JavaScript Basics

Agenda

Introduction
Language
Functional Program
Objects
OOP

Introduction

What is Javascript?
Origins
ECMAScript Standardization
Uses
Pervasive

Introduction

JS (abbrev) is a high-level, dynamic, untyped, and interpreted run-time language. It has been standardized in the ECMAScript language specification.

Alongside HTML and CSS, JavaScript is one of the three core technologies of World Wide Web content production.

History in brief

1995 Livescript invented by Brendan Eich at Netscape

2005 Jesse James coins Ajax

2008 ECMA Script 5 (ES5) standardized

Careful..

JavaScript™

Uses

Client side scripting
Server side using Node.JS
Utilities on development machine
Even Embedded Systems (RED)

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

Jeff Atwood’s Law

Language

JS Syntax
Statement
Comment
Variables
Operators
Arithmetic
Assignment
Data Types
Functions

JS Syntax

Rules for writing JS Programs
JS is dynamic typed language

Statements are separated by ‘;’
Statements are simple or block

Composed of values, operators, keywords and comments

JS Values

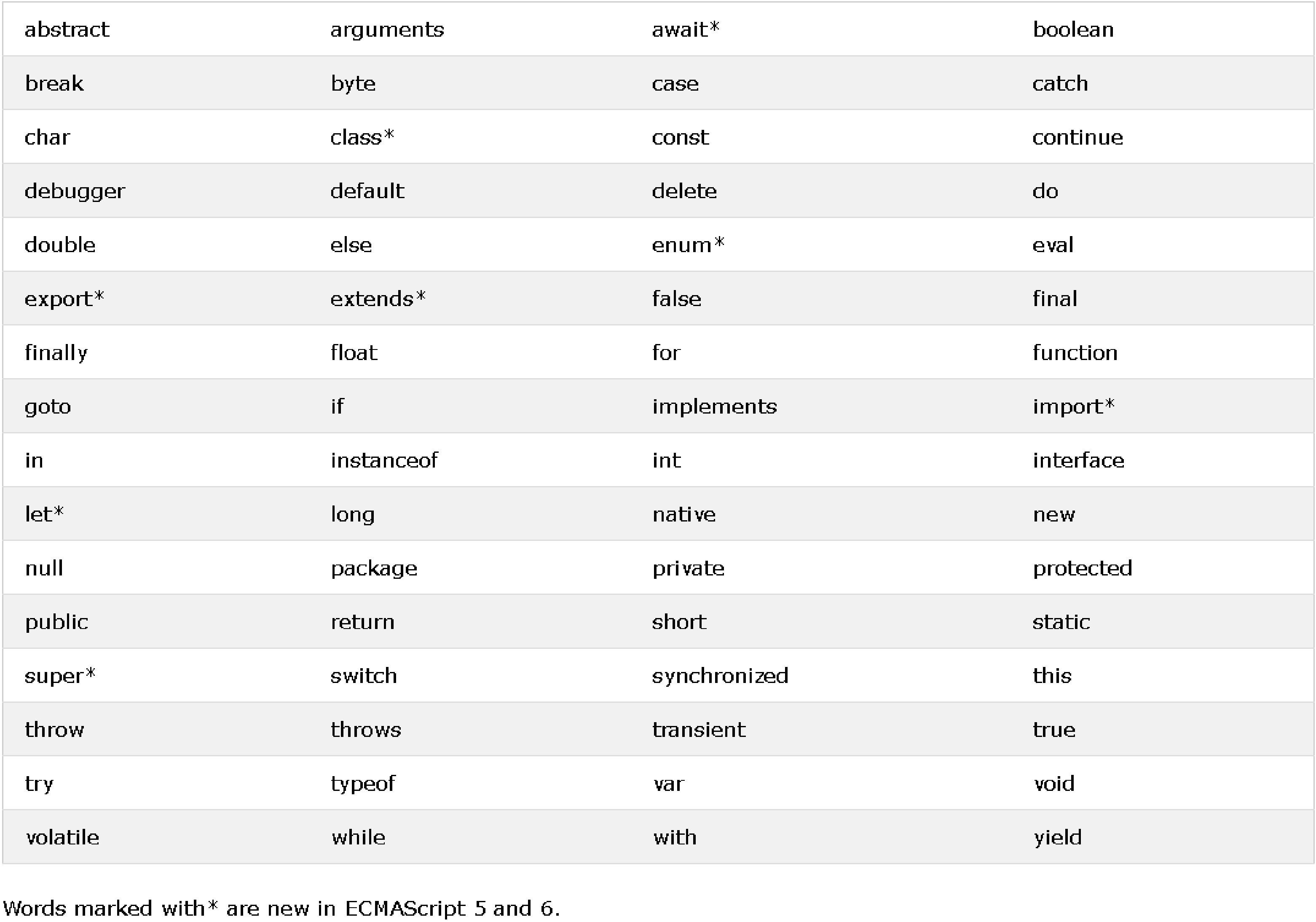
Fixed values – literals
Variable values - vars

Literals – Number, Strings
Vars can hold any type of literals.

