**1. Features of ES6**

ECMAScript 6, also known as ES6 or ECMAScript 2015, introduced significant improvements to the JavaScript language. One of the most notable features is block-scoped variable declarations using let and const, which replaced the problematic var keyword used in older versions.

Another powerful addition is the arrow function syntax (=>), which allows for more concise and readable function expressions, especially for callback functions. Template literals, using backticks (`), make it easy to embed expressions within strings and write multi-line strings without escaping characters.

ES6 also introduced destructuring assignments, default parameters, spread/rest operators, new data structures like Set and Map, classes and class inheritance, modules , Promises for async programming, and enhanced object literals.

**2. Explain JavaScript let**

The let keyword was introduced in ES6 to improve the way variables are declared. Unlike var, which has function scope, let has block scope, meaning it is only accessible within the block or statement it is declared in.

Variables declared using let can be updated later, but they cannot be re-declared in the same scope. Another benefit is that it avoids the issue of hoisting confusion, where variables declared with var can be accessed before declaration with an undefined value.

**3. Differences Between var and let**

The primary difference between var and let lies in their scope and behavior. var is function-scoped, while let is block-scoped. Variables declared with var can be re-declared within the same scope, whereas let will throw an error if you try to redeclare it.

While both are hoisted, let is not initialized at the time of hoisting. This creates a 'temporal dead zone' where the variable exists but cannot be accessed until the declaration is evaluated.

**4. Explain JavaScript const**

The const keyword is used to declare variables that are read-only and block-scoped, similar to let. Once a variable is declared with const, it must be initialized and cannot be reassigned afterward.

However, if a const variable holds an object or an array, the contents of the object or array can still be modified. Only the reference to the object is fixed.

**5. ES6 Class Fundamentals**

ES6 introduced the class syntax to provide a more intuitive and cleaner way to create objects and deal with inheritance. A class typically includes a constructor() method that is automatically called when an object is instantiated using the new keyword.

Although JavaScript is not a class-based language at its core, the ES6 class syntax provides syntactic sugar that helps developers write more organized and readable code.

**6. ES6 Class Inheritance**

In ES6, class inheritance is implemented using the extends keyword. This allows one class to inherit from another, enabling code reuse and structure.

The subclass can use the super() keyword to call the parent class’s constructor and methods. This is useful when building on the functionality of a parent class while adding or modifying certain features in the child class.

**7. ES6 Arrow Functions**

Arrow functions are a concise way to write function expressions using the => syntax. They do not bind their own this context and capture the this value of the enclosing lexical context.

They can also omit parentheses for single parameters and use implicit returns for single expressions, making code more compact and readable.

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

Set is a collection of unique values. It does not allow duplicates and maintains insertion order. Common methods include add(), delete(), has(), and clear().

Map is a collection of key-value pairs, where keys can be of any type. It preserves insertion order and provides methods like set(), get(), delete(), and has().

# **CricketApp Project Code**

**1. App.js**

import React from 'react';  
import ListofPlayers from './components/ListofPlayers';  
import IndianPlayers from './components/IndianPlayers';  
  
function App() {  
 const flag = true; // Change to false to switch view  
  
 return (  
 <div className="App">  
 {flag ? <ListofPlayers /> : <IndianPlayers />}  
 </div>  
 );  
}  
  
export default App;

