ES6 Features

ECMAScript 2015 commonly known as ES6 or ES2015 released on June 2015. ES5 was released on December 2009. It would then take almost six years for the next version of ECMAScript to be released. So, there are many exciting feature in ES6.

Here's the list of the top 10 best ES6 features for a busy javascript developer(in no particular order):

1. Default Parameters in ES6
2. Template Literals in ES6
3. Multi-line Strings in ES6
4. Destructuring Assignment in ES6
5. Enhanced Object Literals in ES6
6. Arrow Functions in ES6
7. Promises in ES6
8. Block-Scoped Constructs Let and Const
9. Classes in ES6
10. Modules in ES6

1. Default Parameters in ES6

In ES6, we can put the default values right in the signature of the functions.

```
var calculateArea = function(height = 50, width = 80) {
    // write logic

    ...
}
```

In ES5, we were using logic OR operator.

```
var calculateArea = function(height, width) {
   height =  height || 50;
   width = width || 80;
   // write logic
    ...
}
```

2. *Template Literals in ES6*

In ES6, we can use a new syntax `${PARAMETER}` inside of the back-ticked string.

```
var name = `Your name is ${firstName} ${lastName}.`
```

In ES5, we have to break string like below.

```
var name = 'Your name is ' + firstName + ' ' + lastName + '.'
```

3. *Multi-line Strings in ES6*

In ES6, it is very simple. Just use back-ticks.

```
let poemData = `Johny Johny Yes Papa,
                Eating sugar? No, papa!
                Telling lies? No, papa!
                Open your mouth Ah, ah, ah!`
```

In ES5, we had to use below approach.

```
var poemData = 'Johny Johny Yes Papa,\n'
             + 'Eating sugar?  No, papa!\n'
             + 'Telling lies? No, papa!\n'
             + 'Open your mouth Ah, ah, ah!'
```

4. Destructuring Assignment in ES6

The destructuring assignment syntax is a JavaScript expression that makes it possible to unpack values from arrays, or properties from objects, into distinct variable.

```
var o = {p: 42, q: true};
var {p, q} = o;

console.log(p); // 42
console.log(q); // true
```

In ES5, we had to use below approach.

```
var o = {p: 42, q: true};
var p = o.p;
var q = o.q;
console.log(p); // 42
console.log(q); // true
```

5. Enhanced Object Literals in ES6

Object literals make it easy to quickly create objects with properties inside the curly braces.

```
function getLaptop(make, model, year) {
   return {
      make,
      model,
      year
   }
}

getLaptop("Apple", "MacBook", "2015");
```

In ES5, we had to follow below syntax.

```
function getLaptop(make, model, year) {
   return {
     make: make,
     model: model,
     year: year
   }
}
getLaptop("Apple", "MacBook", "2015");
```

6. Arrow Functions in ES6

The fat arrows are amazing because they would make your `this` behave properly, i.e., `this` will have the same value as in the context of the function— it won't mutate.

```
$('.btn').click((event) => {
   this.doSomething()
});
```

In ES5, we had to use _this = this or .bind(this).

```
var _this = this;
$('.btn').click(function(event){
   _this.doSomething();
});
```

For more info:
https://developer.mozilla.org/en/docs/Web/JavaScript/Reference/Functions/Arrow_functions

7. Promises in ES6

Promises are used for asynchronous execution. In ES6, we can use promise with arrow function shown below.

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

In ES5, we need to use setTimeout().

```
setTimeout(function(){
   console.log('Yay!');
}, 1000);
```

For more info: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise

8. Block-Scoped Constructs Let and Const

`let` is a new `var` which allows to scope the variable to the blocks.So, the main difference between let and var is, `var` is scoped to the nearest function block and `let` is scoped to the nearest *enclosing* block, which can be smaller than a function block.

In ES6,

```
function calculateAmount(boolVal) {
   let amount = 0;

   if(boolVal) {
      let amount = 1; // scope of this amount ends with next closing bracket
   }

   return amount;
}
console.log(calculateAmount(true)); // output: 0
```

In ES5,

```
function calculateAmount(boolVal) {
   var amount = 0;
   if(boolVal) {
      var amount = 1;
   }
   return amount;
}
console.log(calculateAmount(true)); // output: 1
```

When it comes to `const`, it’s just an immutable and it’s also block-scoped like `let`.

9. Classes in ES6

We can create class in ES6 using “*class”* keyword. Classes creation and usage in ES5 was a pain in the rear, because there wasn’t a keyword class.

```
class Profile {
   constructor(firstName, lastName = '') { // class constructor
      this.firstName = firstName;
      this.lastName = lastName;
   }

   getName() { // class method
     console.log(`Name: ${this.firstName} ${this.lastName}`);
   }
}


let profileObj = new Profile('Kavisha', 'Talsania');
profileObj.getName(); // output: Name: Kavisha Talsania
```

I am not going to show you example of ES5 class creation, because there are many flavors.

10. Modules in ES6

In ES6, there are modules with `import` and `export` operands.

```
export var userID = 10;
export function getName(name) {
   ...
};
```

We can import userID variable and getName method using import statement .

```
import {userID, getName} from 'module';
console.log(userID); // 10
```

There are other many ES6 feature, you can find all list here.

ES6 spread and rest operators are also trending these days. You can get more knowledge about spread and rest opertors from below link: