CoGrammar

Welcome to this session:

Task Walkthrough - React - Local State Management and Events

The session will start shortly...

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

Safeguarding & Welfare

We are committed to all our students and staff feeling safe and happy; we want to make sure there is always someone you can turn to if you are worried about anything.

If you are feeling upset or unsafe, are worried about a friend, student or family member, or you feel like something isn't right, speak to our safeguarding team:

Ian Wyles
Designated Safeguarding Lead

Simone Botes

Nurhaan Snyman

Rafiq Manan

Ronald Munodawafa

Tevin Pitts

Scan to report a safeguarding concern

or email the Designated Safeguarding Lead:
Ian Wyles
safeguarding@hyperiondev.com

CoGrammar HyperionDev

Skills Bootcamp Cloud Web Development

- 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**

Skills Bootcamp Cloud Web Development

- For all **non-academic questions**, please submit a query: ***www.hyperiondev.com/support***

- **Report a safeguarding incident: *www.hyperiondev.com/safeguardreporting***

- We would love your feedback on lectures: ***Feedback on Lectures***

- If you are hearing impaired, please kindly use your computer's function through Google chrome to enable captions.

CoGrammar

React - Local State Management and Events

November 2024

Learning Outcomes

- **Set up and initialize a React project** using Vite.
- **Use React state and props to manage and share data** between components.
- **Implement interactive functionalities** such as updating balances and performing calculations dynamically.
- **Design reusable and styled components** to create a visually appealing application.

Lecture Overview

→ Presentation of the Task

→

→ Task Walkthrough

CoGrammar

React Task

Imagine you’re creating a Daily Expense Tracker App to help users keep track of their expenses and budget. Your task is to build an interactive app that allows users to log expenses, update their daily budget, and calculate how much money they have left to spend for the day.

Think of this app as a digital wallet that updates in real time with every expense logged or budget change made. It’s an engaging way to learn React while solving a problem most of us face daily—staying on budget! 💸

React Task

- ❖ Create a parent component (App.jsx) that manages the overall budget and logs the expenses.
- ❖ Create two child components:
  - ➢ **BudgetManager:** Handles updates to the daily budget.
  - ➢ **ExpenseLogger:** Manages expense inputs and logs them to a list.
- ❖ Use useState in the parent (App.jsx) to store:
  - ➢ The daily budget.
  - ➢ An array of expenses.
  - ➢ The remaining balance.

What is the primary role of state in React?

A. Styling components
B. Managing dynamic data in a component
C. Fetching data from an API
D. Navigating between pages

What does the term “lifting state up” refer to in React?

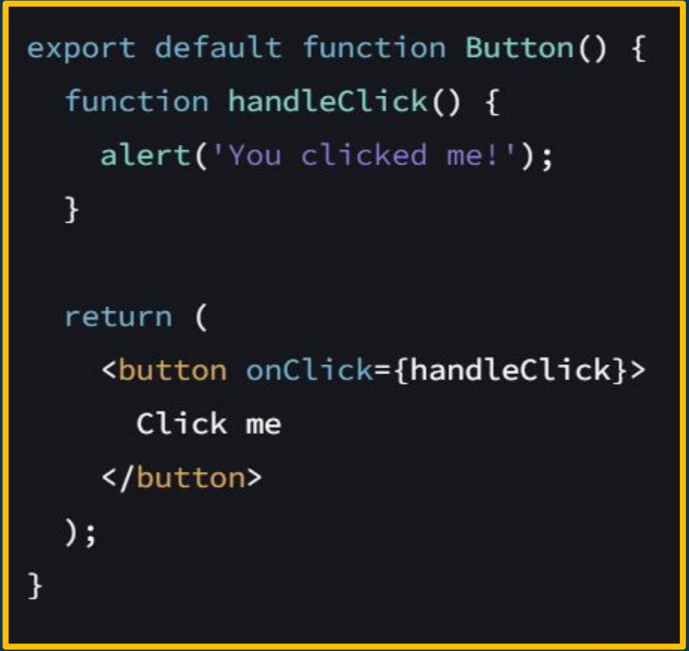
A. Moving state from one application to another
B. Adding animations to components
C. Styling components dynamically
D. Sharing state between sibling components via a common parent

CoGrammar

Events Handlers

- 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.
- Events like onClick, onChange, onSubmit, etc., enable users to interact with components, triggering updates and actions.
- To add an event handler, you will first **define a function** and then **pass it as a prop** to the appropriate JSX tag.

