How Do You Pass Props from a Parent Component to a Child Component in React?

freecodecamp.org/learn/front-end-development-libraries-v9/lecture-working-with-data-in-react/how-do-you-pass-props-from-a-parent-component-to-a-child-component-in-react

In the previous lessons, we learned how to build small components in React like this:

```
function Greeting() {
  const developerName = "Jessica";
  return <h1>Hi {developerName}!</h1>;
}
```

We can choose to nest this component inside another parent component or the root component like this:

```
function App() {
  return <Greeting />;
}

function Greeting() {
  const developerName = "Jessica";
  return <h1>Hi {developerName}!</h1>;
}
```

While this code will run fine and display the result of `Hi Jessica!` on the screen, it is not that flexible of a component.

What if we wanted to display a different name like Naomi, Tom, or Oliver? This is where React props comes in. Props, which is short for properties, is the way for parent components to pass data down to the child component. Props can be of any type: strings, numbers, booleans, objects, or arrays.

Let's update our example from earlier to now accept a `name` prop:

```
function App() {
  return <Greeting name="Jessica" />;
}

export default App;

function Greeting(props) {
  console.log(props);
  return <h1>Hi {props.name}!</h1>;
}
```

For the child component called `Greeting` we are now using `props.name` instead of hardcoding the name `"Jessica"`. We are also logging `props` to the console which is showing as an object.

Then, inside of the parent `App` component, we are passing the value to the `name` prop so it can be passed down to the child. The result will be the same on the screen like earlier, but now we have created a more flexible component.

Now we have the ability to reuse the child component several times and pass in different names each time:

```
function App() {
  return (
    <>
      <Greeting name="Naomi" />
      <Greeting name="Tom" />
      <Greeting name="Jessica" />
      <Greeting name="Oliver" />
    </>
  );
}
```

You can also choose to use object destructuring in the props to make it more readable. Here's how you could rewrite the `Greeting` component:

```
function Greeting({ name }) {
  return <h1>Hi {name}!</h1>;
}
```

This code achieves the same result but makes it clearer which props the component is expecting to receive.

Sometimes, you can have a lot of properties that you have to pass as props. Instead of passing them one by one, you can use the spread operator (`...`), after converting them to an object.

Here is an example of a new child component called `DeveloperCard`:

```
function DeveloperCard({ name, age, country }) {
  return (
    <div className="developer-card">
      <h1>Developer: {name}</h1>
      <p>Age: {age}</p>
      <p>Country: {country}</p>
    </div>
  );
}
```

This `DeveloperCard` component accepts three props: `name`, `age`, and `country`.

In the parent `App` component, we can use the spread syntax to pass all the properties from an object as individual props to the child component:

```
function App() {
  const developerObj = {
    name: "Alice",
    age: 30,
    country: "USA",
  };

  return (
    <div className="App">
      <DeveloperCard {...developerObj} />
    </div>
  );
}
```

This is particularly useful when working with arrays of objects and passing multiple sets of properties to child components. For example, you might have a list of developers where each object in the array has the same structure but represents a different person.

You will learn more about how to render lists in arrays in future lessons.

Using props in React makes your components more flexible and reusable, allowing you to build more complex UIs. However, it's important to note that props are immutable, meaning they cannot be changed once passed to a component. If you need to handle user input and modify data, you should use state instead. You'll learn more about managing state in future lessons.

Questions

In React, how do you pass a prop named `message` with the string `Hello` to a child component?

`<ChildComponent message="Hello" />`

`<ChildComponent props.message="Hello" />`

`<ChildComponent>message="Hello"</ChildComponent>`

`<ChildComponent {message: "Hello"} />`

How would you access a prop named `userName` inside a functional child component, assuming the props are passed as a `props` object?

`this.props.userName`

`props.userName`

`this.userName`

`children.userName`

What is the correct way to pass all properties of an object as individual props to a child component?

`<ChildComponent props={objectName} />`

`<ChildComponent {...objectName} />`

`<ChildComponent objectName />`

`<ChildComponent>{objectName}</ChildComponent>`