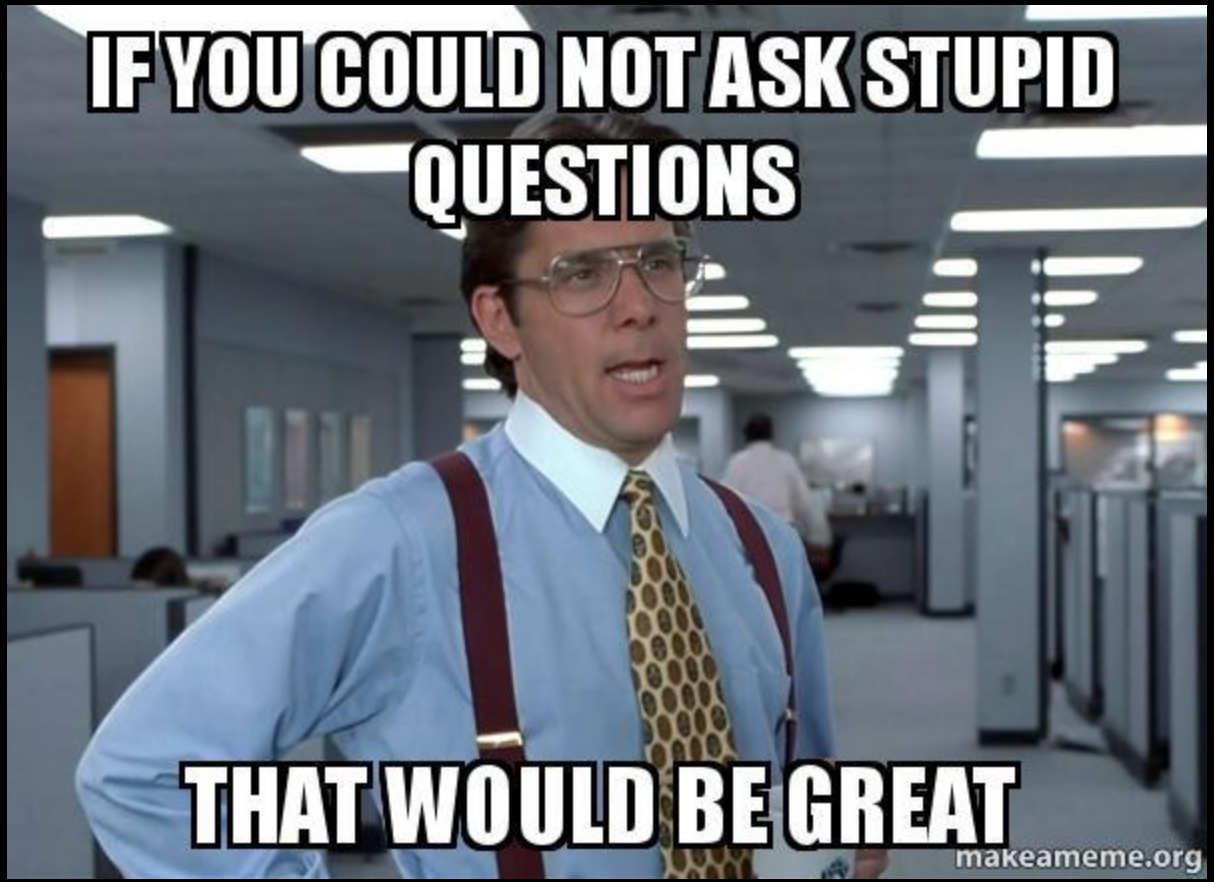

accenture
improving
It's what we do. ™

Building Web Applications without a Framework


Simon MacDonald
@macdonst@mastodon.online
Simon MacDonald
@macdonst@mastodon.online

Ever had a **dependency** break?


IF YOU COULD NOT ASK STUPID QUESTIONS
THAT WOULD BE GREAT
makeameme.org

“I very frequently get the question: **'What's going to change in the next 10 years?'**

I almost never get the question: **'What's not going to change in the next 10 years?'**

And I submit to you that that second question is actually the more important of the two -- because you can build a business strategy around the things that are stable in time.”

- Jeff Bezos

Complications

Web browsers do not break

- Web standards ensure there is consistent behavior
- Browser automatically upgrade themselves and stay ‘evergreen’
- To remain competitive browsers are backwards compatible

Web browsers are backwards compatible, eh

Can **our** source code be stable?

