**1. Features of ES6**

ES6 introduced major updates to JavaScript, including:

* let and const for block-scoped variables
* Arrow functions (=>)
* Classes and class inheritance
* Default parameters in functions
* Destructuring (arrays and objects)
* Modules (import and export)
* Promises for asynchronous programming
* Map and Set data structures

**2. JavaScript let**

* let is used to declare block-scoped variables.
* It is not hoisted like var, meaning it's not accessible before declaration within its block.
* Useful inside loops and conditional blocks where scope must be limited.

**js**

let x = 10;

if (true) {

let x = 20;

}

console.log(x); // 10

**3. Difference between var and let**

| **Feature** | **var** | **let** |
| --- | --- | --- |
| Scope | Function-scoped | Block-scoped |
| Hoisting | Hoisted with undefined | Hoisted but not initialized |
| Redeclaration | Allowed | Not allowed in the same scope |
| Use Case | Legacy code | Modern, safer declaration |

**4. JavaScript const**

* const declares block-scoped constants.
* Once assigned, its value cannot be changed (immutable reference).
* Useful for declaring constants and objects that shouldn’t be reassigned.

**js**

const PI = 3.14;

**5. ES6 Class Fundamentals**

* Classes in ES6 are syntactic sugar over JavaScript's prototype-based inheritance.
* They make object-oriented code cleaner and more readable.

class Person {

constructor(name) {

this.name = name;

}

greet() {

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

}

}

**6. ES6 Class Inheritance**

* Classes can inherit properties and methods from other classes using the extends keyword.
* The super() function is used to call the parent class constructor.

class Student extends Person {

constructor(name, grade) {

super(name);

this.grade = grade;

}

getDetails() {

return `${this.name} is in grade ${this.grade}`;

}

}

**7. Arrow Functions**

* Arrow functions provide a shorter syntax for writing functions.
* They do not bind their own this keyword (lexical this).
* Cannot be used as constructors.

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

**8. Set and Map**

**Set:**

* Collection of unique values.
* No duplicate entries allowed.
* Useful for filtering unique items.

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

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

**Map:**

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

const myMap = new Map();

myMap.set('name', 'Gobika');

myMap.set('age', 21);

console.log(myMap.get('name')); // Gobika