JAVASCRIPT COURSE

PART ONE

ICT HUB

JS

IN THIS CLASS

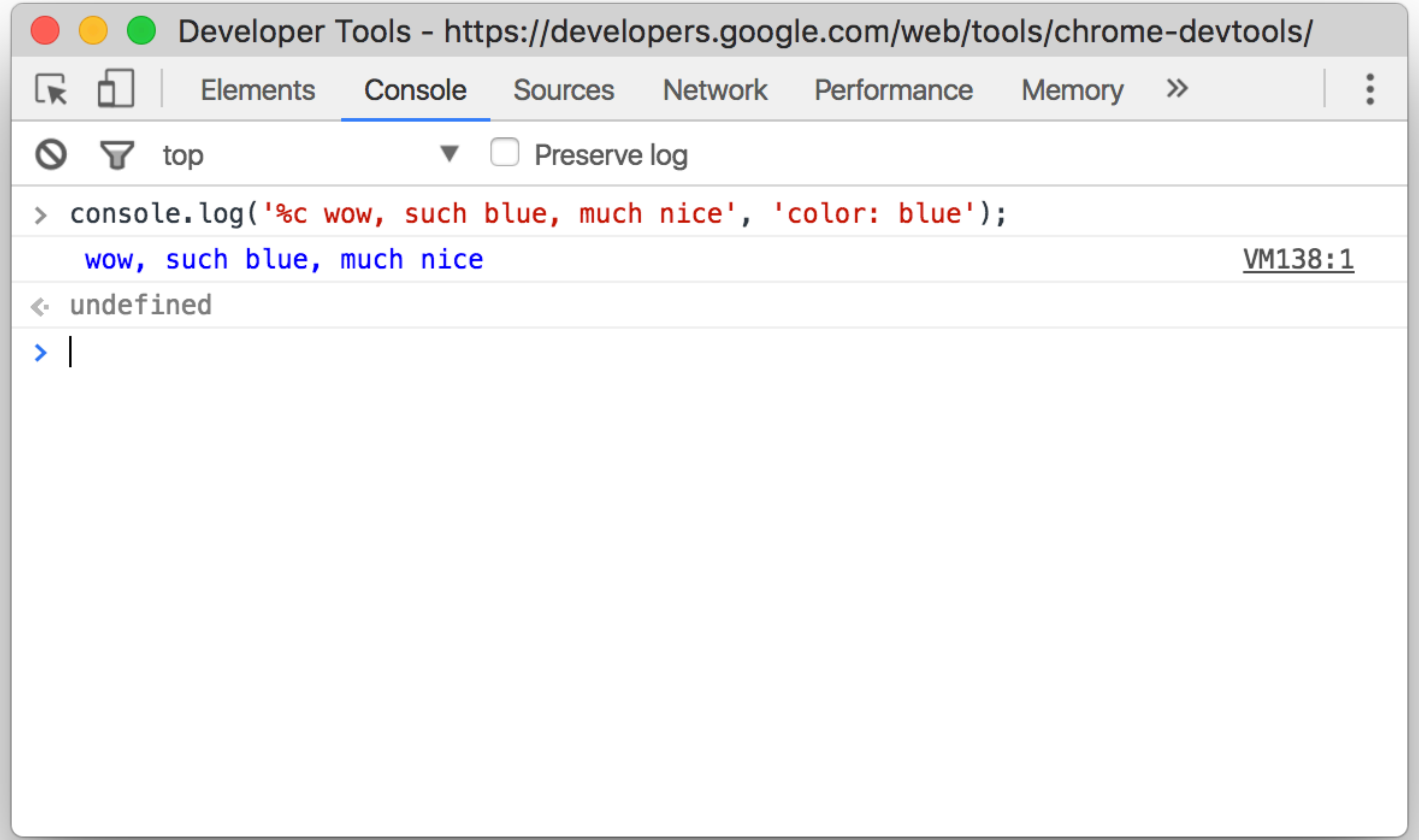
- Variables
- Types
- Operators
- Objects
- Built-in methods
- Value comparison
- Conditionals
- Loops

BEFORE WE BEGIN

Focus on ES6

Run the examples in
Chrome Dev Tools

- Chrome's Main Menu > More Tools > Developer Tools
- Right-click a page element and select Inspect
- Command+Option+I (Mac) or Control+Shift+I (Windows, Linux)


Developer Tools - https://developers.google.com/web/tools/chrome-devtools/
Elements
Console
Sources
Network
Performance
Memory
top
Preserve log

```
> console.log('%c wow, such blue, much nice', 'color: blue');
  wow, such blue, much nice                                    VM138:1
<· undefined
>
```

