

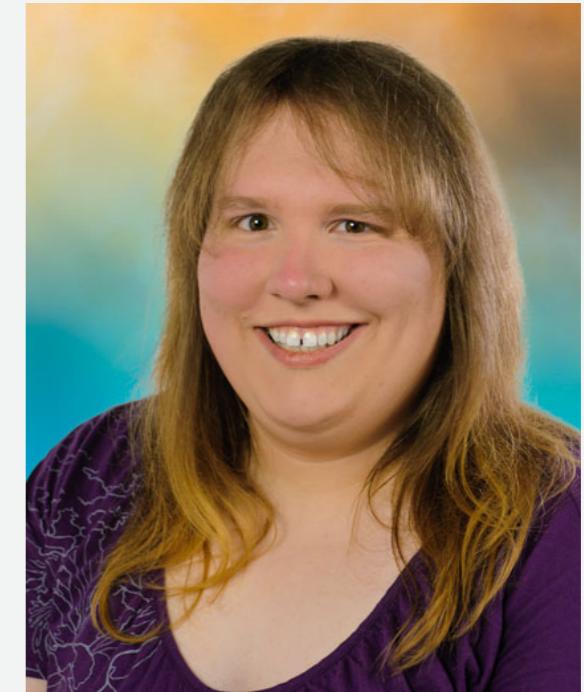
Modern JavaScript and PhoneGap

Kerri Shotts • @kerrishotts

<https://kerrishotts.github.io/pgday/>

About Me

- Used PhoneGap for over six years
- Authored Five books about PhoneGap
- Apache Cordova committer
- One of many moderators at:
 - [Cordova Google Group](#)
 - [PhoneGap Adobe Forums](#)



2009



President of the
United States of America



iPhone 3GS

By nvog86 - Own work, CC BY-SA 3.0, <https://commons.wikimedia.org/w/index.php?curid=20238837>

iOS 3
ES3 (1999)

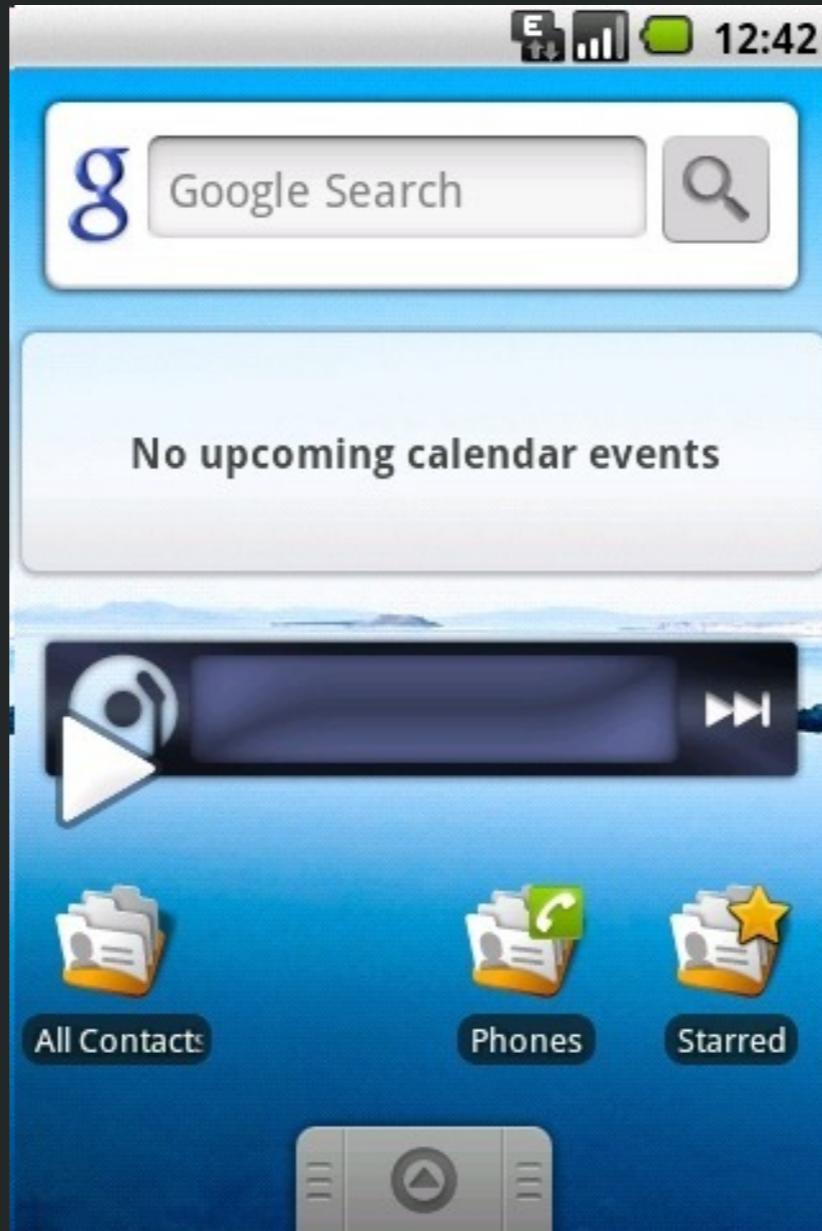




HTC Hero

By AlvinPing at English Wikipedia, CC BY-SA 3.0, <https://commons.wikimedia.org/w/index.php?curid=8406455>

Android 1.5 ES3 (1999)





PhoneGap is born!



ECMAScript 5

JS

By Ramaksoud2000 via Chris Williams - Wikipedia via GitHub logo.js, Public Domain, <https://commons.wikimedia.org/w/index.php?curid=18434372>

ES5

- The version we all know and love (~ish?)
 - 'use strict';
 - map , reduce , Object.create , Object.freeze , trim !
 - JSON parsing
- Supported by all modern mobile web views¹
 - iOS 6+, IE 10+, Edge (forever), Android 4.4+
 - But not in 2009 – *no one supported it*

¹ <http://caniuse.com/#feat=es5>

2012



Still President of the
United States of America

A black and white photograph of an iPhone 5 smartphone. The phone is oriented vertically, showing its front side. The screen is dark, and the body is a dark color. At the top edge, there is a camera lens and a microphone hole. The bottom edge features a home button. The text "iPhone 5" is overlaid in large, white, sans-serif letters across the center of the phone's body.

iPhone 5

iOS 6

93% support
for ES5

<http://kangax.github.io/compat-table/es5/>



By Self-taken screenshot from an iPhone 5., <https://en.wikipedia.org/w/index.php?curid=37500638>

