

Why JavaScript Is Costing You Visibility (and what to use instead)



“Rendering JavaScript” isn’t a thing



A URL returns some HTML



But the HTML isn't the page



The browser constructs the page



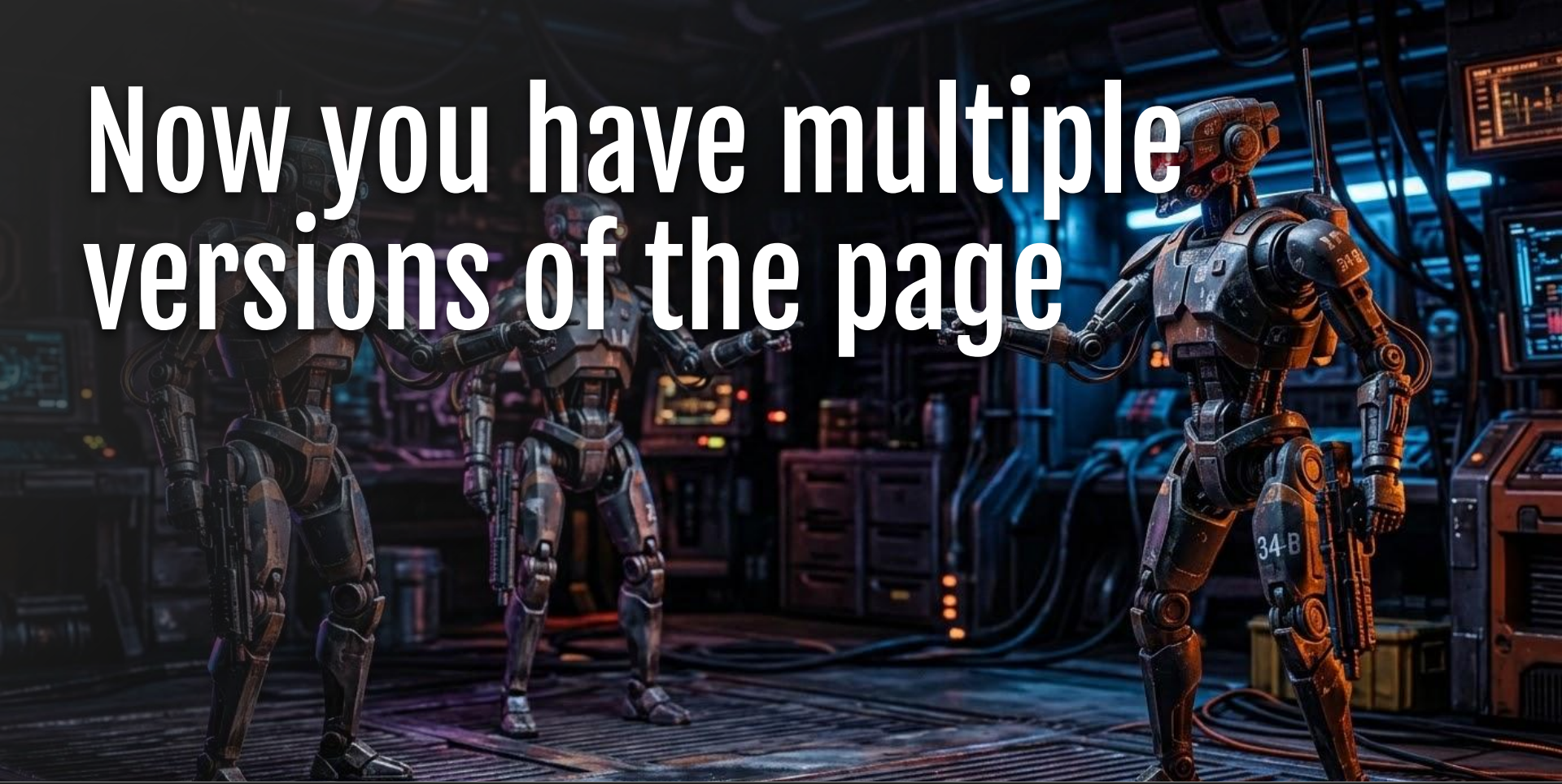
JavaScript can rewrite anything/everything



Google executes some of it



Now you have multiple versions of the page



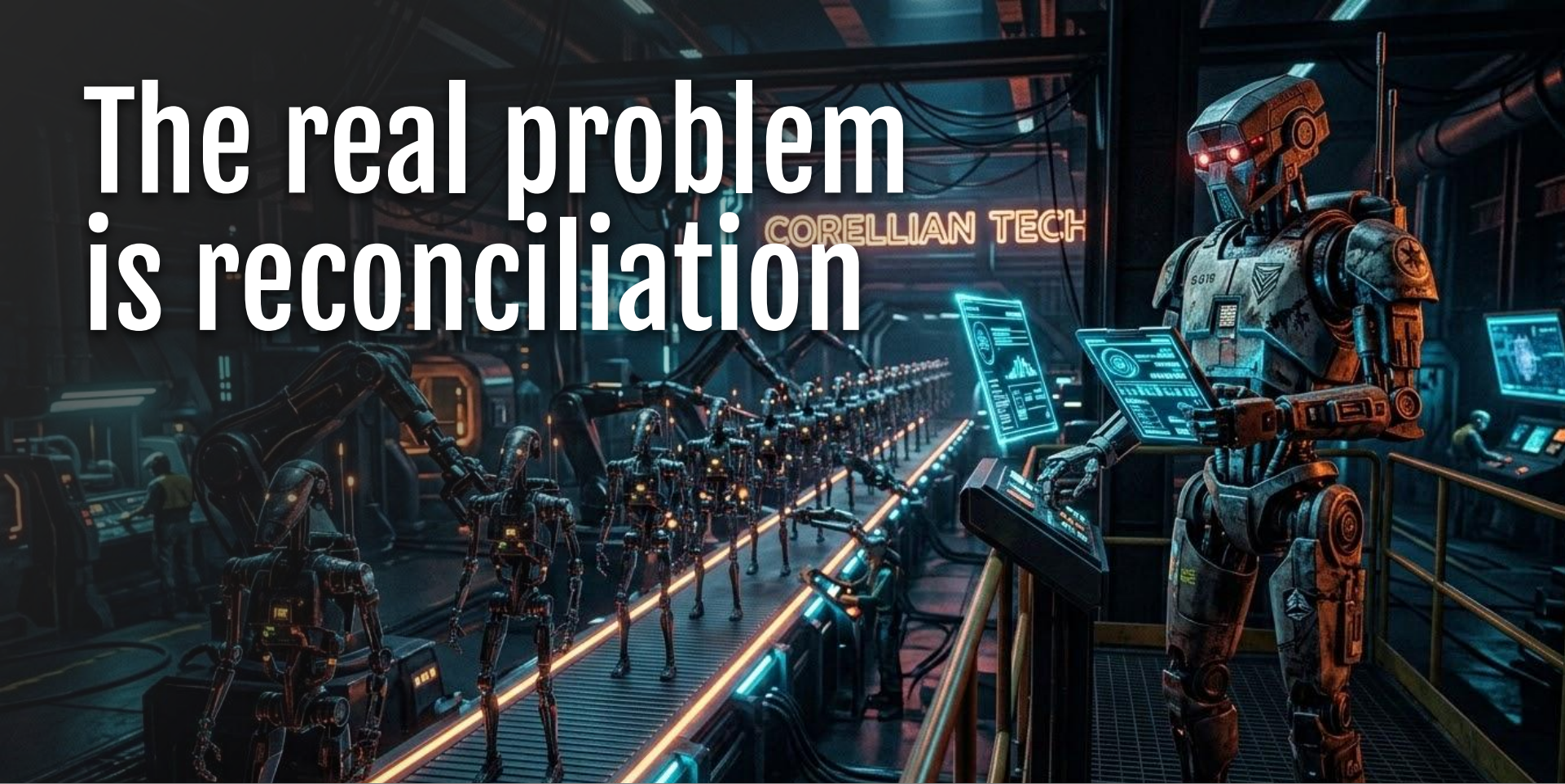
Those versions can disagree



View source, vs inspect



The real problem is reconciliation



None of this is new



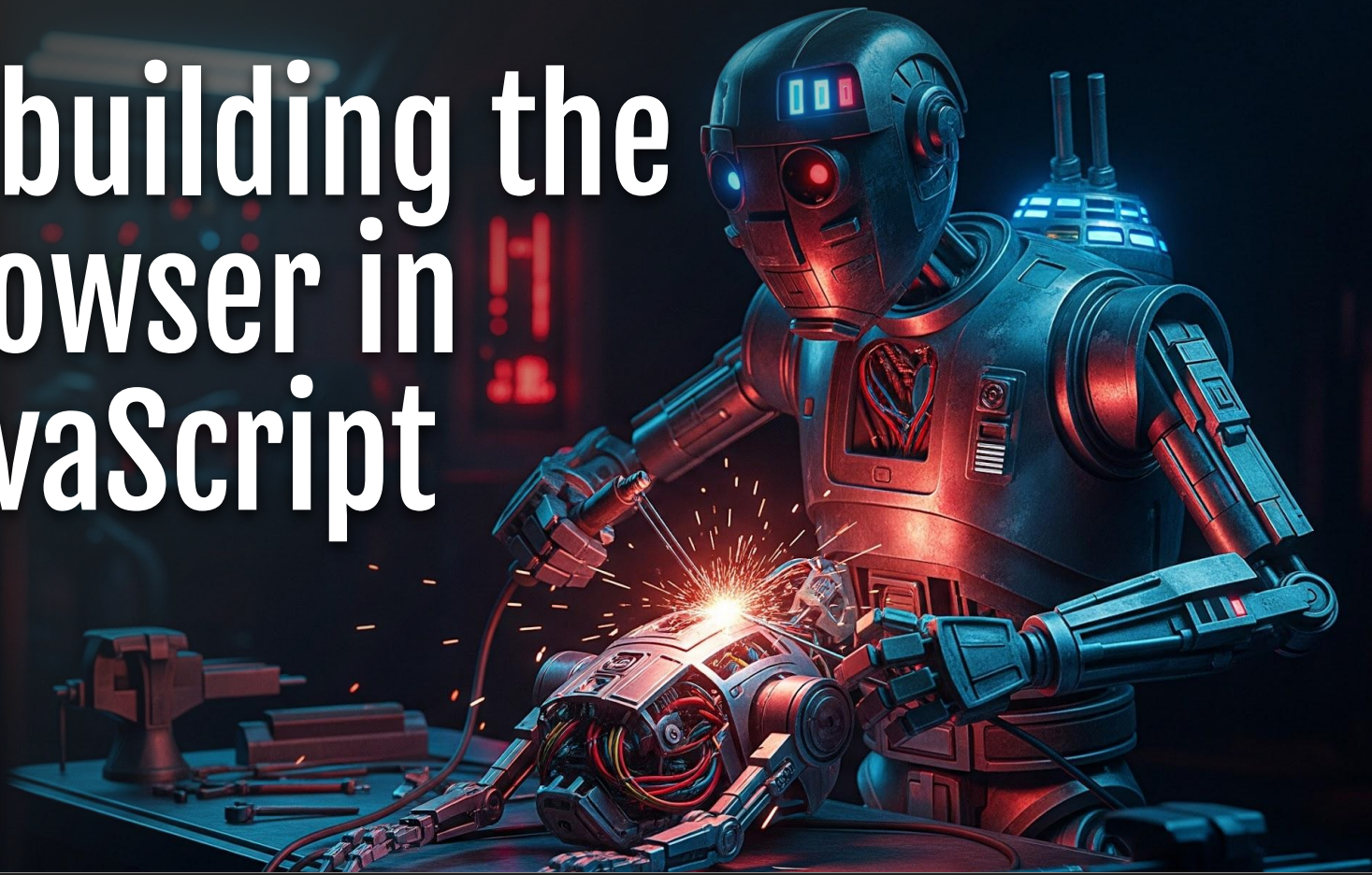
JavaScript changed jobs



