

A silhouette photograph of a construction site at sunset. In the foreground, several workers are silhouetted against a bright sky, working on a structure made of vertical rebar or scaffolding. In the background, a large construction crane with its arm extended is silhouetted against the orange and yellow hues of the setting sun.

Modern JavaScript Development

(A Bootcamp for Java Developers)

Mail: jonas.band@ gmail.com

Twitter: [@jbandi](https://twitter.com/jbandi)

**Most of the people writing in JavaScript are not programmers.
They lack the training and discipline to write good programs.**

- Douglas Crockford, 2001

**A very large group of developers still thinks of JavaScript as a
joke.**

- Davy Brion, 2011

We strongly suggest to adopt JavaScript as a first class language.

- ThoughtWorks Technology Radar, 2011

**The ecosystem around JavaScript as a serious application
platform continues to evolve.**

- ThoughtWorks Technology Radar, January 2014

**Through 2014, improved JavaScript performance will begin to
push HTML5 and the browser as a mainstream enterprise
application development environment.**

- Gartner, October 2013

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

- Atwood's Law, 2007

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



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](https://twitter.com/@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

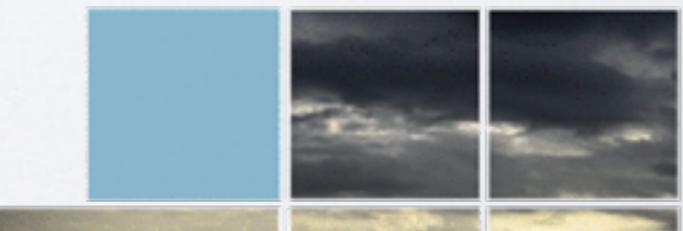
JavaScript: An Overview



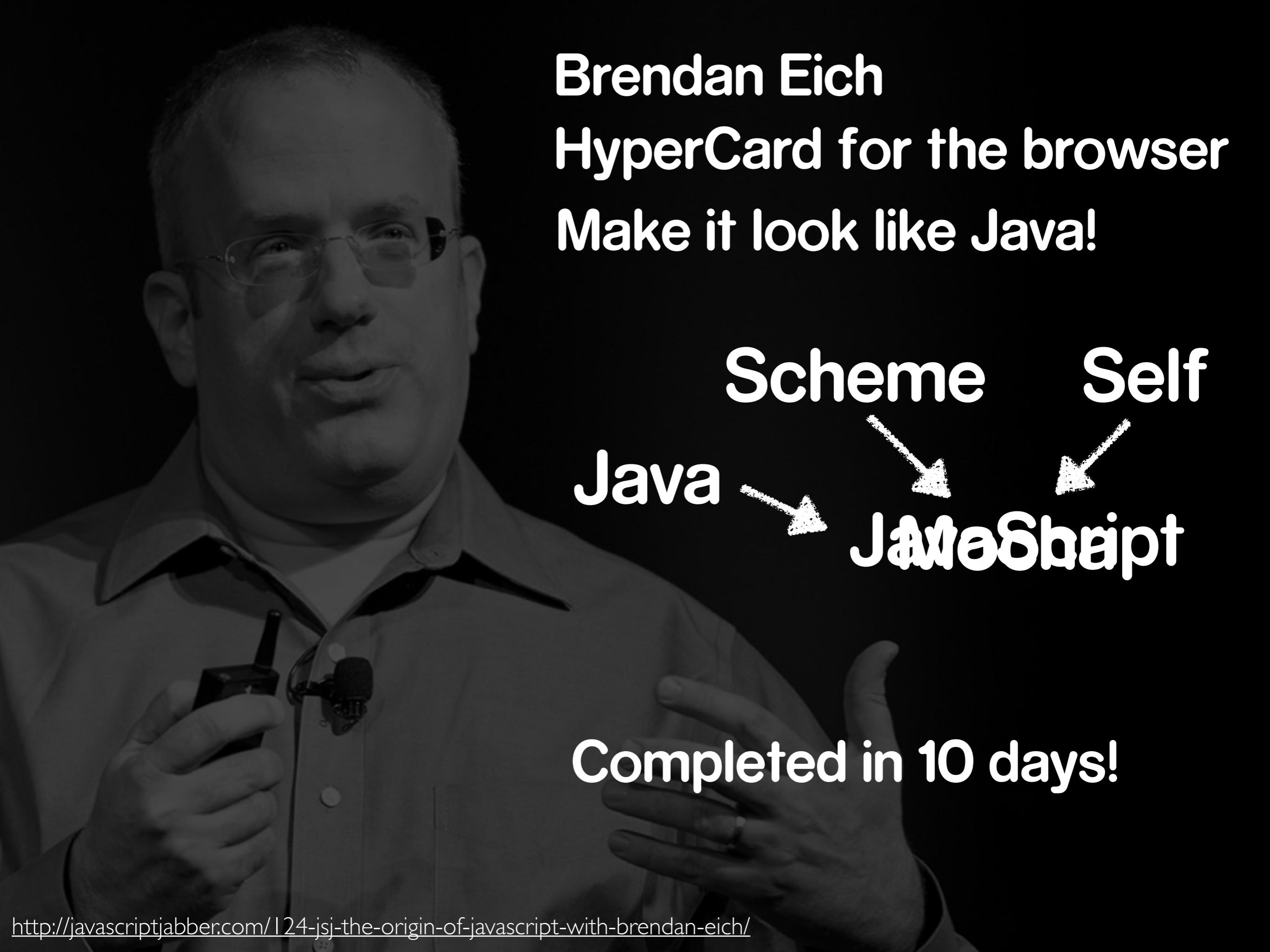
jQuery
write less, do more.



Building
Windows 8 Apps
with **JavaScript**



Windows
Development
Series



Brendan Eich
HyperCard for the browser
Make it look like Java!

Java Scheme Self

↓ ↓ ↓

JavaSript

Completed in 10 days!

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
- Work in Progress (since 2008): ES6 “Harmony” & 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

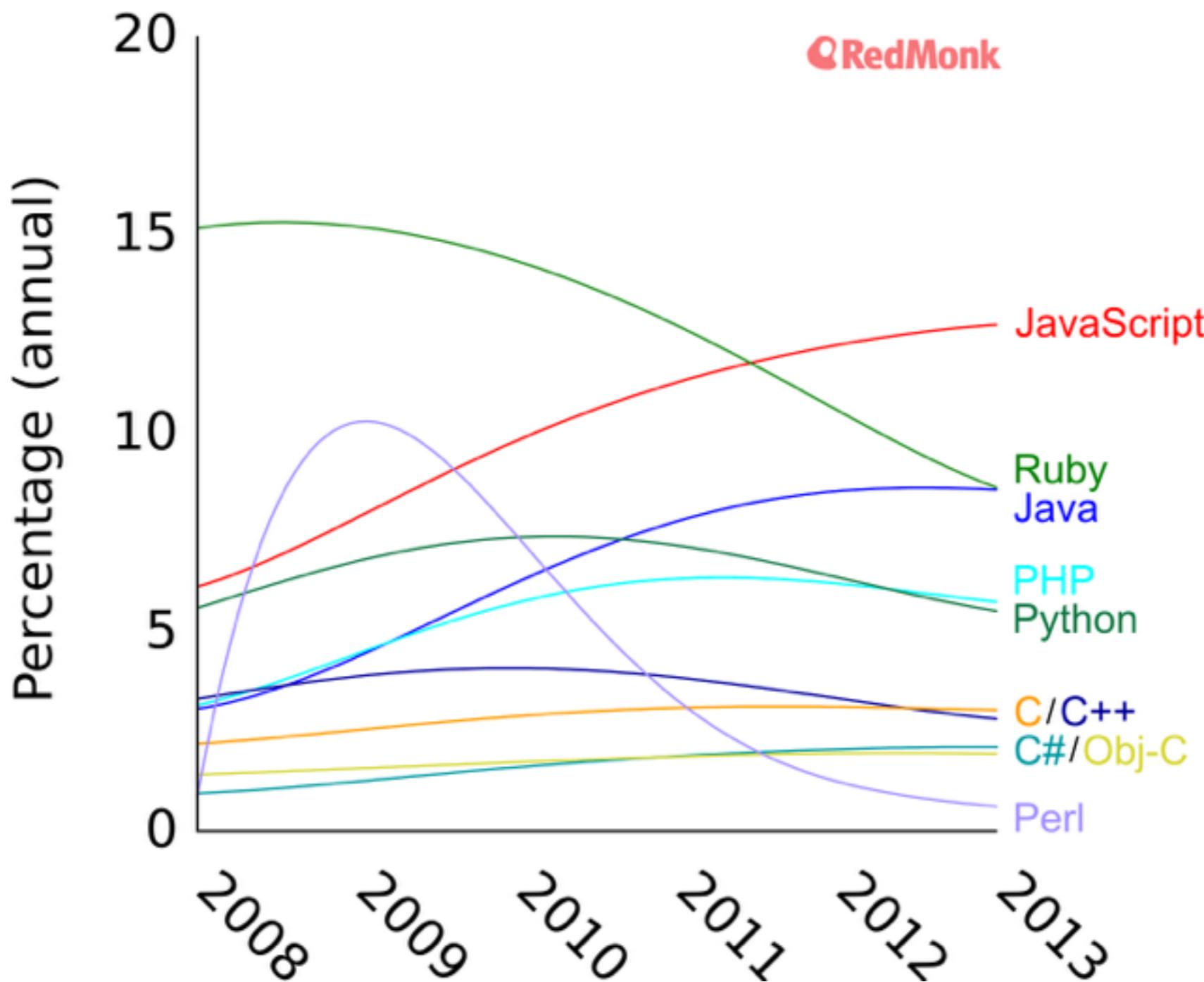
Beyond the Browser



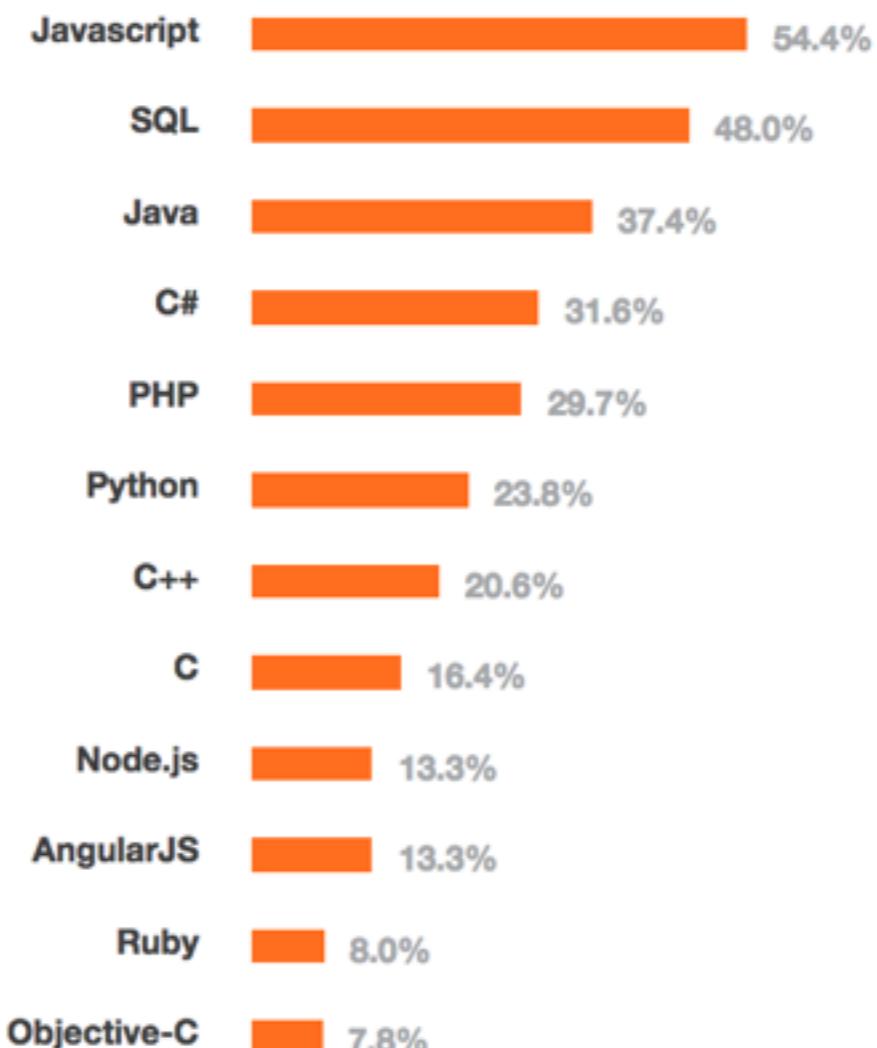
Building
Windows 8 Apps
with JavaScript



New GitHub repositories



2015 2014 2013



21,982 responses

java x 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 x 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# x 784367

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

324 asked today, 4135 this week

LARGE JAVASCRIPT APPLICATIONS



? LoC



? LoC

Cloud9 IDE
Your code anywhere, anytime



? LoC

Lucidchart

ACTIVE ECOSYSTEM



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



write less, do more.

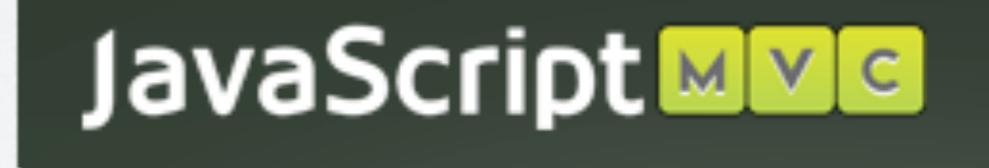


A “Second Generation” of JavaScript Frameworks



Spine

Build Awesome JavaScript
MVC Applications



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



A “Third Generation” of JavaScript Frameworks



ANGULARJS
by Google



Complete Client-Side
Application Development

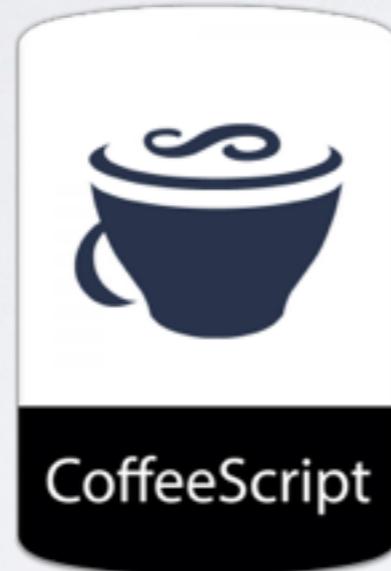


JavaScript as a Compilation Target

"The Assembly Language of the Web"



gwtproject.org



cappuccino-project.org



Clojure Script



Google Traceur



flow
by Facebook

Dart

BABEL
babeljs.io

Hands-On: A naive JavaScript App



IMPROVEMENTS

- Unobtrusive JavaScript
- Separate Code from Markup



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

REFACTORED...

- Using jQuery to abstract Browser Differences
- A professional Build:
 - Linting
 - Packaging (Concatenation, Minification)
 - Testing



Libraries / Frameworks

IDE
Language
dependency management



Toolset
compile
Package

Write Code

Build

Test

Test-Frameworks

Deploy



WS



Language
IDE

Toolset
compile
Package

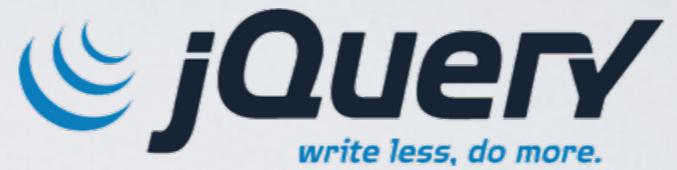


Write Code

Build

Deploy

Test

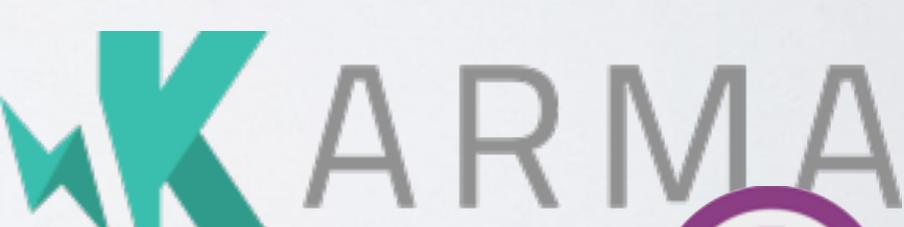


Libraries / Frameworks

dependency management



Bower



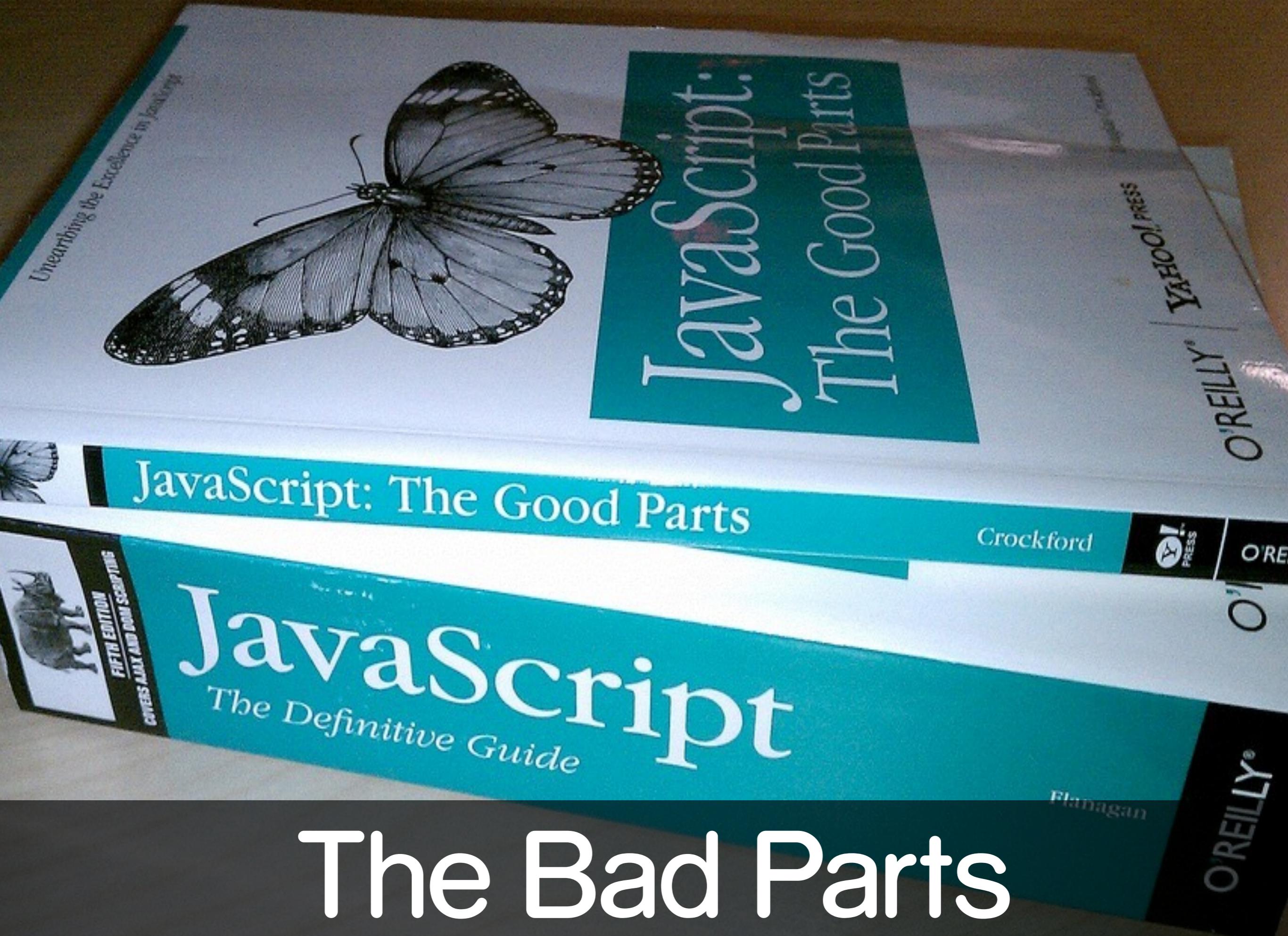
Behavior-Driven JavaScript

The Language

```
function initialize() {
    console = document.getElementById("console");
    canvas = document.getElementById("canvas");
    context = canvas.getContext("2d");

    canvas.width = width;
    canvas.height = height;
```

The Bad Parts



Language Constructs

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

...everything else can
be implemented

- Object
- Function

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());
```

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
```

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();
```

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();
```

Real World



CALCULATOR DEMO

Demo: 03-Patterns/06-RevealingModule/RevealiingModuleDemo

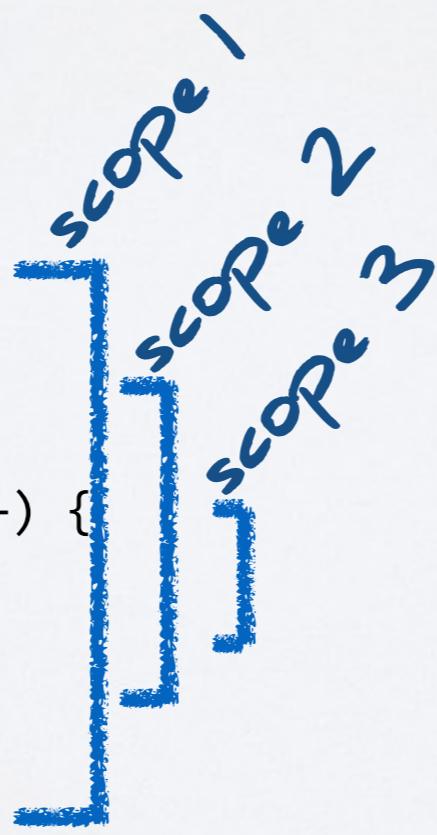
"SPAGHETTI" DEMO

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!

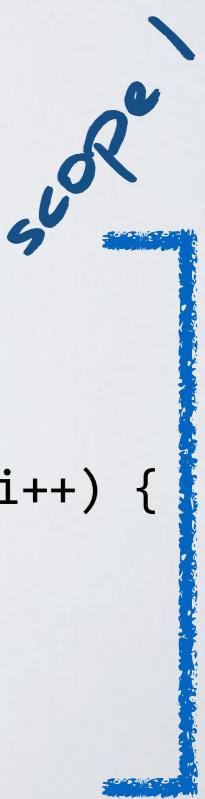
C, Java, C#:

```
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);  
}
```

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);
}
```

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`.

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);
```

EXERCISES



99-LanguageExercises

Questions?



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