CoGrammar

Welcome to this session:

Task Walkthrough - React - Routing

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 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

- ❖ **Set up and configure React Router** using hooks for navigation and route handling.
- ❖ **Use useNavigate and useLocation hooks** to manage app navigation programmatically.
- ❖ **Create dynamic, reusable components** to display recipes and manage state across pages.
- ❖ **Implement conditional rendering** to toggle features like login/logout and favorite recipes.
- ❖ **Style the app using Bootstrap or custom CSS** to enhance user experience.

Lecture Overview

→ Presentation of the Task
→ Introduction to Routing
→ React Router
→ React Router Hooks
→ Task Walkthrough

CoGrammar

Routing Task

Welcome to the Recipe Hub, where your love for cooking meets the power of React! In this task, you'll build a personalized recipe-sharing app where users can log in, explore mouth-watering recipes, save their favorites, and learn about the app's story—all while enjoying smooth navigation and a seamless user experience. 🌟

This is your chance to create an app that combines functionality, interactivity, and mouth-watering visuals. Let your imagination run wild and make this hub a delicious treat for your users. Let's get cooking! 👩‍🍳👨‍🍳

Here’s what your app will include:

1. **Home Page:** A warm welcome! Users can log in and out with their name, and the app will greet them personally.
2. **Recipes Page:** A feast for the eyes! Display recipes as stylish cards, complete with ingredients, preparation time, and a button to save favorites.
3. **Favorites Page:** A curated collection of favorite recipes saved by the user—perfect for planning your next meal!
4. **About Page:** Share the story behind your Recipe Hub, showcase stunning food visuals, and provide contact details for budding chefs to reach out.

What is the purpose of the useNavigate hook in React Router?

A. To style components dynamically
B. To programmatically navigate between routes
C. To fetch data from APIs
D. To manage component state

CoGrammar

What does the useState hook do in React?

A. Manages state within a functional component
B. Fetches data from APIs
C. Handles CSS styling
D. Manages routing logic

CoGrammar

Routing

Definition and Use Cases

- Routing can be termed as the **conditional rendering** of components based on the **URL** in the browser.
- Routing allows users to **navigate between different pages or views** within a web application.
- Routing with plain HTML/CSS used to be **file based**, the anchor (<a></a>) were used to create hyperlinks that link to different web pages which were the different (.html) files in your project.

CoGrammar

Routing in React

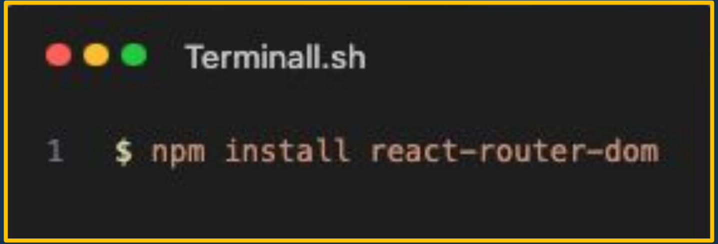
- In the context of React, **client side routing** is executed.
- This allows your app to **update the URL from a link click** without making another request for another document from the server, making your application **render immediately**.
- In simple terms, routing in React involves **dynamically updating the content** of the website without reloading the entire page.
- Routing in React is mostly implemented using **routing libraries** or frameworks. Two common libraries in use for a seamless routing experience are **React Router DOM** and **Reach Router**.

CoGrammar

React Router DOM

Achieves client side routing in your React application by using its inbuilt routing APIs.

