ES2015 / ES6

ECMAScript 2015

also known as

ECMAScript 6

New Features

- arrows
- classes
- enhanced object literals
- template strings
- destructuring
- default + rest + spread
- let + const
- iterators + for..of
- generators
- Unicode
- modules
- module loaders
- map + set + weakmap + weakset
- proxies
- symbols
- subclassable built-ins
- promises
- math + number + string + array + object APIs
- binary and octal literals
- reflect api
- tail calls

Arrows aka Fat Arrow Functions

- Arrows are a function shorthand using the => syntax
- They are syntactically similar to lambda expressions in C#
- Unlike functions, arrows share the same lexical this as their surrounding code

```
var bob = {
  _name: "Bob",
  _friends: [],
  printFriends() {
    this._friends.forEach(f =>
      console.log(this._name + " knows " + f));
  }
}
```

Enhanced Object Literals

- Object literals are extended to support:
  - setting the prototype at construction
  - defining methods
  - making super calls
  - computing property names with expressions

```
var obj = {
    // Setting the prototype
    __proto__: theProtoObj,
    // Shorthand for ‘handler: handler’
    handler,
    // Methods
    toString() {
      // Super calls
      return "d " + super.toString();
    },
    // Computed (dynamic) property names
    [ 'prop_' + (() => 42)() ]: 42
};
```

AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

Template Literals

- Template literals are string literals allowing embedded expressions
- You can use multi-line strings and string interpolation features with them
- Template literals are enclosed by the back-tick (\` \`)

```
`string text`

`string text line 1
 string text line 2`

`string text ${expression} string text`

tag `string text ${expression} string text`

var a = 5;
var b = 10;
console.log(`Fifteen is ${a + b} and not ${2 * a + b}.`);
```

AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

Destructuring

- The destructuring assignment syntax is a JavaScript expression that makes it possible to extract data from arrays or objects into distinct variables
  - Is similar to features present in languages such as Perl and Python

```
var a, b, rest;
[a, b] = [1, 2]
console.log(a) // 1
console.log(b) // 2

[a, b, ...rest] = [1, 2, 3, 4, 5]
console.log(a) // 1
console.log(b) // 2
console.log(rest) // [3, 4, 5]

({a, b} = {a:1, b:2})
console.log(a) // 1
console.log(b) // 2
```

Array destructuring

Object Destructuring

```
var o = {p: 42, q: true};
var {p, q} = o;

console.log(p); // 42
console.log(q); // true

//  A variable can be assigned its value with destructuring separate from its declaration
var a, b;
({a, b} = {a:1, b:2});

// Assigning to new variable names
var o = {p: 42, q: true};
var {p: foo, q: bar} = o;

console.log(foo); // 42
console.log(bar); // true

// + default values and more
```

Default

- Callee-evaluated default parameter values

```
function f(x, y=12) {
  // y is 12 if not passed (or passed as undefined)
  return x + y;
}

f(3) == 15;  // True
```

Rest & Spread

- Rest

```
function f(x, ...y) {
  // y is an Array
  return x * y.length;
}
f(3, "hello", true) == 6;  // True
```

- Spread

```
function f(x, y, z) {
  return x + y + z;
}
// Pass each elemement of array as argument
f(...[1,2,3]) == 6;  // True
```

Let & Const

- let is the new var – uses block scoping
- const is also block scoped

```
function f() {
  {
    let x;
    {
      // okay, block scoped name
      const x = "sneaky";
      // error, const
      x = "foo";
    }
    // error, already declared in block
    let x = "inner";
  }
}
```

From var to let

```
var x = 3;
function func(randomize) {
    if (randomize) {
        var x = Math.random();
        return x;
    }
    return x;
}
var y = func(false);
```

- What is the value of y?

```
let x = 3;
function func(randomize) {
    if (randomize) {
        let x = Math.random(); // (A) scope: whole function
        return x;
    }
    return x; // accesses the x from line A
}
var y = func(false);
```

References & Links

- ES2015 overview
  https://github.com/lukehoban/es6features
- Exploring ES6
  http://exploringjs.com/es6/
- ES2015 support
  https://kangax.github.io/compat-table/es6/