onJavaScript

lukas.weber@socialbakers.com

LEVEL 1

HISTORY

- 1995 JavaScript (Netscape Navigator 2.0, Brendan Eich)
- 1996 JScript (Internet Explorer 3)

DARK AGE™

012645

08284808

"Optimized for IE4" a 

GIFs

Mix of HTML and <script> tags

Form validation

Obtrusive code

Security ...wasn't

FIRST BROWSER WAR ERA

- JavaScript competitors
  - Macromedia Flash
  - JAVA Applet
- Netscape vs. Microsoft
- Microsoft "Won" with Internet Explore 5
- XMLHttpRequest

LEVEL 2

Web Renaissance

- **AJAX a Web 2.0**
- **Rise of PrototypeJS, Mootools and jQuery**
- **Unobtrusive Javascript**
- **CSS Selectors**
- **Still mash of HTML and JS**
- **Security concerns**

AJAX

- **XMLHttpRequest**
- **"Let client's CPU do the work" mentality**
- **Less data over network**
- **Better UX - almost instant responses**
- **Move from web pages to web applications**

Second Browser War Era

- **Internet Explorer became ancient nightmare of web designers and developers**
- **Firefox a took the mainstream**
- **Then Chrome arrived**
- **...Microsoft didn't recovered until IE 9**

LEVEL 3

JavaScript Today

- **All browsers standardized* incl. IE**
- **HTML5**
- **Server-Side JavaScript**
- **Mobile web**
- **You name it!**

...JavaScript Everywhere

***mostly**

IT WILL BURN ITSELF IN YOUR MEMORY FOREVER!

All other experiences shrink into insignificance as you witness this greatest of all motion pictures.

SEE

Trial by Fire of Modern Transgressors

The World's Most Beautiful Sinners
paying eternally for their shame

The Carnage of the Pleasure-Mad

The Judgment Seat of the Damned

The Cliff of Frozen Mortals

The Dread Black River of Woe

The Sea of Boiling Pitch

The Crater of Doom

The Lake of Flame

The Rain of Fire

Thousands of Restless Souls Drifting Forever through Hell's Caverns

Dependency Hell

AMD IS FOR BROWSERS

```
1 // helper.js
2 define("helper", ["stdio"], function (stdio) {});

1 // main.js
2 require(["helper"], function (helper) {});
```

CommonJS is for server

```
1 // helper.js
2 exports.helper = function (stdio) {};
```

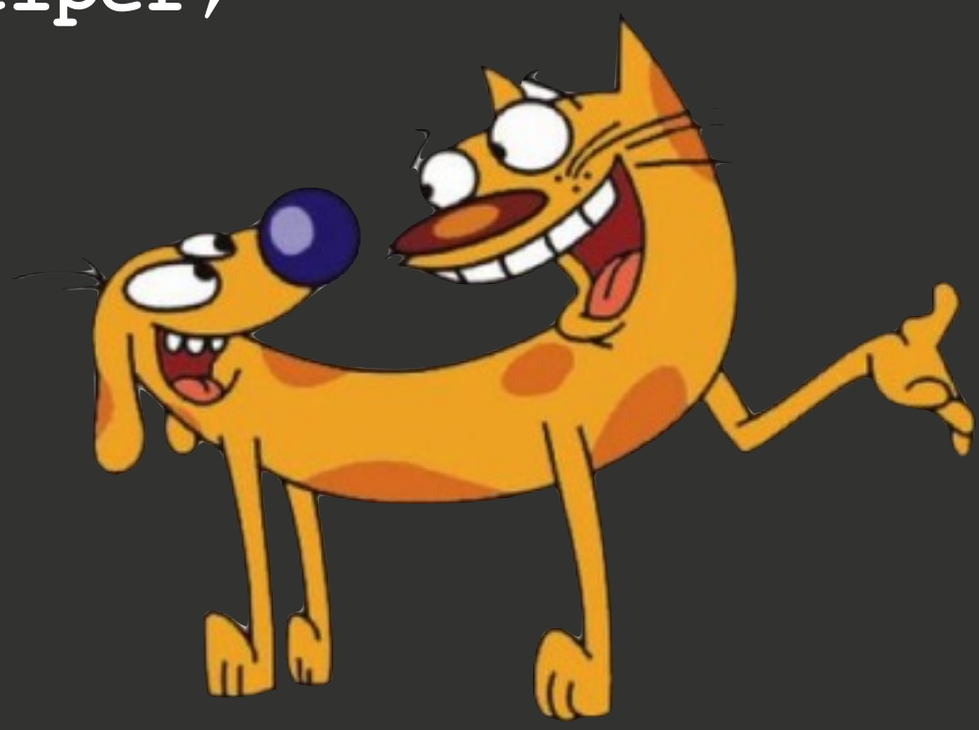
```
1 // main.js
2 var stdio = require("stdio");
2 var helper = require("./helper")(stdio);
```