CODE COMMENTS

// Single line comments

```
/*
  Multi line
  comments
*/
```

VARIABLES

- let — variable (ES 6)
- const — constant (ES 6)
- var — variable

```
const a = 5; // 5
a = 6;       // ❌
```

```
let a = 5; // 5
a = 6;     // 6
```

```
var a = 5; // 5
a = 6;     // 6
```

let vs var

let is block scoped
var is function scoped

TYPES

Static typing

vs

Dynamic (Weak) typing

- String
- Number
- Boolean
- Null
- Undefined
- Symbol (new to ES6)
- Object

Primitive values — immutable (except Object)

OPERATORS

- Assignment (=)
- Math (+, -, * and /)
- Compound Assignment (+=, -=, *= and /=)
- Increment/Decrement (++ and --)
- Equality (==, !=, === and !==)
- Comparison (>, >=, < and <=)
- Logical (&& and ||)
- Object Property Access (.)

OBJECTS

An object is a collection of properties, and a property is an association between a name (or key) and a value.

A key value is either
a String or a Symbol value.

BUILT-IN OBJECTS

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects

ARRAYS

Arrays are list-like objects
whose prototype has methods to perform
traversal and mutation operations.

Neither the length of a JavaScript array nor the types of its elements are fixed.

Arrays cannot use strings as element indexes
(as in an associative array), but must use integers.

JavaScript arrays are zero-indexed:
the first element of an array is at index 0,
and the last element is at the index equal to
the value of the array's length property minus 1.

```
const arr = ['first element', 'second element'];
const mixedArray = ['first element', 2, false];
```

FUNCTIONS

Functions are one of the fundamental building blocks in JavaScript.
A function is a JavaScript procedure, a set of statements, that performs a task or calculates a value.

```
function sum(a, b) {
    return a + b;
}

sum(1, 2); // 3
```

BUILT-IN METHODS

VALUE COMPARISON

COERCION

Type conversion

Explicit and implicit

```
const a = '42';          // '42'
const b = Number(a); // 42
const a = '42';    // '42'
const b = a * 1;  // 42
```

TRUTHY & FALSY

Falsy

- Empty string ('')
- Zero and invalid number (0, −0, NaN)
- null and undefined
- false

Everything else is truthy

EQUALITY

== and ===

Difference between == and ===

!= and !==

http://dorey.github.io/JavaScript-Equality-Table/

More info:
https://github.com/getify/You-Dont-Know-JS
"Types & Grammar", chapter 4

INEQUALITY

<, >, <= and >=

```
41 < 42;                // true
'hello' < 'world';  // true
41 < '42';              // ?
```

CONDITIONALS

Conditionals control behavior in JavaScript and determine whether or not pieces of code can run.

IF … ELSE

The most common type of conditional
is the if statement, which only runs if
the condition enclosed in parentheses () is truthy.

```
if (41 < 42) {
    console.log('Woohoo!'); // Woohoo
}
```

```
if (42 > 41) {
    console.log('Woohoo!');
} else {
    console.log('Nope'); // Nope
}
```

```
if (a > b) {
    console.log('a');
} else if (a < b) {
    console.log('b');
} else {
    console.log('equal');
}
```

TERNARY OPERATOR

```
if (42 > 41) {
    console.log('Woohoo!');
} else {
    console.log('Nope'); // Nope
}
```

```
const ok = (42 > 41) ? 'Woohoo!': 'Nope';
console.log(ok); // Nope
```

expression ? true : false

SWITCH STATEMENT

The switch statement evaluates an expression, matching the expression's value to a case clause, and executes statements associated with that case.

```
switch (fruit) {
    case 'Oranges':
        console.log('Oranges!');
        break;
    case 'Apples':
        console.log('Apples!');
        break;
    default:
        console.log('Other fruit');
}
```

LOOPS

FOR

loops through a block of code N times

```
const fruit = ['apples', 'oranges'];

for (let i = 0; i < fruit.length; i++) {
    console.log(fruit[i]); // apples, oranges
}
```

FOR ... IN

loops through the properties of an object

WHILE

loops through a block of code
while a specified condition is true

```
let i = 0
while (i < 3) {
    text += 'The number is ' + i + ‘\n';
    i++;
}
```

```
// The number is 0
// The number is 1
// The number is 2
```

DO … WHILE

Same as while,
but it will execute the code block once,
before checking if the condition is true

```
let i = 0
do {
    text += 'The number is ' + i + ‘\n';
    i++;
}
while (i < 3);
```

```
// The number is 0
// The number is 1
// The number is 2
```

```
let i = 4
do {
    text += 'The number is ' + i + ‘\n';
    i++;
}
while (i < 3);
```

```
// The number is 4
```

