ECMAScript 6

A bright new future is coming…

ES2015 Arrow functions

ES5

- function(a){ return a*2 }
- function(){ return “foo” }
- function (a,b) {
  return a + b
  }

ES2015

- a => a*2
- () => “foo”
- (a,b) => {
  return a + b
  }

ES2015 Arrow functions

```
3 setTimeout(function () {
4     return console.log('FIN');
5   }, 500);
```

```
setTimeout(() => console.log('FIN'), 500);
```

ES2015 Arrow functions

- Sintaxis mucho más breve y legible
- "this" se toma del contexto (léxico), por lo que no hace falta **.bind** o **this = that**, etc.
- Además de this, también tiene las siguientes variables léxicas: **arguments**, **super**

ES2015
“destructuring”
(y valores por defecto en funciones)

```
1   var obj = { num:1, text:'foo'};
2   var obj2 = { num: 2 };
3
4   var {num, text} = obj;
5
6   function print({ num, text = 'hey' }){
7     console.log('Num:', num, 'Text:', text);
8   }
9
10  print(obj); //Num: 1 Text: foo
11  print(obj2);//Num: 2 Text: hey
```

ES2015 - clases

```
class Foo {
  constructor(x){
    this._prop = x;
  }

  doSomething(){
    console.log(this._prop + ' is doing something');
  }
}
```

ES2015 - herencia

```
class Boo extends Foo {
  constructor(x){
    super(x)
  }
  doSomethingNew(){
    super.doSomething();
    console.log('Desde Boo!');
  }
}
```

ES2015 - import y export

- Un intento de estandarizar los dos formatos de módulos más comunes
  - **CommonJS** (node.js)
  - **AMD** (require.js)
  - Aún no hay soporte nativo 100%, pero está en camino (webpack 2, HTTP/2, System.js, etc)

ES2015 - import y export

- Un módulo tiene dos tipos de exportaciones
  - Con nombre (named export)
  - Por defecto (default export, 1 por módulo)

ES2015 - import y export

```
// Modulo A
export function hola(){
  return "hola!";
}

export default adios(){
  return "adiós!"
}

// Modulo B
import adios from './moduloA';
import { hola } from './moduloA';
// en una sola línea
import adios, { hola } from './moduleA';
```

ES2015 - String

- Nuevos métodos en el objeto String
- "abc".**startsWith**("a") // true
- "abc".**endsWith**("b") // false
- "abc".**includes**("b") // true

ES2015 - String templates

```
const first = 'Michael',
      last = 'Jordan';

const fullName = `${first} ${last}`;
```

```
const multilinea = `
Esta es una cadena
con varias lineas
`;
```

ES2015 - Operador spread

```
const user = {
  first: 'Michael',
  last: 'Jordan'
}
const anotherUser = {
  ...user,
  last: 'Knight'
}
// { first: 'Michael', last: 'Knight '}
const complexObj = {
  number: 1,
  data: [1, 2, 3]
}
const anotherComplexObj = {
  ...complexObj,
  data: [ ...complexObj.data, 4 ]
}
//{ number: 1, data: [1, 2, 3, 4] }
```

ES2015 - Operador spread

- **OJO**: para usar el { ...spread }, tenemos que añadir un nuevo preset a babel: "stage-0"

- **npm install -D babel-preset-stage-0**

- En **.babelrc**:

```
{
   "presets": ["es2015", "react", "stage-0"]
}
```

ES2015 - Arrays

- Array.map(*func(elemento)*)
  Devuelve un nuevo array, resultado de aplicar la función a cada elemento

- Array.reduce(*func(acumulador, elemento*), *inicial*)
  Recorre el array y devuelve un nuevo valor, resultado de aplicar la función a cada elemento, acumulando un valor

ES2015 - Arrays

```
1  var numeros = [1,2,3];
2  var cuadrados = numeros.map(x => x*x);
3  //[1,4,9]
4
5  var suma = numeros.reduce((acc, x) => {
6    return acc + x;
7  }, 0);
8  //6
```

ES2015 - Arrays

- Array.filter(*predicado*)
  Devuelve un nuevo array con los elementos para los cuales la función *predicado* devuelve “true”

- Array.find(*predicado*)
  Devuelve el primer **elemento** del Array que cumple el predicado

ES2015 - Arrays

```
10  var impares = numeros.filter(x => x % 2 === 1);
11  //[1, 3]
12
13  var n = numeros.find(x => x === 4);
14  //undefined
```