❖ To use React Router in your application, you need to install it first using npm or yarn



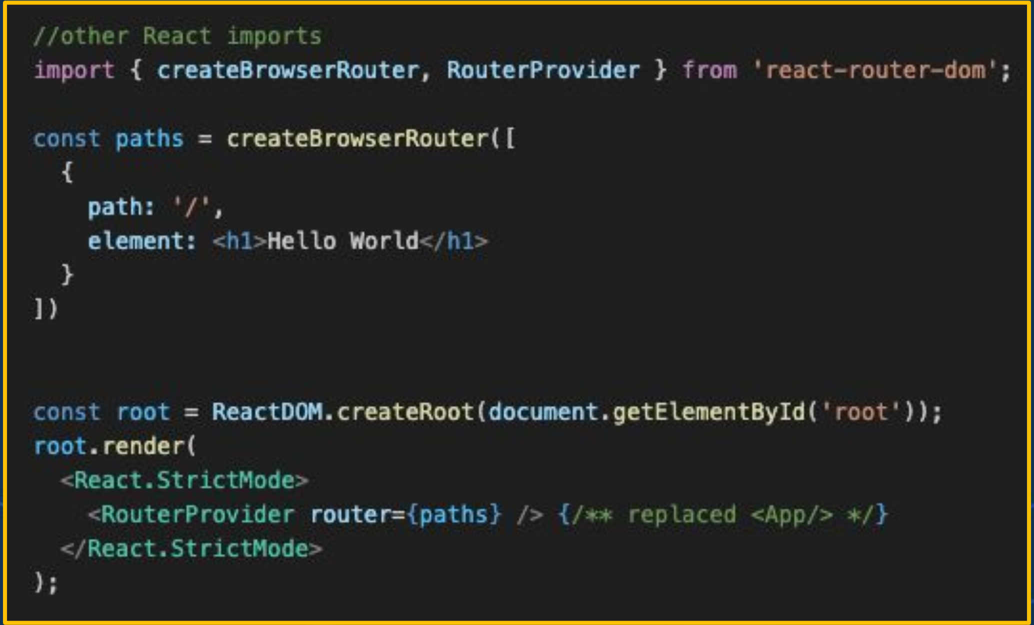
Terminall.sh

```
$ npm install react-router-dom
```

CoGrammar

Configuration

- ❖ After installing React Router, you need to configure your app to use it. This will be done in the root of you Javascript file (main.jsx).



```
//other React imports
import { createBrowserRouter, RouterProvider } from 'react-router-dom';

const paths = createBrowserRouter([
  {
    path: '/',
    element: <h1>Hello World</h1>
  }
])


const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <RouterProvider router={paths} /> {/** replaced <App/> */}
  </React.StrictMode>
);
```

CoGrammar

Configuration

❖ From the configuration example shown, we made two important imports:

1. **createBrowserRouter**: this configures Browser Router which enables client side Routing in our React application.
   - ➢ It is a function that takes in a list of available paths in our application, the paths will be defined by objects.
   - ➢ Currently, we've only created one path which is the home path using a '/' and it renders a <h1> text saying Hello World.

CoGrammar

Configuration

2. **RouterProvider**: All path objects created by the createBrowserRouter API are passed to the provider component as a value of the router prop to render your app and enable routing.

❖ After this configuration, upon running your React server, you will have a text displaying Hello World on the home page.

