FRONT-END DEVELOPMENT

IT'S JUST...

HTML + CSS + JavaScript*

* an oversimplification

HOW DO YOU GET THOSE THINGS TO THE USER?

And how do you ensure those things are maintainable?

APPEALING

» Fast feedback

» Tangible/visual

» Easy to learn

» Fun

UNAPPEALING

» The user is in control of browser, device, dimensions, etc.*

» Framework choices & pace-of-change*

» Available technologies*

» Changing platform*

» Browser variation

» Hard to master

* Depending on your PoV

“The Web is the most hostile software engineering environment imaginable”

-- Douglas Crockford

HTML

Hypertext Markup Language

A set of elements for describing documents and relationships between them.

```
<!DOCTYPE html>
<html lang="en-NZ">
<head>
  <meta charset="utf-8">
  <title>Summer of Tech</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <h1>Summer of Tech</h1>
  <p>
    Hi
  </p>
</body>
</html>
```

¯\(ツ)/¯

Check out a reference (like MDN)

Following the spec is pretty straightforward

span and div are generic elements to use when there's not a specific element to use

(so they tend to be used a lot by JavaScript libraries when creating custom controls)

HTML5

#marketing

New HTML, CSS & JavaScript features that make the web a better platform for writing applications.

JAVASCRIPT

By many measures the world's most popular programming language.

» Released in 1995

» Renaissance started around 2006-2007

» Rise of Node.js from 2010-2011

» In the browser allows you to change HTML and the CSS applied **after** page load

» BTW, it's single threaded

WAT!?!

```
var input = ['10', '10', '10', '10', '10'];
var result = input.map(parseInt);

// 10, NaN, 2, 3, 4
console.log(result);
```

Gary Bernhardt's original lightning talk vs. Brendan Eich's JavaScript at 17

JAVASCRIPT LOOKS LIKE JAVA AND C#, BUT IT'S VERY DIFFERENT TO THOSE LANGUAGES

#marketing

More like Scheme than Java

No block scope (until ES6's const and let)

Dynamic types and type coercion

this is weird

You can do OO, but it's more fun and powerful to work with functions (IMO)

EXAMPLE

```
function f () {
  console.log(a.value, b.value, this.value);
}

var a = { value: 'a' };
var b = { value: 'b' };
var el = document.querySelector('input[type="text"]');

el.addEventListener('blur', f, false);
f.apply(a);
```

STRATEGY PATTERN VS. 1ST CLASS FUNCTIONS

```
var input = ['10', '10', '10', '10', '10'];
var result = input.map(parseInt);

// 10, NaN, 2, 3, 4
console.log(result);
```

STATIC ANALYSIS HELPS

JSLint / JSHint / ESLint

All have editor + command line support

JSHint + ESLint are configurable

ESLint lets you write plugins
(it's the new hotness)

UNDERSTANDING JAVASCRIPT HELPS MORE

JavaScript is flexible so it's easy to write Ruby-style, Java-style, whatever-style code.

Don't do that. Or if you really want to consider a language that compiles to JavaScript.

COMPILE TO JAVASCRIPT LANGUAGES

CoffeeScript / TypeScript / Scala.js /
ClojureScript / Dart / etc

JavaScript

ES3

```
function square (n) {
  return n * n;
}

var input = [1, 2, 3, 4, 5];
var result = [];
var i;

for (i = 0; i < input.length; i++) {
  result.push(square(input[i]));
}
```

ES5

```
var input = [1, 2, 3, 4, 5];

var result = input.map(function (n) {
  return n * n;
});
```

