I DIDN'T KNOW CSS COULD DO THAT!

Matteo Fogli
DevFest Alps 2024

Hello there 👋

This is (not) the web developer you've been looking for

MODO

we code design

So what’s new with CSS?

A lot:

Architectural foundations

- 🚀 Trigonometric functions
- 🆓 Complex nth-* selection
- 🧑‍💻 Scope
- 🤘 Nesting

Typography

- 🚀 Initial-letter
- 🆓 Text-wrap balance/pretty

Color

- 🚀 Color level 4
- 🆓 Color-mix function
- 🧑‍💻 Relative color syntax

Responsive design

- 🚀 Container queries
- 🆓 Style queries
- 🧑‍💻 :has selector
- 🤘 Update media query
- 🆕 Scripting media query
- 👨‍💻 Transparency media query

Interaction

- 🚀 View transitions
- 🆓 Linear-easing function
- 🧑‍💻 Scrollend
- 🤘 Scroll-driven animations
- 🆕 Deferred timeline attachment
- 👨‍💻 Discrete property transitions
- 🤩 Starting-style rule
- 🆒 Overlay animation

Components

- 🚀 Popover
- 🆓 Hr in select
- 🧑‍💻 User-valid/invalid pseudo classes
- 🤘 Exclusive accordion

… but *Can I Use* it?

ENTER: BASELINE

Baseline

Baseline is available on MDN, Can I Use, and web.dev

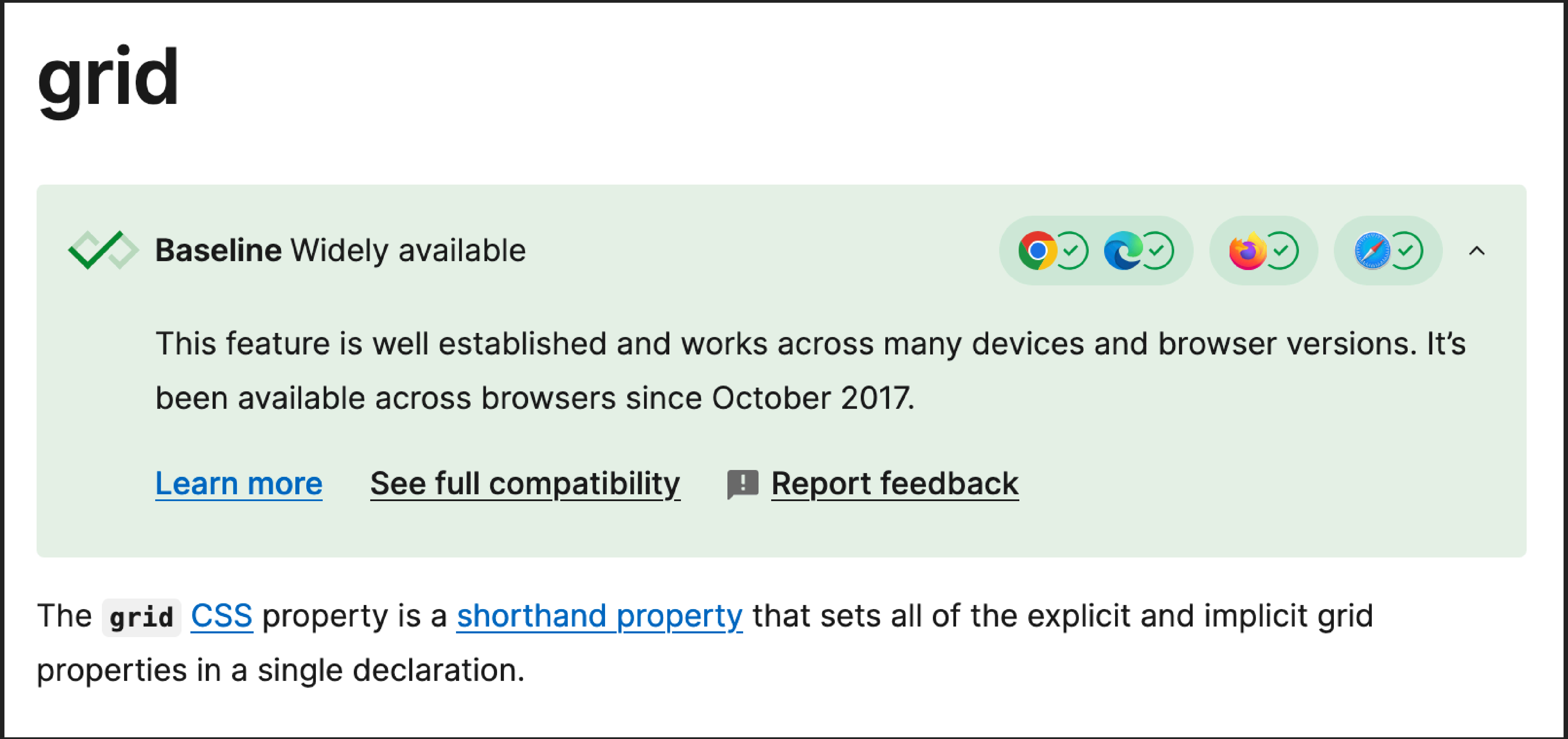
Speaker notes

Baseline is a cross-functional initiative to provide better clarity on browser feature availability ⏭️

This is a good proxy for deciding when to drop a JS implementation and use a *progressively enhanced* CSS solution ⏭️

- Baseline is an initiative of the **WebDX Community Working Group**
- https://developer.mozilla.org/en-US/blog/baseline-unified-view-stable-web-features/
- https://web.dev/baseline
- https://web-platform-dx.github.io/web-features/

ENTER: BASELINE



grid

Baseline Widely available

This feature is well established and works across many devices and browser versions. It’s been available across browsers since October 2017.

Learn more See full compatibility Report feedback

The `grid` CSS property is a shorthand property that sets all of the explicit and implicit grid properties in a single declaration.

Baseline is available on MDN, Can I Use, and web.dev

How do features become part of Baseline?

Baseline has two stages:

Newly available: The feature works across the latest devices and browser versions. The feature might not work in older devices or browsers.

Widely available: The feature is well established and works across many devices and browser versions. It’s been available across browsers for at least 2½ years (30 months).

Speaker notes

- 2 stages: newly available and widely available
- it might take more than 30 months for a feature to be widely available, we count from the release date of newly available
- what about features that don't make it yet? ⏭️
- these feature are publicly available, not behind a flag
- it means the standardization process is complete and the spec final
- browsers

How do features become part of Baseline?

Features that have not yet landed in Baseline are:

Limitedly Available: The feature works only behind a flag or on specific browsers and versions.

How do features become part of Baseline?

Baseline is calculated using the following core browser set:

- Apple Safari (macOS and iOS)
- Google Chrome (desktop and Android)
- Microsoft Edge (desktop)
- Mozilla Firefox (desktop and Android)

Interop (2024)

A cross-browser effort to reach a state where each technology works exactly the same in every browser.

- Accessibility
- CSS Nesting
- Custom Properties
- Declarative Shadow DOM
- font-size-adjust
- HTTPS URLs for WebSocket
- IndexedDB
- Layout
- Pointer and Mouse Events
- Popover
- Relative Color Syntax
- requestVideoFrameCallback
- Scrollbar Styling
- @starting-style and transition-behavior
- Text Directionality
- text-wrap: balance
- URL

Speaker notes

a cross browser initiative to push forward some features during the year in order to reach newly available

- improve the interoperability of the web
- uses suite of tests
- involves browser vendors on shared vision for each year's baseline

IS THIS BECAUSE YOU HATE JAVASCRIPT?

Absolutely not!

Speaker notes

all this focus on CSS… is this because you hate javascript?

I love the web

I love CSS

and I love JavaScript AND TypeScript

The web is an incredible platform but every tool has a use, and we are seeing an explosion of them... it would be foolish to pass the chance to use them!

So why the fixation with JavaScript?

BUT WHY WORK AGAINST THE BROWSER?

Here’s what we get for free

- Running off the main thread
- No re–rendering
- Progressively enhanced
- Better performance

Speaker notes

I'm a huge performance freak, I sweat about milliseconds and removing JS code is my first go-to stop for performance optimizations. Not only do we ship less code (and have to mantain less code!), we also avoid having to think about loading strategies and deferring UX enhnacements to when the scripting is eventually ready (unless of course we want huge INP metrics and very angry users staring at a frozen screen)

Also, I don't like people that "HATE". Well, there are some good reasons to hate, and it gets political the very minute you think about it, but **we are a community** (of designers, developers, builders, makers, visionaries…) We teach, we don't hate. We share, we make progress together. And this — i hope — is what we'll do today.

- declarative language easier than JS
- no need to worry about implementation details
- less code shipped
- Accessibility

The Principle of Least Power

There's an inverse relationship between the power of the language and how easy it is to learn.

- If you can do it with HTML, 👉 use HTML
- If you can't do it with HTML, 👉 use CSS
- If you can't do it with HTML or CSS, 👉 use Javascript

Speaker notes

There’s also en economic / philosophical motive:

- these principles are from the **HTML First manifesto** https://html-first.com/
- I don't share *all* of the opinions but these principles are sane
- HTML First promotes a style of writing web software that favours using the native capabilities and languages of the browser and reducing layers of abstraction (languages and toolchains) on top of them
- HTML is the *least powerful language* but has the **lowest learning curve**, and javascript is the most powerful but has the highest learning curve
- it's a manifesto for the efficiency of the web (and our sanity of mind as dev), although I don't share every opinion in there (YMMV always applies)

COMPLEX CONDITIONAL DOM STATES

the power of the `:has` selector

Speaker notes

`:has` is one of the most powerful and revolutionizing selectors that has landed in CSS

after years, we're finally able to select a parent element based on the conditions of the children (and much more)

THE PARENT SELECTOR

`:has` gives us the power to select an element based on the *state* or presence of children elements.

It unlocks a whole range of possibilities that previously required conditionally applying classes to the DOM via JavaScript

Speaker notes

we don't have to keep state nor patch the DOM to adapt layouts and designs to specific states or conditions

The basics

```
element:has(.child)
```

```
element:has(> .direct-child)
```

```
element:has(:state)
```

```
element:has(:state) .child
```

Speaker notes

`:state` can be anything like `:focus`, `:hover`, `:checked`, and even `[disabled]` or any other `[attribute]`

argument can be anything

The not so basics

```
element:has(:not(:state))
```

```
element:has(.logical, .or)
```

```
element:has(.logical):has(.and)
```

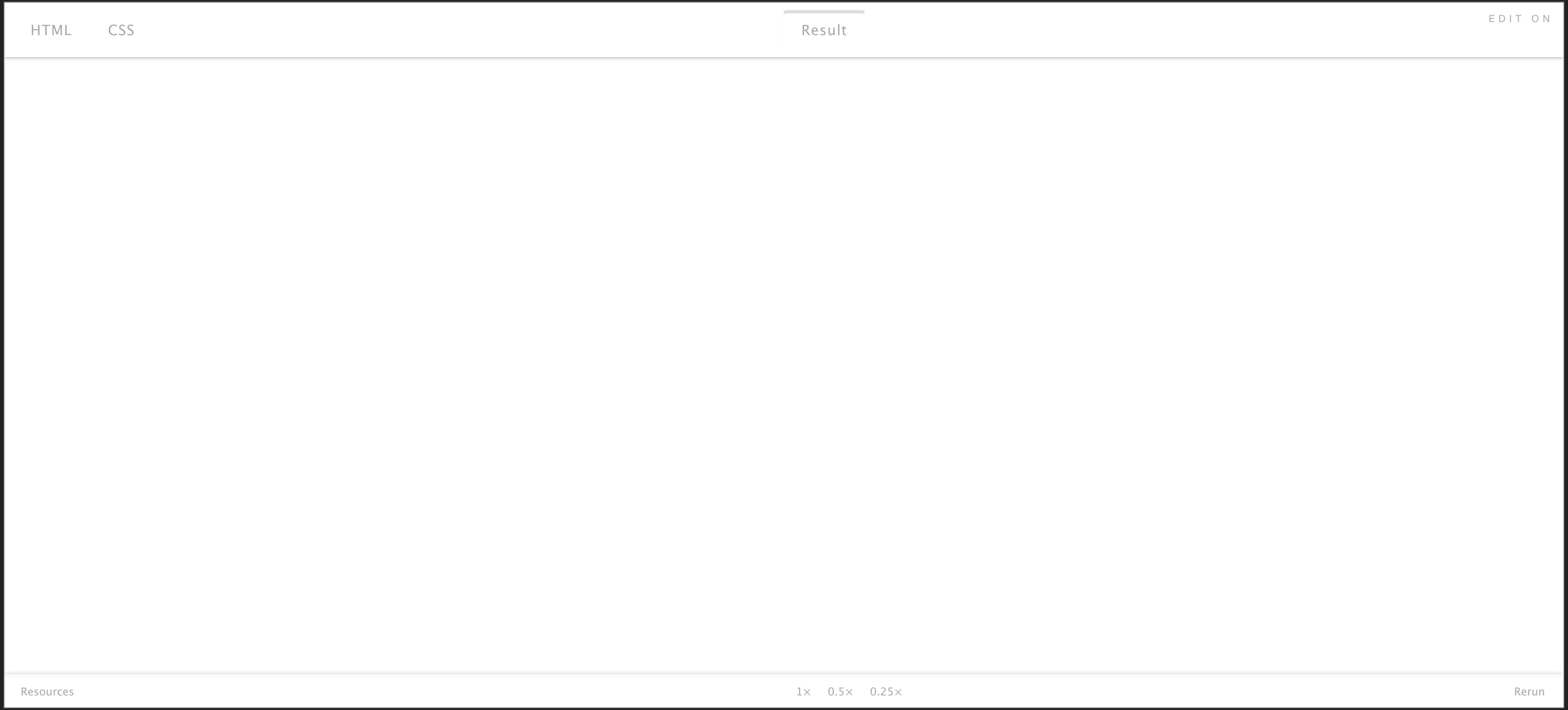
Speaker notes

select hierarchically vs select horizontally

select with a logical OR

select with a logical AND

Change parent element based on child:hover

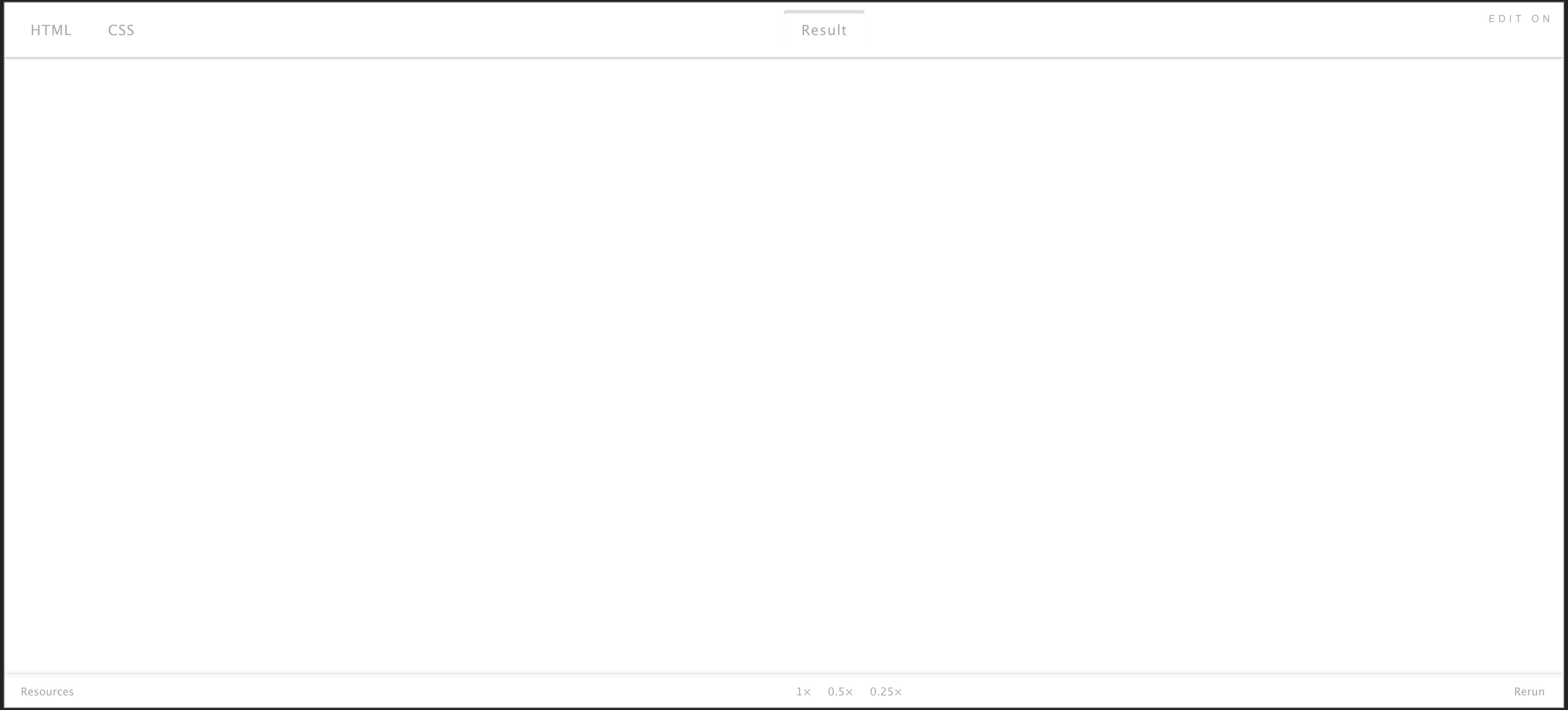


Speaker notes

great pen by Michelle Barker

https://codepen.io/michellebarker/pen/vYzqaNO

Change parent element based on content

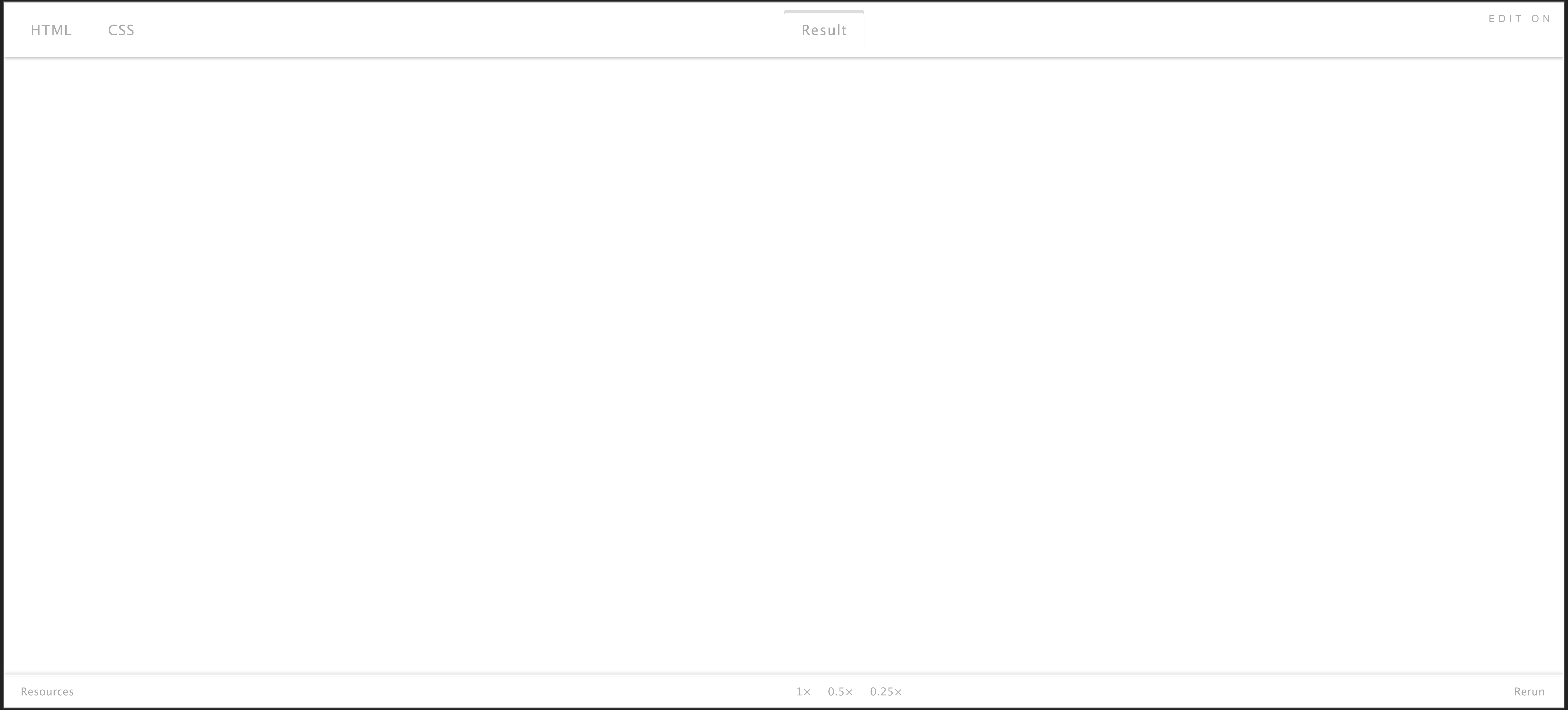


Speaker notes