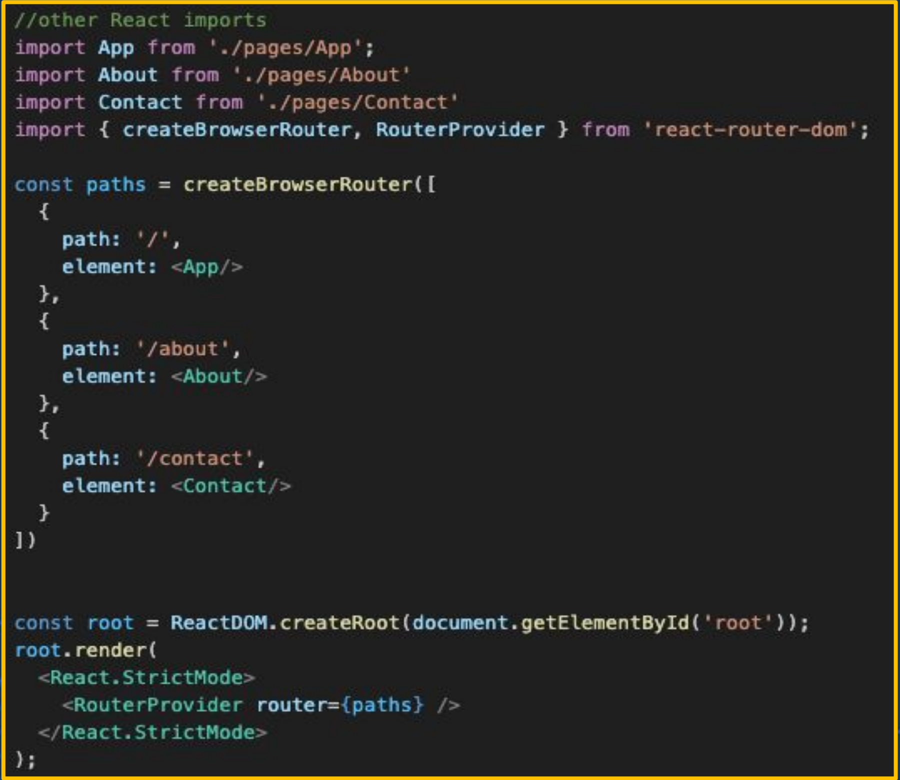
CoGrammar

Multiple Pages

- ❖ Having multiple pages in our React app is one of the main achievements of routing.
- ❖ We do this by creating **other path objects** and **pointing** the path elements to their specific components.
- ❖ The **element property** of the path object will be replaced by a **React component from your project**.
- ❖ In this case, we have three components representing three pages and all are stored in a folder called pages for best practice purpose.

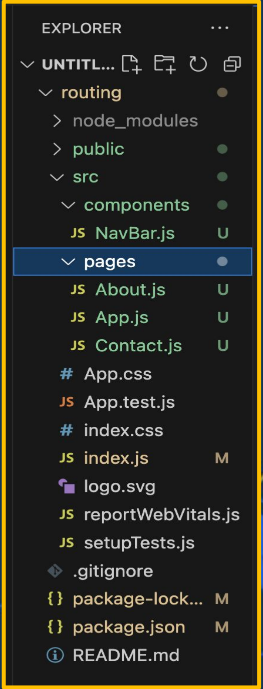
CoGrammar

Multiple Pages



```
//other React imports
import App from './pages/App';
import About from './pages/About'
import Contact from './pages/Contact'
import { createBrowserRouter, RouterProvider } from 'react-router-dom';

const paths = createBrowserRouter([
  {
    path: '/',
    element: <App/>
  },
  {
    path: '/about',
    element: <About/>
  },
  {
    path: '/contact',
    element: <Contact/>
  }
])


const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <RouterProvider router={paths} />
  </React.StrictMode>
);
```

Folder structure:



```
EXPLORER ...
∨ UNTITL...
  ∨ routing
    > node_modules
    > public
    ∨ src
      ∨ components
        JS NavBar.js        U
      ∨ pages
        JS About.js         U
        JS App.js           U
        JS Contact.js       U
      # App.css
      JS App.test.js
      # index.css
      JS index.js           M
      logo.svg
      JS reportWebVitals.js
      JS setupTests.js
    .gitignore
    {} package-lock...      M
    {} package.json         M
    ⓘ README.md
```

