CSS

Web Almanac 2022

Content team

*Hello content team! This is your personal doc to collaborate on and plan the contents of your chapter. Click **Request edit access** above to get started.*

Please add your name and email address below so we can @tag each other in the comments. You can also subscribe to all comments by opening the comment history, clicking the notification bell, and selecting All.

Authors: *Rachel Andrew*, ...
Reviewers: *Chris Lilley (chris@w3.org), Jens Oliver Meiert (jens@meiert.com), ...*
Analysts: *Rick Viscomi*

The objective of your chapter is to write a data-driven answer to this big question:

"What is the state of CSS in 2022?"

Learn more about the chapter lifecycle and refer to your chapter's tracking issue on GitHub for more info. Thank you all for your contributions!

Outline

The purpose of this section is to define the scope of the chapter by creating an ordered list of all of the topics to be explored. You can think of this outline as the chapter's table of contents. This list will become your narrative, so consider how the content should be sequenced and how much additional depth is needed for major topics. You may choose to start with last year's outline and add or remove content as needed. Every chapter must have an introduction and conclusion, but everything in between is up to you.

Every chapter must also be data-driven, so for each topic in the outline below, clearly enumerate which metrics you'll need to substantiate your narrative. Work with your analysts to clarify what data is needed and how the results should be formatted. For example, if you're measuring the usage of a particular HTTP header value, you can measure it as the percentage of pages having that header, as the percentage of headers having that value, as a distribution of values, what the largest value is, etc. Clarify those expectations upfront so that the analysts know how to write the corresponding queries and whether the metrics are even feasible in the dataset.

***Complete the outline by** May 15*
***Implement any custom metrics** by June 1*
***Run all queries** by August 1*

Note from Yr Humble Lead: This is largely the outline of the 2021 almanac's CSS chapter, as it had a great scope and it can be useful to see how things change year over year. New things being proposed for addition this year *are in italics*. Please add your own proposed additions, questions, or comments!

- Introduction
- Methodology
- Usage
    - Style sheet size
    - Numbers of <link> and <style> elements
    - Numbers of rules per sheet / per page
- Selectors and the Cascade
    - Class/ID names
    - Specificity / !important
    - Pseudos
        - :focus customization / removal
        - :focus-visible use

        - *Common generated content*
        - *:has*
    - Attributes
    - *@layer*
- Values and Units
    - Length
    - Percentages
    - Calc()
    - Global values (inherit/all/etc.)
- Custom Properties
    - Names
    - Types
    - Properties
    - Functions
    - Complexity
- Colors
    - Formats
        - *Color fallback strategies?*
        - *Focus on new formats hwb(), color() etc.*
    - color-scheme
    - Alpha use
    - Names
    - *Accent-color*
    - *color-mix()*
- Gradients and Images
    - Gradient types
    - Formats of images in CSS (GIF/JPG/PNG/WEBM/SVG/AVIF/gradients)
    - Number of images in CSS
    - Weight of images in CSS
    - Pixel size of images in CSS
- Layout
    - Methods
    - Box sizing
    - Flexbox
    - Grid
        - *subgrid*
    - Multicol
        - *Use of fragmentation properties in multicol break-before, break-after, break-inside*
- Transitions/Animations

    - Frequency of Tr/An
    - Properties used
    - Durations and timings
    - Average animation/transition duration on the web?
    - Keyframe counts/positions
- Visual Effects
    - Blend/mix modes
    - Filters
    - Masks/clip paths
- Responsive Design (media queries)
    - Queried features
        - prefers-reduced-motion, prefers-color-scheme, prefers-reduced-data
        - *Hover / any-hover, pointer / any-pointer*
        - *forced-colors*
    - Breakpoints
    - Properties changed in queries
    - *New media query syntax from level 4 spec*
    - *Container Queries*
- Feature Queries (@supports)
    - Number of queries
    - Properties queried
    - Properties changed in queries
    - Vendor prefixes (?)
- Internationalization
    - Direction
    - Logical properties
    - Ruby
- CSS in JS
    - Any CSS/JS
    - Houdini
    - **Anything else?**
- Libraries
    - Sass/SCSS
    - **Anything else?**
- *CSS for print*
    - *Pages with print stylesheets*
    - *Usage of fragmentation properties in print stylesheets*
    - *Usage of @page*
- Meta

    - Repetition
    - Shorthands/longhands
    - Syntax errors
    - Mystery properties
    - *Polyfill usage*
    - **Anything else?**
- Conclusion

Draft

The purpose of this section is to iterate on a full draft of the chapter after all of the metric results are ready. Data visualizations for inline figures can be copy/pasted in from the results spreadsheet.

***Peer review and edit by** September 1*
***Final markdown submitted by** September 15*

Introduction

CSS is the language used to lay out and format web pages and other media. It is one of the three main languages of the web, joining HTML which is used for structure, and JavaScript for behavior.

The past few years have seen a flurry of new CSS features. Many of these have taken inspiration from things developers were already doing with JavaScript or in preprocessors, others provide methods of doing things that were impossible a few years ago. Having new features available is one thing, but are developers actually using them in their production web pages and applications? It is this question we will try to answer with data.

In this chapter we use the data to find out what developers actually use in production, rather than the features most talked about on Twitter, showcased at conferences, or found in clever demos. We can see which of the new features are being adopted, which old techniques are falling out of use, and the legacy techniques that are stubbornly remaining in our stylesheets.

Usage

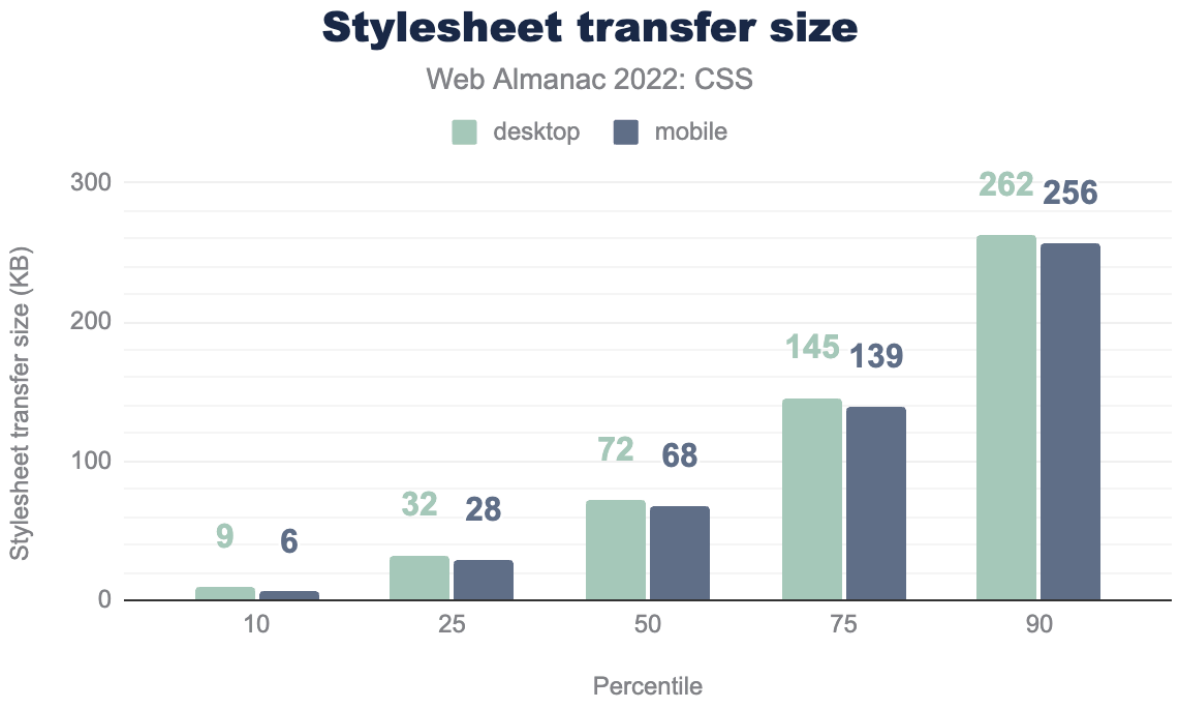

Figure 1.1. Distribution of the stylesheet transfer size by page.

Each year, we see that CSS grows in size, and 2022 was no exception. Other than the 25th percentile, which dropped a percentage point, each percentile showed a small increase in size. At the 90th percentile the increase was almost 7%, a similar increase to that seen between 2020 and 2021. Mobile stylesheets remain slightly smaller than those served to desktop.

The desktop page with the greatest CSS weight was slightly smaller than last year at 62,631 KB. The largest mobile stylesheet had risen from 17,823 KB to 78,543 KB, thankfully this was an exception.

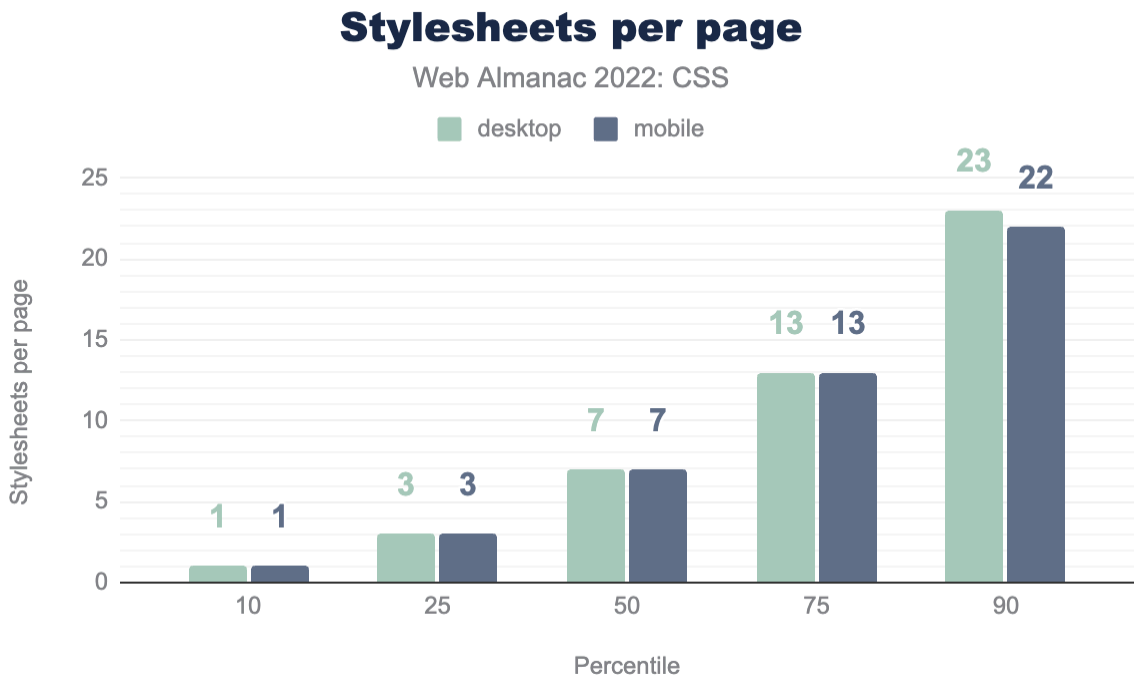

Figure 1.2. Distribution of the number of stylesheets per page

The number of stylesheets per page has remained almost identical to 2021, with an increase of one for mobile at the 50th percentile.

Last year the record was broken for the number of stylesheets loaded by a single page at 2,368. This year we found one site loading 1,387 stylesheets on mobile, still a significant amount.

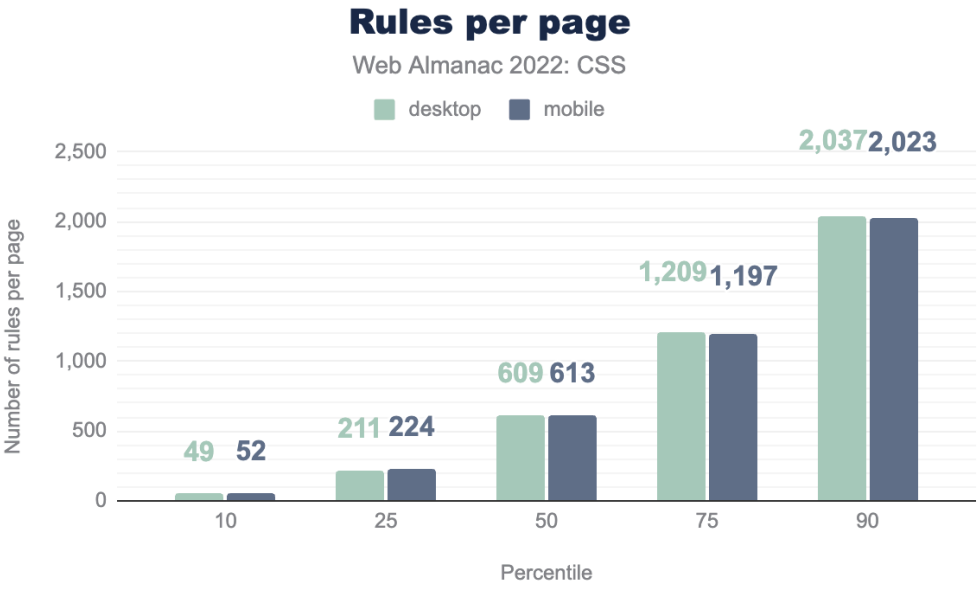

Figure 1.3. Distribution of the total number of style rules per page.

Taking a look at the number of style rules in a page showed an increase across all percentiles, the lower percentiles showing more rules for mobile, the higher percentiles more for desktop. These increases are substantial. Desktop rules for the 50th percentile increased by 130 rules, and the 90th percentile by 202.

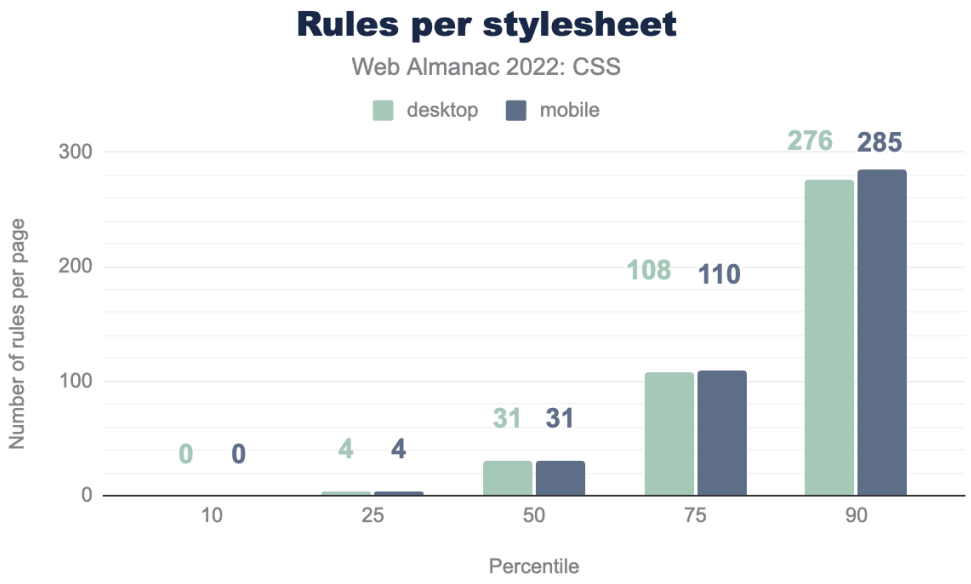

Figure 1.4. Distribution of the number of rules per stylesheet.

We can see from the total number of stylesheets loaded, that typically people are breaking their CSS down into multiple stylesheets. At the 50th percentile this works out as 31 rules per stylesheet, growing to 276 rules on desktop and 285 rules for mobile at the 90th percentile.

Selectors and the cascade

2022 saw a shake-up with regard to the cascade with @layer landing in all engines. This new at-rule enables the grouping of selectors into layers, the order of precedence of the layers can then be managed.

It's a little early to see widespread usage of this new method of managing the cascade, but let's take a look at how selector usage has evolved.

Class names

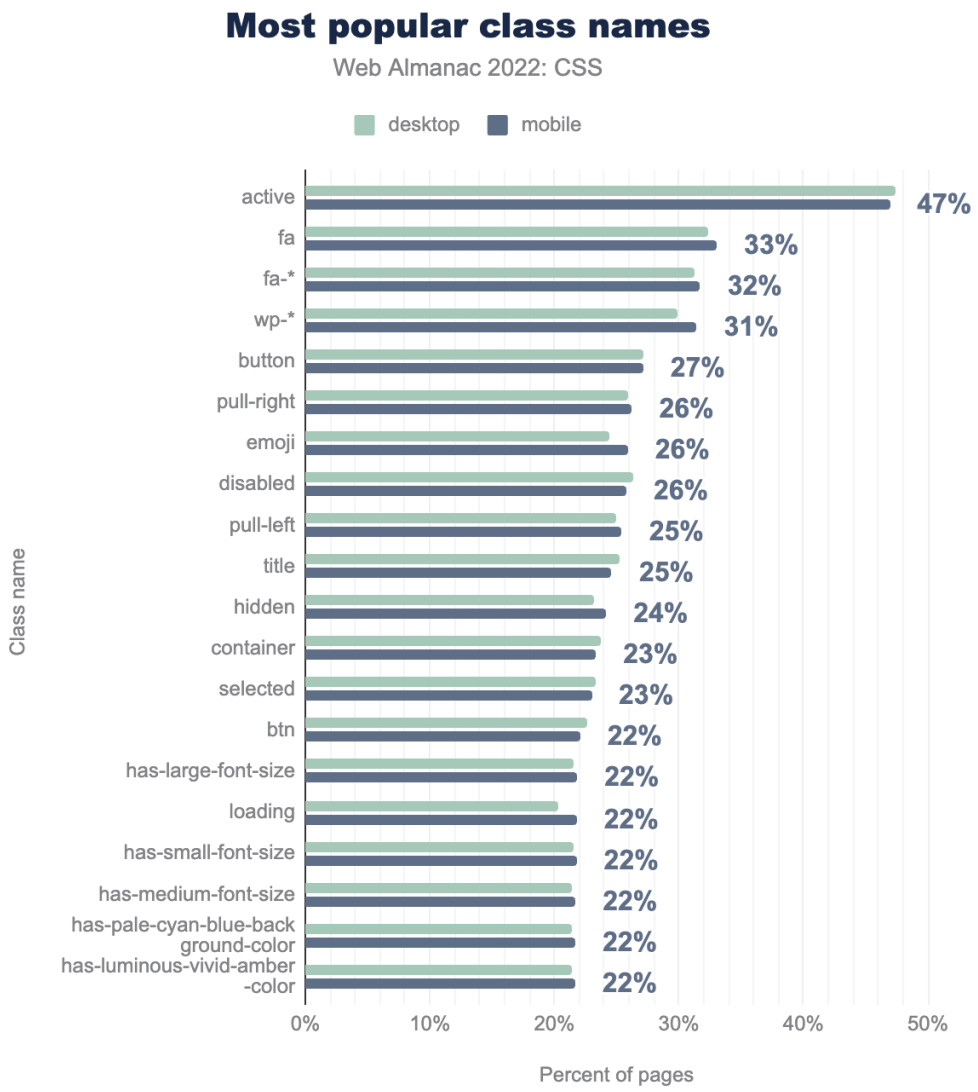

Figure 1.5. The most popular class names by the percent of pages.

As in 2020 and 2021 the most popular class name on the web is active. The fa, fa-* prefixes for Font Awesome still coming second and third. However, wp-* class names have crept up the rankings, moving to fourth place. They now show up on 31% of pages, having been at 20% in 2021. We also see class names such as has-large-font-size appearing, these are used in the new WordPress Block Editor.

Clearfix has disappeared from the top 20, it is now found on only 10% of pages, a very clear indication that float-based layouts are vanishing from the web.

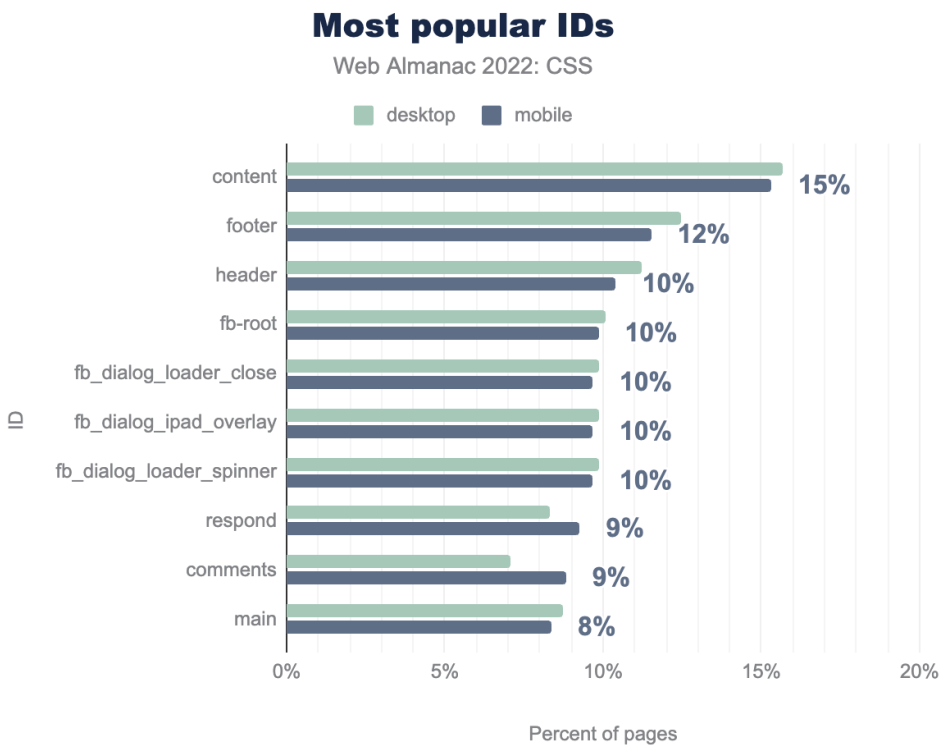

Figure 1.6. The most popular ID names by percent of pages.

The name `content` is once again the most popular ID name, followed by `footer`, and `header`. The IDs starting with `fb_` indicate use of Facebook widgets. In 2021 IDs beginning with `rc-`, indicating use of Google's reCAPTCHA system were seen on 7% of pages, and are still seen with the same frequency, despite being pushed out of the top ten by the Facebook ID names.

!important

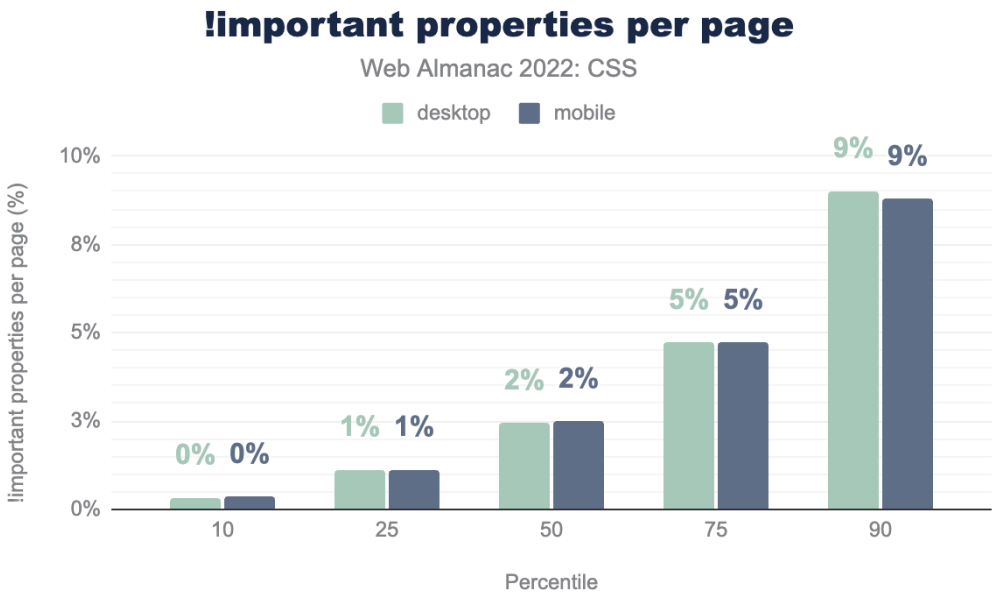

Figure 1.7. The distribution of the number of !important properties per page.

The use of `!important` has slightly increased for the top two percentiles this year. As `@layer` usage takes hold it will be interesting to see how this impacts the use of this property, typically used to deal with specificity issues.

In terms of what `!important` is applied to, the top properties remain unchanged. However, `position` has fallen out of the top ten, to be replaced with `font-size`.

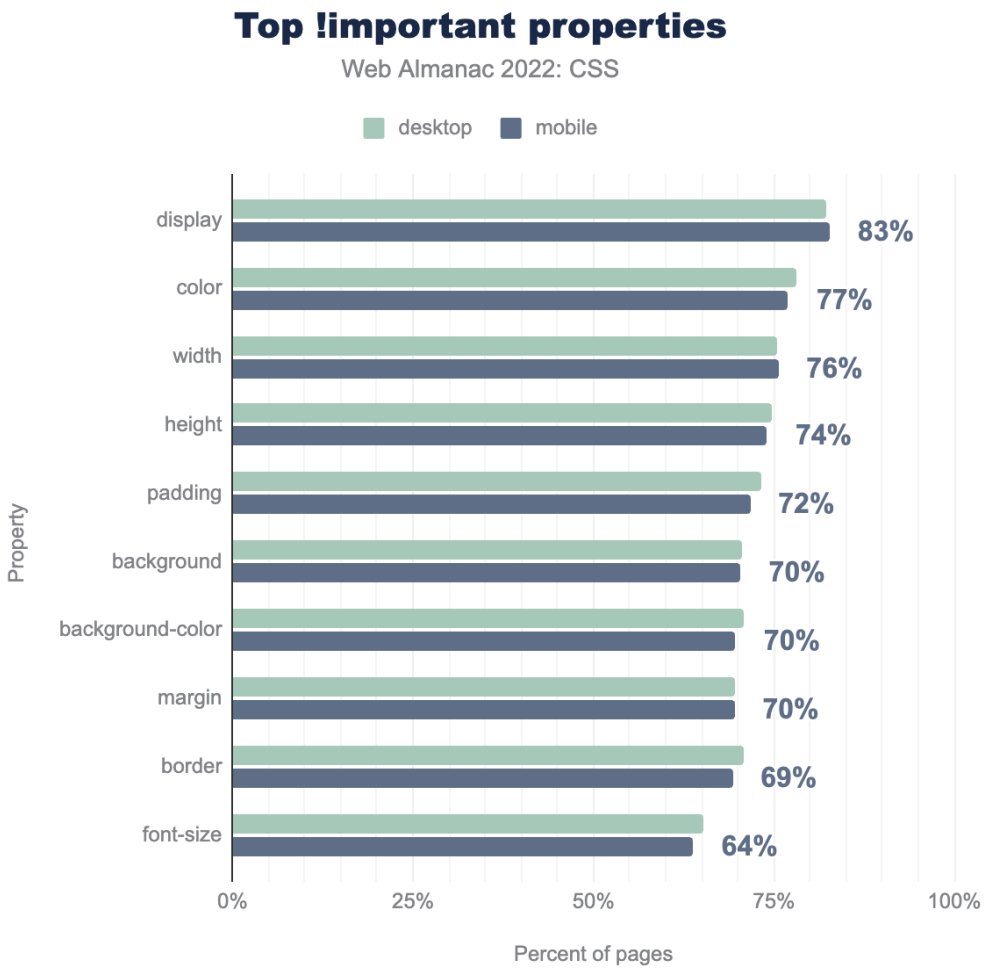

Figure 1.8. The top properties that !important is applied to by percent of pages.