and another great example by Jen Simmons. here the cards span two columns when they include an image

https://codepen.io/jensimmons/pen/bGoMydw

Change previous sibling

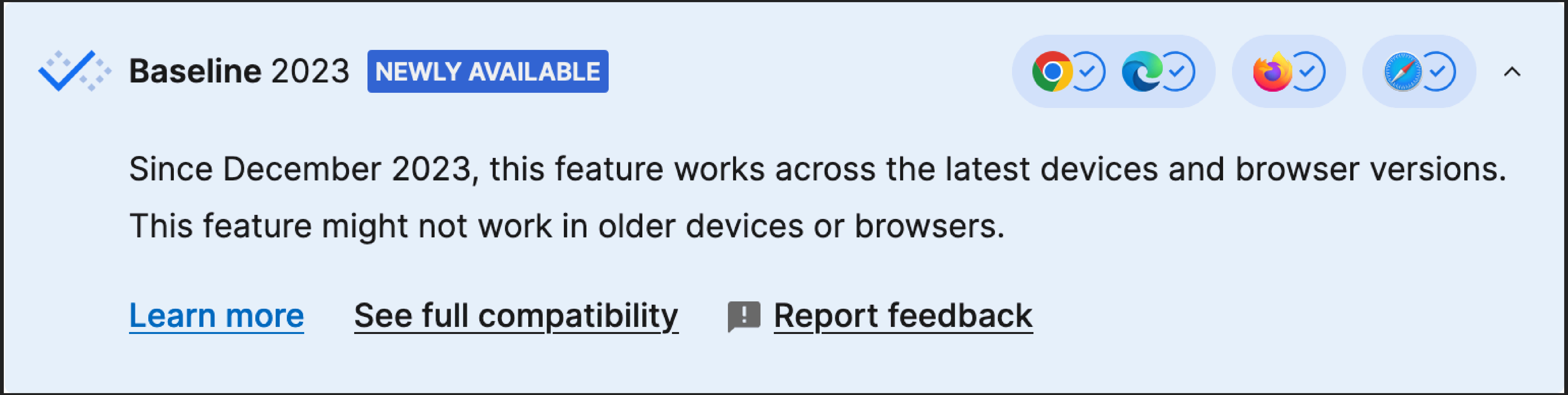


Speaker notes

https://codepen.io/web-dot-dev/pen/XWOqoPL

inspired by a majestic talk by Sanne t' Hooft https://sinds1971.nl/cssday/ see https://codepen.io/shooft/live/bGmMZEP

Baseline Newly Available



Baseline 2023 NEWLY AVAILABLE

Since December 2023, this feature works across the latest devices and browser versions. This feature might not work in older devices or browsers.

Learn more See full compatibility Report feedback

Feature detection

```css
figure {
  /* widely supported CSS styles here */
}

@supports selector(figure:has(caption)) {
  figure:has(caption) {
    /* newly available CSS styles here */
  }
}
```

`:has` caveats and usage hints

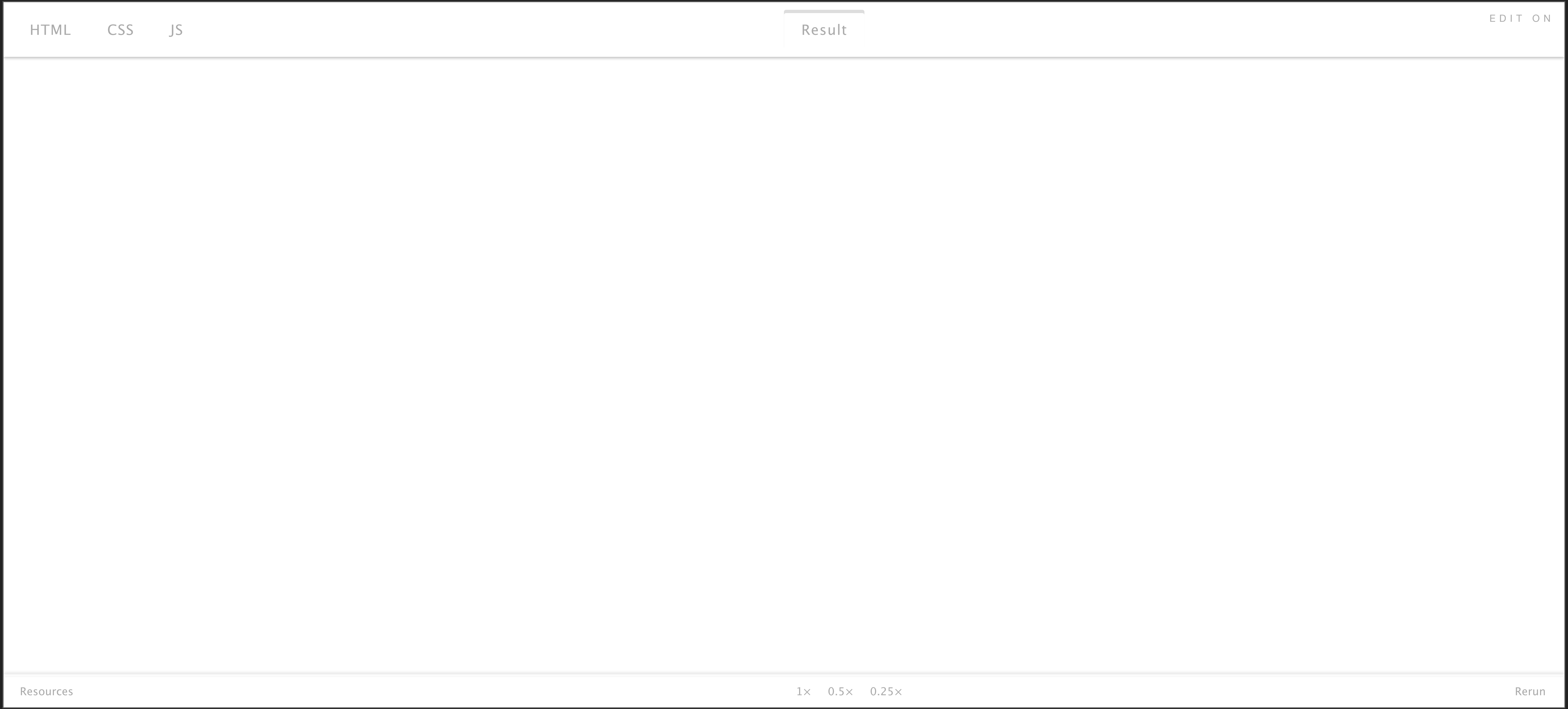
- not forgiving (use `:where()`)
- takes highest specificity of argument selectors
- keep as specific as possible
- *might* have performance issues with very big DOM trees (real world test don’t surface this issue)