Two kinds of change

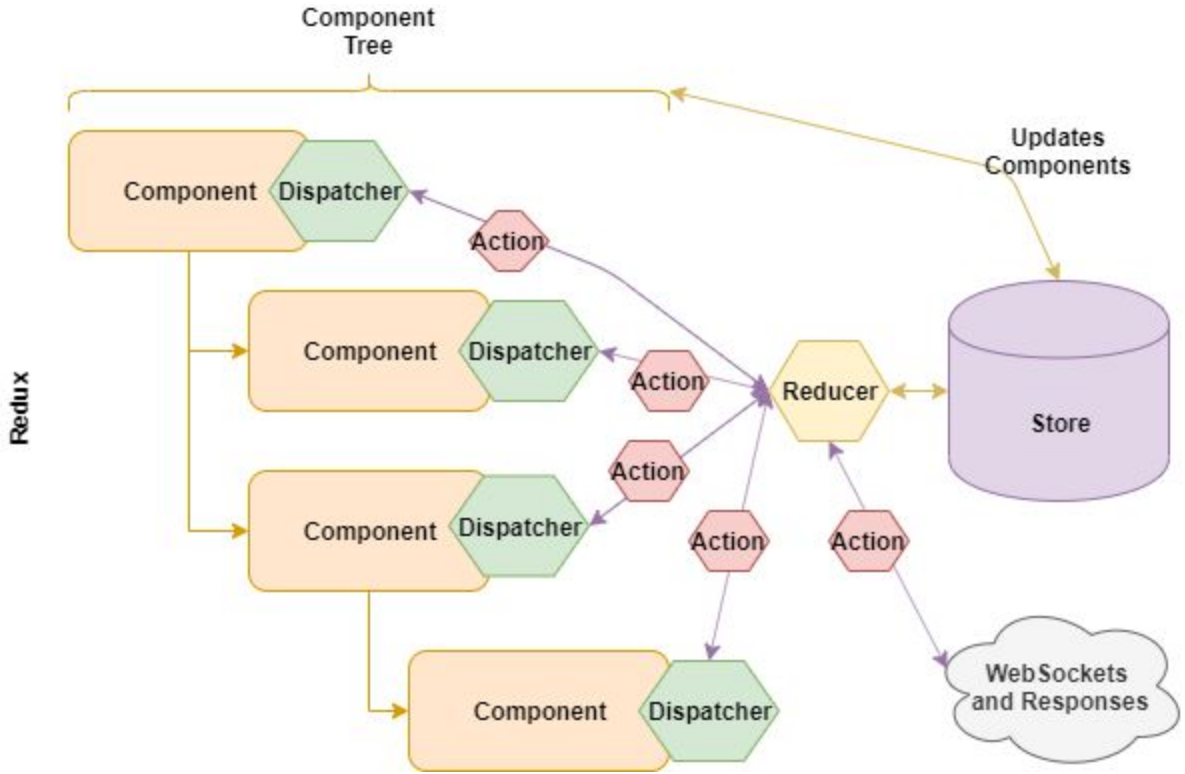
💔 Breaking change (removal)

- Angular 1 -> 2
- React 18
- TypeScript 4.8

✅ Additive change

- HTTP 1 -> HTTP 2
- XMLHttpRequest -> Fetch
- async / await
- <script type=module>
- woff -> woff2
- S3.listObjectsV2

Complications



React: Options for State Management

Templating systems

views/pages/index.ejs

Copy

```
<!DOCTYPE html>
<html lang="en">
<head>
  <%- include('../partials/head'); %>
</head>
<body class="container">

<header>
  <%- include('../partials/header'); %>
</header>

<main>
  <div class="jumbotron">
    <h1>This is great</h1>
    <p>Welcome to templating using EJS</p>

    <h2>Variable</h2>
    <p><%= tagline %></p>

    <ul>
      <% mascots.forEach(function(mascot) { %>
        <li>
          <strong><%= mascot.name %></strong>
          representing <%= mascot.organization %>,
          born <%= mascot.birth_year %>
        </li>
      <% }); %>
    </ul>
  </div>
</main>

<footer>
  <%- include('../partials/footer'); %>
</footer>

</body>
</html>
```

Non standard dialects (JSX *is not* JS)

App.js

```
1  import { useState } from 'react';
2
3  function MyButton() {
4    const [count, setCount] = useState(0);
5
6    function handleClick() {
7      setCount(count + 1);
8    }
9
10   return (
11     <button onClick={handleClick}>
12       Clicked {count} times
13     </button>
14   );
15 }
16
17 export default function MyApp() {
18   return (
19     <div>
20       <h1>Counters that update separately</h1>
21       <MyButton />
22       <MyButton />
23     </div>
24   );
25 }
26
```

Non standard dialects (Svelte)

App.svelte

```svelte
<script>
  let x = 7;
</script>

{#if x > 10}
  <p>{x} is greater than 10</p>
{:else if 5 > x}
  <p>{x} is less than 5</p>
{:else}
  <p>{x} is between 5 and 10</p>
{/if}
```

The problem w transpilers is obfuscated runtime code.

