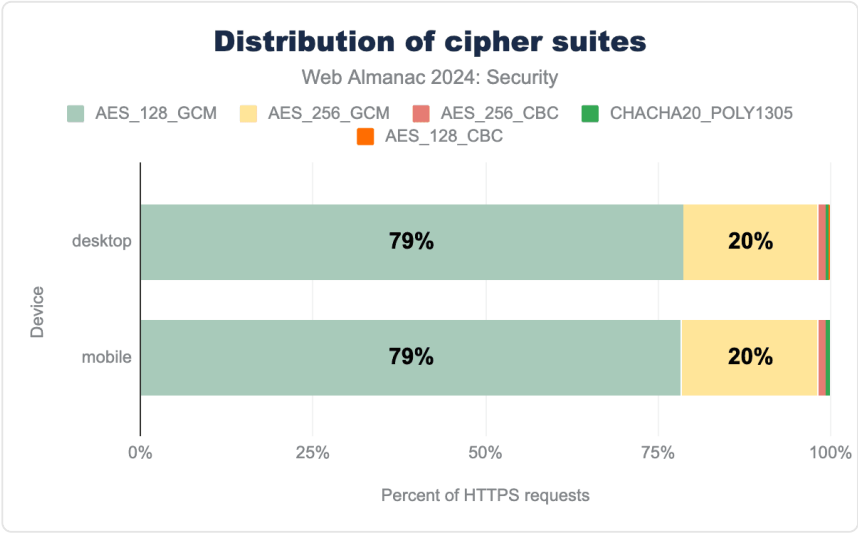


Figure 11.4. The distribution of cipher suites in use.

TLS1.3 makes forward secrecy⁴¹² required, which means it is highly supported on the web. Forward Secrecy is a feature that assures that in case a key in use is leaked, it cannot be used to decrypt future or past messages sent over a connection. This is important to ensure that adversaries storing long-term traffic cannot decrypt the entire conversation as soon as they are able to leak a key. Interestingly, the use of forward secrecy dropped by almost 2% this year, to 95%.

410. <https://datatracker.ietf.org/doc/html/rfc8446#page-133>

411. <https://go.dev/blog/tls-cipher-suites>

412. https://wikipedia.org/wiki/Forward_secrecy

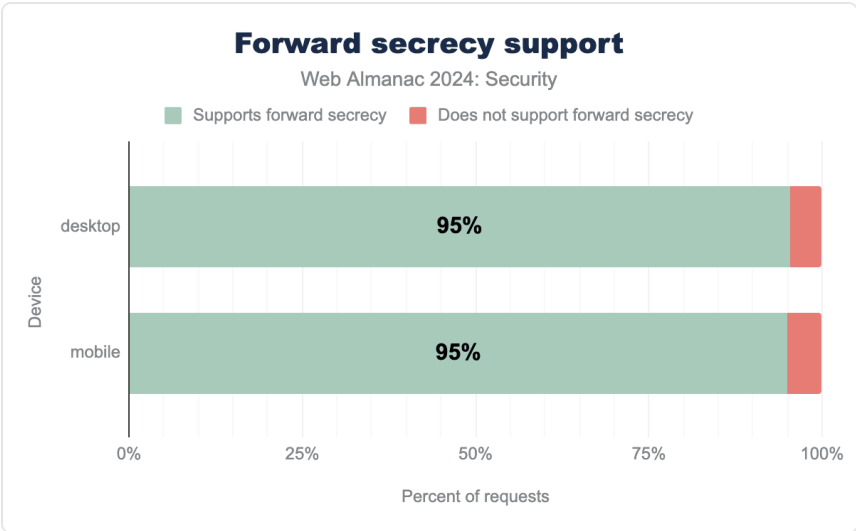


Figure 11.5. The percentage of requests supporting forward secrecy.

Certificate Authorities

In order to use TLS, servers must first get a certificate they can host, which is created by a Certificate Authority⁴¹³ (CA). By retrieving a certificate from one of the trusted CAs, the certificate will be recognized by the browser, thus allowing the user to use the certificate and therefore TLS for their secure communication.

413. <https://www.ssl.com/faqs/what-is-a-certificate-authority/>

Issuer	Desktop	Mobile
R3	44.3%	45.1%
GTS CA 1P5	6.1%	6.6%
E1	4.2%	4.3%
Sectigo RSA Domain Validation Secure Server CA	3.3%	3.1%
R10	2.6%	2.8%
R11	2.6%	2.8%
Go Daddy Secure Certificate Authority - G2	2.0%	1.7%
cPanel, Inc. Certification Authority	1.7%	1.8%
Cloudflare Inc ECC CA-3	1.5%	1.3%
Amazon RSA 2048 M02	1.4%	1.3%

Figure 11.6. The percentage of sites using a certificate issued by a specific issuer.

R3 (an intermediate certificate from Let's Encrypt⁴¹⁴) still leads the charts, although usage dropped compared to last year. Also from Let's Encrypt are the E1, R10 and R11 intermediary certificates that are rising in percentage of websites using them.

R3 and E1 were issued in 2020 and are only valid for 5 years⁴¹⁵, which means it will expire in September 2025⁴¹⁶. Around a year before the expiry of intermediate certificates, Let's Encrypt issues new intermediates that will gradually take over from the older ones. This March, Let's Encrypt issued their new intermediates⁴¹⁷, which include R10 and R11 that are only valid for 3 years. These latter two certificates will take over from R3 directly, which should be reflected in next year's Almanac.

Along with the rise in the number of Let's Encrypt issued certificates, other current top 10 providers have seen a decrease in their share of certificates issued, except for GTS CA 1P5 that rose from close to 0% to over 6.5% on mobile. Of course it is possible that at the time of our analysis a CA was in the process of switching intermediate certificates, which could mean they serve a larger percentage of sites than reflected.

414. <https://letsencrypt.org/>

415. <https://letsencrypt.org/2024/03/19/new-intermediate-certificates.html>

416. <https://letsencrypt.org/2024/03/19/new-intermediate-certificates.html>

417. <https://letsencrypt.org/2024/03/19/new-intermediate-certificates.html>

56%

Figure 11.7. The percentage of pages that use a Let's Encrypt issued certificate on mobile.

When we sum together the use of all certificates of Let's Encrypt, we find that they issue over 56% of the certificates currently in use.

HTTP Strict Transport Security

HTTP Strict Transport Security (HSTS)⁴¹⁸ is a response header that a server can use to communicate to the browser that only HTTPS should be used to reach pages hosted on this domain, instead of first reaching out over HTTP and following a redirect.

30%

Figure 11.8. The percentage of requests that have a HSTS header on mobile.

Currently, 30% of responses on mobile have a HSTS header, which is a 5% increase compared to 2022. Users of the header can communicate directives to the browser by adding them to the header value. The `max-age` directive is obligated. It indicates to the browser the time it should continue to only visit the page over HTTPS in seconds.

⁴¹⁸. <https://developer.mozilla.org/docs/Web/HTTP/Headers/Strict-Transport-Security>

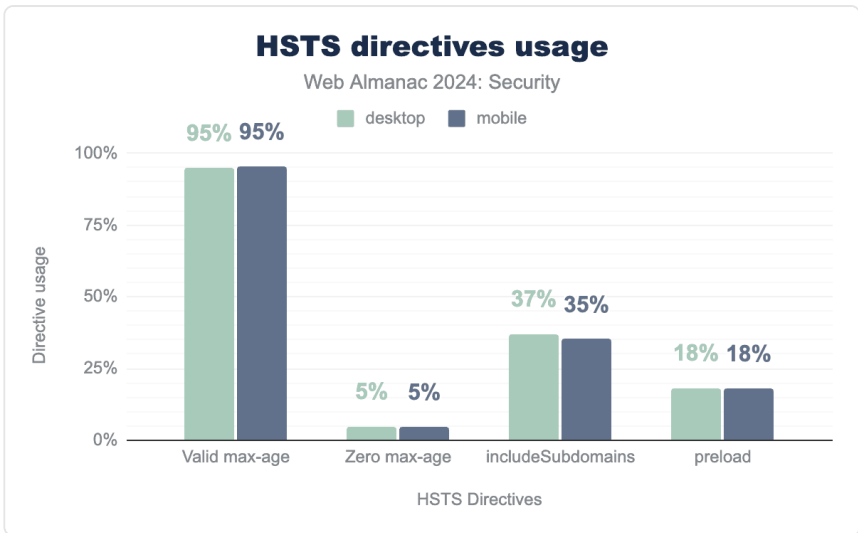


Figure 11.9. The usage of specified HSTS directives.

The share of requests with a valid `max-age` has remained unchanged at 95%. The other , optional, directives (`includeSubdomains` and `preload`) both see a slight increase of 1% compared to 2022 to 35% and 18% on mobile respectively. The `preload` directive, which is not part of the HSTS specification⁴¹⁹, requires the `includeSubdomains` to be set and also requires a `max-age` larger than 1 year (or 31,536,000 seconds).

419. https://developer.mozilla.org/docs/Web/HTTP/Headers/Strict-Transport-Security#preloading_strict_transport_security

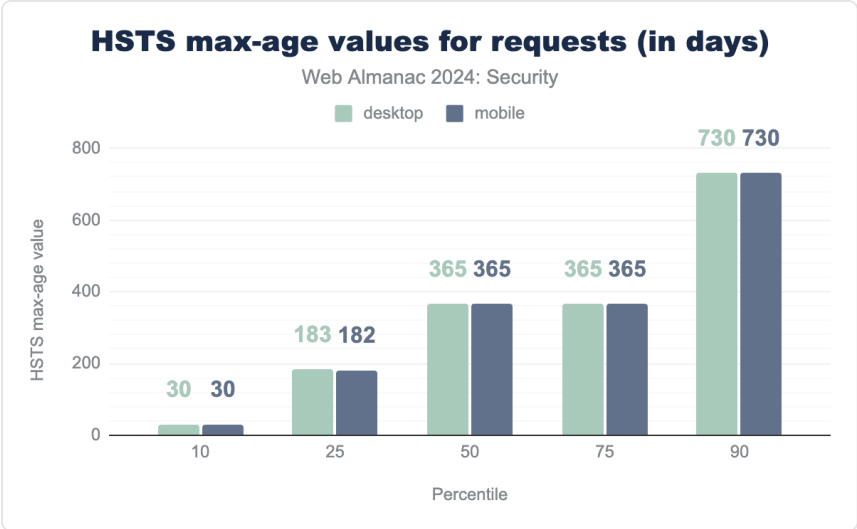


Figure 11.10. The distribution of HSTS max-age values by percentile.

The distribution of valid `max-age` values has remained almost the same as in 2022, with the exception that the 10th percentile on mobile has decreased from 72 to 30 days. The median value of `max-age` remains at 1 year.

Cookies

Websites can store small pieces of data in a user’s browser by setting an HTTP cookie. Depending on the cookie’s attributes, it will be sent with every subsequent request to that website. As such, cookies can be used for purposes of implicit authentication, tracking or storing user preferences.

When cookies are used for authenticating users, it is paramount to protect them from abuse. For instance, if an adversary gets ahold of a user’s session cookie, they could potentially log into the victim’s account.

To protect their user’s from attacks like Cross-Site Request Forgery (CSRF)⁴²⁰, session hijacking⁴²¹, Cross-Site Script Inclusion (XSSI)⁴²² and Cross-Site Leaks⁴²³, websites are expected to securely configure authentication cookies.

420. <https://owasp.org/www-community/attacks/csrf>

421. https://owasp.org/www-community/attacks/Session_hijacking_attack

422. https://owasp.org/www-project-web-security-testing-guide/v41/4-Web_Application_Security_Testing/11-Client_Side_Testing/13-Testing_for_Cross_Site_Script_Inclusion

423. <https://xleaks.dev/>

Cookie attributes

The three cookie attributes outlined below enhance the security of authentication cookies against the attacks mentioned earlier. Ideally, developers should consider using all attributes, as they provide complementary layers of protection.

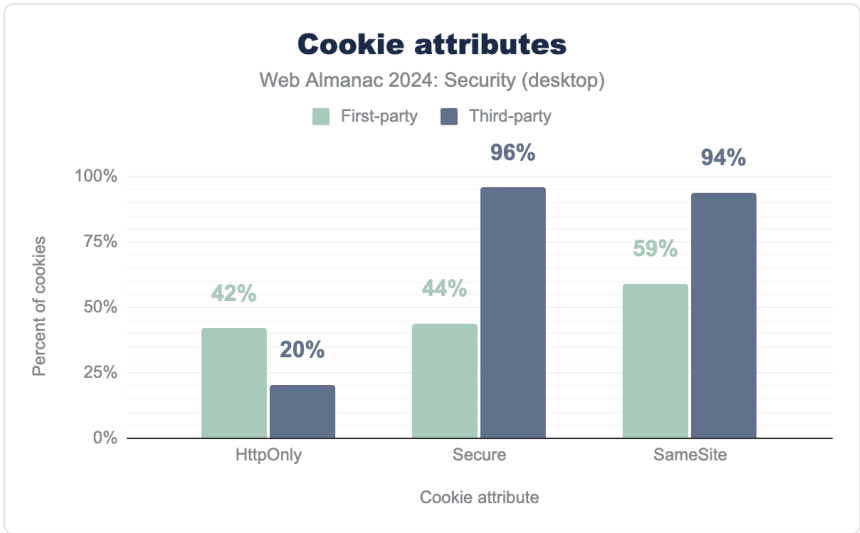


Figure 11.11. Cookie attributes (desktop).

HttpOnly

By setting this attribute, the cookie is not allowed to be accessed or manipulated through the JavaScript `document.cookie` API. This prevents a Cross-Site Scripting (XSS)⁴²⁴ attack from gaining access to cookies containing secret session tokens.

With 42% of cookies having the `HttpOnly` attribute in a first-party context on desktop, the usage has risen by 6% compared to 2022. As for third-party requests, the usage has decreased by 1%.

Secure

Browsers only transmit cookies with the `Secure` attribute over secure, encrypted channels, such as HTTPS, and not over HTTP. This ensures that man-in-the-middle attackers cannot

424. <https://owasp.org/www-community/attacks/xss/>

intercept and read sensitive values stored in cookies.

The use of the `Secure` attribute has been steadily increasing over the years. Since 2022, an additional 7% of cookies in first-party contexts and 6% in third-party contexts have been configured with this attribute. As discussed in previous editions of the Security chapter, the significant difference in adoption between the two contexts is largely due to the requirement that third-party cookies with `SameSite=None` must also be marked as `Secure`. This highlights that additional security prerequisites for enabling desired non-default functionality are an effective driver for the adoption of security features.

SameSite

The most recently introduced cookie attribute, `SameSite`, allows developers to control whether a cookie is allowed to be included in third-party requests. It is intended as an additional layer of defense against attacks like CSRF.

The attribute can be set to one of three values: `Strict`, `Lax`, or `None`. Cookies with the `Strict` value are completely excluded from cross-site requests. When set to `Lax`, cookies are only included in third-party requests under specific conditions, such as navigational GET requests, but not POST requests. By setting `SameSite=None`, the cookie bypasses the same-site policy and is included in all requests, making it accessible in cross-site contexts.

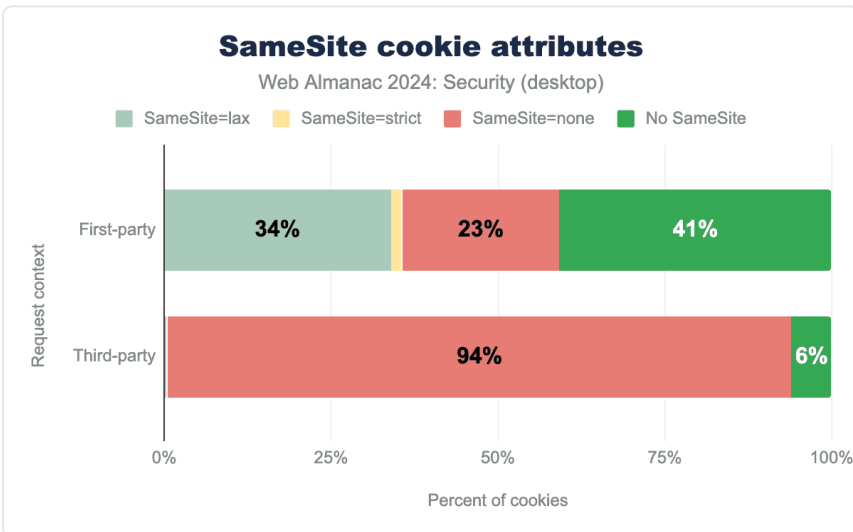


Figure 11.12. SameSite cookie attributes (desktop).

While the relative number of cookies with a `SameSite` attribute has increased compared to

2022, this rise is largely attributable to cookies being explicitly excluded from the same-site policy by setting `SameSite=None`.

It's important to note that all cookies without a `SameSite` attribute are treated as `SameSite=Lax` by default. Consequently, a total of 75% of cookies set in a first-party context are effectively treated as if they were set to `Lax`.

Prefixes

Session fixation⁴²⁵ attacks can be mitigated by using cookie prefixes like `__Secure-` and `__Host-`. When a cookie name starts with `__Secure-`, the browser requires the cookie to have the `Secure` attribute and to be transmitted over an encrypted connection. For cookies with the `__Host-` prefix, the browser additionally mandates that the cookie includes the `Path` attribute set to `/` and excludes the `Domain` attribute. These requirements help protect cookies from man-in-the-middle attacks and threats from compromised subdomains.

Type of cookie	<code>__Secure-</code>	<code>__Host-</code>
First-party	0.05%	0.17%
Third-party	0.00%	0.04%

Figure 11.13. `__Secure-` and `__Host-` prefixes (desktop).

The adoption of cookie prefixes remains low, with less than 1% of cookies using these prefixes on both desktop and mobile platforms. This is particularly surprising given the high adoption rate of cookies with the `Secure` attribute, the only prerequisite for cookies prefixed with `__Secure-`. However, changing a cookie's name can require significant refactoring, which is presumably a reason why developers tend to avoid this.

Cookie age

Websites can control how long browsers store a cookie by setting its lifespan. Browsers will discard cookies when they reach the age specified by the `Max-Age` attribute or when the timestamp defined in the `Expires` attribute is reached. If neither attribute is set, the cookie is considered a session cookie and will be removed when the session ends.

425. https://owasp.org/www-community/attacks/Session_fixation

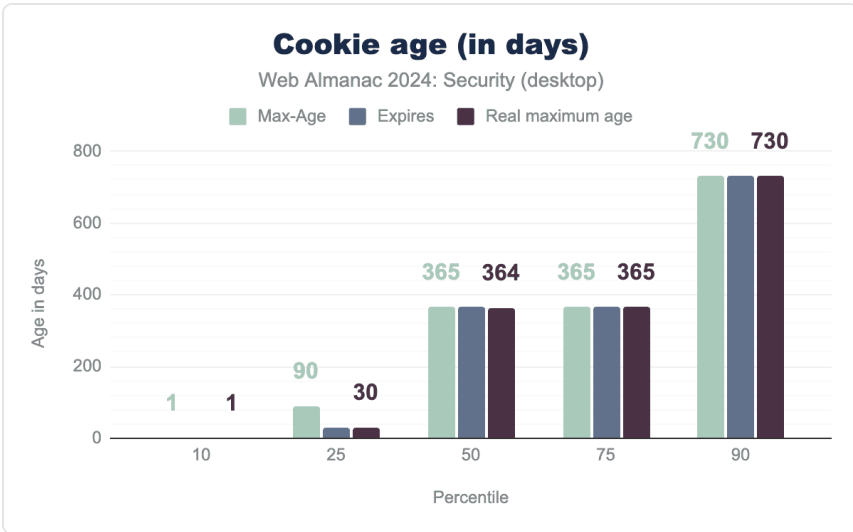


Figure 11.14. Cookie age in days (desktop).

The distribution of cookie ages has remained largely unchanged compared to lat year⁴²⁶. However, since then, the cookie standard working draft⁴²⁷ has been updated, capping the maximum cookie age to 400 days. This change has already been implemented in Chrome⁴²⁸ and Safari. Based on the percentiles shown above, in these browsers, more than 10% of all observed cookies have their age capped to this 400-day limit.

Content inclusion

Content inclusion is a foundational aspect of the Web, allowing resources like CSS, JavaScript, fonts, and images to be shared via CDNs or reused across multiple websites. However, fetching content from external or third-party sources introduces significant risks. By referencing resources outside your control, you are placing trust in those third parties, which could either turn malicious or be compromised. This can lead to so-called supply-chain attacks, like the recent polyfill incident where compromised resources affected hundreds of thousands of websites⁴²⁹. Therefore, security policies that govern content inclusion are essential for protecting web applications.

426. <https://almanac.httparchive.org/en/2022/security#cookie-age>

427. <https://httpwg.org/http-extensions/draft-ietf-httpbis-rfc6265bis.html#name-cookie-lifetime-limits>

428. <https://developer.chrome.com/blog/cookie-max-age-expires>

429. <https://www.darkreading.com/remote-workforce/polyfillio-supply-chain-attack-smacks-down-100k-websites>

Content Security Policy

Websites can exert greater control over their embedded content by deploying a Content Security Policy (CSP)⁴³⁰ through either the `Content-Security-Policy` response header or by defining the policy in a `<meta>` tag. The wide range of directives available in CSP allows websites to specify, in a fine-grained manner, which resources can be fetched and from which origins.

In addition to vetting included content, CSP can serve other purposes as well, such as enforcing the use of encrypted channels with the `upgrade-insecure-requests` directive and controlling where the site can be embedded to protect against clickjacking attacks using the `frame-ancestors` directive.



Figure 11.15. Relative increase in adoption for Content-Security-Policy header from 2022.

The adoption rate of CSP headers increased from 15% of all hosts in 2022 to 19% this year. This amounts to a relative increase of 27%. Over these two years, the relative increase was 12% between 2022 and 2023, and 14% between 2023 and 2024.

Looking back, overall CSP adoption was only at 12% of hosts in 2021, so it's encouraging to see that growth has remained steady. If this trend continues, projections suggest that CSP adoption will surpass the 20% mark in next year's Web Almanac.

Directives

Most websites utilize CSP for purposes beyond controlling embedded resources, with the `upgrade-insecure-requests` and `frame-ancestors` directives being the most popular.

430. <https://developer.mozilla.org/docs/Web/HTTP/CSP>

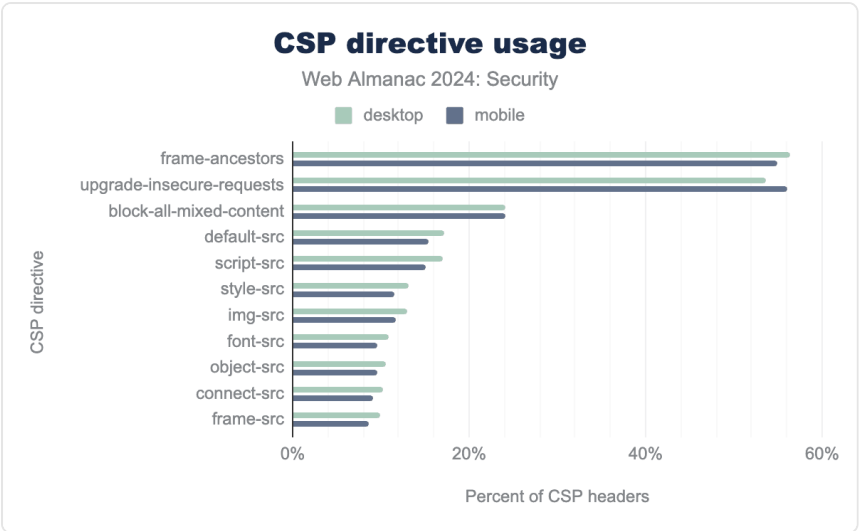


Figure 11.16. Most common directives used in CSP.

The `block-all-mixed-content` directive, which has been deprecated in favor of `upgrade-insecure-requests`, is the third most used directive. Although we observed a relative decrease of 12.5% for desktop and 13.8% for mobile in its usage between 2020 and 2021, the decline has since slowed to an average yearly decrease of 4.4% for desktop and 6.4% for mobile since 2022.

Policy	Desktop	Mobile
upgrade-insecure-requests;	27%	30%
block-all-mixed-content; frame-ancestors 'none'; upgrade-insecure-requests;	22%	22%
frame-ancestors 'self';	11%	10%

Figure 11.17. Most prevalent CSP headers.

The top three directives also make up the building blocks of the most prevalent CSP definitions. The second most commonly used CSP definition includes both `block-all-mixed-content` and `upgrade-insecure-requests`. This suggests that many websites use `block-all-mixed-content` for backward compatibility, as newer browsers will ignore this directive if `upgrade-insecure-requests` is present.

Directive	Desktop	Mobile
upgrade-insecure-requests	-1%	0%
frame-ancestors	5%	3%
block-all-mixed-content	-9%	-13%
default-src	-9%	-6%
script-src	-3%	-2%
style-src	-8%	-2%
img-src	-3%	9%
font-src	-4%	8%
connect-src	3%	17%
frame-src	4%	16%
object-src	16%	17%

Figure 11.18. Relative usage change of CSP directives.

All other directives shown in the table above are used for content inclusion control. Overall, usage has remained relatively stable. However, a notable change is the increased use of the `object-src` directive, which has surpassed `connect-src` and `frame-src`. Since 2022, the usage of `object-src` has risen by 15.9% for desktop and 16.8% for mobile.

Among the most notable decreases in usage is `default-src`, the catch-all directive. This decline could be explained by the increasing use of CSP for purposes beyond content inclusion, such as enforcing HTTP upgrades to HTTPS or controlling the embedding of the current page – situations where `default-src` is not applicable, as these directives don't fallback to it. This change in CSP purpose is confirmed by the most prevalent CSP headers listed in Figure 17, which all have seen an increase in usage since 2022. However, directives like `upgrade-insecure-requests` and `block-all-mixed-content`, while part of these most common CSP headers, are being used less overall, as seen in Figure 18.

Keywords for `script-src`

One of the most important directives of CSP is `script-src`, as curbing scripts loaded by the website hinders potential adversaries greatly. This directive can be used with several attribute

keywords.

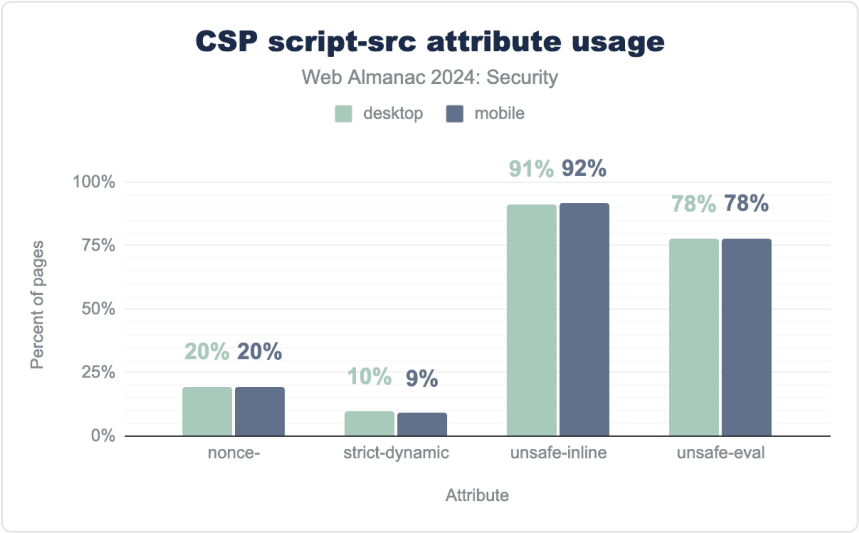


Figure 11.19. Prevalence of CSP `script-src` keywords.

The `unsafe-inline` and `unsafe-eval` directives can significantly reduce the security benefits provided by CSP. The `unsafe-inline` directive permits the execution of inline scripts, while `unsafe-eval` allows the use of the `eval` JavaScript function. Unfortunately, the use of these insecure practices remains widespread, demonstrating the challenges of avoiding use of inline scripts and use of the `eval` function.

Keyword	Desktop	Mobile
nonce-	62%	39%
strict-dynamic	61%	88%
unsafe-inline	-3%	-3%
unsafe-eval	-3%	0%

Figure 11.20. Relative usage change of CSP `script-src` keywords.

However, the increasing adoption of the `nonce-` and `strict-dynamic` keywords is a positive development. By using the `nonce-` keyword, a secret nonce can be defined, allowing only inline scripts with the correct nonce to execute. This approach is a secure alternative to the `unsafe-inline` directive for permitting inline scripts. When used in combination with

the `strict-dynamic` keyword, nonced scripts are permitted to import additional scripts from any origin. This approach simplifies secure script loading for developers, as it allows them to trust a single nonced script, which can then securely load other necessary resources.

Allowed hosts

CSP is often regarded as one of the more complex security policies, partly due to the detailed policy language, providing fine-grained control over resource inclusion.

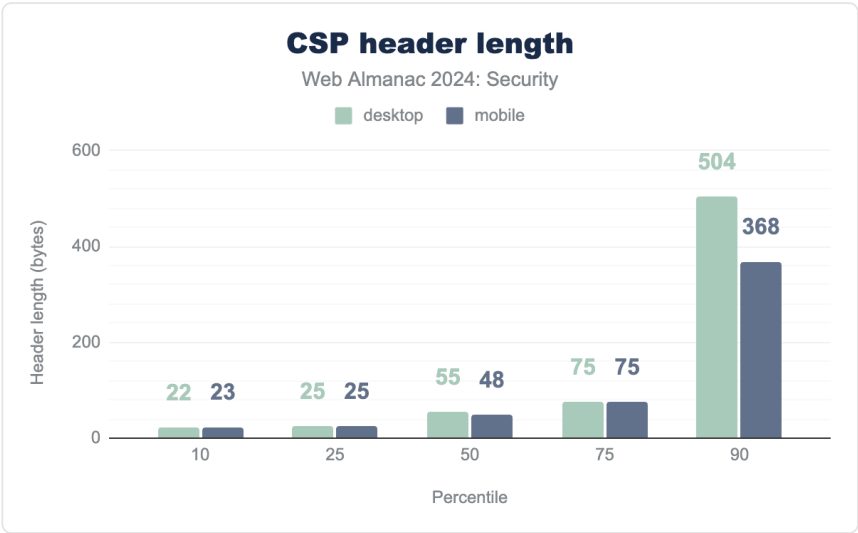


Figure 11.21. CSP header length.

Reviewing the observed CSP header lengths, we find that 75% of all headers are 75 bytes or shorter. For context, the longest policy shown in Figure 17 is also 75 bytes. At the 90th percentile, desktop policies reach 504 bytes and mobile policies 368 bytes, indicating that many websites find it necessary to implement relatively lengthy Content Security Policies. However, when analyzing the distribution of unique allowed hosts across all policies, the 90th percentile shows just 2 unique hosts.

The highest number of unique allowed hosts in a policy was 1,020, while the longest Content Security Policy header reached 65,535 bytes. However, this latter header is inflated by a large number of repeated `,` characters for unknown reasons. The second longest CSP header, which is valid, spans 33,123 bytes. This unusually large size is due to hundreds of occurrences of the `adservice.google` domain, each with variations in the top-level domain. Excerpt:

adservice.google.com adservice.google.ad adservice.google.ae ...

This suggests that the long tail of excessively large CSP headers is likely caused by computer-generated exhaustive lists of origins. Although this may seem like a specific edge case, it highlights a limitation of CSP: the lack of regex functionality, which could otherwise provide a more efficient and elegant solution to handle such cases. However, depending on the websites implementation, this issue could also be solved by employing the `strict-dynamic` and `nonce-` keyword in the `script-src` directive, which enables the allowed script with nonce to load additional scripts.

The most common HTTPS origins included in CSP headers are used for loading fonts, ads and other media fetched from CDNs:

Host	Desktop	Mobile
https://www.googletagmanager.com	0.41%	0.32%
https://fonts.gstatic.com	0.34%	0.27%
https://fonts.googleapis.com	0.33%	0.27%
https://www.google-analytics.com	0.33%	0.26%
https://www.google.com	0.30%	0.26%
https://www.youtube.com	0.26%	0.23%
https://*.google-analytics.com	0.25%	0.23%
https://connect.facebook.net	0.20%	0.19%
https://*.google.com	0.19%	0.19%
https://*.googleapis.com	0.19%	0.19%

Figure 11.22. Most frequently allowed HTTP(S) hosts in CSP policies.

As for WSS origins, used for allowing WebSocket connections to certain origins, the following were found the most common:

Host	Desktop	Mobile
wss://*.intercom.io	0.08%	0.08%
wss://*.hotjar.com	0.08%	0.07%
wss://www.livejournal.com	0.05%	0.06%
wss://*.quora.com	0.04%	0.06%
wss://*.zopim.com	0.03%	0.02%

Figure 11.23. Most frequently allowed WS(S) hosts in CSP policies.

Two of these origins are related to customer service and ticketing (`intercom.io` , `zopim.com`), one is used for website analytics (`hotjar.com`), and two are associated with social media (`www.livejournal.com` , `quora.com`). For four out of these five websites, we found specific instructions on how to add the origin to the website’s content security policy. This is considered good practice, as it discourages website administrators from using wildcards to allow third-party resources, which would reduce security by allowing broader access than necessary.

Subresource Integrity

While CSP is a powerful tool for ensuring that resources are only loaded from trusted origins, there remains a risk that those resources could be tampered with. For instance, a script might be loaded from a trusted CDN, but if that CDN suffers a security breach and its scripts are compromised, any website using one of those scripts could become vulnerable as well.

Subresource Integrity (SRI)⁴³¹ provides a safeguard against this risk. By using the `integrity` attribute in `<script>` and `<link>` tags, a website can specify the expected hash of a resource. If the hash of the received resource does not match the expected hash, the browser will refuse to render the resource, thereby protecting the website from potentially compromised content.

23%

Figure 11.24. Desktop sites using SRI.

431. https://developer.mozilla.org/docs/Web/Security/Subresource_Integrity

SRI is used by 23.2% and 21.3% of all observed pages for desktop and mobile respectively. This amounts to a relative change in adoption of 13.3% and 18.4% respectively.

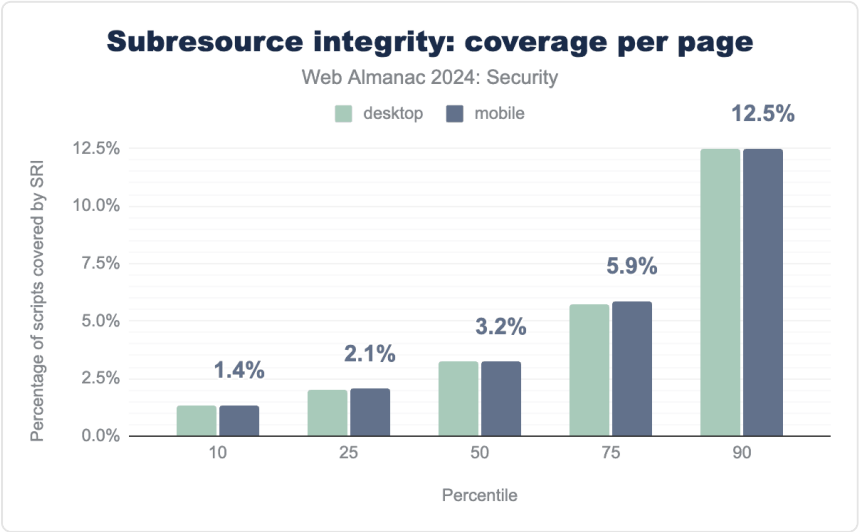


Figure 11.25. SRI coverage per page.

The adoption of Subresource Integrity seems to be stagnating, with the median percentage of scripts per page checked against a hash remaining at 3.23% for both desktop and mobile. This figure has remained virtually unchanged since 2022.

Host	Desktop	Mobile
www.gstatic.com	35%	35%
cdnjs.cloudflare.com	7%	7%
cdn.userway.org	6%	6%
static.cloudflareinsights.com	6%	6%
code.jquery.com	5%	6%
cdn.jsdelivr.net	4%	4%
d3e54v103j8qbb.cloudfront.net	2%	2%
t1.daumcdn.net	2%	1%

Figure 11.26. Most common hosts from which SRI-protected scripts are included.

Most of the hosts from which resources are fetched and protected by SRI are CDNs. A notable difference from 2022's data is the absence of `cdn.shopify.com` from the top hosts list (previously 22% on desktop and 23% on mobile). This is due to Shopify having dropped SRI in favor of similar functionality provided by the `integrity` attribute of `importmap`, which they explain in a blogpost⁴³².

Permissions Policy

The Permissions Policy⁴³³ (formerly known as the Feature Policy) is a set of mechanisms that allow websites to control which browser features can be accessed on a webpage, such as geolocation, webcam, microphone, and more. By using the Permissions Policy, websites can restrict feature access for both the main site and any embedded content, enhancing security and protecting user privacy. This is configured through the `Permissions-Policy` response header for the main site and all its embedded `<iframe>` elements. Additionally, web administrators can set individual policies for specific `<iframe>` elements using their `allow` attribute.

+1.3%

Figure 11.27. Relative increase in adoption of the `Permissions-Policy` header from 2022.

In 2022, the adoption of the `Permissions-Policy` header saw a significant relative increase of 85%. However, from 2022 to this year, the growth rate has drastically slowed to just 1.3%. This is expected, as the Feature Policy was renamed to Permissions Policy at the end of 2020, resulting in an initial peak. In the following years, growth has remained very low since the header is still supported exclusively by Chromium-based browsers.

432. <https://shopify.engineering/shipping-support-for-module-script-integrity-in-chrome-safari#>

433. https://developer.mozilla.org/docs/Web/HTTP/Permissions_Policy

Header
interest-cohort=()
geolocation=(),midi=(),sync- xhr=(),microphone=(),camera=(),magnetometer=(),gyroscope=(),fullscreen=(self),payment=
accelerometer=(), autoplay=(), camera=(), cross-origin-isolated=(), display-capture=(self), encrypted-media=(), fullscreen=*, geolocation=(self), gyroscope=(), keyboard-map=(), magnetometer=(), microphone=(), midi=(), payment=*, picture-in-picture=(), publickey-credentials-get=(), screen-wake-lock=(), sync-xhr=(), usb=(), spatial-tracking=(), gamepad=(), serial=()
accelerometer=(self), autoplay=(self), camera=(self), encrypted-media=(self), fullscreen=(self), geolocation=(self), gyroscope=(self), magnetometer=(self), microphone=(self), midi=(self), payment=(self), usb=(self)
accelerometer=(), camera=(), geolocation=(), gyroscope=(), magnetometer=(), microphone=(), payment=(), usb=()
browsing-topics=()
geolocation=self

Figure 11.28. Most prevalent Permission Policies.

Only 2.8% of desktop hosts and 2.5% of mobile hosts set the policy using the `Permissions-Policy` response header. The policy is primarily used to exclusively opt out of Google's Federated Learning of Cohorts (FLoC); 21% of hosts that implement the `Permissions-Policy` header set the policy as `interest-cohort=()`. This usage is partly due to the controversy that FLoC sparked during its trial period. Although FLoC was ultimately replaced by the Topics API, the continued use of the `interest-cohort` directive highlights how specific concerns can shape the adoption of web policies.

All other observed headers with at least 2% of hosts implementing them, are aimed at restricting the permission capabilities of the website itself and/or its embedded `<iframe>` elements. Similar to the Content Security Policy, the Permissions Policy is “open by default” instead of “secure by default”; absence of the policy entails absence of protection. This approach aims to avoid breaking website functionality when introducing new policies. Notably, 0.28% of sites explicitly use the `*` wildcard policy, allowing the website and all embedded `<iframe>` elements (where no more restrictive `allow` attribute is present) to request any permission - though this is the default behavior when the Permissions Policy is not set.

The Permissions Policy can also be defined individually for each embedded `<iframe>` through

its `allow` attribute. For example, an `<iframe>` can be permitted to use the geolocation and camera permissions by setting the attribute as follows:

```
<iframe src="https://example.com" allow="geolocation 'self'; camera
*;"></iframe>
```

Out of the 30.4 million `<iframe>` elements observed in the desktop crawl, 35.2% included the `allow` attribute. This marks a significant increase compared to **even just the previous month**, when only 14.4% of `<iframe>` elements had the `allow` attribute - indicating that its usage has more than doubled in just one month. A plausible explanation for this rapid change is that one or several widely-used third-party services have propagated this update across their `<iframe>` elements. Given the ad-specific directives we now observe (displayed in the table below, row 1 and 3) - none of which were present in 2022 - it is likely that an ad service is responsible for this shift.

Directive	Desktop	Mobile
join-ad-interest-group	43%	44%
attribution-reporting	28%	280%
run-ad-auction	25%	24%
encrypted-media	19%	18%
autoplay	18%	18%
picture-in-picture	12%	12%
clipboard-write	10%	10%
gyroscope	9%	10%
accelerometer	9%	10%
web-share	7%	7%

Figure 11.29. Most prevalent `allow` attribute directives.

Compared to 2022, the top 10 most common directives are now led by three newly introduced directives: `join-ad-interest-group`, `attribution-reporting` and `run-ad-auction`. The first and third directives are specific to Google’s Privacy Sandbox. For all observed directives in the top 10, almost none were used in combination with an origin or

keyword (i.e., `'src'`, `'self'`, and `'none'`), meaning the loaded page is allowed to request the indicated permission regardless of its origin.

Iframe sandbox

Embedding third-party websites within `<iframe>` elements always carries risks, though it might be necessary to enrich a web application's functionality. Website administrators should be aware that a rogue `<iframe>` can exploit several mechanisms to harm users, such as launching pop-ups or redirecting the top-level page to a malicious domain.

These risks can be curbed by employing the `sandbox` attribute on `<iframe>` elements. Doing this, the content loaded within is restricted to rules defined by the attribute, and can be used to prevent the embedded content from abusing capabilities. When provided with an empty string as value, the policy is strictest. However, this policy can be relaxed by adding specific directives, of which each has their own specific relaxation rules. For example, the following `<iframe>` would allow the embedded webpage to run scripts:

```
<iframe src="https://example.com" sandbox="allow-scripts"></iframe>
```

The `sandbox` attribute was observed in 19.9% and 19.8% of `<iframe>` elements for desktop and mobile respectively, a slight drop from the 22.1% and 21.2% reported in 2022. Much like the sudden spike in `allow` attribute usage mentioned in the previous section, this decline could be attributed to a change in the modus operandi of an embedded service, where the `sandbox` attribute was omitted from the template `<iframe>`.

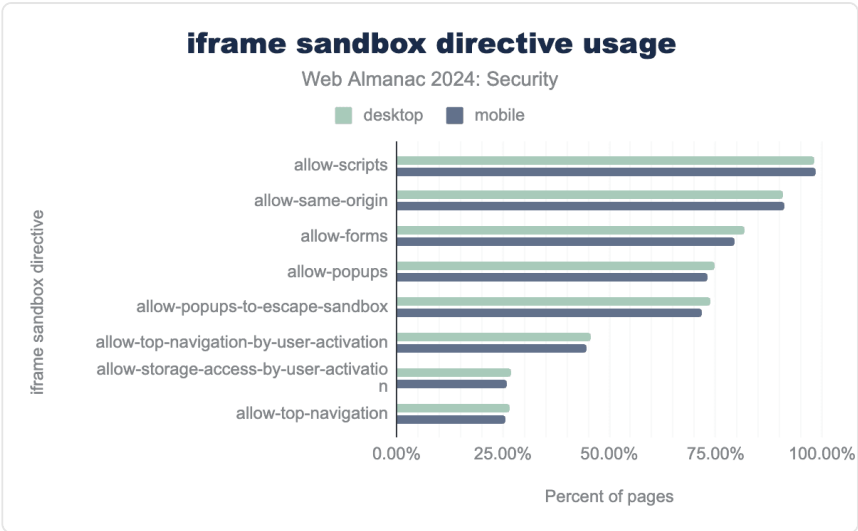


Figure 11.30. Prevalence of sandbox directives on iframes.

More than 98% of pages that have the `sandbox` attribute set in an iframe, use it to allow scripts in the embedded webpage, using the `allow-scripts` directive.

Attack preventions

Web applications can be exploited in numerous ways, and while there are many methods to protect them, it can be difficult to see the full range of options. This challenge is heightened when protections are not enabled by default or require opt-in. In other words, website administrators must be aware of potential attack vectors relevant to their application and how to prevent them. Therefore, evaluating which attack prevention measures are in place is crucial for assessing the overall security of the Web.

Security header adoption

Most security policies are configured through response headers, which instruct the browser on which policies to enforce. Although not every security policy is relevant for every website, the absence of certain security headers suggests that website administrators may not have considered or prioritized security measures.

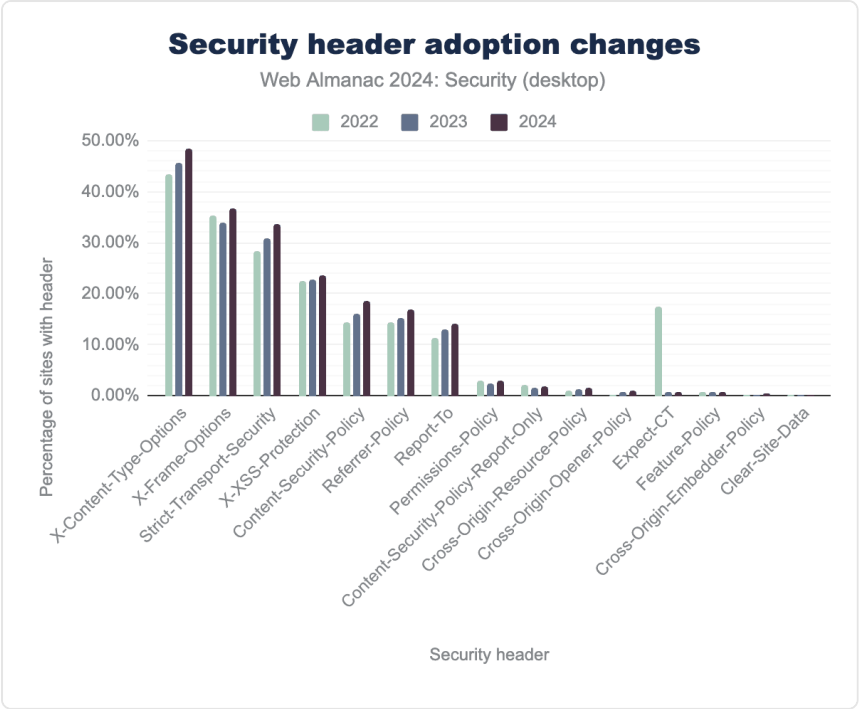


Figure 11.31. Adoption of security headers for site requests in desktop pages.

Over the past two years, three security headers have seen a decrease in usage. The most notable decline is in the `Expect-CT` header, which was used to opt into Certificate Transparency⁴³⁴. This header is now deprecated because Certificate Transparency is enabled by default. Similarly, the `Feature-Policy` header has decreased in usage due to its replacement by the `Permissions-Policy` header. Lastly, the `Content-Security-Policy-Report-Only` header has also declined. This header was used primarily for testing and monitoring the impact of a Content Security Policy by sending violation reports to a specified endpoint. It's important to note that the `Report-Only` header does not enforce the Content Security Policy itself, so its decline in usage does not indicate a reduction in security. Since none of these headers impact security, we can safely assume that the overall adoption of security headers continues to grow, reflecting a positive trend in web security.

The strongest absolute risers since 2022 are `Strict-Transport-Security` (+5.3%), `X-Content-Type-Options` (+4.9%) and `Content-Security-Policy` (+4.2%).

434. https://developer.mozilla.org/docs/Web/Security/Certificate_Transparency

Preventing clickjacking with CSP and X-Frame-Options

As discussed previously, one of the primary uses of the Content Security Policy is to prevent clickjacking attacks. This is achieved through the `frame-ancestors` directive, which allows websites to specify which origins are permitted to embed their pages within a frame. There, we saw that this directive is commonly used to either completely prohibit embedding or restrict it to the same origin (Figure 17).

Another measure against clickjacking is the `X-Frame-Options` (XFO) header, though it provides less granular control compared to CSP. The XFO header can be set to `SAMEORIGIN`, allowing the page to be embedded only by other pages from the same origin, or `DENY`, which completely blocks any embedding of the page. As shown in the table below, most headers are configured to relax the policy by allowing same-origin websites to embed the page.

Header	Desktop	Mobile
SAMEORIGIN	73%	73%
DENY	23%	24%

Figure 11.32. `X-Frame-Options` header values.

Although deprecated, 0.6% of observed `X-Frame-Options` headers on desktop and 0.7% on mobile still use the `ALLOW-FROM` directive, which functions similarly to the `frame-ancestors` directive by specifying trusted origins that can embed the page. However, since modern browsers ignore `X-Frame-Options` headers containing the `ALLOW-FROM` directive, this could create gaps in the website's clickjacking defenses. However, this practice may be intended for backward compatibility, where the deprecated header is used alongside a supported Content Security Policy that includes the `frame-ancestors` directive.

Preventing attacks using Cross-Origin policies

One of the core principles of the Web is the reuse and embedding of cross-origin resources. However, our security perspective on this practice has significantly shifted with the emergence of micro-architectural attacks like Spectre and Meltdown, and Cross-Site Leaks (XS-Leaks⁴³⁵) that leverage side-channels to uncover potentially sensitive user information. These threats have created a growing need for mechanisms to control whether and how resources can be rendered by other websites, whilst ensuring better protection against these new exploits.

This demand led to the introduction of several new security headers collectively known as

435. <https://xsleaks.dev/>

Cross-Origin policies: `Cross-Origin-Resource-Policy` (CORP), `Cross-Origin-Embedder-Policy` (COEP) and `Cross-Origin-Opener-Policy` (COOP). These headers provide robust countermeasures against side-channel attacks by controlling how resources are shared and embedded across origins. Adoption of these policies has been steadily increasing, with the use of Cross-Origin-Opener-Policy nearly doubling each year for the past two years.

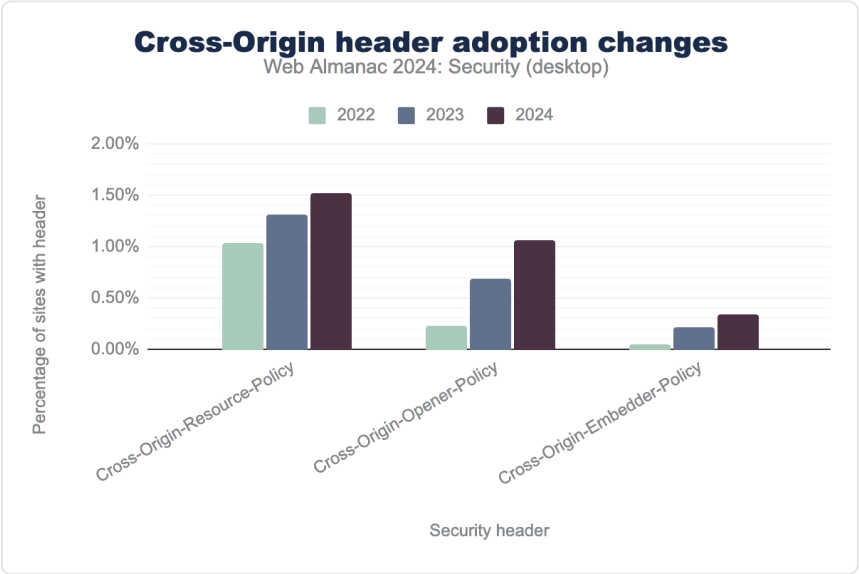


Figure 11.33. Usage of Cross-Origin headers in 2022, 2023 and 2024.

Cross Origin Embedder Policy

The Cross Origin Embedder Policy⁴³⁶ restricts the capabilities of websites that embed cross-origin resources. Currently, websites no longer have access to powerful features like `SharedArrayBuffer` and unthrottled timers through the `Performance.now()` API, as these can be exploited to infer sensitive information from cross-origin resources. If a website requires access to these features, it must signal to the browser that it intends to interact only with cross-site resources via credentialless requests (`credentialless`) or with resources that explicitly permit access from other origins using the `Cross-Origin-Resource-Policy` header (`require-corp`).

436. <https://developer.mozilla.org/docs/Web/HTTP/Headers/Cross-Origin-Embedder-Policy>

COEP value	Desktop	Mobile
unsafe-none	86%	88%
require-corp	7%	5%
credentialless	2%	2%

Figure 11.34. Prevalence of COEP header values.

The majority of websites that set the `Cross-Origin-Embedder-Policy` header indicate that they do not require access to the powerful features mentioned above (`unsafe-none`). This behavior is also the default if the COEP header is absent, meaning that websites will automatically operate under restricted access to cross-origin resources unless explicitly configured otherwise.

Cross Origin Resource Policy

Conversely, websites that serve resources can use the `Cross-Origin-Resource-Policy` response header to grant explicit permission for other websites to render the served resource. This header can take one of three values: `same-site`, allowing only requests from the same site to receive the resource; `same-origin`, restricting access to requests from the same origin; and `cross-origin`, permitting any origin to access the resource. Beyond mitigating side-channel attacks, CORP can also protect against Cross-Site Script Inclusion (XSSI). For instance, by disallowing a dynamic JavaScript resource from being served to cross-origin websites, CORP helps prevent the leaking of scripts with sensitive info.

CORP value	Desktop	Mobile
cross-origin	91%	91%
same-origin	5%	5%
same-site	4%	4%

Figure 11.35. Prevalence of CORP header values.

The CORP header is primarily used to allow access to the served resource from any origin, with the `cross-origin` value being the most commonly set. In fewer cases, the header restricts access: less than 5% of websites limit resources to the same origin, and less than 4% restrict them to the same site.

Cross Origin Opener Policy

Cross Origin Opener Policy⁴³⁷ (COOP) helps control how other web pages can open and reference the protected page. COOP protection can be explicitly disabled with `unsafe-none`, which is also the default behavior in absence of the header. The `same-origin` value allows references from pages with the same origin and `same-origin-allow-popups` additionally allows references with windows or tabs. Similar to the Cross Origin Embedder Policy, features like the `SharedArrayBuffer` and `Performance.now()` are restricted unless COOP is configured as `same-origin`.

COOP value	Desktop	Mobile
same-origin	49%	48%
unsafe-none	35%	37%
same-origin-allow-popups	14%	14%

Figure 11.36. Prevalence of COOP header values.

Nearly half of all observed COOP headers employ the strictest setting, `same-origin`.

Preventing attacks using Clear-Site-Data

The `Clear-Site-Data` header allows websites to easily clear browsing data associated with them, including cookies, storage, and cache. This is particularly useful as a security measure when a user logs out, ensuring that authentication tokens and other sensitive information are removed and cannot be abused. The header's value specifies what types of data the website requests the browser to clear.

Adoption of the `Clear-Site-Data` header remains limited; our observations indicate that only 2,071 hosts (0.02% of all hosts) use this header. However, this functionality is primarily useful on logout pages, which the crawler does not capture. To investigate logout pages, the crawler would need to be extended to detect and interact with account registration, login, and logout functionality – an undertaking that would require quite some effort. Some progress has already been made in this area by security and privacy researchers, such as automating logins to web pages⁴³⁸, and automating registering⁴³⁹.

437. <https://developer.mozilla.org/docs/Web/HTTP/Headers/Cross-Origin-Opener-Policy>

438. <https://www.ndss-symposium.org/wp-content/uploads/2020/02/23008-paper.pdf>

439. <https://dl.acm.org/doi/pdf/10.1145/3589334.3645709>

Clear site data value	Desktop	Mobile
"cache"	36%	34%
cache	22%	23%
*	12%	13%
cookies	4%	6%
"cache", "storage", "executionContexts"	3%	4%
"cookies"	2%	2%
"cache", "cookies", "storage", "executionContexts"	2%	2%
"storage"	2%	2%
"cache", "storage"	1%	1%
cache, cookies, storage	1%	1%

Figure 11.37. Prevalence of `Clear-Site-Data` headers.

Current usage data shows that the `Clear-Site-Data` header is predominantly used to clear cache. It’s important to note that the values in this header must be enclosed in quotation marks; for instance, `cache` is incorrect and should be written as `"cache"`. Interestingly, there has been significant improvement in adherence to this syntax rule: in 2022, 65% of desktop and 63% of mobile websites were found using the incorrect `cache` value. However, these numbers have now dropped to 22% and 23% for desktop and mobile, respectively.

Preventing attacks using `<meta>`

Some security mechanisms on the web can be configured through `meta` tags in the source HTML of a web page, for instance the `Content-Security-Policy` and `Referrer-Policy`. This year, 0.61% and 2.53% of mobile websites enable CSP and Referrer-Policy respectively using `meta` tags. This year we find that there is a slight increase in the use of this method for setting the Referrer-Policy yet a slight decrease for setting CSP.

Meta tag	Desktop	Mobile
includes Referrer-policy	2.7%	2.5%
includes CSP	0.6%	0.6%
includes not-allowed policy	0.1%	0.1%

Figure 11.38. The percentage of hosts enabling different policies using a meta tag.

Developers sometimes also try to enable other security features by using the `meta` tag, which is not allowed and will thus be ignored. Using the same example as in 2022, 4976 pages try to set the `X-Frame-Options` using a `meta` tag, which will be ignored by the browser. This is an absolute increase compared to 2022, but only because there were more than twice as many pages included in the data set. Relatively, there is a slight decrease from 0.04% to 0.03% on mobile pages and 0.05% to 0.03% on desktop pages.

Web Cryptography API

Web Cryptography API⁴⁴⁰ is a JavaScript API for performing basic cryptographic operations on a website such as random number generation, hashing, signature generation and verification, and encryption and decryption.

440. <https://www.w3.org/TR/WebCryptoAPI/>

Feature	Desktop	Mobile
CryptoGetRandomValues	56.9%	53.2%
SubtleCryptoDigest	1.9%	1.7%
SubtleCryptoImportKey	1.7%	1.6%
CryptoAlgorithmSha256	1.6%	1.3%
CryptoAlgorithmEcdh	1.3%	1.3%
CryptoAlgorithmSha512	0.2%	0.2%
CryptoAlgorithmAesCbc	0.2%	0.1%
CryptoAlgorithmSha1	0.2%	0.2%
SubtleCryptoEncrypt	0.2%	0.1%
SubtleCryptoSign	0.1%	0.1%

Figure 11.39. The usages of features of the Web Cryptography API.

In comparison to the last Almanac, the `CryptoGetRandomValues` continued to drop and it did so at a much higher rate in the past two years, dropping down to 53%. Despite that drop, it clearly continues to be the most adopted feature, far ahead of the other features. After `CryptoGeRandomValues`, the next five most used features have become more widely adopted, rising from under 0.7% to adoption rates between 1.3% and 2%.

Bot protection services

Because bad bots remain a significant issue on the modern web, we see that the adoption of protections against bots has continued to rise. We observe another jump in adoption from 29% of desktop sites and 26% of mobile sites in 2022 to 33% and 32% respectively now. It seems that developers have invested in protecting more mobile websites, bringing the number of protected desktop and mobile sites closer together.

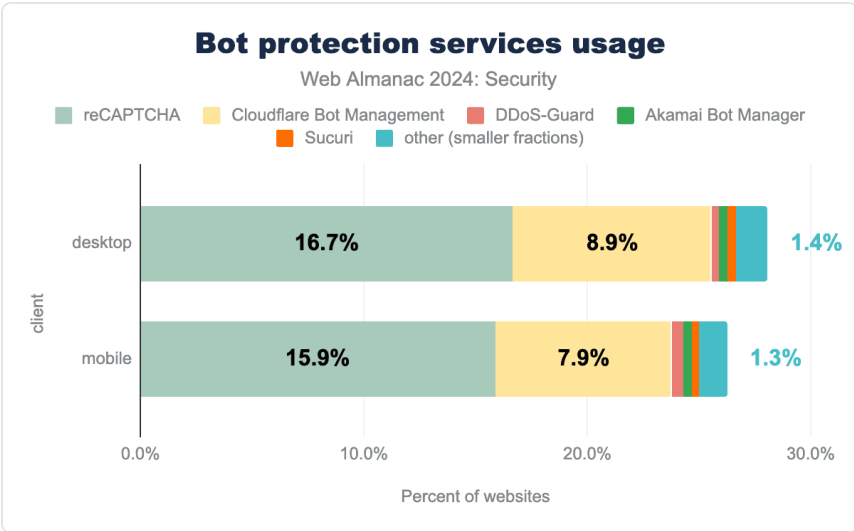


Figure 11.40. The distribution of bot protection services in use.

reCAPTCHA remains the largest protection mechanism in use, but has seen a reduction in its use. In comparison, Cloudflare Bot Management has seen an increase in adoption and remains the second largest protection in use.

HTML sanitization

A new addition to major browsers are the `setHTMLUnsafe` and `ParseHTMLUnsafe` APIs, that allow a developer to use a declarative shadow DOM from JavaScript⁴⁴¹. When a developer uses custom HTML components from JavaScript that include a definition for a declarative shadow DOM using `<template shadowrootmode="open">...</template>`, using `innerHTML` to place this component on the page will not work as expected. This can be prevented by using the alternative `setHTMLUnsafe` that makes sure the declarative shadow DOM is taken into account.

When using these APIs, developers must be careful to only pass already safe values to these APIs because as the names imply they are unsafe, meaning they will not sanitize input given, which may lead to XSS attacks.

441. <https://developer.chrome.com/blog/new-in-chrome-124#dsd>

Feature	Desktop	Mobile
ParseHTMLUnsafe	6	6
SetHTMLUnsafe	2	2

Figure 11.41. The number of pages using HTML sanitization APIs.

These APIs are new, so low adoption is to be expected. We found only 6 pages in total using `parseHTMLUnsafe` and 2 using `setHTMLUnsafe`, which is an extremely small number relative to the number of pages visited.

Drivers of security mechanism adoption

Web developers can have many reasons to adopt more security practices. The three primary ones are:

- Societal: in certain countries there is more security-oriented education, or there may be local laws that take more punitive measures in case of a data breach or other cybersecurity-related incident
- Technological: depending on the technology stack in use, it might be easier to adopt security features. Some features might not be supported and would require additional effort to implement. Adding to that, certain vendors of software might enable security features by default in their products or hosted solutions
- Popularity: widely popular websites may face more targeted attacks than a website that is less known, but may also attract more security researchers or white hat hackers to look at their products, helping the site implement more security features correctly

Location of website

The location where a website is hosted or its developers are based can often have impacts on adoption of security features. The security awareness among developers will play a role, as they cannot implement features that they aren't aware of. Additionally, local laws can sometimes mandate the adoption of certain security practices.

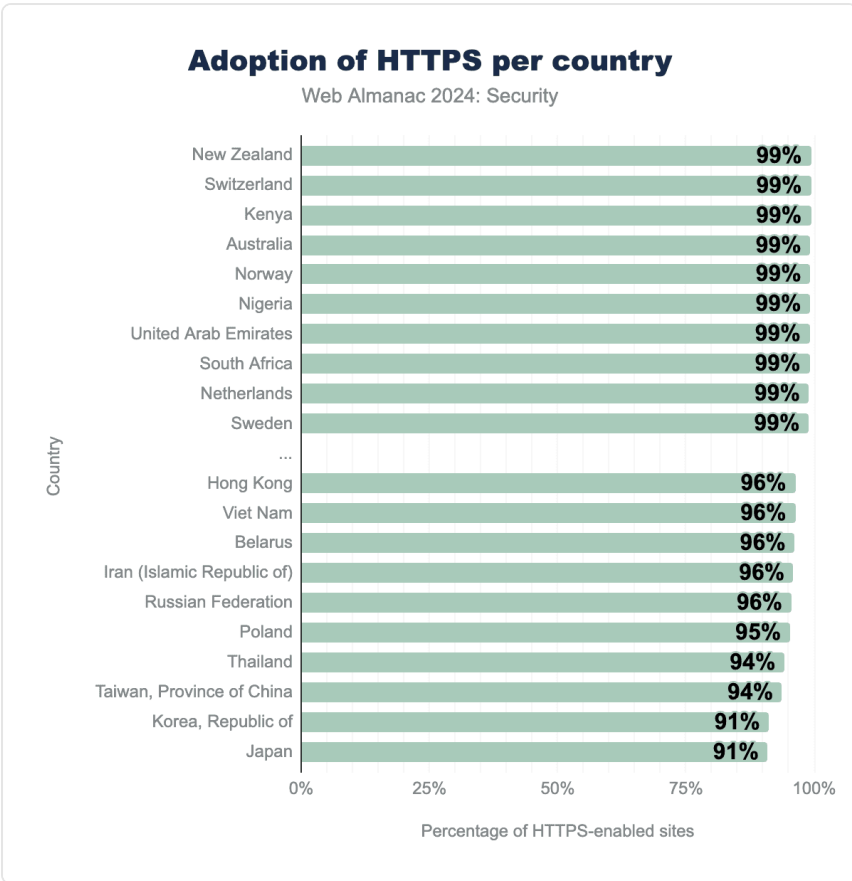


Figure 11.42. The adoption of HTTPS per country; top and bottom 10 countries.

New Zealand continues to lead in the adoption of HTTPS websites, however, many countries are following the adoption extremely closely as the top 9 countries all reach adoption of over 99%! Also the trailing 10 countries have all seen a rise in HTTPS adoption by 9% to 10%, with all countries now reaching adoption above 90%! This shows that almost all countries continue their efforts in making HTTPS the default mode.

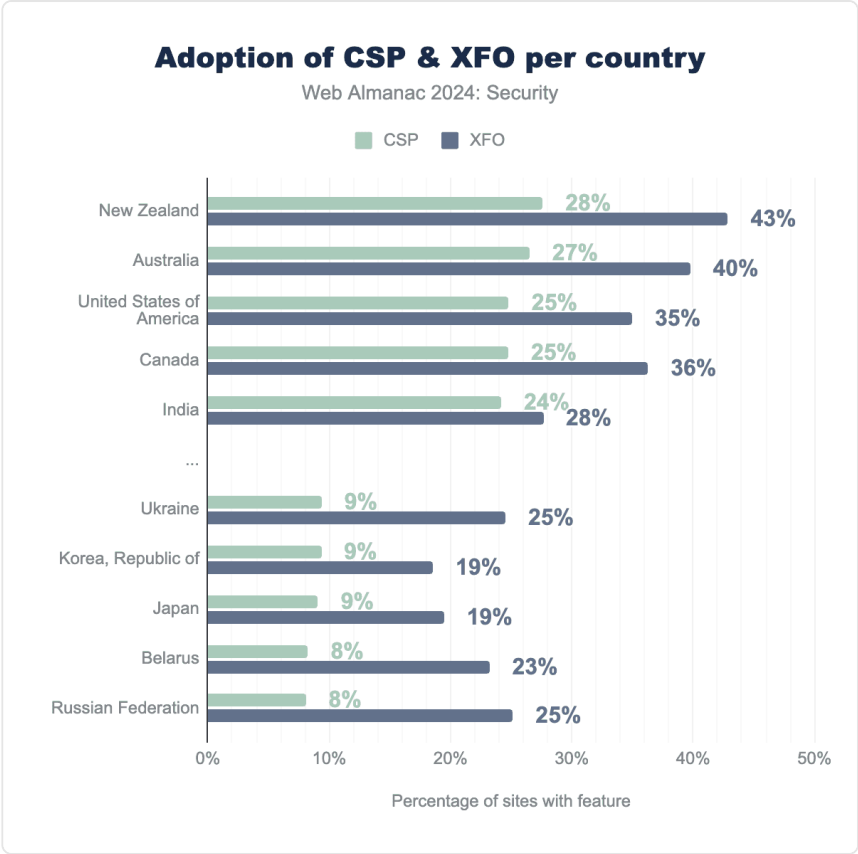


Figure 11.43. The adoption of CSP and XFO per country; top and bottom 5 countries.

We see that the top 5 countries in terms of CSP adoption have CSP enabled on almost a quarter of their websites. The trailing countries have also seen an increase in the use of CSP, albeit a more moderate one. In general the adoption of both XFO and CSP remains very varied among countries, and the gap between CSP and XFO remains equally large if not larger compared to 2022, reaching up to 15%.

Technology stack

Many sites on the current web are made using large CMS systems. These may enable security features by default to protect their users.

Technology	Security features
Wix	Strict-Transport-Security (99.9%), X-Content-Type-Options (99.9%)
Blogger	X-Content-Type-Options (99.8%), X-XSS-Protection (99.8%)
Squarespace	Strict-Transport-Security (98.9%), X-Content-Type-Options (99.1%)
Drupal	X-Content-Type-Options (90.3%), X-Frame-Options (87.9%)
Google Sites	Content-Security-Policy (99.9%), Cross-Origin-Opener-Policy (99.8%), Cross-Origin-Resource-Policy (99.8%), Referrer-Policy (99.8%), X-Content-Type-Options (99.9%), X-Frame-Options (99.9%), X-XSS-Protection (99.9%)
Medium	Content-Security-Policy (99.2%), Strict-Transport-Security (96.4%), X-Content-Type-Options (99.1%)
Substack	Strict-Transport-Security (100%), X-Frame-Options (100%)
Wagtail	Referrer-Policy (55.2%), X-Content-Type-Options (61.7%), X-Frame-Options (72.1%)
Plone	Strict-Transport-Security (57.1%), X-Frame-Options (75.2%)

Figure 11.44. Security features in use by selected CMS systems.

It's clear that many major CMS's that are hosted by the providing company and where only content is created by users, such as Wix, SquareSpace, Google Sites, Medium and Substack, roll out security protections widely, showing adoption of HSTS, X-Content-Type-Options or X-XSS-Protection in the upper 99% adoption rates. Google sites continues to be the CMS that has the highest number of security features in place.

For CMS's that can be easily self-hosted such as Plone or Wagtail, it is more difficult to control

the rollout of features because the CMS creators have no way to influence the update behavior of users. Websites hosted using these CMS's could be online without change in security features for a long time.

Website popularity

Large websites often have a high number of visitors and registered users, of which they might store highly sensitive data. This means they likely attract more attackers and are thus more prone to targeted attacks. Additionally, when an attack succeeds, these websites could be fined or sued, costing them money and/or reputational damage. Therefore, it can be expected that popular websites invest more in their security to secure their users.

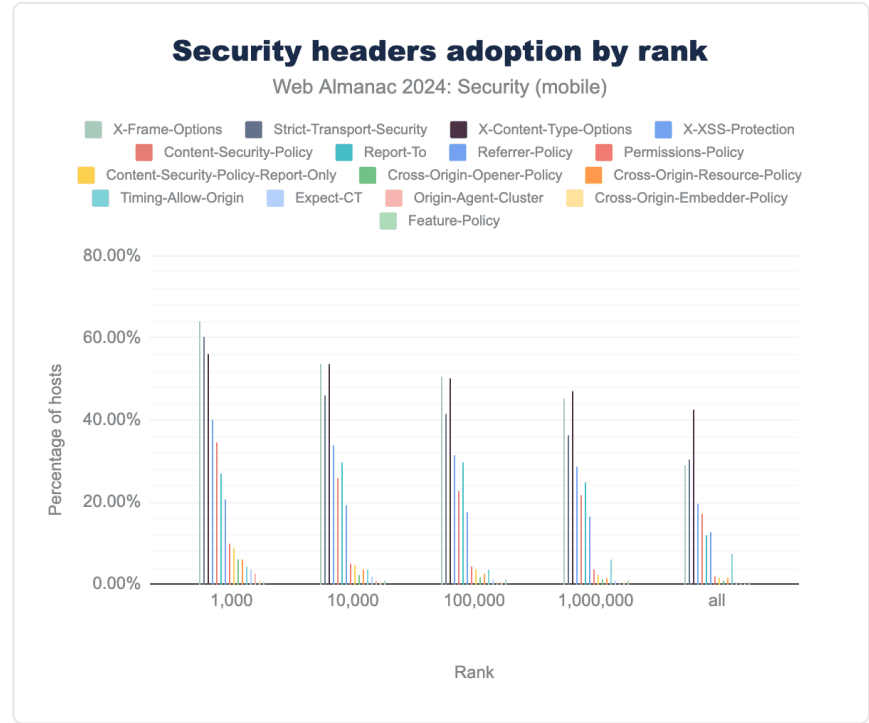


Figure 11.45. Security header adoption by website rank according to the April 2024 CrUX.

We find that most headers, including the most popular ones: X-Frame-Options, Strict-Transport-Security, X-Content-Type-Options, X-XSS-Protection and Content-Security-Policy, always have higher adoptions for more popular sites on mobile. 64.3% of the top 1000 sites on mobile have HSTS enabled. This means the top 1000 websites are more invested in only sending traffic over HTTPS. Less popular sites can still have HTTPS enabled,

but don't add a `Strict-Transport-Security` header as often, which may lead users to repeatedly visit the site over plain HTTP.

Website category

In some industries, developers might keep more up to date with security features they may be able to use to better secure their sites.

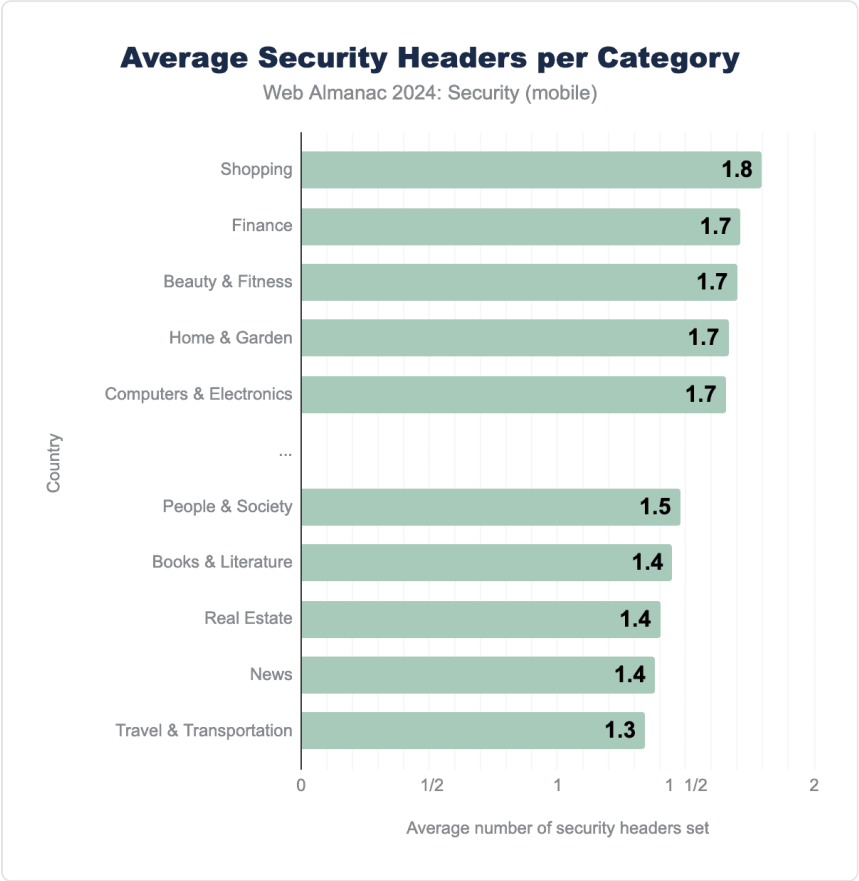


Figure 11.46. The average number of security headers by website category; top and bottom 5 categories.

We find that there is a subtle difference in the average number of security headers used depending on the categorization of the website. This number does not directly show the overall security of these sites, but might give an insight into which categories of industry are inclined to

implement more security features. We see that shopping and finance lead the list, both industries that deal with sensitive information and high amounts of monetary transactions, which may be reasons to invest in security. At the bottom of the list we see news and travel & transportation. Both are categories in which a lot of sites will host content relating to their respective topics, but may not handle much sensitive data compared to sites in the top categories on the list. In general, this trend seems to be weak.

Malpractices on the Web

Although cryptocurrencies remain popular, the number of cryptominers on the web has continued to decrease over the past two years, with no notable spikes in usage anymore as was described in the 2022 edition of the Web Almanac.

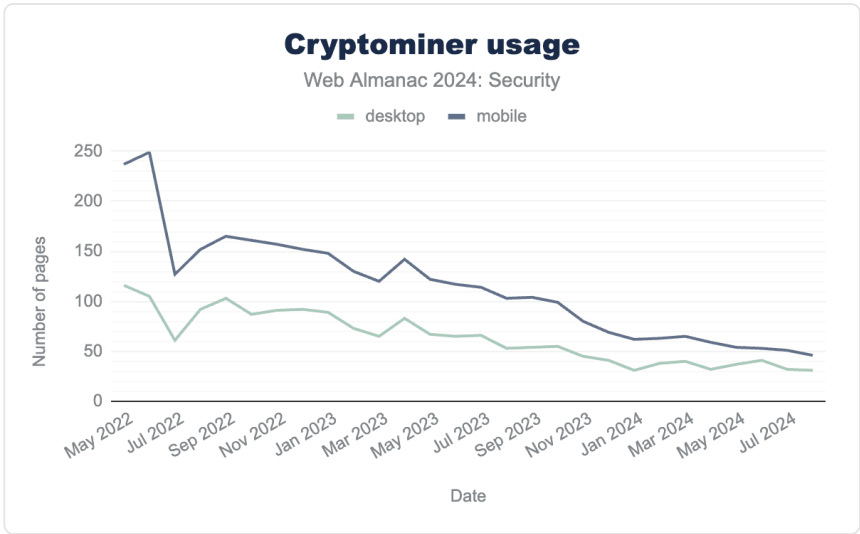


Figure 11.47. The number of cryptominers in use over time; from May 2022 to Jul 2024.

When looking at the cryptominer share, we see that part of the Coinimp share has been overtaken by JSEcoin, while other miners have remained relatively stable, seeing only minor changes. With the low number of cryptominers found on the web, these relative changes are still quite minor.

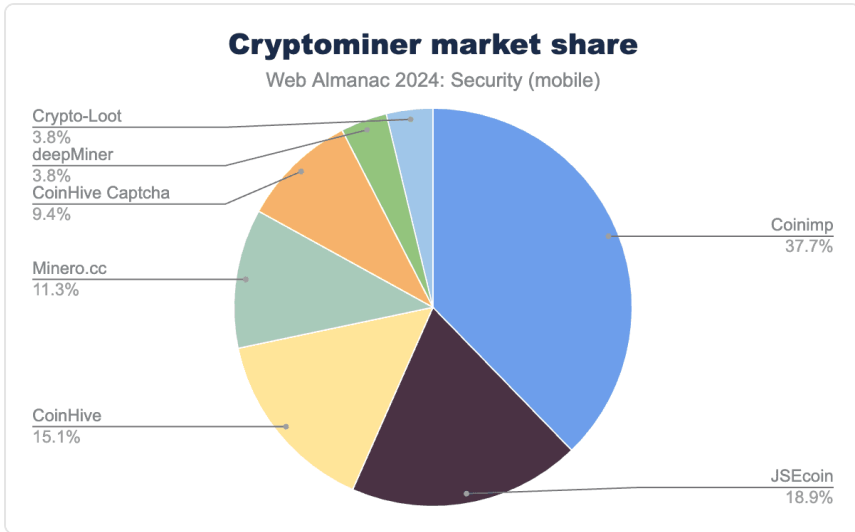


Figure 11.48. The cryptominer market shares.

One should note that the results shown here may be a underrepresentation of the actual state of the websites infected with cryptominers. Since our crawler is run once a month, not all websites that run a cryptominer can be discovered. For example, if a website is only infected for several days, it might not be detected.

Security misconfigurations and oversights

While the presence of security policies suggests that website administrators are actively working to secure their sites, proper configuration of these policies is crucial. In the following section, we will highlight some observed misconfigurations that could compromise security.

Unsupported policies defined by `<meta>`

It's crucial for developers to understand where specific security policies should be defined. For instance, while a secure policy might be defined through a `<meta>` tag, it could be ignored by the browser if it's not supported there, potentially leaving the application vulnerable to attacks.

Although the Content Security Policy can be defined using a `<meta>` tag, its `frame-ancestors` and `sandbox` directives are not supported in this context. Despite this, our observations show that 1.70% of pages that use CSP in a `<meta>` tag on desktop and 1.26% on mobile incorrectly used the `frame-ancestors` directive in the `<meta>` tag. This is far lower

for the disallowed `sandbox` directive, which was defined for less than 0.01%.

COEP, CORP and COOP confusion

Due to their similar naming and purpose, the COEP, CORP and COOP are sometimes difficult to discern. However, assigning unsupported values to these headers can have a detrimental effect on the website's security.

Invalid COEP value	Desktop	Mobile
same-origin	3.22%	3.05%
cross-origin	0.30%	0.23%
same-site	0.06%	0.04%

Figure 11.49. Prevalence of invalid COEP header values.

For instance, around 3% of observed COEP headers mistakenly use the unsupported value `same-origin`. When this occurs, browsers revert to the default behavior of allowing any cross-origin resource to be embedded, while restricting access to features like `SharedArrayBuffer` and unthrottled use of `Performance.now()`. This fallback does not inherently reduce security unless the site administrator intended to set `same-origin` for CORP or COOP, where it is a valid value.

Additionally, only 0.26% of observed COOP headers were set to `cross-origin` and just 0.02% of CORP headers used the value `unsafe-none`. Even if these values were mistakenly applied to the wrong headers, they represent the most permissive policies available. Therefore, these misconfigurations are not considered to decrease security.

In addition to cases where valid values intended for one header were mistakenly used for another, we identified several minor instances of syntactical errors across various headers. However, each of these errors accounted for less than 1% of the total observed headers, suggesting that while such mistakes exist, they are relatively infrequent.

Timing-Allow-Origin wildcards

Timing-Allow-Origin is a response header that allows a server to specify a list of origins that are allowed to see values of attributes obtained through features of the Resource Timing API⁴⁴². This means that an origin listed in this header can access detailed timestamps regarding the

442. https://developer.mozilla.org/docs/Web/API/Performance_API/Resource_timing

connection that is being made to the server, such as the time at the start of the TCP connection, start of the request and start of the response.

When CORS is in effect, many of these timings (including the ones listed above) are returned as 0 to prevent cross-origin leaks. By listing an origin in the Timing-Allow-Origin header this restriction is lifted.

Allowing different origins access to this information should be done with care, because using this information the site loading the resource can potentially execute timing attacks. In our analysis we find that out of all responses with a Timing-Allow-Origin header present, 83% percent of `Timing-Allow-Origin` headers contain the wildcard value, thereby allowing any origin to access the fine grained timing information.



Figure 11.50. The percentage of `Timing-Allow-Origin` that are set to the wildcard (`*`) value.

Missing suppression of server information headers

While *security by obscurity* is generally considered bad practice, web applications can still benefit from withholding excessive information about the server or framework in use. Although attackers can still fingerprint certain details, minimizing exposure - particularly regarding specific version numbers - can reduce the likelihood of the application being targeted in automated vulnerability scans.

This information is usually reported in headers such as `Server`, `X-Server`, `X-Backend-Server`, `X-Powered-By`, `X-AspNet-Version`.

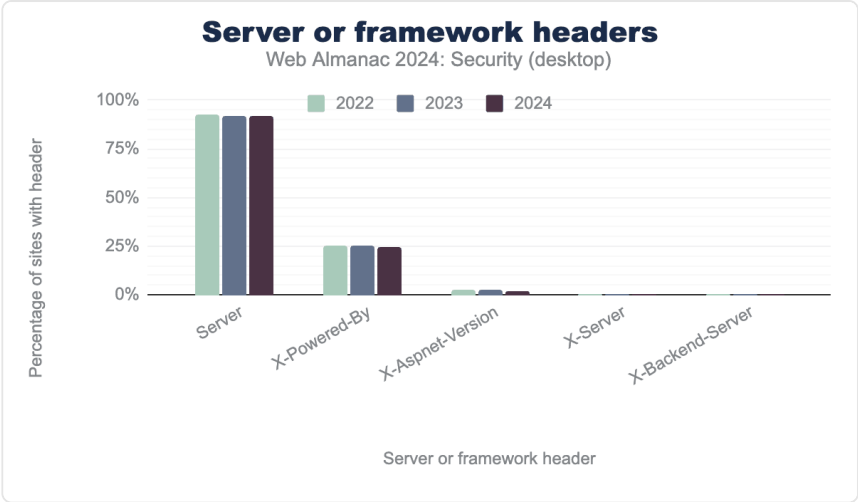


Figure 11.51. Prevalence of headers used to convey information about the server.

The most commonly exposed header is the `Server` header, which reveals the software running on the server. This is followed by the `X-Powered-By` header, which discloses the technologies used by the server.

Header value	Desktop	Mobile
PHP/7.4.33	9.1%	9.4%
PHP/7.3.33	4.6%	5.4%
PHP/5.3.3	2.6%	2.8%
PHP/5.6.40	2.5%	2.6%
PHP/7.4.29	1.7%	2.2%
PHP/7.2.34	1.7%	1.8%
PHP/8.0.30	1.3%	1.4%
PHP/8.1.28	1.1%	1.1%
PHP/8.1.27	1.0%	1.1%
PHP/7.1.33	1.0%	1.0%

Figure 11.52. Most prevalent `X-Powered-By` header values with specific framework version.

Examining the most common values for the `Server` and `X-Powered-By` headers, we found that especially the `X-Powered-By` header specifies versions, with the top 10 values revealing specific PHP versions. For both desktop and mobile, at least 25% of `X-Powered-By` headers contain this information. This header is likely enabled by default on the observed web servers. While it can be useful for analytics, the header's benefits are limited, and thus it warrants to be disabled by default. However, disabling this header alone does not address the security risks of outdated servers; regularly updating the server remains crucial.

Missing suppression of `Server-Timing` header

The `Server-Timing` header is defined in a W3C Editor's Draft⁴⁴³ as a header that can be used to communicate about server performance metrics. A developer can send metrics containing zero or more properties. One of the specified properties is the `dur` property, that can be used to communicate millisecond-accurate timings that contain the duration of a specific action on the server.

We find that `server-timing` is used by 6.4% of internet hosts. Over 60% of those hosts include at least one `dur` property in their response and over 55% even send more than two. This means that these sites are exposing server-side process durations directly to a client, which can be used for exploiting. Because the `server-timing` may contain sensitive information, the use is now restricted to the same origin, except when `Timing-Allow-Origin` is used by the developer as discussed in the previous section. However, timing attacks can still be exploited directly against servers without the need to access cross-origin data.

443. <https://w3c.github.io/server-timing/>

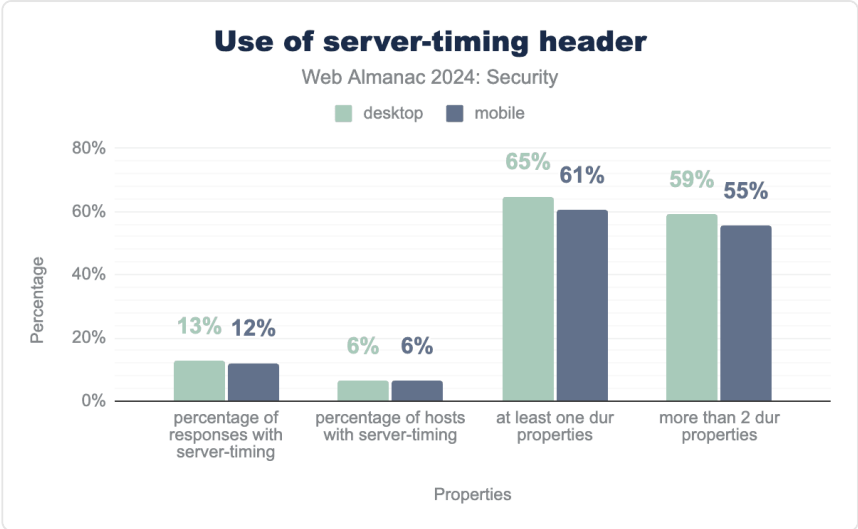


Figure 11.53. The usage of the server-timing header and relative usage of `dur` properties.

`.well-known` URIs

`.well-known` URIs⁴⁴⁴ are used as a way to designate specific locations to data or services related to the overall website. A well-known URI is a URI whose path component begins with the characters `/ .well-known/`.

`security.txt`

`security.txt` is a file format that can be used by websites to communicate information regarding vulnerability reporting in a standard way. Website developers can provide contact details, PGP key, policy, and other information in this file. White hat hackers and penetration testers can then use this information to report potential vulnerabilities they find during their security analyses. Our analysis shows that 1% of websites currently use a `security.txt` file, showing that they are actively working on improving their site's security.