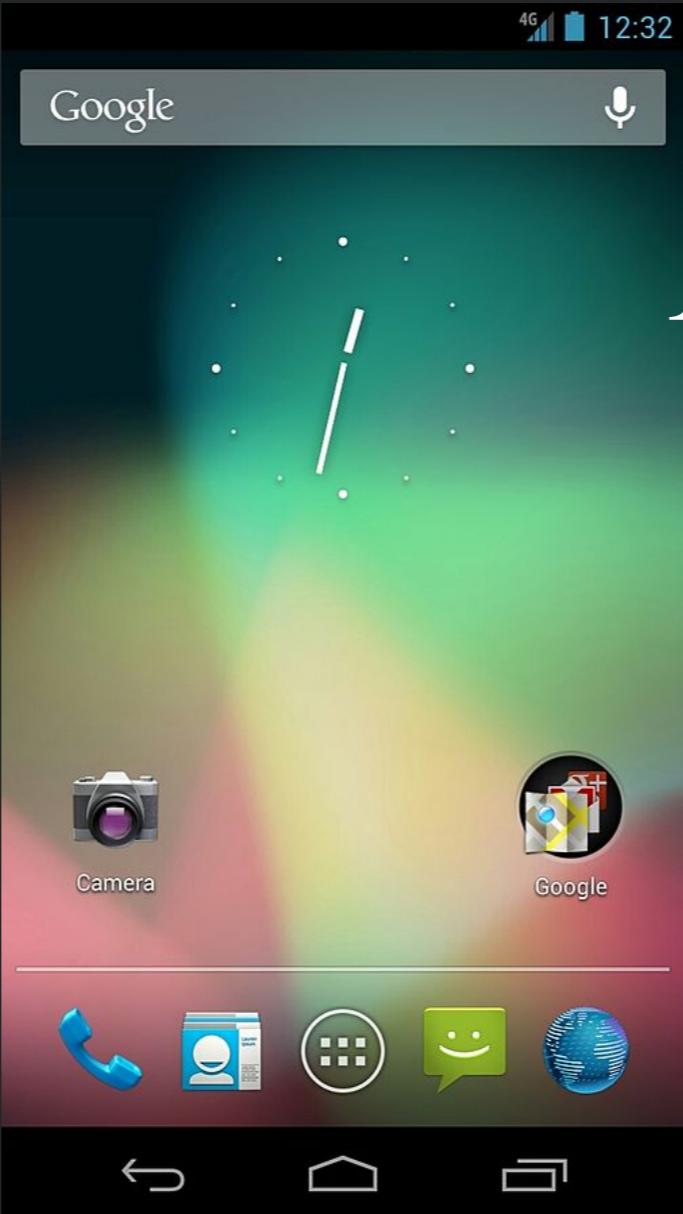


Samsung Galaxy S3

Android 4.1 Jelly Bean

85% support
for ES5

<http://kangax.github.io/compat-table/es5/>



2015



Second term 😊



ES2015 (née ES6)



iOS 9

54% ES2015

Android 5.1

25% ES2015

ES2015 (ES6)

It had been quite a while, so the list is long...

Block-scoped <code>let</code> & <code>const</code>	Destructuring and named parms
Default parameters	Rest and Spread operator (<code>...</code>)
<code>for...of</code> loops and Iterators	Arrow functions (<code>=></code>)
Template strings & interpolation	Improved literals (object, <code>0b10</code>)
Generators (<code>*</code> / <code>yield</code>)	Symbols, Maps & Sets, Promises
<code>class</code> syntactic sugar & <code>super</code>	Modules (<code>import</code> , <code>export</code>)

1: <https://github.com/lukehoban/es6features#readme>; **not** a complete representation of *all* features

2016



We'll miss you! 😢

Brexit

Wait... what?

iOS 10

100% ES2015 *

* Except module implementation; source: <http://kangax.github.io/compat-table/es6/>

Android (Chrome 50)

92% ES2015

* Except module implementation; source: <http://kangax.github.io/compat-table/es6/>

ES2016 / ES7

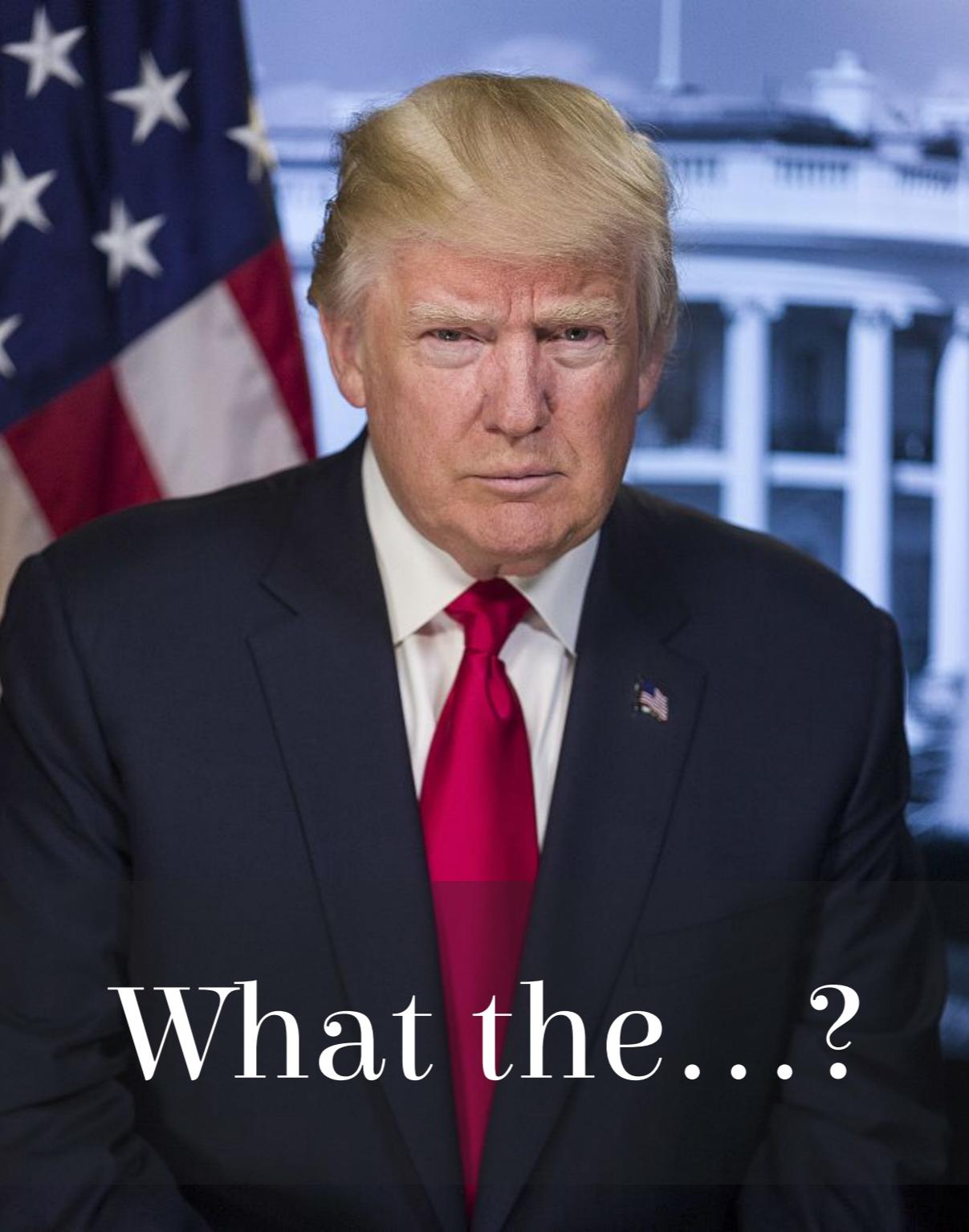
ES2016 (ES7)

