INTRO TO ES6/2015

Jamal O'Garro
Software Engineer + Instructor

WHAT'S BEING COVERED TODAY

- Let and Const
- Block Scoping
- Template Strings
- Iterators
- Fat Arrow Syntax
- Class Syntax
- Promises
- Generators

BRIEF HISTORY OF JAVASCRIPT

- Created by Netscape as a browser scripting language
- Node.js allows JS to be run on the server
- ES6 standard is announced

WHAT IS ECMASCRIPT?

- Specification that JavaScript is developed against

WHAT IS ES6?

- Most recent version of the ECMAScript specification
- Introduces powerful new features and new syntax
- Is available for use now in most modern browsers

WHY USE ES6?

- Allows us to write “safer” JavaScript
- Get to use a “nicer” syntax
- It’s the future and the future is now!

NEW LANGUAGE FEATURES

LET + CONST

```
let myVar = 'some value';

const myConstant = 'some constant value';
```

LET + CONST RECAP

- “let” and “const” allow us to have block scoping
- “const” allows us to create immutable data
- We also have the ability to create blocks on-the-fly

TEMPLATE STRINGS

```
console.log(`Hello my name is ${this.first}`);
```

TEMPLATE STRINGS RECAP

- Allow us to perform string interpolation in JavaScript
- Prevents us from having to concat strings
- Get a performance boost over string concatenation

FAT ARROW FUNCTIONS

```
arr.reduce((prev, curr) => prev + curr);
```

FAT ARROW RECAP

- Provide a new syntax to represent lambda functions
- Give us a lexical “this” in functions that aren’t direct properties of objects

NEW OBJECT SYNTAX

```
let first = 'John';
let last = 'Doe';

let person = { first, last };
```

NEW OBJECT SYNTAX RECAP

- Syntactic sugar for creating properties of objects

CLASSES

```
class Person  {
    constructor(first, last) {
        this.first = first;
        this.last = last;
    }

    sayHello() {
        console.log('Hello, my name is ' + this.first);
    }
}
```

CLASSES RECAP

- Syntax “sugar” for constructor functions
- Easy inheritance
- Super keyword
- Clean way to create static methods

NEW DATA TYPES

ITERATORS

```
let arr = [1,2,3,4,5,6];

for (let element of arr) {
    console.log(element);
}
```

ITERATOR VS. ITERABLE

- Iterator - allows us to produce a sequence of values (Generators, etc.)

- Iterable - Means an object can be iterated (String, Array, Map, Set, etc.)

PROMISES

```
let promise = new Promise((resolve, reject) => {
    setTimeout(() => resolve('some data'), 1000);

    setTimeout(() => reject('there was an error'), 2000);
});
```

PROMISE RECAP

- Allow us to work with Promises natively in the language

GENERATORS

```
function* generatorFunction(){
    let v1 = yield asyncFunc1();

    let v2 = yield asyncFunc2();

    let v3 = yield asyncFunc3();
}
```

WHY ARE GENERATORS USEFUL?

- Allow us to pause a function
- Control of the function's execution externally
- Can pass data to the paused execution
- Can be used for asynchronous coding that "looks" synchronous

SUMMARY

WHAT DID WE NOT COVER TODAY?

- Symbols
- Maps
- Weak Maps
- Sets
- Weak Sets
- Some features that are not ready (at least not in V8 and Node)

WHAT'S CURRENTLY NOT AVAILABLE?

- Destructured assignment
- Parameter Default Values
- Rest
- Spread
- Modules
- Proxies
- Typed Array

HOW TO USE WHAT'S NOT READY?

- Polyfills
- Transpilers
- Babel.js
- Tracuer
- TypeScript

THANK YOU!!!

Jamal O'Garro
Software Engineer + Instructor