**React**

**Handson 9**

**1.Features of ES6 (ECMAScript 6):**

* **let and const**, block-scoped variable declarations.
* **Arrow Functions,** shorter syntax for functions (() => {}).
* **Classes,** introduced class-based syntax for OOP.
* **Template Literals**, use backticks (`) and ${} for interpolation.
* **Default Parameters,** set default values in functions.
* **Destructuring**, extract values from arrays or objects easily.
* **Spread and Rest Operators**, ... used for expanding or collecting items.
* **Modules (import/export),** enables modular JavaScript code.
* **Promises,** handle asynchronous operations more effectively.
* **for...of Loop,** iterates over iterable objects.
* **Set and Map,** new data structures for unique values and key-value pairs.

**2.Explain javascript let:**

* Let lets you declare block-scoped variables.
* Unlike var, let does not hoist the variable to the top of the scope.
* This is useful to avoid variable leaks or redeclarations in loops and conditions.

let x = 10;

if (true) {

let x = 20;

console.log(x);

}

console.log(x);

**3.Differences between var and let:**

|  |  |  |
| --- | --- | --- |
| **Feature** | **var** | **let** |
| Scope | Function-scoped | Block-scoped |
| Hosting | Hoisted and initialized as undefined | Hoisted but not initialized (TDZ) |
| |  | | --- | | Redeclaration | | Allowed | Not allowed in same scope |
| Becomes window property | Becomes window property | Does not become window property |

**4.Explain JavaScript const:**

* const declares block-scoped constants.
* They must be initialized when declared.
* You cannot reassign them, but you can still modify the contents of objects.

**5.Explain ES6 Class Fundamentals:**

* Syntactic sugar for JavaScript's prototype-based inheritance.
* It supports constructors, methods, and static methods.

class Person {

constructor(name) {

this.name = name;

}

greet() {

console.log(`Hi, I'm ${this.name}`);

}

}

**6.Explain ES6 Class Inheritance:**

* Use extends to create a subclass.
* Use super() to call the parent constructor.

class Student extends Person {

constructor(name, course) {

super(name);

this.course = course;

}

showCourse() {

console.log(`${this.name} studies ${this.course}`);

}

}

**7. Define ES6 arrow functions:**

* A shorter way to write functions.
* It does not have its own "this"; it inherits from the parent scope.

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

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

**output:**

**8**

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

**1.Set:**

* Stores unique values of any type.
* No duplicates are allowed.

**Example:**

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

console.log(mySet);

**2.Map:**

* Stores key-value pairs.
* Keys can be any type, not just strings like in objects.

**Example:**

const myMap = new Map();

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

myMap.set(1, "One");

console.log(myMap.get("name"));

**Output:**

Aish

**// src/IndianPlayers.js:**

import React from 'react';

const IndianPlayers = () => {

  const oddPlayers = ['Ravi', 'Surya', 'Manoj'];

  const evenPlayers = ['Deepak', 'Neeraj', 'Ishaan'];

  const T20players = ['Ravi Sharma', 'Neeraj Kumar'];

  const RanjiTrophyPlayers = ['Ajay Patel', 'Kunal Verma', 'Tarun Singh'];

  const mergedPlayers = [...T20players, ...RanjiTrophyPlayers];

  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, i) => (

          <li key={i}>{player}</li>

        ))}

      </ul>

    </div>

  );

};

export default IndianPlayers;

**// src/ListofPlayers.js:**

import React from 'react';

const players = [

  { name: 'Arjun', score: 92 },

  { name: 'Siddharth', score: 48 },

  { name: 'Karthik', score: 74 },

  { name: 'Ramesh', score: 63 },

  { name: 'Ajay', score: 58 },

  { name: 'Rahul', score: 88 },

  { name: 'Nikhil', score: 67 },

  { name: 'Vivek', score: 71 },

  { name: 'Ankit', score: 61 },

  { name: 'Harsha', score: 53 },

  { name: 'Tejas', score: 97 }

];

const ListofPlayers = () => {

  const lowScorers = players.filter(player => player.score < 70);

  return (

    <div>

      <h2>List of Players</h2>

      <ul>

        {players.map((p, i) => (

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

        ))}

      </ul>

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

      <ul>

        {lowScorers.map((p, i) => (

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

        ))}

      </ul>

    </div>

  );

};

export default ListofPlayers;

**//src/App.js:**

import React from 'react';

import ListofPlayers from './ListofPlayers';

import IndianPlayers from './IndianPlayers';

function App() {

  const flag = true;

  return (

    <div className="App">

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

    </div>

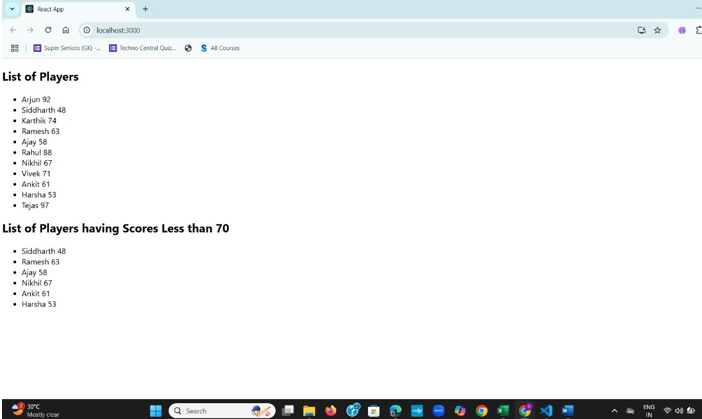
  );

}

export default App;

**Output:**

**When flag is true:**

****

**When flag is false:**

**A screenshot of a computer

AI-generated content may be incorrect.**