Case sensitive var names
Eg data, context, keepScripts, i, j

Coding convention : camelCase

JS reserved words



abstract	arguments	await*	boolean
break	byte	case	catch
char	class*	const	continue
debugger	default	delete	do
double	else	enum*	eval
export*	extends*	false	final
finally	float	for	function
goto	if	implements	import*
in	instanceof	int	interface
let*	long	native	new
null	package	private	protected
public	return	short	static
super*	switch	synchronized	this
throw	throws	transient	true
try	typeof	var	void
volatile	while	with	yield

Words marked with* are new in ECMAScript 5 and 6.

JS Comments

Single Line //
Multi-line /* */

JS Operators

Arithmetic (+, -, *, /, %, ++, --)

Assignment (=, +=, -=, *=, /=, %=)

Bitwise (~, |, &, ^=, <<, >>, >>>)

Logical (||, &&, !)

Comparison (==, !=, >, <, >=, <=, ===, !==)

Ternary ?

Comma ,

JS Data Types

Important concept
Use to process app data
Build complex data structures and algorithms

JS Data Types

Primitive

Number, String, Boolean, NULL, Undefined

typeof of primitive type

typeof of null is object

typeof of undefined is undefined

JS Data Types

Complex

Object, Array, Functions

Typeof returns Object

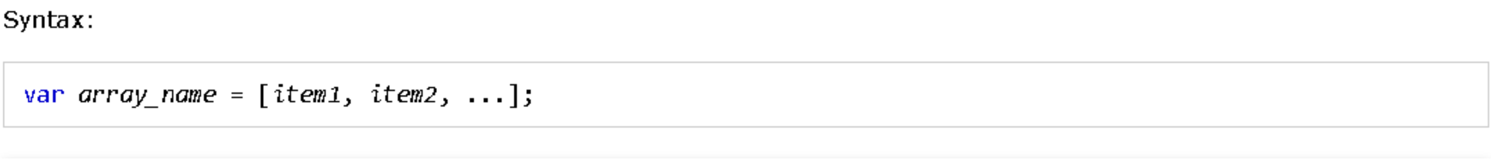
JS Data Types - Objects

Key:value
Length property
Behaviors as functions

```
{ [ nameValuePair1[, nameValuePair2[, ...nameValuePairN] ] ] }
```

JS Data Types - Arrays

Length property



Syntax:

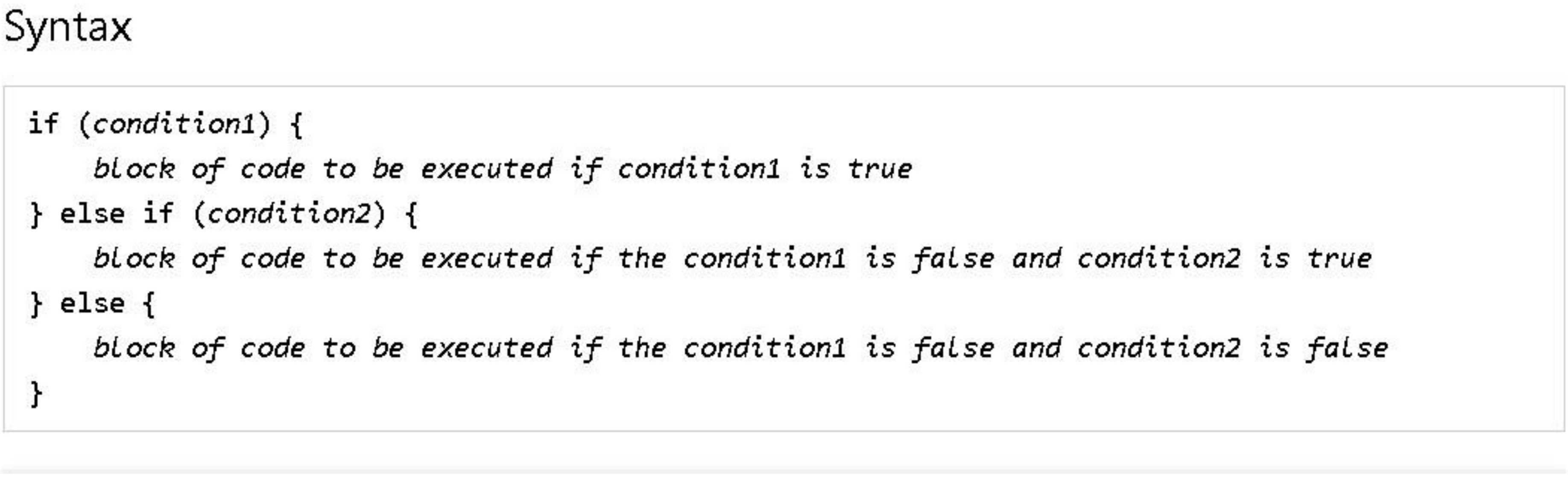
```
var array_name = [item1, item2, ...];
```

JS condition statement

If else

Switch case

JS condition statement

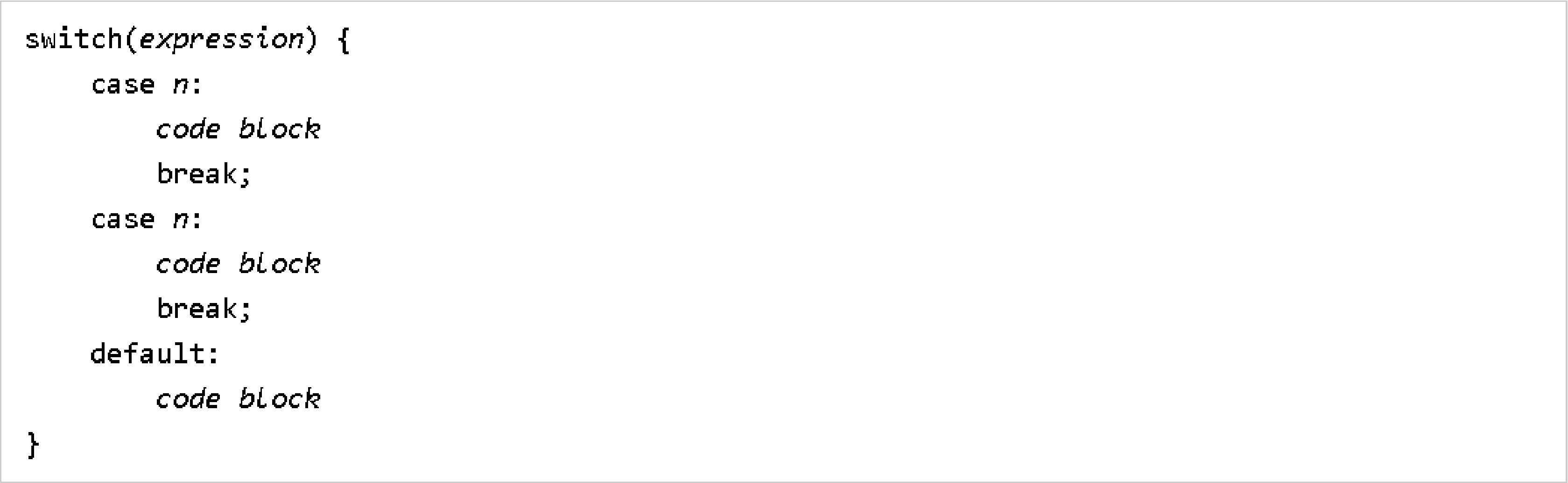


Syntax

```
if (condition1) {
    block of code to be executed if condition1 is true
} else if (condition2) {
    block of code to be executed if the condition1 is false and condition2 is true
} else {
    block of code to be executed if the condition1 is false and condition2 is false
}
```

JS condition statement

Syntax



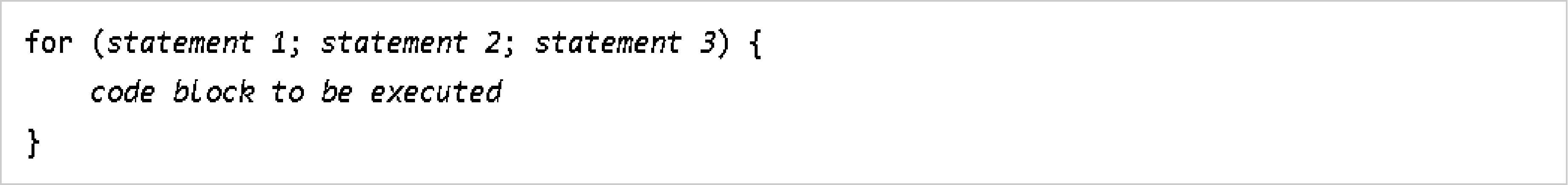
```
switch(expression) {
    case n:
        code block
        break;
    case n:
        code block
        break;
    default:
        code block
}
```

JS Loop construct

for

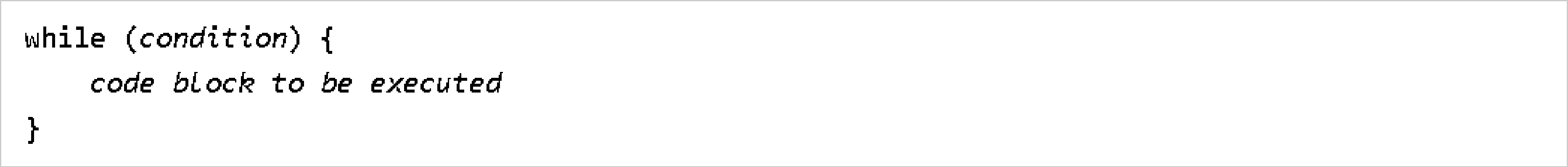
While

JS for Loop construct



```
for (statement 1; statement 2; statement 3) {
    code block to be executed
}
```

JS while Loop construct



```
while (condition) {
    code block to be executed
}
```

