Launching Interop 2026

By Jake Archibald

Posted on February 12, 2026 in Featured Article, Firefox, Uncategorized, and Web Developers

The Interop Project is a cross-browser initiative to improve web compatibility in areas that offer the most benefit to both users and developers.

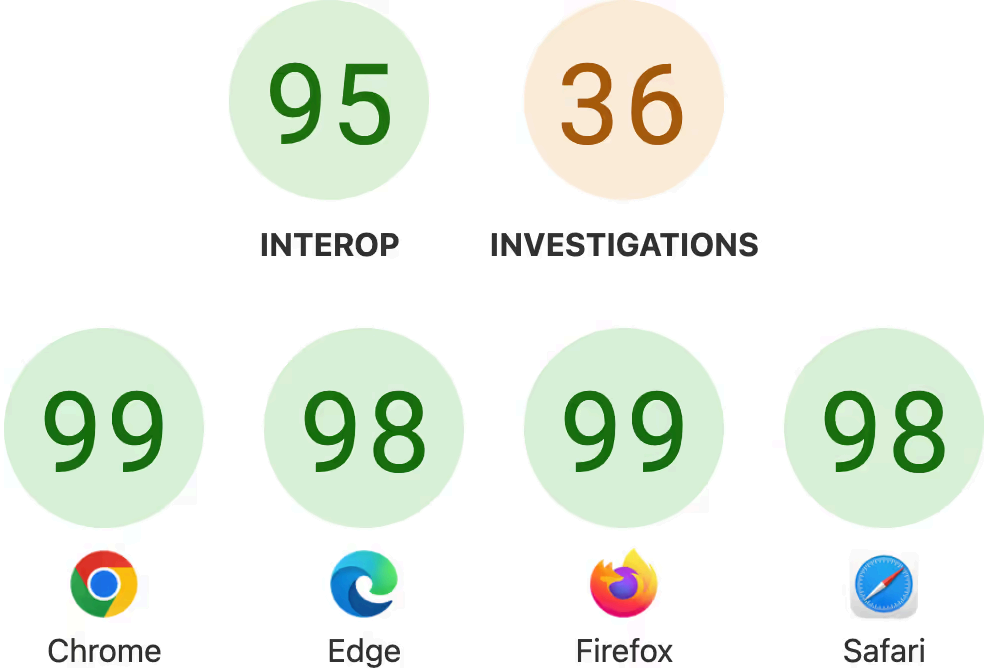
The group, including Apple, Google, Igalia, Microsoft, and Mozilla, takes proposals of features that are well defined in a sufficiently stable web standard, and have good test suite coverage. Then, we come up with a subset of those proposals that balances web developer priorities (via surveys and bug reports) with our collective resources.

We focus on features that are well-represented in Web Platform Tests as the pass-rate is how we measure progress, which you can track on the Interop dashboard.

Once we have an agreed set of focus areas, we use those tests to track progress in each browser throughout the year. And after that, we do it all again!

But, before we talk about 2026, let's take a look back at Interop 2025…

Interop 2025



Firefox started Interop 2025 with a score of 46, so we're really proud to finish the cycle on 99. But the number that really matters is the overall Interop score, which is a combined score for all four browsers – and the higher this number is, the fewer developer hours are lost to frustrating browser differences.

The overall Interop score started at 25, and it's now 95. As a result, huge web platform features became available cross-browser, such as Same-Document View Transitions, CSS Anchor Positioning, the Navigation API, CSS @scope, and the URLPattern API.

That's the headline-grabbing part, but in my experience, it's way more frustrating when a feature is claimed to be supported, but doesn't work as expected. That's why Interop 2025 also focused on improving the reliability of existing features like WebRTC, CSS Flexbox, CSS Grid, Pointer Events, CSS backdrop-filter, and more.

But it's not just about passing tests

With some focus areas, in particular CSS Anchor Positioning and the Navigation API, we noticed that it was possible to achieve a good score on the tests while having inconsistent behavior compared to other browsers.

In some cases this was due to missing tests, but in some cases the tests contradicted the spec. This usually happens when tests are written against a particular implementation, rather than the specified behavior.

I experienced this personally before I joined Mozilla – I tried to use CSS Anchor Positioning back when it was only shipping in Chrome and Safari, and even with simple use-cases, the results were wildly inconsistent.

Although it caused delays in these features landing in Firefox, we spent time highlighting these problems by filing issues against the relevant specs, and ensured they got priority in their working groups. As a result, specs became less ambiguous, tests were improved, and browser behavior became more reliable for developers.

Okay, that's enough looking at the past. Let's move on to…

Interop 2026

Over 150 proposals were submitted for Interop 2026. We looked through developer feedback, on the issues themselves, and developer surveys like The State of HTML and The State of CSS. As an experiment for 2026, we at Mozilla also invited developers to stack-rank the proposals, the results of which we used in combination with the other data to compare developer preferences between individual features – this is something we want to expand on in the future.