Events Handlers



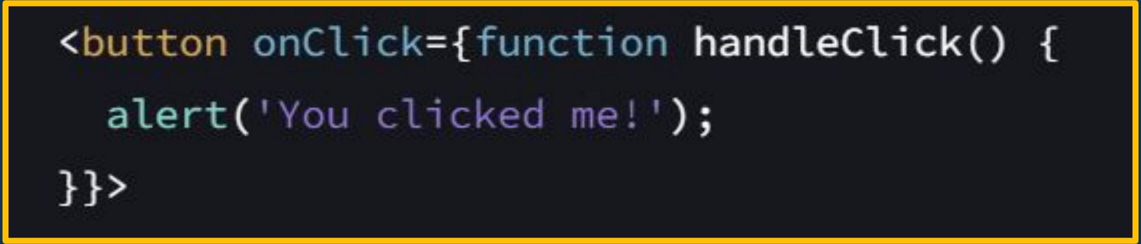
```
export default function Button() {
  function handleClick() {
    alert('You clicked me!');
  }

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

CoGrammar

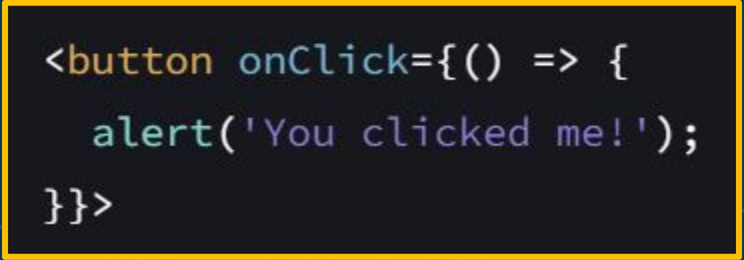
Events Handlers

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



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

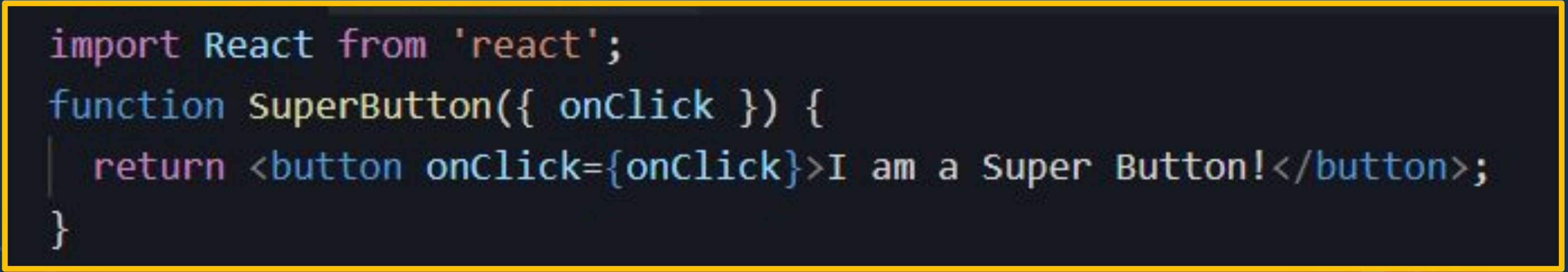
- Or, more concisely, using an arrow function:



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

Events Handlers

- ❖ 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>;
}
```

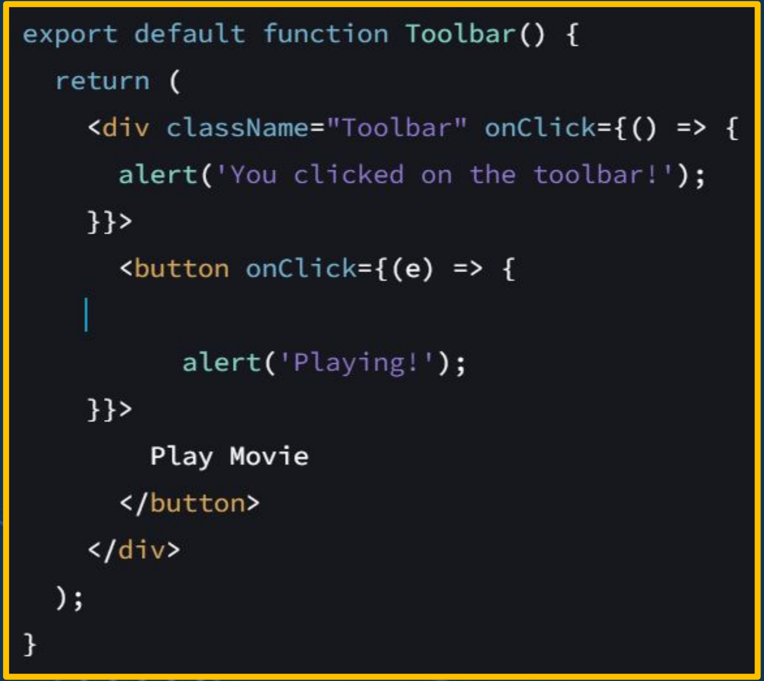
CoGrammar

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.
- The **event object** is used to find out more information about the event, like the type, target and control the default behaviour of the component.
- It also lets you stop the propagation.

