What is ES6?

ES6 or the ECMAScript 2015 is the 6th and major edition of the ECMAScript language specification standard. It defines the standard for the implementation of JavaScript and it has become much more popular than the previous edition ES5.  
  
ES6 comes with significant changes to the JavaScript language. It brought several new features like, let and const keyword, rest and spread operators, template literals, classes, modules and many other enhancements to make JavaScript programming easier and more fun. In this article, we will discuss some of the best and most popular ES6 features that we can use in your everyday JavaScript coding.

* **let and const Keywords**
* **Arrow Functions**
* **Multi-line Strings**
* **Default Parameters**
* **Template Literals**
* **Destructuring Assignment**
* **Enhanced Object Literals**
* **Promises**
* **Classes**
* **Modules**

**Understanding these Features**

**1. let and const keywords :**

The keyword "let" enables the users to define variables and on the other hand, "const" enables the users to define constants. Variables were previously declared using "var" which had function scope and were hoisted to the top. It means that a variable can be used before declaration. But, the "let" variables and constants have block scope which is surrounded by curly-braces "{}" and cannot be used before declaration.

let i = 10;

console.log(i); //Output 10

const PI = 3.14;

console.log(PI); //Output 3.14

**2. Arrow Functions**

ES6 provides a feature known as Arrow Functions. It provides a more concise syntax for writing function expressions by removing the "function" and "return" keywords.  
  
Arrow functions are defined using the fat arrow (=>) notation.

// Arrow function

let sumOfTwoNumbers = (a, b) => a + b;

console.log(sum(10, 20)); // Output 30

It is evident that there is no "return" or "function" keyword in the arrow function declaration.  
  
We can also skip the parenthesis in the case when there is exactly one parameter, but will always need to use it when you have zero or more than one parameter.  
  
But, if there are multiple expressions in the function body, then we need to wrap it with curly braces ("{}"). We also need to use the "return" statement to return the required value.

**3. Multi-line Strings**

ES6 also provides Multi-line Strings. Users can create multi-line strings by using back-ticks(`).  
  
It can be done as shown below :

let greeting = `Hello World,

Greetings to all,

Keep Learning and Practicing!`

**4. Default Parameters**

In ES6, users can provide the default values right in the signature of the functions. But, in ES5, OR operator had to be used.

//ES6

let calculateArea = function(height = 100, width = 50) {

// logic

}

//ES5

var calculateArea = function(height, width) {

height = height || 50;

width = width || 80;

// logic

}

**5. Template Literals**

ES6 introduces very simple string templates along with placeholders for the variables. The syntax for using the string template is ${PARAMETER} and is used inside of the back-ticked string.

let name = `My name is ${firstName} ${lastName}`

**6. Destructuring Assignment**

Destructuring is one of the most popular features of ES6. The destructuring assignment is an expression that makes it easy to extract values from arrays, or properties from objects, into distinct variables.  
  
There are two types of destructuring assignment expressions, namely, Array Destructuring and Object Destructuring. It can be used in the following manner :

//Array Destructuring

let fruits = ["Apple", "Banana"];

let [a, b] = fruits; // Array destructuring assignment

console.log(a, b);

//Object Destructuring

let person = {name: "Peter", age: 28};

let {name, age} = person; // Object destructuring assignment

console.log(name, age);

**7. Enhanced Object Literals**

ES6 provides enhanced object literals which make it easy to quickly create objects with properties inside the curly braces.

function getMobile(manufacturer, model, year) {

return {

manufacturer,

model,

year

}

}

getMobile("Samsung", "Galaxy", "2020");

**8. Promises**

In ES6, Promises are used for asynchronous execution. We can use promise with the arrow function as demonstrated below.

var asyncCall = new Promise((resolve, reject) => {

// do something

resolve();

}).then(()=> {

console.log('DON!');

})

**9. Classes**

Previously, classes never existed in JavaScript. Classes are introduced in ES6 which looks similar to classes in other object-oriented languages, such as C++, Java, PHP, etc. But, they do not work exactly the same way. ES6 classes make it simpler to create objects, implement inheritance by using the "extends" keyword and also reuse the code efficiently.  
  
In ES6, we can declare a class using the new "class" keyword followed by the name of the class.

class UserProfile {

constructor(firstName, lastName) {

this.firstName = firstName;

this.lastName = lastName;

}

getName() {

console.log(`The Full-Name is ${this.firstName} ${this.lastName}`);

}

}

let obj = new UserProfile('John', 'Smith');

obj.getName(); // output: The Full-Name is John Smith

**10. Modules**

Previously, there was no native support for modules in JavaScript. ES6 introduced a new feature called modules, in which each module is represented by a separate ".js" file. We can use the "import" or "export" statement in a module to import or export variables, functions, classes or any other component from/to different files and modules.

export var num = 50;

export function getName(fullName) {

//data

};

import {num, getName} from 'module';

console.log(num); // 50

**Summary**

In this article, we learned about the ECMAScript 2015 or ES6, which defines the standard for JavaScript implementation.

We also learned about the top 10 features of ES6 which makes it very popular.

ES6 provides classes, modules, arrow functions, template literals, destructuring assignments and many more features which make working with JavaScript a lot easier.