REACT.JS

Presented by Codemonk

WHAT IS WEB DEVELOPMENT?

- Web development is the process of building websites and web applications.
- It involves three core technologies:
  - HTML – Structure of the webpage
  - CSS – Styling and design
  - JavaScript (JS) – Adds interactivity and dynamic features

WHY REACT?

- React is a JavaScript library developed by Facebook.
- Helps build dynamic, reusable UI components.
- Advantages:
  - Fast rendering using Virtual DOM
  - Component-based architecture
  - Easy to scale and maintain
  - Large community & ecosystem

INSTALLING NODE.JS AND NPM

DOWNLOAD NODE.JS

- Step 1:
  🔗 Visit https://nodejs.org/
- Step 2:
  Choose the LTS (Recommended) version
- Step 3:
  Click Download → Run the .msi installer file

INSTALLING NODE.JS AND NPM

INSTALLATION STEPS

Double-click downloaded file
Accept License Agreement
Keep default settings
Ensure ✅ “Install npm” is checked
Click Install → Wait till it finishes
Click Finish

VERIFY INSTALLATION

Open Command Prompt / Terminal and run:

```
node -v

npm -v
```

If both show version numbers — installation is successful!
Example Output:

```
v22.0.0

10.5.0
```

CREATE REACT APP USING VITE

Open terminal and run :

```
npm create vite@latest
```

Then:

- ✓ Project name: my-react-app
- ✓ Select a framework: React
- ✓ Select a variant: JavaScript

NAVIGATE AND INSTALL :

```
cd my-react-app

npm install

npm run dev
```

CREATE REACT APP USING VITE

OUTPUT

Development server starts
URL shown in terminal:

http://localhost:5173/

Open in browser → you’ll see “Vite + React” welcome screen

FOLDER STRUCTURE OVERVIEW

FOLDER/FILE		DESCRIPTION
node_modules/	-	Installed dependencies
index.html	-	Entry file
src/	-	React components
main.jsx	-	Renders root component
App.jsx	-	Main UI component

JSX (JAVASCRIPT XML)

React uses JSX to write HTML inside JavaScript.

Example:

```
const name = "codemonk";
return <h1>Hello, {name}!</h1>;
```

- **You can use JavaScript inside {}.**
- **JSX must have one parent element.**

PROPS (PASSING DATA)

- Props = Properties
- Used to pass data from one component (parent) to another (child)
- They make components reusable and dynamic

Think of props like function parameters!

EXAMPLE WITHOUT PROPS

```
function Welcome() {
  return <h1>Hello, codemonk!</h1>;
}

function App() {
  return <Welcome />;
}
```

The name “codemonk” is fixed — not reusable ❌

EXAMPLE WITH PROPS

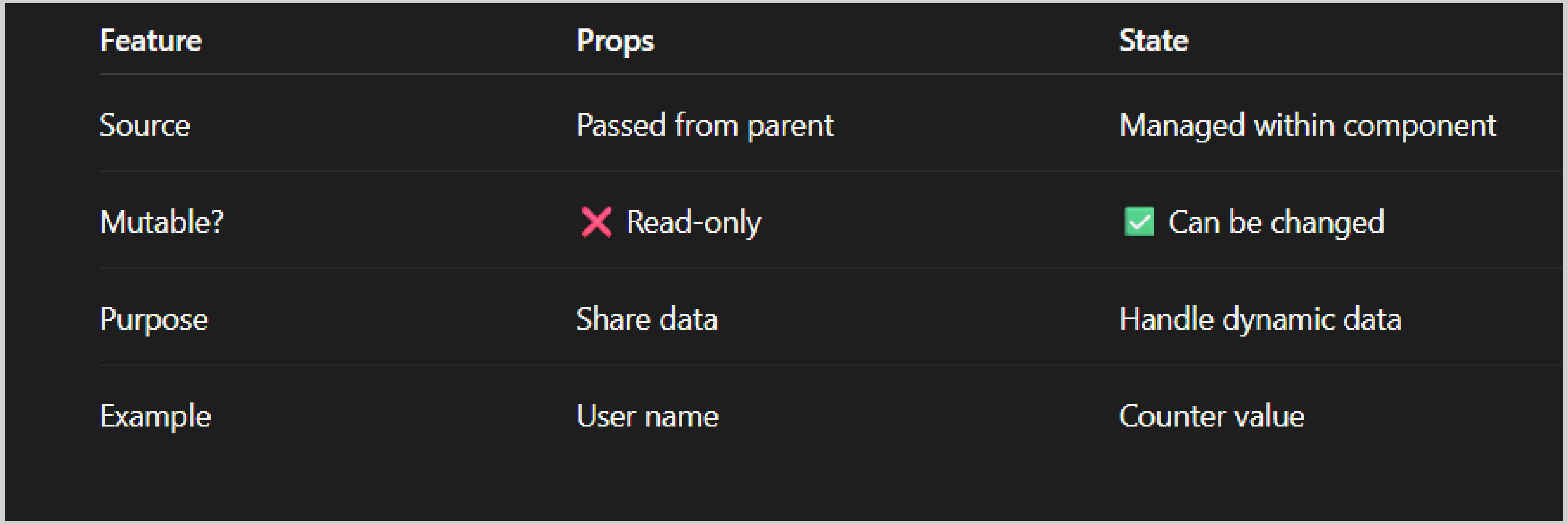
```
function Welcome(props) {
  return <h1>Hello, {props.name}!</h1>;
}

function App() {
  return (
    <>
      <Welcome name="code" />
      <Welcome name="monk" />
      <Welcome name="team" />
    </>
  );
}
```

