CoGrammar

Welcome to this session:

Task Walkthrough

Skills Bootcamp - Task 18

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 Full-Stack Software 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 Full-Stack Software 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 Objectives

- Recall the basic structure of a React component.
- Explain how React manages user interactions through event handlers.
- Implement state management in a React component using useState.
- Distinguish between parent and child components and their respective responsibilities.
- Assess the usability of the Todo List app and suggest improvements (e.g., better styling, additional features).

Which of the following is a valid React event handler for a form input change?

A. onChange={handleClick}
B. onClick={handleChange}
C. onSubmit={handleInputChange}
D. onChange={handleInputChange}

Lecture Overview

- ➔ Presentation of the Task
- ➔ Introduction
- ➔ Task Walkthrough

Task: Build an Interactive To-Do List

- **Objective**: Apply your knowledge of state and events in React to create a To-Do list application.
- **Requirements**:
  - Users enter a task in the input field.
  - Clicking Add Task adds the task to the list and clears the input.
  - Clicking a task toggles its completion status.
  - Clicking the Delete button removes the task.

Setting up the App

- Initialise a React application using {npm create vite@latest
- Include Bootstrap/tailwind CSS for styling
- Initialise state
  - Tasks: An array to hold the list of tasks
  - Task: A string to hold the current input value

App.jsx

```
5    const [tasks, setTasks] = useState([]);
6    const [task, setTask] = useState("");
```

Snipped

Adding a task

- Adds a new task to the tasks array
- Prevents adding empty tasks with the if condition
- Resets the input field after adding a task

App.jsx

```
8     // Add a task
9     const addTask = () => {
10      if (task.trim() === "") return;
11      setTasks([...tasks, { text: task, completed: false }]);
12      setTask("");
13    };
```

Snipped

CoGrammar

Displaying a task

- Use map to render each tasks as a list item
- Applies conditional styling to completed tasks

App.jsx

```jsx
{/* Task List */}
<ul className="list-group mt-4">
  {tasks.map((t, index) => (
    <li
      key={index}
      className={`list-group-item d-flex justify-content-between align-items-center ${
        t.completed ? "list-group-item-success" : ""
      }`}
    >
      <span
        onClick={() => toggleComplete(index)}
        style={{ textDecoration: t.completed ? "line-through" : "none", cursor: "pointer" }}
      >
        {t.text}
      </span>
      <button className="btn btn-danger btn-sm" onClick={() => deleteTask(index)}>
        Delete
      </button>
    </li>
  ))}
</ul>
```

Snipped

CoGrammar

Marking a task as complete

- Toggles the completed state of the task
- Update the task array using map

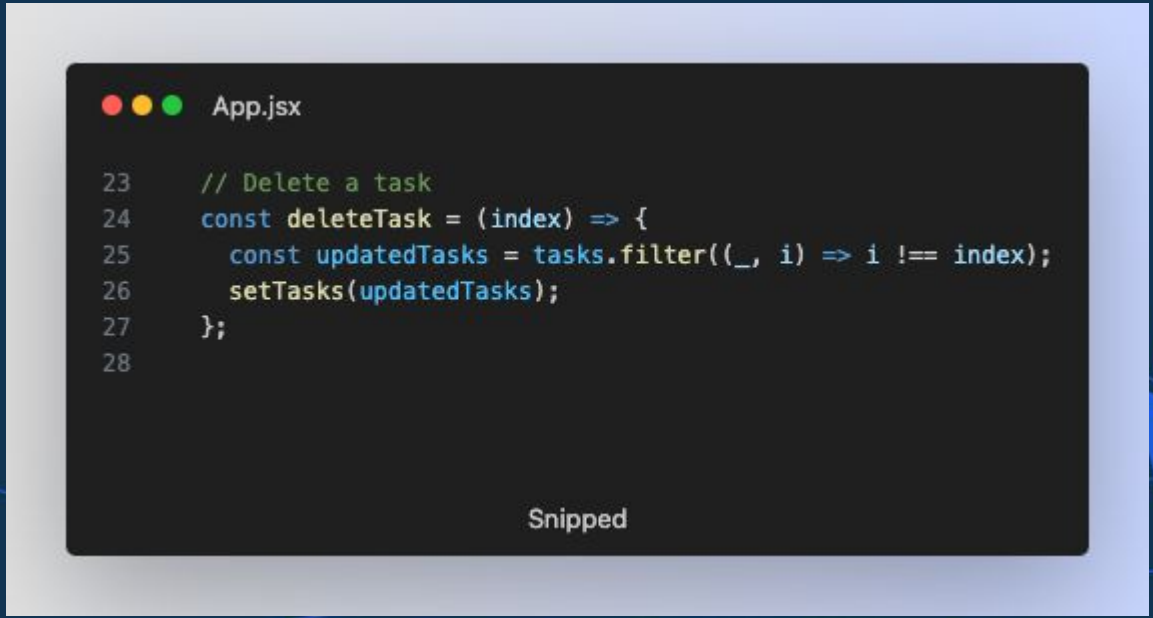
App.jsx

```
15    // Mark task as completed
16    const toggleComplete = (index) => {
17      const updatedTasks = tasks.map((t, i) =>
18        i === index ? { ...t, completed: !t.completed } : t
19      );
20      setTasks(updatedTasks);
21    };
22
```

Snipped

Deleting a task

- Removes the task at the specified index using filter
- Updates the state with the filtered task list



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