444. <https://datatracker.ietf.org/doc/html/rfc8615>

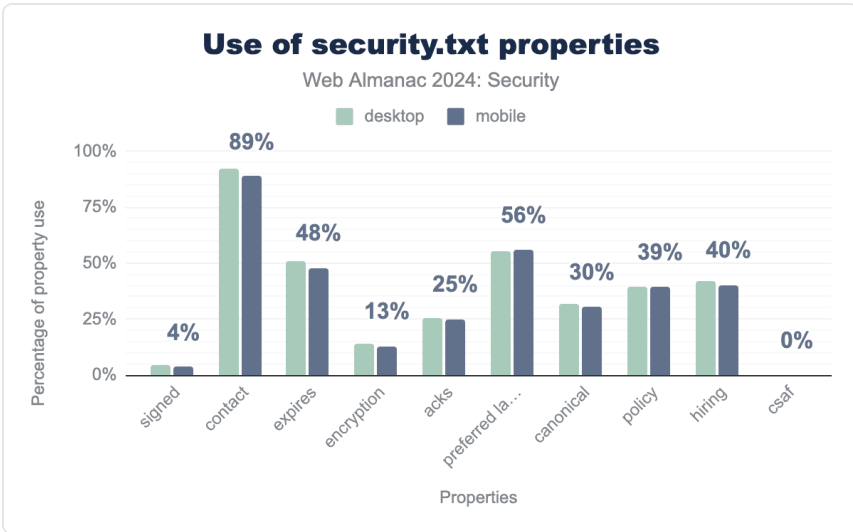


Figure 11.54. The usage of security.txt properties.

Most of the security.txt files include contact information (88.8%) and a preferred language (56.0%). This year, 47.9% of security.txt files define an expiry, which is a giant jump compared to the 2022 2.3%. This can largely be explained by an update to the methodology, as the analysis only includes text files this year instead of simply all responses with code 200, thereby significantly lowering the false positive rate. It does mean that less than half of the sites that use security.txt are following the standard that (among other requirements) defines the expires property as required. Interestingly, only 39% of the security.txt files define a policy, which is the space developers can indicate what steps a white hat hacker that found a vulnerability should take to report the vulnerability.

change-password

The `change-password` well-known URI is a W3C specification in the editor's draft state, which is the same state it was in in 2022. This specific well-known URI was suggested as a way for users and softwares to easily identify the link to be used for changing passwords, which means external resources can easily link to that page.

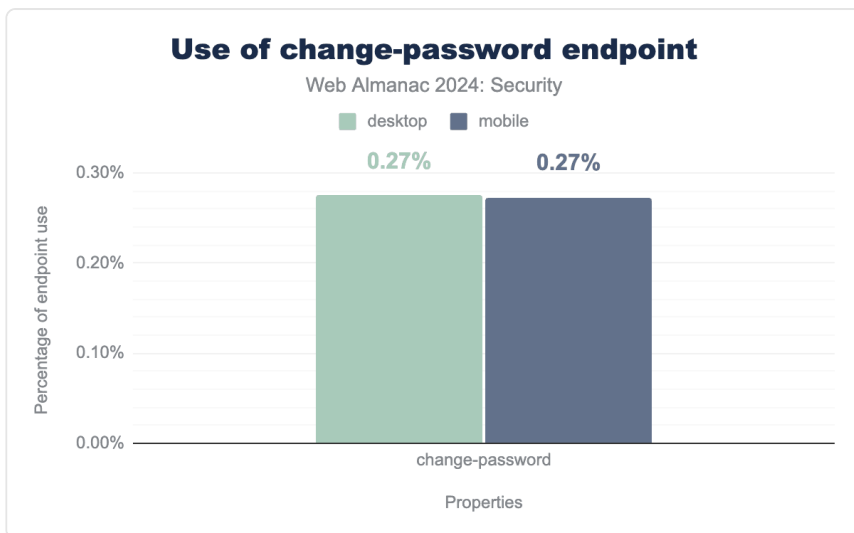


Figure 11.55. The usages of the `change-password` well-known endpoint.

The adoption remains very low. At 0.27% for both mobile and desktop sites it slightly decreased for desktop sites from 0.28% in 2022. Due to the slow standardization process it is not unexpected that the adoption does not change much. We also repeat that websites without authentication mechanisms have no use for this url, which means it would be useless for them to implement it.

Detecting status code reliability

In a specification that is also still an editor's draft⁴⁴⁵, like in 2022, a particular well-known URI is defined to determine the reliability of a website's HTTP response status code. The idea behind this well-known URI is that it should never exist in any website, which in turn means navigating to this well-known URI should never result in a response with an `ok-status`. If it redirects and returns an "ok-status", that means the website's status codes are not reliable. This could be the case when a redirect to a specific '404 not found' error page occurs, but that page is served with an ok status.

445. <https://w3c.github.io/webappsec-change-password-url/response-code-reliability.html>

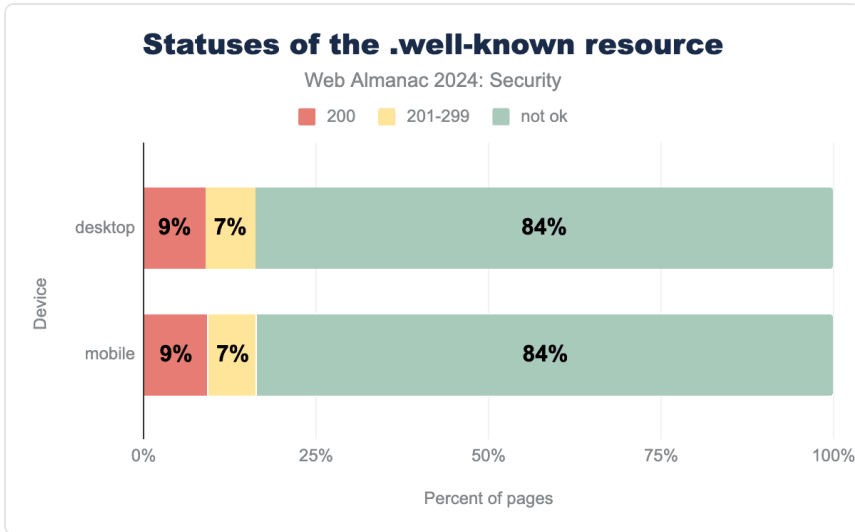


Figure 11.56. The distribution of statuses returned for the `.well-known` endpoint to assess status code reliability.

We find a similar distribution as in 2022, where 83.6% of pages respond with a not-ok status, which is the expected outcome. Again, one reason that these figures may not change much is the fact that the standard is stuck in the editor's draft status and the standardization is slow.

Sensitive endpoints in `robots.txt`

Finally, we check whether or not `robots.txt` includes possibly sensitive endpoints. By using this information, hackers may be able to select websites or endpoints to target based on the exclusion in `robots.txt`.

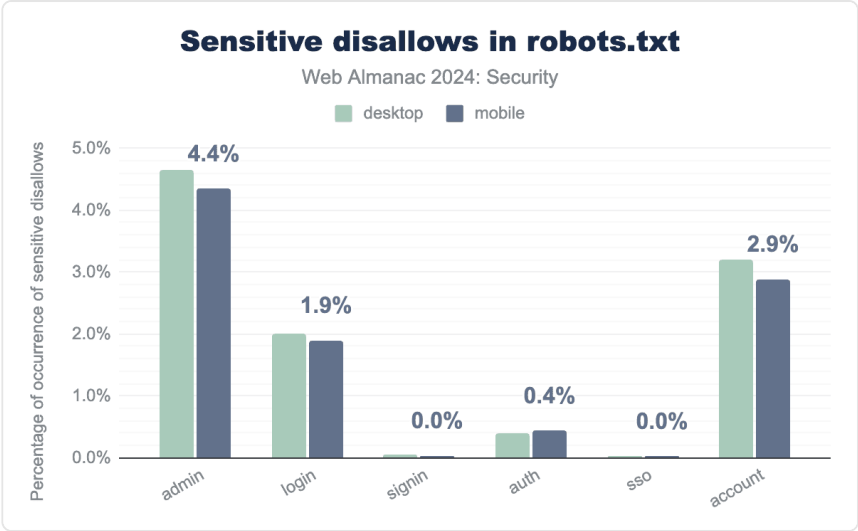


Figure 11.57. The percentage of sites including specified endpoints in their robots.txt.

We see that around 4.3% of websites include at least one `admin` entry in their `robots.txt` file.

This may be used to find an admin-only section of the website, which would otherwise be hidden and finding it would rely on attempting to visit specific subpages under that url. `login`, `signin`, `auth`, `sso` and `account` point to the existence of a mechanism where users can log in using an account they created or received. Each of these endpoints are included in the robots.txt of a number of sites (some of which may be overlapping), with `account` being the more popular one at 2.9% of websites.

Indirect resellers in `ads.txt`

The `ads.txt` file is a standardized format that allows websites to specify which companies are authorized to sell or resell their digital ad space within the complex landscape of programmatic advertising. Companies can be listed as either direct sellers or indirect resellers. Indirect resellers, however, can leave publishers - sites hosting the ads.txt file - more vulnerable to ad fraud because they offer less control over who purchases ad space. This vulnerability was exploited in 2019 by the so-called 404bot scam⁴⁴⁶, resulting in millions of dollars in lost revenue.

446. <https://www.fraud0.com/resources/ads-txt/>

77%

Figure 11.58. The percentage of desktop ad publishers that entirely avoid indirect resellers.

By refraining from listing indirect sellers, website owners help prevent unauthorized reselling and reduce ad fraud, thereby enhancing the security and integrity of their ad transactions. Among publishers that host an ads.txt file, 77% for desktop and 42.4% for mobile avoid resellers entirely, curbing potential fraud.

Conclusion

Overall, this year's analysis highlights promising trends in web security. HTTPS adoption is nearing 100%, with Let's Encrypt leading the charge by issuing over half of all certificates, making secure connections more accessible. Although the overall adoption of security policies remains limited, it's encouraging to see steady progress with key security headers. Secure-by-default measures, like the `SameSite=Lax` attribute for cookies, are driving website administrators to at least consider important security practices.

However, attention must also be given to poor configurations or even misconfigurations that can weaken these protections. Issues like invalid directives or poorly defined policies can prevent browsers from enforcing security effectively. For instance, 82.5% of all `Timing-Allow-Origin` headers allow any origin to access detailed timing information, which could be abused in timing attacks. Similarly, only 1% of websites enable security issue reporting via `security.txt`, and many still expose their PHP version, an unnecessary risk that can reveal potential vulnerabilities. On the bright side, most of these issues represent low-hanging fruit—addressing them typically requires minimal changes to website implementations.

As the number of security policies grows, it's essential for policymakers to focus on reducing complexity. Reducing implementation friction will make adoption easier and minimize common mistakes. For example, the introduction of cross-origin headers designed to prevent cross-site leaks and microarchitectural attacks has already caused confusion, with directives from one policy mistakenly applied to another.

Although new attacks will undoubtedly emerge in the future, demanding new protections, the openness of the security community plays a crucial role in developing sound solutions. As we've seen, the adoption of new measures may take time, but progress is being made. Each step forward brings us closer to a more resilient and secure Web for everyone.

Authors



Gertjan Franken

@GJFR_ GJFR gertjan-franken <https://gjfr.dev>

Gertjan Franken is a postdoctoral researcher with the DistriNet Research Group⁴⁴⁷ at KU Leuven. His research spans various aspects of web security and privacy, with a primary focus on the automated analysis of browser security policies. As part of this research, he maintains the open-source tool BugHog⁴⁴⁸ for pinpointing bug lifecycles.



Vik Vanderlinden

@vikvanderlinden vikvanderlinden vikvanderlinden <https://vikvanderlinden.be/>

Vik Vanderlinden is a PhD candidate in Computer Science at the DistriNet Research Group⁴⁴⁹ at KU Leuven. His research focuses on web and network security, primarily focusing on timing leaks in web applications and protocols.

447. <https://distrinet.cs.kuleuven.be/>

448. <https://github.com/DistriNet/BugHog>

449. <https://distrinet.cs.kuleuven.be/>

Part III Chapter 12

CMS



Written by Jonathan Wold, Lora Raykova, and Niko Kaleev

Reviewed by Raelene Morey and Dan Knauss

Analyzed by Nurullah Demir and Jonathan Wold

Edited by Barry Pollard and Raelene Morey

In this chapter, we interpret the evolving landscape of the Content Management System (CMS) and its increasing influence on how users experience content on the web. We aim to explore both the broader CMS ecosystem and the unique characteristics of web pages created through these platforms.

CMS platforms are pivotal in the collective effort to build a fast, resilient web. By examining their current state, asking critical questions, and identifying areas for future exploration, we can better understand their impact on web performance and user experience.

This year, we've approached the data with curiosity and expertise in several popular CMSs. We encourage you to view our analysis through the lens of CMS variability and the diverse types of content they support.

What is a CMS?

A **Content Management System (CMS)** is a tool that allows individuals and organizations to

create, manage, and publish digital content, particularly on the web. A web-based CMS enables the seamless creation and management of websites while prioritizing user experience for both creators and visitors.

CMS platforms offer a variety of features for users to build websites, from user-friendly templates to more customizable options that require user input in the design and structure of the site. They also include administrative tools to simplify content management.

CMSs vary significantly in their approach to site creation. Some offer ready-made templates and drag-and-drop block builders, while others require users to design layouts and site structures. Regardless of approach, a CMS is supported by an ecosystem that typically includes hosting providers, extension developers, web agencies, and site builders.

In this chapter of the Web Almanac, we explore the entire ecosystem surrounding CMS platforms for 2023 and 2024, respectively. When we refer to a CMS, we mean the platform itself and the associated services and tools that form its ecosystem.

Based on Wappalyzer's CMS definition⁴⁵⁰, our dataset identifies 249 individual CMS platforms.

Some CMSs are open source, like WordPress and Joomla, while others are proprietary, such as Wix and Squarespace. These platforms offer hosting options, from free and self-hosted plans to premium, enterprise-level services.

The CMS landscape is a diverse and interconnected ecosystem, with platforms that differ in functionality, scale, and user experience.

CMS Adoption

Our analysis covers both desktop and mobile websites. While most URLs appeared in both datasets, some were accessed exclusively by desktop or mobile devices. We analyzed desktop and mobile results separately to account for these differences and avoid discrepancies.

450. <https://www.wappalyzer.com/technologies/cms>

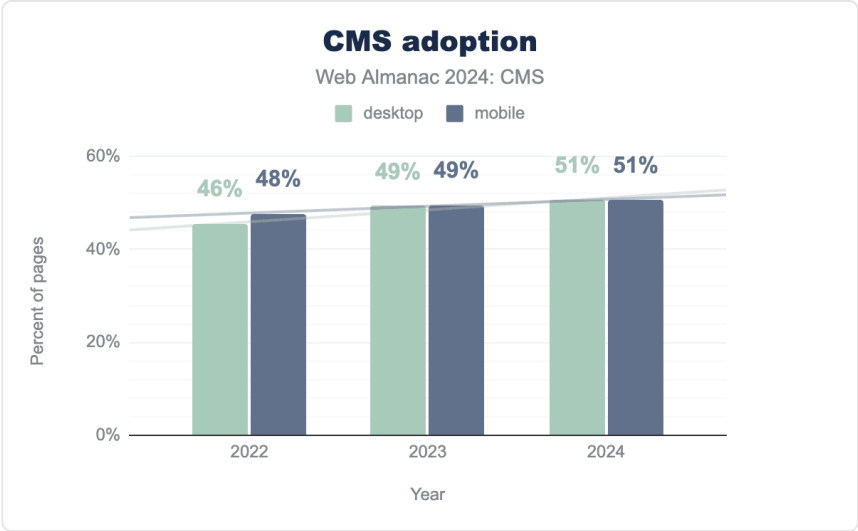


Figure 12.1. CMS adoption.

As of June 2024, 51% of the websites in the Web Almanac’s desktop dataset were powered by a CMS, a steady increase compared to 2022. The mobile dataset shows an increase from 48% in 2022 to 51% in 2024.

Looking closer at the raw desktop figures, we see a positive trend, with a clear bump registered in 2023. This is supported by data in both absolute and percentage terms, with the number of desktop URLs tracked by HTTP Archive (and the source CrUX dataset) growing from 5.4 million in July 2022 to 12.7 million in June 2024. The number of tracked mobile URLs, respectively, also reflects a booming growth in mobile device usage, with an increase of 8.6 million mobile URLs in the dataset for 2023 and a slight drop in 2024.

It’s important to note that our analysis differs from other commonly used datasets, such as W3Techs. These deviations are due to differing research methodologies and definitions of what qualifies as a CMS, which impact the final statistics.

For instance, as mentioned earlier, Wappalyzer uses a more strict definition of a CMS than we do, excluding some significant platforms that appear in W3Tech’s reports. You can learn more about our CMS criteria in the Methodology.

CMS adoption by geography

As of June 2024, CMS adoption worldwide has grown steadily, matching our dataset’s increased number of tracked URLs.

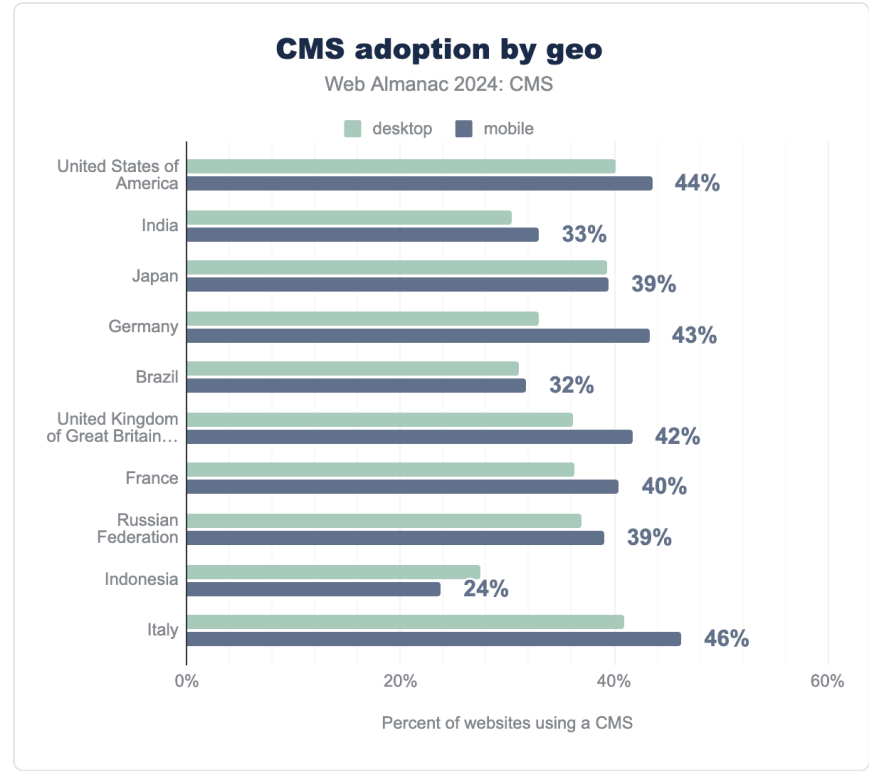


Figure 12.2. CMS adoption by geography.

This year, unlike our analysis in 2022, we differentiate between countries and regions to offer better insights into geographical CMS usage.

CMS adoption is highest in Italy and Spain, where 46% to 44% (40% to 41% in 2022) of mobile sites are built with a CMS. Brazil and Indonesia have the lowest adoption rates, with only 32% and 24%, respectively. Japan is seeing steady growth in mobile CMS adoption—39% in 2024 compared to 32% in 2022. Conversely, India shows a slight decrease in adoption (2% since June 2022). This can be attributed to the growing dataset and tracked URL increase, which helps us better understand the Indian web development market.

The year-over-year (YoY) analysis for mobile results shows consistent growth across countries, putting to rest some of our 2022 speculations⁴⁵¹ for a wholesale drop in CMS adoption.

Let's explore what the CMS adoption rate by region reveals.

451. <https://almanac.httparchive.org/en/2022/cms#cms-adoption-by-geography>

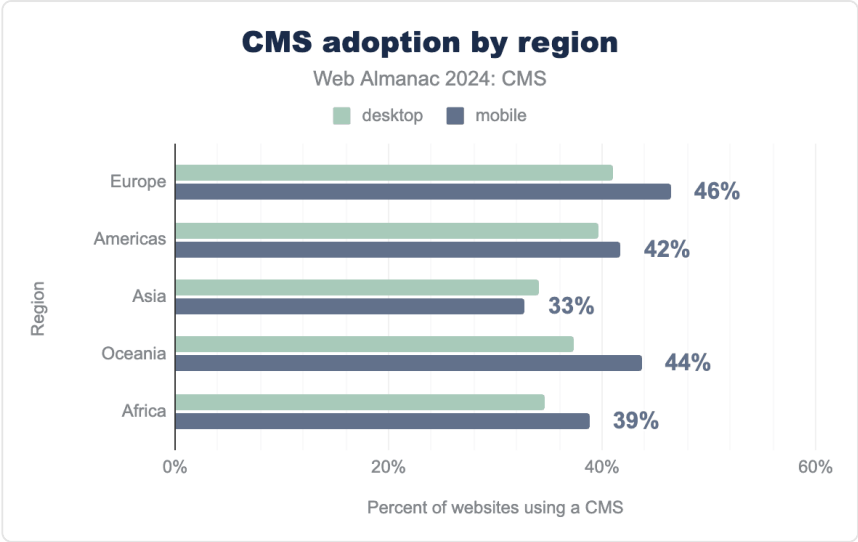


Figure 12.3. CMS adoption by region.

Following the trends in the country breakdown, Europe takes the lead with 46% for mobile results in CMS adoption rates. There are no discrepancies in the generated numbers followed by Oceania and the Americas. (44% and 42%, respectively.). It’s important to note that North America has a 44% rate of mobile CMS adoption across 1.5 million site pages, as shown in our CMS adoption by subregion graph below.

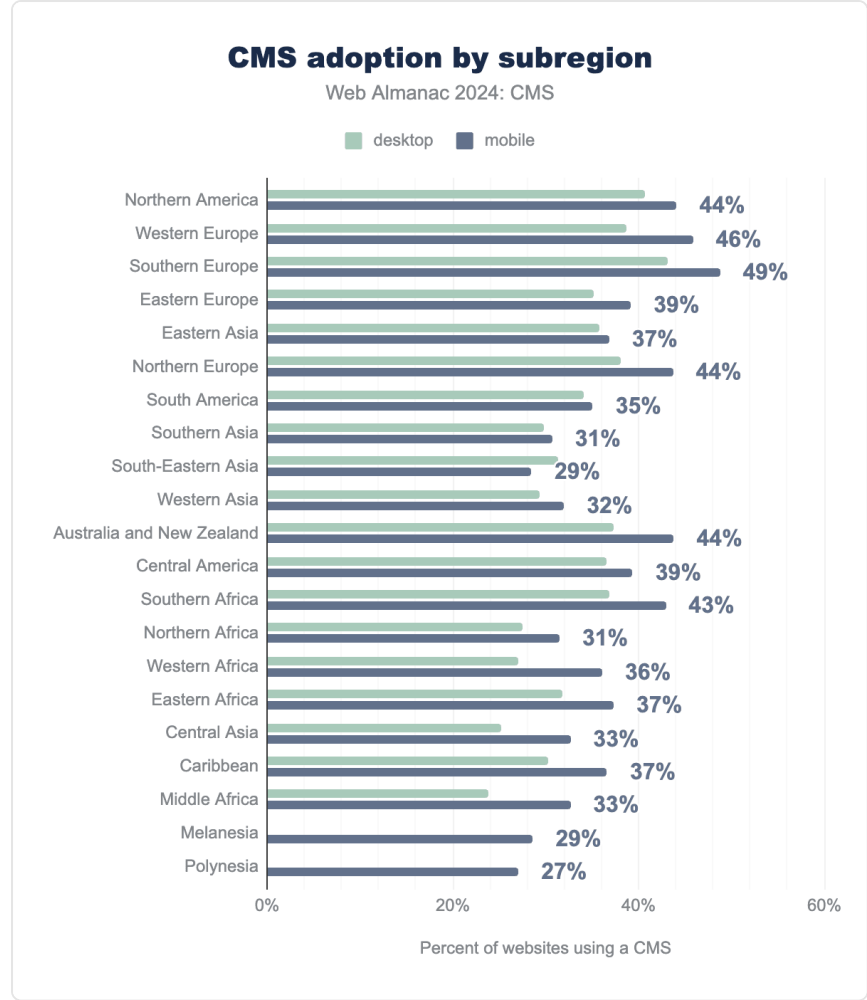


Figure 12.4. CMS adoption by subregion.

CMS adoption by rank

We examined CMS adoption by the estimated rank of the sites included within the dataset.

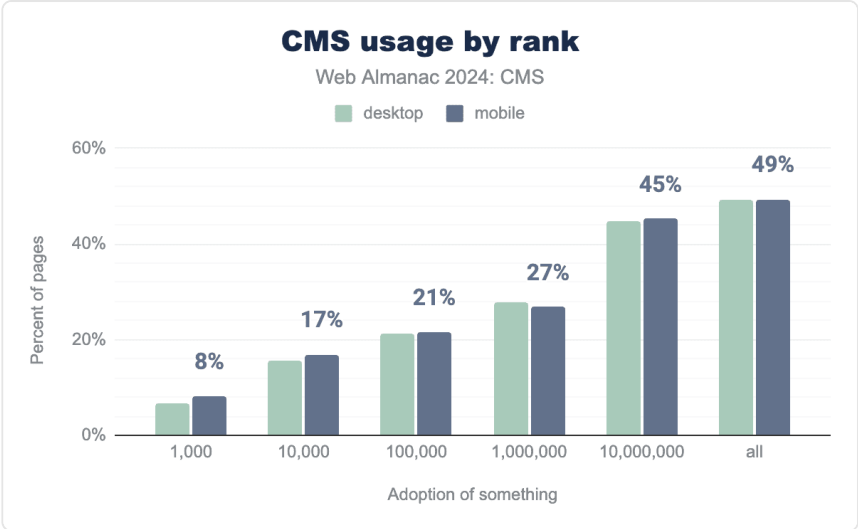


Figure 12.5. CMS usage by rank.

According to the dataset, CMSs are used by approximately 8% of the top 1,000 websites for both desktop and mobile, even though 49% of all mobile sites in the dataset use a CMS.

This has been an ongoing trend for the past few years, most likely attributed to business's size relative to its web development needs. Smaller businesses (represented by a large chunk of our dataset here) tend to use popular CMSs for their affordability and usability. In these cases, it's often easy to identify their CMS.

In contrast, larger businesses with higher-ranking websites often use custom-built CMS solutions that we can't readily identify. They are also more likely to obfuscate the identity of their CMS. It is improbable that more than 90% of the top 1,000 would forgo a CMS entirely. It's much more likely that they don't appear in our dataset.

Top-ranking websites aside, this year, a significant CMS adoption decrease was observed across all rank groups. The total percentage of websites using a CMS dropped 15% from 39% in 2022, which we suspect is primarily due to the increased size of our dataset in 2024.

A potentially correlated trend is the adoption of "headless" CMSs and the move to separate content—and the CMS that powers it—from the frontend experience offered to end-users. Another plausible explanation is the smaller size of our dataset compared to the tracked URLs used for the YoY CMS adoption rate, where we observe consistent growth.

Most popular CMSs

Let’s look a little more into the top CMS’s.

Top 5 CMS adoption growth

Among all websites that use an identifiable CMS, WordPress sites account for the majority of the relative market share—with over 35% adoption on mobile in 2024—followed by Wix (2.8%), Joomla (1.5%), Squarespace (1.5%), and Drupal (1.2%).

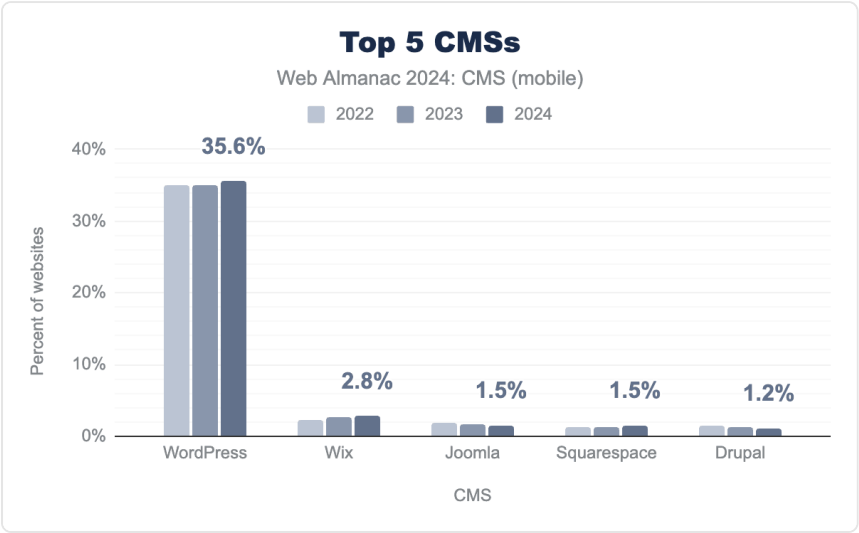


Figure 12.6. Top 5 CMSs Year on Year.

Comparing YoY, Drupal and Joomla continue to decline in market share while Squarespace and Wix grow (0.5% and 0.8%, respectively). WordPress continues its ascent, increasing 0.6% on mobile over 2023-2024 This represents a slower pace of growth than in previous years.

CMS user experience

With the introduction of Core Web Vitals four years ago, user experience has become a priority. That said, while users were mainly comparing CMS platforms based on ease of use, number of plugins/extensions, and themes available, they added one additional criterion—the default user experience offered by a particular platform.

To examine these experiences, we gathered data from the Chrome User Experience Report (CrUX)⁴⁵² and interrogated three specific metrics:

- Core Web Vitals
- Lighthouse scores
- Resource weights

Core Web Vitals

Google's Core Web Vitals are a set of three user-centric performance metrics that measure critical aspects of user experience, focusing on loading speed, interactivity, and visual stability:

- Largest Contentful Paint (loading)
- Interaction to Next Paint (interactivity)
- Cumulative Layout Shift (visual stability)

Since their introduction in 2020 and becoming a ranking signal in 2021, CWV are now recognized as essential to delivering a good user experience across the web.

If you're interested in how websites perform against the Core Web Vitals on a larger scale, the Performance chapter covers this topic in greater detail.

In this section, we are interested in looking at the Core Web Vitals specifically in the context of CMS platforms.

There are 200+ known CMS platforms, but we narrowed our list to the top 10 most used CMSs, considering they have more than 85% market share. We used the Core Web Vitals Technology Report⁴⁵³, which provides a global overview of how different technologies perform in relation to Google's Core Web Vitals.

Below is the percentage of sites on each platform that score "good" (LCP under 2.5s; INP under 200ms; CLS below 0.1) for all three Core Web Vitals:

452. <https://developer.chrome.com/docs/crux>

453. <https://lookerstudio.google.com/reporting/55bc8fad-44c2-4280-aa0b-5f3f0cd3d2be/page/M6ZPC>

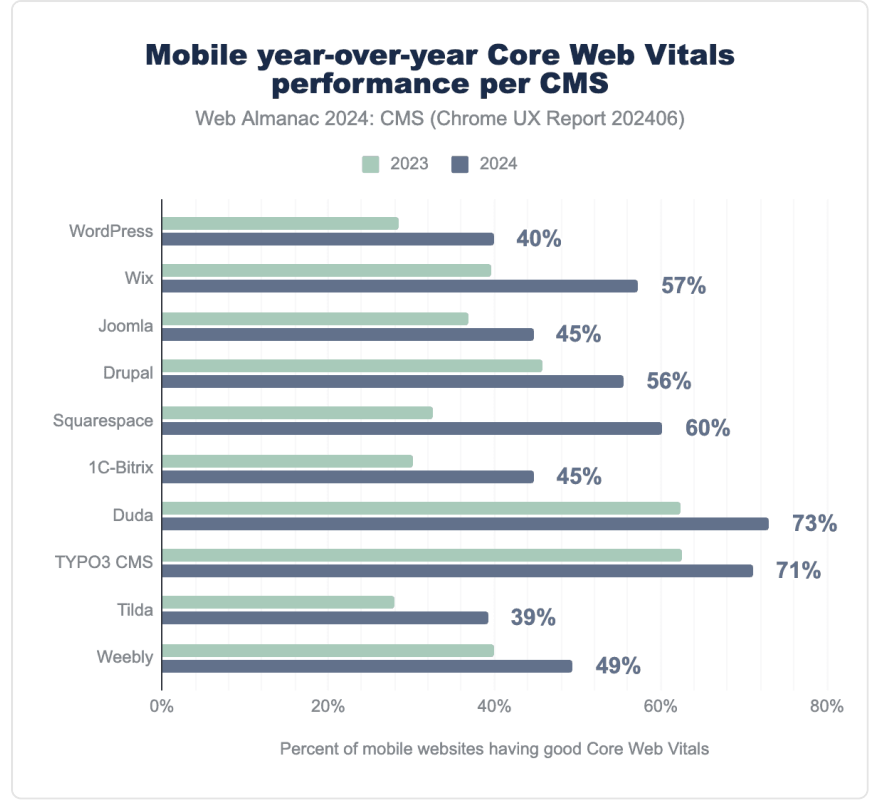


Figure 12.7. Mobile year-over-year Core Web Vitals performance per CMS.

We decided to focus primarily on mobile results for three reasons:

1. As of July 2024, mobile has a 68.29% global traffic share
2. Mobile devices have resource and connectivity limitations, which paint a more realistic picture of how everyday users experience the web
3. All CMS platforms, and technologies in general, score better on desktop

Putting that aside, we can see that all 10 CMS platforms improved their Core Web Vitals, with Squarespace leading the board with a whopping 28% YoY improvement. The top 3 include Wix with 18% and 1C-Bitrix with 14%. The list then goes as follows: WordPress (11% YoY), Tilda (11% YoY), Duda (11% YoY), Drupal (10% YoY), and Weebly (10% YoY). At the bottom are TYPO3 CMS with 9% and Joomla with 8%.

These YoY improvements are as remarkable as the overall Core Web Vitals pass rate the top 10 CMSs collectively provide. For reference, the global CWV pass rate for all origins as of June

2024 is 51%.

Having platforms like Duda, TYPO3 CMS, and Squarespace with results way north of that is a testament to the efforts these platforms have put into enhancing user experience.

Now, let's drill into the three Core Web Vitals to see where each platform can improve and which metrics improved the most since 2023.

Largest Contentful Paint (LCP)

Largest Contentful Paint measures how long it takes for the largest above-the-fold element to load on a page. Anything under 2.5 seconds is considered a "good" LCP score.

This metric represents how quickly users can see what is likely the most meaningful content on a web page.

Of the three Core Web Vitals, LCP is the hardest to optimize. That's why LCP pass rates are the lowest and LCP is considered the bottleneck to passing CWVs.

The reason why LCP is such a challenging metric is that there are a lot of moving parts when it comes to its optimization. You need to:

1. Ensure the LCP resource starts loading as early as possible.
2. Ensure the LCP element can be rendered as soon as its resource finishes loading.
3. Reduce the load time of the LCP resource as much as you can without sacrificing quality.
4. Deliver the initial HTML document as fast as possible.

However, the top 10 CMSs showed some impressive results when it comes to their YoY LCP improvements and overall scores:

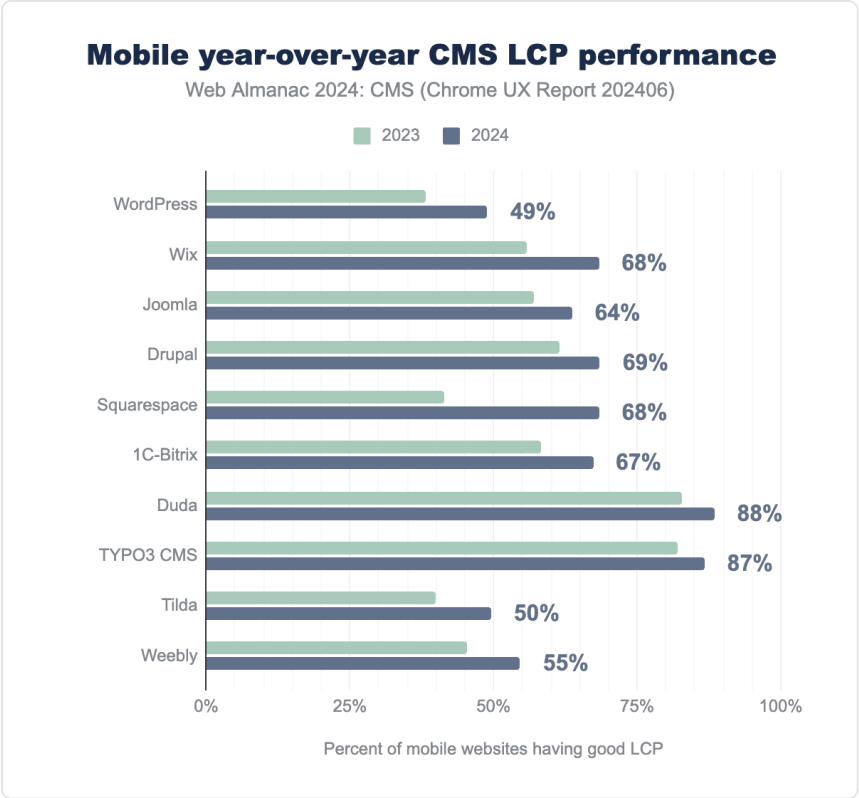


Figure 12.8. Mobile year-over-year CMS LCP performance.

Before diving into the YoY stats, take note that the global LCP pass rate (as of June 2024) is 63.4%. As you can see from the graph, more than half of the platforms achieve better results.

Considering how much they have improved since 2023, Squarespace is again ahead of the other CMSs with a YoY improvement of 27%. Wix is in second place with 13%, and WordPress is third with 11%. The remaining platforms improved by less than 10%.

Cumulative Layout Shift (CLS)

Cumulative Layout Shift (CLS) is a metric used to measure layout stability. It reflects how much content unexpectedly shifts on the screen.

A website is considered to have good CLS if at least 75% of all site visits score 0.1 or lower.

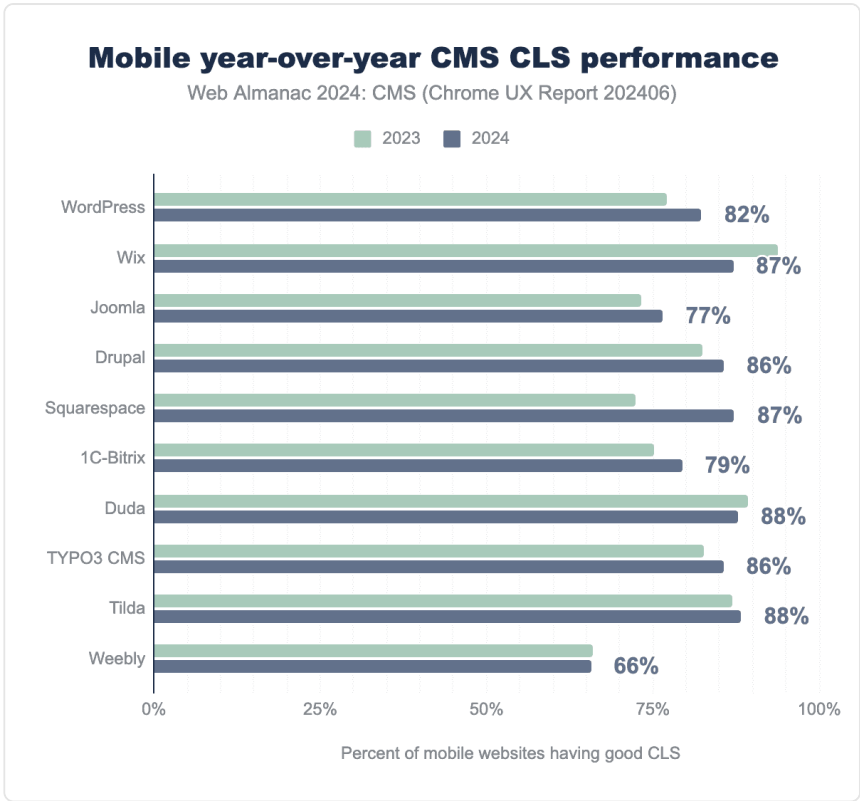


Figure 12.9. Mobile year-over-year CMS CLS performance.

Comparing yearly data shows that the CLS results aren't as impressive as the LCP or the overall CWV pass rates. Most platforms made little progress, while Duda, Wix, and Weebly struggled to improve their 2023 results.

Interaction to Next Paint (INP)

After being announced in 2023 and in an experimental phase for the rest of the year, Interaction to Next Paint officially replaced First Input Delay (FID) as the new interactivity Core Web Vitals metric on March 12, 2024.

INP assesses a page's responsiveness to user interactions by observing the latency of all qualifying interactions during a user's visit to a page. The final INP value is the longest interaction observed.

Simply put, the newest Core Web Vital measures the time from the interaction (for example, a

mouse click) until the browser can update (or paint) the screen.

An INP below or at 200 milliseconds means a page has good responsiveness.

Considering the whole interaction latency, including input delay, processing time, and presentation delay, the introduction of INP led to a global drop in the Core Web Vitals pass rate.

That said, the YoY improvement and overall CWV pass rate demonstrated by the top 10 CMSs is even more impressive, considering the big changes INP introduced.

When it comes to their INP score, the majority of platforms achieve a pass rate of 80% or above:

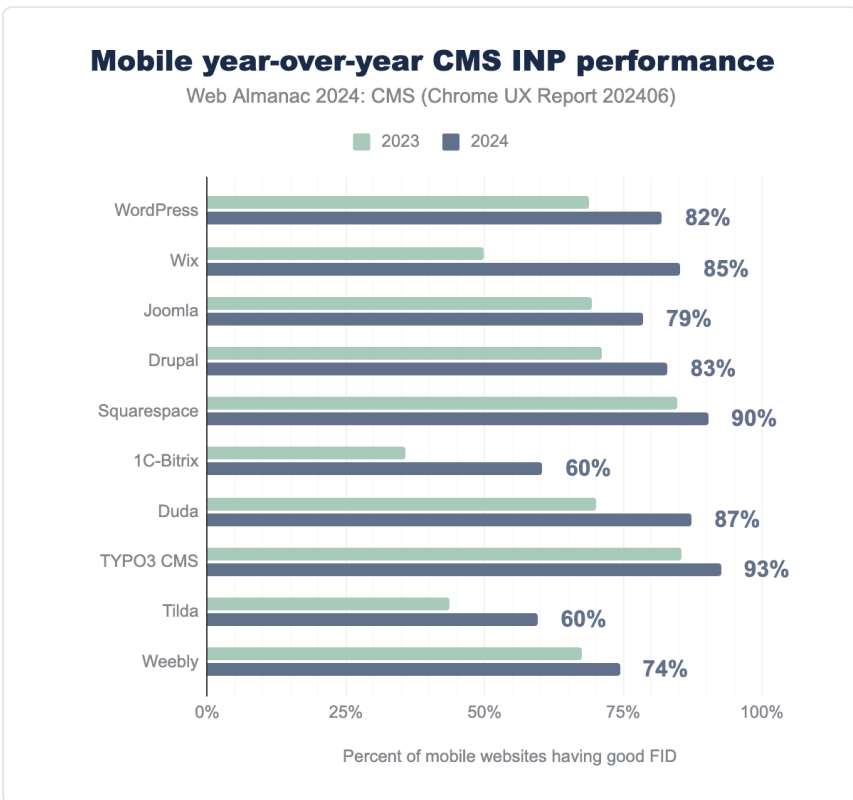


Figure 12.10. Mobile year-over-year CMS INP performance.

The CMSs that fail to pass include Tilda and 1C-Bitrix, which, despite the YoY improvement, still struggle to achieve the global standard of 84.1%.

The Deprecation of FID

First Input Delay is no longer a Core Web Vitals metric. Furthermore, Chrome officially deprecated support for FID on September 9, 2024.

The primary reasons for INP replacing FID include the scope of measurement and comprehensiveness.

Looking at the Web Almanac's previous editions we can see that all websites had a good FID score on desktop, and nearly all on mobile. This data should indicate that visitors rarely experience a laggy website.

Unfortunately, the reality is that the web often does have a responsiveness problem. And FID was no longer accurately measuring it.

The issue with FID was that it only measured the delay for the first input, ignoring the processing time and presentation delay and failing to capture the page's responsiveness throughout the user's entire session.

In contrast, INP evaluates the responsiveness of the entire page throughout the user's session, providing a more comprehensive and accurate representation of the user experience.

Lighthouse

Lighthouse⁴⁵⁴ is an open-source, automated tool designed to improve the quality of web pages by providing a set of audits that evaluate websites based on 4 categories:

- Performance
- Accessibility
- SEO
- Best Practices

Lighthouse generates reports with lab data that offer developers actionable suggestions for enhancing website performance. However, it's important to note that Lighthouse scores do not directly impact the real-world field data collected by CrUX⁴⁵⁵. You can further explore how Lighthouse lab scores and field data differ⁴⁵⁶.

The HTTP Archive runs Lighthouse on mobile and desktop web pages, simulating a slow 4G

454. <https://developers.google.com/web/tools/lighthouse/>

455. <https://developers.google.com/web/tools/chrome-user-experience-report>

456. <https://web.dev/lab-and-field-data-differences/>

connection with throttled CPU performance for mobile.

By analyzing this data, we can gain a different perspective on CMS performance through synthetic tests, which also capture metrics not monitored by CrUX.

Performance score

The Lighthouse performance score⁴⁵⁷ is a weighted average of several scored metrics.

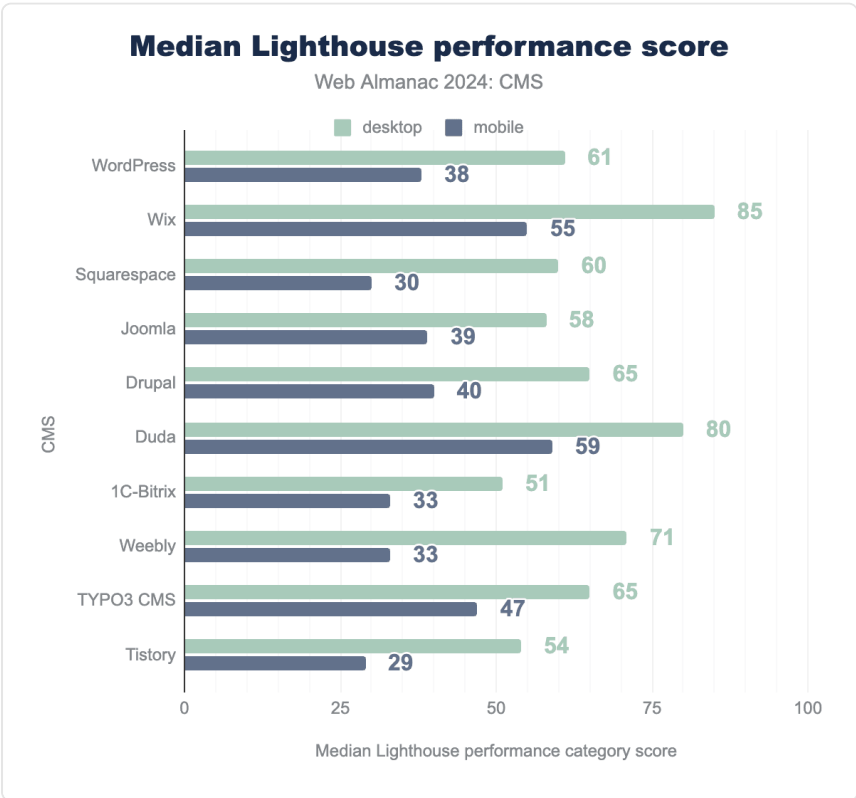


Figure 12.11. Median Lighthouse performance score.

Compared to the 2022 results, where the median performance scores for most mobile platforms ranged from about 19 to 35, we observe impressive improvements for both mobile and desktop websites in 2024.

457. <https://web.dev/performance-scoring/>

Wix and Duda stand out as clear leaders with median desktop performance scores of 85 and 80, respectively, and Duda achieving higher scores on mobile. Next come Weebly, Drupal, TYPO3 CMS, WordPress, and Squarespace, representing median performance scores marked with the orange “Needs Improvement” status, according to the Lighthouse color-coding scheme.

The overall positive trend is due to native and technological improvements in browsers and CMSs, which indicates an overall recognition of the importance of high-quality web performance.

The major increase shown by proprietary platforms such as Wix and Duda is partly connected to how the CMSs operate—meaning they benefit from a streamlined, centralized development that allows greater speed for innovation in the performance field.

As we’ve concluded in previous years, the lower mobile scores are an opportunity for smarter solutions, and optimizations focused on low-end devices with unstable network connections similar to those Lighthouse attempts to emulate. Moreover, it’s inherent for mobile devices to fall behind their desktop counterparts, given the faster CPUs and networks available. Nonetheless, the 2024 results are encouraging, and we’ll continue to track how CMS platforms fare regarding Lighthouse performance metrics.

Year-over-year performance trends

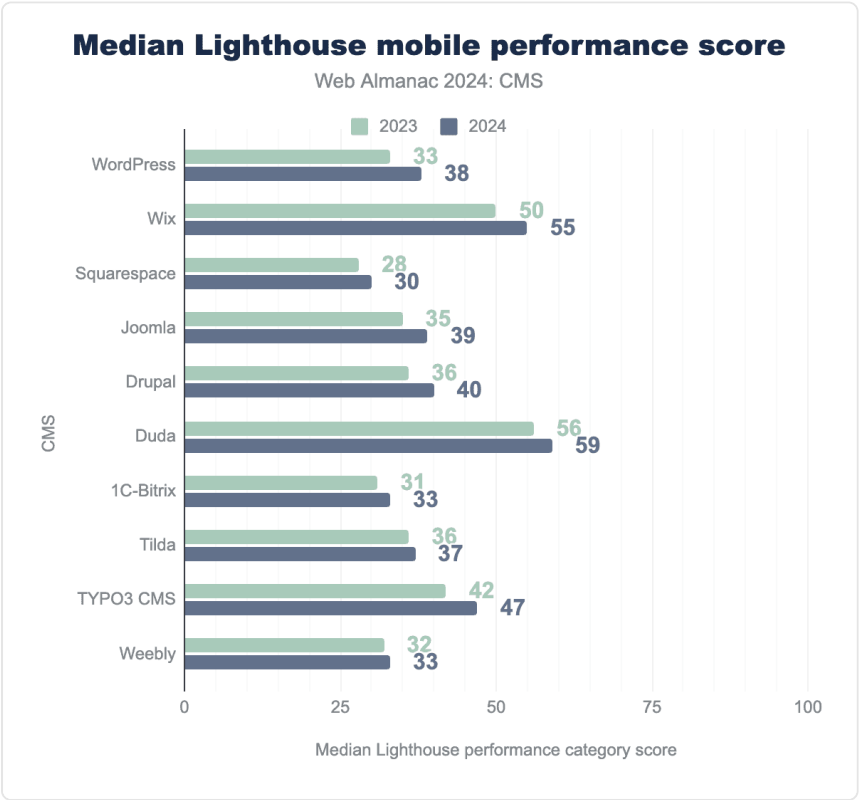


Figure 12.12. Median Lighthouse mobile performance score.

The YoY performance data from 2023 to 2024 reveals an encouraging trend of incremental improvements among top CMS platforms. Duda and Wix continue to lead in mobile performance, with Duda improving from a median Lighthouse score of 56 to 59 and Wix rising from 50 to 55. WordPress also shows improvement, increasing from 33 to 38, alongside Joomla and Drupal, which moved from 35 to 39 and 36 to 40, respectively. These results reflect a broader industry focus on optimizing mobile performance, although some platforms, such as Squarespace and Weebly, showed only minor gains. These varied improvements highlight the ongoing challenges and priorities across CMS platforms as they work toward enhanced user experiences on mobile devices.

SEO score

Search Engine Optimization (or SEO) is the practice of improving the quality and quantity of visitors to a website or a web page from a website to make it more easily found in search engines. This topic is covered in our SEO chapter, but it also relates to CMSs.

A CMS and content on it is generally set up to serve as much information to search engine crawlers as possible to make it as easy as possible for them to index site content appropriately in search engine results. Compared to a custom-built website, one might expect a CMS to provide good SEO capabilities, and the Lighthouse scores in this category remain appropriately high.

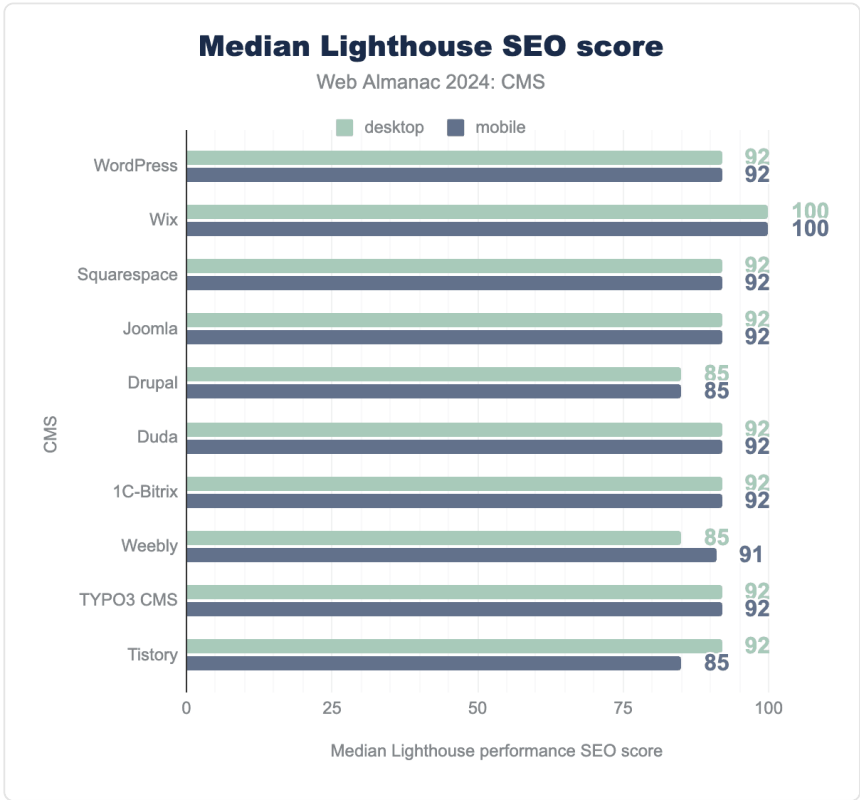


Figure 12.13. Median Lighthouse SEO score.

The median SEO scores in the top 10 platforms range from 85-100, an impressive increase from

82-92 in 2022⁴⁵⁸. Given the high median performance scores, it is no surprise Wix takes the lead in SEO as well, scoring a perfect 100 for both mobile and desktop. With median scores as high as 92 for mobile and desktop, the runner-ups provide users with robust SEO best practices.

Year-over-year SEO trends

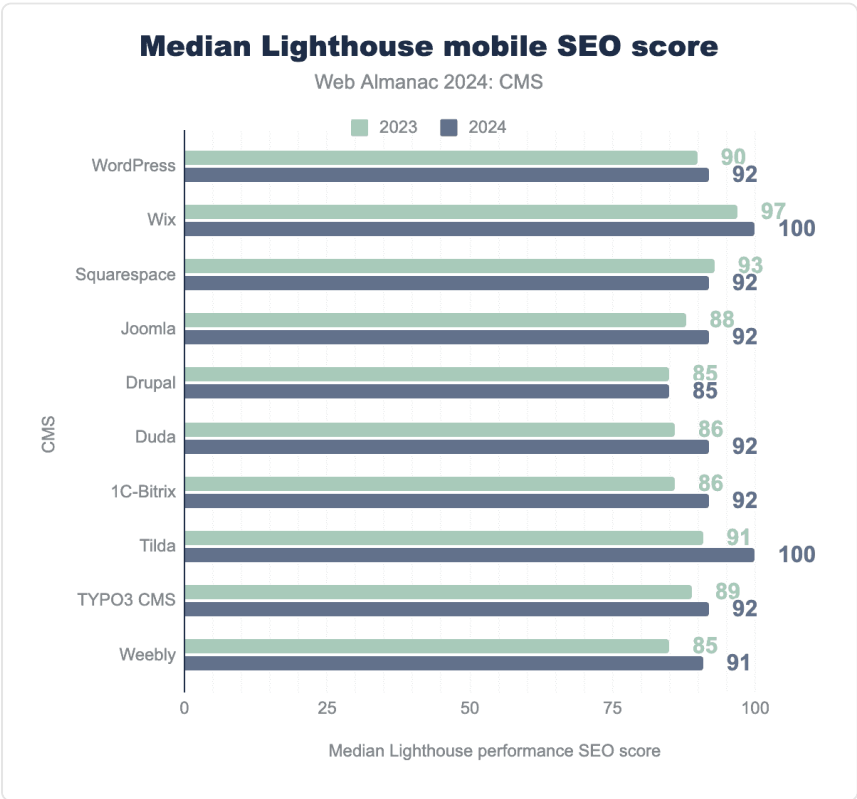


Figure 12.14. Median Lighthouse mobile SEO score.

The year-over-year comparison from 2023 to 2024 reveals consistently high SEO scores across the top CMS platforms, with most maintaining or slightly improving their median scores. This stability at the high end suggests that CMS platforms are prioritizing SEO best practices, ensuring that users have solid foundations for search engine visibility.

⁴⁵⁸. <https://almanac.httparchive.org/en/2022/cms#seo-score>

Accessibility score

An accessible website is a site designed and developed so that people with disabilities can use them. Web accessibility also benefits people without disabilities, such as those on slow internet connections. Read more in our Accessibility chapter.

Lighthouse provides a set of accessibility audits and returns a weighted average of all of them. See [scoring details⁴⁵⁹](#) for a full list of how each audit is weighted.

Each accessibility audit is either a pass or a fail, but unlike other Lighthouse audits, a page doesn't get points for partially passing an accessibility audit. For example, if some elements have screen reader-friendly names but others don't, that page gets a zero for the screen reader-friendly names audit.

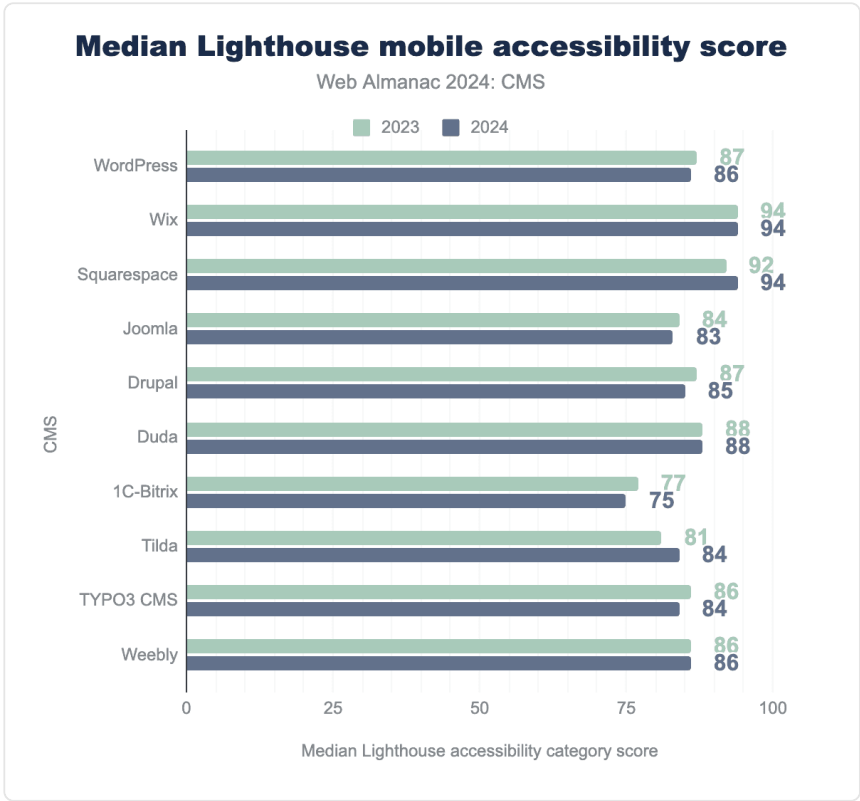


Figure 12.15. Median Lighthouse mobile accessibility score.

459. <https://web.dev/accessibility-scoring/>

In 2024, the median Lighthouse accessibility score for the top 10 CMSs ranges between 74 and 95, a slight improvement from 77 to 91 in 2022⁴⁶⁰. Squarespace has lost the top space by a fraction to Wix, which hits the 94 and 95 marks for mobile and desktop, respectively. 1C-Bitrix had the lowest accessibility scores in 2024 and shows a slight decrease of 2 points for both desktop and mobile, perhaps reflecting that the same sites are delivered to both desktop and mobile devices.

Year-over-year accessibility trends

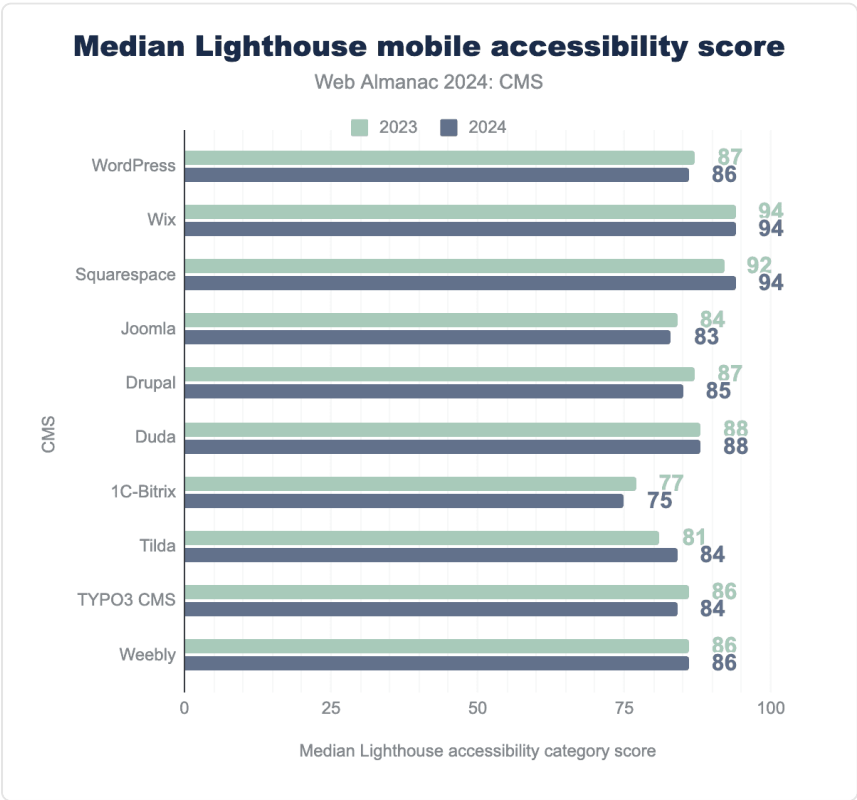


Figure 12.16. Median Lighthouse mobile accessibility score.

The 2023 to 2024 comparison reveals largely stable accessibility scores across top CMS platforms, with most showing minimal changes. Wix shares the lead with Squarespace, holding steady with a high score of 94 for both years. Squarespace improved from 92 in 2023 to match Wix's 94 in 2024. Notably, WordPress, Joomla, Drupal, and TYPO3 all saw minor declines.

460. <https://almanac.httparchive.org/en/2022/cms#accessibility-score>

These scores suggest a steady emphasis on accessibility across the proprietary CMSs, with the open source CMSs demonstrating room for improvement.

Best practices

Lighthouse's best practices⁴⁶¹ audit evaluates whether web pages adhere to widely accepted web standards across various metrics. These include critical factors such as:

- HTTPS support,
- console error elimination,
- deprecated APIs avoidance,
- browser compatibility optimization,
- security protocols,
- and more.

By following these best practices, developers can enhance both the functionality and user experience of their websites, ensuring a more secure, stable, and reliable performance across different browsers and devices.

461. <https://web.dev/lighthouse-best-practices/>

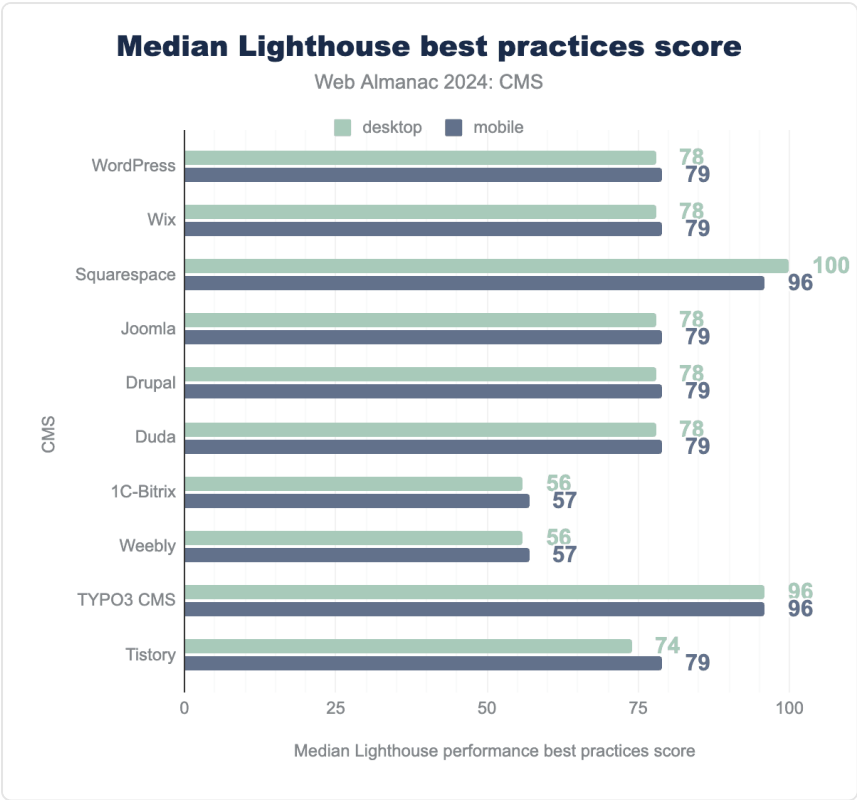


Figure 12.17. Median Lighthouse best practices score.

Our 2024 analysis shows significant changes across the board compared to the results in 2022. Squarespace takes the lead from Wix, with the highest median best practices score of 100, while many of the other top 10 platforms share a score of 78 (a slight improvement since 2022).

While most other CMSs show worse numbers in the best practices audits, TYPO3 CMS claims the second place with a 96-median score for both mobile and desktop, compared to 83 and 92 (mobile and desktop, respectively) in 2022.

Year-over-year Best Practices Trends

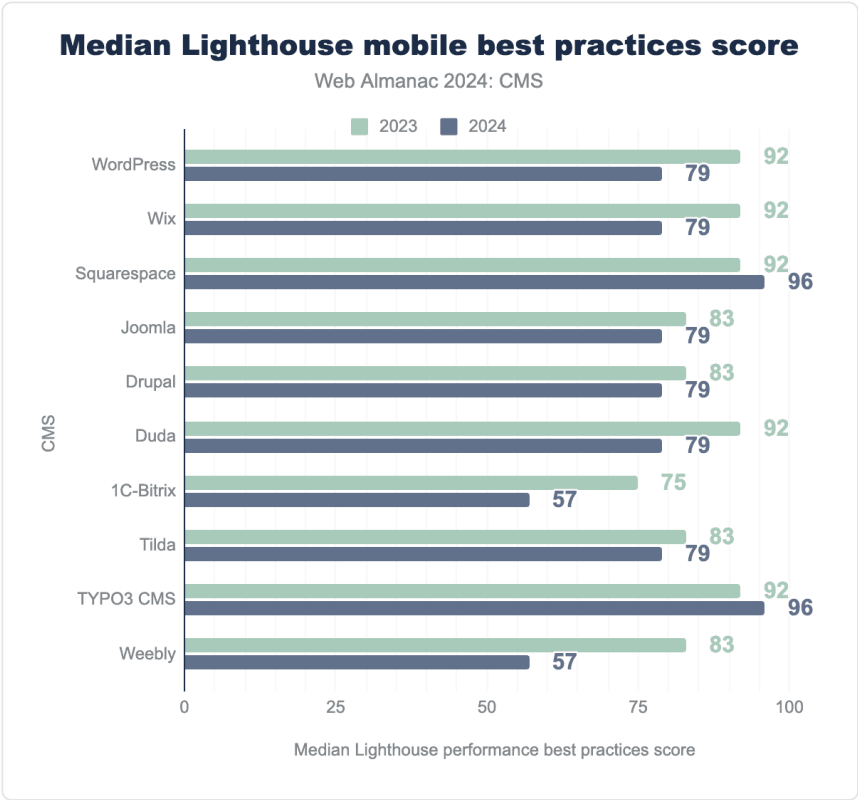


Figure 12.18. Median Lighthouse mobile best practices score.

The YoY comparison from 2023 to 2024 shows stable best practices scores across most CMS platforms. Squarespace and TYPO3 CMS share the top position in 2024, each with a best practices score of 96. Most other platforms, including WordPress, Wix, Joomla, Drupal, Duda, and Tilda, maintained steady scores of 79, reflecting consistent adherence to web standards. In contrast, 1C-Bitrix and Weebly scored lower at 57, highlighting areas where these platforms could improve in best practices compliance.

Resource weights

We leveraged HTTP Archive data to analyze the resource weight across various CMS platforms, aiming to uncover opportunities for performance optimization. While page loading

speed isn't solely determined by the number of downloaded bytes, reducing the amount of data required to load a page offers several advantages.

Fewer bytes mean lower bandwidth costs, reduced carbon emissions, and faster performance—particularly for users on slower connections. This analysis sheds light on areas where resource optimization can have a meaningful impact on both user experience and sustainability. Read the Page Weight chapter for a more detailed analysis.

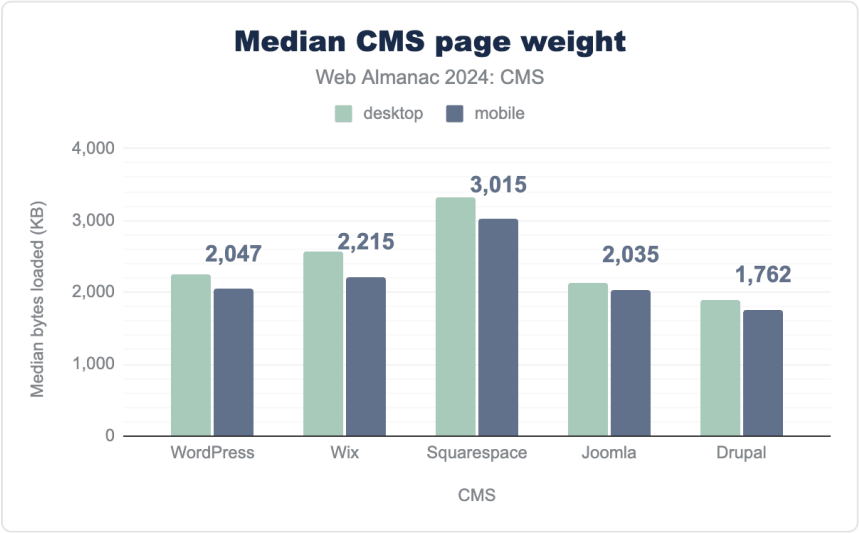


Figure 12.19. Median CMS page weight.

Although we observed a steady trend in increasing page weight over the past several years, in 2024, almost all top five CMSs show notable improvements. Drupal's median page weight has dropped to ~1.7 MB compared to ~2.1 MB in 2022. WordPress and Joomla now hover closely to ~2 MB compared to ~2.3 MB in 2022. Wix is the only CMS on the board that shows a slight increase in median page weight—2.2 MB compared to 2.1MB in 2022. Squarespace continues to deliver the heaviest median page weight of ~3 MB compared to ~3.5MB in 2022.

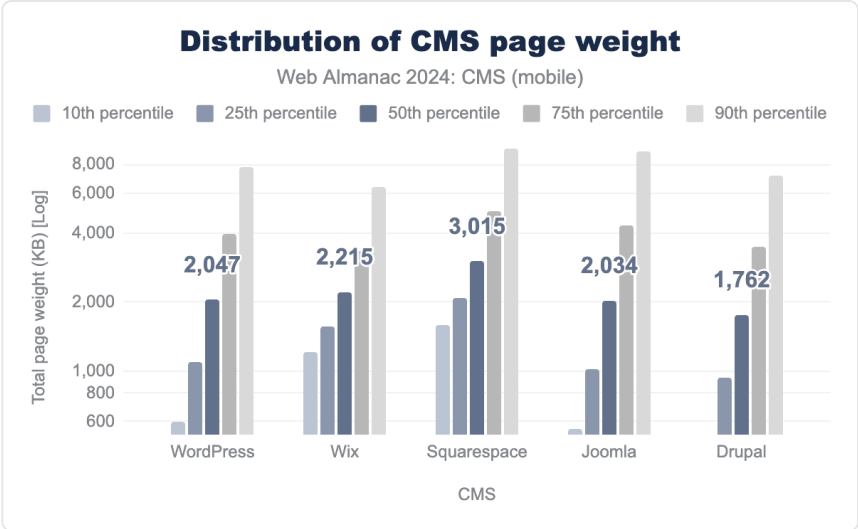


Figure 12.20. Distribution of CMS page weight.

The distribution of page weight in each platform’s percentiles is substantial. Page weight is probably related to the differences in user content across web pages, the number of images used, plugins installed, etc. The smallest pages delivered per platform come from Drupal, followed closely by WordPress—both improving on their results from 2022 (2.1 MB and 2.3 MB, respectively).

This year, Drupal only sends 524 KB for their 10th percentile of visits, while Joomla and WordPress are not falling far behind—561 KB and 598 KB, respectively. With a notable decrease, but still serving the largest pages here, is Squarespace, with ~9.4 MB delivered for their 90th percentile of visits, a ~2 MB decrease compared to 2022 data.

Page weight breakdown

Page weight refers to the total size, measured in kilobytes, of all the resources loaded on a web page. These resources—such as images, JavaScript, CSS, HTML, and fonts—collectively influence the page’s performance.

Below, we analyze and compare the resource weight across different CMS platforms, providing insights into how each CMS contributes to the overall page weight.

Images

Images are a significant resource on most websites, often accounting for the largest portion of page weight. In CMS platforms, where visuals are heavily relied upon for design and engagement, image optimization is essential. High-resolution images can slow down page load times if not properly compressed or served in modern formats like WebP. Read more in our Media chapter.

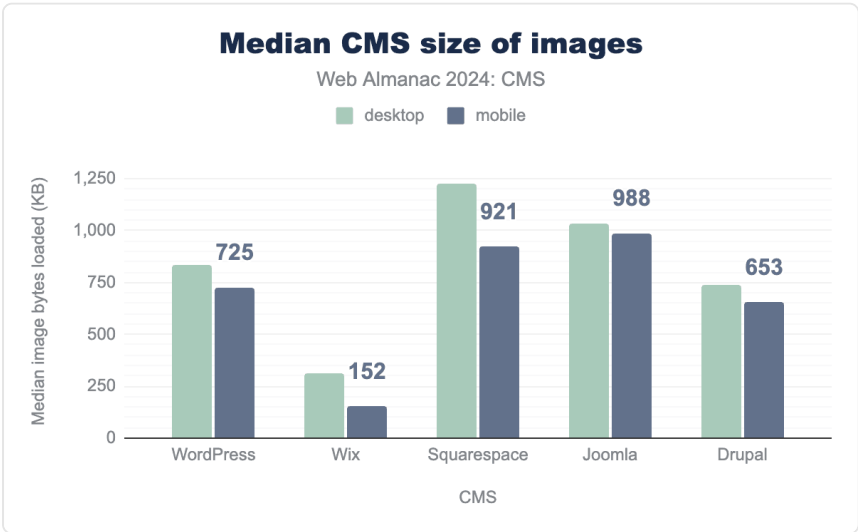


Figure 12.21. Median CMS size of images.

In 2024, Wix continues to deliver substantially fewer image bytes, with only 152 KB delivered for the median of mobile views (138 KB less than in 2022). This suggests good use of image compression and lazy image loading. All the other top four platforms deliver below 1 MB of images—a significant improvement compared to 2022 data (WordPress 1,1 MB; Squarespace ~1,7 MB, Joomla ~1,5 MB, and Drupal ~1,1 MB).

Advanced image formats greatly improve compression, enabling resource savings and faster site loading. WebP is commonly supported in all major browsers today, with over 95% support. In addition, newer image formats continue to gain popularity and adoption, namely AVIF.

We can examine the usage of the different image formats across the top CMSs:

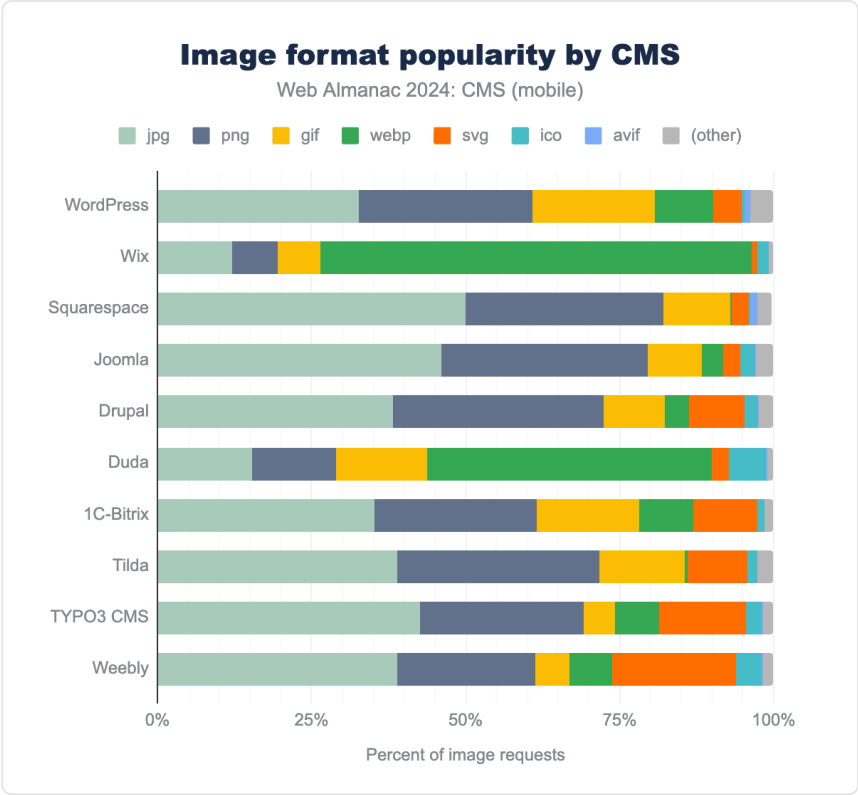


Figure 12.22. Image format popularity by CMS for mobile.

In 2024, Wix and Duda continue to make the most use of WebP, at ~75% and ~60% adoption, respectively, while the rest show minor increases.

Although WebP is now widely supported, we still observe that platforms are underutilizing the format. With the growing support of WebP, it seems all platforms have work to do to reduce the usage of the older JPEG and PNG formats without compromising on image quality.

As of WordPress 5.8, WordPress supports the WebP format and can be set to automatically convert uploaded images to WebP. However, the popularity of the format seems to have plateaued compared to the 2022 data. This can be explained by the more holistic approach the core Performance team at WordPress has chosen toward general performance improvements on the platform. Read more in the section WordPress in 2024.

JavaScript

JavaScript powers much of the interactive functionality on modern websites. CMS platforms frequently use various JS libraries and frameworks to enable features like dynamic content loading, form validation, and user engagement tools. However, excessive or poorly optimized JavaScript can negatively impact performance. Find more detailed information in our JavaScript chapter.

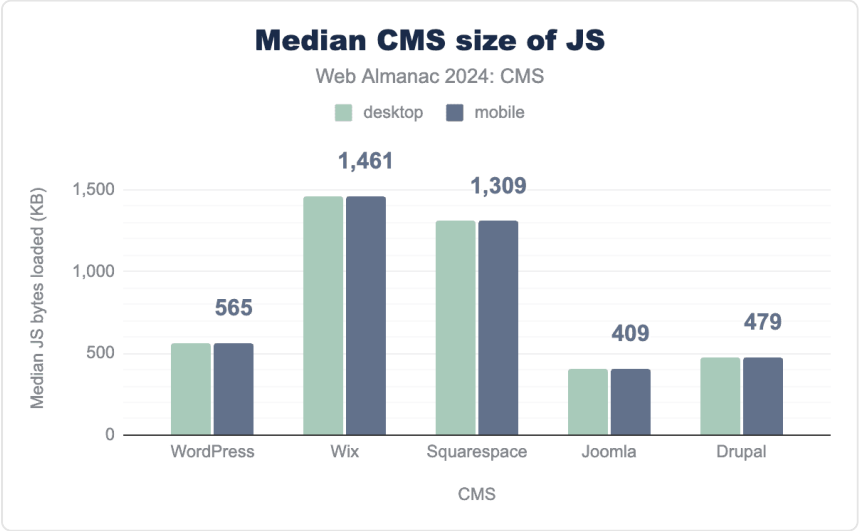


Figure 12.23. Median CMS size of JS.

In 2024, we observe rising JavaScript usage across the board. Almost all top five CMSs deliver more JavaScript compared to 2022 data, with Squarespace seeing a major increase, from 997 KB to ~1.3 MB. Wix still delivers the most JavaScript with a slight increase, from ~1.3 MB to ~1.4 MB. Drupal and WordPress show minor JavaScript growth, with Joomla delivering the least JS at 409 KB (an improvement from 2022’s 452KB).

HTML

HTML forms the structural backbone of any web page, defining the layout and content. CMS platforms automatically generate much of the HTML code, sometimes leading to bloated and inefficient markup. While HTML is typically lightweight compared to other resources, unoptimized or overly complex HTML structures can still affect render time and user experience. Find more detailed information in the Markup chapter.

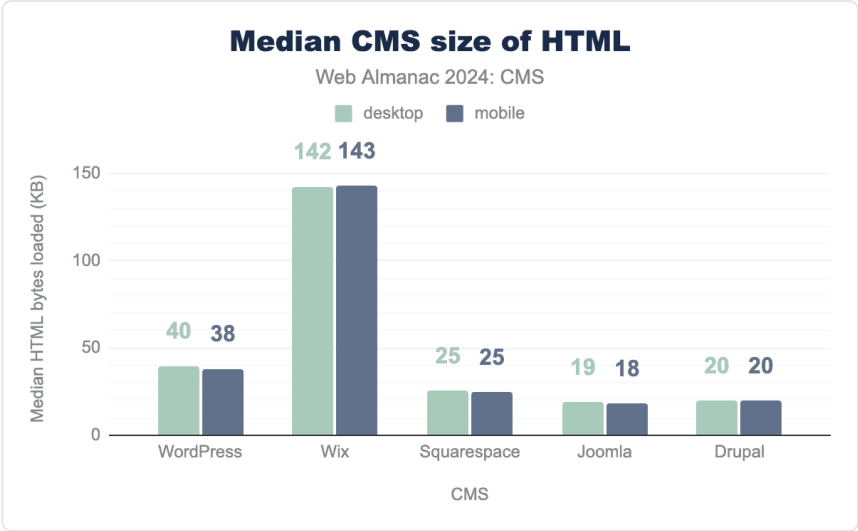


Figure 12.24. Median CMS size of HTML.

Examining the HTML document sizes, we can see that most of the top CMSs deliver a median HTML size of ~22 KB–38 KB. The only exception is Wix, which delivers ~142 KB, a notable increase over 2022’s results. This may suggest extensive use of inlined resources and shows an area that can be further improved.

CSS

CSS controls the visual styling of a website, dictating elements like layout, colors, and fonts. On CMS platforms, themes and templates often come with extensive CSS files that may include unused or redundant styles. Large CSS files can increase page weight and slow down rendering times.

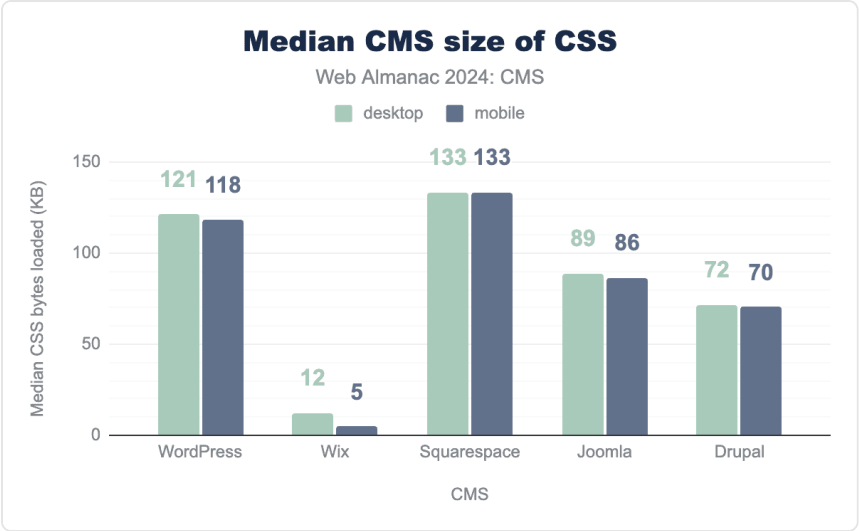


Figure 12.25. Median CMS size of CSS.

In 2024, we see a different distribution between platforms that strengthens the case for inlining CSS. Wix delivers the fewest CSS resources, dropping from 9 KB to 5 KB on mobile views in 2022. Squarespace delivers significantly more CSS this year—133 KB from 89 KB in 2022. WordPress comes second in CSS delivery with a slight increase from 2022 numbers. Drupal also shows a minor increase, while Joomla is the only CMS platform that delivers less CSS in 2024.

Fonts

CMS-based websites frequently offer a variety of fonts to enhance branding and visual appeal. However, fonts can introduce additional weight to a page, especially when multiple font families or variations are loaded. Unoptimized font loading can delay text rendering and impact the user experience. Explore the Fonts chapter for a detailed breakdown.

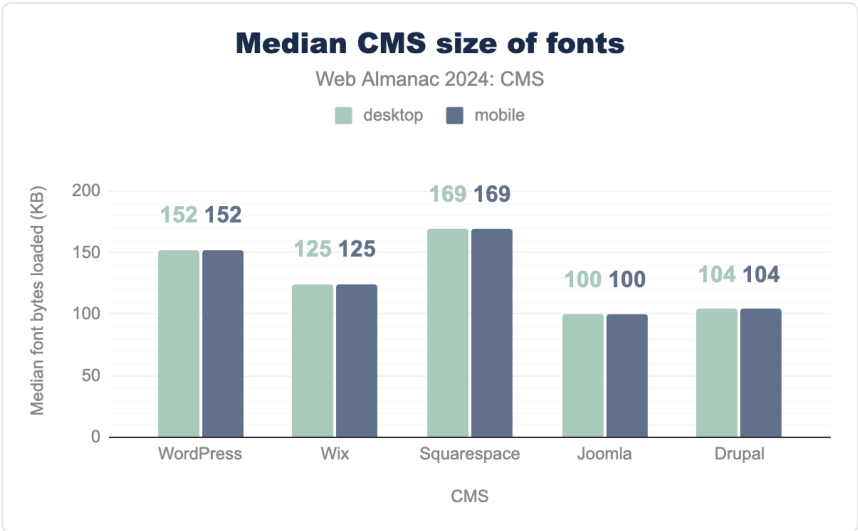


Figure 12.26. Median CMS size of fonts.

This year, all top five CMSs show changes in the amount of font bytes delivered. Squarespace continues to deliver the highest amount at 169 KB (a significant decrease from 202 KB in 2022). Wix also shows a slight decrease in font file delivery, while WordPress, Joomla, and Drupal all have increased the amount of font bytes served. It would be interesting to see if the growing integration of Google-hosted fonts will impact next year’s results.

WordPress in 2024

36%

Figure 12.27. Percentage of mobile sites using WordPress.

Of the over 16 million mobile sites in this year’s crawl, WordPress is used by 5.7 millions sites for a total of 36% of sites. By comparison, the next closest CMS is Wix, with 456,253 sites or 3% of sites.

WordPress’s global dominance stems from two main factors—a community that maintains and improves the functionality of the open-source project, and the CMS’s flexibility in serving a wide range of websites and users.

Furthermore, the availability of tens of thousands of plugins, themes, and page builders allows users from all kinds of technical backgrounds to choose WordPress as their go-to CMS option. However, these easy ways to expand functionality can be a double-edged sword, especially with regards to performance.

In the following section, we review page builder adoption, along with improvements introduced by the Core Performance Team.