output:

```
Hello, code!
Hello, monk!
Hello, team!
```

WHAT IS STATE?



Feature	Props	State
Source	Passed from parent	Managed within component
Mutable?	❌ Read-only	✅ Can be changed
Purpose	Share data	Handle dynamic data
Example	User name	Counter value

USING USESTATE() HOOK

To add state in functional components, use React Hook useState()

Syntax:

```
const [variable, setVariable] = useState(initialValue);
```

Example:

```
const [count, setCount] = useState(0);
```

USING USESTATE() HOOK

COUNTER EXAMPLE

```
import React, { useState } from 'react';

function Counter() {
  const [count, setCount] = useState(0);

  return (
    <><h1>Count: {count}</h1>
      <button onClick={() => setCount(count + 1)}>Increase</button>
      <button onClick={() => setCount(count - 1)}>Decrease</button>
      <button onClick={() => setCount(0)}>Reset</button>
    </>
  );
}
```

USING USESTATE() HOOK

COUNTER EXAMPLE

```
import React, { useState } from 'react';

function Counter() {
  const [count, setCount] = useState(0);

  return (
    <><h1>Count: {count}</h1>
      <button onClick={() => setCount(count + 1)}>Increase</button>
      <button onClick={() => setCount(count - 1)}>Decrease</button>
      <button onClick={() => setCount(0)}>Reset</button>
    </>
  );
}
```

USING USEEFFECT() HOOK

USEEFFECT() ALLOWS YOU TO PERFORM SIDE EFFECTS IN REACT COMPONENTS.

“SIDE EFFECTS” = ACTIONS THAT HAPPEN OUTSIDE THE NORMAL UI RENDERING.

EXAMPLES OF SIDE EFFECTS:
✅ FETCHING DATA FROM AN API
✅ UPDATING THE DOM MANUALLY
✅ SETTING UP TIMERS OR EVENT LISTENERS

USING USEEFFECT() HOOK

Syntax of useEffect()

```
useEffect(() => {

  // Code to run (side effect)

}, [dependencies]);
```

Parts:

- Callback function: Code that runs after render
- Dependency array: Controls when the effect runs

USING USEEFFECT() HOOK

Example 1 — Run on Every Render

```
import { useEffect, useState } from "react";

function Counter() {
 const [count, setCount] = useState(0);

 useEffect(() => {
  console.log("Component rendered!");
 });

 return (
  <><h1>{count}</h1>
   <button onClick={() => setCount(count + 1)}>Increase</button>
  </>
 );
}
```

Output:
Logs message every time component re-renders.

USING USECONTEXT() HOOK

- useContext() allows you to share data globally across components
- No need to pass props manually through every level
- Think of it as a global storage box that any component can access.

