Week – 7 React HandsOn

## **Objectives**

* List the features of ES6
* **let** and **const** for variable declarations.
* **Arrow functions** (()=>{}) for shorter function syntax and lexical this.
* **Classes** (class MyClass {}) for OOP-style programming.
* **Template literals** using backticks `Hello ${name}`.
* **Default parameters** in functions.
* Explain JavaScript let
* Block-scoped (limited to {} where declared).
* Can be updated but **not** re-declared in the same scope.
* Identify the differences between var and let

|  |  |  |
| --- | --- | --- |
| **Feature** | **var** | **let** |
| Scope | Function-scoped | Block-scoped |
| Hoisting | Hoisted (initialized undefined) | Hoisted but not initialized |
| Redeclaration | Allowed in same scope | Not allowed in same scope |

* Explain JavaScript const
* Block-scoped like let.
* **Must** be assigned a value at declaration.
* Cannot be reassigned, but object/array contents can be mutated.
* Explain ES6 class fundamentals

class Player {

constructor(name, score) {

this.name = name;

this.score = score;

}

display() {

return `${this.name} scored ${this.score}`;

}

}

* Explain ES6 class inheritance

class Cricketer extends Player {

constructor(name, score, type) {

super(name, score);

this.type = type;

}

info() {

return `${this.display()} in ${this.type}`;

}

}

* Define ES6 arrow functions

const getScore = (score) => score > 50 ? "High" : "Low";

* Identify set(), map()

let s = new Set([1,2,3,3]); // {1,2,3}

let m = new Map();

m.set("Virat", 100);

m.get("Virat"); // 100

9.React HandsOn

// src/App.js

import React from "react";

import ListofPlayers from "./ListofPlayers";

import Scorebelow70 from "./Scorebelow70";

import OddPlayers from "./OddPlayers";

import EvenPlayers from "./EvenPlayers";

import ListofIndianPlayers from "./ListofIndianPlayers";

export default function App() {

const flag = false; // change to false to switch display

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 IndianTeam = [

"Sachin1",

"Dhoni2",

"Virat3",

"Rohit4",

"Yuvaraj5",

"Raina6"

];

const IndianPlayers = [

"Mr. First Player",

"Mr. Second Player",

"Mr. Third Player",

"Mr. Fourth Player",

"Mr. Fifth Player",

"Mr. Sixth Player"

];

if (flag) {

return (

<div>

<h1>List of Players</h1>

<ListofPlayers players={players} />

<hr />

<h1>List of Players having Scores Less than 70</h1>

<Scorebelow70 players={players} />

</div>

);

} else {

return (

<div>

<h1>Indian Team</h1>

<div>

<h1>Odd Players</h1>

<OddPlayers team={IndianTeam} />

<hr />

<h1>Even Players</h1>

<EvenPlayers team={IndianTeam} />

</div>

<hr />

<div>

<h1>List of Indian Players Merged:</h1>

<ListofIndianPlayers players={IndianPlayers} />

</div>

</div>

);

}

}

// src/EvenPlayers.js

import React from "react";

export default function EvenPlayers({ team }) {

const [, second, , fourth, , sixth] = team;

return (

<div>

<p>Second : {second}</p>

<p>Fourth : {fourth}</p>

<p>Sixth : {sixth}</p>

</div>

);

}

// src/ListofIndianPlayers.js

import React from "react";

export default function ListofIndianPlayers({ players }) {

// Merge example

const extraPlayers = [

"Mr. Seventh Player",

"Mr. Eighth Player"

];

const mergedPlayers = [...players, ...extraPlayers]; // Spread operator

return (

<ul>

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

<li key={i}>{p}</li>

))}

</ul>

);

}

// src/ListofPlayers.js

import React from "react";

export default function ListofPlayers({ players }) {

return (

<ul>

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

<li key={i}>

{p.name} {p.score}

</li>

))}

</ul>

);

