Improving the web developer experience one feature at a time

Patrick Brosset – FOSDEM 2026

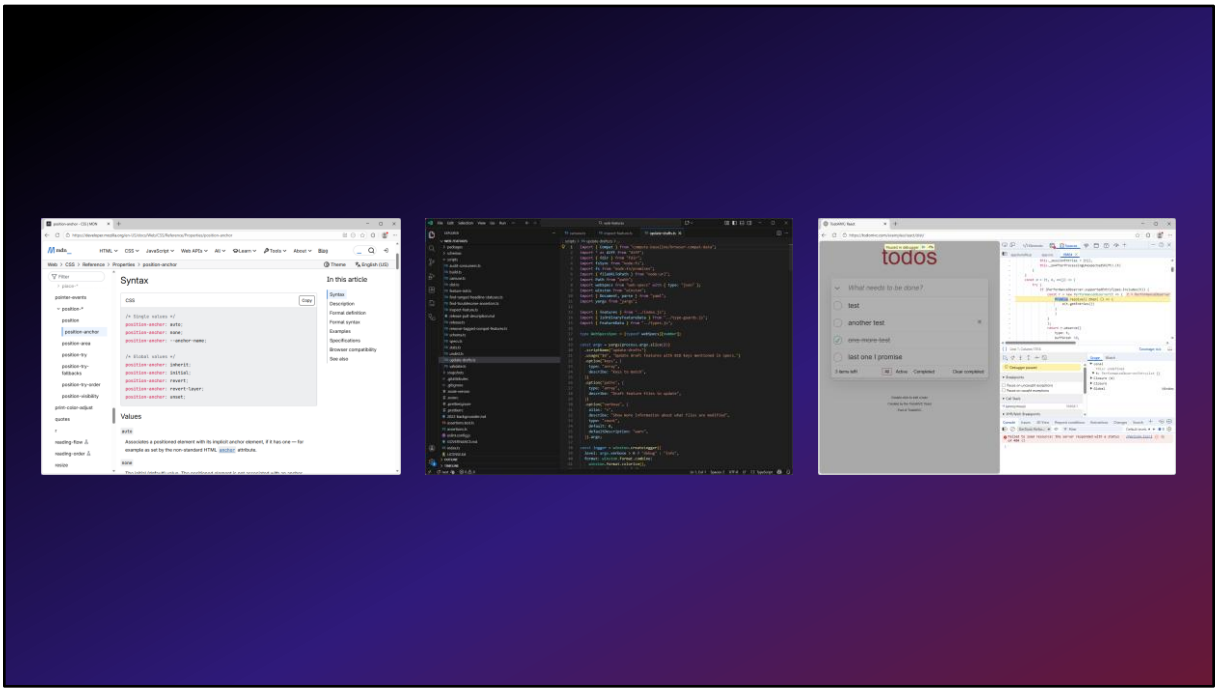
Hello everyone.
It's great to be here, and I'm very honored to open this Browser and Web Platform track with my talk.

As front-end developers, building for the web, we rely on open-source software a lot.
Much of the engines of the browsers we build for are open-source projects.
The very frameworks and libraries we rely on for our sites and apps are also very often open-source.

This entire conference is proof that this is a thriving ecosystem.

But it's not only about the code.
Code is not what I’m here to talk about.

There are two things web devs, and browser engineers too, rely on a lot that's not code: docs and data.
So let's talk about data.

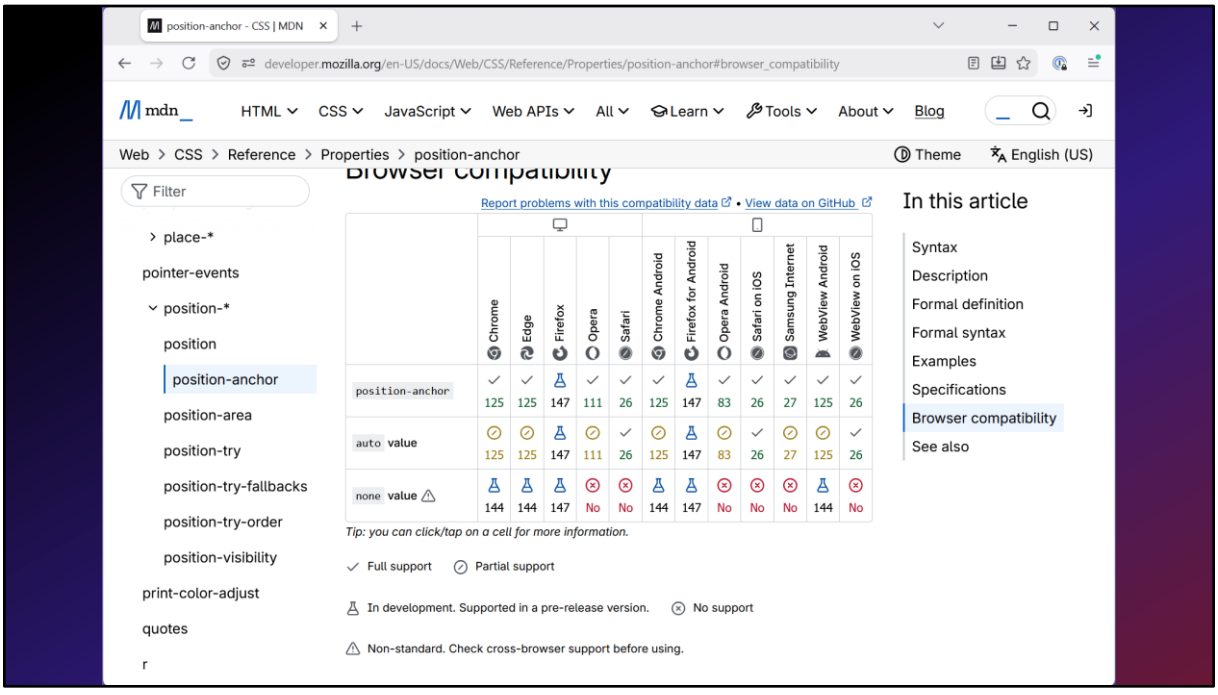


Think about a typical day as a web developer.

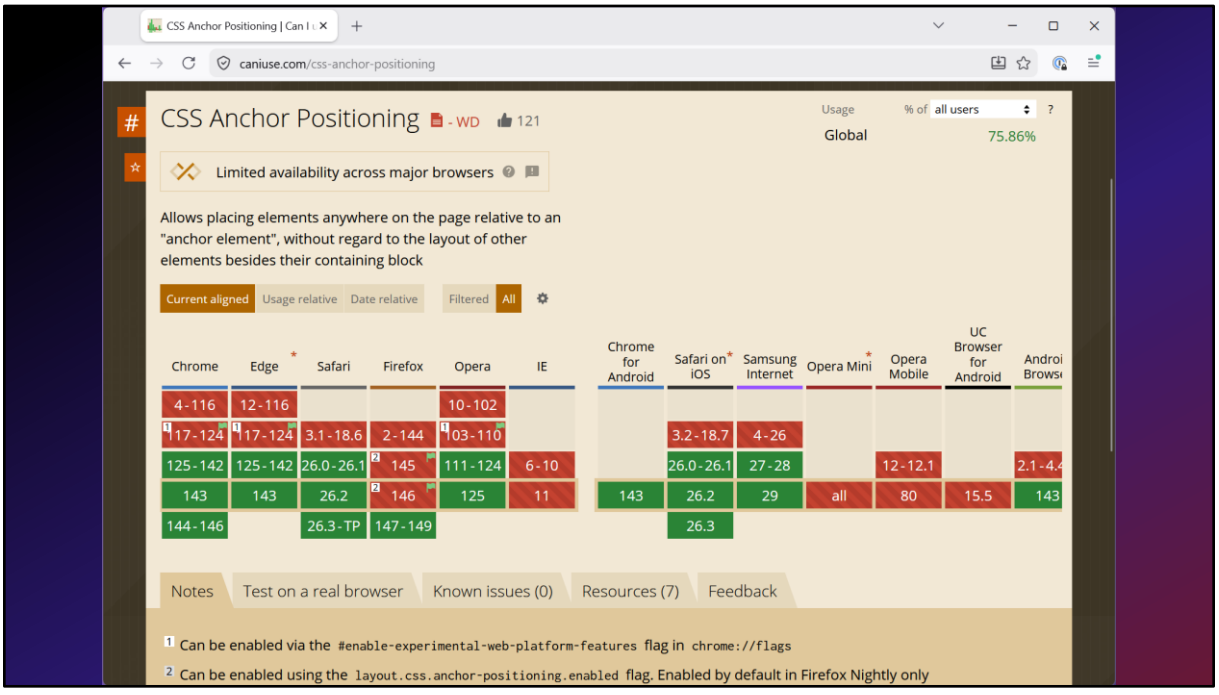
Obviously, you write code. But you also depend on tools to write and debug that code,
and on documentation to even know how to write it in the first place.

