

The Future of JavaScript – ES6

Rami Sayar - @ramisayar

Technical Evangelist

Microsoft Canada

Agenda

- What's ECMAScript6?
- Learning about Block Scoping
- Learning about Destructuring
- Learning about Iterators and Generators

Assumptions

- Working knowledge of JavaScript and HTML5.

Note: Slides will be made available. Follow @ramisayar.

I've seen the

FUTURE

It's in my

BROWSER





code. ship. repeat.

WORDCAMP VANCOUVER - @RAMISAYAR

Why Should You Care?

JAVASCRIPT



ALL THE THINGS

What is ECMAScript?

- ECMAScript is the scripting language standardized by Ecma International as ECMA-262.
- ECMAScript implementations include JavaScript, JScript and ActionScript.
- Most commonly used as the basis for client-side scripting on the Web => JavaScript.

Where is ECMAScript Now?

Edition	Date Published	Notes
1	June 1997	First edition.
2	June 1998	Editorial changes. Aligning with ISO standard.
3	December 1999	Added regex, string handling, new control statements, try/catch, etc.
4	ABANDONED	
5	December 2009	Strict mode subset, clarification, harmonization between real-world and the spec. Added support for JSON and more object reflection.
5.1	June 2011	Aligning with ISO standard.
6	June 2015	NEW SYNTAX
7	WIP	Very early stage of development.

ECMAScript 6

- Significant update to the language and major JavaScript engines are implementing features as we speak.
- Also known as ES6 or ECMAScript 2015 or Harmony or JavaScript 2015.
- ES6 draft was ratified on June 17, 2015. Read it here: <http://www.ecma-international.org/ecma-262/6.0/>
- Status Tables:
 - [Kangax](#)
 - [ES6 Matrix](#) by Thomas Lahn:

ECMAScript 6

Note: The below might be out of date by the time you read it.

ECMAScript [5](#) **6** [7](#) [intl](#) [non-standard](#) [compatibility table](#)

Please note that *some of these tests* represent **existence**, not functionality or full conformance.

Sort by number of features?

Show obsolete platforms?

Show unstable platforms?

■ V8
 ■ SpiderMonkey
 ■ JavaScriptCore
 ■ Chakra
 ■ Carakan
 ■ kJS
 ■ Other

● Useful feature (1 point)
 ● Significant feature (2 points)
 ● Landmark feature (4 points)

Feature name	Current browser	Compilers/polyfills						Desktop browsers										
		Traceur	Babel + core-js ⁽¹⁾	Closure	JSX ⁽²⁾	Type-Script	es6-shim	IE 10	IE 11	Edge ⁽³⁾	FF 31 ESR	FF 38	FF 39	FF 40	CH 43, OP 30 ⁽⁴⁾	CH 44, OP 31 ⁽⁴⁾	CH 45, OP 32 ⁽⁴⁾	SF 6.1 SF 7
Optimisation																		
• proper tail calls (tail call optimisation)	0/2	0/2	1/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2
Syntax																		
• default function parameters	0/6	3/6	5/6	4/6	0/6	3/6	0/6	0/6	0/6	0/6	3/6	3/6	3/6	3/6	0/6	0/6	0/6	0/6
• rest parameters	5/5	4/5	4/5	2/5	3/5	3/5	0/5	0/5	0/5	5/5	3/5	4/5	4/5	4/5	0/5	0/5	0/5	0/5
• spread (...) operator	10/12	12/12	12/12	2/12	1/12	2/12	0/12	0/12	0/12	10/12	8/12	12/12	12/12	12/12	0/12	0/12	0/12	0/12
• object literal extensions	6/6	6/6	6/6	4/6	5/6	5/6	0/6	0/6	0/6	6/6	0/6	6/6	6/6	6/6	3/6	6/6	6/6	0/6
• for...of loops	5/8	8/8	8/8	5/8	1/8	2/8	0/8	0/8	0/8	5/8	4/8	6/8	6/8	6/8	6/8	6/8	6/8	0/8
• octal and binary literals	4/4	2/4	4/4	2/4	0/4	2/4	0/4	0/4	0/4	4/4	2/4	4/4	4/4	4/4	4/4	4/4	4/4	0/4
• template strings	3/3	3/3	3/3	3/3	3/3	3/3	0/3	0/3	0/3	2/3	0/3	3/3	3/3	3/3	3/3	3/3	3/3	0/3
• RegExp "y" and "u" flags	1/2	1/2	1/2	0/2	0/2	0/2	0/2	0/2	0/2	1/2	1/2	1/2	1/2	1/2	0/2	0/2	0/2	0/2
• destructuring	0/32	26/32	29/32	16/32	12/32	21/32	0/32	0/32	0/32	0/32	17/32	23/32	23/32	23/32	0/32	0/32	0/32	0/32
• Unicode code point escapes	2/2	1/2	1/2	1/2	0/2	1/2	0/2	0/2	0/2	2/2	0/2	0/2	0/2	1/2	0/2	2/2	2/2	0/2
• new.target	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2
Bindings																		
• const	8/8	6/8	6/8	6/8	0/8	4/8	0/8	0/8	8/8	8/8	3/8	8/8	8/8	8/8	5/8	5/8	5/8	1/8
• let	8/10	8/10	8/10	8/10	0/10	6/10	0/10	0/10	8/10	8/10	0/10	0/10	0/10	0/10	5/10	5/10	5/10	0/10
• block-level function declaration ⁽¹³⁾	Yes	Yes	Yes	Yes	No	No	No	No	Yes	Yes	No	No	No	No	Yes	Yes	Yes	No
Functions																		
• arrow functions	9/11	9/11	9/11	8/11	7/11	8/11	0/11	0/11	0/11	9/11	7/11	7/11	8/11	8/11	0/11	0/11	0/11	0/11
• class	0/23	16/23	19/23	7/23	15/23	14/23	0/23	0/23	0/23	0/23	0/23	0/23	20/23	20/23	0/23	0/23	0/23	0/23

Getting ES6

- ES6 in the Browser
 - Microsoft Edge (Windows 10) has excellent ES6 support!
 - Go to `about:flags` & turn on “Enable experimental JavaScript features” flag
 - Chrome Canary
 - Go to `chrome://flags` & turn on “Enable Experimental JavaScript”
 - Firefox Nightly or Firefox Developer Edition

Getting ES6

- ES6 in Node.js: `--v8-options` flag

```
node --v8-options | grep harmony
```

```
--harmony_typeof #(enable harmony semantics for typeof)
```

```
--harmony_scoping #(enable harmony block scoping)
```

```
--harmony_modules #(enable harmony modules (implies block scoping))
```

```
--harmony_proxies #(enable harmony proxies)
```

```
--harmony_collections #(enable harmony collections (sets, maps, and weak maps))
```

```
--harmony #(enable all harmony features (except typeof))
```

Let's take a look!

ES6 – Block Scoping

ES6 – Block Scoping

```
var foo = 'JSOpenDay';  
console.log(foo); // Prints 'JSOpenDay'  
if (true) {  
    var foo = 'BAR';  
    console.log(foo); // Prints 'BAR'  
}  
console.log(foo); // Prints 'BAR'
```




[Challiyil Eswaramangalath Pavithran Vipin](#)

[Attribution-ShareAlike 2.0 Generic License](#)



ES6 – Block Scoping

- Scoping in JS is at the function-level for variables and functions.

```
var foo = 'JS';  
console.log(foo); // Prints 'JS'  
if (true) {  
  var foo = 'BAR';  
  console.log(foo); // Prints 'BAR'  
}  
console.log(foo); // Prints 'BAR'
```

```
var foo;  
foo = 'JS';  
console.log(foo); // Prints 'JS'  
if(true) {  
  foo = 'BAR';  
  console.log(foo); // Prints 'BAR'  
}  
console.log(foo); // Prints 'BAR'
```

ES6 – Block Scoping

- ‘Hoisting’ in JavaScript

```
var foo = 'JS';
if(!bar) {
  console.log(foo + ' ' + bar);
  // Prints 'JS undefined'
}
var bar = '2015';
console.log(foo + ' ' + bar);
// Prints 'JS 2015'
```

```
var foo, bar;
foo = 'JS';
if(!bar) {
  console.log(foo + ' ' + bar);
  // Prints 'JS undefined'
}
bar = '2015';
console.log(foo + ' ' + bar);
// Prints 'JS 2015'
```

- Variables are ‘hoisted’ to the top even if they will never be executed in any statement.
- You can enforce ‘hoisting’ syntactically with JSLint ‘onevar’.

ES6 – Block Scoping

- Scoping in JS is at the function-level for variables and functions.

```
var foo = 'JS';  
console.log(foo); // Prints 'JS'  
if (true) {  
  var foo = 'BAR';  
  console.log(foo); // Prints 'BAR'  
}  
console.log(foo); // Prints 'BAR'
```

```
var foo;  
foo = 'JS';  
console.log(foo); // Prints 'JS'  
function foobar() {  
  var foo = 'BAR';  
  console.log(foo); // Prints 'BAR'  
}  
foobar();  
console.log(foo); // Prints 'JS'
```

ES6 – Block Scoping

- ES6 introduces 'let' & 'const'.
- Variables declared with 'let' are scoped to the block statement.
- This is inline with other C-like languages like Java, C++, etc.

ES6 – Block Scoping

- Variable 'foo' declared with 'let'.

```
let foo = 'JS';  
console.log(foo); // Prints 'JS'  
if (true) {  
  let foo = 'BAR';  
  console.log(foo); // Prints 'BAR'  
}  
console.log(foo); // Prints 'JS'
```

- No 'hoisting' of variables when declared with 'let'.

ES6 – Block Scoping

- Variable 'foo' declared with 'const' is also scoped to the block.

```
const foo = 'JS';  
console.log(foo); // Prints 'JS'  
if (true) {  
  let foo = 'BAR';  
  console.log(foo); // Prints 'BAR'  
}  
// foo = 'BAR';  
// The above line causes "SyntaxError: Assignment to constant variable."  
console.log(foo); // Prints 'JS'
```

ES6 – Destructuring

ES6 – Destructuring

- Destructuring is a syntax feature that allows you to pattern match values to variables or properties thus extracting data.

```
var [foo, bar, ABC] = ['bar', 'foo', 3];  
console.log(foo + ' ' + bar + ' ' + ABC);  
// Prints 'bar foo 3'
```

```
var foo = 'bar', bar = 'foo', ABC = 3;  
console.log(foo + ' ' + bar + ' ' + ABC);  
// Prints 'bar foo 3'
```


ES6 – Destructuring

- Can be used to swap variables without a temporary variable.



[Mary Shattock](#)

[Attribution-ShareAlike 2.0 Generic License](#)

ES6 – Destructuring

- Can be used to swap variables without a temp var... like in Python.

```
var [foo, bar] = ['bar', 'foo'];  
[foo, bar] = [bar, foo];  
console.log(foo + ' ' + bar);  
// Prints 'foo bar'
```

ES6 – Destructuring

- Destructuring is a syntax feature that allows you to pattern match values to variables or properties.
- Not limited to arrays, you can apply destructuring to objects.

ES6 – Destructuring

```
// Simple example without assigning new names
var {x, y} = {x: "X", y: "Y"};
console.log(x); // X
console.log(y); // Y

// getTalk() returns -> { speaker: { name: "Rami" },
title: "Future of JS"}

var { title: talk_title, speaker: { name: speaker_name
}} = getTalk();
console.log(talk_title); // "Future of JS"
console.log(speaker_name); // "Rami"
```

ES6 – Destructuring

- Destructuring is a syntax feature that allows you to pattern match values to variables or properties.
- Can be used to name parameter positions, **AWESOME!**

```
function g({name: x}) {  
  console.log(x);  
}  
g({name: 5})
```

ES6 – Destructuring

- Destructuring is a syntax feature that allows you to pattern match values to variables or properties.

```
// Fail-soft destructuring
```

```
var [a] = [];
```

```
a === undefined;
```

```
// Fail-soft destructuring with defaults
```

```
var [a = 1] = [];
```

```
a === 1;
```

ES6 – Iterators & Generators

ES6 – Iterators & Generators

- Object that knows how to access the **next** item in a collection.
- Iterators provide next() function that returns an object containing two properties, done && value:

```
{  
  done: true, // true|false  
  value: undefined // any variable  
}
```

ES6 – Iterators

```
function makeIterator(array) {  
  var nextIndex = 0;  
  return {  
    next: function() {  
      return nextIndex < array.length ?  
        {value: array[nextIndex++], done: false} :  
        {done: true};  
    }  
  }  
}
```

ES6 – Iterators

- You can also use the `for... of` loop (This is different from `for... in`).
- You can also use iterators with arrays.

```
var evangelists = ['@ramisayar', '@tommylee'];  
for (let item of iterator)  
  console.log(item);  
  // prints "@ramisayar" etc.
```

ES6 – Generators

- Generators are factories for iterators. They are functions that continue execution on `next()` and save their context for re-entrances.
- Generators introduce **function *** and **yield**.
- Generators can replace callbacks.

ES6 – Generators

```
function *foo () {  
  var x = 2;  
  while(true) {  
    x = x * x;  
    yield x;  
  }  
}  
var answer = foo ();  
answer.next ().value; // 4  
answer.next ().value; // 16
```

ES6 – Generators

```
function *foo() {  
  var x = 1, next = 1;  
  while(true) {  
    x = x * next;  
    next = yield x;  
  }  
}
```

```
var answer = foo(); answer.next().value; //1  
answer.next(2).value; //2 answer.next(2).value; //4
```

ES6 – Modules & Classes



I Am Developer

@iamdeveloper



Follow

I think I've had milk last longer than some JavaScript frameworks.



