# JavaScript-ES6 Features

Here's a summary of some advanced features in ECMAScript 6 (ES6) that can enhance your JavaScript coding experience:

## 1. Let and Const Variables

`let` is used to declare block-scoped variables, while `const` is used for constants.

```
 // Using let
for (let i = 0; i < 5; i++) {
    console.log(i);
}
console.log(i); // ReferenceError: i is not defined

// Using const
const name = 'John Doe';
name = 'Jane Doe'; // TypeError: Assignment to constant variable.
```

## 2. Arrow Functions

**Arrow functions** provide a more concise syntax for writing simple function expressions.

```
 // Function expression using the old way
var add = function(x, y) {
    return x + y;
};

// Using arrow function
const add = (x, y) => { return x + y; };
```

## 3. Destructuring

**Destructuring** allows you to assign values to variables based on the properties of an object or array.

```
 // Object destructuring
let person = { firstName: 'John', lastName: 'Doe' };
let { firstName, lastName } = person;
console.log(firstName); // John
console.log(lastName);  // Doe

// Array destructuring
let numbers = [1, 2, 3];
let [numOne, numTwo] = numbers;
console.log(numOne); // 1
console.log(numTwo); // 2
```

### 4. Promises

**Promises** provide a way to handle asynchronous operations in a more readable and maintainable way.

```
 // Using callbacks (old way)
function add(x, y, callback) {
    setTimeout(() => {
        callback(null, x + y);
    }, 1000);
}

add(1, 2, (err, result) => {
    console.log(result); // 3
});

// Using Promises (new way)
function add(x, y) {
    return new Promise((resolve, reject) => {
        setTimeout(() => {
            resolve(x + y);
        }, 1000);
    });
}

add(1, 2).then(result => console.log(result)).catch(err => console.error(err));
```

### 5. Modules and Import/Export

**ES6 modules** provide a way to write reusable code in JavaScript.

```
 // exporting function
export function add(x, y) {
    return x + y;
}

// importing function
import {add} from './math.js';
console.log(add(1, 2)); // 3
```

## 6. Classes

**ES6 classes** provide a way to write object-oriented code in JavaScript.

```
 class Person {
    constructor(name) {
        this.name = name;
    }

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

let person = new Person('John Doe');
person.greet(); // Hello, my name is John Doe.
```

## 7. Template Literals

**Template literals** provide a way to write strings in a more readable and maintainable way.

```
 // Using template literal
let str = `Hello, ${name}!`;
console.log(str); // Hello, John!

// Using interpolation
let num = 10;
let str = "The number is: " + num;
console.log(str); // The number is: 10

// Using tag function (ES6 feature)
function hello(strings, ...values) {
    return strings.join(values);
}

let name = 'John Doe';
let greeting = hello`Hello, ${name}!`;
console.log(greeting); // Hello, John!
```

---

*Curated by Brajesh Kumar*