

A WORLD WITHOUT BUILDS



CascadiaJS 2025



CascadiaJS



nearform.com/open-source

Hi!

 I'm Ryan Roemer

 I lead technology & OSS at Nearform

 I like & dislike builds

Let's look at some history...

- **1995:** We met JavaScript.
- **1995-2001:** We wrote JavaScript that actually executed in the browser

A vintage beige computer keyboard with a small monitor in the background.

And it looked like this...

Legacy - movies.js

```
1 // ⚠ Have to add properties to global `this`  
2 this.movies = [  
3   { name: "The Princess Bride", year: 1987 },  
4   { name: "The Emperor's New Groove", year: 2000 },  
5 ];
```

Legacy - index.html ↲

```
1 <script src="movies.js">
2   /* ⚠ attached to global `this` as `movies` */
3 </script>
4 <script>
5   // 🕒 ... ye old `var`s and `+` string concatenation!
6   var html = "";
7   for (var i = 0; i < movies.length; i++) {
8     html += "<p>" + movies[i].name + " (" + movies[i].year + ")</p>";
9   }
10
11  var root = document.getElementById("root");
12  root.innerHTML = html;
13 </script>
```

And then ...

- 2001: A **minifier!** (*JSMin*)
- 2010-2015: **Bundlers!**
 - *RequireJS, Browserify, Webpack, ...*
- 2015-on: **Fancier bundlers**
 - *Rollup, Parcel, Vite, ...*

And now...

A modern simple stack usually entails at least a **bundler**, a **transpiler**, and a **style library**.

We run **minified, heavily transformed** code.

We no longer *run* the code we *write*.



And that's OK!

But...

How far could we go without a build?

Spoilers

- This is just a timeline and greatest hits
- Skipping over a ton of technical detail
- Most large scale web apps probably still **need builds**
- We cheat (✨)
 - *Buildless* still runs **builds**

The Interesting Parts

- What works **natively**?
 - No webpack/vite, no TypeScript/JSX/Babel, ...
 - Just **pure JavaScript** running in a browser?
- When can we push the build to **runtime** (e.g. CDN)?
- What can you actually get away with (e.g. Web Components, React)?

Another timeline

The road to buildless...

2015: ECMAScript Modules (ESM)

Named exports

```
1 export const movies = [ ... ];
```

Static and dynamic imports

```
1 // Static import
2 import { movies } from "./movies.js";
3
4 // Dynamic import
5 const { movies } = await import("./movies.js");
```



Let's see some ESM!

Modern - movies.js

```
1 // ⚡ first class ESM export
2 export const movies = [
3   { name: "Moana", year: 2016 },
4   { name: "The Mitchells vs. The Machines", year: 2021 },
5 ];
```

Modern - index.html ↗

```
1 <script type="module">
2   //💡 first class ESM import (static or dynamic).
3   //⚠️ easy to import from relative paths, but what about bare names?
4   import { movies } from "./movies.js";
5
6   const root = document.getElementById("root");
7   root.innerHTML = movies
8     .map((m) => `<p>${m.name} (${m.year})</p>`)
9     .join("");
10 </script>
```



Yay!

... but then we hit all the hard parts

2018: Import Maps

The problem: Can't import "lodash" (or any dep!)

```
1 import lodash from "lodash"; // ⚠ Doesn't work
```

The solution: map a package name to a URL

```
1 <script type="importmap">{
2   "imports": {
3     "lodash": "https://esm.sh/lodash@^4.17.21"
4   }
5 }</script>
```

2018-2021: Import Maps

- 2018: Another problem - not actually **shipped**
- 2018: es-module-shims (✨) to the rescue!
 - **Rewriting** and **polyfilling** not-yet-supported features
- 2021: Finally, broadly **shipped**

2017+: CDNs

The problem: react only ships CommonJS.

The solution: CDNs get more savvy with lazy ESM conversion (✨).

Enter esm.sh, jspm.io, unpkg.com, ...