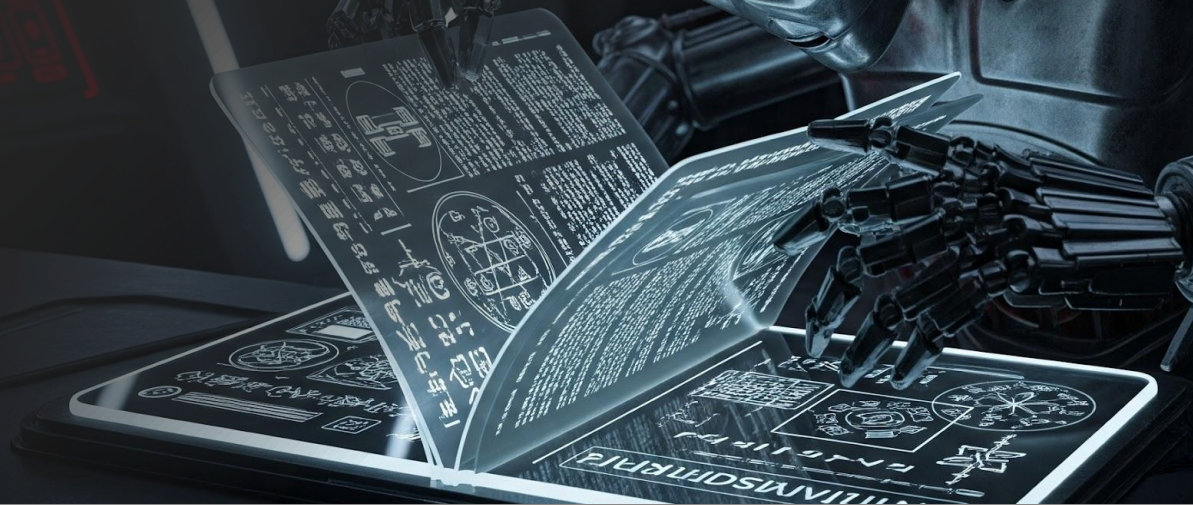
Software developers started building apps



Rebuilding the browser in JavaScript



Reinventing the wheel



The infinite framework arms race

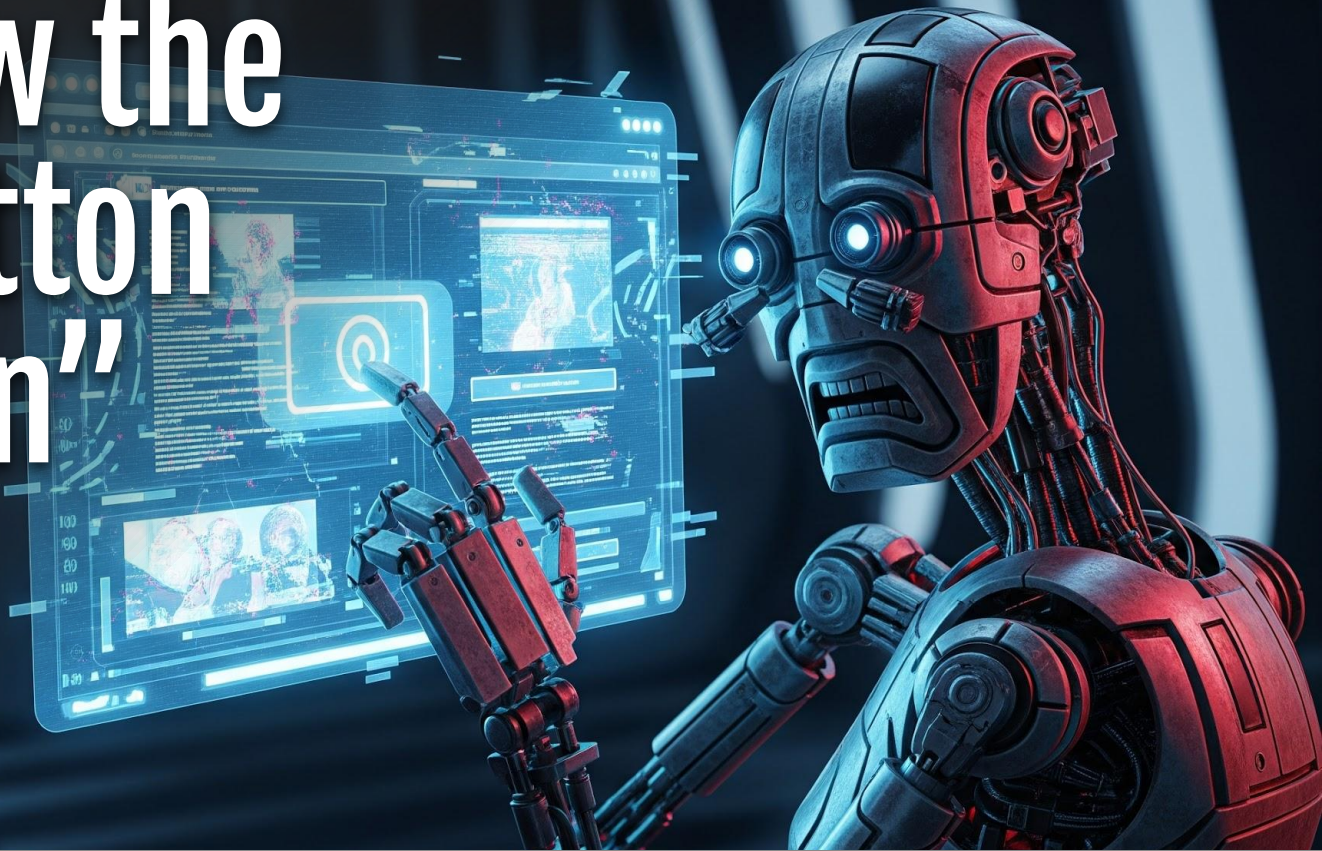


Jono Alderson

independent technical SEO consultant



“Oh, now the back button is broken”



