Welcome to the CoGrammar Lecture: Custom Hooks

The session will start shortly...

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

CoGrammar

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
- Report a **safeguarding** incident: www.hyperiondev.com/safeguardreporting
- We would love your **feedback** on lectures: Feedback on Lectures

CoGrammar

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

CoGrammar

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

CoGrammar

Learning Objectives

- ❖ Explain the purpose and advantages of creating custom hooks in React
- ❖ Design and implement custom hooks
- ❖ Utilize custom hooks for state management
- ❖ Demonstrate the use of custom hooks for handling side effects
- ❖ Incorporate custom hooks in functional components

CoGrammar

React Hooks

JavaScript functions that allow functional components to access React features, like state and side effects.

- ❖ Before the Hooks, **class components** were used, which allowed internal state to be managed and lifecycle events to be handled directly.
- ❖ React Hooks allow us to work with React components in a **simpler and more concise** way, without having to write classes.
- ❖ Hooks also make our code more **readable** and **maintainable**.
- ❖ There are many types of hooks, and **custom hooks** can be defined as well.
- ❖ This lecture will be covering custom hooks

CoGrammar

Introduction to Custom Hooks

- ❖ Custom hooks are functions that let you "hook into" React state and lifecycle features from function components. They can be reused across multiple components.
- ❖ They help in avoiding code duplication and abstracting component logic, making your code cleaner and easier to maintain.

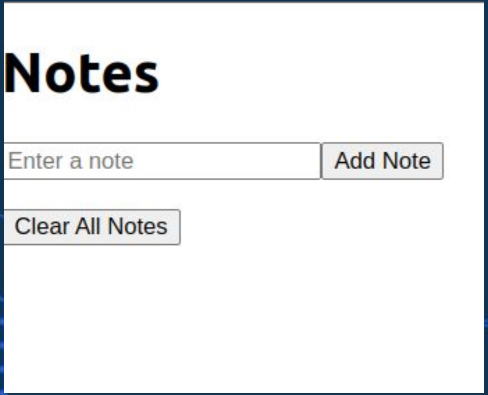
Rules of Custom Hooks

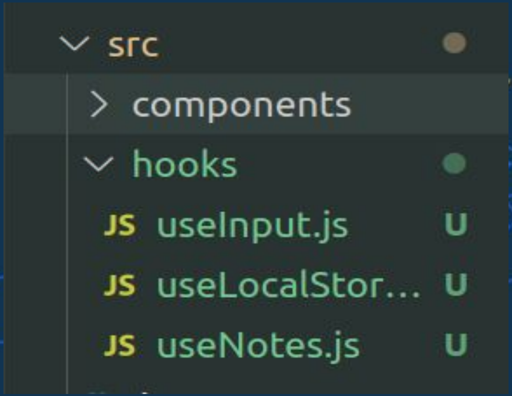
- ❖ Naming: Custom hooks must start with use (e.g., useForm).
- ❖ Calling Hooks: Only call hooks at the top level of a function, not inside loops, conditions, or nested functions.

CoGrammar

Setting Up the Projects

- Initialize the project with Create React App:
  - npx create-react-app notes-app
  - cd notes-app
- Create a basic project structure with src/hooks for custom hooks and src/components for React components.
- Create a simple notes User Interface(UI)