JS Loop variations

For / In

Do While

Break

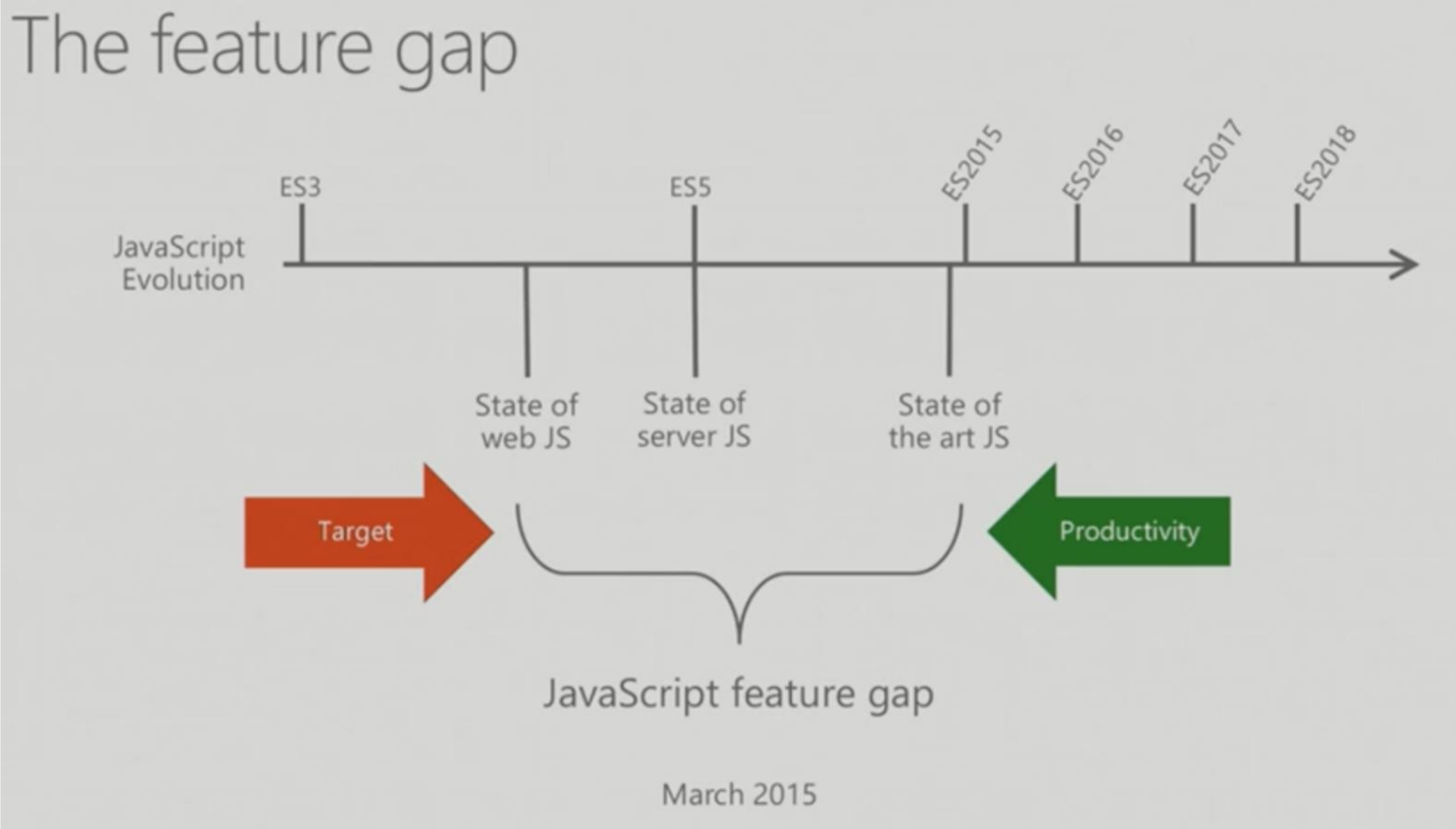
Continue

JS Functions

Function object

OOP in Javascript

Using concept of classes
Objects
Methods
Properties
Uses newer ES6 syntax



The feature gap

JavaScript Evolution