Fewer features, but still important:

Exponent (**)

Array.prototype.includes()

2017



What the...?

ES2017

Ah, that's better...

ES2017

Finally – proper string padding!

async / await

String padding 😊

Shared memory

Atomics

ES2018 and beyond

<https://esdiscuss.org>

A quick intro to ES2015+

Block. Scoped. Variables. Finally.

```
let i = 50;
for (let i = 0; i < 100; i++) {
  console.log(i); // 0...99
}
console.log(i); // 50
```

Constant references too:

```
const SECONDS_IN_A_MINUTE = 60;
const CREATE_ACTION = "action";
```

Constants aren't constant

Only the reference is constant:

```
const DEFAULT_OPTIONS = {  
    quality: 50  
};
```

```
DEFAULT_OPTIONS.quality = 100;  
console.log(DEFAULT_OPTIONS.quality); // 100
```

Dang it, *this*!

```
1 var app = {  
2   text: "Hello, PG Day Attendees!",  
3   sayHi: function() { alert(this.text); },  
4   start: function() {  
5     document.querySelector("#clickme")  
6       .addEventListener("click", this.sayHi, false);  
7   }  
8 }  
9  
10 app.start();
```

A close-up shot of a Star Trek character, likely Captain Jean-Luc Picard, wearing a red uniform with a black collar. He has his eyes closed and is holding his head in his hands, appearing distressed or exhausted. The background shows the interior of a starship with a window showing a yellowish planet.

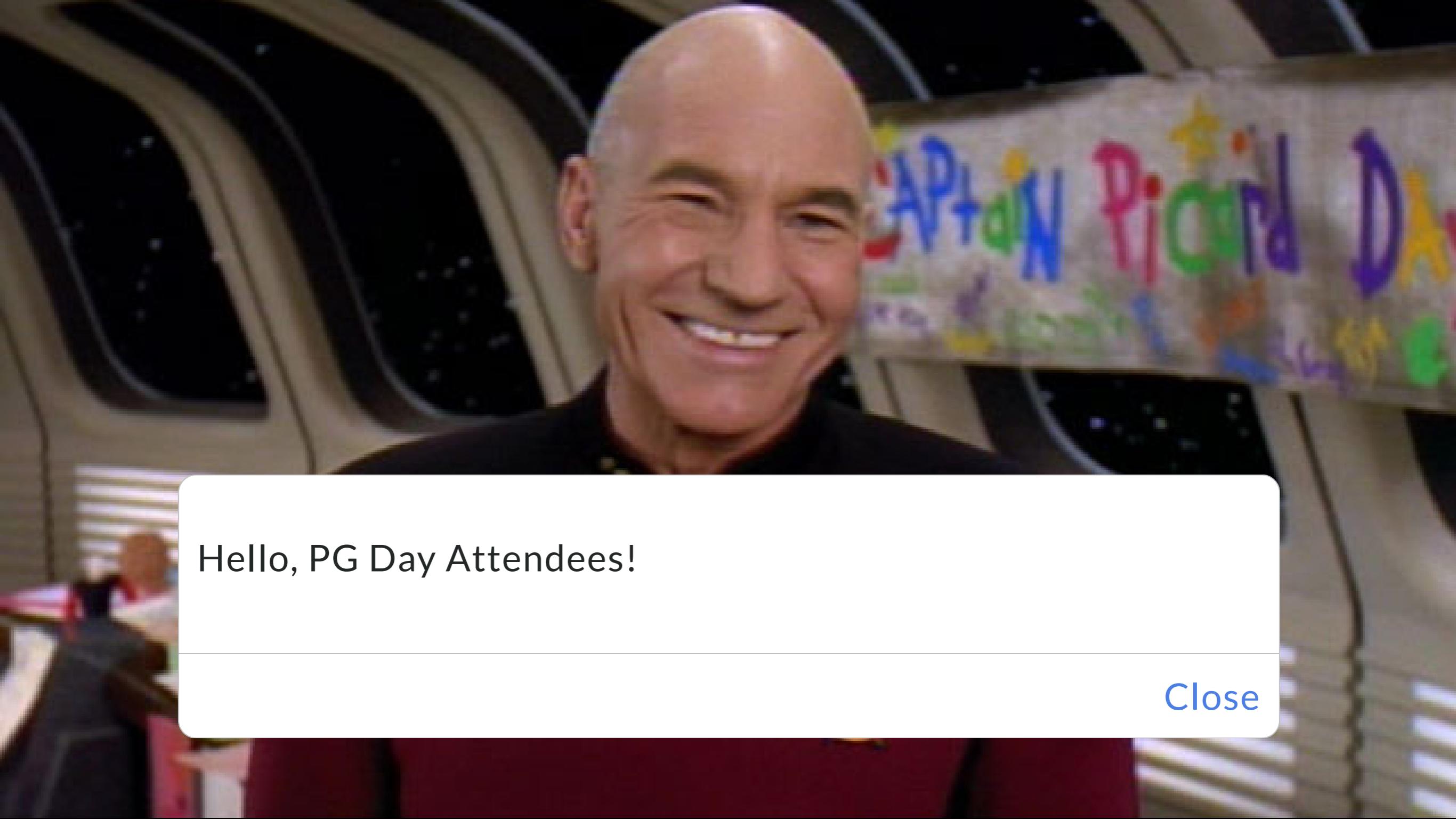
undefined

[Close](#)

Arrow functions

```
1 class App {  
2     constructor() { this.text = "Hello, PG Day Attendees!" }  
3     sayHi() { alert(this.text); }  
4     start() {  
5         document.querySelector("#clickme")  
6             .addEventListener("click", () => this.sayHi(), false);  
7     }  
8 }  
9 const app = new App();  
10 app.start();
```

Line 6 ES5 equivalent: .addEventListener("click", (function() { this.sayHi(); }).bind(this), false)

A close-up photograph of a man with a shaved head, smiling broadly. He is wearing a dark, button-down shirt. In the background, there are large, metallic, curved structures, possibly part of a ship's interior. A colorful banner with the words "Captain Picard" in various fonts and colors is visible behind him.

