**Exercise 9**

**Objectives**

1. **List the features of ES6.**

ES6 (ECMAScript 2015) introduced major improvements to JavaScript. It added **let, const**, arrow functions, default parameters, and template literals. Introduced **classes**, **modules**, and **enhanced object literals**. New data structures like **Set** and **Map** were added. It also brought **destructuring**, **spread/rest operators**, and **Promises** for async programming.

1. **Explain JavaScript let.**

**let** is used to declare block-scoped variables in JavaScript. Unlike **var**, variables declared with **let** are not hoisted to the top of their scope. **let** can be updated but **not re-declared** in the same scope. Useful in loops and conditional blocks for cleaner and predictable scoping. Prevents common bugs caused by variable leakage outside blocks.

1. **Identify the differences between var and let**

var is function-scoped, while let is block-scoped. var declarations are hoisted and initialized with undefined; let is not initialized. let helps avoid accidental re-declarations, while var allows it. let is more predictable in loops and asynchronous operations. Overall, let is safer and preferred over var in modern JavaScript.

1. **Explain JavaScript const**

const declares **read-only** block-scoped variables. The variable **must be initialized** at the time of declaration. It cannot be reassigned, but if it's an object or array, its **contents can be modified**. Useful for defining constants and values that shouldn't change. Helps make code more predictable and easier to debug.

1. **Explain ES6 class fundamentals**

ES6 introduces class syntax as syntactic sugar over prototype-based OOP. A class contains a constructor and methods, defined using the class keyword. You can create objects using the new keyword with the class name. Classes support encapsulation, inheritance, and clean code organization. Makes object-oriented JavaScript more readable and familiar to Java/C++ users.

1. **Explain ES6 class inheritance**

You can extend a class using the extends keyword to create child classes. The super() function is used inside the child constructor to call the parent constructor. Inherited methods and properties can be overridden or reused. Promotes code reuse and modular design patterns. Inheritance in ES6 follows a more classical model, unlike prototypes in older JS.

1. **Define ES6 arrow functions**

Arrow functions are a shorter syntax for writing functions using =>. They do **not have their own this** — they inherit this from the parent scope. Best used for callbacks, array operations, and simple one-liners. Cannot be used as constructors and don’t bind arguments. Improves code readability and reduces boilerplate in function expressions.

1. **Identify set(), map()**

Set is a collection of unique values — duplicates are not allowed. Map is a collection of key-value pairs with keys of any type. Both preserve insertion order, unlike plain objects. Set methods include .add(), .has(), .delete() and Map includes .set(), .get(). Useful for faster lookups, uniqueness checks, and structured data storage.

**Hands On Practice**

1. **Create a React Application named “cricketapp”**

npx create-react-app cricketapp

1. **ListofPlayers**

* **Declare an array with 11 players and store details of their names and scores using the map feature of ES6**
* **Filter the players with scores below 70 using arrow functions of ES6**.

**ListofPlayers.js**

import React from 'react';

import players from './PlayersData';

function ListofPlayers(){

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

  return(

    <div>

      <h2>All Players</h2>

      <ul>

        {players.map((player,index) => (

          <li key={index}>{player.name} - {player.score}</li>

        ))}

      </ul>

      <h3>Players with score less than 70</h3>

      <ul>

        {lowScorers.map((player,index) => (

          <li key={index}>{player.name} - {player.score}</li>

        ))}

      </ul>

    </div>

  );

}

export default ListofPlayers;

1. **IndianPlayers**

* **Display the Odd Team Player and Even Team players using the Destructuring features of ES6**
* **Declare two arrays T20players and RanjiTrophy players and merge the two arrays and display them using the Merge feature of ES6**

**IndianPlayers.js**

import React from 'react';

import players from './PlayersData';

function IndianPlayers()

{

  const allPlayers = players;

  const oddPlayers = allPlayers.filter((\_,index) => index % 2 !== 0);

  const evenPlayers = allPlayers.filter((\_,index) => index % 2 === 0);

  const[odd1, odd2, odd3, ... restOdd] = oddPlayers;

  const[even1, even2, even3,... restEven] = evenPlayers;

  const T20Players = allPlayers.slice(0, 5);

  const RanjiPlayers = allPlayers.slice(5);

  const mergedPlayers = [...T20Players, ...RanjiPlayers];

  return(

    <div>

      <h2>Odd Players</h2>

      <ul>

        <li>{odd1.name}</li>

        <li>{odd2.name}</li>

        <li>{odd3.name}</li>

        {restOdd.map((players, index) => (

          <li key={index}>{players.name}</li>

        ))}

      </ul>

      <h2>Even Players</h2>

      <ul>

        <li>{even1.name}</li>

        <li>{even2.name}</li>

        <li>{even3.name}</li>

        {restEven.map((players,index) => (

        <li key={index}>{players.name}</li>

        ))}

      </ul>

      <h2>Merged T20 Players and Ranji Players</h2>

      <ul>

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

          <li key={index}>{player.name} - {player.score}</li>

        ))}

      </ul>

    </div>

  );

}

export default IndianPlayers;

1. **Output**

When Flag=true

**App.js**

import React from 'react';

import IndianPlayers from './IndianPlayers';

import ListofPlayers from './ListofPlayers';

function App()

{

  const flag = true;

  return(

    <div>

      <h1>Cricket App</h1>

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

    </div>

  );

}

export default App;

A screenshot of a computer

AI-generated content may be incorrect.

When Flag=false

**App.js**

import React from 'react';

import IndianPlayers from './IndianPlayers';

import ListofPlayers from './ListofPlayers';

function App()

{

  const flag = false;

  return(

    <div>

      <h1>Cricket App</h1>

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

    </div>

  );

}

export default App;

A screenshot of a computer

AI-generated content may be incorrect.A white background with black and white clouds

AI-generated content may be incorrect.