Selector specificity

Percentile	Desktop	Mobile
10	0,1,0	0,1,0
25	0,1,2	0,1,3
50	0,2,0	0,2,0
75	0,2,0	0,2,0
90	0,3,0	0,3,0

Figure 1.9. Distribution of the median specificity per page.

Except for desktop at the 25th percentile, median specificity values are exactly the same as last year, remaining constant over the past two years. These values indicate the flattened specificity created by methodologies such as BEM.

Pseudo-classes and -elements

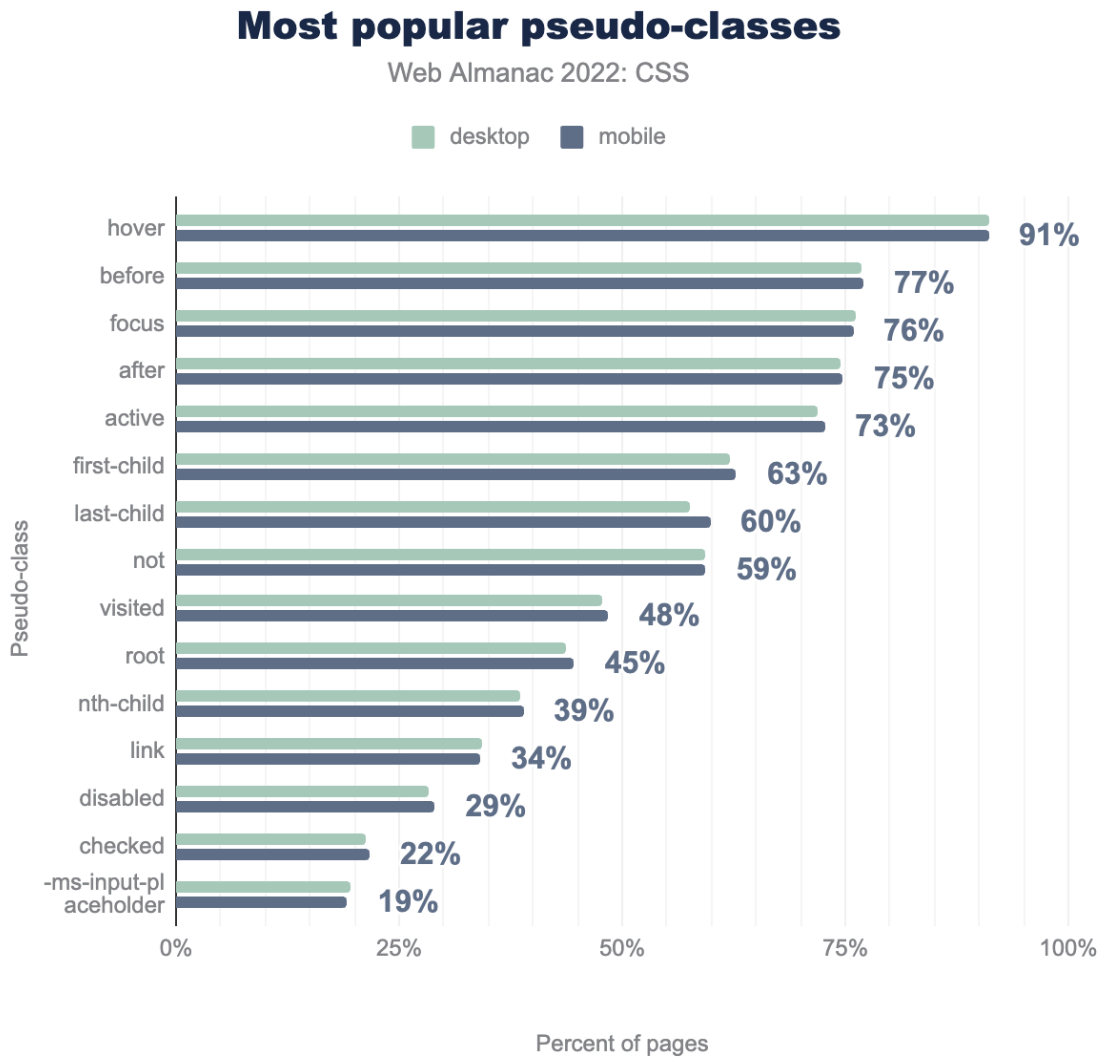

Figure 1.10. Most popular pseudo-classes by percent of pages.

Once again the user-action pseudo-classes `:hover`, `:focus`, and `:active` are in the top three spots. The negation pseudo-class `:not()` also continues its rise in popularity, along with `:root`, likely used to create custom properties.

Last year it was noted that `:focus-visible`, a way to style elements in focus in a way that better matches user expectations, appeared in less than 1% of pages. The property has been available in all three major engines since March 2022, and is now found on 10% of desktop and 9% of mobile pages.

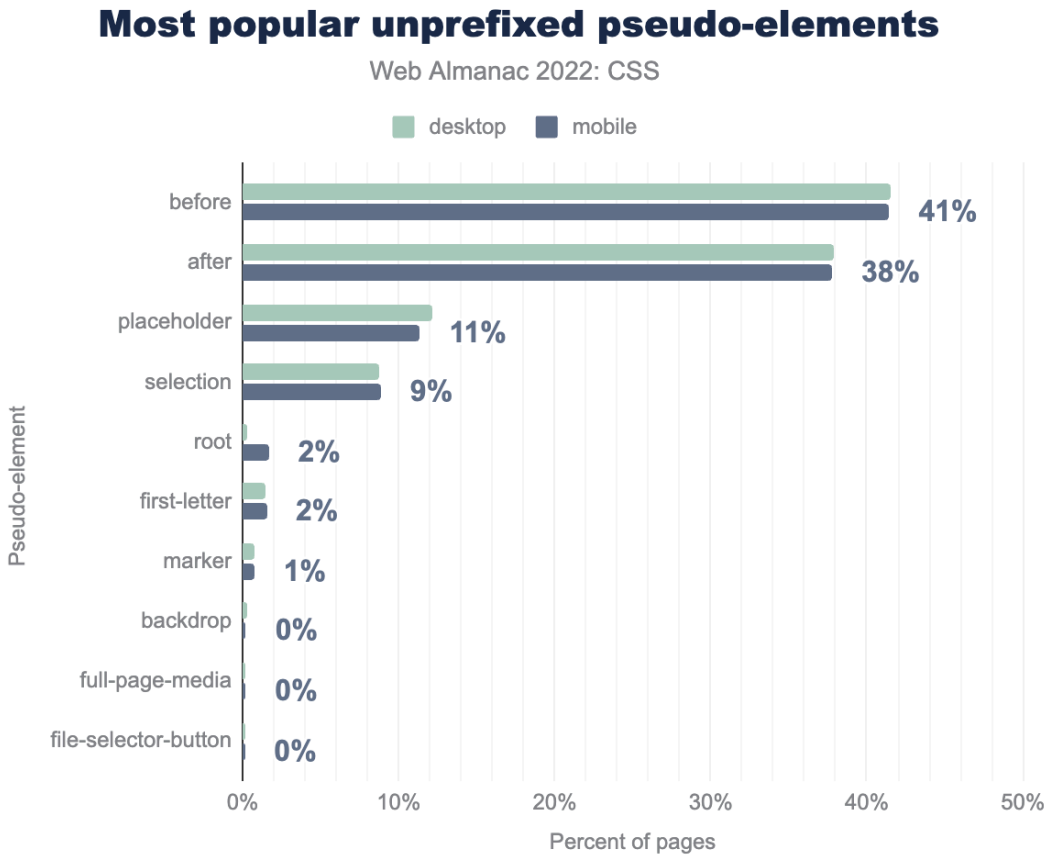

Figure 1.11. Most popular pseudo-elements by percent of pages.

We filter out any prefixed, and therefore browser-specific, pseudo-elements. These are typically used to select interface components or parts of browser chrome, and we are interested in the pseudo-elements developers are actually using.

The use of `::before` and `::after` has increased since last year. These are used to insert generated content into the document. By checking usage of the `content` property, it is possible to see that this is most often being used to insert an empty string, used for styling purposes. Generated content is one way to style a grid area without needing to add an element, perhaps this has contributed to the rise in usage of these properties?

Use of the `::marker` pseudo-element has now made 1%, showing that people are slowly starting to take advantage of the ability to select and style list markers.

Attribute selectors

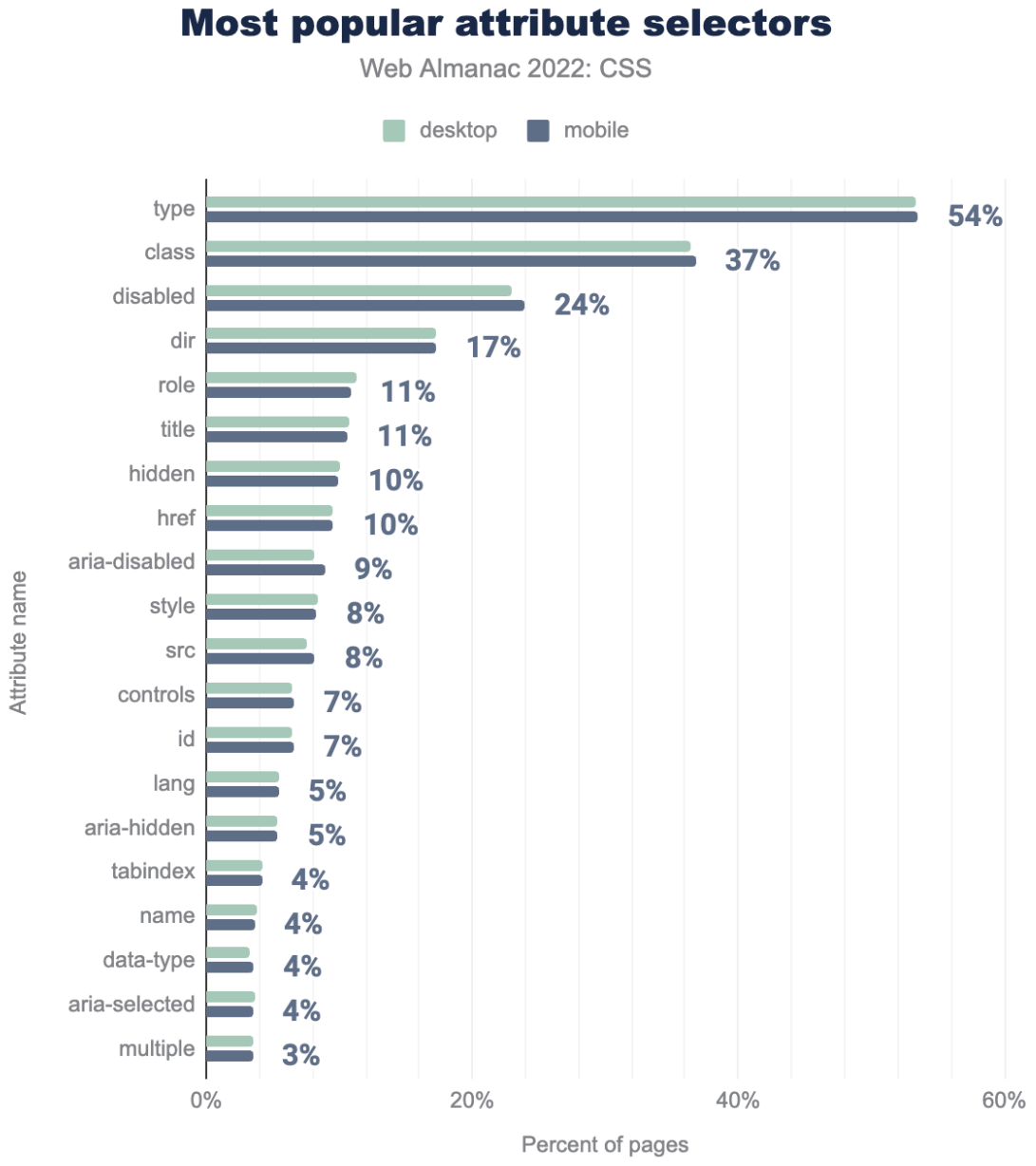

Figure 1.12. Most popular attribute selectors by percent of pages.

Values and Units

CSS provides multiple ways to specify values and units, either in set lengths, or calculations based on global keywords.

Length

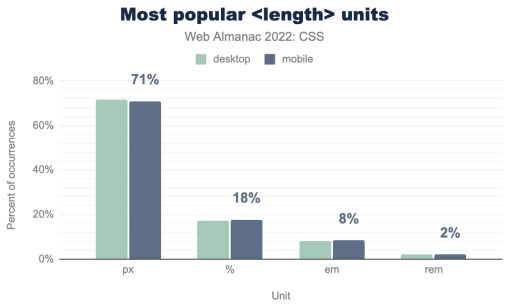

Figure 1.13. Most popular `<length>` units by percent of pages.

Pixel lengths remain the most popular at 71%, the same percentage as in 2021. The spread of usage remains roughly the same too.

Property	px	<number>	em	%	rem	pt
font-size	(up 1.9%) 70.89%	(down 0.1%) 1.90%	(down 0.8%) 15.24%	(down 0.2%) 4.84%	(up 0.6%) 5.62%	(down 0.5%) 1.51%
border-radius	(down 0.5%) 64.45%	(down 1.3%) 19.72%	(up 0.1%) 3.13%	(up 1.2%) 11.19%	(up 1.5%) 1.51%	
line-height	(down 5.4%) 48.58%	(up 4.1%) 35.09%	(down 0.1%) 12.94%	(down 0.8%0 2.22%	(up 1.2%) 1.17%	0.00%
border	(down 1%) 70.03%	(up 0.4%) 28.37%	(down 0.4%) 1.60%			
text-indent	(down 5.4%) 25.60%	(up 12.8%) 64.78%	(down 3.5%) 4.51%	(down 2.9%) 5.11%		
vertical-align	(down 26.2%) 2.84%	(down 9.2%) 2.78%	(up 39.4%) 94.38%			
gap	(up 4.4%) 25.42%	(down 5.8%) 10.19%	(up 31.9%) 32.89%		(down 30.5%)31.51%	
margin-inline-start	(down 30.9%) 7.07%	(up 2.6%) 48.61%	(up 30.3%) 44.32%			
grid-gap	(up 5.3%) 68.30%	(down 1.4%) 9.65%	(down 2%) 6.97%		(down 0.9%) 15.08%	
margin-block-end	(down 1.2%) 2.79%	(up 53.9%) 84.94%	(down 52.7%) 12.27%			
padding-inline-start	(down 4.4%) 28.63%	(up 11.2%) 16.20%	(down 9.5%) 52.53%		(up 2.6%) 2.64%	
mask-position	(up 1.4%) 1.42%	(up 2.9%) 2.92%	(down 14.1%) 35.93%	(up 9.7%) 59.73%		

Figure 1.14. Distribution of length types per property.

The up and down arrows on this chart show the change from the results in 2021. As seen last year, in the majority of cases there is a shift away from using pixels, in favor of other length units. Once again, the `vertical-align` property saw a huge drop in pixel and `<number>` use, and a large rise in em use.

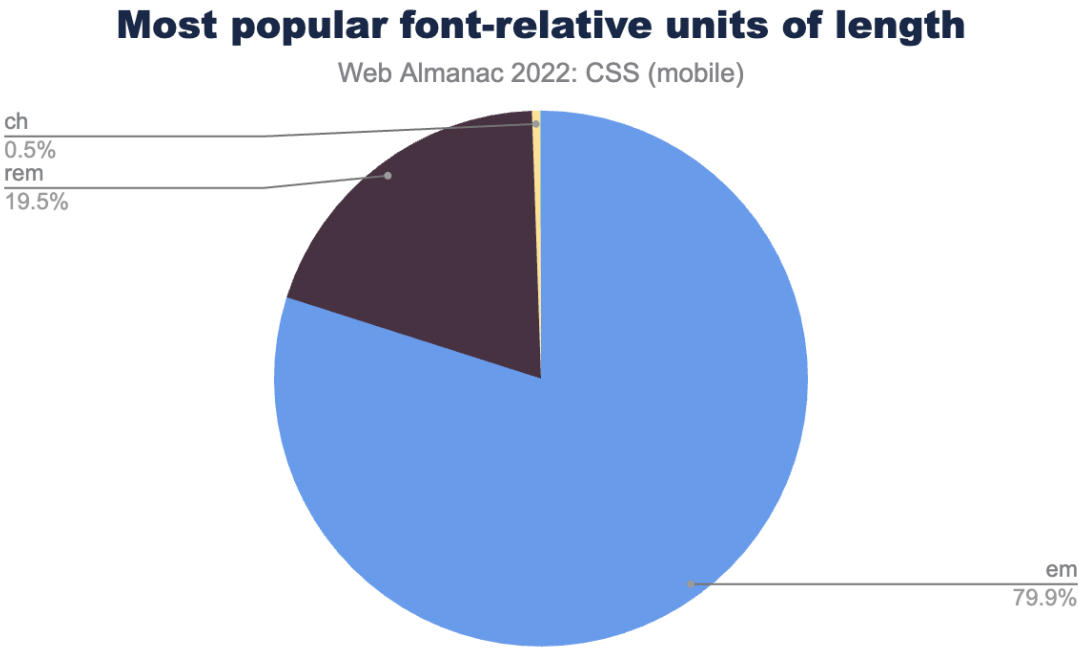

Figure 1.15. The most popular font-relative length units.

While em remains the most popular method of sizing fonts, the swing to rem continues with a small (just under two point) increase over last year.

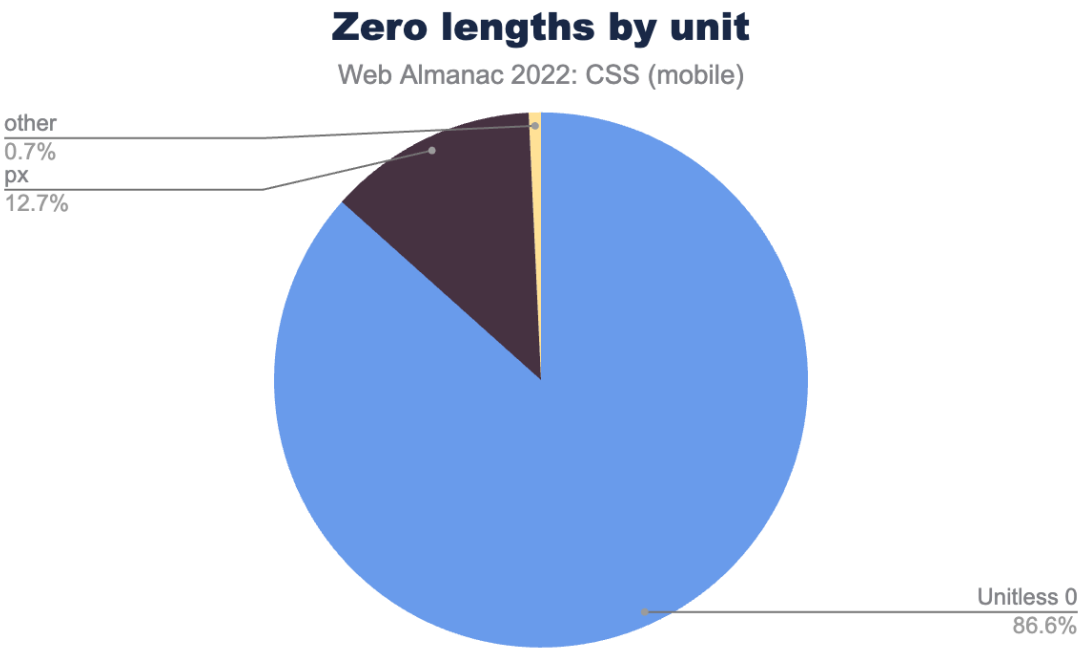

Figure 1.16. The units (or lack thereof) used on zero-length values.

There are a few properties that allow bare `<number>` units (for example, `line-height`), but `<length>` values have a special case where a length of zero does not require a unit. When we looked at all zero-length values, almost 87% of them omitted the unit, this is a small decrease from last year. Nearly all of those zero lengths that included a unit used pixels (0px).

Calculations

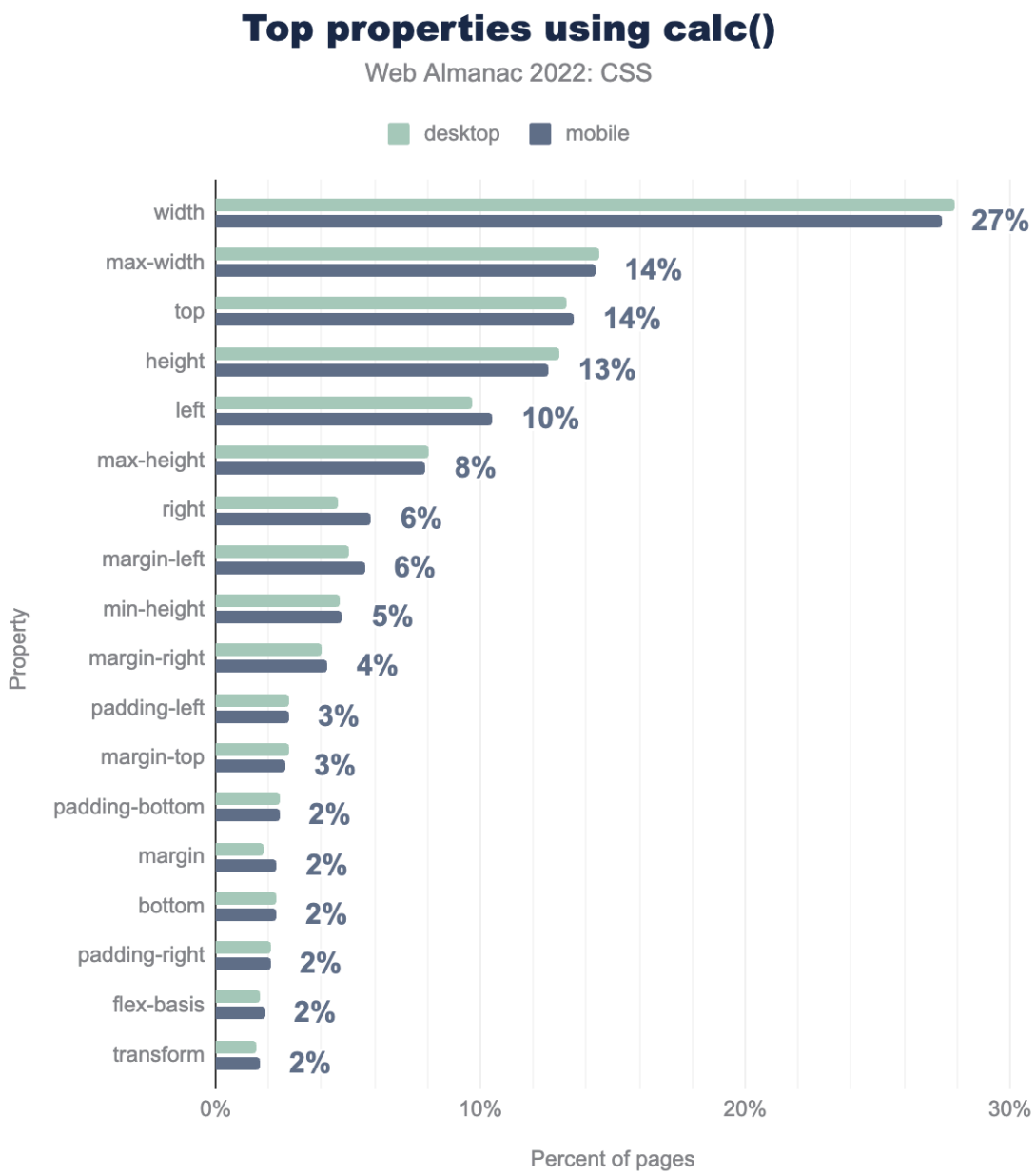

Figure 1.17. The most popular properties using `calc()` functions.

As in previous years, the most popular use of `calc()` is in values for width. This use has dropped 12% points, however, `max-width` has increased in popularity by 9 points.

