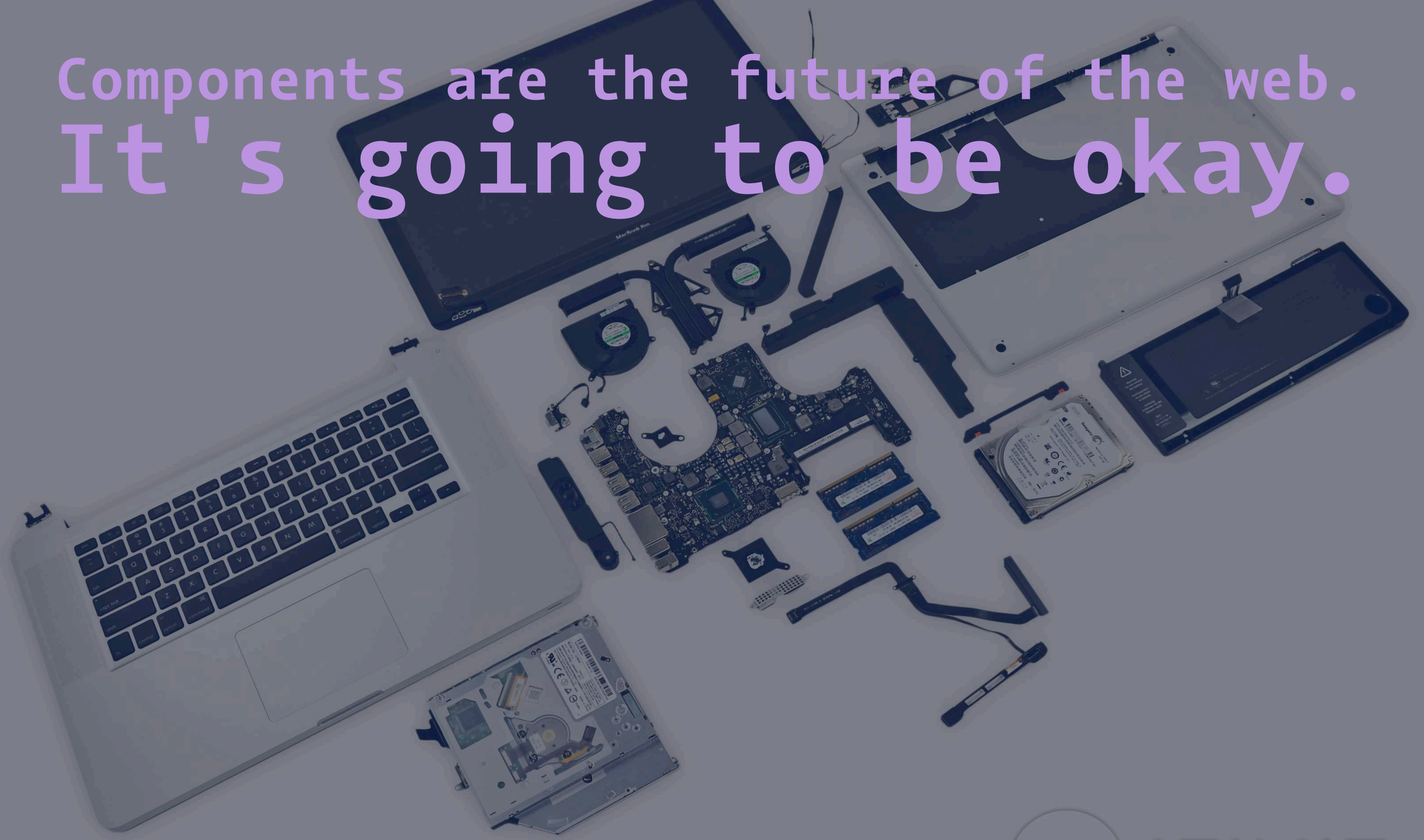


Components are the future of the web.
It's going to be okay.





Tessa Thornton

Front-end developer.

I work at Shopify.

In Toronto.

