CoGrammar

Welcome to this session:

Local State Management and Events (Tutorial)

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.

Learning Outcomes

By the end of this lesson, learners should be able to:

- **Learn different methods to handle events** in React components, including inline handlers, method binding, and arrow functions.
- **Define the concept of state management** in React and identify how the state is managed within components.
- **Implement the useState() hook** to declare and manage local state in React functional components.

Why do components in React start with a capital letter?

A. React requires component names to start with a capital letter to distinguish them from regular HTML elements.
B. It is a convention to improve code readability and maintainability.
C. React automatically converts lowercase component names to uppercase during rendering.
D. It helps to define components that will be passed as props to other components.

CoGrammar

What is the main difference between props and state in React?

A. Props are used to pass data from parent to child components, and state is used to store data that changes over time within a component.
B. Props are used to manage internal data, while state is used to pass data from parent to child components.
C. Props are mutable, meaning they can be changed, while state is immutable.
D. Props are used only in class components, and state is used only in functional components.

CoGrammar

What is the primary difference between a stateful and stateless component in React?

A. A stateful component has internal state that can be modified, while a stateless component only receives props and cannot modify its state.
B. A stateless component has internal state, while a stateful component does not.
C. A stateful component is used for UI rendering, while a stateless component is used for business logic.
D. There is no difference; both are the same in React.

CoGrammar

Lecture Overview

- → Recap on Components
- → Recap on Props
- → Recap on Events in React
- → Recap on State Management
- → Implement the useState( ) hook
- → Spread Operator & filter( ) method
- → Tutorial (Live Coding)

What are events in React?

- ❖ Events in React are occurrences or actions, such as user inputs (clicking, typing) or system-driven actions, that React components can respond to.
- ❖ **Synthetic Events:** React provides a "synthetic event" system, which standardizes browser events. This ensures consistent behavior across various browsers.

CoGrammar

React Components

- ❖ A Component is considered as the **core building blocks** of a React application.
- ❖ It makes the task of building UIs much easier.
- ❖ Each component exists in the **same space**, but they **work independently** from one another and **merge all in a parent component**, which will be the **final UI of your application**.
- ❖ All React components have their own structure, methods as well as APIs. They can be reusable as per your need.

CoGrammar

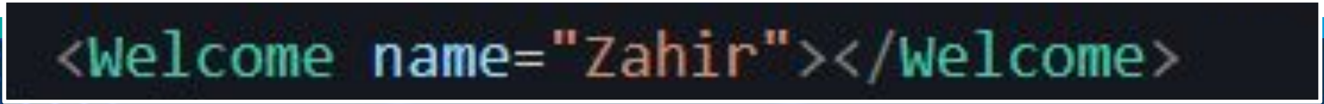
Functional Components

- ❖ To write this component, we will be using JSX. JSX enables you to write HTML-like markup within a JavaScript file, keeping rendering logic and content seamlessly integrated.
- ❖ When creating React components it is extremely important that you always adhere to the following two rules:
  - ➢ **Always** start component names with a **capital letter**. React treats components starting with lowercase letters as DOM elements.
  - ➢ A component should **never** modify its **own props**.

CoGrammar

What are Props in React?

- **Props are inputs to components**: Props (short for "properties") are used to pass data from a parent component to a child component in React. They are read-only and cannot be modified by the child component.
- **Props allow customization of components**: By passing different values through props, you can customize the behavior and appearance of a component for different use cases without altering its internal structure.
- **Props are immutable**: Once a prop is passed to a child component, it cannot be changed by that child. The child component can only access and use the values, not modify them.
- **Props can be any data type**: Props can hold any type of data, including strings, numbers, objects, arrays, and functions, enabling flexible and dynamic component design.
- **Props enable reusability**: Because props make components configurable, they allow you to reuse the same component with different data, improving code modularity and maintainability.



```
<Welcome name="Zahir"></Welcome>
```

CoGrammar

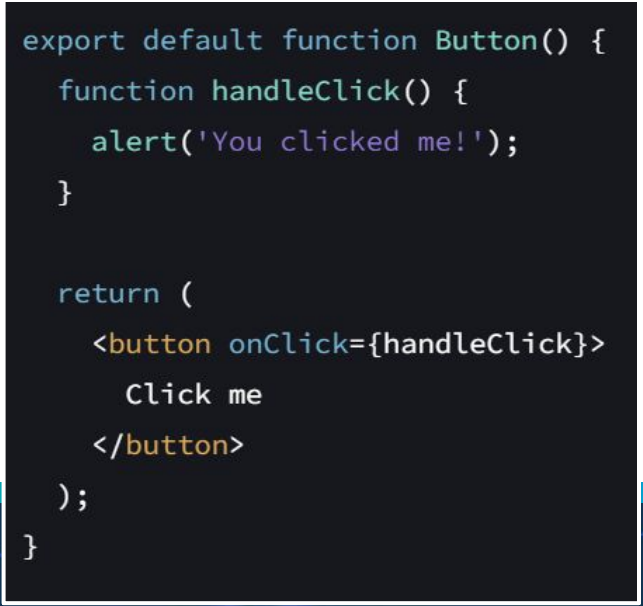
What are events handlers in React?

- ❖ 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.

CoGrammar

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.

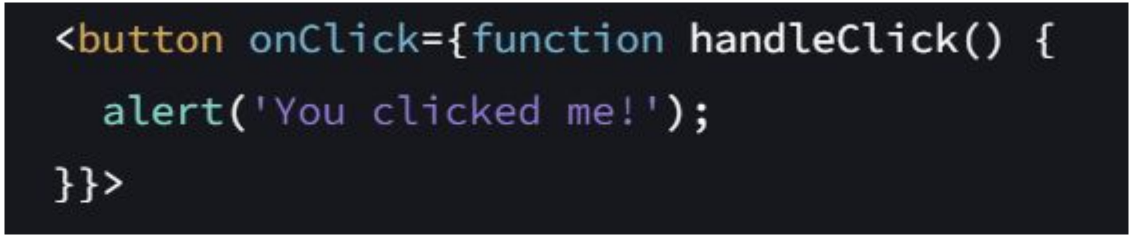


```
export default function Button() {
  function handleClick() {
    alert('You clicked me!');
  }

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

CoGrammar

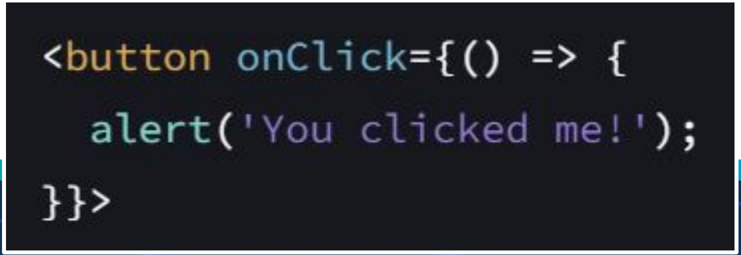
Adding event handlers

- 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:

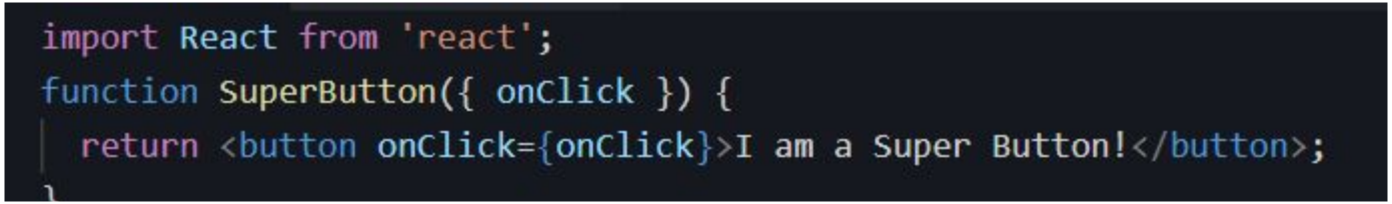


```
<button onClick={() => {
  alert('You clicked me!');
}}>
```

CoGrammar

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>;
```

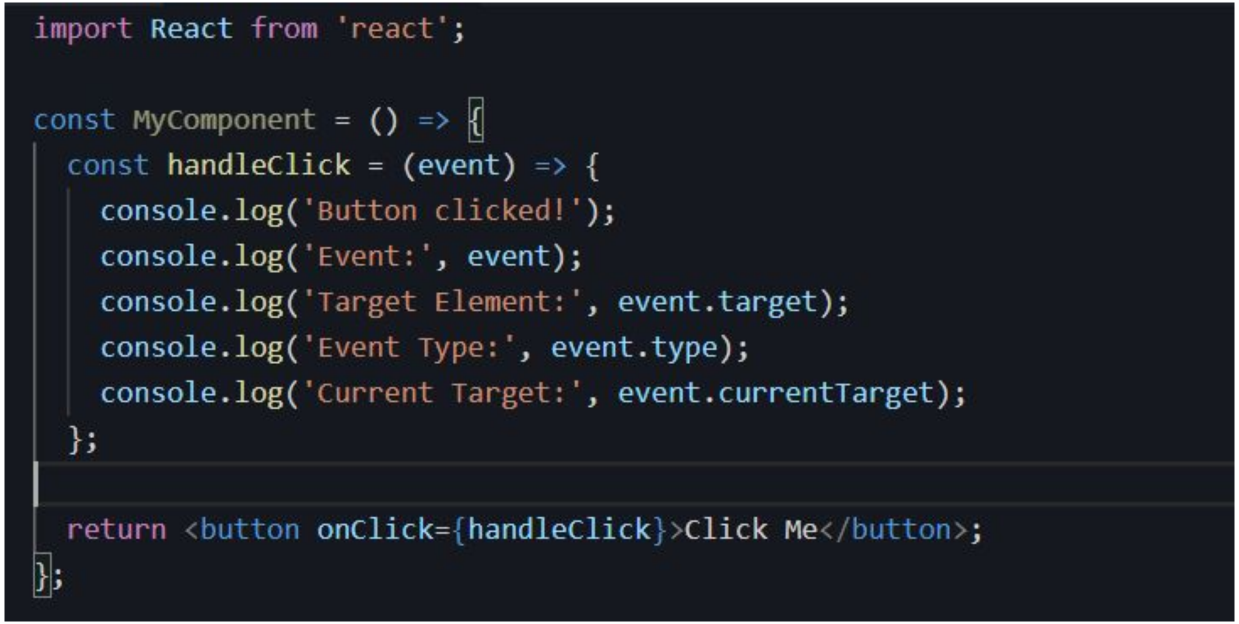
CoGrammar

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.

CoGrammar

Event Object

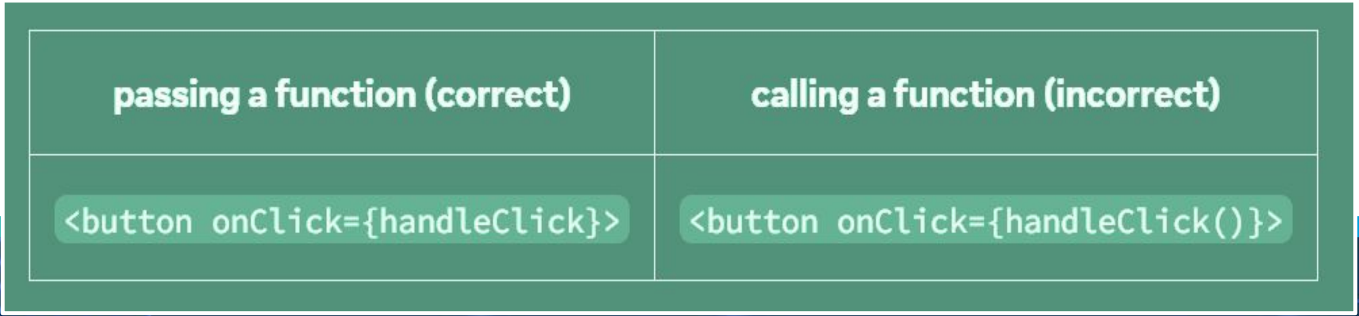


```
import React from 'react';

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>;
};
```

CoGrammar

Remember

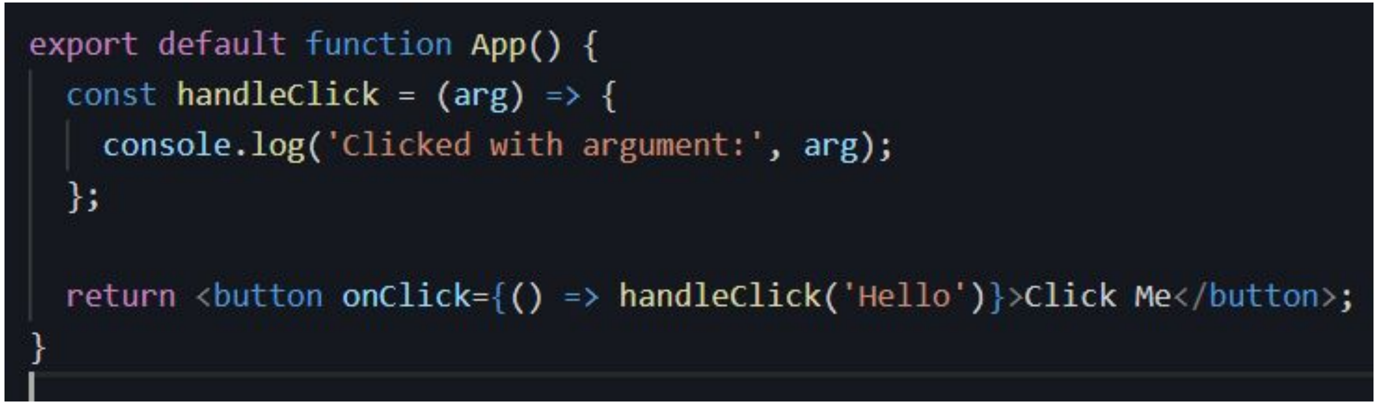
- ❖ Functions passed to event handlers must be passed, not called.
- ❖ The () at the end of handleClick() fires the function immediately during rendering, without any clicks. This is because JavaScript inside the JSX **{ }** executes right away.