THE “RANGE” SELECTOR

Speaker notes

- this is a very powerful but rarely known baseline widely available feature
- once again, it allows us to drop some rendering cycles and JS DOM patches, deferring to CSS layout adaptations and optimizations
- CSS Selectors Level 4 for `of` extension

`:nth-child(n + B)` `:nth-child(-n + B)`

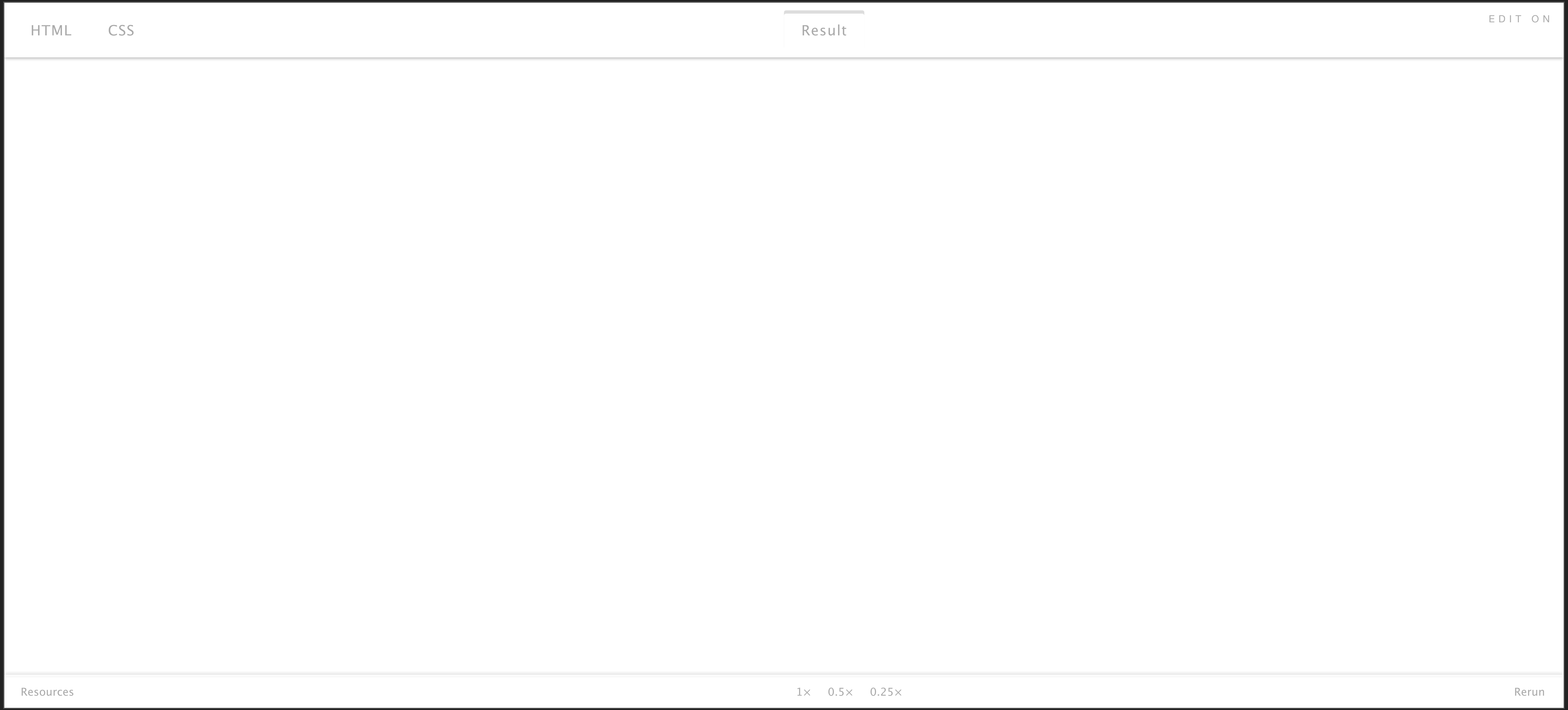


Speaker notes

- we can select ranges (first *n* elements, last *n* elements, *n* to *m* elements)
- we can count elements (combining `:nth-child` and `:nth-last-child` with `:has`)

https://codepen.io/pecus/pen/bGXwVNw

`:nth-child(An + B of .selector)`



Speaker notes

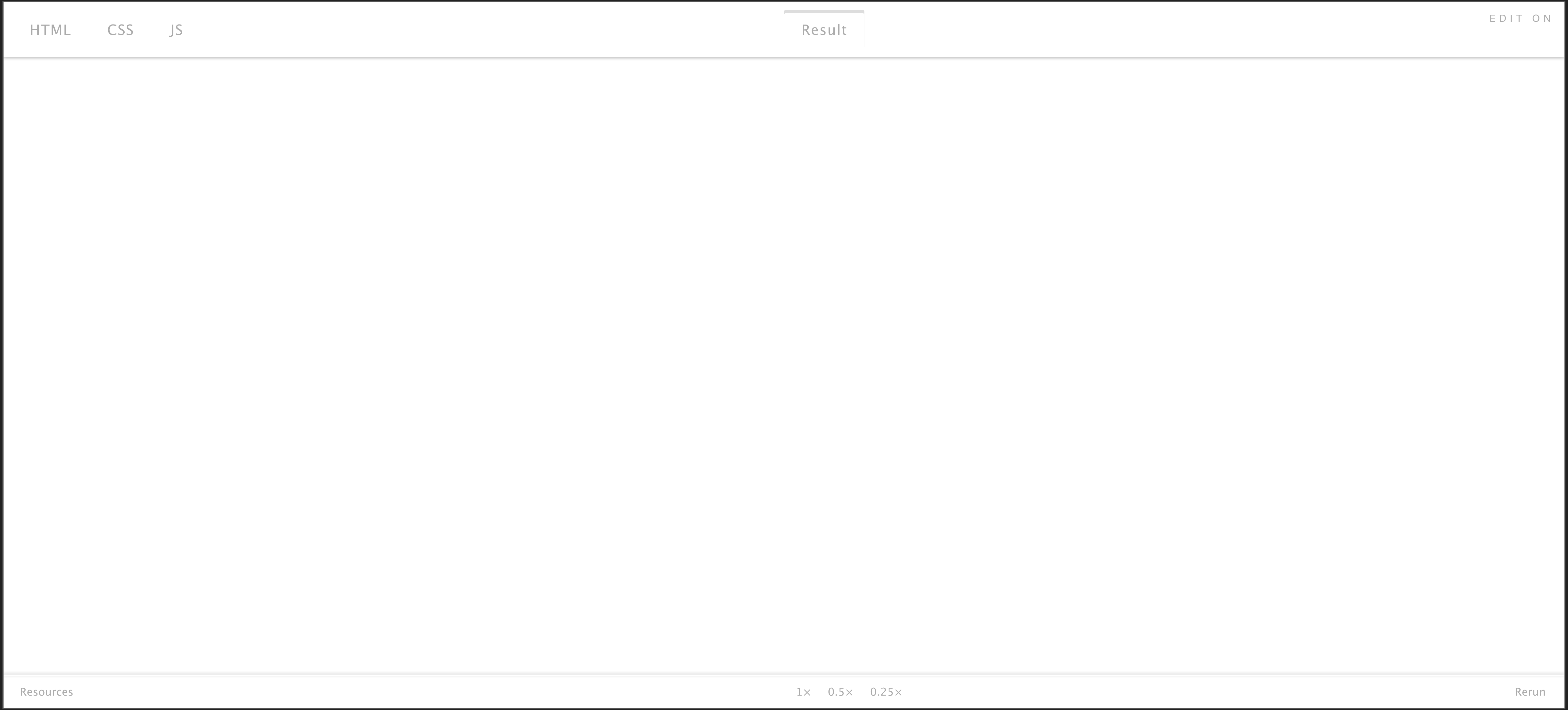
CSS Level 4 update. Allows to *pre-filter* elements based on a selector before counting them.

The element that matches `:nth-child(2 of .highlight)` has a pink outline.

The element that matches `.highlight:nth-child(2)` has a green outline.

https://codepen.io/web-dot-dev/pen/oNMRaQq

`:nth-child(An + B of .selector)`



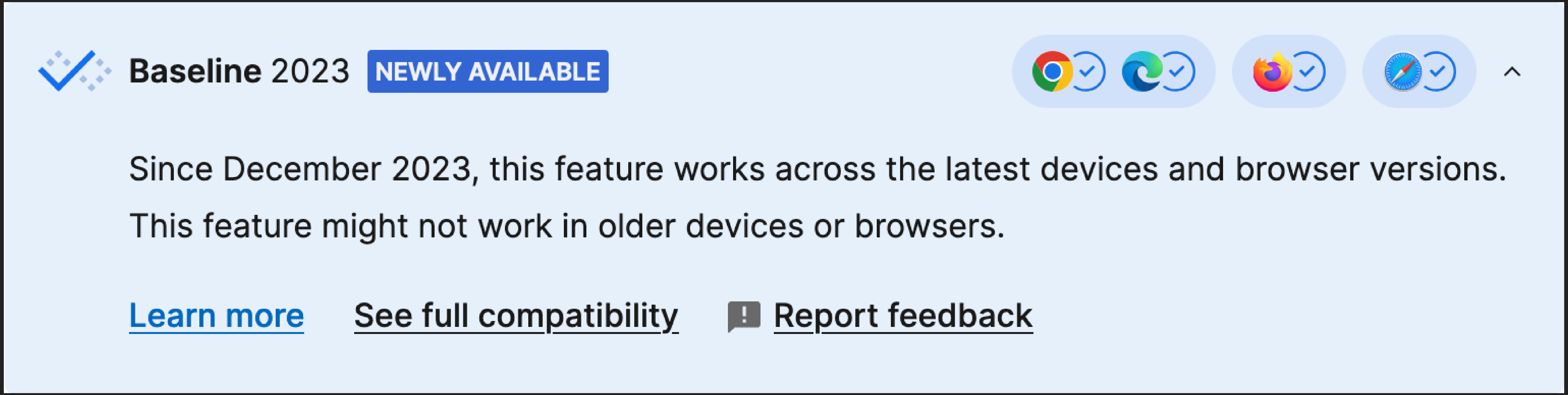
Speaker notes

the most classic case for adoption is alternating table rows (AKA the Zebra effect).

when rows are filtered, `:nth-child()` would not potentially select alternating rows, because hidden rows are still child of the parent element (and therefore counted). the `of` selector argument allows to pre-filter the set and keep the alternating rows effect always visible

https://codepen.io/pecus/pen/poMPvqL

`of <selector>` Baseline Newly Available



Baseline 2023 NEWLY AVAILABLE

Since December 2023, this feature works across the latest devices and browser versions. This feature might not work in older devices or browsers.