}

// src/OddPlayers.js

import React from "react";

export default function OddPlayers({ team }) {

// Destructuring

const [first, , third, , fifth] = team;

return (

<div>

<p>First : {first}</p>

<p>Third : {third}</p>

<p>Fifth : {fifth}</p>

</div>

);

}

// src/Scorebelow70.js

import React from "react";

export default function Scorebelow70({ players }) {

// Arrow function + filter

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

return (

<ul>

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

<li key={i}>

{p.name} {p.score}

</li>

))}

</ul>

);

}

10. ReactJS-HOL

### **1. Define JSX**

JSX (**JavaScript XML**) is a syntax extension for JavaScript used in React to describe UI elements in a way that looks similar to HTML.

### **2. Explain ECMA Script**

ECMAScript is the standardized scripting language specification that JavaScript implements. ES6 (ECMAScript 2015) introduced modern features such as:

* let and const
* Arrow functions
* Template literals
* Classes
* Destructuring
* Modules (import/export)

### **3. Explain React.createElement()**

React.createElement() is the core React API used to create React elements.

### **4. Explain how to create React nodes with JSX**

In JSX, React nodes (UI elements) are created using HTML-like tags inside JavaScript code.

### **5. Define how to render JSX to DOM**

React uses ReactDOM.createRoot() and .render() to mount JSX into a specific DOM element:

### **6. Explain how to use JavaScript expressions in JSX**

Any valid JavaScript expression can be placed inside {} in JSX.

### **7. Explain how to use inline CSS in JSX**

Inline styles in JSX are written as an object, not a string, with property names in camelCase.

App.js

// src/App.js

import React from "react";

import "./App.css";

export default function App() {

// Heading

const element = "Office Space";

// Office list

const offices = [

{ Name: "DBS", Rent: 50000, Address: "Chennai", img: "/images/office1.jpg" },

{ Name: "Regus", Rent: 65000, Address: "Bangalore", img: "/images/office2.jpg" },

{ Name: "WeWork", Rent: 55000, Address: "Mumbai", img: "/images/office3.jpg" }

];

return (

<div>

<h1>{element}, at Affordable Range</h1>

{offices.map((office, index) => {

// Conditional rent color

const rentStyle = {

color: office.Rent <= 60000 ? "red" : "green"

};

return (

<div key={index} className="office-card">

<img

src={office.img}

width="25%"

height="25%"

alt={office.Name}

/>

<h2>Name: {office.Name}</h2>

<h3 style={rentStyle}>Rent: Rs. {office.Rent}</h3>

<h3>Address: {office.Address}</h3>

</div>

);

})}

</div>

);

}

App.css

.office-card {

margin-bottom: 30px;

}

11. ReactJS-HOL

### **1. Explain React Events**

React events are similar to DOM events but work consistently across all browsers because they’re wrapped in **SyntheticEvent** objects. Examples include onClick, onChange, onSubmit, etc.

### **2. Explain Event Handlers**

Event handlers are functions that run when an event is triggered. In React, you attach them as props to elements:

### **3. Define Synthetic Event**

A Synthetic Event is React’s wrapper around the native browser event, ensuring cross-browser compatibility and performance. All React events are instances of SyntheticEvent.

### **4. Identify React Event Naming Convention**

* Event names in JSX use **camelCase** (e.g., onClick, not onclick).
* The handler is passed as a **function reference**, not as a string.

App.js

import React, { Component } from 'react';

import CurrencyConvertor from './CurrencyConvertor';

class App extends Component {

constructor(props) {

super(props);

this.state = {

counter: 5

};

}

// Increments counter and says hello

increment = () => {

this.setState({ counter: this.state.counter + 1 });

alert("Hello! Member1");

};

// Decrements counter

decrement = () => {

this.setState({ counter: this.state.counter - 1 });

};

// Function with argument

sayMessage = (msg) => {

alert(msg);

};

// Synthetic event example

handleClick = (e) => {

alert("I was clicked");