ES6

Modules are for?

```
1 // helper.js
2 export function helper (stdio) {};
```

```
1 // main.js
2 import * from io;
3 import helper from helper;
```



npm

- node package manager
- server side
- client side via ***Browserify***

Bower

- Web package manager
- bower.json
- simplicity of NPM

Build Systems

- Gruntfile.js
- Declarative
- FS based

- Gulpfile.js
- Imperative
- Streams

webpack
MODULE BUNDLER

CommonJS

Lot of Magic™

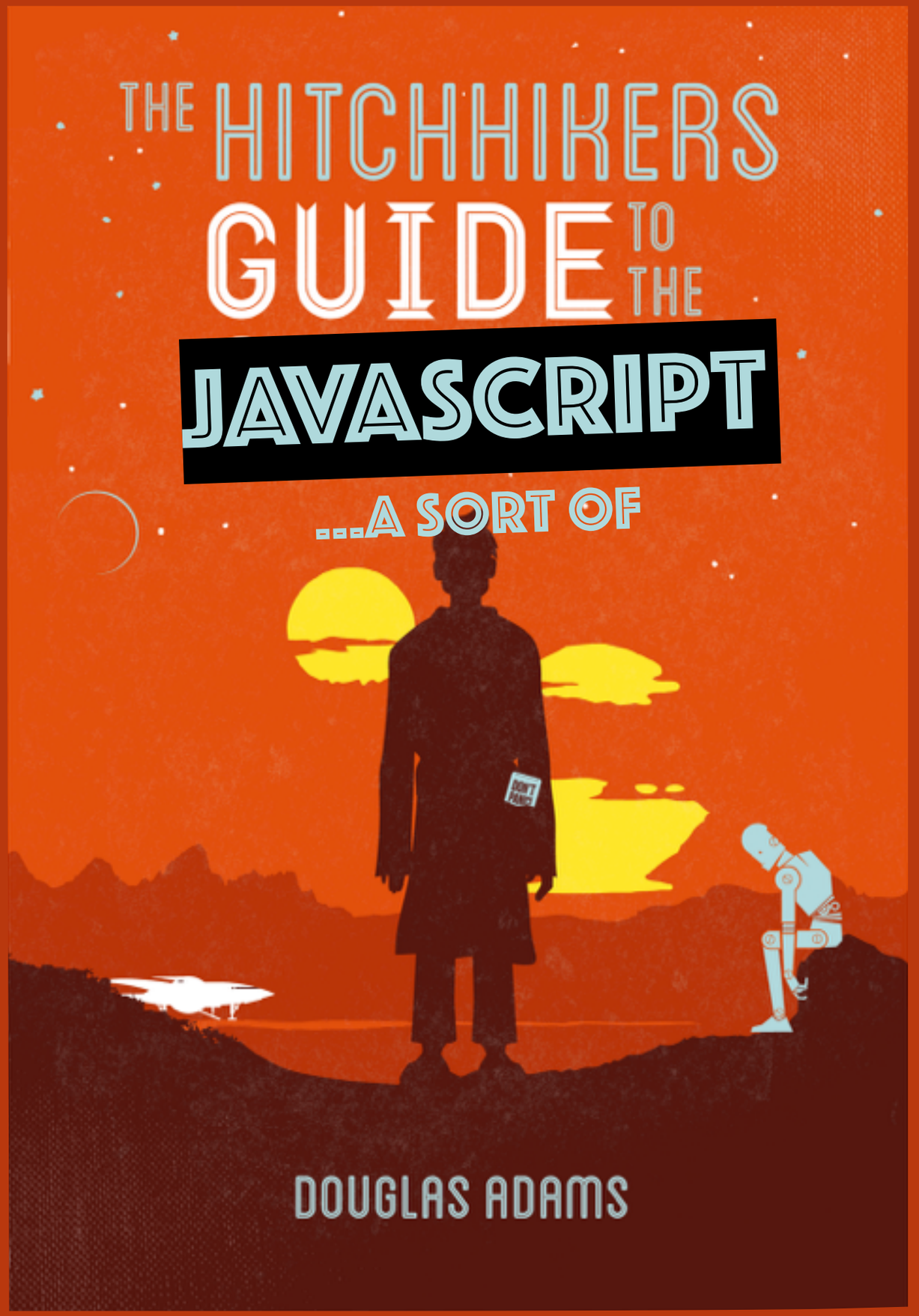
Transformations

browserify

CommonJS/AMD

Code Splitting

Loaders


THE HITCHHIKERS
GUIDE TO THE
JAVASCRIPT
...A SORT OF
DOUGLAS ADAMS

funkyzeit mit brüno

IN	AUS
SPA	jQuery
MVC, MVVM, MVP, HMVC, ...	Monolithic **"do it all"** frameworks
Mobile first	Desktop first
Unidirectional data flow	Event-based mess and global state

Neither column is right,
there is lot of "fad" in programming

Transpilled JS

EXPRESIVITY

SIMPLICITY

EFFECTIVITY

SAFETY

PRODUCTIVITY

CoffeeScript

TypeScript

ClojureScript

Objective-J

AtScript

Dart

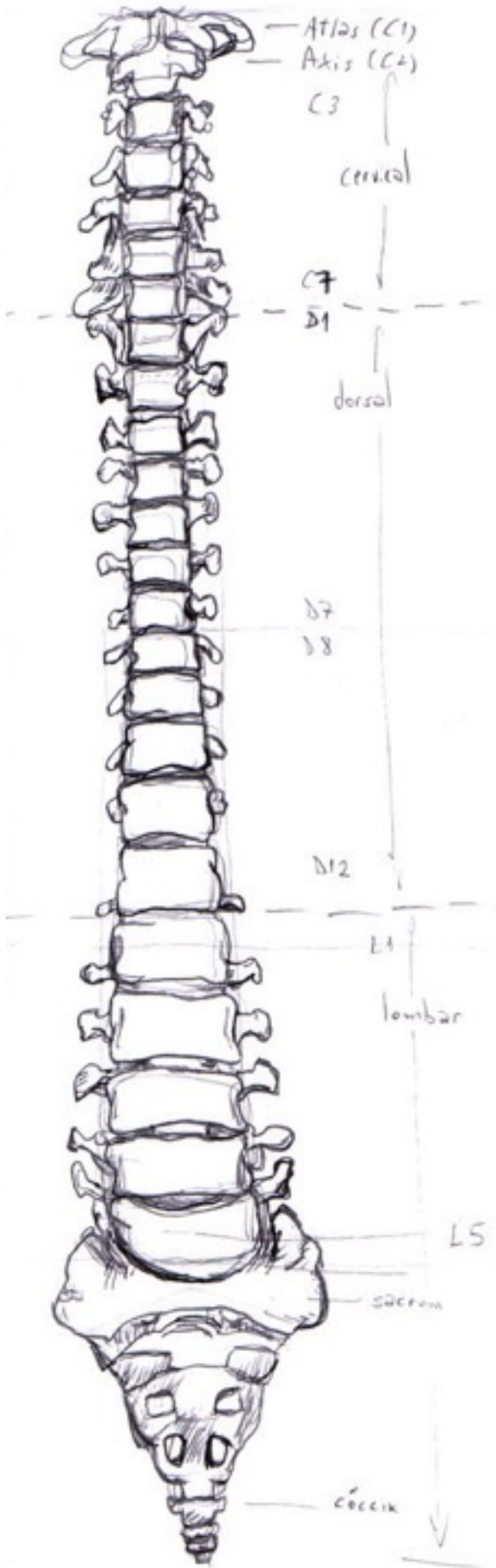
JSX

Sweet.js

elm

asm.js

Closure Compiler



Backbone.js

- First MV(whatever) I met on the web
- Still pretty good
- But very barebones
- Which makes good building block

- "Backbone with included batteries"
- Declarative, $directives directly in HTML
- Dependency injection
- Two-way data binding ...and dirty checking
- Version 2.0 will break everything
- by Google but not publicly used

Closure Tools

- Very old (2005) but not outdated, just misunderstood
- by Google, internal project without public spotlight
- Library, Templates and Compiler
- Everything finetuned to work together

Closure Compiler

- Dependency management
- Type checking
- Various code speed and size optimization
- Dead code removal
- Minification and obfuscation
- Actively developed

Coolest unknown javascript tool ™

React

- Facebook
- Inspired by Game Engines
- Virtual DOM & Event System
- 60 fps
- Extremely easy to understand
- State handling

the performance

What is performance?

Measured performance

work / time

- Hard numbers like FPS
- Memory and CPU demands

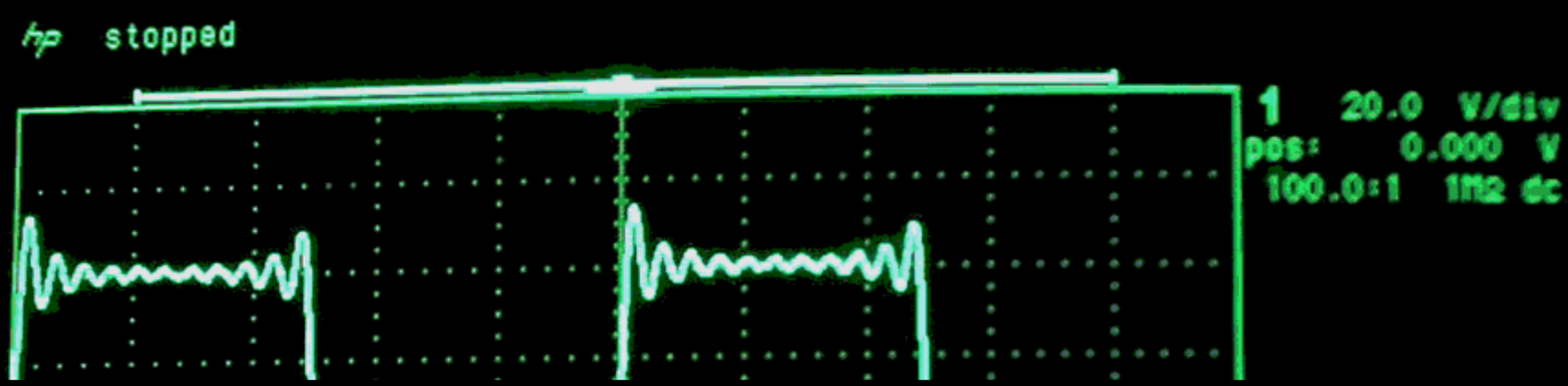
Perceived performance

boredom x time

- Psychological aspects
- Broken dopamine loops and flow

Easy side of performance

- Target to **30-60 FPS** (33.3ms or 16.7ms per execution branch)
- Memory and CPU are still constraint on Mobile devices
- Use **WebWorkers**
- Batch work to **smaller batches** (use queues or CSP)
- Moving with stuff? Use requestAnimationFrame()



Common bottlenecks

- Memory leaks - use incremental heap dumps to find them
- Drawing performance - at 99% caused by DOM manipulation
  - do all updates in one time
  - disconnect nodes when lot of operations is needed
- Computation performance - stack introspection FTW, Flame Graphs
- Network - reduce number of request for first render

Hard part

- User experience - very hard to make it right at first shot and even after many attempts, it will be always just half luck :)

- It's like security - it's nothing you can just get and have, it's process you have to follow

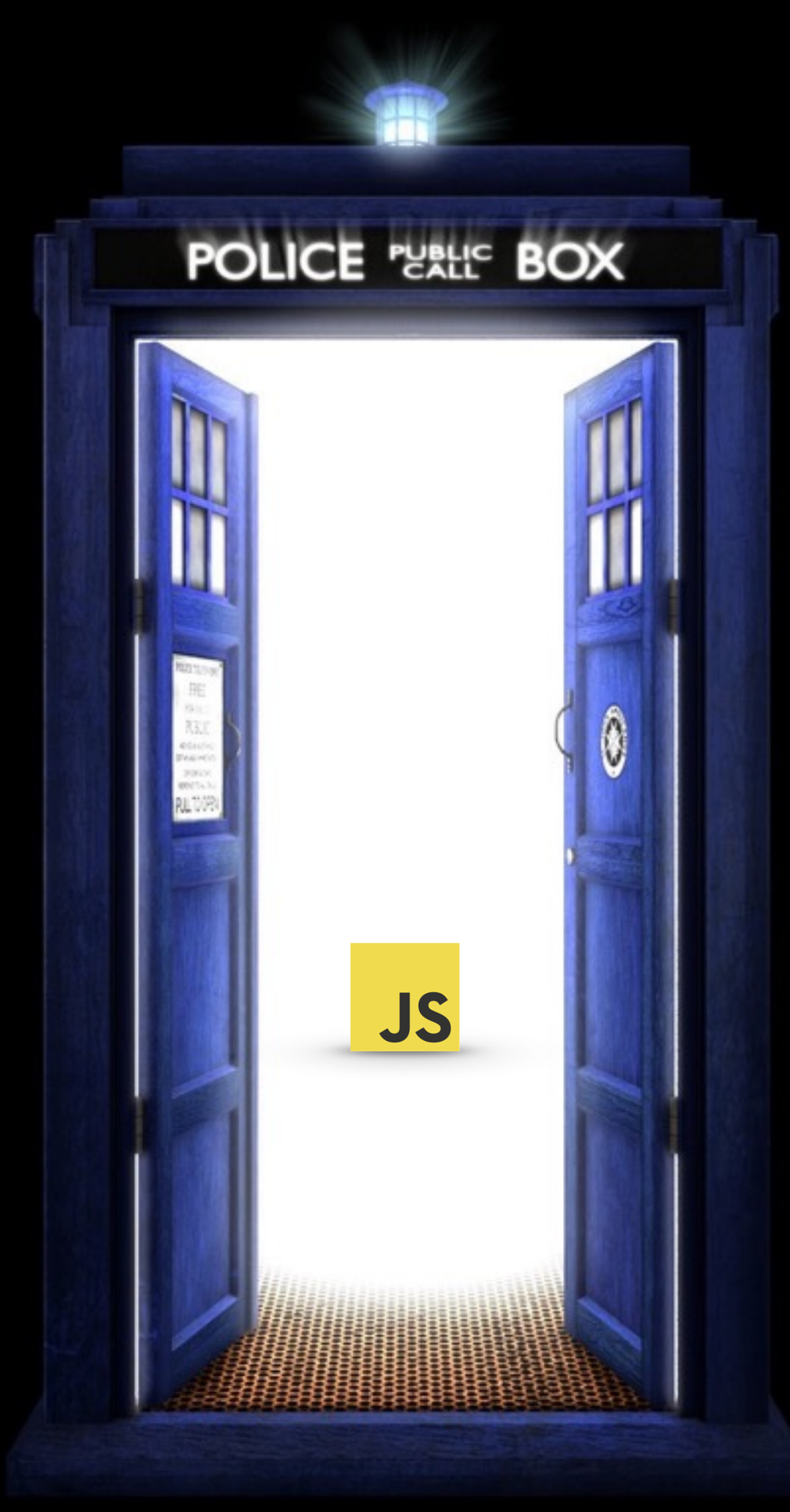
- Even (measurably) fastest app can feel snail-slow to user, because of unnecessary waiting, bad navigation, network lags, broken flows, etc.

- Figure out usability first, then take care of code (if you're not doing CS)

- Learn to understand to your users and your UX designers (*this one is really hard:)*)

Future

- HTTP2
- ECMAScript 6
- WebComponents
- WebGL
- Native development
- asm.js?



Thank you!

lukas.weber@socialbakers.com
@techbakers