Learn more See full compatibility Report feedback

Speaker notes

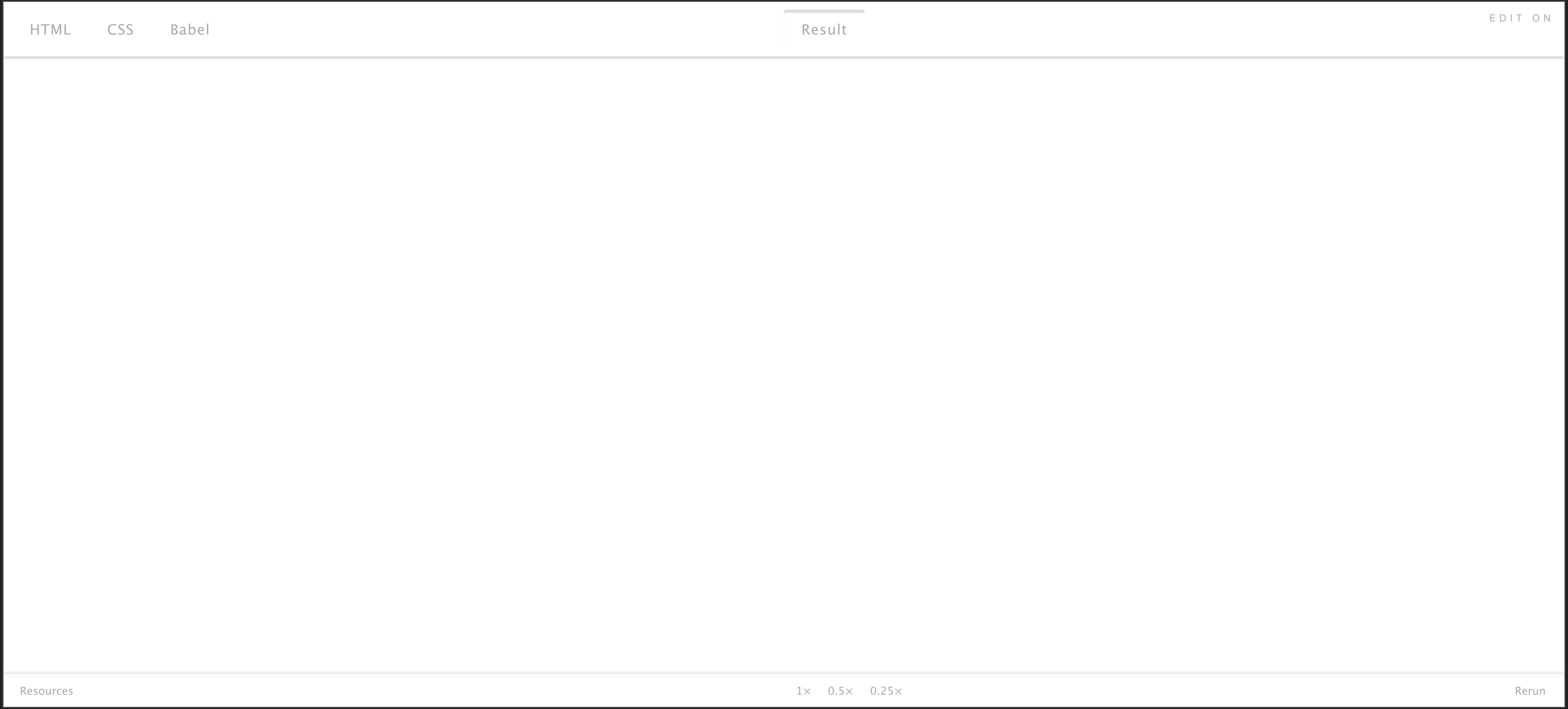
`:nth-child()` selector is baseline widely available since 2015 (Hello IE)

Typographically Accurate Text Wrapping

Speaker notes

- we have eyes trained on print
- typography is beautiful
- yet another thing we used JS for

`text-wrap: balance`



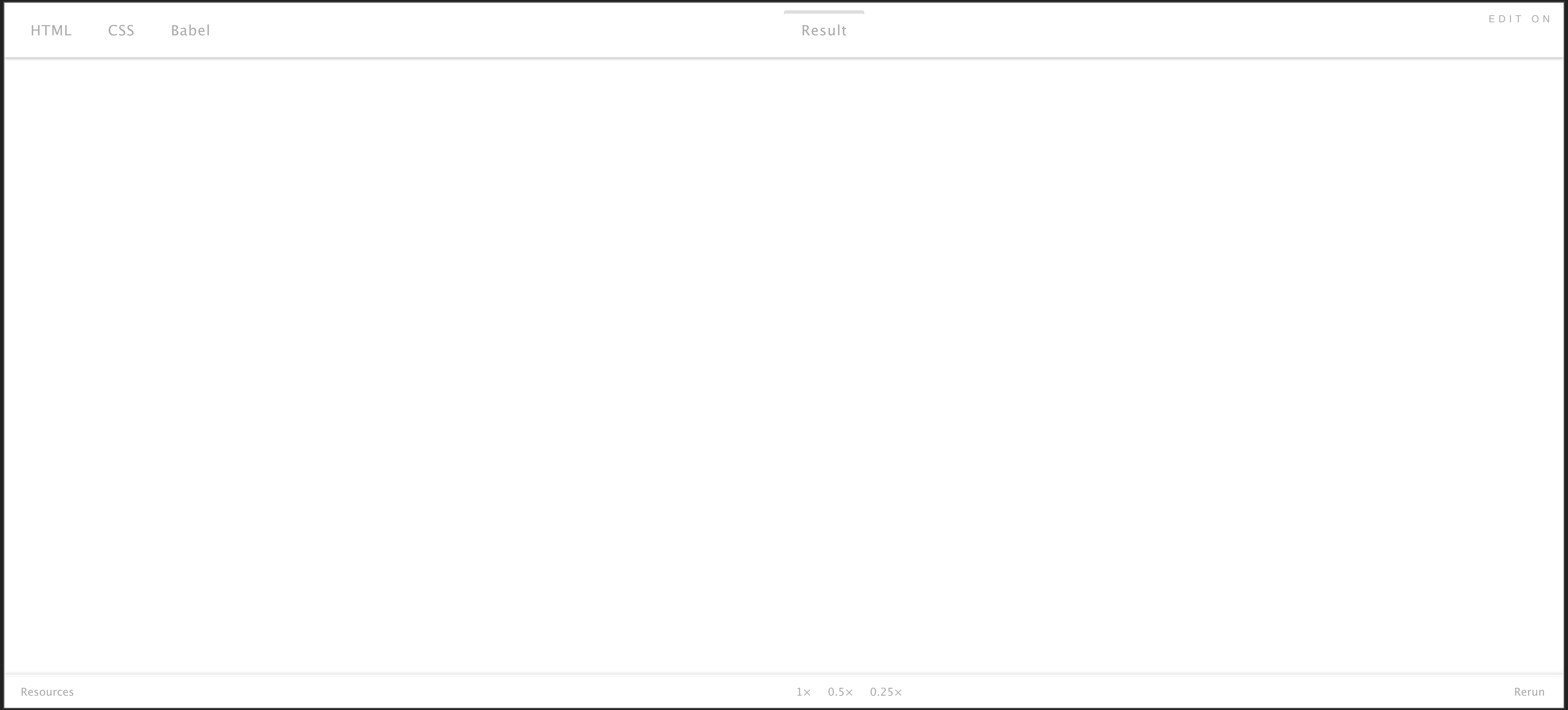
Replaces: https://react-wrap-balancer.vercel.app/

Speaker notes

- shorthand for CSS properties: `text-wrap-mode` and `text-wrap-style`
- `balance`: best balances the number of characters on each line, enhancing layout quality and legibility.
- computationally expensive
- only supported for a limited number of lines (6 Chromium, 10 Firefox)
- make sure to apply only to headline
- will not change the element box size
- only works if text wraps, so make sure to specify a max width (use logical sizes combined with char units for best effect: `max-inline-size: 80ch`)
- works with web fonts, no need to wait for the font observer to ensure that the font has loaded to balance the headline
- While we're at it...