COFFEESCRIPT

```
input = [1, 2, 3, 4, 5]

result = input.map((n) -> n * n)
```

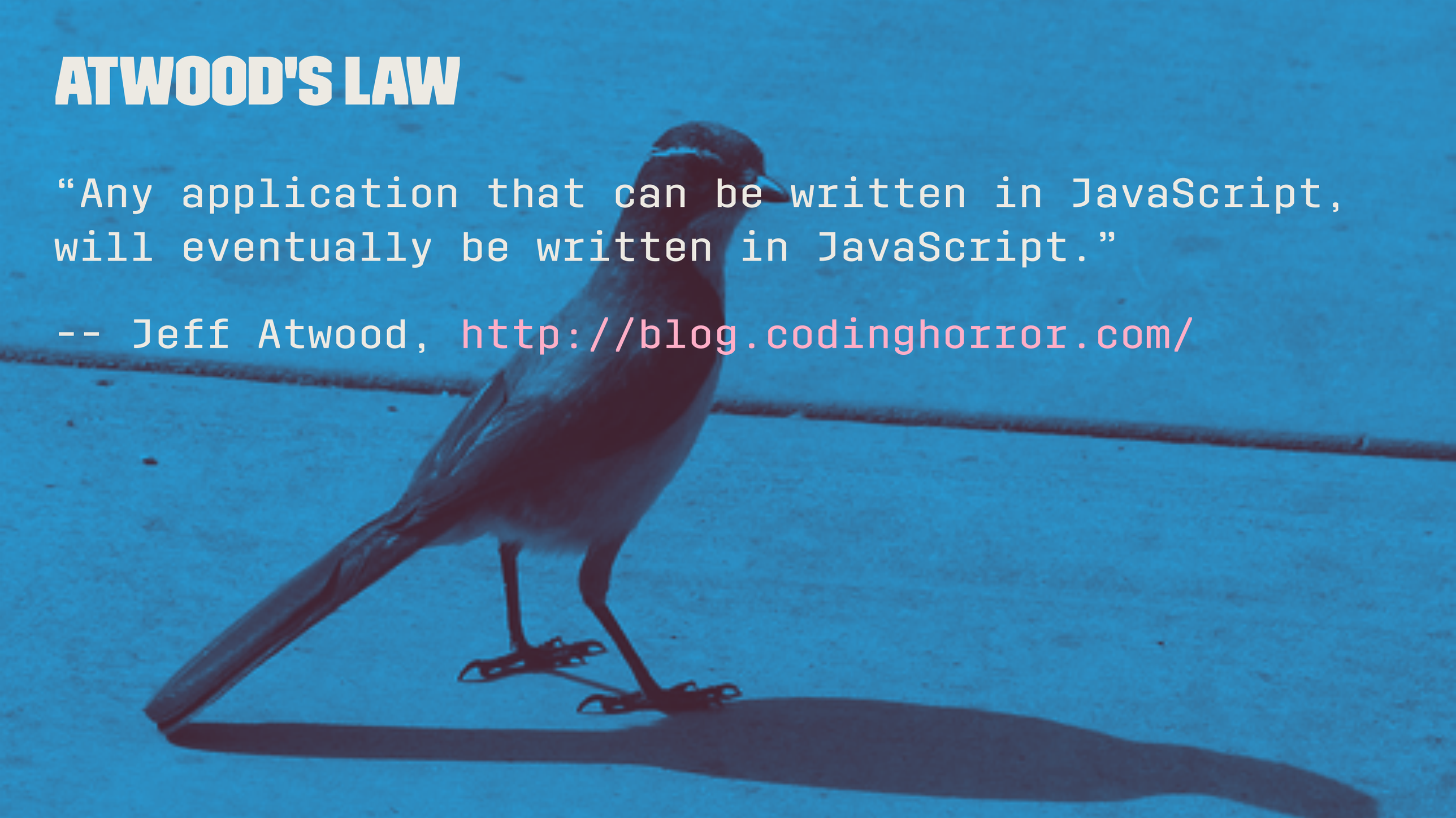
TYPESCRIPT

```
var input:number[] = [1, 2, 3, 4, 5];

var result = input.map((n: number) => {
  return n * n;
});
```

ES6

```
const input = [1, 2, 3, 4, 5];

const result = input.map(n => n * n);
```

ATWOOD'S LAW



“Any application that can be written in JavaScript, will eventually be written in JavaScript.”

