




# Welcome to the CoGrammar Lecture: React.js

**The session will start shortly...**

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



# 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](http://www.hyperiondev.com/support)
- Report a **safeguarding** incident:  
[www.hyperiondev.com/safeguardreporting](http://www.hyperiondev.com/safeguardreporting)
- We would love your **feedback** on lectures: [Feedback on Lectures](#)

# 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**

# 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

# Learning Objectives

- ❖ Understand the core concepts of React.js, a popular JavaScript library for building user interfaces.
- ❖ Explore the fundamental building blocks of React applications, known as Elements, and learn how to create and render them.
- ❖ Gain familiarity with JSX syntax, which allows mixing HTML with JavaScript for more efficient UI development.



# React Introduction

- ❖ ReactJS is a declarative, efficient, and flexible JavaScript library for building reusable UI components.
  - ❖ It is an open-source, component-based front end library responsible only for the view layer of the application.
  - ❖ A ReactJS application is made up of multiple components, each component responsible for outputting a small, reusable piece of HTML code.
  - ❖ The components are the heart of all React applications. These Components can be nested with other components to allow complex applications to be built of simple building blocks.
- 



# Rendering in React

- ❖ React renders HTML to the web page by using a function called **createRoot()** and its method **render()**.
- ❖ The **createRoot()** function takes one argument, an HTML element. The purpose of the function is to define the HTML element where a React component should be displayed.
- ❖ The **render()** method is then called to define the React component that should be rendered.





# Rendering in React

```
const rootElement = document.getElementById('root');  
const root = createRoot(rootElement);  
  
root.render(  
  <StrictMode>  
    <App />  
  </StrictMode>  
);
```

# React JSX

- ❖ JSX(JavaScript Extension), is a React extension which allows writing JavaScript code that looks like HTML.
- ❖ In other words, JSX is an HTML-like syntax used by React that extends ECMAScript so that HTML-like syntax can co-exist with JavaScript/React code.
- ❖ JSX allows you to write HTML/XML-like structures (e.g., DOM-like tree structures) in the same file where you write JavaScript code, then preprocessor will transform these expressions into actual JavaScript code.

```
<div>  
  <h1>Hello</h1>
```

# Why use JSX?

- ❖ It is faster than regular JavaScript because it performs optimisation while translating the code to JavaScript.
- ❖ Instead of separating technologies by putting markup and logic in separate files, React uses components that contain both. We will learn components in a further section.
- ❖ It is type-safe, and most of the errors can be found at compilation time.
- ❖ It makes easier to create templates.

# JSX Attributes

- ❖ JSX use attributes with the HTML elements same as regular HTML.
- ❖ JSX uses camelcase naming convention for attributes rather than standard naming convention of HTML such as a class in HTML becomes className in JSX because the class is the reserved keyword in JavaScript.

```
<div>  
  <h1 className="hello">The Final Countdown</h1>  
</div>
```

# JSX Attributes

- ❖ In JSX, we can specify attribute values in two ways:
  - As String Literals: We can specify the values of attributes in double quotes.

```
<h2 className="firstAttribute">Hello Zahir</h2>
```

- As Expressions: We can specify the values of attributes as expressions using curly braces {}

```
<h2 className={varName}>Hello Zahir</h2>
```

# JSX Comments

- ❖ JSX allows us to use comments that begin with `/*` and ends with `*/` and wrapping them in curly braces `{}` just like in the case of JSX expressions.

```
{/* This is a comment in JSX */}
```

# JSX Styling

- ❖ To set inline styles, you need to use camelCase syntax.

```
export default function App() {  
  let myStyle = {  
    fontSize: 80,  
    fontFamily: 'Courier',  
    color: '#003300',  
  };  
  
  return (  
    <div>  
      <h2 style={myStyle}>Hello Zahir</h2>  
    </div>  
  );  
}
```



# Let's Breathe!

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





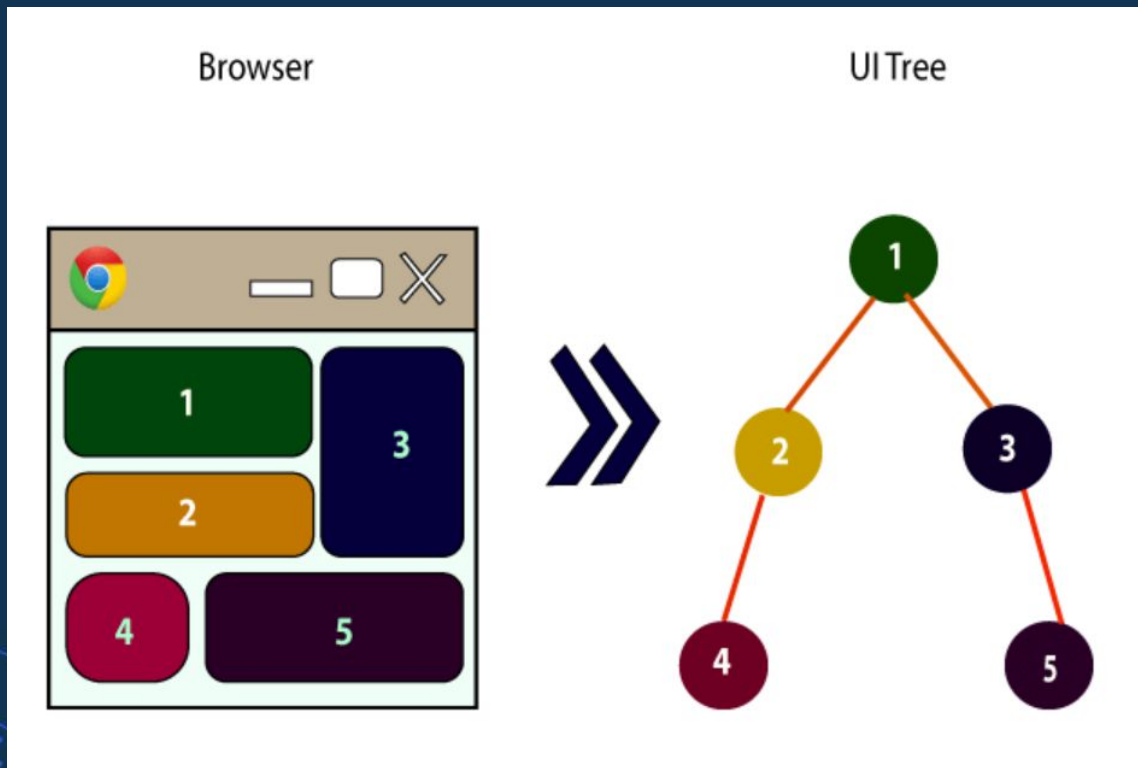


# React Components

- ❖ 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.



# React Components



# Functional Components

```
import React from 'react';  
function WelcomeMessage(props) {  
  return <h1>Welcome to the , {props.name}</h1>;  
}  
  
export default WelcomeMessage;
```

# Props

- ❖ Components can be passed props, which stands for properties.
- ❖ Props are like function arguments, and you send them into the component as attributes.

```
<h2 style={myStyle}>HELLO Zahir</h2>  
<Welcome name="Zahir"></Welcome>
```

# Questions and Answers





# Thank you for attending



Department  
for Education

CoGrammar