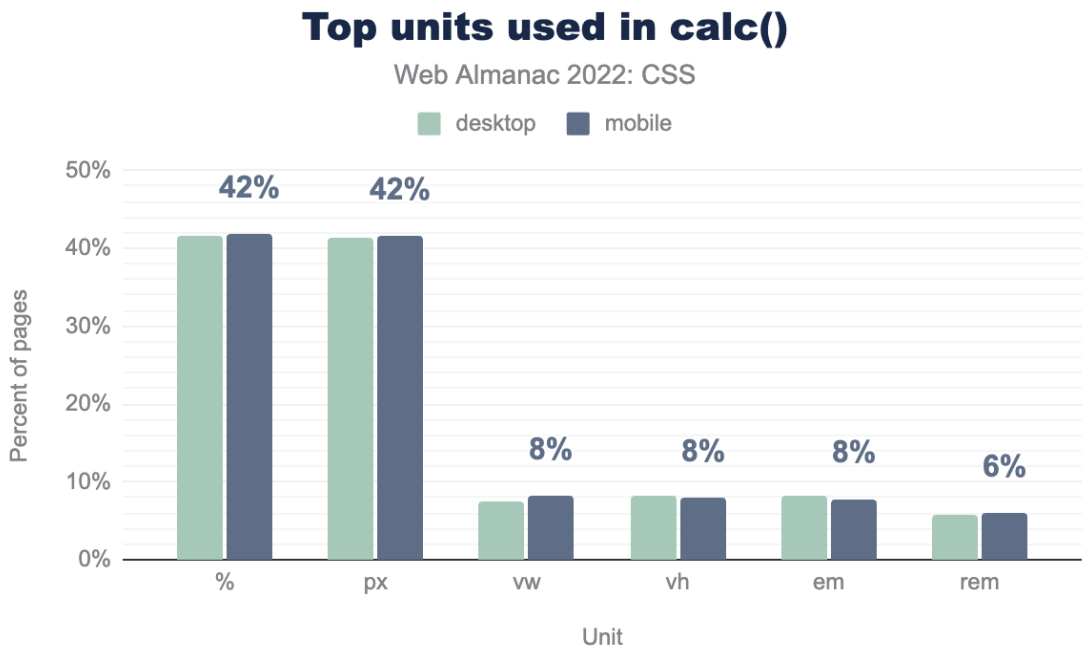

Figure 1.18. The most popular length units used in `calc()` functions.

The percentage of sites using pixels in calculations has decreased 9 points, it is now level with percentages at 42%. There is a significant increase in usage for other values, the viewport units vw and vh both increased from 2% to 8% this year, em increased the same amount, and use of rem doubled from 3% to 6%.

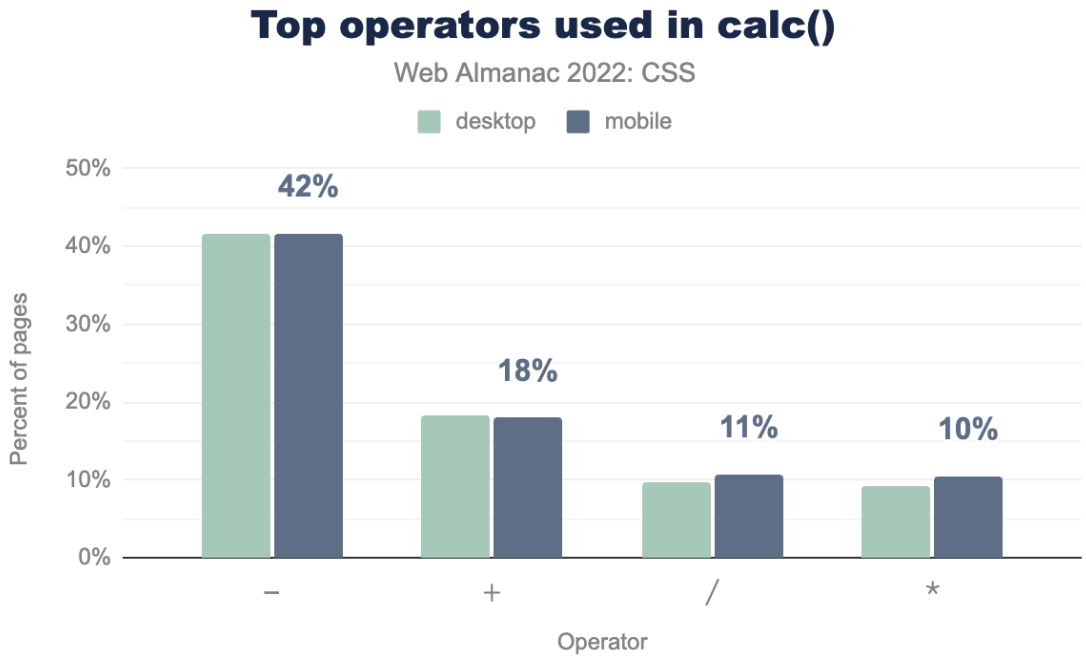

Figure 1.19. The most popular operators used in `calc()` functions.

Subtraction remains the clear favorite in terms of calculation operators, but all four top values saw a drop since 2021, other than addition, which remained the same.

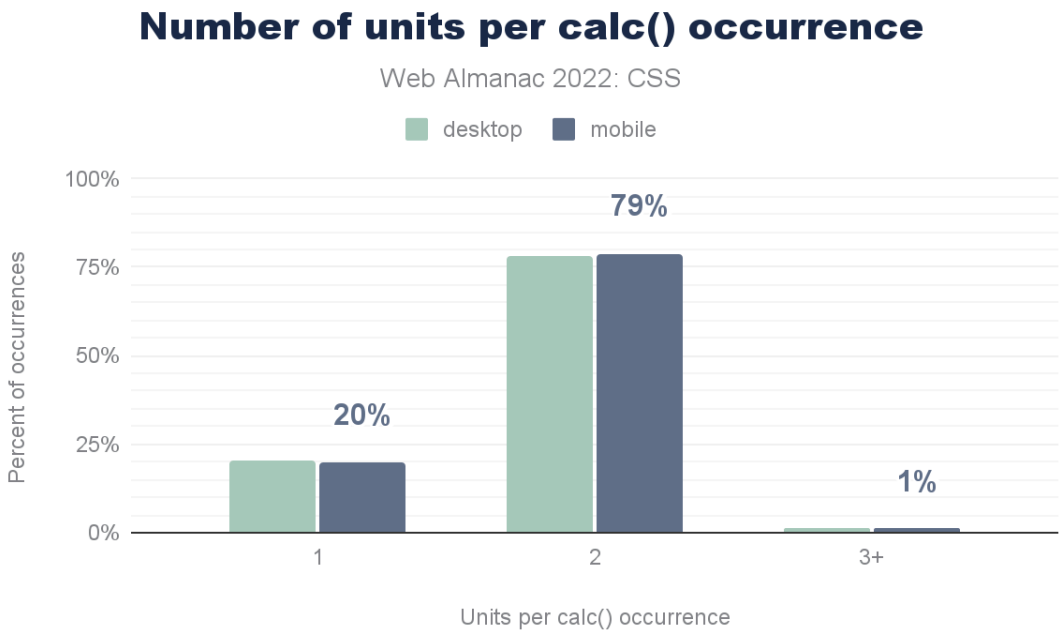

Figure 1.20. The number of unique units used in `calc()` values.

As last year, `calc()` values tend to be fairly simple. The majority using two values, such as the common use case of subtracting a fixed length such as pixels from a percentage. There was a small rise in one unit values, and a small drop in two units.

Global keywords

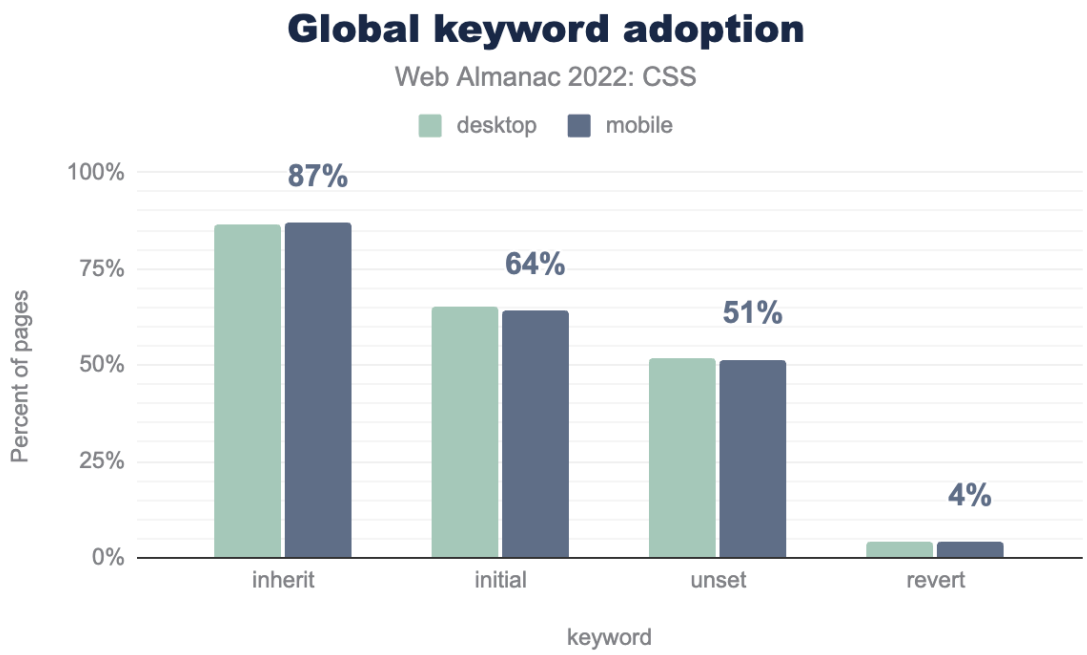

Figure 1.22. Usage of global keyword values.

Last year the use of global keywords had risen significantly, in 2022 inherit is found in the same percentage of pages, however the other three values have increased in use. The newer value of revert has increased from 1% to 4%.

Custom Properties

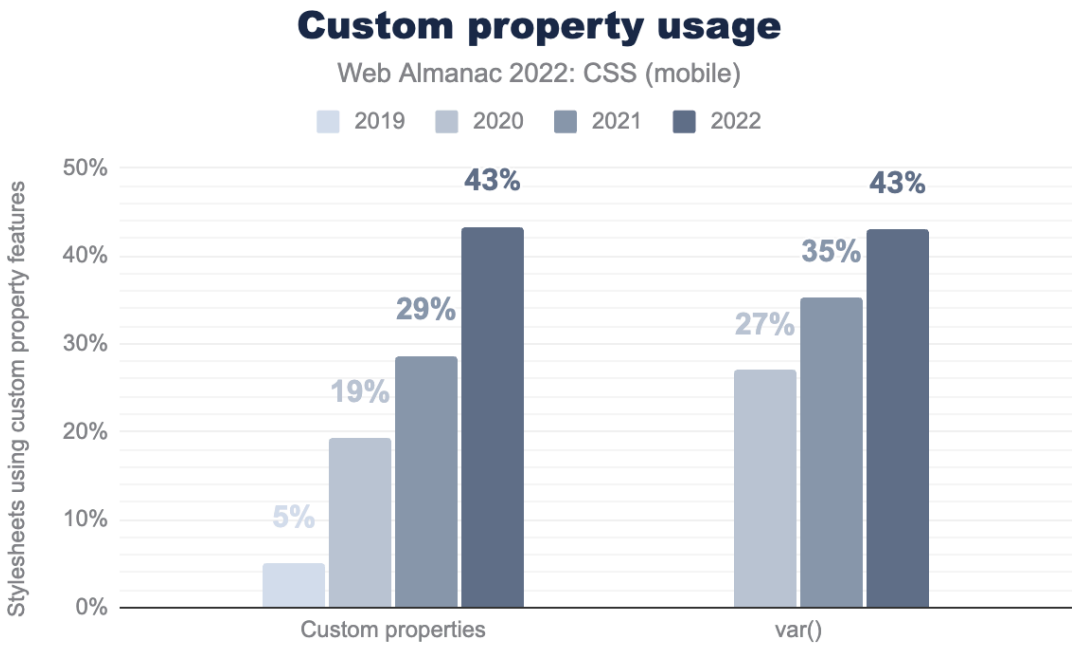

Figure 1.23. Usage of custom properties over the past four years.

Custom properties (sometimes known as CSS variables) have seen a huge surge in use, the growth between 2021 and 2022 is no exception. 43% of pages, for both desktop and mobile are using custom properties and have at least one var() function.

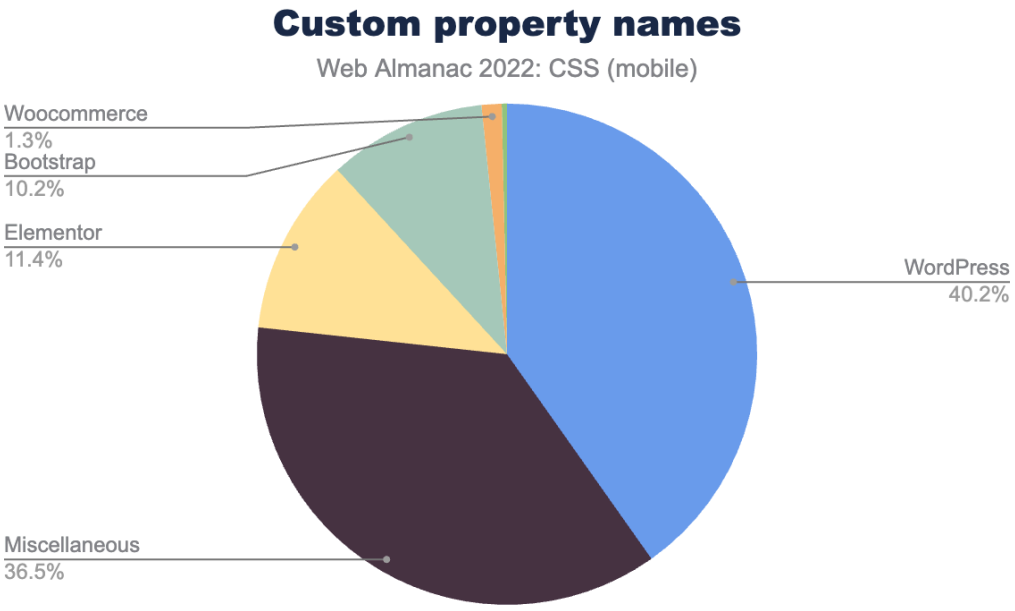

Figure 1.24. Source of common custom property names.

As seen last year, WordPress is the driver for the most common custom property names, these are easily identifiable by the `–wp–` prefix. Following these we once again found a lot of color names `–white`, `–blue`, and so on, used to assign a particular shade of that color.

Types

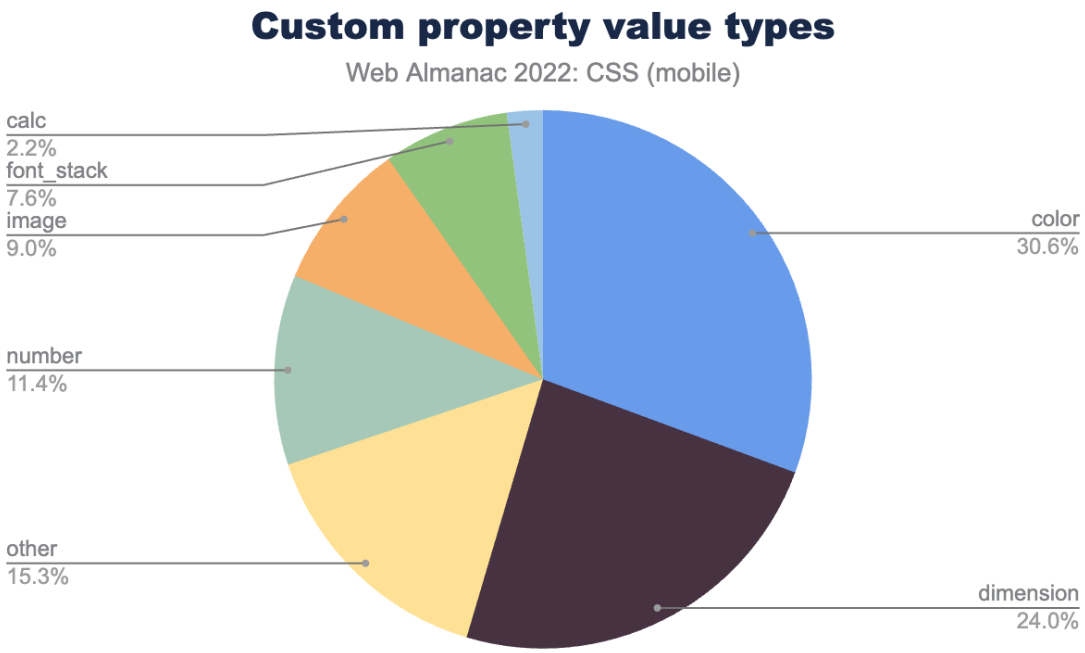

Figure 1.25. Distribution of custom property value types.

The value of a custom property includes a type. For example, `--red: #EF2143` is assigning a color value to `--red`, whereas `--multiplier: 2.5` is assigning a number value. The types have changed a little since last year. While we know that setting a color is common use of custom properties, and the amount of pages where color types are found are increasing, in terms of the share of usage this has dropped from 40% to 30%. Entering this distribution is `calc()`, and images as a value type.

Properties

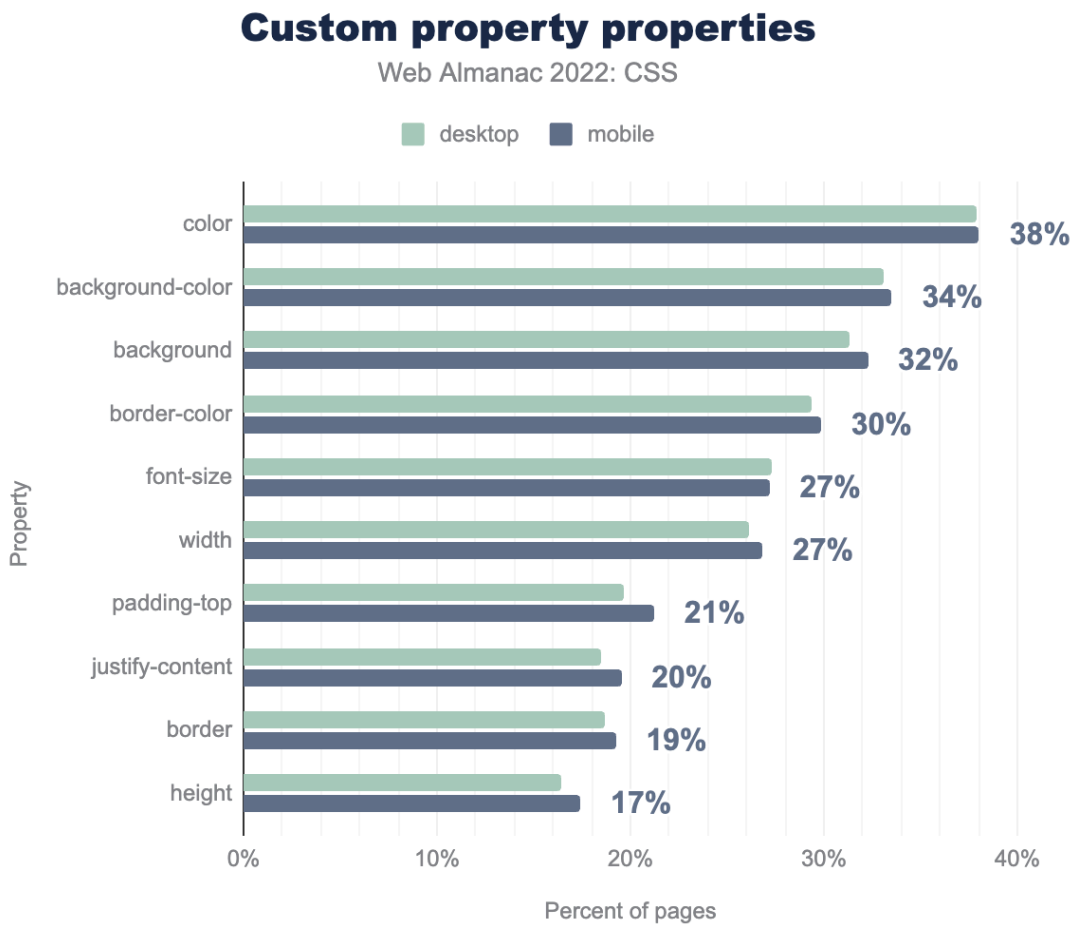

Figure 1.26. The most popular custom property properties by percent of pages.

While the number of pages including these properties has increased, the properties that have custom properties as a value have remained in roughly the same order as last year. Custom properties are most likely to be used for `color`, unsurprisingly as creating color schemes is an obvious use of this functionality. Using the `var()` function to set `font-size` has moved from 10th place to 5th in the list however, and setting the alignment value of justify-content has moved into the top ten. In 2021 5% of mobile, and 4% of desktop pages were using custom properties to set this alignment value, this has jumped to 20%. From the data it looks as if some of this increase is due to WordPress usage, 5% of pages use the `–navigation-layout-justify` custom property, for example.

Functions

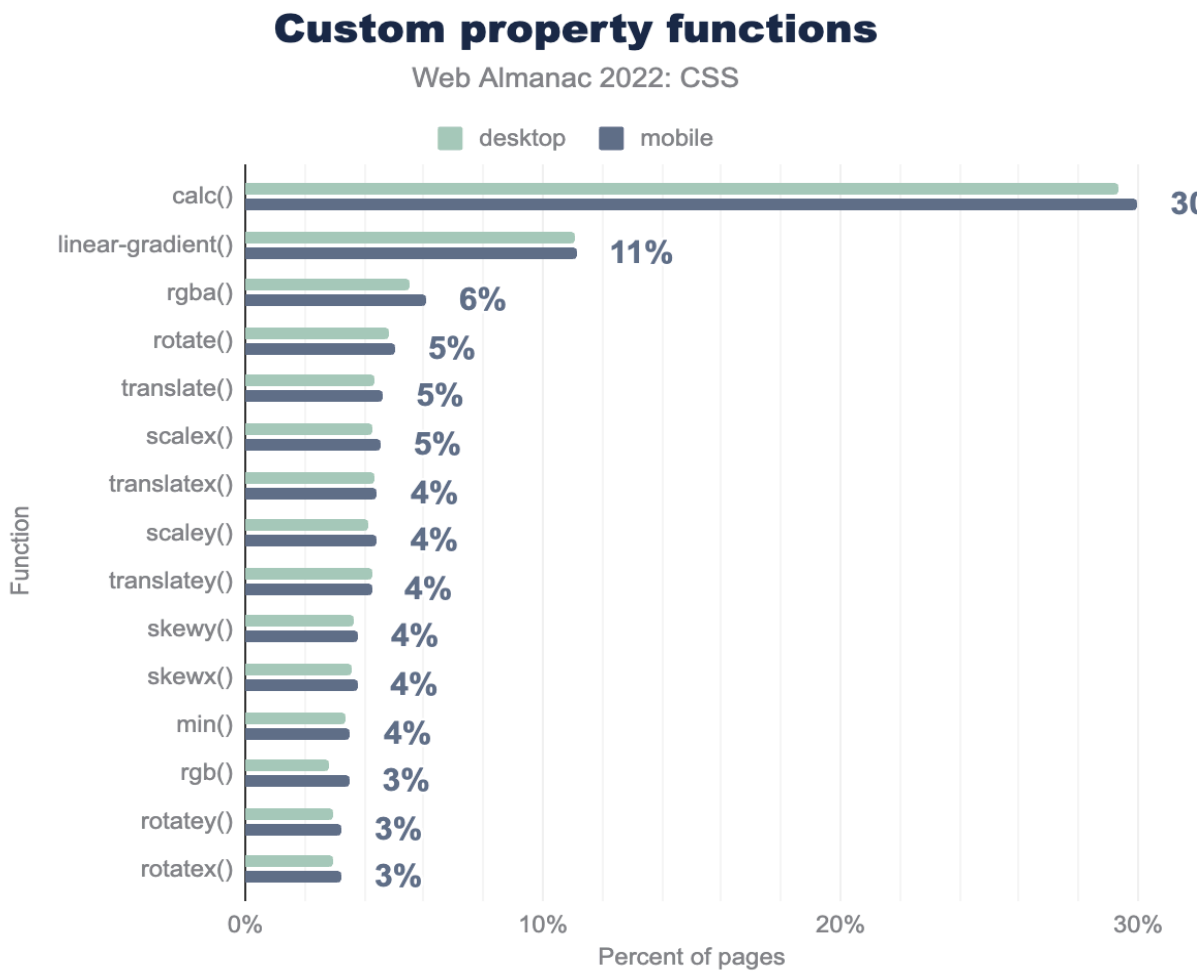

Figure 1.27. The most popular custom property functions by percent of pages.

We saw that `calc()` has started to be notable as a value type for custom properties, and it is by far the most commonly seen function used in this way. It is followed by `linear-gradient()`, and the `rgba()` function used to set RGB color values with an alpha channel. After this are the various functions used for transitions and animations, showing a growing use of custom properties in this area.

Complexity

It's possible to include custom properties in the values of other custom properties. Consider this example from the 2020 Web Almanac:

```
:root {
  --base-hue: 335; /* depth = 0 */
  --base-color: hsl(var(--base-hue) 90% 50%); /* depth = 1 */
  --background: linear-gradient(var(--base-color), black); /* depth = 2 */
```

}

As the comments in the previous example show, the more of these sub-references are chained together, the greater the depth of the custom property.

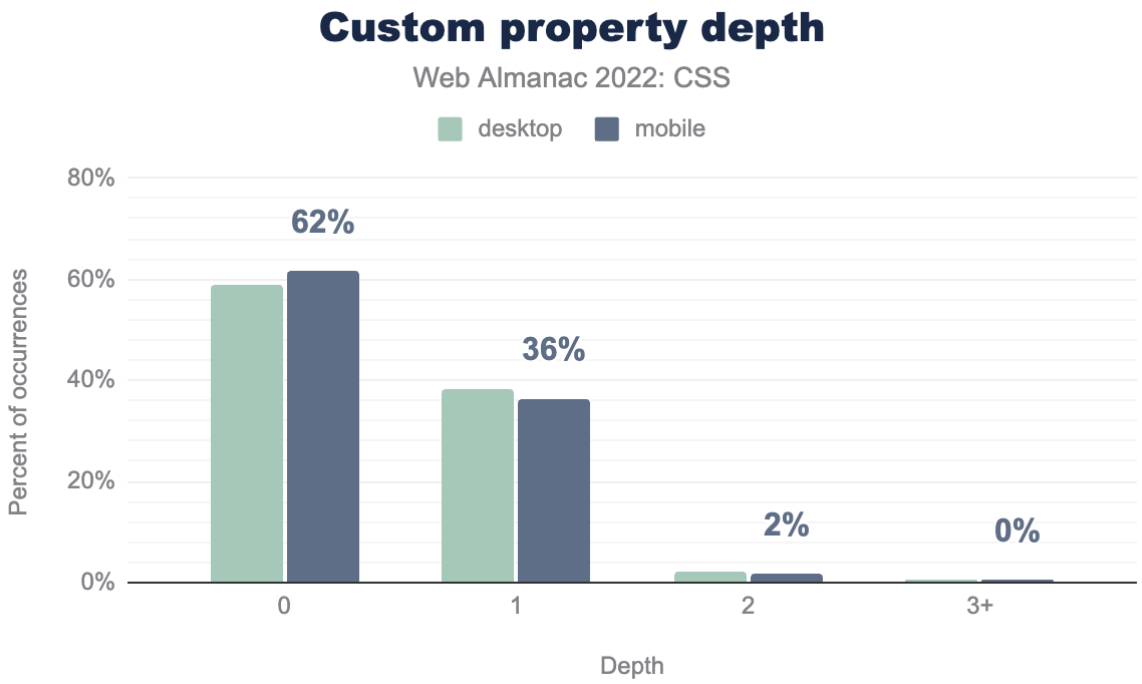

Figure 1.28. The distribution of custom property depth.

