Rendering Object Properties With JSX

#### **Learning Outcomes Addressed**

 2.Render object properties using JSX

 3.Destructure an object with ES6 syntax

 4.Use string interpolation to construct a string with object property values

**Rendering Object Properties With Class Components In JSX**

### Rendering Object Properties With JSX

Before starting, know that this activity uses class components instead of functional components.

There are key differences between class components and functional components.

* Class components use a render function, where functional components returns the JSX directly.
* Class components access props via this, where functional components access props via the components arguments, which also supports object destructuring.
* To define class components, you use the class keyword, where functional components use a JavaScript function, arrow function syntax is also supported.

**Class Component Example**

import React from 'react';  
   
class Welcome extends React.Component {  
  render() {  
    return <h1>Hello, {this.props.name}</h1>;  
  }  
}

**Functional Component Example**

import React from 'react';  
   
const Welcome = (props) => {  
  return <h1>Hello, {props.name}</h1>;  
};

Here is some documentation if you'd like to read further into the topic.

* [React Component](https://reactjs.org/docs/react-component.html)
* [Six Reasons to use React Hooks instead of classes](https://blog.bitsrc.io/6-reasons-to-use-react-hooks-instead-of-classes-7e3ee745fe04)

#### Time for the activity

JSX is a JavaScript extension that allows you to treat HTML elements as JavaScript variables in the same file. It's used in conjunction with React to keep code intuitive and efficient by making it clear what elements on your DOM are connected to variables within the logic of your application.

In this exercise you'll use JSX to do a few operations that you have already done using JavaScript and ES6. You'll render object properties on your page to display in the form of a constructed sentence that uses all of the state object properties. The steps to achieve this include updating the state, taking those values out of the state within the render function, constructing a string, and then using the template code provided to show that string in the DOM using the React.createElement function.

Here's an exercise in which you will:

1. Render object properties on your page using some basic JSX
2. Use ES6 object deconstruction syntax
3. Construct a string with object property values inside using string interpolation.

Provided is a file called reactIntro.jsx (notice the file extension is not .js like previous activities). This file uses the React functionality to show the string (or element) returned by the render function in the DOM.

#### Task instructions

First you'll need to edit the initializeState function in order to update the state to have the following values:

1. Set the initialized field to equal the Boolean value true
2. Set the productName field to equal the string "Rice Krispies"
3. Set the productDescription field to equal the string "a cereal made of popped rice"
4. Set the productPrice field to equal the string "$3.00"

In order to update the state in a way that React will know to re-trigger the render function with updated values, you'll need to use the this.setState function. You'll learn about state later. For now, it's enough to know that state is an object where changes to it can cause React to re-render content showing on the DOM (such as when a variable value changes and you want to make sure the DOM shows the new value). An example of how to use this function to update the state is shown below:

this.setState({ initialized: true });

It's important that you follow this syntax when setting the state rather than accessing it directly, such as:

this.state = newStateVariable;

This is because the setState() function updates it in a way that React knows to re-render the component.

Next you'll use ES6 syntax in the render function. Provided already in the render function is the deconstruction of this.state to define the variable initialized within render. You'll deconstruct the remaining values (productName, productDescription, and productPrice) from the state object.

Finally, using the ES6 feature of string interpolation. You'll use the values from this.state to construct the following sentence to be stored in the stringToReturn variable.

The product name is Rice Krispies, product description is a cereal made of popped rice, and product price is $3.00.

An example of string interpolation can be seen below:

const name = "Bob", time = "today";  
`Hello ${name}, how are you ${time}?

In the desired state of this activity, the page loads with a button with the text Click here to initialize your state. After clicking the button, the button will be replaced by the sentence stored in the stringToReturn variable.

Hints:

* This exercise can be completed by using the examples provided and editing them to the specifications of the instructions. Don't over-complicate it!

Task

Show the sentence in the browser exactly as it's shown in the instructions by using setState, object deconstruction, and string interpolation.

import "./styles.css";

import { useState } from "react";

export default function App() {

const [state, setState] = useState({

initialized: false,

productName: null,

productDescription: null,

productPrice: null

});

let initializeState = () => {

return state;

};

const { initialized } = state;

const str = ``;

if (state.initialized) {

return <h1>{str}</h1>; } else {

return (

<>

<button onClick={() => initializeState()}>

Click here to initialize your state{" "}

</button>

</>

);

}

}