

Capítulo 1: JavaScript

Capítulo 2: Angular 8



CIBERTEC

Curso: Angular 8.0 Front -End Application Developer

Capítulo N° 1

Tema 1: Java Script







2019



Objetivos

Al finalizar el capítulo, podrás:

- Identificar las novedades de ES6, lo último de JS.
- Elaborar falsas API REST para agilizar el desarrollo front.



Agenda

- Tema 1: Novedades de ECMAScript 6.
- Tema 2: Crear una API REST falsa.

ECMAScript 6

- ES6 o ECMAScript 2015, es la 6a versión de la especificación del lenguaje ECMAScript o como comúnmente lo conoces, JavaScript. Esta nueva versión trae consigo muchas novedades para el lenguaje y fue, oficialmente, aprobada el 17 de junio de 2015.
- La implementación de estas mejoras en los principales motores de JavaScript como V8 de Chrome o Spider Monkey de Mozilla, se encuentra en desarrollo constante.



ECMAScript 6

- ES6 trae muchos cambios significativos al lenguaje, veamos algunos de ellos:
 - Block Scope: Let, const
 - Arrow Function
 - Rest and default parameters
 - Classes
 - Template Literals
 - Destructuring
 - Spread operator
 - Promise

ECMAScript 6

- **Let:** nos permite definir el alcance de una variable a un nivel de bloque de código, definido entre “{..}”; fuera de ese bloque, la variable no puede ser alcanzada.

```
function sum(numbers) {  
  let result = 0;  
  for (let i = 0; i < numbers.length; i++) {  
    result += numbers[i];  
  }  
  return result;  
}  
sum([10,20,30]); // 60
```

ECMAScript 6

- **Const:** al igual que con Let, nos permite definir el alcance de una variable a nivel de bloque, adicional a ello, previene que, una vez definido su valor, éste no cambie.

```
let Circle = {  
  area: function (radio) {  
    const PI = 3.1416;  
    return PI * radio * radio;  
  }  
}  
let area = Circle.area(10); // area: 314.16
```

ECMAScript 6

- **Arrow function:** conocida en otros lenguajes (C#, Java) como expresiones lambda, arrows o flechas son abreviaciones de funciones, utilizando el operador =>

```
let numbers = [3, 6, 9];  
  
let squared = numbers.map(n => n * n);  
// squared: [9, 36, 81]
```

ECMAScript 6

- **Rest and default parameters:** el intérprete de JavaScript no valida que se cumpla con la cantidad de parámetros que se definen en una función; por ello, al ejecutarla, se pueden pasar más parámetros, como ningún parámetro y la función se va a ejecutar igual. Para estos escenarios la nueva versión de ES6 nos provee estas dos características:

```
const sum = (a = 0, b = 0, ...rest) => {  
  let result = a + b;  
  rest.forEach(n => result += n);  
  return result;  
};  
  
sum(1, 5, 6, 8); // 20
```

ECMAScript 6

- **Class:** esta nueva versión implementa las clases, tal como se conocen en POO, porque la naturaleza de JS es trabajar con la estructura de **Prototype**.

```
class Circle {  
  PI = 3.1416;  
  constructor(radius) {  
    this.radius = radius;  
  }  
  area() {  
    return this.PI * this.radius * this.radius;  
  }  
}  
  
let c1 = new Circle(10);  
c1.area(); // 314.16
```

ECMAScript 6

- **Template literals:** son un tipo especial de cadena con formato, similares a la interpolación en otros lenguajes como Ruby. Se definen con un par de caracteres back-tick (`), a diferencia de las cadenas normales que usan comillas sencillas o dobles.

```
const greet = (name, gender= 'Mr.', greeting= 'Hello') => {  
  return `${greeting}! ${gender} ${name}`;  
};  
  
greet('Manuel');  
// Hello! Mr. Manuel
```

ECMAScript 6

- **Destructuring:** esta característica nos permite descomponer un array u objeto, para asignarlo a un conjunto de variables.

```
const [a, , c] = [100, 200, 300];  
sum(a, c); // 400  
  
const user = {  
  name: 'Pedro',  
  surname: 'Perez',  
  gender: 'Mr.',  
  email: 'pp@gmail.com'  
};  
  
const { name, gender } = user;  
greet(name, gender); // Hello! Mr. Pedro
```

ECMAScript 6

- **Spread operator:** genera una lista de valores, a partir de un array. En el caso de los objetos, extrae los valores por “key” y “value”.

```
const numbers = [1, 8, -5, 10];  
Math.min(...numbers) // -5  
  
const o1 = { user: 'Pedro', email: 'pp@gmail.com', age: 20 };  
const o2 = {...o1};
```



ECMAScript 6

- **Promise:** es usado para computaciones asíncronas, la cual representa un valor que puede estar disponible ahora, en el futuro o nunca.

```
const API = 'https://jsonplaceholder.typicode.com/posts?_limit=10';  
  
fetch(API)  
  .then(res => res.json())  
  .then(data => {  
    console.log(data);  
  });
```



Ejercicio Nº 1.1: Ejecutar las novedades de ES6

Al finalizar el laboratorio, podrás:

- Aplicar las novedades de ES6.



Crear una API REST falsa

- Para simular una API REST, usaremos la librería **json-server**.
- Se define un archivo JSON, que será como nuestra base de datos.
- Se utilizan los verbos http (POST, PUT, PATCH o DELETE) para modificar el JSON.
- Instalación:

```
npm install -g json-server  
json-server -w db.json
```



Ejercicio Nº 1.2: Diseñar una API REST falsa para el curso

Al finalizar el laboratorio, podrás:

- Diseñar una API REST falsa para simular un backend listo.



Lecturas adicionales

Para obtener información adicional, puede consultar:

- [ES6 - New Features](#)
- [Soporte de ECMAScript 2015](#)
- [You Don't Know JS](#)
- [Create A REST API with JSON-Server](#)
- [JSON-Server as a Fake REST API](#)
- [My JSON Server](#)



Resumen

En este capítulo, aprendiste:

- Las novedades de ES2015, nos permite reusar códigos y aplicar, mejores patrones de diseño y paradigmas, como la programación orientada a objetos.
- Crear un API REST falsa, nos saca de apuros cada vez que queremos avanzar con el desarrollo frontend sin depender de un backend listo.



Tarea Nº 1: Recopilar información sobre los frameworks JavaScript basados en componentes

Investigar sobre los frameworks JavaScript que agilizan el desarrollo web, basado en componentes.