We created monsters



Meanwhile, the browser caught up



It's (unfortunately) our
job to fix this



```

<section class="accordion" id="faq">
  <h3 class="acc-item">
    <button class="acc-btn" aria-expanded="false" aria-controls="a1">What is X?</button>
    </h3>
    <div id="a1" class="acc-panel" role="region" aria-hidden="true">Answer A...</div>

  <h3 class="acc-item">
    <button class="acc-btn" aria-expanded="false" aria-controls="a2">How do I Y?</button>
    </h3>
    <div id="a2" class="acc-panel" role="region" aria-hidden="true">Answer B...</div>
</section>

<style>
  .acc-panel { max-height: 0; overflow: hidden; transition: max-height .25s ease; }
  .acc-btn[aria-expanded="true"] + .acc-panel { max-height: 20rem; }
  .acc-btn { width: 100%; text-align: left; }
</style>

<script>
(function () {
  const acc = document.querySelector("#faq");
  const buttons = acc.querySelectorAll('.acc-btn');

  function closeAll() {
    buttons.forEach(btn => {
      btn.setAttribute('aria-expanded', 'false');
      const panel = document.getElementById(btn.getAttribute('aria-controls'));
      panel.setAttribute('aria-hidden', 'true');
    });
  }

  function toggle(btn) {
    const expanded = btn.getAttribute('aria-expanded') === 'true';
    closeAll(); // one-at-a-time behavior, remove if multi-open
    btn.setAttribute('aria-expanded', String(!expanded));
    const panel = document.getElementById(btn.getAttribute('aria-controls'));
    panel.setAttribute('aria-hidden', String(expanded));
  }

  buttons.forEach(btn => {
    btn.addEventListener('click', () => toggle(btn));
    btn.addEventListener('keydown', e => {
      if (e.key === ' ' || e.key === 'Enter') { e.preventDefault(); toggle(btn); }
      if (e.key === 'ArrowDown') { e.preventDefault(); btn.closest('.acc-item').nextElementSibling?.nextElementSibling?.previousElementSibling?.querySelector('.acc-btn')?.focus(); }
      if (e.key === 'ArrowUp') { e.preventDefault(); btn.closest('.acc-item').previousElementSibling?.previousElementSibling?.querySelector('.acc-btn')?.focus(); }
    });
  });
})();
</script>

```



```
<details class="faq">  
  <summary>What is X?</summary>  
  <p>Answer A...</p>  
</details>
```

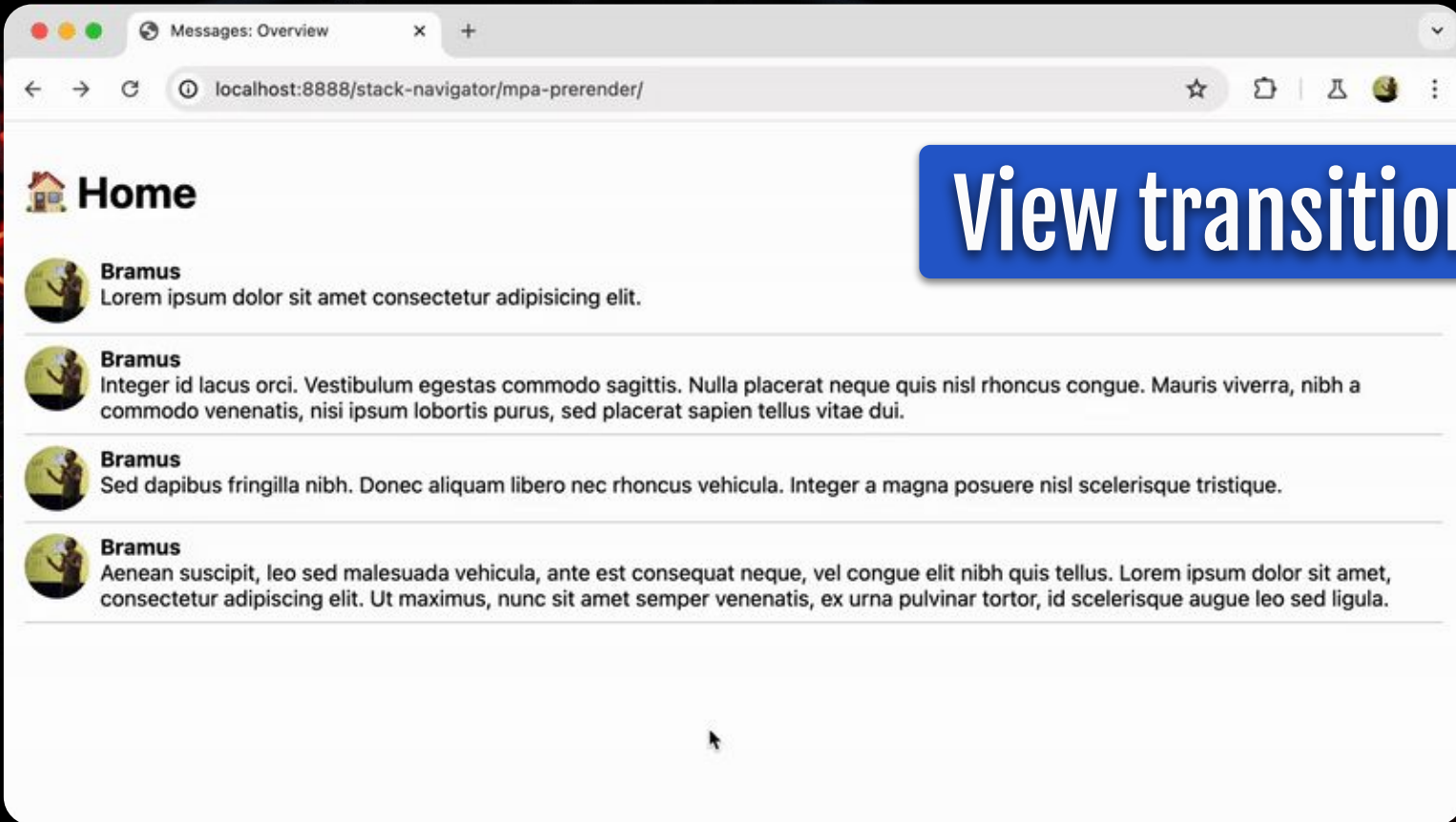
```
<style>  
summary {cursor:pointer;list-style:none}  
summary::after {content:'▸';float:right;transition:.2s}  
details[open] summary::after {transform:rotate(90deg)}  
</style>
```