@tessthornton

A romanticized and slightly
inaccurate history of web
apps

```
<form action="submit.php" method="POST">
  <input name="name" type="text">
  <select name="favourite_color">
    <option value="00ffff">Red</option>
    <option value="ff00ff">Blue</option>
    <option value="ff0000">Yellow</option>
  </select>
  <input type="submit">
</form>
```

Then JavaScript grew up

```
<form>
```

```
...
```

```
<button onclick="submitForm()">Submit</button>
```

```
</form>
```

~~<form>~~

BAD

~~...~~

~~<button onclick="submitForm()">Submit</button>~~

~~</form>~~

Strict separation of concerns!

- HTML is for **content**
- JavaScript is for **behaviour**
- CSS is for **presentation**

and so it was written

```
<button id="formSubmitButton">Submit</button>
```

```
<script>
```

```
$("#formSubmitButton").click(submitForm);
```

```
</script>
```

and we called it **Best Practices**

Remember when Our best practices were killing us?



... We did it again.

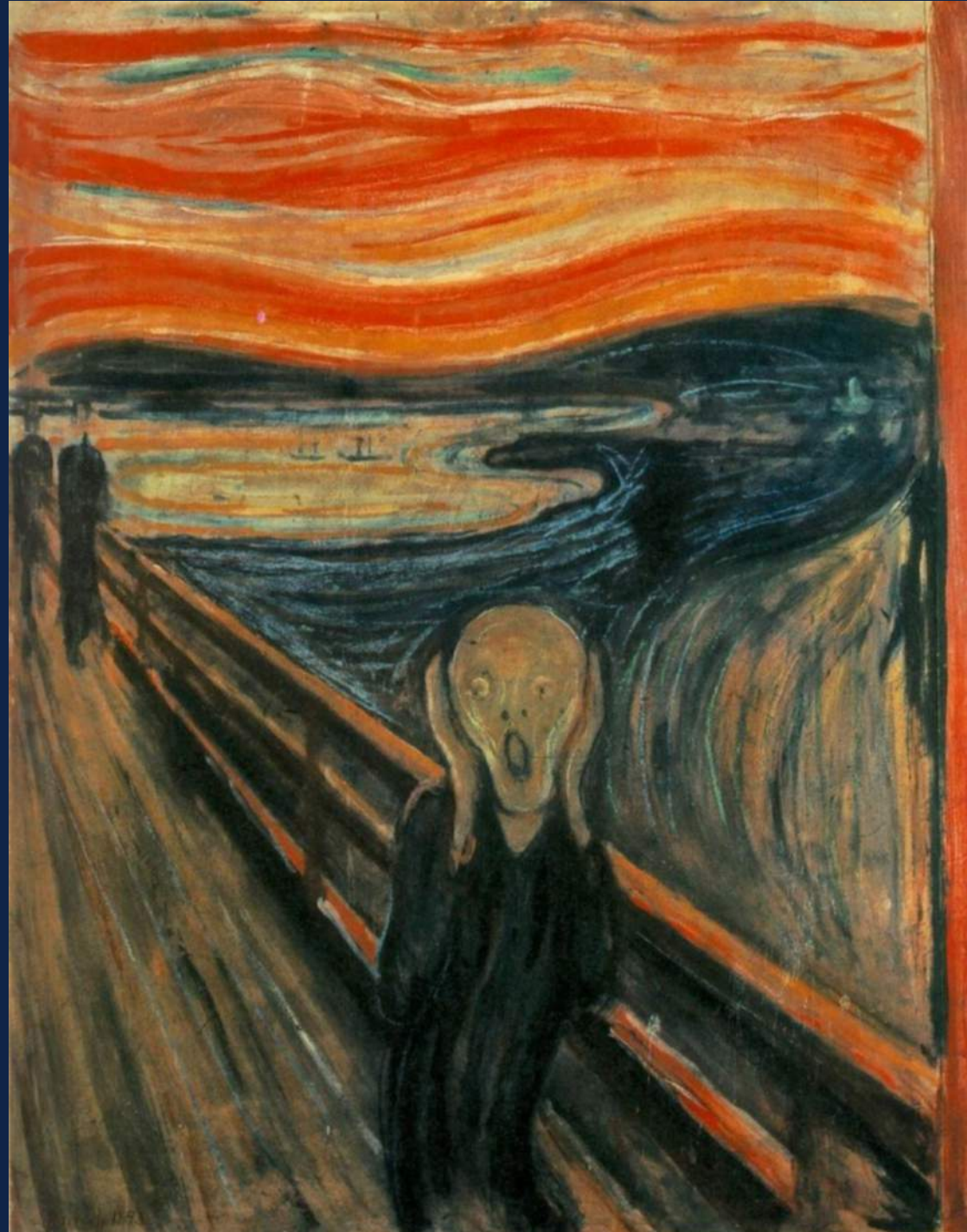
In the fast-changing world of JavaScript-heavy interactive web applications, these best practices haven't kept up