As seen in 2021 the vast majority of custom properties had a depth of zero, meaning that they did not include the values of other custom properties in their value. There has been a small increase in the number of properties with a depth of one, and a small decrease in the number with a depth of two. However, it does not seem from the data that our use of custom properties has become much more complex in the past year.

Colors

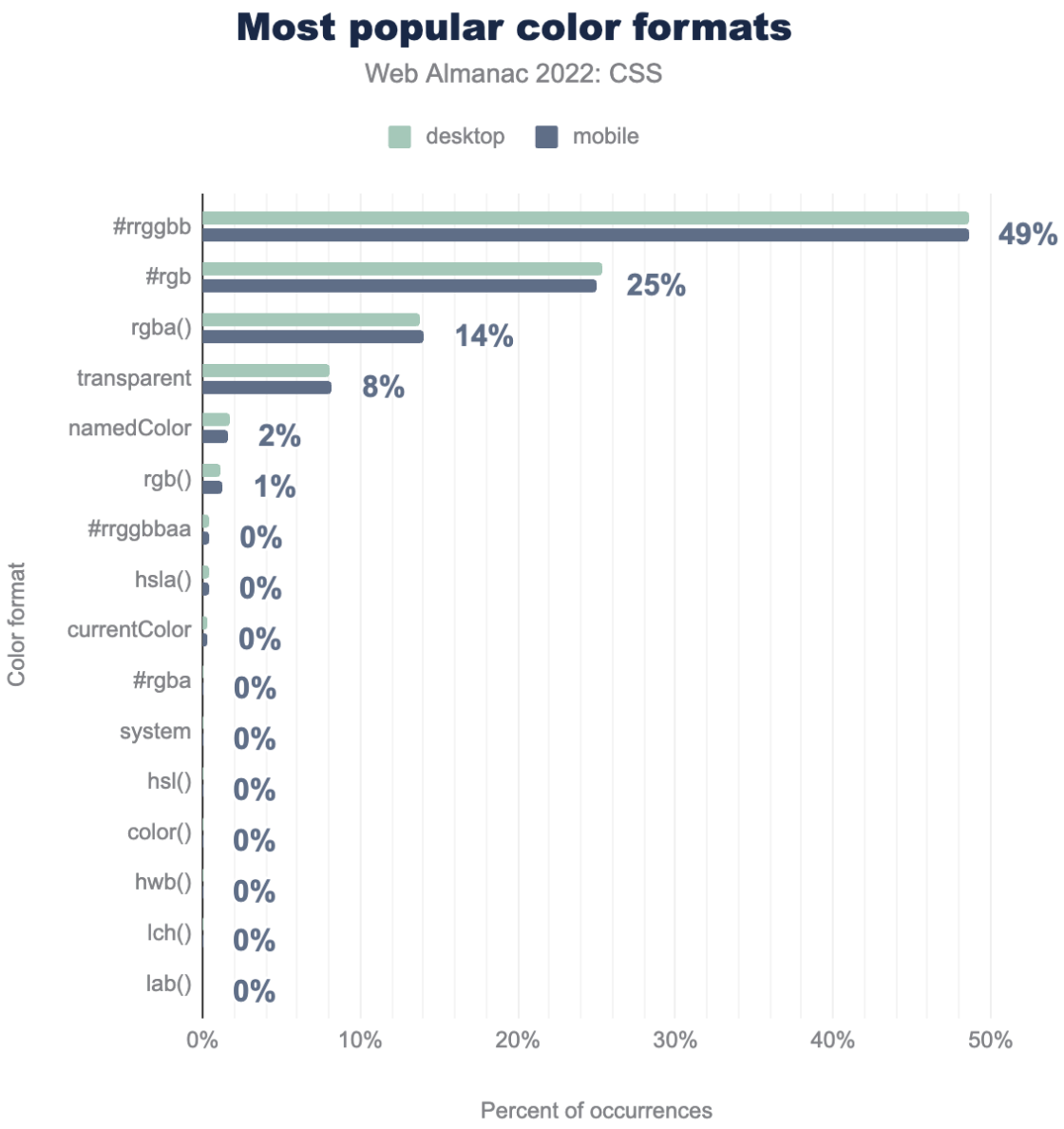

Figure 1.29. The most popular color formats by percent of occurrences.

The use of the time-honored six-digit #RRGGBB syntax remains unchanged since 2021, being used in half of color declarations. Despite the widespread availability of eight-digit #RRGGBBAA hex, the `rgba()` form is the most widely used way to add an alpha component, likely because it was implemented in browsers much earlier.

The usage of other values showed a similar story, the web community hasn't yet started to take advantage of other color formats, even widely supported ones such as `hsl()`.

8% of pages use the keyword transparent, making it the most popular named color. 2% of pages use other named colors, white being the most popular followed by black, at the other end of the scale mediumspringgreen languishes as the least popular color.

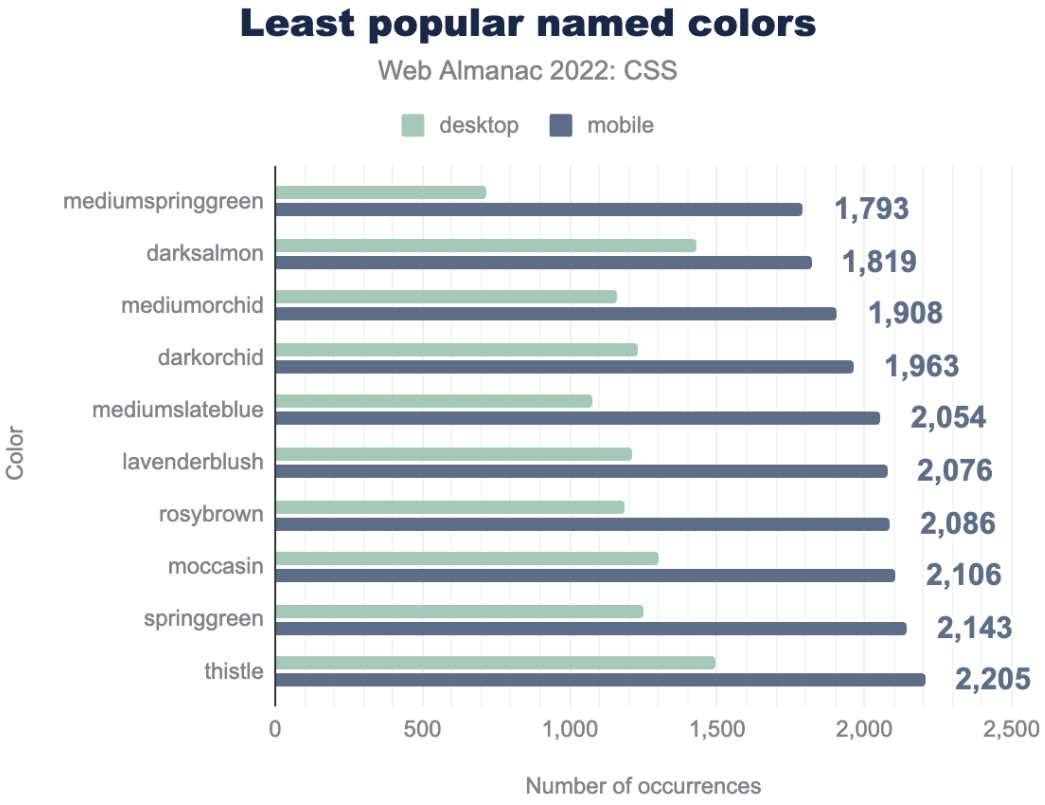

Figure 1.30. The least popular named colors by number of occurrences.

Alpha support and use

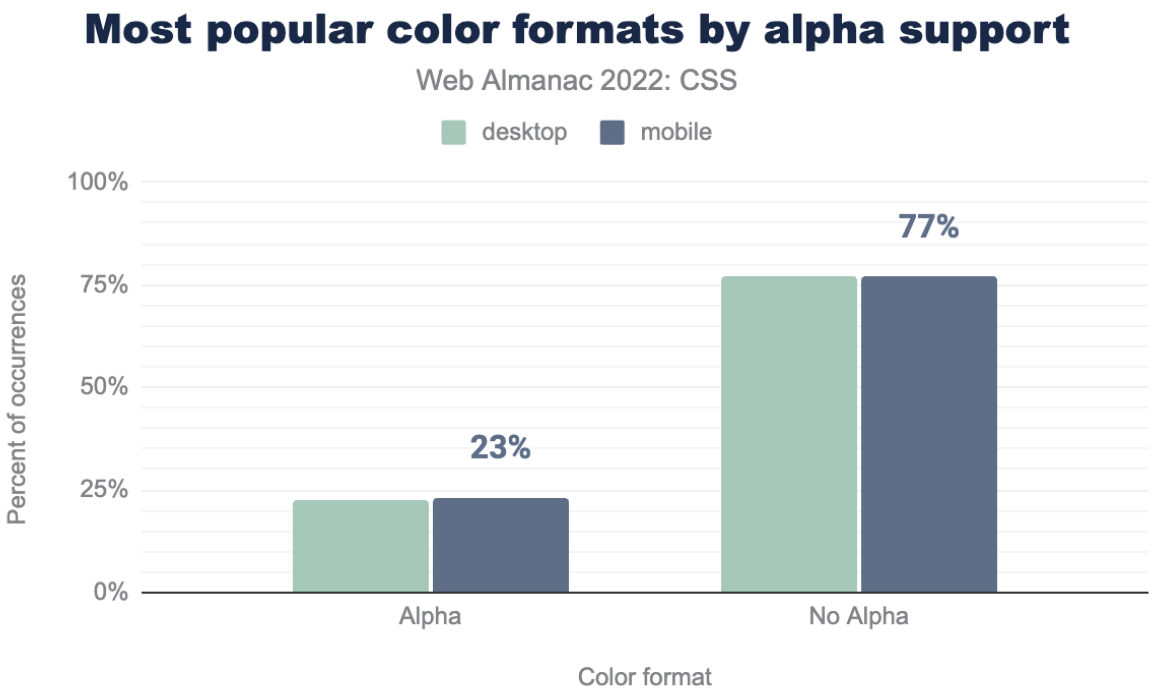

Figure 1.31. The most popular color formats by alpha support.

The `rgba()` function is the third most popular color format, used substantially more than the `rgb()` form, presumably in order to make use of alpha channel support. We looked at the occurrences of values with and without alpha support, to find that 77% of color formats used do not have support for an alpha channel.

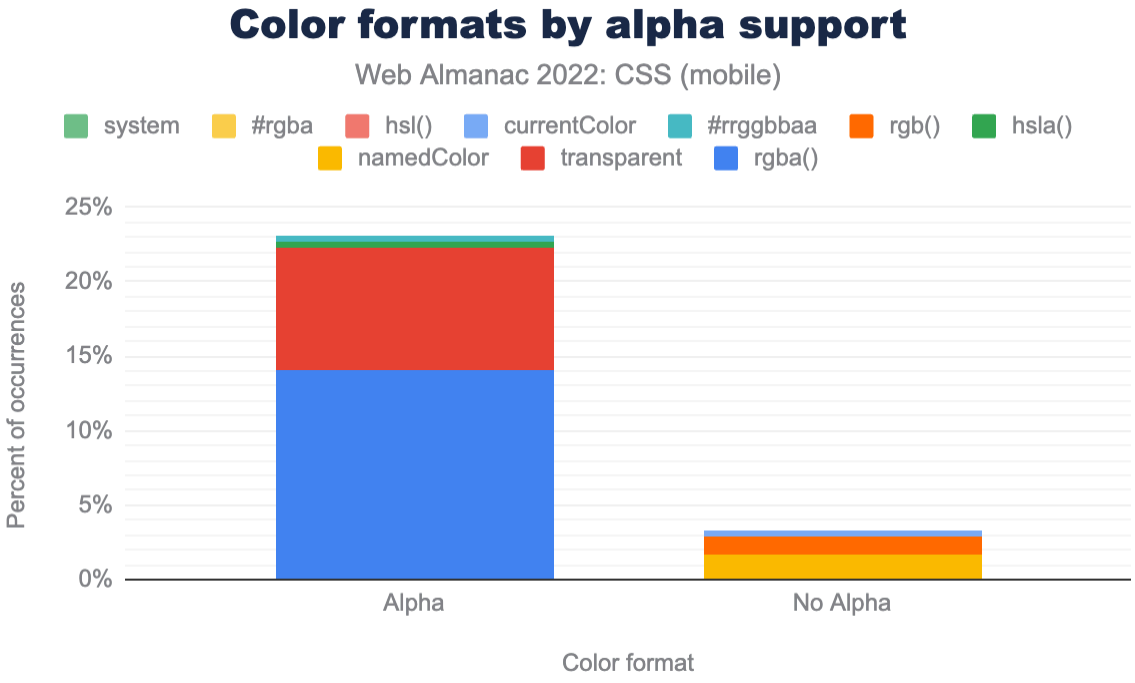

Figure 1.32. Distribution of color formats by alpha support.

As we would expect from other data, rgba() is the most popular alpha-supporting format in use, followed by the transparent keyword. Other formats such as hsla() barely feature.

New color properties and values

There are interesting things happening in the world of color. In addition to new color spaces, we have a number of color-related properties and values. We wondered if any of these were making an impact on the data.

The accent-color property lets you add your brand color as an accent color to notoriously hard to style form elements such as checkbox, radio buttons, and range sliders. Perhaps due to the fact it has only been available in all engines since March this year, it still shows less than 0.3% usage.

Another property becoming available in all engines this year is color-scheme, a property that lets you specify which color schemes (light or dark) a component can be rendered in. This property is, somewhat surprisingly, so far only found in 0.2% of pages.

Gradients and Images

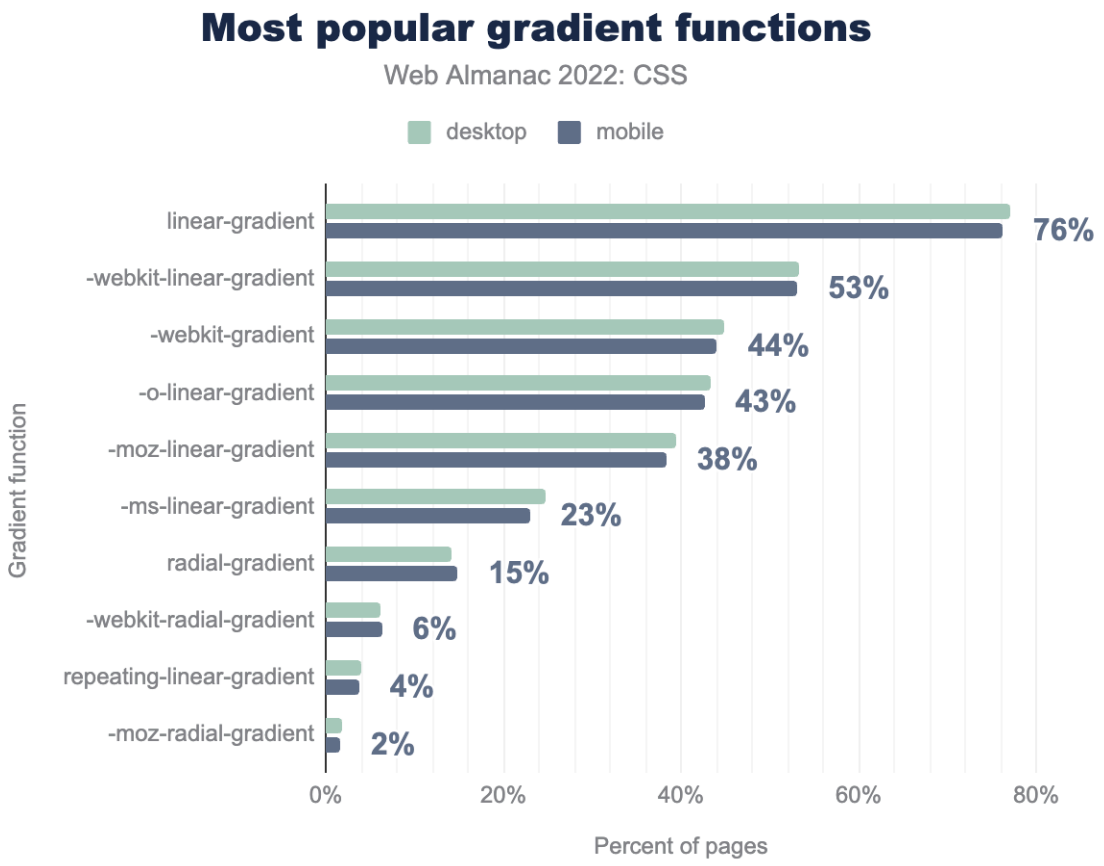

Figure 1.33. The most popular gradient functions by percent of pages.

Linear gradients continue as the leading choice, appearing on a slightly higher percentage of pages than in 2021, however gradient use stays pretty much the same for the last two years. There is still a very high frequency of prefix use when it comes to the `linear-gradient` property, despite this having been supported unprefixed in all engines for over nine years.

Image formats

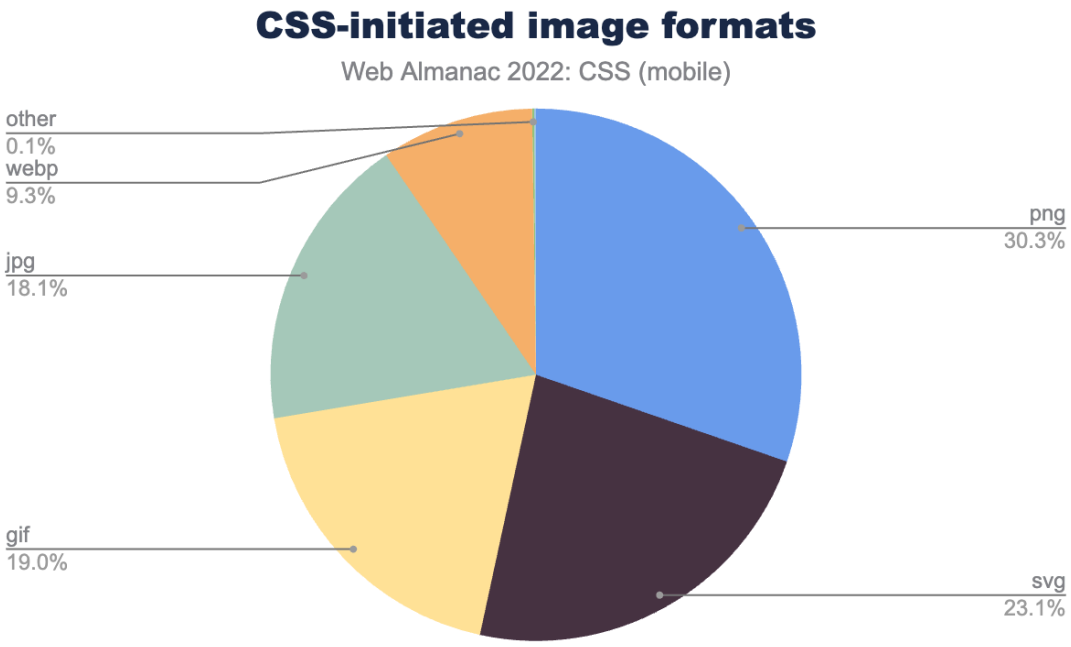

Figure 1.34. Image formats as loaded from CSS.

This chart breaks down the image formats of images loaded from CSS. It does not include images loaded from HTML, just those that appear in a style rule. There has been a significant swing away from PNG (down from 44% to 30%), with SVG and WEBP each seeing an increase of 6 percentage points.

Number of images in CSS

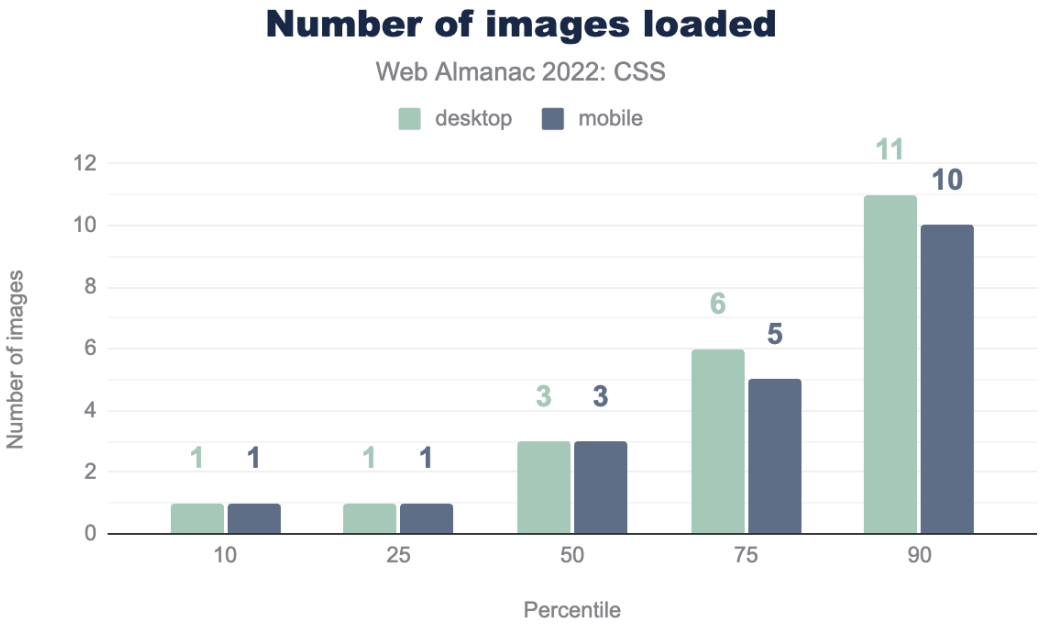

Figure 1.35. Distribution of number of images loaded from CSS.

The number of images loaded from CSS remains the same as in 2021. CSS doesn't cause many image loads: the lower two percentiles came in at one image each, and even the 90th percentile hovered around 10 images, across all image types.

Weight of images in CSS

While CSS doesn't cause many image loads, the weight of those images is important. The data showed that image weight has increased from 2021, despite the fact that the number of images has stayed the same.

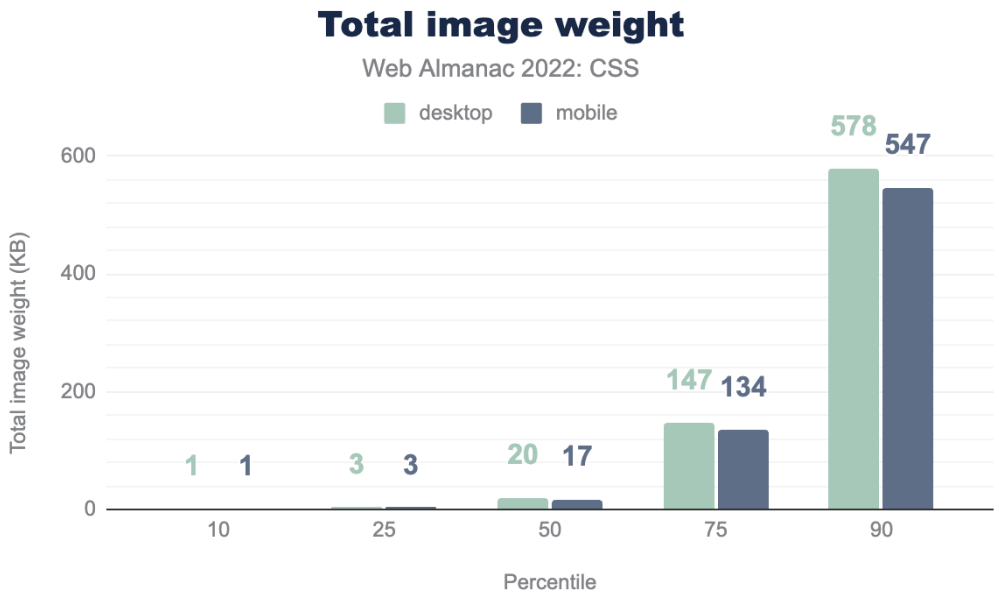

Figure 1.36. Distribution of total weight of images loaded from CSS.

The median page, on mobile, has increased image weight by 1KB to 17KB. At the upper end of the chart however, at the 90th percentile we see an increase of 67KB on mobile and 42KB on desktop. As in 2021, the weight on mobile is consistently lower on mobile, an indication that developers are trying to serve smaller images to mobile contexts.

Pixel size of images in CSS

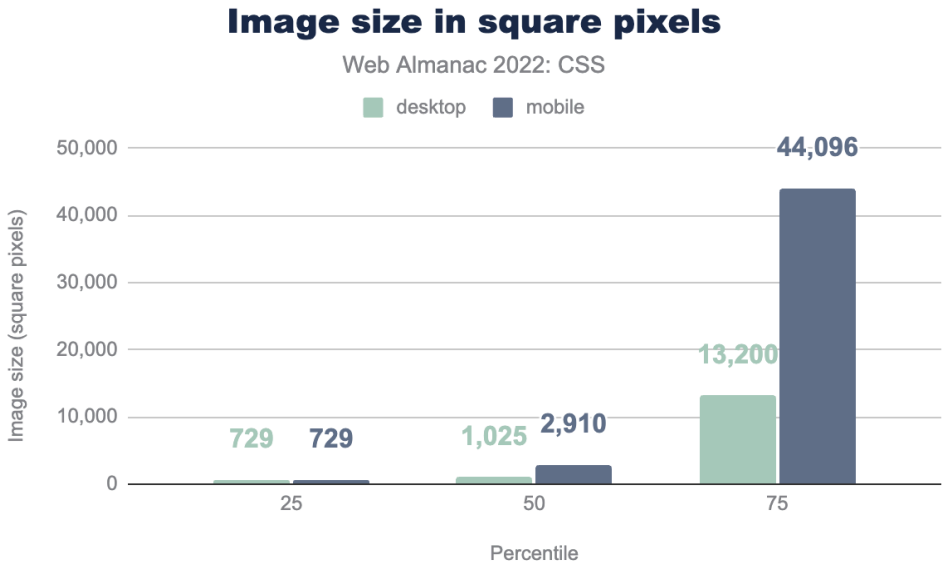

Figure 1.37. Distribution of sizes of images loaded from CSS.

