**9.React-HOL**

**List the features of ES6:**

* Arrow functions
* let and const for variable declarations
* Classes and inheritance
* Template literals
* Default parameters
* Destructuring assignment
* Rest and Spread operators
* Promises
* Modules (import/export)
* Map and Set objects

**Explain JavaScript let:**

* let allows you to declare block-scoped variables.
* Unlike var, it does not allow redeclaration in the same scope.

**Identify the differences between var and let:**

| **Feature** | **var** | **let** |
| --- | --- | --- |
| Scope | Function-scoped | Block-scoped |
| Redeclaration | Allowed | Not allowed |
| Hoisting | Yes (initialized as undefined) | Yes (not initialized) |

**Explain JavaScript const:**

* Used to declare constants (block-scoped).
* The value cannot be reassigned but objects/arrays can be mutated.

**Explain ES6 class fundamentals:**

* class keyword is used to define object blueprints.
* Contains constructors and methods.

class Car {

constructor(brand) {

this.brand = brand;

}

display() {

return `Brand is ${this.brand}`;

}

}

**Explain ES6 class inheritance:**

* Uses extends and super() to inherit parent class properties/methods.

lass Car {

constructor(brand) {

this.brand = brand;

}

}

class Model extends Car {

constructor(brand, model) {

super(brand);

this.model = model;

}

}

**Define ES6 arrow functions:**

* Shorter syntax for functions.
* Does **not** bind this.

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

**Identify set() and map():**

* Set: Collection of **unique** values.
* Map: Key-value pairs with keys of any type.

**CODE: ListofPlayers.js**

import React from 'react';

const ListofPlayers = () => {

  const players = [

    { name: "Mr. Jack", score: 50 },

    { name: "Mr. Michael", score: 70 },

    { name: "Mr. John", score: 40 },

    { name: "Mr. Ann", score: 61 },

    { name: "Mr. Elisabeth", score: 61 },

    { name: "Mr. Sachin", score: 95 },

    { name: "Mr. Dhoni", score: 100 },

    { name: "Mr. Virat", score: 84 },

    { name: "Mr. Jadeja", score: 64 },

    { name: "Mr. Raina", score: 75 },

    { name: "Mr. Rohit", score: 80 },

  ];

  const filteredPlayers = players.filter(p => p.score < 70); // using arrow function

  return (

    <div>

      <h2>List of Players</h2>

      <ul>

        {players.map((p, index) => (

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

        ))}

      </ul>

      <hr />

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

      <ul>

        {filteredPlayers.map((p, index) => (

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

        ))}

      </ul>

    </div>

  );

};

export default ListofPlayers;

**IndianPlayers.js**

import React from 'react';

const IndianPlayers = () => {

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

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

  const T20players = [

    "Mr. First Player",

    "Mr. Second Player",

    "Mr. Third Player"

  ];

  const RanjiPlayers = [

    "Mr. Fourth Player",

    "Mr. Fifth Player",

    "Mr. Sixth Player"

  ];

  const merged = [...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>

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

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

        ))}

      </ul>

    </div>

  );

};

export default IndianPlayers;

**app.js**

import React from 'react';

import ListofPlayers from './ListofPlayers';

import IndianPlayers from './IndianPlayers';

function App() {

  const flag = false; // set to false to view IndianPlayers

  return (

    <div className="App">

      <h1>React App</h1>

      {flag ? <ListofPlayers /> : <IndianPlayers />}

    </div>

  );

}

export default App;

**OUTPUT**



