



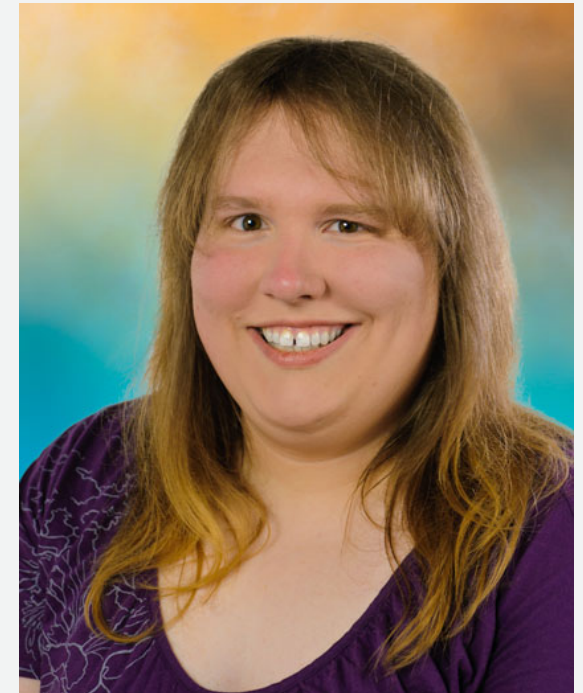
Modern JavaScript and PhoneGap

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](#)
- Just started at Adobe



2009



iPhone 3GS



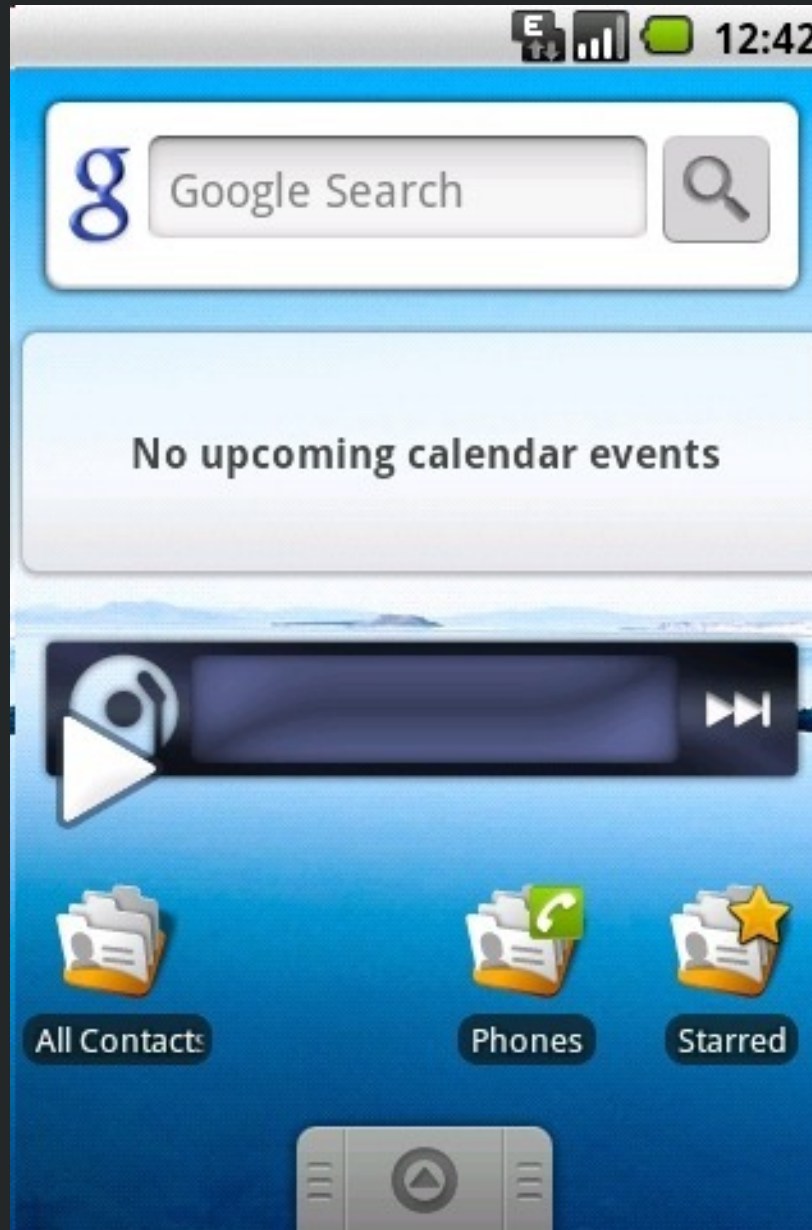


iOS 3

ES3 (1999)

HTC Hero





Android 1.5

ES3 (1999)



5

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>

A swirling blue and white vortex, resembling a time warp or a deep ocean current, with a bright white light at the center. The year 2012 is written in a large, white, serif font across the middle of the image.

