# React Syllabus one shot

/ Learn Coding

# 1. What is React?

React is a JavaScript library used for building user interfaces, particularly for single-page applications. It allows developers to create reusable UI components and efficiently update and render them as data changes.

# 2. Milestones of React?

- **2011:** Created by Jordan Walke at Facebook.
- **2013:** Open-sourced at JSConf US.
- **2015:** React Native launched.
- **2017:** Fiber (React 16) introduced.
- **2020:** React 17 released, focusing on backward compatibility.
- **2022+:** React grows with Suspense, Concurrent Mode, and more.

# 3. Why Use React?

- Reusable Components: Simplifies development.
- Virtual DOM: Ensures fast updates.
- Community Support: Large ecosystem of libraries.
- Declarative: Focuses on what to do, not how to do it.

# 4. Why is **React** So Famous?

Backed by Facebook.
Excellent for dynamic web apps.
Strong developer tools like React DevTools.
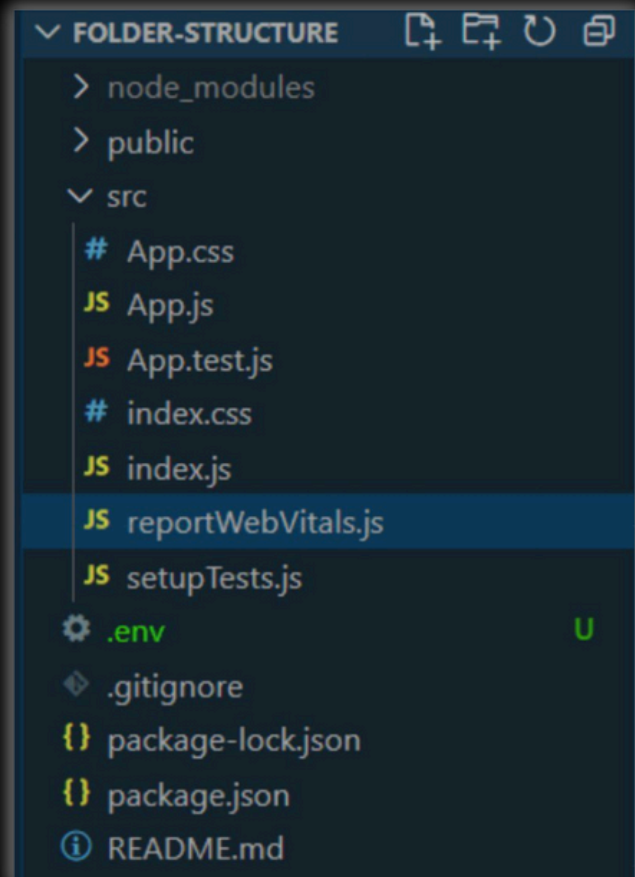Highly adaptable with libraries like Redux and Context API.

# 5. Folder Structure of **React**

src folder: Core app files (e.g., App.js, components, assets).
public folder: Static files (e.g., index.html).
node_modules: Installed npm packages.
package.json: App metadata and dependencies.



```
∨ FOLDER-STRUCTURE        ⎘ ⎗ ↻ ⧉
    > node_modules
    > public
    ∨ src
        #  App.css
        JS App.js
        JS App.test.js
        #  index.css
        JS index.js
        JS reportWebVitals.js
        JS setupTests.js
        ⚙  .env                        U
        ◈  .gitignore
        {} package-lock.json
        {} package.json
        ⓘ  README.md
```

# 6. Setting and Installing Environment for <span style="color:#FFC000">React</span>

• Install Node.js and npm.
• Run npm create vite@latest
• Navigate to the folder:
• Start development: npm start.

# 7. Import & Export

Import: Use import to bring modules into a file.
Example: import Component from './Component';
Export: Share code across files.
Example: export default Component;

# 8. Real DOM vs. Virtual DOM

• Real DOM: Updates the whole DOM tree.
• Virtual DOM: Updates only the changed elements, improving performance.