Is HTML **really** for content?

```
<select id="countryList"></select>
<script>
$.get('countrylist.json', function(data) {
    $('#countryList').populateOptions(data);
});
</script>
```



```
<input type="range" min="0" max="10" step="2" value="6">
```



HTML isn't just for content,
and it never was

```
<input type="range" min="0" max="10" step="2" value="6">
```

```
<picture>
```

```
  <source srcset="images/extralarge.jpg" media="(min-width: 1000px)">
```

```
  <source srcset="images/large.jpg" media="(min-width: 800px)">
```

```
  <img srcset="images/medium.jpg" alt="A giant stone face ">
```

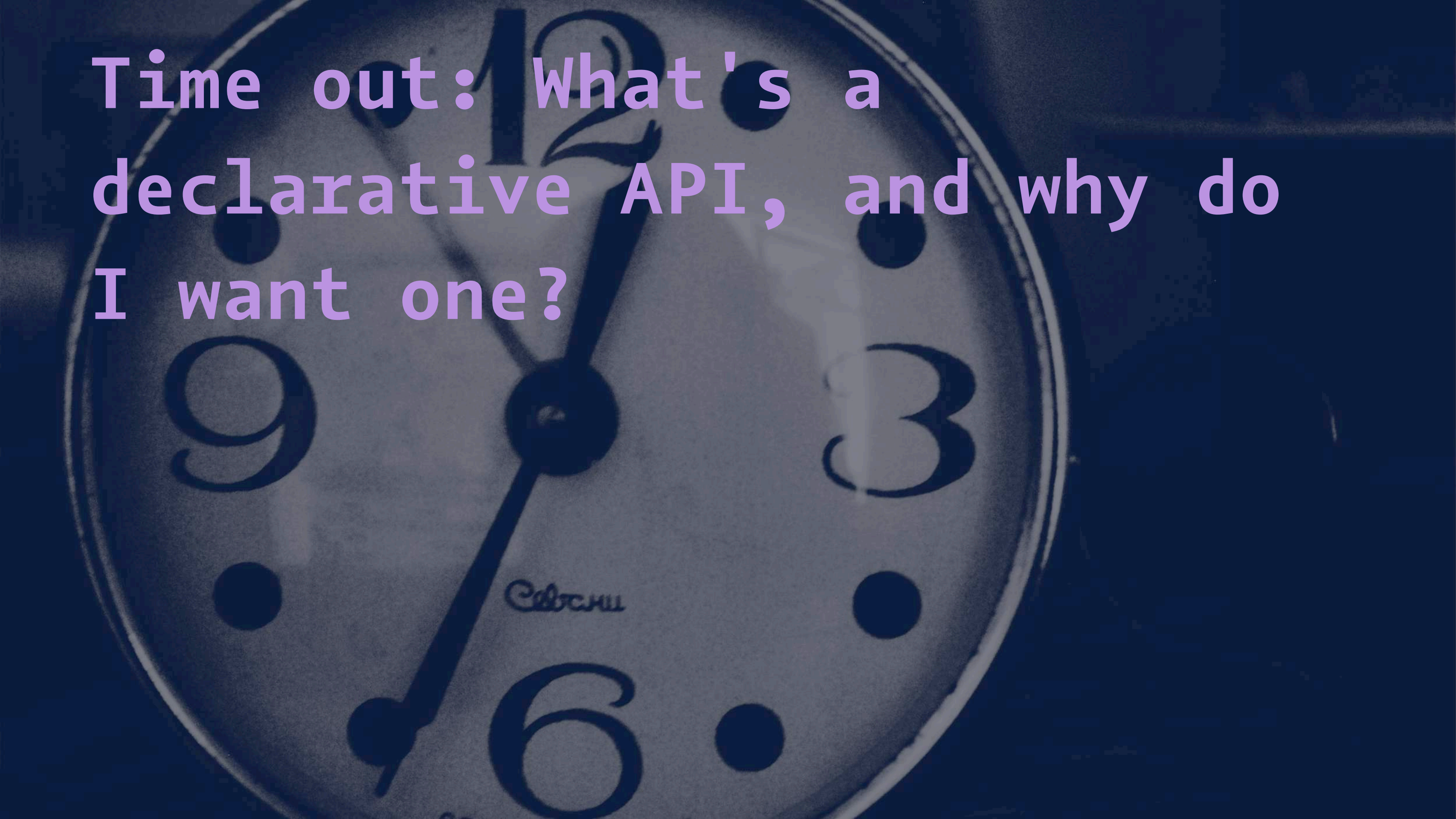
```
</picture>
```

```
<select name="provinces">
```

```
  <option value="AB">Alberta</option>
```

```
  <option value="BC">British Columbia</option>
```

```
</select>
```