Both take data to be as useful as possible.
Both depend on massive amounts of data, to give you the best environment for you to learn, write, and debug.

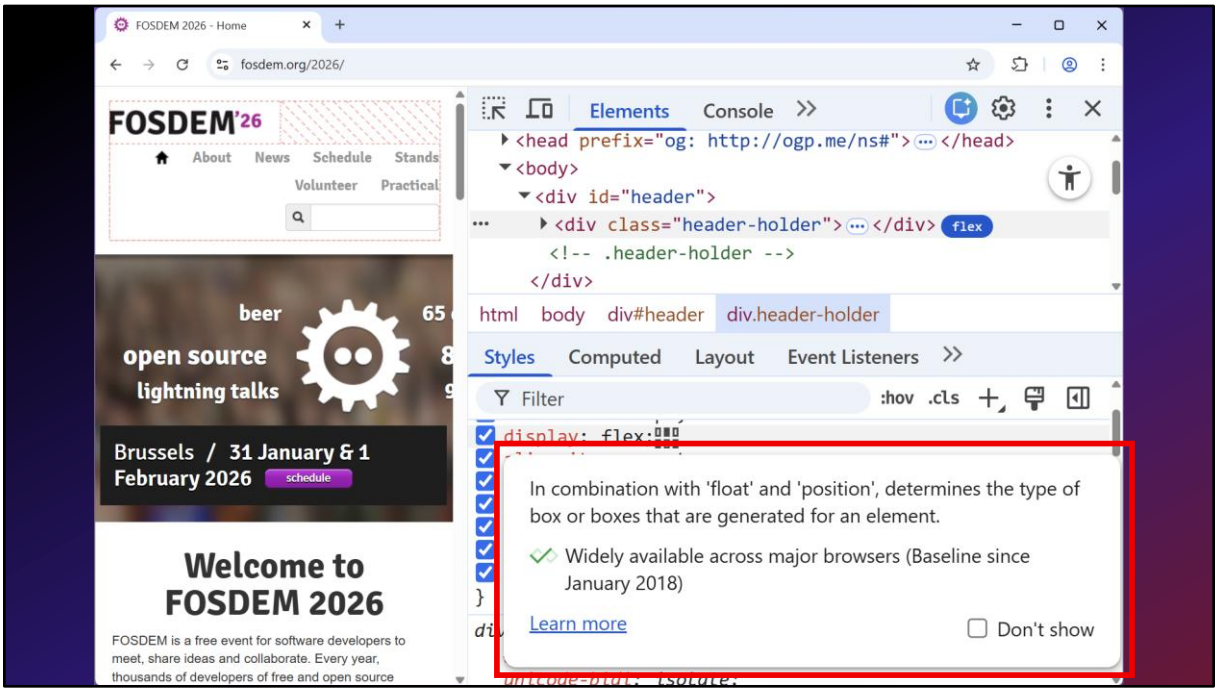
These tools and docs, they need an understanding of what you're doing and the features you're using, in order to help you get things done.



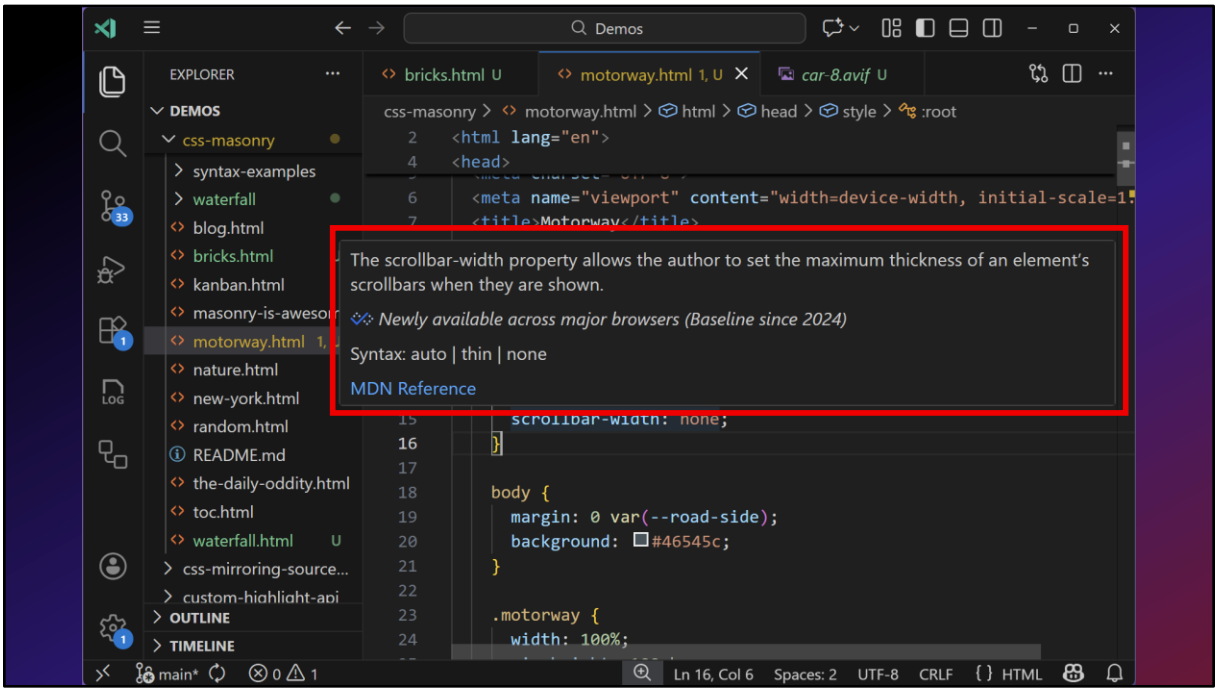
Familiar with this?


CSS Anchor Positioning | Can I u
caniuse.com/css-anchor-positioning
CSS Anchor Positioning - WD 121
Usage
% of all users
Global
75.86%
Limited availability across major browsers
Allows placing elements anywhere on the page relative to an "anchor element", without regard to the layout of other elements besides their containing block
Current aligned
Usage relative
Date relative
Filtered
All
Chrome
Edge
Safari
Firefox
Opera
IE
Chrome for Android
Safari on iOS
Samsung Internet
Opera Mini
Opera Mobile
UC Browser for Android
4-116
12-116
10-102
17-124
17-124
3.1-18.6
2-144
103-110
3.2-18.7
4-26
125-142
125-142
26.0-26.1
145
111-124
6-10
26.0-26.1
27-28
12-12.1
143
143
26.2
146
125
11
143
26.2
29
all
80
15.5
143
144-146
26.3-TP
147-149
26.3
Notes
Test on a real browser
Known issues (0)
Resources (7)
Feedback
1 Can be enabled via the #enable-experimental-web-platform-features flag in chrome://flags
2 Can be enabled using the layout.css.anchor-positioning.enabled flag. Enabled by default in Firefox Nightly only

And this?

