



Welcome to this session: Introducing React Elements and Components

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

Skills Bootcamp Cloud Web 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

By the end of this lesson, learners should be able to:

- **Set up and initialise a React project** using Vite as the build tool and NPM for dependency management.
- **Implement and use JSX** to create dynamic, declarative components in React.
- **Create and style reusable React components** that display structured information.
- **Implement and display data** from JavaScript objects and arrays within JSX.




Who initially created React?

- A. Facebook (Now Meta).
- B. Jordan Walke.
- C. Mark Zuckerberg.
- D. The React Team at Facebook.



Why would we prefer to use Vite over Create React App (CRA) for React development? (Select all that apply)

- A. Faster Build and Hot Module Replacement (HMR).
 - B. Better Performance with ES Modules.
 - C. Out-of-the-box Support for Modern JavaScript Features.
 - D. Smaller Bundle Sizes.
 - E. Simpler and More Configurable.
 - F. Built-in Support for Frameworks and Libraries.
 - G. More Modern Development Experience.
- 

Lecture Overview

- Introduction to React
- Introduction to JSX
- React Elements
- React Components

Introduction to React

- ❖ 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 code**.
- ❖ The components are the heart of all React applications. These components can be **nested within other components** to allow complex applications to be built of simple building blocks.

Common React terms

- ❖ JSX
 - Stands for JavaScript XML. JSX is a JavaScript extension syntax used in React to easily write HTML and JavaScript together.
- ❖ React elements
 - The building blocks of React applications.
- ❖ React components
 - Small, reusable pieces of code that return a React element to be rendered to the page.

Common React terms

- ❖ Props
 - Inputs to a React component. They are data passed down from a parent component to a child component.
- ❖ State
 - An object that holds data and information related to a React component. It can be used to store, manage, and update data within the application.
- ❖ Hooks
 - These let you use different React features from your components.

DOM for an HTML Table

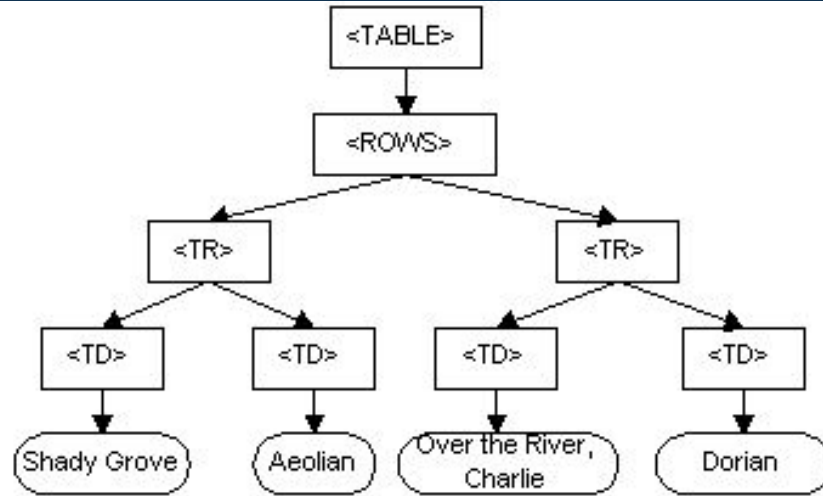


Table (Robie, J., Texcel Research, 1998).

<https://www.w3.org/TR/WD-DOM/introduction.html>

Virtual DOM

- ❖ An efficient in-memory representation of the actual HTML document.
- ❖ When changes occur in the application, React updates the virtual DOM first.
- ❖ It then compares the updated virtual DOM with its previous version to identify what has changed.
 - This comparison process, known as "reconciliation", allows React to determine the specific changes needed in the real DOM. Instead of rewriting the entire DOM, only the parts that have changed are updated, this helps to enhance the overall performance greatly.

Why do we use React?

- ❖ Building interactive websites easily.
- ❖ Reusable pieces.
- ❖ Efficient updates.
- ❖ Large and helpful community.
- ❖ Good for big projects.
- ❖ Learn once, use everywhere.
- ❖ Job market demand.
- ❖ Backed by Meta.