Page builders

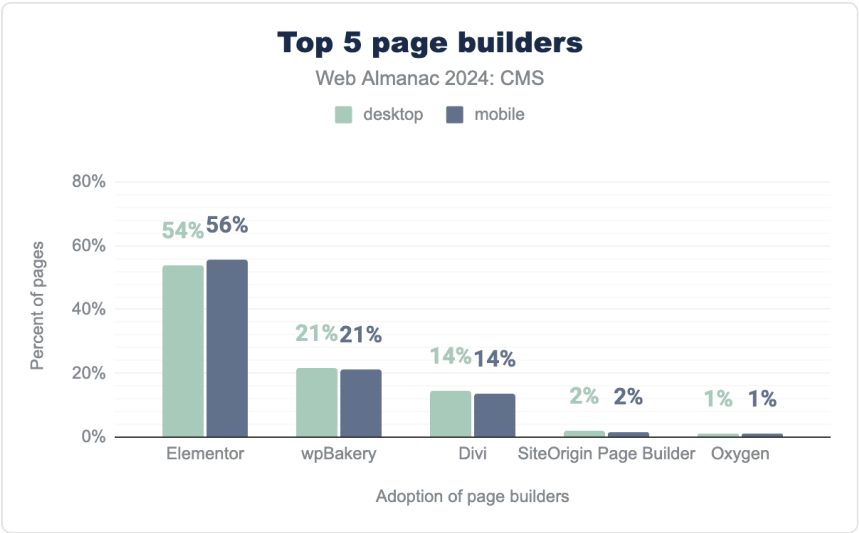


Figure 12.28. Top 5 page builders.

WordPress users often leverage a “page builder” that provides an interface within the CMS for content management. Thanks to the improvement of Wappalyzer’s detection methods, we can follow page builder adoption trends compared to our 2022 results.

We discovered that of the WordPress sites attributed to a page builder, Elementor and WPBakery remain the clear winners (13% and 12% increase for mobile, respectively), with Divi, SiteOrigin, and Oxygen trailing behind.

As we see it today, page builders can significantly influence a site’s performance. Historically, page builders have been anecdotal indicators of poor performance. As one example, our dataset indicates that it’s not uncommon for websites to have multiple page builders installed, significantly increasing the resources loaded by a site.

Thanks to native performance improvements and advancements, however, page builders tend to offer leaner alternatives, which are becoming increasingly popular among site owners. Paired with best practices when developing a website, they can now achieve better overall performance in WordPress, more of which we cover in the next section.

Latest performance improvements in WordPress

The WordPress Core Performance Team monitors, enhances, and promotes performance in WordPress core and its surrounding ecosystem. The group was established in 2021 to increase the performance of WordPress Core, which in turn improved the performance of the average WordPress site.

Fast forward to 2024, and it's safe to say the team has exceeded expectations, merging in a significant number of performance updates across each major release of WordPress.

Furthermore, since they first kicked off the project in November 2021, WordPress's overall Core Web Vitals pass rate has surged by 12%.

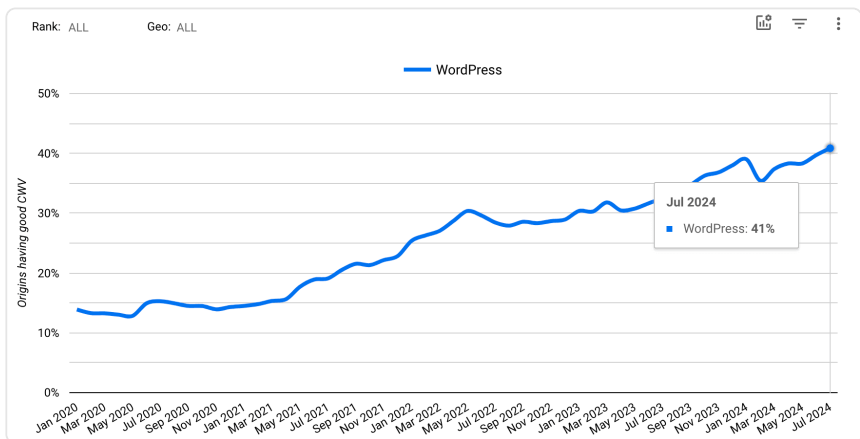


Figure 12.29. Trend of WordPress Core Web Vitals pass rates.

These are some of the enhancements that led to these results:

- **WordPress 6.0:** Improvements to the WordPress Caching API⁴⁶², taxonomy term queries⁴⁶³, and performance increase for sites with large user counts⁴⁶⁴

462. <https://make.wordpress.org/core/2022/04/29/caching-improvements-in-wordpress-6-0/>

463. <https://make.wordpress.org/core/2022/04/28/taxonomy-performance-improvements-in-wordpress-6-0/>

464. <https://make.wordpress.org/core/2022/05/02/performance-increase-for-sites-with-large-user-counts-now-also-available-on-single-site/>

- **WordPress 6.3:** More than 170 performance updates⁴⁶⁵ merged into the core, improvements in LCP and TTFB for block and classic themes, added support for the *fetchpriority="high"*⁴⁶⁶ attribute on images, introduced script loading strategies, which adds support for loading scripts with *defer* or *async*.
- **WordPress 6.4:** Merged over 100 performance updates and implemented script loading strategies⁴⁶⁷ for frontend scripts in core and bundled themes.
- **WordPress 6.5:** Introduced multiple performance improvements⁴⁶⁸, including support for the AVIF image format and a faster localization system.

The team also released the Performance Lab⁴⁶⁹ plugin, which is a collection of performance-related “feature projects” that may eventually be merged into the WordPress core software:

- Image Placeholders
- Modern Image Formats
- Performant Translations
- Embed Optimizer (experimental)
- Enhanced Responsive Images (experimental)
- Image Prioritizer (experimental)

Speculative Loading⁴⁷⁰ is another plugin that is part of the Performance Lab and has been recently made available. This plugin enables support for the Speculation Rules API⁴⁷¹, allowing the definition of rules to dynamically prefetch or prerender specific URLs based on user interactions. We discuss this API more in the next section. By default, it is set to prerender WordPress frontend URLs when a user hovers over a relevant link, allowing users to experience instant page load times.

Since its release, the plugin's adoption has steadily grown, reaching over 30,000 active installations as of this chapter's writing.

465. <https://make.wordpress.org/core/2023/08/07/wordpress-6-3-performance-improvements/>

466. <https://make.wordpress.org/core/2023/07/13/image-performance-enhancements-in-wordpress-6-3/>

467. <https://make.wordpress.org/core/2023/10/17/script-loading-changes-in-wordpress-6-4/>

468. <https://make.wordpress.org/core/2024/04/23/wordpress-6-5-performance-improvements/>

469. <https://wordpress.org/plugins/performance-lab/>

470. <https://wordpress.org/plugins/speculation-rules/>

471. https://developer.mozilla.org/docs/Web/API/Speculation_Rules_API

Emerging trends and technologies

Although we didn't have enough data regarding adoption and real-world impact to add to this year's edition, some emerging technologies promise to boost the overall performance of all CMS platforms and the web as a whole.

That's why we decided to include them still, keep an eye on them, and revisit their impact in 2025.

Speculation Rules API

The Speculation Rules API is a JSON-defined API developed by Google to enhance the performance of subsequent page navigation, leading to faster rendering times and improved user experiences. We already saw how the Speculative Loading WordPress plugin made use of this in the previous section.

This API enables developers to specify which URLs should be dynamically prefetched or prerendered:

- **Prefetching:** Fetches resources (like pages or assets) in the background before they are requested by the user, reducing load time when the user eventually navigates to the prefetched content.
- **Prerendering:** Fully renders a page in the background, so it is immediately available with no load time when the user navigates to it.

Furthermore, the API improves performance based on user behaviors, such as hovering over links or predicted navigation patterns, allowing content to load more quickly when needed.

To fine-tune when speculations should fire, developers can adjust the “eagerness” setting:

- **Eager:** Speculation rules are triggered immediately when conditions are met.
- **Moderate:** Speculation occurs after a short delay, such as when a user hovers over a link for at least 200 milliseconds, indicating some level of intent.
- **Conservative:** Speculation is triggered only with more definitive user actions, such as tapping or clicking on a link, signaling a higher likelihood of navigation.

These eagerness levels allow developers to balance performance optimization with resource management, ensuring that the browser preloads or prerenders content at appropriate times.

Long Animation Frames API (LoAF)

The Long Animation Frames API (LoAF) is designed to improve upon the Long Tasks API. It was shipped with Chrome 123 and provides developers with more actionable insights to address responsiveness issues and improve Interaction to Next Paint (INP).

Responsiveness refers to how quickly a page reacts to user interactions, ensuring updates are painted without delay. For INP, a response time under 200 milliseconds is ideal, though even shorter times may be necessary for animations.

Instead of measuring individual tasks, LoAF focuses on long animation frames, defined as frames that take longer than 50 milliseconds to render, helping identify blocking work more effectively.

Artificial Intelligence (AI)

Artificial Intelligence (AI) is reshaping how users build, manage, and optimize websites. AI-driven tools and plugins are becoming more prevalent, enabling automation, personalization, and enhanced user experiences.

Regarding the WordPress ecosystem, it seems like the community has yet to fully embrace the AI trend. In a thread⁴⁷² from May 2023, the core WordPress team and contributors exchanged opinions about the role of AI in the CMS.

Following the discussion, several highlights stand out:

- **AI should remain in the plugin space:** Since AI integrations currently rely on third-party systems and pricing, it's more likely to be adopted through plugins rather than Core—at least until AI models are fast enough to run directly on servers.
- **Developer Experience (DX) as a focus for innovation:** Some suggest that addressing current DX limitations, especially in the block editor, should be a priority. Enhancing extensibility could allow plugins more freedom to experiment with AI integrations.
- **AI for collaboration:** Others propose using AI to enhance collaboration and workflows, such as adding AI chatbots as a new user type as part of Phase 3 of the Gutenberg roadmap⁴⁷³. A bolder idea is integrating a personal AI assistant into the admin panel for business support.

472. <https://make.wordpress.org/core/2023/05/02/lets-talk-wordpress-core-artificial-intelligence/>

473. <https://en-au.wordpress.org/about/roadmap/>

WordPress core aside, AI has been adopted by numerous plugins that offer content generation, personalization, chatbots, and more. So, it's safe to say that AI has already started changing the WordPress ecosystem.

Conclusion

CMS platforms have continued their upward trajectory in 2024, becoming increasingly integral to the fabric of the internet as they support a diverse range of content creators, businesses, and users worldwide. With adoption rates steadily rising, CMSs are not only shaping how people create and manage content but also enhancing how users experience and engage with the web.

This year, the industry-wide focus on performance and user experience has deepened, with CMS platforms showing steady improvements across Core Web Vitals and Lighthouse scores. Many CMSs have embraced optimization strategies that enhance loading speed, interactivity, and accessibility, reflecting a shared commitment to a user-first web. The adoption of Interaction to Next Paint (INP) as a Core Web Vital has given us a more comprehensive measure of responsiveness, a new standard for page load times, and higher expectations for browsing experiences across devices and network conditions.

Challenges persist, however. As CMSs expand their capabilities and adopt new technologies, balancing added functionality with performance remains crucial. Issues such as page weight and JavaScript load continue to impact some platforms, especially on mobile, underscoring the importance of ongoing optimization and adherence to best practices.

Looking ahead, we're excited to see how CMSs will continue to evolve, with emerging technologies like AI transforming content creation and speculative loading improving performance. Although they're still in their early stages of development, these new technologies have the potential to redefine web performance and engagement. As we expand our datasets and refine our methodologies, we'll aim to provide an even clearer picture of the state of the web — and the CMS. Here's to a faster, more accessible, and user-friendly web—let's keep making it better together.

Authors



Jonathan Wold

 @sirjonathan  sirjonathan  <https://jonathanwold.com>

Jonathan Wold is an Open Web advocate with more than 20 years focused on the WordPress ecosystem. Beyond his love for WordPress and the Open Web, he enjoys reading widely, strategy games, acting, rock climbing, and occasionally writing in third-person.



Lora Raykova

 LoraRaykova  lraykova

Lora is a Content Manager at NitroPack with 8+ years of experience developing in-depth, specialized content for SaaS companies in the CEE region.



Niko Kaleev

 niko-kaleev  niko-kaleev-8760a9131

Niko is a Content Writer at NitroPack with 5+ years of experience in dissecting nuanced topics like hosting, Core Web Vitals, web performance metrics and optimizations.

Part III Chapter 13

Ecommerce



Written by Jonathan Pagel

Reviewed by Nurullah Demir

Analyzed by Jonathan Pagel

Edited by Niko Kaleev

Introduction

In this chapter, we review the state of ecommerce on the web. An ecommerce website is an “online store” that sells physical or digital products. When building your online store, there are several types of platforms to choose from:

1. **Software-as-a-Service (SaaS)** platforms like Shopify minimize the technical knowledge required to open and manage an online store. They do this by restricting access to the codebase and removing the need to worry about hosting.
2. **Platform-as-a-Service (PaaS)** solutions, such as Adobe Commerce (Magento), provide an optimized technology stack and hosting environment while allowing full codebase access.
3. **Self-hosted platforms**, such as WooCommerce.
4. **Headless platforms**, like CommerceTools, that are “API-as-a-service.” They provide the ecommerce backend as a SaaS, while the retailer is responsible for building and hosting the frontend experience.

Note that some platforms may fall into more than one of these categories. For example, Shopware offers SaaS, PaaS, and self-hosted options.

Our goal is to give an overview of the current state of the ecommerce ecosystem as well as its development in the last year. The first part focuses on ranking the platforms based on their usage and Core Web Vitals/Lighthouse performance results. The second part is more experimental. We checked different eCom technologies and how they have evolved over time.

Platform detection

We used an open-source tool called Wappalyzer⁴⁷⁴ to detect technologies used by websites. Wappalyzer can identify content management systems, ecommerce platforms, JavaScript frameworks and libraries, and more.

Limitations

Our methodology has some limitations that may affect its accuracy.

Limitations in recognizing ecommerce sites

Firstly, there are some limitations to our ability to recognize an ecommerce site:

- Wappalyzer must have detected an ecommerce platform.
- for this analysis we only analyzed home pages. If the ecommerce platform is hosted within a subdirectory it may be excluded from this analysis.
- A headless implementation reduces our ability to detect the platform in use. One of the primary methods to identify an ecommerce platform is by recognizing standard HTML or JavaScript components. Therefore, a headless website that does not use the ecommerce platform front end makes it challenging to detect.

Accuracy of metrics and commentary

The accuracy of metrics and commentary may also be affected by the following limitations:

474. <https://www.wappalyzer.com/>

- Trends observed may be influenced by changes in detection accuracy rather than reflecting industry trends. For example, an ecommerce platform may appear to become more popular simply because the detection method has improved.
- All website requests were made from the United States. If a website redirects based on geographic location, the final version analyzed will be the U.S.-specific one.
- We used the Chrome UX Report⁴⁷⁵ dataset, which includes only data from real-world Chrome sessions, not representing the user experience across all browsers.

Top ecommerce platforms

In total, we detected nearly 2.5 million websites built on ecommerce platforms in 2024, representing approximately 21% of all the websites analyzed. The most widely used ecommerce platform is WooCommerce, followed by Shopify and Squarespace.

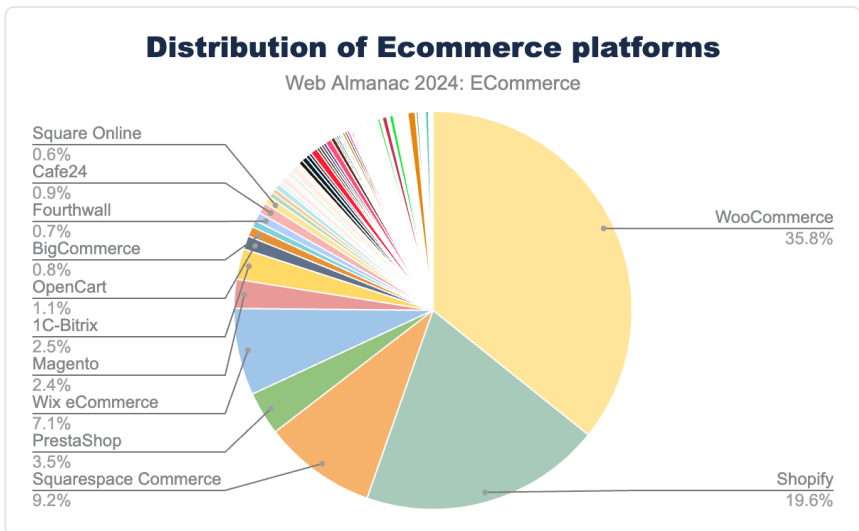


Figure 13.1. Distribution of Ecommerce platforms.

WooCommerce (36%) and Shopify (20%) dominate the ecommerce platform landscape. OpenCart is the last of the 362 detected shop systems that manage to secure a share above 1% of the market.

475. <https://developer.chrome.com/docs/crux>

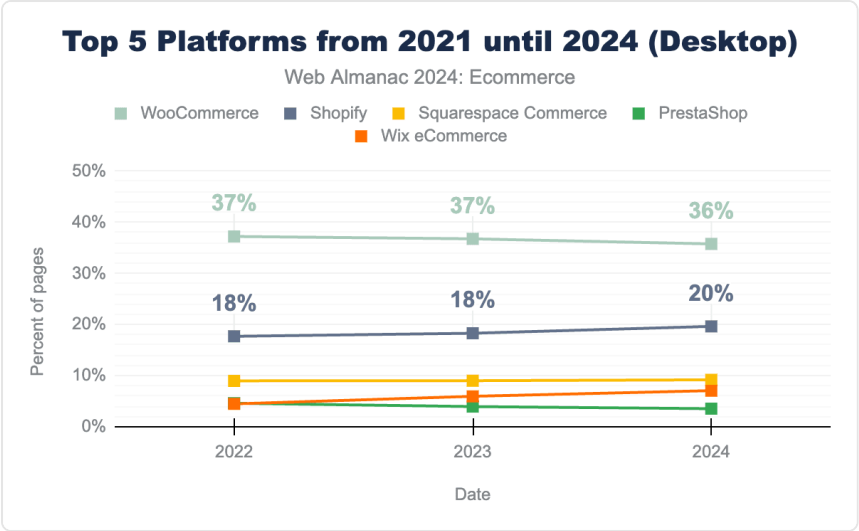


Figure 13.2. Top 5 Ecommerce Platforms from 2021 until 2024.

Over the years, the top five platforms have remained relatively consistent. However, Wix ecommerce surpassed PrestaShop in 2023, moving from 5th to 4th place. Trends indicate that the open-source project WooCommerce is slightly losing market share, decreasing from 37% in 2022 to 36% in 2024, while its commercial competitor, Shopify, is gaining market share in the same period (increasing from 18% to 20%).

Top ecommerce platforms by rank

Using the Chrome User Experience Report data, we looked at ecommerce platforms by rank.

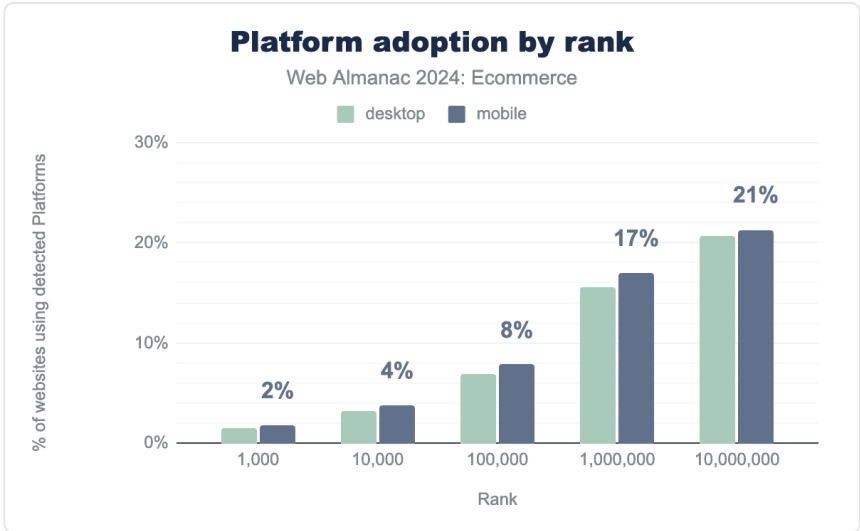


Figure 13.3. Platform adoption by rank for desktop and mobile.

Our data shows that only a few ecommerce platforms are represented in the top 1,000 websites, while about 20% of the top 10 million websites use an ecommerce platform. This difference could be because the top 1,000 sites often use custom solutions.

Position	10,000	100,000	1,000,000	10,000,000
1	Salesforce Commerce Cloud	Shopify	Shopify	WooCommerce
2	Fourthwall	Magento	WooCommerce	Shopify
3	Amazon Webstore	Salesforce Commerce Cloud	Magento	Squarespace Commerce
4	Magento	WooCommerce	PrestaShop	PrestaShop
5	SAP Commerce Cloud	Amazon Webstore	1C-Bitrix	Magento

Figure 13.4. Top ecommerce platforms by rank.

Compared to the overall web, there are noticeable differences in platform popularity among the top 10 million websites. For instance, Wix ecommerce loses its position in the top five platforms, while Magento joins the top five. In the top one million sites, Shopify overtakes WooCommerce as the most popular platform, while Squarespace and Wix ecommerce fall out of the top five and below the top 20.

In the top 100,000 websites, Salesforce Commerce Cloud and Amazon Webstore emerge among the most used platforms, with Shopify still holding the number one spot. Finally, in the top 10,000 websites, none of the previously leading platforms are represented in the top five, which are instead dominated by commercial solutions such as Fourthwall, SAP, and Salesforce Commerce Cloud.

Top ecommerce platform by geography

There are quite a few differences in preferences between geographies. We used additional data from the CrUX dataset, which categorizes the most visited websites per geographical area. For example, `google.com`, while an American website, is also one of the most visited websites by all German internet users.

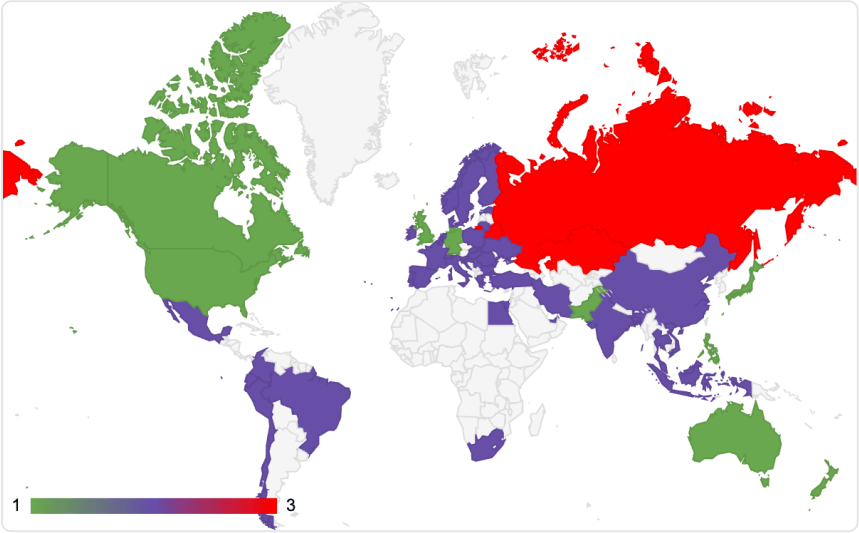


Figure 13.5. Top ecommerce Platform by Country.

We can see that three leading platforms take the top spot in each country: WooCommerce (violet), Shopify (green), and 1C-Bitrix (red). The map visualizes only these three due to the limitations of Google Sheets.

Core Web Vitals in ecommerce

Google's Core Web Vitals are three key performance metrics designed to evaluate crucial

aspects of user experience, focusing on loading speed, interactivity, and visual stability. Introduced in 2020 and adopted as a ranking signal in 2021, those metrics, among many others, determine how high a page is ranking in the Google search results.

Here's the percentage of sites on each platform that achieve a "good" score, meaning they meet the performance thresholds for all three Core Web Vitals: LCP (Largest Contentful Paint) under 2.5 seconds, INP (Interaction to Next Paint) under 200 milliseconds, and CLS (Cumulative Layout Shift) below 0.1.

- **LCP under 2.5 seconds:** This measures how long it takes for the largest visible element on the page to load. Achieving this threshold ensures users can view the main content quickly without excessive delays.
- **INP under 200 milliseconds:** This measures the time from a user's interaction (such as a click or tap) to the browser updating (or painting) the screen. A score under 200 milliseconds means the page is highly responsive and provides a smooth user experience.
- **CLS below 0.1:** This tracks the visual stability of the page by measuring how much elements move unexpectedly as they load. A score below 0.1 indicates minimal shifts, ensuring a more stable visual experience for use.

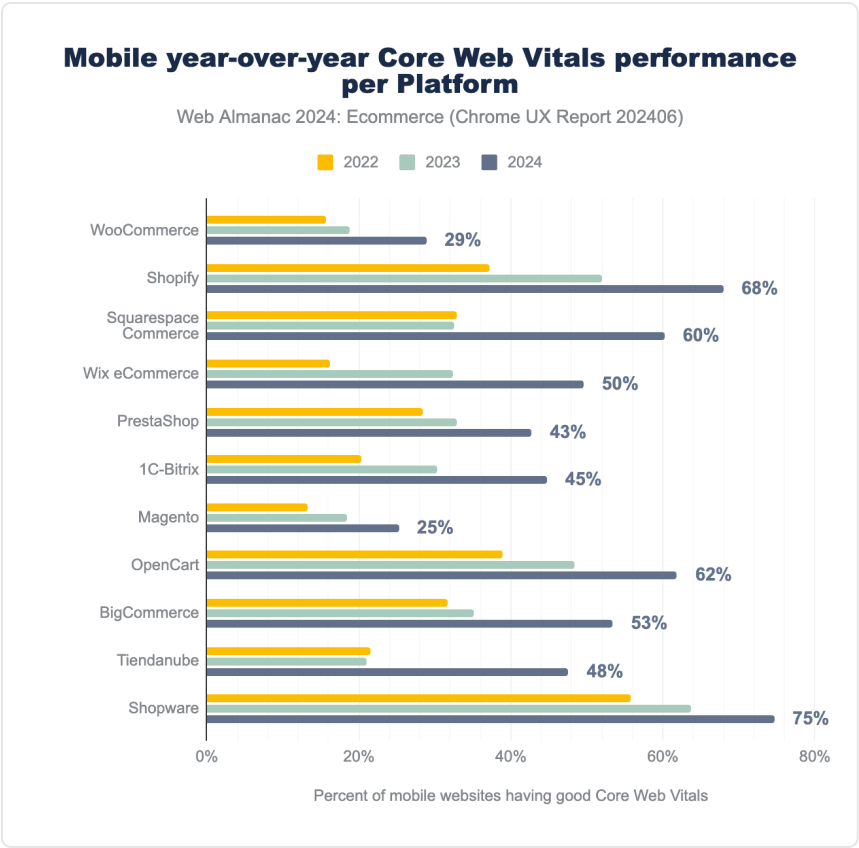


Figure 13.6. Mobile year-over-year Core Web Vitals performance per platform.

Some platforms made impressive improvements, like Squarespace, which increased from 33% good scores in 2022 to 60% in 2024. Other platforms, like Magento and WooCommerce, are still struggling with real-world user experiences. This chart, and the other charts of this section, consider mobile performance only since most web traffic comes from mobile, and it's more challenging to reach top scores.

Largest Contentful Paint (LCP)

Largest Contentful Paint (LCP) measures how long it takes for the main content of a webpage to load and become visible to users. It specifically tracks the loading time of the largest element on the screen, like a big image or a block of text, which makes it a good indicator of how quickly users can see something meaningful on your page.

A good LCP time should be under 2.5 seconds. If it takes too long, it can make the website feel slow, and users might leave. To improve LCP, you can optimize images, make server responses faster, and minimize blocking scripts so that key content shows up more quickly.

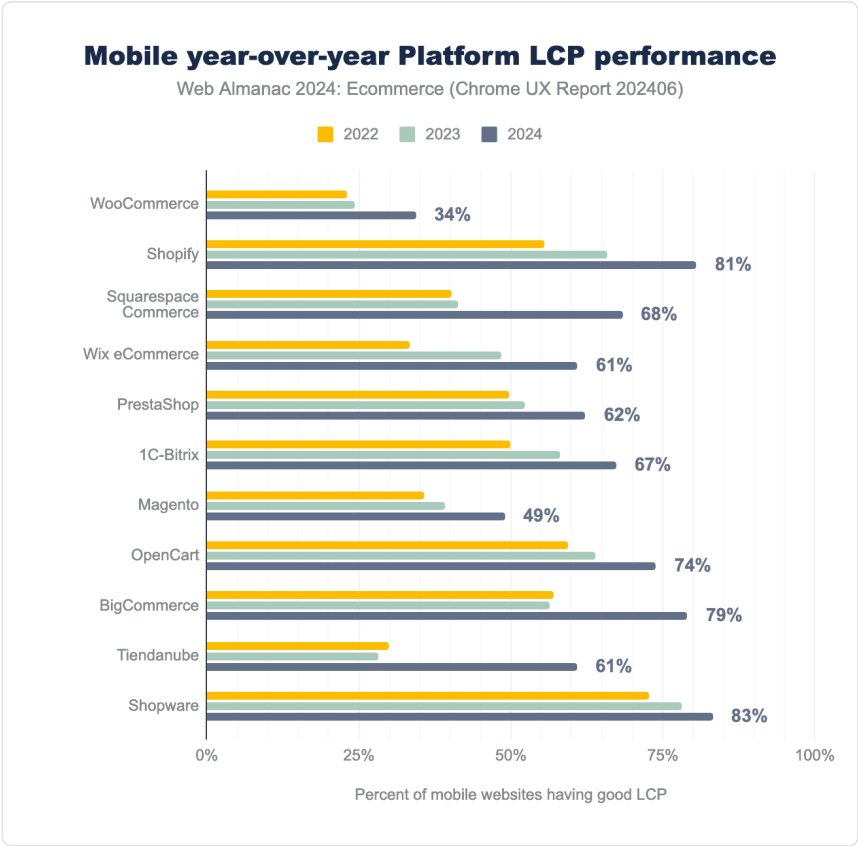


Figure 13.7. Mobile year-over-year Platform LCP performance.

Despite these challenges, the top 10 ecommerce platforms have shown significant year-over-year improvements in their LCP scores. Platforms like Shopify, OpenCart, and Shopware have consistently had good LCP pass rates since 2022, while Tiendanube, a popular platform in Argentina, made impressive progress, increasing its pass rate from 28% in 2022 to 61% in 2024. On the other hand, WooCommerce lags behind with a pass rate of just 34%.

Cumulative Layout Shift (CLS)

Cumulative Layout Shift (CLS) measures how stable the layout of a page is by tracking how

much content unexpectedly shifts as the page loads. A good CLS score means that 75% or more of a website's visits register a score of 0.1 or lower, indicating a stable, user-friendly experience.

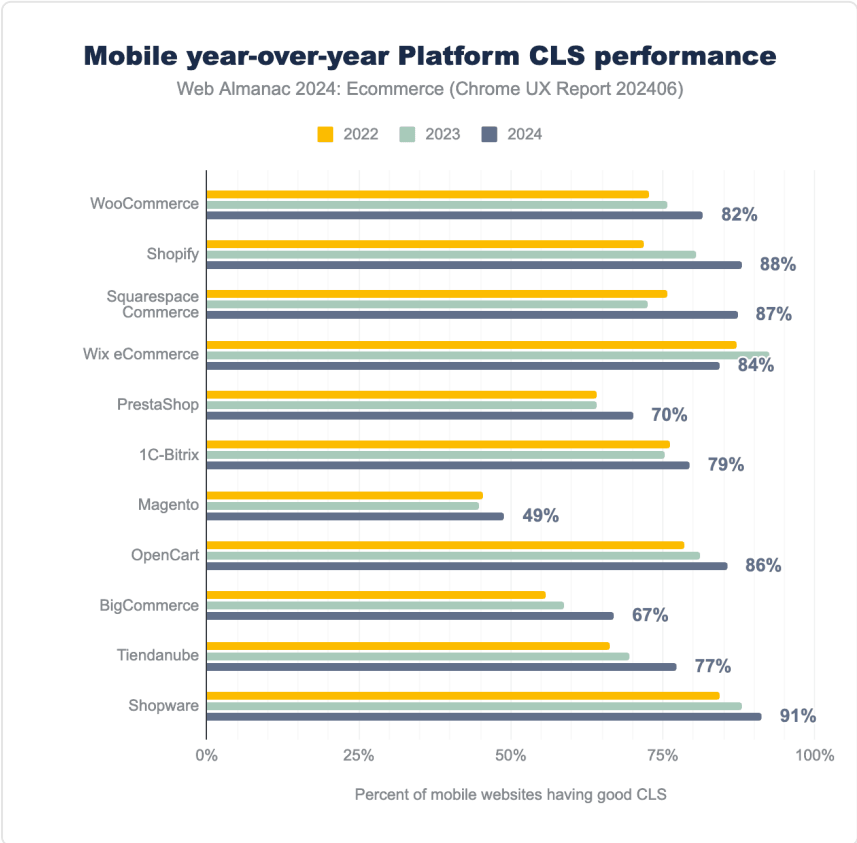


Figure 13.8. Mobile year-over-year Platform CLS performance.

In comparison to LCP, improvements in CLS have been less dramatic. Many platforms have only made modest progress, with Magento struggling more than others. WooCommerce, while facing challenges in other metrics, performs exceptionally well in CLS, similar to many other platforms.

Interaction to Next Paint (INP)

INP captures the time taken from the moment a user interacts with a page (e.g., clicking a button) to when the browser visually responds to that interaction. A good INP score is 200

milliseconds or less⁴⁷⁶, which ensures a smooth and responsive experience.

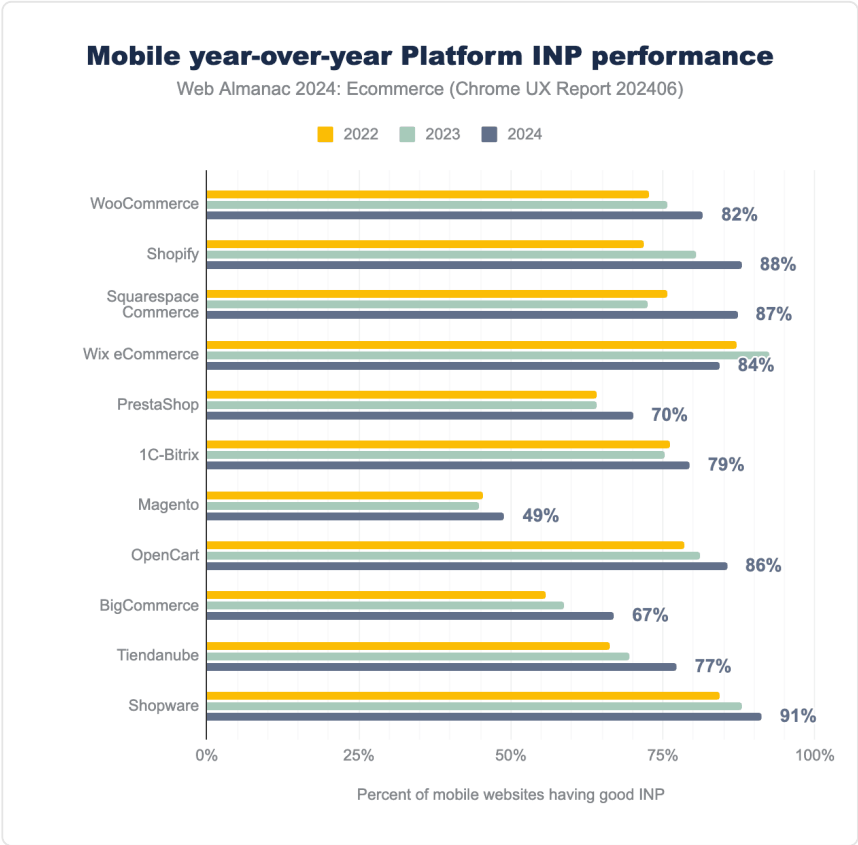


Figure 13.9. Mobile year-over-year Platform INP performance.

While most platforms have improved their INP scores, some still lag behind, including Magento and BigCommerce, with pass rates of 49% and 67%, respectively. However, most platforms have a pass rate above 75%, indicating substantial progress across the industry.

Lighthouse

Lighthouse⁴⁷⁷ is an open-source, automated tool for auditing web page quality. It provides metrics and reports on aspects like performance, accessibility, best practices, and search engine

476. <https://web.dev/articles/inp>

477. <https://developers.google.com/web/tools/lighthouse/>

optimization (SEO).

It generates reports with lab data that offer developers actionable suggestions for enhancing websites. However, it's important to note that Lighthouse scores do not directly impact the real-world field data collected by CrUX⁴⁷⁸. The HTTP Archive runs Lighthouse on mobile and desktop web pages, simulating a slow 4G connection with throttled CPU performance. Lighthouse Performance is a lab-based assessment of website performance tailored to specific test profiles for desktop and mobile users. To better understand the differences between both metrics, refer to this article⁴⁷⁹.

While Core Web Vitals offer real-world data, they provide a limited set of metrics. Lighthouse, on the other hand, gives us lab data for a wide variety of metrics. You can also run a Lighthouse test for this page using Chrome DevTools. In the following section, we focus on ecommerce systems that were detected more than 50,000 times to emphasize the systems people commonly use and recognize in our charts. However, the complete data is available in the sheets, and for exceptional cases, we also mention smaller shop systems.

Performance

The Lighthouse performance score⁴⁸⁰ is a metric that summarizes the overall performance of a web page, mainly focusing on how quickly and smoothly it loads and becomes usable for visitors. This score ranges from 0 to 100, where higher scores indicate better performance.

The Lighthouse performance score is a weighted average of the five metric scores—First Contentful Paint (10%), Speed Index (10%), Largest Contentful Paint (25%), Total Blocking Time (30%), Cumulative Layout Shift (25%).

478. <https://developers.google.com/web/tools/chrome-user-experience-report>

479. <https://web.dev/articles/lab-and-field-data-differences>

480. <https://developer.chrome.com/docs/lighthouse/performance/performance-scoring>

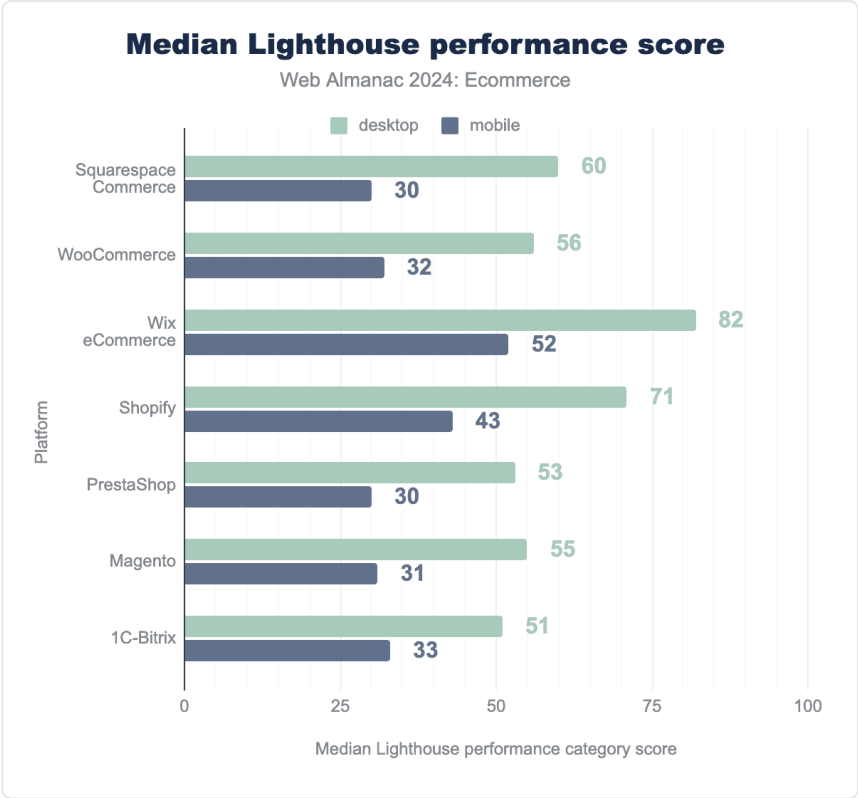


Figure 13.10. Median Lighthouse performance score for various ecommerce platforms on desktop and mobile.

We can see that Wix ecommerce performs very well on both desktop and mobile compared to other systems. This is surprising, as the ranking data shows that it is primarily used for websites outside the top 10 million, which are likely less professional stores. This performance could also be due to the limited customization options compared to open-source systems like WooCommerce.

If we lower the threshold to platforms that appear at least 5,000 times instead of 50,000 times, Gumroad scores very well, with a median score of 87 on desktop and 59 on mobile. Additionally, Argentina's most popular shop system, Tiendanube, also scores well, with 74 on desktop and 58 on mobile.

Accessibility

Accessibility is an increasingly important topic, not just for moral reasons but also for legal reasons. Lighthouse uses the Axe framework⁴⁸¹ to determine the accessibility score. The accessibility score shows how easy it is for everyone, including people with disabilities, to use your website. The score ranges from 0 to 100, with a higher number meaning the website is easier for more people to use.

The score is based on a series of tests. For example, it checks if images have descriptions for people using screen readers and if buttons are clearly labeled. It also looks for issues like proper use of headings and good color contrast to ensure the website is understandable and easy to navigate. We also have an entire chapter dedicated to accessibility that goes into more detail.

It's important to note that a good accessibility score does not necessarily mean the website is fully accessible, as many problems cannot be detected by automated tools. For example, an incorrect description of an image is an issue, but Lighthouse won't flag it.

481. <https://github.com/dequelabs/axe-core/blob/develop/doc/rule-descriptions.md>

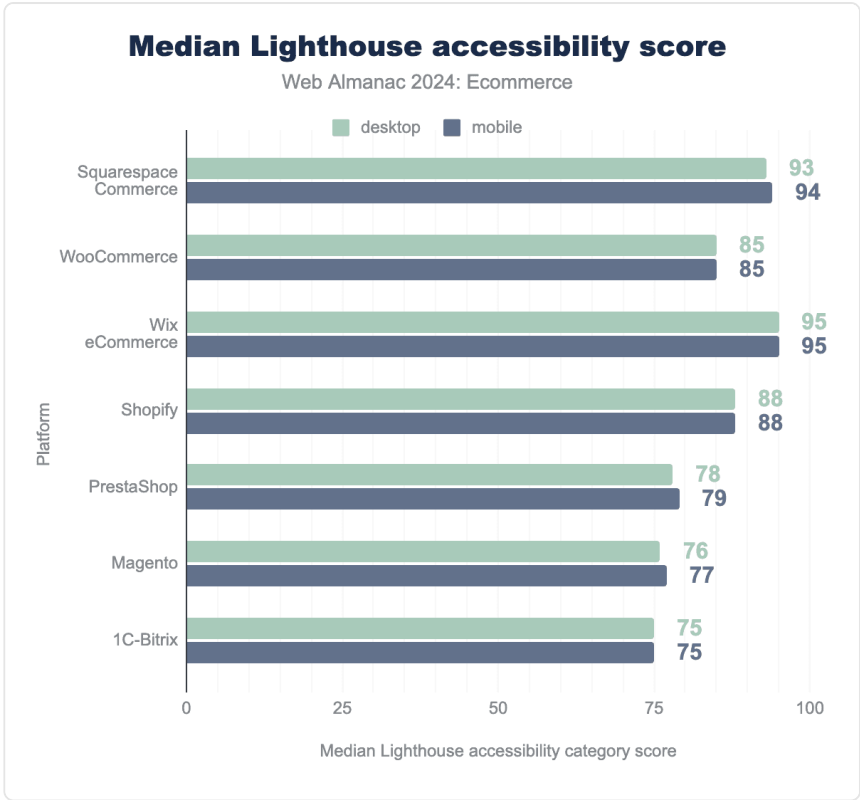


Figure 13.11. Median Lighthouse accessibility score for various ecommerce platforms on desktop and mobile.

The commercial platforms Wix, Squarespace, and Shopify perform well, while none of the systems have a notable bad result.

SEO

The Lighthouse SEO score shows how well a website is set up to be found by search engines like Google. The score ranges from 0 to 100, with a higher score meaning a site is better optimized for search engines, which helps more people find it. The test checks for meta description, title, and correct headings structure.

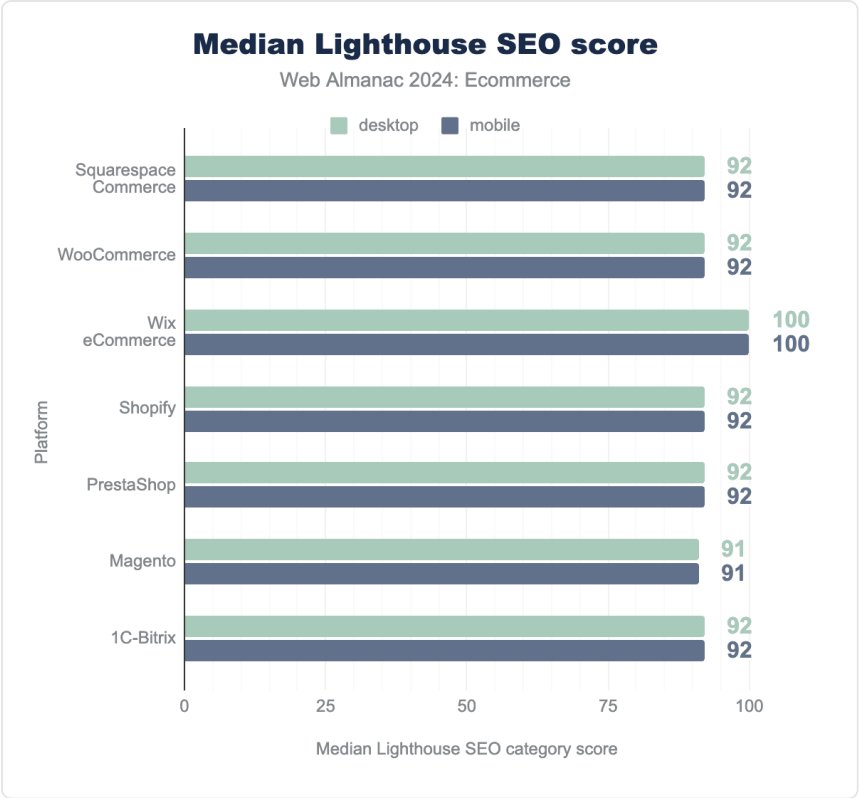


Figure 13.12. Median Lighthouse SEO score for various ecommerce platforms on desktop and mobile.

The importance of SEO is reflected in the lighthouse results since most platforms score really well, led by Wix, which has a perfect median score of 100.

Best practices

The Best practices score shows how well your website follows general rules for good web development, focusing on making the site secure and reliable. It checks things like whether your site uses secure HTTPS connections, if JavaScript features are used safely, and if images and resources load properly. The goal is to ensure your website avoids common problems that could make it less secure, slow, or unreliable for users.

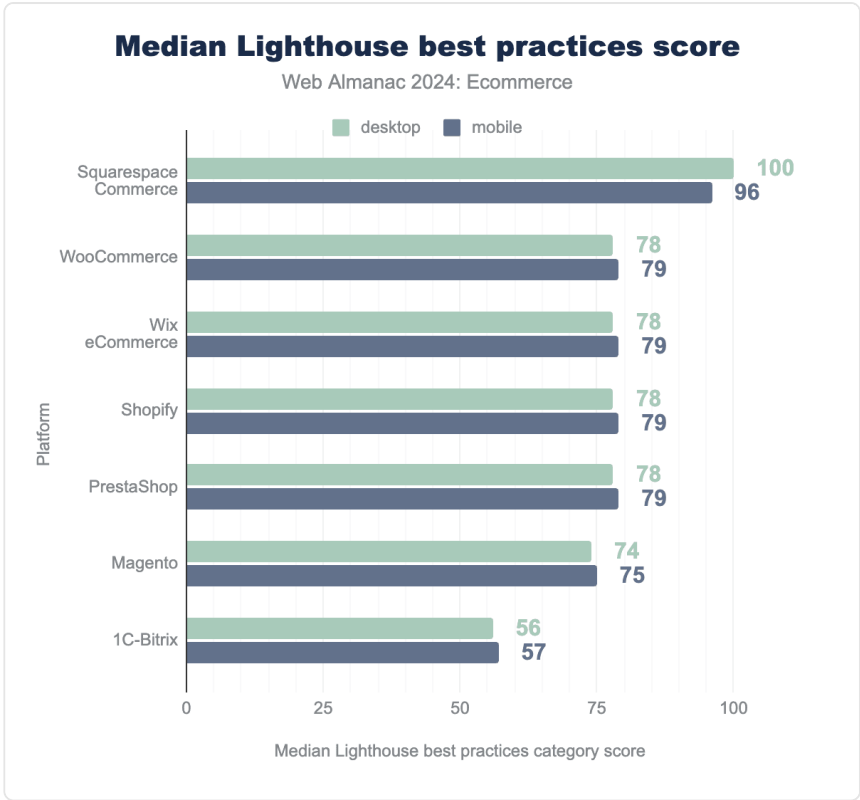


Figure 13.13. Median Lighthouse best practices score for various ecommerce platforms on desktop and mobile.

Squarespace is leading by far among platforms with more than 50,000 pages in our dataset, with a 22-point difference over the second place, WooCommerce.

Payment providers

The detection of payment providers is not as advanced or precise as the detection of different ecommerce platforms. This may be because we only analyzed the home page rather than the entire website, which might not provide clear indications of the payment providers used. You can refer to the checkpoint on the Wappalyzer GitHub⁴⁸² profile to understand how different payment providers are detected. Multiple payment providers can be used on the same website. In such cases, the website is counted for each payment provider detected.

482. <https://github.com/HTTPArchive/wappalyzer>

The following section focuses on each website with detected payment providers, even if no ecommerce system was identified.

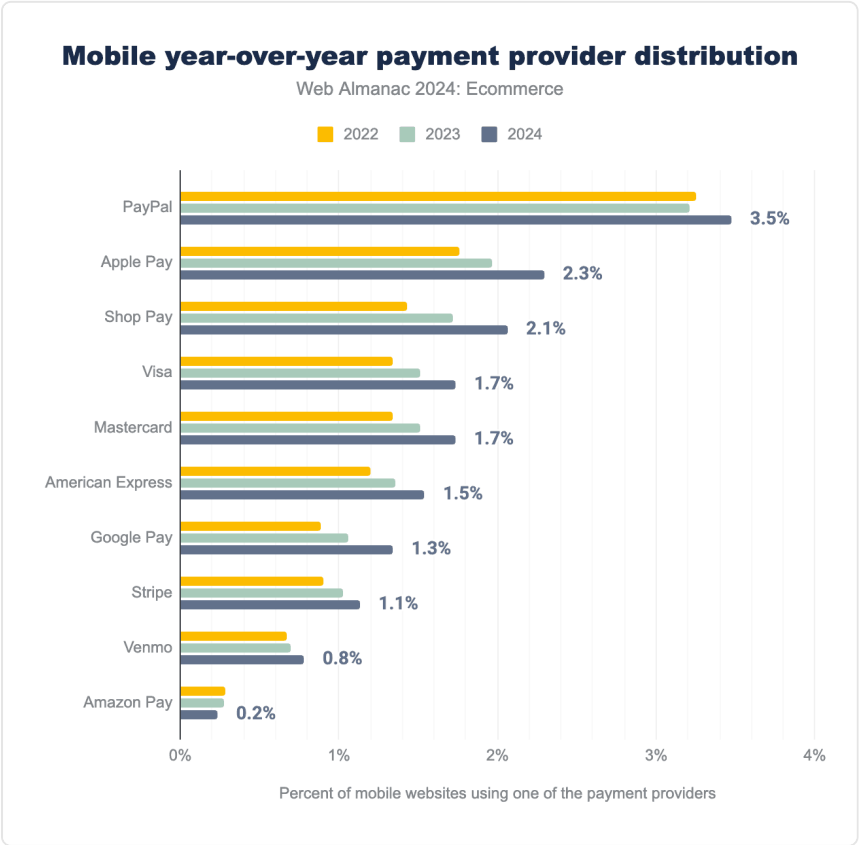


Figure 13.14. Mobile year-over-year payment provider distribution.

The data reveals that PayPal is the most commonly detected payment method on mobile websites, appearing on 3.5% of all pages in the dataset. This means PayPal was found on approximately 560,000 mobile pages out of more than 16 million analyzed.

Apple Pay ranks second, being detected more frequently than Google Pay, which shows its growing presence in mobile ecommerce. Meanwhile, Shop Pay, a payment solution provided by Shopify, secures third place in the rankings.

Conclusion

Ecommerce is still evolving, with platform preferences varying by region and website size. While WooCommerce remains the go-to platform for many, Shopify has steadily gained ground, especially among higher-traffic websites. Interestingly, platforms like Wix ecommerce perform well in terms of user experience metrics despite being more popular with smaller sites. Overall, we can observe improvements in most metrics, from performance to accessibility, over the past few years, benefiting everyone.

While ecommerce platforms are diverse and well distributed among different providers, a few key players dominate technologies like payment systems. It will be interesting to see how this landscape continues to evolve in the coming years.

Author



Jonathan Pagel

 [jcmpagel](#)  [jonathan-pagel](#)  <https://jonathanpagel.com>

Jonathan Pagel studied ecommerce in his bachelor's degree and has since been interested in the field, particularly in the areas of speed optimization and accessibility for shops and websites. Currently, he is freelancing in this field and pursuing a Master's in AI and Society.

Part III Chapter 14

Jamstack



Written by Mike Neumegen

Reviewed by Thom Krupa

Analyzed by Estela Franco

Edited by David Large

Introduction

Remember Jamstack? While the term itself may not be widely used anymore, the core concepts remain as relevant as ever.

At its heart, Jamstack is a delivery architecture that promotes prerendering pages whenever possible. For the 2024 Web Almanac Jamstack chapter, we'll explore the three main delivery architectures commonly used today:

1. **Prerendered:** Build pages ahead of time and serve them as static files.
2. **Hybrid:** mix of static and dynamic pages, using technologies like Incremental Static Regeneration (ISR) and Edge Functions to blur the lines between them.
3. **Dynamic:** Pages are generated on the server for each request.

It's become more common to take a **Static-first** approach as the more static a website is, the faster, more scalable, secure, and eco-friendly it tends to be. Prerendering, hybrid, and dynamic

approaches are valuable tools in a web developer's toolkit and are each suited to different situations. Let's see how these categories are evolving.

Defining the dataset

So, how do we determine which delivery approach a site is taking? It would be ideal if there were a clear marker to neatly classify every site in one of our categories. Unfortunately, there is no such marker and it's one of the biggest challenges of reporting on these architectures.

Previous Jamstack Web Almanac chapter have taken two approaches:

1. Focus on websites with a detectable Static Site Generator. This gave us an accurate picture as we could be sure these sites were using targeted technologies. The problem is that many Static Site Generators don't leave a fingerprint by default, so the dataset skews toward frameworks like Next.js and Nuxt that do. To complicate things further, these heavier JavaScript frameworks are capable of both prerendering pages and dynamically rendering them, so which category do we put them in?
2. The second approach, pioneered in the 2022 Jamstack chapter⁴⁸³ used thresholds for Largest Contentful Paint⁴⁸⁴, Cumulative Layout Shift⁴⁸⁵, and Caching to build the dataset. This approach broadens the dataset by including more prerendered sites that don't have a clear Static Site Generator fingerprint. The downside is that it blurs the definition of prerendering by focusing on frontend performance. Prerendering will often improve delivery speed, but that's where the scope ends in our view. What happens after delivery like a faster LCP is a happy byproduct.

Both approaches have merit: the first uses data that clearly defines a prerendered site, while the second relies on indicators. This year, we're combining the two with a scoring system to get the best of both worlds. We developed this scoring system to reflect our confidence level in whether a site is prerendered. Strong indicators, like being generated by 11ty or hosted on GitHub Pages, earn 100 points, while softer signals, such as a fast Time to First Byte (TTFB), receive lower points—50 in this case.

We then total up the points and put them in one of the following categories:

483. <https://almanac.httparchive.org/en/2022/jamstack>

484. <https://web.dev/articles/lcp>

485. <https://web.dev/articles/cls>

Prerendered

If a site scores 100 points or over, we call it Prerendered. It means the site is statically generated or shows multiple signs of being decoupled. This is a high bar and contains ~0.5% of total sites in 2024.

Hybrid

If a site scores between 50 and 99 points, we put it in the Hybrid category. These are websites that might be static but don't have enough fingerprints for us to tell, or they may not be purely static but share some shared approaches and philosophies. For example, a Nuxt site hosted on Netlify or a Gatsby site with fast TTFB. While Hybrid is a more lenient category, it's still a high bar representing ~5% of sites in the data set in 2024.

Dynamic

Anything scoring below 50 points falls into the Dynamic category. These sites either rely on server-side rendering or don't have enough fingerprints to put in the other categories. Dynamic represents the majority of the web and ~94.5% of the dataset.

Scoring

Let's dive into how scoring works.

Detectable Static Site Generator

This is a huge tell. Based on the SSG, we can give a score on how likely the website is prerendered. These are ballparks so while we can't confidently say that 30% of Next.js sites are purely static, it's one data point that when combined with others, gives us confidence.

SSG	Points
Astro	50
Bridgetown	100
Docusarus	100
Eleventy	100
Gatsby	30
Gridsome	100
Hexo	100
Hugo	100
Jekyll	100
Mintlify	70
Next.js	30
Nextra	70
Nuxt	30
Octopress	100
Pelican	100
Retype	100
Scully	70
VuePress	100

Figure 14.1. SSG Points Distribution.

Hosting Providers

GitHub Pages can only serve static content so we know if it's hosted on that platform, it must be prerendered. Netlify and Vercel both have many static websites, but also have dynamic functionality. We've tried to account for this in the scoring.

Hosting Provider	Points
GitHub Pages	100
Netlify	30
Tiiny Host	100
Vercel	30

Figure 14.2. Hosting Provider Points Distribution.

TTFB

Time to First Byte⁴⁸⁶ is how long the server took to return anything to the browser. Our thinking behind using this metric is that if all the server needs to do is return a static file, the TTFB will be very fast, whereas if the server needs to do dynamic processing to generate the page, it will take longer. Of course TTFB can be gamed if the server sends *something* while it takes its time generating the page, which is why we're giving a good TTFB a score of 50 and not 100.

Server Response Time	Points
Less than or equal to 800ms	50
More than 800ms and less than or equal to 1800ms	25
More than 1800ms	0

Figure 14.3. Server Response Time Points Distribution.

Caching

Caching gives us a good sense of how static a site might be. If a site uses aggressive caching headers, it's a strong sign that the server is doing minimal work, which feels very static-like. On the other hand, sites with long but less aggressive headers still suggest a static approach, but with less certainty.

486. <https://web.dev/articles/ttfb>

Cache header	Points
max - age of two weeks or older and does not require revalidation.	100
max - age of two weeks or older and requires revalidation.	50

Figure 14.4. Cache Header Points Distribution.

ETag

ETags help with cache validation by letting the browser check if a resource has changed since it was last fetched. We see this as a small nudge toward a static approach since it shows the server is leaning on cached content instead of regenerating it every time.

Etag	Points
Etag present	10
No ETag	0

Figure 14.5. Etag Points Distribution.

Set - Cookie

A `Set - Cookie` header nudges toward dynamic behavior. It typically means the server is handling things like sessions which suggests the content isn't purely static.

Set Cookie	Points
Set-Cookie present	-10
No Set-Cookie	0

Figure 14.6. Set Cookie Points Distribution.

Limitations in the dataset

It's worth reiterating, this segmentation is hard. These are rough estimates based on the data available and some miscategorization is inevitable. It's also the first year we've used this scoring method and is something we can refine in years to come.

To minimize this, we've set a high bar for what qualifies as Prerendered and Hybrid. It's likely

these categories are larger than represented in this report.

Analysis

Now that our categories are defined, let's see how they're trending in 2024.

Static Site Generators

SSGs are the technology that performs the prerendering, which makes them a natural starting point for our analysis. Keep in mind not all Static Site Generators leave a fingerprint. Tools like 11ty, for example, don't leave any trace by default, so they aren't reflected in this dataset.

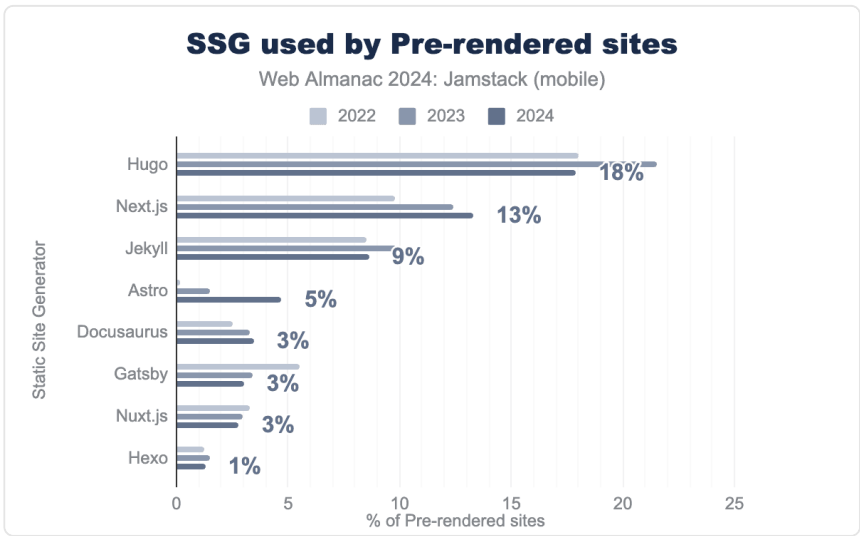


Figure 14.7. Static Site Generator usage for Prerendered sites.

We see Hugo leading the Prerendering category by a sizable margin for the past 3 years. Hugo is known for being a fast and flexible SSG with a community that values performance so it's not surprising to see it at the top of the list.

Next.js continues its dominance on the web and is steadily closing the gap on Hugo in the Prerendering category. It's worth noting that with our scoring, we're considering almost all known Hugo sites as Prerendered, whereas Next.js needs multiple indicators to be considered Prerendered.

Meanwhile, Jekyll, Gatsby, and Hexo have seen stable or declining usage in the Prerendered

category, which reflects the reduced development going into these frameworks.

Docusaurus is on a steady rise which is impressive considering it's only used for documentation.

The real stand out here is Astro though which grew 3x in 2024! It's now a major player in the Prerendered category and could become the dominant framework if this trend continues.

Let's take a look at the Hybrid category:

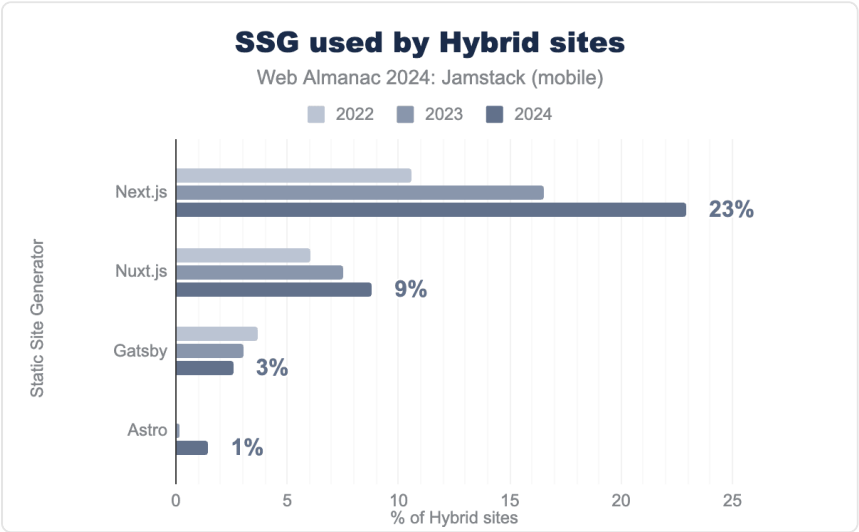


Figure 14.8. Static Site Generator usage for Hybrid sites.

We see Next.js extending its lead in the Hybrid category with a 39% increase in the past year. Next.js is a highly flexible framework capable of Prerendering, Dynamic Rendering, and Incremental Static Regeneration (updating specific static pages on-demand without requiring a full site rebuild). Its versatility for everything from an information website to a fully-fledged web application is one of the big reasons we're seeing its growing usage.

Nuxt continued to grow steadily within the Hybrid category, while Astro saw rapid growth in 2024.

Performance

Prerendered sites are known for their performance because the server can respond instantly rather than dynamically rendering a response for each request. This results in faster load times and less strain on the server. Let's see how that plays out in our analysis.

The median site page transfer size is an interesting place to start as we would expect it to be a significant factor in overall site performance.

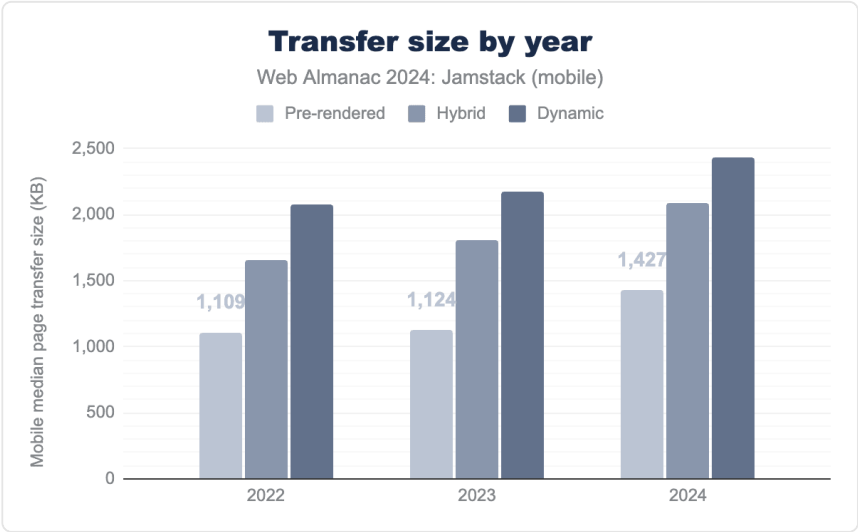


Figure 14.9. Transfer size by year.

On average, Prerendered sites have much smaller transfer sizes coming in at 43% of the Dynamic sites.

The jump between Prerendered and Hybrid sites likely reflects their different use cases with Prerendered sites more likely to be static, informational sites, while Hybrid sites might lean more into eCommerce and web applications which can lead to larger transfer sizes.

Let's break down where this extra weight is coming from.

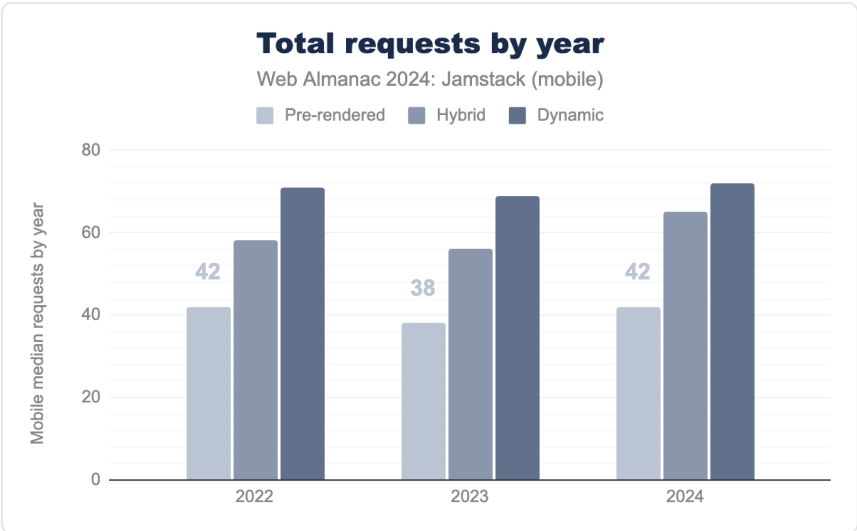


Figure 14.10. Total requests by year.

Looking at the total number of requests could explain some of the differences in total page weight between categories. From 2022 to 2024 we see the total requests stay about the same for Prerendered sites, a slight increase with Dynamic, and the largest increase in the Hybrid category (16% in 2024).

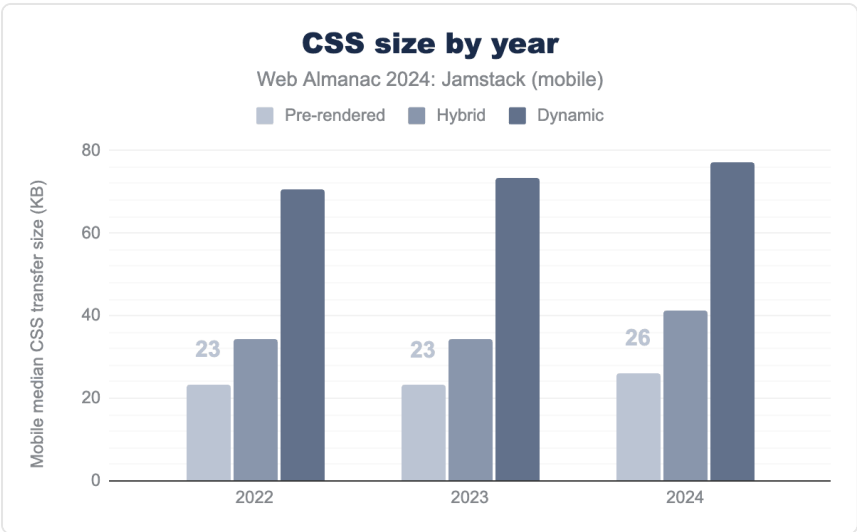


Figure 14.11. CSS size by year.

The total transfer of CSS provides another perspective and shows a slight upward trend across all categories. The notable jump between Hybrid and Dynamic sites stands out. While further analysis is needed to pinpoint the exact cause, a likely explanation is that Prerendered and Hybrid sites tend to be hand-crafted by web developers. In contrast, Dynamic sites often rely on themes, website builders, and plugins which are notorious for having bloated web assets.

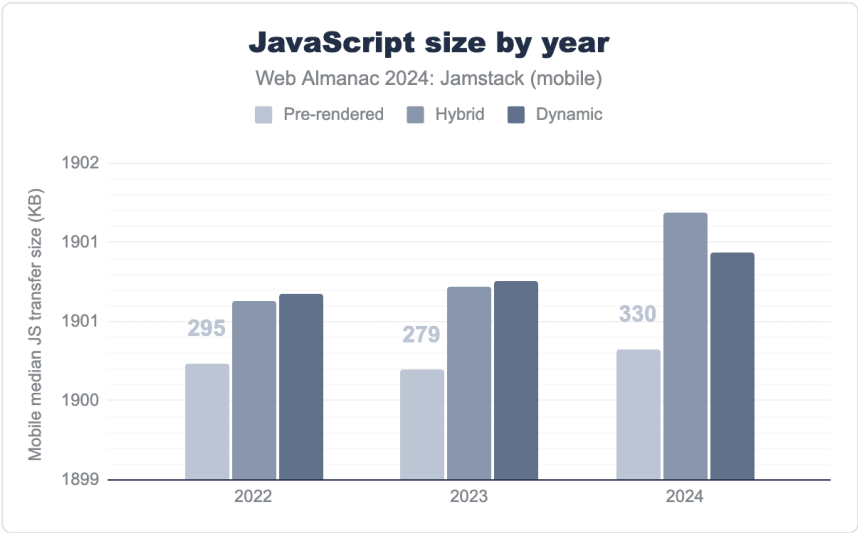


Figure 14.12. JavaScript size by year.

The total transfer of JavaScript offers one of the most interesting insights with a sharp increase in JavaScript size for Hybrid sites, surpassing that of the Dynamic category in 2024.

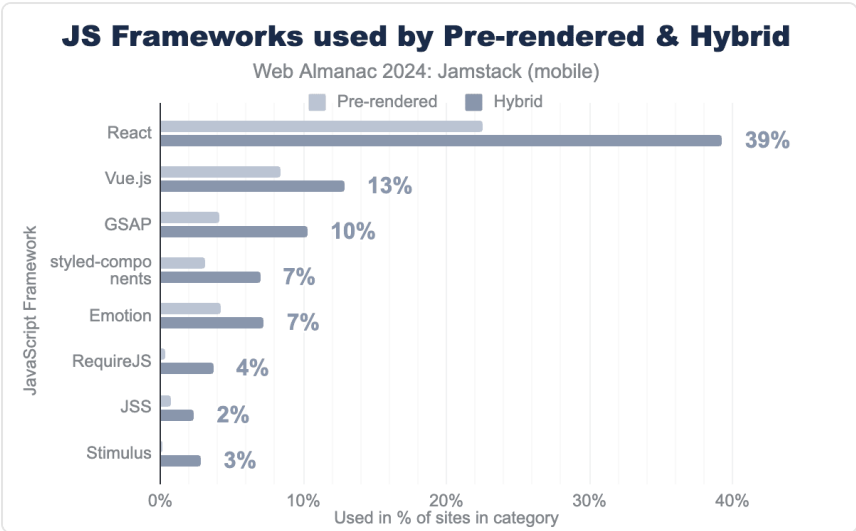


Figure 14.13. JavaScript Frameworks used by Prerendered and Hybrid sites.

When we break down the JavaScript frameworks used by Prerendered and Hybrid websites, we see heavier frameworks like React are more commonly used in the Hybrid category.

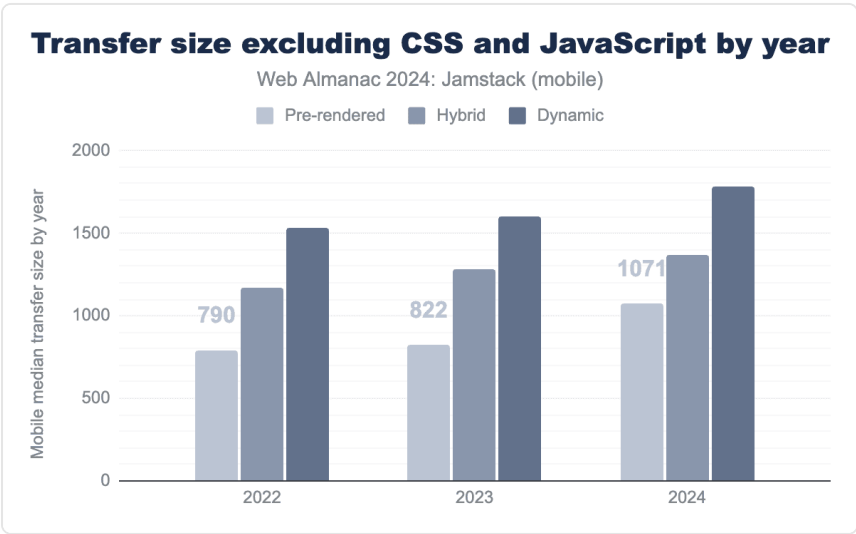


Figure 14.14. Transfer size excluding CSS and JavaScript by year.

If we remove CSS and JavaScript from the total transfer size, we're left with HTML, images,

fonts, and other media. When we compare this filtered total median transfer size, there’s still a noticeable gap between each category. This corresponds to the total number of requests made in each category, which likely reflects a higher usage of images and fonts in those categories.

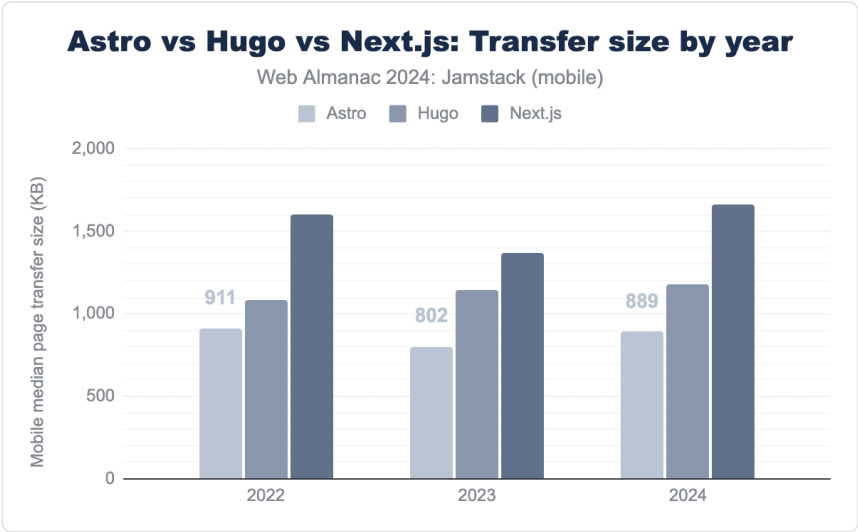


Figure 14.15. Astro vs Hugo vs Next.js: Transfer size by year.

Focusing on the three standout Static Site Generators in the Prerendered category—Astro, Hugo, and Next.js—gives us another lens to analyze page weight. Note: this analysis only includes sites in the Prerendered category to keep the comparison fair.

Astro takes numerous steps to ship only the minimal data required, from Astro Islands and zero JavaScript by default to an asset optimization pipeline. It’s great to see its dedication to performance reflected in the data.

Hugo sees a step up from Astro in page weight. Hugo has many of the same types of asset optimization pipeline as Astro, though the Astro pipeline is more deeply integrated into the framework, which could explain the difference.

Next.js shows a sizable increase in page weight. Next.js ships with its bundled runtime for routing and hydration, as well as the React library, both of which contribute to a higher page weight baseline.

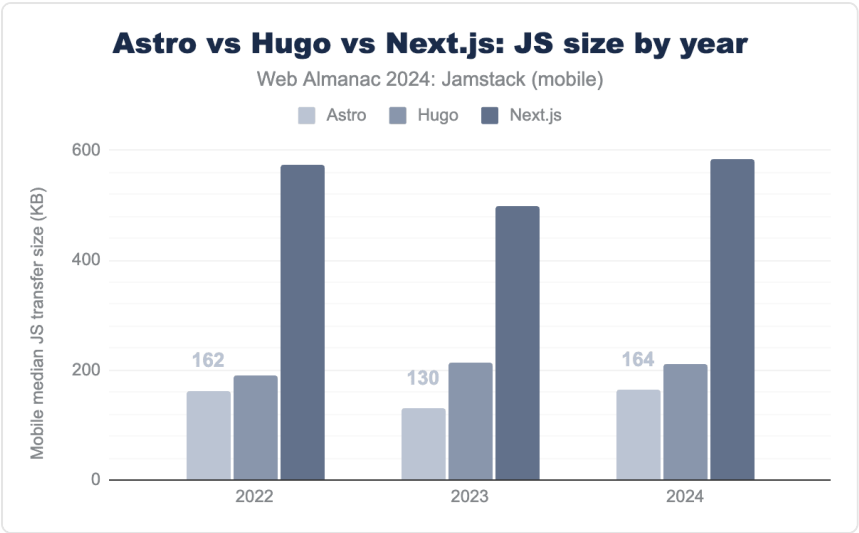


Figure 14.16. Astro vs Hugo vs Next.js: JavaScript size by year.

Breaking this down to purely the JavaScript shipped, we can see how heavy the JavaScript bundle is for Next.js coming in at 3.5x larger than Astro's.

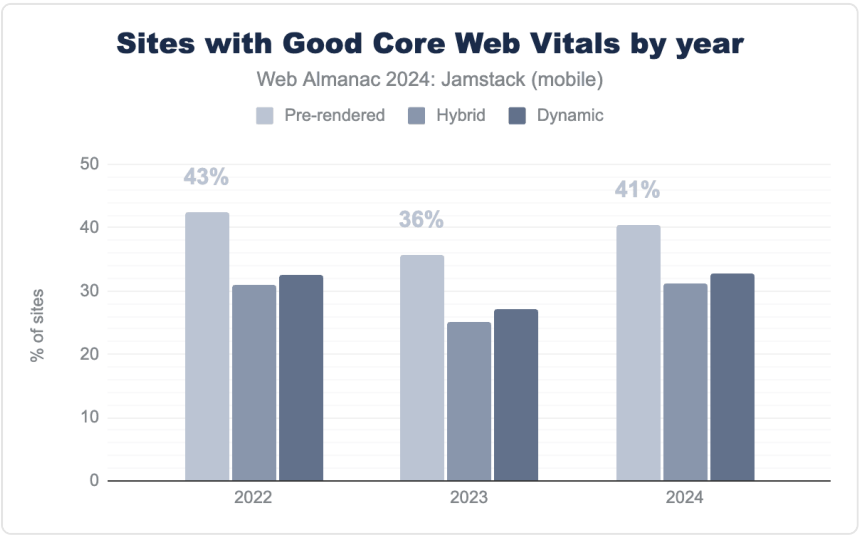


Figure 14.17. Sites with Good Core Web Vitals by year.

Let's see how the delivery approaches impact Core Web Vitals. We use Google's definition of

having Good Core Web Vitals⁴⁸⁷ where a site has:

- **Largest Contentful Paint (LCP)** under 2.5 seconds for 75% of users
- **Cumulative Layout Shift (CLS)** less than 0.1 for 75% of users
- **Interaction to Next Paint (INP)** under 200 milliseconds for 75% of users

Prerendering typically results in a faster Time to First Byte (TTFB), which will naturally improve LCP: the faster the browser receives the HTML, the sooner it can start fetching assets and render the page. These sites also tend to receive more attention to details like CLS, as they are often handcrafted by developers. Hybrid has a heavy use of JavaScript frameworks and largest JavaScript payloads, which likely explains it having the lowest percentage of Good Core Web Vitals.

Growth

So, how are these architectures being adopted across the web?

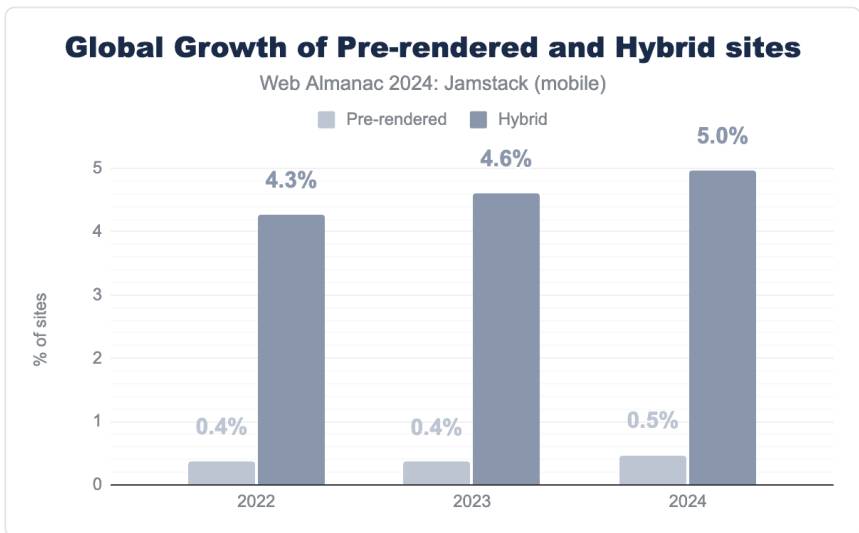


Figure 14.18. Global Growth of Prerendered and Hybrid sites.

Looking at the full dataset, we see that while Prerendered and Hybrid architectures are on the rise, they're still relatively niche compared to the rest of the web. This makes sense, as most

487. <https://web.dev/articles/vitals#core-web-vitals>

websites rely on website builders or GUIs, rather than the developer-focused tools common in the Prerendered and Hybrid categories.

We see more growth if we zoom into the sites with the most traffic:

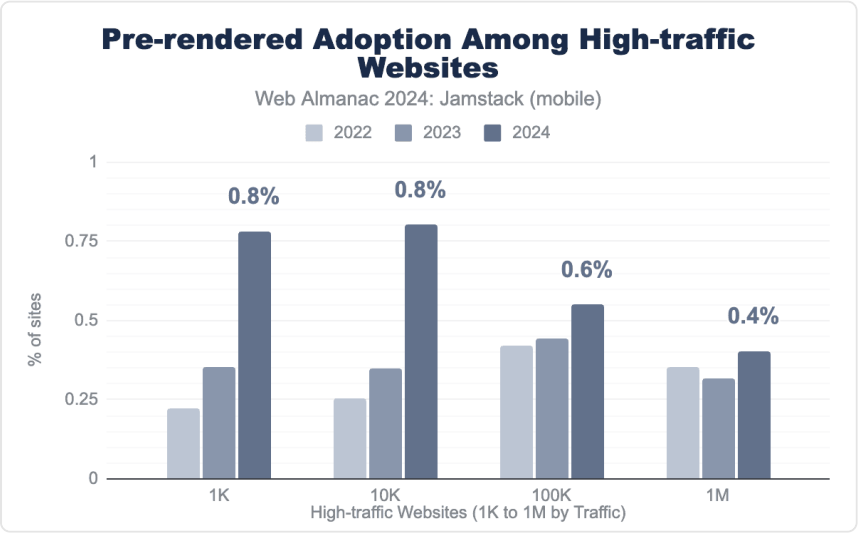


Figure 14.19. Prerendered Adoption Among High-traffic Websites.

There's been significant growth in Prerendered adoption among the top 1k and 10k most popular sites with growth leveling off at the 1M mark and beyond. These high-traffic sites care a great deal about performance, SEO, security, and stability, principles that align perfectly with the Prerendered approach.

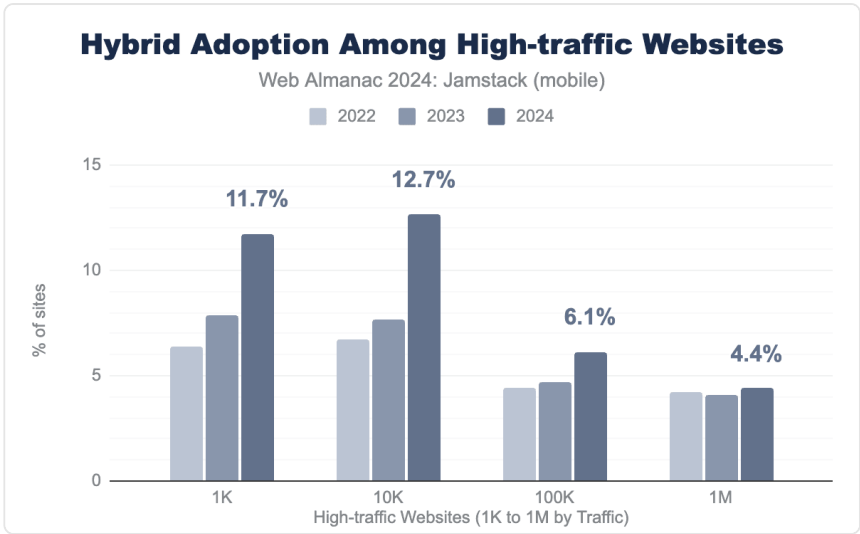


Figure 14.20. Hybrid Adoption Among High-traffic Websites.

It's a similar story with Hybrid which now powers over 12% of the most popular 10k websites.

Conclusion

The standout this year is the combined 67% growth of Prerendered and Hybrid architectures in the top 10k high-traffic websites in 2024—over 12% of these popular sites now use these approaches. While Prerendering and Hybrid delivery remains niche in the context of the entire web, it's rapidly gaining traction where its benefits are most valued.

Given how much of the web is inherently static, this trend is promising. By adopting a static-first model and limiting dynamic rendering to dynamic content, the web could not only be faster, and lighter weight, but also significantly more environmentally friendly.

Hugo continues to lead as the top Static Site Generator in the Prerendering space, with Next.js gaining ground. Astro saw the largest growth of any Prerendering framework in 2024, an impressive feat considering it's one of the newest frameworks in this space.

The growing presence of Next.js and Astro in both Prerendered and Hybrid categories signals a shift toward hybrid architectures that give developers more control over static and dynamic content generation. Astro's focus on improving performance and reducing page weight looks to be paying off and has made serious inroads in catching up on Next.js.

While the term Jamstack may no longer be widely used, its evolution continues to shape the future in powering top-tier websites, paving the way for a faster, more stable, secure, and eco-friendly web.