After carefully examining all the proposals, the Interop group has agreed on 20 focus areas (formed of 33 proposals) and 4 investigation areas. See the Interop repository for the full list, but here are the highlights:

New features

As with 2025, part of the effort is about bringing new features to all browser engines.

Cross-document View Transitions allow transitions to work across documents, without any JavaScript. The sub-features `rel="expect"` and `blocking="render"` are included in this focus area..

Scroll-driven animations allow you to drive animations based on the user's scroll position. This replaces heavy JavaScript solutions that run on the main thread.

WebTransport provides a low-level API over HTTP/3, allowing for multiple unidirectional streams, and optional out-of-order delivery. This is a modern alternative to WebSockets.

CSS container style queries allow you to apply a block of styles depending on the computed values of custom properties on the nearest container. This means, for example, you can have a simple `--theme` property that impacts a range of other properties.

JavaScript Promise Integration for Wasm allows WebAssembly to asynchronously 'suspend', waiting on the result of an external promise. This

simplifies the compilation of languages like C/C++ which expect APIs to run synchronously.

CSS attr() has been supported across browsers for over 15 years, but only for pseudo-element content. For Interop 2026, we're focusing on more recent changes that allow attribute values to be used in most CSS values (with URLs being an exception).

CSS custom highlights let you register a bunch of DOM ranges as a named highlight, which you can style via the `::highlight(name)` pseudo-element. The styling is limited, but it means these ranges can span between elements, don't impact layout, and don't disrupt things like text selection.

Scoped Custom Element Registries allow different parts of your DOM tree (such as a shadow root) to use a different set of custom elements definitions, meaning the same tag name can refer to different custom elements depending on where they are in the DOM.

CSS shape() is a reimagining of `path()` that, rather than using SVG path syntax, uses a CSS syntax, allowing for mixed units and `calc()`. In practice, this makes it much easier to design responsive `clip-paths` and `offset-paths`.

And more, including CSS contrast-color, accent-color, dialog closedby, popover="hint", fetch upload streams, IDB getAllRecords(), media pseudo-classes such as :playing, and the Navigation API's precommitHandler.

Existing feature reliability improvements

Like in previous years, the backbone of Interop is in improving the reliability of existing features, removing frustrations for web developers.

In 2026, we'll be focusing these efforts on particular edge cases in:

- Range headers & form data in fetch
- The Navigation API
- CSS scroll snap
- CSS anchor positioning
- Same-document View Transitions
- JavaScript top-level await
- The event loop
- WebRTC
- CSS user-select
- CSS zoom

Some of these are carried over from 2025 focus areas, as shortcomings in the tests and specs were fixed, but too late to be included in Interop 2025.

Again, these are less headline-grabbing than the shiny new features, but it's these edge cases where us web developers lose *hours of our time*. Frustrating, frustrating, hours.

Interop investigations

Sometimes, we see a focus area proposal that's clearly important, but doesn't fit the requirements of Interop. This is usually because the tests for the feature aren't sufficient, are in the wrong format, or browsers are missing automation features that are needed to make the feature testable.

In these cases, we identify what's missing, and set up an investigation area.

For interop 2026, we're looking at...

Accessibility. This is a continuation of work in 2025. Ultimately, we want browsers to produce consistent accessibility trees from the same DOM and CSS, but before we can write tests for this, we need to improve our testing infrastructure.

Mobile testing. Another continuation from 2025. In particular, in 2026, we want to figure out an approach for testing viewport changes caused by dynamic UI, such as the location bar and virtual keyboard.

JPEG XL. The current tests for this are sparse. Existing decoders have more comprehensive test suites, but we need to figure out how these relate to browsers. For example, progressive rendering is an important feature for developers, but how and when browsers should do this (to avoid performance issues) is currently being debated.

WebVTT. This feature allows for text to be synchronised to video content. The investigation is to go through the test suite and ensure it's fit for purpose, and amend it where necessary.

It begins... again

The selected focus areas mean we've committed to more work compared to the other browsers, which is quite the challenge being the only engine that isn't owned by billionaires. But it's a challenge we're happy to take on!

Together with other members of the Interop group, we're looking forward to delivering features and fixes over the next year. You can follow along with the progress of all browsers on the Interop dashboard.

If your favorite feature is missing from Interop 2026, that doesn't mean it won't be worked on. JPEG XL is a good example of this. The current test suite meant it wasn't a good fit for Interop 2026, but we've challenged the JPEG XL

team at Google Research to build a memory-safe decoder in Rust, which we're currently experimenting with in Firefox, as is Chrome.

Interop isn't the limit of what we're working on, but it is a cross-browser commitment.

If you're interested in details of features as they land in Firefox, and discussions of future features from spec groups, you can follow us on:

- BlueSky
- Mastodon
- LinkedIn
- Threads
- YouTube
- TikTok
- Instagram

Partner Announcements

This is a team effort, and we've all made announcement posts like this one. Get other members' take on it:

- Apple
- Google
- Igalia
- Microsoft

About Jake Archibald

More articles by Jake Archibald…