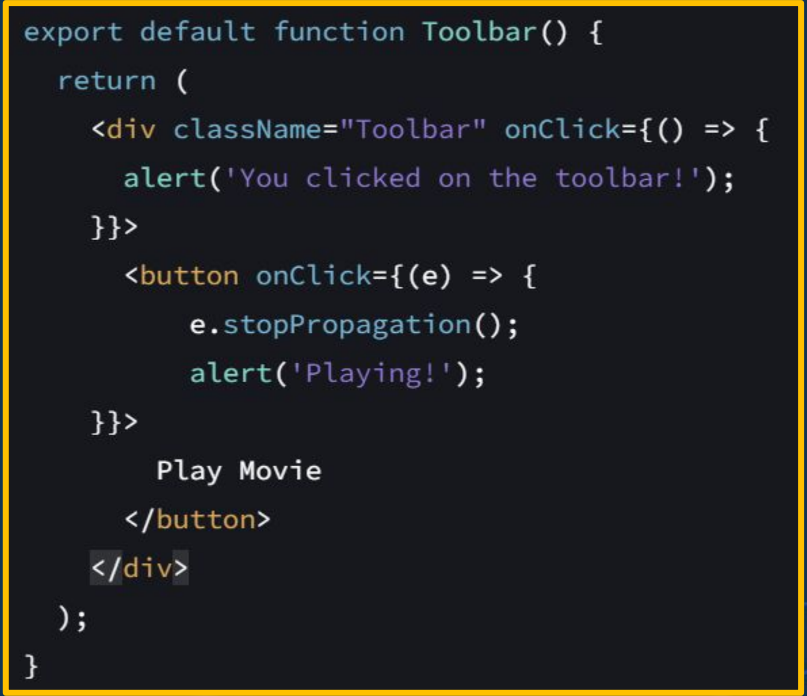
CoGrammar

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>
  );
}
```



```
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>
  );
}
```

CoGrammar

State Management

- ❖ **State management** is the process of **handling and updating data** within a React application.
- ❖ It allows components to **maintain their internal state** and **respond to user interactions effectively**.
- ❖ In React, **state** refers to **an object that represents the current condition of a component**.

CoGrammar

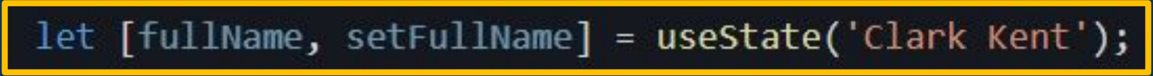
State Management

- **Stateful components** have the ability to **hold and modify their state**, which **affects their rendering and behavior**.
- When a component's state changes, React automatically **re-renders the component** to reflect the updated state.
- **Changes to state trigger a re-render of the component and its child components**, ensuring that the UI stays in sync with the underlying data.

CoGrammar

useState Hook

- In functional components, we use the **useState hook** to introduce stateful behavior.
- **React hooks** are powerful functions that can be used in a functional component to manipulate the state of the component.
- The useState hook allows us to **declare state variables** and **update them within the component.**



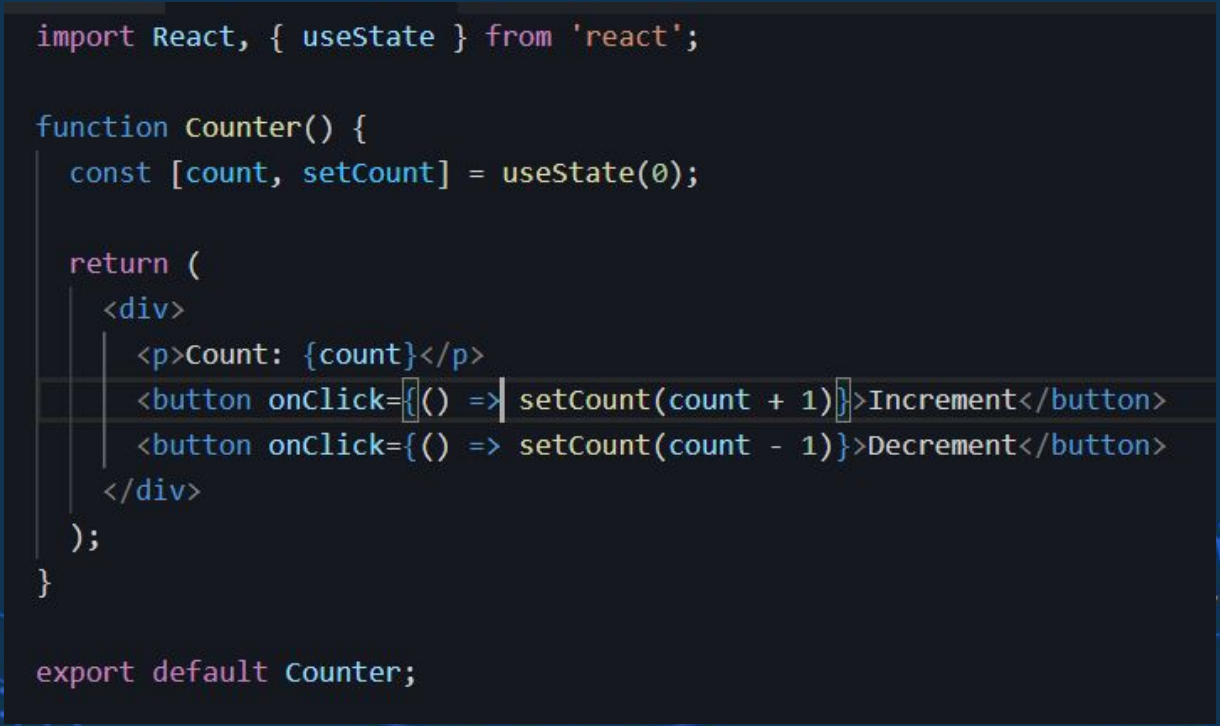
```
let [fullName, setFullName] = useState('Clark Kent');
```

state: Represents the current value of the state variable.

setState: A function used to update the state variable and trigger re-rendering.

CoGrammar

Example: Counter Component



