React - Events

Topics : React JS
Written on January 02, 2024

In React, handling events is similar to handling events in regular HTML, with some differences due to the fact that React uses a synthetic event system for cross-browser compatibility. Here's how you can handle events in React:

1. Basic Event Handling:

```
import React from 'react';

class MyComponent extends React.Component {
handleClick = () => {
console.log('Button clicked!');
};

render() {
return (
<div>
<button onClick={this.handleClick}>Click me</button>
</div>
);
}
}

export default MyComponent;
```

In this example, the `handleClick` method is called when the button is clicked.

2. Passing Parameters to Event Handlers:

If you need to pass additional parameters to an event handler, you can use an arrow function:

```
import React from 'react';

class MyComponent extends React.Component {
handleClick = (param) => {
console.log(`Button clicked with param: ${param}`);
};

render() {
return (
<div>
```

```
<button onClick={() => this.handleClick('example')}>Click me</button>
</div>
);
}
}

export default MyComponent;
```

3. Using `bind` for Event Handlers:

Another way to handle parameterized event handlers is by using `bind`:

```
import React from 'react';

class MyComponent extends React.Component {
handleClick(param) {
console.log(`Button clicked with param: ${param}`);
}

render() {
return (
<div>
<button onClick={this.handleClick.bind(this, 'example')}>Click me</button>
</div>
);
}
}

export default MyComponent;
```

4. Updating State with Events:

You often use events to trigger updates to the component's state, which in turn triggers a re-render:

```
import React from 'react';

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

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

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

```
</div>
);
}
}

export default Counter;
```

5. Synthetic Events:

React uses a synthetic event system to ensure consistent behavior across different browsers. The `onClick`, `onChange`, and other event handlers receive synthetic events that wrap the native browser events.

```
import React from 'react';

class MyComponent extends React.Component {
handleChange = (event) => {
console.log(`Input value: ${event.target.value}`);
};

render() {
return (
<div>
<input type="text" onChange={this.handleChange} />
</div>
);
}
}

export default MyComponent;
```

6. Preventing Default Behavior:

To prevent the default behavior of an event, such as form submission, you can use `event.preventDefault()`:

```
import React from 'react';

class MyComponent extends React.Component {
handleSubmit = (event) => {
event.preventDefault();
console.log('Form submitted!');
};

render() {
return (
<div>
<form onSubmit={this.handleSubmit}>
<button type="submit">Submit</button>
</form>
</div>
);
```

```
}
}

export default MyComponent;
```