React Lecture - 8

Events

Handling Events

Introduction

React lets you add event handlers to your JSX. Event handlers are your own functions that will be triggered in response to interactions like clicking, hovering, focusing form inputs, and so on.

Adding Event Handler

To add an event handler, you will first define a function and then pass it as a prop to the appropriate JSX tag. For example, here is a button that doesn't do anything yet:

```
export default function Button() {
  return (
    <button>
      I don't do anything
    </button>
  );
}

```

You can make it show a message when a user clicks by following these three steps:

1. Declare a function called handleClick inside your Button component.

2. Implement the logic inside that function (use alert to show the message).

3. Add onClick={handleClick} to the <button> JSX.

```
export default function Button() {
  function handleClick() {
    alert('You clicked me!');
  }

  return (
    <button onClick={handleClick}>
      Click me
    </button>
  );S
}

```

Adding Event Handlers

You defined the handleClick function and then passed it as a prop to <button>. handleClick is an event handler. Event handler functions:

Are usually defined inside your components.

Have names that start with handle, followed by the name of the event.

By convention, it is common to name event handlers as handle followed by the event name. You'll often see **onClick={handleClick}**, **onMouseEnter={handleMouseEnter}**, and so on.

Adding Event Handlers

Alternatively, you can define an event handler inline in the JSX:

```
<button onClick={function handleClick() {
  alert('You clicked me!');
}}>
```

Or, more concisely, using an arrow function:

```
<button onClick={() => {
  alert('You clicked me!');
}}>
```

Adding Event Handlers

Functions passed to event handlers must be passed, not called. For example:

passing a function (correct)	calling a function (incorrect)
`<button onClick={handleClick}>`	`<button onClick={handleClick()}>`

The difference is subtle. In the first example, the handleClick function is passed as an onClick event handler. This tells React to remember it and only call your function when the user clicks the button.

Adding Event Handlers

When you write code inline, the same pitfall presents itself in a different way:

passing a function (correct)	calling a function (incorrect)
`<button onClick={() => alert('...')}>`	`<button onClick={alert('...')}>`

Reading Props in Event Handlers

Because event handlers are declared inside of a component, they have access to the component's props. Here is a button that, when clicked, shows an alert with its message prop:

```
function AlertButton({ message, children }) {
  return (
    <button onClick={() => alert(message)}>
      {children}
    </button>
  );
}

export default function Toolbar() {
  return (
    <div>
      <AlertButton message="Playing!">
        Play Movie
      </AlertButton>
      <AlertButton message="Uploading!">
        Upload Image
      </AlertButton>
    </div>
  );
}

```