Understanding ES6

ES6 or ECMAScript 2015 is the 6th edition of the ECMAScript language specification standard. It defines the standard for the implementation of JavaScript and 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` keywords, 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.

ES6 Features

1. Let and Const Keywords

The keyword `let` enables the users to define variables, and `const` enables the users to define constants. Variables were previously declared using `var` which had function scope and were hoisted to the top, meaning that a variable can be used before declaration. However, `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 function
let sumOfTwoNumbers = (a, b) => a + b;
console.log(sumOfTwoNumbers(10, 20)); // Output: 30
```

We can also skip the parenthesis when there is exactly one parameter, but we always need to use it when there are zero or more than one parameter.

```
// Single parameter
let square = (x) => x * x;
console.log(square(5)); // Output: 25

// Multiple parameters
let sum = (a, b) => {
  return a + b;
};
console.log(sum(10, 20)); // Output: 30
```

3. Multi-line Strings

ES6 also provides Multi-line Strings. Users can create multi-line strings by using back-ticks (`).

```
let greeting = `Hello World,
                Greetings to all,
                Keep Learning and Practicing!`;
console.log(greeting);
```

4. Default Parameters

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

```
// ES6
let calculateArea = function (height = 100, width = 50) {
  // logic
  return height * width;
};
console.log(calculateArea()); // Output: 5000

// ES5
var calculateArea = function (height, width) {
  height = height || 50;
  width = width || 80;
  // logic
  return height * width;
};
console.log(calculateArea()); // Output: 4000
```

5. Template Literals

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

```
let firstName = "John";
let lastName = "Doe";
let name = `My name is ${firstName} ${lastName}`;
console.log(name); // Output: My name is John Doe
```

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.

Array Destructuring

```
let fruits = ["Apple", "Banana"];
let [a, b] = fruits; // Array destructuring assignment
console.log(a, b); // Output: Apple Banana
```

Object Destructuring

```
let person = { name: "Peter", age: 28 };
let { name, age } = person; // Object destructuring assignment
console.log(name, age); // Output: Peter 28
```

7. Enhanced Object Literals

ES6 provides enhanced object literals, making it easy to quickly create objects with properties inside curly braces.

```
function getMobile(manufacturer, model, year) {
  return {
    manufacturer,
    model,
    year,
  };
}
console.log(getMobile("Samsung", "Galaxy", "2020")); // Output: { manufacturer:
'Samsung', model: 'Galaxy', year: '2020' }
```

8. Promises

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

```
var asyncCall = new Promise((resolve, reject) => {
  // do something
  resolve();
}).then(() => {
  console.log("Done!");
});
```

9. Classes

Previously, classes never existed in JavaScript. Classes are introduced in ES6, which look similar to classes in other object-oriented languages such as C++, Java, PHP, etc. However, they do not work exactly the same way. ES6 classes make it simpler to create objects, implement inheritance using the `extends` keyword, and also reuse code efficiently.

```
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.

```
// Exporting variables and functions
export var num = 50;
export function getName(fullName) {
  // logic
}

// Importing variables and functions
import { num, getName } from "module";
console.log(num); // Output: 50
```