CITY UNIVERSITY
LONDON

ES6

Aris Markogiannakis
City University - Short Courses
CS3606 : JavaScript 2: Advanced Javascript for Websites and Web

Today

- We will go through the following:
  - Babel
  - ES6 – ES5 differences
  - Webpack

ES6

- ES6 - also known as Harmony, es-next, ES2015 is the latest finalised specification of the language
- ES6 specification was finalized in June 2015
- Future versions of the specification will follow the ES[YYYY]
- But check the Compatibility Table
  - http://kangax.github.io/compat-table/es6/

Tooling

- ES6 is not supported by all browsers
- We need a JavaScript to JavaScript Transpiler
- What a transpiler does:
- Allow to compile code in the latest version into older versions of the languages
- As browser supports gets better, we will transpile ES2016, ES2017 into ES6 and beyond
- We will need better source mapping functionality
- They’re the most reliable way to run ES6
- We use **Babel** as a transpiler for converting our code from ES6 to ES5

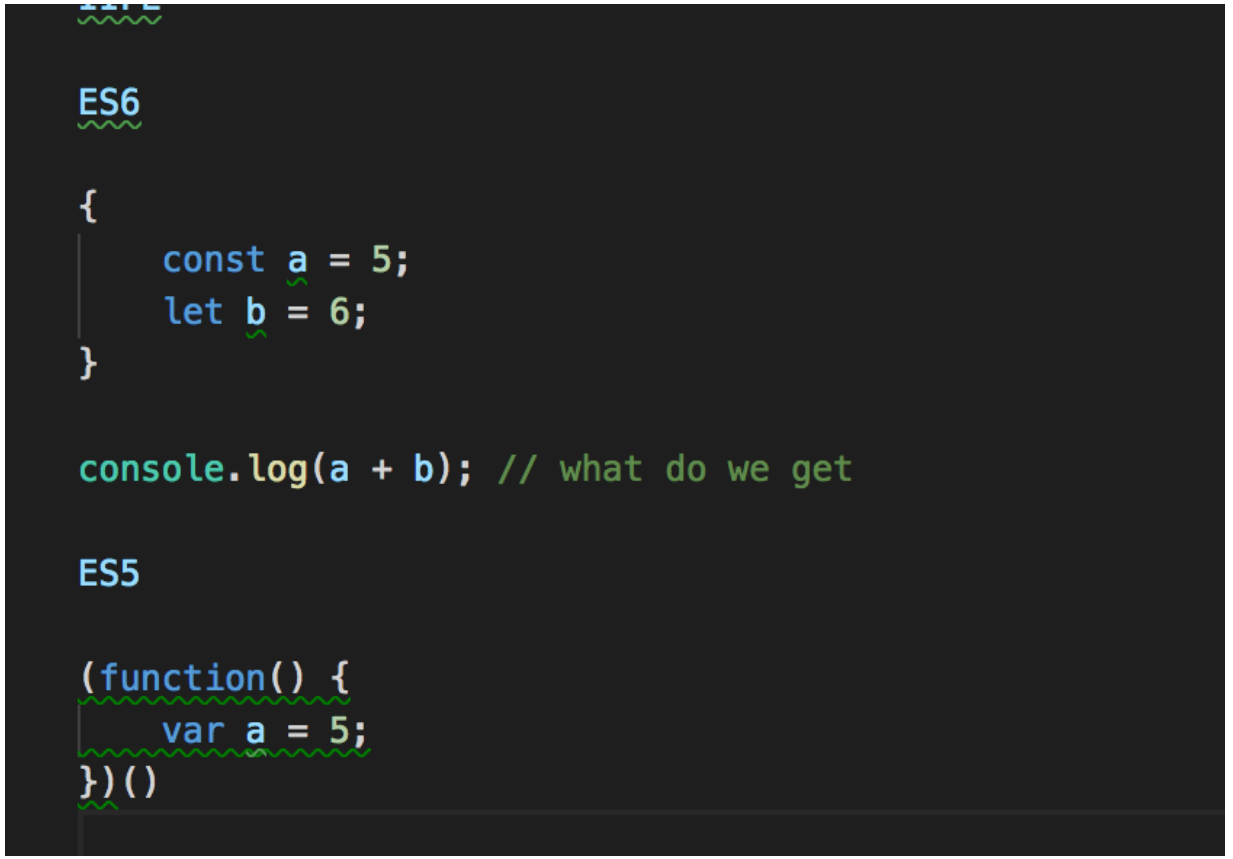
Differences – variables declaration

- Var for global variables

- Const for values that don’t change
- let for variables declaration that will change and only inside blocks.

IIFE

An immediately-invoked function expression (or **IIFE**, pronounced "iffy") is a JavaScript programming language idiom which produces a lexical scope using JavaScript's function scoping.



String concatenation

- When we want to mix variables with strings and variables

```
// Strings

let name = "Aris";
let job = "Web Developer";

// es5
console.log(name + 'is a' + job);

// es6
console.log(`{${name} is a ${job}`);

var myArray = ['January', 'February', 'March'];
var item = myArray[Math.floor(Math.random()*myArray.length)];

// Print the word that starts with a characters or string
console.log(item.startsWith('J'));

// Repeat the string X times
console.log(`${name}`.repeat(5));
```