Hello, PG Day Attendees!

[Close](#)

Arrow function quirks

No parameters? Use parentheses:

```
[1, 2, 3].forEach(() => console.log("hi"));
```

One parameter? Convention is no parentheses:

```
[1, 2, 3].map(i => i * 2);
```

Multiple parameters? Use parentheses:

```
[1, 2, 3].map((i, idx) => i * idx);
```

Arrow function returns

Single line arrow functions should use implicit return:

```
[1, 2, 3].map(i => i * 2);
```

Multi-line arrow functions should use blocks:

```
[1, 2, 3].map(i => {
  let x = Math.floor(Math.random() * 100);
  return i * x;
});
```

Arrow function ambiguity

But what if we return an object? This won't work:

```
[1, 2, 3].map(i => {i: i * 2}); // is equivalent to  
[1, 2, 3].map(i => {  
    i: // obviously not what  
    i * 2; // we want :-(  
});
```

Instead, wrap the return in parentheses:

```
[1, 2, 3].map(i => ({i: i * 2}));
```

Template Strings

Single quotes, double quotes, and now... backticks!? Just after I remapped underscore to the backtick, no less!

```
function sayHello(name) {  
  return `Hello, ${name}!`;  
}  
  
sayHello("World"); // Hello, World!
```

Template Strings

Complex expressions (*use with care*):

```
function sayComplexHello(name) {  
  return `Hello, ${name ? name : "Jane Doe"}!`;  
}
```

```
sayComplexHello("Alex");      // Hello, Alex!  
sayComplexHello();           // Hello, Jane Doe!
```

Promises, Promises

More concise with arrow functions:

```
function getPos(options) {
  return new Promise((resolve, reject) => {
    navigator.geolocation.getCurrentPosition(
      resolve, reject, options);
  });
}
```

Promises, Promises, Promises

Easier-to-read chaining:

```
getPos.then(pos => {
  console.log(JSON.stringify(pos));
}).catch(err => {
  console.error(err);
});
```

Destructuring

Do be careful with how far you nest, though.

```
function gotPos(data) {  
  let {timestamp, coords:{latitude, longitude}} = data;  
  console.log(` ${latitude}, ${longitude}@${timestamp}`);  
}  
function gotError(err) {  
  console.log(`Error ${err.code}: ${err.message}`);  
}  
getPos().then(gotPos).catch(gotError);
```

Destructuring

Not just for objects; arrays work too:

```
function divide(a, b) {  
  if (b === 0) {  
    return [undefined, new Error("Divide by zero")];  
  } else {  
    return [a / b, null];  
  }  
}  
  
let [results, error] = someFunction();
```

async / await (ES2017)

```
async function start() {  
  try {  
    let pos = await getPos(),  
        {coords:{latitude, longitude}} = pos;  
    console.log(`${latitude}, ${longitude}`);  
  } catch(err) {  
    console.log(`Error ${err.code}: ${err.message}`);  
  }  
}
```

Note: `async` poisons the call tree; all callers must also be `async` or treat the response like a `promise`.

Array-like conversion

If only I had a € for every time I've written:

```
var elList = document.querySelectorAll("a"),
elArr = [].slice.call(elList, 0);
```

Array-like conversion

Using the standard library:

```
let elArr = Array.from(document.querySelectorAll("a"));
```

Using the spread operator:

```
let elArr = [...document.querySelectorAll("a")];
```

Rest

Easy variable arguments:

```
function sum(start = 0, ...nums) {  
  return nums.reduce((acc, val) => acc + val, start);  
}  
console.log(sum(1, 5, 10, 99)); /* 115 */
```

Named Parameters & Defaults

```
function getPicture({quality = 50, width = 512,  
                    height = 512} = {}) {  
  return new Promise((resolve, reject) => {  
    navigator.camera.getPicture(resolve, reject, {  
      allowEdit: false,  
      correctOrientation: true, quality,  
      targetWidth: width, targetHeight: height,  
    });
  });
}
```

Named Parameters & Defaults

```
// use all the defaults  
getPicture().then(/*...*/);
```

```
// specify only quality  
getPicture({quality:75}).then(/*...*/);
```

```
// specify only height & width  
getPicture({height: 1024, width: 1024}).then(/*...*/)
```

Modules

Static Analysis, FTW!

math.js:

```
export function add(a, b) { return a+b; }
```

index.js:

```
import {add} from "./math.js";
console.log(add(4, 3));      // 7
```

Cool! Where can I use it?

Native support is a moving target

OS	ES2015	ES2016	ES2017
Android (Chrome)	97% (51+)	100% (55+)	53% (56+)
Windows (Edge 15)	100%	100%	39%
Windows (Edge 14)	93%	-	-
iOS 10.3	100%	100%	98%
iOS 10	100%	61%	42%
iOS 9	54%	-	-

Sources: [ES2015](#), [ES2016+](#)