-- Jeff Atwood, http://blog.codinghorror.com/

CSS

CSS is easy

That makes it really hard

ANY DAY NOW

I'm still waiting for the CSS renaissance.

```
body {
  font-family: Arial, Helvetica, sans-serif;
  background-color: #fff;
  color: #333;
}

p {
  font-style: italic;
}

#thebest {
  text-decoration: underline;
  text-transform: uppercase;
}

.snazzy {
  color: pink !important;
  font-weight: bold;
}
```

CSS SPECIFICITY

Ugh

```
!important > inline > id > class > element
```

```
<p id="thebest" class="snazzy" style="padding-left: 20px;">
  What is even happening?
</p>
```

CSS PRE + POST PROCESSORS

» Sass & Compass

» LESS

» Rework

» PostCSS

» cssnext

VARIABLES

```
$primary-color: #333;

body {
  color: $primary-color;
}
```

IMPORTS

```
@import "components/button";
@import "components/modal";

.foo {
  color: fuschia;
}
```

NESTING

```
.calendar {
  border: 1px solid #555;

  .calendar-day {
    border: 1px solid #555;
    color: blue;
    font-size: 14px;
  }
}
```

PREFIXES - COMPASS

```
.foo {
  @include background(linear-gradient(to bottom right, #333, #0c0));
}
```

=>

```
.foo {
  background: url('...');
  background: -webkit-gradient(linear, 0% 0%, 100% 100%, color-stop(0%, #333333), color-stop(100%, #00cc00));
  background: -moz-linear-gradient(top, #333333, #00cc00);
  background: -webkit-linear-gradient(top, #333333, #00cc00);
  background: linear-gradient(to bottom right, #333333, #00cc00);
}
```

PREFIXES - AUTOPREFIXER

```
.foo {
  background: linear-gradient(to bottom right, #333, #0c0);
}

=>

.foo {
  background: -webkit-linear-gradient(top left, #333, #0c0);
  background: linear-gradient(to bottom right, #333, #0c0);
}
```

And when the unprefixed form is universally supported autoprefixer will stop generating that rule

A FEW YEARS AGO

We thought CSS pre-processors were the hotness, but...

```
.nesting {
  .stuff {
    .is-easy {
      color: black;

      &.dangerous {
        color: blue;
      }
    }
  }
}
```

=>

```
.nesting .stuff .is-easy {
  color: black;
}

.nesting .stuff .is-easy.dangerous {
  color: blue;
}
```

“When writing CSS for big projects, it’s the stuff outside the braces that counts...”

-- Harry Roberts, http://csswizardry.com

NAMING CONVENTIONS

- SMACSS
- BEM

SMACSS

```
body {}                                 /* Base styles */

.layout-sidebar {}                      /* .layout-{name} */

.modal {}                               /* .{moduleName} */

.modal--header {}                       /* .{moduleName}--{subComponent} */

.button {}                              /* .{moduleName} */

.button-is-disabled {}                  /* .{moduleName}-is-{stateName} */

.button-default {}                      /* .{moduleName}-{subModule} */

.button-primary {}                      /* .{moduleName}-{subModule} */
```

IN HTML

```
<div class="modal">
  <!-- .modal--header is a subcomponent -->
  <div class="modal--header">Modal</div>
  <div class="modal--body">
    ...
  </div>
  <div class="modal--footer">
    <!-- .button-primary is a submodule -->
    <a href="#" class="button button-primary">Save</a>
    <a href="#" class="button button-cancel">Cancel</a>
  </div>
</div>
```

TL;DR

By following strong conventions and minimising specificity naming conventions keep your CSS more maintainable.

CSSLint is also a thing.

DOCUMENT OBJECT MODEL (DOM)

```
// find an element in the document
const el = document.getElementById('content');

// listen to events on that element
el.addEventListener('click', function (e) {
  e.preventDefault();
  // modify that element somehow (e.g. changing styling)
  el.classList.add('link-is-clicked');
}, false);
```

HTTP

Hypertext Transfer Protocol

1991 - v 0.9
1996 - v 1.0
1999 - v 1.1
2015 - v 2.0*

* but servers (but CDNs!)