https://codepen.io/pecus/pen/jOgrxQN

`text-wrap: pretty`

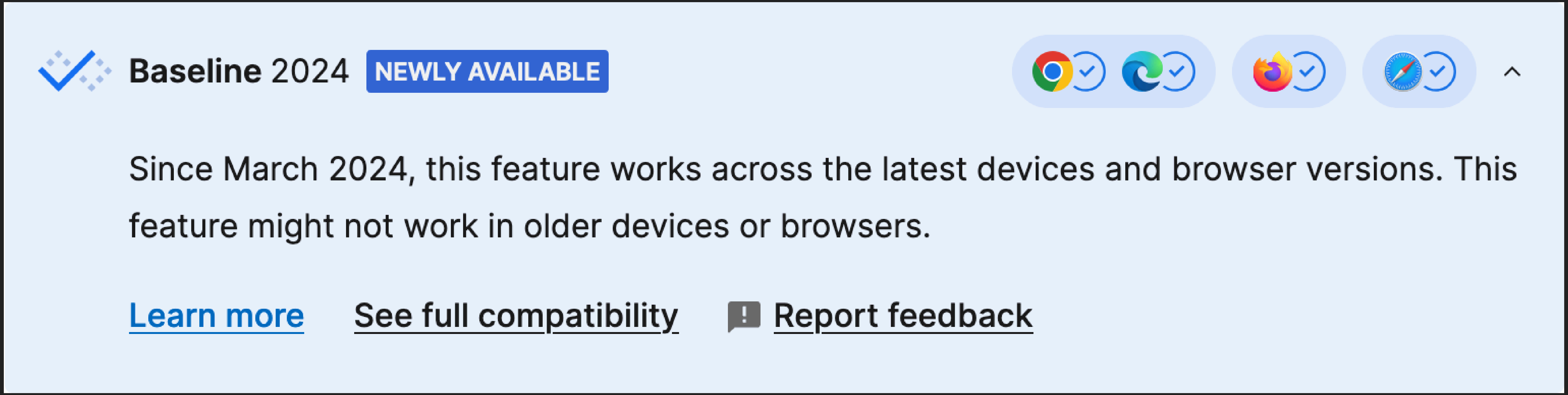


Speaker notes

`pretty`: same behavior as wrap, except that the user agent will use a slower algorithm that favors better layout over speed. This is intended for body copy where good typography is favored over performance

https://codepen.io/pecus/pen/gOVMyQd

Baseline Newly Available



Baseline 2024 NEWLY AVAILABLE

Since March 2024, this feature works across the latest devices and browser versions. This feature might not work in older devices or browsers.

Learn more See full compatibility Report feedback

Some limitations apply.

Speaker notes

- `text-wrap` is actually a shorthand for `text-wrap-mode: wrap` and `text-wrap-style: pretty|balance` but Chrome only understands the shorthand syntax
- this is a perfect feature to illustrate **progressive enhancement**. you don't even need to use `@supports`

Feature detection

(gracefully falls back without `@supports`)

```css
:where(h1,h2,h3,h4,h5,h6) {
  /* widely supported CSS styles here */
}

@supports (text-wrap: pretty) {
  :where(h1,h2,h3,h4,h5,h6) {
    /* newly available CSS styles here */
  }
}
```

SCROLL DRIVEN ANIMATIONS

Speaker notes

- this is actually a trip down the rabbit hole
- **Scroll driven animations!** super fluid *scrub* animations based on scroll and element entering into/out of viewport
- we get a new timeline, 2 new functions (`scroll()` and `view()`) and a couple of new CSS properties (`scroll-timeline` and `animation-range`)
- we also get butter smooth animations running off the main thread with no scroll hijacking, and no need for debouncing or throttling events taking away precious resources from the main thread
- we don't get (yet) scroll driven animations

Speaker notes

** must read **

https://scroll-driven-animations.style

demos and tools + a full video course to explore rather complex concepts related to scroll driven animations, such as understanding how to limit the range of the animation, how to scope the same animation to multiple elements, and a how to replicate popular effects that used to require expensive JS with a few lines of CSS

Scroll driven animations extend CSS animations

```css
@keyframes spin {
  to {
    rotate: y 1800deg;
  }
}
.animate-me {
  animation:
    spin 1s
    ease-in-out
    infinite
}
```

Speaker notes

- builds on CSS animations we're all familiar with

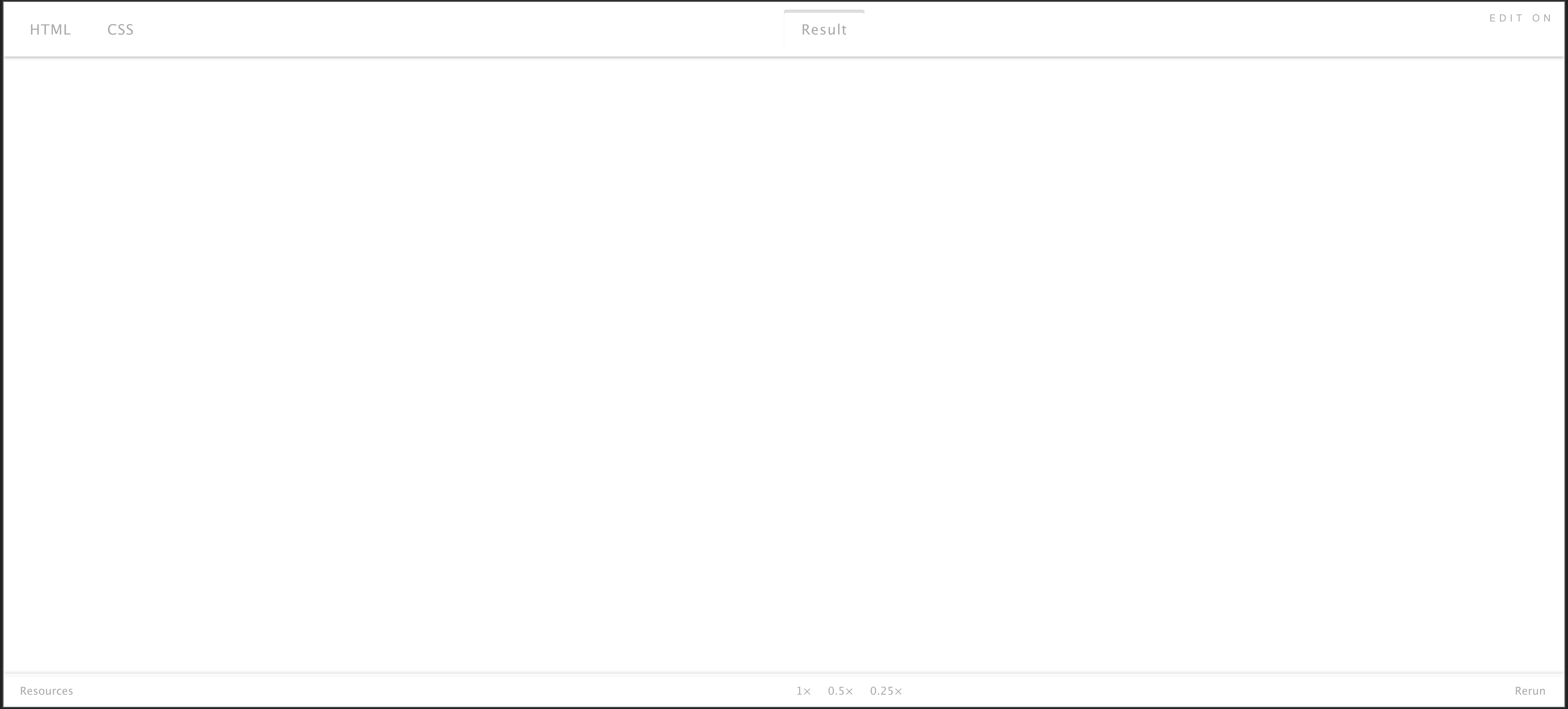
Scroll driven animations extend CSS animations

```css
@keyframes spin {
  to {
    rotate: y 1800deg;
  }
}
.animate-me-scroll {
  animation: spin linear;
  animation-timeline: scroll(block root);
  /* animation-range: 50% 100%; */
}
```

Speaker notes

again, we get a new timeline, 2 new functions (`scroll()` and `view()`) and a couple of new CSS properties (`scroll-timeline` and `animation-range`both shorthand for `scroll-timeline-axis` `scroll-timeline-name` and `animation-range-start` `animation-range-end` respectively)

