Chapter 2:
How to Create React Components

React Components

- In React, **a component** is a single independent unit of a user interface (UI).
- What you write inside a component will determine what should appear on the browser screen at a given time.
- In the previous chapter, we created an **App component** that returns a heading element:

```
function App() {
  return <h1>Hello World</h1>
}

export default App
```

...React Components

- **A component** is made up of a function that returns a single UI element.
- All React components are saved under the **.jsx** file extension.
- What is **JSX**? It's an extension of JavaScript that produces JavaScript-powered HTML elements.

…React Components

- How to Return Multiple Elements With Fragments:
- **A component** must always return a single element.
- When you need to return multiple elements, you need to wrap all of them in a single element like a

```
function App() {
  return (
    <div>
      <h1>Hello World!</h1>
      <h2>Learning to code with React</h2>
    </div>
  )
}

export default App
```

...React Components

- But this will make your application render one extra element in the browser.
- To avoid cluttering your application, you can render an empty tag <> like this:

```
function App() {
  return (
    <>
      <h1>Hello World!</h1>
      <h2>Learning to code with React</h2>
    </>
  )
}

export default App
```

...React Components

- The empty tag above is a React feature called a **Fragment**.
- By using a **Fragment**, your component won't render an extra element to the screen.

```
import { Fragment } from 'react';

function App() {
  return (
    <Fragment>
      <h1>Hello World!</h1>
      <h2>Learning to code with React</h2>
    </Fragment>
  )
}

export default App
```

...React Components

- But you don't need to explicitly state the Fragment tag. Using an **empty tag** <> is enough.
- ❖ How to **Render** to the Screen
- **React component** using the ReactDOM library, which you've seen previously when viewing the **main.jsx** file.
- You need to have **an HTML** file as the source from which your React component is rendered.
- Usually, a very basic HTML document wi
  enough, as you can see in the **index.htm**

```
<body>
  <div id="root"></div>
  <script type="module" src="/src/main.jsx"></script>
</body>
```

…React Components

- Notice how ReactDOM is imported from react-dom package,
- and the **document.getElementById('root')** is used to select the element :

```
import React from 'react'
import ReactDOM from 'react-dom/client'

function App() {
  return <h1>Hello World!</h1>
}

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

How to Write Comments in React

- Writing comments in React components is similar to how you comment in regular JavaScript code.
- You can use the double forward slash syntax **//** to comment any code.

```
function App() {
  return (
    <>
      <h1>Hello World!</h1>
      <h2>Learning to code with React</h2>
    </>
  )
}

// export default App
```

…Comments in React

- When you want to comment the code **inside the return statement**,
- you need to use the curly brackets, forward slash, and asterisk format **{/* comment here */}** as shown below:

```
function App() {
  return (
    <>
      <h1>Hello World!</h1>
      {/* <h2>Learning to code with React</h2> */}
    </>
  )
}
```

Compose Multiple Components as One

- **Composing components** is the process of forming the user interface by using loosely coupled components.

```
export default function ParentComponent() {
  return (
    <>
      <UserComponent />
      <ProfileComponent />
      <FeedComponent />
    </>
  );
}

function UserComponent() {
  return <h1> User Component </h1>;
}

function ProfileComponent() {
  return <h1> Profile Component </h1>;
}

function FeedComponent() {
  return <h1> Feed Component</h1>;
}
```

Compose Multiple Components as One

- The composition of many components will form a single tree of React components in a top-down approach.
- The tree will then be rendered into the DOM through the ReactDOM.render() method: