**WEEK – 07**

**ReactJS-HOL**

**Superset ID: 6262264**

**EXERCISE 9:**

**OBJECTIVES :**

1. **List the Features of ES6**

ES6 (ECMAScript 2015) introduced many useful features to modernize JavaScript:

* let and const for block-scoped variables
* Arrow functions (()=>{})
* Template literals (`Hello, ${name}`)
* Default function parameters
* Destructuring assignment
* Spread (...arr) and Rest (...args) operators
* Classes and class inheritance
* Promises for asynchronous code
* Modules (import, export)
* New collections: Set, Map
* Enhanced object literals
* for...of loop

1. **Explain JavaScript let**

* let declares variables with block-level scope.
* Variables declared with let can be updated but cannot be re-declared in the same scope.

**EXAMPLE:**

let name = "Ruby";

name = "Developer"; // Allowed

let name = "Engineer"; // Error in same block

1. **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 |
| Use | Outdated | Preferred in modern JS |

1. **Explain JavaScript const.**

* const is used for constant variables with block scope.
* Once assigned, the value cannot be changed.
* For objects, properties can be modified, but the reference stays constant.

**EXAMPLE:**

const name = "Ruby";

const user = { id: 101 };

user.id = 102; // Allowed

user = {}; // Not allowed

1. **Explain ES6 Class Fundamentals**

* ES6 introduced class to define object blueprints using constructors and methods.

**EXAMPLE:**

class Person {

constructor(name) {

this.name = name;

}

greet() {

return `Hello, ${this.name}`;

}

}

const p = new Person("Ruby");

console.log(p.greet()); // Hello, Ruby

**6. Explain ES6 Class Inheritance.**

* Inheritance is done using the extends keyword.
* super() is used to call the parent constructor.

**EXAMPLE:**

class Student extends Person {

constructor(name, department) {

super(name);

this.department = department;

}

getDetails() {

return `${this.name} belongs to ${this.department} department`;

}

}

const s1 = new Student("Ruby", "EEE");

console.log(s1.getDetails()); // Ruby belongs to EEE department.

**7.Define ES6 Arrow Functions.**

* Arrow functions are shorter and don't bind their own this.

**EXAMPLE:**

const greet = name => `Hello, ${name}`;

console.log(greet("Ruby")); // Hello, Ruby.

**8. Identify Set() and Map()**

**Set:**

* Collection of unique values (no duplicates).

**EXAMPLE:**

const ids = new Set();

ids.add(1);

ids.add(2);

ids.add(1); // Ignored

console.log(ids); // Set(2) {1, 2}

**Map:**

* Stores key-value pairs.
* Keys can be strings, numbers, or objects.

**EXAMPLE:**

const profile = new Map();

profile.set("name", "Ruby");

profile.set("role", "Engineer");

console.log(profile.get("name")); // Ruby

**Create a React Application named “cricketapp” with the following components:**

***ListofPlayers.js (for flag = true):***

import React from 'react';

import '../App.css';

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 below70 = players.filter(player => player.score < 70);

return (

<div className="App">

<h2>List of Players</h2>

<ul>

{players.map((p, i) => (

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

))}

</ul>

<hr />

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

<ul>

{below70.map((p, i) => (

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

))}

</ul>

</div>

);

};

export default ListofPlayers;

***IndianPlayers.js (for flag = false):***

import React from 'react';

import '../App.css';

const IndianPlayers = () => {

const T20 = ['Sachin1', 'Dhoni2', 'Virat3'];

const Ranji = ['Rohit4', 'Yuvraj5', 'Raina6'];

const AllPlayers = [...T20, ...Ranji];

const positionWords = ['First', 'Second', 'Third', 'Fourth', 'Fifth', 'Sixth'];

const OddPlayers = AllPlayers.filter((\_, i) => i % 2 === 0); // index: 0,2,4

const EvenPlayers = AllPlayers.filter((\_, i) => i % 2 !== 0); // index: 1,3,5

return (

<div className="App">

<h2>Odd Players</h2>

<ul>

{OddPlayers.map((player, i) => (

<li key={i}>{positionWords[i \* 2]} : {player}</li>

))}

</ul>

<h2>Even Players</h2>

<ul>

{EvenPlayers.map((player, i) => (

<li key={i}>{positionWords[i \* 2 + 1]} : {player}</li>

))}

</ul>

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

<ul>

{AllPlayers.map((\_, i) => (

<li key={i}>Mr. {positionWords[i]} Player</li>

))}

</ul>

</div>

);