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

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>;
}
```

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

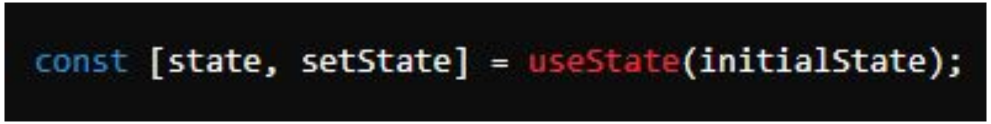
useState( ) Hook

- Before React introduced **hooks**, only **class components** could have state, but with useState, **functional components** can now manage and update state without needing to be converted to class components.
- **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.**

CoGrammar

useState( ) syntax

- **Declaration**: You call useState inside a functional component to declare a state variable and a function to update that state.
- **Syntax**:

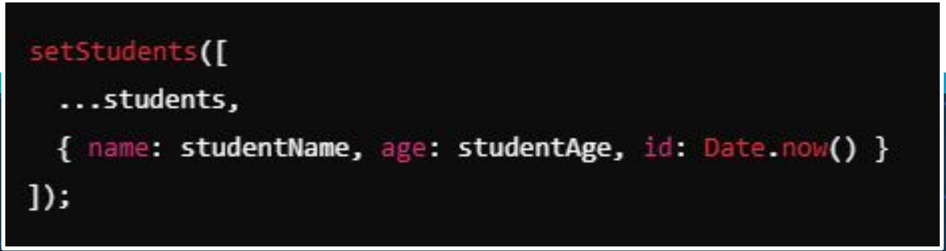


```
const [state, setState] = useState(initialState);
```

  - **state** is the variable that holds the current state value.
  - **setState** is the function used to update that state.
  - **initialState** is the initial value of the state.

The spread operator (...) in JavaScript

- ❖ A powerful feature that allows you to copy the elements of an array or object into another array or object.
- ❖ Example explanation:
  - ➢ **students** is an array that holds the current list of students.
  - ➢ The **spread operator** (...students) takes all the elements from the students array and places them into the new array. This ensures that the new array includes all of the existing students.
  - ➢ **{ name: studentName, age: studentAge, id: Date.now() }** is creating a new student object with the name, age, and a unique id.

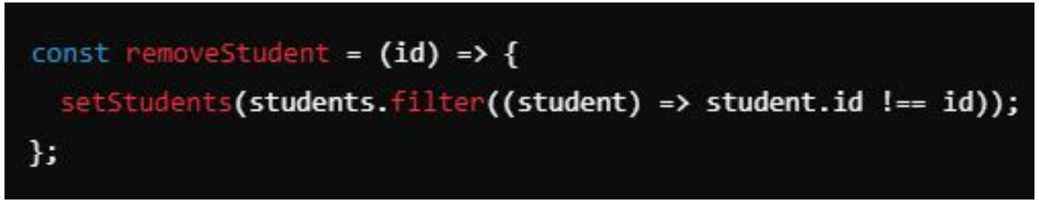


```
setStudents([
  ...students,
  { name: studentName, age: studentAge, id: Date.now() }
]);
```

CoGrammar

The filter( ) method in JavaScript

❖ It is used to create a **new array** that contains only the elements that **meet a specific condition**. It does not modify the original array but instead returns a new array with the filtered results.



```
const removeStudent = (id) => {
  setStudents(students.filter((student) => student.id !== id));
};
```

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”**.

What is the purpose of the spread operator (...) in JavaScript?

A. It is used to combine multiple objects or arrays into one, or to spread elements in an array or properties in an object.
B. It is used to copy the values from one variable to another.
C. It is used to define a function that accepts multiple arguments in a specific format.
D. It is used to destructure an object or array into individual values.

CoGrammar

What is the purpose of the filter( ) method in JavaScript?

A. It creates a new array with all elements that pass a test provided by a function.
B. It modifies the original array by removing elements that don't pass a test.
C. It returns the first element that matches a condition from an array.
D. It sorts an array in ascending order based on a given condition.

CoGrammar

Let's take a break

CoGrammar

Questions and Answers

CoGrammar

Thank you for attending

CoGrammar

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education