};

render() {

return (

<div style={{ padding: "20px" }}>

<div>{this.state.counter}</div>

<button onClick={this.increment}>Increment</button>

<button onClick={this.decrement}>Decrement</button>

<button onClick={() => this.sayMessage("welcome")}>Say welcome</button>

<button onClick={this.handleClick}>Click on me</button>

<CurrencyConvertor />

</div>

);

}

}

export default App;

Currencyconvertor.js

import React, { Component } from 'react';

class CurrencyConvertor extends Component {

constructor(props) {

super(props);

this.state = {

amount: '',

currency: ''

};

}

handleSubmit = (e) => {

e.preventDefault();

if (this.state.currency.toLowerCase() === "euro") {

let euroAmount = this.state.amount \* 80; // Example conversion rate

alert(`Converting to Euro Amount is ${euroAmount}`);

} else {

alert("Currency not supported");

}

};

render() {

return (

<div style={{ marginTop: "20px" }}>

<h2 style={{ color: "green" }}>Currency Convertor!!!</h2>

<form onSubmit={this.handleSubmit}>

<div>

Amount:

<input

type="number"

value={this.state.amount}

onChange={(e) => this.setState({ amount: e.target.value })}

/>

</div>

<div>

Currency:

<input

type="text"

value={this.state.currency}

onChange={(e) => this.setState({ currency: e.target.value })}

/>

</div>

<button type="submit">Submit</button>

</form>

</div>

);

}

}

export default CurrencyConvertor;

12. ReactJS-HOL

## 1) Conditional rendering

Conditional rendering means returning different React elements based on state or props, similar to using if/else in regular JS

## 2) Element variables

Store elements in variables before returning JSX, which helps choose what to render

## 3) Preventing rendering

Return null from a component to render nothing. This is useful for hiding UI (like a banner) without unmounting parents or changing layout logic.

App.js

import { useState } from "react";

import GuestPage from "./GuestPage";

import UserPage from "./UserPage";

function LoginButton({ onClick }) {

return <button onClick={onClick}>Login</button>;

}

function LogoutButton({ onClick }) {

return <button onClick={onClick}>Logout</button>;

}

// Example of a component that may render nothing

function WarningBanner({ show, message }) {

if (!show) {

return null; // prevents rendering

}

return <div style={{ color: "crimson", marginBottom: 12 }}>{message}</div>;

}

export default function App() {

const [isLoggedIn, setIsLoggedIn] = useState(false);

const handleLogin = () => setIsLoggedIn(true);

const handleLogout = () => setIsLoggedIn(false);

// Element variable example for switching buttons

let actionButton;

if (isLoggedIn) {

actionButton = <LogoutButton onClick={handleLogout} />;

} else {

actionButton = <LoginButton onClick={handleLogin} />;

}

return (

<div style={{ padding: 32, fontFamily: "sans-serif" }}>

<h1>{isLoggedIn ? "Welcome back" : "Please sign up."}</h1>

{/\* Demonstrates preventing rendering \*/}

<WarningBanner

show={!isLoggedIn}

message="Guests can only browse flights. Login to book."

/>

{/\* Conditional page rendering \*/}

{isLoggedIn ? <UserPage /> : <GuestPage />}

<div style={{ marginTop: 16 }}>{actionButton}</div>

</div>

);

}

Guestpage.js

export default function GuestPage() {

const flights = [

{ id: 1, from: "NYC", to: "SFO", time: "08:30", price: 320 },

{ id: 2, from: "LAX", to: "SEA", time: "11:10", price: 140 },

{ id: 3, from: "DAL", to: "MIA", time: "16:45", price: 210 },

];

return (

<section>

<h2>Available Flights (Browse only)</h2>

<ul>

{flights.map(f => (

<li key={f.id}>

{f.from} → {f.to} at {f.time} — ${f.price}

</li>

))}

</ul>

<p style={{ opacity: 0.7 }}>

Booking is disabled for guests. Please login to book tickets.

</p>

</section>

);

}

Index.js

import React from "react";

import { createRoot } from "react-dom/client";

import App from "./App";

createRoot(document.getElementById("root")).render(

<React.StrictMode>

<App />

</React.StrictMode>

);

Userpage.js

import { useState } from "react";

export default function UserPage() {

const flights = [

{ id: 1, from: "NYC", to: "SFO", time: "08:30", price: 320 },

{ id: 2, from: "LAX", to: "SEA", time: "11:10", price: 140 },

{ id: 3, from: "DAL", to: "MIA", time: "16:45", price: 210 },

];

const [selectedId, setSelectedId] = useState(null);

const selected = flights.find(f => f.id === selectedId);

const handleBook = () => {

if (!selected) return;

alert(

`Booked ${selected.from} → ${selected.to} at ${selected.time} for $${selected.price}`

);

};

return (

<section>

<h2>Book Tickets</h2>

<label>

Choose flight:

<select

value={selectedId ?? ""}

onChange={e => setSelectedId(Number(e.target.value))}

style={{ marginLeft: 8 }}

>

<option value="" disabled>

Select a flight

</option>

{flights.map(f => (

<option key={f.id} value={f.id}>

{f.from} → {f.to} at {f.time} — ${f.price}

</option>

))}

</select>

</label>

<div style={{ marginTop: 12 }}>

<button onClick={handleBook} disabled={!selected}>

Book

</button>

</div>

</section>

);

}

13. ReactJS-HOL

## Conditional Rendering: All Major Techniques

* if / else (return different JSX)
  + Use plain JavaScript control flow to return one component tree or another.[legacy.reactjs+1](https://legacy.reactjs.org/docs/conditional-rendering.html)
* Ternary operator ? :
  + Concise inline choice between two JSX branches inside return.[w3schools+2](https://www.w3schools.com/react/react_conditional_rendering.asp)
* Logical AND (&&)
  + Render a fragment only when a condition is truthy; renders nothing when false.[geeksforgeeks+2](https://www.geeksforgeeks.org/reactjs/reactjs-conditional-rendering/)
* switch statement
  + Useful when choosing among several cases (e.g., layout modes, tabs).

## Rendering Multiple Components

Render the three sections side by side or stacked by returning a container with each sub-component, optionally separated by condition checks (e.g., show Books only when books.length > 0).

## Lists with map() and List Components

* map() transforms arrays into arrays of JSX elements or sub-components, ideal for dynamic lists.[geekster+2](https://www.geekster.in/articles/map/)
* Encapsulate list rendering in a reusable List component that accepts data via props and maps to child items

## Keys in React

* Keys let React identify which list items changed, were added, or removed, enabling efficient reconciliation.[dhiwise+2](https://www.dhiwise.com/post/react-lists-and-keys-the-key-to-efficient-rendering)
* Use a stable, unique id from the data as key; avoid array indexes if item order can change or items can be inserted/removed

## Extracting Components With Keys

* When a list item is extracted to its own component (e.g., <BookItem />), pass the key on the array element where you map, not inside BookItem; React reads keys from siblings in that list’s scope.

export default function BlogList({ blogs }) {

return (

<div>

{blogs.map(b => (

<article key={b.id} style={{ marginBottom: 12 }}>

<h2>{b.title}</h2>

<strong>{b.author}</strong>

<p>{b.body}</p>

</article>

))}

</div>

);

}

export default function BookItem({ book }) {

return (

<div>

<h3>{book.bname}</h3>

<h4>{book.price}</h4>

</div>

);

}

import BookItem from './BookItem';

export default function BookList({ books }) {

if (!books || books.length === 0) return <p>No books available.</p>;

return (

<ul style={{ listStyle: 'none', padding: 0, margin: 0 }}>

{books.map(b => (

<li key={b.id}>

<BookItem book={b} />

</li>

))}

</ul>

);

