# 📘 ES6 JavaScript Features Explained

### 🔹 List of ES6 Features

* let and const for block-scoped variables
* Arrow functions (=>)
* Class syntax with inheritance
* Template literals
* Default function parameters
* Destructuring assignment
* Enhanced object literals
* Spread and rest operators
* Promises for asynchronous operations
* Modules (import/export)
* New data structures: Set and Map

### 🔹 let in JavaScript

* Used to declare block-scoped variables.
* Cannot be redeclared in the same scope.
* Hoisted but not initialized.

let score = 10;

if (true) {

let score = 20;

console.log(score); // 20

}

console.log(score); // 10

### 🔹 Differences Between var and let

| **Feature** | **var** | **let** |
| --- | --- | --- |
| Scope | Function-scoped | Block-scoped |
| Hoisting | Hoisted (initialized as undefined) | Hoisted (not initialized) |
| Redeclaration | Allowed | Not allowed |

### 🔹 const in JavaScript

* Declares block-scoped constants.
* Must be initialized at declaration.
* Cannot be reassigned.
* Mutable object properties are allowed.

const config = { theme: "dark" };

config.theme = "light"; // ✅ allowed

### 🔹 ES6 Classes

* Defined with the class keyword.
* Use constructor() for initial setup.
* Methods are added directly inside the class body.

class User {

constructor(name) {

this.name = name;

}

sayHi() {

return `Hi, ${this.name}`;

}

}

### 🔹 ES6 Class Inheritance

* Use extends to inherit from a parent class.
* Use super() to call the parent constructor.

class Animal {

constructor(name) {

this.name = name;

}

speak() {

return `${this.name} makes a sound.`;

}

}

class Dog extends Animal {

speak() {

return `${this.name} barks.`;

}

}

### 🔹 Arrow Functions

* Shorter function syntax
* Implicit return (if no curly braces)
* Lexical this binding

const greet = name => `Hello, ${name}`;

const add = (a, b) => a + b;

### 🔹 Set and Map

#### ✅ Set

* Stores unique values.
* No duplicates allowed.

const items = new Set([1, 2, 2, 3]);

console.log(items); // Set { 1, 2, 3 }

#### ✅ Map

* Stores key-value pairs.
* Keys can be objects, functions, or any type.

const ages = new Map();

ages.set("Sahil", 22);

console.log(ages.get("Sahil")); // 22