Arrow functions – also called
“fat **arrow**” **functions**, from CoffeeScript (a transcompiled language) — are a more concise syntax for
writing **function** expressions. They utilize a new token, => , that looks like a fat**arrow**. **Arrow functions** are anonymous and change the way this binds in **functions**.

```
const salaries = ["40000", "200000", "300000"];

function salaryAfterTax(salarytext) {
    var salary = parseFloat(salarytext);
    var salary_after_tax = 0;

    salary_after_tax =  salary - (salary*((salary > 40000) ? 40 : 20))/100;

    return salary_after_tax;
}

ES5

var salaries_vat = salaries.map(function(salary){
    return salaryAfterTax(salary);
});

console.log(salaries_vat);

ES6

var salaries_vates6 = salaries.map(salary => salaryAfterTax(salary));

console.log(salaries_vates6);
```

Destructuring

A JavaScript expression that makes it possible to unpack values from arrays, or properties from objects into distinct variables.

```
// Destructuring
// ES5
var firstSalary = salaries[0];
var secondSalary = salaries[1];

const [firstSalaryes6, secondSalaryes6] = salaries;


const employee=  {
    name: 'John',
    role: 'Senior Dev'
}


const {name, role} = employee;

console.log(name);
```

Spread Operators

Allows an iterable such as an array expression or string to be expanded

in places were zero or more arguments (for function calls) or elements (for array literals) are expected, or an object expression to be expanded in places where zero or more key-value.

```
function calculateTax(salary1, salary2) {
    if (salary1 > salary2) {
        console.log('salary1 is bigger than salary2');
    }
}


//What if you wanted to add one more value to compare
let newsalaries = [40000, 200000, 300000];

function getAverageSalary(a, b, c) {
   return (a + b +c)/3;
}

console.log(getAverageSalary(...newsalaries));

const family = ['John', 'suzan', 'peter', 'mike'];
const relatives = ['James', 'Kit', 'Rosa', 'Maria'];

const fullfamily = [...family, ...relatives];

console.log(fullfamily);
```

Map

Object holds key-value pairs. Any value (both objects and primitive values) may be used as either a key or a value.

```
// Maps - build in type

const  question = new Map();
question.set('question', 'What is the most expensive car?');
question.set(1, 'Lamborgini');
question.set(2, 'Ferrari');
question.set(3, 'porce');
question.set('correct', 3);
question.set(true, 'Correct answer is :D');
question.set(false, 'Wrong answer');

console.log(question.get('question'));
console.log(question);

if (question.has(4)){
    question.delete(4);
}

question.clear();
```

Our first ES6

- Before we start we need to install node.js
- To install node you this link
- To run node you will need to open your command prompt on windows or terminal on mac
- Once you have got it installed open the terminal and write the following command:

node -v --- that will return the version of your node

Setting up your project

Create a folder on your projects area let say “myfirstes6”

- Open the folder with Visual studio, then from the top menu select View and then Integrated terminal.
- Type on your terminal: npm init

package.json

You will be asked some questions please write back the answers (see an example to your bottom)

This will create a file - **package.json** on your folder.

```
name: (Lecture6) introtoes6
version: (1.0.0) 0.0.1
description: my first time in es6
entry point: (index.js) index.js
test command:
git repository:
keywords:
author: Your name
license: (ISC) isc
Sorry, license should be a valid SPDX license expression (without "LicenseRef"), "UNLICENSED", or "SEE LICENSE IN <filename>" and license is similar to the valid
license: (ISC)
About to write to /Users/aris/Documents/Lectures-code/Lecture6/package.json:

{
  "name": "introtoes6",
  "version": "0.0.1",
  "description": "my first time in es6",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Your name",
```

Installing babel

- We now need to install babel

npm install babel-core babel-loader@7 babel-preset-es2015 webpack --save-dev

- Open package.json and add make the changes on the scripts area:

"scripts": {
"webpack": "webpack --mode development",
"start": "http-server"
},

Webpack

- Weback is a module that helps us to build our projects
- Create inside the main folder a file called webpack.config.js and copy the code from the right.
- Create a folder called src and then inside that a folder called app.
- Inside the src create a file called **index.js**

```
const path = require('path');
module.exports = {
  entry: { main: './src/index.js' },
  output: {
    path: path.resolve(__dirname, 'dist'),
    filename: 'app.bundle.js'
  },
  module: {
    rules: [
      {
        test: /\.js$/,
        exclude: /node_modules/,
        use: {
          loader: "babel-loader"
        }
      }
    ]
  }
};
```

First ES6 class

- Ok we are going to write our first class,
- Add inside a file called food.js inside the app.js
- Our class has a constructor that sets the instance
- And two methods one that returns all the values as string one that prints it

```
"use strict";

// Food is a base class

export default class Food {

  constructor (name, protein, carbs, fat) {

     this.name = name;

     this.protein = protein;

     this.carbs = carbs;

     this.fat = fat;

  }


  toString () {

     return `${this.name} | ${this.protein}g P :: ${this.carbs}g C ::
   ${this.fat}g F`

  }


  print () {
```

Subclass

- Inside your index.js file add the following code.

- Because we are using webpack we are able to import our class and then use it to create a new instance

```
import Food from './app/food.js';\

const chicken_breast = new Food('Chicken Breast', 26, 0, 3.5);


chicken_breast.print(); // 'Chicken Breast | 26g P :: 0g C :: 3.5g F'


console.log(chicken_breast.protein); // 26 (LINE A)
```

Html file setup

Create an html file at the root of your folder as index.html and add the following code.

- On your terminal run
- npm run webpack
- This command will build your code and minify it and save everything to one file.
- Open another tab on your terminal (press the + sign) and run
- npm run start
- This command will start a webserver on your computer.
- Go to your browser on http://localhost:8080

```
<!doctype html>
<html>


<head>
<title>First es6</title>
<meta charset="UTF-8">
<meta name="description" content="">
<meta name="viewport" content="width=device-width, initial-scale=1">
</head>
<body>
<script src="build/app.bundle.js"></script>
</body>
</html>
```

Result

The result is the following