USING USECONTEXT() HOOK

- Steps to Use Context

Step Description

- Create a context using createContext()
- Wrap components with Context.Provider
- Use useContext() hook to access data

USING USECONTEXT() HOOK

CREATING CONTEXT

```
import React, { createContext, useState, useContext } from 'react';

// Create a context with a default value
const MyContext = createContext();

const MyProvider = ({ children }) => {
  const [state, setState] = useState(0);
  const incri=()=>{setState(state+1);}
const decri=()=>{setState(state-1);}
  return (
    <MyContext.Provider value={{ state, setState , incri, decri}}>
      {children}
    </MyContext.Provider>
  );
};
```

CONSUMING CONTEXT WITH USECONTEXT()

Now, you can consume this context in any child component using the useContext hook.

```
const MyComponent = () => {
  const { state, setState,incri,decri } = useContext(MyContext);

  return (
    <div>
      <p>{state}</p>
      <button onClick={() => incri()}>Update Context</button>
 <button onClick={() => decri()}>Update Context</button>
    </div>
  );
};
```

WRAP YOUR APP WITH THE PROVIDER

Finally, you wrap your main app component with the provider so that any child component can access the context.

```
const App = () => {
 return (
  <MyProvider>
   <MyComponent />
  </MyProvider>
 );
};

export default App;
```

WHAT IS ONCHANGE IN REACT?

onChange is an event handler that runs a function whenever the value of an input changes

like typing in a textbox, selecting a dropdown, or toggling a checkbox.

ONCHANGE IN REACT?

Simple Example – Text Input

```
import React, { useState } from "react";

function App() {
  const [name, setName] = useState("");

  const handleChange = (event) => {
    setName(event.target.value); // update state with the input value
  };

  return (
    <div>
      <h2>Enter your name:</h2>
      <input type="text" value={name} onChange={handleChange} />
      <p>Hello, {name}</p>
    </div>
  );
}

export default App;
```

Output:

```
Enter your name:
[codemonk]
Hello, codemonk
```

WHAT IS MAP()?

- map() is a JavaScript array method
- It loops over an array and returns a new array

```
import React from "react";

function App() {
  const names = ["code", "monk", "team"];

  return (
    <div>
      <h2>Names List</h2>
      <ul>
        {names.map((name, index) => (
          <li key={index}>{name}</li>
        ))}
      </ul>
    </div>
  );
}

export default App;
```

WHAT ARE FETCH AND AXIOS?

Both are used to make HTTP requests to servers — for example, to get data from an API or send data to one.

Using fetch()
fetch() is a built-in JavaScript function (no need to install anything).

Using axios
axios is a third-party library — install it first:

```
npm install axios
```

WHAT ARE FETCH AND AXIOS?

Both are used to make HTTP requests to servers — for example, to get data from an API or send data to one.

Using fetch()

fetch() is a built-in JavaScript function (no need to install anything).

Using axios

axios is a third-party library — install it first:

```
npm install axios
```

```
import React, { useEffect, useState } from "react";
import axios from "axios";

function FetchAxios() {
  const [fetchData, setFetchData] = useState([]);
  const [axiosData, setAxiosData] = useState([]);

  useEffect(() => {
    // Using fetch()
    fetch("https://jsonplaceholder.typicode.com/users")
      .then((res) => res.json())
      .then(setFetchData)
      .catch((err) => console.error("Fetch Error:", err));

    // Using axios
    axios
      .get("https://jsonplaceholder.typicode.com/users")
      .then((res) => setAxiosData(res.data))
      .catch((err) => console.error("Axios Error:", err));
  }, []);
```

```
 return (
  <div style={{ display: "flex", gap: "50px" }}>
   <div>
    <h3>Using Fetch()</h3>
    {fetchData.length ? (
     fetchData.map((u) => <p key={u.id}>{u.name}</p>)
    ) : (
     <p>Loading or Error...</p>
    )}
   </div>

   <div>
    <h3>Using Axios</h3>
    {axiosData.length ? (
     axiosData.map((u) => <p key={u.id}>{u.name}</p>)
    ) : (
     <p>Loading or Error...</p>
    )}
   </div>
  </div>
 );
}

export default FetchAxios;
```

LET'S BEGIN OUR FIRST MINI PROJECT

Build a small React app that fetches and displays product data from the Fake Store API.

API URL:

https://fakestoreapi.com/products

THANK YOU