CoGrammar

Navigating through React Router pages

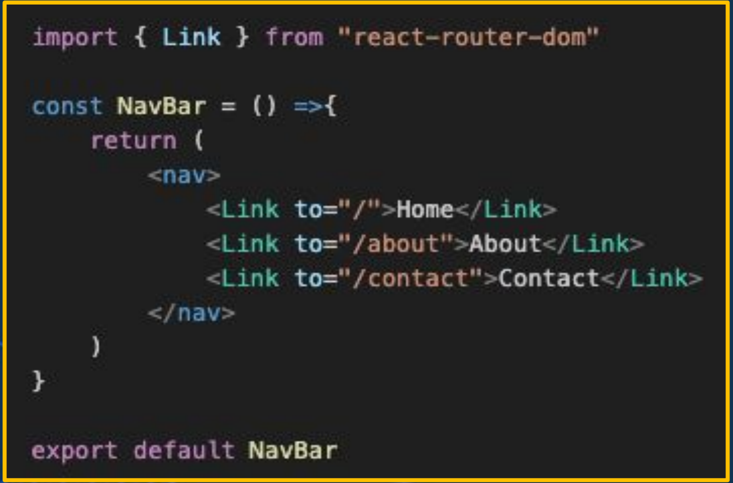
- For hyperlinks, we are used to utilizing the <a> tag in HTML. Using <a href=''> causes a page refresh which can lead to losing an application's state.
- To achieve complete client side routing with React Router, we use its <Link> element to navigate from page to page. Instead of the {href='/path'} attribute in <a> tags, the link element provides a { to='/path'} property to direct the link to the desired URL path.
- The <Link> element does not cause a page refresh hence the application's state cannot be lost.

CoGrammar

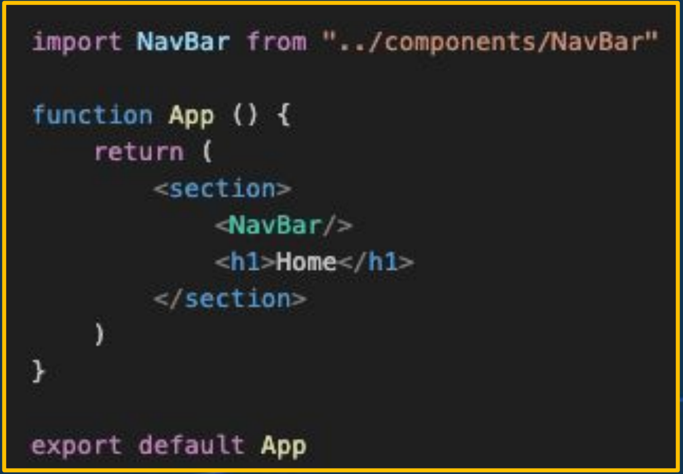
Example

Note that the structure of the App component is also implemented on the About and Contact component

- ❖ The { Link } element is imported from ‘react-router-dom’
- ❖ You can also user { NavLink } to know whether a page is active or not.



```
import { Link } from "react-router-dom"

const NavBar = () =>{
    return (
        <nav>
            <Link to="/">Home</Link>
            <Link to="/about">About</Link>
            <Link to="/contact">Contact</Link>
        </nav>
    )
}

export default NavBar
```



```
import NavBar from "../components/NavBar"

function App () {
    return (
        <section>
            <NavBar/>
            <h1>Home</h1>
        </section>
    )
}

export default App
```