REQUEST / RESPONSE

Clients make requests to servers

Servers provide responses to clients

REQUEST METHODS

GET / POST / DELETE / PUT / HEAD / OPTIONS / PATCH

A GET REQUEST

```
GET / HTTP/1.1
Host: www.summeroftech.co.nz
Connection: keep-alive
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_3) AppleWebKit/537.36
  (KHTML, like Gecko) Chrome/43.0.2357.81 Safari/537.36
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-NZ,en;q=0.8,en-US;q=0.6
```

AND ITS REPONSE

```
HTTP/1.1 200 OK
Server: nginx/1.4.6 (Ubuntu)
Date: Mon, 01 Jun 2015 18:22:18 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 8421
Connection: keep-alive
X-Powered-By: PHP/5.5.24
Vary: Accept-Encoding,Cookie
Cache-Control: max-age=3, must-revalidate
WP-Super-Cache: Served supercache file from PHP


...
```

XHR / AJAX, JSON & APIS

```
const byId = s => document.getElementById(s);
const template = Handlebars.compile(byId('template'));
const el = byId('users');

byId('button').addEventListener('click', function (e) {
  e.preventDefault();

  fetch('/users.json').then(function(response) {
    return response.json();
  }).then(function(json) {
    el.innerHTML = template(json);
  }).catch(function(e) {
    console.log('parsing failed', e);
  });
}, false);
```

BROWSERS

» Global stats are hard to make conclusions about

» Depends on markets
https://www.modern.ie/en-us/ie6countdown

» Get your own stats (e.g. Xero dropped IE 10 last week)

» User agent strings are easy to fake

» Feature detection (e.g. Modernizr)

It used to be much much worse.

JAVASCRIPT LIBRARIES & FRAMEWORKS

» jQuery for DOM normalisation

» Backbone

» Angular / Angular 2

» Ember

» React

Massive ecosystem, with lots of options.

Today it's less about protecting you from browser variation, more about how to structure applications.

HTTP SERVERS

Apache / IIS

Nginx / Node / Tomcat / etc

LET'S TALK ABOUT ASYNC

or

HOW THE HECK DOES NODE.JS SCALE WHEN JAVASCRIPT IS SINGLE THREADED!?!?!

It's used in production by PayPal, eBay, LinkedIn, Walmart, Yahoo!, Uber, Trello, New York Times, News Corp and many others.

SERVER GENERATED CONTENT

Maybe less of a thing today, but PHP, ASP, JSP, Ruby on Rails, etc. are all about generating HTML documents dynamically based on user input, session, database state, etc.

In Node land Express or hapi are pretty good options to start with.

FE OPS

So...

Transpiling, combining, linting, testing, measuring, etc. all that code needs tools.

There are lots of tools for server-side environments like Java and .NET.

Increasing maturity and complexity in the front-end has lead to new tools and new specialisation here.

NODE TOOLING

Node basically does evented I/O (reading/writing to/from file system, std in/out, etc.).

It's really easy to use to build little CLI tools.

Oh, and they're fast! Starting up Node is way cheaper than starting up a JVM.

Tools like Grunt, Gulp and Yeoman make task automation and project scaffolding pretty simple.

WEB VS. NATIVE

Today's hot drama (still!?).

Web means the best reach, best device support, but you have to work harder.

Loads of "native" apps use web views for parts of their app.

It's not a binary choice and web technology makes it easy to try something.

THE FUTURE

» Single Page Applications*

» Bigger, more ambitious apps

» Smaller, more fine-grained modules

» Mainstreaming of functional programming

» Isomorphic apps

» Cloud infrastructure / platform-as-a-service

* we're sort of there now, but there is still some resistance

FUNCTIONAL PROGRAMMING

Immutability

Referential transparency

Testability

Predictability

“The secret to building large apps is never build large apps. Break your applications into small pieces. Then, assemble those testable, bite-sized pieces into your big application.”