This is bad. But it gets worse.

```
(self.webpackChunk_N_E = self.webpackChunk_N_E || []).push([[59172], {
  95627: function(t, e, r) {
    var i = r(15686).Buffer,
      n = r(4163);
    !function() {
      var e,
        a,
        o,
        s,
        f,
        h,
        d,
        c,
        u = {
          7160: function(t, e, r) {
            var i = e;
            i.bignum = r(711),
            i.define = r(495).define,
            i.base = r(853),
            i.constants = r(7335),
            i.decoders = r(6701),
            i.encoders = r(3418)
          },
          495: function(t, e, r) {
            var i = r(7160),
              n = r(3782);
            function a(t, e) {
              this.name = t,
              this.body = e,
              this.decoders = {},
              this.encoders = {}
            }
            e.define = function(t, e) {
              return new a(t, e)
            },
            a.prototype._createNamed = function(t) {
              var e;
              try {
                e = r(6144).runInThisContext("(function " + this.name + "(entity) {\n
this._initNamed(entity);\n})")
              } catch (i) {
                e = function(t) {
                  this._initNamed(t)
                }
              }
              return n(e, t), e.prototype._initNamed = function(e) {
                t.call(this, e)
              }, new e(this)
            },
            a.prototype._getDecoder = function(t) {
```

12k LOC is totally normal for a trivial bundle.

This is not efficient.

