## List the features of ES6

Features:

* let and const
* Arrow Functions
* Template Literals
* Default Parameters
* Destructuring Assignment
* Spread and Rest Operators (...)
* Classes
* Promises
* Map & Set

## Explain JavaScript let

let is a keyword introduced in ES6 to declare variables in a block-scoped manner, offering a safer and more predictable alternative to the older var.

## Identify the differences between var and let

Differences:

* Var is Function-scoped, while let is Block-scoped ({})
* Var is hoisted to the top of function or global scope and initialized as undefined. Let is hoisted, but in a Temporal Dead Zone (TDZ) until declaration
* Var, when used in loops, Does not create a new scope per iteration whereas, let creates a new scope per iteration

## Explain JavaScript const

const (short for constant) is a way to declare block-scoped variables whose value cannot be reassigned after initialization. It was introduced in ES6 (ECMAScript 2015) to encourage the use of immutable bindings.

Just like let, variables declared with const are accessible only within the block {} where they are defined and a const declaration must be assigned a value when declared.

## Explain ES6 class fundamentals

class was introduced in ES6 as syntactic sugar over JavaScript’s existing prototype-based inheritance. It provides a cleaner and more intuitive way to create objects and handle inheritance, like classes in languages like Java or Python.

Key Fundamentals:

* Class Declaration
* Creating Instances
* Constructor Method
* Inheritance with extends
* Getters and Setters

## Explain ES6 class inheritance

Inheritance in ES6 allows one class (called a subclass) to inherit properties and methods from another class (called a superclass), using the extends and super keywords.

Example: class Dog extends Animal {}

## Define ES6 arrow functions

Arrow functions were introduced in ES6 (ECMAScript 2015) as a more concise way to write functions. They also inherit the this context from their enclosing scope, making them especially useful in callbacks and functional programming. Example:

**function add(a, b) {**

**return a + b;**

**}** 🡪 Traditional function

**const add = (a, b) => a + b;** 🡪 Arrow function

## Identify set(), map()

A Set is a built-in object that lets you store unique values, whether primitive or object references. Key features:

* No duplicates allowed.
* Order is preserved (insertion order).
* Iterable

A Map is a built-in object that holds key-value pairs and remembers the original insertion order of the keys. Key features:

* Keys can be of any type (not just strings).
* Maintains insertion order.
* Avoids key collisions (unlike plain objects).