-- Justin Meyer, JavaScriptMVC

COMMUNITY

» Meetups

» Conferences

» Github

» npm

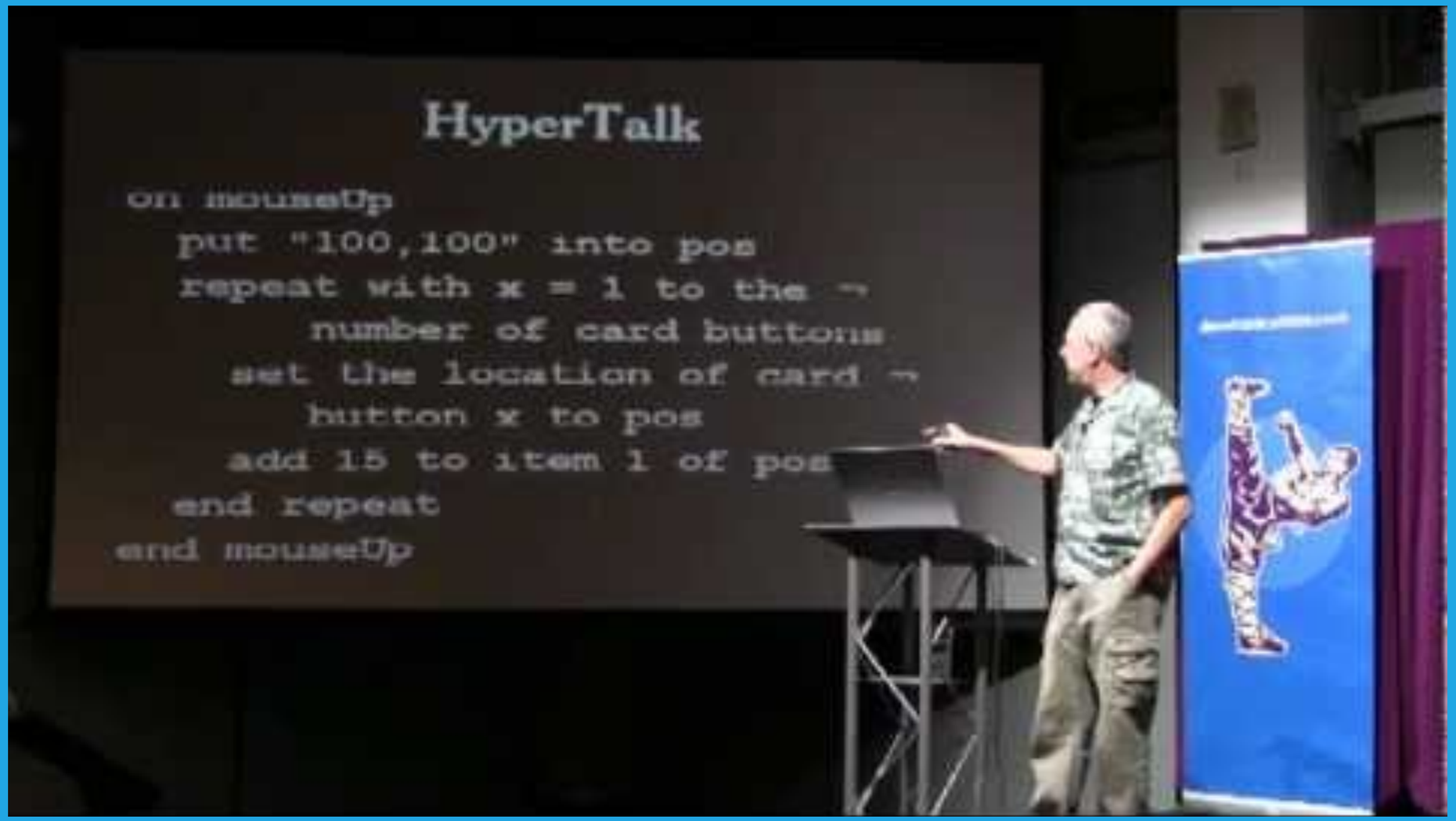
» social media

» podcasts

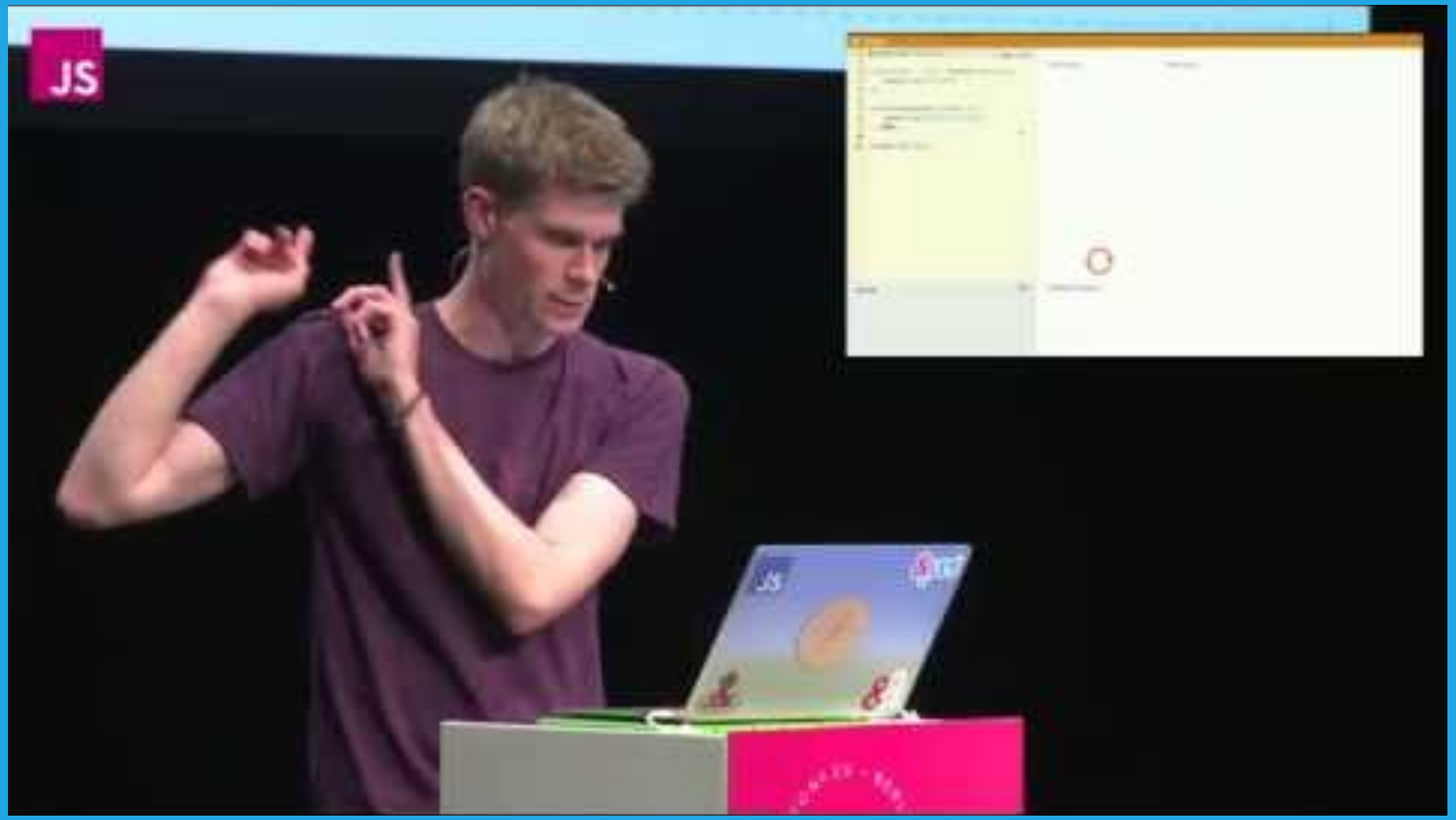
RESOURCES

Videos / Podcasts / Books / References / Tools

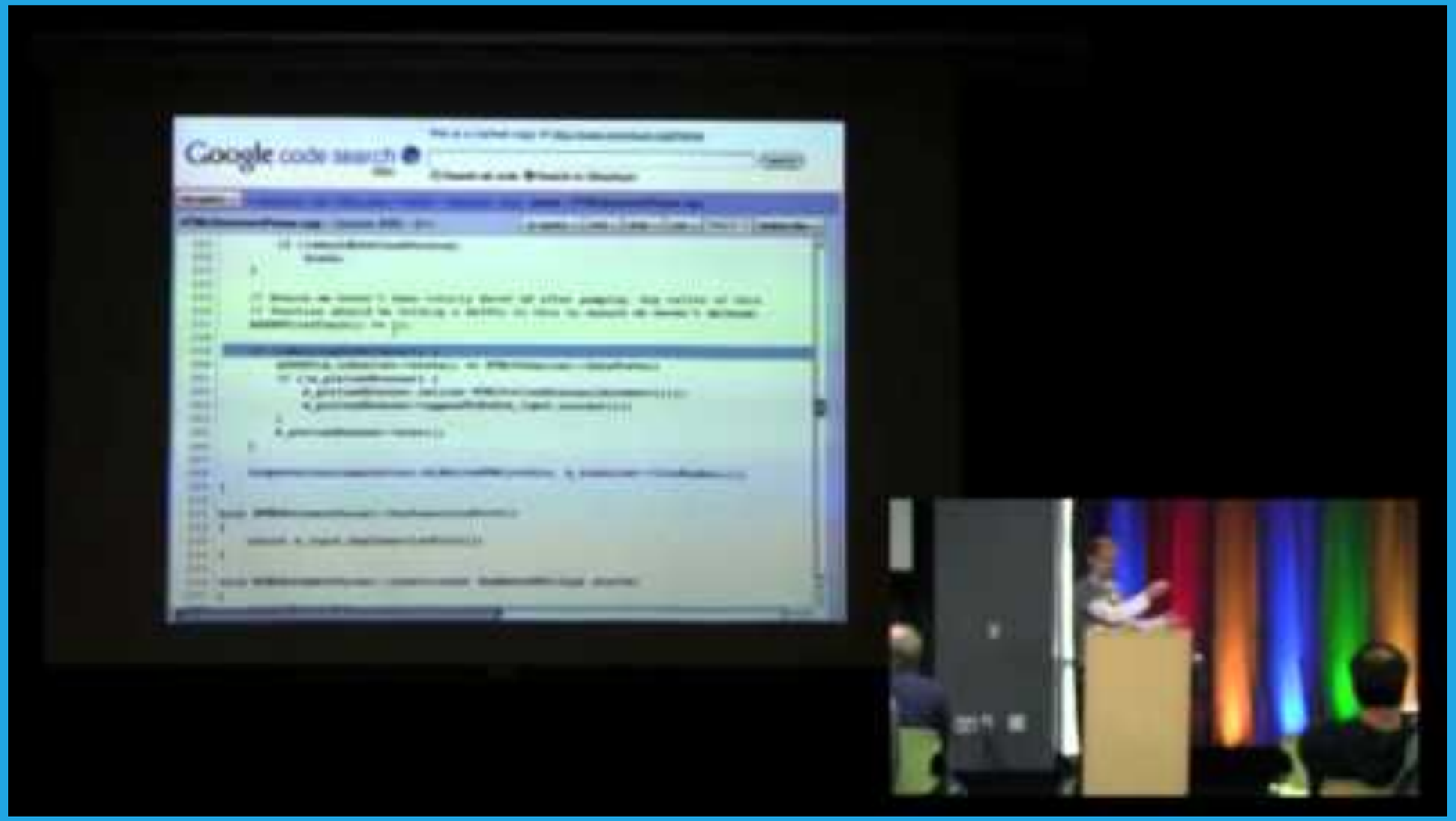
CROCKFORD ON JAVASCRIPT



WHAT THE HECK IS THE EVENT LOOP ANYWAY?



HOW WEB BROWSERS WORK



READ THESE

How Browsers Work
http://www.html5rocks.com/en/tutorials/internals/howbrowserswork/

A Baseline for Front-End [JS] Developers
http://rmurphey.com/blog/2015/03/23/a-baseline-for-front-end-developers-2015/

Caching Tutorial for Web Authors and Webmasters
https://www.mnot.net/cache_docs/

JavaScript Allongé
https://leanpub.com/javascript-allonge

LISTEN TO THESE

» Shop Talk Show - http://shoptalkshow.com/

» JavaScript Jabber - http://devchat.tv/js-jabber/

BOOKMARK THESE

» http://nodeschool.io

» http://caniuse.com

» http://devdocs.io

PLAY WITH THESE

» Dabblet - http://dabblet.com

» JSFiddle - https://jsfiddle.net

» CodePen - http://codepen.io

» ES6Fiddle - http://www.es6fiddle.net

» Babel - https://babeljs.io

» ESLint - http://eslint.org

TWEET AT ME

@wrumsby

BIRD PICUTRES

» profil by Olivier Bacquet

» A bird in UCD campus by asinensis

» On the edge by Ville Miettinen

» Seagull on One Leg by Walter Rumsby