A dark, atmospheric photograph of four cooling towers emitting plumes of white steam against a cloudy sky.

Let's ship a React app!

React - movies.js

```
1 // ⚡ same as before
2 export const movies = [
3   { name: "Moana", year: 2016 },
4   { name: "The Mitchells vs. The Machines", year: 2021 },
5 ];
```

React (1) - index.html ↶

```
1 <!-- ⚡ The magic -- map bare names to URLs -->
2 <!-- ✨ React now has ESM exports! -->
3 <script type="importmap">
4 {
5   "imports": {
6     "htm": "https://esm.sh/htm@^3?external=react,react-dom",
7     "react": "https://esm.sh/react@18.2.0?",
8     "react/jsx-runtime": "https://esm.sh/react@18.2.0/jsx-runtime?",
9     "react-dom": "https://esm.sh/react-dom@18.2.0?",
10    "react-dom/client": "https://esm.sh/react-dom@18.2.0/client?"
11  }
12 }
13 </script>
```

React (2) - index.html

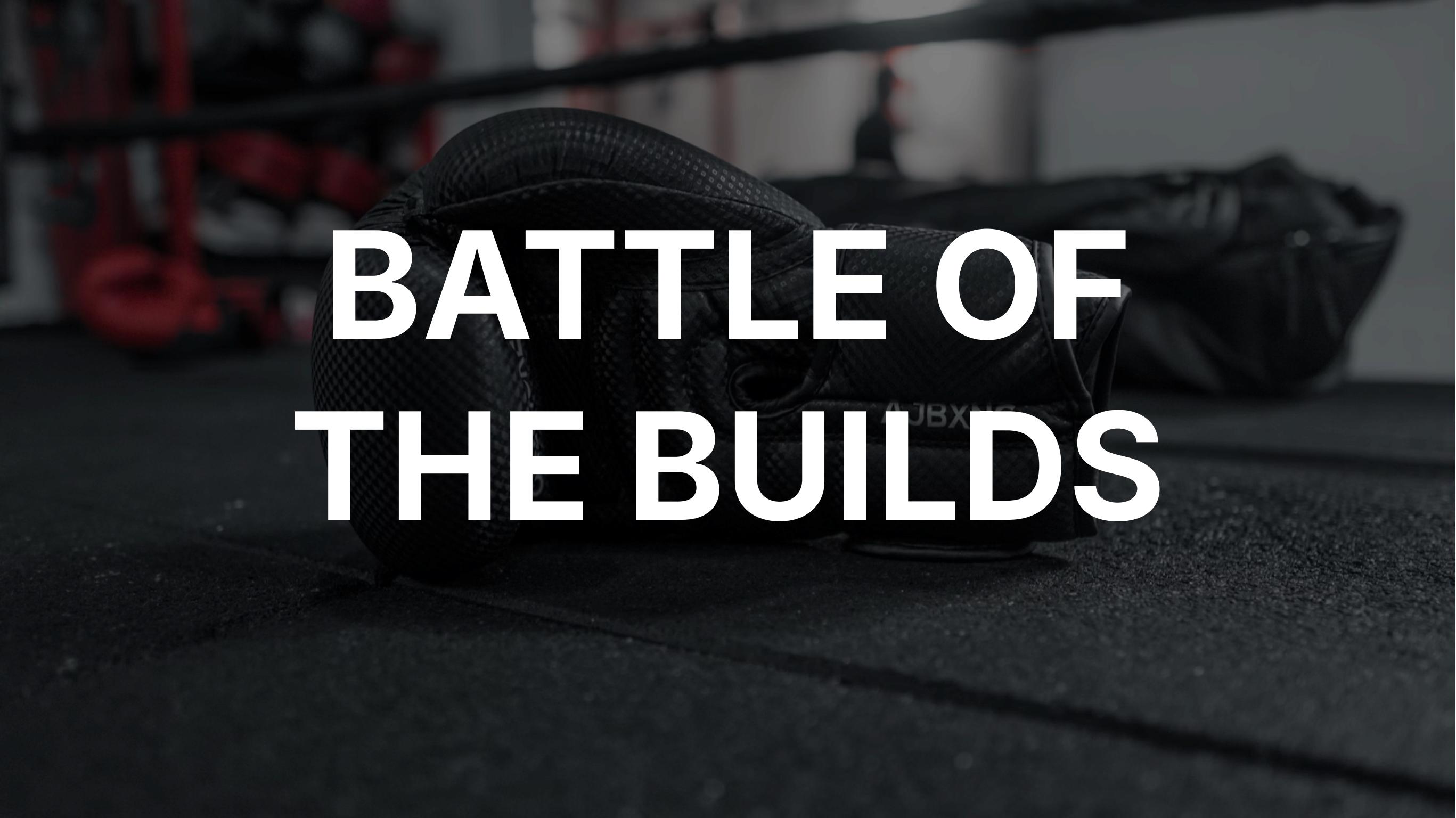
```
1 <script type="module">
2   import { Fragment, createElement } from "react"; //💡 Bare names work now!
3   import { createRoot } from "react-dom/client";
4   import htm from "htm";
5   import { movies } from "./movies.js";
6
7   const html = htm.bind(createElement); //✨ JSX-like template literals!
8   const Movie = ({ name, year }) => html`<p>${name} (${year})</p>`;
9   const App = () => html`
10     <${Fragment}>
11       ${movies.map((movie, i) => html`<${Movie} key=${i} ...${movie} />`)}
12     <${Fragment}>`;
13
14   const root = createRoot(document.getElementById("root"));
15   root.render(html`<${App} />`);
16 </script>
```

