1. Features of ES6

ES6 introduced several new features to JavaScript that made the language more powerful, readable, and suitable for large-scale applications. Key features include:

- let and const for block-scoped variable declarations
- Arrow functions (=>) for cleaner function syntax
- Classes for easier object-oriented programming
- Template literals using backticks (` ``) for string interpolation
- Destructuring assignment for arrays and objects
- Default function parameters
- Rest (...args) and spread operators (...array)
- Promises for asynchronous programming
- Map and Set data structures
- Enhanced object literals
- Modules (import/export) for code organization

2. JavaScript let

The let keyword declares variables that are **block-scoped**, meaning they only exist within the nearest enclosing {} block.

- Variables declared with let can be updated but **not re-declared** in the same scope.
- let is preferable over var for cleaner scoping and fewer bugs.

3. Difference between var and let

Feature	**var**	**let**
Scope	Function-scoped	Block-scoped
Redeclaration	Allowed	Not allowed in the same scope
Hoisting	Hoisted (initialized as undefined)	Hoisted but not initialized
Use case	Legacy JS	Modern JavaScript

4. JavaScript const

const is used to declare **constants**. Like let, it's **block-scoped**. However, it **must be initialized during declaration**, and **cannot be reassigned**.

While the reference cannot change, the **contents of objects or arrays** can still be modified.

5. ES6 Class Fundamentals

ES6 introduced a clean, concise syntax to create classes.

6. ES6 Class Inheritance

ES6 supports inheritance using the extends and super keywords.

- extends sets up prototype-based inheritance.
- super is used to call methods from the parent class.

7. ES6 Arrow Functions

Arrow functions provide a shorter syntax for writing functions and **do not bind their own this context**, which makes them ideal for callbacks.

8. Set and Map in ES6

Set

A Set is a collection of **unique values** (no duplicates).

Features:

- No duplicates allowed
- Can store any type of value (primitive or object)
- Useful for filtering unique values

Map

A Map holds **key-value pairs** and remembers the original insertion order of the keys. Unlike objects, keys in a Map can be of **any type**.

Features:

- Keys can be objects, functions, or any primitive
- Maintains insertion order
- More performant and flexible than plain objects for frequent additions/removals