**2. 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 />);

**3. ListofPlayers.js**

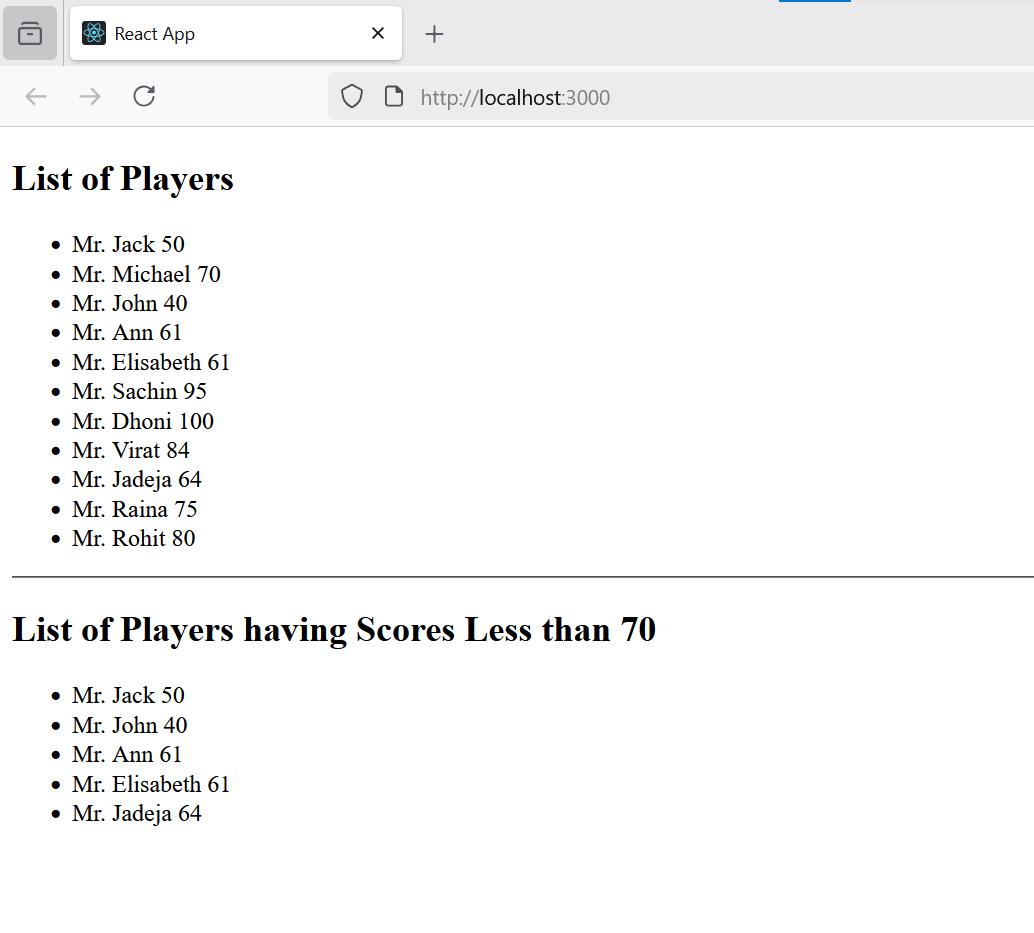
import React from 'react';  
  
const ListofPlayers = () => {  
 const players = [  
 { name: "Mr. Jack", score: 50 },  
 { name: "Mr. Michael", score: 70 },  
 { name: "Mr. John", score: 40 },  
 { name: "Mr. Ann", score: 61 },  
 { name: "Mr. Elisabeth", score: 61 },  
 { name: "Mr. Sachin", score: 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 lowScorers = players.filter(player => player.score < 70);  
  
 return (  
 <div>  
 <h2>List of Players</h2>  
 <ul>  
 {players.map((player, index) => (  
 <li key={index}>{player.name} {player.score}</li>  
 ))}  
 </ul>  
 <hr />  
 <h2>List of Players having Scores Less than 70</h2>  
 <ul>  
 {lowScorers.map((player, index) => (  
 <li key={index}>{player.name} {player.score}</li>  
 ))}  
 </ul>  
 </div>  
 );  
};  
  
export default ListofPlayers;

**4. IndianPlayers.js**

import React from 'react';  
  
const IndianPlayers = () => {  
 const teamPlayers = [  
 "Sachin1", "Dhoni2", "Virat3",  
 "Rohit4", "Yuvraj5", "Raina6"  
 ];  
  
 const [first, second, third, fourth, fifth, sixth] = teamPlayers;  
  
 const T20Players = [  
 "Mr. First Player",  
 "Mr. Second Player",  
 "Mr. Third Player"  
 ];  
  
 const RanjiPlayers = [  
 "Mr. Fourth Player",  
 "Mr. Fifth Player",  
 "Mr. Sixth Player"  
 ];  
  
 const mergedPlayers = [...T20Players, ...RanjiPlayers];  
  
 return (  
 <div>  
 <h2>Odd Players</h2>  
 <p>First : {first}</p>  
 <p>Third : {third}</p>  
 <p>Fifth : {fifth}</p>  
 <hr />  
 <h2>Even Players</h2>  
 <p>Second : {second}</p>  
 <p>Fourth : {fourth}</p>  
 <p>Sixth : {sixth}</p>  
 <hr />  
 <h2>List of Indian Players Merged:</h2>  
 <ul>  
 {mergedPlayers.map((player, index) => (  
 <li key={index}>{player}</li>  
 ))}  
 </ul>  
 </div>  
 );  
};  
  
export default IndianPlayers;

**Output**

When flag=true;



When flag=false;