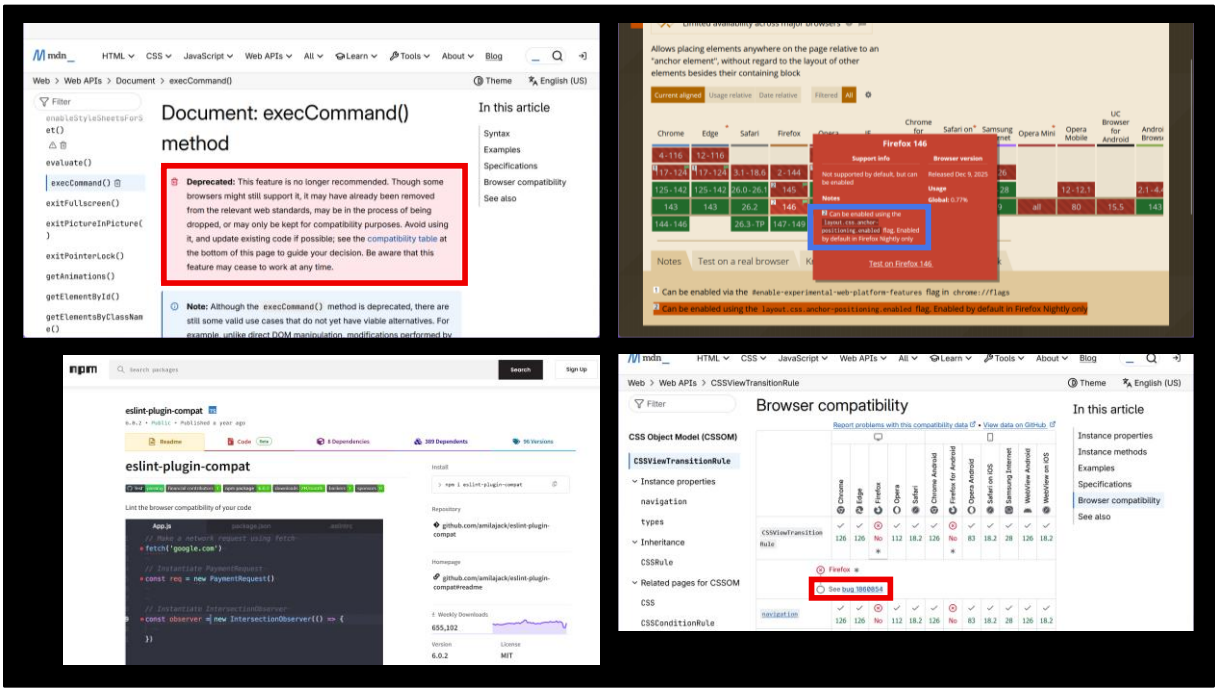


What about docs/compat tooltips in browser DevTools?



And hover cards in your editor or IDEs?

All these examples rely, in some way, on open-source data which you benefit from every day as a web developer
and without which much of your work would be a lot harder.



These were kind of obvious examples, here are more subtle ones which might still have a big impact on web devs experience:

Knowing that a feature is deprecated, or currently behind a flag, or blocked by a particular bug are all pieces of data that can come in very useful in the life of a web developer.

Imagine docs without up to date browser compatibility data, or deprecated warnings.
Imagine CI or linter tools that don't know how to check if the features you use work in your target browsers.

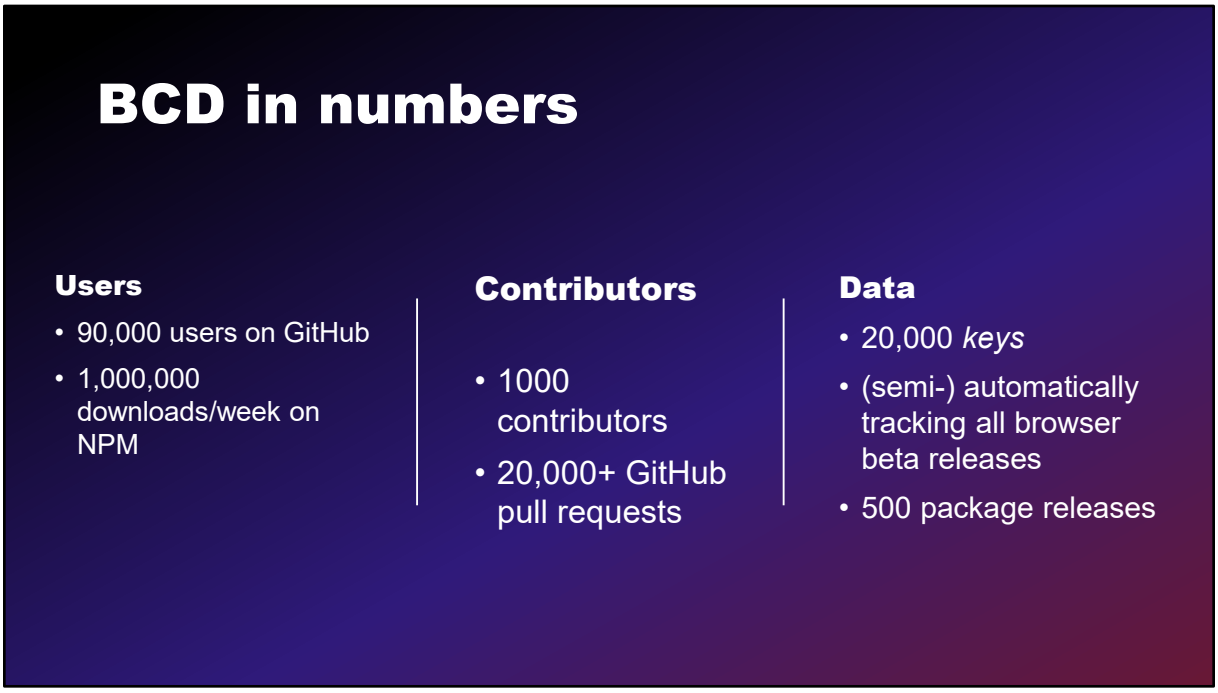
If you think about it, up to date web data is a critical infrastructure of the web industry which, as developers, we don't really think about.

browser-compat-data (BCD)

Let’s look at one of the critical data sources, probably the most important:

BCD, or browser-compat-data.

It’s the data behind all the compatibility tables on MDN
but also on many of the caniuse.com pages.
It’s used by VS Code, ESlint, babel, Bun, and many others.



- Used by 90,000 people, according to GitHub
- Downloaded 1,000,000 times per week, according to NPM
- More than 1000 contributors on GitHub
- 20,000 commits.
- 20,000 “keys”, where each key corresponds to one tiny specific aspect of a web technology, like a single css property, or a single JS method parameter, etc.
- Almost 500 releases
- Semi-automatically tracking browser beta releases, for the most up to date data.

So, a very active project.

Main contributors

Most active contributors being:

- Open Web Docs, a vendor-neutral initiative to ensure the long-term health of web platform docs and data.
- MDN team. BCD is under the MDN org on GitHub.
- A few individuals at browser vendor companies, super helpful as they often know best as to which versions a given feature is supported in.
- A bunch of extremely helpful external contributors.

Using BCD

1. **Install the dependency**

```
npm install @mdn/browser-compat-data
```

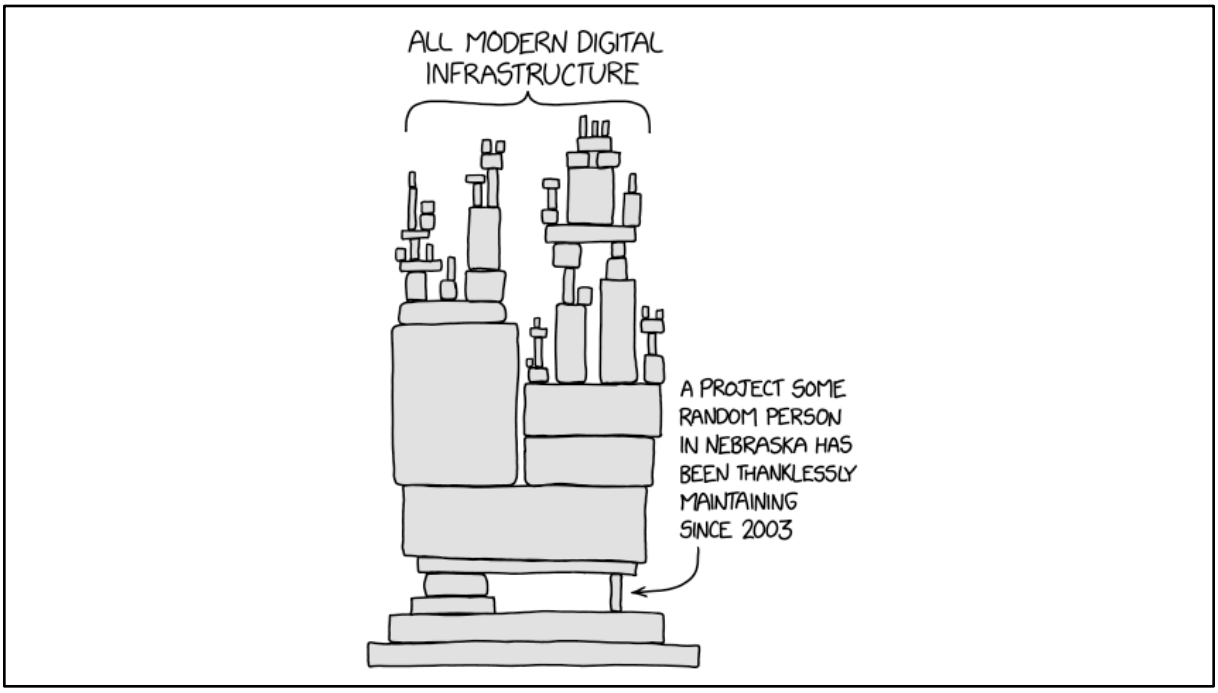
2. **Import the data in your project**

```
import bcd from '@mdn/browser-compat-data' with { type: 'json' };
```

3. **Use the data!**

```
const support = bcd.css.properties.background.__compat;
```

Super easy to use too.



Intermission

Open-source projects need funding to last, remain relevant, secure, etc.
There's a whole track about this at this very conference.

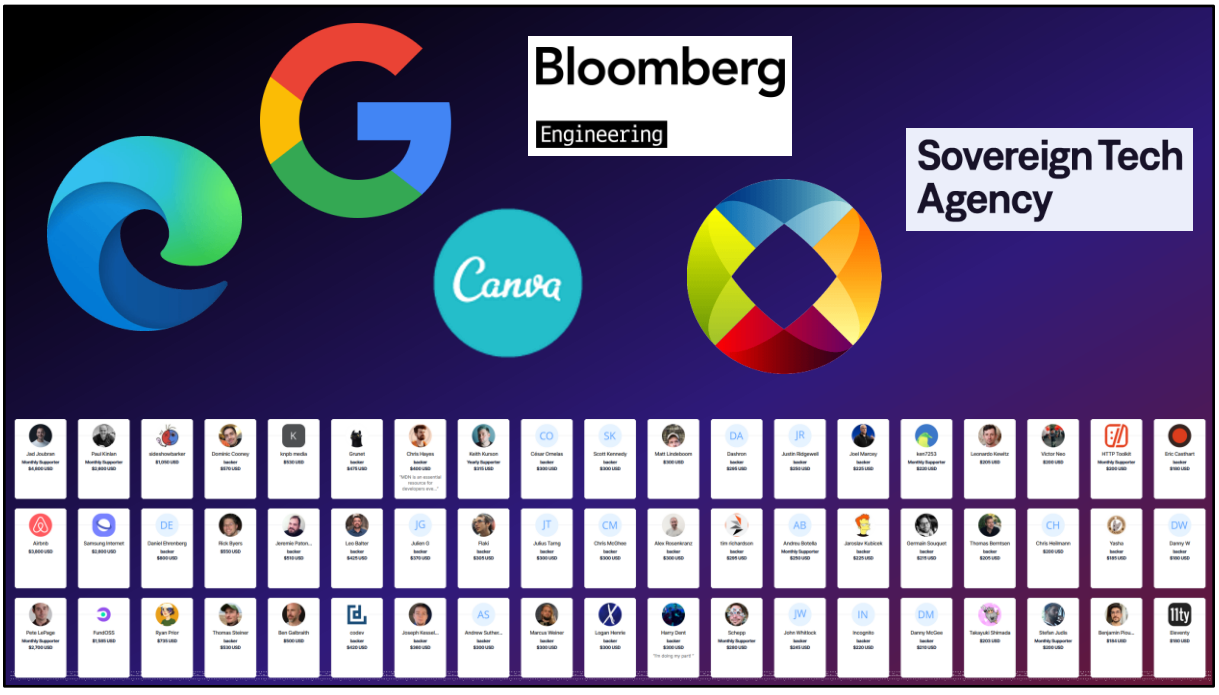
It's very simple: if work is needed, then someone must do that work and get paid for it.

Funding remains one of the biggest challenges, and something these projects constantly have in mind,
and need to prioritize, sometimes instead of doing technical work for the project.

So, big shoutout to all the sponsors and maintainers of BCD and other similar projects.

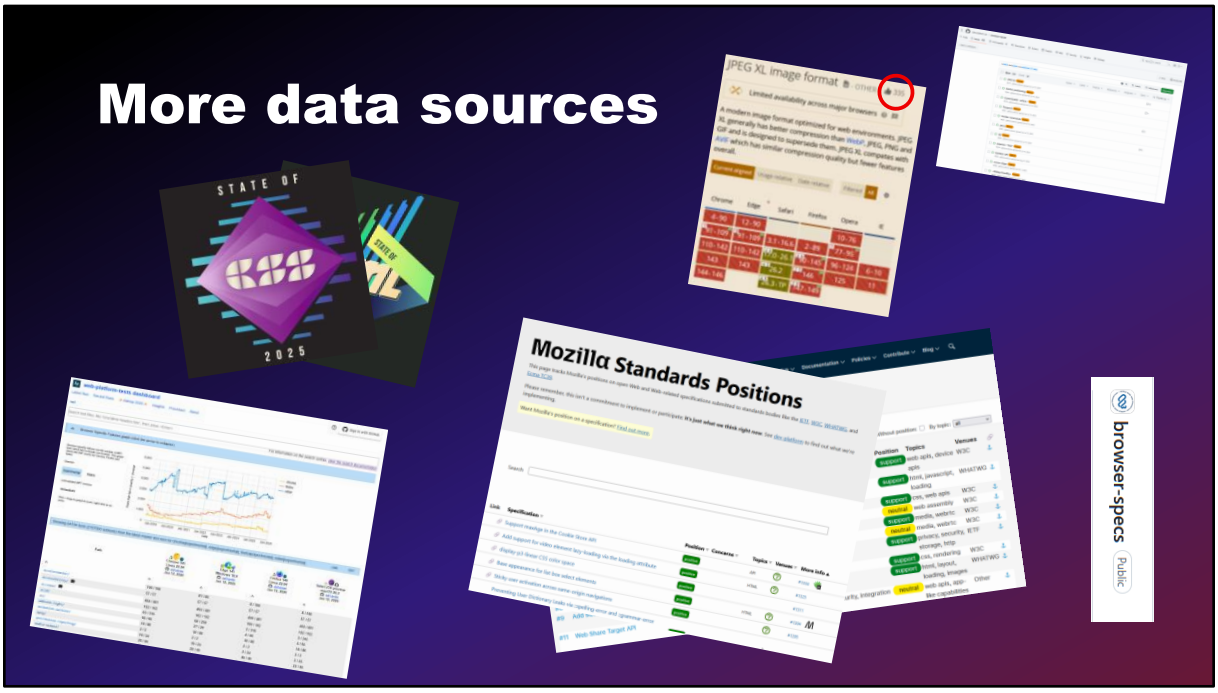
For BCD, Mozilla employs a few engineers to help maintain the data and its infrastructure, but it wouldn't be possible without outside help too.

Open Web Docs, which I also mentioned before, does a lot of the work that's needed.
Very active with updating data, reviewing PRs, but also maintaining and running a tool that updates BCD based on latest browser beta releases.



OWD is (or has been) funded by these companies, and a lot of individual donors via opencollective and github.

If you want to help make a difference, you can donate to OWD.
Or, even better, convince your company to become a sponsor.



And there are more data sources out there, which benefit web developers, sometimes indirectly.
Some of which browser companies make extensive use of, to prioritize their work.