};

export default IndianPlayers;

***App.js (Switch Components Using flag):***

import './App.css';

import ListofPlayers from './components/ListofPlayers';

import IndianPlayers from './components/IndianPlayers';

function App()

{

const flag = true; **// Set false to show IndianPlayers component**

return (

<div className="App">

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

</div>

);

}

export default App;

***App.css (Add This Styling):***

.App

{

font-family: Arial, sans-serif;

padding: 20px;

}

h2

{

font-weight: bold;

margin-top: 30px;

}

hr

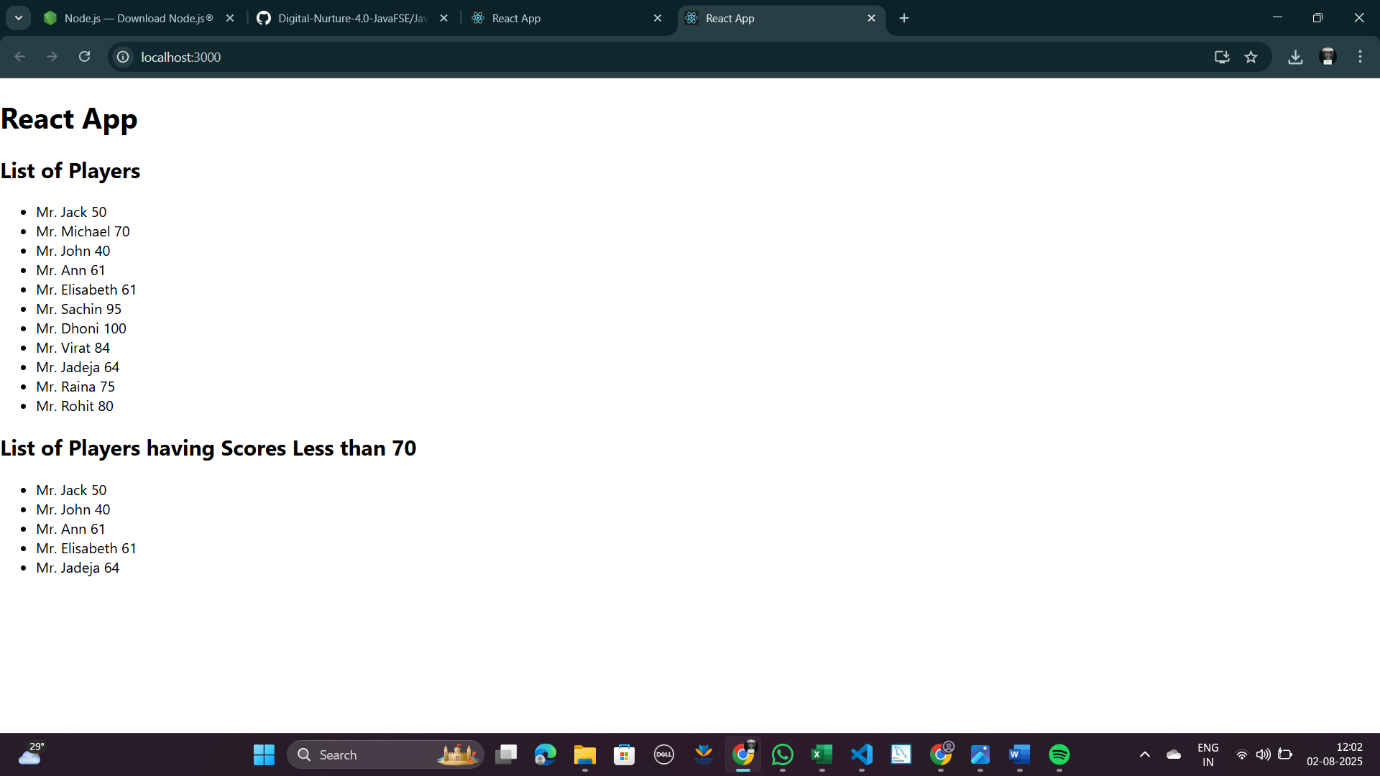
{

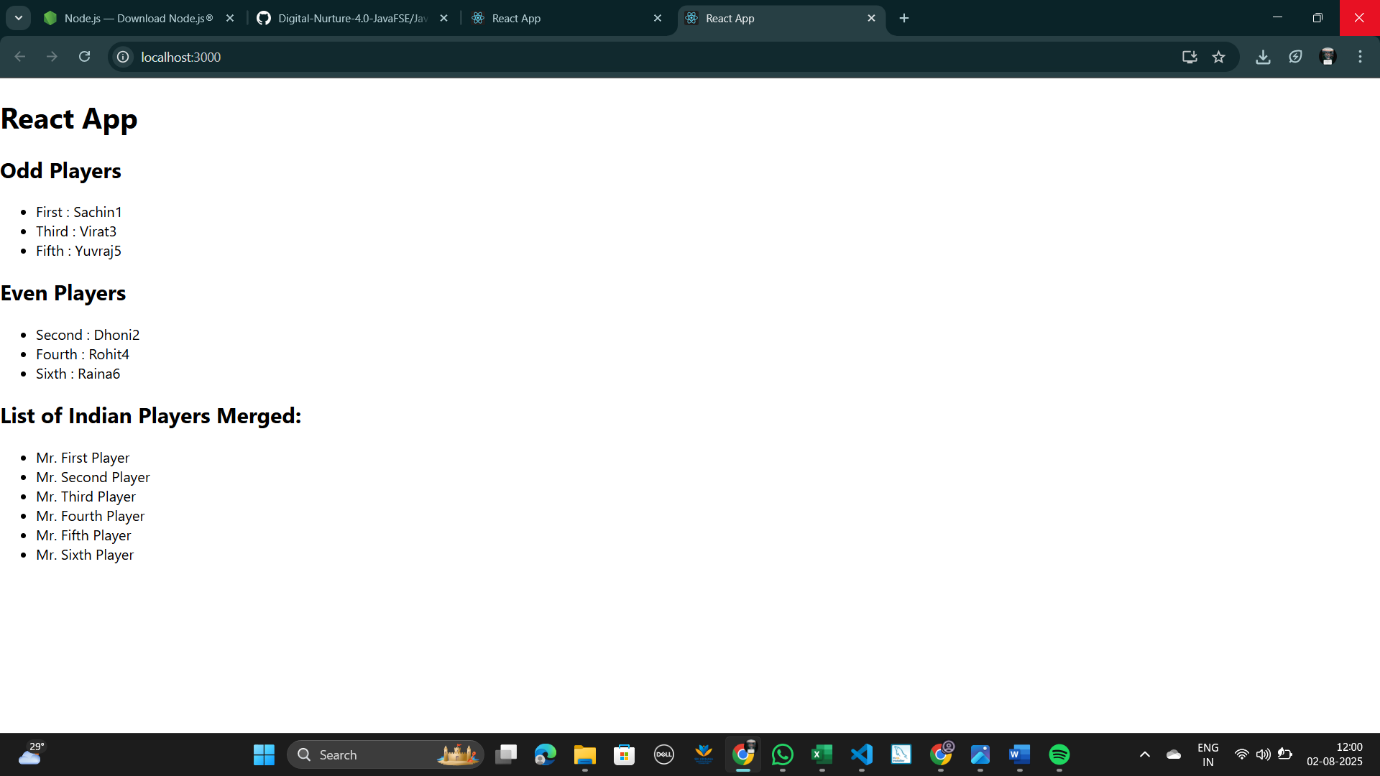
margin: 20px 0;

border: 1px solid #ccc;

}

**OUTPUT:**

****

****

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**