THINGS YOU SHOULD BE PLAYING WITH





View transitions



Create view transition in 1 CSS line

View transitions

[Next page](#)

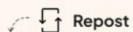
```
@view-transition {  
  navigation: auto;  
}
```

theosoti.com

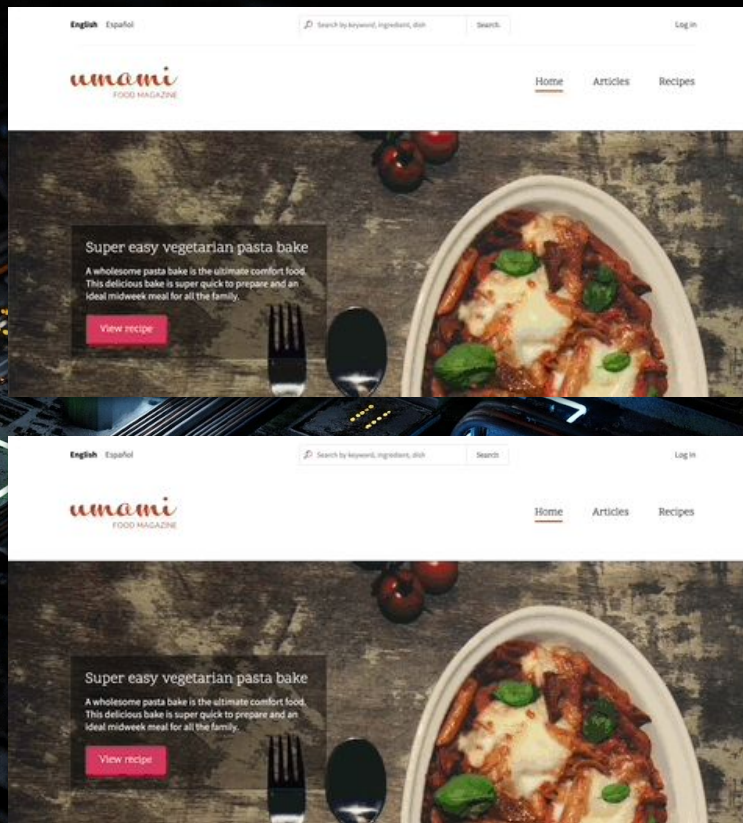


Theodore Soti
@TheoSoti

Checkout my blog
→ theosoti.com



Repost



Jono Alderson

Independent technical SEO consultant



<ViewTransition>

Canary

The `<ViewTransition />` API is currently only available in React's Canary and Experimental channels.

[Learn more about React's release channels here.](#)

`<ViewTransition>` lets you animate elements that update inside a Transition.

```
import {ViewTransition} from 'react';
```

```
<ViewTransition>  
  <div>...</div>  
</ViewTransition>
```



Horizontal Cards

Scroll Buttons

Scroll Markers

Scroll-State Queries

Scroll Initial Target

Anchor

Full Bleed

Buttons and markers both anchored to the bottom



Jam Strut

Ut enim ad minim veniam, quis nostrud
exercitation ullamco laboris nisi ut aliquip
ex ea commodo consequat.

Check it out



Pure CSS carousels



Jono Alderson

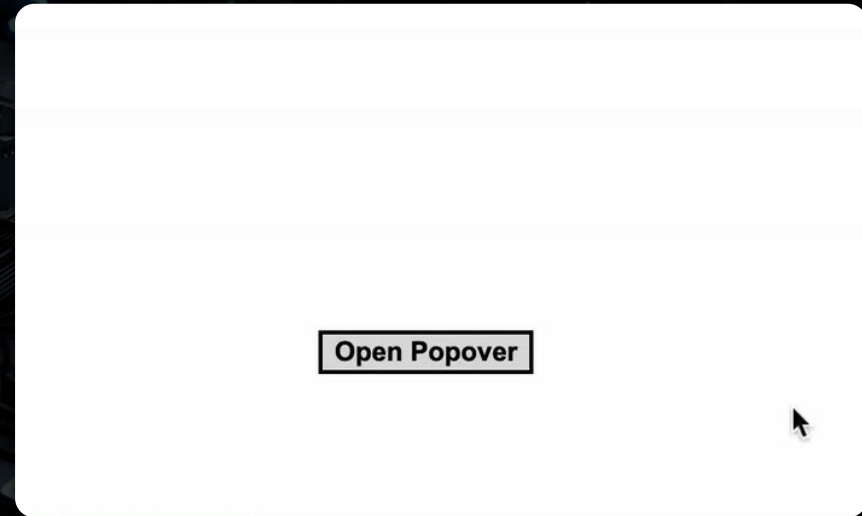
independent technical SEO consultant



Popover

```
<button popovertarget="mypopover">  
  Toggle popover  
</button>
```

