React - Components

Topics : React JS
Written on January 02, 2024

In React, components are the building blocks of user interfaces. A React application is typically composed of multiple components that encapsulate specific parts of the UI. Components are reusable, modular, and can be nested within one another. There are two main types of React components: functional components and class components.

Functional Components:

Functional components are JavaScript functions that take props (short for properties) as arguments and return React elements. They are also known as stateless or presentational components.

```
// Example of a functional component
const WelcomeMessage = (props) => {
return <h1>Hello, {props.name}!</h1>;
};
```

Functional components are primarily used when the component doesn't have its own state or lifecycle methods. With the introduction of React Hooks, functional components can now have state and side effects using hooks like `useState` and `useEffect`.

```
// Example of a class component
import React, { Component } from 'react';

class WelcomeMessage extends Component {
render() {
return <h1>Hello, {this.props.name}!</h1>;
}
}
```

Class Components:

Class components are ES6 classes that extend `React.Component`. They have their own state, lifecycle methods, and can handle complex logic. However, with the introduction of React Hooks, functional components can now handle state and lifecycle methods, reducing the need for class components in many cases.

```
// Example of a class component
import React, { Component } from 'react';

class WelcomeMessage extends Component {
render() {
```

```
return <h1>Hello, {this.props.name}!</h1>;
}
}
```

Class components have a `render` method, and they can manage their state using `setState`. They are used when local state or lifecycle methods are required.

JSX and Rendering Components:

In both functional and class components, JSX is used to define the structure of the component. JSX is a syntax extension for JavaScript that looks similar to XML or HTML. It allows you to write UI elements in a concise and readable way.

```
const App = () => {
return (
<div>
<WelcomeMessage name="John" />
<Counter />
</div>
);
};
```

In this example, the `App` component renders the `WelcomeMessage` and `Counter` components.

Component Lifecycle:

Class components have lifecycle methods, such as `componentDidMount`, `componentDidUpdate`, and `componentWillUnmount`. These methods allow you to perform actions at different stages of a component's lifecycle.

```
class ExampleComponent extends React.Component {
componentDidMount() {
console.log('Component mounted!');
}

componentDidUpdate(prevProps, prevState) {
console.log('Component updated!');
}

componentWillUnmount() {
console.log('Component will unmount!');
}

render() {
return <p>Example Component</p>;
}
}
```

Props:

Props allow you to pass data from a parent component to a child component. They are read-only and should not be modified within the component.

```
const ParentComponent = () => {
return <ChildComponent message="Hello from parent!" />;
};

const ChildComponent = (props) => {
return <p>{props.message}</p>;
};
```

State:

State is used for handling the internal state of a component. State can be modified using the `setState` method, triggering a re-render of the component.

```
class Counter extends React.Component {
constructor(props) {
super(props);
this.state = {
count: 0,
};
}

incrementCount = () => {
this.setState({ count: this.state.count + 1 });
};

render() {
return (
<div>
<p>Count: {this.state.count}</p>
<button onClick={this.incrementCount}>Increment</button>
</div>
);
}
}
```

Key React Concepts:

1. **Props:** Data passed to a component.
2. **State:** Internal data managed by a component.
3. **Lifecycle Methods:** Methods that are called at different stages of a component's lifecycle.
4. **Hooks:** Functions that add state and lifecycle features to functional components.
5. **Context:** A way to share values like themes, authentication status, etc., between components without explicitly passing them through props.