This is an interesting chart which shows that at the lower end of the chart people are serving images of around the same size to desktop and mobile, at the 50th and 75th percentile pages are serving far larger images to their mobile users than they do to desktop. What the data shows is that people are serving much wider images to their mobile users, perhaps to try to account for tablets in landscape mode.

Layout

We have many options to choose from when doing layout on the web, and most sites will be using a variety of these methods. A simple search of the data, looking for property and value combinations to detect layout methods in use, gives us the following table.

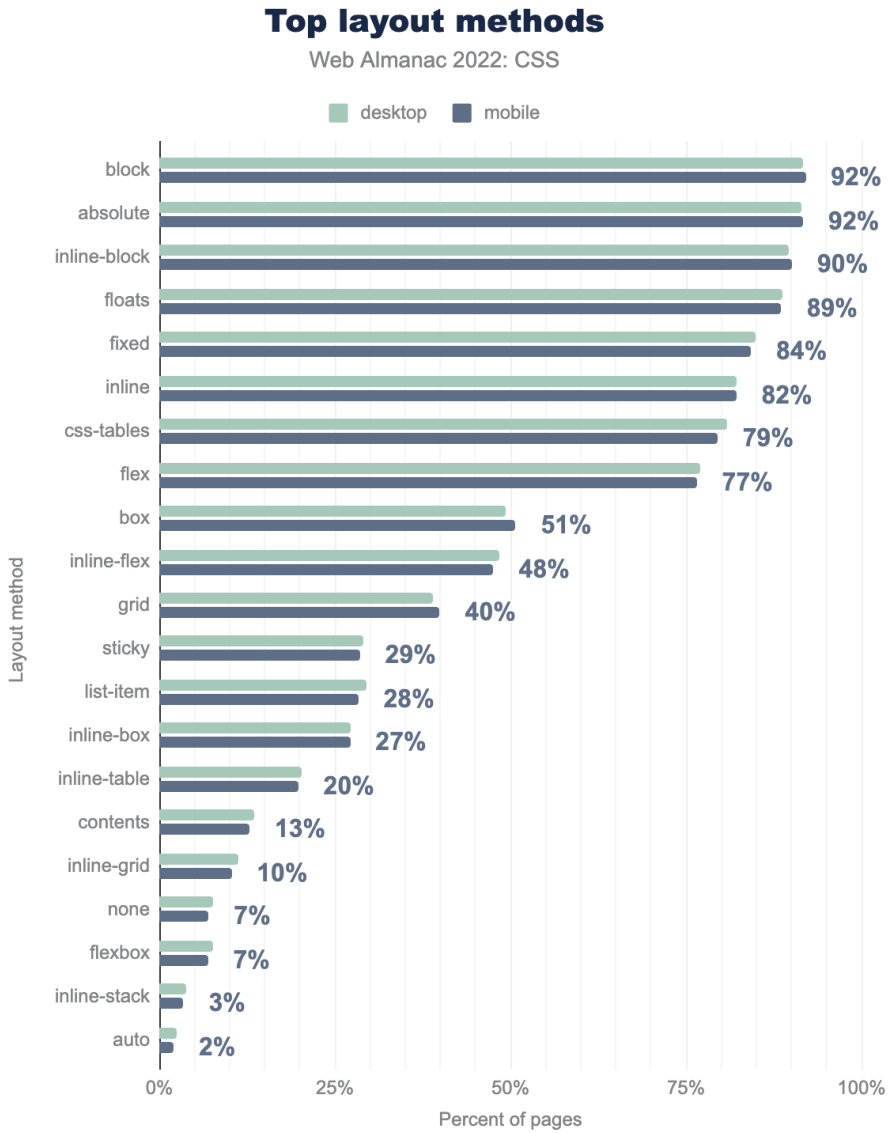

Figure 1.38. Layout methods by percent of pages.

This chart doesn't tell us the main layout method used on a page. It indicates that a property or value appears in the CSS for those pages. For example, 51% of pages are using the old 2009 version of flexbox, with display: box. It's likely this has been added for backwards compatibility, perhaps via a tool such as Autoprefixer.

Flexbox and grid adoption

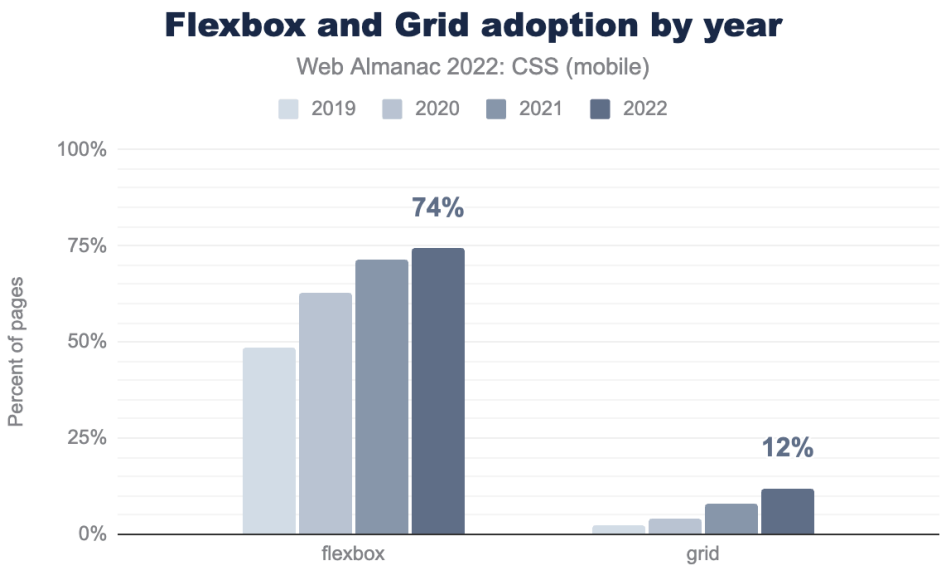

Figure 1.39. Flexbox and grid adoption over the past four years.

Flexbox and grid usage continues to grow. In 2021 flexbox adoption was 71%, it's now at 74%. Grid has jumped from 8% to 12%. Note that, in contrast to the previous section, what is measured here is the percentage of pages that are actually using Flexbox or Grid for layout, as opposed to the pages that simply have some sort of Flexbox or Grid property in their stylesheet.

Grid adoption is reasonably slow, we feel this may be due to the prevalence of frameworks being used for layout, many of which have based their layout on flexbox.

We also took a look at a couple of values of flex and grid properties that are newer to us, to see how adoption of these new features was developing.

The value of content for the flex-basis property is an explicit instruction for the browser to look at the intrinsic content size of the item, rather than any width set

on it. It's a newer value, at the time of writing not available in the release version of Safari. Currently, only 0.5% of mobile and 0.6% of desktop sites use this value.

The subgrid value for grid-template-rows and grid-template-columns is, at the time the queries were run, only supported in Firefox. Perhaps unsurprisingly, it appears in only 211 mobile and 212 desktop pages in the entire dataset. As the value is part of the Interop 2022 project, we will be interested to see how support grows once this becomes interoperable.

Box sizing

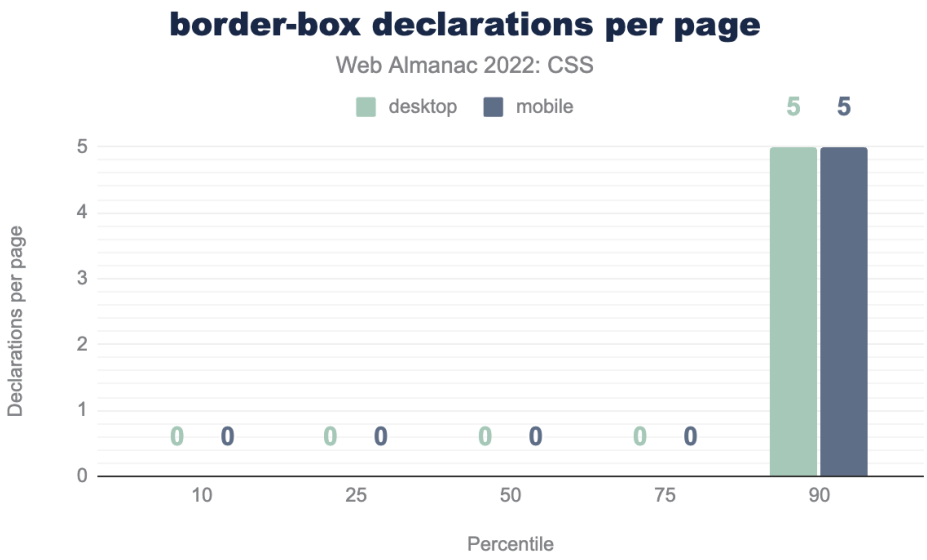

Figure 1.40. Distribution of number of border-box declarations per page.

The web has overwhelmingly voted to reject the original W3C box model in favor of box-sizing: border-box. The number of pages using this property and value combination has risen slightly again to over 90% of pages. Almost half of all pages analyzed apply border-box sizing to every element on the page via the universal selector (*). This approach may help explain why the median number of border-box declarations per page is so low across the bottom four percentiles.

Around 22% of pages use border-box on checkboxes and radiobuttons, we then see a lot of .wp- classes again, showing that WordPress is responsible for the use on 20% of pages analyzed.

Multicolumn

23%

Figure 1.41. The percentage of pages using multicolumn layout

Use of multicolumn layout has increased once again, it's now found on 23% of pages, a rise of 3 points since 2021.

The aspect-ratio property

2%

Figure 1.42. The percentage of pages using the aspect-ratio property

We were interested in the adoption of the new `aspect-ratio` property. This became interoperable towards the end of 2021, so it will be interesting to see usage of this property grow over time.

Transitions and animations

The animation property appears on 77% of mobile pages (the same as last year) and a slight increase on desktop to 76.8%. The transition property is even more popular, it's found on 85% of mobile and 85.6% of desktop pages. The desktop frequency has dropped slightly by around 4 percentage points since 2021.

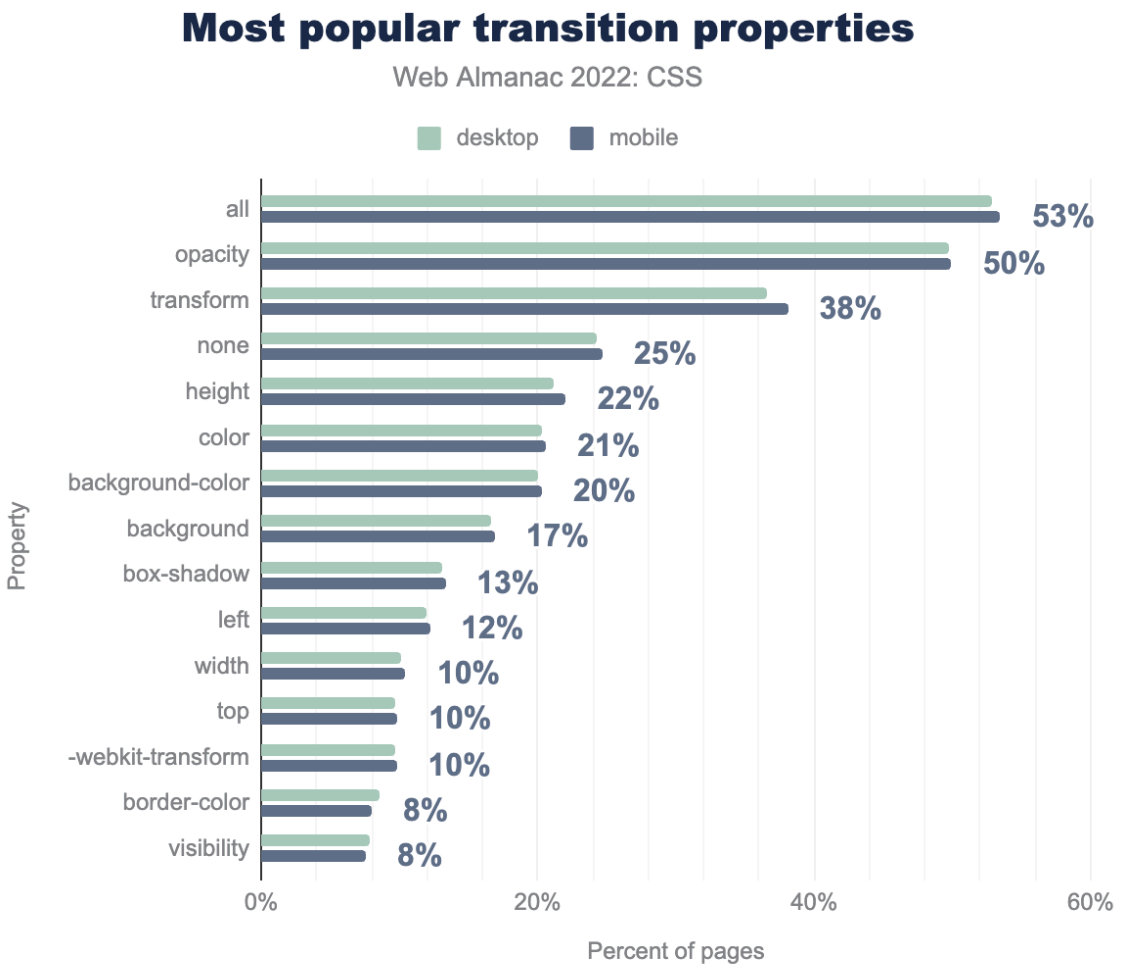

Figure 1.43. The most popular transition properties by percent of pages.

As seen last year the most common usage is to apply transitions to all animatable properties with the `all` keyword. This usage has grown to 53% (up 7 percentage points), it is followed by `opacity`, at 50% pf pages including transitions.

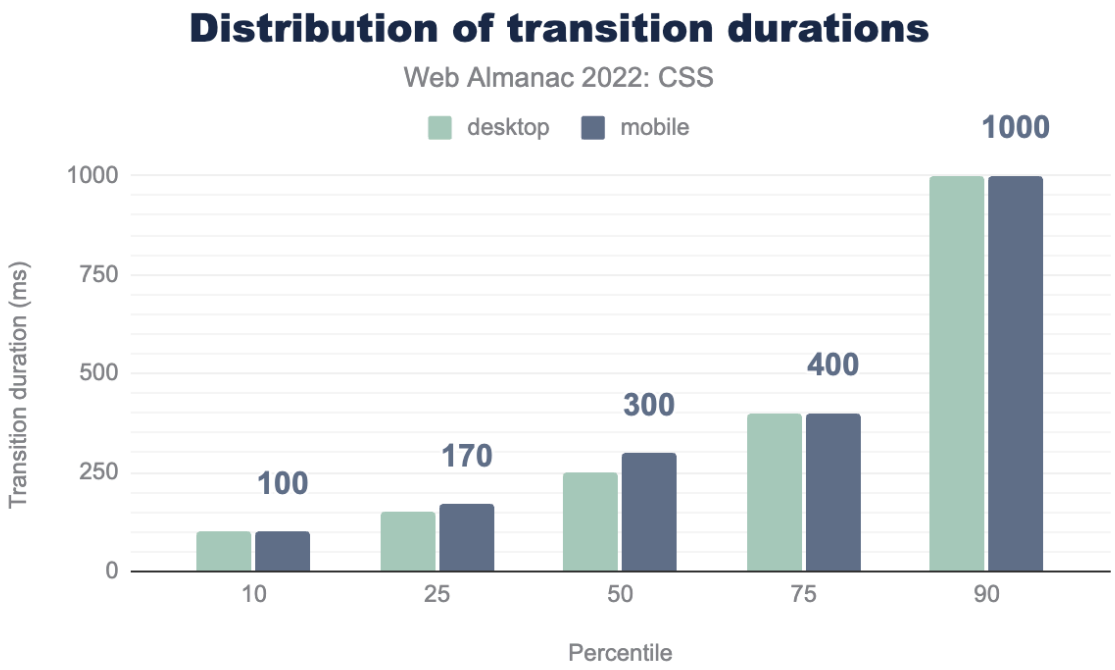

Figure 1.44. Distribution of transition durations.

Looking at the duration of transitions we see a change from last year. In 2021, at the 90th percentile the median transition duration was half a second, this has now jumped to 1 second. We see increases across all top four percentiles.

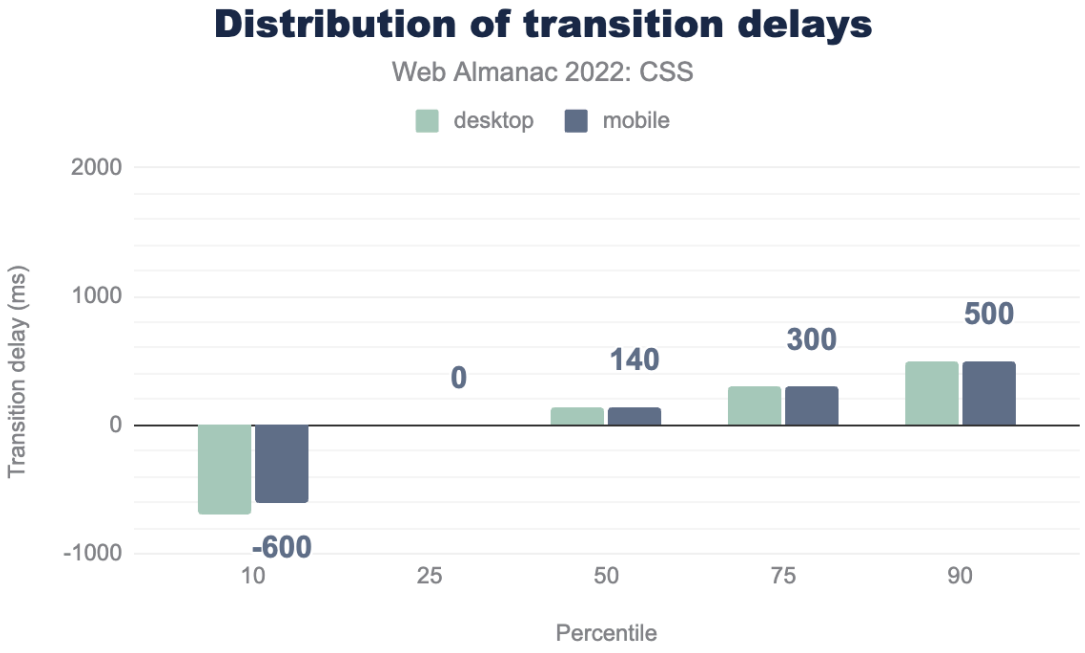

Figure 1.45. Distribution of transition delays.

The distribution of transition delays has also changed. The 90th percentile delay has dropped from 1.7 seconds to half a second. Though the 10th percentile median delay is now over half a negative second. This is seen when a transition starts partway through the resulting animation.

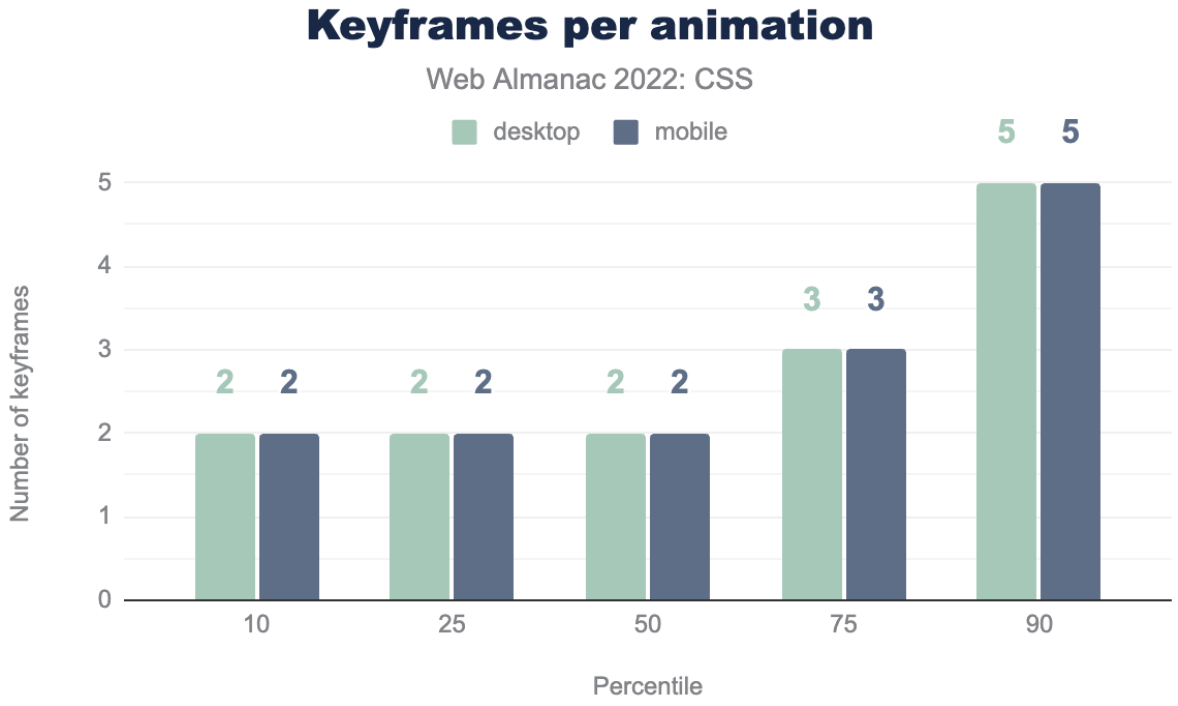

Figure 1.46. Distribution of keyframes per animation.

We also looked at the average number of keyframes used per animation, and found one site that used an astonishing 6,995 keyframes. This was unusual however, and even at the 90th percentile, the number of keyframes per animation is five on both desktop and mobile.

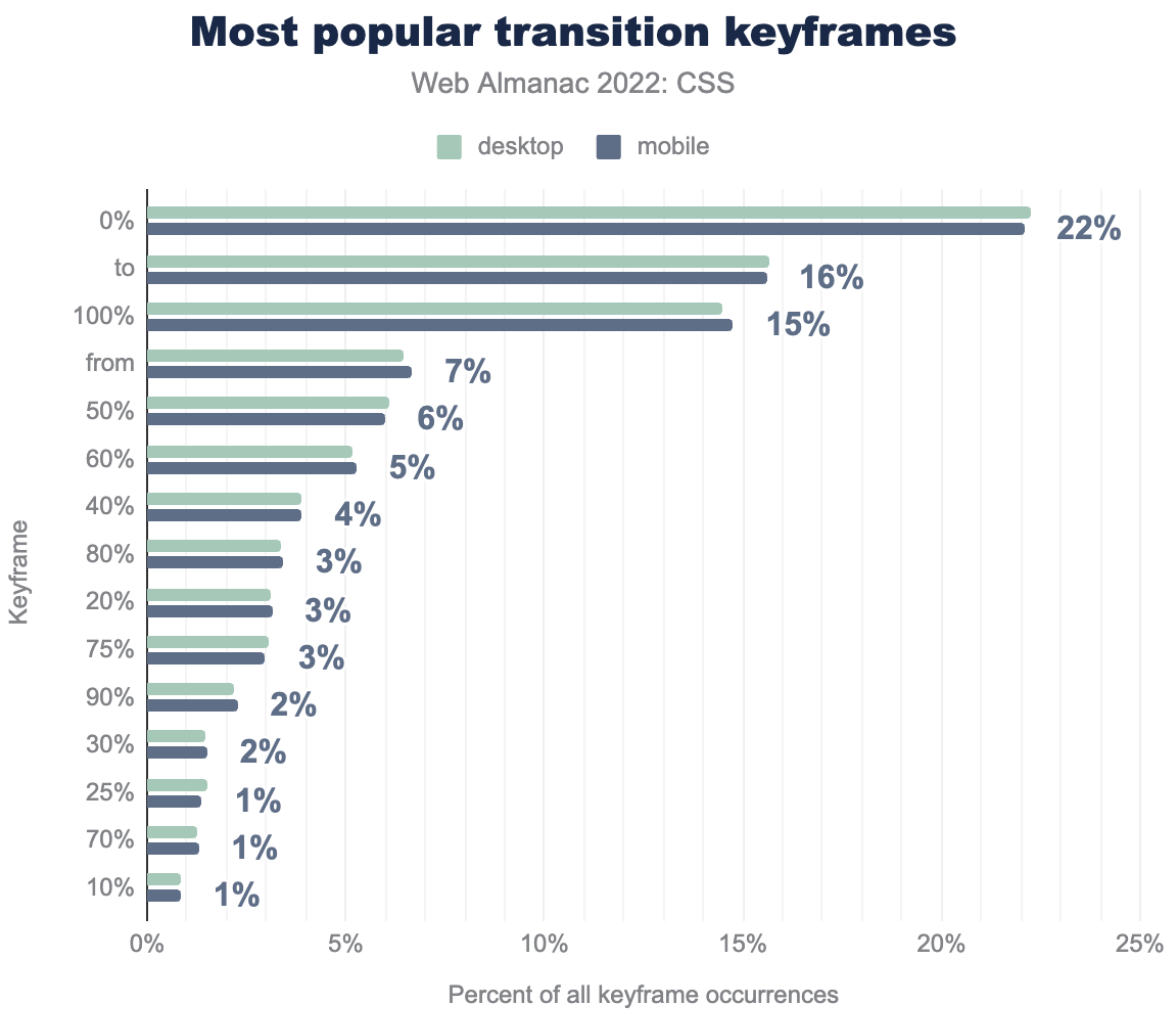

Figure 1.47. The most popular transition keyframes by percent of occurrences.

As you might expect the most popular stops are at 0% to and from 100%, followed by 50%. Developers generally set these stops at 10% intervals, only 1% of pages use 33%, for example.