# 9. JSX

JSX (JavaScript XML) allows writing HTML-like syntax in JavaScript.
Example:
<h1>Hello, World!</h1>.
JSX is compiled to React.createElement calls.

# 10. Components

Body:
Functional Components: Written as functions (e.g., function
Component()).
Class Components: Use ES6 classes and support lifecycle methods.

# 11. CSS / Tailwind CSS / CSS Modules

- **CSS:** Standard styling language (e.g., style.css).
- **Tailwind CSS:** Utility-first CSS framework.
- **CSS Modules:** Scoped CSS for components (Component.module.css)

```
                                                    tailwind css
1   css
2   /* style.css */ button { color: ■blue; }
3   Tailwind CSS:
4   jsx
5   <button className="text-blue-500">Click Me</button>
6   CSS Modules:
7   jsx
8   import styles from './Button.module.css'; <button className=
    {styles.blueButton}>Click
9   Me</button>;
```

# 12. Conditional Rendering

Render elements based on conditions.
Example:

```
1  {isLoggedIn ? <h1>Welcome Back!</h1> : <h1>Please Sign
   In</h1>}
2
```

# 13. Props

Props are arguments passed to components for
dynamic data.
Example:

```
1 ∨ function Greet(props) { return <h1>Hello, {props.name}!</h1>;
    } <Greet
2   name="John" />;
```

# 14. Calling Function

Call functions on events like clicks.
Example:

```
1  function handleClick() { alert("Button Clicked!"); } <button
2  onClick={handleClick}>Click Me</button>;
```

# 15. React Router DOM

Routing library for navigation in React apps.
Example:

```
1  import { BrowserRouter, Route, Link } from 'react-router-dom',
2  <BrowserRouter>
3  <Link to="/about">About</Link>
4  <Route path="/about" component={About} />
5  </BrowserRouter>;
6  Export
```

# 16. State - React Variable / Hooks

State holds dynamic data and can be updated using
hooks like
useState.
Example:

```
1  const [count, setCount] = useState(0);
2  <button onClick={() => setCount(count + 1)}>Count:
3  {count}</button>;
```

# 17. Events Handling / Forms

Capture user inputs and manage form submissions.
Example:

```
1  function handleSubmit(e)
2  {
3    e.preventDefault();
4    alert("Form Submitted!");
5  }
6  <form onSubmit={handleSubmit}>
7    <input type="text" placeholder="Enter name" />
8    <button type="submit">Submit</button> </form>;
```

# 18. List & Keys

Render lists using map() with unique keys.
Example:

```
1  const items = ["Apple", "Banana", "Cherry"];
2  <ul>{items.map((item, index) => <li key={index}>{item}</li>)}
   </ul>;
```

# 19. Rendering JSON Data

Display JSON data by iterating over it.
Example

```
1  const items = ["Apple", "Banana", "Cherry"];
2  <ul>{items.map((item, index) => <li key={index}>{item}</li>)}
   </ul>;
```

# 20. useEffect

Run side effects in functional components.
Example:

```
1  useEffect(() => {
2    console.log("Component mounted!");
3  }, []);
```

# 21. Context API

Share state globally without pop drilling.
Example:

```
1  const UserContext = React.createContext();
2  <UserContext.Provider value="John">
3   <App />
4  </UserContext.Provider>;
```

# 22. Fetching API

Retrieve data from an API using fetch or libraries like Axios
Example:

```
tailwind css
1   useEffect(() => {
2     fetch('https://api.example.com/data')
3       .then(res => res.json())
4       .then(data => console.log(data));
5   }, []);
6
```

# 23. Deployment

Host your React app on platforms like **Vercel**, **Netlify**, or **GitHub Pages**.
Deploy to Vercel: Connect GitHub repository and deploy.
See More In React One Shot Tutorial