Time out: What's a
declarative API, and why do
I want one?

Declarative vs. Imperative programming

Imperative code instructs the compiler what steps to take to achieve the desired result

Declarative code specifies what the desired result is and lets the compiler figure out the implementation

HTML has declarative APIs

```
<input type="range" min="0" max="10" step="2" value="6">
```

[jsbin](#)

We didn't really learn from those built-in examples

```
<button id="launchSalesModal">Launch Modal</button>
```

```
<script>
```

```
$("#launchSalesModal").on("click", function() {  
  library.launchModal({  
    el: "#salesModal"  
  });  
});
```

```
</script>
```

Like, *what is this even*:

```
<button id="launchSalesModal">Launch Modal</button>
```

What about this bit of JavaScript:

```
<script>  
$("#launchSalesModal").on("click", function() {});  
</script>
```

So what does `$("#launchSalesModal")` even do?

It gets even worse when we do things like ~data binding~

```
<input type="text">
```

```
<p></p>
```

```
<script>
```

```
$("#input").on("keyup", function(e) {
```

```
    $("#p").text(this.value);
```

```
});
```

```
</script>
```



*We're writing JavaScript
like we don't have control
over our HTML*

We think our HTML should be dumb, but then we have to do all sorts of shit in JS to make up for it

So what happens when we re-introduce some declarative attributes to our HTML?

2009: Meet Angular.js


```
<a href="" ng-click="archive()">archive</a>
```



Angular's HTML made a lot of people very angry

```
<a href="" ng-click="archive()">archive</a>
```

*"I don't like how it breaks the separation between
html/js/css"*

"I hate the way the HTML looks"

"Isn't this a step backwards???"

-- some nerds on twitter

```
<a href="" ng-click="archive()">archive</a>
```

OMG separation of concerns

I have a lot of things to say
about this




```
<a href="" ng-click="archive()">archive</a>
```

OMG inline event handlers

— it's ok



```
<a href="" ng-click="archive()">archive</a>
```

But it looks bad and you should feel bad



— no

**FORWARD, NOT BACKWARD
UPWARD, NOT FORWARD**



**AND ALWAYS TWIRLING, TWIRLING, TWIRLING
TOWARDS FREEDOM**

If we're ok with this then...

```
<a href="" ng-click="archive()">archive</a>
```



... let's slide down this slippery slope

What if you could make up your own declarative attributes?

```
<button onClick="openModoal()" modal-target="#salesModal">Open</button>
```

WHAT IF YOU COULD MAKE UP YOUR OWN DECLARATIVE ELEMENTS??

```
<modal-trigger-button target="#salesModal">Open</modal-trigger-button>
```


Welcome to the future it is
called web components

What are Web Components?