CoGrammar

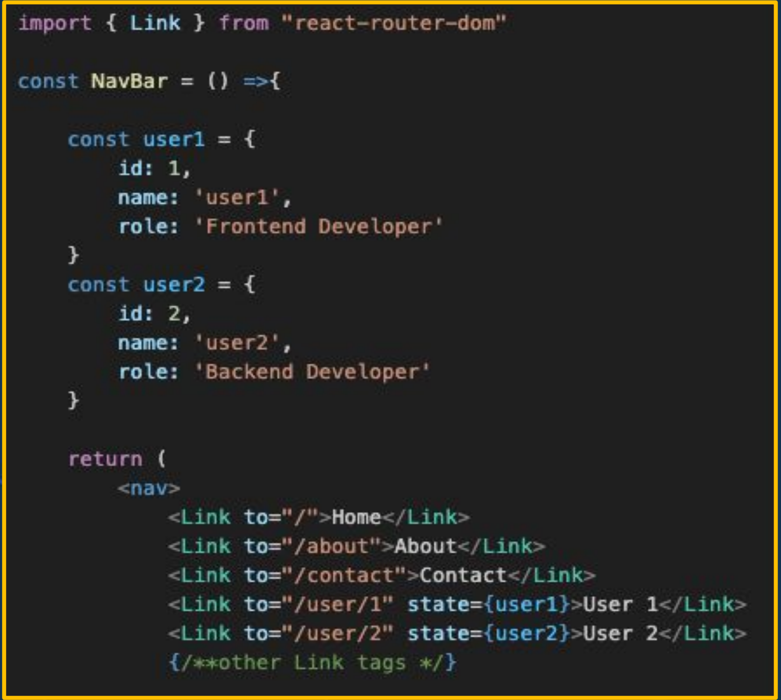
Passing State Variables

- ❖ State can be passed via the <Link> element in the same way we pass props to components. We use an extra prop called { state }.
- ❖ State can also be passed via a useNavigate hook provided by React Router which returns a function that lets you navigate programmatically.
- ❖ To access the state, we use a useLocation hook which returns a location object with the state property containing the state passed from the component.

CoGrammar

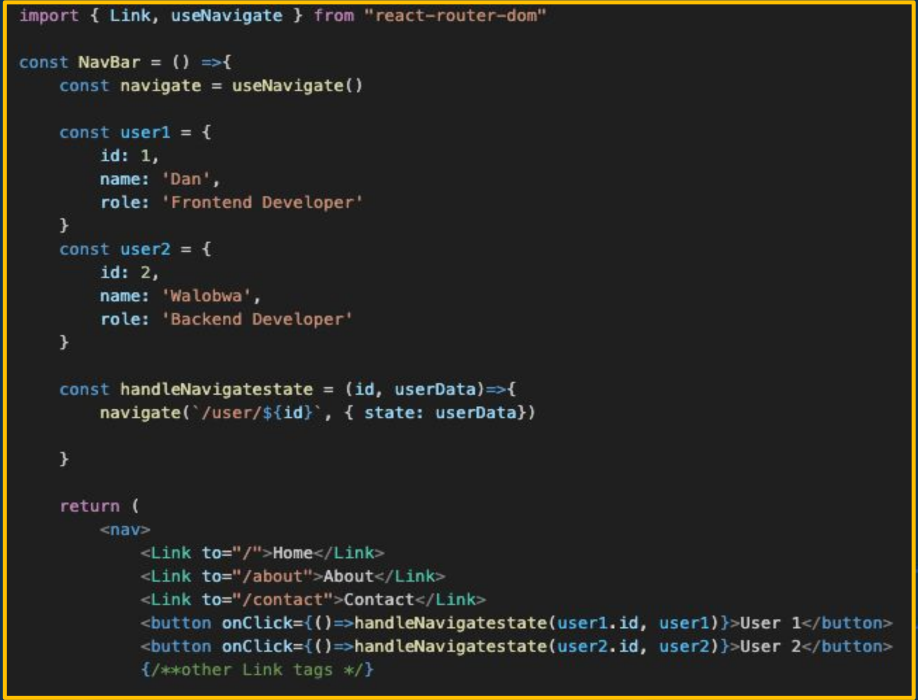
Passing State

Using <Link state={data}>



```
import { Link } from "react-router-dom"

const NavBar = () =>{

    const user1 = {
        id: 1,
        name: 'user1',
        role: 'Frontend Developer'
    }
    const user2 = {
        id: 2,
        name: 'user2',
        role: 'Backend Developer'
    }

    return (
        <nav>
            <Link to="/">Home</Link>
            <Link to="/about">About</Link>
            <Link to="/contact">Contact</Link>
            <Link to="/user/1" state={user1}>User 1</Link>
            <Link to="/user/2" state={user2}>User 2</Link>
            {/**other Link tags */}
```