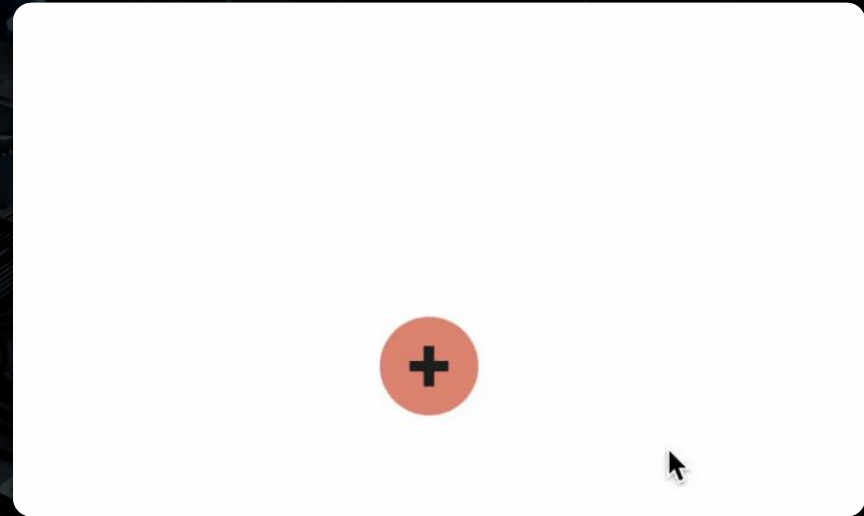
```
<div id="mypopover" popover>  
  Popover content  
</div>
```



Popover

```
<button popovertarget="mypopover">  
  Toggle popover  
</button>
```

```
<div id="mypopover" popover>  
  Popover content  
</div>
```

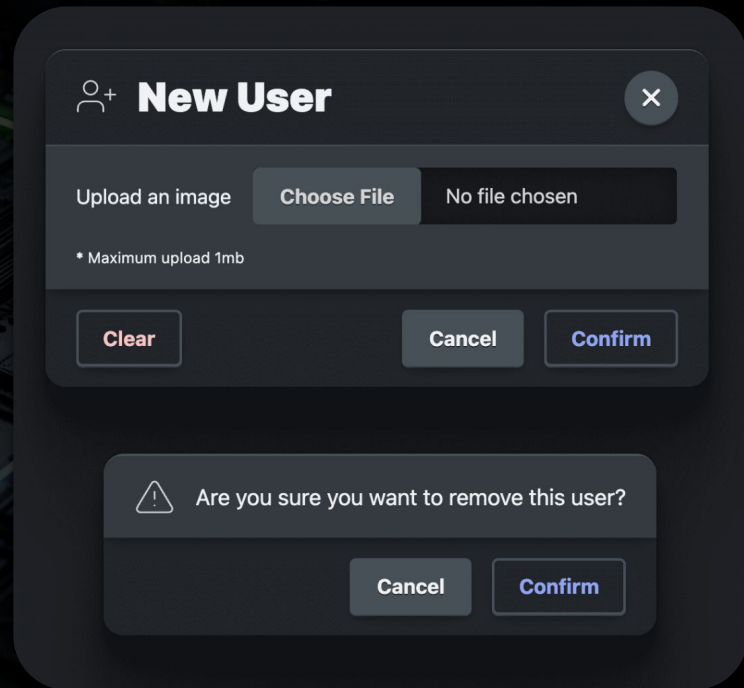


Dialog

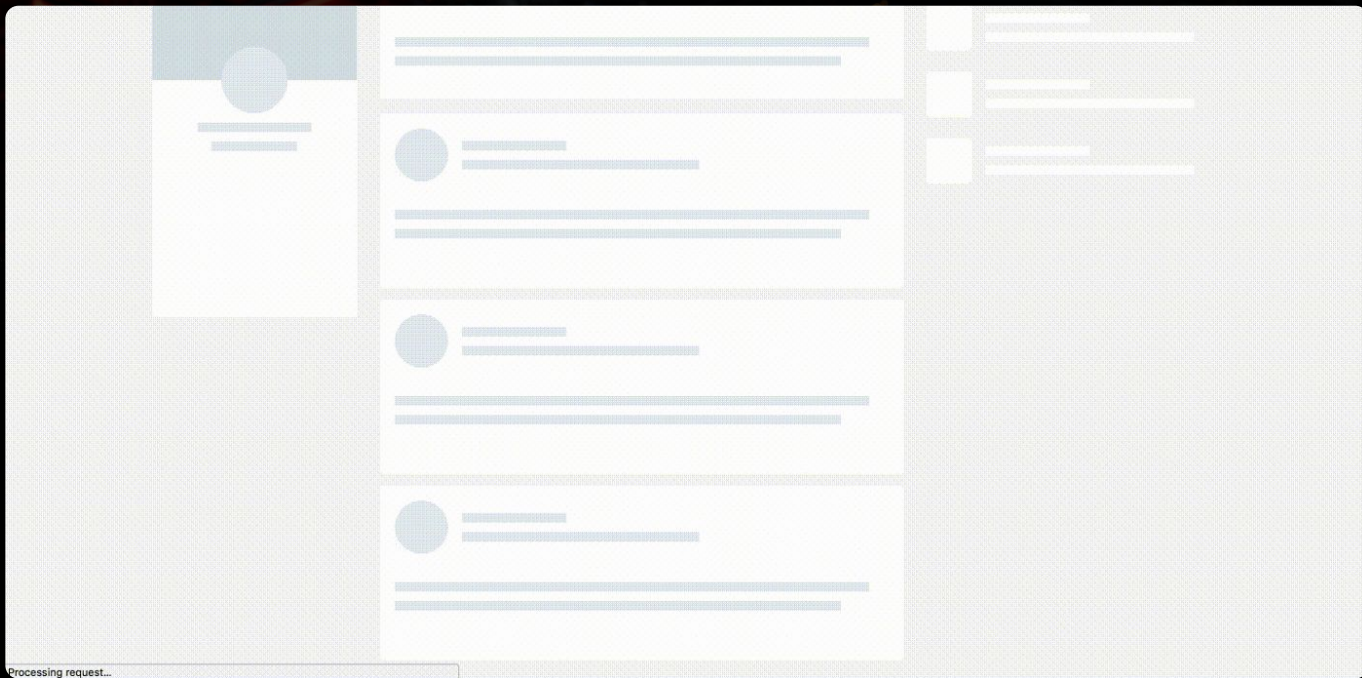
```
<button onclick="example.showModal()">  
  Open  
</button>
```

```
<dialog id="example">  
  <p>Hello world</p>
```

```
  <button onclick="example.close()">  
    Close  
  </button>  
</dialog>
```