- State of CSS/JS/HTML surveys
  useful for prioritization.
  Results are accessible as JSON on the repo, which can then be queried via GraphQL.
- Developer signals repo.
  Very new. Starting to get integrated into various places, such as caniuse.com.
  Allows us to capture use cases from developers about features which aren't yet interoperable.
- Browser specs.
  Helps with tracking the status of specs, ...
- Standards positions.
  Makes it easy for developers and other browser vendors understand the positions of Mozilla and Apple on various features, and whether they are likely

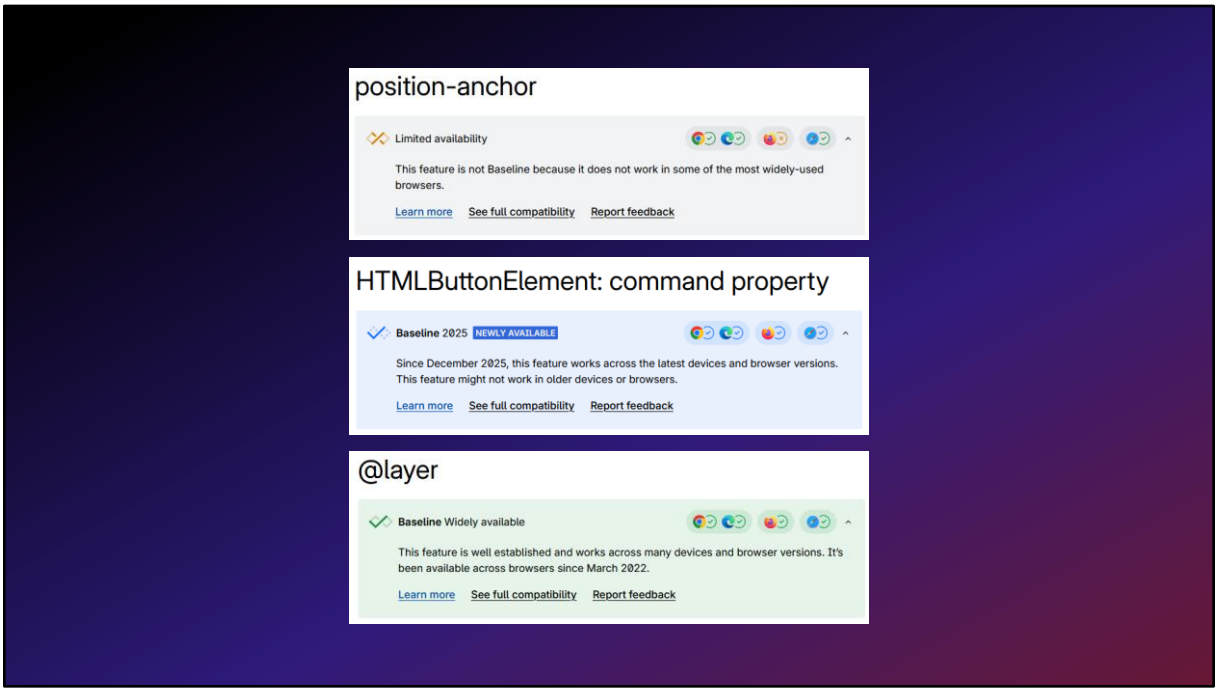
to ever become interoperable in their current state. Available as JSON data too.
- WPT test results
  Allow us to track the implementation quality of features in browsers.

So there's a lot of useful web-related data sources we can use to eventually make the web platform a better place.
It'd be nice if they were all interconnected though. More on that next.

Let’s talk about one more web data-related open source project: web-features

It's unlikely that you've heard about it
But you might have heard about Baseline.

Baseline is this simplified status for features of the web platform
It gives you some sense of a feature's availability across browsers.

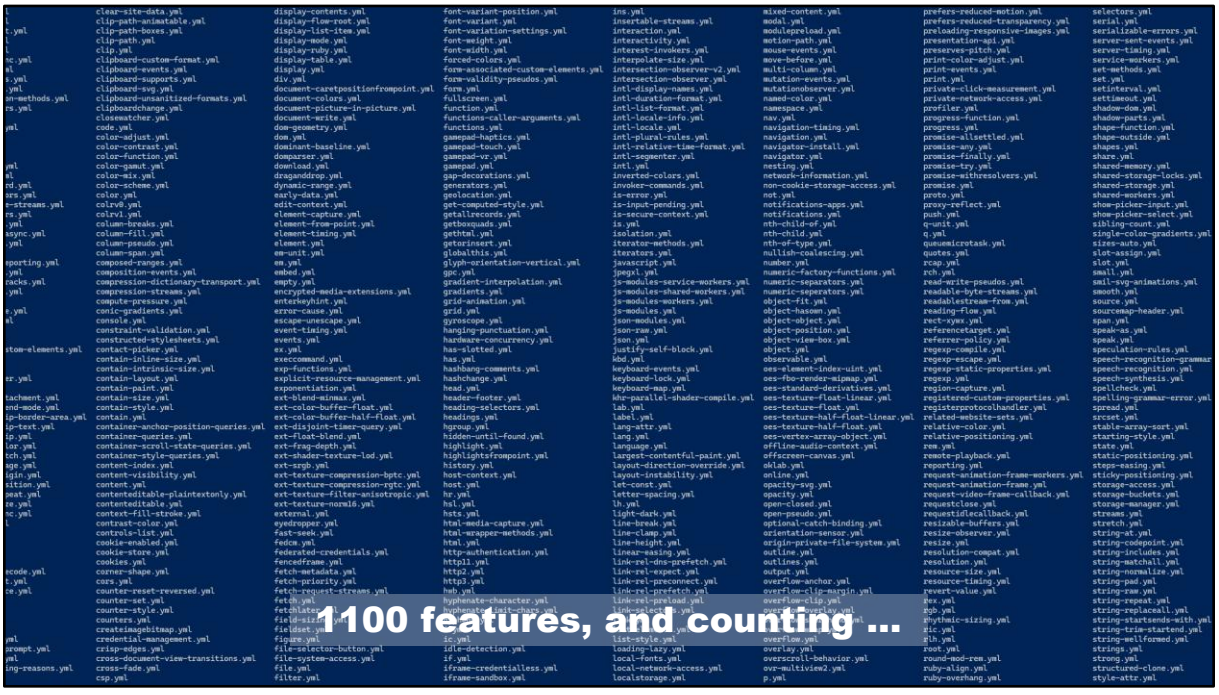


For example, Anchor Positioning is not Baseline, because it's not yet supported across all of the browsers that Baseline considers.

Invoker commands is Baseline Newly Available, because it's now supported in all browsers that Baseline considers, but it's not been very long yet.

Finally, CSS layers is Baseline Widely Available, because it's been supported in all browsers that Baseline consider for long enough that's considered much more stable and safer to use.

It doesn't replace careful testing and consideration of your specific use case, but it makes certain decisions faster.



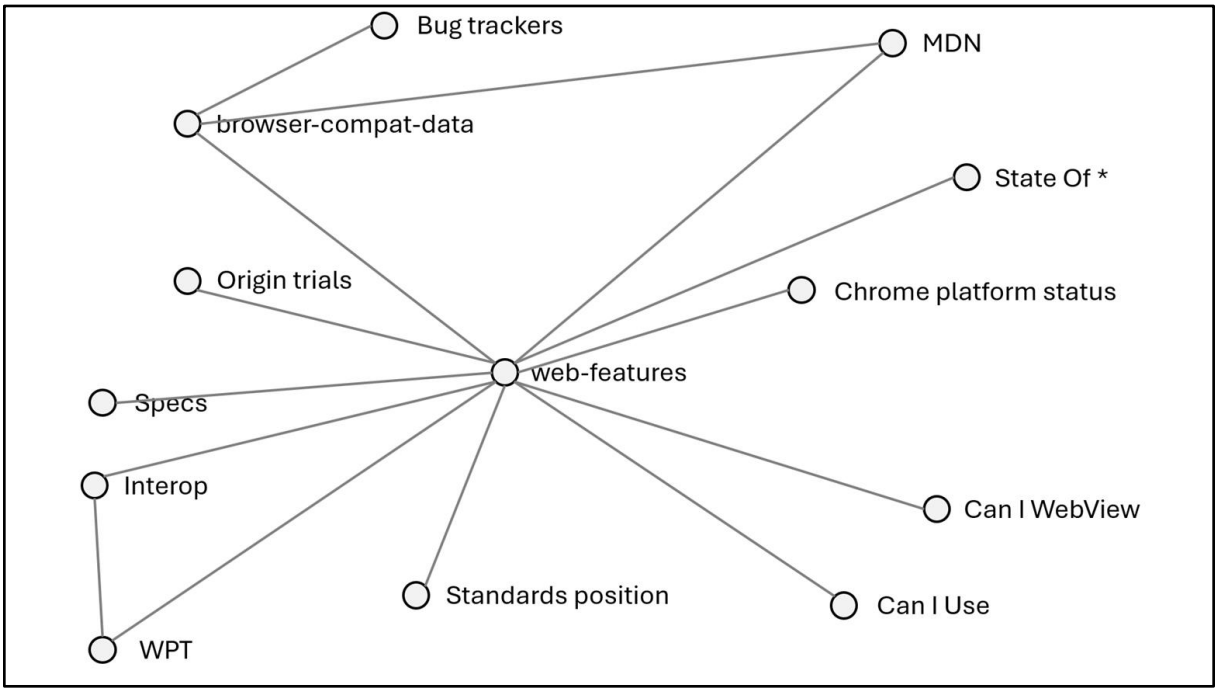
So, Baseline is made possible by the web-features project.
But what is web-features exactly?