Using useNavigate hook

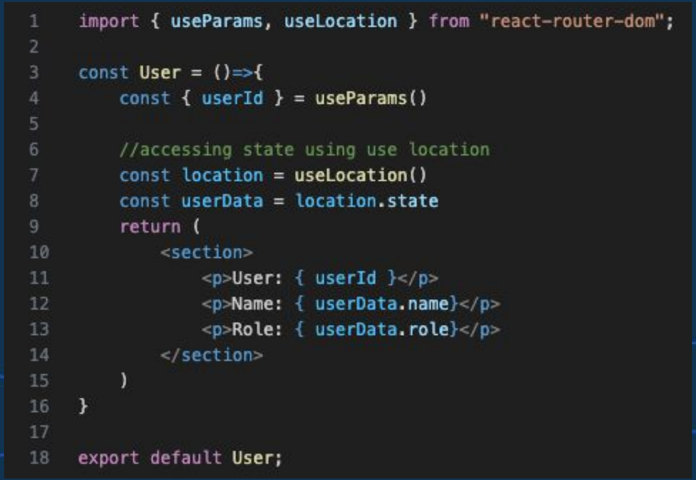


```
import { Link, useNavigate } from "react-router-dom"

const NavBar = () =>{
    const navigate = useNavigate()

    const user1 = {
        id: 1,
        name: 'Dan',
        role: 'Frontend Developer'
    }
    const user2 = {
        id: 2,
        name: 'Walobwa',
        role: 'Backend Developer'
    }

    const handleNavigatestate = (id, userData)=>{
        navigate(`/user/${id}`, { state: userData})

    }

    return (
        <nav>
            <Link to="/">Home</Link>
            <Link to="/about">About</Link>
            <Link to="/contact">Contact</Link>
            <button onClick={()=>handleNavigatestate(user1.id, user1)}>User 1</button>
            <button onClick={()=>handleNavigatestate(user2.id, user2)}>User 2</button>
            {/**other Link tags */}
```

CoGrammar

useLocation hook

- The useLocation hook is used to access the state passed from its respective dynamic path. We access state from the location object returned by the useLocation hook.
- You need to import useLocation from React Router in order to use it. This gives access to data passed from both the <Link> element and the useNavigate hook.



```
import { useParams, useLocation } from "react-router-dom";

const User = ()=>{
    const { userId } = useParams()

    //accessing state using use location
    const location = useLocation()
    const userData = location.state
    return (
        <section>
            <p>User: { userId }</p>
            <p>Name: { userData.name}</p>
            <p>Role: { userData.role}</p>
        </section>
    )
}

export default User;
```

CoGrammar

Routing Task

Welcome to the Recipe Hub, where your love for cooking meets the power of React! In this task, you'll build a personalized recipe-sharing app where users can log in, explore mouth-watering recipes, save their favorites, and learn about the app's story—all while enjoying smooth navigation and a seamless user experience. 🌟

This is your chance to create an app that combines functionality, interactivity, and mouth-watering visuals. Let your imagination run wild and make this hub a delicious treat for your users. Let's get cooking! 👩‍🍳👨‍🍳

Here’s what your app will include:

1. **Home Page:** A warm welcome! Users can log in and out with their name, and the app will greet them personally.
2. **Recipes Page:** A feast for the eyes! Display recipes as stylish cards, complete with ingredients, preparation time, and a button to save favorites.
3. **Favorites Page:** A curated collection of favorite recipes saved by the user—perfect for planning your next meal!
4. **About Page:** Share the story behind your Recipe Hub, showcase stunning food visuals, and provide contact details for budding chefs to reach out.

Which React Router hook is used to access the current location?

A. useParams
B. useNavigate
C. useLocation
D. useState

CoGrammar

What is a benefit of using hooks like useNavigate over <Route> components?

A. They remove the need for JSX
B. They simplify dynamic navigation programmatically
C. They manage state more efficiently than useState
D. They automatically style components

CoGrammar

Summary

React Router Hooks

- ★ useNavigate: Programmatic navigation between routes.
- ★ useLocation: Accessing and managing the current route.

State Management

- ★ Using useState for toggling login/logout states and managing favorites.

Dynamic Content Rendering

- ★ Creating recipe cards dynamically with .map().

Conditional Rendering

- ★ Showing or hiding components based on user interaction or application state.

Styling and Navigation

- ★ Using useLocation to highlight active routes.
- ★ Styling components with custom CSS or Bootstrap.

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