RETWEETS

1,397

FAVORITES

875



6:22 AM - 4 Dec 2014

On Frameworks...

ES6 – Classes

```
class SkinnedMesh extends THREE.Mesh {  
  constructor(geometry, materials) {  
    super(geometry, materials);  
  
    this.idMatrix = SkinnedMesh.defaultMatrix();  
    this.bones = [];  
    this.boneMatrices = [];  
    // ...  
  }  
}
```

ES6 – Classes

```
update(camera) {  
    //...  
    super.update();  
}  
get boneCount() {  
    return this.bones.length;  
}  
set matrixType(matrixType) {  
    this.idMatrix = SkinnedMesh[matrixType]();  
}
```

ES6 – Classes

```
static defaultMatrix() {  
    return new THREE.Matrix4();  
}  
}
```

ES6 – Modules

- Modularization in software architecture is extremely important.
- Plenty of attempts to implement modules in JavaScript. CommonJS and Asynchronous Module Definition (AMD) patterns had 100s of different implementations.
- Node.js had a built-in module system.
- ES6 Modules borrow the best concepts from CommonJS and AMD.

ES6 – Modules

- The implementation is defined by the JavaScript host.
- Implicitly asynchronous loading.
- Two keywords: “import” & “export”

ES6 – Modules

```
// math.js
export var pi = 3.141593;

export function add(num1, num2) {
  return num1 + num2;
}

export function circle_area(r) {
  return pi * r * r;
}
```

ES6 – Modules

```
// app.js
import * as math from "math";
alert("2π = " + math.add(math.pi, math.pi));
```

```
// otherApp.js
import {circle_area, pi} from "math";
alert("Area of Circle with Radius of 5 = " +
circle_area(5));
```

ES6 – Module Loading API

- Programmatically load modules like in AMD with `system.import`
- Why?
 - Customize how modules are mapped to files.
 - Automatically lint modules on import.
 - Automatically compile CoffeeScript/TypeScript on import.
 - Use other module systems...



[Eva Blue](#)

[Attribution 2.0 Generic](#)

WORDCAMP VANCOUVER - @RAMISAYAR

ES6 – Who Has It?

Note: The below might be out of date by the time you read it.

	IE11	Edge	FF40	Chrome 43	Node	io.js
const	8/8	8/8	8/8	5/8	1/8	5/8
let	8/10	8/10	Flag	5/10	0/10	5/10
block-level function declaration	Yes	Yes	No	Yes	Flag	Yes
destructuring	0/32	0/32	23/32	0/32	0/32	0/32
classes	0/23	Flag	20/23	Flag	0/23	Flag
generators	0/22	Flag	19/22	15/22	Flag	15/22

Source: <http://kangax.github.io/compat-table/es6>

Going Back In Time

- Google Traceur, ES6 Compiler: <https://github.com/google/traceur-compiler>
- Babel, ES6 Compiler: <https://babeljs.io/>
 - Special support for JSX & React
 - Support for extensions and plugins
- Continuum, ES6 Virtual Machine written with ES3: <https://github.com/Benvie/continuum>
 - Theoretically, support goes all the way back to IE6.



[JD Hancock](#)