}

export function CourseList({ courses }) {

return (

<div>

{courses.map(c => (

<div key={c.id} style={{ marginBottom: 12 }}>

<h2>{c.name}</h2>

<h4>{c.date}</h4>

</div>

))}

</div>

);

}

export function CourseGrid({ courses }) {

return (

<div style={{ display: 'grid', gridTemplateColumns: '1fr 1fr', gap: 16 }}>

{courses.map(c => (

<div key={c.id} style={{ border: '1px solid #ddd', padding: 12 }}>

<h2>{c.name}</h2>

<h4>{c.date}</h4>

</div>

))}

</div>

);

}

// Higher-order component for conditional access

export default function withAuth(Component) {

return function Wrapped(props) {

return props.isLoggedIn ? (

<Component {...props} />

) : (

<p>Please log in to view this section.</p>

);

};

}

.container {

display: grid;

grid-template-columns: 1fr 1fr 1fr;

gap: 24px;

padding: 24px;

}

.col { padding: 12px; border-left: 6px solid #0a6; }

.col:first-child { border-left: none; }

.h1 { margin-top: 0; }

.toolbar { padding: 12px; display: flex; gap: 8px; flex-wrap: wrap; }

import { useState } from 'react';

import './App.css';

import { books, blogs, courses } from './data';

import BookList from './components/BookList';

import BlogList from './components/BlogList';

import { CourseList, CourseGrid } from './components/CourseList';

import withAuth from './components/withAuth';

// Protect BlogList behind a simple auth check (HOC)

const ProtectedBlogs = withAuth(BlogList);

export default function App() {

// State flags to demonstrate different conditional rendering patterns

const [showBooks, setShowBooks] = useState(true); // logical AND (&&)

const [showBlogs, setShowBlogs] = useState(true); // ternary ? :

const [layout, setLayout] = useState('list'); // switch-case

const [isLoggedIn, setIsLoggedIn] = useState(true); // HOC condition

const [loading, setLoading] = useState(false); // early return

const [error, setError] = useState('');

// Early returns for loading and error states

if (loading) return <p style={{ padding: 24 }}>Loading...</p>;

if (error) {

return (

<div style={{ padding: 24 }}>

<p style={{ color: 'crimson' }}>{error}</p>

<button onClick={() => setError('')}>Clear error</button>

</div>

);

}

return (

<div>

{/\* Small toolbar to toggle conditions \*/}

<div className="toolbar">

<button onClick={() => setShowBooks(s => !s)}>Toggle Books</button>

<button onClick={() => setShowBlogs(s => !s)}>Toggle Blogs</button>

<button onClick={() => setLayout(l => (l === 'list' ? 'grid' : 'list'))}>

Toggle Courses Layout

</button>

<button onClick={() => setIsLoggedIn(a => !a)}>

{isLoggedIn ? 'Log out' : 'Log in'}

</button>

<button onClick={() => setLoading(l => !l)}>Toggle Loading</button>

<button onClick={() => setError(e => (e ? '' : 'Something went wrong'))}>

Toggle Error

</button>

</div>

{/\* IIFE used only to show another conditional style around a section \*/}

{(() => {

return (

<div className="container">

{/\* Books \*/}

<div className="col">

<h1 className="h1">Book Details</h1>

{showBooks && <BookList books={books} />}

</div>

{/\* Blogs \*/}

<div className="col">

<h1 className="h1">Blog Details</h1>

{showBlogs ? (

<ProtectedBlogs blogs={blogs} isLoggedIn={isLoggedIn} />

) : (

<p>No blogs to display.</p>

)}

</div>

{/\* Courses \*/}

<div className="col">

<h1 className="h1">Course Details</h1>

{(() => {

switch (layout) {

case 'grid':

return <CourseGrid courses={courses} />;

case 'list':

default:

return <CourseList courses={courses} />;

}

})()}

</div>

</div>

);

})()}

</div>

);

}

import React from 'react';

import ReactDOM from 'react-dom/client';

import App from '*./App*';

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(<App />);