`scroll()`



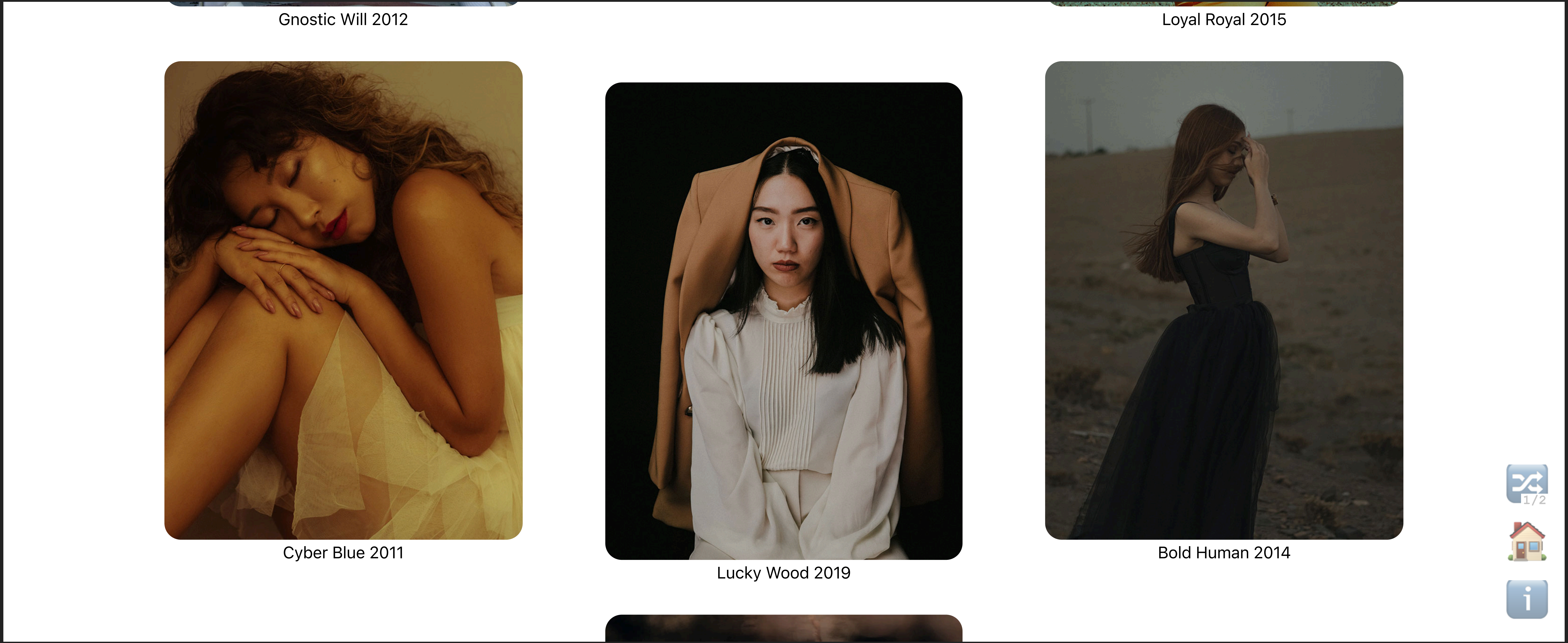
(partially) Replaces: GSAP, Lenis

Speaker notes

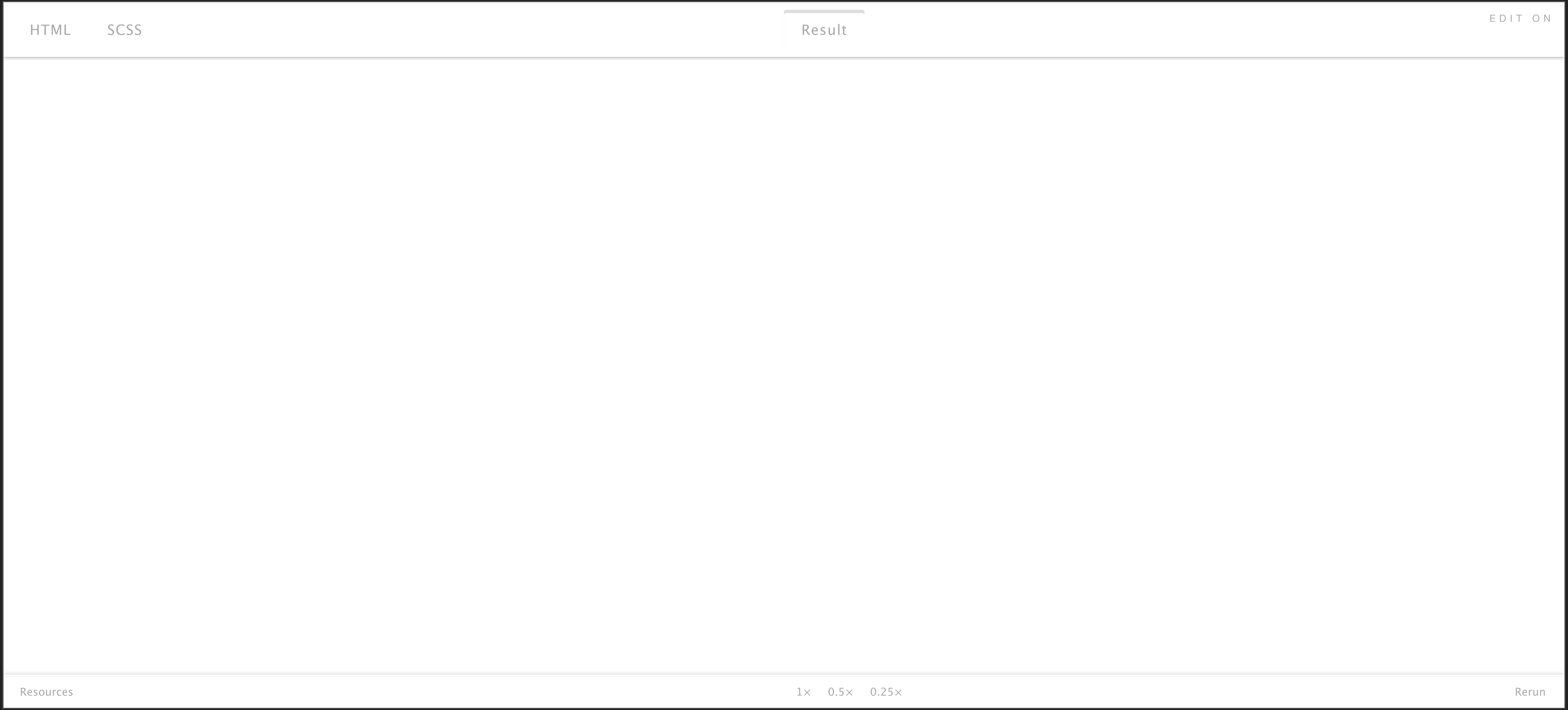
we're tracking the *root* scroller without customizations, so our timeline scrubs from the top of the document to the bottom.

Another great pen by Michelle Barker https://codepen.io/michellebarker/pen/JjxBzvO

scroll()


Gnostic Will 2012
Loyal Royal 2015
Cyber Blue 2011
Lucky Wood 2019
Bold Human 2014
1/2

view()

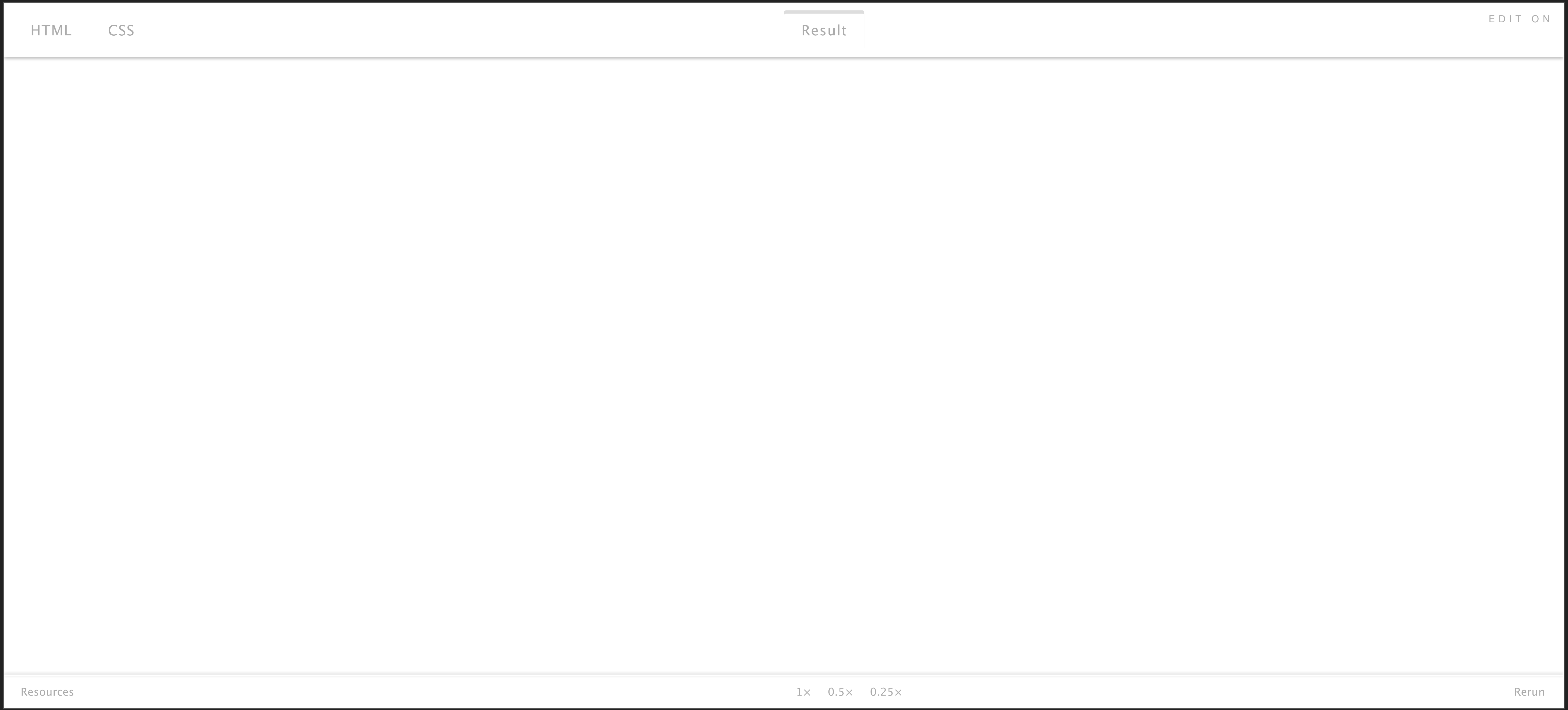


Speaker notes

a basic demo, progressively enhanced, that lets you appreciate how smooth these animations are and how you can go bonkers with ideas and effects

https://codepen.io/pecus/pen/RwXpBed

Measure direction and velocity...



