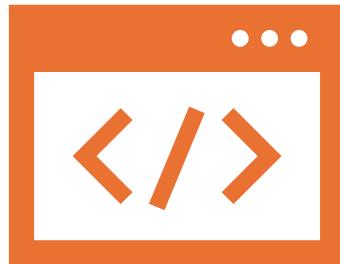


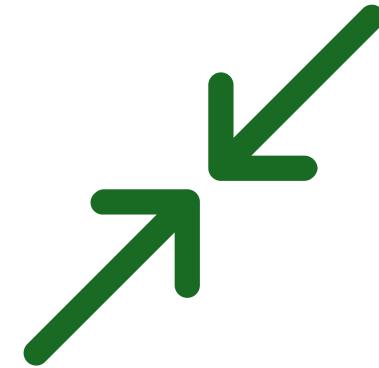
# REACT JS

Prof:- Gautam Singh

# What is React?



React is an open-source JavaScript library developed by Facebook (Meta) for building fast and interactive user interfaces, especially Single Page Applications (SPA).



👉 React focuses on the UI (View layer) of an application.

# Why React exists?

**Before React:**

Web pages reload  
completely after  
every action

Code was hard to  
manage

DOM  
manipulation was  
slow

**React solves this  
using:**

Virtual DOM

Component-  
based  
architecture

# Advantages of React JS Component Based Architecture

UI is divided into small reusable pieces

Easy to maintain & reuse

Virtual DOM

Faster updates than real DOM

Better performance 

Reusable Components

Write once, use many times

# Disadvantages of React JS

JSX, Hooks,  
State, Props  
can confuse  
beginners

Only UI  
Library

Needs extra  
libraries for  
routing, state  
management

SPA may  
face SEO  
issues  
(solved using  
Next.js)

# What are Components in React?

A component  
is a reusable  
piece of UI.

Think of  
components  
like:

Button

Header

Footer

Navbar

# Types of Components

- Functional Components  (Most used)
- Class Components  (Old)
  -  Functional Component (Practical)
  - ```
function Hello() {
```
  - ```
    return <h1>Hello React!</h1>;
```
  - ```
}
```
- ```
export default Hello;
```

# Using Component in App.js

- import Hello from "./Hello";
- function App() {
- return (
- <div>
- <Hello />
- </div>
- );
- }
- export default App;

# + . Import & o Export in React

## Why Import / Export?

- To share components, functions, variables between files.
  - ♦ Export Types
- ① Default Export
  - export default function Header() {
  - return <h1>Header Component</h1>;
  - }
- Import
  - import Header from "./Header";
- ② Named Export
  - export const Footer = () => {
  - return <h1>Footer</h1>;
  - };
- Import
  - import { Footer } from "./Footer";

+

.

o

# State in React

- State is used to store data that can change.
- ✓ React re-renders component when state changes.
  - ◆ useState Hook (Practical)
- import { useState } from "react";
- function Counter() {
- const [count, setCount] = useState(0);
- return (
  - - ## Count: {count}
    - <button onClick={() => setCount(count + 1)}>+</button>
  -
- });
- }
- export default Counter;

# Important Points

- State is local
- Do not mutate state directly
- Use setter function
-  Wrong:
  - `count = count + 1;`
-  Correct:
  - `setCount(count + 1);`

# Props in React

- Props (Properties) are used to pass data from parent to child.

-  Props are read-only.

-  Example (Practical)

- Parent Component

- function App() {

- return <Student name="Rahul" age={21}>;

- }

# Child Component

- function Student(props) {
- return (
- <div>
- <h2>Name: {props.name}</h2>
- <h2>Age: {props.age}</h2>
- </div>
- );
- }

# Destructuring Props

```
function Student({ name, age }) {
```

```
    return <h2>{name} - {age}</h2>;
```

```
}
```

# Axios in React (Fake API Fetch)

- Axios is a promise-based HTTP client used to fetch data from APIs.
  - ◆ Install Axios
  - npm install axios
  - ◆ Fake API (JSONPlaceholder)
  - <https://jsonplaceholder.typicode.com/users>

# Hooks

- `useEffect()` runs side effects like:
  - API calls
  - Data fetching
  - DOM updates
- Syntax:
  - `useEffect(() => {  
 // code  
}, []);`

