**Practical 18: ReactJS Installation Steps**

**Code:**

Install Node.js & npm:  
node -v  
npm -v  
  
Create React app (using Create React App):  
npx create-react-app my-app  
cd my-app  
npm start  
  
Open http://localhost:3000 in browser.

**Expected Output:**

Expected Output:  
- node -v: Displays Node.js version (e.g., v18.17.0)  
- npm -v: Displays npm version (e.g., 9.6.7)  
- npx create-react-app: Creates new React application with project structure  
- npm start: Opens development server at http://localhost:3000 showing React welcome page

**Practical 19: Basic "Hello World" React Component**

**Code:**

import React from 'react';  
  
function App() {  
return <h1>Hello World!</h1>;  
}  
  
export default App;

**Expected Output:**

Expected Output:  
Browser displays a webpage with "Hello World!" as a large heading (h1 tag).  
The text appears styled according to default browser h1 styling.

**Practical 20: React Router Setup**

**Code:**

import React from 'react';  
import { BrowserRouter as Router, Routes, Route, Link } from 'react-router-dom';  
  
function Home() { return <h2>Home Page</h2>; }  
function About() { return <h2>About Page</h2>; }  
  
function App() {  
return (  
<Router>  
<nav><Link to="/">Home</Link> | <Link to="/about">About</Link></nav>  
<Routes>  
<Route path="/" element={<Home />} />  
<Route path="/about" element={<About />} />  
</Routes>  
</Router>  
);  
}  
  
export default App;

**Expected Output:**

Expected Output:  
Navigation bar with "Home | About" links at the top.  
- Clicking "Home" displays "Home Page" heading  
- Clicking "About" displays "About Page" heading  
- URL changes dynamically (/, /about) without page refresh

**Practical 21: jQuery Slide Effects**

**Code:**

<!DOCTYPE html>  
<html>  
<head>  
<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>  
</head>  
<body>  
  
<p id="text">Hello! Slide me!</p>  
<button id="up">Slide Up</button>  
<button id="down">Slide Down</button>  
<button id="toggle">Slide Toggle</button>  
  
<script>  
$("#up").click(()=>$("#text").slideUp());  
$("#down").click(()=>$("#text").slideDown());  
$("#toggle").click(()=>$("#text").slideToggle());  
</script>  
  
</body>  
</html>

**Expected Output:**

Expected Output:  
Page displays "Hello! Slide me!" text with three buttons:  
- "Slide Up": Makes text disappear with upward sliding animation  
- "Slide Down": Makes text appear with downward sliding animation  
- "Slide Toggle": Alternates between hiding/showing text with slide effect

**Practical 22: API Integration and Charting in React**

**Code:**

// Install: npm install axios chart.js react-chartjs-2  
import React, { useEffect, useState } from 'react';  
import axios from 'axios';  
import { Line } from 'react-chartjs-2';  
  
function Weather() {  
const [data,setData] = useState({labels:[], datasets:[]});  
  
useEffect(()=>{  
axios.get('https://api.openweathermap.org/data/2.5/onecall?lat=35&lon=139&appid=YOUR\_API\_KEY')  
.then(res=>{  
const labels = res.data.daily.map(d => new Date(d.dt\*1000).toLocaleDateString());  
const temps = res.data.daily.map(d => d.temp.day);  
setData({labels, datasets:[{label:'Temp', data:temps, borderColor:'blue', fill:false}]});  
});  
},[]);  
  
return <Line data={data} />;  
}  
  
export default Weather;

**Expected Output:**

Expected Output:  
Line chart displaying weather temperature data:  
- X-axis: Dates (fetched from API)  
- Y-axis: Temperature values  
- Blue line connecting temperature points over multiple days  
- Chart title showing "Temp" as dataset label  
Note: Requires valid OpenWeatherMap API key to function

**Practical 23: Simple Login Form with State Management**

**Code:**

import React, { useState } from 'react';  
  
function Login() {  
const [user,setUser] = useState('');  
const [pass,setPass] = useState('');  
const [msg,setMsg] = useState('');  
  
const handleLogin = () => setMsg(`Logged in as ${user}`);  
  
return (  
<div>  
<input placeholder="Username" value={user} onChange={e=>setUser(e.target.value)} />  
<input placeholder="Password" type="password" value={pass} onChange={e=>setPass(e.target.value)} />  
<button onClick={handleLogin}>Login</button>  
<p>{msg}</p>  
</div>  
);  
}  
  
export default Login;

**Expected Output:**

Expected Output:  
Login form with:  
- Username input field (placeholder: "Username")  
- Password input field (placeholder: "Password", hidden text)  
- "Login" button  
- When login is clicked: displays "Logged in as [username]" below the form  
- Form inputs update in real-time as user types

**Practical 24: React Event Handling**

**Code:**

import React from 'react';  
  
function Events() {  
const handleClick = () => alert('Button clicked!');  
return <button onClick={handleClick}>Click Me</button>;  
}  
  
export default Events;

**Expected Output:**

Expected Output:  
Simple webpage with a "Click Me" button.  
When button is clicked: Browser alert popup appears with message "Button clicked!"  
User must click "OK" to close the alert and return to the page.