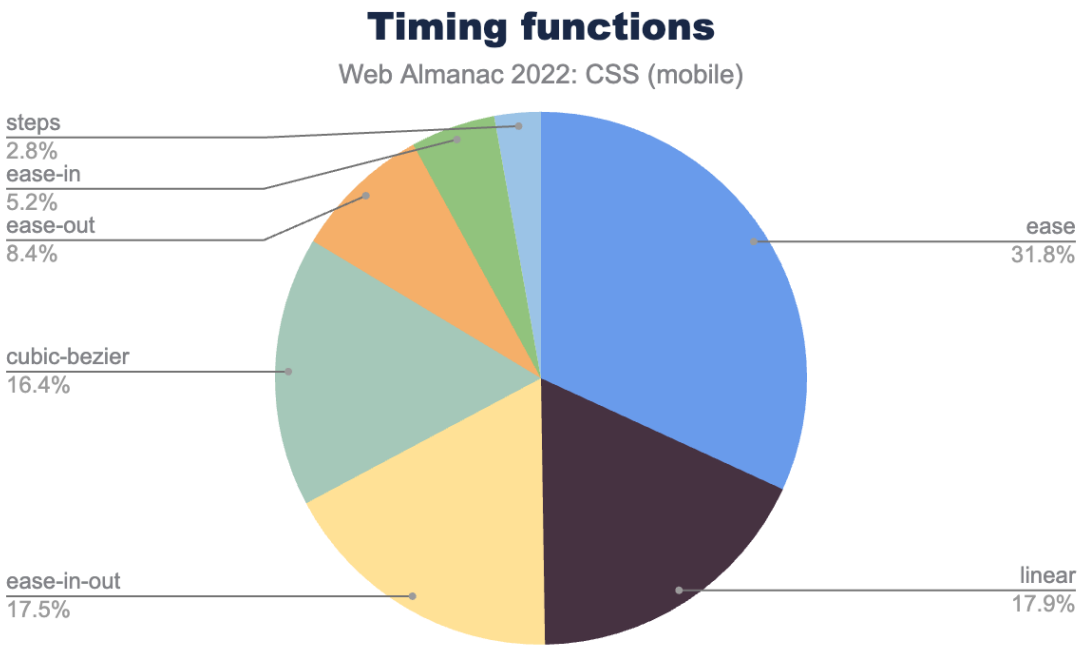

Figure 1.48. Distribution of timing functions.

There has been little change in the distribution of timing functions used during transitions when compared to 2021. As then, the clear leader is ease.

To understand what developers are using animations for, we take a look at the names used for the animation classes. For example, anything with spin in the classname is deemed to be rotate. Rotate animations were the most popular, as in 2021. However the percentage has dropped from 18% to 13%, with bounce animations moving from 5th place to 3rd place in the list.

As last year, the high showing for unknown/other is due to a prevalence of the classname ‘a’, which we can’t map to a specific animation type.

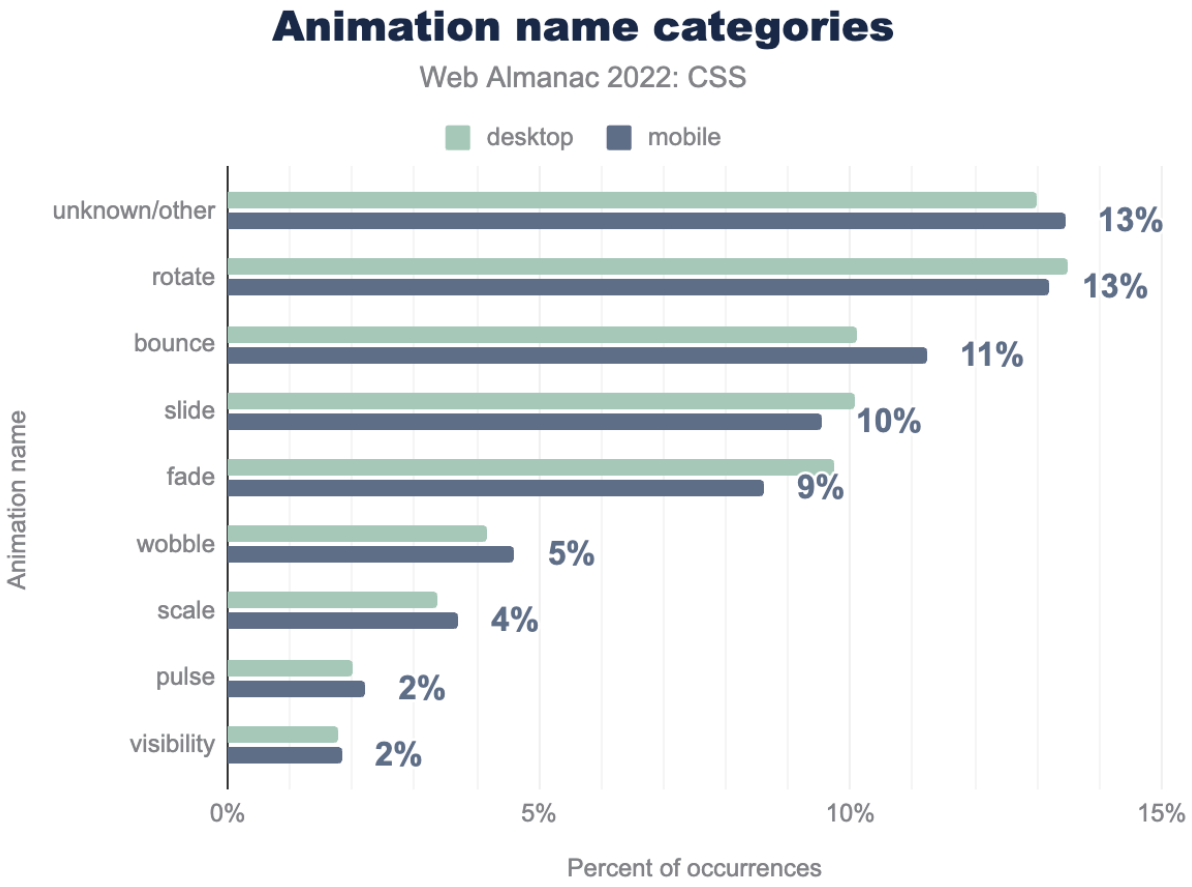

Figure 1.49. Types of animations as identified by animation name.

Visual Effects

18%

Figure 1.50. The percentage of pages using blend modes

We looked at some visual effects being used in CSS. For example, 18% of desktop pages define styles on the `background-blend-mode` or `mix-blend-mode` properties.

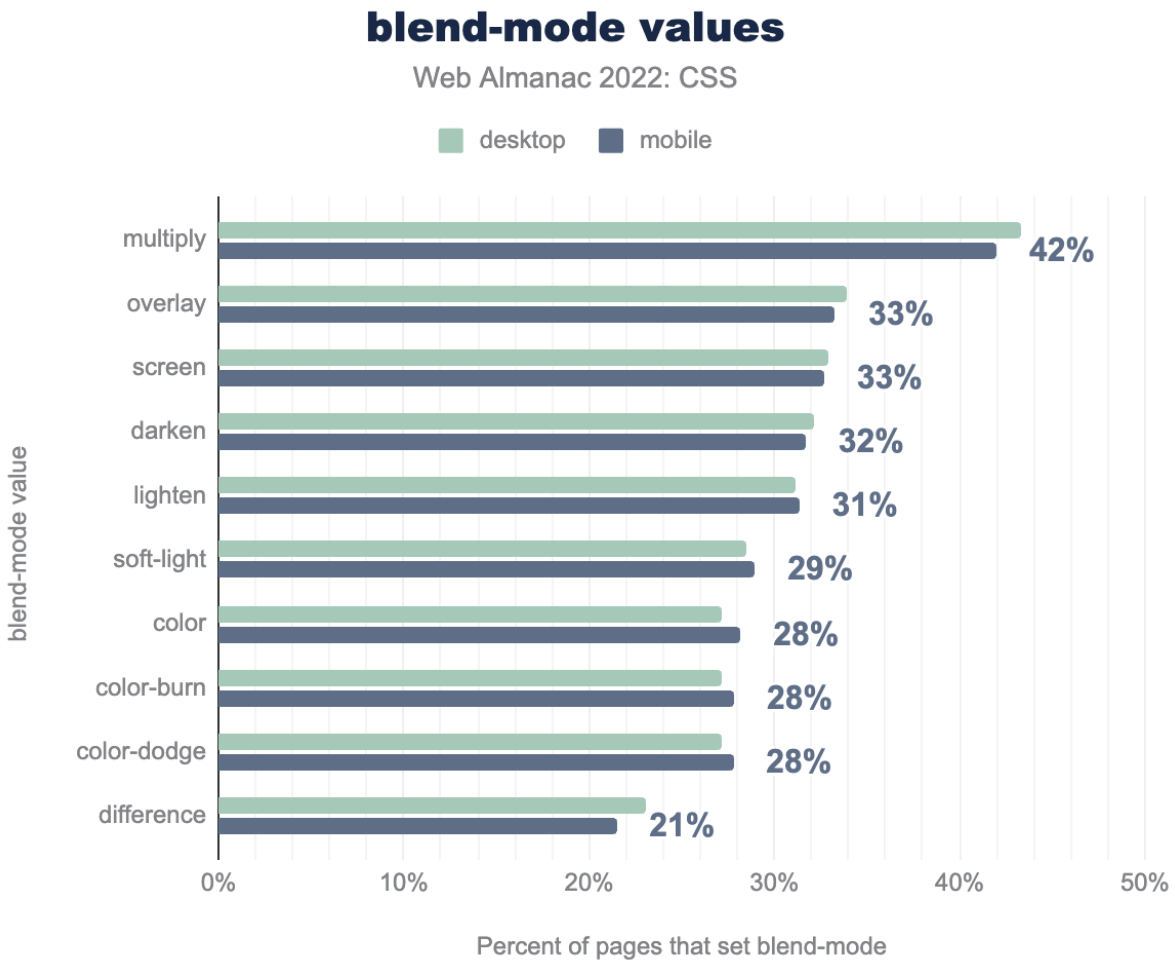

Figure 1.51. Most popular blend modes used on pages that set blend mode.

The most frequently seen value for blend modes was multiply, seen in 42% of pages. However there is a fair distribution of other values too.

Around 18% of pages were using a custom property `var(--overlay-mix-blend-mode)`, a specific name that must come from a library or tool of some sort.

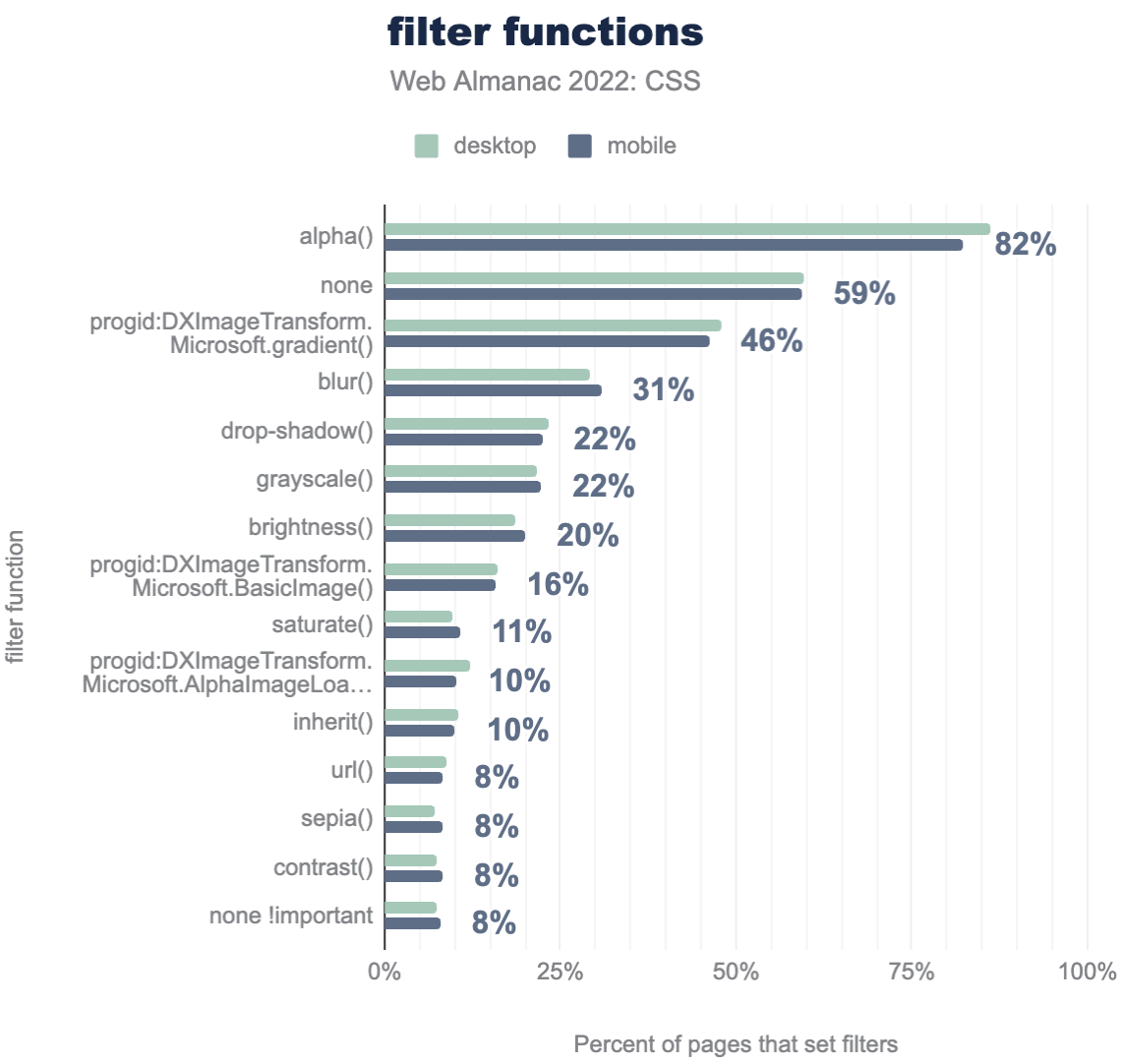

Figure 1.52. Most popular filter functions used on pages that set filters.

Of the percentage of pages that have set filters to apply graphical effects, 82% are using the `alpha()` value, which is non-standard and used for Internet Explorer 8 and below. We also see a high usage of the Microsoft.gradient() filter.

Of the standard values, 31% of pages use `blur()` making it the most popular value after none.

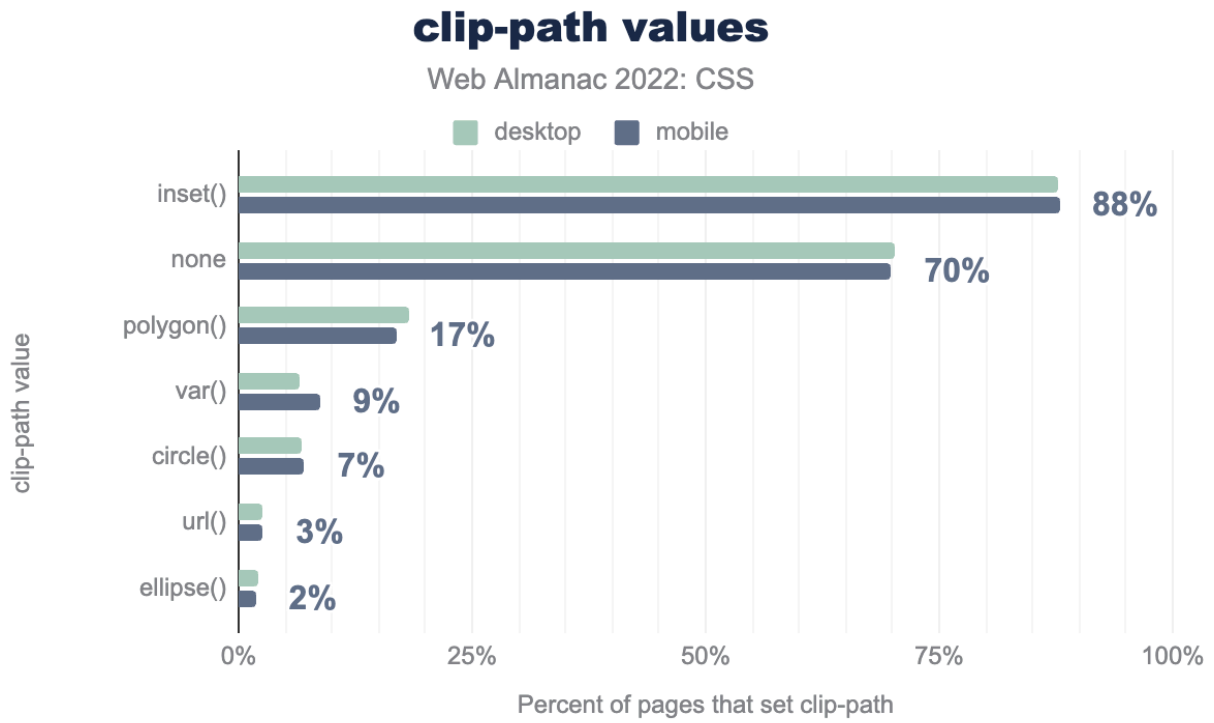

Figure 1.53. Popular clip-path values in pages that set clip-path().

In pages that use clip-path to clip an element, the vast majority are using inset(), the value that simply insets the box of the element, 88% of pages using clip-path have used this function.

After that, and the value none, most developers have chosen to use a polygon(), which is the value that gives the most flexibility to define your own path.

Responsive Design

While many developers are eagerly anticipating container queries, and new layout methods such as flexbox and grid can often enable a design to work well on multiple screen sizes, media queries are used in the majority of pages for responsive design.

When developers write media queries, they most often test the width of the viewport. max-width and min-width were the most popular queries by far, the same as in 2020 and 2021. There was no ranking change in the third and fourth place results either.

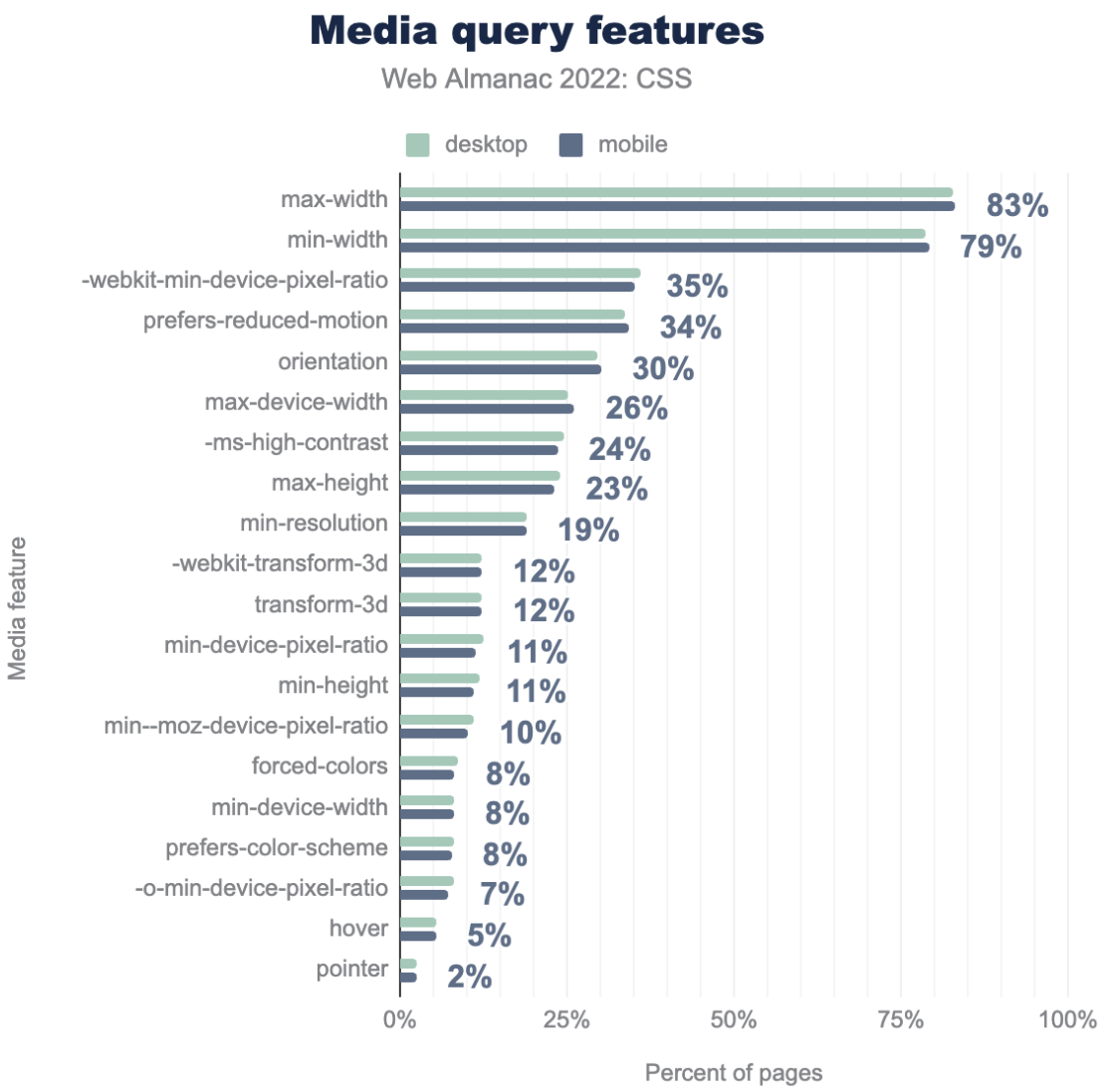

Figure 1.54. Popular media query features.

The prefers-reduced motion media query however, which was noted in 2021 as rising in the rankings, has now edged out orientation to take the fourth spot. This is due to a 2% rise for `prefers-reduced-motion` but also a drop of 4% for `orientation`.

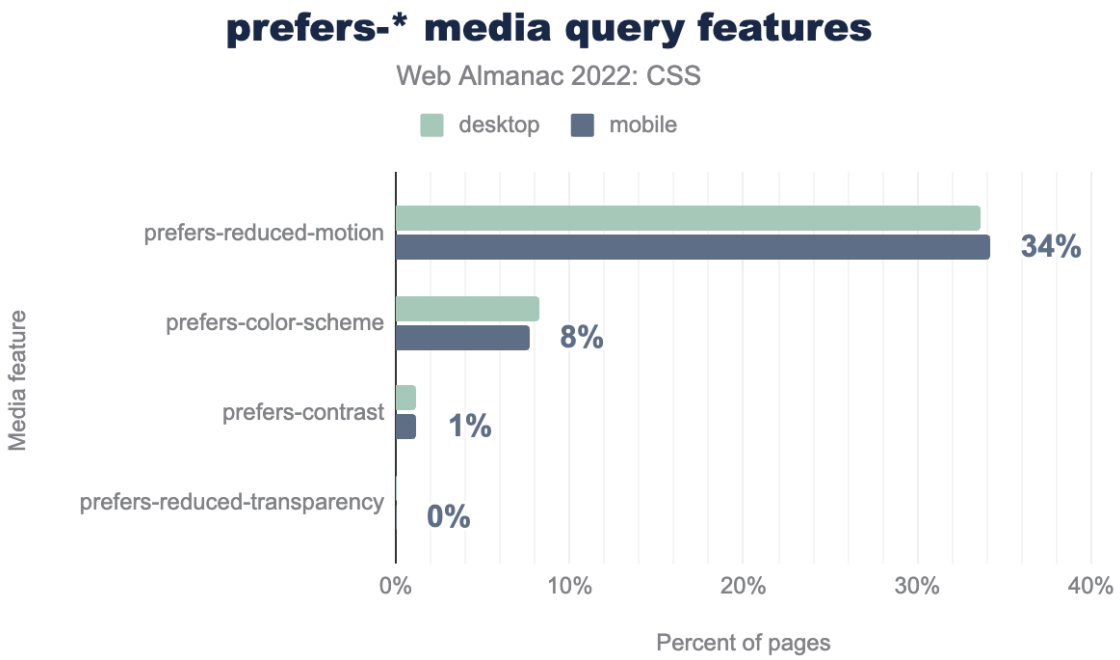

Figure 1.55. Use of user preference features by percent of pages.

If we just look at the `prefers-` user preference features, we can see that `prefers-reduced-motion` is by far the most popular, due to good browser support plus the prevalence of animations and transitions on the web. The `prefers-color-scheme` feature, checking to see if the user has set a preference for a light or dark scheme, has increased in use slightly, as the use of dark mode on websites and applications becomes more popular.

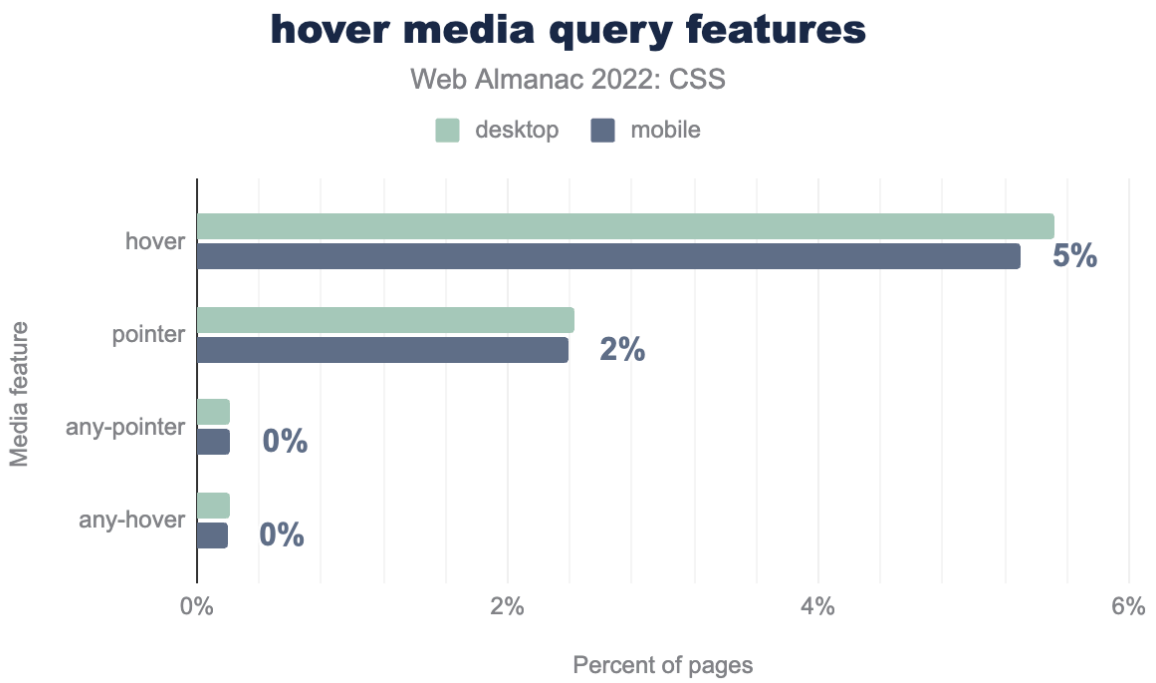

Figure 1.56. Use of hover and pointer media features.