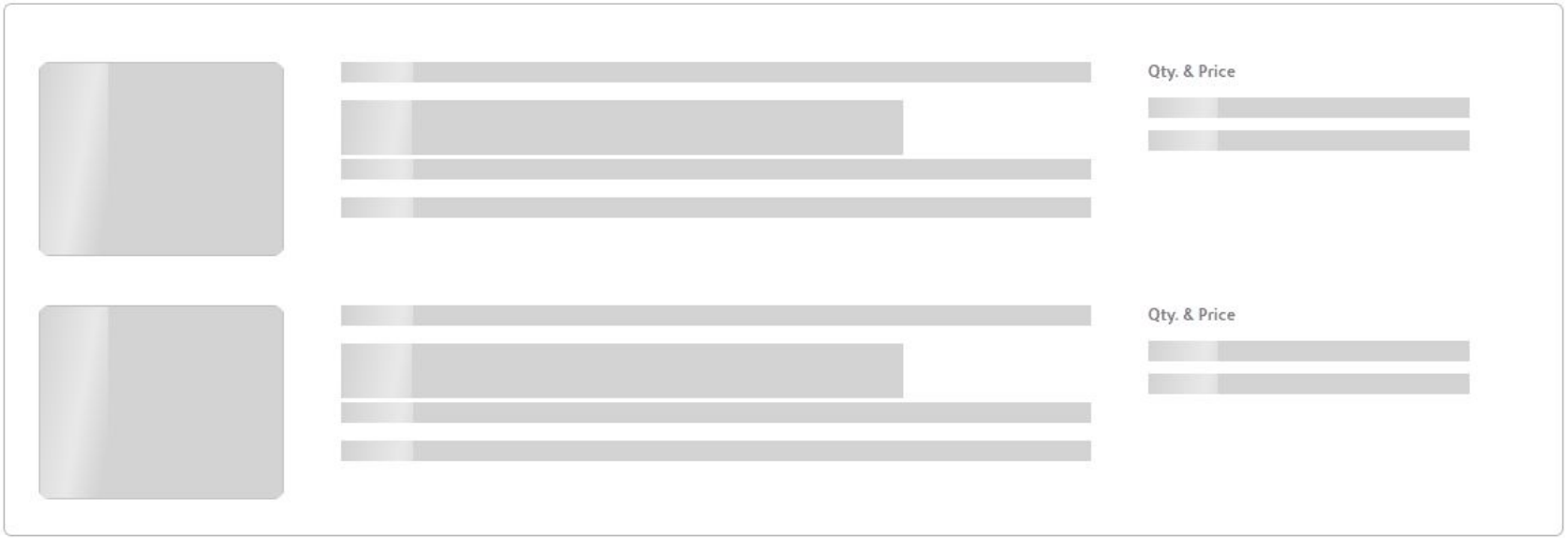
And terrible DX to debug.

```
      e.Decipheriv = s.Decipheriv,
      e.createDecipheriv = s.createDecipheriv,
      e.getCiphers = s.getCiphers,
      e.listCiphers = s.listCiphers,
      f = p(6587),
      e.DiffieHellmanGroup = f.DiffieHellmanGroup,
      e.createDiffieHellmanGroup = f.createDiffieHellmanGroup,
      e.getDiffieHellman = f.getDiffieHellman,
      e.createDiffieHellman = f.createDiffieHellman,
      e.DiffieHellman = f.DiffieHellman,
      h = p(4078),
      e.createSign = h.createSign,
      e.Sign = h.Sign,
      e.createVerify = h.createVerify,
      e.Verify = h.Verify,
      e.createECDH = p(9942),
      d = p(9783),
      e.publicEncrypt = d.publicEncrypt,
      e.privateEncrypt = d.privateEncrypt,
      e.publicDecrypt = d.publicDecrypt,
      e.privateDecrypt = d.privateDecrypt,
      c = p(6445),
      e.randomFill = c.randomFill,
      e.randomFillSync = c.randomFillSync,
      e.createCredentials = function() {
        throw Error("sorry, createCredentials is not implemented yet\nwe accept pull
requests\nhttps://github.com/crypto-browserify/crypto-browserify")
      },
      e.constants = {
        DH_CHECK_P_NOT_SAFE_PRIME: 2,
        DH_CHECK_P_NOT_PRIME: 1,
        DH_UNABLE_TO_CHECK_GENERATOR: 4,
        DH_NOT_SUITABLE_GENERATOR: 8,
        NPN_ENABLED: 1,
        ALPN_ENABLED: 1,
        RSA_PKCS1_PADDING: 1,
        RSA_SSLV23_PADDING: 2,
        RSA_NO_PADDING: 3,
        RSA_PKCS1_OAEP_PADDING: 4,
        RSA_X931_PADDING: 5,
        RSA_PKCS1_PSS_PADDING: 6,
        POINT_CONVERSION_COMPRESSED: 2,
        POINT_CONVERSION_UNCOMPRESSED: 4,
        POINT_CONVERSION_HYBRID: 6
      },
      t.exports = b
    }()
  }
}]);
```

Do you even know what your tools are doing inside those bundles?

```
1   (self.__LOADABLE_LOADED_CHUNKS__=self.
    __LOADABLE_LOADED_CHUNKS__||[]).push([[9158],{9158:(_,e,r)⇒
    {var o=r(85081)("toLower",r(46129),r(39478));o.placeholder=r
    (48051),_.exports=o},46129:(_,e,r)⇒{var o=r(26478);_.
    exports=function(_){return o(_).toLowerCase()}}}]);
```

Static != Dynamic




Too many moving parts.

Can we simplify this situation?