**Practical 25: Searchable List Component**

**Code:**

import React, { useState } from 'react';  
  
function SearchList() {  
const [search,setSearch] = useState('');  
const items = ['Apple','Banana','Orange','Mango'];  
return (  
<div>  
<input placeholder="Search" onChange={e=>setSearch(e.target.value)} />  
<ul>{items.filter(i=>i.toLowerCase().includes(search.toLowerCase())).map(i=><li key={i}>{i}</li>)}</ul>  
</div>  
);  
}  
  
export default SearchList;

**Expected Output:**

Expected Output:  
Search interface with:  
- Search input field (placeholder: "Search")  
- List showing: Apple, Banana, Orange, Mango  
- As user types in search box, list filters in real-time:  
 \* "app" shows only "Apple"  
 \* "an" shows "Banana", "Orange", "Mango"  
 \* Empty search shows all items

**Practical 26: Simple Calculator Component**

**Code:**

// Create a new React app:  
// 1 . npx create-react-app calculator-app  
// cd calculator-app  
// npm start  
  
//2 . Create a Calculator.js component.  
  
import React, { useState } from "react";  
  
const Calculator = () => {  
const [input, setInput] = useState("");  
return (  
<div>  
<input value={input} readOnly />  
{"1234567890+-\*/".split("").map(c => <button onClick={()=>setInput(input+c)}>{c}</button>)}  
<button onClick={()=>setInput(eval(input))}>=</button>  
<button onClick={()=>setInput("")}>C</button>  
</div>  
);  
};  
export default Calculator;

**Expected Output:**

Expected Output:  
Calculator interface with:  
- Read-only display field showing current input  
- Number buttons: 1, 2, 3, 4, 5, 6, 7, 8, 9, 0  
- Operation buttons: +, -, \*, /  
- "=" button: calculates and shows result  
- "C" button: clears the display  
Example: Click 5, +, 3, = shows "8" in display

**Practical 27: Basic Product Page Component**

**Code:**

// Create a new React app:  
// 1 . npx prod-react-app prod-app  
// cd calculator-app  
// npm start  
  
//2 . Create a pord.js component.  
  
import React from "react";  
  
const ProductPage = () => {  
const product = {name:"Phone", price:"$599", images:["https://via.placeholder.com/150"], reviews:["Good","Nice"]};  
return (  
<div>  
<h1>{product.name}</h1>  
<h2>{product.price}</h2>  
{product.images.map((img,i)=><img key={i} src={img} alt="" />)}  
<h3>Reviews:</h3>  
<ul>{product.reviews.map((r,i)=><li key={i}>{r}</li>)}</ul>  
<button>Add to Cart</button>  
</div>  
);  
};  
export default ProductPage;

**Expected Output:**

Expected Output:  
Product page displaying:  
- Large heading: "Phone"  
- Price: "$599" as subheading  
- Product image: 150x150 placeholder image  
- Reviews section with:  
 \* "Reviews:" heading  
 \* Bulleted list: "Good", "Nice"  
- "Add to Cart" button at bottom

**Practical 28: Short Dynamic Website (HTML/CSS/JS)**

**Code:**

<!DOCTYPE html>  
<html lang="en">  
<head>  
<meta charset="UTF-8">  
<meta name="viewport" content="width=device-width, initial-scale=1.0">  
<title>Short Dynamic Website</title>  
<style>  
body { font-family: Arial, sans-serif; margin: 0; padding: 0; background: #f5f5f5; }  
header { background: #bf6934; color: white; padding: 15px; text-align: center; }  
nav { background: #333; padding: 10px; text-align: center; }  
nav a { color: white; margin: 0 10px; text-decoration: none; }  
nav a:hover { text-decoration: underline; }  
main { padding: 20px; text-align: center; }  
button { padding: 10px 20px; margin-top: 10px; cursor: pointer; }  
#dynamicText { color: #4CAF50; font-weight: bold; }  
</style>  
</head>  
<body>  
  
<header>  
<h1>My Short Dynamic Website</h1>  
</header>  
  
<nav>  
<a href="#home" onclick="showSection('home')">Home</a>  
<a href="#about" onclick="showSection('about')">About</a>  
<a href="#contact" onclick="showSection('contact')">Contact</a>  
</nav>  
  
<main>  
<section id="home">  
<h2>Welcome to the Home Page</h2>  
<p>This is a short dynamic website built with HTML, CSS, and JavaScript.</p>  
<button onclick="changeText()">Click Me!</button>  
<p id="dynamicText">Hello World!</p>  
</section>  
  
<section id="about" style="display:none">  
<h2>About Us</h2>  
<p>We create simple, dynamic websites using modern web technologies.</p>  
</section>  
  
<section id="contact" style="display:none">  
<h2>Contact Us</h2>  
<p>Email: contact@example.com</p>  
<p>Phone: +1234567890</p>  
</section>  
</main>  
  