ES3

ES5

ES2015

ES2016

ES2017

ES2018

State of web JS

State of server JS

State of the art JS

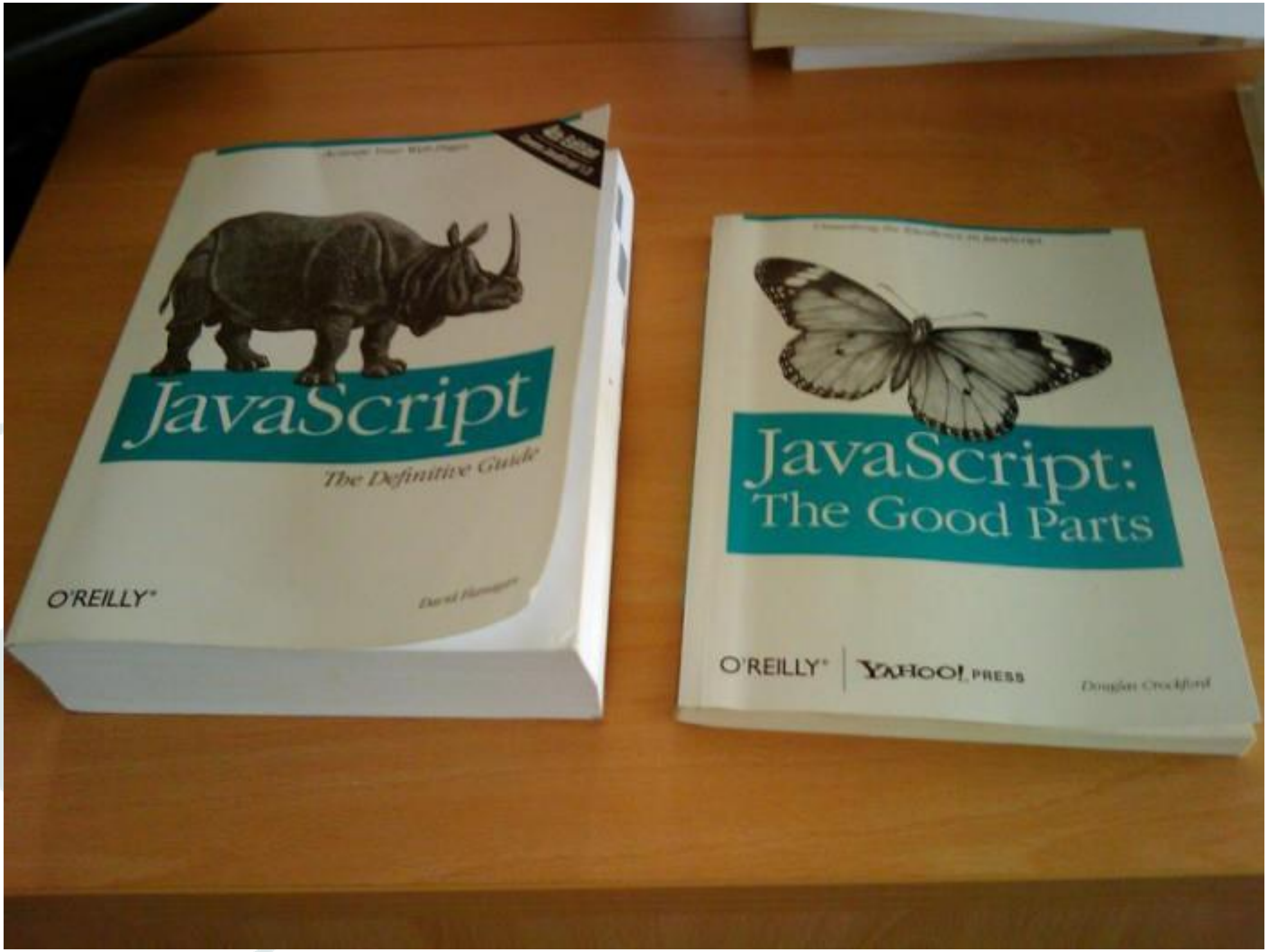
Target

Productivity

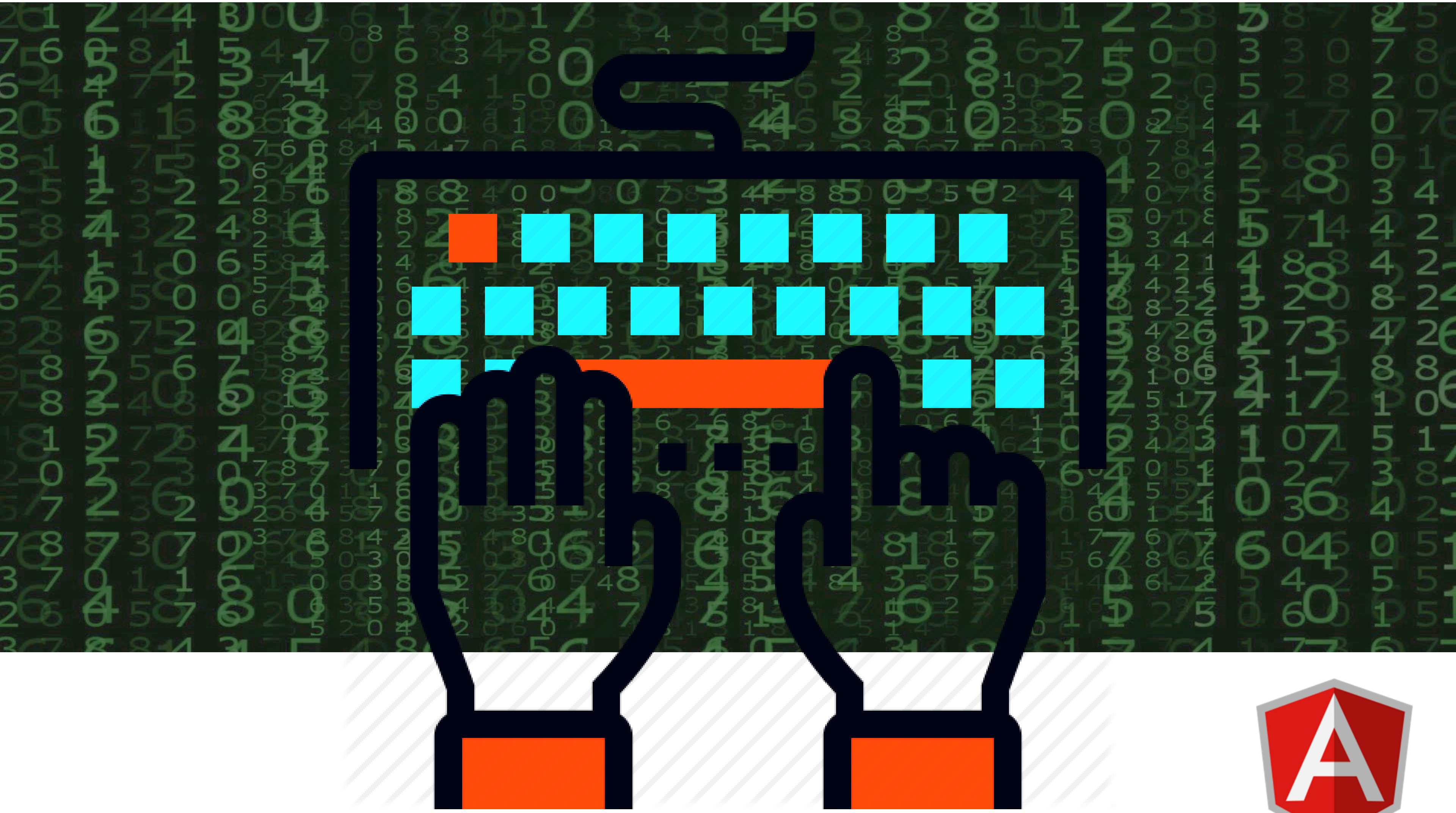
JavaScript feature gap

March 2015

JavaScript References:



Lets code

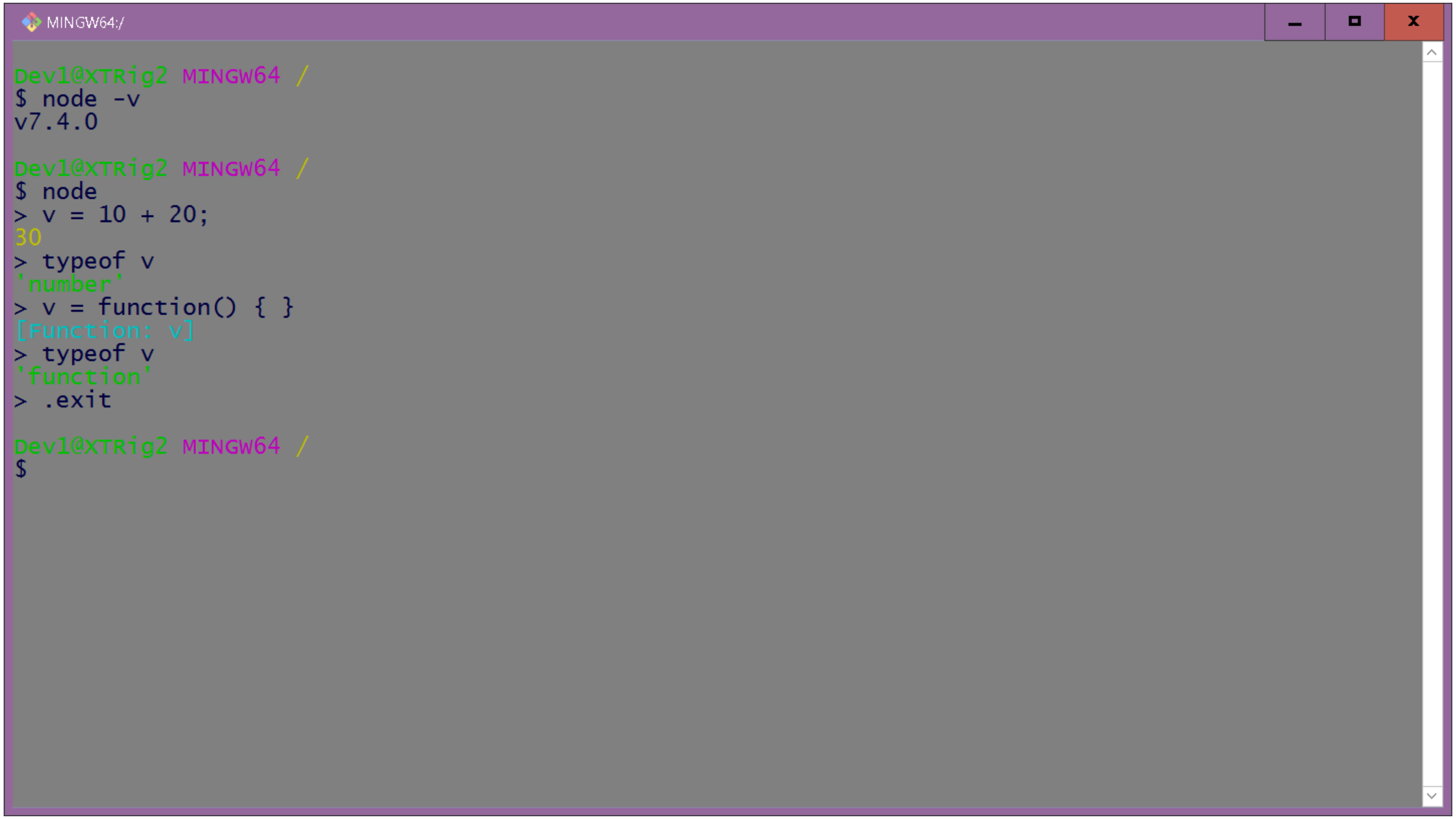


Ways to Run

node REPL

Chrome Debugger

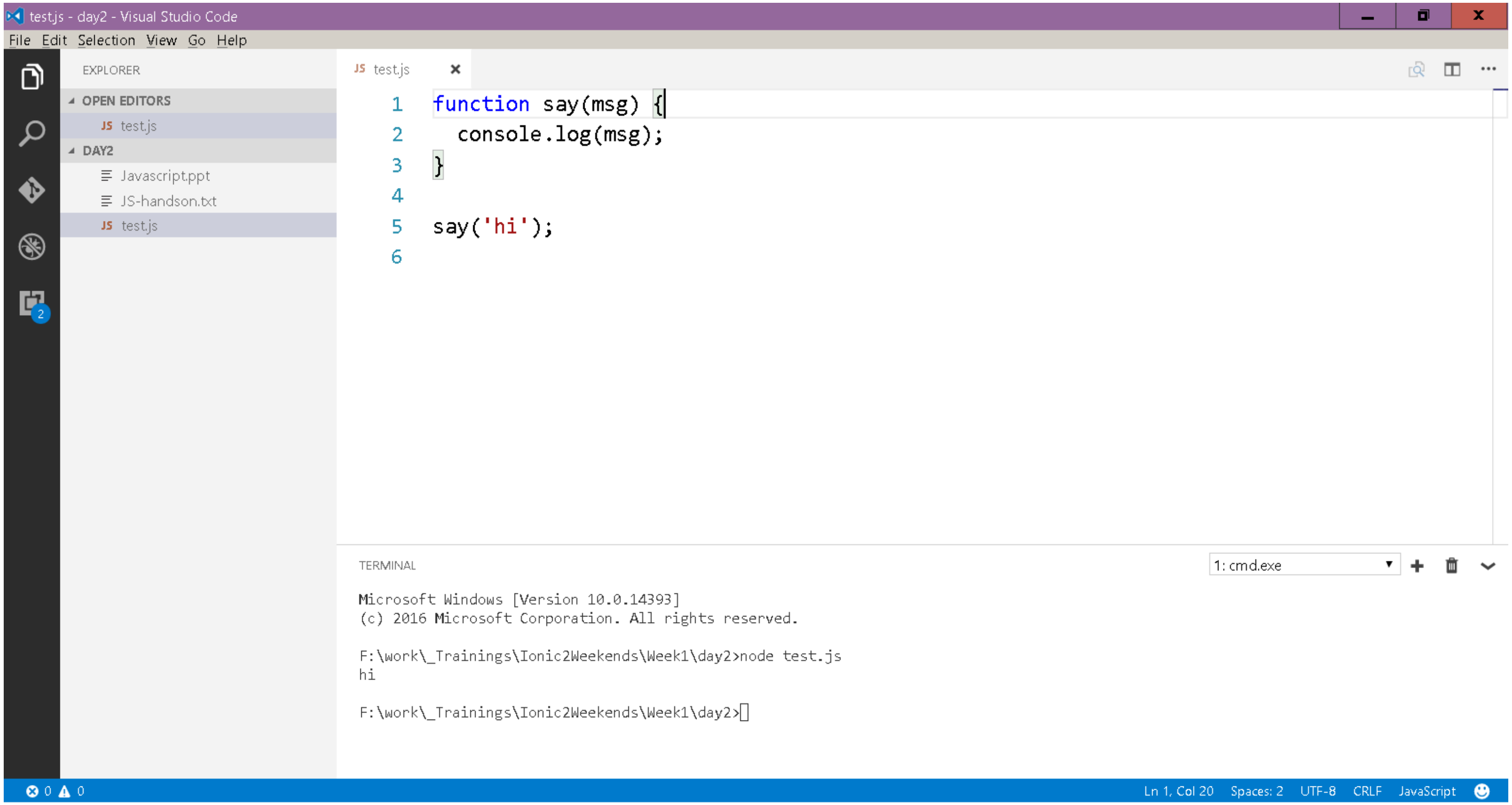
VS Code Terminal



```
Dev1@XTRig2 MINGW64 /
$ node -v
v7.4.0

Dev1@XTRig2 MINGW64 /
$ node
> v = 10 + 20;
30
> typeof v
'number'
> v = function() { }
[Function: v]
> typeof v
'function'
> .exit

Dev1@XTRig2 MINGW64 /
$
```



test.js - day2 - Visual Studio Code
File Edit Selection View Go Help
EXPLORER
OPEN EDITORS
test.js
DAY2
Javascript.ppt
JS-handson.txt
test.js
TERMINAL
1: cmd.exe
Ln 1, Col 20 Spaces: 2 UTF-8 CRLF JavaScript