Author



Mike Neumegen

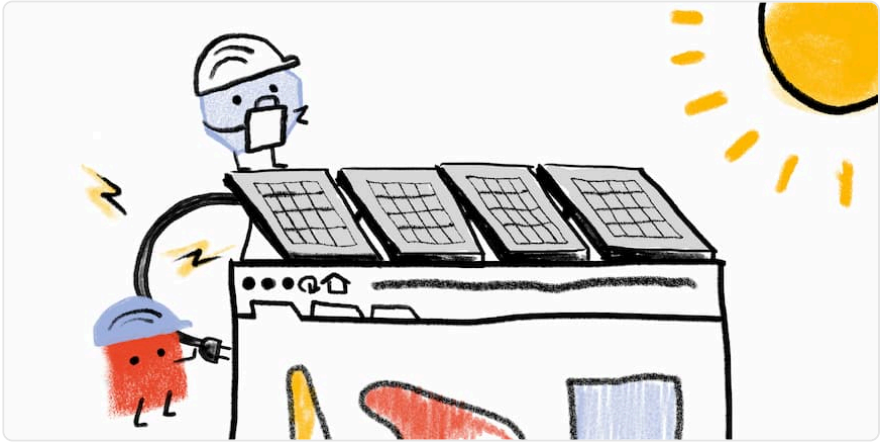
✉ @mikeneumegen 🌐 @https://fosstodon.org/@mikeneu 🔊 mneumegen 🔗 mneumegen
🌐 <https://mikeneumegen.com/>

Mike Neumegen passionate about building a web that's accessible, fast, simple, and secure and grounded in strong fundamentals. He's the co-founder of CloudCannon⁴⁸⁸, a content management platform that brings Git-based workflows to content editors.

488. <https://cloudcannon.com/>

Part III Chapter 15

Sustainability



Written by Laurent Devernay Satyagraha, Burak Güneli, Ines Akrap, Alexander Dawson, Mike Gifford, and Tim Frick

Reviewed by Rafael Bonalume Lebre

Analyzed by Lucia Harcegova, Burak Güneli, and Mike Gifford

Edited by Caleb Queern

Introduction

In the current chapter, we will rely as much as possible on the 2022 Sustainability chapter⁴⁸⁹. If you haven't yet, you should read it right now. Yes, recycling/reusing is a major part of sustainability.

Since the 2022 Almanac, the field of digital sustainability has advanced considerably. That said, it is very much in its infancy, as this is a complex problem. We cannot know, with absolute certainty, what the full effects of our digital lives are on our physical planet. What we can be confident in is that the full impacts are bigger than generally accounted for. Water, land, rare minerals, and electricity are all consumed by our “clean” digital interfaces, and toxic waste is often produced.

In addition, it has been recognized by many working in sustainability that the impact of other

489. <https://almanac.httparchive.org/en/2022/sustainability>

primary fields such as web performance, accessibility, security, privacy, etc. can lead to emissions as a secondary consequence of poor planning and decision-making.

Since the last Almanac, we have seen the creation of open standards, open source software, and open data on the impacts of our digital lives. Although still in development, we have the beginning of an evidence-based approach to emission reductions. There are courses, certifications, books, videos, podcasts, and conferences on the subject⁴⁹⁰. Yet, this is still an issue that takes backstage with other efforts to reduce emissions.

We need to get better at reducing the impacts of our data centers and start seeing digital devices as part of a circular economy. But more importantly, we need to begin reducing our demand for more and bigger digital experiences.

Where possible, we will be leveraging the Draft W3C Sustainable Web Community Group Web Sustainability Guidelines (WSG). We will be building on the experience of the 2022 Almanac to highlight data to give us a snapshot of where we are today.

But first, here's a quick recap of what happened in the sustainability field since 2022.

What's new in web sustainability?

Since 2022, we have seen a steep increase in awareness of web sustainability. Along the way, a lot has happened.

The Software Carbon Intensity (SCI) Specification⁴⁹¹ has become an ISO standard⁴⁹². Based on the rate of carbon emissions, this specification is a way to track the environmental efficiency of a digital service. This could be a good starting point. However, it could be important to go further through additional ecological factors (depletion of abiotic resources, water usage, etc) and consider all steps of the lifecycle of a project to reach for frugality and moderation.

This quest for efficiency saw some other achievements such as carbon-aware code⁴⁹³ and major cloud providers aiming for a reduction of their environmental impacts, mostly based on carbon emissions. Sustainability also met some drawbacks⁴⁹⁴ because of the rise of artificial intelligence, which in itself gets massive media coverage but not necessarily for its environmental impacts. This emergence hindered the efforts of some major cloud providers to reach for the reduction of environmental impacts. This helps illustrate that efficiency is not enough and frugality should be our top priority. It also highlights that widely adopting new technologies without considering their environmental (and social) impacts should be avoided.

490. <https://w3c.github.io/sustyweb/intro.html#resources>

491. <https://sci.greensoftware.foundation/>

492. <https://greensoftware.foundation/articles/sci-specification-achieves-iso-standard-status>

493. <https://hackernoon.com/carbon-aware-computing-next-green-breakthrough-or-new-greenwashing>

494. <https://thenewstack.io/sustainability-how-did-amazon-azure-google-perform-in-2023/>

This is where we should mention new repositories for best practices:

- The W3C (World Wide Web Consortium) Sustainable Web Community Group published the Web Sustainability Guidelines (WSG) 1.0⁴⁹⁵. These offer a higher perspective on web sustainability and should help teams adopt sustainability on a larger scale (more on this later in this chapter).
- France institutions also released the General policy framework for the ecodesign of digital services⁴⁹⁶. The purpose here is to offer a framework for sustainable digital services and to aim for a wider adoption of these best practices.
- An ISO standard for Digital Services Ecodesign⁴⁹⁷ is also on the way.

More and more books⁴⁹⁸ are also being published, such as Building Green Software⁴⁹⁹ by Anne Currie, Sarah Hsu and Sara Bergman.

In addition to this, tools for estimating the environmental impacts of the web are still evolving and new ones keep appearing. Some existing tools (such as Screaming Frog SEO and Webpagetest) are adding features to estimate environmental impacts. As such, the Sustainable Web Design Model⁵⁰⁰ is often used. However, accurately estimating impacts is still an important topic and no consensus has been reached yet. As is often the case with environmental considerations, the topic remains complex.

To give more context to all these breakthroughs, more general studies about the environmental impacts of digital should be conducted worldwide, as is already the case in France⁵⁰¹. Such a large perspective would help estimate the benefits of all the ongoing efforts but also give insights on where more focus is needed.

Limitations and hypothesis

There are many ways to assess environmental impacts, but there are a few things to keep in mind:

- Most free tools only rely on transferred data, the number of HTTP requests, and DOM size. This is insufficient to capture the overconsumption related to animations, heavy calculations (especially using JS) but also benefits from dark mode. To achieve this, other metrics are needed, such as CPU/GPU, Energy, and

495. <https://w3c.github.io/sustyweb/>

496. <https://en.arcep.fr/news/press-releases/view/n/environnement-rge-sn-170524.html>

497. <https://www.iso.org/standard/86105.html>

498. <https://w3c.github.io/sustyweb/intro.html#books>

499. <https://www.oreilly.com/library/view/building-green-software/9781098150617/>

500. <https://sustainablewebdesign.org/estimating-digital-emissions/>

501. <https://en.arcep.fr/news/press-releases/view/n/the-environnement-210324.html>

RAM/memory usage should be measured.

- Tools usually stick to page loading and sometimes scrolling. This is not always relevant to real usage and often misses some obvious things, such as accepting cookies, client-side cache, playing videos, etc. Real user journeys should be measured, based on analytics and user feedback.
- GHG is often used as a proxy for environmental impacts, but this is not enough. To avoid impact transfers and greenwashing, other indicators should be used, as stated by the ADEME⁵⁰² (PDF, French, 980 kB).

These notions are detailed in this article about the environmental impacts of web elements⁵⁰³

Embedded environmental impact needs to be factored into our calculations. The web should not require users to update their devices every 2-3 years. We need to ensure that the lifespans of our digital infrastructure are extended.

Intersectional environmental issues

Sustainability ultimately involves people, as the planet will ultimately take care of itself. Building a just society that supports the needs of its people within the boundaries of its population is key. Throughout our lives, all of us require accommodation. Our ability to navigate the physical and digital world changes, and a sustainable economy supports that. One in five people has a permanent disability, and everyone will have both temporary and situational disabilities throughout their life. For more on this, you should read the Accessibility chapter.

In building sustainable digital interfaces, we must consider the user, including those with disabilities. Sustainable digital interfaces allow users to quickly navigate to accomplish their tasks. An inaccessible interface may ultimately require a user to take a less sustainable path to accomplish their goals.

Similarly, one must consider human diversity. Race, class, gender, and sexual identity have been used to divide people. A sustainable web should promote climate justice.

For further information, refer to:

- The WSG 1.0 - SC 3.5.

502. https://bibliothèque.ademe.fr/ged/7595/R_f_rentiel_rcp_services_num_riques.pdf

503. <https://greenspector.com/en/reduce-the-weight-of-a-web-page-which-elements-have-the-greatest-impact/>

Legal obligations and reporting standards

Digital sustainability is becoming a greater concern in the regulatory landscape. Recent legislative changes worldwide have introduced requirements that everyone must comply with.

Where organizations once faced minimal obligations, they are now confronted with new regulations designed to start accounting for their environmental impact. The urgency of the climate crisis has accelerated this shift, with governments increasingly holding businesses accountable for their ecological footprint.

The European Union currently leads the world in terms of sustainability regulation. The Corporate Sustainability Reporting Directive (CSRD) and the European Sustainability Reporting Standards (ESRSs) mandate that large corporations report on Scope 1, 2, and 3 emissions annually, encompassing direct, indirect, and value chain emissions. Scope 1 covers direct greenhouse gas (GHG) emissions made by a business, Scope 2 covers indirect emissions from a business such as electricity to heat or cool a building, and Scope 3 covers all other emissions associated, not with the company directly, but that the organization is indirectly responsible for.

Meanwhile, the Ecodesign for Sustainable Products Regulation (ESPR) and Energy Efficiency Directive focus on mandating physical infrastructure (such as data centers) and products towards circular economies and general sustainability principles. Furthermore, the Corporate Sustainability Due Diligence Directive (CS3D) mandates that organizations foster sustainable behavior relating to workplaces and employees.

Additionally, The Green Claims Directive explicitly legislates around claims of sustainability or related terms (such as being green) to fight against greenwashing. The Digital Operational Resilience Act (DORA) mandates the sustainability of operations relating to DevOps activities. The EU Artificial Intelligence (AI) Act provides necessary regulation around Artificial Intelligence and the environmental impact of such technologies.

It should be noted that many of these European Union laws apply to both citizens and residents of the EU. This also applies to companies who do business with Members of the EU or with people living within the EU. Like the GDPR, the ramifications of these laws have global significance.

Two best practice repositories from the EU known as the Handbook of Sustainable Design of Digital Services (GR491) and the European Digital Rights and Principles have become widely associated with their legislative move towards sustainability.

France established itself as an early leader in digital sustainability through the publication of its REEN legislation alongside the General Policy Framework for the Ecodesign of Digital Services (RGESN). This framework provides clear and actionable guidance on best practices, setting a

high standard for sustainable digital service design. There is a huge focus on training engineers on how to build more energy-efficient digital products/services. With its circular economy anti-waste law AGECE also in effect, and its other best practice AFNOR Spec 2201 also available, the country is working hard to tackle environmental issues.

In Germany, progress is also being made regarding sustainability through the development of the Energy Efficiency Act (EnEeG) which mandates that data centers use less energy on the grounds of climate protection, and the Supply Chain and Due Diligence Act (SCDDA) legislating on the human side of sustainability providing legal protections across supply chains.

California's new emissions reporting law, the Climate Corporate Data Accountability Act (S.B. 253) along with its sister legislation Climate-Related Financial Risk Act (CFRA), are a landmark regulation package with potential global ramifications. They will require companies with revenues exceeding \$1 billion and business operations in California to disclose their greenhouse gas emissions. Given California's role as a global hub for digital technology, these laws will likely influence practices and standards internationally, making it essential for digital agencies worldwide to take note.

India is the final country of note that has crafted sustainability legislation, in this case, the Business Responsibility and Sustainability Report (BRSR) which is much like the EU's CSRD, except for those living or trading in this nation, enforcing reporting scope emissions.

Much like web accessibility, digital sustainability is becoming a non-negotiable aspect of doing business. Non-compliance with the emerging sustainability standards is no longer an option, and businesses face a range of potential penalties.

The regulations affecting digital sustainability are diverse. Some, like the CSRD, focus on enhancing transparency and accountability in emissions reporting, while others aim to prevent greenwashing by enforcing accurate and responsible communication about products and services.

Standards such as the Global Reporting Initiative (GRI) provide essential frameworks to help digital agencies meet these new obligations. It may be helpful to think of laws to dictate what must be done, standards to provide the roadmap for compliance, and best practices to ensure that compliance is achieved effectively and ethically.

Digital professionals should look at what legislation, standards, and best practices for digital sustainability apply to them prior to the creation of their product or service and monitor any developments as their product evolves (to meet an evolving compliance landscape).

The benefits of adopting sustainable practices from the offset extend beyond mere compliance; they offer a competitive edge in a market increasingly driven by the need for excellent customer experience and increasing environmental considerations and can lead to time and

cost savings when not having to be retrofitted into existing works. As this report outlines, integrating sustainability into your business strategy is not just about avoiding penalties, but about seizing new opportunities in a rapidly evolving landscape.

Organizations are beginning to add a /sustainability page to the footer of their webpage to allow them to report on their CO2 emissions as well as describe what they are doing to reduce emissions for their digital services.

For further information on specific regulations and standards, refer to:

- WSG: Web Sustainability Laws & Policies⁵⁰⁴
- WSG: Draft Sustainable Tooling And Reporting (STAR) 1.0⁵⁰⁵
- GRI: Global Reporting Initiative⁵⁰⁶
- GR491, The Handbook of sustainable design of digital services | ISIT⁵⁰⁷
- WERF: Web Ecodesign Reference Framework⁵⁰⁸
- Apolitical: Keeping tech sustainable⁵⁰⁹

Evaluating the environmental impact of websites

Evaluating the environmental impact of a website is everything but an easy task. It usually involves assessing the energy consumption and carbon footprint associated with its operation, from the servers that host it to the devices that access it. This evaluation considers various factors, including the efficiency of the hosting infrastructure, the amount of data transferred during page loads, the use of renewable versus non-renewable energy by data centers, and the overall optimization of the website's code and assets. Websites that are heavy in images, videos, and other large files require more energy to load and transmit, contributing to higher carbon emissions. By understanding and optimizing these elements, website owners can reduce their digital carbon footprint, contributing to a more sustainable web and helping to minimize their environmental impact.

It is very important to keep in mind that there is still no available tool that would allow for precise measurement of every single part of the system, allowing for a nice overall sum. Therefore, we can measure isolated parts of the systems our teams work on or resources to the

504. <https://w3c.github.io/sustyweb/policies.html>

505. <https://w3c.github.io/sustyweb/star.html>

506. <https://www.globalreporting.org/how-to-use-the-gri-standards/gri-standards-english-language/>

507. <https://gr491.isit-europe.org/en/>

508. <https://github.com/cnumr/best-practices/blob/main/README.en.md>

509. <https://apolitical.co/solution-articles/en/keeping-tech-sustainable>

use of the models that allow us to do an estimation based on specific inputs. One model that is currently used in many different website carbon calculators and is being worked on and improved is the Sustainable Web Design Model⁵¹⁰. This model can be a slightly controversial way of estimating, as well as any currently out there. However, it has been present for a while, allowing comparison over time, and simplifying this complex process allows for better understanding and, hopefully, recognition of the problem itself.

Usually, we talk about CO₂ but it would be more accurate to talk about eqCO₂ or CO₂e: values equivalent to CO₂ emissions but rather applies to all kinds of greenhouse gasses (GHG). Also, other environmental impacts should be considered (as is the case in Life Cycle Assessment for instance) to avoid impact transfers (e.g. when reducing GHG emissions proves detrimental to water consumption) and debatable claims such as “carbon neutrality”.

For further information, refer to:

- The WSG 1.0 - SC 2.1, 2.25, and 5.5.

A quick note on assessing environmental impacts

The queries from last year have been updated to reflect the new data structure. For most, there has been no effective difference in the results.

The update of the CO₂ emissions calculations is reflected in the global page, SSG, CMS, and e-commerce emissions calculation. We have updated the calculations to the version 4 model of emissions calculation based on Sustainable Web Design Model⁵¹¹.

Main differences:

- The global carbon intensity has changed from 442g/kWh to 494g/kWh.
- Previously the website CO₂ per visit was a 'simple' equation considering the page weight (data transferred in GB), multiplied by the estimated energy usage of loading one GB (0,81) and multiplying it with the carbon intensity.
- In the updated formula segments the energy consumption by data centers, network, and energy consumed by user devices for greater insight.
- Each system segment is further broken down into two categories — operational and embodied emissions.

510. <https://sustainablewebdesign.org/estimating-digital-emissions/>

511. <https://sustainablewebdesign.org/estimating-digital-emissions/>

- **Operational:** The emissions attributed to the use of the devices in a segment.
- **Embodied:** The emissions attributed to the production of the devices in a segment.
- Each segment is attributed a certain weight in the calculation reflecting its estimated contribution to the total emissions -> for example, each segment has its estimated carbon intensity for both operation and embodied energy consumption.
- At the end, this allows us to estimate this for each segment, and then sum the totals to get the total estimated emissions.

Page weight

The Sustainable Web Design Model uses data transfer as a proxy. While we can debate whether this input gives us an accurate carbon emission estimate, it's clear that the increasing bloat of websites over the years is contributing to the problem. More data needs more servers to store them, more energy to deliver them, and more processing power to display them on-screen to end users. Over the years, there has been a visible upward trend, so let's look at that data for 2024.



Figure 15.1. Page weight.

Page weight represents the amount of data transferred to access the web page (based only on

HTTP requests). It is recommended that this metric be kept as low as possible to ensure a fast load and good user experience for all users. On the 90th percentile, the average mobile site weighs around 7.2 MB, while desktop ones are around 8 MB. The good news is that these numbers are slightly lower than the ones in the 2022 Almanac⁵¹². The bad news is that these numbers are still way too high. When we talk about sustainability, we usually touch upon inclusion, and to make sure the majority of Internet users can access the page and have a decent user experience, page weight should be kept below 1MB, ideally around 500KB⁵¹³.

It is also important to emphasize that these numbers are coming for a so-called lab test, meaning that pages were accessed via script and not real users. With many websites these days implementing lazy loading strategies (request and load assets once they are needed, through native lazy-loading for images and iframes or progressive hydration for dynamic components), as well as loading and processing extra scripts once the user has given consent for it (and the script does not include any user interaction emulation), average page weight is sure to be an even bigger number.

For further information, refer to:

- The WSG 1.0 - SC 2.6 and 5.27.

Weight by content type

With data showing that an average page weighs around 8MB, the next logical question is where all those kilobytes are coming from. Modern pages are composed of many different pieces, including basic building blocks such as HTML, CSS, and JS, as well as fonts and images to enhance visual presentation.

512. <https://almanac.httparchive.org/en/2022/sustainability#page-weight>

513. <https://infrequently.org/2021/03/the-performance-inequality-gap/>

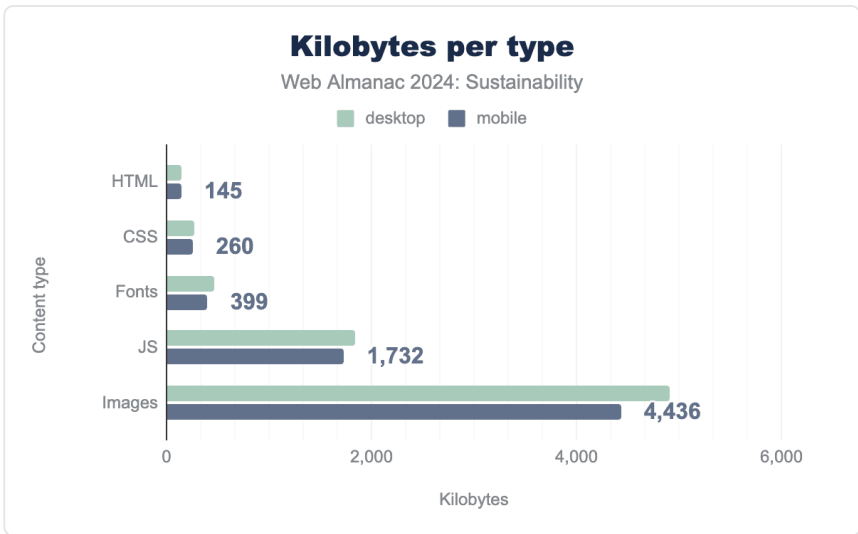


Figure 15.2. Kilobytes per type.

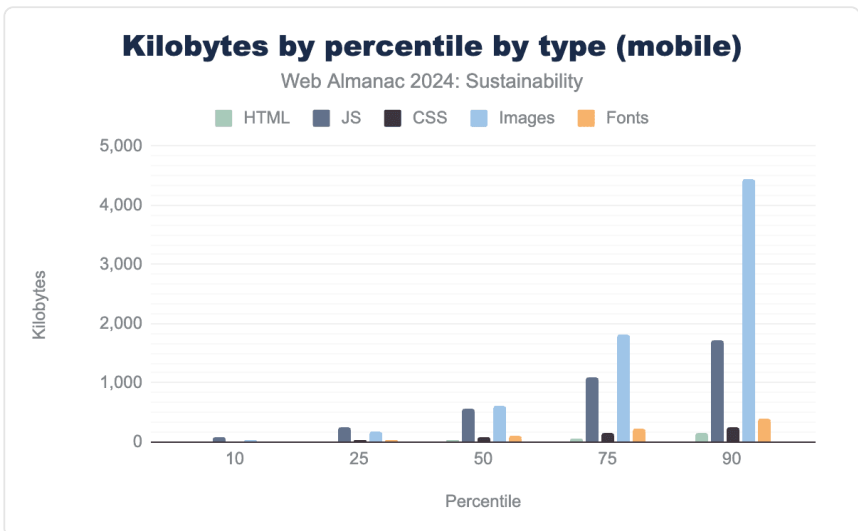


Figure 15.3. Kilobytes by percentile by type (mobile).

From the 90th percentile of collected data, it is clear that the largest portion, more than half of the page's total weight, belongs to images. This is quite an expected result and one of the parts where small optimizations can have the biggest impact. Slightly surprising and worrying is that there is still a very small gap between desktop and mobile numbers, not only in images but across all types. This shows us that even though mobile-first became a concept over 15 years

ago, most teams still don't optimize their mobile sites to account for the limitations of mobile networks, phone plans, and smaller screens (for which smaller images are usually enough). In the end, it may look mobile-friendly, but the experience isn't.

It is also not surprising that the second largest portion is JavaScript code. Modern frameworks, dependencies, packages, as well as legacy code easily accumulate a larger amount of JavaScript code. The biggest problem is that script processing is part of the page loading that needs the most power from the CPU and consumes the most energy. Also, security should be a concern, check the Security chapter for more on this.

Compared to images and JavaScript, the weight of CSS doesn't look too large. However, the important question covered later in the chapter is how much of this code, both CSS and JS, is actually being used. In addition to these considerations, the processing cost of such code should be considered (CPU/Memory usage).

The size of font files may look reasonable if we consider that a typical custom font file can be over 200 KB⁵¹⁴. This size, though, can be brought down significantly by subsetting font files to only characters that are needed for the site content. The median font file with an English-only subset of characters should be around 12 KB⁵¹⁵, which is 6% of the original 200 KB and could bring us closer to having a great-looking page that is below 1 MB.

More about this topic can be found in the Fonts and Page Weight Chapter.

For further information, refer to:

- The WSG 1.0 - SC 2.6 and 5.27.

Carbon emissions

As stated above, back in 2022, we used the Sustainable Web Design (SWD) model to estimate carbon emissions based on page weight. As we explained many times, this is inaccurate but constantly improving. As such a new version of the model was recently published. This has a considerable impact on the results so we decided to recalculate emissions from the 2022 data before proceeding with the 2024 data. In both cases, we exclude data from the 100 percentile, which is some kind of “worst-case scenario” and way above other percentiles thus making the diagrams less readable.

514. <https://sustainablewebdesign.org/has-the-design-used-the-minimum-number-of-custom-fonts/>

515. <https://www.phpied.com/bytes-normal-web-font-study-google-fonts/>

Emissions for 2022

Based on the V4 of the SWD model, we get these global results for emissions:

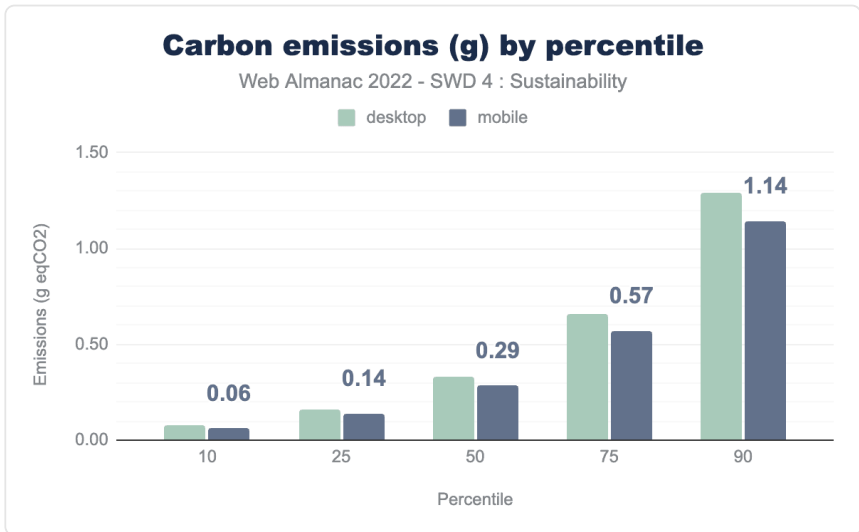


Figure 15.4. Carbon emissions (g) by percentile for 2022.

If you take a look at the results from the 2022 Sustainability⁵¹⁶ chapter, you'll notice that these new estimations are slightly lower. This is why we recalculated them. Otherwise, we could have been led to think that emissions from web pages diminished significantly between 2022 and 2024 (spoiler: this is not the case). With this in mind, we can now look at the data for 2024.

Emissions for 2024

Based on the SWD V4 model, carbon emissions for web pages in 2024 are as follows:

⁵¹⁶. <https://almanac.httparchive.org/en/2022/sustainability#carbon-emissions>

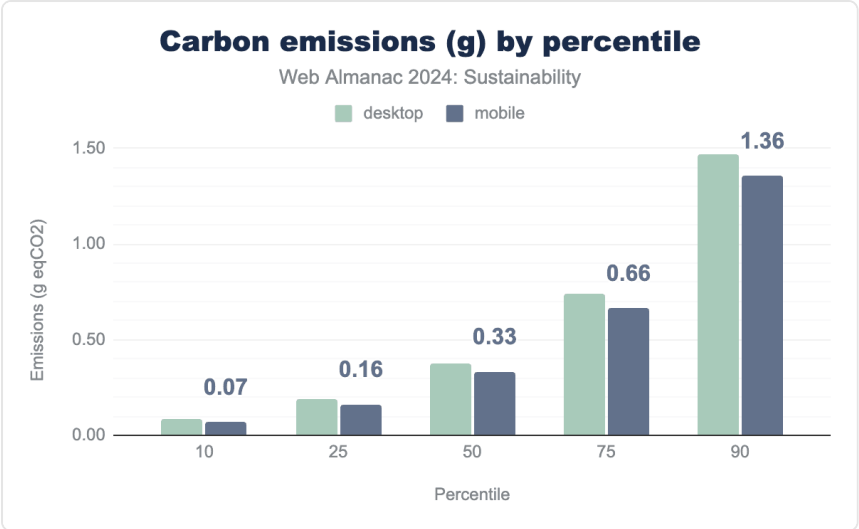


Figure 15.5. Carbon emissions (g) by percentile for 2024.

Even if the results are quite similar to those from 2022, we notice a slight increase in carbon emissions, which is even more significant for the 75 and 90 percentile. This is bad news as we should focus on reducing all our carbon emissions. This is not surprising since page weight has been globally on the rise for many years. More on this in the Page Weight chapter. The goal of the Sustainability chapter is precisely to raise awareness on this but also to provide recommendations to improve things.

We can also take a look at the share of emissions for different resources (HTML/JS/CSS/images/fonts), distinguishing mobile and desktop:

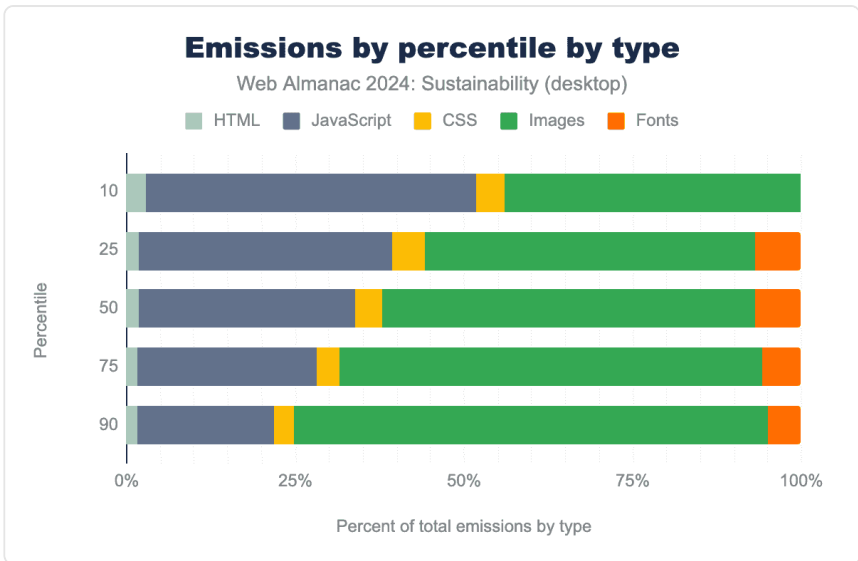


Figure 15.6. Emissions by percentile by type for 2024 (desktop).

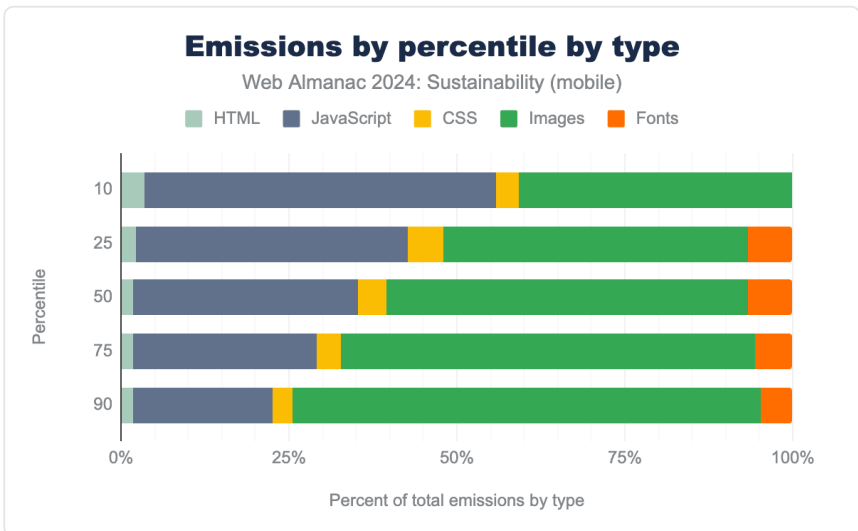


Figure 15.7. Emissions by percentile by type for 2024 (mobile).

This looks quite similar to what we found in 2022, leading to the same conclusions. We should however insist on the fact that images are usually easier to process than JS and CSS. Here, JS is a major offender regarding transferred data but is usually quite impactful on memory, CPU, and GPU, which then leads to additional environmental impacts that are not yet considered in the

SWD model. For instance, soliciting more CPU/GPU/memory could have an impact on the battery discharge of your smartphone, thus forcing you to change the battery or device sooner than expected and possibly making the website less performant on older devices.

Some “meta” considerations

Somewhere along writing this chapter, someone in the team wondered what would be the total carbon emissions related to the HTTP Archive monthly crawl. Each month, data is collected from millions of web pages to monitor the state of the web. For instance, the Web Almanac is based on this data. Find more about the methodology.

Read other reports from the HTTP Archive⁵¹⁷.

We ran a query on the data collected for the 2024 Web Almanac and found that the total amount of transferred data would be around 201,66 TB. Using the SWD model, this would amount to 27,7 T CO₂. Based on a CO₂ converter⁵¹⁸, this is approximately as many carbon emissions as a thermal car driving for 127 298 km (going around the Earth for more than 3 times) or manufacturing 323 smartphones. Things only add up when considering this crawl is done monthly.

But you should also keep in mind that:

- This is a necessity to have updated data.
- Some of these pages gather millions of visits each month.
- That does not prevent us from thinking about ways to reduce these impacts (and I hope that will be considered soon).

Going further

It should be noted that while many assume sustainability to be a task primarily (or purely) focused on reducing carbon (GHG) emissions, the considerations of digital sustainability are not as simple as they would initially appear. There are not only several ways in which sustainability can be measured and a variety of different resource types that must be accounted for, but a range of variables that can impact the resulting calculation.

Carbon emissions can ultimately be measured through the amount of energy (electricity) consumed through the act of processing actions (such as how many objects are rendered to the screen and the energy intensity of CPU, GPU, and RAM required for this activity), and as such, it

517. <https://httparchive.org/reports>

518. <https://impactco2.fr/outils/comparateur>

can be more easily calculated, and any reductions made where beneficial. However, other natural resources are consumed by Internet infrastructure or devices connected to the web such as water (for cooling equipment), materials and chemicals (e.g. printing), and e-waste (old devices being thrown away). The sustainability impact of these issues must also be accounted for in the lifecycle chain of a product or service.

For further information, refer to:

- The WSG 1.0 - SC 4.1 and 5.5.

Number of requests

Requests are generated whenever a file is needed to load a page, providing insight into the page's impact on the network and servers. They are even used to calculate environmental impact. Analyzing these requests helps us uncover opportunities for optimization, which will be valuable as we explore different types of assets and external requests.

It's essential to minimize the number of requests. Setting an initial cap of 25 is a positive step, but it's often challenging to meet this target due to the presence of trackers and similar elements.

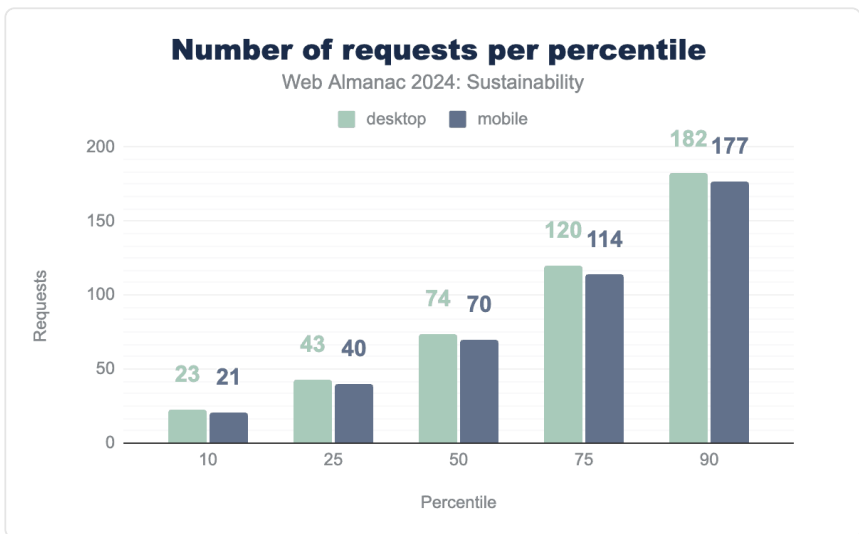


Figure 15.8. Number of requests by percentile.

From the extracted data, we can see that the number of requests is quite similar for mobile and desktop pages, which is not ideal. To respect mobile networks as well as plan limitations, it

would be great to see fewer resources being loaded on mobile, which would result in fewer requests. The amount of requests, in general, is quite high, so let's see what resources these requests are calling for.

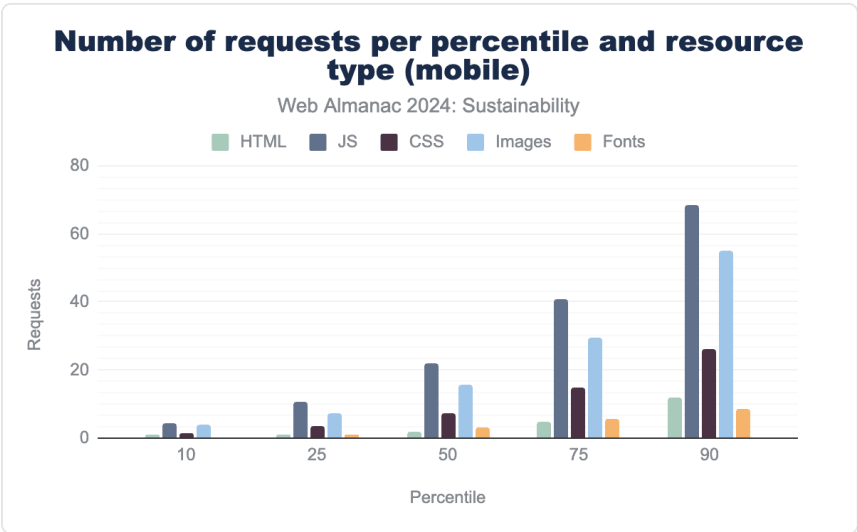


Figure 15.9. Number of requests per percentile and resource type (mobile).

The not-so-nice surprise here is that most of the requests, around 70 in the 90th percentile, are retrieving JS files, followed by over 50 for images. We can see that the pattern repeats itself across all the percentiles. This is very interesting since, in the report from 2022, the number of requests retrieving images was bigger than the ones retrieving JS. The follow-up question is whether the change in the number of requests also impacts the size of the retrieved assets. So, let's check that.

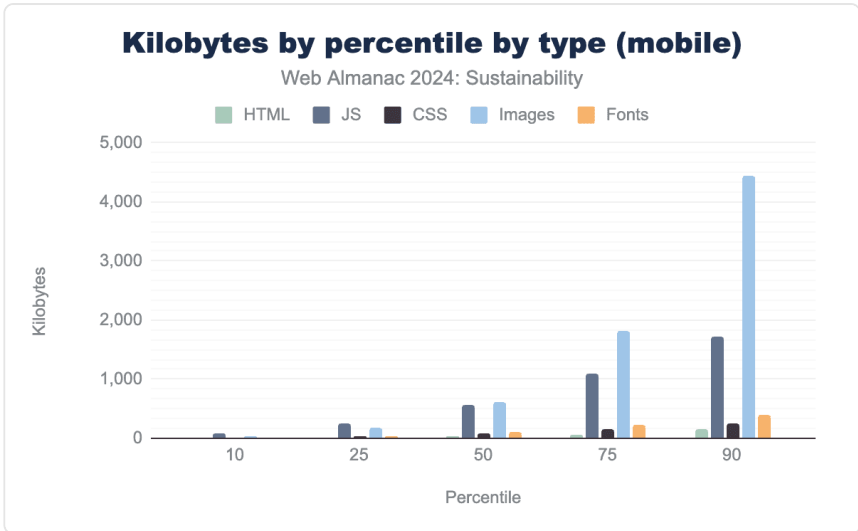


Figure 15.10. Kilobytes by percentile by type (mobile).

The size of retrieved Images is almost double the size of retrieved JavaScript in the 90th percentile, so the “old” pattern is still there. This leads us to the conclusion that sites are loading fewer images but slightly heavier and, in general, more JavaScript, which is not a good trend since that leads to the need for more processing power and can exclude users with aging devices from accessing sites.

For further information, refer to:

- The WSG 1.0 - SC 3.1 and 4.9.

More sustainable hosting

One of the simplest ways to reduce your carbon footprint is by choosing a sustainable hosting provider⁵¹⁹. Navigating this world can be quite tricky as no provider will truly be 100% carbon neutral as emissions always exist in some form, however, how they power their equipment, maintain their hardware, and deal with waste and redundancy can significantly impact the amount of emissions created. Therefore picking the right supplier⁵²⁰ can have a considerable impact on your ability to mitigate the emissions you produce.

One of the most critical features of a sustainable host is how they get their energy

519. <https://dodonut.com/blog/how-to-choose-the-best-green-web-hosting-provider/>

520. <https://www.thegreenwebfoundation.org/tools/directory/>

requirements. Hosts may get their electricity supply from normal power companies which in turn (depending on where they are located) generate a significant proportion of their energy from carbon-emitting coal or gas. Having a hosting provider that generates its energy requirements from solar, wind, tidal, and other natural sources will be more beneficial to the planet.

Other factors to consider are how the hosting provider cools its equipment. Many hosts require water cooling (and this can be an issue where freshwater can be a valuable limited resource), suppliers that use natural cooling in colder climates (for example) may be preferable.

Hosts that allow you to monitor your energy requirements (PUE, WUE, CUE⁵²¹) and even the amount of energy you are utilizing (CPU, GPU, RAM) on an active and logged basis can help you scale your hosting to reduce waste. If the host also recovers, recycles, and upcycles⁵²² equipment rather than has a throwaway culture, this can reduce e-waste (the same goes for keeping equipment as long as possible).

Compensating for emissions at the server-side can be a complex task and transparency with hosting providers can also be a bit of a minefield, but providers are gradually becoming more sustainably minded and with specialist providers out there who can help you reduce your carbon footprint, taking steps to house your product or service in a greener space can be relatively straight forward.

Note: You may be able to use APIs and infrastructure to model your website or application based on the environmental situation at that time. Carbon-aware computing⁵²³ is a relatively new concept but it has some interesting viewpoints.

For further information, refer to:

- The WSG 1.0 - Section 4.

How many of the sites listed in the HTTP Archive run on green hosting?

To help organizations and individuals choose “greener” hosting, the Green Web Foundation⁵²⁴ maintains a dataset of providers matching the “green” criteria⁵²⁵. With the list expanding and updating regularly it is not easy to track this metric, however, let’s see what is its current state.

521. <https://submer.com/blog/pue-cue-and-wue-what-do-these-three-metrics-represent-and-which-is-one-is-the-most-important/>

522. <https://www.scientificamerican.com/article/reduce-reuse-recycle-why-all-3-rs-are-critical-to-a-circular-economy/>

523. <https://hackernoon.com/carbon-aware-computing-next-green-breakthrough-or-new-greenwashing>

524. <https://www.thegreenwebfoundation.org/>

525. <https://www.thegreenwebfoundation.org/what-we-accept-as-evidence-of-green-power/>

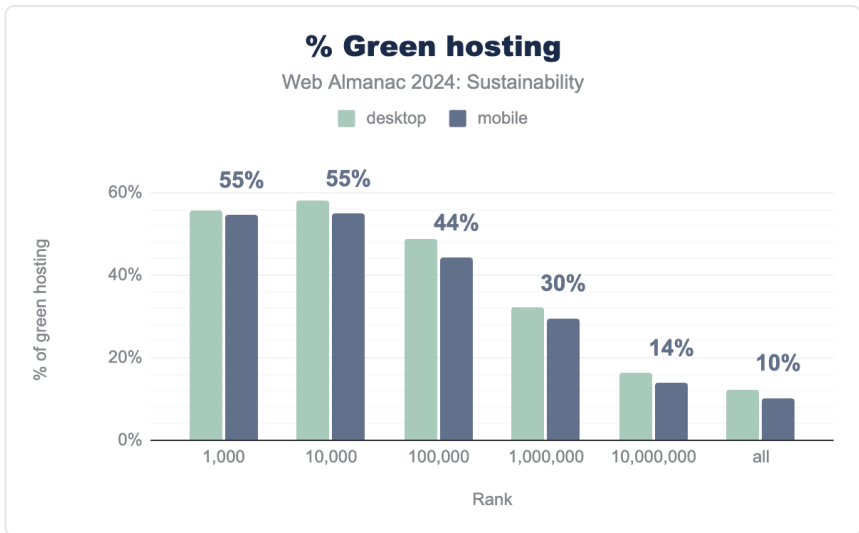


Figure 15.11. % Green hosting.

We can see that only 14% of mobile sites, and slightly more desktop ones from the HTTP Archive are hosted using “green hosting” providers. That is a slight increase from 2022, when this number was 10%, however, it shows us that progress in this area is extremely slow and that there is still a long way to go in both aspects: encouraging website owners to switch to “greener” hosting provides, as well as more providers offering “greener” hosting.

There is a significant difference in top-ranked sites, showing that 55% of sites ranked in the top 10000 are considered to be hosted green. This number also has a slightly better jump from 2022, when it was 48% for the top 10000 ranked sites. As good as this may sound it can easily be attributed to the fact that many bigger hosting providers such as Amazon and Google are considered to be “green”.

Reducing the environmental impact of websites

Understanding the environmental impact of websites is only the first step; action is crucial. With our updated insights into website footprints, particularly regarding unused code and font usage, we can now explore more targeted and effective strategies for mitigation. Let’s examine how to translate this knowledge into tangible, sustainable web practices.

For further information, refer to:

- The WSG 1.0 - Sections 2, 3, 4, and 5.

Avoiding waste

Avoiding waste remains one of the most effective ways to reduce the environmental impact of websites.

- **Minimize unnecessary content and code:** Many websites still carry superfluous features and content. Be critical in assessing the necessity of each page element. Our analysis shows that unused CSS and JavaScript continue to be major contributors to page bloat. Ruthlessly eliminate unused code and unnecessary scripts.
- **Optimize processing efficiency:** JavaScript remains a significant source of energy consumption on user devices. Our data reveals a concerning trend of increased JavaScript usage. Whenever possible, opt for lighter alternatives or consider if the functionality is truly needed. Static HTML and CSS are often sufficient and far less resource-intensive.
- **Prioritize lightweight assets:** While text remains the most eco-friendly content type, our font analysis shows that even typography can contribute to waste. Choose the default system fonts where possible, and when using custom fonts, optimize their delivery. For images and videos, ensure they're compressed and only loaded when necessary.
- **Design for longevity and accessibility:** Create websites that function well on older devices and slower connections. This approach not only reduces environmental impact but also improves accessibility. By supporting older technology, we encourage longer device lifespans, which is one of the most impactful ways to reduce digital environmental footprint.
- **Implement smart loading strategies:** Use techniques like lazy loading, code splitting, and conditional loading to ensure that assets are only delivered when they're needed. This approach can significantly reduce unnecessary data transfer and processing.

Loading unused assets

You should only load assets that are essential for displaying the current view of the page. This can be achieved through techniques like lazy-loading, critical CSS extraction, and patterns such as Import on Interaction⁵²⁶ and Import on Visibility⁵²⁷. It's also crucial to load assets, particularly images and fonts, at the appropriate size for each client device. In this section, we'll focus

526. <https://www.patterns.dev/vanilla/import-on-interaction>

527. <https://www.patterns.dev/vanilla/import-on-visibility>

primarily on minimizing unused code and optimizing font loading, as our data shows these remain significant contributors to unnecessary page weight.

For further information, refer to:

- The WSG 1.0 - SC 2.7, 2.15-9, 2.23, 3.7, 3.18, and 3.19.

Fonts

For optimal sustainability, system fonts⁵²⁸ remain the best choice, requiring no additional downloads. When custom fonts are necessary, consider these strategies to minimize environmental impact:

- Use the WOFF2 format for superior compression and broad support.
- Implement variable fonts⁵²⁹ to replace multiple font files with a single, versatile one.
- It would be nice to also use subsets⁵³⁰ or use a tool such as subfont⁵³¹.
- For custom fonts, use analysis tools to remove unnecessary glyphs or features.
- If using third-party font services, leverage their optimization options, but consider self-hosting for better control.

Remember, each additional font weight or style increases payload. Balance aesthetic needs with sustainability goals. For icons, consider Scalar Vector Graphics (SVG) instead of icon fonts for better efficiency and accessibility.

By thoughtfully approaching typography, we can create visually appealing websites while reducing data transfer and energy consumption.

For more detailed information, refer to our dedicated Fonts chapter and stay updated with the latest web font optimization practices.

For further information, refer to:

- The WSG 1.0 - SC 2.18 and 2.19.

528. <https://www.smashingmagazine.com/2015/11/using-system-ui-fonts-practical-guide/>

529. <https://the-sustainable.dev/reduce-the-weight-of-your-typography-with-variable-fonts/>

530. <https://everythingfonts.com/subsetter>

531. <https://github.com/Munter/subfont>

Unused CSS

The environmental impact of excess code extends beyond mere inefficiency. It directly translates to increased energy consumption and a larger carbon footprint for both servers and user devices. While CSS frameworks boost development efficiency, they often introduce substantial amounts of unused styles. This bloat not only affects page load times but also unnecessarily increases data transfer and processing requirements. In an era of ever-growing global internet usage, the cumulative effect of this excess code on energy consumption is significant.

Modern development tools have made identifying and eliminating unused CSS more accessible. Features like Chrome DevTools' Coverage analysis⁵³² offer powerful analysis to trim stylesheets. However, the practice of loading all CSS upfront for caching purposes presents a nuanced challenge. While it can reduce server requests and improve performance for returning visitors, it potentially increases the initial carbon cost of page loads.

As web applications grow in complexity, striking a balance between comprehensive styling and sustainable practices becomes increasingly crucial. Unused CSS not only impacts user experience through slower initial loads but also contributes to increased energy usage in rendering and processing. Addressing unused CSS stands out as a tangible step developers can take to reduce the digital ecosystem's environmental impact.

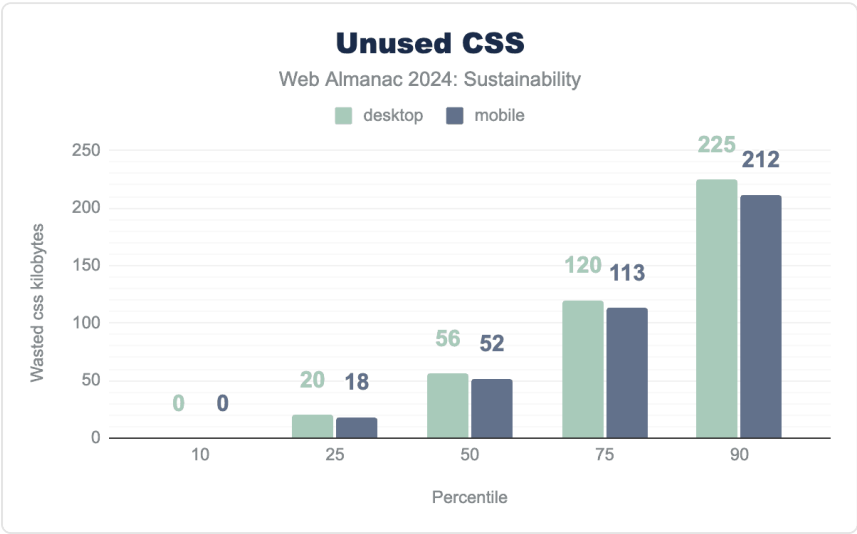


Figure 15.12. Unused CSS.

532. <https://developer.chrome.com/docs/devtools/coverage/>

Comparing the data from 2022 and 2024, we see some subtle changes. The 10th percentile remains unchanged, with no unnecessary CSS loaded, which is a positive sign but there's a slight increase in unused CSS across the remaining other percentiles.

These changes, while small, suggest a general trend towards slightly larger CSS footprints. This could be attributed to the growing complexity of web applications, the adoption of more feature-rich CSS frameworks, or an increase in the use of CSS for advanced styling and animations. The data reveals that by the 90th percentile, websites are still loading over 200KB of unused CSS. This represents a significant amount of unnecessary data transfer and processing, which can impact both performance and sustainability.

Despite these small increases, the overall picture hasn't changed dramatically since 2022. This suggests that while there have been some efforts to optimize CSS usage, there's still considerable room for improvement, especially for websites with higher percentiles.

For further information, refer to:

- The WSG 1.0 - SC 3.4.

Unused JavaScript

Unused JavaScript significantly impacts energy consumption and carbon footprints of both servers and user devices. While JavaScript frameworks enhance development efficiency, they often introduce substantial unused code, affecting page load times and increasing data transfer unnecessarily.

Modern techniques like tree shaking and code splitting are crucial for optimizing JavaScript. The following steps will most probably help you to decrease your unused Javascript:

- Tree shaking eliminates dead code from the final bundle, particularly effective with ES6 modules.
- Code splitting breaks code into smaller chunks, loading only what's necessary for the current functionality.
- Carefully evaluating the JavaScript libraries and frameworks.
- Regularly doing code audits and refactoring to get bundle sizes smaller.

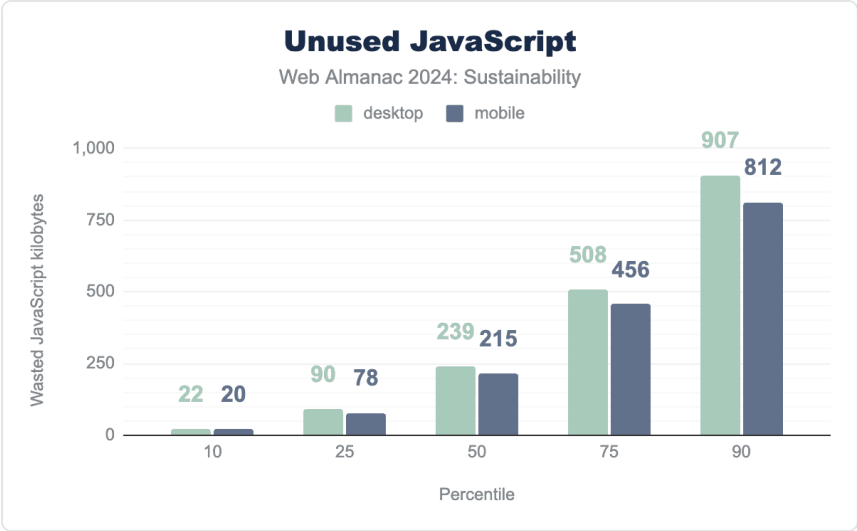


Figure 15.13. Unused JS.

Comparing 2022 and 2024 data reveals significant increases across all percentiles.

By the 90th percentile, websites now load over 900KB of unused JavaScript for desktop and over 800KB for mobile, significantly impacting performance and sustainability. The disappearance of 0KB values at the 10th percentile indicates even the most optimized sites now include some unused JavaScript.

The substantial increase in unused JavaScript between 2022 and 2024 underscores the urgent need for the web development community to adopt these optimization techniques for improved performance and sustainability.

For further information, refer to:

- The WSG 1.0 - SC 3.3 and 3.4.

Other technical optimizations

Different resource types have different energy intensity levels and this should be taken into account⁵³³ when choosing what to include with your product or service. Syntax languages such as HTML and CSS [PDF]⁵³⁴ are fairly simple but must render each component to the screen (creating emissions).

533. <https://greenspector.com/en/reduce-the-weight-of-a-web-page-which-elements-have-the-greatest-impact/>

534. <https://websitesustainability.com/cache/files/research23.pdf>

CSS has the added complexity of simple state-based interactions that can develop additional emissions (due to repainting) such as animations and transitions. JavaScript being a more technical language also has rendering requirements that can accumulate as the build complexity increases. It can also be used both on the client and server-side and can develop impacts while interacting with the page (due to its ability to manage state).

All server-side languages (such as PHP) will also have an impact on resource utilization and energy efficiency [PDF] (based on performance⁵³⁵). Images⁵³⁶ and media also have to be processed on the screen and due to their size and quality can be fairly impactful.

Optimizing your content

Now that we've examined the environmental impact of websites, particularly focusing on unused CSS, JavaScript, and font usage, let's turn our attention to optimization. While removing unnecessary content and functionalities remains crucial, this year we're emphasizing the importance of efficiency in what remains.

In this section, we'll explore how to make your essential content as sustainable as possible. We'll delve into optimizing images, videos, animations, and fonts - elements that significantly contribute to a website's environmental footprint. By fine-tuning these components, we can substantially reduce resource consumption without compromising user experience.

For further information, refer to:

- The WSG 1.0 - SC 2.8, 2.14, and 2.21.

Mobile-first design

Adopting a mobile-first approach not only enhances user experience but also offers significant sustainability benefits. By prioritizing essential content and functionality for mobile devices, we naturally create leaner, more efficient websites that consume less energy and bandwidth across all platforms.

This approach encourages developers to critically evaluate each element's necessity, leading to reduced data transfer and processing requirements. However, it's crucial to ensure that this doesn't result in multiple, device-specific versions of your site, which could potentially increase overall resource consumption.

When implemented thoughtfully, mobile-first design can lead to faster load times, lower energy

535. <https://attractivechaos.github.io/plb/>

536. <https://michaelandersen93.substack.com/p/greening-the-web-a-study-on-low-carbon>

consumption, and a smaller carbon footprint for your digital presence. It aligns well with other sustainability practices like minimizing unused code and optimizing assets.

For more detailed insights on how mobile-first impacts website performance, refer to our Performance chapter.

For further information, refer to:

- The WSG 1.0 - SC 3.14.

Image optimization

Images remain a significant contributor to web page weight and energy consumption.

Optimizing images offers substantial sustainability benefits: Reduced data transfer, lowering energy use in data centers and network infrastructure. Decreased processing requirements on user devices possibly result in lower energy consumption. Faster load times, potentially reducing user wait time and associated energy consumption.

When implementing image optimization, balance file size reduction with maintaining necessary quality. Utilize modern formats, responsive loading techniques, and appropriate compression levels. For detailed performance implications and technical implementations of image optimization strategies, refer to our Performance chapter.

For further information, refer to:

- The WSG 1.0 - SC 2.15.

Format (WebP/AVIF)

The adoption of modern image formats continues to be a crucial factor in web sustainability. WebP, with its impressive compression and wide browser support, remains a go-to format for optimizing images. Meanwhile, AVIF is gaining traction, offering even better compression in many cases. Let's look at how the usage of these formats has changed from 2022 to 2024:

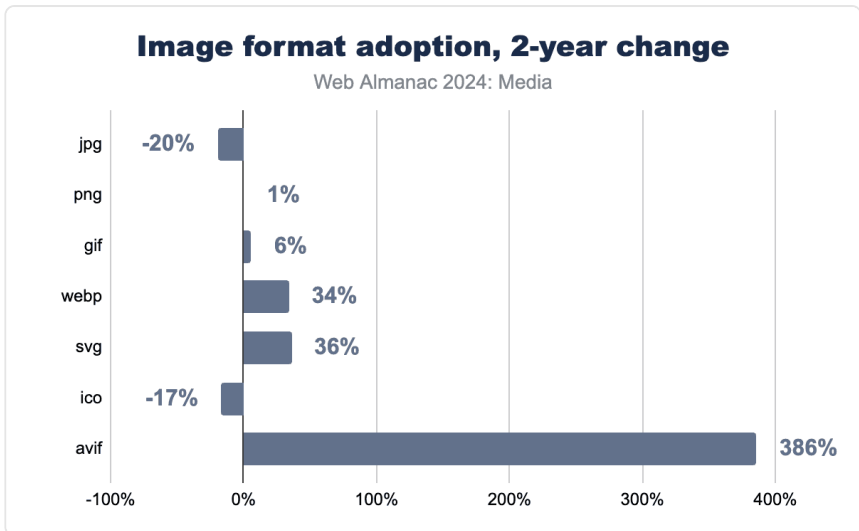


Figure 15.14. Image format adoption, 2-year change.

This data shows the evolving landscape of image format adoption. WebP has grown significantly, with a 34% increase in usage. Although AVIF shows an impressive 386% increase, it can be misleading since for desktop clients the usage of AVIF format is only at 1.40%, and for mobile clients 1.05%. Traditional formats like JPEG are seeing a decline, while PNG and GIF usage remains relatively stable. Current statistics for 2024 provide a clearer picture of where we stand:

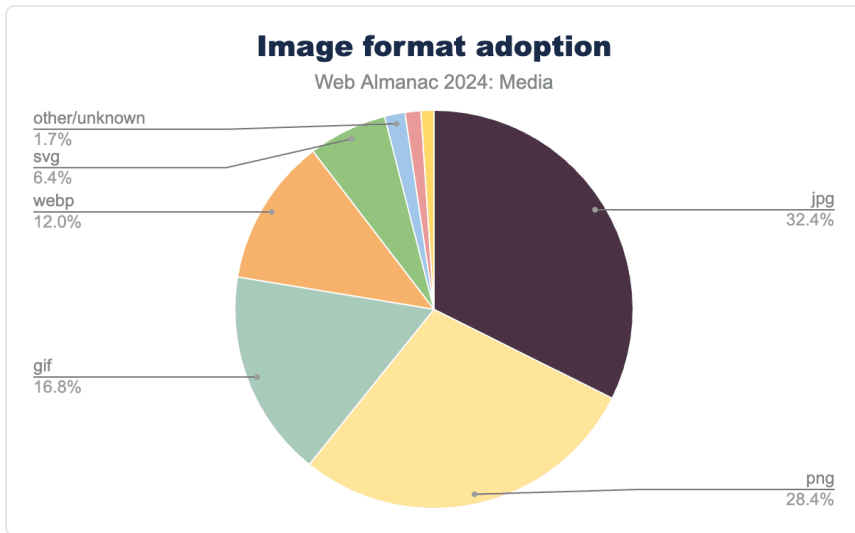


Figure 15.15. Image format adoption.

Despite the clear benefits, many websites have yet to fully embrace these modern formats. The potential for reducing page weight and improving loading times remains significant. For optimal sustainability:

1. Use WebP as your primary format, falling back to JPEG or PNG for older browsers.
2. Consider implementing AVIF for browsers that support it (with a fallback), as it often provides superior compression.
3. For icons and simple graphics, optimized SVG remains the best choice. Consider inlining frequently used SVGs directly in your HTML to reduce HTTP requests.
4. For JPEG/WebP you can aim for 80-85% quality; adjust based on visual inspection.
5. Use `srcset` and `sizes` attributes to serve appropriate image sizes.
6. Lazy load non-critical images. Use `loading="lazy"` (native HTML attribute) for images below the fold.
7. Strip metadata. Remove unnecessary EXIF data to reduce file size (thus also avoiding potential privacy issues).
8. Implement a content negotiation strategy to serve the most efficient format based on browser capabilities.
9. Periodically review and re-optimize images as new techniques emerge.

Remember, while adopting these formats can significantly reduce data transfer and processing requirements, it's crucial to balance compression with maintaining necessary image quality. Over-compression can lead to degraded user experience and potential re-uploads, negating some of the sustainability benefits.

For more detailed information on implementing these formats and their performance implications, refer to our Performance chapter.

Responsiveness, size, and quality

As the diversity of devices accessing the web continues to grow, delivering appropriately sized images remains crucial for both performance and sustainability. Responsive image techniques allow us to serve optimized images for each scenario, reducing unnecessary data transfer and processing. It's worth remembering that image quality doesn't always need to be at maximum.

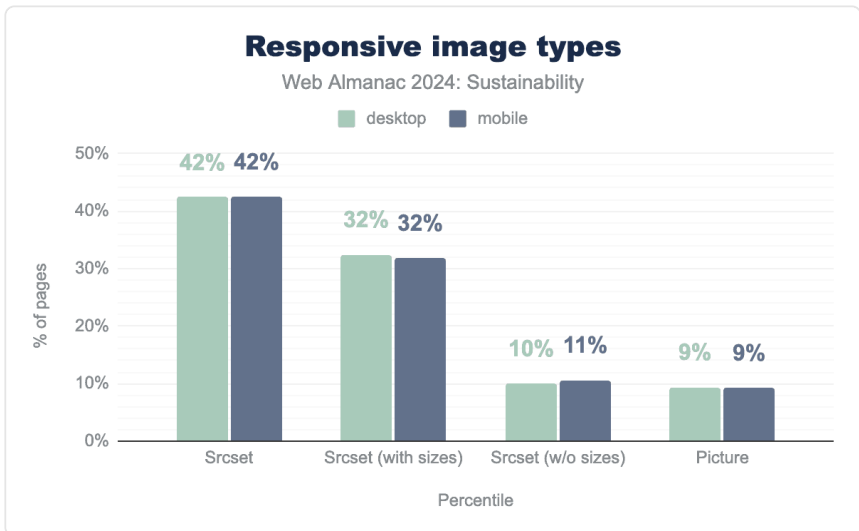


Figure 15.16. Responsive image types.

The data shows encouraging progress in the adoption of responsive image techniques:

1. Overall usage of the `srcset` attribute has increased significantly, from about 33-34% in 2022 to 42% in 2024 across both desktop and mobile.
2. The use of `srcset` with the `sizes` attribute, which provides more precise control over image selection, has also grown from around 25-26% to 32%.
3. There's a slight increase in the use of `srcset` without `sizes`, which, while not ideal, still offers some responsiveness benefits.
4. Usage of the `<picture>` element has seen a modest increase from 8% to around 9.3% across both desktop and mobile.

This trend indicates a growing awareness among developers about the importance of responsive images. However, there's still considerable room for improvement, as less than half

of websites are currently utilizing these techniques.

To optimize your images for sustainability:

1. Implement `srcset` and `sizes` attributes to serve appropriately sized images for different viewport sizes.
2. Use the `<picture>` element for art direction and to serve modern formats like WebP and AVIF with appropriate fallbacks.
3. Optimize image quality, aiming for the sweet spot of size reduction without noticeable quality loss.
4. Consider implementing automated tools in your build process to generate and optimize responsive images.

While the trend is positive, there's still a long way to go before responsive images become ubiquitous. As web professionals, we should continue to advocate for and implement these techniques to create more sustainable and performant websites. For more detailed implementation strategies and performance impacts, refer to our Performance chapter and Media chapter.

Lazy-loading

Lazy-loading remains a crucial technique for enhancing both performance and sustainability in web development. By loading images progressively we can significantly reduce initial page load times and unnecessary data transfer. This approach is particularly beneficial for sustainability, as it prevents the loading of images that users may never see, saving energy and resources.

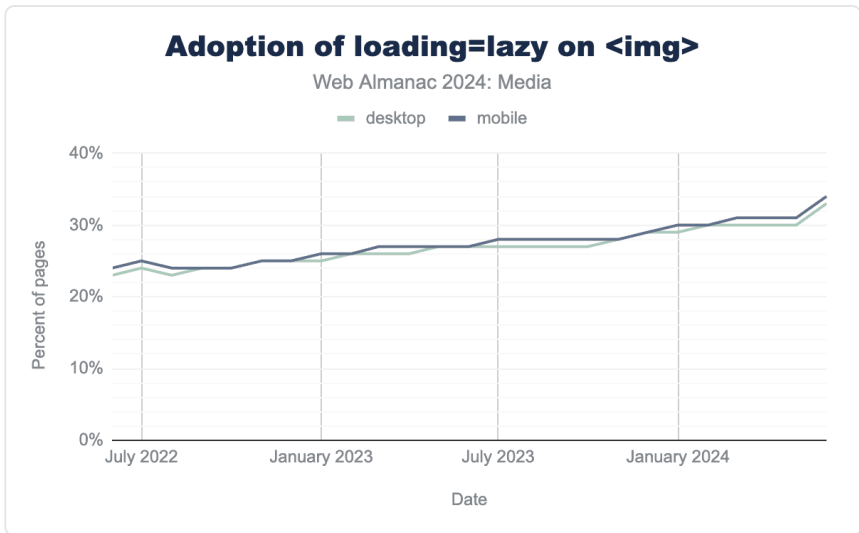


Figure 15.17. Adoption of loading=lazy on `<img>`.

The past two year's data reflects a growing awareness of the importance of optimized image loading. However, there's still considerable room for improvement, as a significant portion of websites have yet to implement any form of lazy-loading. Regarding iframes, the advice remains largely unchanged:

1. Native lazy-loading can be applied to iframes, offering similar benefits as with images.
2. However, for optimal sustainability, consider avoiding iframes altogether when possible. Usage of iframes can lead to unnecessary resource consumption if overused or if embedded content isn't controlled or optimized (e.g., third-party trackers or ads).
3. The facade pattern⁵³⁷ remains a preferred approach for integrating external content like embedded videos or interactive maps.

For deeper insights & analysis, refer to our Performance chapter and Media chapter.

Efficiently encoding images

Image encoding plays a crucial role in web sustainability, directly impacting the amount of data transferred across networks and the energy consumed in the process. Efficient encoding reduces the size of images, which often constitute a large portion of a web page's total size. This

537. <https://developer.chrome.com/docs/lighthouse/performance/third-party-facades>

reduction in data transfer translates to lower energy consumption in data centers, network infrastructure, and user devices. Moreover, smaller, well-encoded images require less processing power to decode and render.

The cumulative effect of efficient image encoding across billions of web pages can lead to substantial global energy savings.

Video

Videos remain among the most resource-intensive elements on websites, significantly impacting sustainability. For more detailed information, refer to our Media chapter. When incorporating third-party videos, utilizing facades is still the recommended approach.

Additionally, configure your videos thoughtfully:

1. Avoid preloading and autoplay to reduce unnecessary data transfer.
2. Implement lazy loading for videos below the fold.
3. Use appropriate compression techniques to reduce file sizes without compromising quality.
4. Consider using adaptive bitrate streaming for varying network conditions.

Remember, every optimization in video delivery can lead to substantial energy savings given the large file sizes involved. Balancing video quality with sustainability goals is key to creating engaging yet environmentally responsible web experiences.

For further information, refer to:

- The WSG 1.0 - SC 2.16.

Preload

Automatically preloading videos is a concern for web sustainability. This practice involves retrieving data that might not be useful for all users, potentially leading to unnecessary data transfer and energy consumption, especially on pages with high traffic volumes. From a sustainability perspective, preloading should ideally be avoided or only initiated upon user interaction.

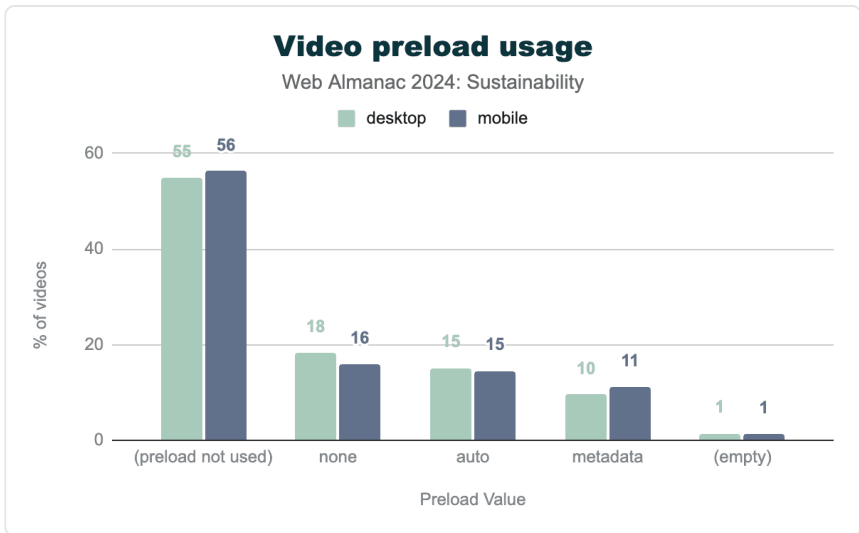


Figure 15.18. Video preload usage.

Comparing the usage of the preload attribute between 2022⁵³⁸ and 2024, we observe some changes. The percentage of websites not using preload has slightly decreased, from 58% to 55% on desktop and from 60% to 56% on mobile. This shift suggests a small increase in the use of preload attributes, which could have implications for sustainability.

It's important to remember that the preload attribute only has three valid values: none, auto, and metadata (default). Using the preload attribute with no value or an incorrect value may be interpreted as 'metadata', which can still involve loading up to 3% of the video to retrieve metadata. From a sustainability standpoint, 'none' remains the most environmentally friendly option.

The slight increase in the use of 'metadata' and the decrease in non-use of preload suggest that more attention needs to be paid to video preloading practices to enhance web sustainability.

For more detailed information on this topic, refer to Steve Souders' 2013 article⁵³⁹ and the web.dev 2017 article on video preloading strategies⁵⁴⁰.

Autoplay

The considerations surrounding autoplay continue to be critical from a sustainability

538. <https://almanac.httparchive.org/en/2022/sustainability>

539. <https://www.stevesouders.com/blog/2013/04/12/html5-video-preload/>

540. <https://web.dev/articles/fast-playback-with-preload>

perspective. Autoplaying videos consume data and energy for content that users might not be interested in viewing, potentially leading to unnecessary resource usage.

It's important to note that the autoplay attribute can override preload settings, as autoplaying naturally requires loading the video content. This forced loading further impacts data consumption and energy use.

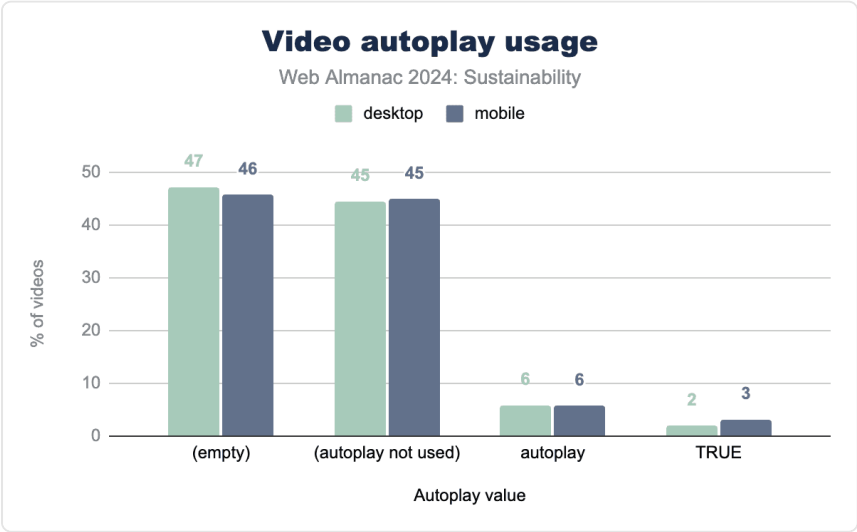


Figure 15.19. Video autoplay usage.

Comparing the usage of autoplay between 2022⁵⁴¹ and 2024, we see some notable changes. The percentage of websites explicitly not using autoplay has decreased, from 53% to 45% on desktop and from 53% to 45% on mobile. This could be a concern for sustainability efforts. Also, we notice a slight increase for websites using an empty value for this attribute, which also triggers autoplay (and is bad for sustainability).

It's crucial to remember that autoplay is a Boolean attribute, meaning its presence, even with an empty value, triggers autoplay behavior. The combined percentage of explicit autoplay usage (including `autoplay`, `TRUE` and other values) has remained relatively stable, around 8% for both desktop and mobile.

Given the sustainability implications, the trend towards more potential autoplay usage (through empty values) is worrying. Developers should be cautious about including the autoplay attribute, even if unintentionally, as it can lead to unnecessary data consumption and energy use. From a sustainability perspective, avoiding autoplay remains a recommended

541. <https://almanac.httparchive.org/en/2022/sustainability>

practice in most cases to reduce unnecessary data transfer and processing.

Animations

Animations continue to be a double-edged sword in web design. While they can enhance user experience, they pose challenges for both accessibility (more on this in the Accessibility chapter) and sustainability.

From a sustainability perspective, animations can be resource-intensive:

- They increase battery consumption and CPU/GPU usage, potentially reducing device longevity, especially on mobile devices.
- Animations often require additional code, which can delay rendering and increase page weight.
- Poorly optimized animations can lead to unnecessary repaints and reflows, further taxing device resources.

Recent data on non-composited animations⁵⁴² provides insight into their usage across websites:

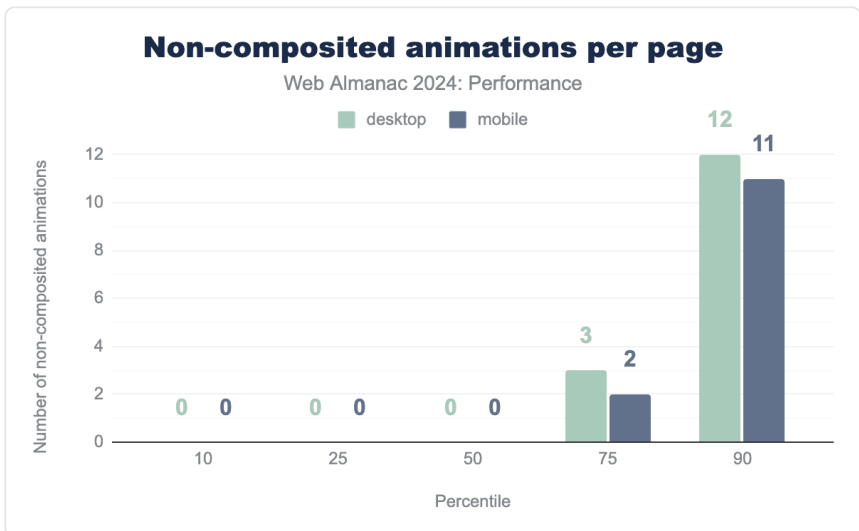


Figure 15.20. Non-composited animations per page.

This data reveals:

542. <https://web.dev/articles/stick-to-compositor-only-properties-and-manage-layer-count>

- At least 50% of websites don't use non-composited animations, which is positive from a sustainability perspective.
- There's a significant jump in usage at the higher percentiles, with the 90th percentile showing 12 animations on desktop and 11 on mobile. This jump could severely impact the accessibility, performance, and sustainability of the webpage.

The case of carousels remains contentious: users, developers, and designers tend to despise them while organizations stick to them.

For arguments against using carousels, visit:

- Should I Use A Carousel⁵⁴³
- Usability Guidelines For Better Carousels UX⁵⁴⁴

If animations are necessary for your design:

- Use CSS animations where possible, as they're generally more performant than JavaScript-based animations.
- Consider using the `prefers-reduced-motion` media query to respect user preferences for reduced motion.
- Lazy-load animations that are not immediately visible on page load.

For further information, refer to:

- The WSG 1.0 - SC 2.17.

Favicon and error pages

Favicons and error pages continue to play a subtle but important role in website performance and sustainability.

Key considerations remain:

- Always include a favicon to prevent unnecessary 404 requests.
- Ensure your favicon is properly cached to reduce repeated requests.

543. <https://shouldiuseacarousel.com/>

544. <https://www.smashingmagazine.com/2022/04/designing-better-carousel-ux/>

- Optimize your 404 page HTML to be as lightweight as possible.
- Set up redirects properly, so that users find the content they are looking for.
- While browsing to a missing page should lead to a 404 page, loading a missing resource should return a text message⁵⁴⁵ rather than the whole 404 HTML page (as is the case with some servers). You should also look for dead links⁵⁴⁶.

For optimal favicon⁵⁴⁷ sustainability:

1. **SVG:** The ideal choice. SVGs are lightweight, scalable, and eliminate the need for multiple sizes, significantly reducing data transfer and storage needs.
2. **Optimized PNG:** A well-optimized PNG (180x180 pixels for iOS in the base directory named `apple-touch-icon.png`) can be a good compromise between file size and broad compatibility. Note the filename and location requirements as browsers will seek this (based on the meta tag if used) if no SVG is provided.
3. A `favicon.ico` with sizes 32x32 and 16x16 for compatibility purposes (older browsers will seek this within your base directory so having the file will reduce errors). These formats, when properly implemented, minimize data transfer, reduce storage requirements, and lower energy consumption across the web ecosystem. Always ensure your chosen format is properly optimized for the best sustainability impact.

545. <https://httpd.apache.org/docs/2.4/en/custom-error.html>

546. <https://www.deadlinkchecker.com/>

547. <https://evilmartians.com/chronicles/how-to-favicon-in-2021-six-files-that-fit-most-needs>

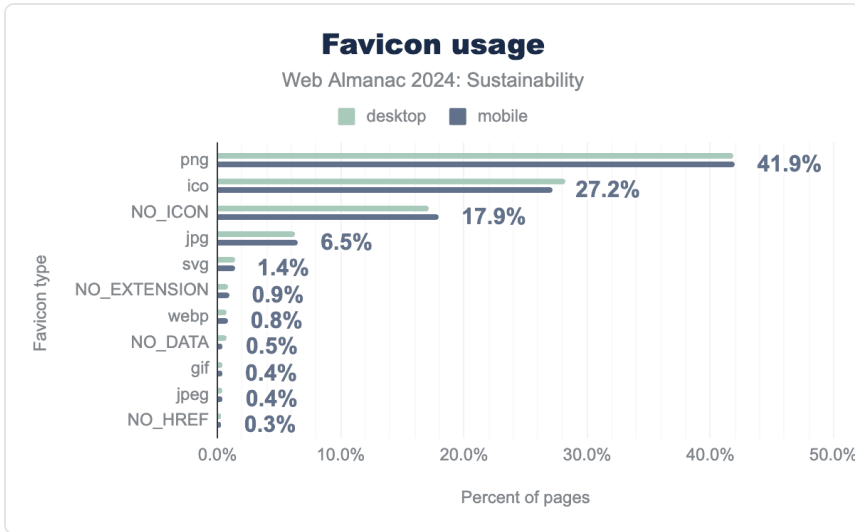


Figure 15.21. Favicon usage.

When we compare the 2024 data to 2021 data⁵⁴⁸ (since there was no favicon data since the 2021 Markup chapter⁵⁴⁹), we can say that the changes from 2021 to 2024 indicate a positive trend towards more sustainable favicon practices. The shift from ICO to more efficient formats like PNG, SVG, and WebP suggests improved awareness of file size and performance impacts. Also, the reduction in missing icons demonstrates better attention to detail, reducing unnecessary server requests. Finally, the growth in SVG and WebP usage, while still small, represents a move towards more sustainable, scalable formats.

While there's still room for improvement, particularly in further adopting highly efficient formats like SVG and WebP, the overall trend suggests that developers are increasingly considering the sustainability implications of even small elements like favicons.

For further information, refer to:

- The WSG 1.0 - SC 3.18 and 4.4.

Optimizing external content

One of the strengths of web development is the ease of integrating external content: from frameworks and libraries to third-party widgets and media. However, this convenience

548. <https://almanac.httparchive.org/en/2021/markup#favicons>

549. <https://almanac.httparchive.org/en/2021/markup#favicons>

shouldn't override considerations of necessity and efficiency. The environmental impact of external content is twofold: the immediate cost of transferring and processing the content, and the ongoing energy consumption from regular updates and continuous connections.

For each external element you consider adding, evaluate:

- **Energy footprint:** Consider both the operational energy (transfer, processing) and embodied energy (storage, infrastructure) costs
- **Resource necessity:** Could the functionality be achieved with a lighter, more energy-efficient solution?
- **Cache efficiency:** How frequently does the resource need to update, and could longer cache times reduce repeated transfers?
- **Network impact:** Will this resource require maintaining persistent connections or trigger frequent frequent background requests?

Third parties

Since third-party requests make up a large portion of requests on the web, it's interesting to make sure they come from "green" hosts. Back in 2022, we estimated that 91% of third-party requests came from "green" hosts. As of 2024, it has risen to a whopping 97%!

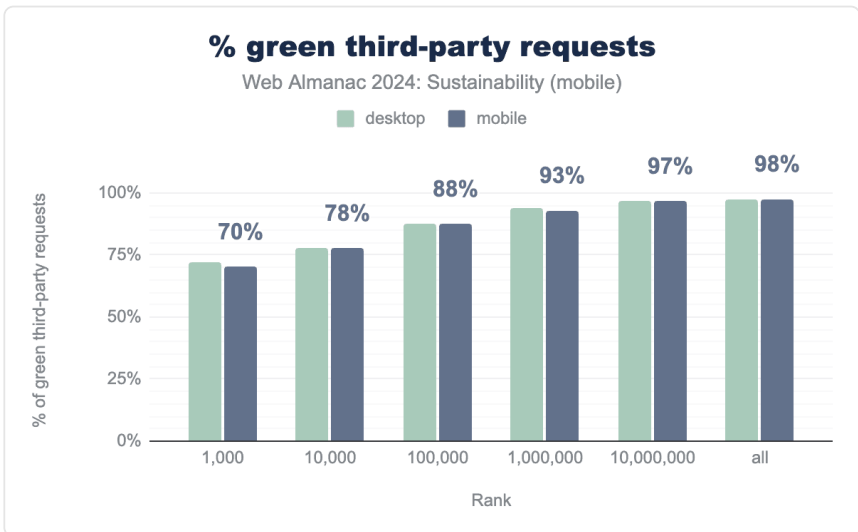


Figure 15.22. % green third-party requests.

This sure looks like really great news but you should keep in mind this is somewhat biased. Most third-parties originate from Google whose servers are considered “green”. More generally, many cloud providers are considered “green” but the truth might be slightly more complicated, as explained in our Green Hosting section.

You should look at this year’s Third-parties chapter for more on all this.

Third-party resources, while often essential for modern web functionality, can significantly impact a website’s sustainability and performance. These external assets, ranging from scripts to stylesheets, can increase page weight and create performance bottlenecks. All of these factors contribute to higher data transfer and energy consumption.

To create more sustainable websites, it’s crucial to regularly audit and optimize third-party usage. This process might involve removing unnecessary services.

Tools like Google’s Lighthouse offer valuable insights into the impact of third-party code on site performance. The Third-Party Usage audit in Lighthouse can help identify how external resources affect page load time and overall performance.

By thoughtfully managing third-party resources, we can strike a balance between necessary functionality and website efficiency. This approach not only improves user experience but also contributes to a more sustainable web ecosystem.

Making third-party requests more sustainable

Making third-party requests more sustainable requires a strategic approach from the earliest stages of development. While outsourcing certain services can potentially reduce development time and redundancy, it’s crucial to rigorously assess each third-party component for its ecological impact.

Prioritize self-hosted content over embedded third-party services whenever possible. This approach gives you greater control over performance and emissions. For unavoidable third-party content, implement a ‘click-to-load’ delay screen using the “import on interaction” pattern. This technique significantly reduces initial page load and unnecessary data transfer.

When evaluating libraries and frameworks, opt for performant alternatives that achieve the same goals with less overhead, keeping in mind that not using such tools could also be a possibility (thus sticking to vanilla JS for instance). Consider creating your own clickable icons and widgets rather than relying on third-party hosted solutions.

By carefully managing third-party integrations and prioritizing user preferences, we can significantly reduce the ecological footprint of our digital products while maintaining functionality and user experience. This is also usually beneficial to performance, security,

privacy, and accessibility.

For further information, refer to:

- The WSG 1.0 - SC 3.7.

For more detailed information on analyzing and optimizing third-party usage, refer to the Chrome Developers documentation⁵⁵⁰.

Implementing technical optimizations

Historically, web performance introduced a lot of recommendations that contribute to efficiency, thus improving sustainability. It also sometimes encourages frugality. But there are still some performance recommendations that could be detrimental to environmental impacts (or at least need some discussion and careful consideration). For instance, CDN should be treated with care (see dedicated section below) and preloading or predictive loading should be avoided because they might result in loading resources that will never be used.

JavaScript

JavaScript has been an important language in the web's growth, enabling dynamic and interactive experiences. However, when not optimized, it can also impact performance and energy consumption. Let's focus on some quick wins: optimizations that are easy to implement and great for sustainability. These tweaks can significantly improve your site's efficiency without sacrificing functionality. For a deeper dive into JavaScript's pros and cons, check out our comprehensive JavaScript chapter.

Minification, tree shaking & code splitting

Optimizing JavaScript through minification, tree shaking, and code splitting remains crucial for improving website performance and sustainability. By focusing on these optimization techniques, we can significantly reduce data transfer, improve load times, and ultimately decrease the energy consumption associated with web browsing. Remember, even small savings per site can lead to substantial cumulative benefits across billions of page views.

550. <https://developer.chrome.com/docs/lighthouse/performance/third-party-summary>

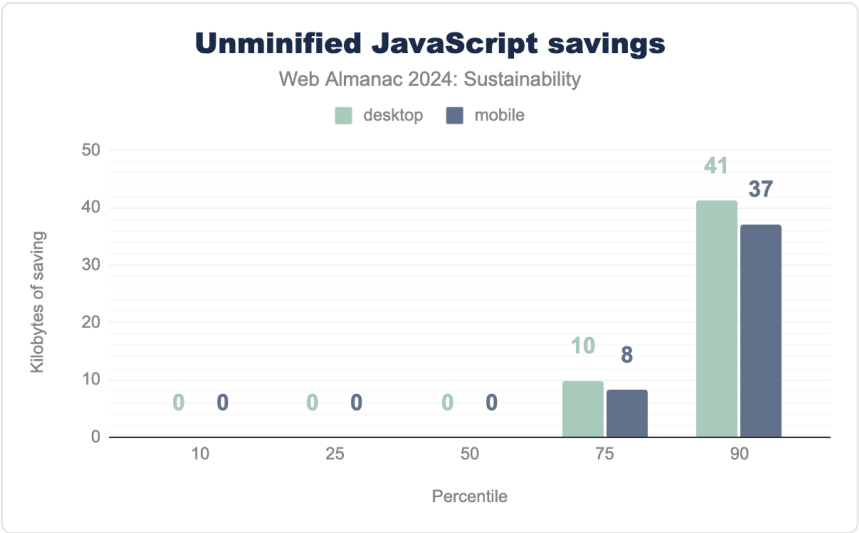


Figure 15.23. Unminified Javascript savings.