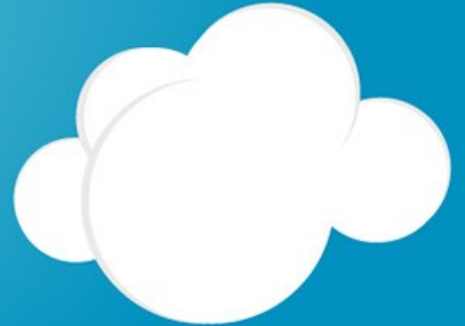
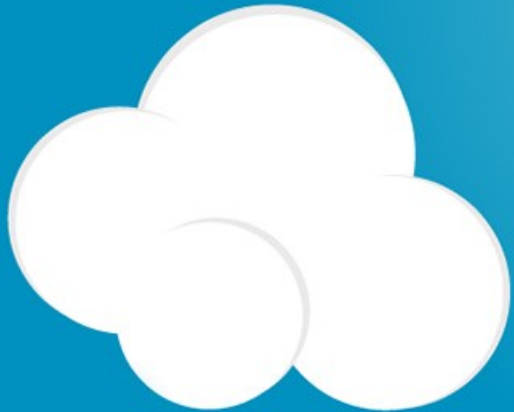
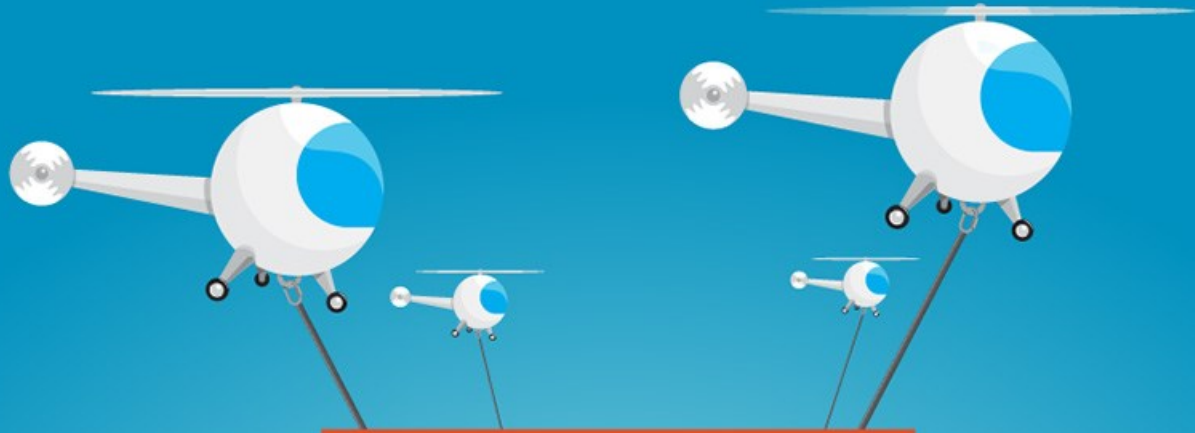
[Attribution 2.0 Generic](#)



Back to the Future

- **xto6**, convert JavaScript to ES6:
<https://github.com/mohebifar/xto6>
- **es6-shim**, adding support for ES6:
<https://github.com/paulmillr/es6-shim>
- **es6-module-loader**, module loader support:
<https://github.com/ModuleLoader/es6-module-loader>

Blank white rectangular area at the top of the page.



Blank white rectangular area at the bottom of the page.

dev.modern.ie

Dev tools for the modern web

Check and debug your site for compatibility with these cross platform tools during development



Virtual machines

Test versions of IE from 6 through 11 using virtual machines you download and manage locally.



RemoteIE

Free test service using Azure RemoteApp to run IE on your Windows, Mac, iOS or Android device.



Quick site scan

Run a quick static code scan on any URL to check for out-of-date libraries, layout issues and accessibility.



Generate screenshots

See how your site renders across 9 common browsers and devices.

What did we learn?

- What's ECMAScript6?
 - Block Scoping
 - Destructuring
 - Modules and Classes
 - Iterators and Generators
-
- There is plenty more in ES6!

Thank You! Questions?

tw: [@ramisayar](https://twitter.com/ramisayar) | gh: [@sayar](https://github.com/sayar)

gist.github.com/sayar/d8f64a80d3a410ba5cba

Resources, References, Links

- [ES6 Compatibility Table](#)
- [ES6 Browser Support](#)
- [What's new in JavaScript?](#)
- [An introduction to ES6 Part 1: Using ES6 Today](#)
- [An introduction to ES6 Part 2: Block Scoping](#)
- [An introduction to ES6 Part 3: Destructuring](#)
- [Tracking ECMAScript 6 Support](#)
- [ES6 \(a.k.a. Harmony\) Features Implemented in V8 and Available in Node](#)
- [React Introduces Support for ES6 Classes](#)

Resources, References, Links

- [ECMAScript 6 Features - Introduction](#)
- [ECMAScript 6 modules: the final syntax](#)
- [The Basics Of ES6 Generators](#)
- [ECMAScript 6 and Block Scope](#)
- [Understanding ES6 Generators](#)
- [MDN - Iterators and generators](#)
- [Classes in JavaScript ES6](#)
- [ECMAScript 6 modules: the future is now](#)

Resources, References, Links

- [es6-shim](#)
- [es6-module-loader](#)
- [Continuum](#)
- [Xto6](#)
- [Koa.js](#)
- [Babel.js](#)
- [traceur-compiler](#)



Microsoft

©2013 Microsoft Corporation. All rights reserved. Microsoft, Windows, Office, Azure, System Center, Dynamics and other product names are or may be registered trademarks and/or trademarks in the U.S. and/or other countries. The information herein is for informational purposes only and represents the current view of Microsoft Corporation as of the date of this presentation. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information provided after the date of this presentation. MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS PRESENTATION.