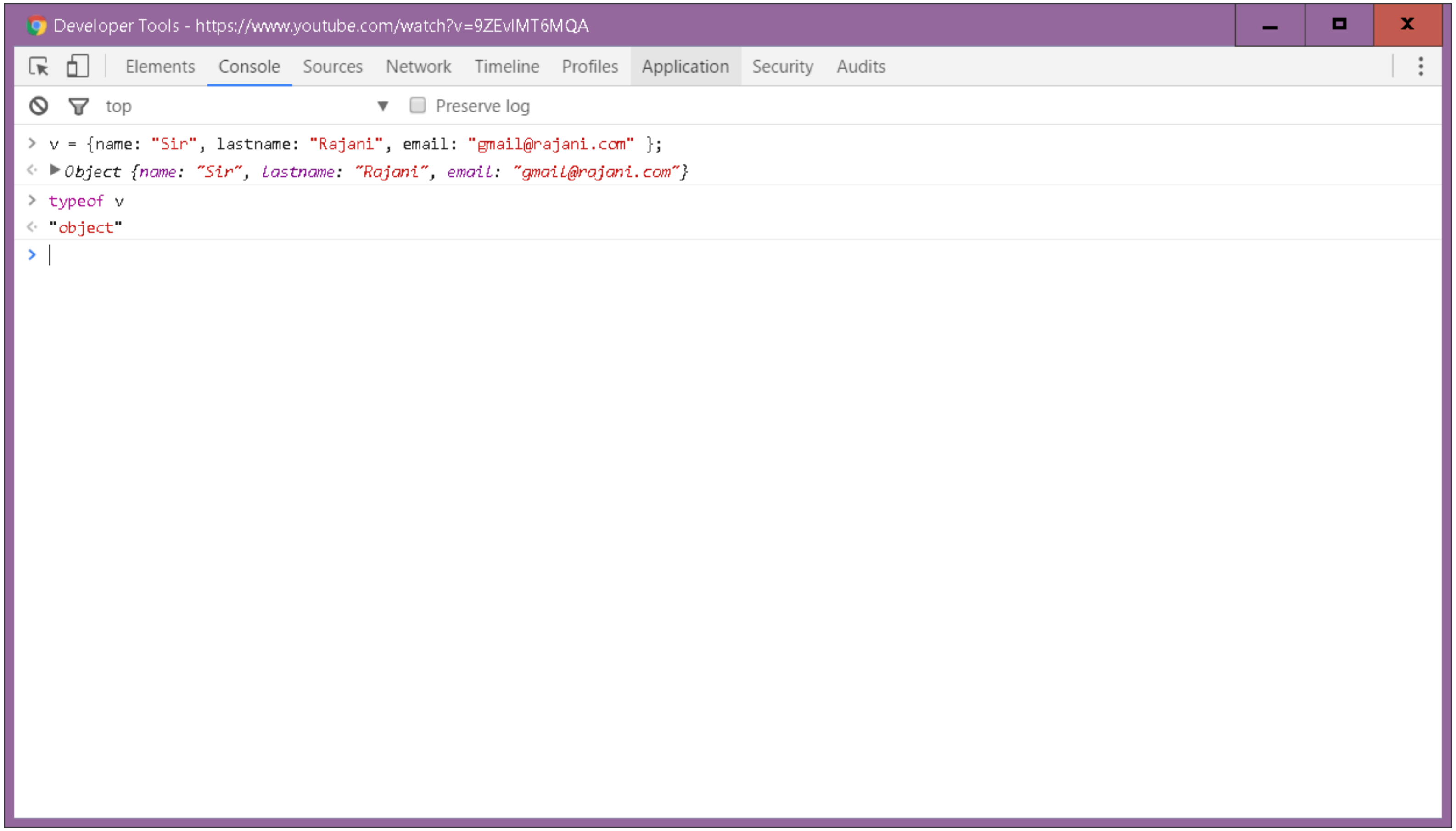
```javascript
function say(msg) {
  console.log(msg);
}

say('hi');
```

```
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

F:\work\_Trainings\Ionic2Weekends\Week1\day2>node test.js
hi

F:\work\_Trainings\Ionic2Weekends\Week1\day2>
```



```
> v = {name: "Sir", lastname: "Rajani", email: "gmail@rajani.com" };
< ▶Object {name: "Sir", lastname: "Rajani", email: "gmail@rajani.com"}
> typeof v
< "object"
>
```

JS 5 – Features

use strict;

multiline strings

new array methods - isArray(), Array.prototype.filter(), Array.prototype.map()

JS Programming – Issues

Spaghetti Programming
Globals

Dynamic data – no type safety!

Difficult to use for large applications

JS ES6 [Correctly ES2015] – Features

const PI = 3.1415;

block scoping - let i;

Arrow functions

Destructuring

Classes

Arrow Functions

```
(v => { return v*v; })
```

Lexical this

```
this.nums.forEach((v) => {
    if (v % 5 === 0)
       this.fives.push(v)
})
```

Destructuring

Lexical this

```
this.nums.forEach((v) => {
    if (v % 5 === 0)
      this.fives.push(v)
})
```

Classes

Class & Object
Constructor
Properties
Methods
Static properties and methods
this keyword

Note: there are no private properties

ES6

ES6 modules, classes & objects

App structuring

Structure an app
Learn about data & services

Thank You!