Compared to 2022, these figures show a slight increase in potential savings at the higher percentiles, indicating that some websites have accumulated more unminified code.

While many sites are effectively minifying JavaScript, there’s still significant potential for savings, especially for larger sites.

As developers, we can make our apps & websites more sustainable by:

- Implement tree shaking to eliminate dead code, which can provide additional savings beyond minification.
- Use code-splitting to load JavaScript on demand, reducing initial payload sizes.

For a deeper dive into JavaScript’s pros and cons, check out our comprehensive JavaScript chapter.

For further information, refer to:

- The WSG 1.0 - SC 3.2.

CSS

Cascading Stylesheets (CSS) have also been a critical feature in the growth of the Internet’s

popularity. The ability to add stylistic flourishes to pages and apps brings a unique visual appeal to content and features. The visual complexity of the Web however brings with it challenges in offering a sustainable product or service.

Web performance plays a critical role as too many page repaints or heavy burdens on the CPU can consume energy on the client-side. CSS also has to consider factors such as how it makes the best use of the form it is being consumed on to avoid triggering asset loading unnecessarily or wasting critical material resources if turned into a physical object (printing).

Let's focus on some quick wins, some of which will be familiar to you from the JavaScript section to help improve your code efficiency without sacrificing functionality.

Minification

While minification for JavaScript is a common practice (as mentioned earlier in this chapter), it's also important to recognize the value of minifying other static text assets⁵⁵¹ such as CSS files due to the data savings that can be obtained. Tools within IDEs can help automate and streamline this process to increase efficiency.



Figure 15.24. Unminified CSS savings.

Compared to 2022, these figures show a slight decrease in potential savings, suggesting a modest improvement in CSS minification practices or potentially a shift towards more efficient

⁵⁵¹. <https://web.dev/articles/optimizing-content-efficiency-optimize-encoding-and-transfer>

CSS authoring. The decrease in potential CSS savings since 2022 indicates progress, but further optimization is possible.

As developers, we can make our apps & websites more sustainable by:

- Consider using CSS Modules approaches, which can help eliminate unused styles.
- Implement critical CSS techniques to inline essential styles and defer the loading of non-critical CSS.

For further information, refer to:

- The WSG 1.0 - SC 3.2-4.

Print stylesheet

Reducing the impact of physical documents is critically important because any single-use item not only uses material resources (such as paper and ink), but there is a risk involved that the item may not be recycled properly, or that its production may not come from a sustainable source. Therefore we must treat the creation of such items as a precious commodity and avoid producing more than is required to reduce excess waste.

Having a print-friendly stylesheet⁵⁵² (using the `@media print` and `@page` at-rule) will allow you to define styles that stylistically affect how the end product will appear on the printed page⁵⁵³. This allows you to organize content to fit onto the page better, remove content that doesn't transcend well from digital to print (such as navigation), and include extra information where appropriate (for URLs alongside links as an example).

When creating a print-friendly stylesheet, be considerate of the user preference towards color or monochrome output, the color of paper used in the printer tray, the size of paper provided for printing, and the orientation of paper (responsive design of a sort that exists in print).

For further information, refer to:

- The WSG 1.0 - SC 2.23 and 3.13.

User preferences (dark mode)

Visitors have their preferred way of browsing websites, and one of the most common

552. <https://www.smashingmagazine.com/2018/05/print-stylesheets-in-2018/>

553. <https://alistapart.com/article/goingtoprint/>

preferences is “lights on or off” otherwise known as the use of dark mode. While this may outwardly appear to be a visual or stylistic choice, there are some sustainability and accessibility considerations to take into account with this choice.

Before we get into the sustainability benefits, it is worth quickly discussing that while the use of dark mode for some people can increase readability, it can affect others negatively by acting as a trigger for people with astigmatism⁵⁵⁴. It is therefore important that the ability to turn on and off features like dark mode be available to visitors and user preferences (on OS or browser) taken into account.

Dark mode itself can be a real benefit for sustainability on OLED screens due to the use of dimmed pixels⁵⁵⁵ (blacks and low colors⁵⁵⁶). Studies have shown that on such devices the reduction of energy use⁵⁵⁷ can vary but it does make a real difference⁵⁵⁸ (and as the screen is the primary energy emitter for handheld devices, this is important).

There are several other user-preference media queries⁵⁵⁹ available to CSS that may (depending on usage) have sustainability benefits for your visitors such as monochrome (to default printing to a single cartridge type), prefers-reduced-motion (to reduce processor-intensive animated effects), and the upcoming `prefers-reduced-data` (that allows designing around low bandwidth devices).

For further information, refer to:

- The WSG 1.0 - SC 3.13.

Including as little code as possible directly in HTML

The practice of inlining JavaScript and CSS directly in HTML can bloat HTML files and potentially harm overall performance and sustainability, as well as security. This issue is particularly prevalent in websites built with Content Management Systems (CMS) and those implementing the Critical CSS method⁵⁶⁰.

The importance of the separation of concerns⁵⁶¹ (HTML, CSS, and JavaScript) cannot be understated as also touched upon earlier in this chapter. While CSS can't defer or asynchronously load assets like JavaScript that would be render-blocking (without reliance on JavaScript itself), it still retains the same key benefit of being able to cache such assets. In doing so, a large library of CSS styles can be re-used among many pages without having to be re-

554. <https://www.boia.org/blog/dark-mode-can-improve-text-readability-but-not-for-everyone>

555. https://www.youtube.com/watch?v=N_6sPd0Jd3g

556. <https://greentheweb.com/energy-efficient-color-palette-ideas/>

557. <https://engineering.purdue.edu/ECE/News/2021/dark-mode-may-not-save-your-phones-battery-life-as-much-as-you-think-but-there-are-a-few-silver-linings>

558. <https://www.businessinsider.com/guides/tech/does-dark-mode-save-battery?r=US&IR=T>

559. <https://polypane.app/blog/the-complete-guide-to-css-media-queries/>

560. <https://web.dev/articles/extract-critical-css>

561. <https://meiert.com/en/blog/what-happened-to-separation-of-concerns/>

downloaded.

For further information, refer to:

- The WSG 1.0 - SC 3.17.

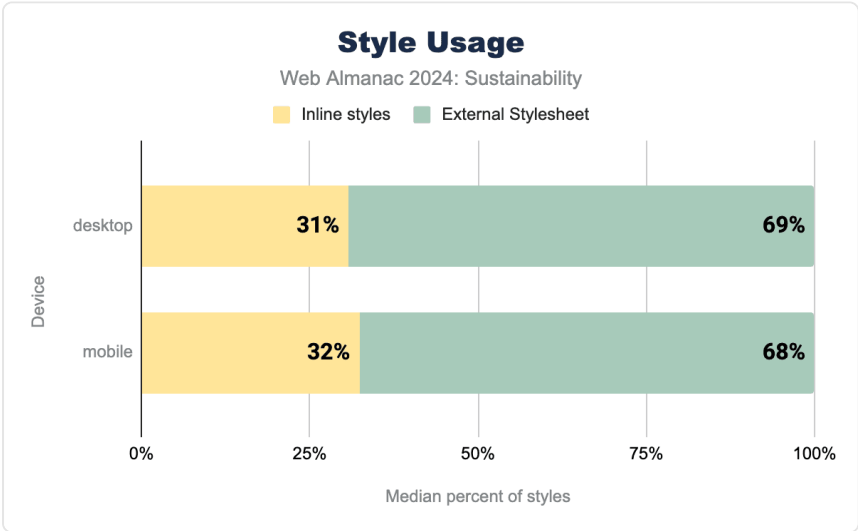


Figure 15.25. Style usage.

When we compare this year's data to 2022 data⁵⁶² we can see the following:

1. Increase in inline stylesheet usage:
 - Desktop: from 25% in 2022 to 31% in 2024.
 - Mobile: from 25% in 2022 to 32% in 2024.
2. A corresponding decrease in external stylesheet usage:
 - Desktop: from 75% in 2022 to 69% in 2024.
 - Mobile: from 75% in 2022 to 68% in 2024.

This trend shows a clear shift towards more inlining of CSS, particularly on mobile devices. While this approach can improve initial render times by reducing HTTP requests, it also presents challenges:

⁵⁶². <https://almanac.httparchive.org/en/2022/sustainability>

- Increased HTML file sizes, potentially slowing down initial page loads.
- Reduced caching efficiency, as inline styles, can't be cached separately from HTML.
- Potential for redundant code across multiple pages. In the end, even though inlining critical CSS could help reduce the initial render time we must be careful to not inline more than that because it could have the exact opposite effect and delay the initial render.

Obsolete code

In the rapidly evolving landscape of web development, obsolete code is an often overlooked source of inefficiency that can significantly impact a website's sustainability footprint.

Obsolete code refers to unnecessary, outdated, or redundant code that remains in a codebase. This can include:

- Deprecated JavaScript and CSS features.
- Polyfills for outdated browsers.
- Unused functions or styles from previous iterations of a site.
- Legacy code supporting discontinued features.

From a sustainability perspective, obsolete code is problematic for several reasons:

1. **Increased File Size:** Obsolete code bloats JavaScript and CSS files, leading to larger file sizes. This results in increased data transfer, consuming more energy both in transmission and processing.
2. **Unnecessary Processing:** Browsers still need to parse and execute obsolete code, even if it's not used. This consumes additional CPU cycles and, consequently, more energy on user devices.
3. **Maintenance Overhead:** Obsolete code makes codebases harder to maintain, potentially leading to inefficient workarounds and further code bloat over time.
4. **Security:** Obsolete code is much more likely to have known security issues that could put users at risk.

For further information, refer to:

- The WSG 1.0 - SC 2.29 and 3.20.

CDN

Content Delivery Networks (CDNs) play a crucial role in optimizing web performance and, by extension, improving the sustainability of web applications. By distributing content across multiple, geographically dispersed servers, CDNs reduce the distance data needs to travel, leading to faster load times and reduced energy consumption.

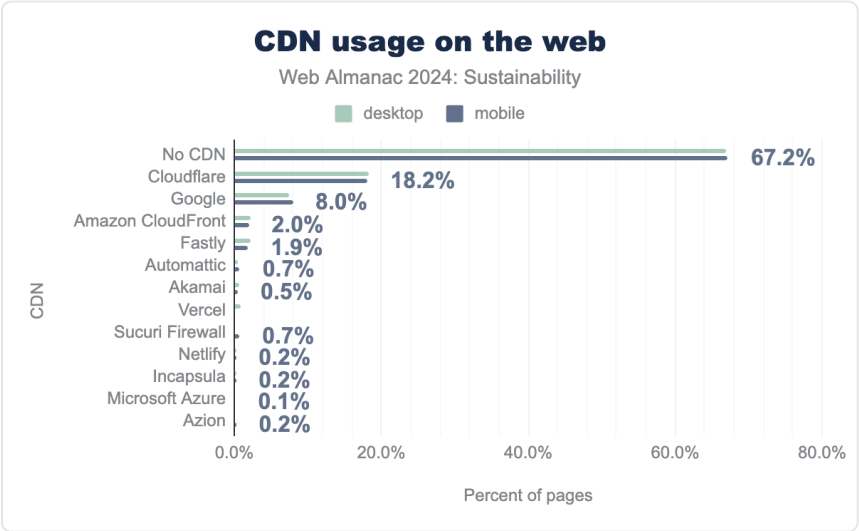


Figure 15.26. CDN usage on the web.

Comparing this data to 2022 data⁵⁶³, we observe several notable trends:

- **Increased CDN Adoption:** The percentage of websites not using a CDN has decreased from 69.7% to 66.88% for desktop and from 71.2% to 67.20% for mobile. This indicates a growing recognition of CDN benefits.
- **Cloudflare's Growth:** Cloudflare has solidified its position as the leading CDN provider, increasing its market share from 16.9% to 18.28% on desktop and from 15.1% to 18.16% on mobile.
- **Google's Expansion:** Google's CDN usage has seen significant growth, rising from 5.2% to 7.40% on desktop and from 6.5% to 7.95% on mobile.
- **Shifts in the Market:** While Fastly has seen a slight decrease in usage, Amazon CloudFront has maintained its position. Newer players like Vercel have entered the

563. <https://almanac.httparchive.org/en/2022/sustainability>

top 10, indicating a dynamic and evolving CDN market.

From a sustainability perspective, the increased adoption of CDNs is a positive trend:

- **Reduced Data Travel:** By serving content from geographically closer locations, CDNs minimize the distance data needs to travel, reducing overall network energy consumption.
- **Improved Caching:** CDNs often provide advanced caching mechanisms, reducing the need for repeated data transfers and server processing.
- **Load Balancing:** By distributing traffic across multiple servers, CDNs can help prevent server overload, potentially reducing energy consumption during traffic spikes.
- **Edge Computing:** Many modern CDNs offer edge computing capabilities, allowing for data processing closer to the end-user, which can further reduce energy consumption.

For a deeper dive into CDNs, check out our comprehensive CDN chapter.

For further information, refer to:

- The WSG 1.0 - SC 4.10.

Text compression

Text compression is a crucial technique for reducing the size of transmitted data, playing a significant role in improving both website performance and sustainability. By compressing text-based resources such as HTML, CSS, and JavaScript before sending them over the network, we can substantially reduce data transfer and, consequently, energy consumption.

Common text compression methods include:

- **Gzip⁵⁶⁴:** Widely supported and effective for most text-based content, typically achieving 60-80% compression ratios.
- **Brotli⁵⁶⁵:** A newer algorithm that often outperforms Gzip, especially for smaller files, with potential compression improvements of 15-25% over Gzip.

564. <https://www.gnu.org/software/gzip/>

565. <https://github.com/google/brotli>

- Zopfli⁵⁶⁶: A Gzip-compatible algorithm that can achieve better compression ratios but at the cost of longer compression times, making it suitable for static content.
- Zstandard (zstd)⁵⁶⁷ is also a serious contender that shows great performance for both compression and decoding.

More on this in this article from Paul Calvano⁵⁶⁸.

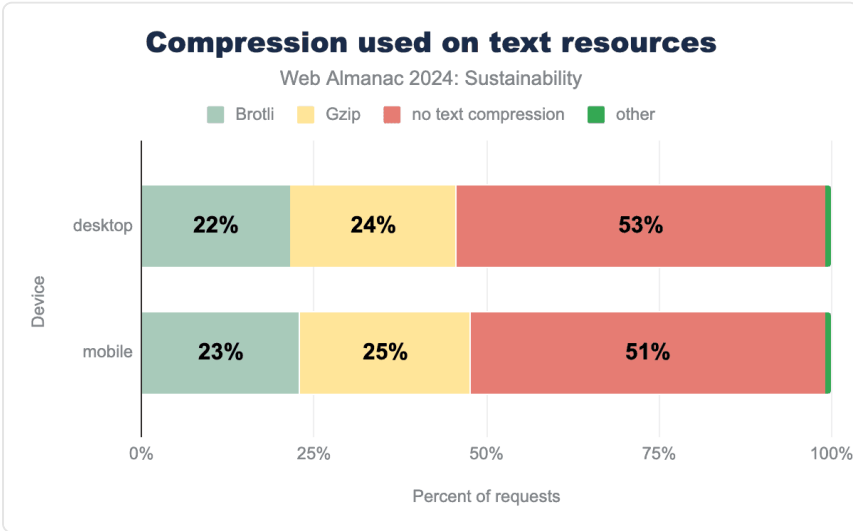


Figure 15.27. Compression used on text resources.

The data shows that over half of websites are not using any form of text compression, with 53.47% of desktop sites and 51.38% of mobile sites sending uncompressed content. This represents a significant missed opportunity for reducing data transfer and, consequently, energy consumption.

Among sites using compression, Gzip is slightly more prevalent than Brotli, being used by 24.05% of desktop sites and 24.65% of mobile sites. Brotli, despite its superior compression capabilities, is used by 21.59% of desktop sites and 23.03% of mobile sites.

Given that text compression can significantly reduce data transfer volumes, the widespread lack of adoption represents a substantial opportunity for enhancing the web's energy efficiency. Encouraging the use of compression, particularly more efficient algorithms like Brotli, could lead to meaningful reductions in data center energy usage, network traffic, and client-side

566. <https://github.com/google/zopfli>

567. <https://facebook.github.io/zstd/>

568. <https://paulcalvano.com/2024-03-19-choosing-between-gzip-brotli-and-zstandard-compression/>

processing requirements.

As we move towards a more sustainable web, text compression remains a key tool in our optimization toolkit, offering a relatively simple yet highly effective way to reduce our digital carbon footprint.

For further information, refer to:

- The WSG 1.0 - SC 4.3.

Caching

Caching is a crucial technique in web optimization that significantly contributes to sustainability efforts. By reducing the need for repeated data transfers and server processing, caching plays a vital role in minimizing the energy consumption associated with web operations.

From a sustainability perspective, effective caching offers several key benefits:

- **Reduced Data Transfer:** By storing resources closer to the user, caching dramatically decreases the amount of data transmitted over networks. This directly translates to lower energy consumption across the web infrastructure.
- **Decreased Server Load:** With fewer requests reaching origin servers, the overall energy usage in data centers can be significantly reduced.
- **Cumulative Environmental Impact:** For frequently accessed resources, the energy savings compound with each cache hit, potentially leading to substantial reductions in overall carbon footprint over time.

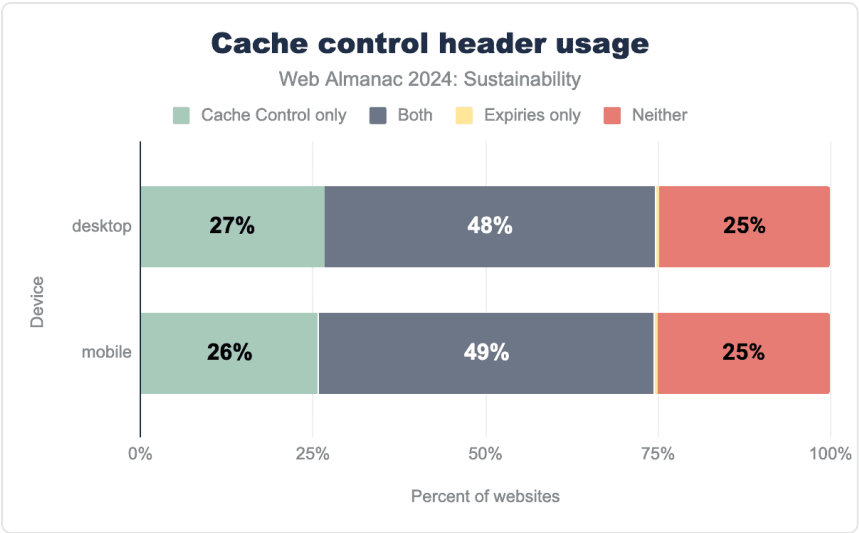


Figure 15.28. Cache control header usage.

An analysis of cache header usage between 2022 and 2024 reveals subtle shifts in caching practices:

- The use of both Cache-Control and Expires headers has slightly decreased, from 51% to 48.08% on desktop and 48.57% on mobile. Conversely, the use of Cache-Control only has increased from 23% to 26.65% on desktop and from 22% to 25.92% on mobile.
- The percentage of sites using neither caching header has remained relatively stable, with a marginal decrease from 25% to 24.89% on desktop and a slight decrease from 26% to 25.13% on mobile.

While these changes are modest, they indicate a trend toward more focused use of Cache-Control headers. From a sustainability perspective, the persistent quarter of websites using no caching headers represents a significant opportunity for improving resource efficiency and reducing unnecessary data transfers.

The shift towards Cache-Control only usage suggests a growing awareness of modern caching best practices, as Cache-Control offers more granular control over caching behavior. However, the overall slight decrease in caching header usage underscores the ongoing need to emphasize the importance of effective caching strategies in creating a more sustainable web ecosystem.

For more detailed information on caching techniques, implementation strategies, and

performance implications, please refer to our comprehensive Performance chapter.

For further information, refer to:

- The WSG 1.0 - SC 4.2.

SEO and sustainability

The advent of Search Generative Experiences in 2024 brings new sustainability concerns to light regarding Search Engine Optimization (SEO). The energy and water requirements⁵⁶⁹ to power these AI-enabled searches are significant. By some estimates, a single AI-powered search can use as much as 30 times more energy⁵⁷⁰ than a traditional search.

As with most web sustainability concerns, challenges are directly proportional to volume. We conduct billions of searches every day. Plus, Search Generative Experiences have been known to produce misleading, inaccurate, or false results. This increases misinformation risk as well as environmental impact when billions of searchers must rethink and re-run their queries to produce more accurate results.

What's more, there's currently no way to opt out of using these features. Web teams and digital marketers are forced to incorporate AI search features into their day-to-day practices.

To make SEO efforts more sustainable, marketers will also need to tackle important questions such as:

- Can AI and other emerging technologies help us execute search campaigns more efficiently or are the social and environmental impacts simply not worth the effort?
- How should we balance the need to consistently produce good content with limitations imposed by organizational resources and our stated climate goals?
- Are our content marketing efforts reaching the right audience and helping them to make better decisions and more sustainable choices? What should we do with old or otherwise outdated content to maintain relevance?
- Do we need to hold onto search and analytics data from seven years ago? Or even three?

With this in mind, here are seven ways to bring more sustainable SEO strategies⁵⁷¹ into your

569. <https://mashable.com/article/ai-environment-energy>

570. <https://www.scientificamerican.com/article/what-do-googles-ai-answers-cost-the-environment/>

571. <https://www.mightybytes.com/blog/sustainable-seo/>

search marketing efforts:

1. Provide measurable benefits to the people and communities you target, helping them find the information they need quickly and without barriers.
2. Utilize E-E-A-T—Experience⁵⁷², Expertise, Authoritativeness, and Trustworthiness—to focus on quality over quantity in search marketing efforts.
3. Create search marketing processes that can be realistically measured and maintained over time through continuous improvement. Include sustainability metrics in these efforts, doing more of what works and less of what doesn't.
4. Reduce negative effects⁵⁷³ where possible to minimize search marketing's impact on the planet.
5. As with any web sustainability initiative, reduce the amount of data you transfer, collect, and store.
6. Well-structured metadata can convey critical information about your website that can be used to educate potential visitors. For many sites, their purpose can be satisfied without a user ever actually visiting their domain.
7. Finally, align your search marketing efforts with the W3C Web Sustainability Guidelines (WSG) 1.0, particularly SC 2.6, 2.14, 2.15, 2.16, 2.19, 3.5, 3.12 and 4.4.

It is possible to redefine success⁵⁷⁴ by including stakeholder perspectives and needs in our marketing, making it an engine of well-being versus the more traditional engine of (often exploitative) growth. As part of these efforts, we also need search platforms to take ownership and action on their own roles in exacerbating the climate crisis.

Sustainable data and content management

Tech platform business models often depend on customers increasing data use over time. As part of their marketing strategies, platforms encourage customers to upgrade and purchase bigger data plans through conversion rate optimization, drip and marketing automation campaigns, notifications, and other means. To increase attractiveness, they regularly offer large amounts of data at relatively low costs.

Regrettably, this incentivizes sloppy data practices and proliferates the myth that data is immaterial or otherwise inconsequential when compared to the benefits of our data-fueled future. Reality, however, tells another story.

For example, the data center industry—currently valued at about \$125.35 billion⁵⁷⁵—is expected

572. <https://developers.google.com/search/blog/2022/12/google-raters-guidelines-e-e-a-t>

573. <https://thechange.com/7-ways-to-align-climate-strategy-with-digital-marketing-strategy-9668a3a3bda>

574. <https://www.mightybytes.com/blog/sustainable-digital-marketing/>

575. <https://www.precedenceresearch.com/data-center-market>

to grow 10%⁵⁷⁶ year over year between now and 2030. Unfortunately, data centers drive a variety of ecological problems⁵⁷⁷, including noise and air pollution, e-waste, increased emissions and energy use, water use, and other impacts. The rapid rise of artificial intelligence⁵⁷⁸ is especially problematic in this regard.

What's more, up to 90% of collected data goes unused⁵⁷⁹ after about three months. Most of this data is never properly disposed of⁵⁸⁰. Plus, poor data governance drives inequality⁵⁸¹, privacy, and security risks as well.

With all of this in mind, it's easy to see why our collective thirst for low-cost, high-volume data solutions drives significant sustainability concerns. This is unfortunate considering that data and analytics literacy are also often strategic differentiators for many organizations.

To make more sustainable data practices⁵⁸² the norm among those who create and manage digital products and services, organizations should redefine their relationships with data. For marketers and product teams, this can be accomplished by improving data governance and more effectively managing content across the products and services they create, subscribe to, or otherwise maintain. The annual Digital Cleanup Day⁵⁸³ can be a good way to raise awareness on this topic.

Data governance and sustainability

Data sustainability requires effective long-term governance strategies. Good data governance⁵⁸⁴ is key for long-term digital governance⁵⁸⁵ to work across an organization's teams and departments. In tandem with sustainability practices, these two disciplines can help organizations better manage web-based products and services over time while simultaneously improving how data is collected, managed, secured, and disposed of.

Specific tactics to improve data governance include:

1. Devise long-term organizational data and digital governance strategies that are grounded in sustainability principles and continuous improvement. Define and track clear KPIs to measure progress over time.
2. Train internal team members on data science to drive good governance. Appoint dedicated leaders to own digital and data governance within the organization.
3. Identify strategic partners⁵⁸⁶ in your digital supply chain⁵⁸⁷ that are aligned with the

576. <https://www.psmarketresearch.com/market-analysis/data-center-market>

577. <https://computing.mit.edu/news/the-staggering-ecological-impacts-of-computation-and-the-cloud/>

578. <https://humanrights.berkeley.edu/projects/climate-impact-of-ai-data-centers/>

579. <https://www.greenergydatacenters.com/eng/blog/90-of-data-sits-unused-how-to-get-rid-and-avoid-digital-waste>

580. <https://www.mightybytes.com/blog/what-is-your-data-disposal-strategy/>

581. <https://atmos.earth/inside-the-fight-for-indigenous-data-sovereignty/>

582. <https://www.mightybytes.com/blog/design-a-sustainable-data-strategy/>

583. <https://cyberworldcleanupday.fr/welcome.html>

584. <https://www.mightybytes.com/blog/sustainable-data-governance-for-marketers/>

585. <https://www.mightybytes.com/blog/building-capacity-with-digital-governance/>

586. <https://www.mightybytes.com/blog/sustainable-marketing-stack/>

organization's sustainability goals.

4. Regularly audit data and digital products and services to reduce risk and waste.
5. Think twice before you collect information and, when you do, document how long this information should be kept.
6. Finally, align data governance efforts with the W3C Web Sustainability Guidelines (WSG) 1.0, particularly SC: 2.1, 2.25, 4.12, 5.6, 5.12, and 5.20.

More sustainable content management

Similarly, content marketing often drives data collection for many organizations. Web forms, landing pages, blog posts, video tutorials, social media posts, and other forms of content all play important roles in an organization's marketing, product management, content governance, and digital sustainability⁵⁸⁸ efforts.

This offers opportunities for teams to support more sustainable content management within their organizations. Examples include:

1. Align content strategy⁵⁸⁹ efforts with specific sustainability goals and your team's capacity.
2. Publish accessible content that uses inclusive language⁵⁹⁰, is easy for a fifth grader to understand, is as long as it needs to be, and is formatted to skim with short paragraphs, bulleted lists, clear headings, and so on.
3. Regularly conduct content audits⁵⁹¹ to ensure your content continues to provide value for stakeholders over time. Use this process to edit, delete, add, clarify, or otherwise revise content as needed.
4. Manage content assets with sustainability in mind: compress, tag, and reuse assets as needed. Use heavy media assets like audio, video, or animation only when necessary. Also, use a digital asset management or design system to keep source files and other assets organized.
5. Audit tools regularly to ensure your team has access to the features they need proportional to your resources. Train or upskill team members on new features or processes.
6. Finally, align content management efforts with the W3C Web Sustainability Guidelines (WSG) 1.0, particularly SC: 2.6, 2.8, 2.14, 2.15, 2.16, 2.21, 2.23, 2.25, 3.5, 5.3, 5.4, and 5.12.

587. <https://www.mightybytes.com/blog/how-to-improve-your-digital-supply-chain/>

588. <https://www.mightybytes.com/blog/digital-sustainability/>

589. <https://www.mightybytes.com/blog/create-a-content-strategy/>

590. <https://www.apa.org/about/apa/equity-diversity-inclusion/language-guidelines>

591. <https://www.mightybytes.com/blog/how-to-run-a-content-audit/>

Popular frameworks, platforms, and CMSs

Online platforms and CMS tools significantly reduce the barriers for individuals and businesses to publish content or conduct commerce online. Additionally, development frameworks and site generators expedite project initiation for web developers, providing pre-set configurations and solutions that address common development issues. Performance is often a consideration in these tools, but for someone implementing a site, it is almost always cheaper to just get a bigger server.

With digital sustainability, scaling is what matters most. A byte here or there has an insignificant sustainability impact if it is just a single page. When that byte is loaded on over a million sites, with trillions of downloads a day, it has a much bigger impact.

The accompanying charts present the median page weight across the top five most popular eCommerce platforms, CMS tools, and site generator tools. These comparisons are to highlight the collective impact of the tools we use.

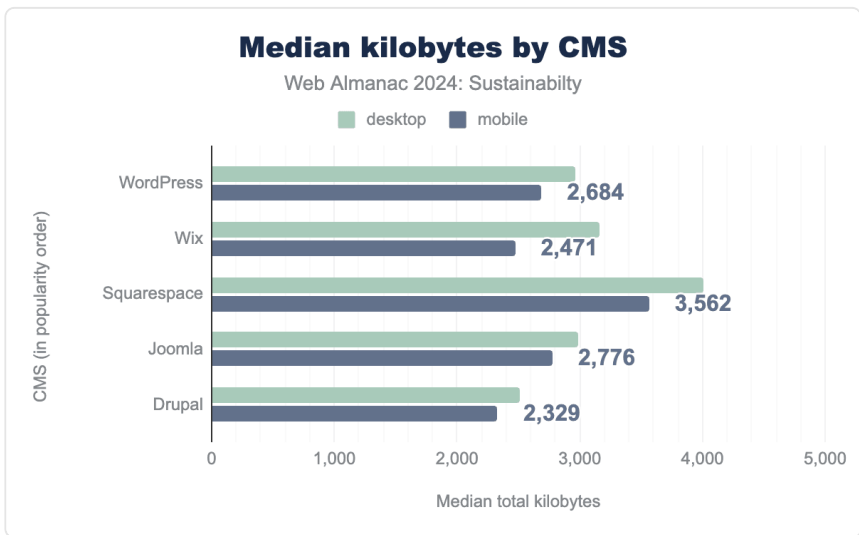


Figure 15.29. Median kB by CMS.

Notably, all report a median mobile page weight above the overall median of 2 MB. Comparing these values to their 2022 median size demonstrates that most websites built with each CMS are heavier than they were 2 years ago, which confirms a tendency that we noticed on global emissions for all pages. The more significant growth is for Squarespace websites on desktop and Drupal still seems to fare better than others (even if there is room for improvement).

Another point of interest across the segments is a noticeable difference in page size between desktop and mobile versions, which often stems from enhanced image optimization on mobile platforms. For instance, within the CMS segment, Wix and Squarespace display a substantial discrepancy in page size between desktop and mobile versions compared to other leading platforms.

As noted above, the model we use for calculating carbon emissions changed between 2022 and 2024. Based on SWD model V4, we recalculated emissions based on 2022 data for more relevant comparison with 2024 data. Here are both :

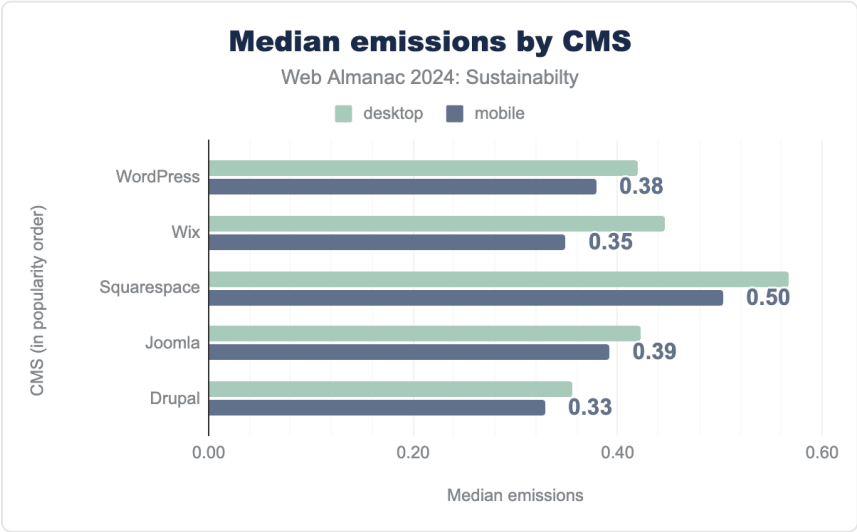


Figure 15.30. Median emissions by CMS - 2024.

For comparison reasons, here are the emissions data from 2022, calculated with the SWD V4 model :

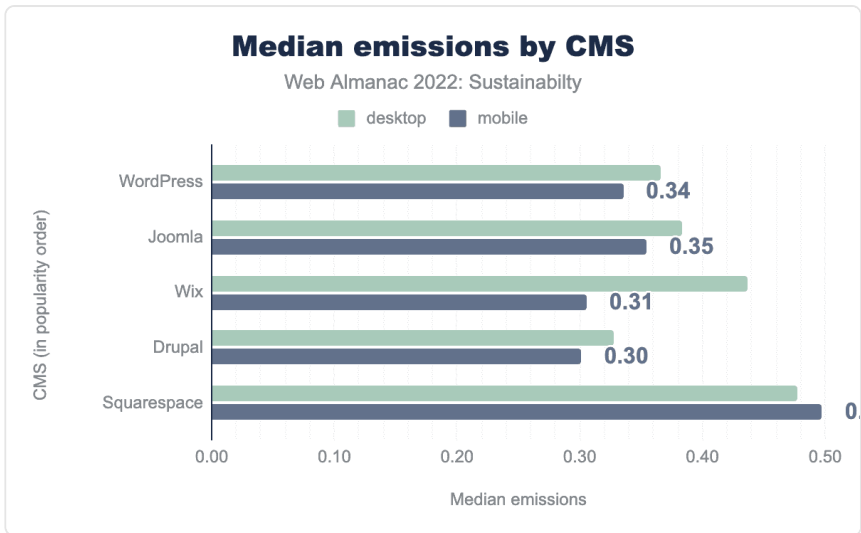


Figure 15.31. Median emissions by CMS - 2022.

The 2024 scan included around 5 million instances of WordPress, 500,000 instances of Wix, 250,000 instances of Squarespace, 250,000 instances of Joomla, and 200,000 instances of Drupal. That's over 6 million unique instances of these five CMS. So, even just generating this report does have a measurable climate impact.

The average of the median emissions in the chart above is about 0.3 grams of carbon. To run these queries on seven million URLs would produce about 2125 kg of CO₂. That's just from sampling the top five CMS. We know that this is just a small sampling of the CO₂ produced through the use of these websites.

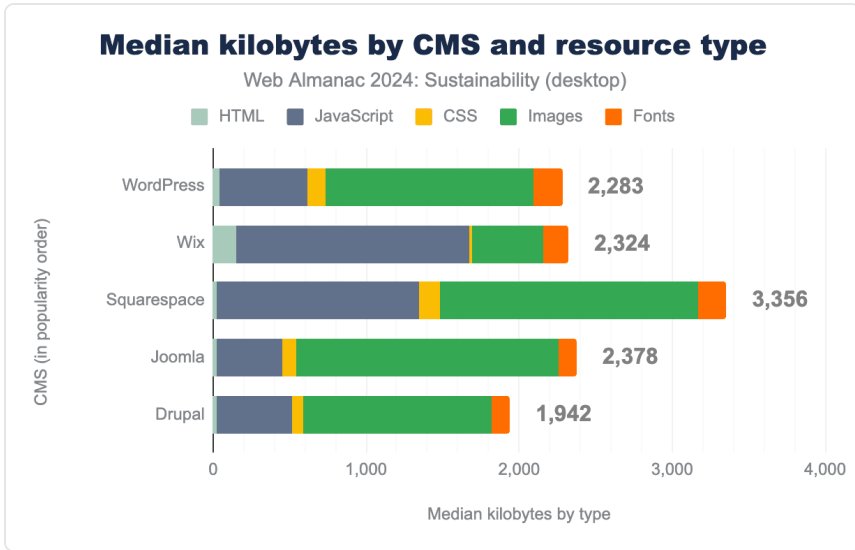


Figure 15.32. Median kB by CMS and resource type.

Note : in diagrams displaying resource types, we only take into account HTML+Javascript+CSS+Images+Fonts, omitting other requests such as video, audio and some “unidentified” 3rd-party services.

Of the five CMS, it is good to see that image sizes went down across the board. Squarespace is still the biggest with an average of 1.3 Mb of images on mobile devices. It is unfortunate to see JavaScript generally increase. Wix now has the largest JavaScript footprint in 2024 at 1.3 Mb. With most of the CMS, HTML is the smallest resource type. The charts above show that Wix seems to implement significantly more aggressive image optimizations on their platform.

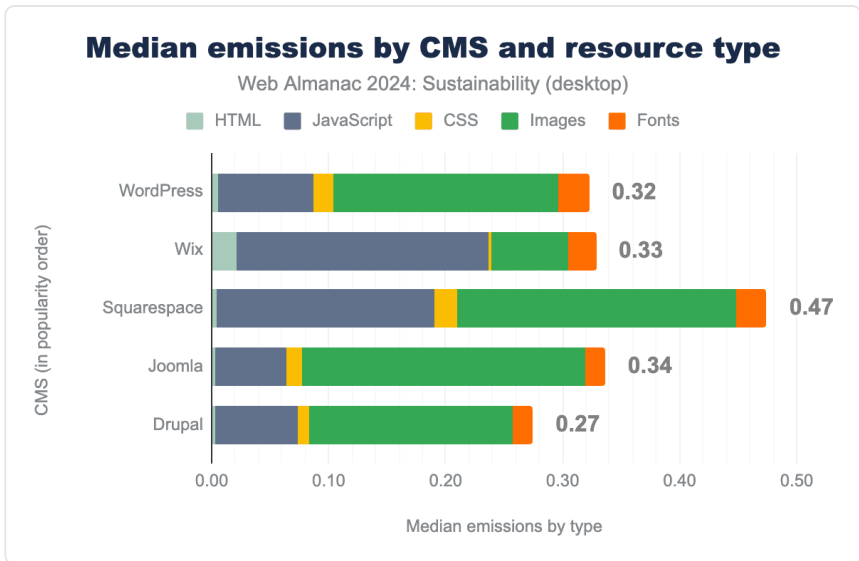


Figure 15.33. Median emissions by CMS and resource type.

It is useful to break down this information between resource types because it also indicates where there is waste. From a CO₂ perspective, it is hard to beat a simple and static HTML/CSS website. We are expecting richer, more interactive, and visual content on the web, but it wasn't always this way. Web fonts⁵⁹² became a part of the web only in 2009. In 2024 they are pretty much expected, seeing just how much custom fonts have become part of CMS implementations. The sites may look a bit more unique, but it comes at the cost of web performance and of course sustainability.

CMS also has a strong role to play in shifting the definition of quality work. Wagtail CMS is a leader⁵⁹³ in this as they have provided a release that focuses on reducing their carbon footprint. WordPress⁵⁹⁴, Drupal⁵⁹⁵, and Joomla⁵⁹⁶ have all made some sustainability initiatives. These communities can have a large impact on what is considered a good product. Through organizations like the Open Web Alliance⁵⁹⁷, there may be even greater collaboration to help implement best practices.

⁵⁹² <https://thehistoryoftheweb.com/web-fonts/>

⁵⁹³ <https://wagtail.org/blog/wagtail-greener-and-leaner/>

⁵⁹⁴ <https://make.wordpress.org/sustainability/>

⁵⁹⁵ <https://www.drupal.org/about/sustainability>

⁵⁹⁶ <https://magazine.joomla.org/all-issues/april-2024/green-websites-help-to-keep-your-feet-dry>

⁵⁹⁷ <https://www.drupal.org/association/blog/drupal-association-co-founds-the-open-website-alliance>

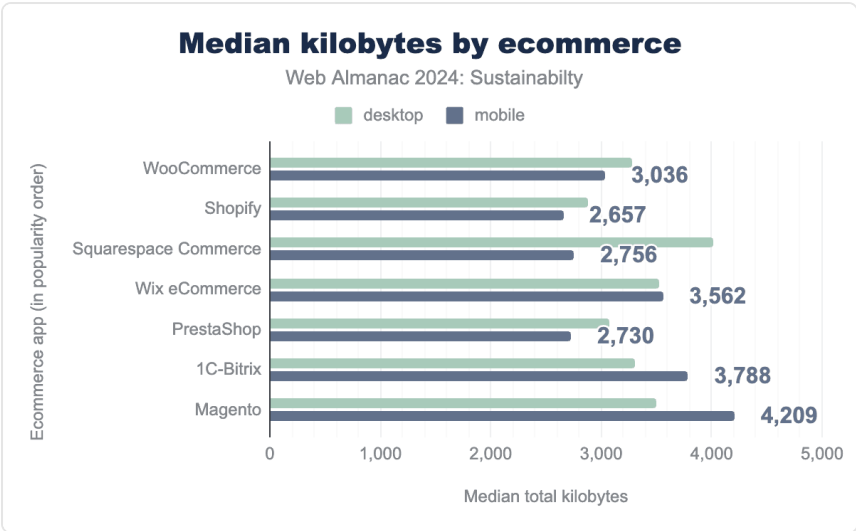


Figure 15.34. Median kB by ecommerce.

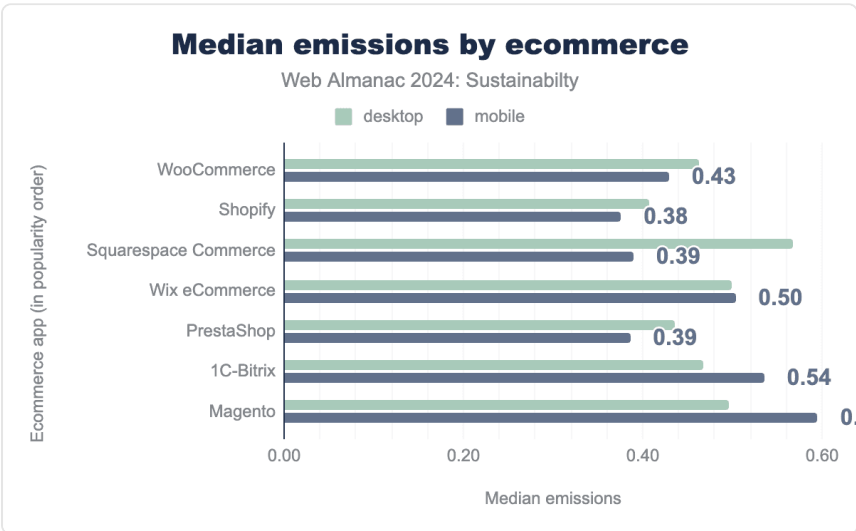


Figure 15.35. Median emissions by ecommerce - 2024.

For comparison reasons, here are the emissions data from 2022, calculated with the SWD V4 model :

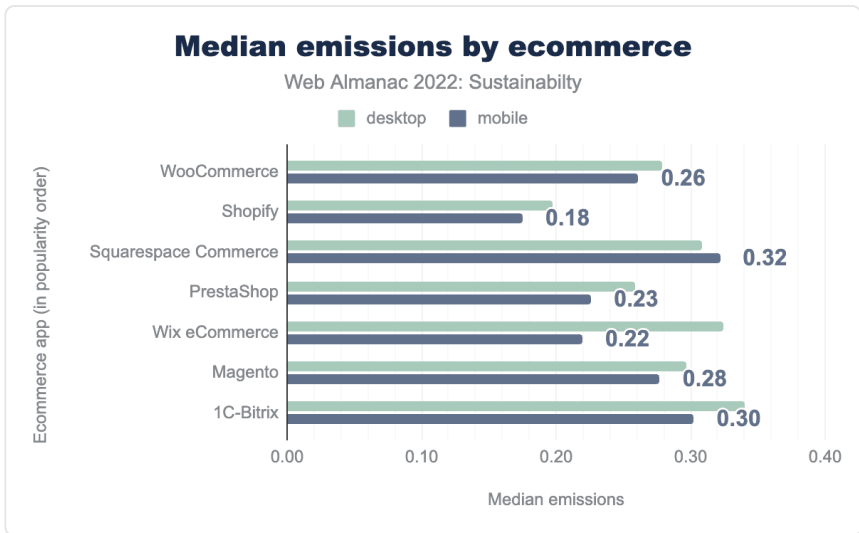


Figure 15.36. Median emissions by ecommerce - 2022.

It is clear comparing the ecommerce platforms with other CMS platforms, that they are substantially heavier in their page-load size and their environmental impact. Since these figures mostly relate to home pages, we can only guess that this could be due to more products or less optimized content. However, this shows that there is still room for improvement.

The Sustainable eCommerce Handbook⁵⁹⁸ could be a great starting point. In any case, you should keep in mind that most of the environmental impacts for eCommerce occur outside of websites (manufacturing, shipping, usage, and end-of-life of sold products, for instance).

598. <https://theecommanager.com/sustainable-ecommerce-handbook/>

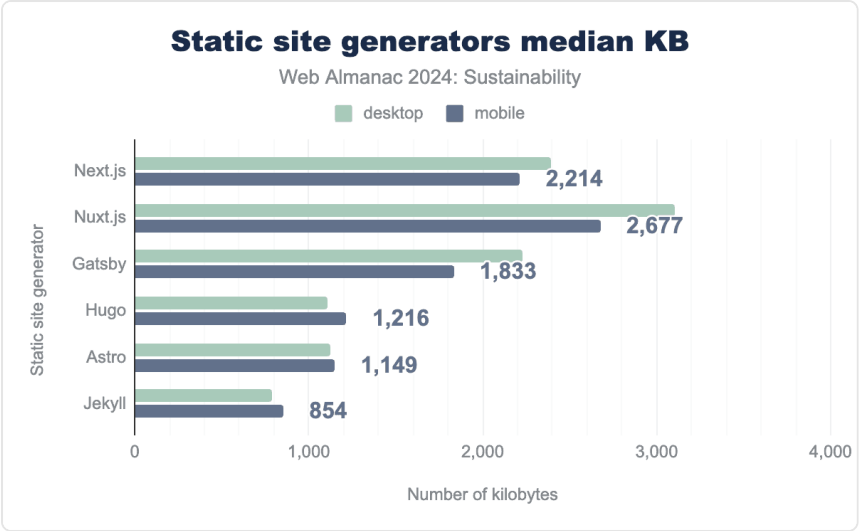


Figure 15.37. Median KB by static site generator.

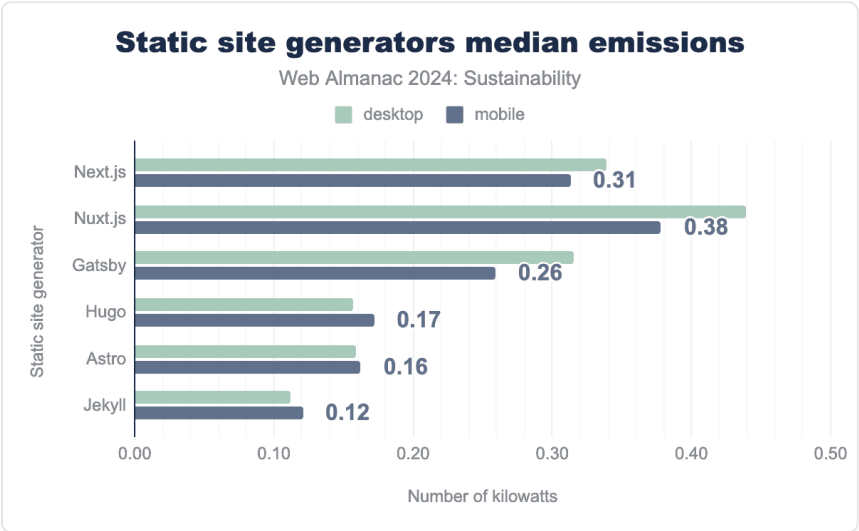


Figure 15.38. Median emissions by static site generator - 2024.

For comparison reasons, here are the emissions data from 2022, calculated with the SWD V4 model :

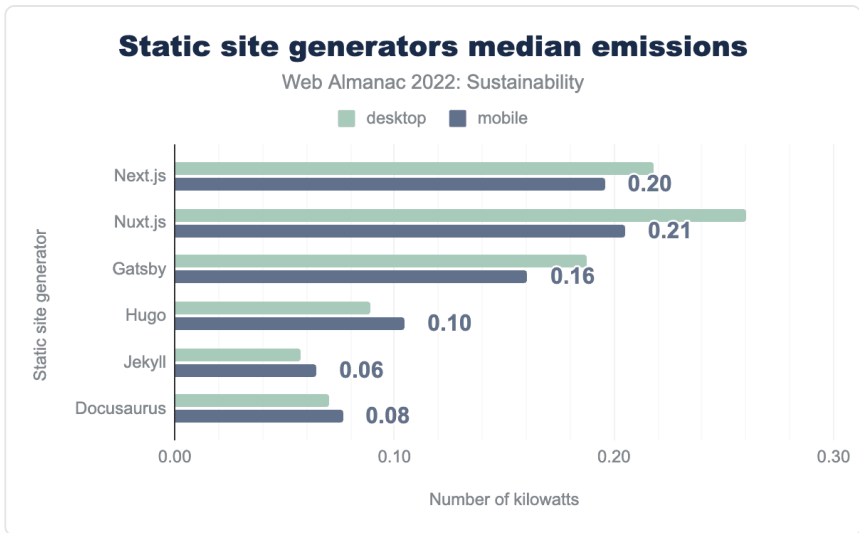


Figure 15.39. Median emissions by static site generator - 2022.

It appears evident that site generators have prioritized optimization as part of their builds. Astro, Hugo, and Jekyll all have emissions substantially lower than all the CMS that have been categorized.

These insights underscore the significant role that platforms and frameworks play in promoting sustainable web development. By setting effective defaults, platform creators and framework developers enable the construction of environmentally friendly websites right from the start.

These exceptions belong to the static site generator category, particularly Hugo and Jekyll, which typically support sites focused on blogs and text-heavy content with minimal JavaScript usage. Static site generators are also often chosen for their performance benefits, suggesting a greater likelihood of optimization beyond the norms of standard CMS-driven sites.

For further information, refer to:

- The WSG 1.0 - SC 3.7, 3.21, and 5.28.

Conclusion

Sustainability is a rapidly evolving and firmly established field that will increasingly have a seat at the table for those creating products and services for the Web. As with accessibility, legislation is helping to drive the need for conformance and the benefits it can bring to both

people and the planet. By considering how you create websites and applications, the ethical decisions behind design and development, taking into account industry best practices, and tackling the digital variables that can have real-world impacts, each one of us can become sustainability advocates.

Starting from the 2022 chapter, we realized that things are moving really fast for web sustainability and we are faced with many challenges:

- There is more and more information out there and it might get difficult to follow: new resources and tools, existing stuff being updated, etc.
- It appears essential to keep an eye on emerging technologies (metaverse faded away as fast as it appeared, AI is everywhere), since all of them have environmental impacts (and possibly benefits too). Solutionism is a growing risk:
 - Offering digital solutions to fight climate change without considering their own impacts.
 - Silver bullet offerings to reduce the environmental impacts of digital.
- Greenwashing is everywhere, even with the best intention in mind.

In the end, we see a generally wider adoption of technical best practices but still an increase of emissions. Efficiency is mandatory but sobriety and frugality are the way to go.

Actions you can take

Creating sustainable websites requires a combination of understanding, planning, and action. While the challenge may seem daunting, there are numerous practical steps you can take today to begin or advance your sustainability journey. From quick assessment tools to comprehensive guidelines, and from planning frameworks to continuous learning resources, this section outlines key approaches that can help you make meaningful progress. Remember that sustainability is not about achieving perfection immediately, but rather about making consistent improvements and informed decisions that collectively reduce our environmental impact.

Quick checks

One of the simplest steps you can take in rapidly evaluating your sustainability situation is using an automated measurement tool. As with accessibility tooling, these only tell part of the story and cannot account for aspects that can only be human-tested (rather than automated), and there are still many issues we do not have accurate data to measure against; but it's still a good

place to dip your toes in the water.

For further information, refer to:

- Ecograder⁵⁹⁹
- Website Carbon⁶⁰⁰
- Are My Third Parties Green?⁶⁰¹

Planning and reporting

Another step forward you can take beyond a rapid evaluation is to put forward some concrete plans or even better, start reporting on your sustainability journey. Measure what you can, document your efforts, and be as open with your findings and progress as you can be. Sustainability is a journey and progress is always beneficial over perfection. Having a sustainability statement will be a great place to provide such plans and successes you have achieved.

For further information, refer to:

- Co2.js⁶⁰²
- How to Write An Effective Sustainability Statement⁶⁰³

WSG

The W3C Sustainable Web community group's Web Sustainability Guidelines are a hugely beneficial resource for anyone seeking to implement sustainability in their website or application. The specification they have produced is broken down into sections based on expertise (UX, Web Development, DevOps, and Business) and additional resources are available (linked to at the top of the specification) which can assist in your understanding of and implementation of sustainability within your product or service.

For further information, refer to:

- Web Sustainability Guidelines⁶⁰⁴

599. <https://ecograder.com>

600. <https://websitescarbon.com>

601. <https://aremythirdpartiesgreen.com/>

602. <https://www.thegreenwebfoundation.org/co2-js/>

603. <https://ecocart.io/sustainability-statement/>

604. <https://w3c.github.io/sustyweb/>

- Sustainable Tooling And Reporting (STAR)⁶⁰⁵
- At-A-Glance Overview⁶⁰⁶
- Introduction to Web Sustainability⁶⁰⁷
- Web Sustainability Laws & Policies⁶⁰⁸
- WSG Quick Reference (Includes PDF)⁶⁰⁹

Up-to-date knowledge

As creators, keeping our knowledge current in an overwhelming industry is a challenge, and in sustainability which is a rapidly evolving field, this is no exception. Reading books, watching relevant videos, taking courses, looking at studies or papers, and keeping up-to-date with the latest standards and best practices are essential. You have taken the first step with this chapter of the Web Almanac, if you enjoyed it, see what else interests you to expand your experience further.

Authors



Laurent Devernay Satyagraha

 ldevernay  laurent-devernay-satyagraha-2610b85  <https://ldevernay.github.io/>

Laurent Devernay Satyagraha is a Digital Sobriety Expert for Greenspector⁶¹⁰. You can find him blogging on his own⁶¹¹ or for this company⁶¹² but almost always about web sustainability. Which makes him either an enthusiast or a monomaniac. Connect with Laurent on LinkedIn⁶¹³.

605. <https://w3c.github.io/sustyweb/star.html>

606. <https://w3c.github.io/sustyweb/glance.html>

607. <https://w3c.github.io/sustyweb/intro.html>

608. <https://w3c.github.io/sustyweb/policies.html>

609. <https://w3c.github.io/sustyweb/quickref.html>

610. <https://greenspector.com/en/home/>

611. <https://ldevernay.github.io/>

612. <https://greenspector.com/fr/blog/>

613. <https://www.linkedin.com/in/laurent-devernay-satyagraha-2610b85/>



Burak Güneli

✉ @burakGuneli 🌐 burakguneli

Burak is Frontend Software Engineer who strive to understand how things work under the hood to unravel their mysteries, especially in Javascript. He is based in Berlin and if you're also living in Berlin, there's a good chance you might bump into each other at an indoor cycling class.



Ines Akrap

✉ @InesAkrap 🌐 ines-akrap 🌐 https://inesakrap.com/

Ines Akrap is a Frontend Software Engineer passionate about optimizing websites to be fast, sustainable, and provide the best user experience for every user. She works in Storyblok as a Solutions Engineer. She enjoys sharing her knowledge through talks, podcasts, workshops, and courses.



Alexander Dawson

✉ @AlexDawsonUK 🌐 @https://mastodon.design/@AlexDawsonUK 🌐 AlexDawsonUK

🌐 alexdawsonuk 🌐 https://alexanderdawson.com/

Alexander Dawson is a Web Developer, Sustainability Researcher, and editor of the Web Sustainability Guidelines⁶¹⁴. You can find more details about him at alexanderdawson.com⁶¹⁵.



Mike Gifford

🌐 @https://mastodon.social/@mgifford 🐦 @mgifford.bsky.social 🌐 mgifford 🌐 mgifford

🌐 https://accessibility.civicaactions.com/

Mike Gifford is CivicActions' Open Standards & Practices Lead. He is also a thought leader on open government, digital accessibility and sustainability. He has served as a Drupal Core Accessibility Maintainer and also a W3C Invited Expert. He is a recognized authoring tool accessibility expert and contributor to the W3C's Draft Web Sustainability Guidelines (WSG) 1.0.

⁶¹⁴. <https://w3c.github.io/sustyweb/>

⁶¹⁵. <https://alexanderdawson.com/>



Tim Frick

✉ [@timfrick](https://twitter.com/timfrick)  [timfrick](https://github.com/timfrick)  <https://www.mightybytes.com/>

Tim Frick is President of Mightybytes⁶¹⁶, a digital agency and Certified B Corp located in Chicago. A seasoned public speaker, he regularly presents on sustainable design, measuring impact, and problem solving in the digital economy. Tim is also the author of four books, including *Designing for Sustainability, A Guide to Building Greener Digital Products and Services*⁶¹⁷, from O'Reilly Media. Connect with Tim on LinkedIn⁶¹⁸.

616. <https://mightybytes.com>

617. <https://www.oreilly.com/library/view/designing-for-sustainability/9781491935767/>

618. <https://www.linkedin.com/in/timfrick/>

Part IV Chapter 16

Page Weight



Written by Dave Smart and Jamie Indigo

Reviewed by Ines Akrap

Analyzed by Burak Güneli

Edited by Montserrat Cano

Introduction

The internet is growing at a rapid pace. Each new page brings with it a bespoke set of resources necessary to render its content, which is expensive as more computing resources are required. These bandwidth requests are competing with the computing resources of generative AI initiatives.

In the United States, the rapidly growing AI demand is poised to drive data center energy consumption to about 6% of the nation's total electricity usage in 2026⁶¹⁹, adding further pressure on grid infrastructures and highlighting the urgent need for sustainable solutions to support continued AI advancement. These generative AI initiatives will in turn rapidly increase the size of the web. Statista estimates 149 zettabytes⁶²⁰ of internet content were created in 2024. In comparison, the years from 2010 to 2018 produced a combined 127.5 zettabytes.

⁶¹⁹. <https://hbr.org/2024/07/the-uneven-distribution-of-ais-environmental-impacts>

⁶²⁰. <https://www.statista.com/statistics/871513/worldwide-data-created/>

In short, resources are becoming increasingly scarce and expensive. With Google now prioritising on-page elements, addressing the issue of page weight has become important. Reducing unnecessary bloat in websites not only enhances user experience and boosts conversions, but also supports sustainability efforts.

As highlighted in discussions about web performance in 2024, heavy websites contribute to inequalities in user access and responsiveness, particularly on lower-end devices, widening the “performance inequality gap.” Alex Russel’s series, *The Performance Inequality Gap*⁶²¹ bring into sharp focus that some of the assumptions that are made on current device performance and capabilities may not be true, and that whilst that devices might be getting more and more powerful, that’s not true for everyone, and there’s a long tail of users who are negatively impacted by web pages with large payloads.

This growing disparity emphasizes the importance of lightweight, efficient web design to ensure equitable access and engagement for all users. Page weight matters, whether you’re experiencing a weak network connection at an inopportune moment or live in a market where access to the internet is charged by the megabyte, inflated page weight decreases the availability of information.

Page weight is an accessibility issue

Large page weight disproportionately affects users who cannot afford top end devices, and fast, high data usage cap connections.

Bloated pages mean that people without access to these have a more expensive, less performant experience of the web, and in extreme cases might even make a page practically unusable.

What is page weight?

Page weight is the byte size of a web page. The web has evolved massively since its birth, and page weight in 2024 isn’t just the HTML from the URL you arrive at. In nearly all cases, it involves the assets needed to load and display that page. Those assets include the following:

- The HTML that comes in the initial response from a server.
- Images and other media (video, audio, etc) that are embedded into the page.

621. <https://infrequently.org/2024/01/performance-inequality-gap-2024/>

- Cascading Style Sheets (CSS)⁶²² for styling the page.
- JavaScript to provide interactivity and functionality.
- Third-Party resources, which can be one or more of the above, from other providers.

Every extra thing added to a web page increases the overall page weight, and every bit ultimately means more work and overhead for the browser in transmitting it across the network, processing, parsing and ultimately rendering and painting it on the screen for the user to consume and interact with.

Some forms of resources carry even greater overheads, especially JavaScript, which also needs to be compiled and executed. This also has a knock-on effect on both sustainability and conversion rates. The heavier a page is, the more carbon emissions and the least possibility of conversions on that page.

You can find out more about how page weight impacts carbon emissions in the Sustainability chapter.

There are various mitigations available to help manage page weight, and its overall effect on load times, but the stark reality is that more weight is always going to involve more work.

The weight effects can be divided into three main categories: storage, transmission, and rendering.

Storage

Every byte of a web page needs to be stored somewhere, and with the nature of how the web works, that usually means being stored in multiple locations

It starts with the web server itself. Pure storage space remains relatively small in cost per Gigabyte, depending on the type of storage. For example, Google's cloud storage is somewhere between \$0.02 and \$0.03 per month in North America⁶²³ or \$0.006 and \$0.025 in Europe. Resources stored in memory on a web server versus on disk for faster access will ramp up in cost far faster than one that lives on disk.

There can be multiple copies of the same resource too, spread across a number of intermediate caches, and even spread across multiple data centers if a CDN when edge caching is employed.

The second part of the equation is these resources also need to be stored on the user's device

622. <https://developer.mozilla.org/Web/CSS>

623. <https://cloud.google.com/storage/pricing#north-america>

when they access a page. Lower-end devices, particularly mobile ones, may be far more restricted as to how much they can hold. Pushing large payloads can overwhelm storage capacity, pushing other valuable resources to be purged from the cache. This can lead to additional costs and performance hits when navigating to a new page that would have reused those resources.

Transmission

The first time you visit a site, all the resources need to be delivered across the internet to your device. Subsequent visits to the same URL, or even other URLs on the same site, might mean that some of the resources can be used from cache, but a significant amount might still need to be retrieved again across the network.

Not all network connections are equal everywhere, it could be a super-fast broadband connection with generous data limits, or it could be a metered, capped slow mobile connection.

So, it is best to think strategically. The bigger the page weight, the longer the transmission of resources will take, and those with slower mobile connections or low data limits will be hit the hardest, which may also affect business.

The best way to optimize the transmission of resources is by serving small resources. In case that is difficult to achieve, using resource hints⁶²⁴ (like preconnect and preload) and fetch priority⁶²⁵ can help with managing the order resources are loaded on page.

Rendering

Before a browser can paint the URL requested onto someone's screen, it needs to gather and process those resources.

The greater the page weight, the longer it will take a browser to get and process all the parts needed, delaying the point where users can read and interact with the page.

Even after loading, excessive page weight can make a page slow to respond to interaction, as the browser is bogged down shuffling large resources.

Page weight by the numbers

The internet blossomed from a place of bare text to the rich, interactive landscape we know

624. <https://web.dev/learn/performance/resource-hints>

625. <https://web.dev/articles/fetch-priority>

today by introducing new content types. Images introduced visual depth, Javascript enabled interactivity, and videos introduced new ways of storytelling.

Each of these technologies also brought more weight to their pages. Before the introduction of HTML 2.0 in 1995, the only asset to weight was HTML, Page weight dramatically increased when RFC 1866⁶²⁶ introduced the `<img>` tag. In 1996, JavaScript stepped on the scale followed by libraries like JQuery a decade later. The first widely recognized single-application framework emerged in 2010 opening the door for JavaScript frameworks like Angular, React, Vue and others to come to market.

Each evolution of page functionality brings with it more weight and file types intended to improve performance while retaining functionality.

We have analyzed common file types, their occurrence and response size to better understand their application. This includes comparisons by device and page type.

File type requests for the median page

To understand the file types associated with page weight, we should look at file type requests for the 50th percentile of pages. This provides a baseline for the impact of each file type overall.

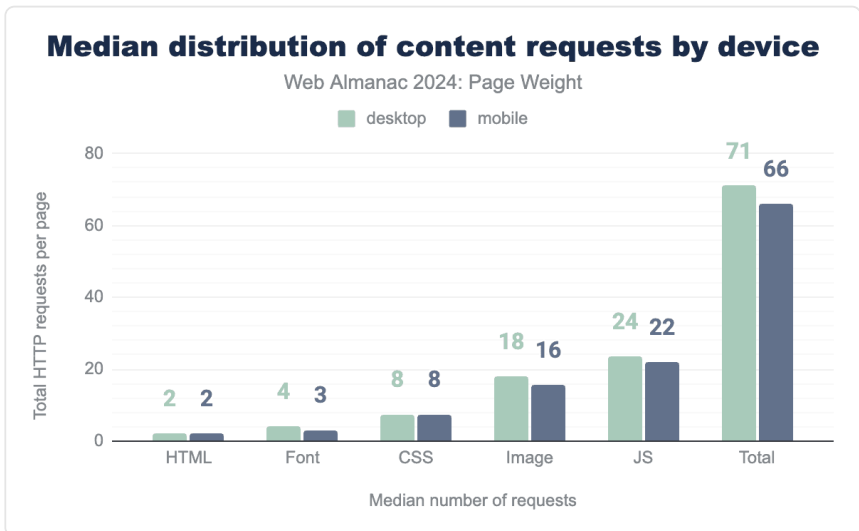


Figure 16.1. Median number of requests by content type and device type.

626. <https://www.rfc-editor.org/rfc/rfc1866>

The total number of requests for desktop pages decreased by 9%, down to 71 from 2022's 76 requests per page. Similarly, mobile decreased from 70 to 66 requests. The count of image file types decreased by 43% in 2024

Desktop pages requested 18 images in 2024 compared to 25 in 2022.

Javascript overtook images as the most requested file type. The median page requested 24 Javascript resources on desktop pages. Mobile saw 22 requests.

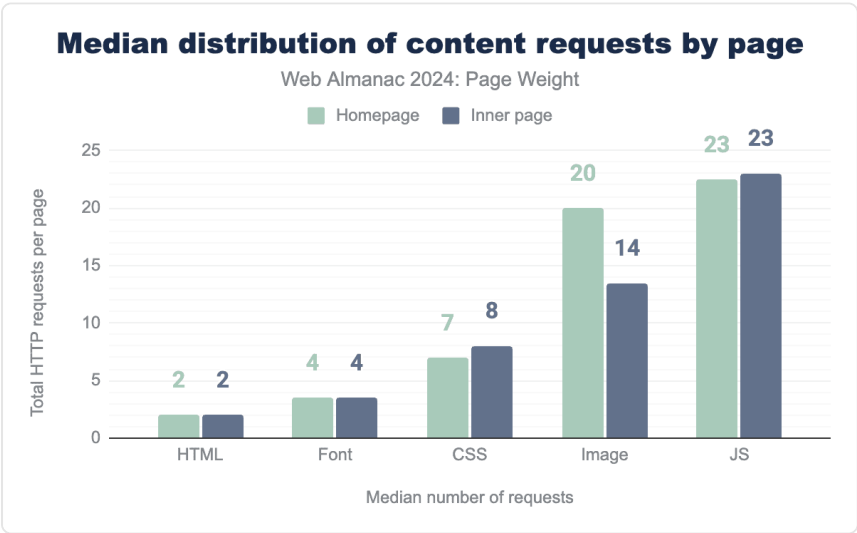


Figure 16.2. Median number of requests by content type and page type.

As we begin dissecting behavior patterns for page weight, it is important to note the impact of inner pages. Throughout this chapter, you will find strong variances that are only visible when homepages are compared to inner pages.

This is noteworthy as our new data will change device type comparisons due to the significant variances. With this in mind, we have also included comparisons filtered to homepages for a more accurate measure to 2022's data set.

Requests volume

Each request a page makes is a component needed to create the intended experience and content it provides. The total number of requests, the number of pieces needed to complete the build, impacts page performance.

Modern browsers are multi-threaded and multi-process. This means they can utilize multiple threads and processes to handle different tasks, including network requests. Each request requires resources to execute, and due to their technical limitations, only a limited number of requests can be completed simultaneously. Like humans, browsers can only do a limited number of things at once.

With this knowledge, the number of requests impacts both page weight and perceived performance.

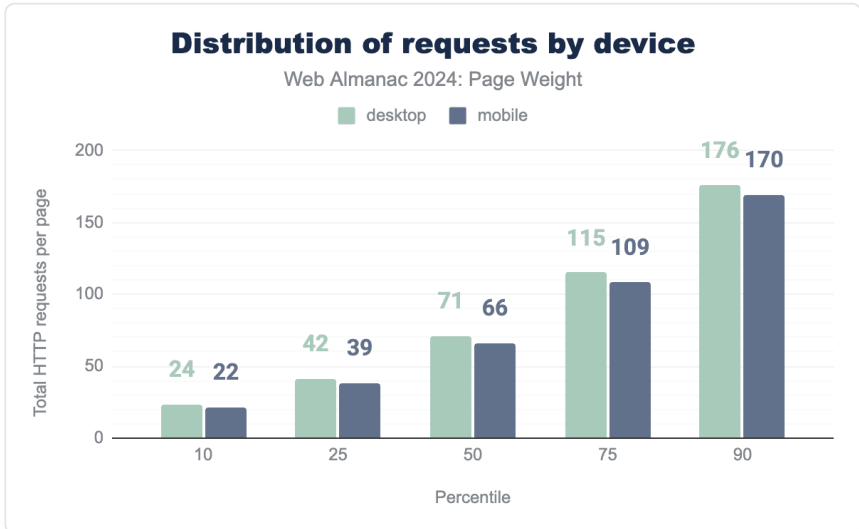


Figure 16.3. Distribution of requests by device type.

The median page makes 71 requests on desktop and 66 on mobile. These numbers include both home and inner pages. When compared to the 2022 distribution, all percentiles show a decrease in the total number of files.

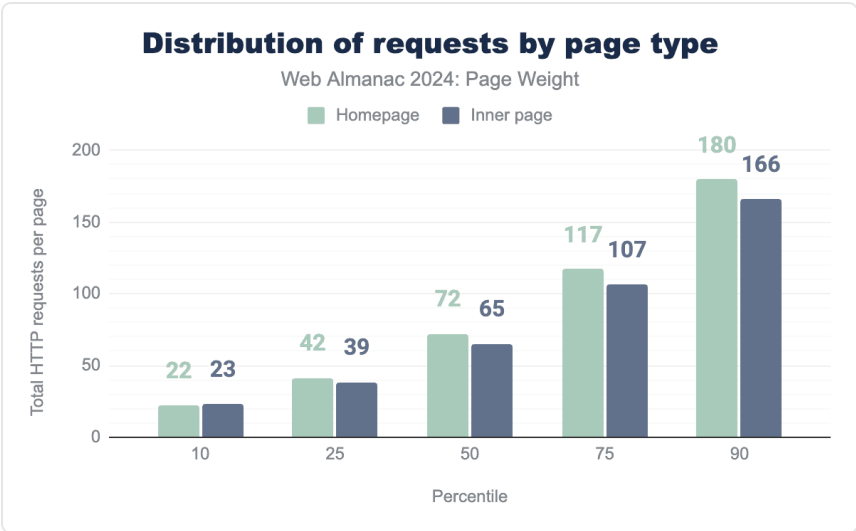


Figure 16.4. Distribution of requests by page type.

The lower all over numbers are impacted by the consistently lower number of requests made by inner pages when compared to homepages. The median homepage calls 72 resources while its inner counterpart requires only 65.

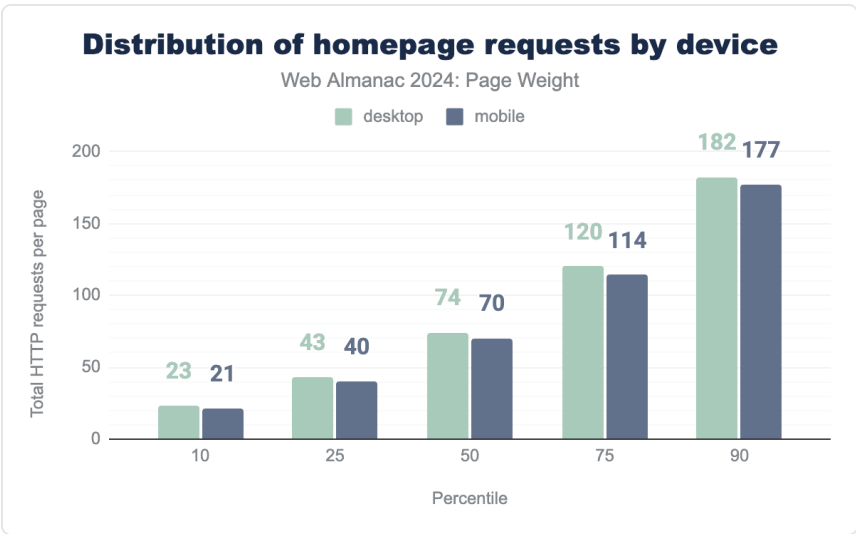


Figure 16.5. Distribution of requests by homepages by device type.

When analyzing only homepages for data congruity, total requests were slightly down in 2024 but consistent with 2022 rates. The median desktop homepage requested 76 resources in 2022 and now requests 74. The mobile median remains unchanged.

Images

Images are static files that are essential for constructing and displaying web pages. As the web becomes increasingly visual, they exemplify the need to balance performance-enhancing technologies with asset byte size.

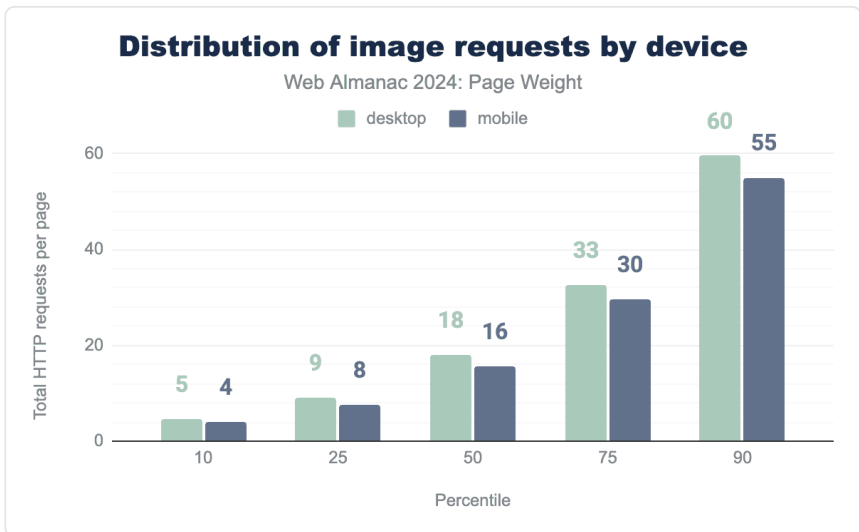


Figure 16.6. Distribution of image requests by device type.

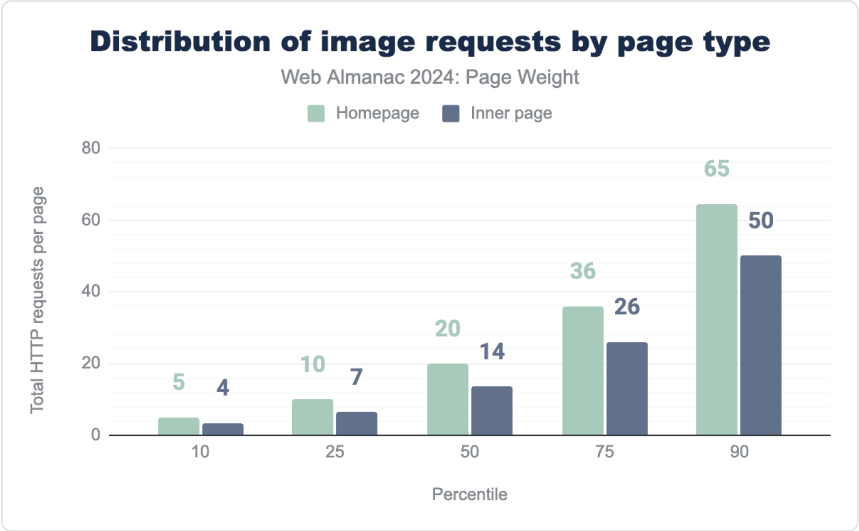


Figure 16.7. Distribution of image requests by page type.

In 2022, we saw the median page request 25 images for desktop and 22 for mobile pages. This is down to 18 for desktop and 16 for mobile.

Decreased image file types does not mean that the web has become less visual. Instead, sites may be switching to CSS effects (such as shadows⁶²⁷ or gradients⁶²⁸) and CSS animations⁶²⁹. These assets can be used to produce resolution-independent assets that always look sharp at every resolution and zoom level, often at a fraction of the bytes required by an image file.

Desktop pages consistently call for more image file types with the gap between desktop and mobile growing steadily and consistently across percentiles. The difference between homepage and inner pages was striking in comparison. Where device type saw relatively consistent numbers, the median homepage called for 20 images compared to just 14 for inner pages.

14,974

Figure 16.8. Image requests made on desktop pages at the 100th percentile.

627. https://www.w3schools.com/css/css3_shadows.asp

628. <https://developer.mozilla.org/Web/CSS/gradient>

629. <https://web.dev/articles/animations-guide>

CSS

CSS, or Cascading Style Sheets, is a style sheet language used to describe the presentation of a document written in a markup language like HTML. In other words, CSS is responsible for the visual styling and layout of web pages.

It allows developers to control the color, font, size, spacing, and many other visual aspects of HTML elements. CSS works in conjunction with HTML, providing a separation of content and presentation.

This separation makes web pages more maintainable, flexible, responsive, and can be used to make a site more performant but substituting byte-heavy image assets with CSS effects and animations.

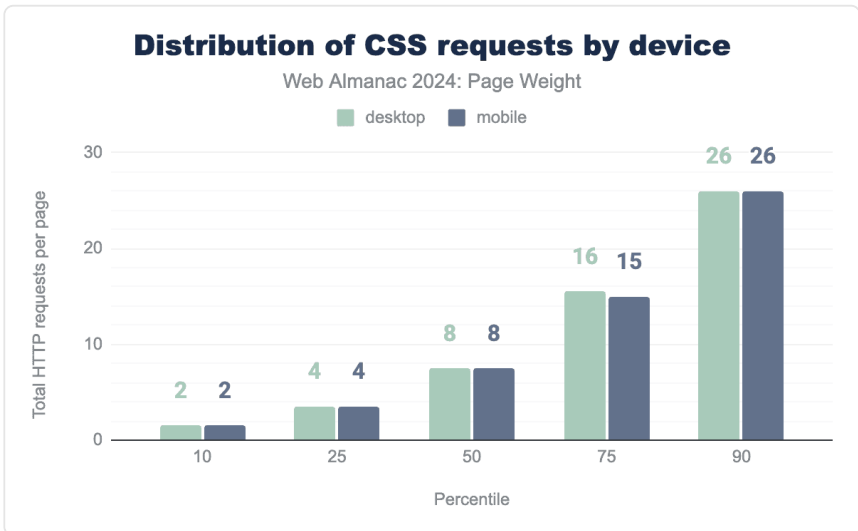


Figure 16.9. Distribution of CSS file requests by device type.

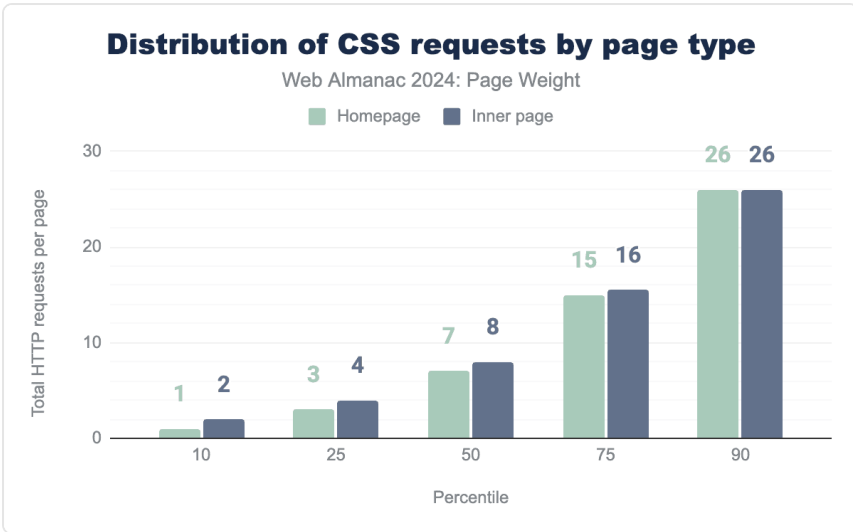


Figure 16.10. Distribution of CSS file requests by page type.

CSS is an essential tool in the web developer’s toolkit across devices and page types. The median desktop and mobile page both called for 8 CSS assets. Percentiles were identical except for a nominal variance at the 75th.

In comparing homepages to inner pages, we saw that homepages consistently called one fewer cascading style sheets until the 90th percentile. At the 100th percentile, we saw inner pages deviate with a spike of 4,879 requests compared to 3,346 on inner pages. While both are high, inner pages are 46% higher.

JavaScript

JavaScript is a high-level, dynamic and interpreted programming language. It is one of the core technologies of the web, enabling interactive web pages and web applications. JavaScript allows developers to add interactivity, animations, and effects to web pages. This includes features such as drop-down menus, image sliders, personalized content, and analytics tracking.

97.8%

Figure 16.11. Mobile home pages using JavaScript.

It is used as a client-side programming language by 97.8% of all mobile home pages, and 98.5%

of inner pages.

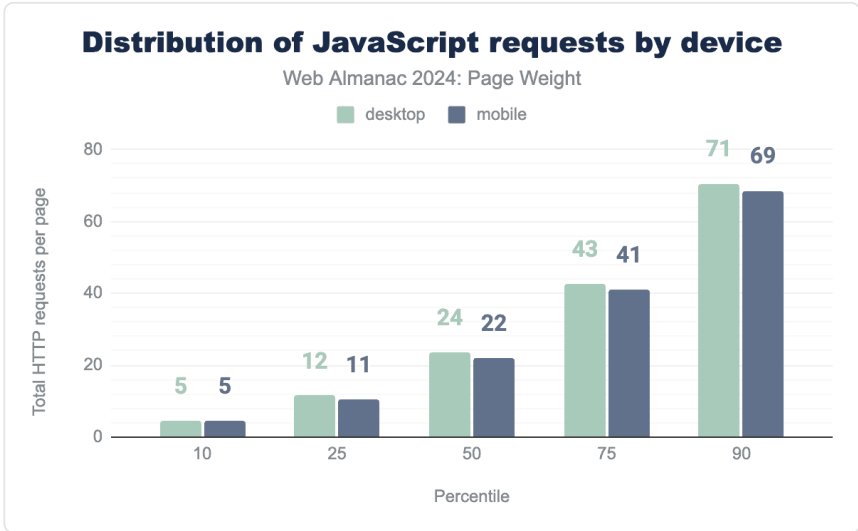


Figure 16.12. JavaScript request distribution by device type.

2024 saw JavaScript overtake images as the dominant file type. The median page requested 24 JS files for desktop and 22 for mobile pages. This is up 8% for desktop and 4.5% for mobile when compared to 2022. The number of JavaScript requests was consistent between inner and homepages through the 90th percentile.

At the 100th percentile, desktop pages and homepages broke away from their counterparts in the number of requests. Desktop pages made 33% more requests; homepages made 31% more. Desktop homepages made requests for 12,676 JavaScript resources. We attempted to reach the page for comment, but the request was still loading at time of publication.

For more information on how JavaScript is being used in 2024, take a look at the JavaScript chapter.

Third-party services

Third-party resources are external assets or services that are integrated into a web page or application, but are hosted and maintained by a different provider. These resources can include things like JavaScript, CSS, fonts, and analytics tools, to name a few. According to the Third Parties chapter, 92% of pages had one or more third party resources. The most called third party resources were scripts, making up 30.5% of requests by content type. The authors also noted a considerable decrease in the number of third parties for lower-ranked websites.

For more insights, refer to the Third Parties chapter.

Other assets

Web pages can utilize a variety of other assets and resources beyond just code, styles, and images. These additional assets contribute to the overall functionality, interactivity, and visual appeal of a web page, working in harmony with the HTML, CSS, and JavaScript to create a complete user experience.

HTML

HTML, or Hypertext Markup Language, is the standard markup language used to create and structure web pages. It provides the foundation for the content and layout of websites, defining elements like headings, paragraphs, lists, links, images, and more. HTML uses a series of tags and attributes to describe the semantic meaning and visual presentation of web page content.

There are several reasons why a page may include more than one HTML request for a single web page, including:

1. **Embedded Resources:** A web page typically loads not just the HTML document, but also additional resources like images, CSS files, JavaScript files, fonts, etc. Each of these external resources will trigger a separate HTTP request to the server to fetch that content.
2. **Dynamically Loaded Content:** Some web pages use JavaScript to dynamically load additional content or data after the initial page load. This could be things like infinite scrolling, AJAX-powered content updates, or lazy-loading of elements. These dynamic requests are in addition to the initial HTML document request.
3. **Preloading/Prefetching:** Web developers may include `<link>` tags with `rel="preload"` or `rel="prefetch"` to instruct the browser to proactively fetch certain resources in advance before they are actually needed. This can improve perceived performance.
4. **Error Handling:** If there are any network errors or server issues when loading a resource, the browser will retry the request, leading to multiple requests for the same content.

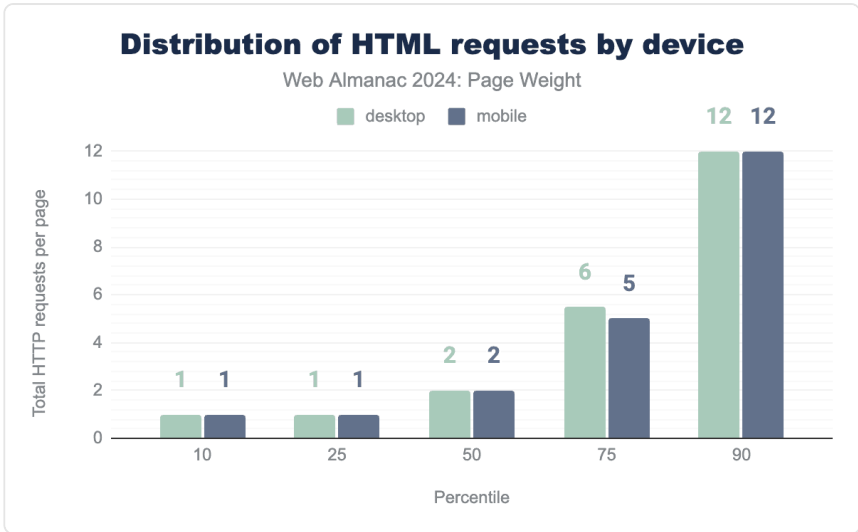


Figure 16.13. HTML requests distribution by percentile.

The median page made two HTML requests., which was consistent across devices and page types. At the 90th percentile, we saw 12 HTML requests. The number spiked dramatically at the 100th percentile, where desktop homepages made 13,389 requests.

Fonts

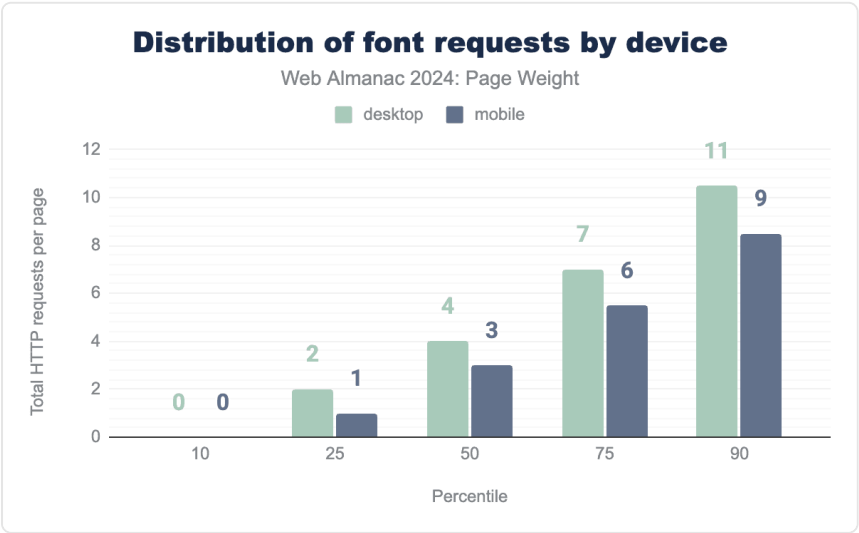


Figure 16.14. Font requests distribution by percentile.