The hover and pointer media features help developers test the capabilities of the device, and the way the user might be interacting with it. They are a better way to discover if a user is using a touchscreen, for example, than screen size alone given the number of large tablets and touchscreen laptops in use.

Both `hover` and `pointer` now appear in the top ten features. The less useful `any-pointer` and `any-hover` see very little use. Using `any-pointer` allows you to determine if a user has access to a fine pointer such as a mouse or trackpad, even if `pointer` indicates they are currently using the touchscreen. Asking a user to switch is definitely not ideal, though a combination of these features could give you a good understanding of the environment a user is working in.

Common breakpoints

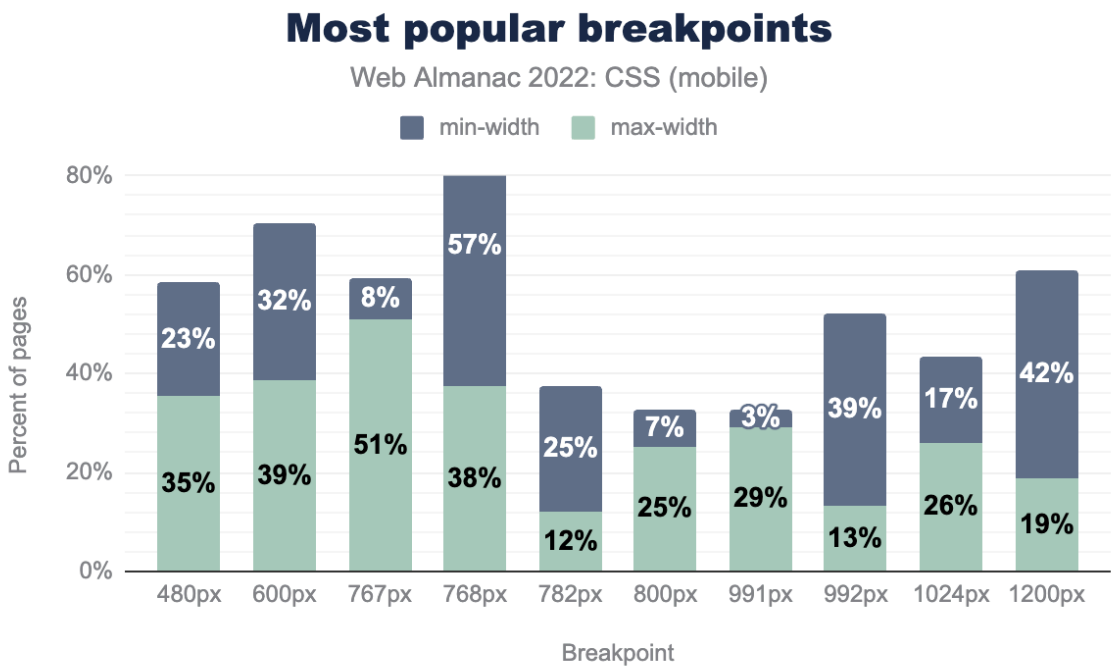

Figure 1.57. Distribution of the most popular breakpoints.

As in the past two years, common breakpoints have changed little. The chart follows the same shape, and the most common breakpoint being a max-width of 767px and min-width of 768px. As noted in 2021, this corresponds with an iPad in portrait mode.

Once again, breakpoints are overwhelmingly set in pixel values, we haven't converted other values to pixels for the chart. The first em value is again 48em, found at position 78.

Properties changed in queries

We looked at the properties that appear within media query blocks, to see which properties people were changing based on breakpoint.

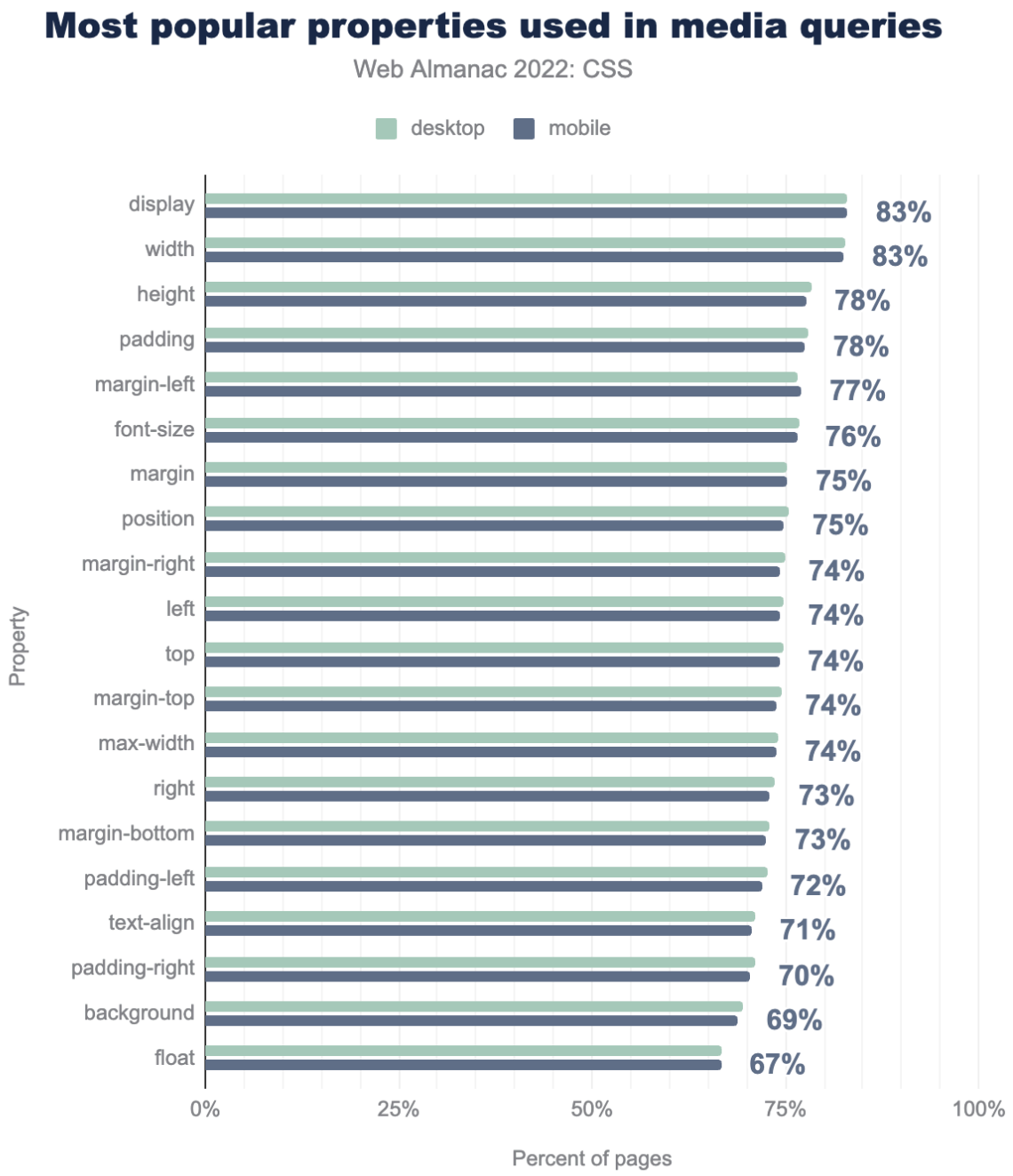

Figure 1.58. Most popular properties found in media query blocks.

The display property is still top of the chart for properties changed within media queries, however there has been some reshuffling in the rankings. These are not as dramatic as they might seem. The color property has vanished from the chart, however this only represents a change from 74% to 67%. It is joined however by a reduction in usage of background-color for 65% to 63%, which makes us wonder if some framework, or perhaps WordPress has stopped using this in a stylesheet.

Another interesting point to note is that in 2020 font-size appeared in 73% of media blocks, and was fifth on the list. In 2021, it showed up in 60% of blocks, appearing at 12th. This year it has gained ground, back to 76% and sixth place.

Feature Queries

Features queries, testing for support of a CSS feature, were found in 40% of mobile pages and 38% of desktop pages. This was down from a figure of 48% in 2021. This may indicate that support for common features tested has become great enough for people not to worry about testing for the feature before use.

The number of feature query blocks per page is 4 at the 75th percentile, and at the 90th percentile 7 for desktop and 8 for mobile. We did find one site however with 1,722 feature query blocks.

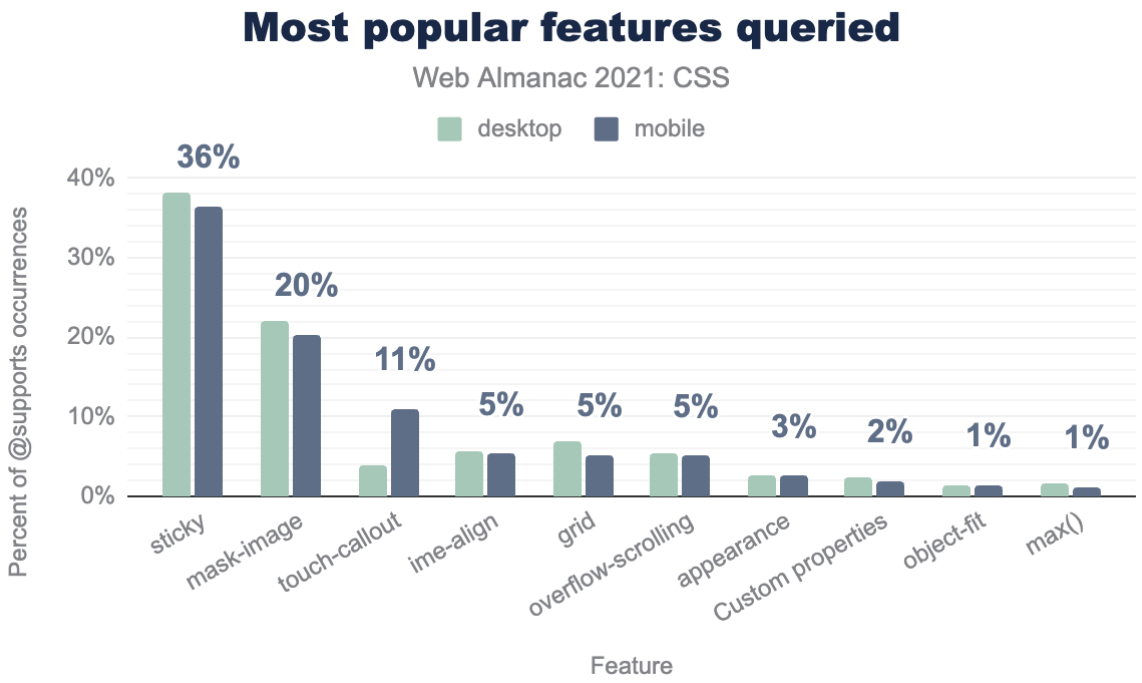

Figure 1.59. Most popular features tested for with feature queries.

As last year, the most popular feature tested for in feature queries was `position: sticky`, however this has fallen from 53% to 36% of occurrences, perhaps due to the improved browser support for this feature.

Non-standard features show up strongly in these tests, with touch-callout (-webkit-touch-callout) and ime-align (-ms-ime-align). The former has grown in usage from 5% to 11%, while ime-align has dropped from 7% to 5%.

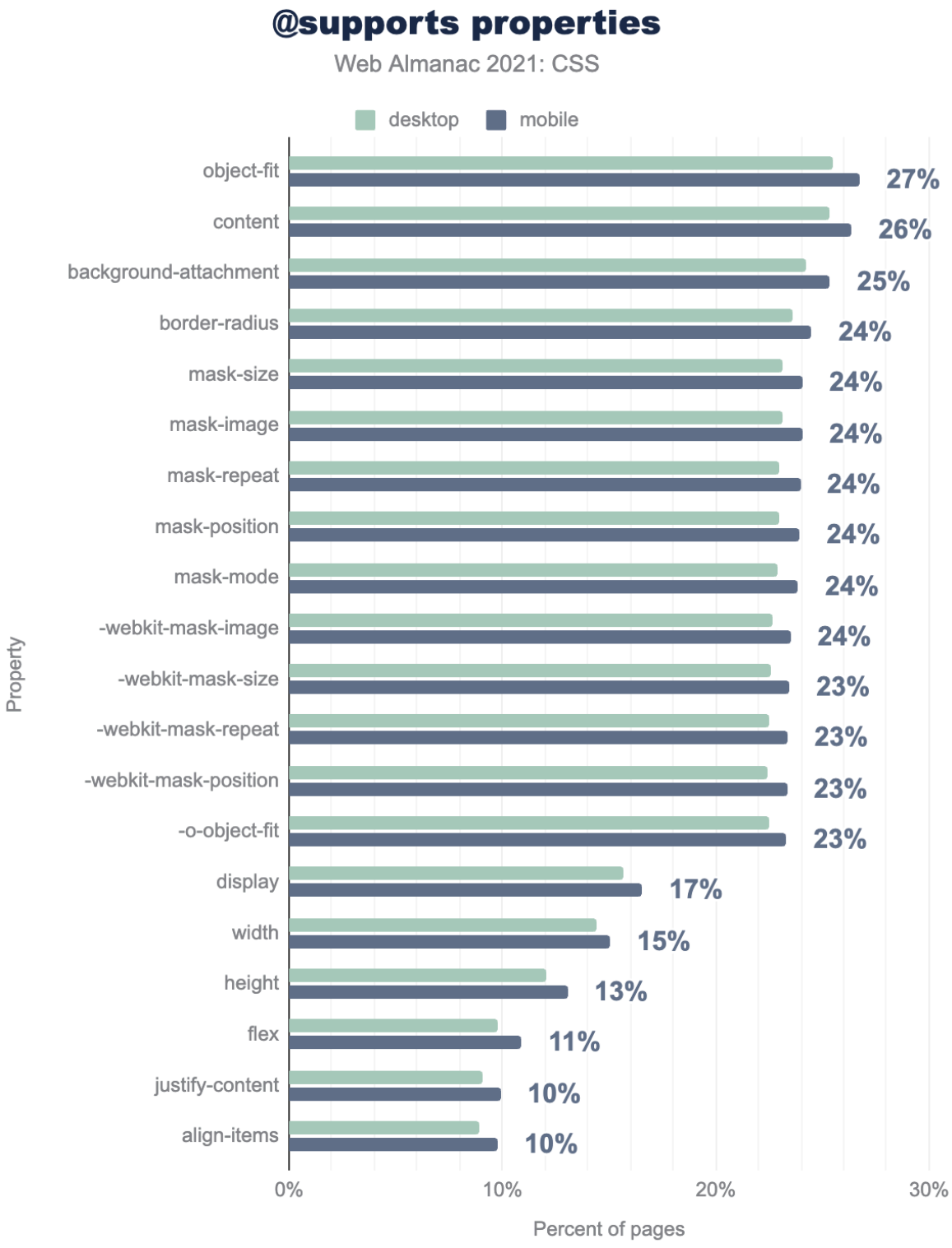

Figure 1.60. Properties used inside feature query blocks by percent of pages.

Having tested for support, which properties are then used inside these feature query blocks? The `object-fit` property came out top, the mask- properties making a good showing, along with their `-webkit-` counterparts. This is likely due to the lack of interoperability for masking until recently, the properties still requiring a `-webkit` prefix for Chrome.

While the `display` property features in the top 20, you have to go a long way down the list to find any grid- properties. The `grid-template-columns` property being found in 2% of feature query blocks.

Internationalization

English is described as a horizontal top to bottom language, because sentences are written horizontally, starting at the top of the page. The script direction runs left-to-right. Arabic, Hebrew and Urdu are also horizontal top to bottom languages, but have a script direction of right-to-left. There are also languages that are written vertically, from top to bottom—such as Chinese, Japanese, and Mongolian. CSS has evolved to better cope with these different writing modes and script directions.

Direction

The number of pages using the direction property to set CSS either on the <body> or <html> element remained unchanged from 2021, with 11% of pages setting it on <html> and 3% on <body>. It's recommended to use HTML, rather than CSS to set direction, so a lower number here matches that best practice.

Logical and physical properties

Logical or flow-relative properties such as border-block-start and values such as start for text-align are useful for internationalization, as they follow the flow of text rather than being tied to the physical dimensions of the screen. Browser support for these properties is now excellent, so we wondered whether we would see more adoption.

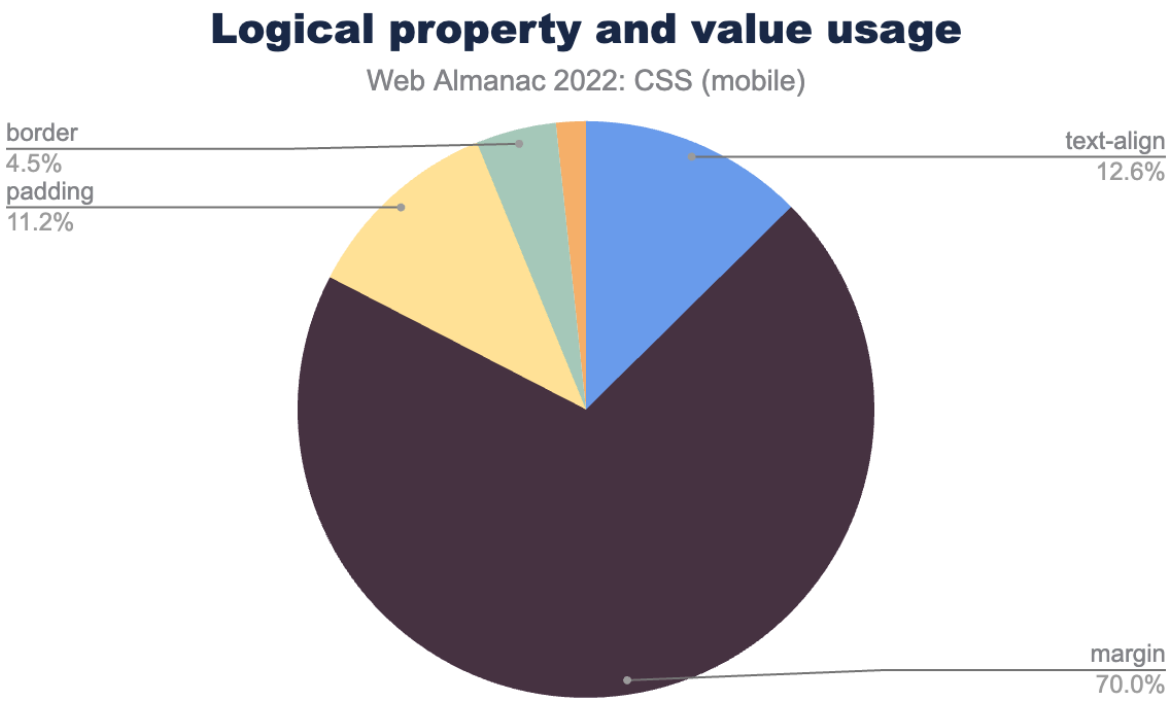

Figure 1.61. The distribution of logical properties used.

Logical property usage has increased slightly from 2021, up from 4% to 5%. However, the chart for 2022 looks very different to the one for 2021. Overwhelmingly people are using logical properties to set margin properties, up to 70% from 26%. The most popular margin- properties are margin-inline-start and margin-inline-end, they are found in 9% of total pages. These are particularly useful for making sure that spacing between a label and following field, for example, works in the same way in a LTR and RTL script.

Ruby

Once again we checked for usage of CSS Ruby, this is a collection of properties used for interlinear annotation, which are short runs of text alongside the base text. Its usage is still tiny, but has increased from 2021. In only 8,157 desktop pages and 9,119 mobile pages were found to be using it—less than 0.1% of all pages analyzed. This year 16,698 desktop, and 21,266 mobile pages—or 0.2% of all pages analyzed—were using it.

CSS in JS

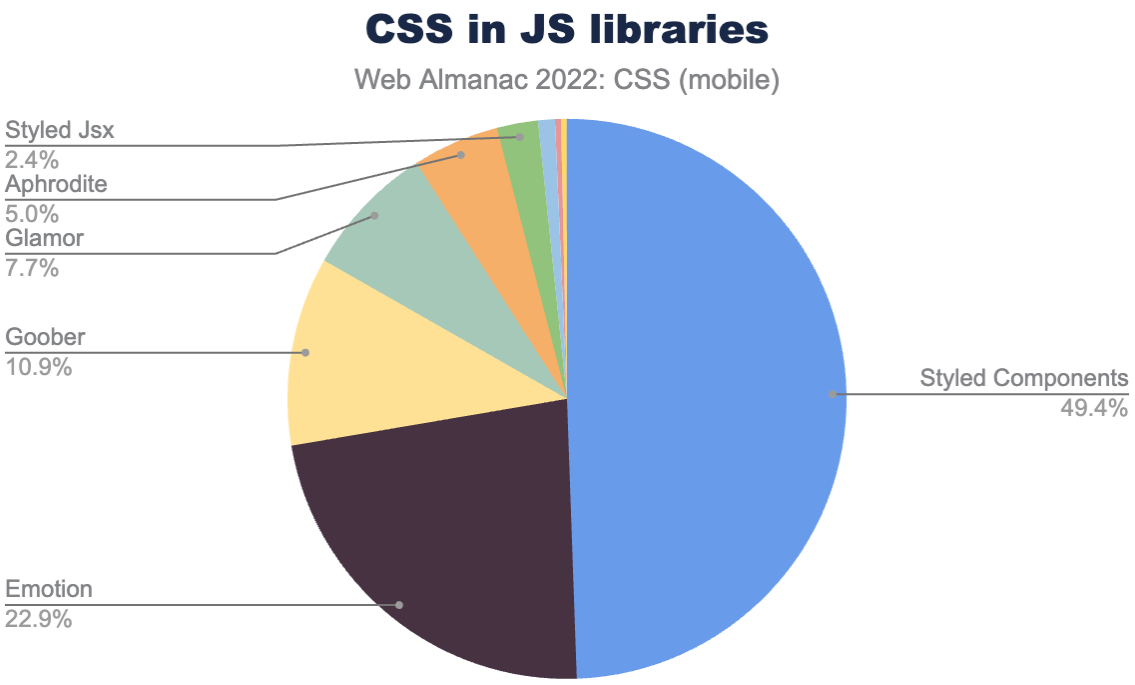

Figure 1.62. Usage of CSS in JS libraries.

The use of CSS-in-JS has not increased from last year, staying at 3%. This usage is almost all from libraries, the most popular of which is Styled Components. This library has dropped in share from 57% to 49%, with a new library entering the mix at almost 11%. Goober describes itself as "a less than 1KB css-in-js solution", and is certainly making some inroads among people who like this type of thing.

Houdini

There is still very little usage of Houdini on the open web, looking at the number of pages using animated custom properties shows only a small increase since 2021. We also looked at usage of the Houdini Paint API. We do find instances of this in use on the web. By looking at the names of worklets used, much of this is this Smooth corners worklet, indicating that people are using it as a progressive enhancement, given that this can fall back nicely to a regular border-radius.

Sass

Preprocessors like Sass can be seen as a good indicator of what developers want to be able to do with CSS, but can't. And, with CSS increasing in power, a common question from developers is whether we need to use Sass at all. We can see from

the rise in custom properties usage, that one common preprocessor use, to be able to have variables or constants, has now a built-in CSS equivalent.

Looking at the function calls shows that color functions are still a very popular use of Sass, something that may well soon be replaced with new CSS color functions such as color-mix(). There are some changes from last year. The darken function has dropped 2 percentage points to 14% and third place. The lighten function has, however, gained a points.

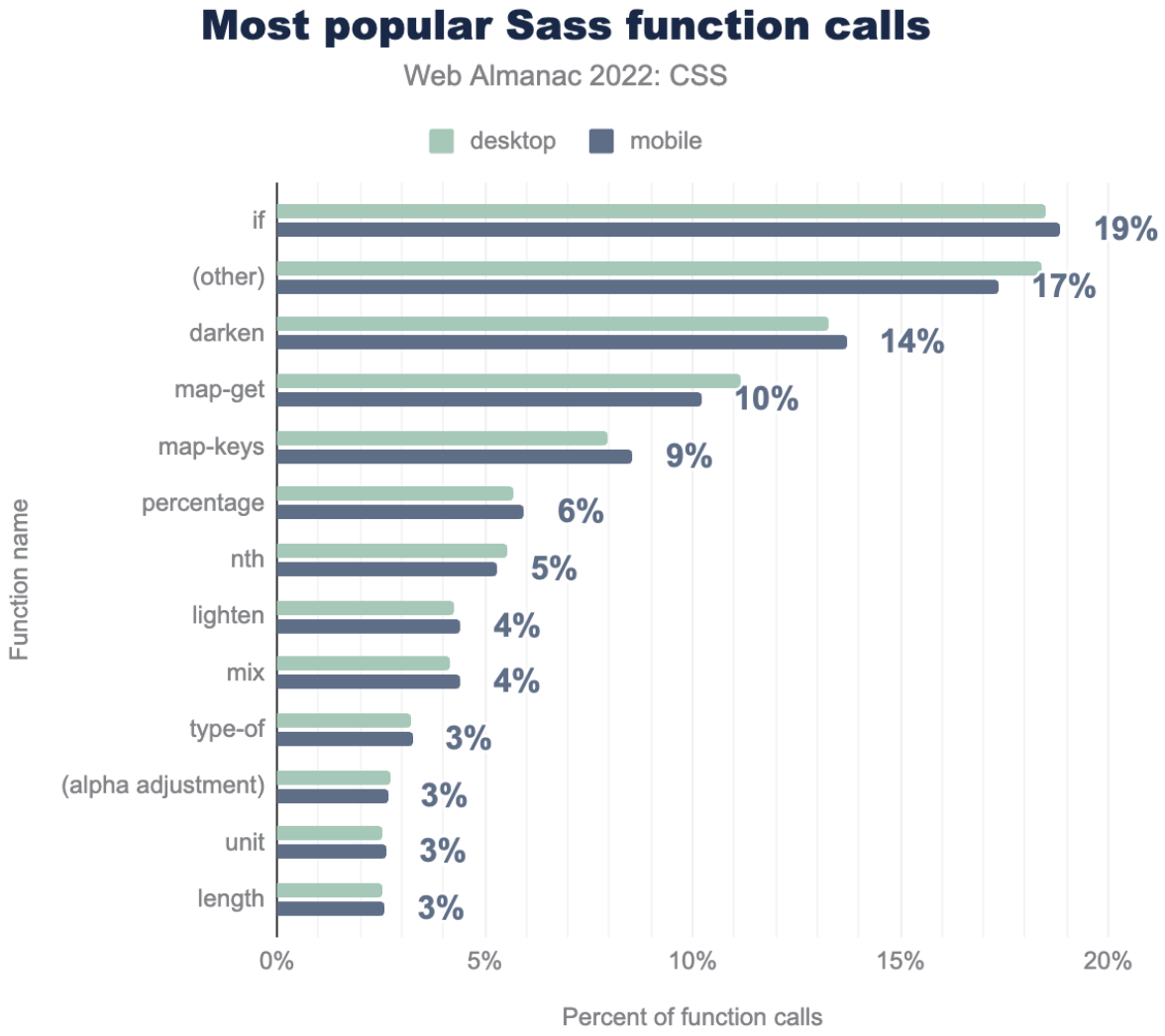

Figure 1.63. Most popular Sass function calls by percent of calls.

