**1. Features of ES6 (ECMAScript 2015)**

ES6 introduced many improvements to JavaScript. Key features:

* let and const for block-scoped variables
* Arrow functions (=>)
* Classes and inheritance (class, extends)
* Template literals (`Hello ${name}`)
* Default parameters (function greet(name = "Guest"))
* Destructuring (const [a, b] = arr;)
* Rest & Spread operators (...args, ...arr)
* Modules (import, export)
* Promises for async code
* Map, Set, WeakMap, WeakSet

**2. Explain JavaScript let**

* let is used to declare **block-scoped** variables (not accessible outside {} block).
* Unlike var, it doesn’t get hoisted to the top of its function with a default undefined.

let count = 5;

if (true) {

let count = 10;

console.log(count); // 10 (new block scope)

}

console.log(count); // 5

**3. Differences Between var and let**

| **Feature** | **var** | **let** |
| --- | --- | --- |
| Scope | Function-scoped | Block-scoped |
| Hoisting | Yes, hoisted with undefined | Yes, but not initialized |
| Redeclaration | Allowed | Not allowed in same scope |
| Temporal Dead Zone | Not applicable | Exists (error before init) |

// var redeclares

var a = 10;

var a = 20; // No error

// let redeclares

let b = 10;

// let b = 20; // Error

**4. Explain JavaScript const**

* const is used to declare **constants** (can't be reassigned).
* Like let, it's **block-scoped**.
* For objects/arrays: The reference is fixed, but contents can change.

const x = 10;

// x = 20; // Error

const user = { name: "Alice" };

user.name = "Bob"; // Allowed

**5. ES6 Class Fundamentals**

* ES6 introduced class syntax for creating objects and constructors.

class Person {

constructor(name) {

this.name = name;

}

greet() {

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

}

}

const p = new Person("John");

p.greet(); // Hello, John

**6. ES6 Class Inheritance**

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

class Animal {

constructor(name) {

this.name = name;

}

speak() {

console.log(`${this.name} makes a sound.`);

}

}

class Dog extends Animal {

speak() {

console.log(`${this.name} barks.`);

}

}

const dog = new Dog("Rex");

dog.speak(); // Rex barks.

**7. ES6 Arrow Functions**

* Shorter function syntax.
* Does **not** have its own this (lexical scope).

// Traditional

function add(a, b) {

return a + b;

}

// Arrow

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

console.log(add(2, 3)); // 5

Use arrow functions for simple logic, callbacks, or array methods.

**8. Set() and Map()**

**➤ Set:**

* Collection of **unique** values.
* No duplicates allowed.

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

set.add(4);

console.log(set); // Set(4) {1, 2, 3, 4}

**➤ Map:**

* Collection of **key-value** pairs.
* Keys can be of any type.

const map = new Map();

map.set("name", "Alice");

map.set(1, "one");

console.log(map.get("name")); // Alice

console.log(map.size); // 2