The Rise of the Transpilers

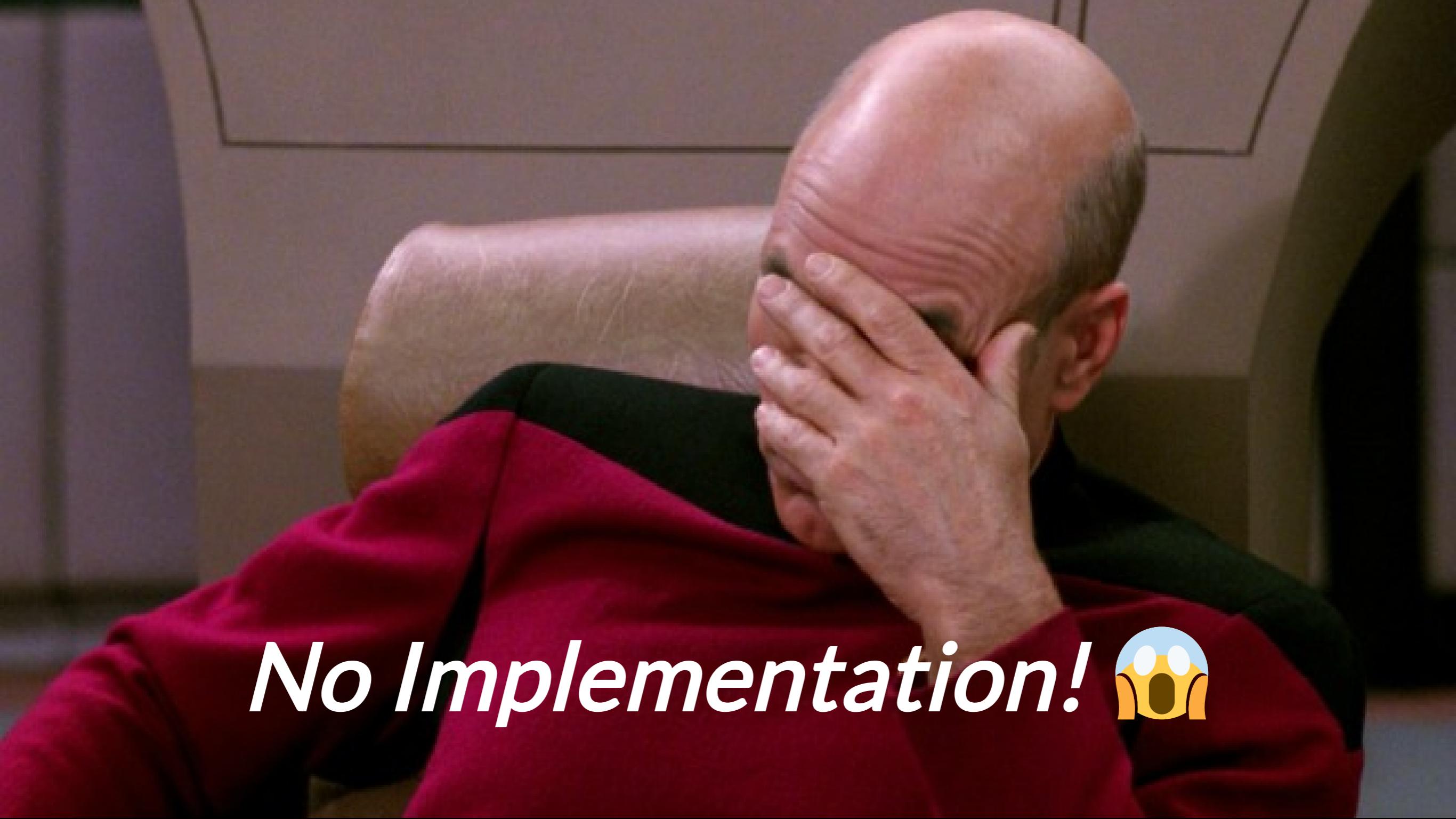
These can all transpile ES2015+* (feature support may vary)

- [Babel](#) (née [es6to5](#))
- [TypeScript](#)
- [Buble](#) **
- [Traceur](#)

* **Note:** Not every ES2015+ feature can be transpiled effectively (if at all), such as proxies, shared memory, atomics, built-in subclassing, and tail call elimination. Also, most transpilers need [core-js](#) to polyfill the standard library.

** Doesn't attempt to transform non-performant or non-trivial ES6 features; *also very young*

*Remember module
support?*



No Implementation! 😱

I lie...

Browsers have *finally* started shipping implementations:

- Available now:
 - Safari 10.1, iOS 10.3
- Behind a flag:
 - Edge 15
 - Firefox 54
 - Chrome and Android WebView 60

Source: <http://caniuse.com/#feat=es6-module>

Native Modules

js/index.js:

```
import Game from "./Game.js";
const game = new Game();
game.start();
```

index.html:

```
<script type="module" src="./js/index.js"></script>
```

There's always a catch

- No “bare” import !
 - Must include the path
 - Must include the extension
 - No aliases
- iOS support does not work in PhoneGap / Cordova
 - Not supported at all by UIWebView
 - WKWebView without a local webserver throws CORS errors
 - WKWebView using the local webserver throws 403 forbidden

A color photograph of an elderly man with white hair and a warm smile. He is wearing a dark red turtleneck sweater over a black collared shirt. He is holding two large, dark-colored wine bottles, one in each hand, with both arms raised slightly. The background is a soft-focus indoor setting with warm lighting.

But we can fix that...

Module support using Bundling



Dependency management & import / export (and CommonJS, AMD, etc.) support

- [Webpack](#)
- [JSPM](#)
- [Browserify](#)

You can do more than just bundling:

- Convert SASS to CSS, lint, copy assets, compress assets, etc.

Execution Options

- Manual
 - Just run each tool's CLI... *every time...*
 - Um... no.
 - Error prone – you might forget!
- Automatic
 - Task runners (`gulp` , `grunt` , etc.)
 - `npm` scripts
 - Plugin or Project hooks

Automating with npm scripts

- Pick your bundler and transpiler
 - Bundler: Webpack 2
 - Transpilers: TypeScript & Babel (showing both configs)
- Install Webpack & Transpiler
- Configure Webpack & Transpiler
- Add scripts to package.json

Install Webpack

Easy (assuming package.json exists):

```
$ | npm install --save-dev webpack
```

Install Transpiler

TypeScript:

```
$ | npm install --save-dev ts-loader typescript core-js
```

Babel:

```
$ | npm install --save-dev babel-loader babel-core  
babel-polyfill babel-preset-es2015  
babel-preset-es2016 babel-preset-es2017  
babel-plugin-transform-runtime
```

Note: core-js is a standard library polyfill; depending on your feature use and targets you may not need it.

Configure Transpiler

```
// tsconfig.json                                // .babelrc
{ "compilerOptions": {                         { "presets": [
    "allowJs": true,                           ["es2015", {
    "target": "es5",                            "loose": true,
    "module": "es2015", // DCR                 "modules": false // DCR
    "lib": ["es2015", ...]                      }] ,
    "inlineSourceMap": true                     "es2016", "es2017"
},                                              ],
    "include":                               "plugins": ["transform-runtime"]
    [ "www.src/es/**/*" ]                      }
}
```

* Don't forget to import `core-js (ts)/ babel-polyfill` in your `index.?s` if targeting older runtimes. DCR = tree shaking

webpack.config.js

```
module.exports = {  
  devtool: "inline-source-map",  
  context: path.resolve(__dirname, "www.src"),  
  entry: { app: ["./es/index.js"] },  
  output: {  
    filename: "bundle.js",  
    path: path.resolve(__dirname, "www", "js")  
  },  
  module: { /*...*/ }  
}
```

webpack.config.js

```
module: {  
  rules: [ {  
    test: /\.([t|j]sx?)$/,  
    exclude: /node_modules/,  
    loader: "ts-loader",           // or babel-loader  
    options: { entryFileIsJs: true } // excl if babel  
  } /*, ... other rules as needed */  
]  
}
```

npm Scripts

```
"scripts": {  
  "sim:ios": "webpack -d && cordova emulate ios",  
  "run:ios": "webpack -d && cordova run ios",  
  "build:ios": "webpack -d && cordova build ios",  
  "build:ios:rel": "webpack -p && cordova build ios --release"  
}
```

-d : debug

-p : production (minifies as well)

```
$ | npm run build:ios
```

Code Splitting

```
module.exports = {
  entry: {
    app: ["./es/index.js"],
    vendor: ["core-js"]
  }, /*...*/
  module: { /*...*/ },
  plugins: [
    new webpack.optimize.CommonsChunkPlugin({
      name: "vendor", filename: "vendor.js"})
  ]
}
```

Automating with Plugin Hooks

`cordova-plugin-webpack-transpiler` transforms at `prepare` -time.

```
$ | cordova plugin add cordova-plugin-webpack-transpiler  
| --variable CONFIG=typescript|babel|...
```

Executes when `prepare` is called: `build` , `run` , `emulate` , etc.

```
$ | cordova build ios # debug mode  
| $ cordova build ios --release # production mode  
| $ cordova run ios --notransform # skip transform/bundling
```

Automating with Templates

Template	Author	Bundler	Transpiler	Frameworks	Automation
cordova-template-webpack-ts-scss	Me	Webpack	TypeScript	Vanilla	cordova
cordova-template-webpack-babel-scss	Me	Webpack	Babel	Vanilla	cordova
cordova-template-framework7-vue-webpack	centrual	Webpack	Babel	Vue, F7	cordova
phonegap-template-react-hot-loader	devgeeks	Webpack	Babel	React	npm
phonegap-vueify	lemaur	Browserify	Babel	Vue	npm

Automation: “cordova” = Cordova hooks; “npm” = npm scripts

Video

Linting

eslint works just fine with ES2015! (tslint for Typescript)

```
$ | npm install --save-dev eslint
```

package.json:

```
"scripts": {  
  "lint": "eslint www.src/es"  
}
```

```
$ | npm run lint      # or, write a plugin/project-level hook!
```

Tests and Coverage

```
$ npm install --save-dev chai mocha mocha-webkit  
webpack-node-externals source-map-support  
istanbul-instrumentor-loader cross-env
```

- Create mocha-webpack.opts
- Create .nycrc
- Create webpack config for testing and coverage
- Add npm scripts

Based on: <https://github.com/istanbuljs/nyc/issues/176#issuecomment-286738545>

mocha-webpack.opts

```
--webpack-config ./webpack.config-test.js  
--require source-map-support/register  
--full-trace  
test/*.js
```

.nycrc

```
{  
  "extension": [".js", ".jsx", ".ts", ".tsx"],  
  "reporter": ["html", "text-summary"],  
  "cache": true,  
  "sourceMap": false,  
  "instrument": false  
}
```

webpack.config-test.js

```
var path = require("path");
var coverage = process.env.NODE_ENV === "cover";
module.exports = {
  context: path.resolve("test"),
  target: "node",
  externals: require("webpack-node-externals")(),
  devtool: "inline-source-map",
  resolve: {
    extensions: [".js", ".jsx", ".ts", ".tsx"],
    modules: ["node_modules"]
  }, module: { /*...*/ }
};
```

webpack.config-test.js

```
module: {  
  rules: [{  
    test: /\.j|ts[x]?/,  
    exclude: /node_modules/,  
    rules: [{ test: () => coverage,  
              enforce: "post",  
              loader: "istanbul-instrumenter-loader?esModules"  
            }, { loader: "ts-loader", // or babel-loader  
                  options: { entryFileIsJs: true } // not if babel  
                }]  
  }]  
}
```

Add npm scripts

Add scripts to package.json:scripts *

```
"test": "mocha-webpack",  
"cover": "cross-env NODE_ENV=cover nyc npm test",
```

And then:

```
$ | npm test  
$ | npm run cover
```

A close-up portrait of Captain Jean-Luc Picard from Star Trek: The Next Generation. He is wearing his iconic red starfleet uniform with a black collar and a white triangular insignia on his chest. He has a serious, contemplative expression, looking slightly off-camera to the left. His hair is thinning and grey. The background is a blurred interior of a starship.

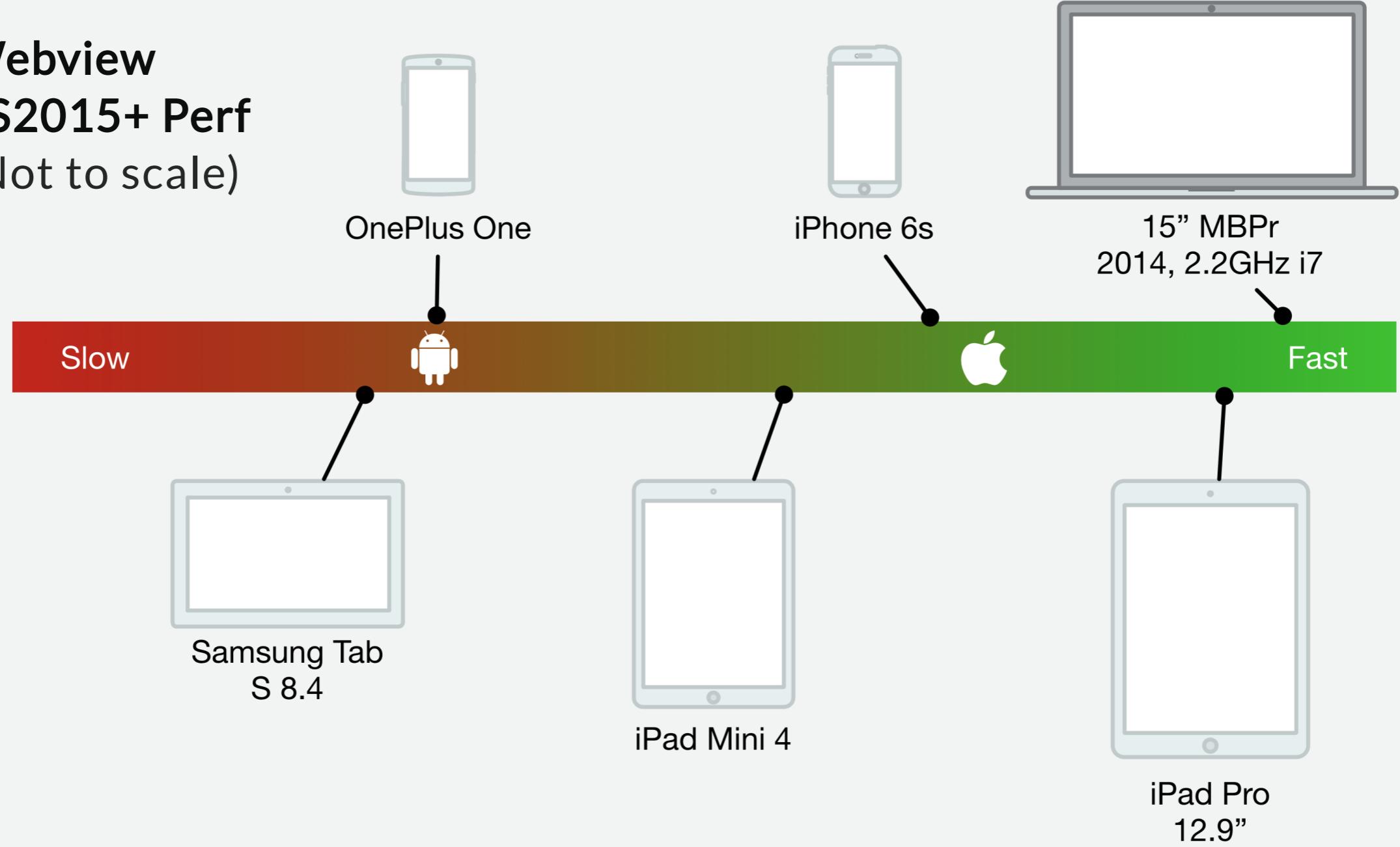
Reality Check...