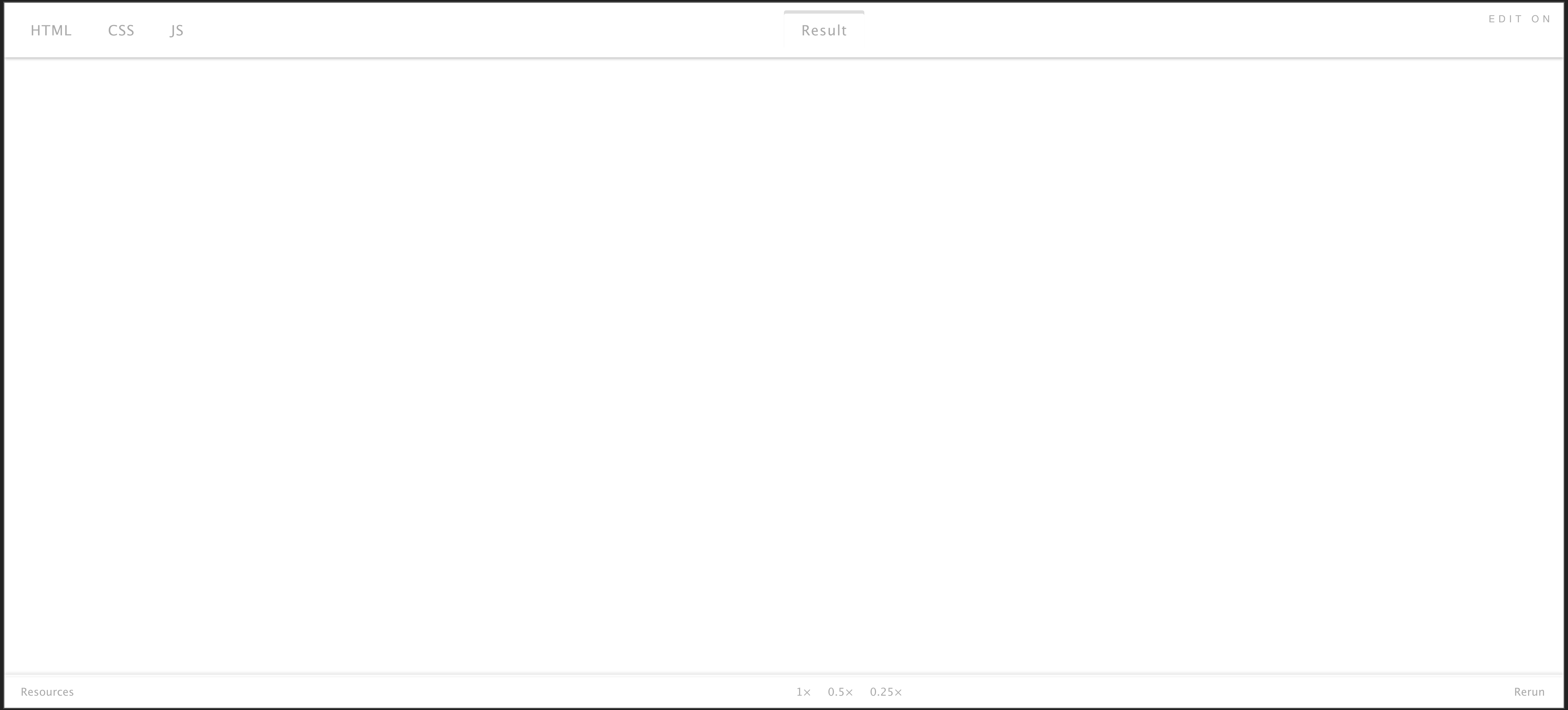
Speaker notes

using `@property` and some basic CSS math (`abs()` and `sign()`), we can track scroll direction and scroll speed/velocity

https://front-end.social/@bramus/113220884843667438

https://codepen.io/pecus/pen/dyxWOxR

… for unthinkable CSS-only effects



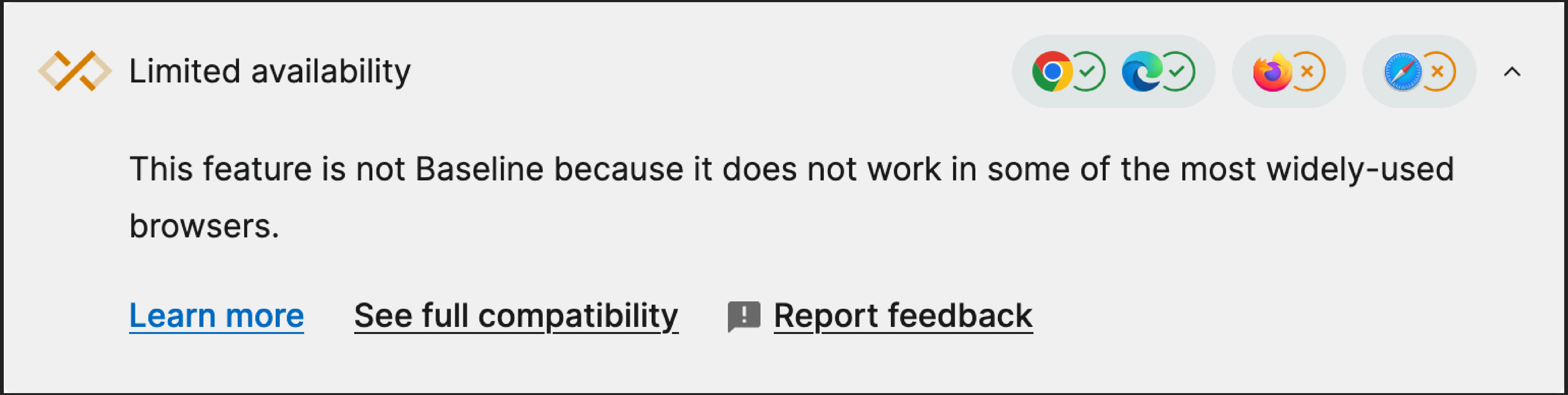
Speaker notes

an incredible hack by fabrizio calderan improving a demo by bramus van damme using `transition-delay` and custom properties

https://front-end.social/@bramus/113220884843667438

https://codepen.io/fcalderan/pen/LYKwyyd

Baseline? not yet



Limited availability

This feature is not Baseline because it does not work in some of the most widely-used browsers.

Learn more   See full compatibility   Report feedback

Feature detection

```
selector {
  /* widely supported CSS styles here */
}

@supports (animation-timeline: scroll()) {
  @media (prefers-reduced-motion: no-preference) {
    /* ensure animations are enabled only for users that did not signal a preference to avoid rapid mo
    selector {
      /* newly available CSS styles here */
    }
  }
}
```

IT'S ALL ABOUT CSS

Speaker notes

so we've gone through some of the newest features of CSS covering animations, Typographically correct wrapping, counter selectors and parent selectors for adaptive *intrinsic* layouts that react conditionally to page states or content count.

but THERE. IS. SO. MUCH. MORE.

oklch()
View Transitions
@swash
abs()
1cap
overlay
inset
color()
anchor()
:focus-visible
light-dark()
image-set()
CSS Nesting

text-box-trim
clamp()
Subgrid
Trigonometric functions
scrollbar-color
scale
Container Queries
@scope
Scroll Snap
:user-valid
caret-color
env()

@property
Scroll Animations
in oklab
:has()
scroll-timeline
ascent-override
accent-color
@starting-style
::backdrop
:fullscreen
aspect-ratio
place-content

top-layer
@layer
Popover API
::marker
view-timeline
initial-letter
color-mix()
override-colors
::cue
:dir()
cross-fade()
gap

Speaker notes

These are only **some** of the newest additions to CSS. Look at this list!

Color Functions would require a whole talk themselves. Container Queries. Trigonometric functions!

It's really a fantastic moment to be a front-end developer and embracing the platform. With everything CSS can offer, *maybe* shifting the weight a little bit from JavaScript.

Do stuff

Teach stuff

Involve everyone

Speaker notes

- they are great oportunities for modernizing, improving and hyping your code and the design of your sites. so **DO STUFF*
- encourage to use CSS, to discover, to build and experiment
- watch videos, read articles, attend conferences, talk to peers, organize meetups, ask your company to host workshops **TEACH STUFF**
- champion CSS and build excitement
- progressive enhancement: educate clients and bosses and negotiate with designers **INVOLVE EVERYONE**
- ask designers to move to CSS, to experiment with you. Share demos, build pens, mix ideas

~~I DIDN'T KNOW CSS~~
~~COULD DO THAT!~~

Speaker notes

This talk's title is "I didn't know CSS could do that!". But maybe, now that you've seen a few of the things the latest CSS can, you can say "I didn’t know *I* could do that in CSS!"

I DIDN'T KNOW I COULD DO THAT WITH CSS!

THANK YOU

Demos: https://codepen.io/collection/ZMgzbg

@pecus@mastodon.social