Destructuring And Props (60:00)

#### **Learning Outcome Addressed**

 6.Use ES6 and JSX to build functionality

Follow the instructions in NextTech to complete the activity.

**React Destructuring And Props**

### ES6 Destructuring - React

React fully embraces ES6 syntax. Therefore, ES6 destructuring adds a lot of benefits to improving your code. Destructuring is a convenient way of creating new variables by extracting some values from data stored in objects or arrays.

In React, destructuring can be used to destructure function parameters or props. React uses props to send data between components. props is an object, and is commonly passed down from a parent component to a child component.

Consider the following React component that displays basic user information:

const UserInfo = props => (  
  <div>  
    <div>  
      <span>First Name: {props.firstName}</span>  
    </div>  
    <div>  
      <span>Last Name: {props.lastName}</span>  
    </div>  
    <div>  
      <span>Email: {props.email}</span>  
    </div>  
    <button onClick={props.saveData}>Save</button>  
  </div>  
);

Notice how props is being used here. To display the first name, you have to write props.firstname. This could get more complex if the props have a nested object structure for example.

Applying destructuring to this code results in the following:

const UserInfo = ({ firstName, lastName, email, saveData }) => (  
  <div>  
    <div>  
      <span>First Name: {firstName}</span>  
    </div>  
    <div>  
      <span>Last Name: {lastName}</span>  
    </div>  
    <div>  
      <span>Email: {email}</span>  
    </div>  
    <button onClick={saveData}>Save</button>  
  </div>  
);

Notice how destructuring props allows for assigning variables to individual props. Then, within the component, all that needs to be included is the variable. Note that props.firstName was replaced by firstName after destructuring.

Note that destructuring is most often used to access whole objects or functions from props, not just inner properties like in this example. You can imagine the strength of destructuring as your application scales and increases complexity. It allows for a neater and more organized code that is more readable.

For example, let's work with a variable called vehicle that is an object. vehicle has a property car which is an object with two properties make and model. We can create variables from the properties make and model by destructuring.

We can choose the name of the variables, but often we keep them the same name as the properties that we are destructuring. This example shows keeping the same name and choosing a different name. The property name of the object you are destructing is on the left side of the : and the new variable name is on the right side of the :.

// create the object  
const vehicle = {  
  car: {  
    make: "Porsche",  
    model: "911"  
  }  
};  
   
// destructuring  
const {  
  car: {   
    make: make, // property name: make, variable name: make  
    model: modelName // property name: model, variable name: modelName  
  }  
} = vehicle;  
   
console.log(make); // prints: Porsche  
console.log(modelName); // prints: 911

**In this activity, your task is to add destructuring props to an existing component that doesn't use destructuring.**

The purpose of this activity is to simplify the component code by using destructuring.

The component Person displays details about a person. These details are passed in props. The structure of the props is as follows:

props = {  
  details: {  
    name: "Maradona",  
    residence: {  
      city: "Tigre",  
      country: "Argentina"  
    }  
  },  
  profession: "soccer"  
};

Make sure you update the props to include these exact values. Otherwise, your task won't be complete.

Your code changes should apply to the Details component inside the destruct.jsx file.

Hints:

* Notice in the starting code how many layers of dot notation is required (*props.person.details.residence.country*) when accessing the data in *props*. The end state should not require dot notation.
* Utilize the technique in the example code where we destructure the *vehicle* object and create variables from the *make* and *model* properties.

Task

Add destructuring to render the Person component.

import "./styles.css";

import { useState } from "react";

function Person() {

const [state, setState] = useState({

details: { name: "", residence: { city: "", country: "" }, profession: "" }

});

return <Details person={state}></Details>;

}

export default Person;

const Details = (props) => {

*// TODO: destructure props*

return (

<div style={{ margin: "1em" }}>

<div id="name">Name: {props.person.details.name}</div>

<div id="city">City: {props.person.details.residence.city}</div>

<div id="country">Country: {props.person.details.residence.country}</div>

<div id="profession">Profession: {props.person.details.profession}</div>

</div>

);

};