<script>  
function changeText() {  
const text = document.getElementById("dynamicText");  
text.innerText = text.innerText === "Hello World!" ? "You clicked the button!" : "Hello World!";  
}  
  
function showSection(sectionId) {  
const sections = document.querySelectorAll("main section");  
sections.forEach(sec => sec.style.display = "none");  
document.getElementById(sectionId).style.display = "block";  
}  
</script>  
  
</body>  
</html>

**Expected Output:**

Expected Output:  
Multi-section website with:  
- Brown header: "My Short Dynamic Website"  
- Black navigation bar: Home | About | Contact  
- Home section (default): Welcome text, "Click Me!" button, dynamic text  
- About section: Information about the website  
- Contact section: Email and phone details  
- Button toggles text between "Hello World!" and "You clicked the button!"  
- Navigation switches between sections dynamically

**Practical 29: Advanced Dynamic Site (React List Management)**

**Code:**

import React, { useState } from "react";  
  
export default function App() {  
const [items, setItems] = useState(["Item 1", "Item 2"]);  
const [newItem, setNewItem] = useState("");  
  
return (  
<div style={{ padding: "20px", fontFamily: "Arial" }}>  
<h1>Advanced Dynamic Site</h1>  
  
<input  
type="text"  
value={newItem}  
placeholder="New Item"  
onChange={e => setNewItem(e.target.value)}  
/>  
<button onClick={() => {  
if(newItem) { setItems([...items, newItem]); setNewItem(""); }  
}}>Add</button>  
  
<ul>  
{items.map((item, i) => (  
<li key={i}>  
{item} <button onClick={() => setItems(items.filter((\_, idx) => idx !== i))}>Remove</button>  
</li>  
))}  
</ul>  
  
{items.length === 0 && <p>No items!</p>}  
</div>  
);  
}  
  
// Run Instructions  
// Create React app:  
// npx create-react-app advanced-site  
// cd advanced-site  
  
// Replace src/App.js with the above code.  
  
// Run the app:  
// npm start  
  
// Open http://localhost:3000

**Expected Output:**

Expected Output:  
Dynamic list management interface:  
- Heading: "Advanced Dynamic Site"  
- Input field with "New Item" placeholder and "Add" button  
- Initial list: "Item 1" and "Item 2", each with "Remove" button  
- Adding new items: Type in input, click Add - item appears in list, input clears  
- Removing items: Click Remove next to any item - item disappears from list  
- Empty state: If all items removed, displays "No items!" message

**Practical 30: Student Management System (React Router)**

**Code:**

// Steps to Set Up  
// Create React app:  
// npx create-react-app student-system  
// cd student-system  
  
// Install React Router:  
// npm install react-router-dom  
  
import React from "react";  
import { BrowserRouter as Router, Routes, Route, Link } from "react-router-dom";  
  
// Pages  
const Home = () => <h2>Welcome to Student Management System</h2>;  
const Registration = () => (  
<div>  
<h2>Registration</h2>  
<input type="text" placeholder="Name" /><br/><br/>  
<input type="email" placeholder="Email" /><br/><br/>  
<button>Register</button>  
</div>  
);  
const Login = () => (  
<div>  
<h2>Login</h2>  
<input type="email" placeholder="Email" /><br/><br/>  
<input type="password" placeholder="Password" /><br/><br/>  
<button>Login</button>  
</div>  
);  
const Contact = () => (  
<div>  
<h2>Contact Us</h2>  
<p>Email: contact@example.com</p>  
<p>Phone: +1234567890</p>  
</div>  
);  
const About = () => <h2>About Us: We manage student data efficiently.</h2>;  
  
function App() {  
return (  
<Router>  
<div style={{ padding: "10px", fontFamily: "Arial" }}>  
<nav style={{ marginBottom: "20px" }}>  
<Link to="/" style={{ margin: "0 10px" }}>Home</Link>  
<Link to="/register" style={{ margin: "0 10px" }}>Register</Link>  
<Link to="/login" style={{ margin: "0 10px" }}>Login</Link>  
<Link to="/contact" style={{ margin: "0 10px" }}>Contact</Link>  
<Link to="/about" style={{ margin: "0 10px" }}>About</Link>  
</nav>  
  
<Routes>  
<Route path="/" element={<Home />} />  
<Route path="/register" element={<Registration />} />  
<Route path="/login" element={<Login />} />  
<Route path="/contact" element={<Contact />} />  
<Route path="/about" element={<About />} />  
</Routes>  
</div>  
</Router>  
);  
}  
  
export default App;  
  
// Run Instructions  
// Start the app:  
// npm start  
  
// Open browser: http://localhost:3000

**Expected Output:**

Expected Output:  
Student Management System with navigation:  
- Navigation bar: Home | Register | Login | Contact | About  
- Home page: "Welcome to Student Management System"  
- Registration page: Form with Name, Email inputs and Register button  
- Login page: Form with Email, Password inputs and Login button  
- Contact page: Displays contact email and phone number  
- About page: Brief description about student data management  
- URLs change dynamically when navigating between pages