- Custom elements
- Shadow DOM
- templates
- HTML imports

An example: tabs

```
<div id="tabs" class="tabs-container">
  <ul class="tabs-nav">
    <li><a href="#tab1">Tab 1</a></li>
    <li><a href="#tab2">Tab 2</a></li>
  </ul>
  <div class="tabs-content">
    <div id="tab1">Tab 1 content</div>
    <div id="tab2">Tab 2 content</div>
  </div>
</div>

<script>
$("#tabs").tabs();
</script>
```

A better way

```
<demo-tabs>  
  <demo-tab title="Tab 1">Tab 1 content</demo-tab>  
  <demo-tab title="Tab 2">Tab 2 content</demo-tab>  
</demo-tabs>
```

```
<template id="template">
  <p>I'm in Shadow DOM. My markup was stamped from a &lt;template&gt;.</p>
</template>
```

```
<script>
var proto = Object.create(HTMLElement.prototype, {
  createdCallback: {
    value: function() {
      var t = document.querySelector('#template');
      var clone = document.importNode(t.content, true);
      this.createShadowRoot().appendChild(clone);
    }
  }
});
document.registerElement('x-foo', {prototype: proto});
</script>
```

```
<x-foo></x-foo>
```

OKAY I WANT SOME WEB
COMPONENTS GIVE ME SOME WEB
COMPONENTS

...it's a little bit complicated right now.

Browser support for these specs is... not great.

YET.

BUMMER

There's some options though

Po1ymer

www.polymer-project.org/


```
<polymer-element name="x-foo">
  <template id="template">
    <p>I'm in Shadow DOM. My markup was stamped from a <template>.</p>
  </template>
</polymer-element>

<x-foo></x-foo>
```

The philosophy behind
components has SPREAD to a
framework near you

Angular Directives



```
<tabset>
  <tab ng-repeat="tab in tabs" heading="{{tab.title}}" active="tab.active">
    {{tab.content}}
  </tab>
</tabset>
```

Angular + Web Components + the future

Angular 2.0 will support Web Components out of the box. Not sure what this will look like yet.

