**Features of ES6**

* let and const for better variable declaration.
* Arrow functions for shorter function syntax.
* Template literals using backticks for string interpolation.
* Classes and inheritance for object-oriented programming.
* Default parameters in functions.
* Destructuring assignment for arrays and objects.
* New data structures like Map and Set.
* Modules for importing and exporting code.

**JavaScript let**

let is used to declare variables that are block-scoped. This means the variable only exists inside the block { } where it is defined. It is preferred over var because it prevents accidental overwriting and provides better control of variable lifetime.

**Difference Between var and let**

* **Scope**:
  + var is function-scoped.
  + let is block-scoped.
* **Hoisting**:
  + var variables are hoisted (moved to the top) and initialized with undefined.
  + let variables are hoisted but not initialized (cannot be accessed before declaration).
* **Re-declaration**:
  + var can be re-declared in the same scope.
  + let cannot be re-declared in the same scope.

**JavaScript const**

const is used to declare variables that cannot be reassigned. Like let, it is block-scoped. However, for objects and arrays declared with const, you can modify their contents but cannot reassign the variable to a new object or array.

**ES6 Class Fundamentals**

Classes in ES6 provide a cleaner way to create objects and handle inheritance. A class can contain:

* A constructor method (runs when an object is created).
* Methods (functions inside the class).
* Properties that belong to the object.

It simplifies object-oriented programming compared to older prototype-based approaches.

**ES6 Class Inheritance**

ES6 allows one class to inherit from another using the extends keyword. The super() method is used to call the parent class’s constructor. This helps reuse code and create hierarchies like Animal → Dog or Vehicle → Car.

**ES6 Arrow Functions**

Arrow functions are a shorter way to write functions using the => syntax.  
Example:

javascript

CopyEdit

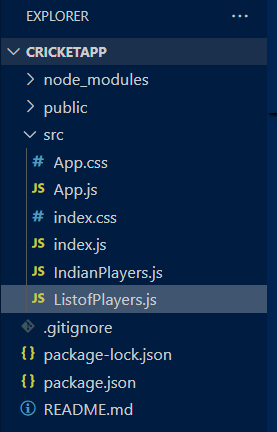
const add = (a, b) => a + b;

They also do not have their own this context, which makes them useful for callbacks and event handlers.

**Set() and Map()**

* Set: A collection of unique values. It automatically removes duplicates.  
  Example: new Set([1,2,2,3]) → {1,2,3}
* Map: A collection of key-value pairs where keys can be any data type (unlike objects that only allow strings or symbols as keys).  
  Example: new Map([['name','John'],['age',30]])

FILE STRUCTURE



**ListOfPlayers.js**

import React from "react";

const ListofPlayers = () => {

  // Players with names and scores

  const players = [

    { name: "Virat Kohli", score: 95 },

    { name: "Rohit Sharma", score: 88 },

    { name: "KL Rahul", score: 65 },

    { name: "Shreyas Iyer", score: 40 },

    { name: "Hardik Pandya", score: 75 },

    { name: "Ravindra Jadeja", score: 60 },

    { name: "R Ashwin", score: 55 },

    { name: "Bhuvneshwar Kumar", score: 35 },

    { name: "Jasprit Bumrah", score: 78 },

    { name: "Mohammed Shami", score: 69 },

    { name: "Axar Patel", score: 82 }

  ];

  // Filter players with score below 70

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

  return (

    <div style={styles.container}>

      <h2>List of Players</h2>

      <h3>All Players</h3>

      <ul style={styles.list}>

        {players.map((p, i) => (

          <li key={i}>

            {p.name} - {p.score}

          </li>

        ))}

      </ul>

      <h3>Players Scored Below 70</h3>

      <ul style={styles.list}>

        {lowScorers.map((p, i) => (

          <li key={i}>

            {p.name} - {p.score}

          </li>

        ))}

      </ul>

    </div>

  );

};

const styles = {

  container: {

    textAlign: "center",

    marginTop: "20px",

    fontFamily: "Arial"

  },

  list: {

    listStyle: "none",

    padding: 0

  }

};

export default ListofPlayers;

**IndianPlayers.js**

import React from "react";

const IndianPlayers = () => {

  // Odd and Even team using destructuring

  const team = ["Virat", "Rohit", "Rahul", "Shreyas", "Hardik", "Jadeja"];

  const [p1, p2, p3, p4, p5, p6] = team;

  const oddPlayers = [p1, p3, p5];

  const evenPlayers = [p2, p4, p6];

  // Merge T20 and Ranji Trophy players

  const T20players = ["Virat", "Rohit", "Hardik"];

  const RanjiTrophyPlayers = ["Pujara", "Rahane", "Ishant"];

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

  return (

    <div style={styles.container}>

      <h2>Odd Team Players</h2>

      <ul style={styles.list}>

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

          <li key={i}>{player}</li>

        ))}

      </ul>

      <h2>Even Team Players</h2>

      <ul style={styles.list}>

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

          <li key={i}>{player}</li>

        ))}

      </ul>

      <h2>Merged Players (T20 + Ranji Trophy)</h2>

      <ul style={styles.list}>

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

          <li key={i}>{player}</li>

        ))}

      </ul>

    </div>

  );

};

const styles = {

  container: {

    textAlign: "center",

    marginTop: "20px",

    fontFamily: "Arial"

  },

  list: {

    listStyle: "none",

    padding: 0

  }

};

export default IndianPlayers;

**App.js**

import React from "react";

import ListofPlayers from "./ListofPlayers";

import IndianPlayers from "./IndianPlayers";

function App() {

  return (

    <div>

      <ListofPlayers />

      <hr />

      <IndianPlayers />

    </div>

  );

}

export default App;

A screenshot of a computer

AI-generated content may be incorrect.

**OUTPUT ON THE BROWSER**

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.