```
import React, { useState } from 'react';

function Counter() {
  const [count, setCount] = useState(0);

  return (
    <div>
      <p>Count: {count}</p>
      <button onClick={() => setCount(count + 1)}>Increment</button>
      <button onClick={() => setCount(count - 1)}>Decrement</button>
    </div>
  );
}

export default Counter;
```

CoGrammar

Sharing State

- Often we want to **share state between multiple components** so when the state changes, **all dependent components re-render**.
- To do this, we **create a new parent component** to handle the state: creating state variables and functions which are called to change state.
- The state variables can then be passed to the child components as props.
- This process is often referred to as **"lifting state up"**.

CoGrammar

React Task

Imagine you're creating a Daily Expense Tracker App to help users keep track of their expenses and budget. Your task is to build an interactive app that allows users to log expenses, update their daily budget, and calculate how much money they have left to spend for the day.

Think of this app as a digital wallet that updates in real time with every expense logged or budget change made. It's an engaging way to learn React while solving a problem most of us face daily—staying on budget! 💸

React Task

- Create a parent component (App.jsx) that manages the overall budget and logs the expenses.
- Create two child components:
  - **BudgetManager:** Handles updates to the daily budget.
  - **ExpenseLogger:** Manages expense inputs and logs them to a list.
- Use useState in the parent (App.jsx) to store:
  - The daily budget.
  - An array of expenses.
  - The remaining balance.

What is the primary role of state in React?

A. Styling components
B. Managing dynamic data in a component
C. Fetching data from an API
D. Navigating between pages

What does the term “lifting state up” refer to in React?

A. Moving state from one application to another
B. Adding animations to components
C. Styling components dynamically
D. Sharing state between sibling components via a common parent

CoGrammar

Summary

- ★ **React Hooks:**
  - useState: How to manage component-level state dynamically.
  - Examples include storing and updating values like the daily budget or logged expenses.
- ★ **Event Handling in React:**
  - How React handles user interactions, such as button clicks or form submissions.
  - Using onClick and onChange to trigger updates to the app's state.
- ★ **State Lifting:**
  - Sharing state between components by "lifting" it to a common parent.
  - How this allows components like BudgetManager and ExpenseLogger to interact seamlessly.

CoGrammar

Q & A SECTION

Please use this time to ask any questions relating to the topic, should you have any.

Thank you for attending

CoGrammar

SKILLS FOR LIFE SKILLS BOOTCAMPS

Department for Education