Bridging the gap

- Flexbox gap
- Why are there gaps in support?
- The problems
  - Interoperability
  - Messaging

Watch the talk Bridging the Gap from Google I/O '22:

https://www.youtube.com/watch?v=fGlhRnp5M-g

When we talk to developers, be that individually or via surveys such as the State of CSS, we repeatedly hear the same things. Developers find it hard to make websites and applications that work well across browsers, and find it hard to know when exciting new features are safe to use.

Flexbox gap

As an example of a problematic property, the `gap` property lets you make gaps between grid or flex items, or columns in a multicol container. While we have had `column-gap` in multicol for a long time, the property first appeared in grid layout as `grid-gap`, along with `grid-column-gap` and `grid-row-gap`.

Just after grid layout landed in browsers, the property was renamed as `gap`, along with `row-gap` and `column-gap`. It was then specified to work in flex layout too. The renaming means that we don't have multiple properties doing the same thing.

```
.box {
  display: flex;
  gap: 1em;
}
```

Firefox shipped the property for flex layouts in October 2018. It didn't show up in Chrome until July 2020, followed by Safari in April 2021. This meant there was a gap of two years and six months

before we could safely use `gap`. In reality, for most developers the wait was much longer, due to needing to support browser versions older than the latest one. Supporting `gap` in flex layout was made more problematic by the fact we can't use feature queries to detect gap support in flex layout. As the `gap` property is supported in grid, `@supports (gap:1em)` will return true.

An additional problem is that when a new feature lands in one browser, people start to talk about it and share demos. This often begins long before the feature is in any stable browser at all. This can be confusing for developers, or at the least, frustrating. Repeatedly, shiny new features are being talked about everywhere, and then you discover you can't actually use them due to lack of support.

Why are there gaps in support?

This is not a post pointing the finger at one browser for being slow. If you look across different platform features, you'll discover that different browsers take the lead on different features.

Most features will be prototyped in one browser. For example, the grid layout specification was first created by Microsoft and implemented in an initial form in Internet Explorer 10. An engineer at Mozilla did a lot of work to figure out how subgrid should behave, and so this feature landed in Firefox first. We're seeing Safari take the lead on some exciting new color functions.

While one browser may take the lead on prototyping, representatives of all browsers (and other organizations) in the CSS Working Group discuss CSS features. It's very important that a feature can be implemented in all browsers, and that it isn't designed in such a way as to make it impossible to implement in one browser. That would lead to a permanent gap in support, and lack of adoption of the feature.

When it comes to implementing a feature however, it needs to be prioritized alongside all the other possible features for that browser. And, sometimes things are held up behind other work that needs to be done to make the browser better. A great example of this is the RenderingNG work in Chromium. This has paved the way for the implementation of subgrid; however, the long gap between Firefox and Chromium shipping subgrid is due to the need for grid layout to be reimplemented in the new rendering engine first.

The problems

We have two problems here. The first is the interoperability problem, the fact that it can take a long time from the point at which a feature lands in one browser to when it is available everywhere.

The second is a problem of messaging. How can we make it clear where the gaps in support are? How do we share new features without causing everyone to have to carefully research each thing to work out how well supported it is?

Interoperability

Browsers are already working together to solve the interoperability issue. Last year Compat 2021 helped to close the gap in support on a number of features, including the gap property in flex layout. This year the Interop 2022 effort is focussing on 15 features, and there has already been movement on some of these.

You can follow progress on the Interop 2022 dashboard.

Messaging

The second problem is something that I'm keen to help with when we talk about features here on web.dev and on developer.chrome.com. I want the status of features to be really clear when you read our content, and for us to provide practical ways to navigate support issues.

We've launched a number of fundamental courses, with more to come. These courses help you to learn how to build for the modern web, using core web platform technologies. Check out:

- Learn CSS
- Learn Forms
- Learn Design
- Learn PWA

We're working to focus our content on this site on those things that you can safely use. We're not completely there yet; however, you should start to see a bit of a shift towards the practical over the coming months.

We also contribute to open web documentation through our support of the Open Web Docs project. This ensures we have first-class documentation on MDN, along with up-to-date browser compatibility data. We then use this data here on web.dev to show support for features. Here's the current support of `gap` in flex layout.

If you want to know more about Chrome's vision for the web, the things we are experimenting with in Origin and Developer trials, then you'll increasingly find that content on our sister site—developer.chrome.com. The content there may well be experimental, or only supported in Chrome right now, but we'd love you to explore it and offer feedback.

It really is an exciting time for the web right now. We hope we can help to bring key features to you more quickly, and be clear about the gaps that do exist, making web development more fun and less frustrating.

Photo by Cristofer Maximilian.

□□□ #3
□ □□□ □□□ 2026-10-31 21:40:25 UTC
□ cocobook-sync □□□ 2026-11-04 17:36:37 UTC