The background of the slide is a photograph taken from an airplane window. It shows a vast landscape of snow-covered mountain peaks in the distance, with a layer of green fields and small bodies of water in the foreground. The sky is a clear, vibrant blue.

How about the real world?

Performance

- Spoiler: it's faster to bundle
 - ... but how much?
 - ... and what does your application actually need?
- Let's compare build vs. no build in an unscientific...



BATTLE OF THE BUILDS

∅ No Build

💼 A Build

Bigger:

- 112 requests
- 826 kB

Slower:

- 250 ms

Smaller:

- 5 requests
- 640 kB

Faster:

- 150 ms

Where might we head next?

- The future that *will be*
 - JSON / CSS modules
 - WASM modules
- The future that *could be* (✨ es-module-shims)
 - Hot reload
 - TypeScript!?

TypeScript - movies.ts

```
1 // ! TypeScript types!
2 interface Movie {
3   name: string;
4   year: number;
5 }
6
7 export const movies: Movie[] = [
8   { name: "Moana", year: 2016 },
9   { name: "The Mitchells vs. The Machines", year: 2021 },
10 ];
```

TypeScript - index.html

```
1 <!-- ✨ Add the shims -->
2 <script src="https://ga.jspm.io/npm:es-module-shims@2.6.2/dist/es-module-shims.js"></script>
3
4 <!-- ✨ `lang="ts"` is cheating -->
5 <!-- 💡 `type="module-shim"` skips native check -->
6 <script type="module" lang="ts">
7   // ✨ Can import `*.ts`
8   import { movies } from "./movies.ts";
9
10  const root = document.getElementById("root") as HTMLElement;
11  root.innerHTML = movies
12    .map((m) => `<p>${m.name} (${m.year})</p>`)
13    .join("");
14 </script>
```

Go play around!

- ─ See the deck at:

ryan-roemer.github.io/a-world-without-builds

- ─ ...and check out the code at:

[ryan-roemer/a-world-without-builds](https://github.com/ryan-roemer/a-world-without-builds)



Thanks!

-  [ryan-roemer](https://github.com/ryan-roemer)
-  [in/ryanroemer](https://www.linkedin.com/in/ryanroemer)
-  [@ryan-roemer.bsky.social](https://bsky.social/@ryan-roemer)
-  ryan.roemer@nearform.com

