Modern JavaScript Development

(A Bootcamp for Java Developers)

Mail: jonas.bandi@gmail.com
Twitter: @jbandi

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

– Atwood's Law, 2007

http://www.codinghorror.com/blog/2007/07/the-principle-of-least-power.html

We are slaves to JavaScript because people have simply started to accept its weirdness and flaws, much like a Stockholm Syndrome phenomenon.

– Chris Richardson, 2013

http://www.youtube.com/watch?v=CN0jTnSROsk&list=WLB09959CADB0E3254

Two Worlds?

AGENDA

JS **JavaScript: An Overview**

Demo: A modern front-end

JS **Closures & Modules**

JS **Language Exercises**

ABOUT ME

Jonas Bandi
jonas.bandi@gmail.com
Twitter: @jbandi

Fragen und Anregungen zum Workshop erwünscht!

Ich unterrichte den CAS "Applikationsentwicklung mit JavaScript und HTML5" an der Berner Fachhochschule.

In den letzten 2 Jahren habe ich viele in-house Schulungen zu den Themen JavaScript, EcmaScript 6, jQuery, AngularJS und HTML5 gehalten. Unter anderem bei UBS, Postfinance, Schweizerische Mobiliar, Bundesamt für Informatik ...

Interessiert an JavaScript Schulungen?
Visit: http://ivorycode.com/#schulung

ADMINISTRATIVES

Unterlagen

Git Repo:
https://github.com/jbandi/JUGS-JS

Initial Checkout:

```
git clone https://github.com/jbandi/JUGS-JS
```

Update:

```
git reset --hard
git pull
```

(setzt alle lokalen Veränderungen zurück)

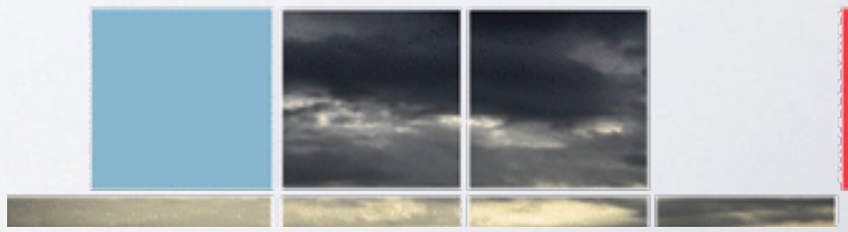
Slides: `<checkout>/00-CourseMaterial`

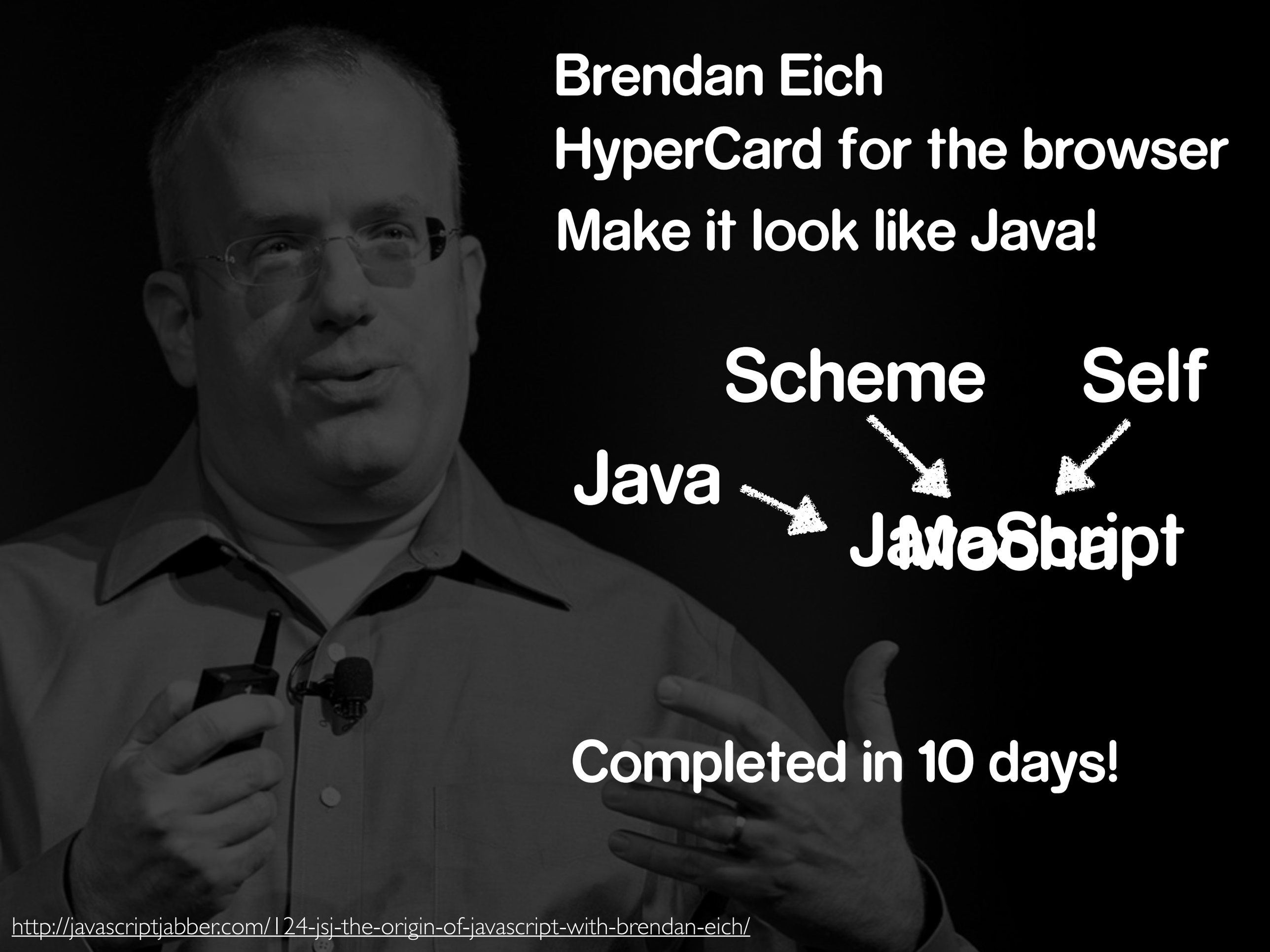
JavaScript: An Overview

jQuery
write less, do more.

Building Windows 8 Apps *with* JavaScript

node.js™





http://javascriptjabber.com/124-jsj-the-origin-of-javascript-with-brendan-eich/

JavaScript != Java

Scott Hanselman
@shanselman

Following

Java is to JavaScript as ham is to hamster, as car is to carpet, and as iron is to irony. h/t @RachelAppel

RETWEETS 338 FAVORITES 176

3:30 PM - 9 Oct 2014

THE HISTORY OF JAVASCRIPT

- 1995: Netscape Navigator 2.0 – LiveScript
  - Cooperation with Sun: JavaScript 1.0
- 1996: IE 3 - JScript
- 1997: ECMAScript, 1998: ISO-Norm
- 2000-2006: Ajax Revolution
- 2009 Node.js created by Ryan Dahl
- 2010: JavaScript 1.8.5 / ECMAScript 5
  - Firefox 4, IE 9
- 2011: ECMAScript 5.1
- June 17th 2015: ECMAScript 2015 (ES 6) final Specification
- Work in Progress: ECMAScript 2016 (ES7)
- Today "JavaScript" is a trademark of Oracle Corporation
- Java SE 6 contains a complete JavaScript Runtime (Rhino)
- Java SE 8 contains a new JS Runtime called “Nashorn”

SUCCESS OF JAVASCRIPT

- JavaScript is the Language of the Web
  - Every OS ships with a WebBrowser
  - Every Smartphone ships with a WebBrowser
  - JavaScript is the most popular VM
- Performance Improvements in Browsers
  - The Second Browser War: ~Factor 10 in JS Perf
- Performance Improvements in Devices
- REST & The Cloud - Simple JS Interfaces to Services
  - JSON
  - No middleware needed - Simplicity!
  - Services directly addressable from Browser

HTML 5

HTML5 explained:

- Html is the structure
- CSS is the colour and style
- JavaScript is everything else

Scott Hanselman: Angle Brackets, Curly Braces http://channel9.msdn.com/Events/Build/2012/3-027

Beyond the Browser

mean

sails

express

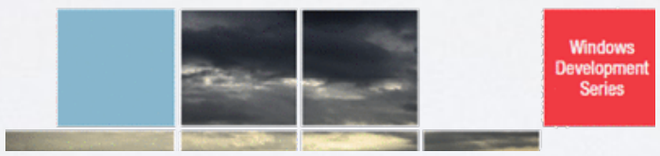
node

Visual Studio Code

ATOM

ELECTRON

Building
Windows 8 Apps
with **JavaScript**



React Native

Native Script

appcelerator®

ionic

PhoneGap

APACHE CORDOVA™

Java 8

Nashorn

OS X Yosemite

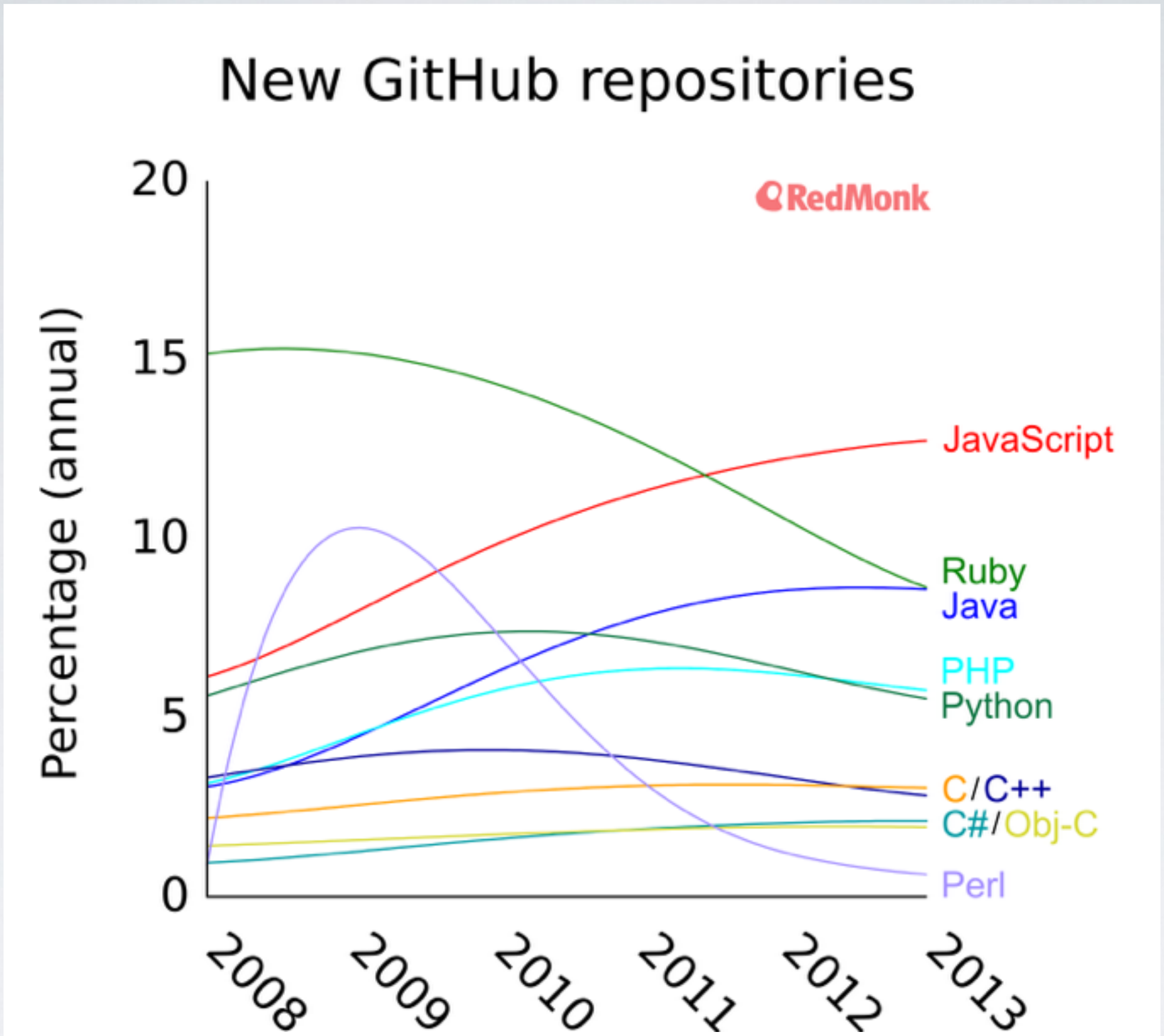
RethinkDB

Apache CouchDB

mongoDB

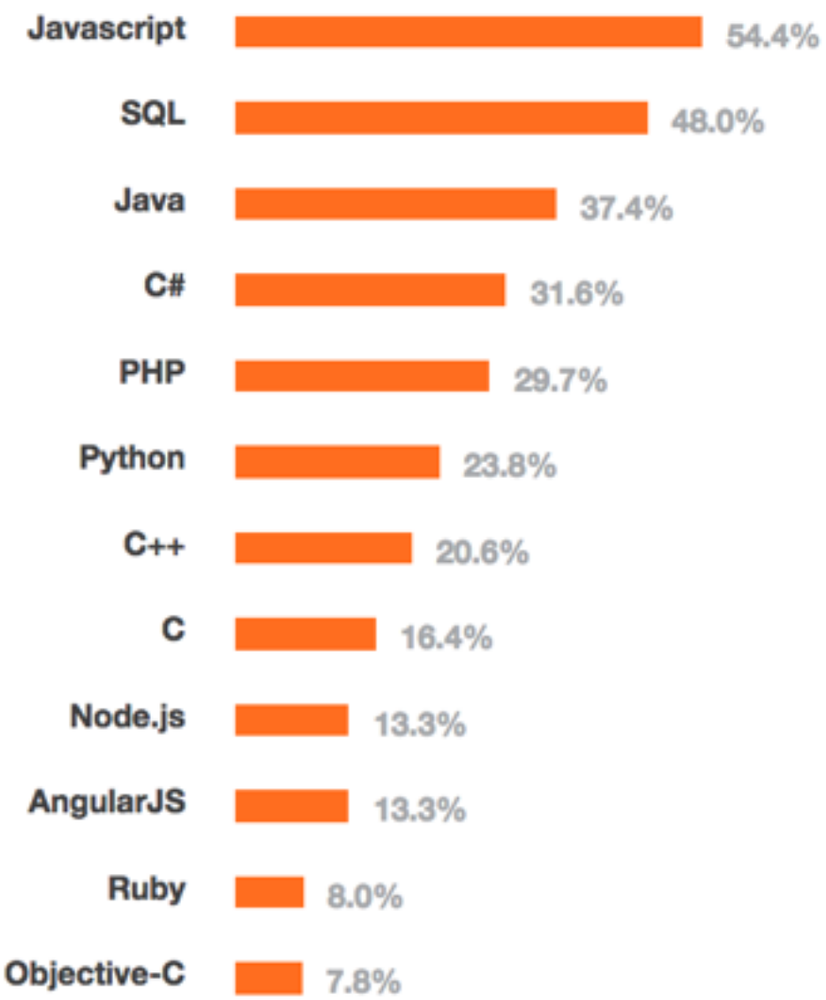
ARDUINO

nodebots.io



http://redmonk.com/dberkholz/2014/05/02/github-language-trends-and-the-fragmenting-landscape/

2015 2014 2013



21,982 responses

java × 841751

a general-purpose programming language designed to be used in conjunction with the Java Virtual Machine (JVM). "Java

629 asked today, 6150 this week

javascript × 833443

JavaScript (not to be confused with Java) is a dynamic, weakly-typed language typically used for client-side scripting. Use

522 asked today, 6344 this week

c# × 784367

a multi-paradigm programming language encompassing strong typing, imperative, declarative, functional, generic, object-

324 asked today, 4135 this week

http://stackoverflow.com/research/developer-survey-2015#tech

http://stackoverflow.com/tags

LARGE JAVASCRIPT APPLICATIONS

? LoC

? LoC

? LoC

ACTIVE ECOSYSTEM

www.npmjs.org

143,345 total packages

20,749,065 downloads in the last day

305,036,905 downloads in the last week

1,254,076,126 downloads in the last month

(Maven Central has 930k artifacts / 109k unique artifacts)

I Am Devloper
@iamdevloper

Following

Being a JavaScript developer in 2014 is like speed dating.

45 seconds with each framework before you have to re-write.

RETWEETS 1,104 FAVORITES 637

8:25 AM - 16 Dec 2014

First Generation Frameworks

jQuery
write less, do more.

Ext JS

dojo
toolkit

prototype

scripty2

script.aculo.us
it's about the user interface, baby!

MOOTOOLS 1.3

YUI

A "Second Generation" of JavaScript Frameworks

BACKBONE.JS

Spine

Build Awesome JavaScript MVC Applications

handlebars

batman.js

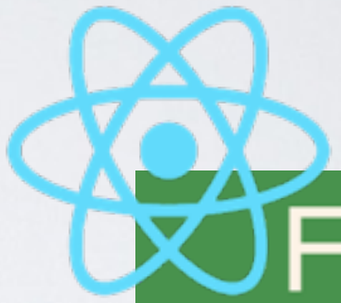
JavaScript MVC

Knockout.

- Client-Side MVC
- Application Structure
- Data-Binding

A "Third Generation" of JavaScript Frameworks

ANGULARJS
by Google



React

Flux
APPLICATION ARCHITECTURE FOR BUILDING USER INTERFACES

DURANDAL

aurelia

Complete Client-Side Application Development

JavaScript as a Compilation Target

"The Assembly Language of the Web"

gwtproject.org

cappuccino-project.org

Clojure Script

Google Traceur

flowtype.org

flow

by Facebook

Dart

BABEL

babeljs.io

https://github.com/jashkenas/coffeescript/wiki/List-of-languages-that-compile-to-JS

Hands-On: A naive JavaScript App

IMPROVEMENTS

- Unobtrusive JavaScript
- Separate Code from Markup

Separated & unobtrusive JavaScript is the foundation for professional JavaScript development.

Libraries / Frameworks

Language

IDE

dependency management

Write Code

Toolset

compile

package

Build

Test

Test-Frameworks

Deploy

Language

IDE

Libraries / Frameworks

dependency management

Bower

Write Code

Toolset

compile

package

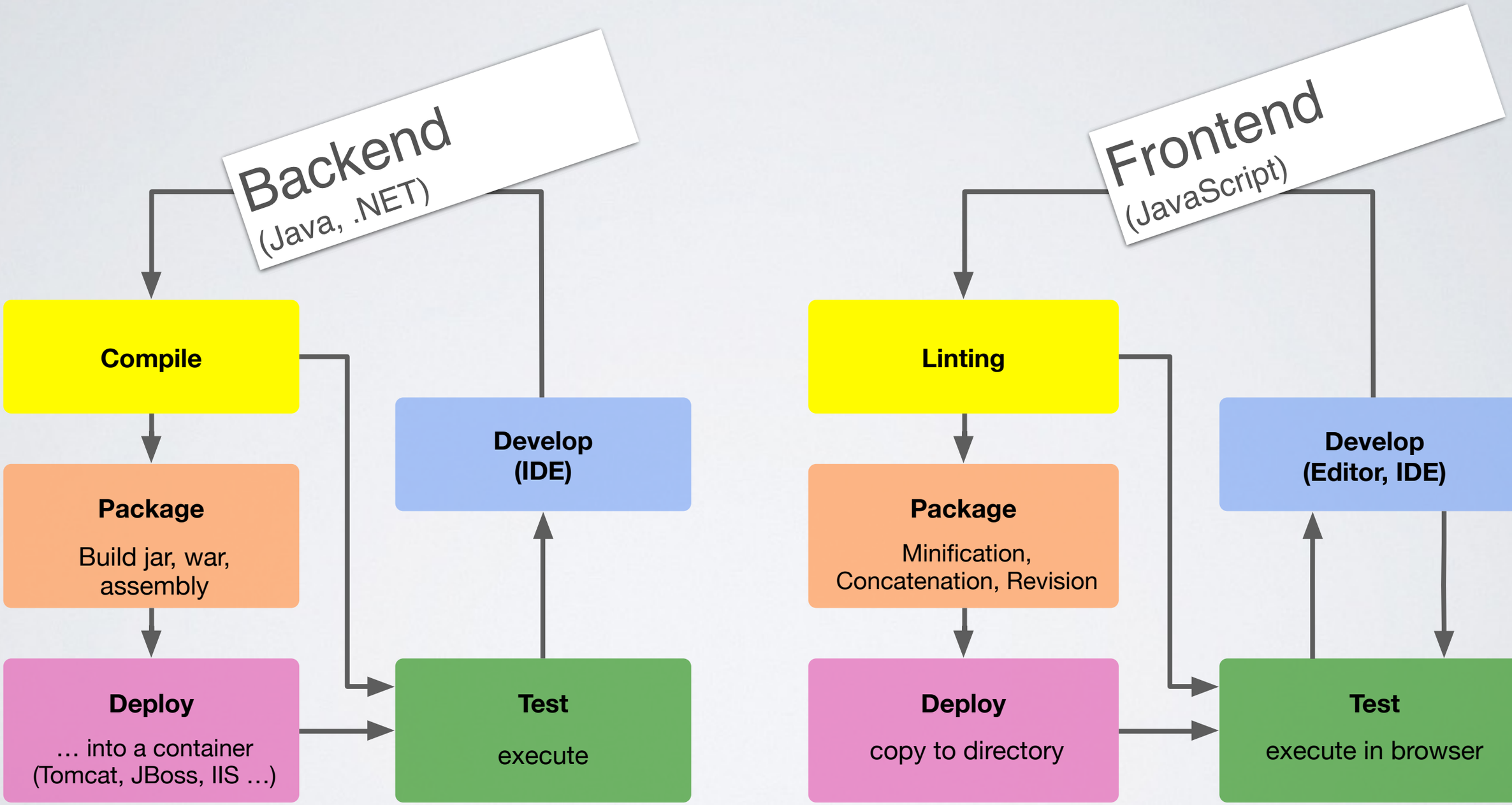
Build

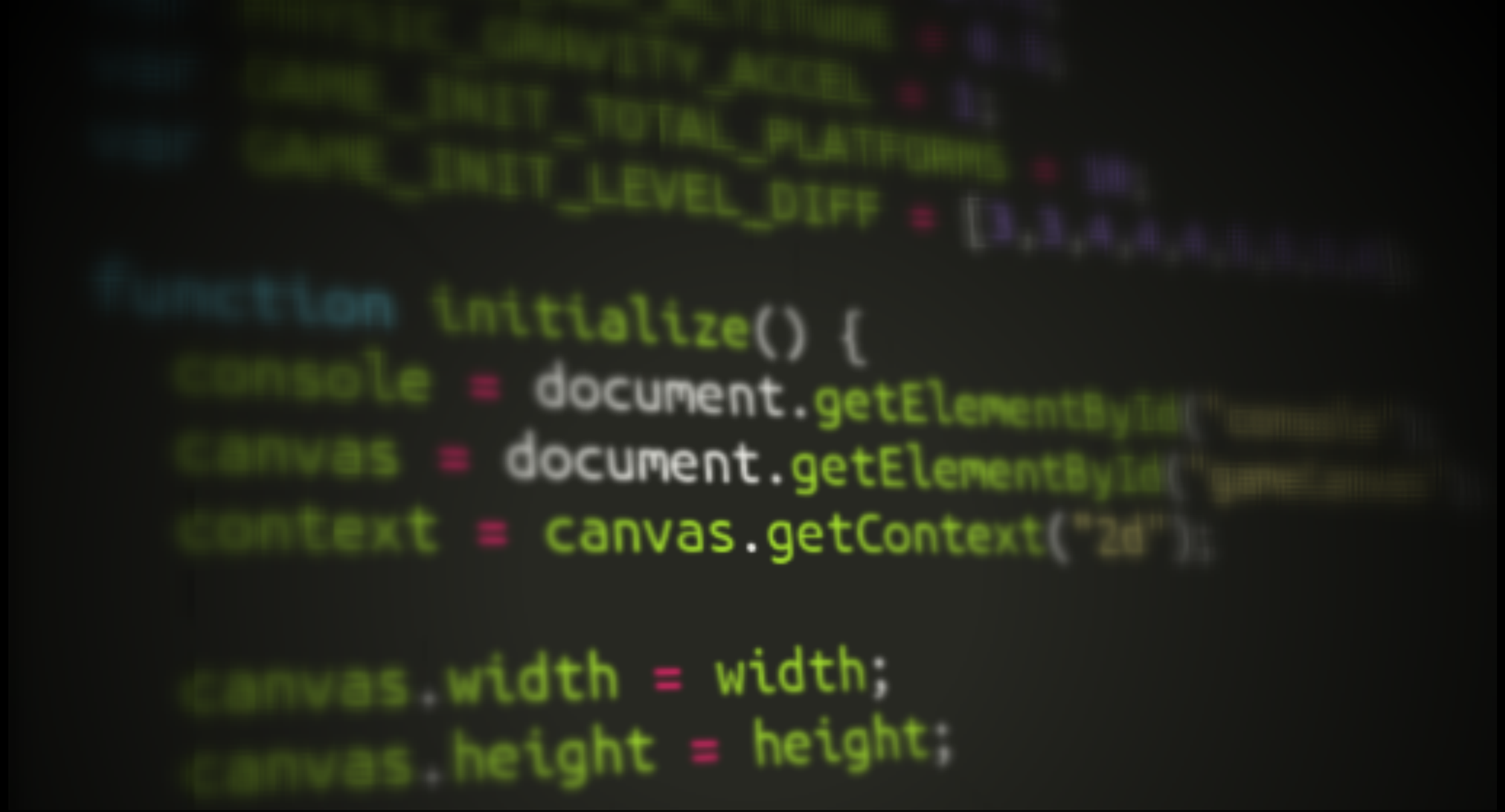
Test

Deploy

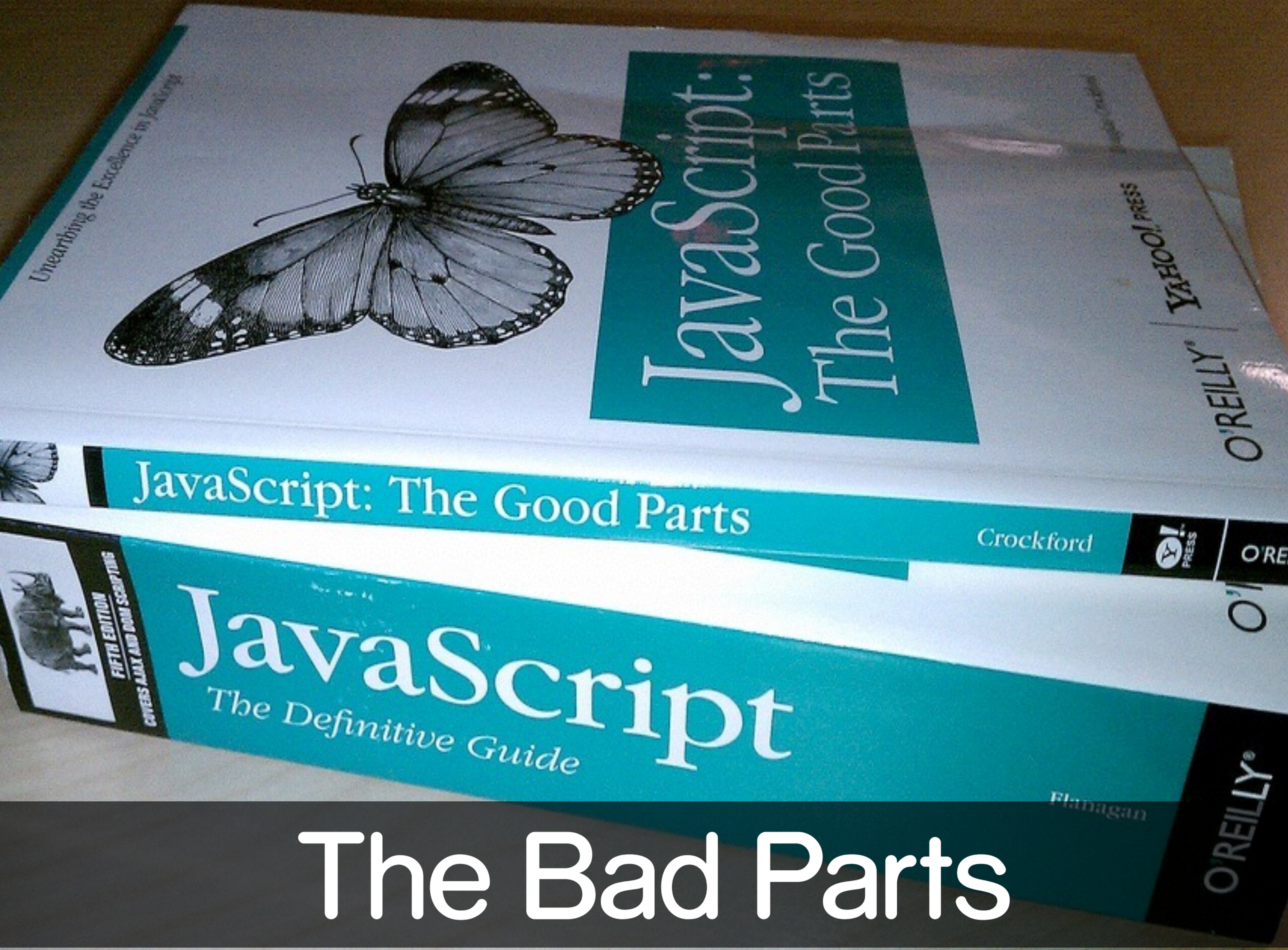
Test-Frameworks

Project Build





The Language



The Bad Parts

Language Constructs

- Object
- Class
- Method
- Constructor
- Packages
- Inheritance

- Object
- Function

...everything else can be implemented

Closures

Inner Functions

- Function can be nested inside functions
- Nested functions can access variables in their parent function's scope

```
function outerFunction() {
    var a = 1;
    function innerFunction() {
        return a + 1;
    };
    return innerFunction();
};
var x = outerFunction();
console.log(x);
```

```
function outerFunction() {
    var a = 1;
    function innerFunction() {
        return a + 1;
    };
    return innerFunction;
};
var x = outerFunction();
console.log(x);
console.log(x());
```

Demo: 03-Patterns/01-Closures/innerFunction.js

Closures

- Inner functions contain the scope of parent functions even if the parent function has returned.
- A closure is a special kind of object that combines two things: a function, and the environment in which that function was created. The environment consists of any local variables that were in-scope at the time that the closure was created.

```
var name = 'Bob';
function sayHello() {
   alert(name);
}

name = 'Tim';
sayHello(); // --> Tim
```

```
var name = 'Bob';
function getGreetingClosure() {
    var text = 'Hello ' + name;
    return function() {
             console.log(text);
             };
}
var sayHello =
getGreetingClosure();

name = 'Tim';
sayHello(); // --> Hello Bob
```

Demo: 30-Patterns/01-Closures

Immediately Invoked Function Expressions (IIFE)

- aka: Self-Invoking Anonymous Function
- The function is executed immediately after creation. No reference to the function and its scope remains.
- A useful pattern for encapsulation and to avoid polluting the global namespace.

```
var calcModule = (function(){

    // private members
    var a = [1,2,3];
    function calculate(){
        return (a[0]*a[1])+a[2];
    }
    function calcAndAdd(){
        return calculate() + 1;
    }
    function calcAndSubtract(){
        return calculate() - 1;
    }

    // public members
    return {
        add: calcAndAdd,
        subtract: calcAndSubtract
    }
})();
var res = calcModule.add();
```

Demo: 03-Patterns/02-IIFE

The Module Pattern

- Use a closure to separate private and public state & members
- Explicitly expose public state & members
- Establish a clear interface for a client of the module.

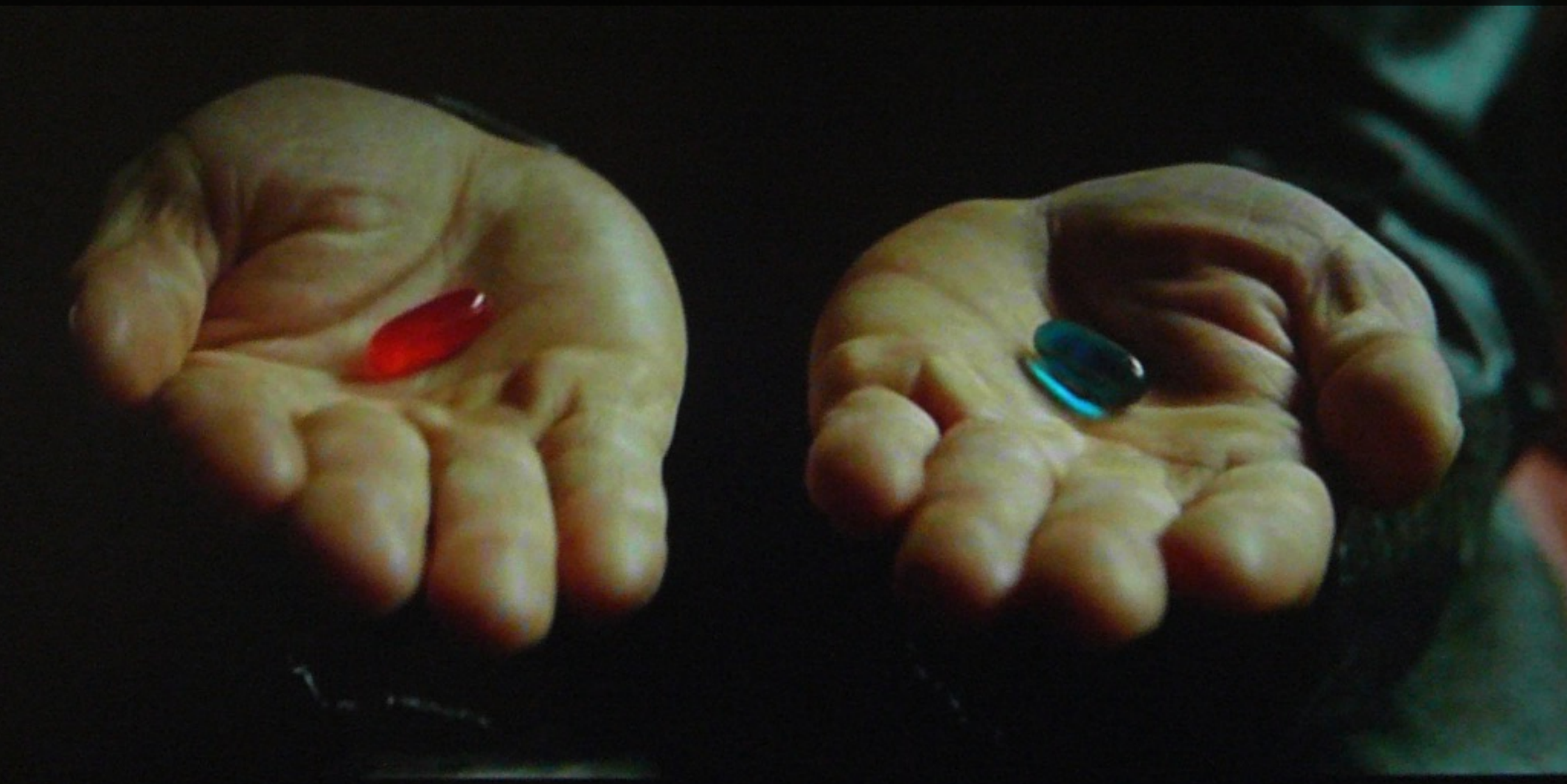
```
var calcModule = (function(){

    // private members
     var a = [1,2,3];
     function calculate(){
         return (a[0]*a[1])+a[2];
     }
     function calcAndAdd(){
         return calculate() + 1;
     }
     function calcAndSubtract(){
         return calculate() - 1;
     }

     // public members
     return {
         add: calcAndAdd,
         subtract: calcAndSubtract
     }
})();
var res = calcModule.add();
```

Demo: 03-Patterns/06-RevealingModule

Real World



CALCULATOR DEMO

Demo: 03-Patterns/06-RevealingModule/RevealiingModuleDemo

"SPAGHETTI" DEMO

Demo: 03-Patterns/03-Spaghetti

Understanding Scope

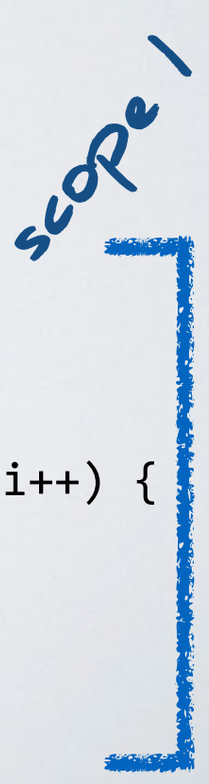
- Scope defines the lifetime of a variable.
- In JavaScript scope is solely provided by functions:
  - aka: Function scoping or lexical scoping
  - There is no block scope!

C, Java, C#:

```
public int add1(int n1, int n2) {
     var sum = n1 + n2;
     if (true) {
          var sum = 42;
          for (int i = 0; i < sum; i++) {
               int forty_two = 42;
          }
     }
     return sum;
}
```

JavaScript:

```
function add1(n1, n2) {
     var sum = n1 + n2;
     if (true) {
          var sum = 42;
          for (int i = 0; i < sum; i++) {
               int forty_two = 42;
          }
     }
     return sum;
};
```



Using a Closure to provide Scope

```
document.body.innerHTML = '';

var nums = [1,2,3];
for (var i = 0; i < nums.length; i++) {
    var num = nums[i];

    var elem = document.createElement('div');
    elem.textContent = num;

    elem.addEventListener('click', function() {
        alert(num);
    });

    document.body.appendChild(elem);
}
```

Demo: 03-Patterns/05-CapturingScope

Using a Closure to provide Scope

```
document.body.innerHTML = '';
var nums = [1,2,3];
for (var i = 0; i < nums.length; i++) {
    var num = nums[i];

    var elem = document.createElement('div');
    elem.textContent = num;

    elem.addEventListener('click', function() {
      var n = num;
      return function(){
          alert(n);// n is captured in the closure

      }
    }());

    document.body.appendChild(elem);
}
```

Demo: 03-Patterns/05-CapturingScope

The Variable `this`

- `this` is the context of a function. The value of this is determined by *how a function is called*:
    1. If the function is called as a constructor with the `new` keyword, `this` is a new object
    2. If the method is called via call() or apply(), `this` is explicitly passed
    3. If the function is called as a method of an object, `this` points to that object
    4. If none of the above are used, `this` is the global object (or `undefined` in strict mode).
- Especially for callbacks it might be tricky to find out the value of `this.`

Demo: 02-Language/02-LanguageConstructs/11-This

Pattern: Capturing `this`

- The value of **`this`** depends on how a function is called
- **`this`** might not be the same when passing a callback and when the callback is executed

```
function Controller(){
  this.count = 0;
  this.increment = function(){
    this.count++;
    alert(this.count);
  }
}
```

```
function Controller(){
  var self = this;
  self.count = 0;
  self.increment = function(){
    self.count++;
    alert(self.count);
  }
}
```

```
var ctrl = new Controller();
var button = document.getElementById('button');
button.addEventListener("click", ctrl.increment);
```

Demo: 03-Patterns/04-CapturingThis

EXERCISES

99-LanguageExercises

Questions?

Interested in JavaScript courses?
Visit: http://ivorycode.com/#schulung