Create a new React app using Vite

- ❖ `npm create vite@latest my-react-app -- --template react`
 - `npm create vite@latest`
 - `npm`: This is the Node.js package manager, which is used to manage dependencies and packages for JavaScript projects.
 - `create`: The create command in this context is used to generate a new project. This command is part of a package called create-vite, which is a tool that helps you scaffold a new Vite project.
 - `vite@latest`: This specifies that you want to use the latest version of Vite to create the project. The `@latest` part ensures you get the most up-to-date version of Vite, which includes all the latest features and fixes.

Create a new React app using Vite

- ❖ `npm create vite@latest my-react-app -- --template react`
 - `my-react-app`
 - This is the name of the directory or project folder that will be created. The project folder will be named `my-react-app`, and all the necessary files for the new React application will be placed inside this folder.
 - `-- --template react`
 - The first `--`: This is used to pass additional flags or arguments to the underlying vite command.
 - `--template react`: This tells Vite to use a specific template when creating the project. In this case, the react template will be used. Vite provides different templates for various frameworks, and react is one of them. This template sets up a project configured for React development, including the necessary dependencies, configurations, and default files for a React app.

Create a new React app using Vite

- ❖ Install dependencies:
 - After setting up the project, install all the necessary dependencies using:
 - `npm install`
- ❖ Run the development server:
 - Finally, to start your development environment and preview your React app, run:
 - `npm run dev`

Rendering in React

```
import { StrictMode } from 'react'
import { createRoot } from 'react-dom/client'
import './index.css'
import App from './App.jsx'

createRoot(document.getElementById('root')).render(
  <StrictMode>
    <App />
  </StrictMode>,
)
```

Understanding React Elements

- ❖ React Elements are the smallest building blocks of a React application.
- ❖ Key concepts:
 - A React Element is a plain JavaScript object that represents the DOM tree.
 - React Elements are immutable, meaning once you create an element, it cannot be changed. React only updates the parts of the DOM that need to change.
 - The **React.createElement()** function is used to create a React Element, although, in most cases, we use JSX to simplify this.

```
// Without JSX
```

```
const element = React.createElement('h1', null, 'Hello, World!');
```

```
// With JSX
```

```
const element = <h1>Hello, World!</h1>;
```

What is JSX?

- ❖ JSX is a JavaScript syntax extension that can make creating React applications a lot quicker and easier.
- ❖ With JSX, if an attribute value is enclosed in quotes “ ” it is a string, and if the value is wrapped in curly braces { } it is an enclosed JavaScript expression.

```
import { StrictMode } from "react";
import { createRoot } from "react-dom/client";

const name = "Hyper Dave";

const element = (<h1>Welcome back, {name}</h1>);

createRoot(document.getElementById("root")).render(

  <StrictMode>

    {element}

  </StrictMode>

);
```

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 technologies**.
- ❖ It is **type-safe**, and most of the errors can be found at compilation time.
- ❖ It makes it **easier to create templates**.

JSX Attributes

- ❖ JSX assign 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 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>  
  );  
}
```

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.

Functional Components

- ❖ To write this component, we will be using JSX. JSX enables you to write HTML-like markup within a JavaScript file, keeping rendering logic and content seamlessly integrated.
- ❖ When creating React components it is extremely important that you always adhere to the following two rules:
 - **Always** start component names with a **capital letter**.
React treats components starting with lowercase letters as DOM elements.
 - A component should **never** modify its **own props**.

Functional Components

```
//Create the Welcome Component
function Welcome({name, age}) {
  return (
    <div>
      { /* This h1 element uses JSX to display the name property of the props object */ }
      <h1>Hello World, {name} you are {age} years old</h1>
    </div>
  );
}
export default Welcome;
```



What is the core purpose of a component in React?

- A. Encapsulate UI and Logic.
- B. Render HTML Content Only.
- C. Handle User Interactions.
- D. Manage Application State.
- E. Reusable Building Blocks for the UI.

Why do we primarily use JSX with React?

- A. To Write HTML-like Code in JavaScript.
- B. To Improve Code Readability and Maintainability.
- C. To Enable Conditional Rendering.
- D. To Automatically Optimize the UI for Performance.
- E. To Provide Better Error Handling and Debugging.

Questions and Answers



Thank you for attending



CoGrammar



Department
for Education