Content visibility



Content visibility

```
<ul>  
  <li class="card">[...]</li>  
  <li class="card">[...]</li>  
  [...]  
</ul>
```

```
.card {  
  content-visibility: auto;  
  contain-intrinsic-size: 400px 300px;  
}
```



Containment

```
.card {  
  contain: layout paint style;  
  /* or: contain: content; */  
}
```



Change management

```
.hero img {  
  transition: transform .3s ease;  
  will-change: transform; /* pre-promotes layer */  
}  
  
.hero img:hover {  
  transform: scale(1.05);  
}
```



Speculation rules

```
<script type="speculationrules">
{
  "prefetch": [
    {
      "where": { "href_matches": "^/" }, // same-origin navigations
      "eagerness": "moderate"           // hover 200ms or pointerdown
    }
  ]
}
</script>
```



Speculation rules

"prefetch": [

```
{ // High-value links you've explicitly marked up
  "where": { "selector_matches": "a[data-prefetch]" },
  "eagerness": "immediate"
},
{ // Exclude sensitive routes
  "where": {
    "and": [
      { "href_matches": "^/" },
      { "not": { "href_matches": "^/(logout|checkout|admin)" } },
      { "not": { "selector_matches": "a[data-prefetch='off']" } }
    ]
  },
  "eagerness": "moderate"
}
]
```

"prerender": [

```
{ // Only prerender links you explicitly opt-in
  "where": { "selector_matches":
    "a[data-prefetch='prerender']" },
  "eagerness": "immediate"
},
{ // Or prerender highly likely next pages
  "where": {
    "and": [
      { "href_matches": "^/article/[^#?]+$" },
      { "selector_matches": "a[rel='next']" }
    ]
  },
  "eagerness": "conservative" // on pointer/touch down
}
]
```



Form controls

Date, Time & Color Pickers

Date

dd/mm/yyyy



Time

--:--



Datetime-Local

dd/mm/yyyy --:--



Month



Week

Week --, ----



Color



File Input

Upload Document

Choose File

No file selected



Form controls

```
form:has(:invalid) {  
  outline: 1px solid red;  
}
```

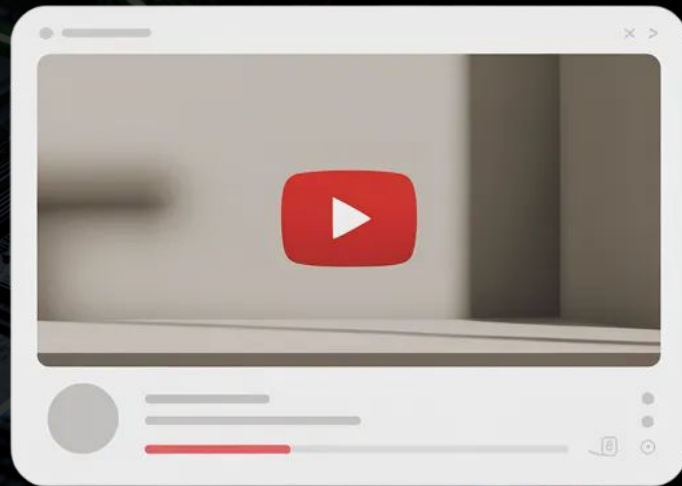
```
.field:has(input:invalid) label {  
  color: red;  
}
```

```
.field.password:has(input:focus) .strength-hint {  
  opacity: 1;  
}
```



Lazy loading <video>

```
<video  
  src="cats.mp4"  
  poster="cat.webp"  
  preload="metadata"  
  loading="lazy"  
>
```



No-vary-search

No-Vary-Search

Fix cache fragmentation from tracking parameters

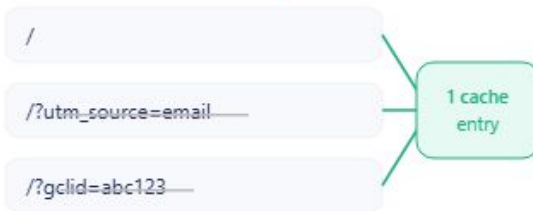
Without



3× downloads

Same page fetched every time

With No-Vary-Search



1× download

Params ignored, cache reused

Add one header:

`No-Vary-Search: params=("utm_source" "utm_campaign" "gclid")`



Creating a scroll-spy with 2 lines of CSS

Published on July 29, 2025

* Introduction

Spec example

scroll-target-group

TLDR; the 2 lines

Smoother navigation

Progressive enhancement

Accessibility

Conclusion

Did you know that there's a new CSS feature **landing in Chrome 140** that let's you very easily create trackable table of contents? This "scroll-spy" effect can be achieved with a new CSS property called `scroll-target-group` combined with the `:target-current` pseudo class.

