




# Welcome to the CoGrammar Lecture: Event Handling

**The session will start shortly...**

Questions? Drop them in the chat. We'll have dedicated moderators answering questions.



# Full Stack Web Development Session Housekeeping

---

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly.  
**(Fundamental British Values: Mutual Respect and Tolerance)**
- No question is daft or silly - **ask them!**
- There are **Q&A sessions** midway and at the end of the session, should you wish to ask any follow-up questions. Moderators are going to be answering questions as the session progresses as well.
- If you have any questions outside of this lecture, or that are not answered during this lecture, please do submit these for upcoming Academic Sessions. You can submit these questions here: [Questions](#)

## Full Stack Web Development Session Housekeeping cont.

---

- For all **non-academic questions**, please submit a query:  
[www.hyperiondev.com/support](http://www.hyperiondev.com/support)
- Report a **safeguarding** incident:  
[www.hyperiondev.com/safeguardreporting](http://www.hyperiondev.com/safeguardreporting)
- We would love your **feedback** on lectures: [Feedback on Lectures](#)

# Skills Bootcamp

## 8-Week Progression Overview

### Fulfil 4 Criteria to Graduation

#### ✓ Criterion 1: Initial Requirements

Timeframe: First 2 Weeks

Guided Learning Hours (GLH):

Minimum of 15 hours

Task Completion: First four tasks

**Due Date: 24 March 2024**

#### ✓ Criterion 2: Mid-Course Progress

**60** Guided Learning Hours

Data Science - **13 tasks**

Software Engineering - **13 tasks**

Web Development - **13 tasks**

**Due Date: 28 April 2024**

# Skills Bootcamp Progression Overview

## ✓ Criterion 3: Course Progress

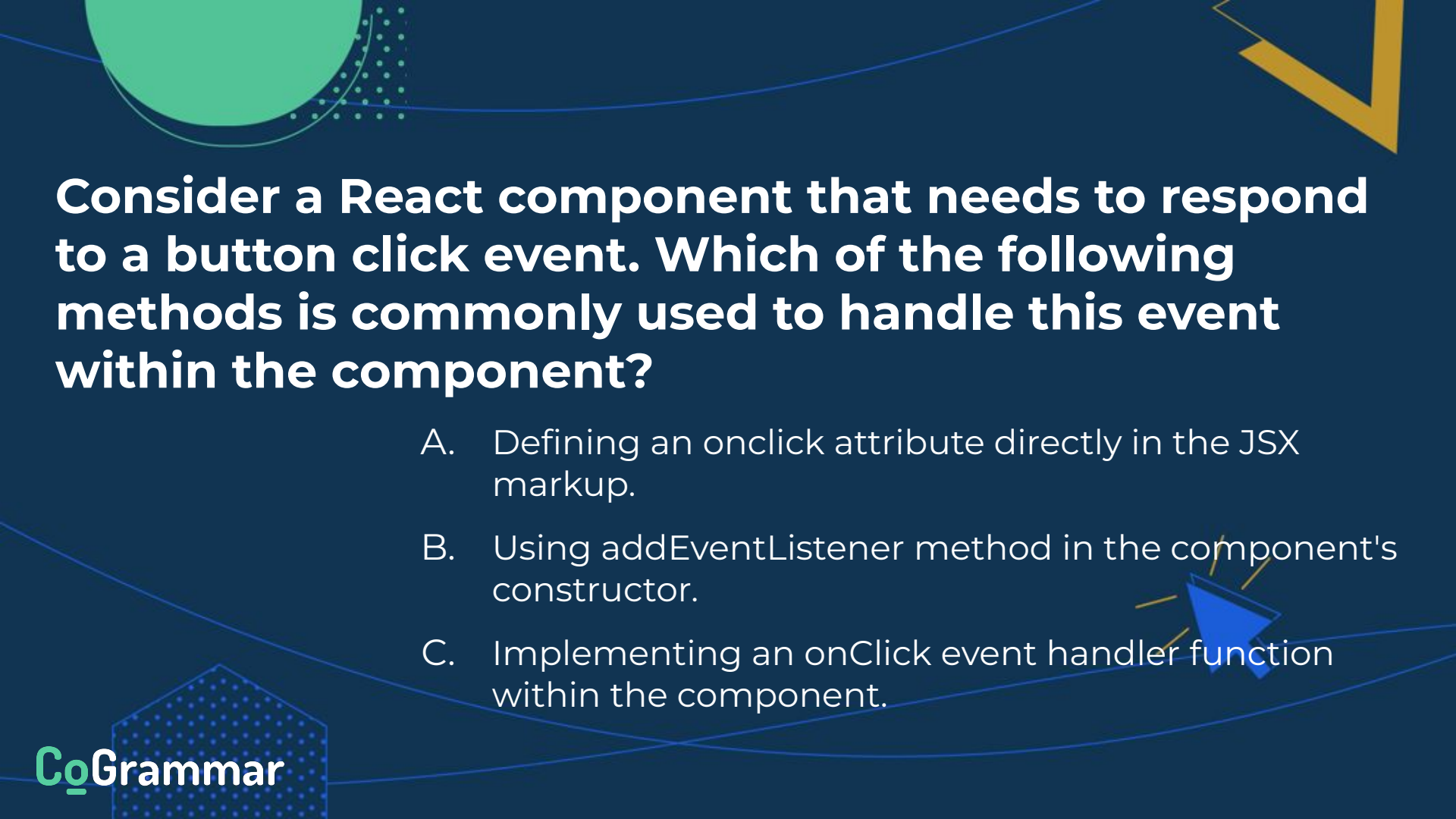
Completion: All mandatory tasks,  
including Build Your Brand and  
resubmissions by study period end  
Interview Invitation: Within 4 weeks  
post-course  
Guided Learning Hours: Minimum of  
112 hours by support end date  
(10.5 hours average, each week)