Web-features is a repository that lists _all_ features of the web.
BCD already does this, but at a level that is extremely granular.
BCD wants to know about each and every attribute, property, event, parameter, and each and every little option.
Web-features doesn't need this. Instead, it wants to track features at a level which makes sense to developers.

If most developers think of, say, view transitions as one feature, then there should be a feature named view transitions in the repo.

The great thing about that isn't the amount of data that comes with each feature.
Because it's mostly just name, description, spec, and the compat data that supports Baseline.
The great thing is that each feature is identified by a unique ID we can use to link to other things.

Now we have a catalog for all the features of the web platform.

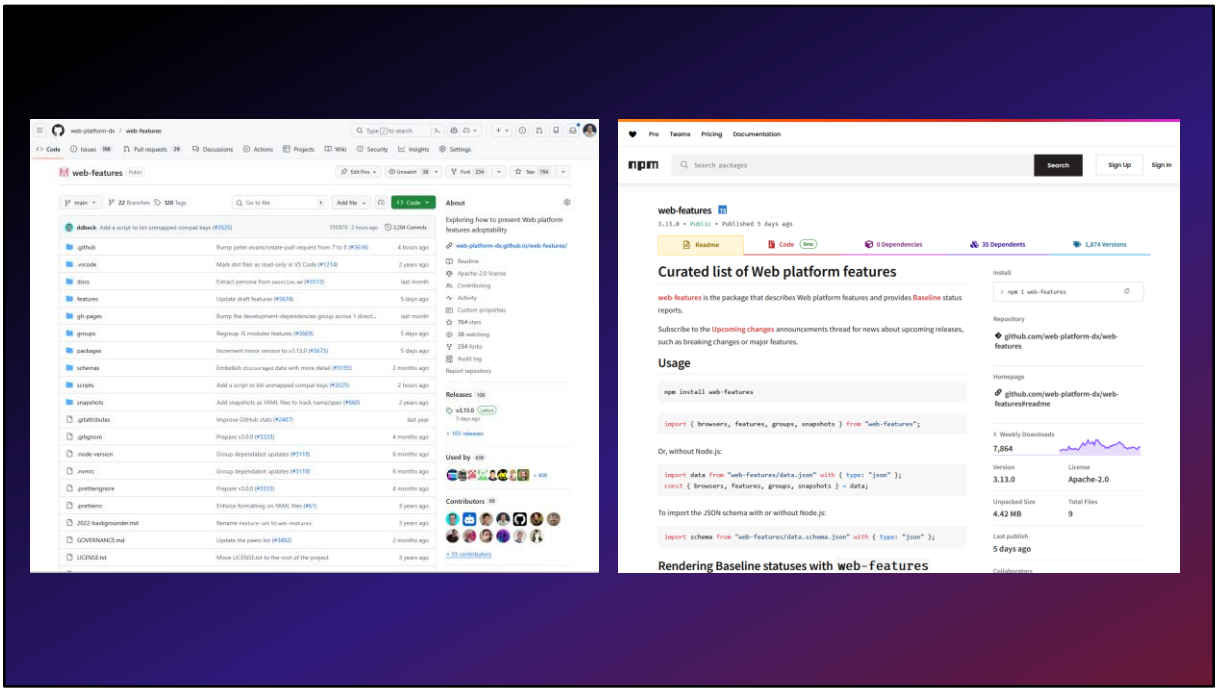


And this allows us to interconnect the other things I mentioned before!

By using web-features IDs, we can connect to docs, Web Platform Test results, State of CSS/JS/HTML survey results, standards positions, developer signals, browser bugs, caniuse pages, specifications, Etc.

The data can be seen as a reference data source for the entire web platform, a hub that connects all other data sources together.

And our goal with the project, through consistent communication about features, is to make it easier for all web devs to understand the web platform, find information about it, make decisions, and be confident about it.



The project lives on GitHub.
Owned by the WebDX community group at W3C, which I co-chair, and which you're more than welcome to join.

70 contributors.
Used by 400 people.
230 forks.
Apache 2.0 license.
With at least 1 release per week to always make sure the compat data is up to date.

Can be used as a dependency via NPM. Or download the JSON data directly from GitHub.
Produced more than 100 releases so far.
Easy to use too.

Using web-features

1. **Install the dependency**

```
npm install web-features
```

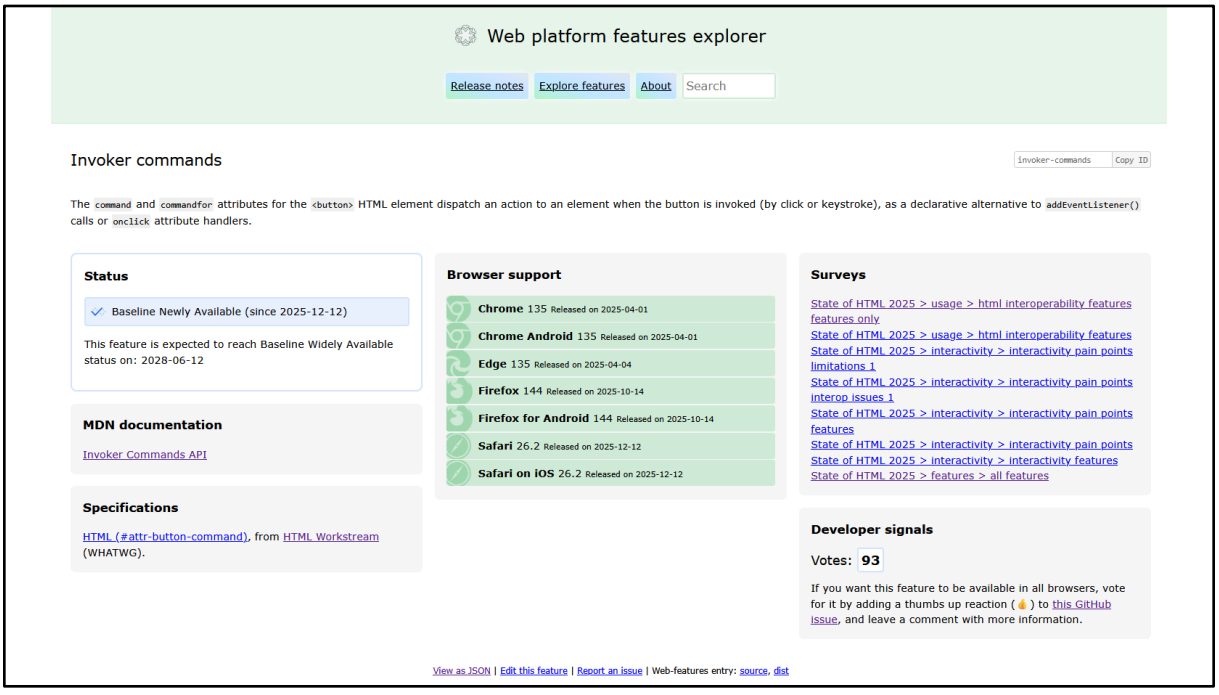
2. **Import the data in your project**

```
import { features } from 'web-features';
```

3. **Use the data!**

```
const data = features.popover;
```

Usage example.
Like BCD, very simple.
NPM is the simplest, but can only download the JSON from the github release.

