Composing Pages Using JSX

JSX (JavaScript XML) is an extension to JavaScript that allows us to write code that closely resembles HTML without having to make it a string. This is the preferred way to use React.

*class* App *extends* *Component* {

    render() {

        return (

            <h1>Hello World</h1>

        );

    }

}

Notice how above we make a react component called App and give it a method called render(){ ... }. This method returns what appears to be an HTML <h1> tag. Whatever content we wish to make visible on the page, we can write it using a syntax we are already familiar with.

We are able to use any HTML tags that we are already familiar with, try out some like:

* table
* ul
* form
* input
* img

Some rules to keep in mind

We aren't free to put in just anything we want to however.

JSX Expressions must have one Parent Element

Consider the example code...

*class* App *extends* *Component* {

    render(){

        return (

            <h1>Hello World</h1>

            <p>This is a paragraph</p>

        );

    }

}

When we are working with JSX, we need to ensure that our code being returned exists inside of one JSX element. The above example will fail to compile with the error message...

./src/App.js

    Line 6: Parsing error: Adjacent JSX elements must be wrapped in an enclosing tag. Did you want a JSX fragment <>...</>?

We can either place the above elements inside of a single tag like <div> or if we don't want to introduce additional elements into the DOM, we are able to make use of a JSX fragment as the error message above alludes to.

*class* App *extends* *Component* {

    render(){

        return (

            <>

                <h1>Hello World</h1>

                <p>This is a paragraph</p>

            </>

        );

    }

}

JSX versions of certain HTML attributes

Consider the following snippet.

<h1 class="my-class">This is HTML</h1>

If we were to try to use this in JSX, we might come up with something like...

*class* App *extends* *Component* {

    render() {

        return (

            <h1 class="my-class">This is JSX</h1>

        );

    }

}

But notice how we have class on line 1 and on line 4. This is more than a little ambiguous, and when we're working with JSX, we will get an error in our console telling us...

Warning: Invalid DOM property `class`. Did you mean `className`?

    in h1 (at App.js:5)

    in App (at src/index.js:6)

When using the HTML class, we will need to instead use the className attribute in JSX.

*class* App *extends* *Component* {

    render() {

        return (

            <h1 className="my-class">This is JSX</h1>

        );

    }

}

Consider the example where we might have a form in HTML.

<form action="/process" method="post">

    <div class="form-group">

        <label for="email">Email:</label>

        <input type="text" id="email" name="username" class="form-control" />

    <div>

</form>

If we were to rewrite this into JSX, how should it know to handle the for on line 3 as the HTML version of for and not as a JavaScript for?

*class* App *extends* *Component* {

    render() {

        return (

            <form action="/process" method="post">

                <div class="form-group">

                    <label htmlFor="email">Email:</label>

                    <input type="text" id="email" name="username" class="form-control" />

                <div>

            </form>

        );

    }

}

<https://reactjs.org/docs/dom-elements.html>