## ✓ Criterion 4: Demonstrating Employability

Final Job or Apprenticeship  
Outcome: Document within 12  
weeks post-graduation  
Relevance: Progression to  
employment or related  
opportunity

# Learning Objectives

- ❖ Master the various ways to handle events in React components, including inline event handlers, method binding, and arrow function syntax.
- ❖ Discover common event types in React such as `onClick`, `onChange`, `onSubmit`, etc., and how to utilise them effectively.
- ❖ Gain practical experience through hands-on examples and exercises to reinforce learning and deepen understanding.




**Consider a React component that needs to respond to a button click event. Which of the following methods is commonly used to handle this event within the component?**

- A. Defining an onclick attribute directly in the JSX markup.
- B. Using addEventListener method in the component's constructor.
- C. Implementing an onClick event handler function within the component.



# 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.
  - ❖ In React.js, event handling is crucial for building interactive and dynamic user interfaces.
  - ❖ Events like `onClick`, `onChange`, `onSubmit`, etc., enable users to interact with components, triggering updates and actions.
  - ❖ Understanding event handling enhances the responsiveness and interactivity of React applications.
- 



# Adding event handlers

- ❖ To add an event handler, you will first define a function and then pass it as a prop to the appropriate JSX tag.

```
export default function Button() {  
  function handleClick() {  
    alert('You clicked me!');  
  }  
  
  return (  
    <button onClick={handleClick}>  
      Click me  
    </button>  
  );  
}
```

# 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!');  
}}>
```

# Passing event handlers as props

- ❖ Often you'll want the parent component to specify a child's event handler.
- ❖ To do this, pass a prop the component receives from its parent as the event handler.

```
import React from 'react';  
function SuperButton({ onClick }) {  
  return <button onClick={onClick}>I am a Super Button!</button>;  
}
```

```
export default function App() {  
  return (  
    <div>  
      <SuperButton  
        onClick={() => {  
          alert('Haaaaayyyyyy!!!');  
        }}  
      />  
    </div>  
  );  
}
```

# Event Object

- ❖ When an event occurs, React automatically passes an event object to the event handler function. This object contains a wealth of information about the event, including the following common properties:
  - target: The DOM element that triggered the event.
  - type: The type of the event (e.g., "click", "change").
  - preventDefault: A method to prevent the default behavior of the event.

# Event Object

```
import React from 'react';

const MyComponent = () => {
  const handleClick = (event) => {
    console.log('Button clicked!');
    console.log('Event:', event);
    console.log('Target Element:', event.target);
    console.log('Event Type:', event.type);
    console.log('Current Target:', event.currentTarget);
  };

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

# Remember

- ❖ Functions passed to event handlers must be passed, not called.
- ❖ The `()` at the end of `handleClick()` fires the function immediately during rendering, without any clicks. This is because JavaScript inside the JSX `{}` and `}` executes right away.

**passing a function (correct)**

```
<button onClick={handleClick}>
```

**calling a function (incorrect)**

```
<button onClick={handleClick()}>
```



# Let's Breathe!

Let's take a small break  
before moving on to  
the next topic.



# onClick Event

- ❖ The onClick event is triggered when a user clicks on an element. It's widely used for handling button clicks, link clicks, or any other click interactions.

```
import React from 'react';

const MyComponent = () => {
  const handleClick = () => {
    console.log('Button clicked!');
  };

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



# onChange Event

- ❖ The onChange event is triggered when the value of an input element changes. It's commonly used for handling user input in forms.

```
const MyComponent = () => {  
  const [value, setValue] = useState('');  
  
  const handleChange = (event) => {  
    console.log('Input changed!');  
    console.log('New value:', event.target.value);  
    setValue(event.target.value);  
  };  
  
  return <input type="text" value={value} onChange={handleChange} />;  
};
```

# onSubmit Event

- ❖ The onSubmit event is triggered when a form is submitted. It's essential for form validation and submission.

```
const MyComponent = () => {  
  const handleSubmit = (event) => {  
    console.log('Form submitted!');  
  };  
  
  return (  
    <form onSubmit={handleSubmit}>  
      <button type="submit">Submit</button>  
    </form>  
  );  
};
```

# preventDefault

- ❖ The `preventDefault` method is called within event handlers to prevent the default behavior of an event. It's commonly used to prevent form submission or link navigation.

```
export default function App() {  
  const handleClick = (e) => {  
    e.preventDefault();  
    console.log('Hayyyyyyy!!!');  
  };  
  
  return (  
    <div>  
      <a href="/items" onClick={handleClick}>  
        Click me  
      </a>  
    </div>  
  );  
}
```

# Passing Arguments to Event Handlers

- ❖ You can pass additional arguments to event handlers using arrow functions.

```
export default function App() {  
  const handleClick = (arg) => {  
    console.log('Clicked with argument:', arg);  
  };  
  
  return <button onClick={() => handleClick('Hello')}>Click Me</button>;  
}
```



# Event propagation

- ❖ Event handlers will also catch events from any children your component might have.
- ❖ We say that an event “bubbles” or “propagates” up the tree: it starts with where the event happened, and then goes up the tree.



# Event propagation

```
export default function Toolbar() {  
  return (  
    <div className="Toolbar" onClick={() => {  
      alert('You clicked on the toolbar!');  
    }}>  
      <button onClick={(e) => {  
        |  
          alert('Playing!');  
        }}>  
        Play Movie  
      </button>  
    </div>  
  );  
}
```



# Stopping propagation

- ❖ That event object also lets you stop the propagation.

```
export default function Toolbar() {  
  return (  
    <div className="Toolbar" onClick={() => {  
      alert('You clicked on the toolbar!');  
    }}>  
      <button onClick={(e) => {  
        e.stopPropagation();  
        alert('Playing!');  
      }}>  
        Play Movie  
      </button>  
    </div>  
  );  
}
```



# How do we access the value of a text box within the event handler function?

- A. Copy the value from the text box using Ctrl+C.
- B. Send a GET request.
- C. Use the event object.



# How do we bind a click event handler to a button in React.js?

- A. Assign the event object to the onClick attribute of the button component.
- B. Create an event object and pass it to the button as a prop.
- C. Assign a function to the onClick attribute of the button component.

# Summary

## ❖ Key Takeaways:

- Event handlers are added as attributes to JSX elements.
- The event object passed to event handlers contains common properties like target, type and preventDefault.
- Events like onClick, onChange and onSubmit are commonly used for handling user interactions.

- ❖ **Homework:** Create a task listing app allowing the user to add, edit and remove tasks from a list.

# Questions and Answers



# Thank you for attending



Department  
for Education

CoGrammar