The median page requested four font files. Through the 90th percentile, font requests remained fairly low and consistent across device and page types. At the 100th percentile, we saw desktop homepages request 3,038 fonts. With that volume of font requests, we speculate this site to be a font repository or a ransom note generator.

Request bytes

Comparing the median page weight over time shows that unfortunately it continues to grow, almost at the same rate. The median page weight is still increasing at almost the same rate, as shown by a comparison over time.

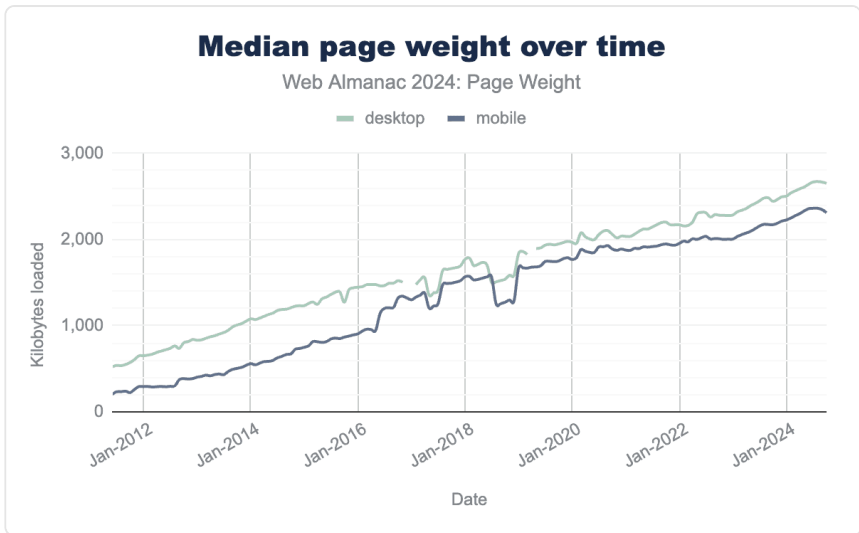


Figure 16.15. Median page weight over time.

The median page weight for a desktop page, as measured in October 2024 is 2,652 KB, for mobile it's a slightly lower, but still weighty 2,311 KB.

Compared to 2022's chapter, both figures are higher, with the median desktop page being 2,312 KB and for mobile it was 2,037 KB. 2024's Mobile page is just 1 KB lighter than 2022's desktop page. In October 2024, the median page weight for a desktop page was 2,652 KB, while the median page weight for a mobile page was 2,311 KB.

Both of these figures are higher than those from 2022. In 2022, the median page weight for a desktop page was 2,312 KB, and the median page weight for a mobile page was 2,037 KB. Notably, 2024's mobile page weight is only 1 KB lighter than 2022's desktop page weight.

1.8 MB

Figure 16.16. How much larger the median mobile page weight has grown in 10 years.

When we compare year to year, desktop grew 8.6%, or 210 KB from Oct 2023 to Oct 2024, and mobile grew 6.4%, or 140 KB.

The median desktop page has increased by 120%, or 1.4 MB, over the past 10 years. The median mobile page has seen a more significant increase of 357%, or 1.8 MB, during the same period. This equates to adding more than a 3.5" floppy disk's worth of data to mobile pages.

Year-over-year, from October 2023 to October 2024, the desktop grew by 8.6%, or 210 KB, and mobile grew by 6.4%, or 140 KB.

According to What Does My Site Cost?⁶³⁰ a web based tool for calculating the cost of web data to end users, the median desktop page could cost a user up to \$0.32 USD, or in some regions up to 1.7% of their Gross National Income.

Content type and file formats

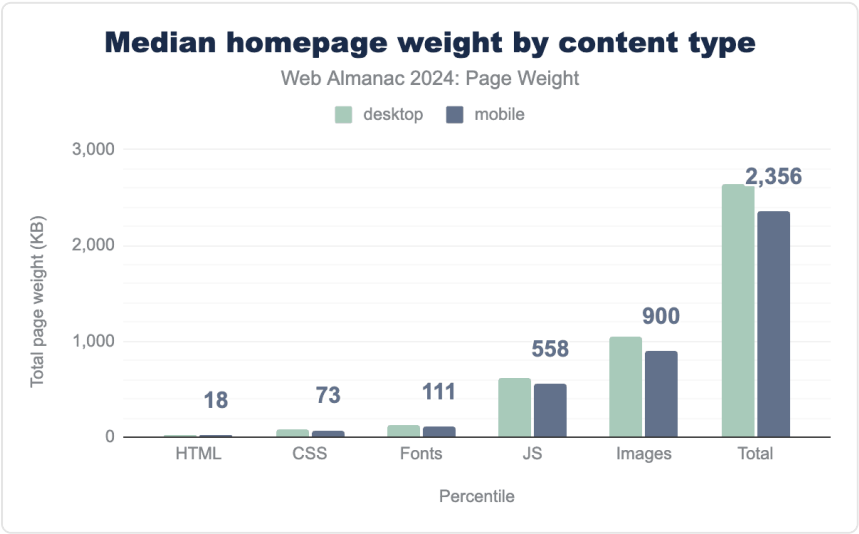


Figure 16.17. Median homepage weight by content and device type.

The predominant resource type for homepages, excluding video, is images. The median desktop page using images requests 1,054 KB, while mobile pages request 900 KB. This shows a small increase from 2022, where desktop pages requested 1,026 KB and mobile pages requested 900 KB.

JavaScript was the second largest contributor to page weight, with the median desktop page serving 613 KB, on mobile pages it's 558 KB. Like images, these both represent growth from 2022's chapter, where it was 509 KB on desktop pages and 461 KB on mobile pages.

630. <https://whatdoesmysitecost.com/#usdCost>

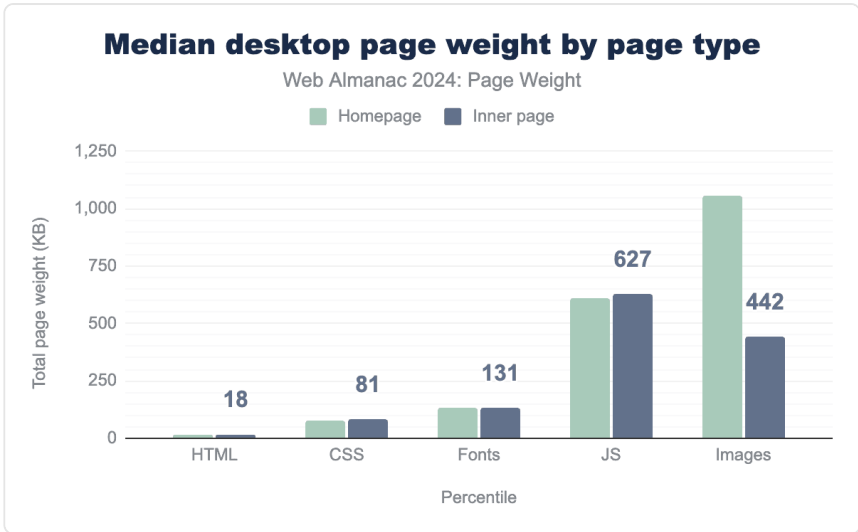


Figure 16.18. Median desktop page weight by page and content type.

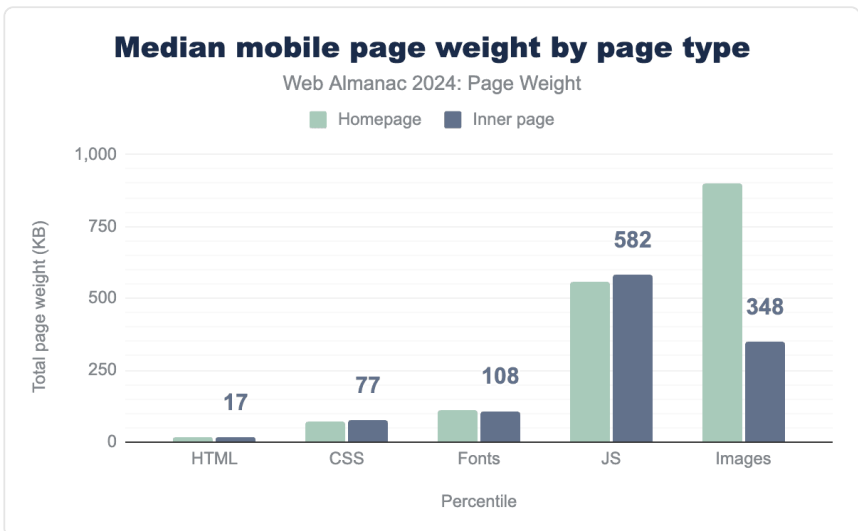


Figure 16.19. Median mobile page weight by page and content type.

For both mobile and desktop pages, inner pages tend to have less bytes of images, and slightly more bytes of JavaScript.

This new analysis also shows that images are not always the biggest component of page weight, as previously thought, and that for inner pages JavaScript took that dubious honor instead.

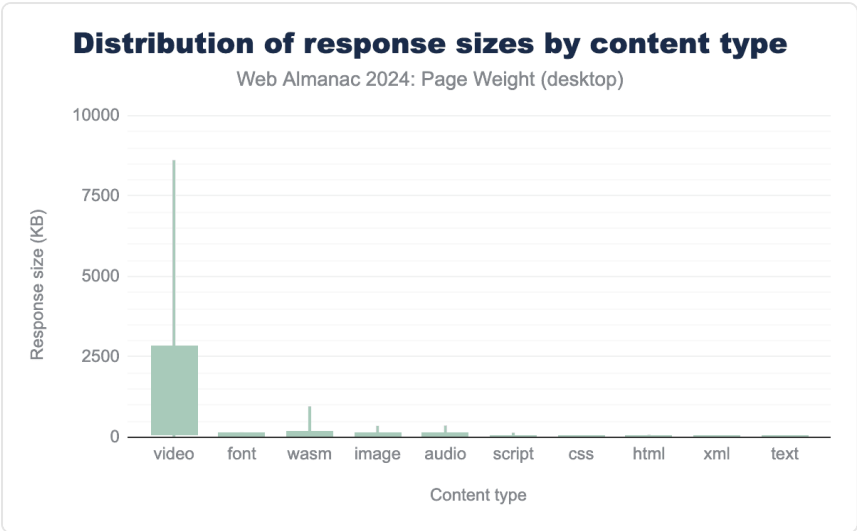


Figure 16.20. Distribution of response sizes by content type.

Images might be the biggest contributors to page weight in all in all, however when looking at the size per request, that switches to Video leading the way, followed by fonts, and WebAssembly (wasm)⁶³¹ files (which weren’t detected in 2022). In 2022’s analysis, audio was the second most weighty, but it slipped to fourth place, behind images this year.

JavaScript Bytes

Increased weight of JavaScript files carries an additional penalty to performance, as not only is the pure size a consideration, a browser needs to parse and execute the JavaScript, which can be a costly process, especially on lower-end devices.

631. <https://web.dev/explore/webassembly>

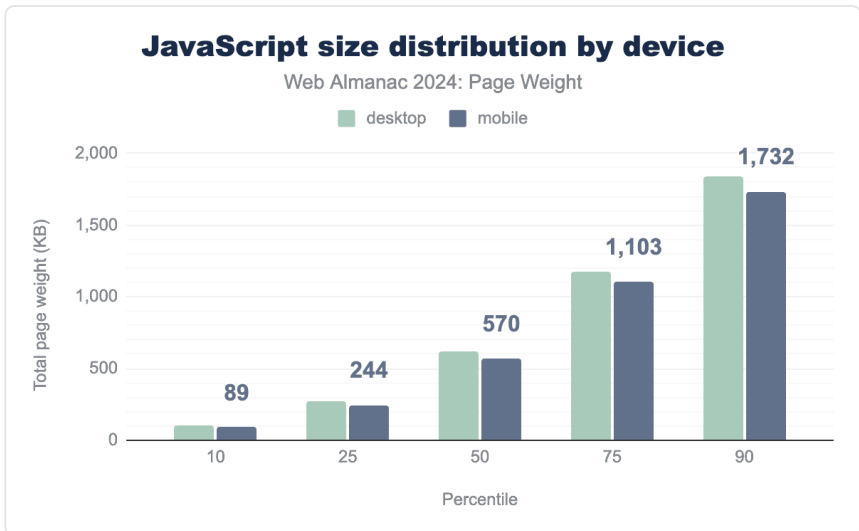


Figure 16.21. Distribution of JavaScript response sizes by device type.

Although the trend is that a desktop page requests more bytes of JavaScript than a mobile page, with the median desktop page requesting 620 KB of JavaScript, and a mobile one 570 KB, the differences aren't huge.

According to Alex Russell's The Performance Inequality Gap, 2024⁶³² study, these are however far above the proposed target of a page load of under 3 seconds at the 75th percentile, which is 365 KB.

At the 75th percentile, both mobile and desktop blast past the proposed 650 KB budget to achieve a 5 second load time, and that is assuming it's a JavaScript-heavy page, and markup is accordingly smaller.

632. <https://infrequently.org/2024/01/performance-inequality-gap-2024/>

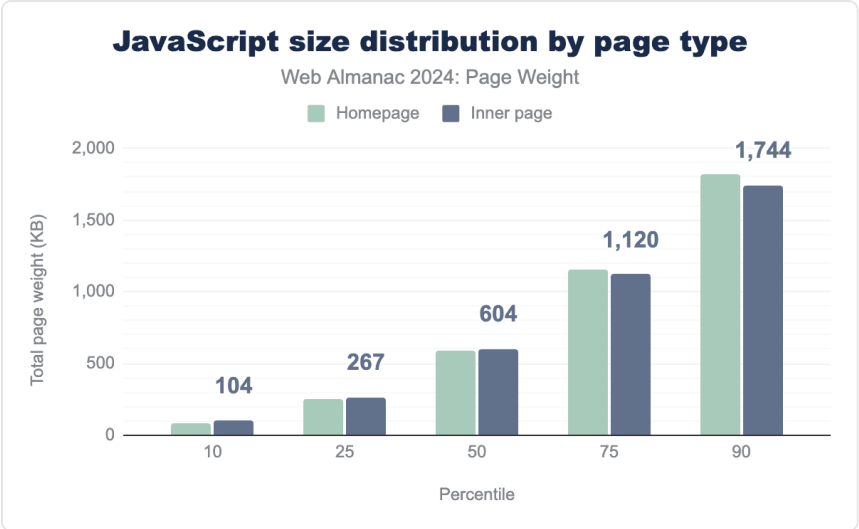


Figure 16.22. Distribution of JavaScript response sizes by page type.

There is not a huge difference between homepage and inner page JavaScript file response sizes. Inner pages have a little more JavaScript up to the 50th percentile, above that the trend is for homepages to have more.

This could point to there being opportunities for developers loading all, or most, JavaScript resources on all pages and represent an opportunity to reduce JavaScript needed overall by tree shaking⁶³³, which is a method of splitting JavaScript files up into more specific ones and only loading them when needed, therefore reducing the wasted JavaScript bytes being downloaded.

633. https://wikipedia.org/wiki/Tree_shaking

CSS bytes

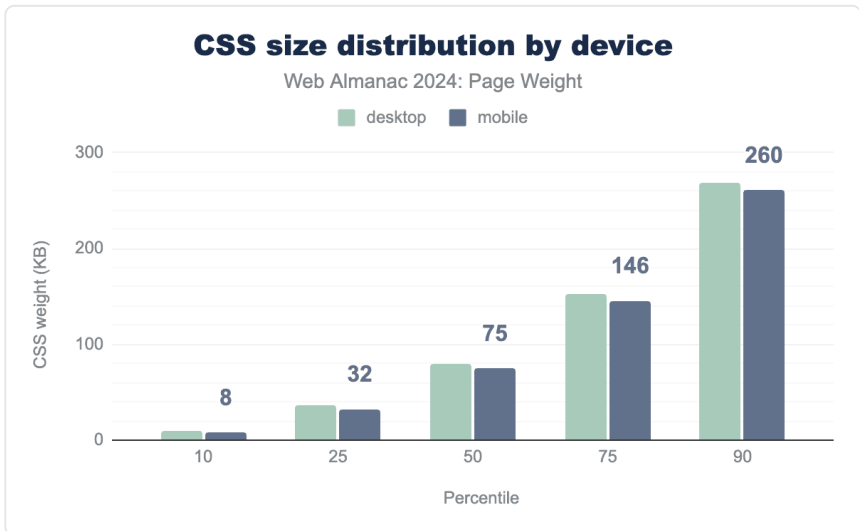


Figure 16.23. Distribution of CSS response sizes by device type.

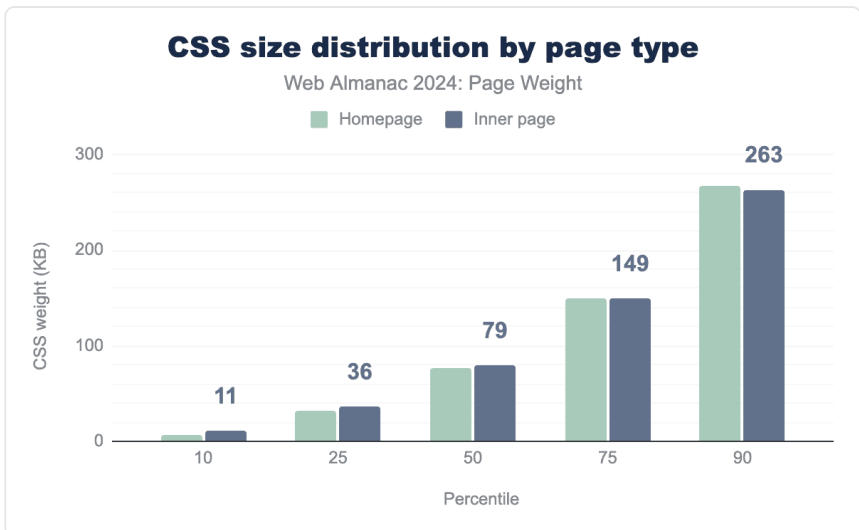


Figure 16.24. Distribution of CSS response sizes by page type.

CSS sizes were slightly larger on desktop than on mobile across all percentiles, and there was very little difference between homepages and inner pages.

This points to the generally adopted method is one set of CSS for all devices and page types. This does potentially point to a missed opportunity to reduce CSS needed overall by tree shaking, as with JavaScript above, but it's always a more nuanced thing with CSS where you are balancing caching and the capabilities of build tools.

Overall, the 76KB of CSS files seems both a little larger than you would hope, but not excessively huge, but keep in mind the best size for a CSS file is as small as it can possibly be. Hopefully that doesn't mean folks are just stuffing it all inline in the head instead.

Image bytes

In past page weight chapters, images have always been the largest contributor to page weight overall, and even though 2024's new inner page data shows that's a trend specific to homepages, it still represents a large component overall.

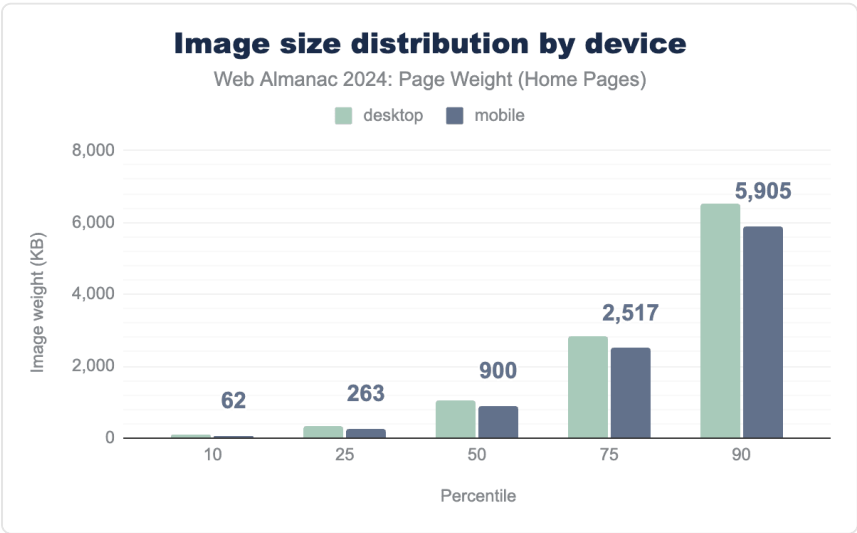


Figure 16.25. Distribution of image response sizes by device type

The median desktop homepage is loading 1,054 KB of images, and mobile ones a little less weighty 900 KB. As noted earlier that's still an increase over 2022 where it was 1,026 KB for desktop pages, 900 KB for mobile. Things soon balloon once you get to the 75th percentile, with 2,822 KB of images for desktop, and 2,517 KB of images for mobile.

In fact, at the median and above, all percentiles were bigger than in 2022 chapter⁶³⁴, however

634. <https://almanac.httparchive.org/en/2022/page-weight#image-bytes>

the more positive findings are that at the 10th and 25th, image bytes were either pretty much stable or down from the previous chapter, pointing to the fact that developers who were already optimising for image file sizes have continued to do so, and might be getting slightly better at it.

It is also pleasing to see that where developers seem to be concentrating on reducing the impact on page weight the most is for mobile users, where page weight can carry the highest penalties. This could be due to folks using responsive image⁶³⁵ serving.

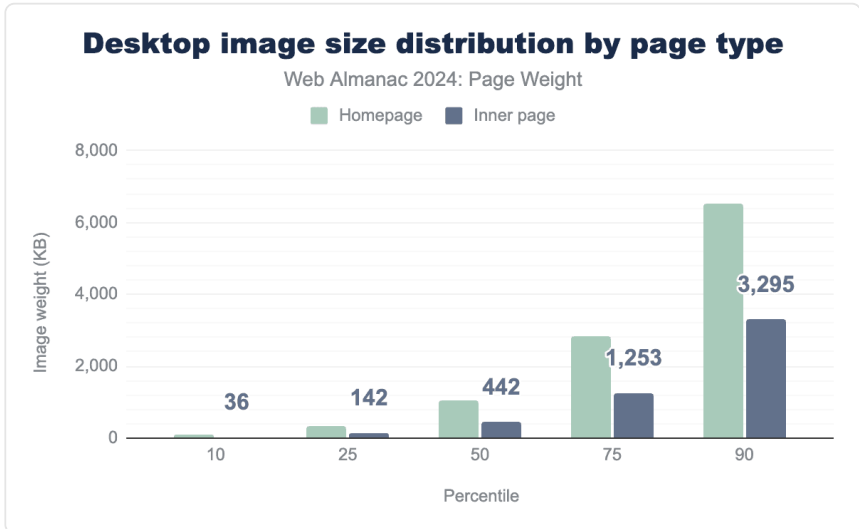


Figure 16.26. Distribution of desktop image response sizes by page type

635. <https://web.dev/articles/serve-responsive-images>

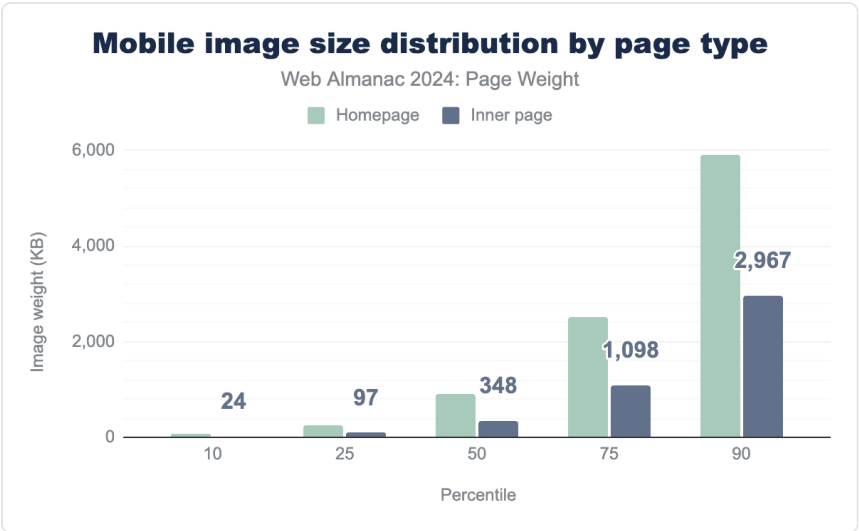


Figure 16.27. Distribution of mobile image response sizes by page type.

Looking at home and inner pages, there is a clear trend for both desktop and mobile for the homepage to have more image bytes, with the median desktop page having 1,054 KB for homepages and 442 KB for inner pages, and on mobile it is 900 KB for homepages and 348 KB. For both desktop and mobile the median inner page carries less than half the image bytes than the homepage.

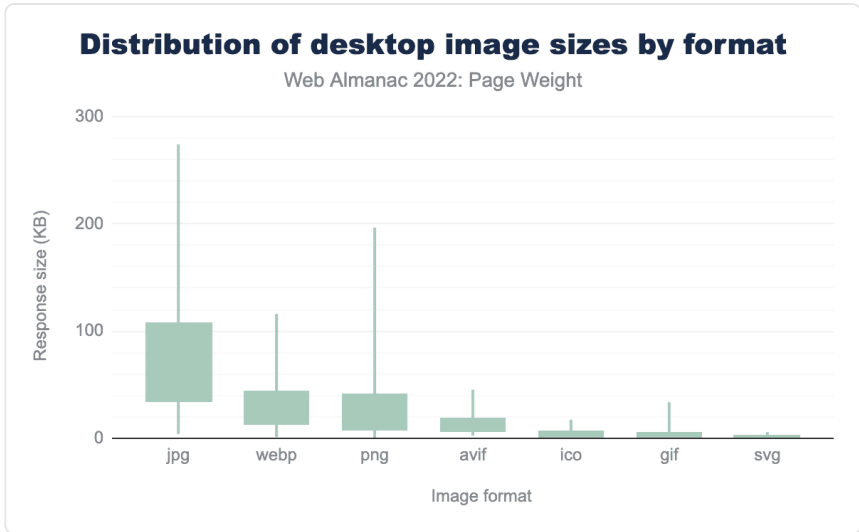


Figure 16.28. Distribution of desktop image sizes by format.

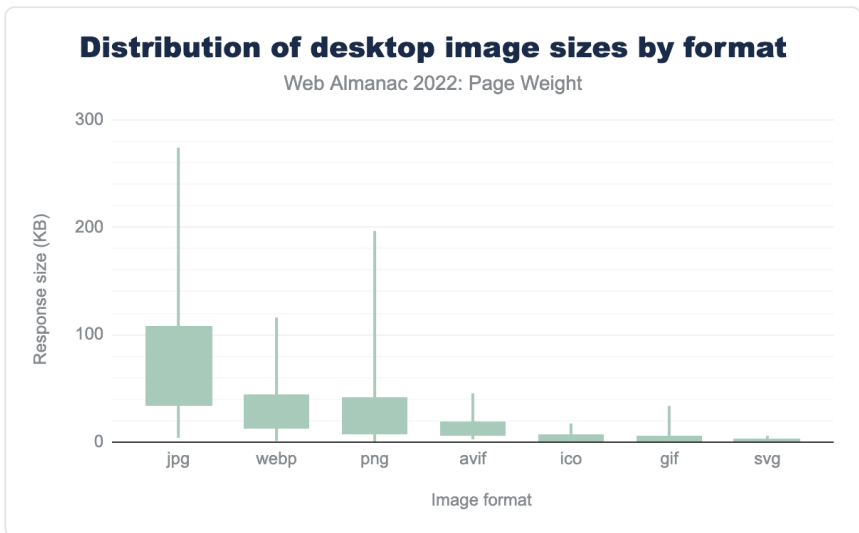


Figure 16.29. Distribution of mobile image sizes by format.

JPG, WebP, and PNG file formats retain their 2022 status as top sources of image weight, for more insights into image format use on the web, visit the Media chapter.

Video bytes

Videos carry a lot of data, for each second of video, there's many images, or frames, and often audio as well. As such, they can significantly add to the weight of a page.

Modern formats help compress and shrink this down, but at some point there is a trade off to be made with file size and quality.

Getting that trade off right and combining with other techniques, like using a facade, can reduce the impact as much as possible.

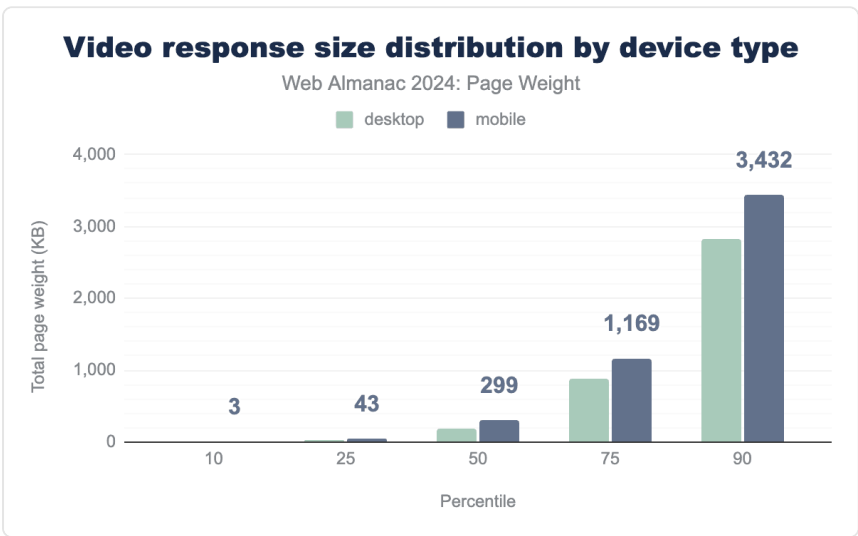


Figure 16.30. Distribution of video response sizes by device type.

The median page requests 194 KB of video for desktop users, and surprisingly a larger 299 KB for mobile users. In fact that trend was present from the 25th percentile onwards, which is a disappointing trend, given that mobile devices are likely to be the ones to benefit the most from reduced page weight.

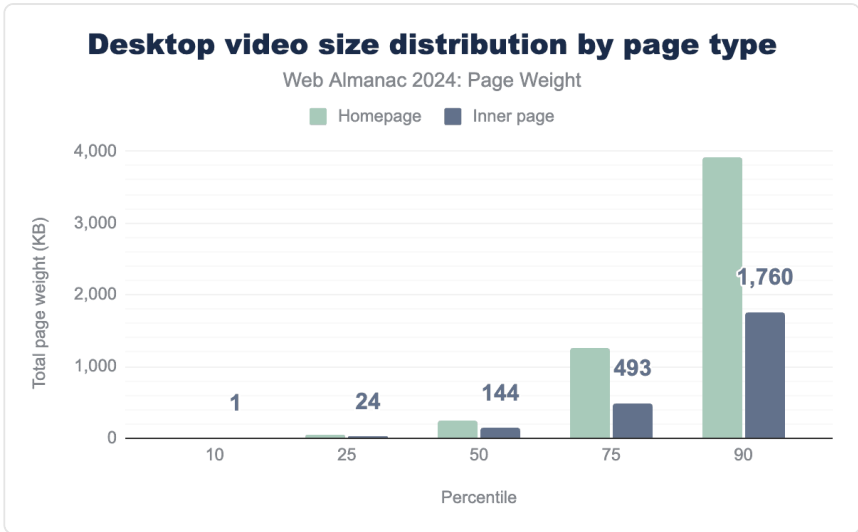


Figure 16.31. Desktop video response size distribution by page type.

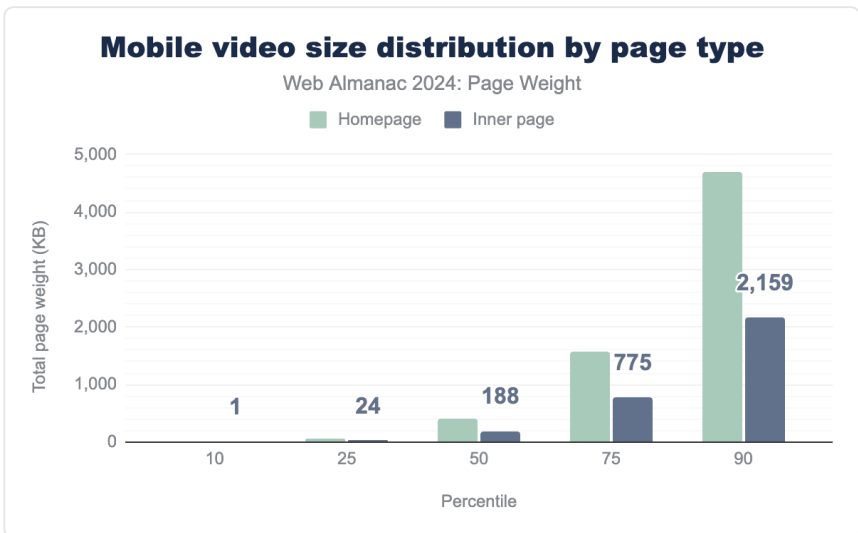


Figure 16.32. Mobile video response size distribution by page type.

Homepages are found to have a larger payload of video resources than inner pages, with the median homepage requesting 243 KB of video resources for desktops and 410 KB for mobile devices. Inner pages were lower at 144 KB for desktop and 188 KB for mobile devices.

This could point to developers favour using video “hero” sections, or otherwise embedding

video on homepages, which tend to focus on the site or business as a whole, rather than adding them into inner pages, which are likely to be more focused pages, perhaps categories and listing pages, or individual articles or products.

Adoption rates of byte-saving technologies

There are a number of things that can be done to reduce page weight. The first one is not so much a technology as an approach, and is on the surface quite simple. Don't send stuff you don't need to.

This means that you should be mindful of what is added to pages, and what is shipped, in short you should be looking to see if what you are adding is really adding value to the user of the page and the business case the page might need to fulfill.

But the aim isn't to remove all features, rather, make sure that what you are adding adds value.

Let's take a look at some of these patterns and approaches to help you deliver your valuable content in more efficient ways, and how often they are being implemented.

Facades for videos & other embeds

Third-party embeds, such as videos, social media posts and other interactive embeds can massively increase page weight. It's simple to click that share button on a video or post and paste the code into your pages without fully being aware of the huge payloads that can come along with it.

Videos you might expect to have a significant number of bytes, but even things like embedding a live chat widget, or even a social media post like a tweet can come with significant overhead, loading a surprising amount of JavaScript to enable interactivity, like clicking the like button or resharing it.

One design pattern that can be a good compromise is using a facade, also known as import on interaction⁶³⁶. The fundamental principle of this is to use a graphical, or simple, non-interactive representation of the embed, which then becomes the interactive, full embed when and if a user clicks on it.

For video, that's often displaying the poster image, which when clicked loads in the full embed. For a social media post, it could be either styled html, or like the video solution, an image that loads the full interactivity when the user clicks on the post.

636. <https://www.patterns.dev/vanilla/import-on-interaction/>

Whilst ultimately, if a user interacts, the larger payload does still need to be loaded, the savings come when many users don't want to interact, watch the video or start a live chat with customer services or sales. The users that do pay that cost are the ones that proactively want to use the feature.

There can be some drawbacks to using facades, these are covered well in the third-party facades article on [web.dev](https://web.dev/637)⁶³⁷. But ultimately, this approach can help save a significant amount of overall page weight.

To look at adoption of facades, we can turn to Lighthouse, which offers a lazy load third-party resources with facades⁶³⁸ audit to see if there are some identifiable resources embedded in the page that might represent an opportunity to use a facade.

Judging adoption is, overall, hard, as we can't reliably test for sites that are implementing facades, because the solution involves the page no longer loading the resources we'd be looking for, so looking at sites that could potentially benefit is more meaningful.

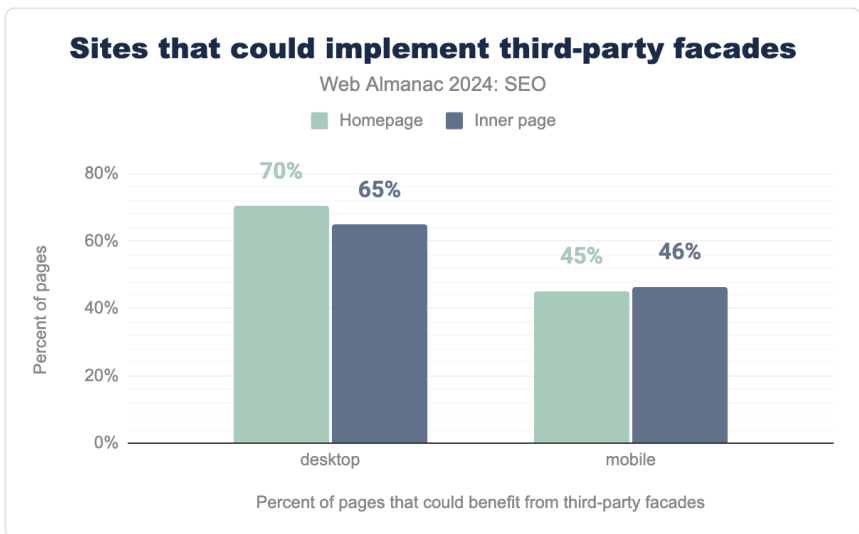


Figure 16.33. Sites that could implement Third-party facades.

70% of desktop homepages were detected as having the opportunity to replace some embeds with facades, for inner pages on desktop it was 65%. For mobile crawls, things were better here, with 45% of mobile homepages potentially being able to benefit, and 46% of mobile inner pages.

637. https://developer.chrome.com/docs/lighthouse/performance/third-party-facades#live_chat_intercom_drift_help_scout_facebook_messenger

638. <https://developer.chrome.com/docs/lighthouse/performance/third-party-facades>

From the data, it would appear that either adoption is higher on pages being served to mobile clients, or developers and publishers are omitting the type of third-party embeds for mobile users altogether.

Although facades were covered in the previous almanac page weight chapter in 2022, the test has changed slightly over the last two years, along with the methodology we used to analyze the data, so direct comparisons are not possible.

Compression

Compression can allow you to shrink the size of your resources before you send them across the network to the requesting client, where they are uncompressed by the client, which for web pages is usually a browser, before being used. Smaller payloads in theory, and usually in practice, make for faster page loads.

For text based files, like HTML, CSS, JavaScript, JSON, SVG, ico and ttf font files, HTTP compression is a powerful ally in reducing page weight as transmitted. Using GZIP or Brotli compression can sometimes significantly reduce the size of text-based resources. Other file types, especially media files like images and videos, do not benefit from HTTP compression, as they are already compressed.

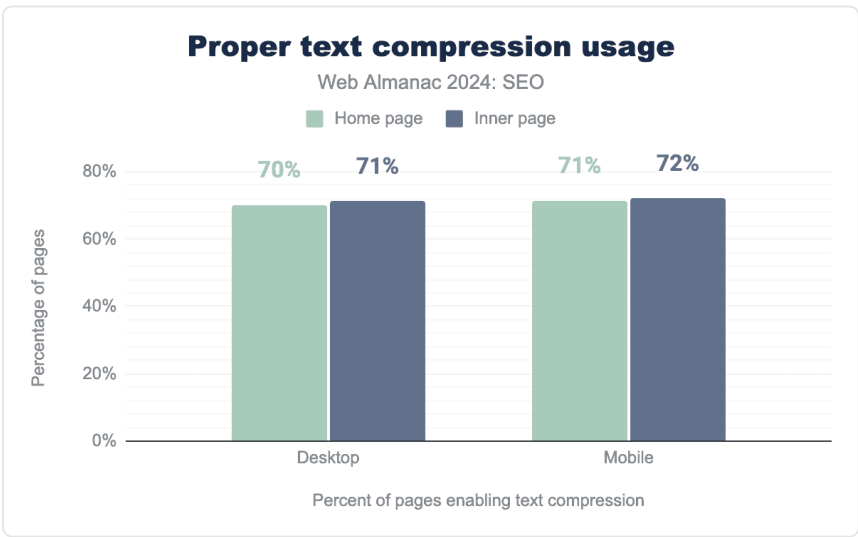


Figure 16.34. Proper text compression usage.

We detected that 70% percent of desktop homepages and a fundamentally similar 71% of inner pages correctly used text compression. The homepage figure represents a drop from 74% in

2022.

For mobile crawls, 71% of homepages and 72% of inner pages correctly used text compression. Comparing homepages to 2022, this also represents a drop from 73%.

The drop in usage, whilst perhaps small, is disappointing, it's certainly not become harder to enable compression, and perhaps easier, especially if you use a CDN like Cloudflare where it's a simple flip of a switch in a dashboard.

It should be noted that compression is not entirely magic, and doesn't make the whole impact of page weight go away for these resources for the client. Ultimately they do need to be decompressed again before use.

It's also not an entirely free process either, it takes work on the server to compress these resources, somewhat mitigateable by caching the compressed resources where possible, and it does take work on the client to decompress them too.

But as a tradeoff, it's normally one worth making, compression techniques are generally well optimised and efficient, and the major bottleneck is the network.

Minification

Minification can reduce the overall size of resources by removing unnecessary characters⁶³⁹, like spaces, returns and code comments, things that aren't needed by a browser to use the resources.

Unlike compression, there's no additional work to be done client side, resources do not need to be unminified. There can be some overhead and work done on the server, if resources are minified on-the-fly, but very often it's best to minify CSS resources up front, they are very likely to be static in nature, and can often be done at build time.

639. <https://developer.mozilla.org/Glossary/Minification>

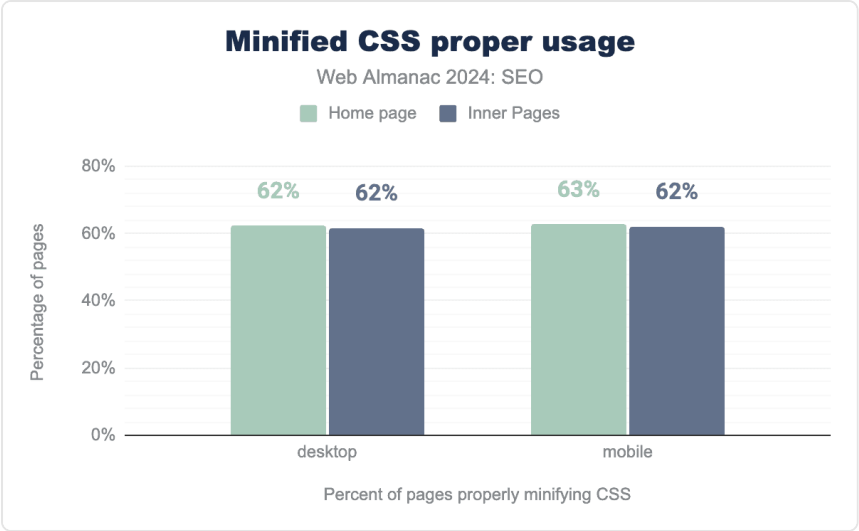


Figure 16.35. Minified CSS proper usage.

In 2024, 62% of homepages had correctly minified CSS, as detected by lighthouse, a significant drop from 2022 where the figure was 84%. For mobile homepages it was 63%, a drop from 2022’s 68%, and inner pages were slightly less at 62%.

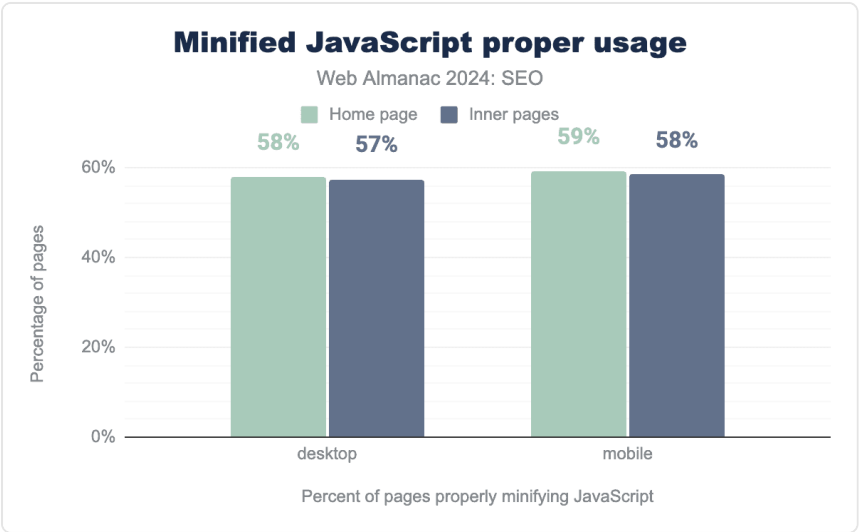


Figure 16.36. Minified JavaScript proper usage.

In 2024, 58% of desktop homepages had correctly minified JavaScript, as detected by Lighthouse, which is a significant drop from 2022 where it was 77%. Inner pages were slightly worse at 57%. For mobile, 59% of homepages passed this test, down from 2022's 64%. Like desktop, inner pages were slightly worse at 58%.

Whilst it is still encouraging that more sites pass this test than don't, it is very disappointing to see that both CSS and Javascript minification is less prevalent in 2024 than in 2022.

Caching and CDNs

The use of caching and CDNs plays a role in managing page weight, as they help reducing page load times by minimising the time required to deliver resources to users. However, it is worth mentioning that CDNs do not reduce page weight.

Resources include both static and dynamic, as well as personalisation, third-party integrations and Edge computing.

On the one hand, caches which are used both on servers and on browsers, allow resources to be reused. Cached content is sent via CDNs, a series of interconnected servers geographically distributed, which reduce the distance between the server requesting the resource and the resource being served. This is particularly important for international websites.

For more insights into CDNs, please refer to the CDN chapter.

Page weight and Core Web Vitals

Core Web Vitals⁶⁴⁰ are a set of performance metrics designed to refine the dangerously ambiguous definition of “performance” into a human-centric measurement. To be a “Good” page, a page must pass three evaluations that measure key moments for users:

1. Is it loading? (Largest Content Paint (LCP))⁶⁴¹
2. Can the user interact? (Interaction to Next Paint (INP))⁶⁴²
3. Is it visually stable? (Cumulative Layout Shift (CLS))⁶⁴³

Core Web Vitals are designed to be an evolving set of metrics. This year, the metric for interactivity changed⁶⁴⁴ from First Input Delay (FID) to Interaction to Next Paint (INP). This change was made because it provided two significant advancements:

640. <https://web.dev/articles/vitals>

641. <https://web.dev/articles/lcp>

642. <https://web.dev/articles/inp>

643. <https://web.dev/articles/cls>

644. <https://developers.google.com/search/blog/2023/05/introducing-inp>

- The first is to shift from a single interaction to include all interactions on the page. In other words, clicking with a mouse, tapping on a device with a touchscreen and pressing a key on either a physical or onscreen keyboard.
- The second is to represent interactivity for sites using JavaScript frameworks accurately, since JavaScript is often what drives interactivity mostly.

When responsiveness as measured by INP is below 200 milliseconds, it is considered to be a good experience and pass the INP performance assessment. Total Blocking Time remains as the lab data equivalent and is for diagnostics when an INP issue is detected.

The changeover in March 2023 saw many JavaScript framework origins drop from passing Good classification. Sites using prominent frameworks like React, Next, Nuxt, and Vue were hit hardest. Sites adapted quickly and by September 2024, the number of passing origins exceeded when evaluated by INP exceeded the count when the FID metric was used.

To gather data, we had to rely on lighthouse’s lab testing audits, which capture LCP and CLS, but not the interaction based metrics of INP or FID. Lab testing does have drawbacks, and real user metrics should always be used to truly assess performance, as detailed in web.dev’s Why lab and field data can be different (and what to do about it)⁶⁴⁵.

We used data from June 2024, the page weights for each percentile and device type are as follows:

Percentile	Desktop Page Weight	Mobile Page Weight
10th	549 KB	471 KB
25th	1,138 KB	995 KB
50th	2,157 KB	1,938 KB
75th	4,169 KB	3,766 KB
90th	8,375 KB	7,680 KB

Figure 16.37. Percentile page weight by device from Lighthouse tests.

Largest Contentful Paint (LCP)

A good score for Largest Contentful Paint⁶⁴⁶ is 2.5 seconds or less. LCP over 4 seconds is considered poor.

645. <https://web.dev/articles/lab-and-field-data-differences>

646. <https://web.dev/articles/lcp#what-is-a-good-lcp-score>

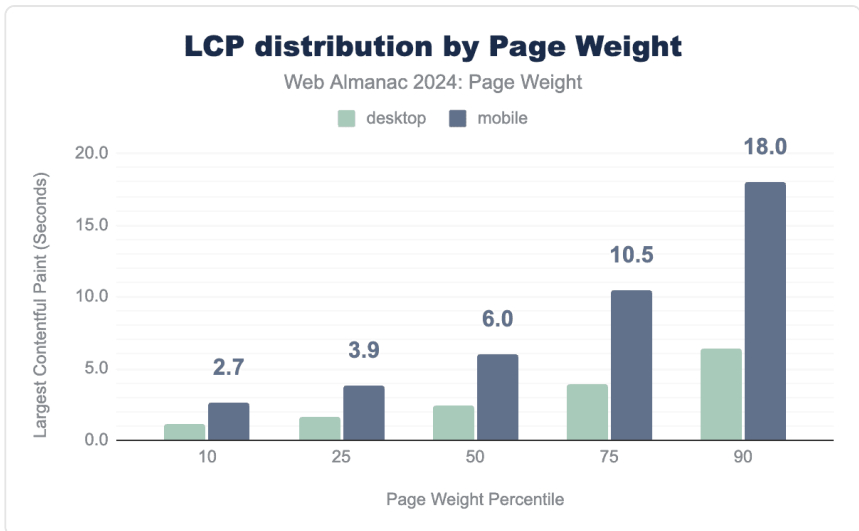


Figure 16.38. Distribution of LCP scores by device type and page weight

There is a clear correlation between the page weight and Largest Contentful Paint, the higher the page weight, the longer the time to LCP. This is especially true for mobile devices with a much steeper curve.

Cumulative Layout Shift (CLS)

A good score for Cumulative Layout Shift⁶⁴⁷ is 0.1 or less. CLS over 0.25 is considered poor. On consideration to keep in mind when looking at this particular metric is it is especially affected by differences in lab and field data, as CLS is effectively measured across the whole life of a page, including interactions and scrolling, where lab tests can only capture the initial load.

647. <https://web.dev/articles/cls#what-is-a-good-cls-score>

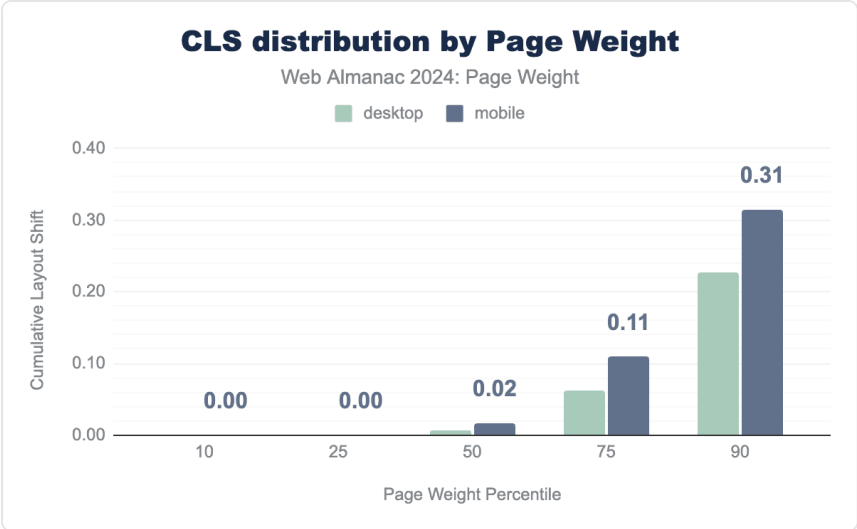


Figure 16.39. Distribution of CLS scores by device type and page weight.

Like LCP, CLS also grew as page weight grew, although the differences between desktop and mobile were more linear. CLS isn't directly a load time issue, so this could point to poorly optimised, ads heavy pages injecting ads into content, pushing content down, or images with large files sizes defined without dimension.

Total Blocking Time

As mentioned above, Interaction to Next Paint, or even the older First Input Delay cannot be accurately measured in lab tests, however, as recommended by web.dev⁶⁴⁸, Total Blocking Time⁶⁴⁹, often shortened to TBT, can be a good proxy metric to see how interactivity might be affected.

A total blocking time of 200 ms or less is considered a good target.

648. <https://web.dev/articles/inp#lab-measurement>
649. <https://web.dev/articles/tbt>

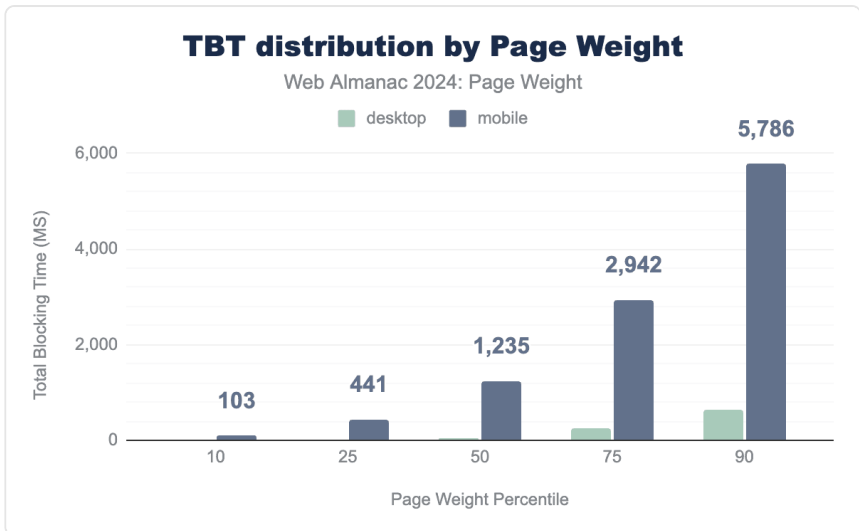


Figure 16.40. Distribution of total blocking time scores by device type.

The trend of larger page sizes negatively impacting performance, as measured by total blocking time, is evident across both mobile and desktop experiences. However, the disparity is particularly significant for mobile users. At the 50th percentile, Time to First Byte (TBT) is 53ms for desktop and 1,235ms for mobile—a difference of 1.2 seconds. This discrepancy becomes even more pronounced at the 90th percentile, with TBT at 648ms for desktop and a staggering 5,786ms for mobile—a massive 5.1 second difference.

It is reasonable to conclude that interactivity is one of the worst hit metrics associated with excessive page weight.

For more data on Core Web Vitals in 2024, visit the Performance chapter.

Conclusion

The growing issue of page weight reflects the need for balance between functionality and accessibility.

While advancements like JavaScript frameworks and rich media have enhanced the web's interactivity and storytelling, they have also introduced significant challenges. Bloated pages disproportionately impact users with limited devices or slow connections, making web experiences less inclusive and equitable.

Sadly this year's data shows the same trend of page weight continuing to increase, for both mobile and desktop users.

By focusing on lightweight, efficient design and adopting practices like compression, caching, and byte-saving technologies, developers can bridge the “performance inequality gap” and ensure the web remains accessible for all.

Ultimately, this approach not only benefits users but also supports sustainability and long-term growth of the web.

Overall, in 2024 page weight is far from a “solved problem”, but there remains a lot of opportunity for developers to embrace byte saving techniques and technologies, with plenty of room left for adoption for things like facades and compression, leading to a lighter, brighter web for us all.

Authors



Dave Smart

🐦 @tamethebots.com 📧 dwsmart 🔗 davewsmart 🌐 https://tamethebots.com

Dave Smart is a developer and technical search engine consultant at Tame the Bots⁶⁵⁰. They love building tools and experimenting with the modern web, and can often be found at the front in a gig or two.



Jamie Indigo

🐦 @not-a-robot.com 📧 fellowhuman1101 🔗 jamie-indigo 🌐 https://not-a-robot.com

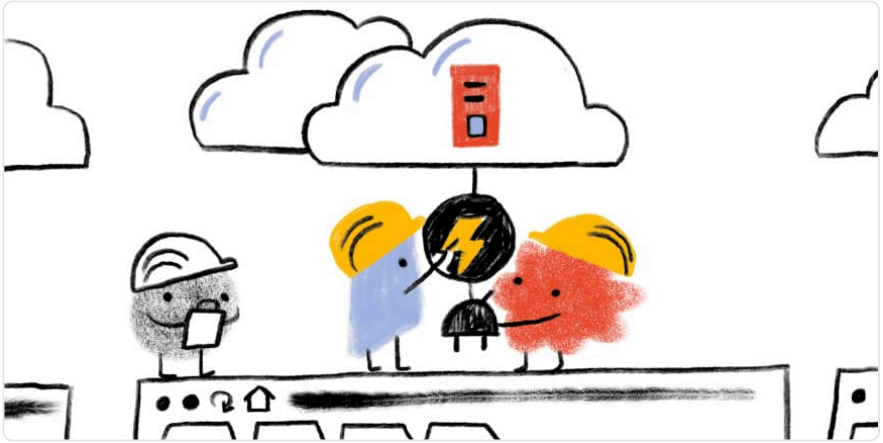
Jamie Indigo isn't a robot, but speaks bot. As director of technical SEO at Cox Automotive⁶⁵¹, they study how search engines crawl, render, and index the web. Jamie loves to tame wild JavaScript and optimize rendering strategies. When not working, they like horror movies, graphic novels, and terrorizing lawful good paladins in Dungeons & Dragons.

650. <https://tamethebots.com>

651. <https://www.coxautoinc.com/>

Part IV Chapter 17

CDN



Written by Joe Viggiano, Jaiganesh Girinathan, and Alex Moening

Reviewed by Caroline Scholles

Analyzed by Jaiganesh Girinathan and Alex Moening

Edited by Caroline Scholles

Introduction

This chapter examines the evolving landscape of Content Delivery Networks (CDNs) and their critical role in today's digital ecosystem. As we move from 2022, when our last CDN chapter came together, and into 2024, CDNs continue to be fundamental in delivering content globally, extending their reach beyond large-scale operations to smaller sites and applications. Their significance has grown in facilitating the delivery of not just static content, but also dynamic and personalized experiences, third-party integrations, and Edge Computing.

A key focus of this chapter is the pivotal role CDNs play in driving the adoption of web standards, protocols, and emerging technologies like HTTP/3 and the Quick UDP Internet Connections (QUIC) protocol. We also explore how CDNs are at the forefront of implementing and popularizing performance optimization techniques.

We believe that in 2024, CDNs continue to not only facilitate performant content delivery, but also serve as comprehensive platforms that integrate first and third-party security solutions.

What is a CDN?

A *Content Delivery Network* (CDN) is a geographically distributed network of servers designed to provide high availability, enhanced performance, and improved security for web content and applications. The primary goal of a CDN is to minimize latency and optimize content delivery by serving data from locations closer to the end user.

The role of CDNs has expanded significantly in recent years, driven by the increasing complexity of web applications, the growth of streaming services, and the rising demands of e-commerce and digital businesses. In 2024, CDNs are crucial infrastructure supporting a wide range of online activities and the increasing sophistication of web applications.

CDNs have evolved far beyond their original function as simple proxy servers. Today's CDN offerings typically include:

- Caching and content optimization for various types of media
- Intelligent routing and load balancing to minimize network hops and optimize performance
- Edge Computing capabilities, allowing for near-real-time processing and personalization
- Robust security features to protect against a wide range of cyber threats
- Analytics and insights to help businesses understand and optimize their web performance

The benefits of utilizing CDNs have expanded beyond simple performance improvements. In 2024, CDNs play a crucial role in enabling global scalability, enhancing security postures, and facilitating the deployment of complex, distributed applications. By pushing more logic to the edge, businesses can create more responsive and personalized user experiences while reducing the load on origin servers.

Lastly, an often overlooked benefit is how CDNs contribute to sustainability by caching content closer to end users and optimizing the size of files, such as videos and images. This translates to lower energy consumption and a smaller carbon footprint associated with content delivery.

Caveats and disclaimers

As with any observational study, there are limits to the scope and impact that can be measured. The statistics gathered on CDN usage for the Web Almanac are focused more on applicable

technologies in use and not intended to measure performance or effectiveness of a specific CDN vendor. While this ensures that we are not biased towards any CDN vendor, it also means that these are more generalized results.

These are the limits to our testing methodology:

- **Simulated network latency:** We use a dedicated network connection that synthetically shapes traffic.
- **Single geographic location:** Tests are run from a single datacenter and cannot test the geographic distribution of many CDN vendors.
- **Cache effectiveness:** Each CDN uses proprietary technology and many, for security reasons, do not expose cache performance or depth of cache.
- **Localization and internationalization:** Just like geographic distribution, the effects of language and geo-specific domains are also opaque to these tests.
- **CDN detection:** This is primarily done through DNS resolution and HTTP headers. Most CDNs use a DNS CNAME to map a user to an optimal data center. However, some CDNs use Anycast IPs or direct A+AAAA responses from a delegated domain which hide the DNS chain. In other cases, websites use multiple CDNs to balance between vendors, which is hidden from the single-request pass of our crawler.
- **IPv6 detection:** Whether or not a CDN is configured to use IPv6 can be inferred if the DNS entry for the domain name contains a AAAA entry and accepts a connection over IPv6. The 2024 test run did not include this capability, however we've ensured this data can be collected for the 2025 Web Almanac.

All of this influences our measurements. These results reflect the support of specific features (for example TLSv1.3, HTTP/2+, Zstandard) per site, but do not reflect actual traffic usage.

With this in mind, here are a few statistics that were intentionally not measured in the context of a CDN:

- Time To First Byte (TTFB)
- Time To Last Byte (TTLB)
- CDN Round Trip Time
- Core Web Vitals
- "Cache hit" versus "cache miss" performance

While some of these could be measured with HTTP Archive dataset, and others by using the CrUX dataset, the limitations of our methodology and the use of multiple CDNs by some sites, will be difficult to measure and so could be incorrectly attributed. For these reasons, we have decided not to measure these statistics in this chapter.

CDN adoption

A web page is composed of following key components:

1. Base HTML page (for example, `www.example.com/index.html` —often available at a more friendly name like just `www.example.com`).
2. Embedded first-party content such as images, css, fonts and javascript files on the main domain (`www.example.com`) and the subdomains (for example, `images.example.com`, or `assets.example.com`).
3. Third-party content (for example, Google Analytics, advertisements) served from third-party domains.

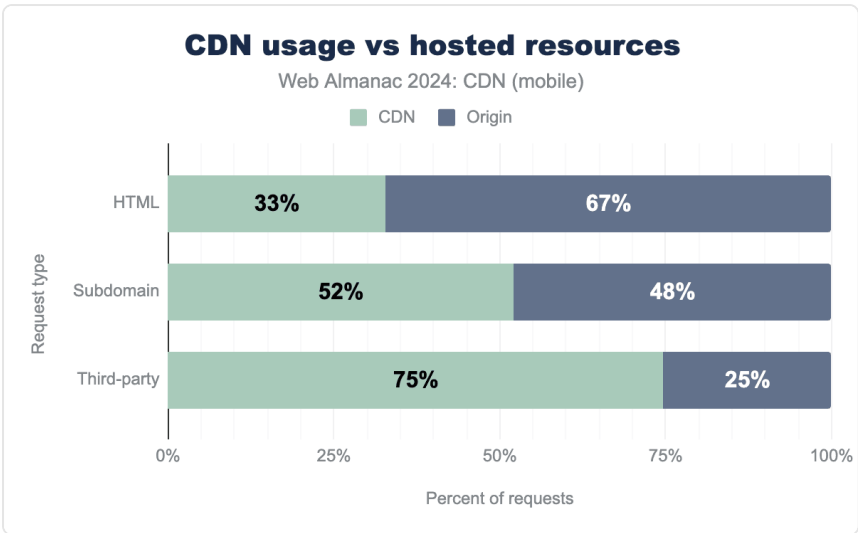


Figure 17.1. CDN usage vs hosted resources on mobile.

The chart above shows the breakdown of requests for different types of content (HTML, Subdomain, and Third-party), showing the share of content served by CDN versus origin on mobile devices (identical figures are observed for desktop).

CDNs are often utilized for delivering static content such as fonts, image files, stylesheet, and

Javascript.. This kind of content doesn't change frequently, making it a good candidate for caching on CDNs proxy servers. We still see CDNs are used more frequently for this type of resource—especially for third-party content, with 75% being served via CDN.

CDNs can provide better performance for delivering non-static content as well as they often optimize the routes and use most efficient transport mechanisms. However, we see that the usage of CDNs for serving HTML still lags considerably behind the other two types of content, with only 33% served via CDN and 77% still being served from the origin.

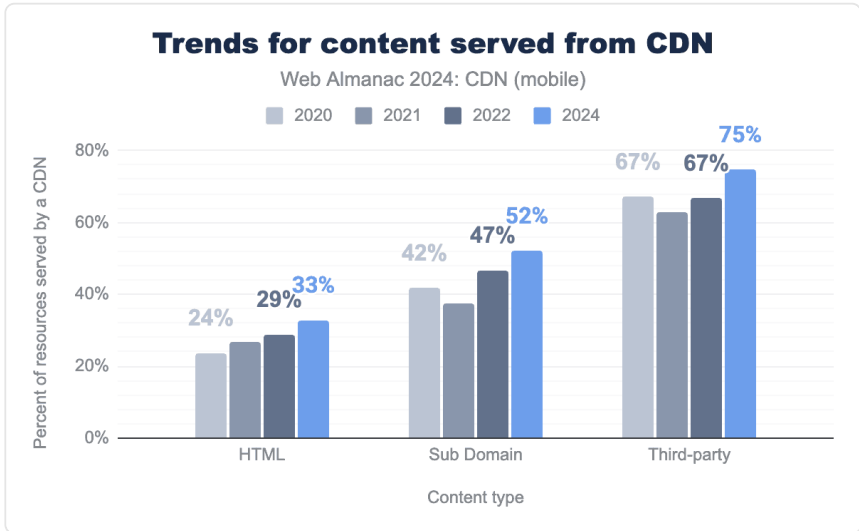


Figure 17.2. Trends for content served from CDN for mobile.

The above figure shows the evolution of different content types served from CDNs over the years.

Whether it be the base HTML pages, sub-domains, or third-parties, 2024 saw increased adoption compared to previous years. The fastest pace was seen in third-party with a 8% increase from 67% in 2022 to 75% in 2024.

These are likely some of the reasons behind this continued trajectory:

- The persistence of remote and hybrid working models continues to drive the need for consistent delivery of content to a geographically dispersed user base.
- Growing security threats led more companies to value CDNs' built-in scalable protections like DDoS mitigation and WAF capabilities.

- Improvements in edge computing enabled more rich personalized experiences while reducing infrastructure compute costs.

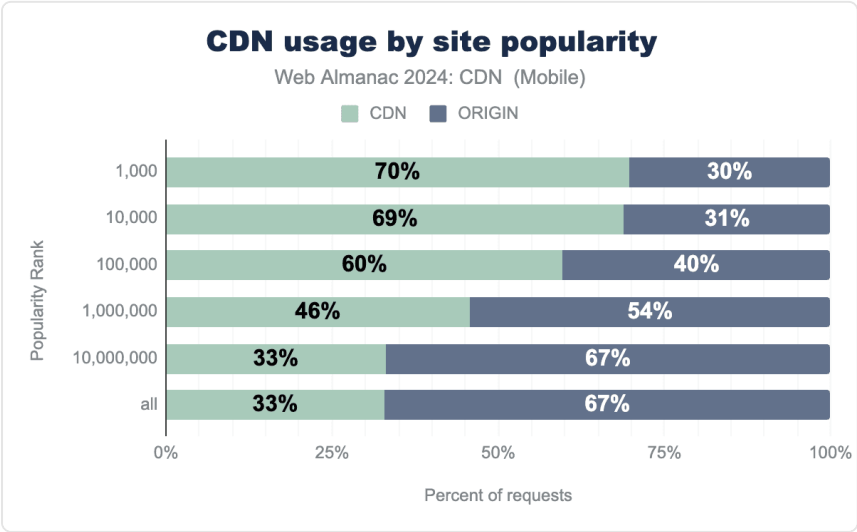


Figure 17.3. CDN usage by site popularity on mobile.

The share of CDN usage has increased over the years, particularly among the most popular websites according to Google Chrome's UX Report (CrUX) classification. As the graph shows, the top 1,000 websites have the highest CDN usage at 70%, followed by the top 10,000 at 69%, and the top 100,000 at 60%. Compared to our latest results, CDN usage among the top 1,000 to 10,000 most popular websites increased by 6%, while CDN usage among the top 100,000 websites rose by 8%.

As mentioned in previous editions, the increase in CDN usage among smaller sites can be attributed to the rise of free and affordable CDN options. Additionally, many hosting solutions now bundle CDNs with their services, making it easier and more cost-effective for websites to leverage this technology.

CDN providers

CDN providers can generally be classified into two segments:

1. **Generic CDNs** – Providers that offer a wide range of content delivery services to suit various use cases, including Akamai, Cloudflare, CloudFront, and Fastly.
2. **Purpose-built CDNs** – Providers tailored to specific platforms or use cases, such as

Netlify and WordPress.

Generic CDNs address broad market needs with offerings that include:

- Website delivery
- Mobile app API delivery
- Video streaming
- Edge Computing services
- Web security offerings

These capabilities appeal to a wide range of industries, which is reflected in the data.

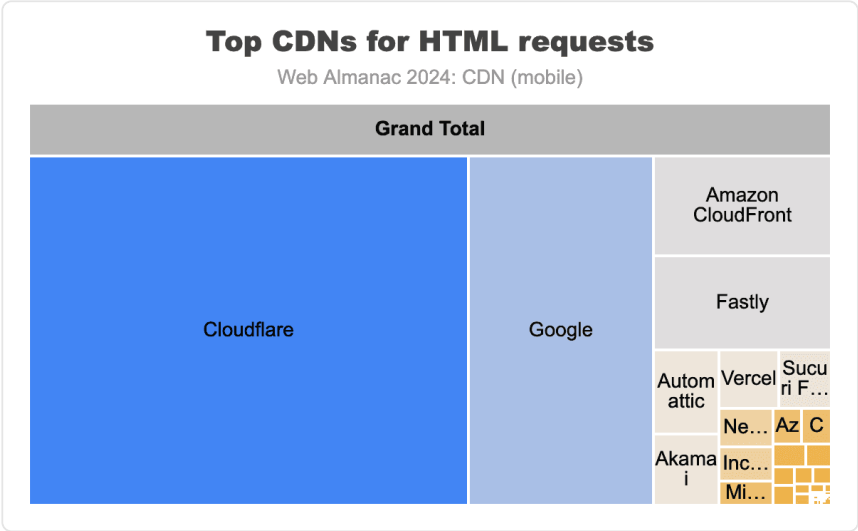


Figure 17.4. Top CDNs for HTML requests on mobile.

The above figure shows the top CDN providers for base HTML requests. The leading vendors in this category are Cloudflare, with a 55% share, followed by Google (23%), Amazon CloudFront (6%), Fastly (6%), Akamai (2%), and Automattic and Vercel, each with a 1% share.

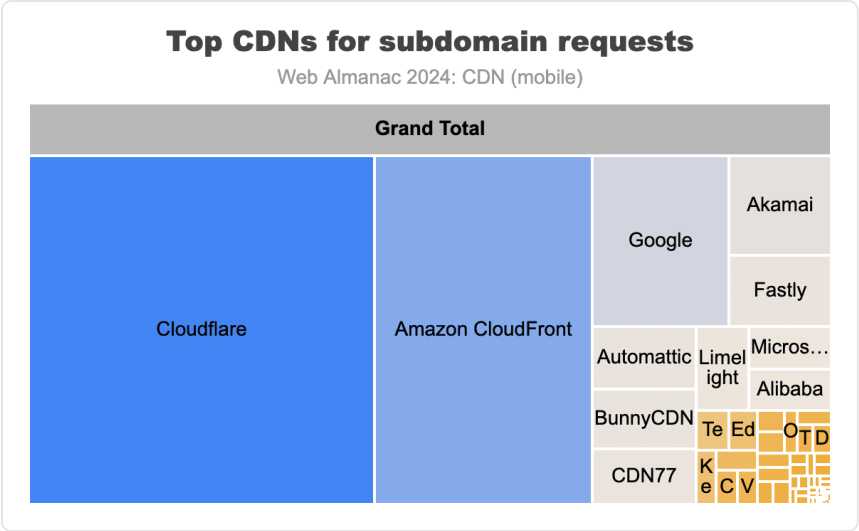


Figure 17.5. Top CDNs for subdomain requests on mobile.

For the subdomain requests, we saw an increase in the share of Amazon Cloudfront (from 19% to 27%).

This is driven by many users having their content on cloud service providers that come with a CDN offering. Utilizing compute and other services alongside the cloud service provider CDN helps users scale their applications and increase performance of delivering services to end users.

The leading vendors in this category are Cloudflare (43%), Amazon CloudFront (27%), Google (8%), and Akamai (3%).

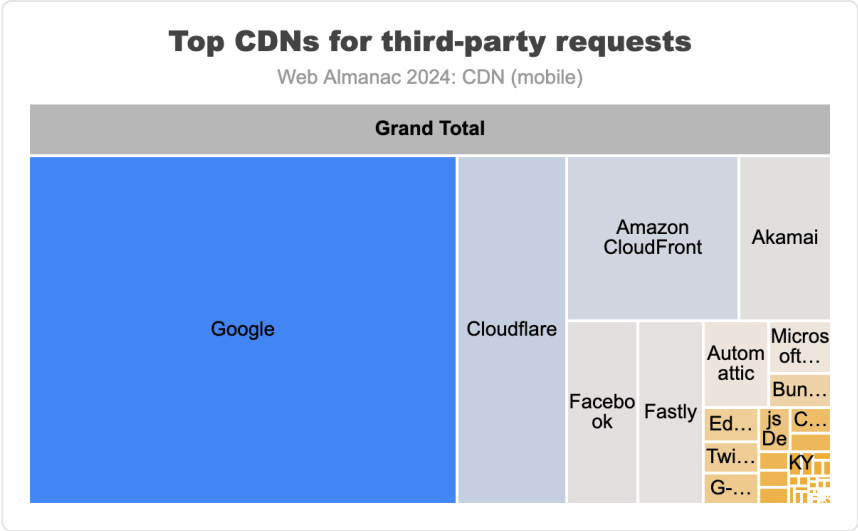


Figure 17.6. Top CDNs for third-party requests on mobile.

The above figure highlights third-party domain usage, with Google leading the list at 54% market share, followed by well-known CDN providers such as Cloudflare (14%), Amazon CloudFront (10%), Akamai (5%), and Fastly (4%). Notably, Facebook also appears prominently in the rankings, holding a 4% share.

While these CDNs have purpose built features that optimize for particular sets of content delivery workflows, many are also attached to larger service offerings either with cloud services, security, and/or edge computing. These services are often delivered or integrated with the CDNs themselves which further drives adoption as part of a broader ecosystem of services.

Third-party providers like Google and Facebook might wholly optimize and purpose build their CDNs to handle high throughput rates for ad delivery and beacon capturing. While others such as Cloudflare or Amazon CloudFront may optimize a subset of features. These more general use CDNs take these features to integrate into managed services such as providing managed API services to a global user base or dynamically injecting javascript to perform client side inspection of devices for web security purposes.

HTTP/3 (HTTP/2+) adoption

Published in June of 2022 by IETF, HTTP/3⁶⁵² is a major revision of the HTTP network protocol,

652. <https://datatracker.ietf.org/doc/html/rfc9114>

succeeding HTTP/2.

The most notable difference in HTTP/3 is that it uses a protocol called QUIC over UDP instead of the traditional TCP. This change improves performance by reducing latency, allowing faster data transmission, especially in environments with high packet loss or network congestion. TLS v1.3 was an improvement in reducing the number of TCP + TLS network protocol handshakes and round trips from the client to server, but QUIC reduces this further without sacrificing security. Another key improvement is the elimination of head-of-line blocking, meaning if one resource experiences delivery issues, other resources can still load independently. With this enhanced multiplexing and robust encryption, QUIC contributes to a more secure and efficient browsing experience.

For website operators, CDNs handle all the complex implementation details while providing automatic fallback to HTTP/2 when needed. This experience enabled by CDNs is a simple configuration change without requiring significant technical investment on the operator's part.

Due to the way HTTP/3 works (see the HTTP chapter for more information), HTTP/3 is often not used for first connections which is why we are instead measuring "HTTP/2+", since many of those HTTP/2 connections may actually be HTTP/3 for repeat visitors (we have assumed that no servers implement HTTP/3 without HTTP/2).

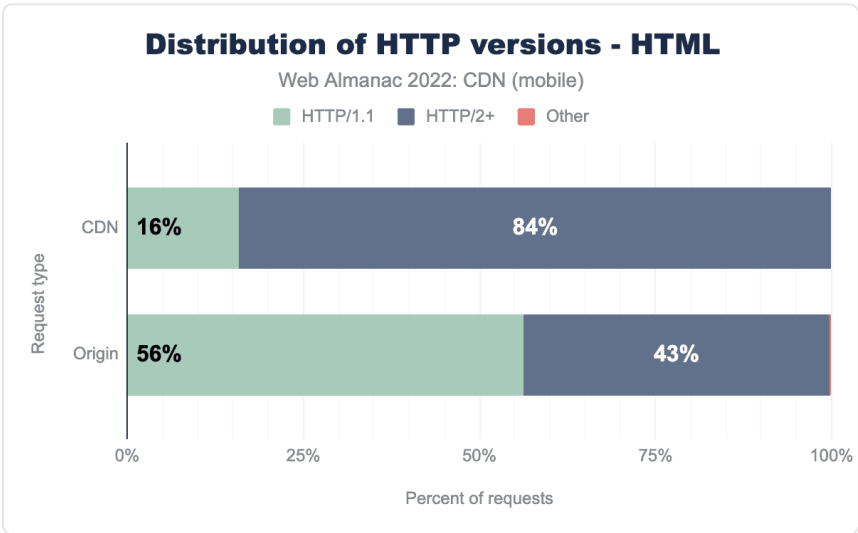


Figure 17.7. Distribution of HTTP versions for HTML (mobile).

In 2022, we observed a stark contrast of HTTP/2+ usage between CDN and origin servers. While a gap persists whereby CDN usage of HTTP/2+ is higher 98% compared to 71%, the origin usage of HTTP/2+ continues to grow in adoption which we can see with 42% in 2022 to

71% in 2024.

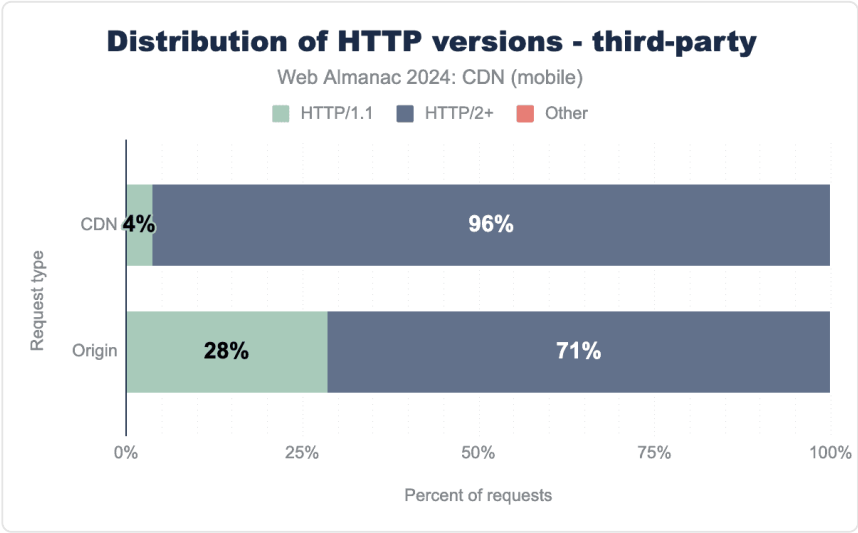


Figure 17.8. Distribution of HTTP versions for third-party requests (mobile).

The trend for HTTP/2+ adoption in third-party domains follows the same as for HTML above. Compared to 2022, HTTP/2+ CDN usage increased from 88% to 96% while origin increased from 47% to 71%.

Overall compared to previous years, while CDNs continue to lead in the adoption of newer HTTP versions, in 2024 origin servers are beginning to catch up. This could be a result of the critical mass of CDN traffic already using HTTP/2+, but we look forward in future chapters to diving deeper into these trends as they unfold.

Compression

Compression remains a fundamental aspect of web content delivery, playing a crucial role in optimizing user experience and website performance. By reducing the size of files transmitted over networks, compression contributes to faster page load times, decreased bandwidth consumption, and improved overall web browsing efficiency. Despite advancements in network speeds and the proliferation of diverse connectivity options, compression continues to be a key factor in enhancing internet experiences across all types of connections.

Within the web ecosystem we observed several commonly used compression algorithms:

- Gzip
- Brotli
- Zstandard (Zstd)

While media files such as JPEG images are already compressed, textual assets such as HTML, stylesheets, javascript, and manifest files can be compressed to optimize performance. Created in 1992, Gzip⁶⁵³ is the longest standing compression widely used, however as we'll see in this chapter Brotli⁶⁵⁴ has become the de facto algorithm for compressing textual data over the web ecosystem. In 2024, we also see the emergence of Zstandard which was developed by Facebook. Each of these algorithms has its strengths and use cases, and their adoption rates vary across the web.

Below is the analysis of compression types used by CDNs and origin servers in the Web Almanac 2024 reveals trends in how web content is optimized for delivery.

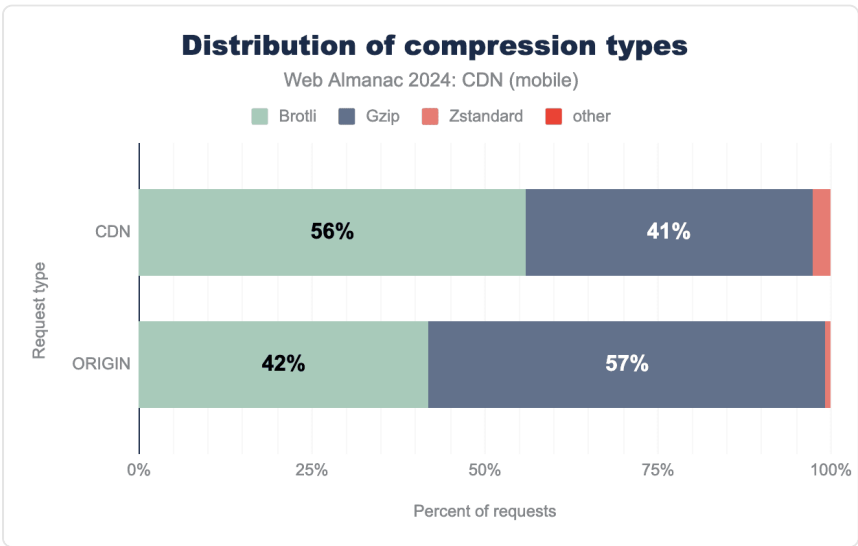


Figure 17.9. Distribution of compression types (mobile).

CDNs are leading in Brotli adoption, with over 55% of CDN served content using Brotli compression up from 47% in 2022. In contrast, less than 42% of content served directly from origin servers uses Brotli, however this is up from 25% from 2022. While Gzip remains the majority compression algorithm used by origin servers, Brotli continues its upward adoption

653. <https://wikipedia.org/wiki/Gzip>

654. <https://wikipedia.org/wiki/Brotli>

trajectory.

Zstandard (Zstd) adoption

Zstandard (Zstd) is a compression algorithm developed by Facebook and released in 2016. It aims to provide a good balance between compression ratio and speed, making it a potential alternative to established algorithms like Gzip and Brotli in web content delivery scenarios.

As of 2024, browser support for Zstandard in web content delivery has improved considerably:

- **Chrome:** Supported by default since version 121 (released January 2024)
- **Edge:** Supported by default since version 121 (released January 2024)
- **Firefox:** Supported behind a flag since version 123 (released March 2024)
- **Safari:** No native support as of the latest version
- **Opera:** Supported by default since version 108 (released January 2024)

This represents a significant shift in Zstandard's availability for web content delivery, with major Chromium-based browsers now offering native support.

Despite the recent improvements in browser support and Zstd's technical capabilities, our data shows that Zstandard adoption for web content delivery remains limited compared to Gzip and Brotli. CDNs only show 2.72% adoption rate of Zstandard and origin servers 0.70%.

While Zstd offers benefits, the real-world performance improvements over Brotli in web scenarios may not yet be fully established or significant enough to drive rapid adoption for all use cases. Zstd's flexibility in compression levels and dictionary compression may be particularly beneficial for certain types of content or delivery scenarios, which could lead to targeted adoption. We look forward to exploring this data more in future chapters.

Distribution of compression types

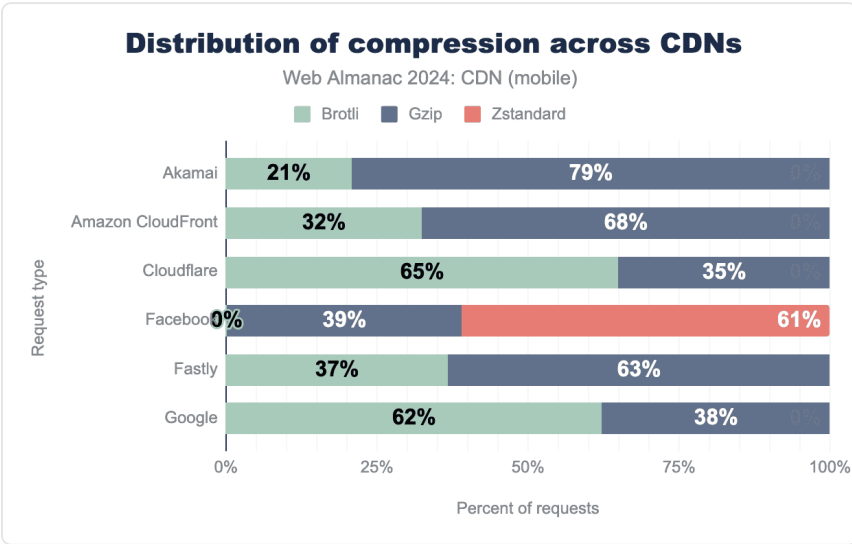


Figure 17.10. Distribution of compressions across CDNs (mobile).

Brotli usage is prevalent on Cloudflare and Google CDNs while Gzip remains the majority across Akamai, Amazon CloudFront, and Fastly. However, when compared to 2022 Brotli continues its broad trend towards more adoption with larger CDN providers. The outlier in our dataset was Facebook which had over 60% adoption of Zstandard. Facebook’s strategy has been to optimize content delivery using their own compression algorithm so this result is expected.

TLS usage

TLS 1.3 adoption

It’s good news that both CDN and origin requests have largely moved away from serving older, less secure TLS versions 1.0 and 1.1. This means that clients are now using more modern and secure protocols.

CDNs have embraced TLS 1.3, with 98% of requests using this latest version. This is great for developers because TLS 1.3 is faster at establishing secure connections, which means websites load more quickly. CDNs are at the forefront of adopting new technologies to optimize content delivery and security and by fronting your application with a CDN, you automatically reap

those benefits with minimal efforts.

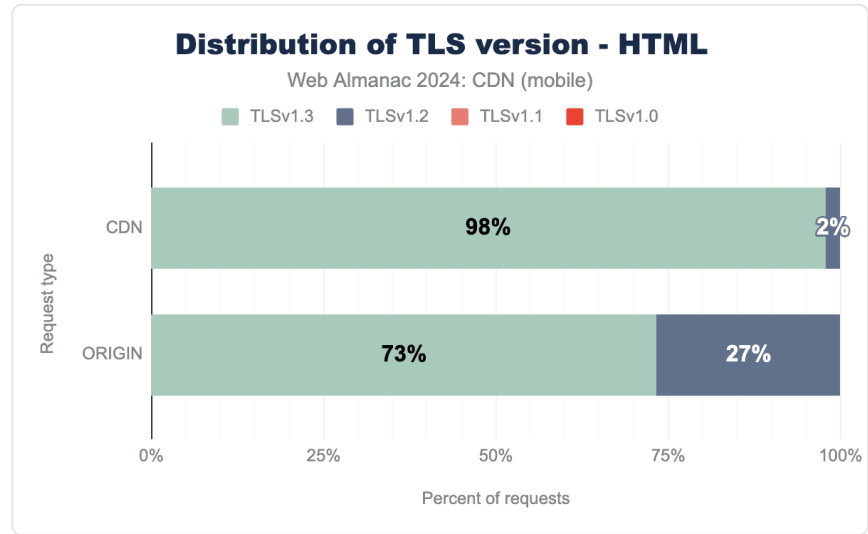


Figure 17.11. Distribution of TLS version for HTML (mobile).