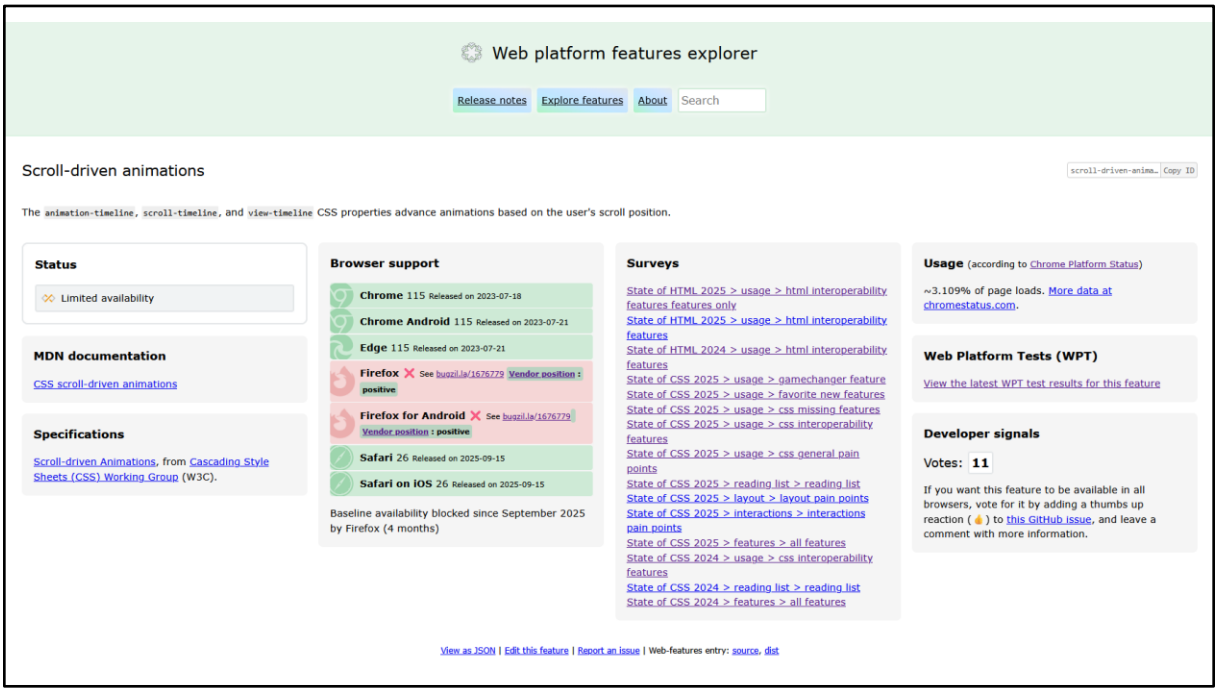


I talked about interconnecting data before, and here is an illustration of this.

On the WebDX group, we also maintain a site called the web platform features explore, which lists all our features + connected data sources.

Here it shows the entry for the invoker commands feature.
And you can see how it not only shows the feature's name, description and baseline status, but other connected data sources too:

- Mdn docs
- Spec
- Surveys which mention the feature
- Developer signals upvotes



Another good example that shows scroll-driven animations with:

- Links to browser bugs
- Links to vendor standard positions
- Usage metrics from chromestatus.com
- WPT test result link

Who uses web-features?

- MDN
- caniuse.com
- caniwebview.com
- VS Code
- JetBrains IntelliJ-based IDEs
- DevTools
- RUMVision
- RUM Archive Insights
- ESLint CSS plugin
- StyleLint Baseline plugin
- browserslist
- webstastus.dev
- Web platform features explorer
- …

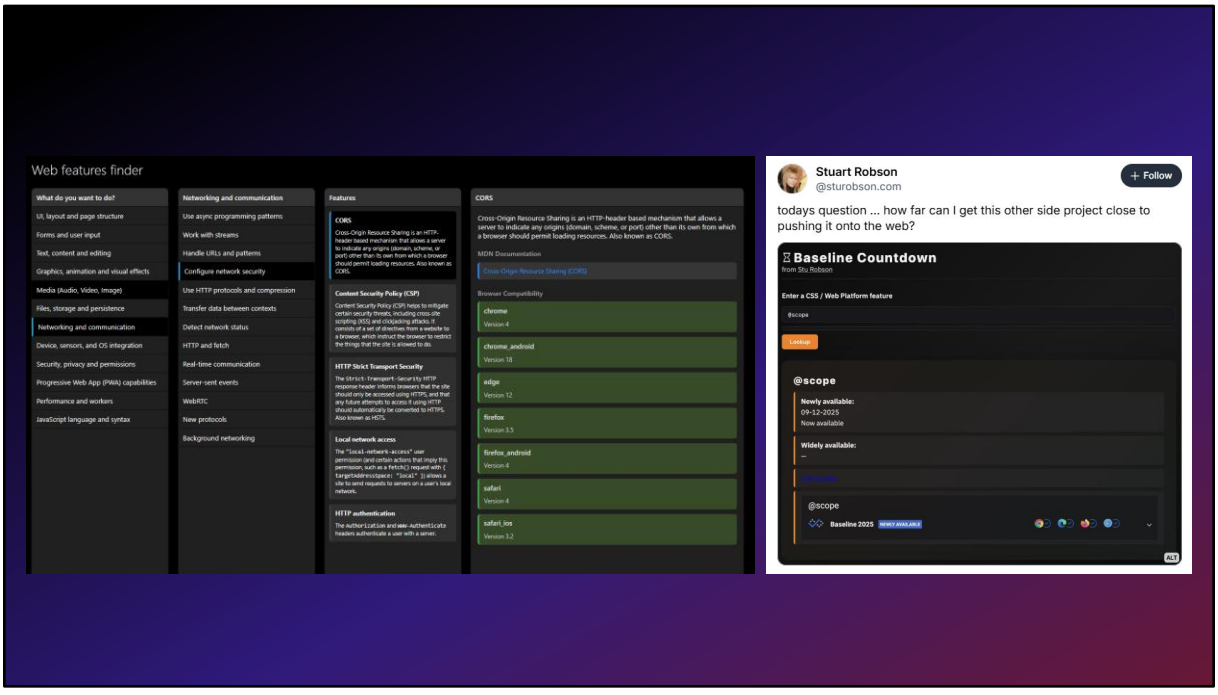
Now let's look at where the web-features data is used today.

The first and most important consumers now are MDN and caniuse.com, which use the data to display Baseline banners.

But many other tools have started to use web-features data now too:
VSCode, WebStorm, browser DevTools.
There's an ESLint Baseline rule too.

Dashboards too:
explorer (feature pages, release notes)
webstatus (with wpt results)

