# Welcome to the CoGrammar

# Tutorial 13: Full Stack App with custom hooks

## The session will start shortly...

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

CoGrammar

## Data Science 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**

CoGrammar

## **Data Science 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

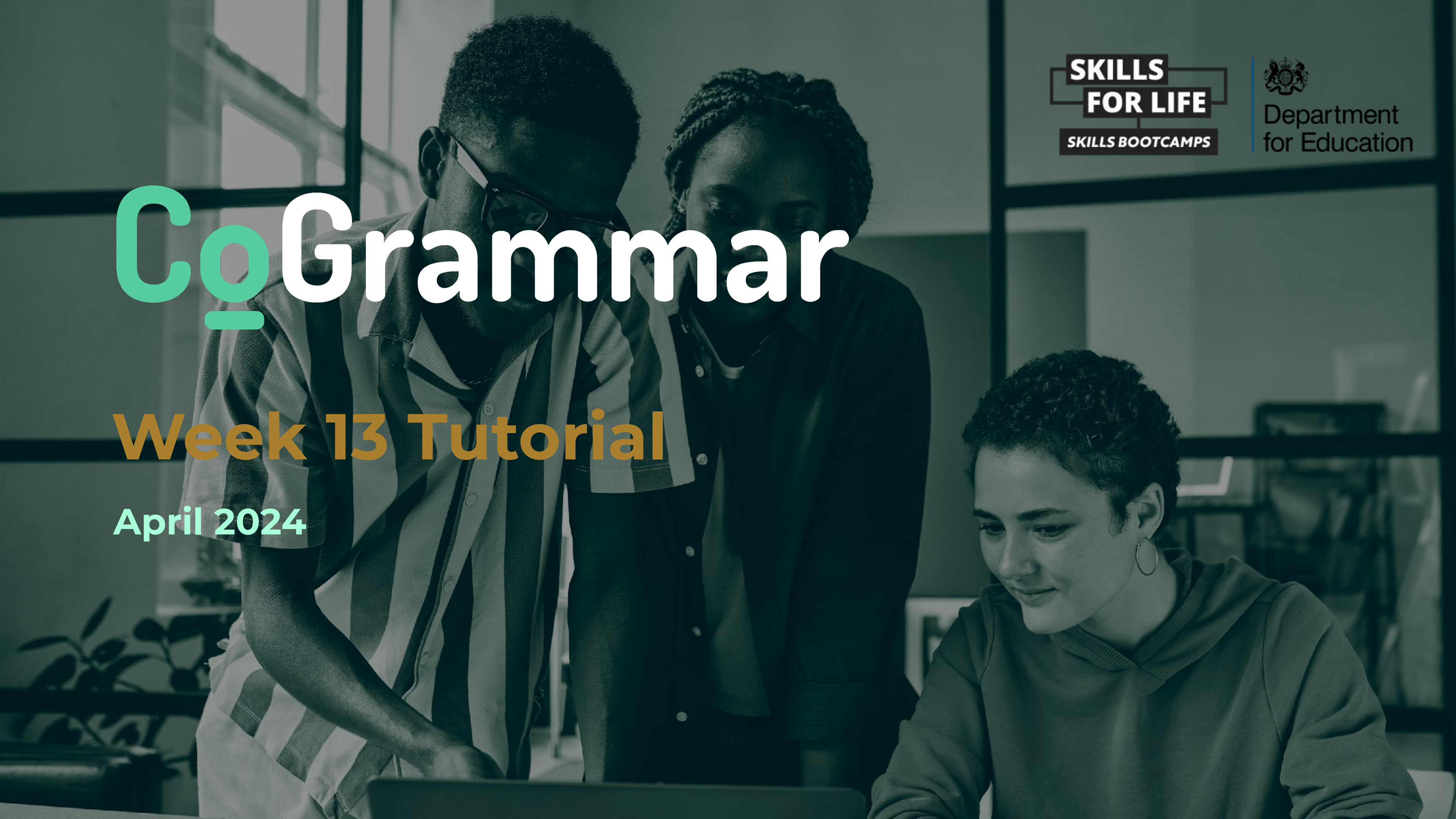
### ✅ Criterion 3: Course Progress

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



SKILLS FOR LIFE
SKILLS BOOTCAMPS
Department for Education

# CoGrammar

## Week 13 Tutorial

April 2024

## Learning objectives

- ❖ In depth understanding of hooks and custom hooks
- ❖ Build a full stack application
  - ➢ Set up a backend
  - ➢ Set up a frontend
  - ➢ Connecting BE and FE
- ❖ Fetch multiple endpoints using a custom hook demonstrating the importance of custom hooks

CoGrammar

## Full Stack Application setup.

CoGrammar

## Let’s code along...

CoGrammar

## Let’s Breathe!

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

CoGrammar

## Custom Hooks Recap

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

# Creating a Custom Hook

❖ In your **src** folder of your client app, create a folder called **hooks** where your custom hooks will be stored.

