**REACT ASSIGNMENT**

**Features of ES6**

ES6 (ECMAScript 2015) brought major improvements to JavaScript, making it more powerful and developer-friendly. Key features include let and const for block-scoped variable declarations, arrow functions for shorter syntax and lexical this, classes and inheritance for object-oriented programming, template literals for cleaner string handling, destructuring for easy variable extraction from arrays/objects, default function parameters, rest/spread operators, and built-in support for modules (import/export). It also introduced Promises for better asynchronous handling and new data structures like Map and Set.

**JavaScript let**

The let keyword in JavaScript is used to declare variables that are limited to the block scope in which they are defined, making it safer and more predictable than var. Unlike var, which is function-scoped and hoisted with an initial value of undefined, let declarations remain in a "temporal dead zone" until initialized, preventing accidental usage before declaration. It is especially useful in loops and conditional blocks where variable scope matters.

**Difference Between var and let**

The primary difference between var and let is scope: var is function-scoped, while let is block-scoped. Variables declared with var are hoisted and initialized with undefined, which can lead to bugs if accessed before declaration. In contrast, let variables are also hoisted but not initialized, and accessing them before declaration results in a reference error. Additionally, let prevents redeclaration in the same scope, unlike var, making code cleaner and less error-prone.

**JavaScript const**

The const keyword in JavaScript is used to declare block-scoped constants that cannot be reassigned after initialization. Like let, it has block scope and is not hoisted in the traditional sense. However, while the variable identifier can't be reassigned, objects or arrays declared with const can still have their contents modified. It's ideal for values that should remain constant, enhancing code clarity and stability.

**ES6 Class Fundamentals**

ES6 introduced the class syntax as a more intuitive and structured way to create objects and handle inheritance in JavaScript. Classes encapsulate data and behavior in a single structure using a constructor method for initialization and define methods directly within the class body. This syntax is cleaner and closer to traditional object-oriented programming languages, making code more readable and easier to manage.

**ES6 Class Inheritance**

ES6 supports class inheritance using the extends and super keywords. A subclass can inherit properties and methods from a parent class, enabling code reuse and the creation of hierarchical structures. The extends keyword establishes the inheritance, and super() is used within the constructor of the child class to call the parent class's constructor. Method overriding is also supported, allowing subclasses to customize behavior.

**Arrow Functions**

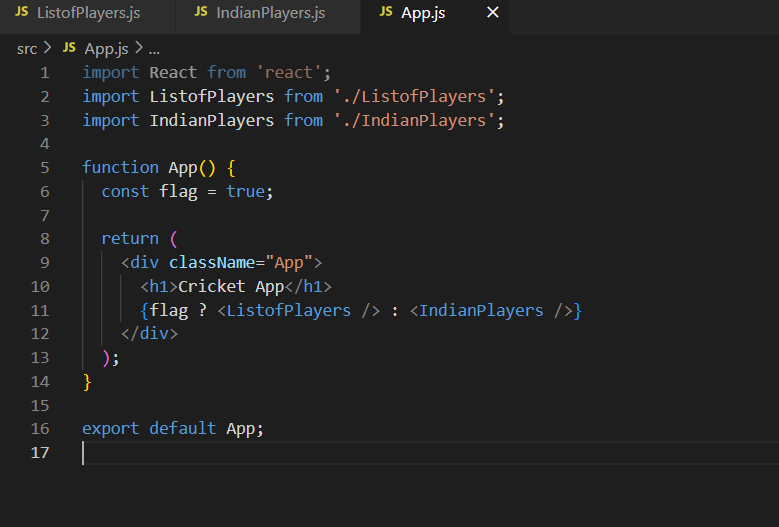
Arrow functions (=>) offer a concise syntax for writing functions in ES6 and inherit the this value from their surrounding context, unlike regular functions that have their own this. They are particularly useful in callbacks and array methods like map, filter, and reduce. Arrow functions do not have their own arguments object and cannot be used as constructors, which makes them unsuitable for certain use cases but ideal for short, functional code snippets.

**Set() and Map()**

Set and Map are new data structures introduced in ES6. A Set is a collection of unique values where duplicates are automatically removed, and it's useful for storing distinct items. A Map stores key-value pairs, where keys can be of any data type, unlike objects which only allow string keys. Both structures provide improved performance and cleaner syntax for managing collections of data compared to traditional objects and arrays.

**HANDS-ON:**

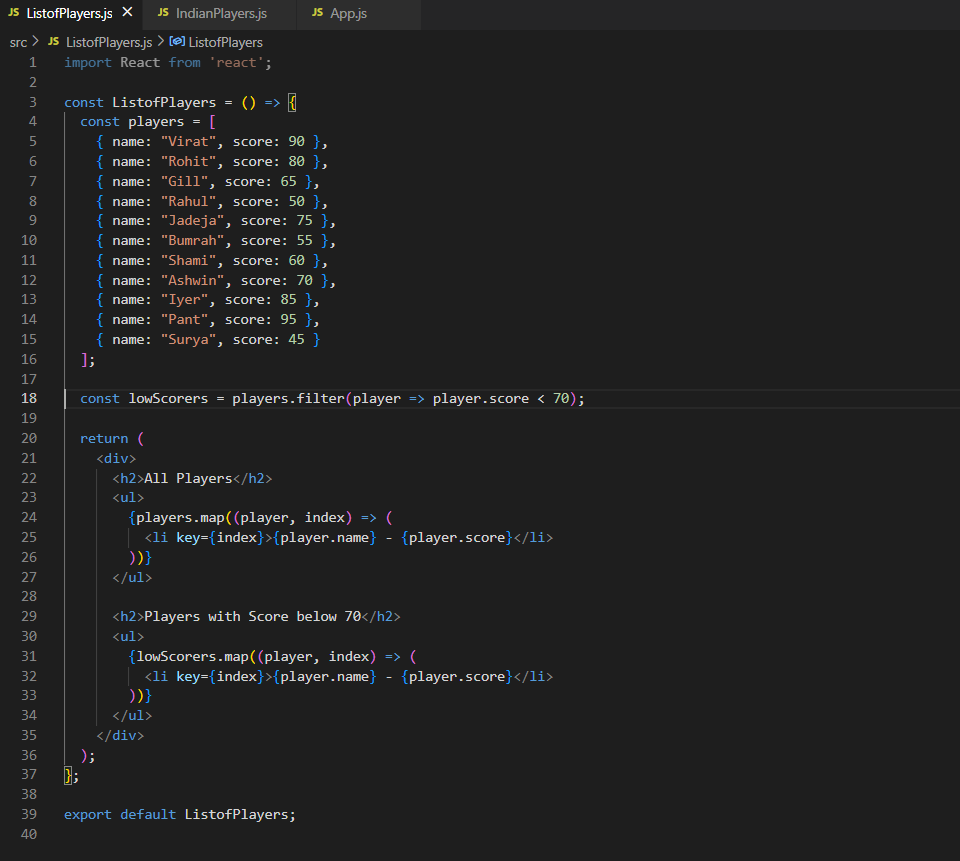
**App.js:**

****

**IndianPlayers.js:**

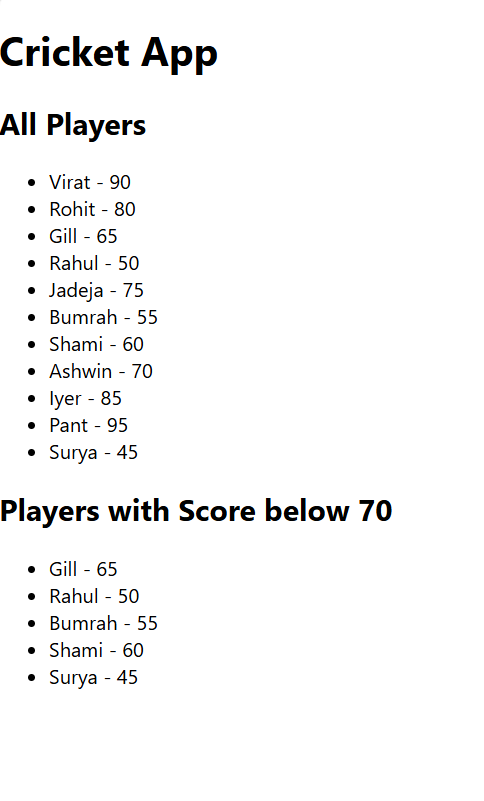
****

**ListofPlayers.js:**

****

**OUTPUT:**

Flag=true

****

Flag=false