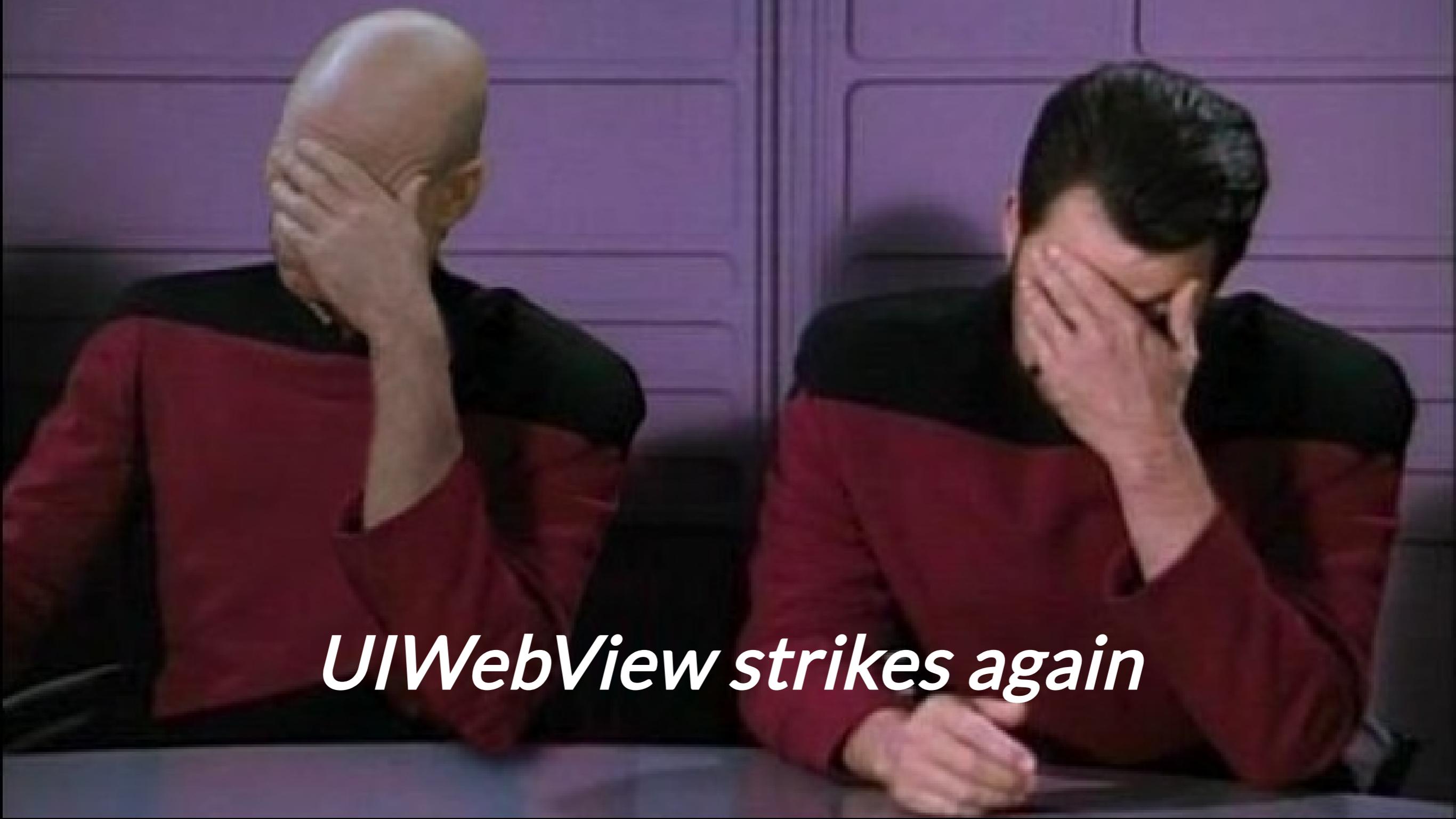
Reality Check...

- ES2015+ is NOT a performance optimization
 - See <https://kpdecker.github.io/six-speed/> (as of 2017-01-04)

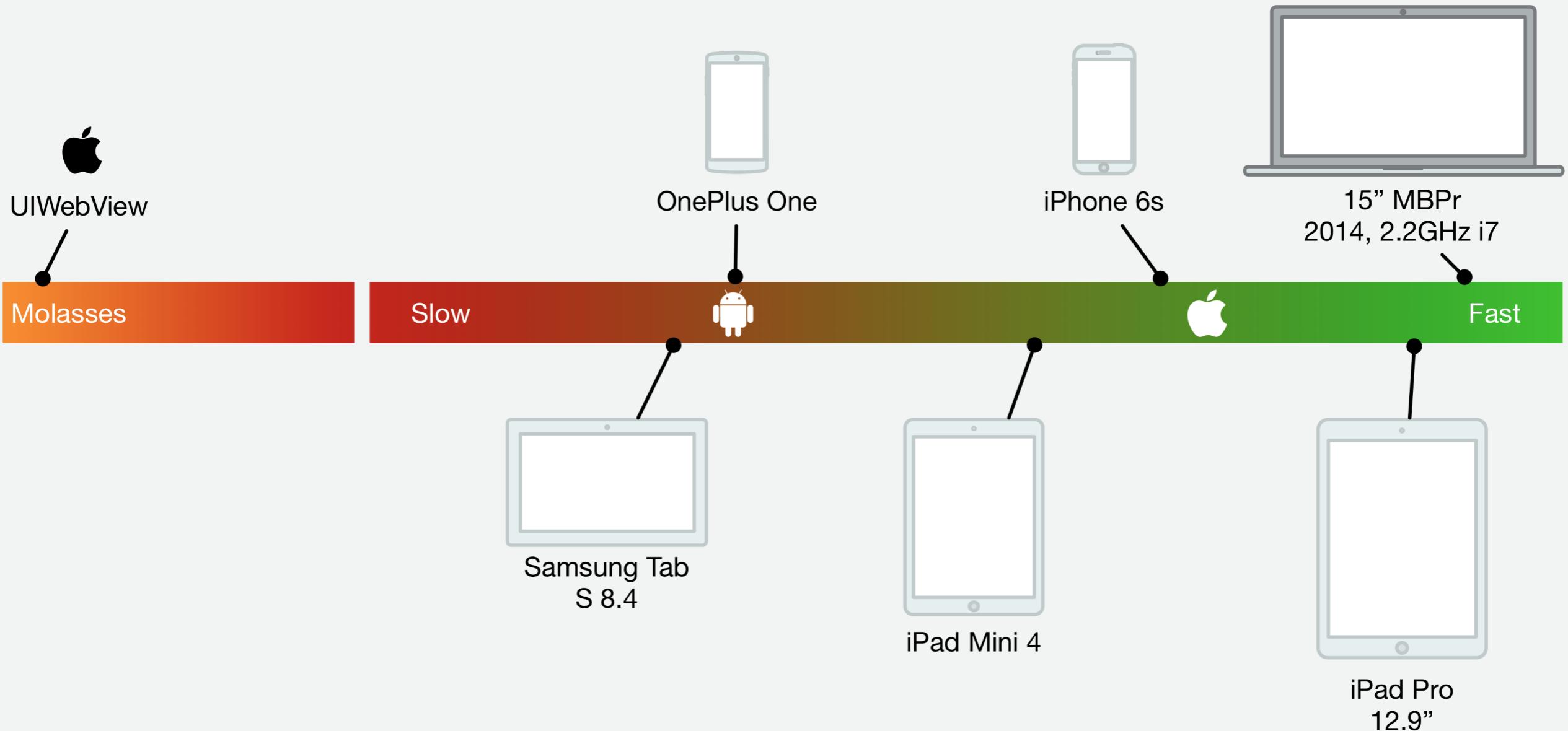
		node 6.9.3 7.3.0		chrome 55 49		firefox 50 53		edge 14.14393 15.14959		safari 10	webkit 604.1.2+
arrow tests	babel	1.2x slower	Identical	Identical	Identical	Identical	Identical	Identical	1.2x slower	Identical	Identical
	typescript	Identical	Identical	Identical	Identical	Identical	Identical	1.2x slower	1.2x faster	Identical	1.2x slower
	es6	Identical	Identical	Identical	Identical	Identical	Identical	Identical	1.2x faster	Identical	Identical
arrow-args tests	babel	Identical	Identical	Identical	Identical	Identical	Identical	Identical	1.5x slower	Identical	Identical
	typescript	Incorrect ⓘ	Incorrect ⓘ	Incorrect ⓘ	Incorrect ⓘ	Incorrect ⓘ	Incorrect ⓘ	Incorrect ⓘ	Incorrect ⓘ	Incorrect ⓘ	Incorrect ⓘ
	es6	Identical	Identical	Identical	Identical	Identical	Identical	Identical	Identical	69x slower	71x slower
arrow-declare tests	babel	1.6x slower	1.5x slower	1.4x slower	1.4x slower	30x slower	23x slower	1.3x slower	1.7x slower	Identical	Identical
	typescript	1.4x slower	1.3x slower	1.3x slower	1.4x slower	15x slower	23x slower	1.5x slower	1.4x slower	Identical	Identical
	es6	1.4x slower	1.3x slower	1.3x slower	1.3x slower	38x slower	56x slower	1.2x slower	1.5x slower	Identical	Identical
bindings	babel	Identical	Identical	Identical	Identical	Identical	Identical	Identical	1.4x slower	Identical	1.2x slower

Webview ES2015+ Perf (Not to scale)



A man with dark hair, wearing a red and black striped sweater over a green collared shirt, sits at a table. He has his hands clasped together near his face, appearing to be in distress or despair. The background is a plain, light-colored wall.

UIWebView strikes again



UIWebView's performance is highly dependent upon language features in use; this is a worst-case representation.

More Reality Checks

- Build step
- Debugging can be “fun”
- Some of the syntax can be a little *sharp* – handle with care

Euuuuggghhh!!!

Way to crush my dreams!

Not really...

- Micro-benchmarks aren't the entire story
 - Engines are continually improving
- Actual performance deltas are highly variable
 - Depends on platform and the language features in use
- Lots of benefits:
 - Expressive and concise
 - Less boilerplate
 - padStart and padEnd ! 😊

Tips

- ES5 still works
- Use ES2015+ as needed and when you're ready
- `var` is alive and well
 - Use where performance is critical (e.g., tight nested loops)
- Declare `const / let` at the top of each scope
 - Benefits Chrome's optimizer
 - Good practice anyway
- Arrow functions aren't drop-in

Tips

- Minify & remove dead code for release builds
 - Reduces bundle sizes and startup time
- Split code bundles
 - Vendor code can be separately bundled
 - Easier to blacklist in debuggers
- Use WKWebView on iOS for best performance

Resources

- <https://esdiscuss.org>
- [ECMAScript 2015 Support in Mozilla](#)
- [ES2015 Compatibility Table](#)
- [2ality - JavaScript and more](#)
- [Can I Use](#)
- [WebKit Feature Status](#)
- [Chrome Platform Status](#)

A close-up, profile view of a man's face, looking towards the left. He has a warm, reddish-brown complexion and is smiling broadly, showing his teeth. His eyes are partially closed in a joyful expression. The background is blurred, suggesting an outdoor setting with warm sunlight.

That's all, folks!

Thanks!

@kerrishotts

<https://kerrishotts.github.io/pgday/>

This slide intentionally left blank