**Module 10 – Single Page Application framework – React**

**9.ReactJS-HOL**

**Features of ES6 (ECMAScript 2015)**

ES6 introduced several powerful features that made JavaScript more modern, cleaner, and easier to write. Key features include:

1. **let and const** – New ways to declare variables with better scoping rules.
2. **Arrow Functions** – A concise way to write functions.
3. **Classes and Inheritance** – Introduced class-based object-oriented programming.
4. **Template Literals** – Easier string interpolation using backticks (`\).
5. **Destructuring** – Extract values from arrays or objects into variables.
6. **Default Parameters** – Assign default values to function parameters.
7. **Rest and Spread Operators** – Handle arrays and function arguments efficiently.
8. **Promises** – Simplify asynchronous programming.
9. **Modules** – Allow exporting and importing code between files.
10. **Map and Set** – New data structures to handle unique and key-value data efficiently.

**JavaScript let**

The let keyword allows us to declare block-scoped variables. Unlike var, let variables are not hoisted to the top of the function or global scope. It also prevents redeclaration within the same scope.

**Example:**

let name = 'John';

name = 'Doe'; // Allowed

console.log(name); // Output: Doe

**Difference Between var and let**

| **Feature** | **var** | **let** |
| --- | --- | --- |
| Scope | Function-scoped | Block-scoped |
| Hoisting | Hoisted, initialized as undefined | Hoisted, but not initialized |
| Redeclaration | Allowed | Not allowed in same scope |
| Best Practice | Avoid using | Preferred for modern JS |

**JavaScript const**

The const keyword is used to declare variables that **cannot be reassigned** after their initial value. It's also **block-scoped** like let. However, const only prevents reassignment of the variable reference—not the contents (in case of objects/arrays).

**Example:**

const age = 25;

age = 30;

**ES6 Class Fundamentals**

ES6 introduced the class keyword to create objects using a more intuitive and clean syntax. It provides a blueprint for creating multiple objects with shared structure and behavior.

**Example:**

class Person {

constructor(name) {

this.name = name;

}

greet() {

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

}

}

const p = new Person('Alice');

p.greet(); // Output: Hello, I am Alice

**ES6 Class Inheritance**

Inheritance allows one class to extend another and reuse its properties and methods. In ES6, the extends keyword is used to create a subclass, and super() is used to call the parent class constructor.

**Example:**

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('Bruno');

dog.speak(); // Output: Bruno barks.

**ES6 Arrow Functions**

Arrow functions provide a shorter syntax for writing functions and do **not bind their own this context**, making them great for callbacks and array methods.

**Syntax:**

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

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

**ES6 set() and map()**

**Set**

* A Set is a collection of **unique values**.
* Duplicate values are automatically ignored.

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

console.log(numbers); // Output: Set { 1, 2, 3 }

**Map**

* A Map is a collection of **key-value pairs**.
* Unlike objects, keys in a Map can be of any type.

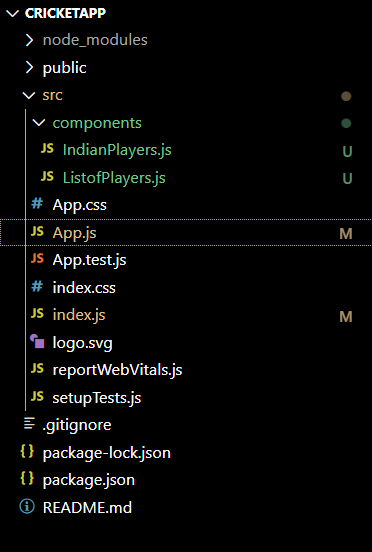
const capitals = new Map();

capitals.set('India', 'New Delhi');

capitals.set('France', 'Paris');

console.log(capitals.get('India')); // Output: New Delhi

**Structure:**



**IndianPlayers.js:**

import React from 'react';

const IndianPlayers = () => {

  const oddPlayers = ['Sachin1', 'Virat3', 'Yuvaraj5'];

  const evenPlayers = ['Dhoni2', 'Rohit4', 'Raina6'];

  const T20Players = ['First Player', 'Second Player', 'Third Player'];

  const RanjiPlayers = ['Fourth Player', 'Fifth Player', 'Sixth Player'];

  const mergedPlayers = [...T20Players, ...RanjiPlayers];

  return (

    <div>

      <h2>Odd Players</h2>

      <ul>

        <li>First : {oddPlayers[0]}</li>

        <li>Third : {oddPlayers[1]}</li>

        <li>Fifth : {oddPlayers[2]}</li>

      </ul>

      <h2>Even Players</h2>

      <ul>

        <li>Second : {evenPlayers[0]}</li>

        <li>Fourth : {evenPlayers[1]}</li>

        <li>Sixth : {evenPlayers[2]}</li>

      </ul>

      <h2>List of Indian Players Merged:</h2>

      <ul>

        {mergedPlayers.map((player, index) => (

          <li key={index}>Mr. {player}</li>

        ))}

      </ul>

    </div>

  );

};

export default IndianPlayers;

**ListofPlayers.js:**

import React from 'react';

const ListofPlayers = () => {

  const players = [

    { name: 'Jack', score: 50 },

    { name: 'Michael', score: 70 },

    { name: 'John', score: 40 },

    { name: 'Ann', score: 61 },

    { name: 'Elisabeth', score: 61 },

    { name: 'Sachin', score: 95 },

    { name: 'Dhoni', score: 100 },

    { name: 'Virat', score: 84 },

    { name: 'Jadeja', score: 64 },

    { name: 'Raina', score: 75 },

    { name: 'Rohit', score: 80 }

  ];

  // List of all players

  const allPlayersList = players.map((player, index) => (

    <li key={index}>Mr. {player.name} {player.score}</li>

  ));

  // Filtered players with score < 70 using arrow function

  const filteredPlayers = players

    .filter(player => player.score < 70)

    .map((player, index) => (

      <li key={index}>Mr. {player.name} {player.score}</li>

    ));

  return (

    <div>

      <h2><strong>List of Players</strong></h2>

      <ul>{allPlayersList}</ul>

      <hr />

      <h2><strong>List of Players having Scores Less than 70</strong></h2>

      <ul>{filteredPlayers}</ul>

    </div>

  );

};

export default ListofPlayers;

**App.js:**

import React from 'react';

import ListofPlayers from './components/ListofPlayers';

function App() {

  return (

    <div className="App">

      <h1>React App</h1>

      <ListofPlayers />

    </div>

  );

}

export default App;

**index.js:**

import React from 'react';

import ReactDOM from 'react-dom/client';

import App from './App';

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(<App />);

**Output:**

