Handling Events in React.js

In this article, we'll explore how to handle events with React.js, one of the most popular JavaScript libraries for building web applications. We'll cover everything from binding events to passing arguments and preventing default behaviours. Let's dive in!

 **by Curious Coder (CodeWithCurious.com)**

Introduction to Handling Events

What are events?

In web development, an event refers to anything that happens on a web page, such as when a button is clicked or a form is submitted. In React, you can handle these events with event handlers.

Why is event handling important?

Event handling is crucial in creating dynamic web applications. By responding to events with custom logic, you can create user interactions that make your application feel more responsive and intuitive.

Binding Events in React

> When you want to bind an event to a component in React, you need to define a function that will be called when the event is triggered. The function is then bound to the component using the "this" keyword.
>
> - React Documentation

Event Handlers in React

Event	Handler
onClick	handleClick
onChange	handleChange
onSubmit	handleSubmit

Here are some examples of common events and their respective handlers in React. Keep in mind that event handlers should be defined as class methods, not within the render method.

Understanding Synthetic Events in React

Synthetic events are a cross-browser wrapper around the native browser events. They have the same interface as native events, including stopPropagation() and preventDefault(), but work identically across all browsers. In addition, synthetic events can be reused to enhance performance.

Passing Arguments to Event Handlers in React

Passing arguments to an event handler is essential when you want to perform actions on the data associated with the event. The most common way to do this is by using arrow functions to create a closure around the event handler function.

1 Example:

```
handleClick = (id) => () => console.log(id);
<button onClick={this.handleClick(itemId)}>Click me</button>
```

Preventing Default Behaviour in React

By default, every HTML element has a predefined behaviour. For example, if you click a link, the browser will navigate to the URL specified in the href attribute. In some cases, you may want to prevent this default behaviour to perform custom actions instead.

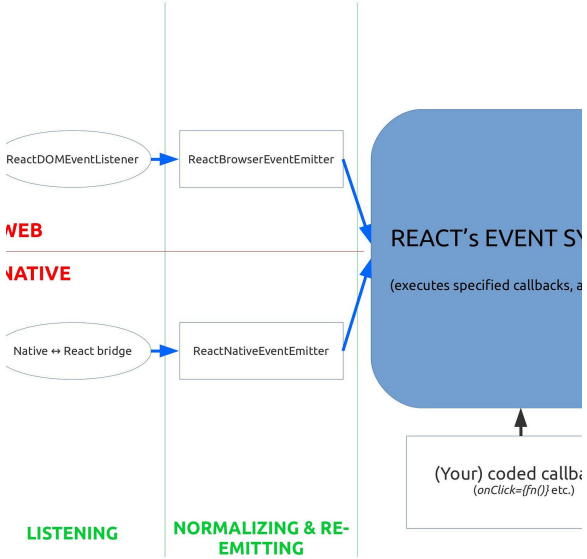
Method 1: Using event.preventDefault()

```
handleClick = (event) => { event.preventDefault(); console.log('Link clicked!'); }
```

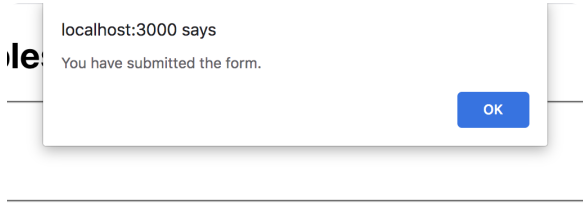
Method 2: Using a button element

```
<button type="button" onClick={this.handleClick}>Click me</button>
```

Example of Handling Events in React



Example of a button click event



Example of a form submission event

Here are a couple of examples of how you can handle events in React. These are simple examples, but they demonstrate the core concepts of event handling in React.