Ember COMPONENTS

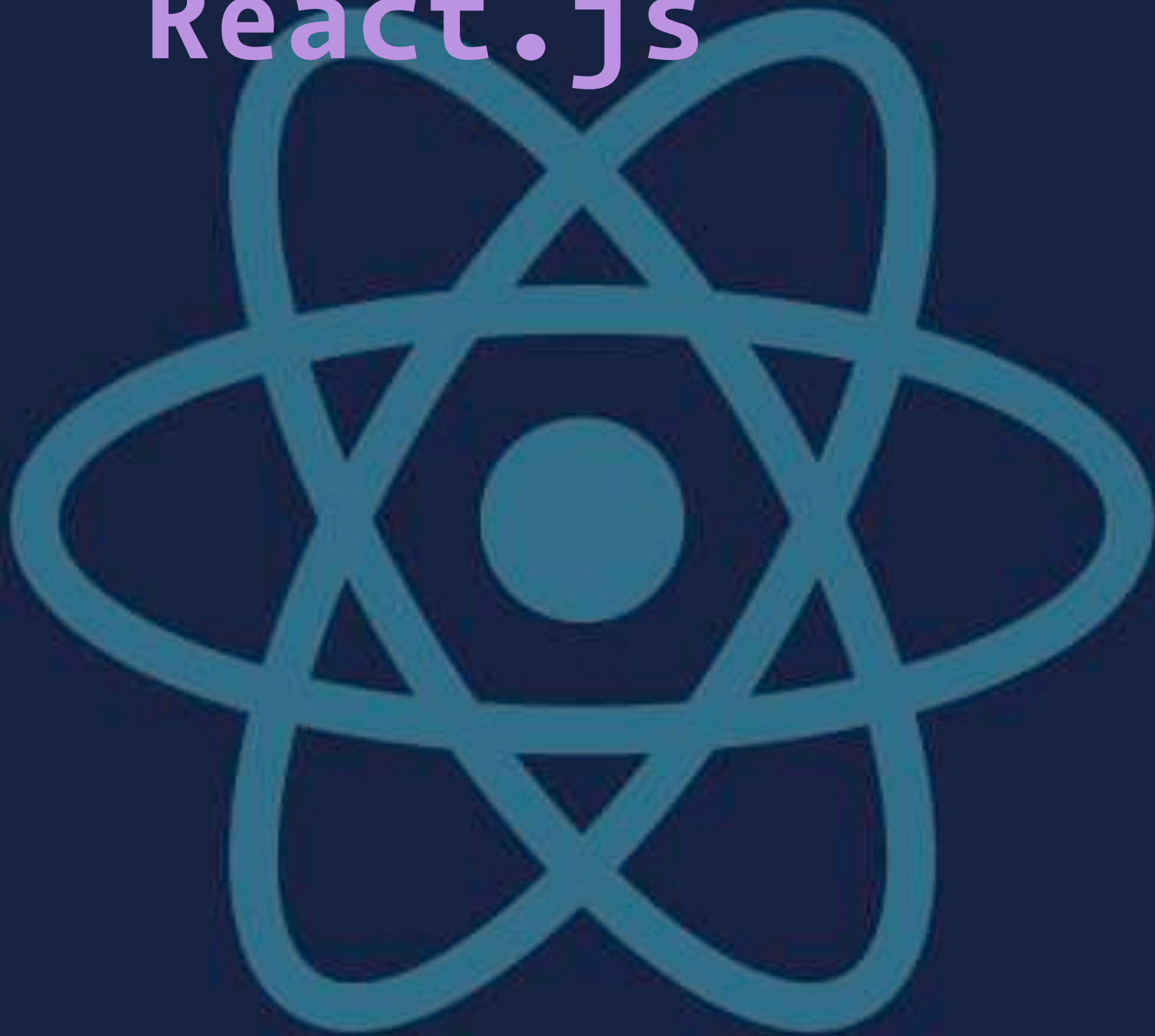
```
{{demo-tabs}}  
  {{demo-tab title="Tab 1"}}Tab 1 content{{/demo-tab}}  
  {{demo-tab title="Tab 2"}}Tab 2 content{{/demo-tab}}  
{{/demo-tabs}}
```



Ember + Web Components + the future

- Ember 2.0 makes components much more important

React.js



**SO HOT RIGHT
NOW**

memegenerator.net

```
React.createClass({
  render: function() {
    return (
      <Tabs>
        <Tabs.Panel title='Tab #1'>
          <h2>Content #1 here</h2>
        </Tabs.Panel>
        <Tabs.Panel title='Tab #2'>
          <h2>Content #2 here</h2>
        </Tabs.Panel>
      </Tabs>
    );
  }
});
```

Even if you don't buy 100% into the components thing, you can still ease your pain with declarative data-binding

- knockout.js

- twine.js

```
<input type="text" bind="color" value="blue">  
<strong bind="color"></strong>
```

In Summary

the "Declarative renaissance" of the web is coming whether you like it or not

- Angular is doing it
- Ember is doing it
- React is doing it
- The browsers themselves are doing it

But you're going to like it

- declarative APIs
- Meaningful HTML
- encapsulation (styles too in the native spec)
- re-usability
- composability

Choose your own adventure

- Angular directives
- Ember components
- React components
- Polymer
- Native (Coming Soon(ish))

You're going to "pollute"
your HTML and you're going
to like it

Links

- [Declarative vs imperative programming](#)
- [Our best practices are killing us](#)
- [The web's declarative, composable future](#)
- [Angular.js](#)
- [Angular 2.0](#)
- [Ember.js](#)
- [Ember road to 2.0](#)

- React.js
- Polymer
- Web Components
- Knockout.js
- Twine.js