

Table of Contents



- [1. Installation and Setup](#)
 - [1.1 Install Node.js](#)
 - [1.2 Create a New React App](#)
- [2. Project Structure](#)
- [3. Basic Component Structure](#)
- [4. JSX \(JavaScript XML\)](#)
- [5. Rendering Components](#)
- [6. Functional vs. Class Components](#)
 - [Functional Component \(Preferred\):](#)
 - [Class Component:](#)
- [7. Props \(Properties\)](#)
- [8. State \(Component State Management\)](#)
- [9. Event Handling](#)
- [10. Conditional Rendering](#)
- [11. Lists and Keys](#)
- [12. Forms in React](#)
- [13. CSS Styling](#)
 - [Inline Styling:](#)
 - [External CSS:](#)
- [14. React Router \(Navigation\)](#)
- [15. Lifecycle Methods \(Class Components\)](#)
- [16. Hooks \(Functional Component Lifecycle\)](#)
- [17. Fetch Data \(API Call Example\)](#)
- [18. Useful CLI Commands](#)

1. Installation and Setup

1.1 Install Node.js

Download from <https://nodejs.org/>

1.2 Create a New React App

```
npx create-react-app my-app
```

```
cd my-app
npm start
```

- npx - Runs the latest version without installing globally.
-

2. Project Structure

```
my-app/
├── public/           # Static files (HTML, favicon)
├── src/              # React components and logic
│   ├── App.js        # Main component
│   ├── index.js       # Renders App component
│   ├── App.css        # Styling
│   └── index.css
├── package.json      # Project dependencies & scripts
└── README.md
```

3. Basic Component Structure

App.js (Functional Component Example):

```
import React from 'react';

function App() {
  return (
    <div>
      <h1>Hello, React!</h1>
    </div>
  );
}

export default App;
```

4. JSX (JavaScript XML)

- JSX allows HTML to be written within JavaScript.
- Example:

```
const title = <h1>Welcome to React</h1>;
const element = <div>{title}</div>;
```

- Notes:
 - JSX must return a single parent element.
 - Use fragments if needed:

```
<>
  <h1>Hello</h1>
  <p>World</p>
</>
```

5. Rendering Components

index.js:

```
import React from 'react';
import ReactDOM from 'react-dom';
import App from './App';

ReactDOM.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
  document.getElementById('root')
);
```

- Renders the App component inside the root div from index.html.

6. Functional vs. Class Components

Functional Component (Preferred):

```
function Greeting() {  
  return <h1>Hello!</h1>;  
}
```

Class Component:

```
import React, { Component } from 'react';  
  
class Greeting extends Component {  
  render() {  
    return <h1>Hello!</h1>;  
  }  
}
```

7. Props (Properties)

- Pass Data to Components:

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

8. State (Component State Management)

Functional Component with useState (React Hooks):

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

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

  return (
    <div>
      <p>Count: {count}</p>
      <button onClick={() => setCount(count +
1)}>Increment</button>
    </div>
  );
}
```

9. Event Handling

```
function ButtonClick() {
  function handleClick() {
    alert('Button Clicked!');
  }

  return <button onClick={handleClick}>Click Me</button>;
}
```

10. Conditional Rendering

```
function Message(props) {
  return (
    <div>
      {props.isLoggedIn ? <h1>Welcome Back!</h1> : <h1>Please
Sign In</h1>}
    </div>
  );
}
```

11. Lists and Keys

```
const names = ['Alice', 'Bob', 'Charlie'];
const listItems = names.map((name) => <li key={name}>{name}</li>);

function NameList() {
  return <ul>{listItems}</ul>;
}
```

12. Forms in React

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

function Form() {
  const [value, setValue] = useState('');

  const handleSubmit = (e) => {
    e.preventDefault();
    alert(`Submitted: ${value}`);
  };

  return (
    <form onSubmit={handleSubmit}>
      <input
        type="text"
        value={value}
        onChange={(e) => setValue(e.target.value)}
      />
      <button type="submit">Submit</button>
    </form>
  );
}
```

13. CSS Styling

Inline Styling:

```
const headingStyle = {  
  color: 'blue',  
  fontSize: '24px'  
};
```

```
<h1 style={headingStyle}>Styled Text</h1>
```

External CSS:

```
h1 {  
  color: red;  
}
```

App.js:

```
import './App.css';
```

14. React Router (Navigation)

```
npm install react-router-dom
```

App.js (Routing Example):

```
import React from 'react';  
import { BrowserRouter as Router, Route, Switch, Link } from 'react-router-dom';  
  
function Home() {  
  return <h1>Home Page</h1>;  
}
```

```

function About() {
  return <h1>About Page</h1>;
}

function App() {
  return (
    <Router>
      <nav>
        <Link to="/">Home</Link>
        <Link to="/about">About</Link>
      </nav>
      <Switch>
        <Route exact path="/" component={Home} />
        <Route path="/about" component={About} />
      </Switch>
    </Router>
  );
}

```

15. Lifecycle Methods (Class Components)

```

class Lifecycle extends React.Component {
  componentDidMount() {
    console.log('Component Mounted');
  }

  componentDidUpdate() {
    console.log('Component Updated');
  }

  componentWillUnmount() {
    console.log('Component Unmounted');
  }

  render() {
    return <h1>Lifecycle Demo</h1>;
  }
}

```

```
    }  
  }  
}
```

16. Hooks (Functional Component Lifecycle)

```
import React, { useEffect } from 'react';  
  
function Example() {  
  useEffect(() => {  
    console.log('Component Mounted');  
  
    return () => {  
      console.log('Component Unmounted');  
    };  
  }, []);  
  
  return <h1>Using useEffect</h1>;  
}
```

17. Fetch Data (API Call Example)

```
import React, { useState, useEffect } from 'react';  
  
function DataFetcher() {  
  const [data, setData] = useState([]);  
  
  useEffect(() => {  
    fetch('https://jsonplaceholder.typicode.com/posts')  
      .then((response) => response.json())  
      .then((data) => setData(data));  
  }, []);  
  
  return (  
    <ul>
```

```
        {data.map((item) => (  
            <li key={item.id}>{item.title}</li>  
        ))}  
    </ul>  
);  
}
```

18. Useful CLI Commands

npm start	# Start development server
npm run build	# Create production build
npm test	# Run tests
npm install <package>	# Install package
npm uninstall <package>	# Uninstall package