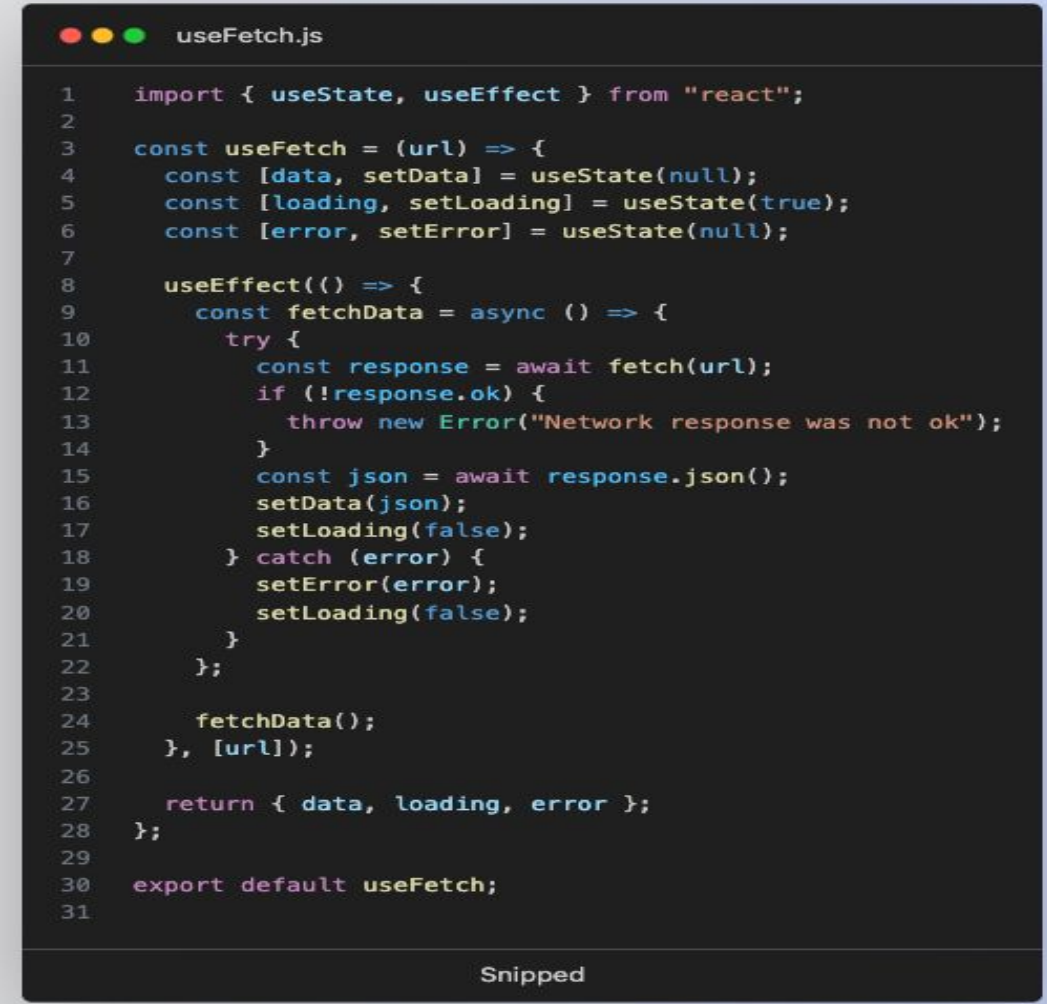
```
src/
│
├── components/
│   ├── Component 1.js
│   └── ...
│
├── hooks/
│   ├── useFetch.js
│   └── ...
├── App.js
└── index.js
```

CoGrammar

## Creating a Custom Hook

- The next code snippet represents a custom hook called useFetch()
- It is a normal function that performs a certain task and returns only necessary data to be utilized by the end user of the application.
- This custom hook will be used to refetch APIs in our components instead of creating new fetch functionalities each and every time.

CoGrammar



useFetch.js

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

const useFetch = (url) => {
  const [data, setData] = useState(null);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);

  useEffect(() => {
    const fetchData = async () => {
      try {
        const response = await fetch(url);
        if (!response.ok) {
          throw new Error("Network response was not ok");
        }
        const json = await response.json();
        setData(json);
        setLoading(false);
      } catch (error) {
        setError(error);
        setLoading(false);
      }
    };

    fetchData();
  }, [url]);

  return { data, loading, error };
};

export default useFetch;
```

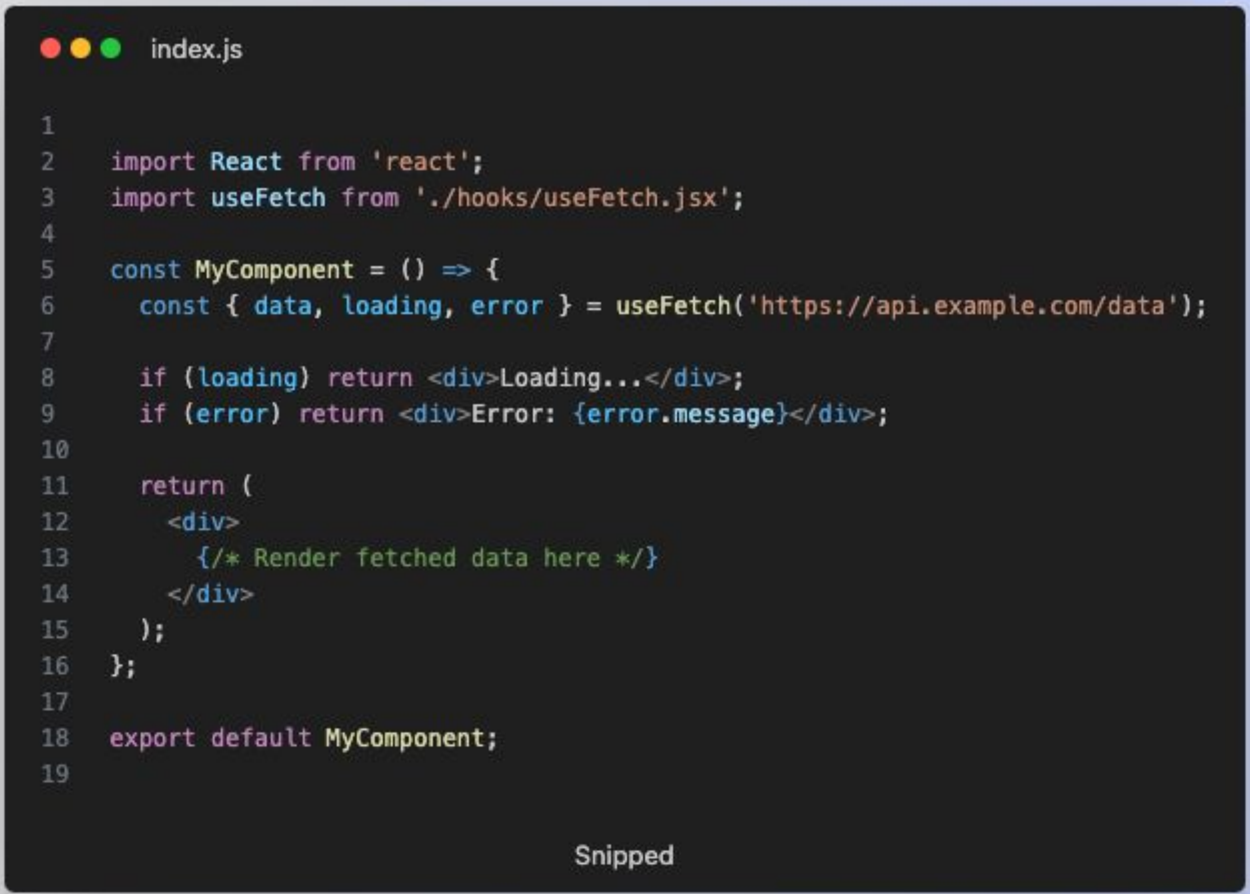
Snipped

CoGrammar

## Using a Custom Hook

- From the next slide, using a Custom hook becomes straightforward, you just need to call the hook and pass in the required argument and it should perform the data fetching process for you.

CoGrammar


index.js

```

import React from 'react';
import useFetch from './hooks/useFetch.jsx';

const MyComponent = () => {
  const { data, loading, error } = useFetch('https://api.example.com/data');

  if (loading) return <div>Loading...</div>;
  if (error) return <div>Error: {error.message}</div>;

  return (
    <div>
      {/* Render fetched data here */}
    </div>
  );
};

export default MyComponent;

```

CoGrammar

## Which aspect of React’s custom hooks do you find most impactful?

A. Enhanced Code Reusability
B. Improved Component Logic Organization
C. Streamlined State Management
D. Optimized Performance

CoGrammar

## Which scenario best describes your primary use case for React custom hooks?

A. Simplifying Complex State Logic

B. Abstracting API fetching

C. Encapsulating Browser API Interactions

D. Managing Lifecycle Events.

## Questions and Answers

CoGrammar

# Thank you for attending

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education

CoGrammar