What Are Components in React, and How Do They Work?

freecodecamp.org/learn/front-end-development-libraries-v9/lecture-introduction-to-javascript-libraries-and-frameworks/what-are-components-in-react-and-how-do-they-work

Components are the building blocks of React applications that allow developers to break down complex user interfaces into smaller, manageable pieces, making it easier to develop and maintain large-scale applications.

The two types of components in React are functional and class-based components. In modern React, developers will use functional components and all of the examples we look at today will be functional components.

At a higher level, you can think of components like JavaScript functions that return elements describing the UI.

This UI is described using JSX, a syntax extension for JavaScript that looks similar to HTML but allows you to write UI elements in a more declarative way.

Let's look at an example of a React component:

```
function Greeting() {
  const name = "John"
  {/* The result will be Hello John*/}
  return <h1 className="title">Hello {name}</h1>;
}
```

In this example, we've defined a component called `Greeting`. The curly braces `{}` inside the `h1` tags enable us to embed JavaScript expressions, allowing us to access the `name` variable within the `h1` element.

We are also applying a `className` called `title` to the `h1` element.

But why are we using `className` instead of `class` like with regular HTML elements?

Well, this is because in JavaScript, `class` is a reserved name. So, we need to use `className` instead.

We are also using a comment in JSX showing what the result will be. It is important to note that you can use regular comment syntax like this but it needs to be wrapped in curly braces in order for it to work:

```
{/* Block Comments */}
```

Another thing you might have noticed is that we are using a capital letter at the beginning of the component name. But why can't we use all lower case letters?

This is because React treats components with a capital letter as custom components, while elements with lowercase letters are considered built-in HTML elements.

When React encounters a lowercase tag, like `<div>` or `<span>`, it assumes it's a standard HTML element. However, if the component name starts with a capital letter, React will treat it as a user-defined component and render it accordingly. This distinction helps React differentiate between native HTML tags and components that you create.

To use this `Greeting` component in our application, we would write something like this:

```
<Greeting />
```

This would render an `h1` element with the text `Hello John` to the page. But take a closer look at the syntax here. When we use the component, it ends with a forward slash and then the greater than symbol.

When working with JSX, all tags and uses of components need to be explicitly closed. So if the component or tag does not have any children, then you need to explicitly close it like shown here:

```
<Greeting /> {/* /> is required */}
```

So far we have only been looking at how to render a single `h1` element. But you can actually render multiple elements.

Let's take a look at the following example code here:

```
function Greeting() {
  const name = "John";
  {/* This will throw an error */}
  return <h1>Hello {name}</h1>
  <p>Nice to meet you.</p>
}
```

We are trying to add another sentence of `Nice to meet you` but it is not rendering on the page correctly.

There seems to be an error message instead. The error message says `Adjacent JSX elements must be wrapped in an enclosing tag.`

The reason why you are getting that error message is because multiple sibling elements need to be wrapped in a parent element. While you could wrap the `h1` and `p` elements in a simple `div`, there is another way to silence the error.

React fragments are used to group elements together. Here is what the revised example will look like:

```
function Greeting() {
  const name = "John";
  return (
    <Fragment>
      <h1>Hello {name}</h1>
      <p>Nice to meet you.</p>
    </Fragment>
  );
}
```

You can also choose to use empty JSX tags which can serve as shorthand for fragments:

```
function Greeting() {
  const name = "John";
  return (
    <>
      <h1>Hello {name}</h1>
      <p>Nice to meet you.</p>
    </>
  );
}
```

In future lessons, we will continue to learn more about how to work with components and JSX. But for now, you've gained a solid introduction to building user interfaces with components, setting a strong foundation for what's to come.

Questions

What is the primary reason React uses `className` instead of `class` for HTML elements in JSX?

`class` is a reserved keyword in JavaScript.

`className` sounds nicer.

Using `className` results in better security in React apps.

Using `className` results in faster performance.

In the context of React components, what is JSX?

A new programming language.

A database query language.

A syntax extension for JavaScript that looks similar to HTML.

A CSS framework.

How would you render a `Message` component in React?

`</ Message>`

`<</Message>`

`Message`

`<Message />`