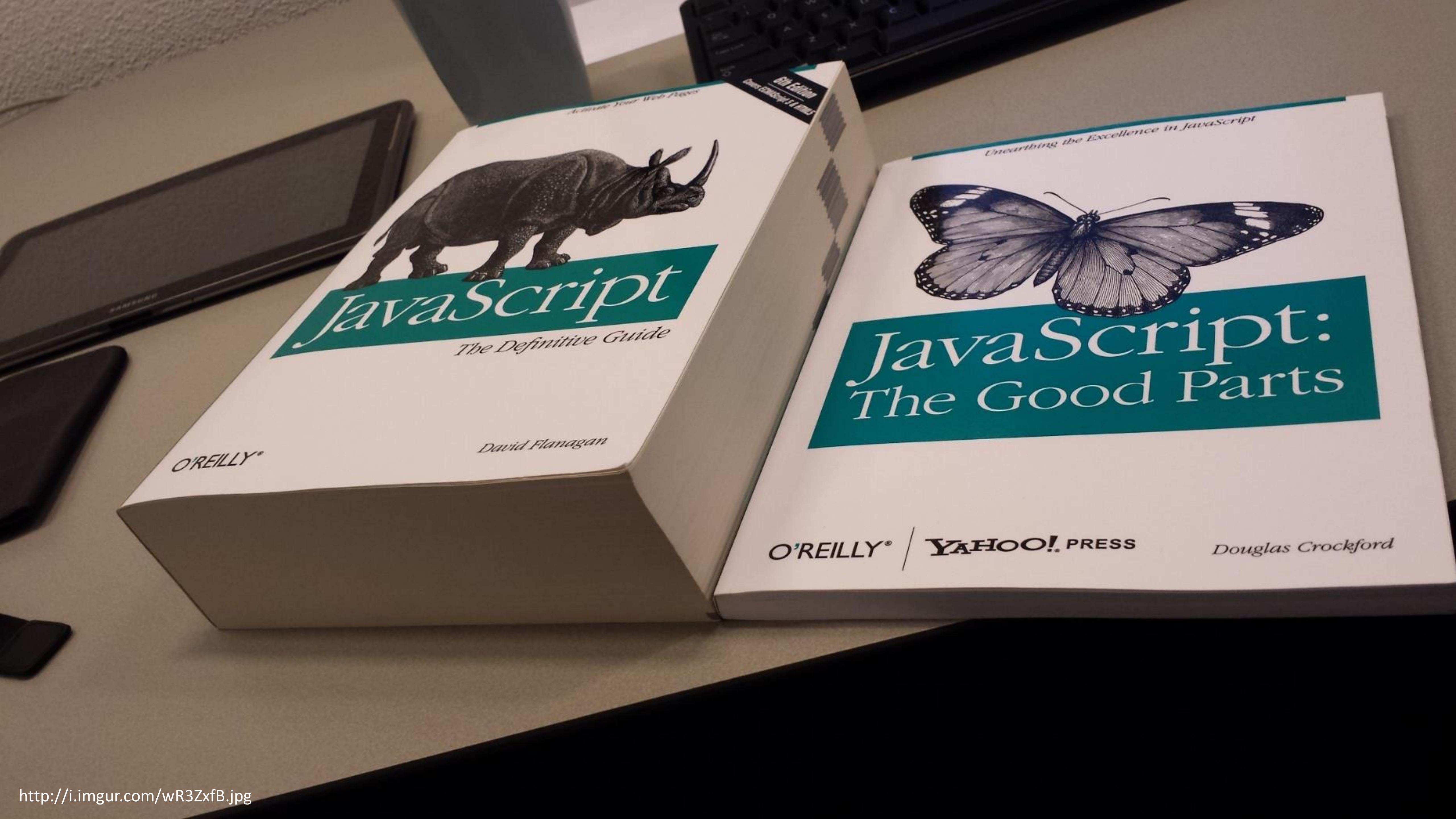
I just met you, and 'this' is crazy, but here's my NaN, so call(me) maybe?

JavaScript is so weird

What makes JavaScript fun?
(and by fun I mean....a little bit strange)

- JavaScript
- Blocks
- Functions
- Null
- Equality
- Truthy/Falsey
- Objects
- this!
- Eval
- parseInt
- NaN
- With
- JSLint/JSHint
- Arrays
- Switches
- more...



http://i.imgur.com/wR3ZxfB.jpg

JavaScript

…is not Java

Java is to JavaScript as car is to carpet

Java is to JavaScript as ham is to hamster

Syntax Error

- Automatic semicolon silliness
  - {}
  - ;

Syntax Error

- Most often, a newline (\n) ends a statement unless...
  - The statement has an unclosed parenthesis ")", array literal, or object literal.
  - The line uses -- or ++
  - Any block statements such as for, while, do, if, or else, and there is no starting bracket "{"
  - After these constructs: break, continue, return, throw

Just say no to single line blocks

One single line

```
if (ok)
 x = true;
```

Can turn into

```
if (ok)
  x = true;
  callFunc();
```

Then looks like

```
if (ok)
  x = true;
  callFunc();
```

In reality it is

```
if (ok) {
x = true;
}
callFunc();
```

Putting the "fun" in functions

```
function functionStatement() {
    // Some code
    return value;
}
var functionExpression = function() {
    // Some code
    return value;
};
```

Putting the "fun" in functions

```
(function () {
    console.log("anonymous function");
})();

(function IIFE() {
    console.log("function expression");
})();
```

Putting the "fun" in functions

```
function main() {
    var x = functionStatement();
    function functionStatement() { … }

    var functionExpression = function() { … }
    functionExpression();
}
```

DEMO

- Fun with functions

Arrays

- There is no such thing
- No dimensions
- No out of bounds errors
- typeof doesn't know the difference

Carry on

- continue statement

  "I've never seen a piece of code that was not improved by removing it"

  -- Crockford

The switch

- Auto fallthrough

```
switch (expression) {
    case expression:
        statements
        [break;]
     case expression:
        statements
        [break;]
    default:
        statements
        [break;]
}
```

Let's get to the truth of the matter

Truthy values

```
'false' (quoted false)
'0'     (quoted zero)
()  (empty functions)
[]  (empty arrays)
{}  (empty objects)
All other values
```

Falsey values

```
false
0    (zero)
''   (empty string)
null
undefined
NaN
```

All things being equal-ish. Maybe. Sort of.

==	===
!=	!==

DEMO

- Truthy and Falsey

Just what's up with *this*, anyway?

Eval - the most evil of all?

Or just a lowly, misunderstood, function?

```
var x = eval("forms[0].option" + optionNumber + ".checked");
```

```
var x = forms[0]["option" + optionNumber].checked;
```

Are you *with* me, or against me?

```
with (objX) {
  // statements
}
```

New, new, don't do

- typed wrappers
  - new object
  - new array
  - new Date

farceInt(fib);

static int parseInt(String s)

static int parseInt(String s, int radix)

parseInt's farce parsing

0X or 0x	16 (hexadecimal)	
0	10 (decimal)	8 (octal)
Everything else	10 (decimal)	

```
var result = parseInt(numericString, 10);
```

NaN

- It claims it's NotANumber, but it's a number.
- Don't use equality operators with NaN
  - Use Number.isNaN() instead
  - Use typeof instead
  - Don't use plain isNaN() – or else nothing is a number!
- ES6 Number.isNaN()

Seems legit

Is legit

```
var add = function() {
    return arguments[0] + arguments[1];
};

console.log(add(4, 4)); // returns 8
```

How to avoid all the odd stuff

- JSLint
- JSHint
- JSFiddle