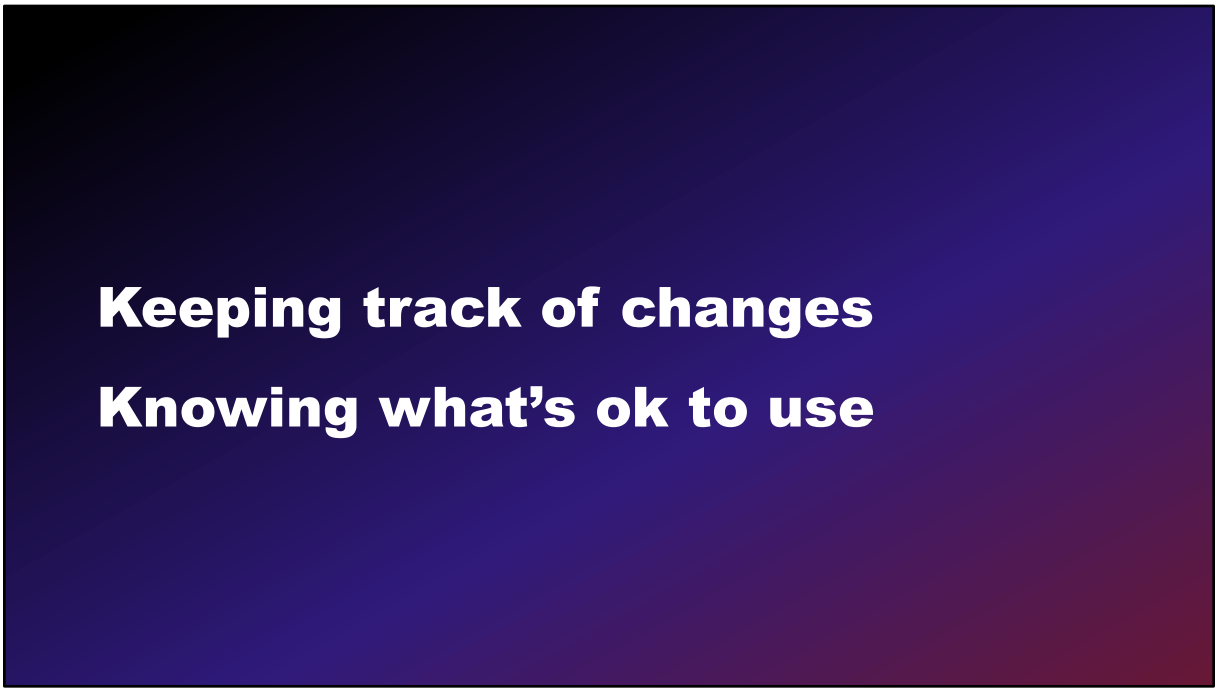
And more coming all the time.



And you can build your own thing with the data too!

On the left is an example of a feature finder idea I had the other day and started to hack on.
It lets you navigate through the web platform, organized by web dev activities, like networking, or file handling, etc.,
and then find the features you can use to implement your tasks.

On the right is someone on social network who shared they were working on a Baseline countdown web page,
To show the time remaining before a feature reaches the Baseline Widely Available stage.



So why is this useful again?

Some of the most frustrating pain points which developers share through surveys and other feedback channels
beside the lack of interoperability
is keeping track of the rapid pace of changes, and knowing what can be used in production safely.

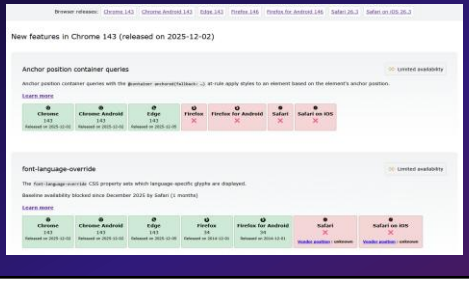
Baseline helps with the latter, practically, by giving a simplified status in prominent places such as editors and docs.

Dashboard sites and other tools built with web-features, can help with the former.

Release notes

RSS feed

Browser releases



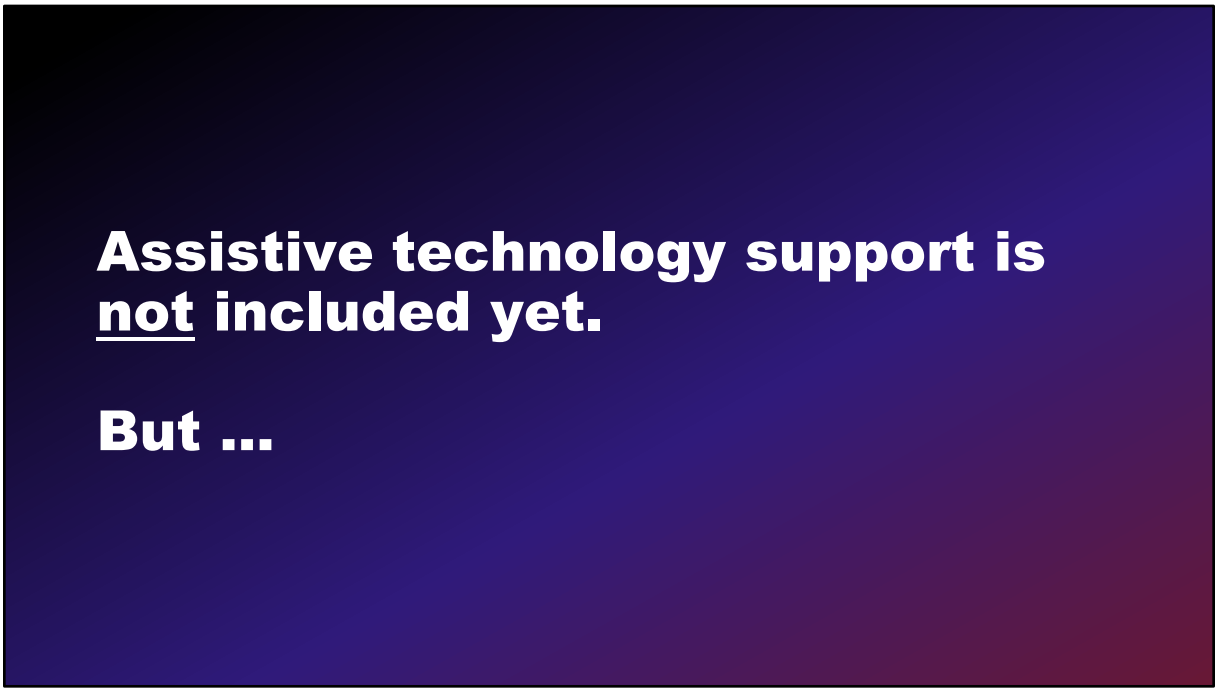
Taking the example of the web features explorer website I mentioned before,

We're experimenting with release notes across browsers.
But also individual brower releases.
And RSS feeds so you can keep track at your own pace, in your own RSS client.

So these are the types of things we can do with the data.
I'm hopeful that the web community can innovate in this area and gradually make it more and more enjoyable to work with the web platform.

What’s not covered (yet)

Alright, so web-features is great!
But like with any in-progress project, there’s always more to do.
So, let’s look at what the project doesn’t cover today yet.



Assistive technology support is one of the most important one.
The Baseline status that web-features computes is based on BCD. BCD does care about bugs in browsers that may prevent a web feature from being accessibility supported.
So that's a start.
But we don't have data about how features work across the matrix of browsers + OS + assistive technologies. So, that part is missing from Baseline.

We're working together with Lola Odelola, who's actually here today and giving a talk later.
Lola is working on a project called ACD, Accessibility Compat Data, which eventually should get us in a position to think about what it means to include accessibility considerations in Baseline.

Progressive enhancement is another pretty important one.
Because a feature doesn't have a Baseline status doesn't mean you can't use it.
You could very successfully use it as a progressive enhancement, or with a fallback or a polyfill.

So, never think of Baseline as a YES/NO status.
Think of it as a YES/Maybe YES.
Nothing replaces careful consideration of your specific audience and use case.

Also, we've been discussing about linking web-features to polyfills and other useful resources,
so that dashboards and tools can start giving you more actionable information than just a thing isn't Baseline.

Partial support, form factor-specific features, etc.

Web features sometimes take years to appear in browsers,
and even when they do, it takes even more years for them to mature enough to be complete and fully interoperable across all browsers.

That means that it's common for features to be partially implemented for a while.

On the web-features repo, we don't yet have a way to deal with this.
A feature is either supported by a browser, or it's not.

That's because the Baseline status we compute is simple by design.
The more nuance we add to it, such as partial support, or support for mobile-only features, etc. makes Baseline more complicated.

We wish to retain Baseline's simple status, but over time add more of the nuance in the underlying data.
This way, the sites and tools that consume the data can keep on showing the nice and simple Baseline banner,

But also add interesting notes about special cases.

So, it's important to continue being a responsible web developers especially these days where vibe coding is a thing.

Baseline is a tool for making some quicker decisions,
it’s not a substitute for actual engineering.

And with that I'll leave you with a few things to keep in mind:

- BCD and web-features are two very important data sources for web developers, even if you don't realize it.
- They take funding to work on. If your company invests or relies on the web, consider helping out. Talk to me.
- Play with web-features and let us know if you build something useful.
- Join the WebDX CG, we're a welcoming bunch of folks who want to make the web a better place.

🙏 **Thank you!**

Find me at ➡️ https://patrickbrosset.com ⬅️

Thank you!