A new case for **progressive enhancement**

Progressive Enhancement

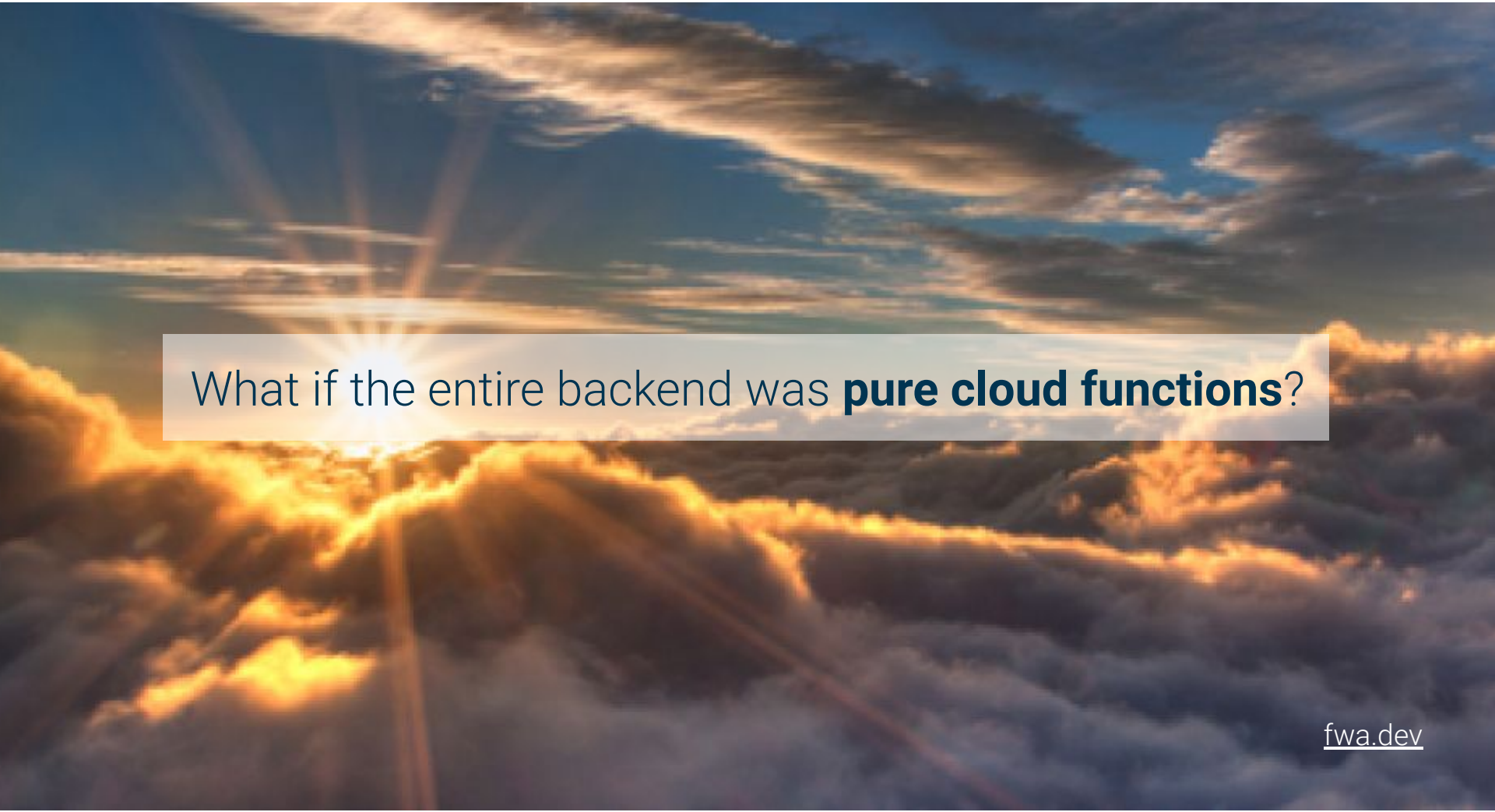
Start with **working HTML anchors and forms**. Make it better with a <script>.

Next generation frontend frameworks are HTML first

We **agree** with HTML-first progressive enhancement;
our **backend framework** always recommended it 👇

Architect

https://arc.codes


What if the entire backend was **pure cloud functions**?
fwa.dev

“Modern” JS

Major problems

1. Brittle incompatible niche ecosystems
2. Non standard template libraries or, worse, opaque programming languages
3. Static, not dynamic, resulting in spinners/skeleton screens


Can our frontend source be pure
standards-based **HTML**, **CSS** and **JS**?

Use The Platform

Web Components

1. Custom Elements
2. Shadow DOM
3. Slots and Templates

Web Component Problems

1. Doesn’t work without JavaScript, i.e. breaks Progressive Enhancement
2. Flash of Unstyled Custom Element
3. Doesn’t play well with native Forms

Enhance is an ***HTML framework***

1. Author HTML pages
2. Use generally available web standards
3. **Progressively enhance working HTML**

enhance.dev

🙏 to the demo gods

Enhance key concepts

- ✔ File based routing with plain HTML
- ✔ Reuse markup with custom elements
- ✔ Built-in utility CSS based on scales rather than absolute values
- ✔ API routes without manually wiring props
- ✔ Progressively enhance with standard JS; no special syntax
- ✔ Fullstack FWA under the hood


enhance.dev
Frontend
Backend

ENHANCE

The HTML framework

Read the Docs:	**enhance.dev/docs**
Join our Discord:	**enhance.dev/discord**
Follow us on Mastodon:	**fosstodon.org/@enhance_dev**