React.js Manual

Beginner to Intermediate Guide

1. Introduction to React

React is a JavaScript library for building user interfaces, primarily for single-page applications. It allows developers to create reusable UI components and efficiently update the UI when data changes.

Why Use React?

1. Component-based architecture
2. Fast rendering using Virtual DOM
3. Reusable and maintainable code
4. Strong community and ecosystem

2. Setting Up React

The easiest way to start a React project is by using Vite or Create React App.

Example using Vite:

```
npm create vite@latest my-app cd my-app npm install npm run dev
```

3. JSX Basics

JSX is a syntax extension that allows you to write HTML-like code inside JavaScript. It makes UI code more readable and expressive.

```
const element = Hello, React!;
```

4. Components

Components are the building blocks of React applications. They can be functional or class-based. Modern React uses functional components with hooks.

```
function Welcome() { return Welcome User; }
```

5. Props

Props (properties) are used to pass data from parent components to child components.

```
function Greeting({ name }) { return Hello {name}; }
```

6. State and Hooks

State allows components to manage and track data changes. Hooks like useState and useEffect enable state and lifecycle features in functional components.

```
import { useState } from 'react'; function Counter() { const [count, setCount] = useState(0); return ( setCount(count + 1)}> Count: {count} ); }
```

7. useEffect

The useEffect hook lets you perform side effects such as data fetching, subscriptions, or manually changing the DOM.

```
useEffect(() => { console.log('Component mounted'); }, []);
```

8. Handling Events

React events are written in camelCase and passed as functions.

```
Click Me
```

9. Conditional Rendering

You can render components conditionally using if statements or ternary operators.

```
{isLoggedIn ? : }
```

10. Lists and Keys

Lists are rendered using the map() function. Each item should have a unique key.

```
items.map(item => {item.name})
```

11. Forms

Controlled components handle form data using state.

```
setName(e.target.value)} />
```

12. Recommended Project Structure

A common React project structure includes components, pages, hooks, services, and assets folders.

13. Best Practices

1. Keep components small and reusable
2. Use meaningful component and variable names
3. Lift state up when necessary
4. Avoid unnecessary re-renders

14. Conclusion

React is a powerful and flexible library for building modern web applications. With a solid understanding of components, props, state, and hooks, you can build scalable UIs efficiently.