How to start writing apps with ES6, Angular 1.x and JSPM

by Martin Micunda

blog: http://martinmicunda.com

github: martinmicunda

twitter: @martinmicunda

ECMAScript 6

ECMAScript 6

- Lots of new features (modules, arrow function, generator, classes, promises etc.)

ECMAScript 6

- Lots of new features (modules, arrow function, generator, classes, promises etc.)
- Features are frozen

ECMAScript 6

- Lots of new features (modules, arrow function, generator, classes, promises etc.)
- Features are frozen
- ES6 is fully compatible with ES5

ECMAScript 6

- Lots of new features (modules, arrow function, generator, classes, promises etc.)
- Features are frozen
- ES6 is fully compatible with ES5
- End of 2014 - specification was finished

ECMAScript 6

- Lots of new features (modules, arrow function, generator, classes, promises etc.)
- Features are frozen
- ES6 is fully compatible with ES5
- End of 2014 - specification was finished
- March 2015 - publication process starts

ECMAScript 6

- Lots of new features (modules, arrow function, generator, classes, promises etc.)
- Features are frozen
- ES6 is fully compatible with ES5
- End of 2014 - specification was finished
- March 2015 - publication process starts
- **June 2015** - formal publication!

ES6 Modules

ES6 Modules

- The future replacement for the AMD and CommonJS module formats.

ES6 Modules

- The future replacement for the AMD and CommonJS module formats.
- The ES6 Specification defines two primary module features:
  - module syntax
  - module loader

ES6 Module Syntax

ES6 Module Syntax

Named exports (several per module)

```
//------ lib.js ------
export const sqrt = Math.sqrt;
export function square(x) {
    return x * x;
}
export function diag(x, y) {
    return sqrt(square(x) + square(y));
}
```

ES6 Module Syntax

Named exports (several per module)

```
//------ lib.js ------
export const sqrt = Math.sqrt;
export function square(x) {
    return x * x;
}
export function diag(x, y) {
    return sqrt(square(x) + square(y));
}

//------ main.js ------
import { square, diag } from 'lib';
console.log(square(11)); // 121
console.log(diag(4, 3)); // 5
```

ES6 Module Syntax

Named exports (several per module)

```
//------ lib.js ------
export const sqrt = Math.sqrt;
export function square(x) {
    return x * x;
}
export function diag(x, y) {
    return sqrt(square(x) + square(y));
}

//------ main.js ------
import { square, diag } from 'lib';
console.log(square(11)); // 121
console.log(diag(4, 3)); // 5

//------ main.js ------
import * as lib from 'lib';
console.log(lib.square(11)); // 121
console.log(lib.diag(4, 3)); // 5
```

ES6 Module Syntax

Named exports (several per module)

```
//------ lib.js ------
export const sqrt = Math.sqrt;
export function square(x) {
    return x * x;
}
export function diag(x, y) {
    return sqrt(square(x) + square(y));
}

//------ main.js ------
import { square, diag } from 'lib';
console.log(square(11)); // 121
console.log(diag(4, 3)); // 5

//------ main.js ------
import * as lib from 'lib';
console.log(lib.square(11)); // 121
console.log(lib.diag(4, 3)); // 5
```

Default exports (one per module)

```
//------ myFunc.js ------
export default function () { ... };
```

ES6 Module Syntax

Named exports (several per module)

```
//------ lib.js ------
export const sqrt = Math.sqrt;
export function square(x) {
    return x * x;
}
export function diag(x, y) {
    return sqrt(square(x) + square(y));
}

//------ main.js ------
import { square, diag } from 'lib';
console.log(square(11)); // 121
console.log(diag(4, 3)); // 5

//------ main.js ------
import * as lib from 'lib';
console.log(lib.square(11)); // 121
console.log(lib.diag(4, 3)); // 5
```

Default exports (one per module)

```
//------ myFunc.js ------
export default function () { ... };

//------ main1.js ------
import myFunc from 'myFunc';
myFunc()
```

ES6 Module Syntax

Named exports (several per module)

```
//------ lib.js ------
export const sqrt = Math.sqrt;
export function square(x) {
    return x * x;
}
export function diag(x, y) {
    return sqrt(square(x) + square(y));
}

//------ main.js ------
import { square, diag } from 'lib';
console.log(square(11)); // 121
console.log(diag(4, 3)); // 5

//------ main.js ------
import * as lib from 'lib';
console.log(lib.square(11)); // 121
console.log(lib.diag(4, 3)); // 5
```

Default exports (one per module)

```
//------ myFunc.js ------
export default function () { ... };

//------ main1.js ------
import myFunc from 'myFunc';
myFunc()

//------ MyClass.js ------
export default class { ... };

//------ main2.js ------
import MyClass from 'MyClass';
let inst = new MyClass();
```

ES6 Module Syntax

Named exports (several per module)

```
//------ lib.js ------
export const sqrt = Math.sqrt;
export function square(x) {
    return x * x;
}
export function diag(x, y) {
    return sqrt(square(x) + square(y));
}

//------ main.js ------
import { square, diag } from 'lib';
console.log(square(11)); // 121
console.log(diag(4, 3)); // 5

//------ main.js ------
import * as lib from 'lib';
console.log(lib.square(11)); // 121
console.log(lib.diag(4, 3)); // 5
```

Default exports (one per module)

```
//------ myFunc.js ------
export default function () { ... };

//------ main1.js ------
import myFunc from 'myFunc';
myFunc()

//------ MyClass.js ------
export default class { ... };

//------ main2.js ------
import MyClass from 'MyClass';
let inst = new MyClass();
```

http://www.2ality.com/2014/09/es6-modules-final.html

Dr. Axel Rauschmayer

ES6 Module Loader

ES6 Module Loader

```
//------ module.js ------
export function square(x) {
    return x * x;
}

//------ index.html ------
<!doctype html>
    <script>
        System.import('module').then(function(module) {
          console.log(module.square(5));
        });
    </script>
```

ES6 Module Loader

```
//------ module.js ------
export function square(x) {
    return x * x;
}

//------ index.html ------
<!doctype html>
    <script>
        System.import('module').then(function(module) {
          console.log(module.square(5));
        });
    </script>
```

http://whatwg.github.io/loader/

ES6 TODAY

ES6 TODAY

May I use ES6 today?

ES6 TODAY

May I use ES6 today?

YES

ES6 TODAY

May I use ES6 today?

YES

How?

ES6 TODAY

May I use ES6 today?

YES

How?

ES6 Compilers (Traceur or 6to5)

ES6 Module Loader

SystemJS

JSPM

Traceur

ES6

```
var seattlers = [
  for (c of customers) {
    if (c.city == "Seattle") {
      name: c.name,
      age: c.age
    }
  }
];
```

ES5

```
var seattlers = (function() {
  var c;
  var $__20 = 0,
      $__21 = [];
  for (var $__22 = customers[
      $traceurRuntime
      .toProperty(Symbol.iterator)](),
      $__23; !($__23 = $__22.next()).done;){
    c = $__23.value;
    if (c.city == "Seattle")
      $traceurRuntime
      .setProperty($__21, $__20++, {
        name: c.name,
        age: c.age
      });
  }
  return $__21;
}());
```

6to5

ES6

```
var seattlers = [
  for (c of customers) {
    if (c.city == "Seattle") {
      name: c.name,
      age: c.age
    }
  }
];
```

ES5

```
var seattlers = Array.from(customers)
  .filter(function (c) {
  return c.city == "Seattle";
}).map(function (c) {
  return {
    name: c.name,
    age: c.age
  };
});
```

ES6 Module Loader Polyfill

- “Dynamically loads ES6 modules in browsers and NodeJS with support for loading existing and custom module formats through loader hooks.”
- It follows JavaScript Loader Spec

SystemJS

- “Universal dynamic module loader - loads ES6 modules, AMD, CommonJS and global scripts in the browser and NodeJS”
- map, paths and global shims (similar to requireJS)
- plugins (CSS, JSON and images)
- ES6 Module Loader polyfill + SystemJS = 16KB (minified and gzipped)

SystemJS

```
//------ hello-amd.js ------
define([], function() {
  return {
    say: function() {
      return 'AMD says HI!';
    }
  };
});
```

SystemJS

```
//------ hello-amd.js ------
define([], function() {
  return {
    say: function() {
      return 'AMD says HI!';
    }
  };
});

//------ hello-cjs.js ------
var obj = {
  say: function speak(msg) {
    return 'CommonJS says HI!';
  }
};
module.exports = obj;
```

SystemJS

```
//------ hello-amd.js ------
define([], function() {
  return {
    say: function() {
      return 'AMD says HI!';
    }
  };
});

//------ hello-cjs.js ------
var obj = {
  say: function speak(msg) {
    return 'CommonJS says HI!';
  }
};
module.exports = obj;

//------ hello-es6.js ------
export class Hello {
  say() {
    return 'ES6 says HI!';
  }
}
```

SystemJS

```
//------ hello-amd.js ------
define([], function() {
  return {
    say: function() {
      return 'AMD says HI!';
    }
  };
});

//------ hello-cjs.js ------
var obj = {
  say: function speak(msg) {
    return 'CommonJS says HI!';
  }
};
module.exports = obj;

//------ hello-es6.js ------
export class Hello {
  say() {
    return 'ES6 says HI!';
  }
}
```

```
//------ main.js ------
import { Hello } from './hello-es6'
import amd from './hello-amd'
import cjs from './hello-cjs'

// es6
var hello = new Hello()
console.log('es6 ' + hello.say());

// amd
console.log('amd ' + amd.say());

// commonjs
console.log('commonjs ' + cjs.say());
```

JSPM

- It’s package manager with a CLI for the SystemJS, built on top of the dynamic ES6 module loader.
- It creates bundle or a self-executing bundle.
- It seems like what the NPM team recommended as a solution to front-end dependency management.

ES6 Compatibility Table

		Compilers/polyfills							Desktop browsers										
	34%	60%	76%	27%	26%	10%	8%	23%	6%	21%	70%	44%	60%	68%	69%	34%	37%	48%
Feature name	Current browser	Traceur	6to5 + core-js[1]	ES6 Transpiler	Closure Compiler	JSX[2]	TypeScript	es6-shim	IE 10	IE 11	IE Technical Preview[3]	FF 31	FF 35	FF 36	FF 37	CH 39, OP 26[4]	CH 40, OP 27[4]	CH 41, OP 28[4]
Optimisation																		
proper tail calls (tail call optimisation)	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No
Syntax																		
default function parameters	0/5	3/5	5/5	3/5	4/5	0/5	3/5	0/5	0/5	0/5	0/5	3/5	3/5	3/5	3/5	0/5	0/5	0/5
rest parameters	0/3	3/3	3/3	2/3	1/3	2/3	2/3	0/3	0/3	0/3	2/3	2/3	2/3	2/3	2/3	0/3	0/3	0/3
spread (...) operator	0/8	8/8	8/8	6/8	2/8	0/8	0/8	0/8	0/8	0/8	4/8	6/8	6/8	8/8	8/8	0/8	0/8	0/8
object literal extensions	0/5	5/5	5/5	5/5	4/5	2/5	2/5	0/5	0/5	0/5	5/5	0/5	5/5	5/5	5/5	0/5	0/5	2/5
for..of loops	4/4	4/4	4/4	3/4	3/4	0/4	0/4	0/4	0/4	0/4	4/4	3/4	3/4	4/4	4/4	4/4	4/4	4/4
octal and binary literals	0/4	2/4	2/4	2/4	2/4	0/4	2/4	0/4	0/4	0/4	2/4	2/4	2/4	4/4	4/4	0/4	4/4	4/4
template strings	0/2	2/2	2/2	2/2	2/2	2/2	1/2	0/2	0/2	0/2	1/2	0/2	2/2	2/2	2/2	0/2	0/2	0/2
RegExp "y" and "u" flags	0/2	1/2	1/2	0/2	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
destructuring	0/15	15/15	15/15	11/15	11/15	7/15	0/15	0/15	0/15	0/15	0/15	8/15	11/15	12/15	12/15	0/15	0/15	0/15
Unicode code point escapes	No	Yes	Yes	Yes	Yes	No	No	No	No	No	Yes	No	No	No	No	No	No	No
Bindings																		
const	1/8	6/8	6/8	6/8	6/8	1/8	0/8	0/8	0/8	8/8	8/8	3/8	3/8	8/8	8/8	1/8	1/8	5/8
let	0/10	8/10	8/10	6/10	8/10	0/10	0/10	0/10	0/10	8/10	8/10	0/10	0/10	0/10	0/10	0/10	0/10	5/10
block-level function declaration[9]	No	Yes	Yes	No	Yes	No	No	No	No	Yes	Yes	No	No	No	No	Flag	Flag	Yes
Functions																		
arrow functions	0/9	7/9	7/9	6/9	7/9	6/9	6/9	0/9	0/9	0/9	8/9	7/9	7/9	7/9	7/9	0/9	0/9	0/9
class	0/11	8/11	10/11	9/11	5/11	8/11	0/11	0/11	0/11	0/11	11/11	0/11	0/11	0/11	0/11	0/11	0/11	0/11
super	0/4	4/4	4/4	4/4	0/4	0/4	0/4	0/4	0/4	0/4	4/4	0/4	0/4	0/4	0/4	0/4	0/4	0/4
generators	10/13	12/13	12/13	0/13	7/13	0/13	0/13	0/13	0/13	0/13	0/13	9/13	11/13	12/13	12/13	10/13	10/13	11/13
Built-ins																		
typed arrays	21/40	0/40	0/40	0/40	0/40	0/40	0/40	0/40	16/40	16/40	40/40	18/40	19/40	19/40	33/40	21/40	21/40	21/40
Map	11/11	10/11	11/11	0/11	0/11	0/11	0/11	11/11	0/11	5/11	11/11	10/11	11/11	11/11	11/11	11/11	11/11	11/11
Set	11/11	10/11	11/11	0/11	0/11	0/11	0/11	11/11	0/11	5/11	11/11	10/11	11/11	11/11	11/11	11/11	11/11	11/11
WeakMap	4/4	0/4	4/4	0/4	0/4	0/4	0/4	0/4	0/4	2/4	4/4	2/4	3/4	4/4	4/4	4/4	4/4	4/4
WeakSet	4/4	0/4	4/4	0/4	0/4	0/4	0/4	0/4	0/4	0/4	4/4	0/4	4/4	4/4	4/4	4/4	4/4	4/4

http://kangax.github.io/compat-table/es6/

Demo

https://github.com/martinmicunda/employee-scheduling-ui

Resources

ES6 Draft Specification

ES6 Features - Luke Hoban

Traceur Compiler

6to5 Compiler

ES6 Compatibility Table

Read Axel Rauschmayer Blog

Understanding ECMAScript 6 - Nicholas C. Zakas

Practical Workflows for ES6 Modules - Guy Bedford

ES6 Module Loader - Guy Betford

SystemJS - Guy Betford

JSPM - Guy Betford

Thanks!

blog: http://martinmicunda.com

github: martinmicunda

twitter: @martinmicunda

Questions?