When we look closely at how TLS is being used by different device types (Mobile vs Desktop), we find identical results where both mobile and desktop have 98% adoption of TLS 1.3 over CDNs. Mobile vs desktop directly to origin servers were nearly identical as well with mobile at 73% and desktop 71%. This represents a significant increase in TLS 1.3 adoption when compared to 2022. Mobile CDN TLS 1.3 traffic represented 87% in 2022 compared to 98% in 2024 and through origin servers from 42% now to 73%.

While origin servers have begun to catch up with adoption of TLS 1.3, this further shows how CDNs drive newer features quicker than when web server operators have to perform software and hardware upgrades for the same new features.

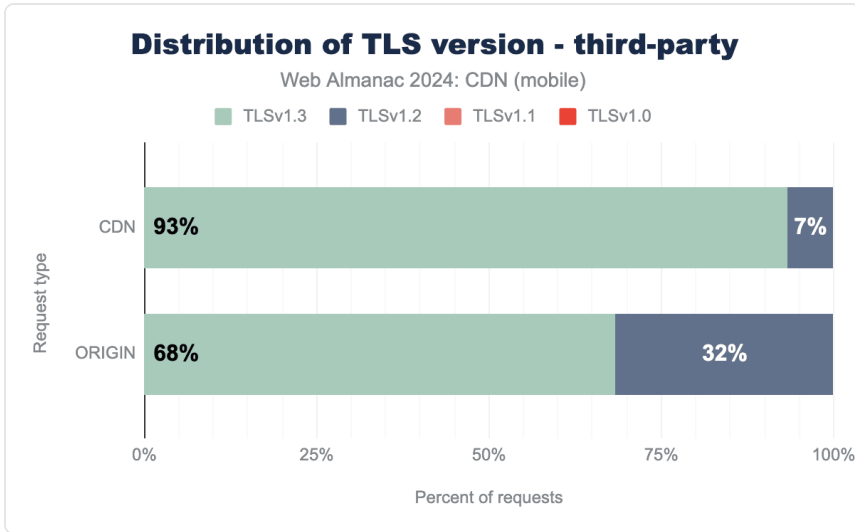


Figure 17.12. Distribution of TLS version for third-party requests (mobile).

We find a similar story with third-party TLS 1.3 adoption with CDNs at 93% and origins at 68%. There was a small increase from 2022 for CDNs TLS 1.3 from 88% to 93% but as with our other results, the origin server increased significantly with 2022 at 26% to now 68%.

TLS performance impact

TLS negotiation times reveal significant differences between CDN and Origin servers, as well as between desktop and mobile devices.

For desktop users, CDN performance is notably faster than Origin servers across all percentiles. The median (p50) TLS negotiation time for CDN is 70 milliseconds, compared to 183 milliseconds for Origin server. This trend is consistent across other percentiles, with CDN outperforming Origin at every level. For instance, at the 90th percentile (p90), CDN negotiation times are 108 milliseconds, while Origin servers take 289 milliseconds - more than 2.5 times longer.

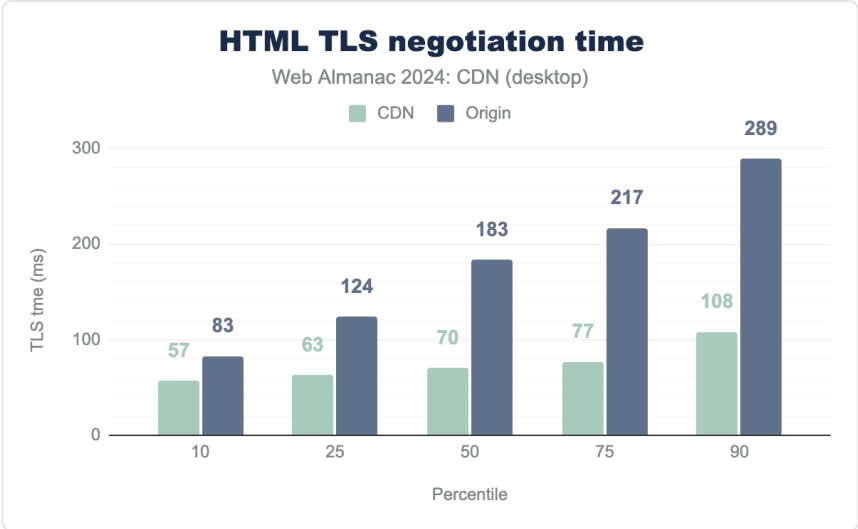


Figure 17.13. HTML TLS negotiation - CDN vs origin (desktop).

Mobile devices show a similar pattern, with CDN performing better than Origin servers, but the overall negotiation times are higher compared to desktop. The median TLS negotiation time for mobile CDN is 196 milliseconds, while for Origin servers it's 316 milliseconds. At the 90th percentile, mobile CDN takes 256 milliseconds, whereas Origin servers require 451 milliseconds.

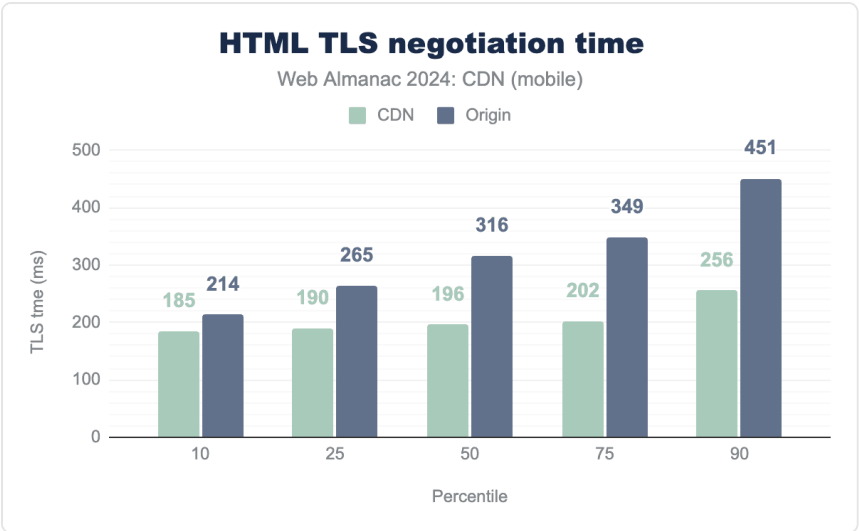


Figure 17.14. HTML TLS negotiation - CDN vs origin (mobile).

Comparing desktop and mobile TLS negotiation times, we observe that mobile devices consistently have longer TLS negotiation times regardless of whether request is served from CDN or Origin server. For example, the median negotiation time for mobile CDN (196 ms) is nearly three times that of desktop CDN (70 ms). This difference is less pronounced for Origin servers, but still significant, with mobile median times (316 ms) being about 1.7 times longer than desktop (183 ms).

The disparity between desktop and mobile performance is likely due to the typically lower processing power and potentially less stable network connections of mobile devices. The superior performance of CDN over Origin server can be attributed to the distributed nature of CDNs, which places content closer to end-users and optimizes for faster connections.

Image formats and optimization

Image formats play a crucial role in Content Delivery Networks (CDNs) and can significantly impact website performance, user experience, and overall efficiency. Modern image file formats like WebP and AVIF offer superior compression compared to traditional formats like JPEG and PNG. This results in smaller file sizes, which leads to faster page load times, reduced bandwidth usage and improved user experience.

Most CDNs can automatically detect the user’s browser capabilities and serve the most appropriate image format. For example: AVIF to Chrome browsers, WebP to Edge browsers,

JPEG to older browsers. They can further resize and cache images on the fly to handle responsive design requirements. This allows site operators to upload a single high resolution image and not having to maintain all its variations as site layout evolves.

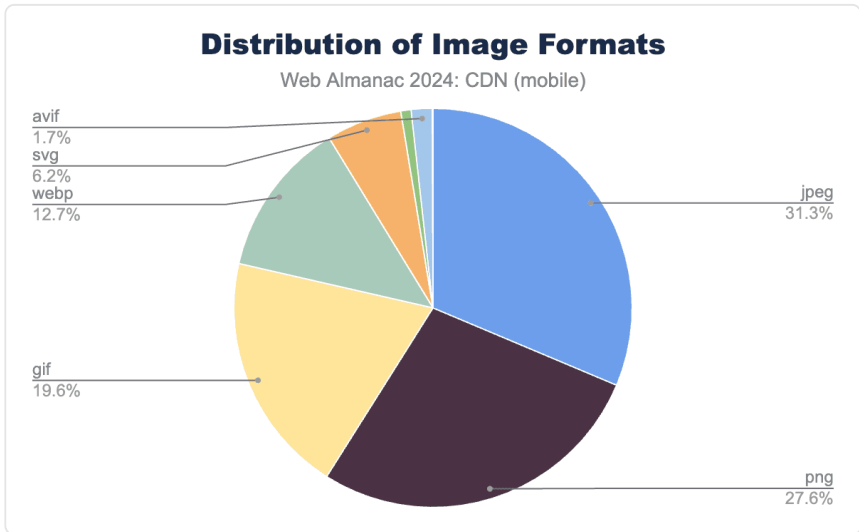


Figure 17.15. Distribution of Image Formats (mobile).

As of 2024, data reveals that while traditional formats like JPEG and PNG still dominate, there's a clear trend towards adopting more efficient and mobile-friendly formats like WebP and SVG. The mobile ecosystem generally shows higher numbers for most formats, reflecting the growing importance of mobile web usage. The presence of newer formats like AVIF suggests an industry push towards more efficient image delivery, which is crucial for improving web performance and user experience across all devices.

Client Hints

First proposed as a way to reduce information from the User-Agent string, Client Hints allows a web server to proactively request data from the client and are sent as part of the HTTP headers. Client Hints are divided into four categories: device, user-agent preferences, user preference media features, and networking. This further is broken down into high and low entropy hints. High entropy hints may provide the ability for the CDN or other entities to fingerprint and thus are typically gated by user permissions or other policies driven by the browser. Low entropy hints are less likely to be provide the ability for the client to be fingerprinted. Low entropy hints may be provided by default depending on user or browser

settings.

Based on the provided information, the server can determine the most optimal resources to respond with to the requesting client. While initially developed for Google Chrome browser, other Chromium based browsers have adopted it, but other popular browsers continue to have limited or no support for Client Hints.

The CDN, origin servers, and client browser must all support Client Hints to be utilized properly. As part of the flow, the CDN can present the Accept-CH HTTP header to the client in order to request which Client Hints a client should include in subsequent requests. We measured clients responses where the CDN provided this header inside the request and measured it across all CDN requests recorded as part of our testing.

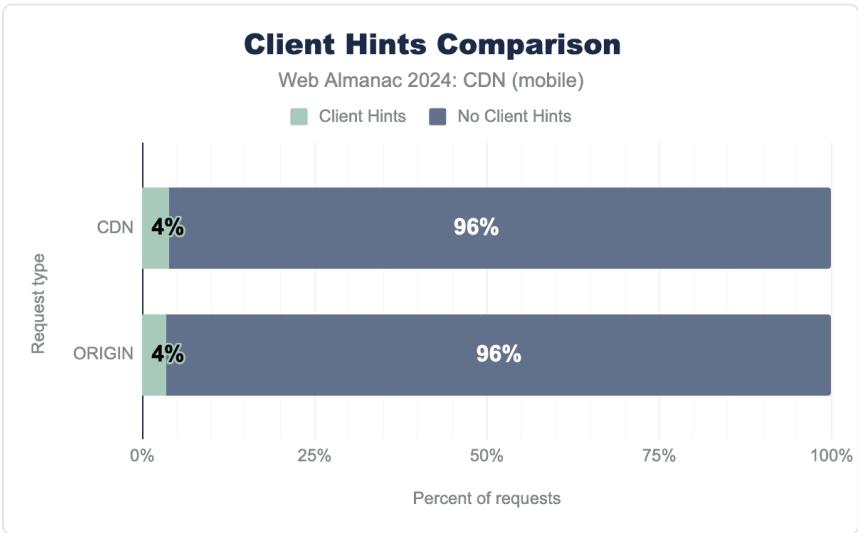


Figure 17.16. Client Hints Comparison (mobile).

In 2022, Client Hints adoption was at less than 1% for mobile requests. While the 2024 result was an increase with less than 4% of the requests for mobile devices indicating the presence of Client Hints, adoption of this capability remains low relative to the overall amount of requests observed. Though not explored in this year's chapter, if Client Hints adoption continues to grow we may in future chapters measure how CDNs are using the Accept-CH header to vary on for caching purposes and a more personalized experience.

Early Hints

Early Hints is the HTTP 103 status code⁶⁵⁵ that allows servers to send preliminary HTTP headers to browsers before the main response is ready. This is particularly valuable for preloading critical resources like stylesheet, JavaScript, and fonts.

While major browsers support Early Hints, we found hardly any adoption across the dataset. However, as seen with other newer and emerging features such as TLSv1.3, CDN's continue to lead the way in driving adoption compared to support going directly to web servers. Even still, we only observed CloudFlare and Fastly support Early Hints in any significant number compared to the rest of the CDN community.

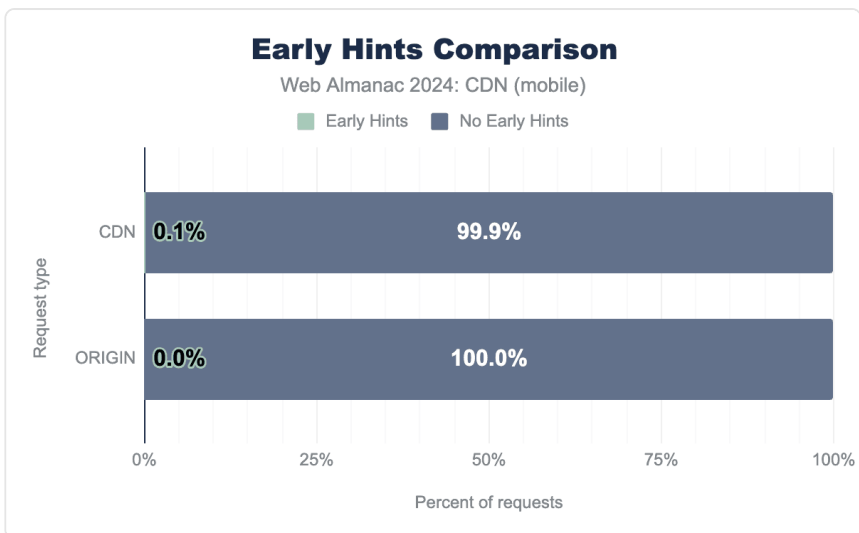


Figure 17.17. Early Hints Comparison (mobile).

As adoption of Early Hints increases we look forward to exploring more the performance implications that hints may provide. In future chapters we hope to observe more CDNs implement Early Hints and be able to show more granular statistics.

Conclusion

In 2022, we observed CDNs be the driving force behind the adoption of new and emerging technologies such as HTTP/3 and in 2024 this trend continued. Whether we look at

655. <https://datatracker.ietf.org/doc/html/rfc8297#section-2>

compression types like Brotli and ZStandard or encryption protocol TLS 1.3, CDNs reduce the heavy lifting of implementation with a simple configuration change rather than upgrading fleets of web servers, load balancers, and other network devices.

We looked into new metrics this year with Early Hints and ZStandard compression. We revisited Client Hints and Image Formats which were added in 2022. In 2025 we look to add more granular details for HTTP/3 and dive into how CDNs are impacting the adoption of IPv6.

All CDNs use custom developed or modified opensource technologies for caching and optimizing the moving of bits across their networks and the public internet. Due to this there are limitations to the insights we can deduce about CDNs from the outside. However, we have crawled the domains and compared the ones on CDNs against those who are not. We can see that CDNs have been an enabler for websites to adopt new web protocols, from the network layer to the application layer.

Content Delivery Networks are becoming increasingly vital to the internet's infrastructure, and their significance shows no signs of waning. Their technology remains essential for businesses that depend on the internet, ensuring seamless operations with speed, reliability, and security at the forefront.

We recommend readers visit the HTTP and Security chapters of the 2024 Web Almanac where several topics in this chapter are expanded on and provide data through a different lens.

Join us again in 2025 as we collect and analyze more data to see what new insights we can share with our readers.

Authors



Joe Viggiano

 [joeviggiano](#)

Joe Viggiano is a Principal Solutions Architect at Amazon Web Services helping Media & Entertainment customers deliver media content at scale.



Jaiganesh Girinathan

 [pgjaiganesh](#)

Jaiganesh Girinathan is a Principal Solutions Architect at Amazon Web Services with the mission to help customers deliver a fast and secure digital experience.



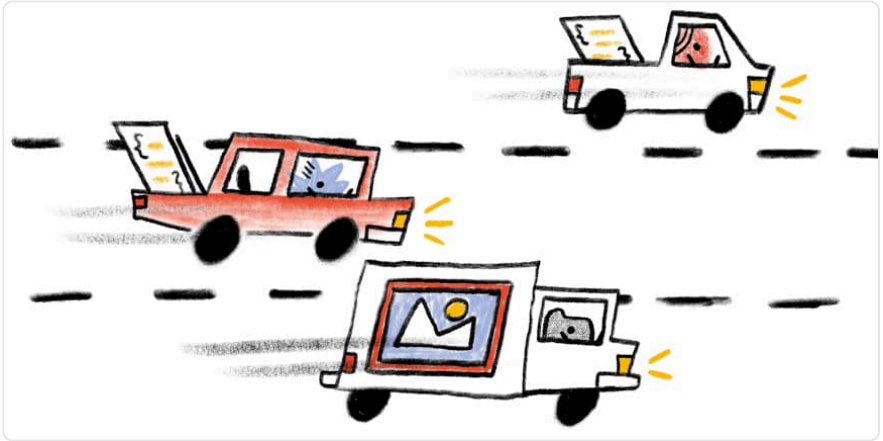
Alex Moening

 AlexMoening

Alex Moening is a Senior Edge Solutions Architect at Amazon Web Services.

Part IV Chapter 18

HTTP



Written by Robin Marx

Reviewed by Barry Pollard and Chris Böttger

Analyzed and edited by Barry Pollard

Introduction

HTTP remains the cornerstone of the web ecosystem, providing the foundation for exchanging data and enabling various types of internet services. It is an actively developed protocol, with the latest version HTTP/3⁶⁵⁶ standardized a little over two years ago, and new options to enable it recently becoming available, such as the new DNS HTTPS records⁶⁵⁷. At the same time, the web platform has been exposing more and more higher-level features that web developers can use to influence when and how resources are requested and downloaded over HTTP. This includes options like Resource Hints⁶⁵⁸ (for example preload and preconnect), 103 Early Hints⁶⁵⁹ and the Fetch Priority API⁶⁶⁰.

In this chapter, we will first look at the current state of HTTP/1.1, HTTP/2 and HTTP/3 adoption, and how their usage has evolved over time. We then consider the new web platform

⁶⁵⁶. <https://datatracker.ietf.org/doc/html/rfc9114>

⁶⁵⁷. https://developer.mozilla.org/docs/Glossary/HTTPS_RR

⁶⁵⁸. <https://web.dev/learn/performance/resource-hints>

⁶⁵⁹. <https://developer.mozilla.org/docs/Web/HTTP/Status/103>

⁶⁶⁰. <https://web.dev/articles/fetch-priority>

features, to get an idea of how well they're supported and how people are using them in practice.

HTTP version adoption

Conceptually, getting an idea of how widespread the adoption of HTTP/2 and HTTP/3 is should be easy: just report how often each of the protocol versions was used to load the observed web pages in our dataset. This is exactly what we've done in the graph below:

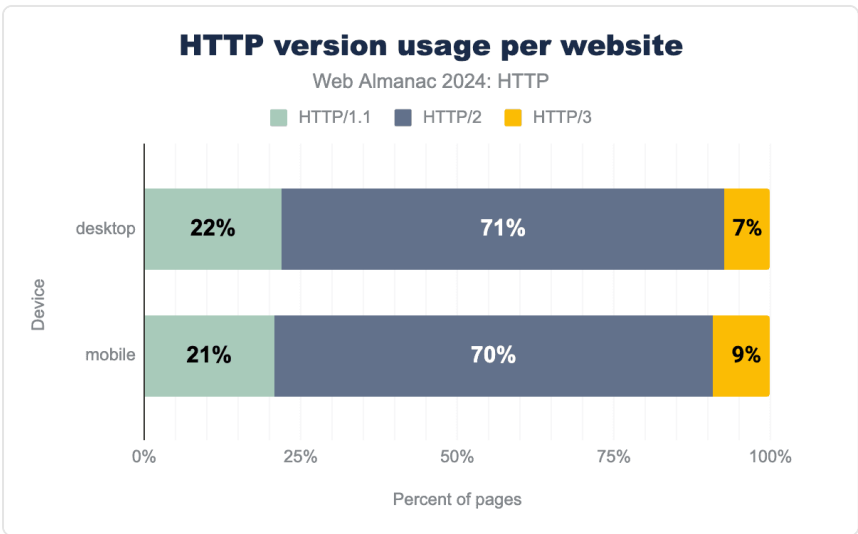


Figure 18.1. Adoption of HTTP versions as a percentage of website home pages.

While these results might look sensible at first glance, they are actually quite misleading. Indeed, as we'll see later, in reality HTTP/3 support across the web is closer to 30% instead of the 7-9% reported above. This discrepancy is due to the fact that HTTP/3 has to be discovered before being used, while the methodology used for the Web Almanac doesn't really lend itself well to the main discovery option (see `HTTP/3 via alt-svc`). This causes a lot of HTTP/3-capable sites to still be loaded over HTTP/2 and thus HTTP/3 is getting underreported. We will discuss this in more detail below, but for now we will bypass this issue by grouping HTTP/2 and HTTP/3 together into a single label of **HTTP/2+** to at least compare them to HTTP/1.1 in general.

As such, we see that only 21-22% of home pages are loaded over HTTP/1.1 in 2024, a marked

difference from 2022 where it was still 34%⁶⁶¹ and especially 2020 where there was basically a 50/50 split⁶⁶² between HTTP/1.1 and HTTP/2+. However, this is just looking at how the main document (the HTML) of the page is loaded. Another way of looking at HTTP adoption is by looking at the version used for all requests (including subresources and third-parties), which skews the results even more towards HTTP/2+:

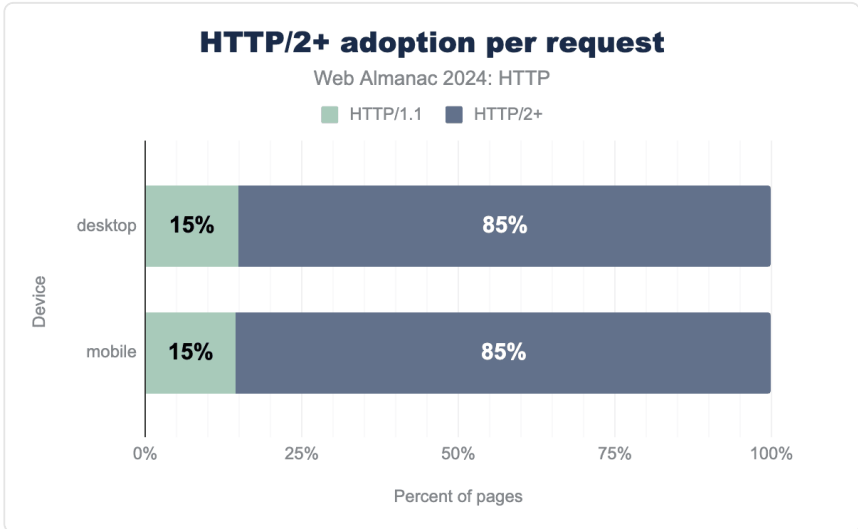


Figure 18.2. Adoption of HTTP/2 and above as a percentage of requests.

This is because many websites will load additional resources from a variety of third-party domains (for example, analytics, plugins/tags, and social media integrations). Due to their scale, these external services often either provide support for HTTP/2+ themselves or make use of a so-called CDN or Content Delivery Network (for example Akamai, Cloudflare or Fastly) that does it for them. In fact, CDNs are used very heavily on the observed websites in our dataset, with a whopping 54% of over 1.3 billion requests in our dataset being served from a CDN!

54%

Figure 18.3. The percentage of over 1.3 billion requests in the Web Almanac dataset that use a CDN.

These companies are typically at the forefront of implementing new standards and protocols.

⁶⁶¹. <https://almanac.httparchive.org/en/2022/http#http2-adoption>

⁶⁶². <https://almanac.httparchive.org/en/2020/http#http2-adoption>

When they enable a new feature, it becomes available to all their customers, usually causing a fast increase in global adoption. As such, we can see that CDNs are still one of the main drivers of HTTP/2+ adoption, with less than 4% of all CDN requests happening over HTTP/1.1. This is again in stark contrast to requests not served from a CDN node, of which up to 29% still use HTTP/1.1.

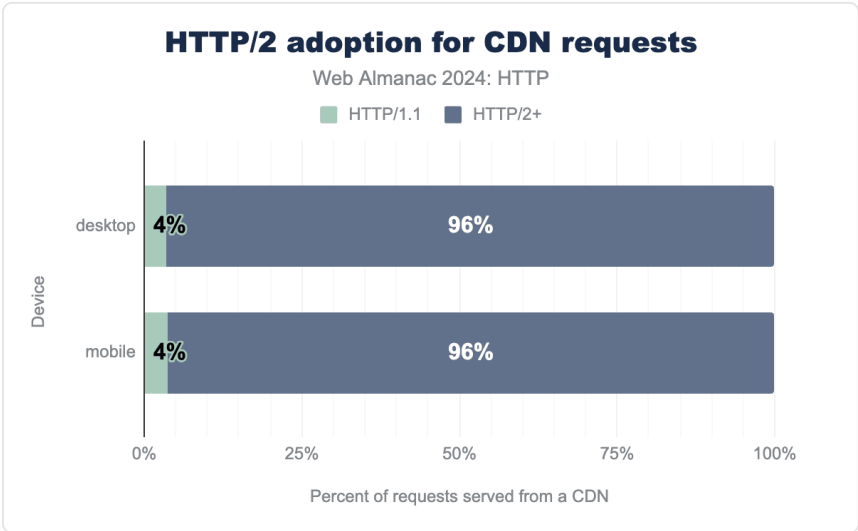


Figure 18.4. Almost all traffic from CDNs is delivered over HTTP/2+.

Combined, these results show that while overall adoption of newer HTTP versions is higher than ever, there are still plenty of sites that are in danger of “being left behind”. These are mostly the (likely smaller) projects that can’t or won’t engage a CDN, but also don’t have the technical know-how or motivation to enable HTTP/2+ themselves at their origins. Conceptually, this makes sense, as the newer versions do have some costs associated with them, both in terms of complexity and—especially for HTTP/3—in terms of CPU usage. And while HTTP/1.1 of course still functions perfectly fine, there might be some performance/efficiency downsides, as only HTTP/2+ allows multiplexing of multiple resources onto one connection, as well as additional features like resource prioritization and header compression.

Still, personally I feel it’s good to have freedom of choice on the web, not just in terms of content, but also the tech stacks that are used, and that includes the protocols. Importantly, the current (over)reliance on CDNs (for performance and security) also does have its own downsides, with some people fearing a “centralization” of the web⁶⁶³ around a few large companies. I don’t think we’re quite there yet, but I have written previously⁶⁶⁴ about the

663. <https://www.mnot.net/blog/2023/12/19/standards-and-centralization>

664. <https://pulse.internetsociety.org/blog/the-challenges-ahead-for-http-3>

worrying fact that it is increasingly only the CDNs/large companies that have the technical know-how to deploy the newer protocols at scale. I don't know the right answer here, but feel it's good to highlight both sides! However you feel about this, it is clear that CDNs already make up a large part of the web today, and that they are driving adoption of the newer HTTP versions.

Let's thus move away from the philosophical back into the (deeply) technical now, and explore why exactly the Web Almanac dataset is only seeing low percentages of HTTP/3 being used!

Discovering HTTP/3 support

Measuring HTTP/3 adoption can be challenging because most modern browsers and clients will not try HTTP/3 the first time they load a page from a domain they haven't seen before. The reasons why are complex and were explained in detail in previous Web Almanacs (2020⁶⁶⁵ and 2021⁶⁶⁶). The short version is that it could take the browser a (very) long time to fall back to HTTP/2 (or HTTP/1.1) if HTTP/3 is not available for the target domain—for example if the server doesn't support it yet, or if the network is blocking the protocol.

This is less of a problem for HTTP/2 and HTTP/1.1, as they both run over the TCP protocol. If the server doesn't support HTTP/2, it can just continue with HTTP/1.1 over the existing TCP connection the browser set up, getting an “instant fallback”. HTTP/3 however is different, as it replaces TCP with a new transport protocol called QUIC⁶⁶⁷ which in turn runs over the UDP protocol. If the browser *only* opens a QUIC+HTTP/3 connection to a server that doesn't support it, that server wouldn't be able to fall back to HTTP/2 or HTTP/1.1, since that's only possible over TCP, not QUIC/UDP. It would have to wait for the HTTP/3 connection to timeout (which can take several seconds) and only then open a new TCP connection for HTTP/2 or HTTP/1.1, which would be very noticeable for the end users.

To prevent users suffering this potential delay, the browser will in practice only try HTTP/3 if it's 100% sure the server will support it. But how can it be 100% sure? Well, if the server/deployment explicitly tells the browser it supports HTTP/3 first, of course!

There are two main ways of doing this:

1. **The `alt-svc` HTTP response header:** the HTTP server advertises HTTP/3 support when a resource is requested over HTTP/2 or HTTP/1.1
2. **The DNS HTTPS resource record:** the DNS server indicates HTTP/3 support during name resolution, before connection establishment

The first option is by far the most popular today, but has the downside that it first needs a

665. <https://almanac.httparchive.org/en/2020/http#deploying-and-discovering-http3>

666. <https://almanac.httparchive.org/en/2021/http#negotiating-http3>

667. <https://www.smashingmagazine.com/2021/08/http3-core-concepts-part1/>

“bootstrapping” connection over HTTP/2 or HTTP/1.1 to discover HTTP/3 support. The second option removes this downside by putting the information directly in the Domain Name System (DNS), which resolves before the first HTTP connection is even made. It is however much newer and thus less supported at the moment of writing.

To get a good idea of real HTTP/3 support in the Web Almanac dataset, we need to consider both separately, as they will give conflicting results.

HTTP/3 via `alt-svc`

As discussed above, if the browser hasn’t connected to a domain before, it will only try HTTP/2 or HTTP/1.1 over TCP, as those are most likely to be supported. For each HTTP/2 or HTTP/1.1 response, the server can then send along a special `alt-svc` HTTP response header, indicating it is guaranteed to support HTTP/3 as well—at least for a specified timeframe: the `ma` (max-age) parameter.



X	Headers	Preview	Response	Initiator	Timing	Cookies
▼ General						
Request URL:	https://www.akamai.com/de					
Request Method:	GET					
Status Code:	● 200 OK					
Remote Address:	2.17.107.128:443					
Referrer Policy:	strict-origin-when-cross-origin					
▼ Response Headers						
Accept-Ch:	DPR, Width, Viewport-Width, Dnlink, Save-Data					
Akamai-Grn:	0.7c6b1102.1729800463.5f0019e					
Alt-Svc:	h3=\":443\"; ma=93600					
Cache-Control:	max-age=604800					

Figure 18.5. `alt-svc` response header example.

`alt-svc` stands for **Alternative Services**: you’re currently using the HTTP/2 service, and there’s also an HTTP/3 service available (usually at UDP port 443). From then on, when the browser needs a new connection to the server, it can try to establish it over HTTP/3 as well! Using this mechanism thus means that HTTP/3 is typically only used from the second page load of a site onward, as the first load will happen over H2 or HTTP/1.1, even if the server supports HTTP/3. And that is the crux of the issue here, as the Web Almanac only measures the first page load by design.

In order to make results comparable and fair across websites, we want to load each page with a fresh browser profile and nothing in the HTTP/file/DNS/alt-svc/... caches. As such, the `alt-svc` mechanism is conceptually useless in our methodology, as we'd only use the initial HTTP/2 connection and never get to HTTP/3. This is why measuring HTTP/3 adoption directly by protocol used in our dataset (as we did in the first image above) is misleading, with HTTP/3 getting underreported.

Note: at this point, you might wonder how we're even seeing HTTP/3 page loads at all since with just `alt-svc` we should be seeing 0% HTTP/3. This is of course mainly due to the use of the 2nd discovery method via DNS, which we'll discuss later.

Let's now look purely at how HTTP/3 support is being announced via `alt-svc` to get an idea of how much support sites claim to have, even if it doesn't actually show up in our dataset:

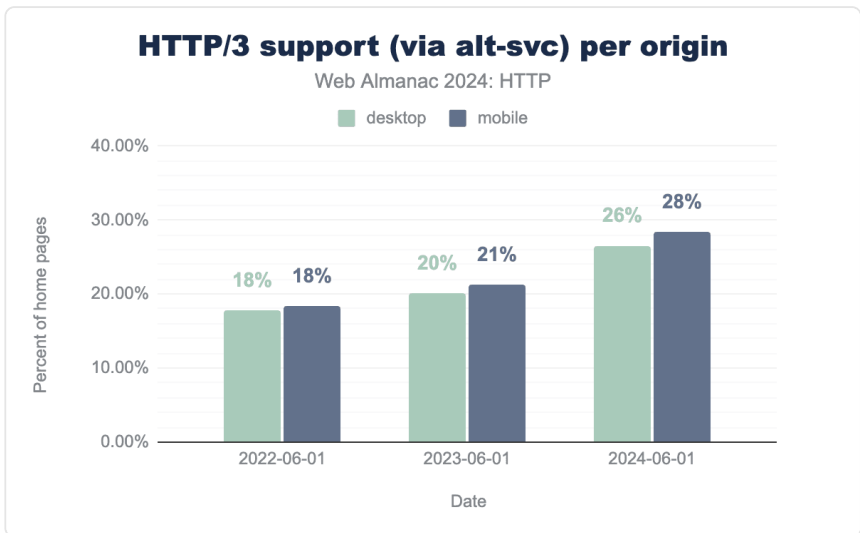


Figure 18.6. HTTP/3 support (via `alt-svc`) has increased steadily since 2022.

When we last looked at HTTP/3 support⁶⁶⁸ in June 2022, the protocol wasn't even fully standardized yet. Still, due to early deployments from some large players, around 18% of sites in the HTTP Archive dataset indicated they had support for HTTP/3. Now, two years later, we can see that support for the new protocol has steadily risen, up to 26% (desktop) to 28% (mobile), a near 10% overall increase. If that still seems low, it's actually quite similar to HTTP/2's evolution, which equally saw around 30% uptake in its second year⁶⁶⁹ after standardization (2017).

⁶⁶⁸. <https://almanac.httparchive.org/en/2022/http#http3-support>

⁶⁶⁹. <https://httparchive.org/reports/state-of-the-web#h2>

It is somewhat interesting to see that mobile home pages advertise a little better support for HTTP/3 than their desktop counterparts. This can potentially be explained by the fact that HTTP/3 will mainly deliver benefits over mobile/cellular networks (and so site owners might want to mainly enable it for that) and because some corporate environments still tend to block HTTP/3 on their networks (making it less interesting to enable for desktop clients).

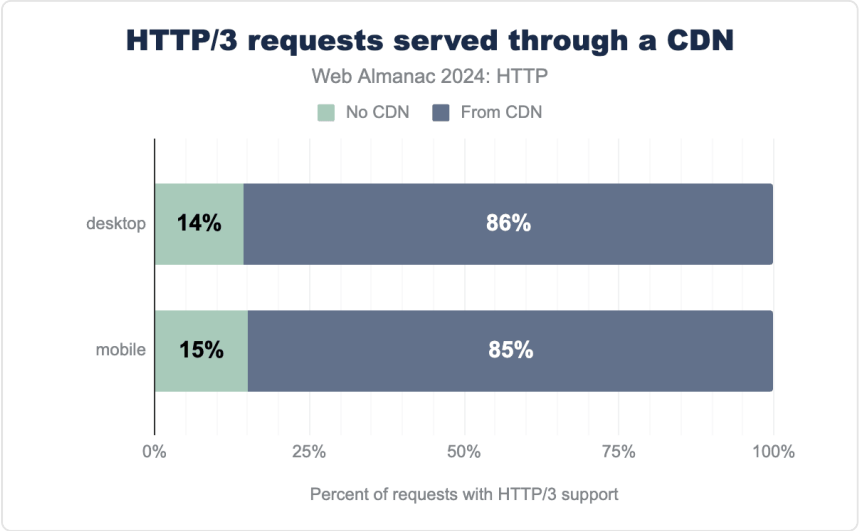


Figure 18.7. HTTP/3 support is driven mainly by CDNs.

Similar to HTTP/2+ above, the support for HTTP/3 comes mainly from the CDNs, but in a quite extreme form in my opinion: around 85% of all HTTP/3 responses seen in our dataset came from a CDN. This compares to around 55% of all HTTP/2+ requests. This indicates that today very few website owners are self-deploying HTTP/3 at their origin and re-emphasizes my point from above that fast adoption of new technologies might (sadly) become a large-company-only thing. This is not entirely unexpected however; a lot of the popular “off the shelf” web servers do not have stable, mature, on-by-default HTTP/3 support yet—including projects like NodeJS, Apache and nginx. Running a scalable HTTP/3 deployment that uses some of the protocol’s more advanced features, like connection migration and 0-RTT, is far from easy. Still, I hope to see more people self-hosting HTTP/3 in the near future.

One important remark to make here is that not all CDNs show equally high HTTP/3 support:

HTTP/3 alt-svc	CDN
> 90%	Facebook, Automattic, jsDelivr
50% - 90%	Cloudflare, Cedexis
10% - 50%	Google, Amazon Cloudfront, Fastly, Microsoft Azure, BunnyCDN, Alibaba, CDN 77
1% - 10%	Akamai, Sucuri Firewall, Azion, KeyCDN
< 1%	Twitter, Vercel, Netlify, OVH CDN, EdgeCast, G-Core CDN, Incapsula

Figure 18.8. Percentage of HTTP/3 alt-svc responses for all requests served by first-party CDNs (with considerable traffic share).

Firstly, we see that some companies seem to go all-in on HTTP/3, with Facebook for example indicating HTTP/3 support on 99.86% of responses! This is in stark contrast to similar companies like Twitter/X, that don't send alt-svc at all. Secondly, even some CDNs that clearly support HTTP/3 rarely reach high percentages, with Cloudflare leading the pack at 78% and Akamai tagging along at just 7%.

This most likely has to do with how exactly the CDNs enable the new protocol. For example, Cloudflare has enabled it by default for their free plans, leading to high (though not universal) use. In contrast, Akamai requires its customers to manually enable the feature in their configuration, which many seem slow/unwilling to do. As such, the amount of HTTP/3 support on the web *could* be much higher than around 28% if all CDN customers would enable it.

Finally, it is somewhat surprising that some more specialized deployments, like Automattic, heavily lean into HTTP/3 (99.92%) while others like Vercel and Netlify show near-zero HTTP/3 support. I can only speculate on the reasons for the latter, but assume it is mostly due to the complexity of setting up and maintaining the new protocol at scale, while these newer up-and-coming companies might prefer to focus on other parts of their stacks first.

Note: These results might not give a 100% accurate impression of what CDNs are actually doing, as firstly the CDN detection logic⁶⁷⁰ might be inaccurate/incomplete (for example, Shopify nor Apple are tracked separately yet), and secondly a lot of the tracked requests are for analytics trackers, which might not be indicative of HTTP/3 support for normal page/resource loads—especially for sites like Facebook.

In conclusion, we see that currently around 27% of all websites in the Web Almanac dataset announce HTTP/3 support through alt-svc. This number could potentially be much higher, if all websites that use a compatible CDN would enable it. At the same time however, the actual number of HTTP/3 requests we've seen in the dataset is much lower, between 7-9%. As we've

670. https://github.com/HTTPArchive/wptagent/blob/5ef2c870a90a3492cd6170893812270d627df107/internal/optimization_checks.py#L67

explained, this is due to the used methodology and most of those 7-9% will come from another HTTP/3 discovery method using DNS records, so let's look at those next.

HTTP/3 via DNS

We now understand that announcing HTTP/3 support via an `alt-svc` HTTP response header has some downsides, leading to the newer protocol only being used from the second connection to a server onwards. To get rid of this inefficiency, the browser would have to discover HTTP/3 support before it even opened the first connection to the server. Luckily, there is still one thing that happens before a connection can be setup and that is: DNS resolution.

The Domain Name System (DNS) is often described as a large phone book, which allows you to translate a hostname (for example `www.example.org`) into one or more IP addresses (with `A` and `AAAA` queries returning IPv4 and IPv6 results respectively). However, in essence, the DNS can also be thought of as just a very large, distributed Key-Value store that can hold other things besides IP addresses (and associated metadata like `CNAME`s) as well. Some examples include listing email details with `MX` records, and using `TXT` records to prove ownership of a domain (for example for Let's Encrypt⁶⁷¹).

The past few years, work has been ongoing to also add other information to the DNS, in particular with the new concept of Service Binding (SVCB) and HTTPS records⁶⁷². These new records provide the client with a lot more information about a service/origin than just its IP addresses: whether it's HTTPS capable, whether it can use the new Encrypted Client Hello⁶⁷³ for extra privacy, or for our purposes, which protocols it supports and on which ports. This is intended to make initial connection setup/service discovery a bit more efficient, as currently this is often done through slower methods like a chain of redirects or the above `alt-svc`, or requires out-of-band methods like HSTS preload⁶⁷⁴. They are also intended to help with complex load balancing setups (for example when combining multiple CDNs).

A full discussion on SVCB would take too much time here however, so we will focus only on how we can announce HTTP/3 support through the new HTTPS record as there are plenty of other⁶⁷⁵ blog posts⁶⁷⁶ and documents⁶⁷⁷ with more details⁶⁷⁸ on the wider applications. Let's look at an example of the HTTPS record in the wild:

671. <https://letsencrypt.org/docs/challenge-types/#dns-01-challenge>

672. <https://www.rfc-editor.org/rfc/rfc9460.html#name-goals>

673. <https://support.mozilla.org/kb/understand-encrypted-client-hello>

674. <https://developer.mozilla.org/docs/Web/HTTP/Headers/Strict-Transport-Security>

675. <https://www.domaintools.com/resources/blog/the-use-cases-and-benefits-of-svcb-and-https-dns-record-types/>

676. <https://blog.cloudflare.com/speeding-up-https-and-http-3-negotiation-with-dns/>

677. <https://www.isc.org/docs/2022-webinar-dns-svcb.pdf>

678. <https://www.netmeister.org/blog/https-rs.html>

```
root@localhost:~# dig +short https blog.cloudflare.com
1 . alpn="h3,h2" ipv4hint=104.18.28.7,104.18.29.7 ipv6hint=2606:4700::6812:1c07,2606:4700::6812:1d07
```

Figure 18.9. DNS HTTPS resource record example.

As we can see, `blog.cloudflare.com` indicates support for both HTTP/3 and HTTP/2 (in order of preference!) via the `alpn="h3,h2"` part of the response. ALPN stands for Application Layer Protocol Negotiation⁶⁷⁹ which was originally a TLS (Transport Layer Security protocol) extension to indicate which application protocols and versions a server supports. For example, to allow the graceful fallback from HTTP/2 to HTTP/1.1 discussed above.

The general approach—and ALPN name—is reused for the DNS HTTPS record as well. Additionally, the example shows the optional `ipv4hint` and `ipv6hint` entries, which allow steering of users to specific endpoints for specific services. For example, if not every single machine in the deployment actually supports HTTP/3 yet, say in a multi-CDN setup.

In conclusion, if a browser queries the DNS for the HTTPS records (which is typically done in parallel or even before A and AAAA queries), and subsequently sees `h3` in the ALPN list, then it is allowed/encouraged to also try HTTP/3 for its first connection to the server. This bypasses the `alt-svc` bootstrapping overhead.

Let's now take a look at how much we've seen the new DNS records being used in the wild in the Web Almanac dataset. Looking at the general use, we see that around 12% of both mobile and desktop pages have an HTTPS record of some kind defined. Not all of those include the `h3` option in their `alpn` section however: that's slightly lower at 9% (desktop) and 10% (mobile):

679. <https://developer.mozilla.org/docs/Glossary/ALPN>

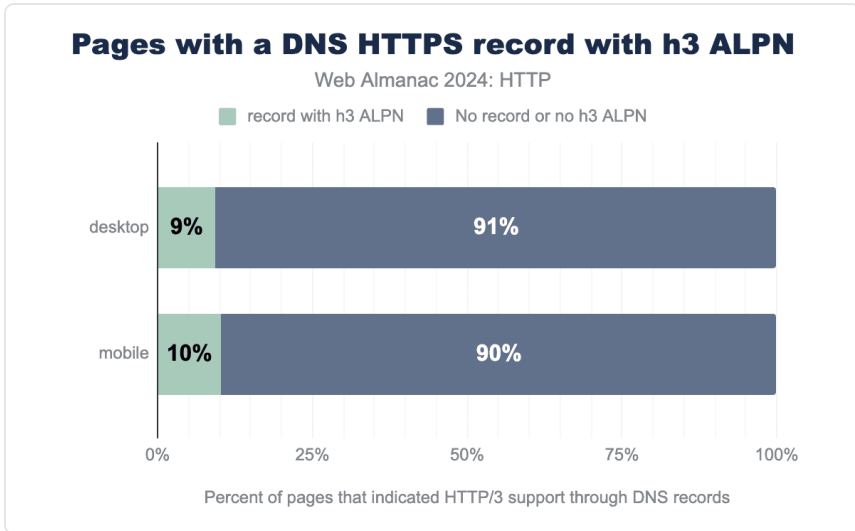


Figure 18.10. HTTP/3 support is regularly being announced through DNS HTTPS records

This means that 9-10% (or over 20 million) of all considered pages in our dataset indicate HTTP/3 support through DNS. However, this does not automatically mean that HTTP/3 is also actually used by the browser. As we showed in the first image of this chapter, only about 7% (desktop) to 9% (mobile) of pages were actually loaded over HTTP/3. This is definitely in the same order of magnitude as the DNS HTTPS adoption, but not quite the same.

This can have many different reasons, including: networks blocking the newer protocol, HTTP/3 somehow losing the “race” to a HTTP/2 connection (we’ll discuss this in the next section, Other considerations), the DNS HTTPS record being misconfigured, the DNS response for the record being delayed, the HTTP/2 connection being reused due to connection coalescing⁶⁸⁰. Still, it shows that this newer/alternative method of indicating HTTP/3 support early in the page loading process has strong potential to improve upon the `alt-svc` approach! This is especially true for our Web Almanac methodology, as we confirmed that 99% of all page loads observed over HTTP/3 were indeed triggered by the presence of DNS HTTPS records (the 1% discrepancy is a bit weird though, and a good topic for future analysis).

It is also interesting to compare this to similar research done by Jan Schaumann in October 2023⁶⁸¹, who found that for over 100 million tested domains, only about 4% provided HTTPS records, which increased to a whopping 25%+ for the top 1 million domains on the Tranco list. He concluded⁶⁸² that DNS HTTPS record adoption is “effectively driven by Cloudflare setting the

680. <https://blog.cloudflare.com/connection-coalescing-with-origin-frames-fewer-dns-queries-fewer-connections/>

681. <https://www.netmeister.org/blog/https-rs.html#current-use>

682. <https://www.netmeister.org/blog/https-rs.html#phints>

records by default on all of their domains”. This would be in-line with our previous findings that it is the big CDNs that drive new feature adoption, so let’s see what our data says:

h3 DNS records	CDN
> 60%	None (yet!)
50% - 60%	Cloudflare
2% - 50%	BunnyCDN, Alibaba
< 2%	Akamai, Amazon Cloudfront, Microsoft Azure, Fastly, Netlify, Google, Incapsula, Azion, Sucuri Firewall, Vercel
< 0.05%	Automattic, OVH

Figure 18.11. Percentage of DNS HTTPS responses with HTTP/3 support for all home pages served by first-party CDNs (with considerable traffic share)

We can indeed see that Cloudflare is clearly the leader of the pack here, setting `h3` in the HTTPS DNS record for over 50% of its hosted sites. Most of the other large CDNs do seem to have some support and are testing the feature, but they rarely get above 2%. An interesting outlier here is Automattic, which had near universal HTTP/3 support via `alt-svc`, but only 0.04% for DNS HTTPS records. Outside this, of the 8.5 million pages not loaded via a CDN, only 0.46% had a DNS HTTPS record configured for HTTP/3 support, again reinforcing our conclusion that (for better or worse) CDNs are the ones driving adoption of cutting edge features at scale.

It will be interesting to track the use of DNS HTTPS and SVCB records in the coming years of the Web Almanac to both see how their adoption evolves, and how that will map to actual HTTP/3 use in the dataset.

Other considerations

In practice, there is even more complexity in the protocol selection/connection setup process used by modern browsers.

One example is an algorithm called *Happy Eyeballs* (yes, really!), which describes how to test for and choose from several different options. This is used to decide between HTTP/2 and HTTP/3, but also IPv4 and IPv6—for which it was originally invented. This algorithm typically “races” different connections against each other and then picks the “winner” to continue the page load on-which means that we sometimes will still see HTTP/2 even though HTTP/3 is supported, if HTTP/2 wins the race. This data is not yet tracked in our dataset, so we don’t really know how often this happens-though in practice this would also heavily depend on testing location and

the network used.

Another example is the concept of *connection coalescing* (which they really should have just called “connection reuse” in my opinion), which says that browsers should prefer to reuse an existing connection instead of opening a new one. In practice, if two domains (`a.com` and `b.com`) share the same TLS certificate, the browser can (and often does) re-use an existing connection to `a.com` to fetch `b.com/main.js`. You can imagine the headaches this gives⁶⁸³ if `a.com` has HTTP/3 enabled and `b.com` does not... We did not yet analyze how often this happens in the Web Almanac dataset, but from personal experience debugging problems with this, I can assure you it’s definitely out there!

96%

Figure 18.12. The percentage of pages that load resources from `cdn.shopify.com` that see both HTTP/2 and HTTP/3 connections to that domain.

Finally, browsers don’t always wait for an existing (HTTP/2) connection to be closed before opening a new (HTTP/3) connection; sometimes they try to switch much more aggressively, even during an ongoing page load over HTTP/2! This can cause “hybrid” page loads, which makes interpreting Real User Monitoring (RUM) metrics and comparing HTTP/2 to HTTP/3 performance quite challenging. Measuring how often this happens in our dataset is a bit tricky, but we tried to get an idea by seeing how many domains had *both* HTTP/2 and HTTP/3 connections opened for them during individual page loads. We find a high dual-protocol usage especially for third parties, with for example `connect.facebook.com` seeing both HTTP/2 and HTTP/3 in the same page load 34% of the time, and `cdn.shopify.com` even switching in 96% of cases. It’s not always this high though, with `www.facebook.com` interestingly only seeing a switch on 12% of pages it’s used on, and various `wp.com` trackers showing only 1-6% (likely because only a few resources are loaded from these domains and there’s no time/need to switch). One caveat is that these switches are also potentially influenced by other factors, such as needing a new connection for CORS reasons or being used in an `iframe`. Still, I believe the data shows that those “hybrid” page loads are quite common and should be taken into account, even though it’s yet unclear which exact impact this has/can have on performance metrics like Largest Contentful Paint.

All of this just reinforces our story so far that there is a lot going on at the network layer nowadays, which makes everything more difficult, from properly judging real adoption from our dataset, to deploying the protocol yourself without CDN support, to correctly analyzing and debugging results from RUM data. While adoption of these newer features is steadily rising due

683. <https://youtu.be/ljtn5iwKWQ?i=362>

to support from big deployments, it will probably level out relatively soon, taking a long time for smaller deployments and especially individuals to start using the newer protocols. Just like we still see a considerable amount of HTTP/1.1 out there, I expect HTTP/2 also won't be going anywhere for quite some time.

Now that I've probably scared you away from ever looking into the internals of protocols ever again, let's consider some higher-level features that allow you to nudge their behaviour without having to understand what ALPN or SVCB stand for.

Higher-level browser APIs

As we've seen in the first part of this chapter, the adoption of HTTP/2 and HTTP/3 are on the rise, and that's a good thing (mostly). These new protocols implement a lot of performance and security best practices that have tangible benefits for end users. For developers however, the protocols and their features often remain a black box, as there are very few ways to tune them directly. You basically just tick a box on your CDN or server configuration, and hope the browser and server get it right.

However, there are a few higher-level features (such as image lazy loading, `async` / `defer` javascript attributes, Resource Hints, the Fetch Priority API), that allow us to influence what happens on the network to an extent. Even though these are not technically always directly tied to HTTP as a protocol, they can have a major impact on how some protocol features (such as connection establishment and resource multiplexing) are used in practice, so we discuss some of them in this chapter.

Resource Hints

Firstly, there are the “Resource Hints”⁶⁸⁴, a group of directives that can be used to guide the browser in various network-related operations, from setting up (parts of) a network connection, over loading a single resource, to doing fetches of entire pages ahead of time. The main ones are `dns-prefetch`, `preconnect`, `preload`, `prefetch`, and `modulepreload`. There was also a `prerender` option which had a bit of a hard time finding its place in the ecosystem, and that use case now moved mostly to the new Speculation Rules API⁶⁸⁵.

684. <https://web.dev/learn/performance/resource-hints>

685. <https://developer.chrome.com/docs/web-platform/prerender-pages>

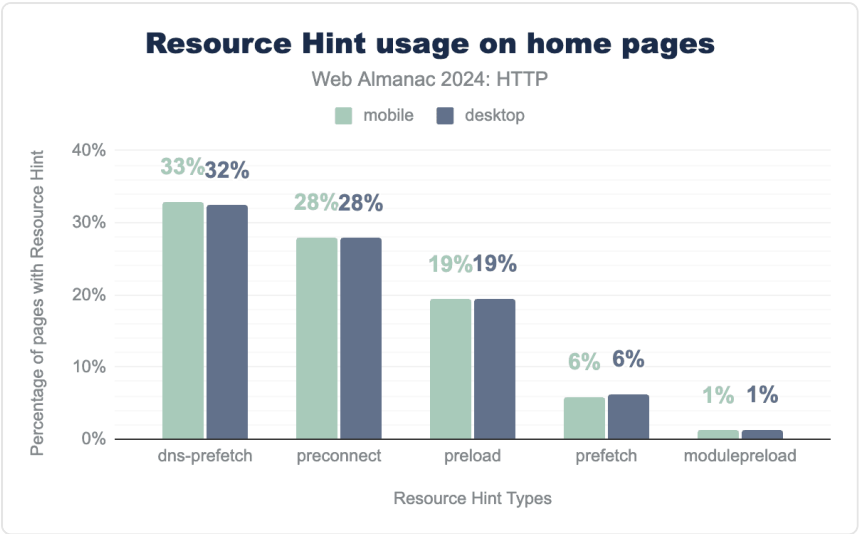


Figure 18.13. Resource Hint usage is quite high in our dataset, especially for `dns-prefetch` and `preconnect`.

These directives have found an impressive uptake over the years, with over 30% of all pages in our dataset using `dns-prefetch`, 28% utilizing `preconnect` and `preload` closing in on 20%.

Personally, I find the high usage of `dns-prefetch` somewhat surprising, as in my opinion you should almost always use `preconnect` instead, as it does more (establishing the full connection including TCP+TLS or QUIC handshakes) with negligible overhead, and is also very well supported⁶⁸⁶. Furthermore (though purely anecdotally), I often see developers use both `dns-prefetch` and `preconnect` right after each other for the same domain, which is almost completely useless nowadays, as `preconnect` automatically includes a DNS lookup as well and is well supported as noted. Still, an overall high use of these directives is good for performance, as it helps hide connection setup latency to subdomains or third-party domains that might host important assets, such as images or fonts.

Some (at least to me) unexpected yet good news is that very few pages seem to overuse these hints. In general, you should rely on the browser to get things right itself—through features like the preload scanner⁶⁸⁷. You should only use the Resource Hints sparingly for those cases where you know the browser doesn't have enough information—for example, when the 3rd party domain/resource is not mentioned in the HTML directly but only in a CSS/JS file. And this is exactly what most people have been doing (yay!): even at the 90th percentile, pages will only

686. <https://caniuse.com/link-rel-preconnect>

687. <https://web.dev/articles/preload-scanner>

use 3 `dns-prefetch`, 2 `preconnect` and 2 `preload` directives, which is great!

<4%

Figure 18.14. The percentage of pages that preload 1 or 2 resources without actually using them.

Specifically for `preload`, we also looked at how many resources are preloaded in vain and not actually used on the page. The results were again quite encouraging: less than 4% of desktop pages uselessly preload 1 or 2 resources, and less than 2% have 3 or more unused preloads!

Still, even though most pages seem to get it right, it's always fun to look at some of the worst offenders. There will always be pages which include way too many hints, usually either due to misconfiguration or a misunderstanding of what the features are supposed to do.

For example, one page had a whopping 3215 preloads! Looking more closely however, it was clear that they are preloading the exact same image over and over again, most likely due to a misconfiguration of their framework/bug in their code.

The second worst offender took it easy with “just” 2583 preloads, all of different versions/subsets of various asian fonts from Google Fonts. Finally, one page preloads an amazing 1259 images, turning them into a “smooth” scrolling background animation; arguably, you could say here preload is actually used in a good way to improve the intended effect, though I wouldn't recommend it in general!

Luckily as well, some of the worst problem cases in our dataset have been fixed since our June 2024 crawl, such as a “Sexy Pirate Poker” site going from 2095 to just 14 preloads (steady as she goes, mateys!).

A final silver lining is that no pages had more than 1,000 `dns-prefetch` or `preconnect` directives. The worst ones utilized a measly 590 and 441 domains respectively.

In general, Resource Hints remain powerful features that should be used sparingly to great effect, something which most developers seem to understand. Let's now take a deeper look at `preload` specifically though, as it is one of the more powerful hints that is widely supported⁶⁸⁸, and it can have a large impact on performance, both positively and negatively (if used incorrectly).

⁶⁸⁸. <https://caniuse.com/link-rel-preload>

Preload

Generally speaking, `preload` should mainly be used to inform the browser of resources that are not linked directly in the main HTML document but that you as a developer know will be important/needed later—such as things loaded dynamically with JS `fetch()` or CSS `@import` and `url()`. Preloading them allows the browser to request them earlier from the server, which may improve performance—but can also degrade it, if you try to preload too much.

Good concrete examples include fonts (which are typically loaded via CSS *and* only requested by the browser when it actually needs them to render text), JS submodules or components imported dynamically, and (Largest Contentful Paint, LCP) images that are loaded via CSS or JS (which you probably shouldn't do in the first place, but sometimes there's no other option).

As we saw above, about 20% of all pages utilize `preload`, and because you have to explicitly indicate which type of resource you're trying to preload with the `as` attribute, we can get some more information on how exactly sites are using it.

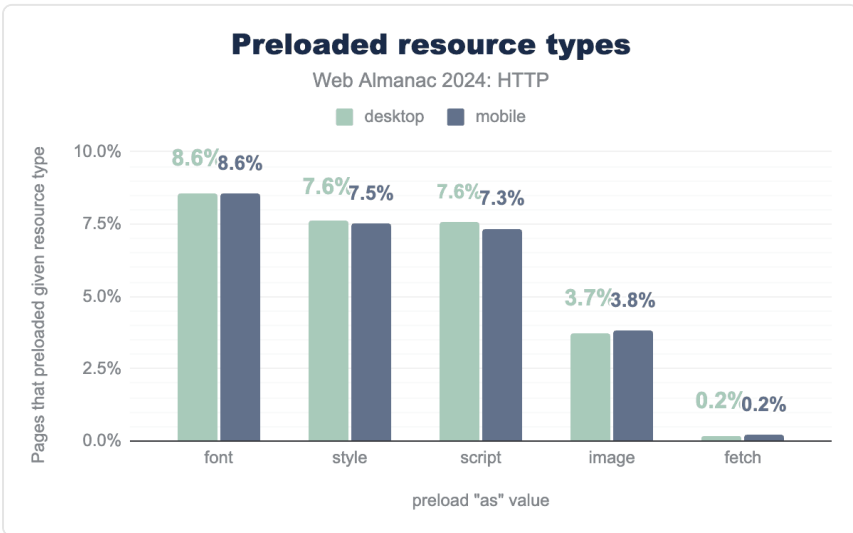


Figure 18.15. Preload is typically used for fonts, stylesheets and scripts.

As expected, fonts are the most popular target for preloads, occurring on about 8.6% of pages.

Somewhat unexpected however was the high use of preload for stylesheets and scripts. These *could* conceptually be useful, but that would mean over 7.5% of pages should probably reduce

their critical path depth⁶⁸⁹, if they need preload to speed that up. In practice people often misuse the feature by preloading resources right before they're mentioned in the HTML; say by having a `<link rel=preload>` for `style.css` the line above the `<link rel=stylesheet>` for the same file, which doesn't really do anything.

A worse problem I've seen is that people `preload` their `async` and `defer` JS files, often those that are loaded via a `<script>` tag at the bottom of the `<body>`. This is not as harmless as it might seem, as these preloads can end up actively delaying other resources—such as actual render blocking JS and important images. This is because the browser doesn't know these scripts will be tagged as `async` / `defer` when they're actually loaded, so it defaults to loading them as if they're render blocking scripts with a high priority!⁶⁹⁰ We didn't run the queries to see how many of the 7.5% of pages misuse `preload` like this, but from my personal experience, it could be pretty widespread.

In contrast, while perhaps too many `async` / `defer` JS scripts are being preloaded, potentially too few JavaScript modules (`<script type=module>`) are benefiting from this hint. While about 9.6% of all desktop pages already use JS modules (which I found impressively high), only about 13% of those (1.24% of all pages) also preload at least one module. I would have expected this to be a bit higher, since the JS modules mechanism has extensive support for dynamically loading code⁶⁹¹, which could benefit from preloads to improve performance.

Potentially, this is because—due to annoying CORS-related reasons⁶⁹²—you actually need a special flavor of preload specifically for JS modules, (predictably) called `modulepreload`. It could be that too few people are aware of this special type to use it, or just that JS modules are too new to have been fully figured out, or because people aren't really using dynamic imports or deep import trees in practice; future Web Almanac analysis will have to tell.

79%

Figure 18.16. The percentage of desktop pages that has an external source (such as an image) as their LCP element.

A different, more positive note is that fewer than 4% of pages preload an image, which I had expected to be a lot higher, as 79% (desktop) to 69% (mobile) of pages have an LCP element that's an external source URL such as an image.

Similar to CSS, it is usually pretty useless⁶⁹³ to preload an image that's already linked to in the

689. <https://www.debugbear.com/blog/avoid-chaining-critical-requests>

690. <https://youtu.be/MV034VqHv5Q?t=838>

691. https://developer.mozilla.org/docs/Web/JavaScript/Guide/Modules#dynamic_module_loading

692. <https://web.dev/articles/modulepreload>

693. <https://youtu.be/p0IFyPufH8Zs?t=2038>

HTML directly (for example as an `img` or `picture` tag), as the browser typically discovers the image early enough even without preload. Luckily, it seems that developers are not making this mistake often: in our dataset, of all the pages that have an `<img>` as their LCP element (46% on desktop, 41% on mobile), less than 1% actually preload that image (0.7% desktop, 0.9% mobile).

In contrast, there are images that *can* benefit from preloading, for example those LCPs loaded via CSS as background images, as those are typically only discovered late. In our dataset, 27% of desktop pages and 25% of mobile pages have a `<div>` as their LCP element, which often means it has a CSS background image. Out of those cases, 2.3% (desktop) and 2% (mobile) actually preload the LCP resource url, more than double than was the case for `<img>`! While that's a good thing, in my opinion people are actually potentially *underutilizing* preload for this use case—though note that you should really only preload the LCP background image, not others!

The astute reader might have noticed that the total amount of pages preloading images (~3.8%) is quite a bit higher than those preloading the LCP element (~1.3% total), which makes me wonder which other images people are preloading then and why...; another thing best left for future analysis!

Finally, as with the general Resource Hints, it's also interesting to look at outliers and obvious mistakes. For `preload` to function, you **MUST** set the `as` attribute, and it can only be set to a select few types⁶⁹⁴: `fetch`, `font`, `image`, `script`, `style`, `track`. Anything else will cause the preload to be in vain. As such, it's interesting to see over 17,000 pages (about 0.11% overall) use an empty value, with 0.03% - 0.01% utilizing invalid (though probable) values like `stylesheet`, `document`, or `video`. Other noteworthy (luckily much less frequent) values include: the fancy `Cormorant Garamond Bold`, the cool `slick`, the spicy `habanero` and the supercalifragilisticexpialidocious `Poppins`.

103 Early Hints

Resource Hints are typically conveyed through `<link rel=XYZ>` tags inside the page HTML's `<head>`. However, though you may not know this, Resource Hints can also be sent in the HTTP response headers for the HTML page instead (albeit with a slightly different syntax⁶⁹⁵). In fact, I'm pretty sure most of you don't know this, since only about 0.04% of all pages (or about 5,500 desktop home pages) utilizes this option—compared to about 20% that use the HTML tags. This is not too surprising though, as there are only very few cases in which the HTTP header option is easier or better than the HTML tag option in my opinion.

694. <https://developer.mozilla.org/docs/Web/HTML/Attributes/rel/preload>

695. <https://olmanac.httparchive.org/en/2021/resource-hints#http-header>

However, this might be changing a bit, with the (relatively) new 103 Early Hints⁶⁹⁶ feature. This mechanism is sometimes called the rightful successor to HTTP Server Push⁶⁹⁷. It allows a server to send back an intermediate 103 response (part of the HTTP 1XX range of status codes) before the actual final response for a request (say a 200 OK or 404 Not Found).

This is especially useful in CDN setups where the HTML is not cached at the CDN edge. In that scenario, the CDN can very quickly send back a 103 Early Hints response to the browser, while it forwards the request for the HTML to the origin server. This 103 response can contain a list of `preconnect` and `preload` Resource Hints (encoded as HTTP response headers), which the browser can start executing while it is still waiting for the final HTML response to come in.

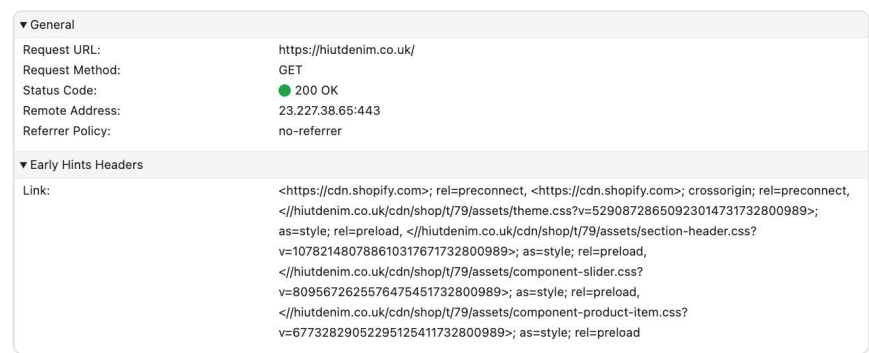


Figure 18.17. 103 Early Hints example.

If everything goes well, the connections to external domains are ready and the preloaded resources are in the browser's cache by the time the HTML comes in, providing an impressive performance boost!

2.9%

Figure 18.18. The percentage of desktop pages using 103 Early Hints.

Disappointingly though, 103 Early Hints adoption has not increased a lot since our first look in 2022⁶⁹⁸: from 1.6% of all desktop pages then, to just 2.9% this year. This is not incredibly surprising however, since properly configuring this feature is not easy, and getting full benefits from it typically requires using a CDN or a similarly distributed deployment.

Support has also been somewhat spotty, with Safari and Firefox only adding support recently

696. <https://developer.mozilla.org/docs/Web/HTTP/Status/103>
697. <https://almanac.httparchive.org/en/2022/http#103-early-hints>
698. <https://almanac.httparchive.org/en/2022/http#103-early-hints>

(with Safari only allowing `preconnect`), it having been disabled for a while on Cloudflare⁶⁹⁹, and the Akamai CDN only making it generally available in July⁷⁰⁰ this year. Still, I would have expected the uptake to be a bit higher, given that Cloudflare has had the feature available even for its free customers since 2022.

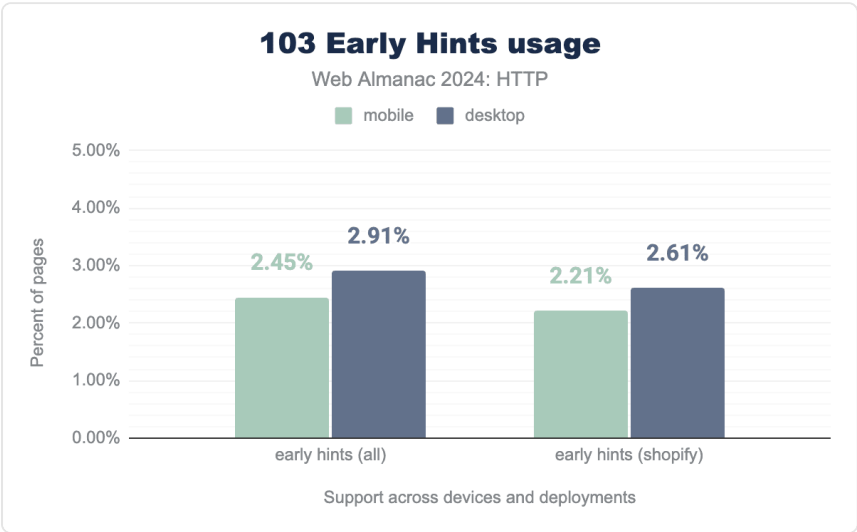


Figure 18.19. Out of 2.9% of all desktop pages that utilize 103 Early Hints, Shopify accounts for 2.6% of them.

Staggeringly, we see that an overwhelming majority of Early Hints comes from just one deployment: Shopify (a Cloudflare customer) accounts for 90% those 2.9% of desktop pages that support Early Hints. This means that only about 39,000 non-Shopify desktop home pages employed the new feature. As such, it's difficult and not very useful to draw broad conclusions about its real potential based on what we see in the Web Almanac's dataset.

Still, there are a few interesting trends (mostly from Shopify's deployment), including that the amount of preconnects is pretty much stable from the 10th up to the 90th percentile (usually preconnecting to `https://cdn.shopify.com` twice, once with and once without `crossorigin`⁷⁰¹). Preload on the other hand is lower at 1 at p75, increasing to 2 at p90.

Interestingly, stylesheets are the most preloaded resource type in Early Hints, about 4x as much as scripts. While fonts were the most popular for preloads in the HTML, here they are the least popular, coming even behind images. This I find quite weird, since in my opinion fonts are still excellent targets for preloading even in Early Hints, while images should probably be

699. <https://community.cloudflare.com/t/early-hints-and-encrypted-client-hello-ech-are-currently-disabled-globally/567730>

700. <https://www.akamai.com/blog/performance/akamai-103-early-hints-prototype-the-results-are-in>

701. <https://csswizardry.com/2023/12/correctly-configure-preconnections/>

avoided, as preloads in Early Hints currently don't support responsive images⁷⁰² (preloads in the HTML do of course⁷⁰³, don't worry)!

In conclusion, while 103 Early Hints is still somewhat absent in our dataset, I feel this will improve over time, as more deployments support it, as it becomes easier to configure—for example, with automated Early Hints⁷⁰⁴—and as more people become aware of its potential!

The Fetch Priority API

While the Resource Hints discussed above can influence when a connection is opened or when a resource is requested, they don't really say much about what happens after that: How is the connection used? How are the resources downloaded? That is the purview of other features, among them the Fetch Priority API⁷⁰⁵ (previously called “Priority Hints”), which helps control how resources are scheduled on HTTP/2 and HTTP/3 connections.

One of the main reasons to switch from HTTP/1.1 to HTTP/2 or HTTP/3 is that you need fewer connections. On the newer protocols, many resources can be “multiplexed” onto a single connection—requested and loaded concurrently—while on HTTP/1.1 you have to open multiple parallel connections to get a similar effect. As each connection has a certain overhead associated with it (TCP+TLS/QUIC handshakes, some memory at the server, competing congestion controllers, ...), setting up and maintaining fewer connections is more efficient.

702. <https://developer.chrome.com/docs/web-platform/early-hints#current-limitations>

703. <https://web.dev/articles/preload-responsive-images>

704. <https://blog.cloudflare.com/smart-hints/>

705. <https://web.dev/articles/fetch-priority>

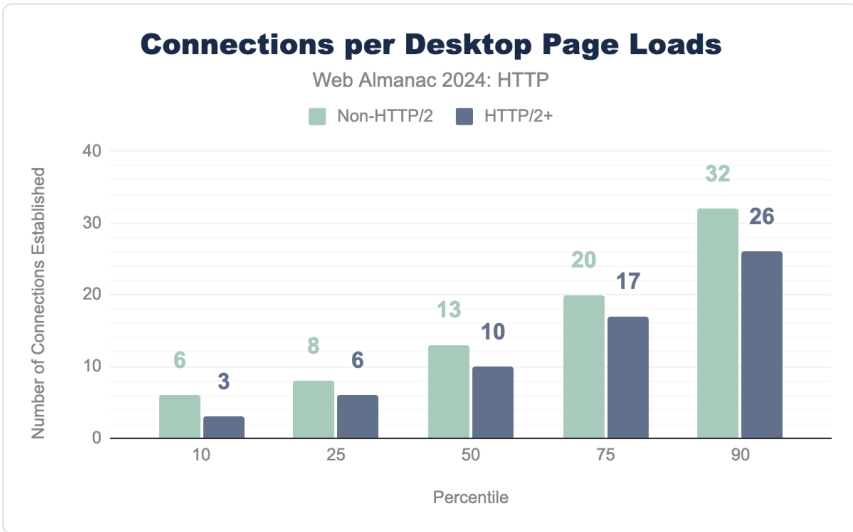


Figure 18.20. HTTP/2 and HTTP/3 generally use fewer connections per page load than HTTP/1.1

The differences here are a bit less clear than a few years ago though; in 2021, there was a 4 connection difference at p50 and p75 between the protocol versions, now this has shrunk to 3—potentially due to aspects like HTTP cache partitioning, crossorigin woes or perhaps more 3rd parties being used. It is still clear however that the new protocols indeed reduce the connection count and improve efficiency and at p10, only half as many connections are needed!

However, now that we have more resources sharing a single underlying connection, that means that we somehow need to decide what gets downloaded first as we typically don't have enough bandwidth to just download everything in one big parallel flow. This “resource scheduling” is governed by a “prioritization mechanism” in HTTP/2 and HTTP/3. How this really works under the hood is a bit too complex to really dig into here however, so we will focus on the basics you need to understand the Fetch Priority API (more details can be found in blogposts⁷⁰⁶, talks⁷⁰⁷, academic papers⁷⁰⁸ and of course, web.dev⁷⁰⁹).

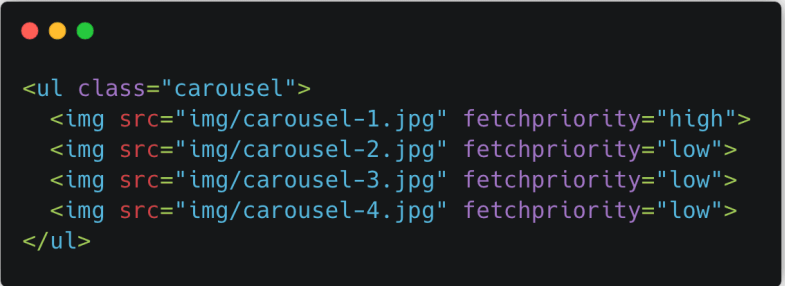
In general, the browser will assign each request a *priority*: an indication of how important it is to the page load. For example, the HTML document and render-blocking CSS in the `<head>` might get `highest` priority, while less critical resources (such as images in the `<body>` or JS tagged as `async` or `defer`) might get `low`. When the server then receives multiple requests from the browser in parallel, it knows in which order to reply: from highest to lowest priority, and following the request order for resources with the same priority value.

706. <https://calendar.perfplanet.com/2022/http-3-prioritization-demystified/>

707. <https://www.youtube.com/watch?v=MV034VqHv5Q>

708. https://herbots.info/public_media/research/anrw2024_h3-eps-in-the-wild_authorversion.pdf

709. <https://web.dev/articles/fetch-priority#effects>



```
<ul class="carousel">
  
  
  
  
</ul>
```

Figure 18.21. Fetch Priority API example to improve image loading behaviour in a carousel component.

Browsers use a complex set of heuristics (“educated guesses”) to determine the priorities of the resources—based on factors like their position in the HTML document, their type, and loading modifiers such as `async` / `defer`. This however also sometimes means the browser gets it wrong, or it simply does not have enough information to make a smarter choice.

A good example here are LCP images: the browser can’t really accurately predict which image will end up being the LCP element just from the HTML. As such, it generally requests all images at the same priority, in discovery order; so if your LCP image is lower in the HTML (below say some images in the menu that’s hidden by default) it will end up loaded later than it probably should.

It is for these reasons that we now have the Fetch Priority API! It lets us tweak/nudge the browser’s default heuristics so it assigns high(er) or low(er) priority values to individual resources—meaning they get loaded earlier/later than they otherwise would. This is done by adding the `fetchpriority` attribute with a value of either `high` or `low` to a resource.

It can be used on many things, not just images but also `<script>` and `<link>` tags and even `fetch()` calls (there, it’s just the `priority` attribute though, because naming things is hard). The Fetch Priority API has been supported in Chrome for a while now, with Safari adding support late last year, and Firefox landing the feature in October 2024. As such, let’s see how it’s being used in the wild!

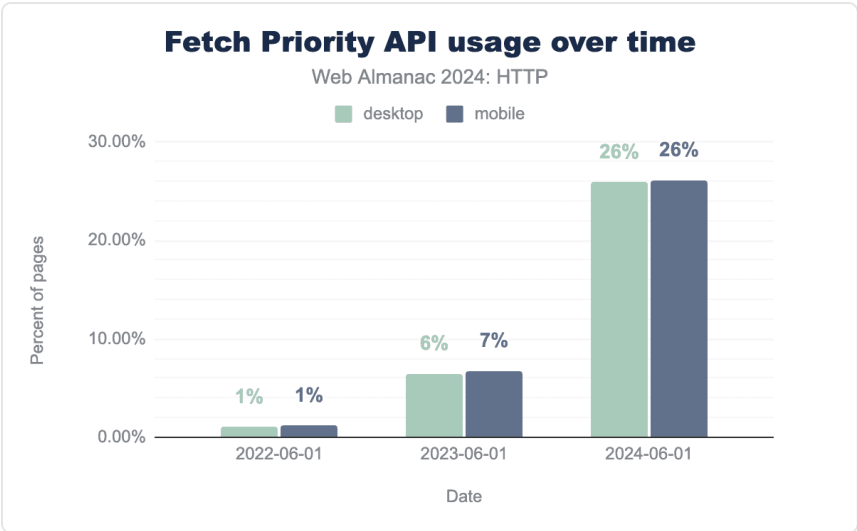


Figure 18.22. Fetch Priority API adoption has soared in 2024

Fetch Priority adoption has positively soared in 2024! Going from around 7% in 2023 to 26% of pages in 2024 using at least one `fetchpriority` attribute is truly a stunning result. Many pages even use more than 1 instance, with the 90th percentile showing 2 uses per page, up to 9 for desktop pages at the 95th (just like with Resource Hints there are of course also people overdoing it, with one page using it an incredible 2382 times!).

5.6%

Figure 18.23. The percentage of desktop pages having an `<img>`-tag with `fetchpriority=high` as their LCP element

Let's look at how `fetchpriority` is being used specifically for LCP images since that is one of the main motivating use cases for this new feature. For example, for those 46% of desktop pages that have an `<img>` element as their LCP, a good 12% (or 5.6% of all desktop pages) also tag it as `fetchpriority=high`; This could/should arguably be (much) higher, but luckily only 0.14% tagged the LCP `<img>` as `fetchpriority=low`, so I'll take it!

Sadly, this is offset by the fact that over 16% of desktop pages does still lazy load their LCP image⁷¹⁰, which you should really never do... luckily, only 4,500 pages were written by cats who don't know whether they want to stay in or go outside (and thus have an LCP element with both

710. <https://web.dev/articles/browser-level-image-lazy-loading>

`loading=lazy` AND `fetchpriority=high`, which is just weird).

We also looked into which initial priority LCP images were requested with in general. By default, images in Chrome have `low` priority, unless they're one of the first 5 images in the HTML, in which case they get `medium` priority. Images can get promoted to `high` priority if the browser determines early on they're in the viewport or, as you'd expect, if `fetchpriority=high` is used.

Again looking at all the desktop pages having an `<img>` element as their LCP, we see that 42% starts at `low` (so they're not even in the top 5!), 45% are `medium` (a bit better) and just 12% get requested as `high` (almost entirely due to `fetchpriority=high`). As such, despite the impressive and rapid uptake of fetch priority in the past 2 years, there is still a large window of opportunity to do some "low hanging fruit" optimizations for many pages by tagging their LCP as `fetchpriority=high`—after first removing `loading=lazy`, of course!

Preload again

The `fetchpriority` attribute can also be used on `<link>` tags. That's not only useful for CSS stylesheets, but also for (you guessed it!) `<link rel=preload>`. So even though we've already spent a lot of time on preload above, let's revisit it here for some added nuance, as fetch priority really makes a difference here too.

Conceptually, preload by itself doesn't/shouldn't change a resource's priority, only when it's discovered and therefore requested by the browser. This is true for images for example: even if you preload your LCP, it will still be `low` priority, unless you also add `fetchpriority=high` to your `preload`, which is often unexpected for people!

In other cases, preload DOES seem to "change" the priority. For example, the case when preloading `async` / `defer` JS (as discussed above as well), where the browser will assign them `high` priority instead of `low` because it doesn't get enough context from the `preload`. In these cases, it can be very useful to use `fetchpriority=low` on the JS preload, to "correct" the browser heuristics to where we know they should be.

73%

Figure 18.24. Percentage of preloads that have `fetchpriority=high` set that are for images

We looked at how people are combining `preload` with `fetchpriority` in our dataset. Of all desktop pages with preloads using `fetchpriority` (about 2% of all desktop pages), an impressive 73% are for images with `fetchpriority=high`. While this can indeed be a good

idea for LCP images, it does have some rough edges and can be a footgun if used incorrectly⁷¹¹ at the top of the `<head>`, actually delaying JS lower down the document. For this reason, nowadays I even recommend just not preloading the LCP in favor of just having it in the HTML with `fetchpriority=high` on the `<img>` directly.

On the other end, 16% of these preloads are for scripts with `fetchpriority=low`, indicating at least some webmasters (what is this, 2005?!) are aware of potential issues with `async` / `defer` there and try to prevent them. For styles, people don't really seem to know what they want (or use cases are diverse), as 3% is loaded as `high` and 5% as `low`. Note that a lot of these nuances are also discussed on web.dev⁷¹², so make sure to read up on things there.

Finally, an interesting 0.06% of pages tries to preload things with a `highest` value for `fetchpriority`, which is not supported (you can only use `high` or `low`)!

Conclusion

In summary, despite HTTP being invented early in the 1990s, its third version is still making waves on the Internet, finding a steadily increasing adoption that should reach 30% soon. This is aided by the introduction of some new capabilities, such as DNS HTTPS records, which make discovering and using HTTP/3—and other newer protocol features—faster and easier.

While the newer protocol versions are typically presented as a black box to developers (indeed, we can't even consciously choose to use HTTP/3 in `fetch()`), some high-level features exist that allow tweaking the underlying behaviours. For example, Resource Hints have become more powerful now they can be used inside 103 Early Hints responses, allowing browsers to preconnect and preload even before the HTML is known. Complementary, the Fetch Priority API can help improve browser heuristics that decide in which order resources are downloaded from a server on HTTP/2's and HTTP/3's heavily multiplexed connections. Developers have found their way to some of these features quite easily (with especially Fetch Priority rising to a 25%+ usage share in a mere 2 years), while remaining hesitant on some others (at less than 3% usage share, 103 Early Hints seems difficult to use or just unknown to many).

Still, there remain challenges ahead. CDNs are the outsized driving force between the fast adoption of many of these new technologies (85% of all HTTP/3 traffic was served through a CDN). This is both “good” and “bad” for the ecosystem in my opinion. Good because they battle-test new technologies quickly and give them a huge market share right out of the gate, ensuring good chances for survival. Bad because this causes the web to become ever more centralized around a few large companies (54% of all requests in our dataset were served from a CDN), with people not using them being at risk of being left behind.

711. <https://youtu.be/pOIFyPuH8Zs?t=2135>

712. <https://web.dev/articles/fetch-priority?hl=en#use-cases>

This goes hand in hand with the increasing complexity of how the web works at the lower layers. Protocols like HTTP/3 are complex to understand, let alone deploy, but even the “simpler” high-level features can be difficult to really apply correctly in practice (the high amount of preloaded scripts is somewhat concerning). There is plenty of potential for misuse and shooting yourself in the foot and not everything is as well documented as it could be (the 16% of pages lazy loading their LCP image is a testament to that).

Still, I see a clear silver lining here. While it is my job to look for the mistakes people make (so I can help fix them!), I was pleasantly surprised to see that most of the obvious mistakes are in fact NOT as widespread as they sometimes seemed from personal experience. Let’s hope it remains that way as these newer features find even further adoption on the wider web. Hopefully, as developers become more familiar with the underlying features of the network protocols, some of the inherent complexity stops being a barrier to adoption, and even HTTP/3 can move beyond its CDN roots. Let’s work together to make that happen!

Author



Robin Marx

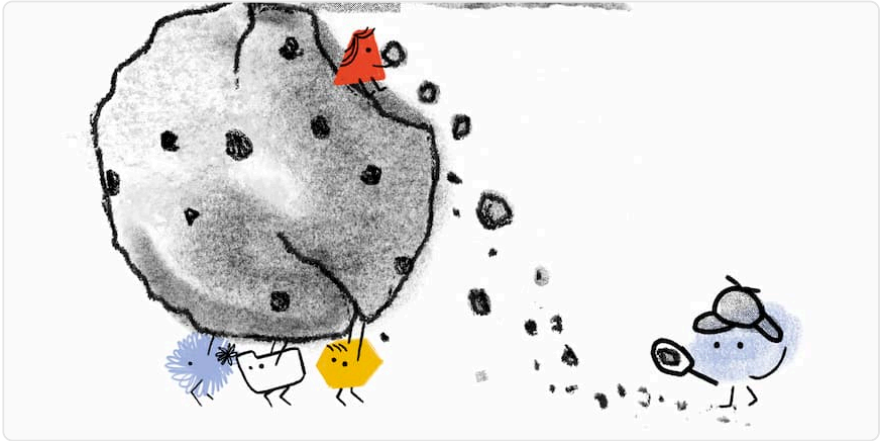
✕ @programmingart 🐦 @programmingart.bsky.social 🎧 rmarx 📺 rmarx

🌐 <http://internetonmars.org>

Dr. Robin Marx is a web performance expert at Akamai Technologies. He studies the performance and operation of modern web protocols such as HTTP/2, HTTP/3, and QUIC. Robin often speaks about web performance at international conferences, making complex situations more understandable for the general public.

Part IV Chapter 19

Cookies



Written by Yohan Beugin, Sam Dutton, and Yana Dimova

Reviewed by Sam Dutton and Rowan Merewood

Analyzed by Yohan Beugin

Edited by Barry Pollard

Introduction

The following chapter of the Web Almanac 2024 is focused on cookies. Cookies have multiple functionalities and are to some extent essential for the web—for example, for authentication, fraud prevention and security. However, some cookies can track users across websites and are utilized to build behavior profiles.

In this chapter, we measure the prevalence and structure of web cookies encountered while visiting mainly the top one million websites during the HTTP Archive crawl of June 2024.

Additionally, we discuss and measure the adoption of alternative mechanisms to third-party cookies that were introduced by Google on Chrome as part of the Privacy Sandbox⁷¹³ initiative to reduce cross-site tracking.

We find that 61% of cookies are set in a third-party context. Generally, third-party cookies can

713. <https://privacysandbox.com/>

be used for online tracking and targeted advertising. For this reason, Google proposed to phase out all third-party cookies and introduce more privacy-friendly options to replace their functionality with the Privacy Sandbox.