INFINITE LOOPS

```
while (true) {
    console.log('Oops');
}
```

READ MORE

You Don’t know JavaScript

Kyle Simpson

https://github.com/getify/You-Dont-Know-JS

JavaScript: The Definitive Guide

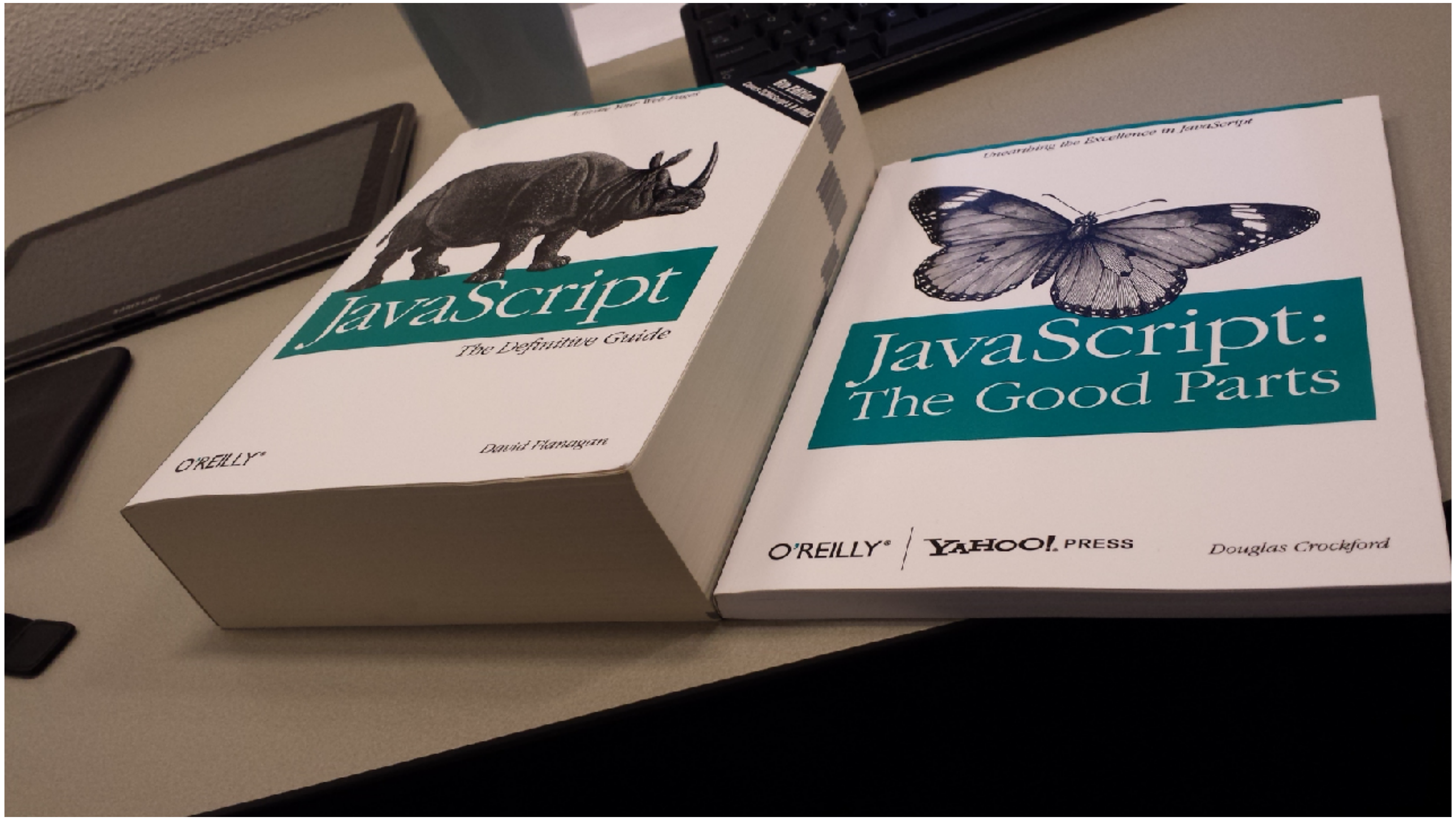
David Flanagan

http://shop.oreilly.com/product/9780596805531.do

JavaScript: The Good Parts

Douglas Crockford

http://shop.oreilly.com/product/9780596517748.do


Activate Your Web Pages
JavaScript
The Definitive Guide
O'REILLY
David Flanagan
Unearthing the Excellence in JavaScript
JavaScript:
The Good Parts
O'REILLY
YAHOO! PRESS
Douglas Crockford

THE END

OF PART ONE