Looking at control flow statements we see a small increase in `@for` and `@each`, however `@while` has increased from 2% to 7%.

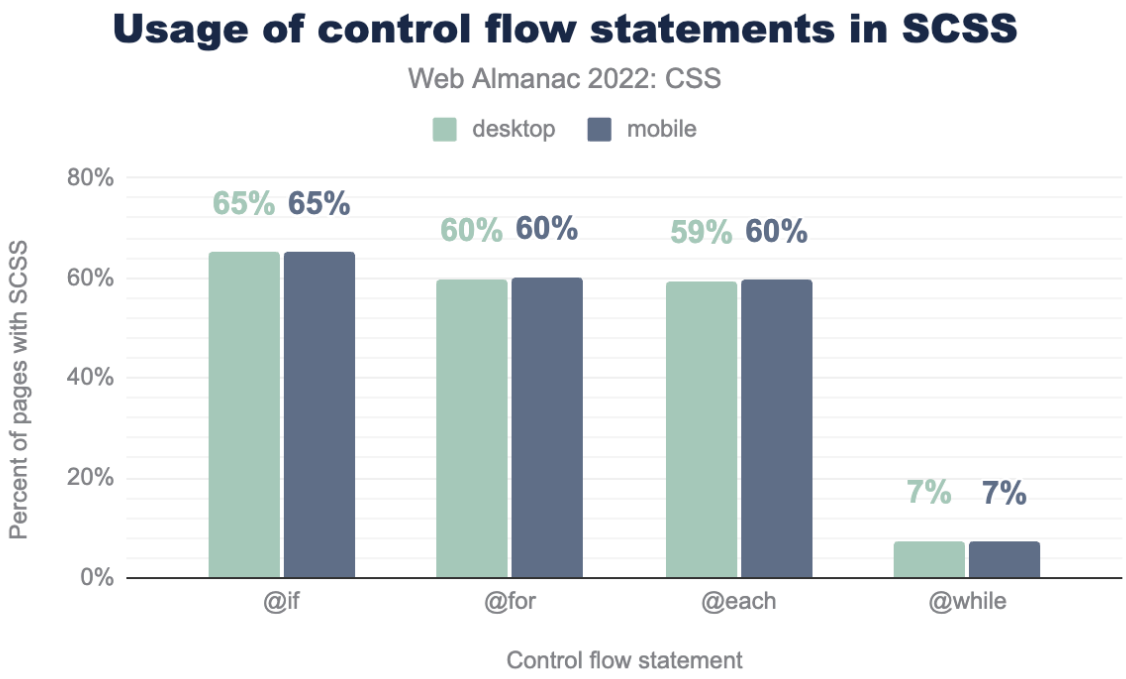

Figure 1.64. Distribution of control flow statements on SCSS.

Nesting is also interesting, given that a future spec for CSS Nesting is currently in development and discussion at the CSS Working Group. Nesting in SCSS sheets is very common, and can be identified by looking for the & character. As with last year pseudo-classes such as :hover, and classes such as .active make up most cases of nesting. All usage increased slightly, however & descendent increased 7 percentage points from 18% to 25%. Implicit nesting is not measured in this survey, as it does not use special characters.

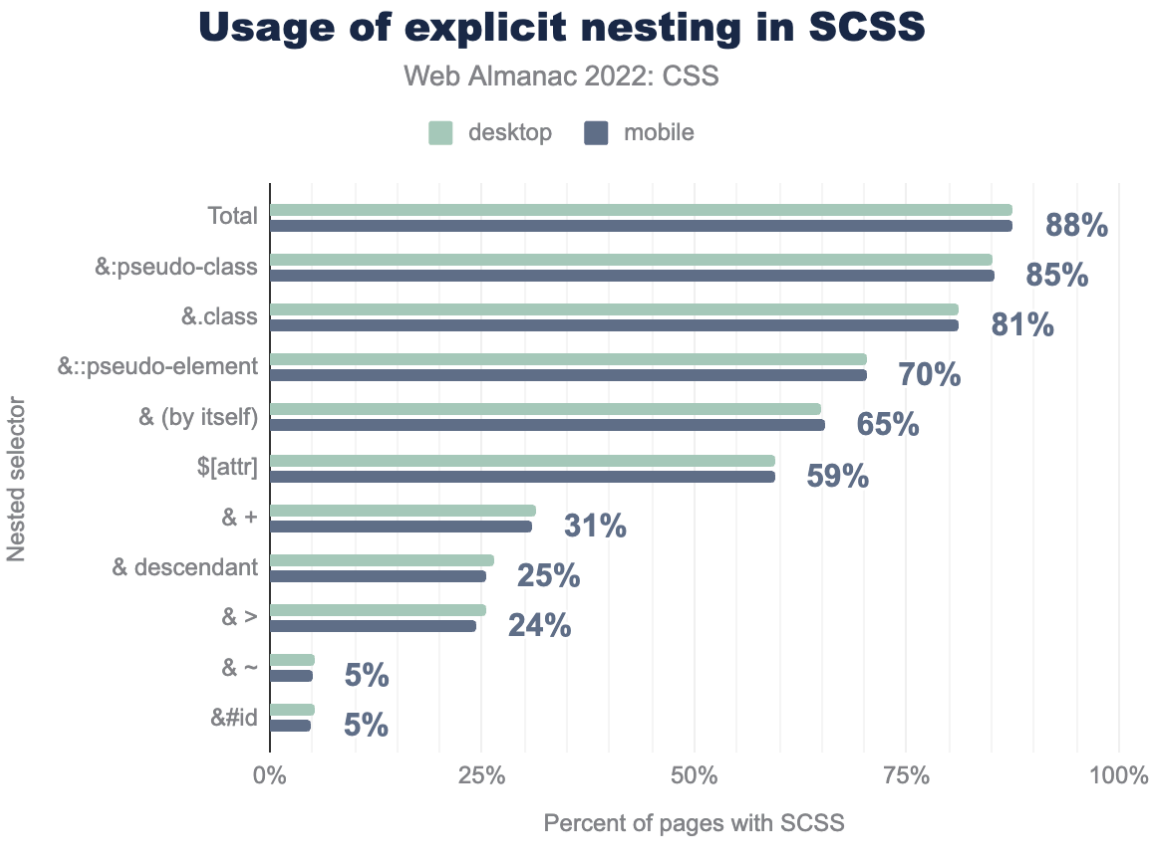

Figure 1.65. Use of explicit nesting in SCSS by percent of pages using SCSS.

CSS for print

5%

Figure 1.66. The percentage of desktop pages with print specific styles

We wondered whether developers were creating print stylesheets to provide a better printed experience, and only 5% of desktop and 4% of mobile sites were doing so.

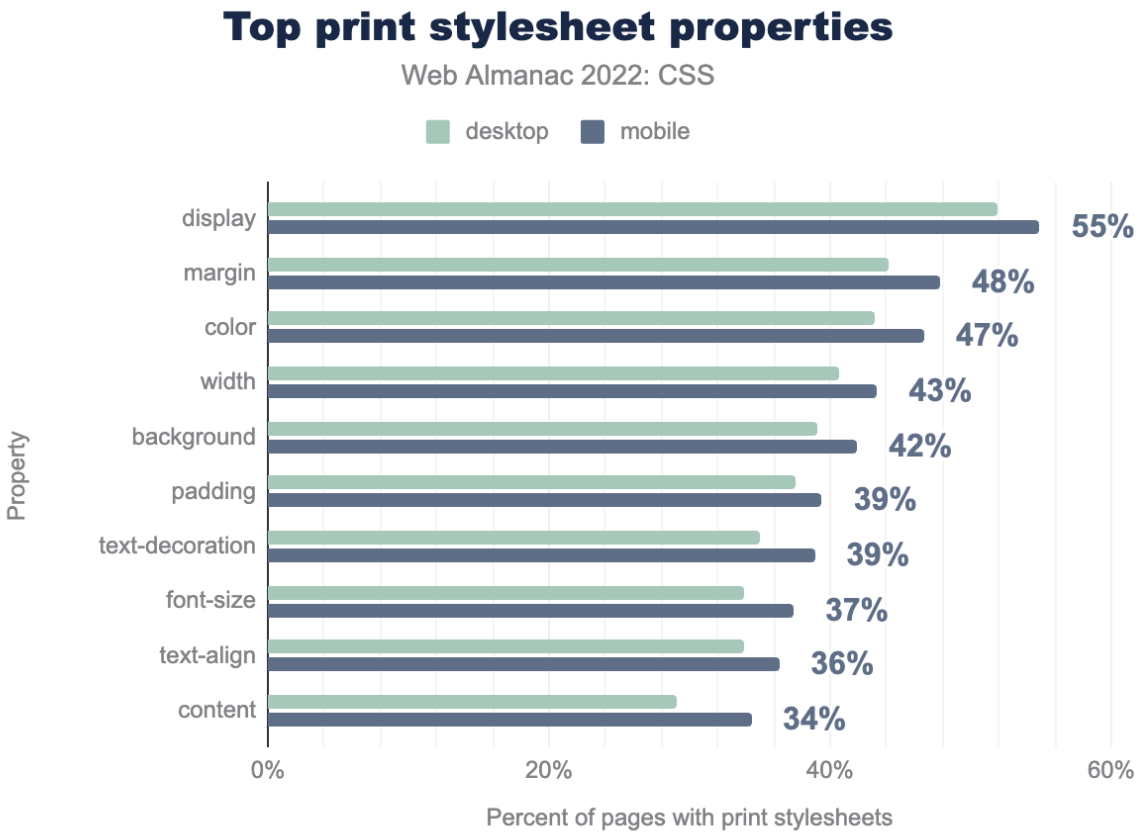

Figure 1.67. The top properties found in print styles on pages that have a print stylesheet.

Of the pages using print styles, over half changed the value of display—perhaps to simplify a grid or flex layout for print. We also see people changing colors, tweaking margin and padding, and setting the font-size. At 34% is the content property, used to insert generated content.

Print is a fragmented medium, the content is fragmented into pages, and we have a set of fragmentation properties that aim to give some control over how these breaks happen. For example, developers usually want to avoid a heading being the last thing on a page, or a caption being disconnected from the figure it relates to.

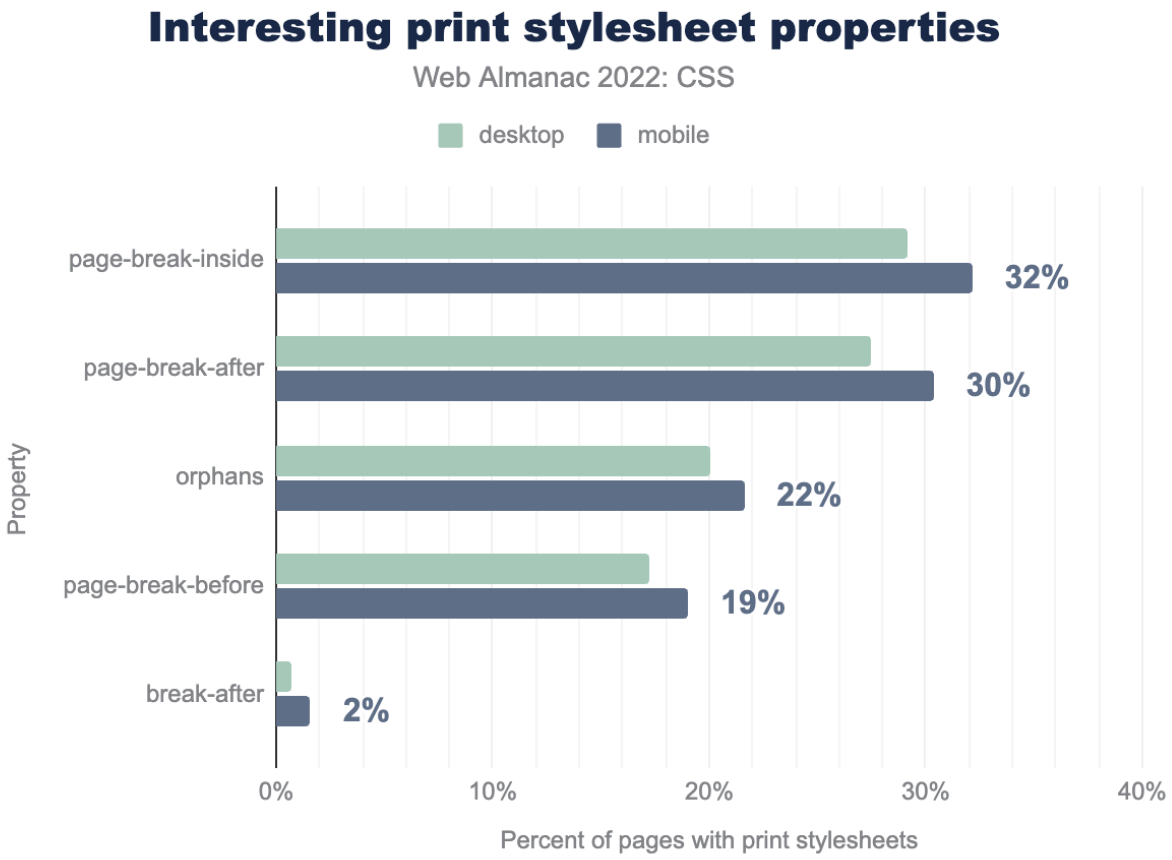

Figure 1.68. Fragmentation properties used in print stylesheets.

We see in this chart that many developers are using the old fragmentation properties of `page-break-inside`, `page-break-after`, and `page-break-before`, rather than the new properties such as `break-before`, which has very low usage.

The orphans property appears in 22% of print stylesheets, despite lacking support in Firefox. This property defines the number of lines that should be left at the bottom of a page before a fragmentation break. The widows property (which sets the number of lines on their own after a fragmentation break) is seen with around the same frequency. It is likely that people are setting the same value for both.

Paged media

There is an entire specification for dealing with Paged Media, and CSS for print. However, this has been poorly implemented in browsers. To find a good implementation of these features you need to use a print specific user agent.

There is some browser support for the @page rule, and its pseudo-classes, and we did find developers using these to set different page properties for the first page, and the left and right pages of a spread.

Pseudo-class	Desktop	Mobile
:first	5,950	7,352
:right	1,548	2,115
:left	1,554	2,101

Figure 1.69. Number of pages found using @page spread pseudo-classes

Of people using these pseudo-classes the use was mostly to set the page margins, and also the size of the page.

Meta

This section rounds up some general information about CSS usage, for example how often declarations are repeated, and common mistakes in CSS.

Declaration repetition

In 2020 and 2021, analysis was done to determine the amount of "declaration repetition". This aims to identify how efficient a stylesheet is by looking for the number of declarations using the same property and value.

In 2021 it was reported there was a slight drop in repetition, this year there is a slight rise. This metric does therefore seem fairly stable year-on-year.

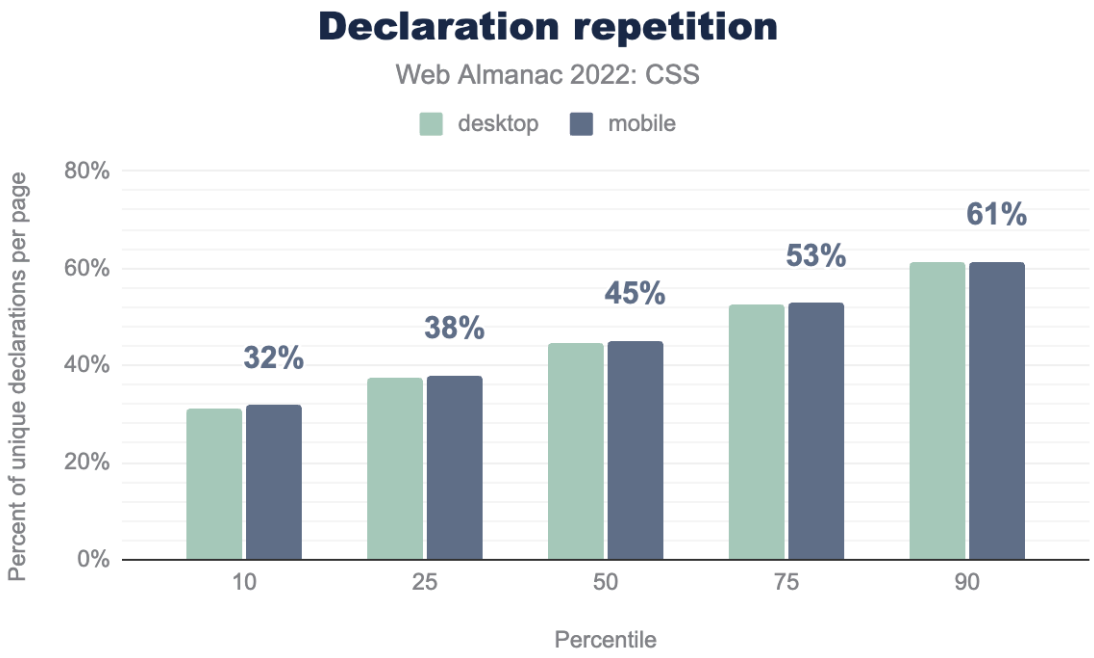

Figure 1.70. Distribution of repetition.

Shorthands and longhands

In CSS a shorthand property is one that can set a number of longhand properties in one declaration. For example, the shorthand background property can be used to set all background longhand properties—background-attachment, background-clip, background-color, background-image, background-origin, background-position, background-repeat, and background-size.

When developers mix shorthand properties like background and longhand properties like background-size in a stylesheet, it is always best to have the longhands come after the shorthands. We looked at instances of this to see which longhands were most common.

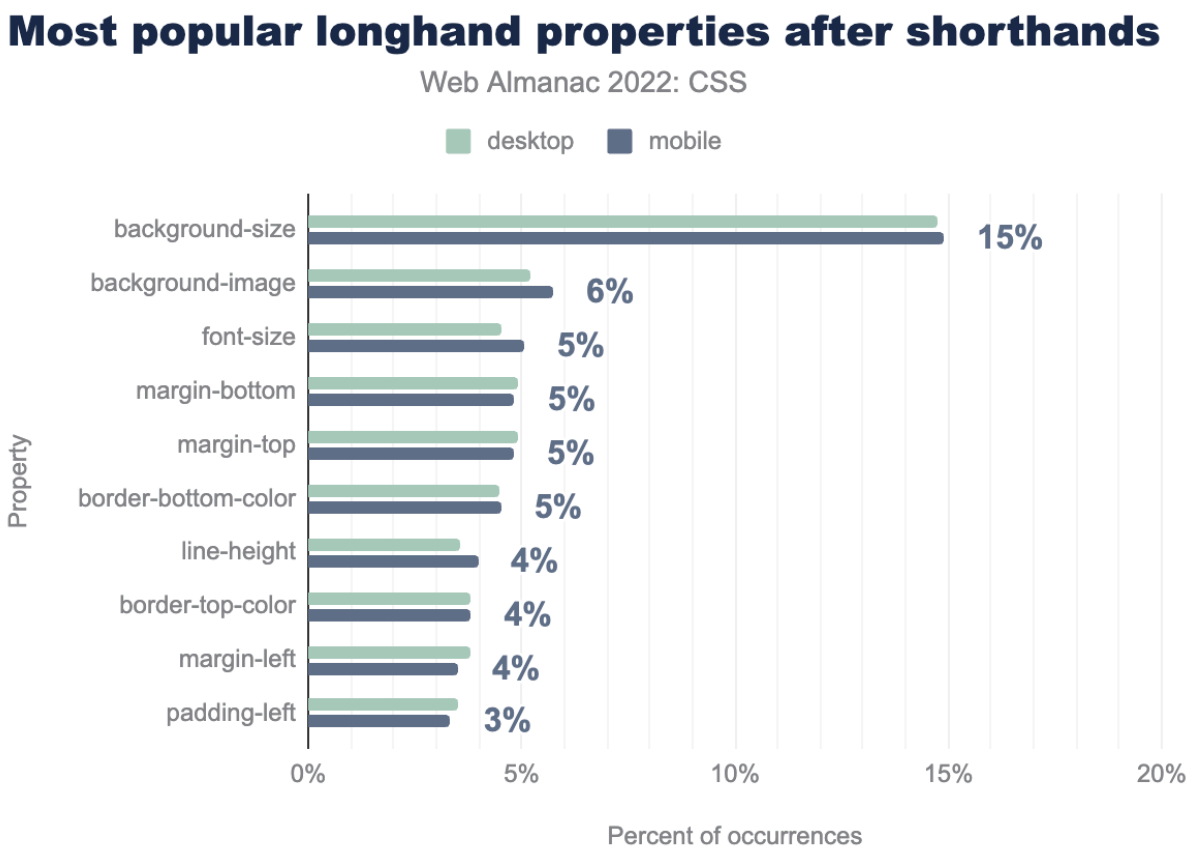

Figure 1.71. The most popular longhand properties that come after shorthands.

As in 2020 and 2021, background-size came out top of the chart, and there was little difference to be seen from 2021.

Unrecoverable syntax errors

To check for unrecoverable errors, we use the Rework CSS parsing engine. An unrecoverable error is one where the error is so bad, the full stylesheet is unable to be parsed by Rework. Last year, 0.94% of desktop pages, and 0.55% of mobile pages contained an unrecoverable error. This year 13% of desktop and 12% of mobile pages had such an error. This seems like a large jump, however due to some changes in methodology (adding size thresholds) it is likely that not all of the instances are unrecoverable errors.

Nonexistent properties

As in previous years we checked for declarations that had valid syntax, but referred to properties that don't actually exist. This includes spelling errors, malformed vendor prefixes, and things developers have just made up.

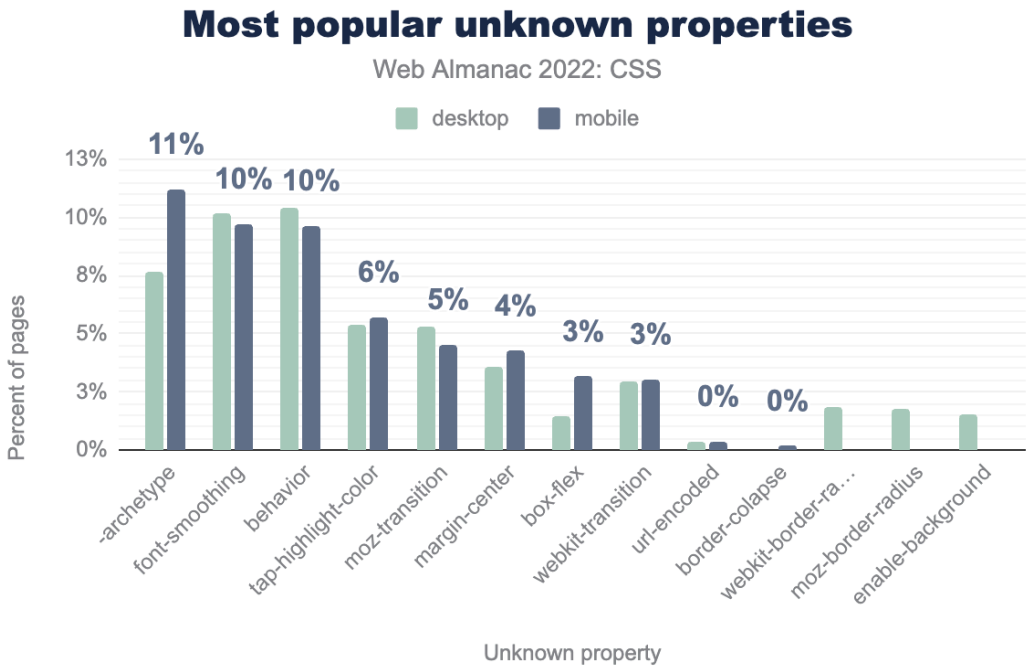

Figure 1.72. The most frequently seen unknown properties.

The top mystery property is -archetype, which is now appearing in 11% of cases of stylesheets with nonexistent properties. This property has jumped from 4% last year to 11% to take the top spot. The second property is font-smoothing with a drop of 4% points from last year. This appears to be an unprefixed version of -webkit-font-smoothing that does not actually exist. The use of the malformed webkit-transition (which should be -webkit-transition) has dropped from 14% to 3%, this makes us think it was perhaps getting into a large number of stylesheets via a framework or other third party, that has since updated to fix the problem.

Conclusion

CSS continues to evolve at a rapid pace, however we can see from the data that new features are adopted quite slowly, even when they have been in all major engines for several years. There are a few highly requested features, such as

container queries, landing in browsers as I write this conclusion. It will be interesting to see whether the uptake for these features will match the demand for them.

Something that has been apparent in this data is how much popular platforms, in particular WordPress, can impact usage statistics. We can see WordPress class and custom property names clearly in the data, what is harder to see are the properties and values used by classes added to the majority of WordPress sites. If WordPress adopts a new feature, as part of one of these standard classes, we should expect to see a sudden uptick in usage.

As noted in last year's conclusion, the data tells a story of gradual, steady adoption of new features (such as grid layout) or best practices (such as using logical rather than physical properties). We look forward to seeing how these changes develop in the years to come.