On the other hand, not all third-party cookies are used for online tracking. Browsers such as Chrome include a number of ways to limit the way that third-party cookies are used. For example, cookies that are partitioned (CHIPS) cannot be accessed across different top-level sites from the one the cookies are set on originally, which makes it impossible to track users across websites. Nonetheless, we find that the most prevalent partitioned cookies are set by domains related to advertising. Another example is the `SameSite` cookies attribute, which ensures that (first-party) cookies are not included in cross-site requests by default. Trackers can disable this setting by explicitly setting the value of the `SameSite` attribute to `None`. Therefore, in practice, we find that for 11% of observed first-party cookies, `SameSite` is set to `None`. Additionally, we observe that the most widely set third-party cookies are used for advertising and analytics, with Google being prevalent on the largest percentage of websites.

First-party cookies can also be used to track recurring users. From our analysis, we conclude that the most prevalent first-party cookies are used for analytics. In theory, because of the same-origin policy, these cookies cannot be used for cross-site tracking. However, by using advanced tracking methods such as cookie syncing and CNAME tracking, trackers can bypass this limitation. We refer to the Privacy chapter for more details on online tracking methods.

Our results indicate both first-party and third-party tracking are common. We show that online tracking by means of cookies is still predominant on the web.

Definitions

First up let's get a common understanding of some of the terms used in this chapter.

HTTP cookie

When a user visits a website, they interact with a web server that can request the user's web browser to set and save an HTTP cookie⁷¹⁴. This cookie corresponds to data saved in a text string on the user's device, and is sent with subsequent HTTP requests to the web server. Cookies are used to persist stateful information about users across multiple HTTP requests, which can allow authentication, session management, and tracking. Cookies are also associated with privacy and security risks.

714. <https://developer.mozilla.org/docs/Web/HTTP/Cookies>

First and third-party cookies

Cookies are set by a web server and there are two types of cookies: **first-party** and **third-party** cookies. First-party cookies are set by the same domain as the site the user is visiting, while third-party cookies are set from a different domain.

Third-party cookies may be from a third party, or from a different site or service belonging to the same “first party” as the top-level site. **Third-party cookies** are really **cross-site cookies**.

For example, imagine that the owner of the domain `example.com` also owns `example.net` and that the following cookies are set for a user visiting `https://www.example.com`:

Cookie Name	Set by	Type of cookie	Reason
cookie_a	www.example.com	First-party	Same domain as visited website
cookie_b	cart.example.com	First-party	Same domain as visited website: subdomains do not matter
cookie_c	www.example.edu	Third-party	Different domain than visited website
cookie_d	tracking.example.org	Third-party	Different domain than visited website
cookie_e	login.example.net	Third-party	Different domain than visited website even if owned by the same owner in this example (cross-site cookie from the same “first party” at the top-level site)

Figure 19.1. Cookie Context.

Privacy & security risks

Web tracking. Cookies are used by third parties to track users across websites and record their browsing behavior and interests. In targeted advertising, this data is leveraged to show users advertisements aligned with their interest. This tracking usually takes place the following way; third-party code embedded on a site can set a cookie that identifies a user. Then, the same third-party can record user activity by obtaining that cookie back when the user visits other websites where it is embedded as well (see also the Privacy chapter). We note that first-party cookies can also be used for online tracking, methods such as cookie syncing allow to bypass

the limitation of third-party cookies and track users across different websites⁷¹⁵.

Cookie theft and session hijacking. Cookies are used to store session information such as credentials (session token) for authentication purposes across several HTTP requests. However, if these cookies were to be obtained by a malicious actor they could use them to authenticate to the corresponding web servers. If cookies are not properly set by web servers, they could be prone to cross-site vulnerabilities such as session hijacking⁷¹⁶, cross-site request forgery (CSRF⁷¹⁷), cross-site script inclusion (XSS⁷¹⁸), and others (see also the Security chapter).

Caveats

You can learn more about the methodology applied by the HTTP Archive for the Web Almanac in 2024 on the Methodology page. There are limitations to that methodology which may impact the results in this chapter:

- Data is collected by automatically visiting websites in a non-interactive way; user interaction could modify the way websites set and use cookies in practice. For example, HTTP Archive's tools do not interact with cookie banners (if any) and so cookies that would be set after interaction with these banners are not observed by our study.
- Websites are visited from servers located in the US that have no cookie set when each independent website visit starts; this is quite different from a user accumulating and saving web cookies while browsing the web. The location from which visits are performed can impact cookie behavior due to regulation and legislation such as GDPR⁷¹⁹.
- For each website, the home page is visited as well as one other page from the same website.
- Most of the results presented in this chapter are based on the top one million most visited websites according to the Chrome User Experience Report (CrUX)⁷²⁰ that were successfully reached during the HTTP Archive crawl of June 2024.
- The cookies collected for the analysis in this chapter were obtained at the end of the visit of each website page by extracting all cookies stored by the web browser in its cookie jar. As a result, the collected data only contains cookies that are deemed valid by the web browser and successfully set. Thus, if websites attempt to set

715. <https://dl.acm.org/doi/abs/10.1145/3442381.3449837>

716. https://developer.mozilla.org/docs/Glossary/Session_Hijacking

717. https://developer.mozilla.org/docs/Web/Security/Practical_implementation_guides/CSRF_prevention

718. https://developer.mozilla.org/docs/Glossary/Cross-site_scripting

719. <https://gdpr-info.eu/>

720. <https://developer.chrome.com/docs/crux>

invalid cookies (too large, attributes mismatch, etc.) they would be missing from our analysis.

Notes

The figures plotted in this chapter indicate in their subtitle (a) the type of client device (**desktop** or **mobile**) that was used to access the websites for the plotted data and (b) the top number of websites visited (according to their CrUX rank⁷²¹). If the information is not specified, it must be on one of the axes of the graph.

Prevalence and structure of cookies

In this section, we report on the prevalence of cookies, their type, and their attributes on the web.

First and third-party prevalence

First-party cookies are set by the same domain as the website that the user is visiting, while third-party cookies are set by a different domain see Definitions. In this analysis, we examine the percentage of cookies set on websites that are first- and third-party across clients (desktop or mobile) and CrUX ranks.

721. <https://developer.chrome.com/blog/crux-rank-magnitude>

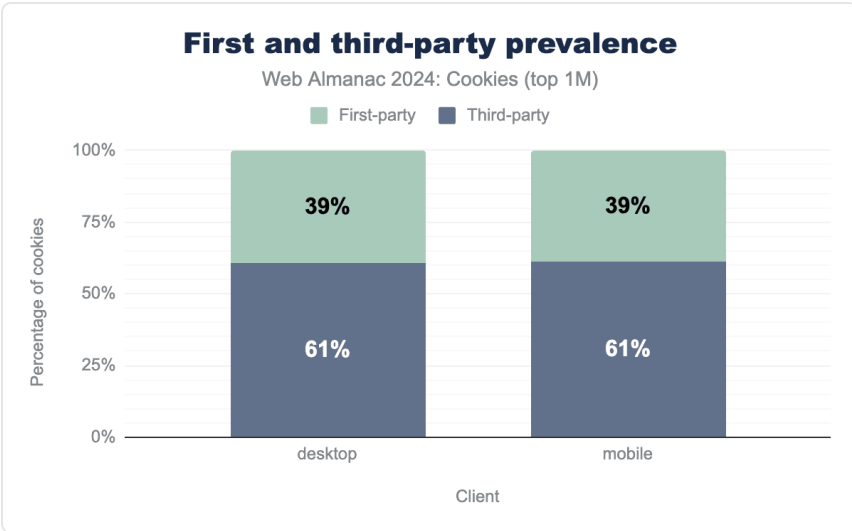


Figure 19.2. First- and third-party prevalence.

On the top one million most visited websites, about 39% of the cookies are first-party and 61% are third-party cookies. Thus, a majority of the cookies set on the web are third-party cookies. We also observe that this distribution is very similar whether these websites are accessed through a desktop or a mobile client. This indicates that overall there is little to no behavior change based on the type of client used. However, some websites may still behave differently and/or use other tracking methods such as fingerprinting depending on the type of client (see the Privacy chapter for more).

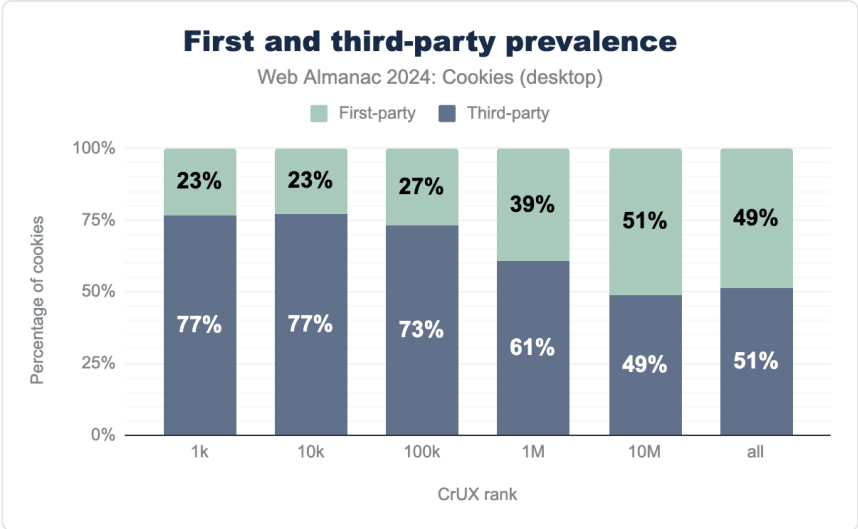


Figure 19.3. First- and third-party prevalence of cookies by rank on desktop clients.

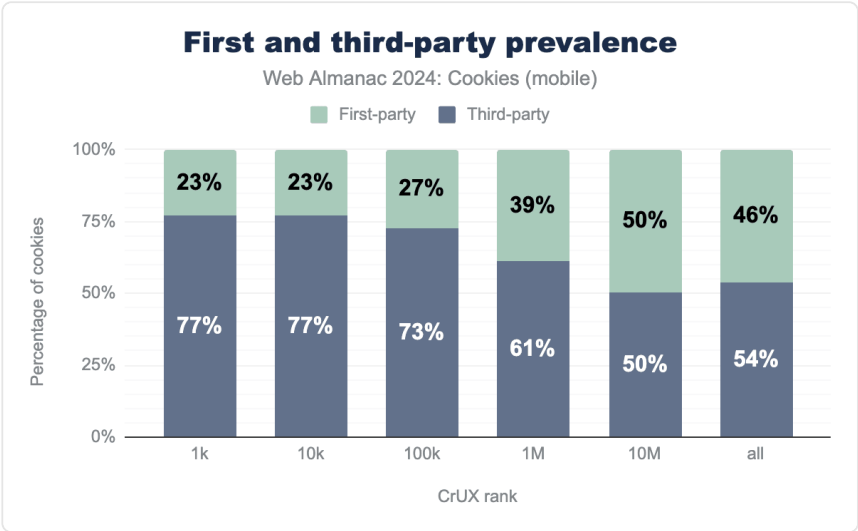


Figure 19.4. First- and third-party prevalence of cookies by rank on mobile clients.

Looking at the prevalence of the type of cookies across website rankings, we observe that more popular websites have a higher proportion of third-party cookies than the ones visited less often. For instance, in comparison to the results reported on the top one million websites, 23% and 77% of the cookies are first and third-party on the top one thousand (top one thousand) websites, respectively. This is likely due to the fact that websites that are the most visited by

users embed more third-party code (that in turn sets more third-party cookies) than less visited ones. Additionally, the prevalence of each cookie type across the ranks is quite similar between desktop and mobile clients; we observe that previous remarks made on the top one million websites also hold across CrUX ranks.

Cookie attributes

Next, we discuss the distribution of different cookie attributes⁷²². Furthermore, we zoom into the use of the `SameSite` cookie attribute. The following two figures show the proportion of first and third-party cookies set on the top one million websites for each client that have one of the following attributes set: `Partitioned`, `Session`, `HttpOnly`, `Secure`, `SameSite`. Before diving into more details for each attribute, let's observe here again the similarity of the distribution of the different attributes between desktop or mobile clients.

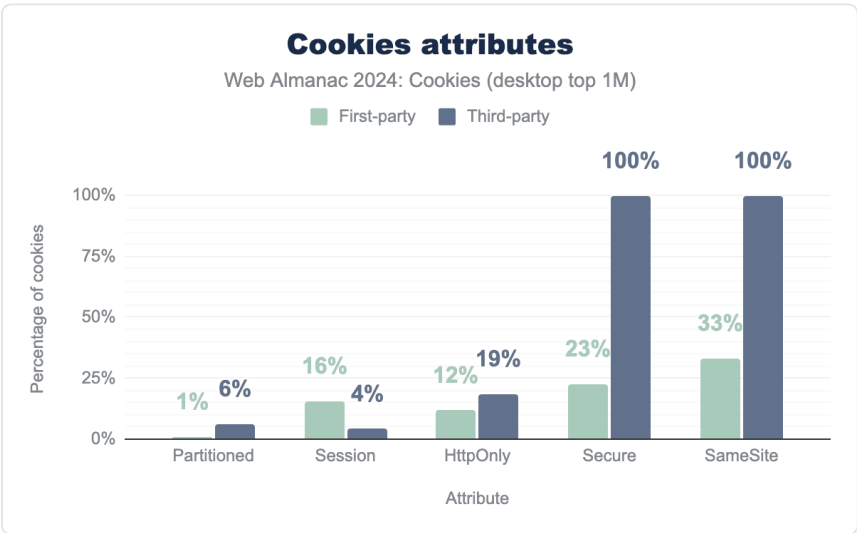


Figure 19.5. An overview of cookie attributes for desktop clients.

722. <https://developer.mozilla.org/docs/Web/HTTP/Headers/Set-Cookie>

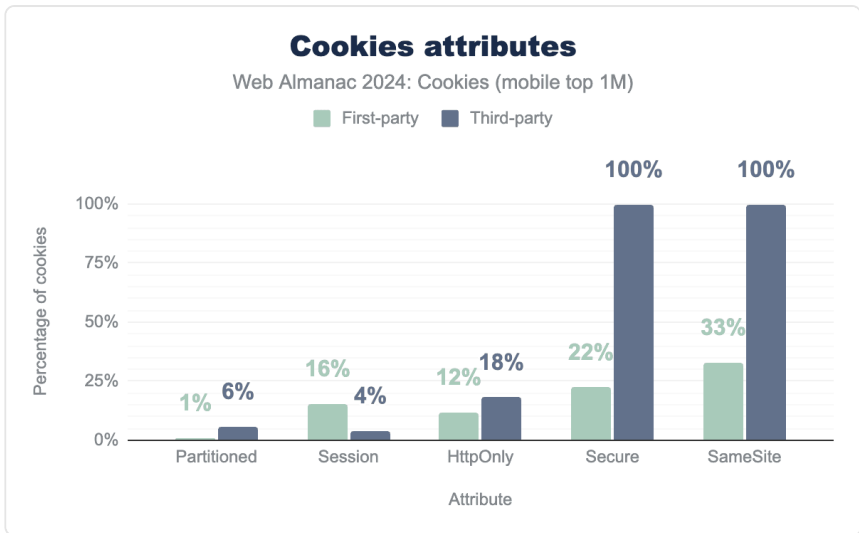


Figure 19.6. An overview of cookie attributes for mobile clients.

Partitioned

Partitioned cookies are stored by compatible browsers⁷²³ using partitioned storage. Cookies that have the `Partitioned` attribute set can only be accessed by the same third party and from the same top-level site where they were created in the first place. In other words, partitioned cookies can not be used for third-party tracking across websites and allow for the legitimate use of third-party cookies on a top-level site. For more details see: Cookies Having Independent Partitioned State (CHIPS)⁷²⁴.

We observe that about 6% of third-party cookies set on desktop or mobile while visiting the top one million websites are partitioned. The next figure shows the most common partitioned cookies (name and domain) that are set in third-party context on the top one million websites. For each client (desktop and mobile) only the top ten partitioned cookies in percentage of websites they are seen on are reported. The top 2 most widely-used partitioned cookies are set by `youtube.com` on 9.9% on desktop and 8.89% mobile websites. The `YSC` cookie is used for security purposes i.e., to prevent fraud and abuse, and expires at the end of the user session, while `VISITOR_INFO1_LIV`'s main purpose is analytics (see Google's documentation⁷²⁵). Most of the cookies listed in the graph are set by advertising domains e.g., `adnxs.com`, `criteo.com`, and `doubleclick.net`.

723. https://developer.mozilla.org/docs/Web/Privacy/Privacy_sandbox/Partitioned_cookies#browser_compatibility

724. https://developer.mozilla.org/docs/Web/Privacy/Privacy_sandbox/Partitioned_cookies

725. <https://policies.google.com/technologies/cookies/embedded?hl=en-US>

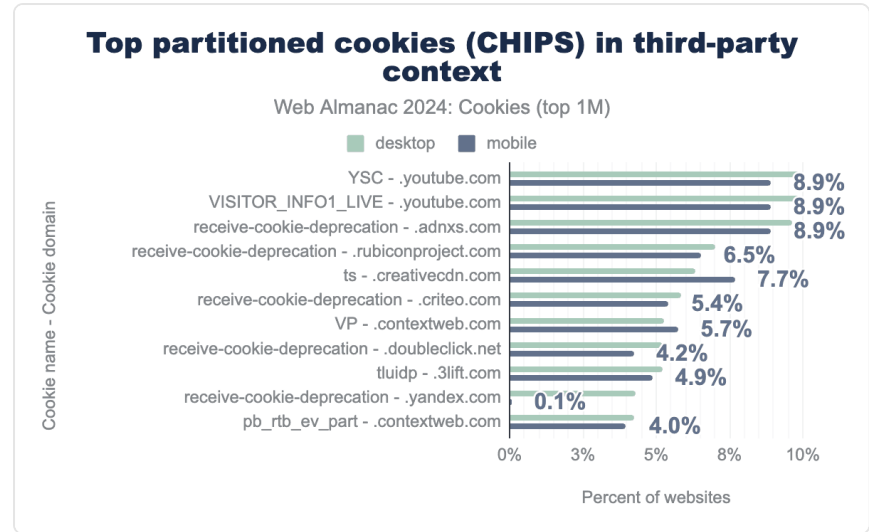


Figure 19.7. Top partitioned cookies (CHIPS) in third-party context.

Perhaps a bit surprising, 1% of all the first-party cookies that are set on the top one million websites (desktop and mobile client) are partitioned. However, partitioning cookies in a first-party context appears to be a bit redundant as first-party cookies are already accessible, by definition, only by that first-party on that top-level site. The following figure displays the top ten partitioned cookies set in first-party context for each client. `receive-cookie-deprecation` is set by domains that participate in the testing phase⁷²⁶ of Chrome's Privacy Sandbox. `cf_clearance` and `csrf_token` are cookies set by Cloudflare to indicate that the user has successfully completed an anti-bot challenge or to identify trusted web traffic, respectively.

726. <https://developers.google.com/privacy-sandbox/private-advertising/setup/web/chrome-facilitated-testing>

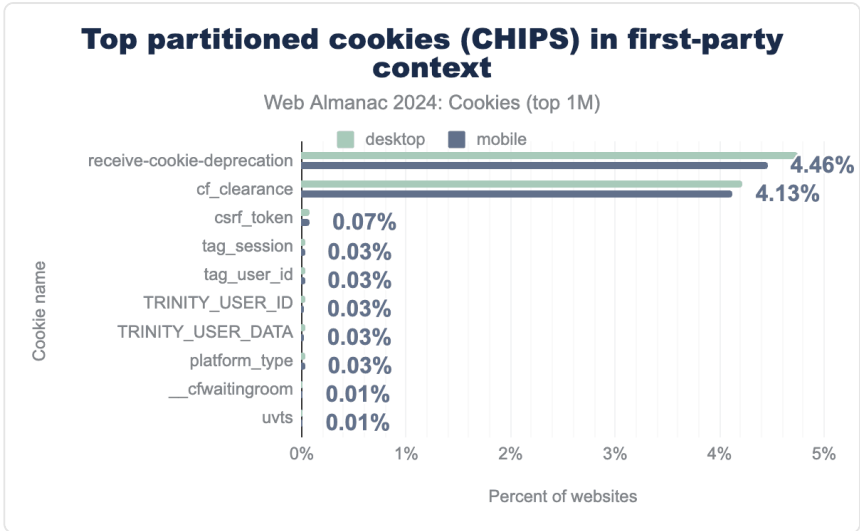


Figure 19.8. Top partitioned cookies (CHIPS) in first-party context.

Session

Session cookies are cookies that are only valid for a single user session. In other words, session cookies are temporary and expire once the user quits the corresponding website they were set on, or closes their web browser, whichever happens first. However, note that some web browsers allow users to restore a previous session on startup, in that case the session cookies set in that previous session are also restored.

The results from our analysis on the top one million websites in June 2024 show that 16% of first-party cookies and only 4% of third-party cookies are session cookies (on both desktop and mobile clients).

HttpOnly

The `HttpOnly` attribute prevents cookies from being accessed by javascript code, this provides some mitigation against cross-site scripting (XSS)⁷²⁷ attacks. Note that setting the `HttpOnly` attribute does not prevent cookies from being sent along `XMLHttpRequest` or `fetch` requests initiated from javascript.

Only 12% of first-party cookies have the `HttpOnly` attribute set, while for third-party cookies

727. https://developer.mozilla.org/docs/Glossary/Cross-site_scripting

19% on desktop and 18% on mobile do.

Secure

Cookies with the `Secure` attribute are only sent to requests made through HTTPS. This prevents man-in-the-middle⁷²⁸ attacks.

For first-party cookies, 23% on desktop and 22% on mobile have the `Secure` attribute and all third-party cookies observed have the `Secure` attribute. Indeed, these third-party cookies also have the `SameSite=None` attribute that requires `Secure` to be set (see the next section).

SameSite

The `SameSite` cookie attribute allows sites to specify when cookies are included with cross-site requests:

- `SameSite=Strict` : a cookie is only sent in response to a request from the same site as the cookie's origin.
- `SameSite=Lax` : same as `SameSite=Strict` except that the browser also sends the cookie on navigation to the cookie's origin site. This is the default value of `SameSite`.
- `SameSite=None` : cookies are sent on same-site or cross-site requests. This means that in order to make third-party tracking with cookies possible, the tracking cookies must have their `SameSite` attribute set to `None`.

To learn more about the `SameSite` attribute, see the following references:

- `SameSite` cookies explained
- “Same-site” and “same-origin”⁷²⁹
- What are the parts of a URL?⁷³⁰

728. <https://developer.mozilla.org/docs/Glossary/MitM>

729. <https://web.dev/articles/same-site-same-origin>

730. <https://web.dev/articles/url-parts>

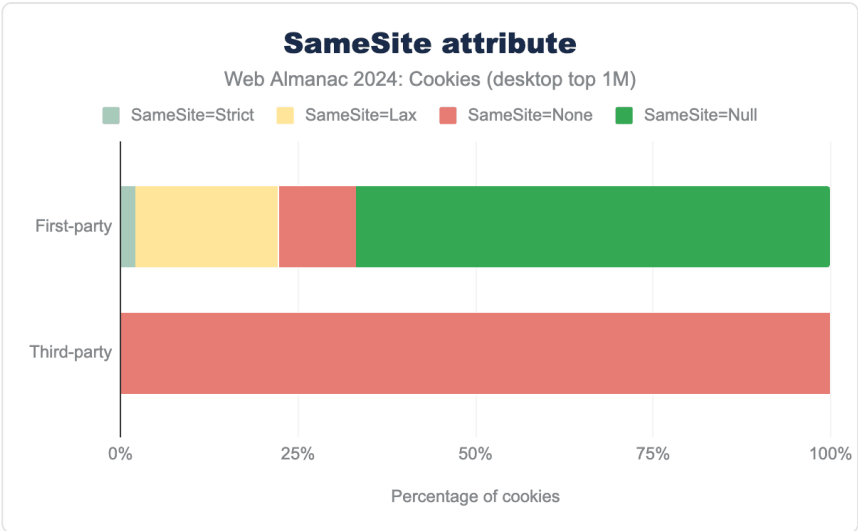


Figure 19.9. SameSite attribute for cookies on desktop client.

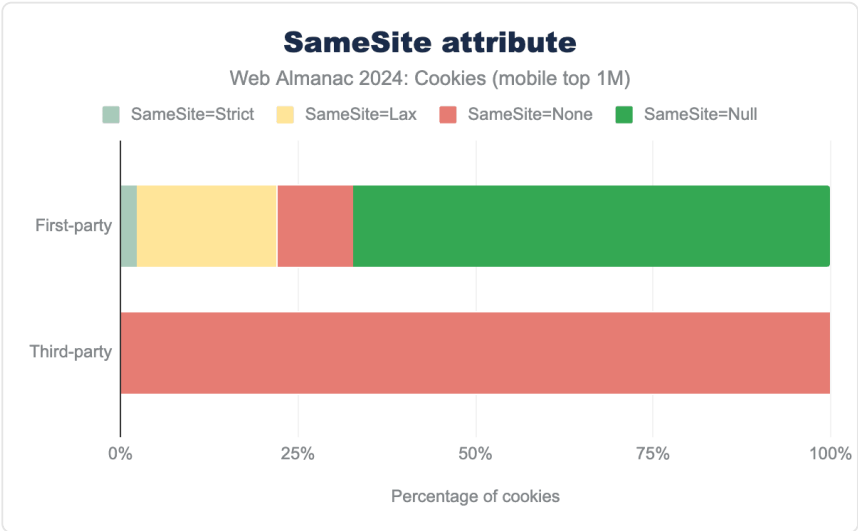


Figure 19.10. SameSite attribute for cookies on mobile client.

We observe that for each client about 33% of the first-party cookies and nearly 100% third-party cookies seen on the top one million websites have a SameSite attribute that is explicitly set when they are created (reminder: SameSite defaults to Lax if not specified). The two bar charts above represent the distribution of this SameSite attribute for first and third-party cookies across clients. We observe that the differences in results across clients is here again

somewhat negligible. Nearly 100% of third-party cookies have `SameSite=None`, and so are sent on cross-site requests. For first-party cookies, about 87% of them have the `SameSite=Lax` (20% explicitly set the attribute, and the remaining 67% are concerned by the default behavior when `SameSite` is not set). 11% of cookies have their `SameSite` attributes explicitly set to have the value `None`. It's hard to determine the exact purpose for which cookies are set, but it is likely that a fraction of these cookies are used to track users in a first-party context. Only 2% of cookies have `SameSite` set to `Strict`.

Cookie prefixes

Two cookie prefixes⁷³¹ `__Host-` and `__Secure-` can be used in the cookie name to indicate that they can only be set or modified by a secure HTTPS origin. This is to defend against session fixation⁷³² attacks. Cookies with both prefixes must be set by a secure HTTPS origin and have the `Secure` attribute set. Additionally, `__Host-` cookies must not contain a `Domain` attribute and have their `Path` set to `/`, thus `__Host-` cookies are only sent back to the exact host they were set on, and so not to any parent domain.

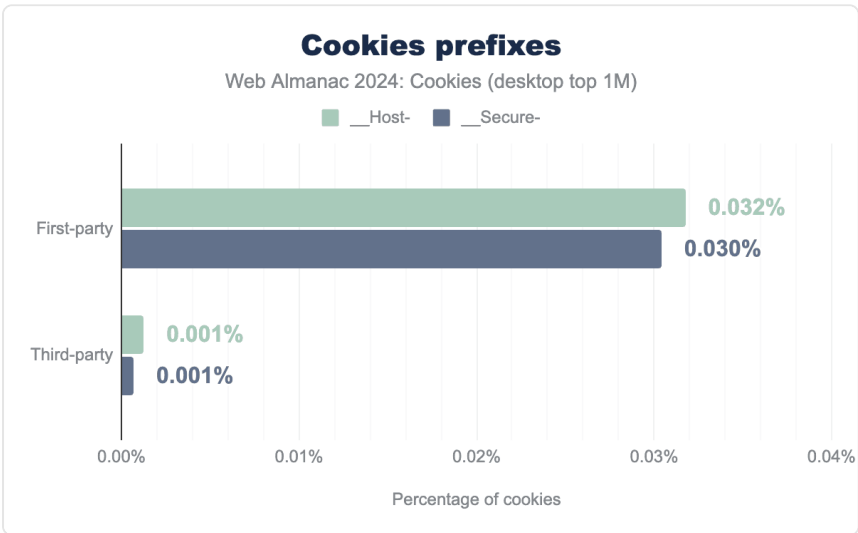


Figure 19.11. Cookie prefixes observed on desktop pages.

731. https://developer.mozilla.org/docs/Web/HTTP/Cookies#cookie_prefixes

732. https://developer.mozilla.org/docs/Web/Security/Types_of_attacks#session_fixation

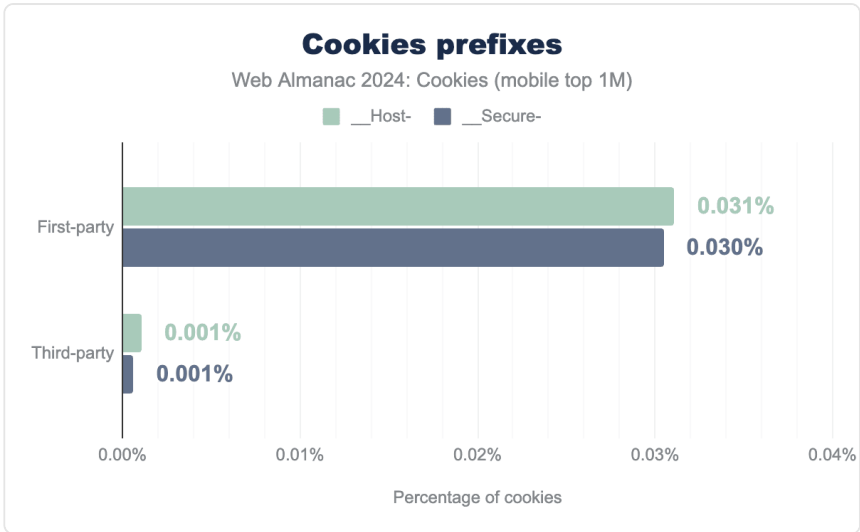


Figure 19.12. Cookie prefixes observed on mobile pages.

We measure that 0.032% and 0.030% of the first-party cookies observed on desktop have the `__Host-` and `__Secure-` prefix set, respectively. These numbers are 0.001% for third-party cookies. These results show the very low adoption of these prefixes and the associated defense-in-depth measure since they were first introduced⁷³³ at the end of 2015.

Top first and third-party cookies and domains setting them

In the following section, we report for each client (desktop and mobile) the top ten first-party cookies, third-party cookies, as well as domains that set them. We comment on a few of them using results from Cookiepedia⁷³⁴ and invite curious readers to refer to this resource for more.

733. <https://datatracker.ietf.org/doc/html/draft-ietf-httpbis-rfc6265bis#section-4.1.3.1>
734. <https://cookiepedia.co.uk/>

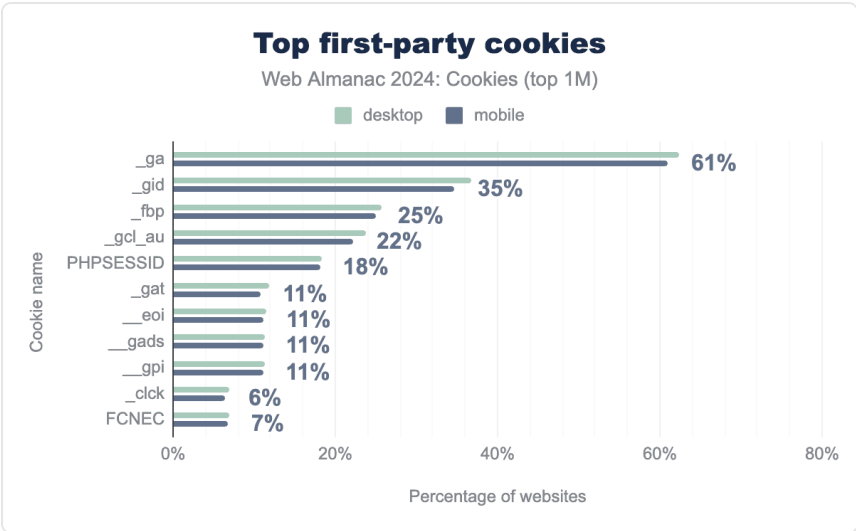


Figure 19.13. Top first-party cookies set.

The first two first-party cookies `_ga` and `_gid` are set by Google Analytics⁷³⁵ to store client identifiers and statistics for site analytics reports, a majority of websites use Google Analytics (more than 60% and 35%, respectively). The third one `_fbp` is set by Facebook and used for targeted advertising on 25% of the websites.

735. <https://business.safety.google/adscokies/>

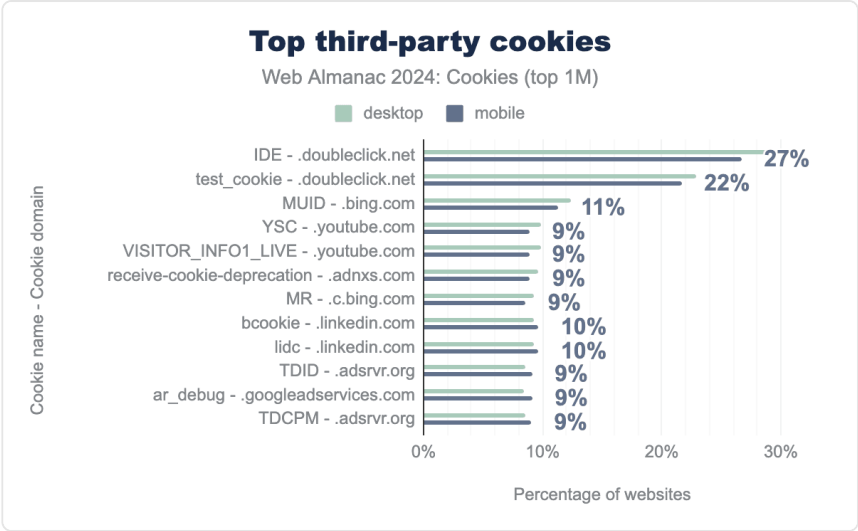


Figure 19.14. Top third-party cookies and domains that set them.

The `IDE` and `test_cookie` cookies are set by `doubleclick.net` (owned by Google) and are the most common third-party cookies observed on the top one million websites; they are used for targeted advertising. DoubleClick checks if a user’s web browser supports third-party cookies by trying to set `test_cookie`. `MUID` from Microsoft comes next and is also used in targeted advertising to store the user’s unique identifier for cross-site tracking.

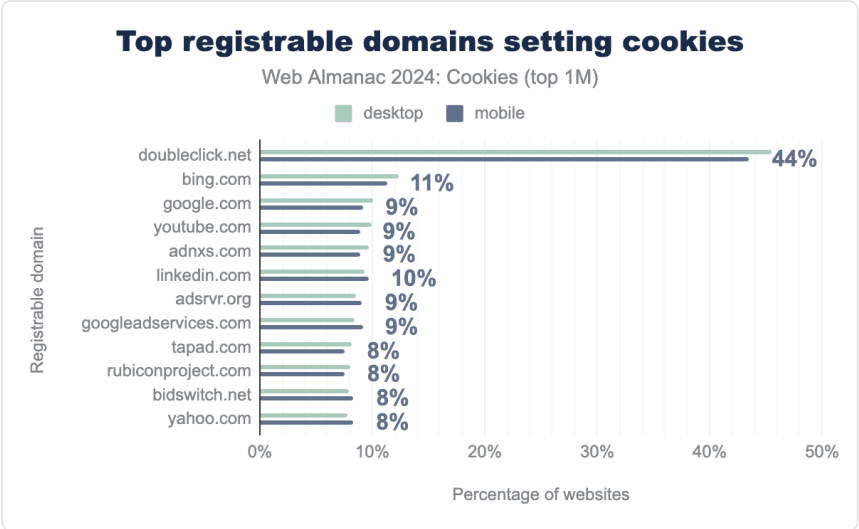


Figure 19.15. Top registrable domains setting cookies.

Among the ten most common domains that set cookies on the web, we only find domains involved in search, targeting, and advertising services. This result outlines the coverage that some third-parties have of the web, for example: Google’s owned advertising platform DoubleClick sets cookies on more than 44% of the top one million websites while others are at about 8% to 12%.

Number of cookies set by websites

Number of cookies (desktop top one million)	First-party	Third-party	All
min	1	1	1
p25	3	2	4
median	7	5	10
p75	13	17	24
p90	22	66	51
p95	46	331	323
max	160	632	662

Figure 19.16. Statistics for number of cookies set on desktop pages.

Number of cookies (mobile top one million)	First-party	Third-party	All
min	1	1	1
p25	3	2	4
median	7	4	9
p75	12	18	24
p90	21	64	52
p95	45	327	316
max	168	604	645

Figure 19.17. Statistics for number of cookies set on mobile pages.

Websites set a median of nine or ten cookies of any type overall, seven first-party cookies, and four or five third-party cookies for mobile and desktop clients, respectively. The tables above report several other statistics about the number of cookies observed per website and the figures below display their cumulative distribution functions (cdf). For example: on desktop a maximum of 160 first-party and 632 third-party cookies are set per website.

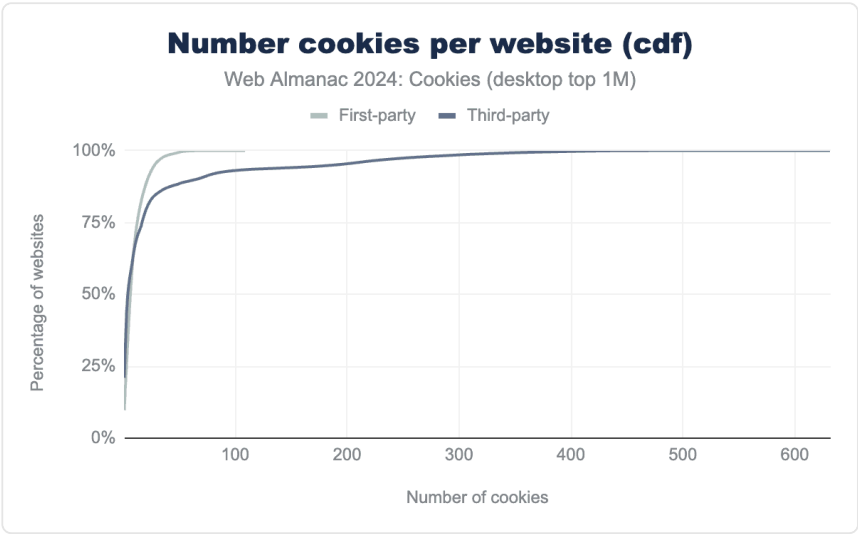


Figure 19.18. Number of cookies per website (cdf) for desktop pages.

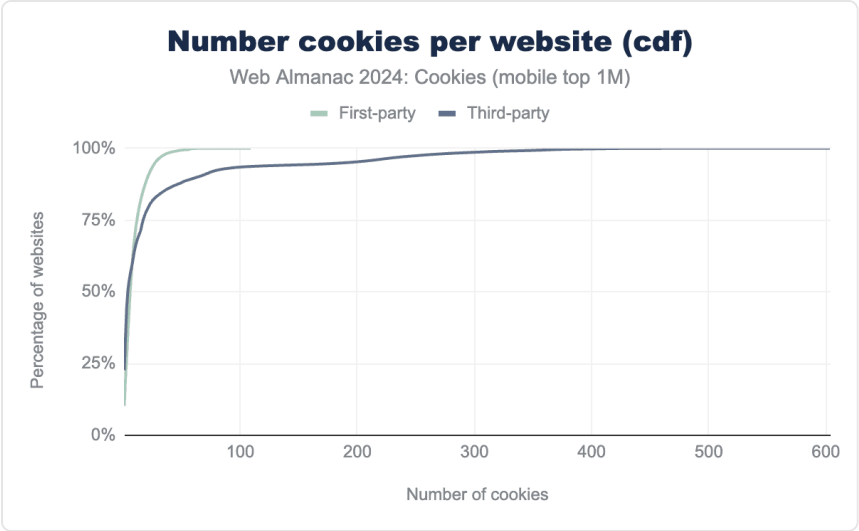


Figure 19.19. Number of cookies per website (cdf) for mobile pages.

We see that more websites have a number of first-party cookies that is closer to the maximum of first-party cookies observed, than for third-party cookies.

Size of cookies

Size of cookies (desktop top one million) in bytes	First-party	Third-party	All
min	1	1	1
p25	26	22	23
median	39	36	37
p75	59	58	58
p90	148	114	128
p95	380	274	348
max	4087	4094	4094

Figure 19.20. Statistics for size of cookies set on desktop pages.

Size of cookies (mobile top one million) in bytes	First-party	Third-party	All
min	1	1	1
p25	26	22	23
median	39	37	38
p75	59	59	59
p90	149	114	130
p95	382	278	352
max	4086	4093	4093

Figure 19.21. Statistics for size of cookies set on mobile pages.

This section focuses on the actual size of these cookies. We find that the median size across all cookies observed on desktop during the HTTP Archive crawl of June 2024 is 37 bytes. This median value is consistent across first and third-party cookies as well as clients. The maximal size that we obtain is at about 4K bytes which is consistent with the limits defined in RFC 6265⁷³⁶. Note that because of the way the HTTP Archive tools work and collect the cookies, if websites try to set cookies larger than the limit of 4K bytes this information would be missing from the data analyzed in this chapter.

The smallest cookies that we observe are of a single byte in size, they are likely set by error by empty `Set-Cookie` headers. Additionally, we also report the cumulative distribution function (cdf) of the size of all the cookies seen on the top one million websites for each client.

Most cookies used for tracking have a size greater than 35 bytes⁷³⁷. The reason for this is that size is related to the tracking capability of cookies: trackers assign identifiers randomly to users in order to be able to re-identify them. So the larger the size (number of bytes) for the identifier, the more unique users they can be assigned to.

736. <https://datatracker.ietf.org/doc/html/rfc6265#section-6.1>
737. https://link.springer.com/chapter/10.1007/978-3-319-15509-8_21

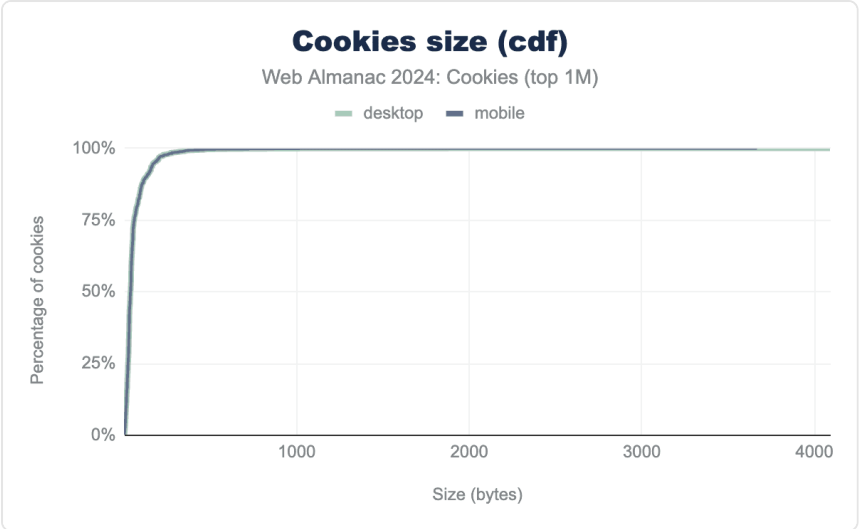


Figure 19.22. Size of cookies per website (cdf) for desktop and mobile pages.

Persistence (expiration)

Age of cookies (desktop top one million) in days	First-party	Third-party	All
min	0	0	0
p25	1	30	30
median	183	365	365
p75	396	365	396
p90	400	400	400
p95	400	400	400
max	400	400	400

Figure 19.23. Statistics for age of cookies set on desktop pages.

Age of cookies (mobile top one million) in days	First-party	Third-party	All
min	0	0	0
p25	1	30	30
median	183	365	365
p75	396	365	390
p90	400	400	400
p95	400	400	400
max	400	400	400

Figure 19.24. Statistics for age of cookies set on mobile pages.

After looking into cookie size, let's now dive into cookie age. Cookies are set to an expiration date when they are created. Recall that session cookies expire immediately after the session is over (see previous section). The median age of first-party cookies is at about 183 days or roughly 6 months, while the median age of third-party cookies is a full year. After less than one day and thirty days, 25% of first-party and third-party cookies expire, respectively. The maximum age among the cookies that we can observe with the instrumentation and collection of the HTTP Archive Tools is of 400 days, this is aligned with the hard limits⁷³⁸ that Chrome imposes on cookie `Expires` and `Max-Age` attribute. Below, are the cumulative distribution functions (cdf) of the age of the cookies set on the top one million websites whether it is on a desktop or mobile client.

738. <https://developer.chrome.com/blog/cookie-max-age-expires>

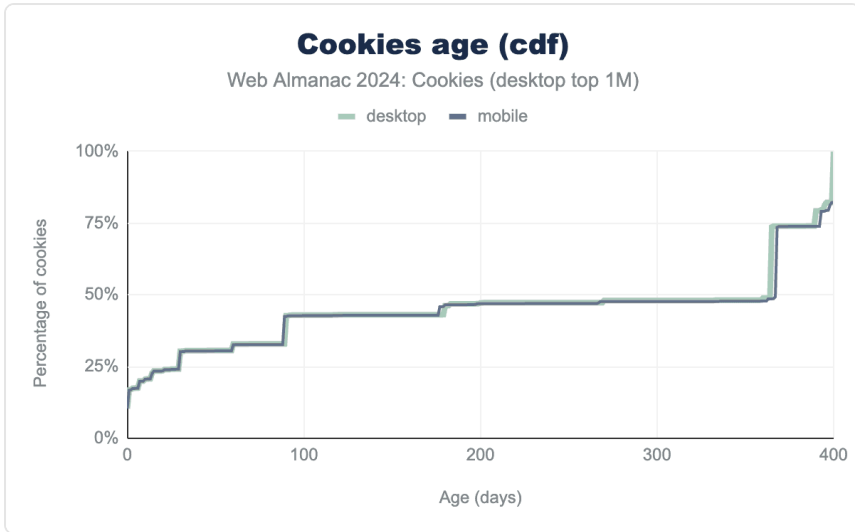


Figure 19.25. Age of cookies per website (cdf) for desktop and mobile pages.

From the graph, we deduce that about 45 % of cookies expire after 90 days. We find the same results for both mobile and desktop clients. Additionally, 75% of cookies have a lifespan of maximum 1 year, while the other half remain stored in the browser for longer than a year. In theory, the longer the lifespan of the cookies, the longer that they can re-identify a recurring user. For this reason, most tracking cookies are typically stored in the browser for a longer time.

Privacy Sandbox initiative

In 2019⁷³⁹, Google announced the launch of the Privacy Sandbox⁷⁴⁰ initiative to reduce cross-site (web) and cross-app (Android) tracking while retaining utility for advertising and other use cases that historically have relied on third-party cookies and other tracking mechanisms.

What is the Privacy Sandbox initiative?

The Privacy Sandbox is composed of more than 20 different proposals⁷⁴¹ that aim to diminish the use of unique identifiers, limiting covert tracking, fighting spam and fraud, showing relevant ads to users, and measuring ad conversions.

Part of Google’s initial plan with the Privacy Sandbox was to deprecate third-party cookies, but

739. <https://blog.google/products/chrome/building-a-more-private-web/>

740. <https://developers.google.com/privacy-sandbox>

741. <https://privacysandbox.com>

in recent updates⁷⁴² Google announced that this was not their intention anymore and that they would rather introduce a “new experience in Chrome that lets people make an informed choice that applies across their web browsing”. At the same time, Google will “continue to make the Privacy Sandbox APIs available and invest in them to further improve privacy and utility”.

We partnered with the Privacy chapter of the Web Almanac 2024 to measure adoption of the Privacy Sandbox APIs on the websites visited by the HTTP Archive crawl and will defer interested readers to their chapter for the analysis of the results. Next, we present an overview of the proposed mechanisms that are part of the Privacy Sandbox and aim at replacing a capability provided by cookies so far.

Topics API

The Topics API⁷⁴³ enables interest-based advertising, without using third-party cookies. The API allows callers (such as ad tech platforms) to access topics of interest that they have observed for a user, but without revealing additional information about the user’s activity.

See the Privacy chapter for some results about the adoption of the Topics API.

Protected Audience

The Protected Audience API⁷⁴⁴ enables on-device ad auctions to serve remarketing and custom audiences, without cross-site third-party tracking. Advertisers can add users to interest groups that are saved by the browser while users are navigating on the web. This allows advertisers to perform retargeted advertising by bidding on the available interest groups the user is part of when they visit a website where an ad auction is performed.

See the Privacy chapter for some results about the adoption of the Protected Audience API.

Attribution Reporting API

The Attribution Reporting API⁷⁴⁵ allows websites and third parties to measure ad conversion, i.e., when a view or a click on an advertisement leads later for example to a purchase. The Attribution Reporting API aims to enable measurement of ad conversion but without the use of cross-site identifiers and cookies.

See the Privacy chapter for some results about the adoption of the Attribution Reporting API.

742. <https://privacysandbox.com/news/privacy-sandbox-update>

743. <https://developers.google.com/privacy-sandbox/private-advertising/topics/web>

744. <https://developers.google.com/privacy-sandbox/private-advertising/protected-audience>

745. <https://developers.google.com/privacy-sandbox/private-advertising/attribution-reporting>

CHIPS

Cookies Having Independent Partitioned State (CHIPS)⁷⁴⁶ allow web developers to specify that they would like the cookies that they are setting to be saved in a partitioned storage, i.e., in a separate cookie jar per top-level site. CHIPS cookies correspond to the partitioned cookies discussed previously in this chapter, in the partitioned section.

Related Website Sets

Related Website Sets⁷⁴⁷ allow websites from the same owner to share cookies among themselves. The creation and submission of a Related Website Set is done at the moment through opening a pull request on a GitHub repository⁷⁴⁸ that Google employees check and merge if deemed valid. Websites that belong to the same related website set must also indicate it by placing a corresponding file at the .well-known URI⁷⁴⁹ `/well-known/related-website-set.json`.

64

Figure 19.26. Number of related primary website sets validated by Google at the time of writing.

Chrome ships with a preloaded file containing related website sets validated by the Chrome team; at the moment of writing (version 2024.8.10.0), there are 64 distinct related website sets. Each related website set contains a primary domain and a list of other domains related to the primary one below one of the following attributes: `associatedSites`, `servicesSites`, and/or `ccTLDs`. These 64 primary domains are each associated with secondary domains as part of their set: 60 sets contain `associatedSites`, 11 `servicesSites`, and 7 `ccTLDs`. We report on the following figure the number of secondary domains for each set. We observe that if a majority of the primary domains are associated with 5 or less secondary domains, `https://journaldesfemmes.com`, `https://ya.ru`, and `https://mercadolibre.com` are linked to 8, 17, and 39 secondary domains among which third party requests are handled as if they were all from the first party, respectively.

746. <https://developers.google.com/privacy-sandbox/cookies/chips>

747. <https://developers.google.com/privacy-sandbox/cookies/related-website-sets>

748. <https://github.com/GoogleChrome/related-website-sets>

749. <https://www.iana.org/assignments/well-known-uris/well-known-uris.xhtml>

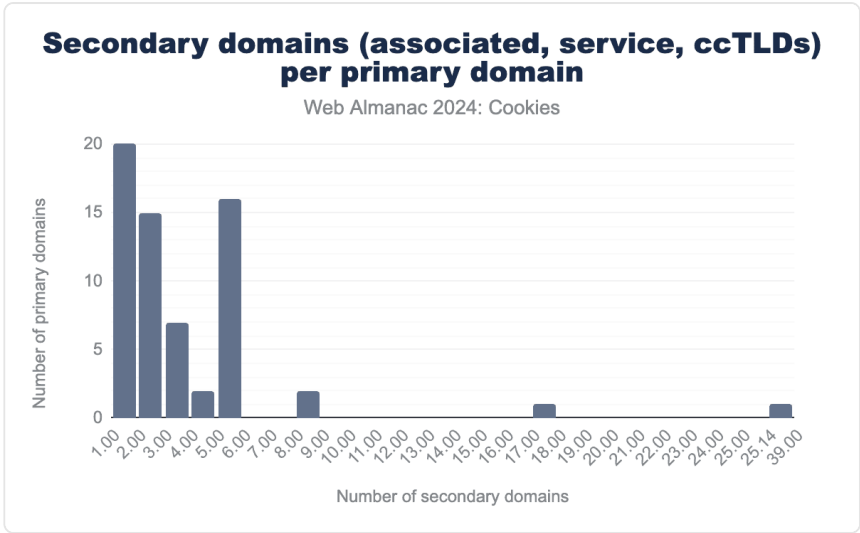


Figure 19.27. Secondary domains per primary domain.

Attestation file

In order to use some of the Privacy Sandbox APIs, API callers have to go through an enrollment⁷⁵⁰ process to declare that they will not abuse these APIs for cross-site re-identification, but only for their intended use cases. The legal implications of this commitment if not respected is quite unclear, but this allows these callers to obtain an attestation file that must be placed at the `.well-known` URI `/.well-know/privacy-sandbox-attestations.json` on the domain they registered to call these APIs from.

Chrome ships with a preloaded file containing a list of domains that have an attestation file registered. Currently, this list contains 257 distinct domains (version `2024.10.7.0`) that have enrolled to call the following APIs: Attribution Reporting, Protected App Signals (Android only), Private Aggregation (Chrome only), Protected Audience, Shared Storage (Chrome only), and Topics.

We used a custom crawler⁷⁵¹ separate from the HTTP Archive tools to obtain and parse these attestation files. We successfully retrieved attestation files for 232 distinct domains with that crawler (some attestation files may be available but not obtained by this crawler due to networking issues for example). Next, we report the proportion of domains that are enrolled for each API on Chrome and Android. We observe that the majority of these origins are enrolled to

750. <https://developers.google.com/privacy-sandbox/private-advertising/enrollment>

751. <https://github.com/privacysandbox/well-known-crawler>

call one of the five Chrome APIs requiring an attestation while the proportion is way less for the Android APIs.

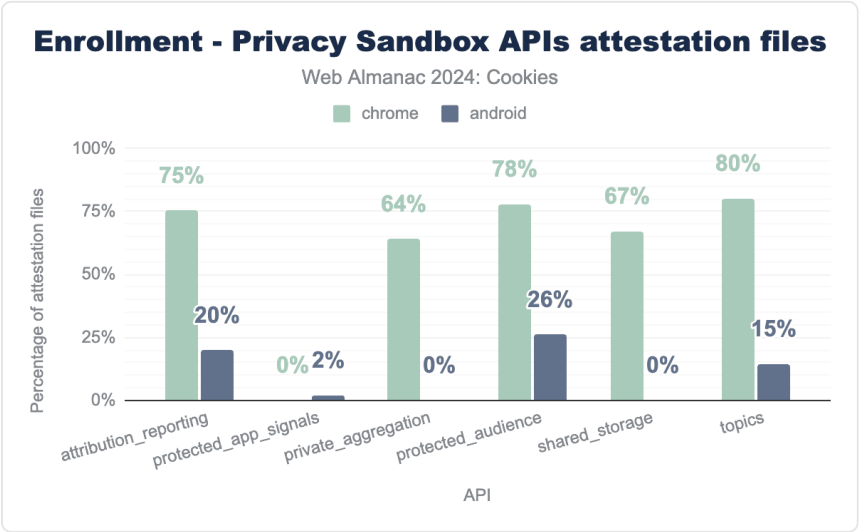


Figure 19.28. Enrollment from Privacy Sandbox APIs attestation files.

Conclusion

In this chapter, we report on the use of cookies on the web. Our analysis allows us to answer multiple questions:

Which type of cookies is set by websites?

We find that the majority of cookies on the web (61%) are third-party. Moreover, more popular websites set significantly more third-party cookies, presumably because they generally include more third-party content. Additionally, we observe that about 6% of third-party cookies are partitioned (CHIPS). Partitioned cookies cannot be used for third-party tracking given that the cookie jar is separate for each website (domain) that the user visits. However, we find that partitioned cookies are predominantly set by advertising domains and are used for analytics.

Which cookie attributes are set?

Out of all cookies set, 16% of first-party cookies and only 4% of third-party cookies are session cookies. The remainder of the cookies are more persistent since they are not deleted when the user closes the browser. Generally, the average lifetime of cookies (the median) is 6 months for

first-party and 1 year for third-party cookies.

Furthermore, for 100% of third-party cookies the `SameSite` attribute is explicitly set to `None`, which allows these cookies to be included in cross-site requests and therefore to track users with them.

Who sets cookies and what are they used for?

The top first-party cookies are mainly used for analytics. Google Analytics, whose primary function is to report on the use of websites by users i.e, first-party analytics, is prevalent on at least 60% of websites. Meta follows its footsteps, by setting first-party cookies on 25% websites.

Third-party cookies also predominantly set by Google: `doubleclick.net` sets a cookie on 44% of websites. Other top trackers have a considerably smaller reach of 8-12% of websites. In general, the most popular third-party cookies belong predominantly to the targeted advertising category.

We conclude the chapter with an overview of the Privacy Sandbox, which aims to replace third-party cookies altogether, and refer to the Privacy chapter for more results.

Authors



Yohan Beugin

yohaan <https://yohan.beugin.org>

Yohan Beugin is a Ph.D. student in the Department of Computer Sciences at the University of Wisconsin–Madison where he is a member of the Security and Privacy Research Group and advised by Prof. Patrick McDaniel. He is interested in building more secure, privacy-preserving, and trustworthy systems. His current research so far has focused on tracking and privacy in online advertising.



Sam Dutton

@sw12 samdutton <https://simpl.info>

Sam Dutton is a Developer Advocate with the Privacy Sandbox team at Google, focused on helping sites migrate away from relying on third-party cookies. Sam grew up in South Australia, went to university in Sydney, and has lived since 1986 in London. He previously worked as a software engineer at BBC R&D and ITN, as a typesetter for Decca Records, and as a researcher at Picador Books.



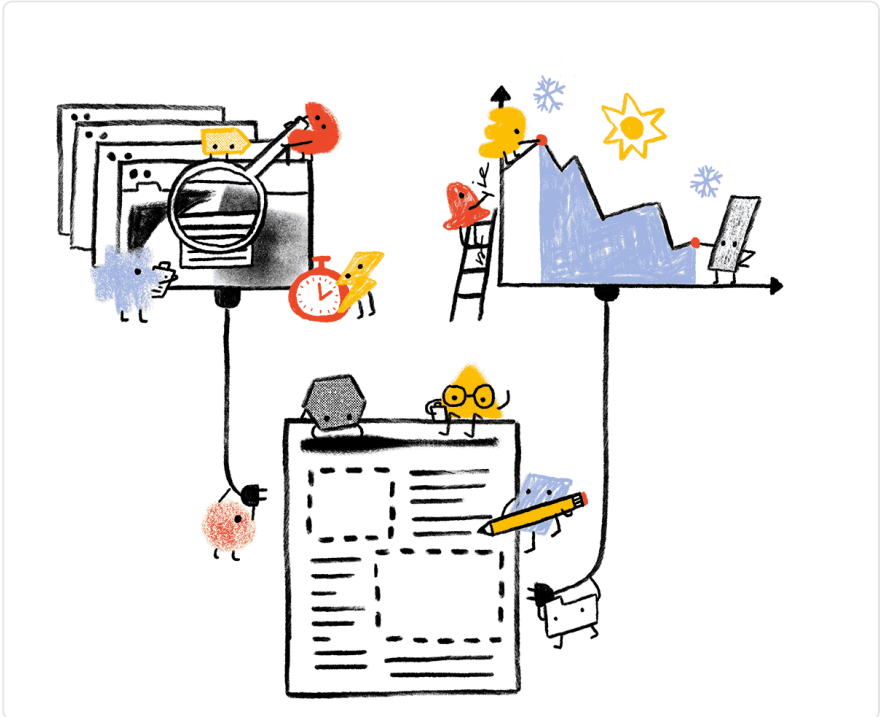
Yana Dimova

 ydimova

Yana Dimova is a PhD student at DistriNet, KU Leuven, focusing on the user’s perspective of privacy and how they can protect it on the web. Her research interests are online tracking, personal data leaks and privacy and data protection law.

Appendix A

Methodology



Overview

The Web Almanac is a project organized by HTTP Archive⁷⁵². HTTP Archive was started in 2010 by Steve Souders with the mission to track how the web is built. It evaluates the composition of millions of web pages on a monthly basis and makes its terabytes of metadata available for analysis on BigQuery⁷⁵³.

The Web Almanac's mission is to become an annual repository of public knowledge about the state of the web. Our goal is to make the data warehouse of HTTP Archive even more

752. <https://httparchive.org>

753. <https://httparchive.org/faq#how-do-i-use-bigquery-to-write-custom-queries-over-the-data>

accessible to the web community by having subject matter experts provide contextualized insights.

The 2024 edition of the Web Almanac is broken into four parts: content, experience, publishing, and distribution. Within each part, several chapters explore their overarching theme from different angles. For example, Part II explores different angles of the user experience in the Performance, Security, and Accessibility chapters, among others.

About the dataset

The HTTP Archive dataset is continuously updating with new data monthly. For the 2024 edition of the Web Almanac, unless otherwise noted in the chapter, all metrics were sourced from the June 2024 crawl. These results are publicly queryable⁷⁵⁴ on BigQuery in tables in the ``httparchive.all.*`` tables for the date `date = '2024-06-01'`.

All of the metrics presented in the Web Almanac are publicly reproducible using the dataset on BigQuery. You can browse the queries used by all chapters in our GitHub repository⁷⁵⁵.

Please note that some of these queries are quite large and can be expensive⁷⁵⁶ to run yourself. For help controlling your spending, refer to Tim Kadlec's post Using BigQuery Without Breaking the Bank⁷⁵⁷.

For example, to understand the number of pages with accessible color contrast, see `color_contrast.sql`⁷⁵⁸:

```
SELECT
  client,
  is_root_page,
  COUNTIF(color_contrast_score IS NOT NULL) AS
total_applicable,
  COUNTIF(CAST(color_contrast_score AS NUMERIC) = 1) AS
total_good_contrast,
  COUNTIF(CAST(color_contrast_score AS NUMERIC) = 1) /
COUNTIF(color_contrast_score IS NOT NULL) AS
```

754. <https://hazfyi/guides/getting-started/>

755. <https://github.com/HTTPArchive/almanac.httparchive.org/tree/main/sql/2024>

756. <https://cloud.google.com/bigquery/pricing>

757. <https://timkadlec.com/remembers/2019-12-10-using-bigquery-without-breaking-the-bank/>

758. https://github.com/HTTPArchive/almanac.httparchive.org/blob/main/sql/2024/accessibility/color_contrast.sql


```

perc_good_contrast
FROM (
  SELECT
    client,
    is_root_page,
    date,
    JSON_VALUE(lighthouse, '$.audits.color-contrast.score')
  AS color_contrast_score
  FROM
    `httparchive.all.pages`
  WHERE
    date = '2024-06-01'
)
GROUP BY
  client,
  is_root_page,
  date
ORDER BY
  client,
  is_root_page;

```

Results for each metric are publicly viewable in chapter-specific spreadsheets, for example JavaScript results⁷⁵⁹. Links to the raw results and queries are available at the bottom of each chapter. Metric-specific results and queries are also linked directly from each figure.

Websites

There are 16,935,953 websites in the dataset. Among those, 16,130,357 are mobile websites and 12,740,973 are desktop websites. Most websites are included in both the mobile and desktop subsets.

HTTP Archive sources the URLs for its websites from the Chrome UX Report. The Chrome UX Report is a public dataset from Google that aggregates user experiences across millions of websites actively visited by Chrome users. This gives us a list of websites that are up-to-date and a reflection of real-world web usage. The Chrome UX Report dataset includes a form factor

⁷⁵⁹. https://docs.google.com/spreadsheets/d/16isMe5_rvmRmJHtK5Je66AhwO8SowGgq0EFqXyjEXw8/edit?gid=1778117656#gid=1778117656

dimension, which we use to get all of the websites accessed by desktop or mobile users.

The June 2024 HTTP Archive crawl used by the Web Almanac used the most recently available Chrome UX Report release for its list of websites. The 202406 dataset was released on Jul 8, 2024 and captures websites visited by Chrome users during the month of June.

Due to resource limitations, the HTTP Archive previously could only test two pages from each website in the Chrome UX report. Be aware that this will introduce some bias into the results because a home page is not necessarily representative of the entire website. This year, we included secondary pages⁷⁶⁰, and many chapters use this new data. Some chapters, however, used just the home pages.

HTTP Archive is also considered a lab testing tool, meaning it tests websites from a datacenter and does not collect data from real-world user experiences. All pages are tested with an empty cache in a logged out state, which may not reflect how real users would access them.

Metrics

HTTP Archive collects thousands of metrics about how the web is built. It includes basic metrics like the number of bytes per page, whether the page was loaded over HTTPS, and individual request and response headers. The majority of these metrics are provided by WebPageTest, which acts as the test runner for each website.

Other testing tools are used to provide more advanced metrics about the page. For example, Lighthouse is used to run audits against the page to analyze its quality in areas like accessibility and SEO. The Tools section below goes into each of these tools in more detail.

To work around some of the inherent limitations of a lab dataset, the Web Almanac also makes use of the Chrome UX Report for metrics on user experiences, especially in the area of web performance.

Some metrics are completely out of reach. For example, we don't necessarily have the ability to detect the tools used to build a website. If a website is built using create-react-app, we could tell that it uses the React framework, but not necessarily that a particular build tool is used. Unless these tools leave detectable fingerprints in the website's code, we're unable to measure their usage.

Other metrics may not necessarily be impossible to measure but are challenging or unreliable. For example, aspects of web design are inherently visual and may be difficult to quantify, like whether a page has an intrusive modal dialog.

⁷⁶⁰. <https://discuss.httparchive.org/t/improving-the-http-archive-pipeline-and-dataset-by-10x/2372>

Tools

The Web Almanac is made possible with the help of the following open source tools.

WebPageTest

WebPageTest⁷⁶¹ is a prominent web performance testing tool and the backbone of HTTP Archive. We use a private instance⁷⁶² of WebPageTest with private test agents, which are the actual browsers that test each web page. Desktop and mobile websites are tested under different configurations:

Config	Desktop	Mobile
Device	Linux VM	Emulated Moto G4
User Agent	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.0.0 Safari/537.36 PTST/ 240613.172707	Mozilla/5.0 (Linux; Android 8.1.0; Moto G (4)) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.0.0 Mobile Safari/537.36 PTST/ 240613.172707
Location	Google Cloud Locations, USA	Google Cloud Locations, USA
Connection	Cable (5/1 Mbps 28ms RTT)	4G (9 Mbps 170ms RTT)
Viewport	1376 x 768px	512 x 360px

Desktop websites are run from within a desktop Chrome environment on a Linux VM. The network speed is equivalent to a cable connection.

Mobile websites are run from within a mobile Chrome environment on an emulated Moto G4 device with a network speed equivalent to a 4G connection.

Test agents run from various Google Cloud Platform locations⁷⁶³ based in the USA.

HTTP Archive’s private instance of WebPageTest is kept in sync with the latest public version and augmented with custom metrics⁷⁶⁴, which are snippets of JavaScript that are evaluated on each website at the end of the test.

The results of each test are made available as a HAR file⁷⁶⁵, a JSON-formatted archive file containing metadata about the web page.

761. <https://www.webpagetest.org/>
762. <https://docs.webpagetest.org/private-instances/>
763. <https://cloud.google.com/compute/docs/regions-zones/#locations>
764. <https://github.com/HTTPArchive/custom-metrics>
765. [https://en.wikipedia.org/wiki/HAR_\(file_format\)](https://en.wikipedia.org/wiki/HAR_(file_format))

Lighthouse

Lighthouse⁷⁶⁶ is an automated website quality assurance tool built by Google. It audits web pages to make sure they don't include user experience antipatterns like unoptimized images and inaccessible content.

HTTP Archive runs the latest version of Lighthouse for all pages. As of the June 2024 crawl, HTTP Archive used the 12.0.0⁷⁶⁷ version of Lighthouse.

Lighthouse is run as its own distinct test from within WebPageTest, but it has its own configuration profile:

Config	Desktop	Mobile
CPU slowdown	N/A	1x/4x
Download throughput	10.0 Mbps	1.6 Mbps
Upload throughput	10.0 Mbps	0.75 Mbps
RTT	40 ms	150 ms

For more information about Lighthouse and the audits available in HTTP Archive, refer to the Lighthouse developer documentation⁷⁶⁸.

Wappalyzer

Wappalyzer⁷⁶⁹ is a tool for detecting technologies used by web pages. There are 108 categories⁷⁷⁰ of technologies tested, ranging from JavaScript frameworks, to CMS platforms, and even cryptocurrency miners. There are over 3,944 supported technologies (a slight increase from 3,805 in 2022).

HTTP Archive runs it's fork of the last open source version of Wappalyzer (v6.10.65), with some extra detections added since.

Wappalyzer powers many chapters that analyze the popularity of developer tools like WordPress, Bootstrap, and jQuery. For example, the CMS chapter relies heavily on the respective CMS⁷⁷¹ category of technologies detected by Wappalyzer.

766. <https://developer.chrome.com/docs/lighthouse/overview>
767. <https://github.com/GoogleChrome/lighthouse/releases/tag/v12.0.0>
768. <https://developer.chrome.com/docs/lighthouse/overview>
769. <https://github.com/HTTPArchive/wappalyzer>
770. <https://github.com/HTTPArchive/wappalyzer/blob/main/src/categories.json>
771. <https://www.wappalyzer.com/categories/cms>

All detection tools, including Wappalyzer, have their limitations. The validity of their results will always depend on how accurate their detection mechanisms are. The Web Almanac will add a note in every chapter where Wappalyzer is used but its analysis may not be accurate due to a specific reason.

Chrome UX Report

The Chrome UX Report⁷⁷² is a public dataset of real-world Chrome user experiences. Experiences are grouped by websites' origin, for example `https://www.example.com`. The dataset includes distributions of UX metrics like paint, load, interaction, and layout stability. In addition to grouping by month, experiences may also be sliced by dimensions like country-level geography, form factor (desktop, phone, tablet), and effective connection type (4G, 3G, etc.).

The Chrome UX Report dataset includes relative website ranking data⁷⁷³. These are referred to as *rank magnitudes* because, as opposed to fine-grained ranks like the #1 or #116 most popular websites, websites are grouped into rank buckets from the top 1k, top 10k, up to the top 10M. Each website is ranked according to the number of eligible⁷⁷⁴ page views on all of its pages combined. This year's Web Almanac makes extensive use of this new data as a way to explore variations in the way the web is built by site popularity.

For Web Almanac metrics that reference real-world user experience data from the Chrome UX Report, the June 2024 dataset (202406) is used.

You can learn more about the dataset in the Using the Chrome UX Report on BigQuery⁷⁷⁵ guide on developer.chrome.com⁷⁷⁶.

Blink Features

Blink Features⁷⁷⁷ are indicators flagged by Chrome whenever a particular web platform feature is detected to be used.

We use Blink Features to get a different perspective on feature adoption. This data is especially useful to distinguish between features that are implemented on a page and features that are actually used.

Blink Features are reported by WebPageTest as part of our regular testing.

772. <https://developer.chrome.com/docs/crux>

773. <https://developer.chrome.com/blog/crux-rank-magnitude>

774. <https://developer.chrome.com/docs/crux/methodology#eligibility>

775. <https://developer.chrome.com/docs/crux/guides/bigquery>

776. <https://developer.chrome.com>

777. https://chromium.googlesource.com/chromium/src/+HEAD/docs/use_counter_wiki.md

Third Party Web

Third Party Web⁷⁷⁸ is a research project by Patrick Hulce, author of the 2019 Third Parties chapter, that uses HTTP Archive and Lighthouse data to identify and analyze the impact of third party resources on the web.

Domains are considered to be a third party provider if they appear on at least 50 unique pages. The project also groups providers by their respective services in categories like ads, analytics, and social.

Several chapters in the Web Almanac use the domains and categories from this dataset to understand the impact of third parties.

Rework CSS

Rework CSS⁷⁷⁹ is a JavaScript-based CSS parser. It takes entire stylesheets and produces a JSON-encoded object distinguishing each individual style rule, selector, directive, and value. See this thread⁷⁸⁰ for more information about how it was integrated with the HTTP Archive dataset on BigQuery.

Parsel

Parsel⁷⁸¹ is a CSS selector parser and specificity calculator, originally written by 2020 CSS chapter lead Lea Verou and open sourced as a separate library. It is used extensively in all CSS metrics that relate to selectors and specificity.

Analytical process

The Web Almanac took about a year to plan and execute with the coordination of more than a hundred contributors from the web community. This section describes why we chose the chapters you see in the Web Almanac, how their metrics were queried, and how they were interpreted.

778. <https://www.thirdpartyweb.today/>

779. <https://github.com/reworkcss/css>

780. <https://discuss.httparchive.org/t/analyzing-stylesheets-with-a-js-based-parser/1683>

781. <https://projects.verou.me/parsel/>

Planning

The 2024 Web Almanac kicked off in March 2024 with a call for contributors⁷⁸². We initialized the project with the same chapters from previous years and the community suggested additional topics, one of which became a new chapter this year: Cookies.

As we stated in the inaugural year's Methodology:

One explicit goal for future editions of the Web Almanac is to encourage even more inclusion of underrepresented and heterogeneous voices as authors and peer reviewers.

To that end, this year we've continued our author selection process⁷⁸³:

- Previous authors were specifically discouraged from writing again to make room for different perspectives.
- Everyone endorsing 2024 authors were asked to be especially conscious not to nominate people who all look or think alike.
- The project leads reviewed all of the author nominations and made an effort to select authors who will bring new perspectives and amplify the voices of underrepresented groups in the community.

We hope to iterate on this process in the future to ensure that the Web Almanac is a more diverse and inclusive project with contributors from all backgrounds.

Analysis

In April and May 2024, data analysts worked with authors and peer reviewers to come up with a list of metrics that would need to be queried for each chapter. In some cases, custom metrics⁷⁸⁴ were created to fill gaps in our analytic capabilities.

Throughout May 2024, the HTTP Archive data pipeline crawled several million websites, gathering the metadata to be used in the Web Almanac. These results were post-processed and saved to BigQuery⁷⁸⁵.

Being our fifth year, we were able to update and reuse the queries written by previous analysts. Still, there were many new metrics that needed to be written from scratch. You can browse all

782. <https://x.com/nrllah/status/1764588403792781823>

783. <https://github.com/HTTPArchive/almanac.httparchive.org/discussions/2165>

784. <https://github.com/HTTPArchive/custom-metrics>

785. <https://console.cloud.google.com/bigquery?p=httparchive&d=almanac&page=dataset>

of the queries by year and chapter in our open source query repository⁷⁸⁶ on GitHub.

Interpretation

Authors worked with analysts to correctly interpret the results and draw appropriate conclusions. As authors wrote their respective chapters, they drew from these statistics to support their framing of the state of the web. Peer reviewers worked with authors to ensure the technical correctness of their analysis.

To make the results more easily understandable to readers, web developers and analysts created data visualizations to embed in the chapter. Some visualizations are simplified to make the points more clearly. For example, rather than showing a full distribution, only a handful of percentiles are shown. Unless otherwise noted, all distributions are summarized using percentiles, especially medians (the 50th percentile), and not averages.

Finally, editors revised the chapters to fix simple grammatical errors and ensure consistency across the reading experience.

Looking ahead

The 2024 edition of the Web Almanac is the fifth in what is mostly an annual tradition (we took a break in 2023) in the web community of introspection and a commitment to positive change. Getting to this point has been a monumental effort thanks to many dedicated contributors and we hope to leverage as much of this work as possible to make future editions even more streamlined.

⁷⁸⁶. <https://github.com/HTTPArchive/almanac.httparchive.org/tree/main/sql/2024>

Appendix B

Contributors



The Web Almanac has been made possible by the hard work of the web community. 92 people have volunteered countless hours in the planning, research, writing and production phases of the 2024 Web Almanac.



Abdul Haddi Amjad

✉ @haddiamjad
 📧 hadiamjad
 🌐 <https://haddiamjad.github.io/>
 Analyst and Author



Alberto Fernandez-de-Retana

✉ @alberto_fdr
 📧 AlbertoFDR
 🌐 <https://albertofdr.github.io/>
 Reviewer



Alex Moening

✉ AlexMoening
 Analyst and Author



Alexander Dawson

✉ @AlexDawsonUK
 📧 <https://mastodon.design/@AlexDawsonUK>
 📧 AlexDawsonUK
 🌐 alexdawsonuk
 🌐 <https://alexanderdawson.com/>
 Author



Andrea Volpini

✉ @cyberandy
 📧 cyberandy
 🌐 <https://wordlift.io/blog/en/entity/>
 andrea-volpini
 Author



Barry Pollard

✉ @tunetheweb
 📧 <https://webperf.social/@tunetheweb>
 📧 tunetheweb.com
 📧 tunetheweb
 🌐 <https://www.tunetheweb.com>
 Analyst, Organizing Committee,
 Developer, Editor, and Reviewer



Beatriz González Mellídez

✉ @b_atish
 📧 bgonzalez
 🌐 https://medium.com/@b_atish
 Reviewer



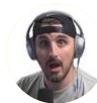
Benjamin Standaert

✉ bstandaert-wustl
 🌐 ben-standaert-15580a1b8
 Analyst and Author



Bram Stein

📧 <https://typo.social/@bram>
 📧 bramstein
 🌐 <http://www.bramstein.com/>
 Author



Brian Clark

✉ @_clarkio
 📧 clarkio
 🌐 <https://www.clarkio.com>
 Reviewer



Brian Kardell

✉ @briankardell
🌐 <https://toot.cafe/@bkardell>
🌐 bkardell
🌐 <https://bkardell.com>
Reviewer



Burak Güneli

✉ @burakGuneli
🌐 burakguneli
Analyst and Author



Caleb Queern

✉ @httpsechheaders
🌐 cqueern
Organizing Committee, Editor, and Project Lead



Caroline Scholles

🌐 carolinescholles
Editor and Reviewer



Charles Berret

🌐 charlesberret
🌐 <https://charlesberret.net/>
Author and Editor



Chris Böttger

🌐 ChrisBeeti
Analyst, Author, Organizing Committee, and Reviewer



Chris Lilley

✉ @svgeesus
🌐 svgeesus
🌐 <https://svgees.us>
Reviewer



Chris Nichols

🌐 cnichols013
🌐 chris-nichols
Analyst



Dan Knauss

🌐 dknauss
🌐 <https://newlocalmedia.com>
Reviewer



Dave Smart

🌐 tamethebots.com
🌐 dwsmart
🌐 davessmart
🌐 <https://tamethebots.com>
Author



David Large

✉ @avidlarge
🌐 <https://fosstodon.org/@avidlarge>
🌐 David-Large
🌐 david-large-4875b81b2
Editor



Dominik Röttsches

🌐 drott
Reviewer



Eric Portis

✉ @etportis
🌐 ericportis.com
🌐 eeeps
🌐 <https://ericportis.com>
Analyst and Author



Estela Franco

✉ @guaca
🌐 <https://toot.cafe/@guaca>
🌐 guaca.bsky.social
🌐 guaca
🌐 estelafranco
🌐 <https://estelafranco.com/>
Analyst and Author



Gertjan Franken

✉ @GJFR_
🌐 GJFR
🌐 gertjan-franken
🌐 <https://gjfr.dev>
Author



Günes Acar

🌐 gunesacar
🌐 <https://gunesacar.net/>
Organizing Committee



Henri Helvetica

✉ @HenriHelvetica
🌐 henrihelvetica
Organizing Committee



Henry Price

📧 henryp25

📄 [henry-price-9ab362b4](https://www.linkedin.com/in/henry-price-9ab362b4)
Analyst



Hidde de Vries

🌐 <https://front-end.social/@hdv>

📧 hidde

🌐 <https://hidde.blog/>
Reviewer



Ines Akrap

📧 @InesAkrap

📧 ines-akrap

🌐 <https://inesakrap.com/>
Author and Reviewer



Ivan Ukhov

📧 IvanUkhov

Analyst and Reviewer



Jaiganesh Girinathan

📧 pgjaiganesh

Analyst and Author



James Gallagher

📧 capjamesg

🌐 <https://jamesg.blog/>
Editor



James Ross

📧 @CherryJimbo

📧 Cherry

📄 [jamesross134](https://www.linkedin.com/in/jamesross134)

🌐 <https://jross.me/>
Editor



Jamie Indigo

📧 not-a-robot.com

📧 fellowhuman1101

📄 [jamie-indigo](https://www.linkedin.com/in/jamie-indigo)

🌐 <https://not-a-robot.com>
Author



Jannis Rautenstrauch

📧 @jannis_r

📧 JannisBush

📄 [jannis-rautenstrauch](https://www.linkedin.com/in/jannis-rautenstrauch)

🌐 <https://jannisbush.github.io/>
Analyst



Jarno van Driel

📧 @JarnoVanDriel

📧 jvandriel

📄 [jarno-van-driel-36a47075](https://www.linkedin.com/in/jarno-van-driel-36a47075)
Reviewer



Jens Oliver Meiert

📧 @j9t

🌐 <https://mas.to/@j9t>

📧 meiert.com

📧 j9t

📄 [meiert](https://www.linkedin.com/in/meiert)

🌐 <https://meiert.com/en/>
Reviewer



Jevgenija Zigisova

📧 @jevgeniazi

📧 imeugenia

📄 [imeugenia](https://www.linkedin.com/in/imeugenia)
Author



Joe Viggiano

📧 joeviggiano

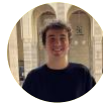
Author



Jonathan Avila

📧 mraccess77

📄 [jonathan-avila-cpwa-2a964a7](https://www.linkedin.com/in/jonathan-avila-cpwa-2a964a7)
Editor



Jonathan Pagel

📧 jcmpagel

📄 [jonathan-pagel](https://www.linkedin.com/in/jonathan-pagel)

🌐 <https://jonathanpagel.com>
Analyst, Author, and Editor



Jonathan Wold

📧 @sirjonathan

📧 sirjonathan

🌐 <https://jonathanwold.com>
Analyst and Author



José Solé

📧 @jmsoleb

📧 jmsole

🌐 <https://www.jmsole.cl/>
Reviewer



Kate Kalceвич

📧 katekalceвич

🌐 katekalceвич
Reviewer



Kevin Farrugia

✉ @imkevdev

🌐 https://webperf.social/@kevinfarrugia

📧 kevinfarrugia

🌐 imkevdev

🌐 https://imkev.dev
Analyst



Laurent Devernay Satyagraha

📧 ldevernay

🌐 laurent-devernay-
satyagraha-2610b85

🌐 https://ldevernay.github.io/
Author



Liam Quin

📧 liamquin

Reviewer



Lora Raykova

📧 LoraRaykova

🌐 lraykova
Author



Lucia Harcegova

📧 Falafelqueen

Analyst



Mandy Michael

✉ @Mandy_Kerr

📧 mandymichael

🌐 https://mandymichael.com/
Reviewer



Matteo Große-Kampmann

✉ @pizzahax

📧 lord-r3

🌐 https://lord-r3.github.io/
Reviewer



Max Ostapenko

📧 max-ostapenko

🌐 max-ostapenko

🌐 https://maxostapenko.com
Analyst, Author, and Developer



Michael Lewittes

✉ @MichaelLewittes

🌐 https://seocommunity.social/@MichaelLewittes

📧 michaellewittes.bsky.social

📧 MichaelLewittes

🌐 michael-lewittes-a-22b831

🌐 https://www.ranktify.com/team
Author and Editor



Michelle O'Connor

Designer



Mika Steinleitner

📧 mika943

Translator



Mikael Araújo

✉ @miknaraujo

📧 mikaelaraujo.bsky.social

📧 mikaelaraujo

🌐 mikael-araujo

🌐 https://www.mikaelaraujo.com
Author



Mike Gifford

🌐 https://mastodon.social/@mgifford

📧 mgifford.bsky.social

📧 mgifford

🌐 mgifford

🌐 https://accessibility.civicactions.com/
Analyst and Author



Mike Neumegen

✉ @mikeneumegen

🌐 https://fosstodon.org/@mikeneu

📧 mneumegen

🌐 mneumegen

🌐 https://mikeneumegen.com/
Author



Miriam Naß

Organizing Committee



Montserrat Cano

📧 montsecano.bsky.social

📧 montsec

🌐 montsecano-senior-digital-marketer

🌐 https://montserrat-cano.com/
Editor

**Niko Kaleev**

niko-kaleev

 niko-kaleev-8760a9131
Author and Editor**Nishu Goel**

@TheNishuGoel

NishuGoel

<https://unravelweb.dev/>
Author**Nurullah Demir**

nrllh

nurullahdemir

<https://ndemir.com>
Analyst, Organizing Committee,
Project Lead, and Reviewer**Onur Güler**

onurglr

Analyst

**Raelene Morey**

raewrites

Editor and Reviewer

**Rafael Bonalume Lebre**

@lebreRafael

 lebreRafael
Reviewer**Raph Levien**

@raphlinus

raphlinus

<https://levien.com>
Reviewer**Rick Viscomi**

@rick_viscomi

rviscomi

<https://rviscomi.dev/>
Reviewer**Robin Marx**

@programmingart

programmingart.bsky.social

rmarx

rmarx

<http://internetonmars.org>
Author**Rowan Merewood**

@rowan_m

https://mastodon.social/@rowan_m

rowan-m

rowanmerewood

<https://merewood.org/>
Reviewer**Ryan Levering**

@rrlevering

rrlevering

Reviewer

**Safin Ainur**

ADSafin

Translator

**Sakae Kotaro**

@beltway7

ksakae1216

<https://ksakae1216.com/>
Translator**Sam Dutton**

@sw12

samdutton

<https://simpl.info>
Author and Reviewer**Saptak Sengupta**

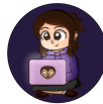
@Saptak013

SaptakS

<https://saptaks.website>
Organizing Committee and Developer**Selman Koral**

krslsman

Developer and Translator

**Shaina Hantsis**

shantsis

Editor

**Shaoor Munir**

@Shaoor_Munir

shaoormunir

<https://shaoormunir.com>
Author



Sia Karamalegos

🌐 <https://front-end.social/@sia>
🦋 sia.codes
🐙 [sia.karamalegos](https://sia.codes)
🌐 [karamalegos](https://sia.codes)
🌐 <https://sia.codes>
Reviewer



Simon Pieters

✉️ [@zcorpan](https://twitter.com/zcorpan)
🐙 zcorpan
Reviewer



Stefan Judis

✉️ [@stefanjudis](https://twitter.com/stefanjudis)
🦋 stefanjudis.com
🐙 stefanjudis
🌐 <https://www.stefanjudis.com>
Analyst and Author



Thom Krupa

✉️ [@thomkrupa](https://twitter.com/thomkrupa)
🐙 thomkrupa
🌐 <https://www.thomkrupa.com/>
Reviewer



Tim Frick

✉️ [@timfrick](https://twitter.com/timfrick)
🐙 timfrick
🌐 <https://www.mightybytes.com/>
Author



Tobias Urban

🐙 turban1988
<https://internet-sicherheit.de/ueber-uns/team/alle-mitarbeiter/urban-tobias-2/>
Author and Organizing Committee



Umar Iqbal

✉️ [@umaarr6](https://twitter.com/umaarr6)
🐙 [UmarIqbal](https://umariqbal.com)
🌐 <https://www.umariqbal.com>
Author



Vik Vanderlinden

✉️ [@vikvanderlinden](https://twitter.com/vikvanderlinden)
🐙 vikvanderlinden
🌐 <https://vikvanderlinden.be/>
Author



Yana Dimova

🐙 ydimova
Author



Yash Vekaria

✉️ [@yekariayash](https://twitter.com/yekariayash)
🐙 [Yash-Vekaria](https://yash-vekaria)
🌐 <https://www.yashvekaria.com/>
Analyst and Author



Yohan Beugin

🐙 yohhaan
🌐 <https://yohan.beugin.org>
Analyst and Author



Yusuf Seyhan

✉️ [@yuseyhan](https://twitter.com/yuseyhan)
🐙 yuseyhan
🌐 <https://webpen.de/>
Designer



Zubair Shafiq

✉️ [@zubair_shafiq](https://twitter.com/zubair_shafiq)
🐙 zubairshafiq
🌐 <http://www.cs.ucdavis.edu/~zubair>
Author