Introduction

Spec example

scroll-target-group

TLDR; the 2 lines

Smoother navigation

Progressive enhancement

Accessibility

Conclusion

now have much more control over creating custom markers (as they no longer are constrained by the limitations of pseudo elements). This demo is just one example of how you can get creative with `scroll-target-group`, but you of course can use it to create more traditional-looking carousels.

"The `scroll-target-group` property allows to overcome such limitations by making HTML anchor elements 'scroll markers'. By specifying fragment identifier authors have 'scroll target into the view' functionality of `::scroll-markers`, but don't have the 'tracking of current scroll marker' one. With `scroll-target-group` property, [the] browser runs an algorithm to determine the 'current scroll marker' and authors can style such anchor element with `:target-current` pseudo class." - [scroll-target-group Explainer](#).

TLDR; the 2 lines of CSS

So TLDR; the 2 lines of CSS that make this happen are:

```
.parent { scroll-target-group: auto; }  
  
:target-current { /* styles for active anchor */ }
```

CSS



[Home](#)[Blog](#)[Docs](#)[Tools](#)[JS/Wasm features](#)[Research](#)

Giving V8 a Heads-Up: Faster JavaScript Startup with Explicit Compile Hints

Published 29 April 2025 · Tagged with [JavaScript](#)

Getting JavaScript running fast is key for a responsive web app. Even with V8's advanced optimizations, parsing and compiling critical JavaScript during startup can still create performance bottlenecks. Knowing which JavaScript functions to compile during the initial script compilation can speed up web page loading.

When processing a script loaded from the network, V8 has to choose for each function: either compile it immediately ("eagerly") or defer this process. If a function that hasn't been compiled is later called, V8 must then compile it on demand.

If a JavaScript function ends up being called during page load, compiling it eagerly is beneficial, because:

- During the initial processing of the script, we need to do at least a lightweight parse to find the function end. In JavaScript, finding the function end requires parsing the full syntax (there are no shortcuts where we could count the curly braces - the grammar is too complex). Doing the lightweight parsing first and the actual parsing afterwards is duplicate work.
- If we decide to compile a function eagerly, the work happens on a background thread, and parts of it are interleaved with loading the script from the network. If we instead compile the function only when it's being called, it's too late to parallelize work, since the main thread cannot proceed until the function is compiled.

You can read more about how V8 parses and compiles JavaScript in [here](#).

Many web pages would benefit from selecting the correct functions for eager compilation. For example, in our experiment with popular web pages, 17 out of 20 showed improvements, and the average foreground parse and compile times reduction was 630 ms.

We're developing a feature, [Explicit Compile Hints](#), which allows web developers to control which JavaScript files and functions are compiled eagerly. Chrome 136 is now shipping a version where you can select individual files for eager compilation.

This version is particularly useful if you have a "core file" which you can select for eager compilation, or if you're able to move code between source files to create such a core file.

You can trigger eager compilation for the whole file by inserting the magic comment

```
///allFunctionsCalledOnLoad
```

*"In our experiment with popular web pages, **17 out of 20** showed improvements, and the average foreground parse and compile times reduction was **630ms**"*

///`allFunctionsCalledOnLoad`

Jono Alderson

independent technical SEO consultant



All of this works together



NATIVE WINS



Render on the server





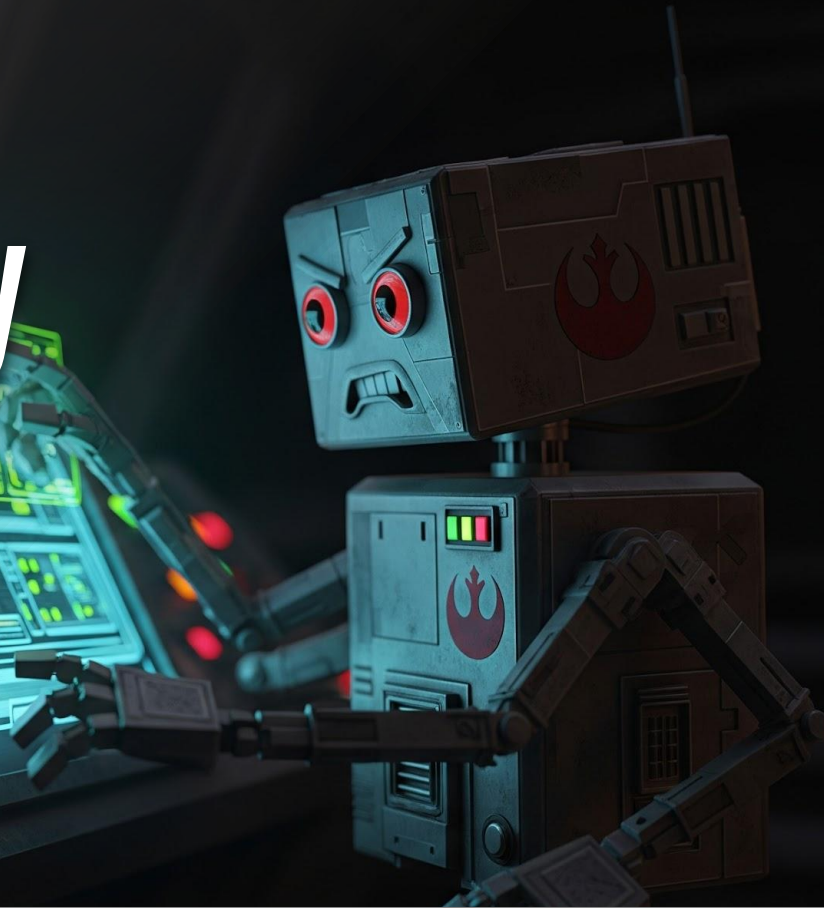
JOIN THE REBELLION



Looking ahead



Structure = Accessibility



Machines are the new users



Browser readiness = AI readiness



Build for humans, Build for machines



Why JavaScript Is Costing You Visibility (and what to use instead)

