## **Objectives**

**1. List the features of ES6**

ES6 introduced many new features to make JavaScript more powerful and easier to write:

* let and const for block-scoped variables
* Arrow functions (=>)
* Classes and class inheritance
* Template literals (`Hello ${name}`)
* Default and rest parameters
* Destructuring (arrays, objects)
* Spread and rest operators (...)
* Promises
* Modules (import / export)
* Set and Map data structures
* Enhanced object literals

**2. Explain JavaScript let**

* let allows you to declare **block-scoped** variables.
* Unlike var, it is **not hoisted** to the top of its scope.
* You **can reassign** a let variable.

**Example:**

let age = 25;

age = 30;

**3. Identify the differences between var and let**

| **Feature** | **var** | **let** |
| --- | --- | --- |
| Scope | Function-scoped | Block-scoped ({}) |
| Hoisting | Yes (initialized as undefined) | Yes, but not initialized |
| Redeclaration | Allowed | Not allowed in same scope |
| Reassignment | Allowed | Allowed |

**4. Explain JavaScript const**

* const creates **block-scoped**, **read-only** variables.
* You **cannot reassign** a const.
* But, **you can modify** the contents of objects/arrays declared with const.
* **Example:**

const pi = 3.14;

pi = 3.14159;

const arr = [1, 2];

arr.push(3);

**5. Explain ES6 class fundamentals**

* ES6 introduced class syntax to write **Object-Oriented JavaScript**.
* Classes have constructors and methods.

**Example:**

class Car {

constructor(brand) {

this.brand = brand;

}

show() {

console.log(`Brand: ${this.brand}`);

}

}

const c1 = new Car("Toyota");

c1.show();

**6. Explain ES6 class inheritance**

* extends keyword allows one class to inherit from another.
* super() is used to call the parent constructor.

**Example:**

class Vehicle {

constructor(type) {

this.type = type;

}

}

class Bike extends Vehicle {

constructor(type, model) {

super(type); // Call parent constructor

this.model = model;

}

}

const b1 = new Bike("Two Wheeler", "Royal Enfield");

**7. Define ES6 arrow functions**

* A short syntax to define functions using =>
* They **do not have their own this**, so useful in callbacks.

**Example:**

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

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

**8. Identify set(), map()**

**Set**

* Collection of **unique values**
* Can store any type

const mySet = new Set();

mySet.add(1);

mySet.add(2);

mySet.add(1); // Duplicate, ignored

**Map**

* Stores **key-value pairs**
* Keys can be any type

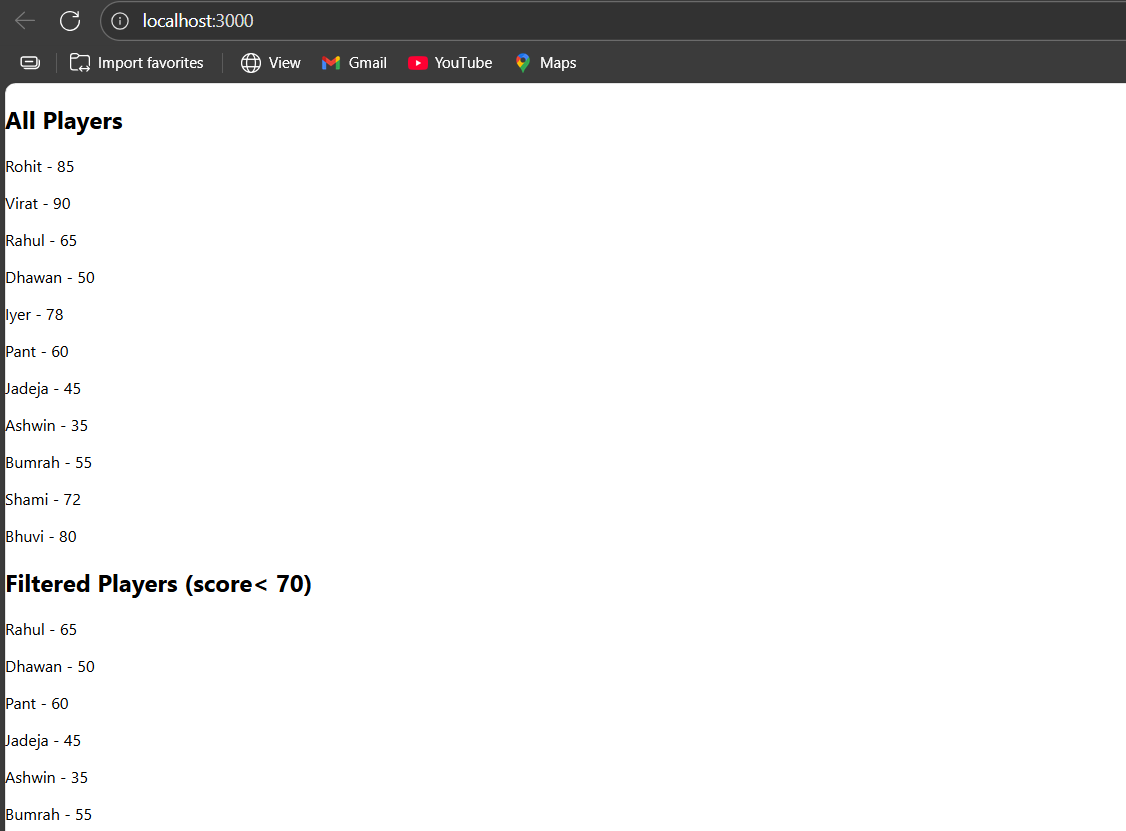
const myMap = new Map();

myMap.set("name", "Sangamesh");

myMap.set(1, "one");

console.log(myMap.get(1)); // "one"

**flag = true;**



**Flag=false**