- + • Fetch API
- o using Axios +  
useEffect  
+ Try  
Catch

```
• import React, { useState, useEffect } from "react";
• import axios from "axios";
• function Users() {
•   const [users, setUsers] = useState([]);
•   const [error, setError] = useState("");
•   useEffect(() => {
•     const fetchUsers = async () => {
•       try {
•         const response = await axios.get(
•           "https://jsonplaceholder.typicode.com/users" );
•         setUsers(response.data);
•       } catch (err) {
•         setError("Failed to fetch users");
•       }
•     }; fetchUsers();
•   }, []);
•   return (    <div>
•     <h1>User List</h1>
•     {error && <p>{error}</p>}
•     {users.map((user) => (
•       <p key={user.id}>{user.name}</p>  ))}  </div>  );
• }
• export default Users;
```

# Flow Explanation

- useEffect runs when component loads
- Axios sends GET request
- Data stored using setUsers
- UI re-renders
- Error handled in catch

# Promises in React

A Promise represents:

Pending

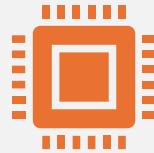
Fulfilled

Rejected

# Promise Example

- const fetchData = new Promise((resolve, reject) => {
  - let success = true;
  - if (success) {
    - resolve("Data fetched");
  - } else {
    - reject("Error occurred");
  - }
  - });
  - fetchData
    - .then((res) => console.log(res))
    - .catch((err) => console.log(err));

# What is Axios & Why We Use It?



Axios is a promise-based HTTP client used to communicate with backend APIs.



👉 It works in browser & Node.js

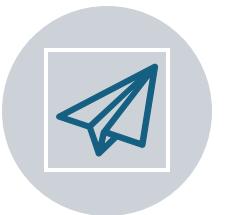


👉 It automatically converts JSON data

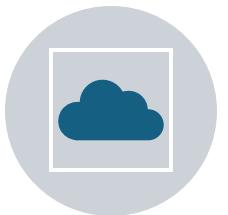
# Uses of Axios



FETCH DATA FROM  
SERVER (GET)



SEND DATA (POST)



UPDATE DATA (PUT /  
PATCH)



DELETE DATA  
(DELETE)



HANDLE HEADERS &  
AUTHENTICATION



ERROR HANDLING  
EASILY



API CALLS INSIDE  
REACT  
COMPONENTS

+

•

# npm install axios

## Axios GET Request (Using then & catch)

- import React, { useState, useEffect } from "react";
- import axios from "axios";
- function UsersThenCatch() {
- const [users, setUsers] = useState([]);
- const [error, setError] = useState("");
- useEffect(() =>
- { axios.get("https://jsonplaceholder.typicode.com/users")
- .then((response) => {
- setUsers(response.data);
- }) .catch((err) => {
- setError("Error fetching users");
- }); }, []);
- return ( <div>
- <h2>Users (then & catch)</h2>
- {error && <p>{error}</p>}
- {users.map((user) => (
- <p key={user.id}>{user.name}</p>
- ))} </div>);}
- export default UsersThenCatch;

# Async & Await

What is async?

async makes a function return a promise

What is await?

await pauses execution until promise resolves

Makes async code look synchronous and cleaner

+

•

- Axios GET  
Using  
async/await  
+ try/catch

- import React, { useState, useEffect } from "react";
- import axios from "axios";
- function UsersAsyncAwait() {
- const [users, setUsers] = useState([]);
- const [error, setError] = useState("");
- useEffect(() => { const fetchUsers = async () => {
- try{
- const response = await axios.get(  
"https://jsonplaceholder.typicode.com/users" );
- setUsers(response.data);
- } catch (err) {
- setError("Something went wrong!");
- } }; fetchUsers();
- }, []);
- return ( <div>
- <h2>Users (async/await)</h2>
- {error && <p>{error}</p>}
- {users.map((user) => (
- <p key={user.id}>{user.name}</p> ))} </div>
- ); }
- export default UsersAsyncAwait;



# JavaScript and ReactJS

- **Author:** Amit Diwan
- **Publisher:** BPB Publications
- • Why this book?
- Combines **JavaScript + React**
- Step-by-step explanation
- Hands-on examples
- Covers:
  - ES6 concepts
  - React components
  - API handling
  - Hooks
  - **Best for:** Practical learning & interview preparation