2012

iPhone 5



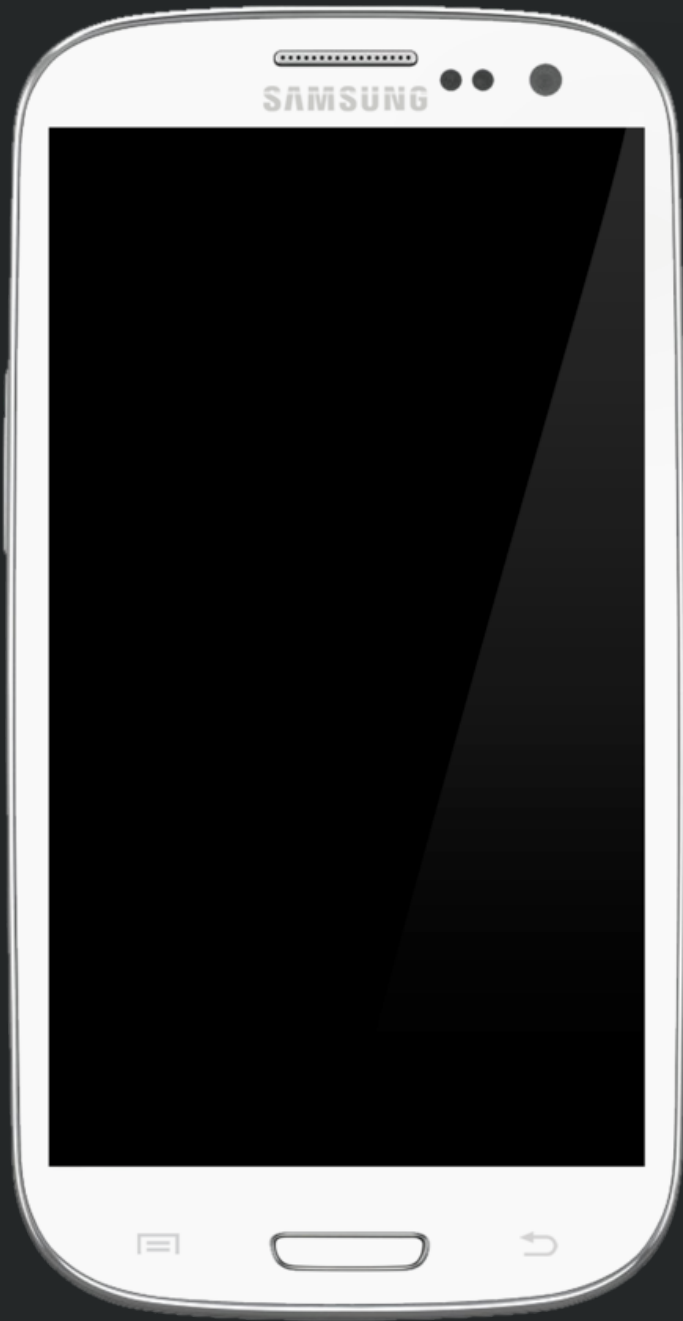


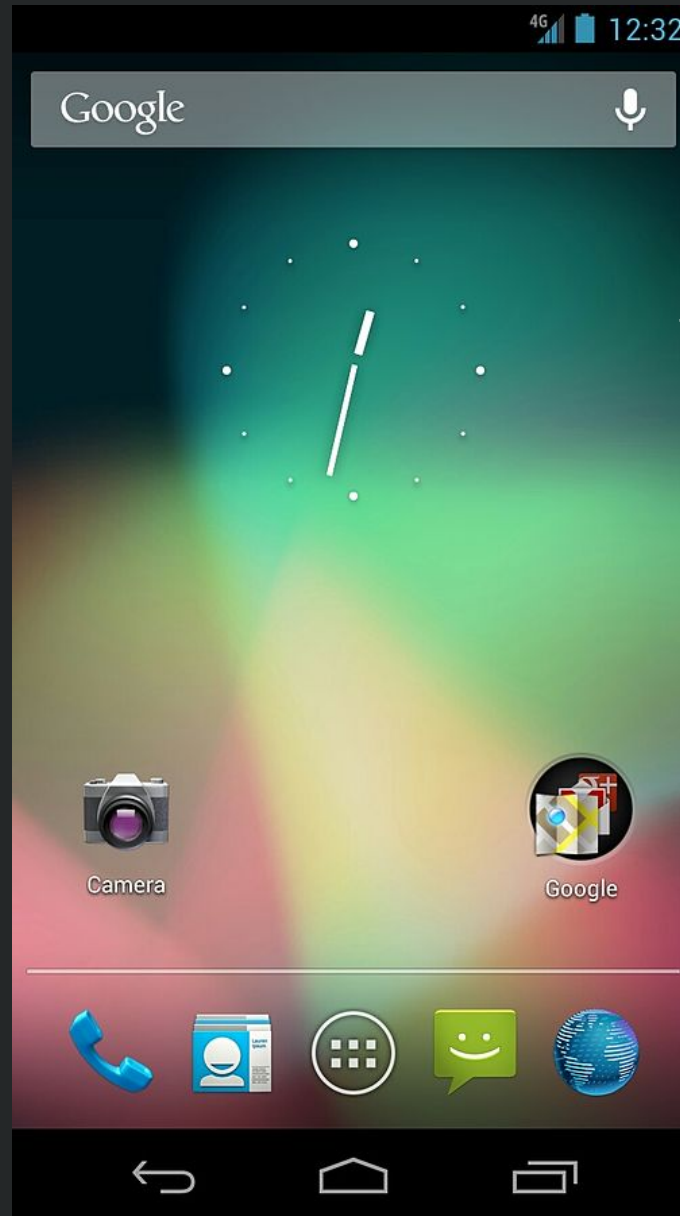
iOS 6

93% support for ES5

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

Samsung Galaxy S3





Android 4.1 Jelly Bean

85% support
for ES5

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

A dynamic, swirling vortex of blue and white light, resembling a time warp or a cosmic storm. The center is a bright, glowing white sphere, from which the swirling patterns of blue and white energy emanate. The overall effect is one of intense motion and energy.

2015

 ES2015 

née ES6

iOS 9

54% ES2015

Android 5.1

25% ES2015

ES2015 (ES6)

New to the language...

Block-scoped <code>let</code> & <code>const</code>	Destructuring and named params
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

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()
```


A swirling blue time vortex with a bright white light at its center. The vortex is composed of many concentric, swirling bands of blue, ranging from deep navy to bright cyan. The center of the vortex is a bright, glowing white light that fades into the blue as it moves outwards. The overall effect is one of intense energy and motion.

2017

ES2017

`async / await`

String padding

Shared memory

Atomics

A swirling blue and white vortex, resembling a time warp or a deep ocean current, with a bright white light at the center. The year 2018 is written in a large, white, serif font across the middle of the image.

2018

ES2018 and beyond

Template Literal Revision: <https://tc39.github.io/proposal-template-literal-revision/>

Stage 3 Proposals

<code>global</code>	Object Rest/spread
async iteration	<code>import()</code>
RegExp improvements	<code>Promise.prototype.finally</code>
BigInt	Class Fields
Optional <code>catch</code> binding	<code>import.meta</code>

<https://esdiscuss.org>

(<https://mail.mozilla.org/listinfo/es-discuss>)

<https://github.com/tc39/ecma262>

A quick intro to ES2015+

Block. Scoped. Variables. Finally.

```
const NUMBER_OF_DOCTORS = 13;
const MY_DOCTOR = "David Tennant";

let i = NUMBER_OF_DOCTORS;
for (let i = 0; i < 100; i++) {
    console.log(MY_DOCTOR, i); // David Tennant 0, ...
}
console.log(i);                // 13
```

Constants !== immutable

The variable can't be reassigned, but the contents are still mutable.

```
const THE_DOCTOR = {  
  person: "Peter Capaldi"  
};
```

```
THE_DOCTOR.person = "Jodie Whittaker";  
console.log(THE_DOCTOR.person); // Jodie Whittaker
```

this is such a pain...

```
var doctor = {  
  greeting: "Hello, %target%! I'm the Doctor!",  
  sayHi: function(evt) {  
    alert(this.greeting.replace(/%target%/g,  
      evt.target.dataset.target));  
  },  
  start: function() {  
    document.querySelector("#sayHiToK9")  
      .addEventListener("click", this.sayHi, false);  
  }  
}
```

**TypeError: undefined is not an object
(evaluating 'this.greeting.replace')**

Arrow functions

```
const doctor = {
  greeting: "Hello, %target%! I'm the Doctor!",
  sayHi: function(evt) {
    alert(this.greeting.replace(/%target%/g,
      evt.target.dataset.target));
  },
  start: function() {
    document.querySelector("#sayHiToK9")
      .addEventListener("click",
        evt => this.sayHi(evt), false);
  }
};
```



Hello, Kg! I'm the Doctor!

Arrow function quirks

Zero or 2+ parameters? Use parentheses:

```
[1, 2, 3].map(() => Math.floor(Math.random() * 100));  
[1, 2, 3].map((i, idx) => i * idx);
```

One parameter? Convention is no parentheses:

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

Need only the second?

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

Arrow function returns

Single line arrow functions use implicit return :

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

Block arrow functions use explicit return :

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


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:
```

```
    i * 2;
```

```
});
```

```
// obviously not what
```

```
// we want :-(
```

Arrow function ambiguity

Instead, wrap the object in parentheses:

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

Or, just use the block form:

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

Template Literals

Multiline and expression interpolation:

```
function sayHelloAndGoodbye(name) {  
    return `Hello, ${name},  
    Goodbye, ${name}`;  
}  
  
console.log(sayHelloAndGoodbye("Doctor"));  
// Hello, Doctor!  
// Goodbye, Doctor!
```

Template Literals

Arbitrary expressions (*use with care*):

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

```
sayComplexHello("Sarah");    // Hello, Sarah!  
sayComplexHello();           // Hello, Doctor!
```

Promises, Promises

More concise with arrow functions:

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

Promises, Promises, Promises

Chaining is easier to read:

```
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.error(`Error received! ${err}`);  
}  
  
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];  
  }  
}  
  
const [results, error] = divide(4, 0);
```

async / await (ES2017)

```
async function start() {  
  try {  
    const pos = await getPos(),  
      {coords:{latitude, longitude}} = pos;  
    console.log(`${latitude}, ${longitude}`);  
  } catch(err) {  
    console.error(`Error received! ${err}`);  
  }  
}
```

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

Array-like conversion

```
// ES 5 way to convert a NodeList to an Array
```

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

```
// ES2015 method
```

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

```
// Can also construct series:
```

```
const series = Array.from({length: 8},  
    (_, idx) => idx * idx); // [0, 1, 4, 9, 16, 25, ...]
```

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();
```

```
// specify only quality
```

```
getPicture({quality:75});
```

```
// specify only height & width
```

```
getPicture({height: 1024, width: 1024});
```


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

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+* to ES5:

- Babel (née es6to5)
- TypeScript
- Bubl  **
- Traceur

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

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

Module Support is problematic

Browsers have only *recently* started shipping implementations:

- Available now:
 - Safari 10.1+, iOS 10.3+
 - Chrome and Android Webview 61+
- Behind a flag:
 - Edge 15
 - Firefox 54

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

```
import Game from "./Game"; // bare import (won't work)  
import Decimal from "Decimal"; // also won't work
```

- No “bare” import !
 - Must include the path
 - Must include the extension
 - No node-style resolution
- iOS module loading does not work in PhoneGap / Cordova

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.

We can supply bundled and unbundled versions:

If you want...

```
<!-- this script will have modules, and executes in  
      modern browsers -->
```

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

```
<!-- this script won't, and will only execute in  
      older browsers * -->
```

```
<script nomodule src="./js/index.js"></script>
```

* Except Safari 10.1. See <https://gist.github.com/samthor/64b114e4a4f539915a95b91ffd340acc>
See <https://jakearchibald.com/2017/es-modules-in-browsers/> for more.

Execution Options

- Manual
- Task runner
 - gulp , grunt , etc.
 - npm scripts
- Automatic
 - Plugin Hooks
 - Project hooks

Automating with npm scripts

- Pick your bundler and transpiler
 - Bundler: Webpack
 - 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
{ "compilerOptions": {
  "allowJs": true,
  "target": "es5",
  "module": "es2015", // DCR
  "lib": ["es2015", ...],
  "sourceMap": true
},
"include":
  ["www.src/es/**/*"]
}
```

```
// .babelrc
{ "presets": [
  ["es2015", {
    "loose": true,
    "modules": false // DCR
  }],
  "es2016", "es2017"
],
"plugins": ["transform-runtime"]
}
```

* 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"})  
  ]  
}
```

... don't forget to update index.html

www/index.html:

```
<body>  
  <script type="text/javascript" src="js/vendor.js">  
  </script>  
  <script type="text/javascript" src="js/bundle.js">  
  </script>  
</body>
```

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/bundle
```

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-template-vue-f7-blank	devgeeks	Webpack	Babel	Vue, F7	npm
phonegap-template-vue-f7-split-panel	devgeeks	Webpack	Babel	Vue, F7	npm
phonegap-template-vue-f7-tabs	devgeeks	Webpack	Babel	Vue, F7	npm
phonegap-vueify	lemaur	Browserify	Babel	Vue	npm

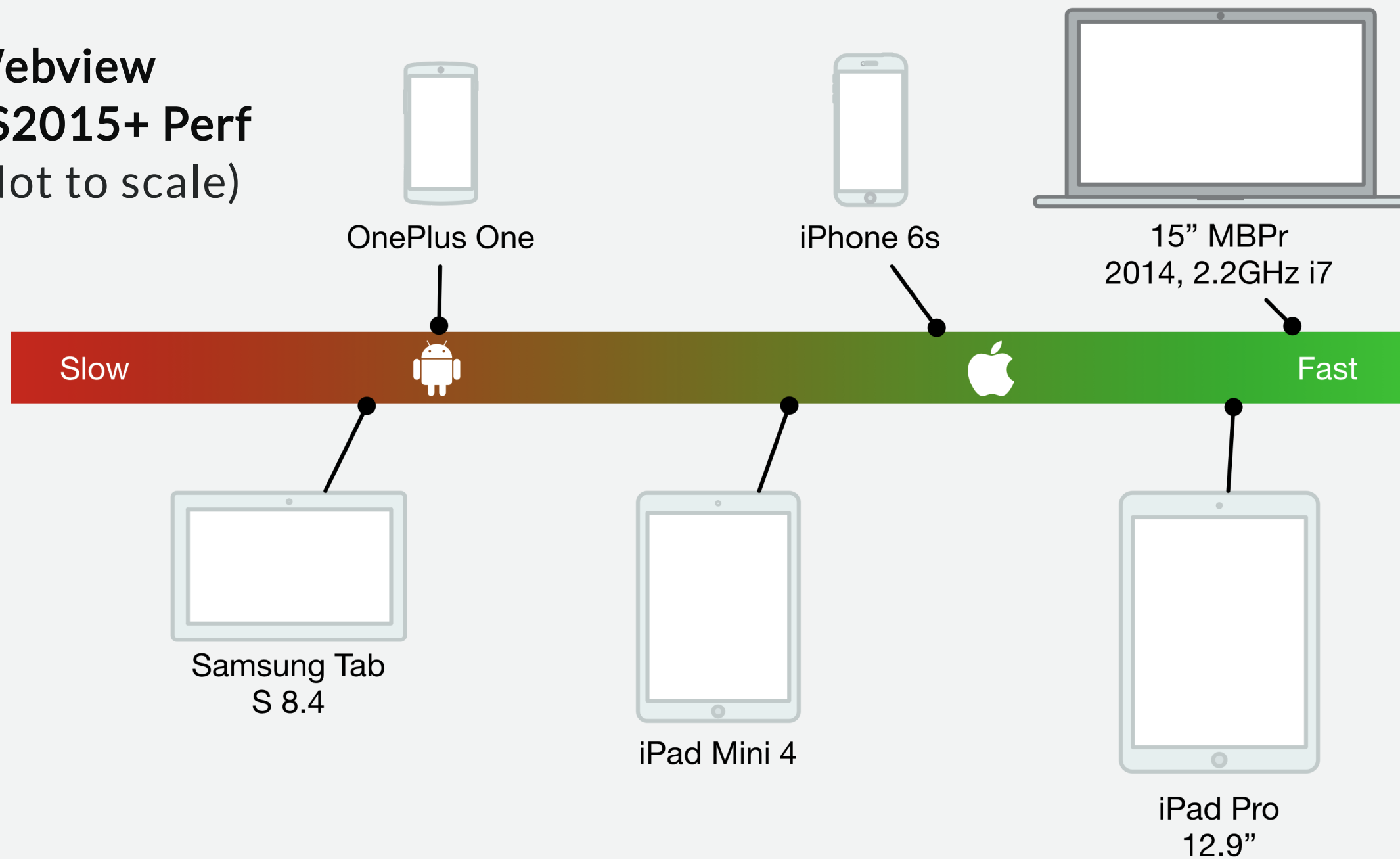
Video

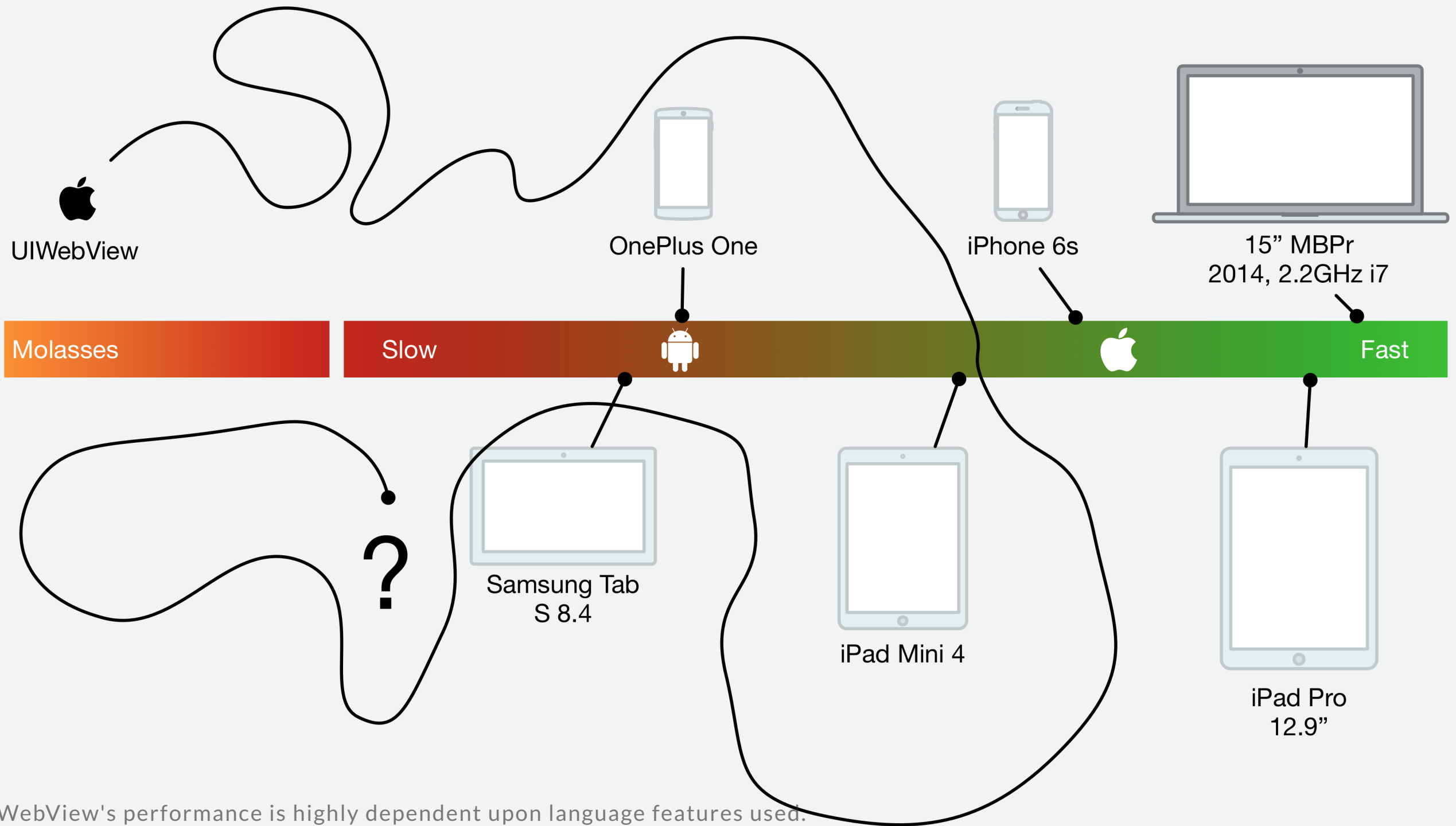
Reality Check...

- Don't switch to ES2015+ because you expect performance improvements
 - See <https://kpdecker.github.io/six-speed/> (as of 2017-01-04)

		node 6.9.37.3.0		chrome 5549		firefox 5053		edge 14.1439315.14959		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

Webview ES2015+ Perf (Not to scale)





More Reality Checks

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

Euuuggghhhh!!!

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
 - Modules, Template literals, and more!

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)
- Arrow functions aren't drop-in
 - `this` is lexically scoped

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

Thanks!

@kerrishotts

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

This slide intentionally left blank