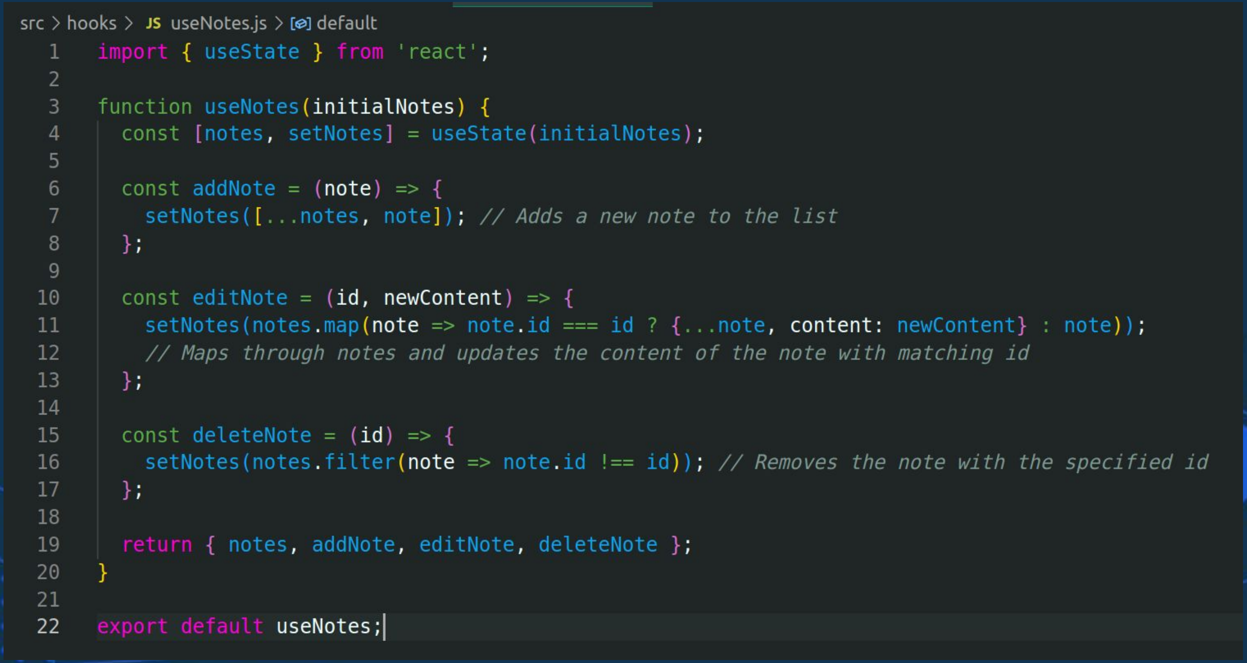


```
src
  components
  hooks
    useInput.js          U
    useLocalStor...      U
    useNotes.js          U
```

CoGrammar

Implementing a Custom Hook - `useNotes`

- The objective of `useNotes` is to manage the state of notes including functionalities to add, edit, and delete notes.

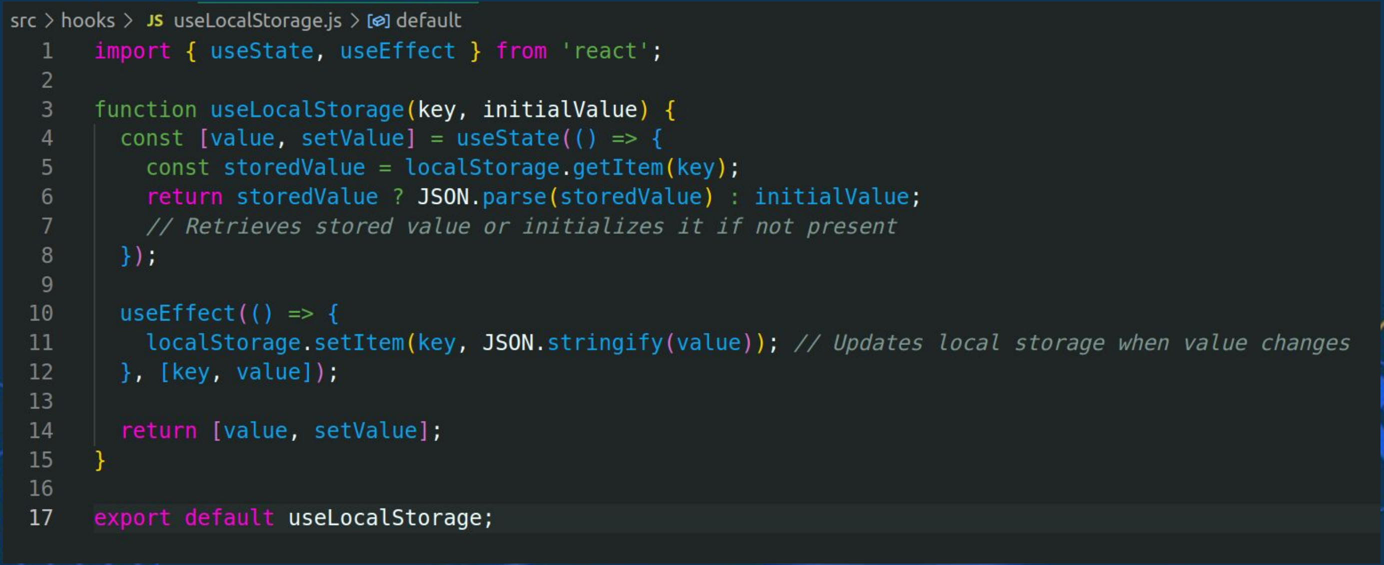

src > hooks > JS useNotes.js > default

```javascript
import { useState } from 'react';

function useNotes(initialNotes) {
  const [notes, setNotes] = useState(initialNotes);

  const addNote = (note) => {
    setNotes([...notes, note]); // Adds a new note to the list
  };

  const editNote = (id, newContent) => {
    setNotes(notes.map(note => note.id === id ? {...note, content: newContent} : note));
    // Maps through notes and updates the content of the note with matching id
  };

  const deleteNote = (id) => {
    setNotes(notes.filter(note => note.id !== id)); // Removes the note with the specified id
  };

  return { notes, addNote, editNote, deleteNote };
}

export default useNotes;
```

CoGrammar

Implementing a Custom Hook - `useLocalStorage`

- The objective of `useLocalStorage` is to sync state(notes’ state) with local storage, making state persistent across browser sessions.



src > hooks > JS useLocalStorage.js > default

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

function useLocalStorage(key, initialValue) {
  const [value, setValue] = useState(() => {
    const storedValue = localStorage.getItem(key);
    return storedValue ? JSON.parse(storedValue) : initialValue;
    // Retrieves stored value or initializes it if not present
  });

  useEffect(() => {
    localStorage.setItem(key, JSON.stringify(value)); // Updates local storage when value changes
  }, [key, value]);

  return [value, setValue];
}

export default useLocalStorage;
```

CoGrammar

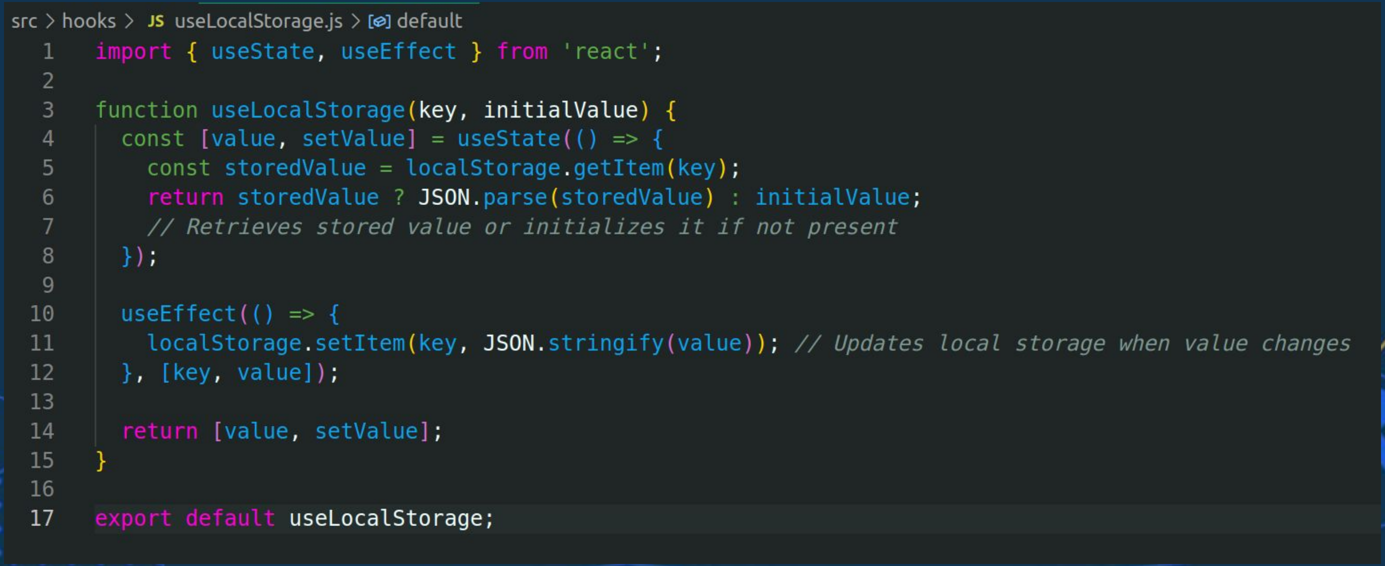
Let’s Breathe!

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

CoGrammar

Implementing a Custom Hook - `useInput`

- The objective of `useInput` is to simplify the management of form input states and changes.

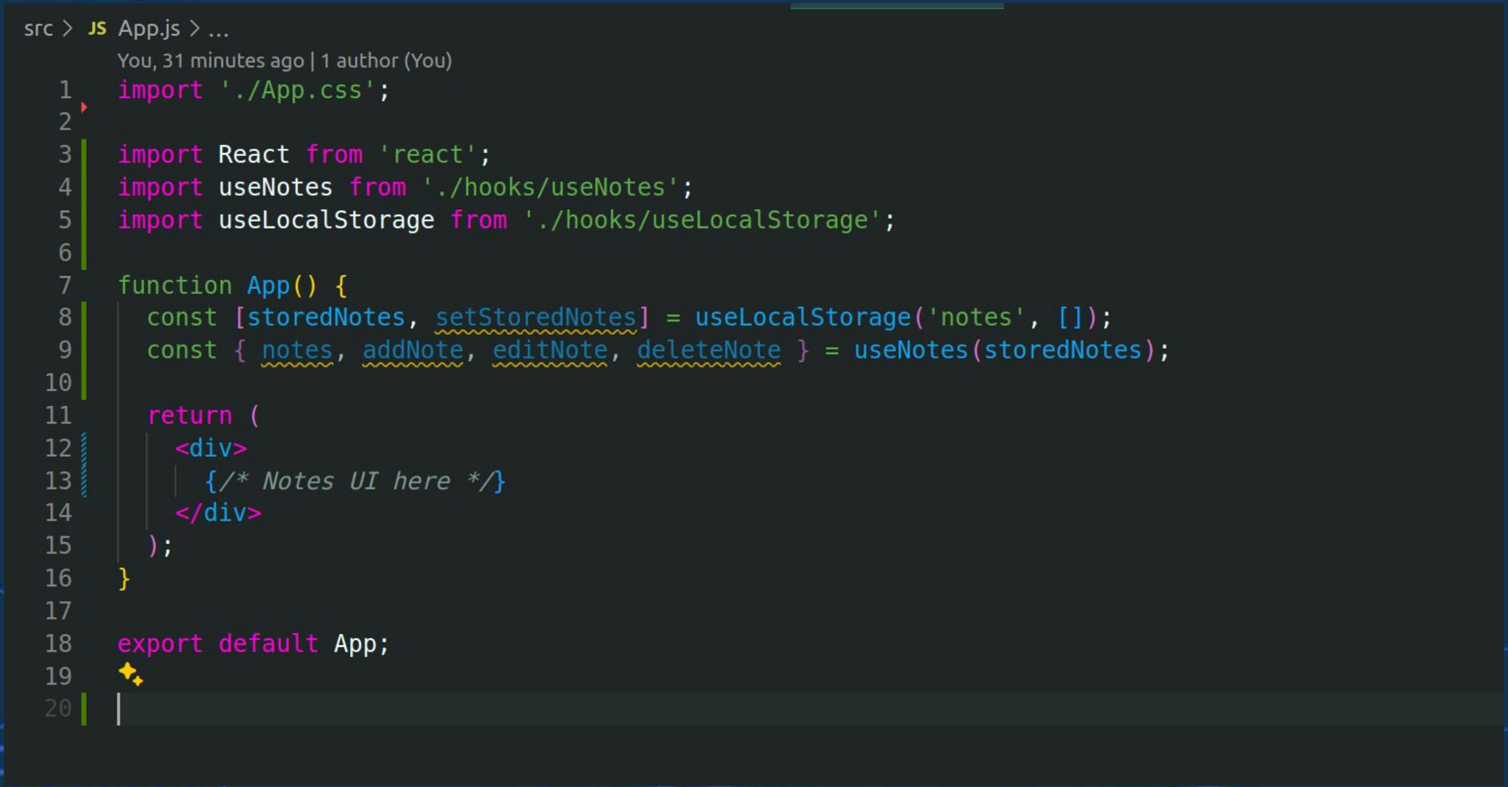


src > hooks > JS useLocalStorage.js > default

```javascript
import { useState, useEffect } from 'react';

function useLocalStorage(key, initialValue) {
  const [value, setValue] = useState(() => {
    const storedValue = localStorage.getItem(key);
    return storedValue ? JSON.parse(storedValue) : initialValue;
    // Retrieves stored value or initializes it if not present
  });

  useEffect(() => {
    localStorage.setItem(key, JSON.stringify(value)); // Updates local storage when value changes
  }, [key, value]);

  return [value, setValue];
}

export default useLocalStorage;
```

CoGrammar

Integrating Hooks into the Application



src > JS App.js > ...

You, 31 minutes ago | 1 author (You)

```
import './App.css';

import React from 'react';
import useNotes from './hooks/useNotes';
import useLocalStorage from './hooks/useLocalStorage';

function App() {
  const [storedNotes, setStoredNotes] = useLocalStorage('notes', []);
  const { notes, addNote, editNote, deleteNote } = useNotes(storedNotes);

  return (
    <div>
      {/* Notes UI here */}
    </div>
  );
}

export default App;

```

CoGrammar

Questions and Answers

CoGrammar

Thank you for attending

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education

CoGrammar