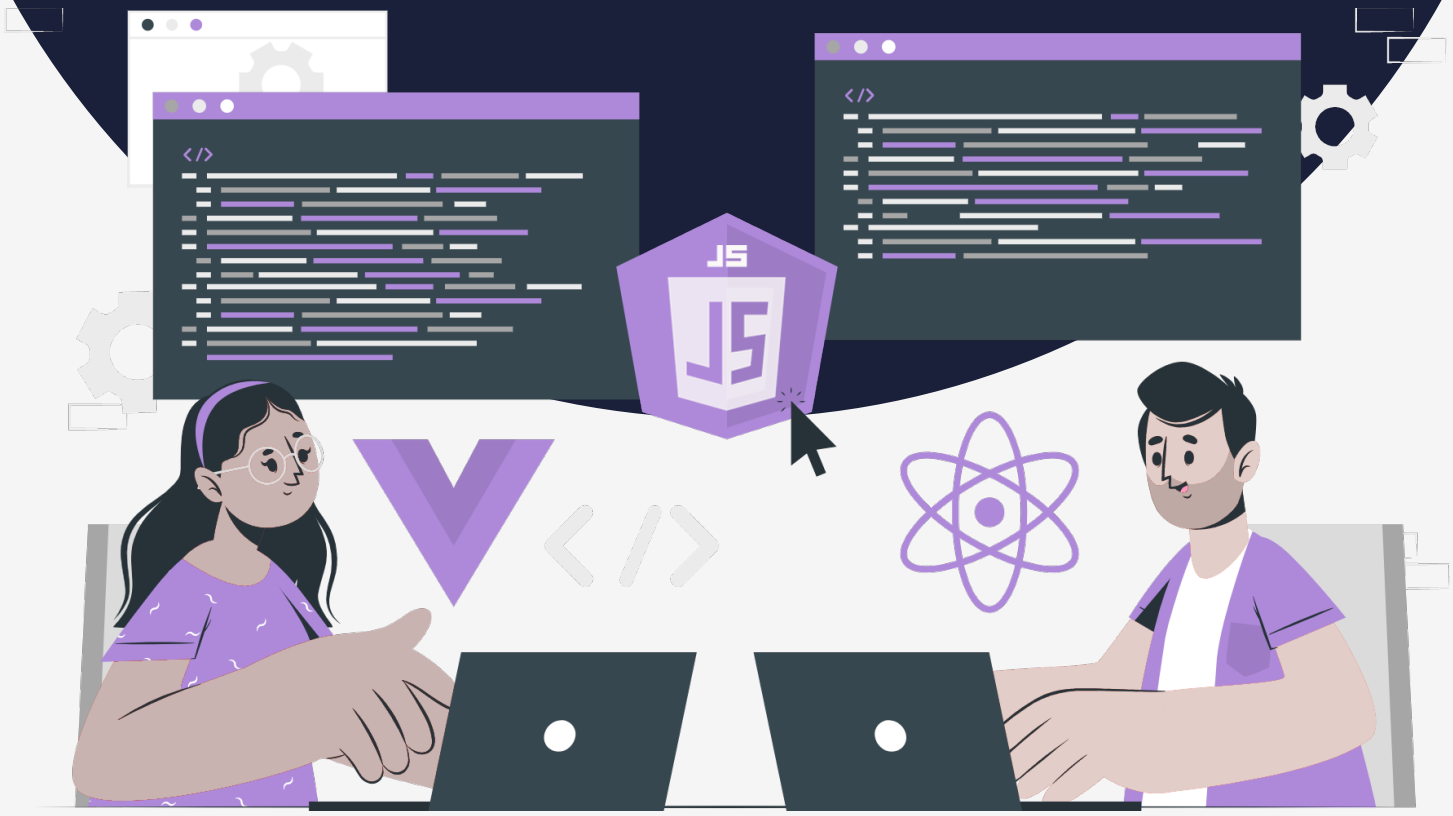
Lesson:

Conditional Rendering JSX component



PW SKILLS

Topics Covered:

1. if/else statements inside JSX components.
2. Ternary operators inside JSX components.
3. short circuit operators inside JSX components.
4. What are HTML attributes?
5. How to pass attributes to Jsx?

If a specified condition is true,then the if/else statement executes a block of code. Another block of code can be executed if the condition is false.

JavaScript

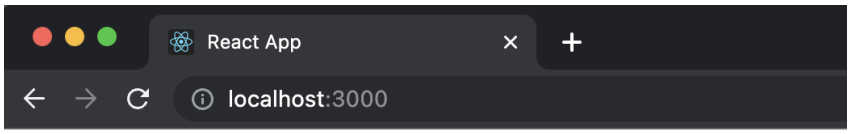
```
function MyComponent(){
// declare a variable and assign it to the value of a boolean
expression
    const isLoggedIn = true;
//declare a variable to hold the JSX element that will be
rendered

let element;

//use an if/else statement to determine which element to
render

if(isLoggedIn){

//if the user is logged in, render a message and a logout a
button

element = (
<div>
   <p> You are Logged in </p>
```

```
   <button> Logout </button>
</div>
);
} else {

//if the user is logged in, render a login form

element=(
   <form>
     <label>
        Email:
         <input type="email"/>
   </label>
   <br/>
   <label>
     Password:
      <input type="password"/>
   </label>
   <br/>
<button>Login </button>
</form>
);

}

//return the element

return element;

}

export default MyComponent;
```

In this example, the if/else statement is used to determine which element to render based on the value of the isLoggedIn variable. If the isLoggedIn variable is true, the component will render a message saying

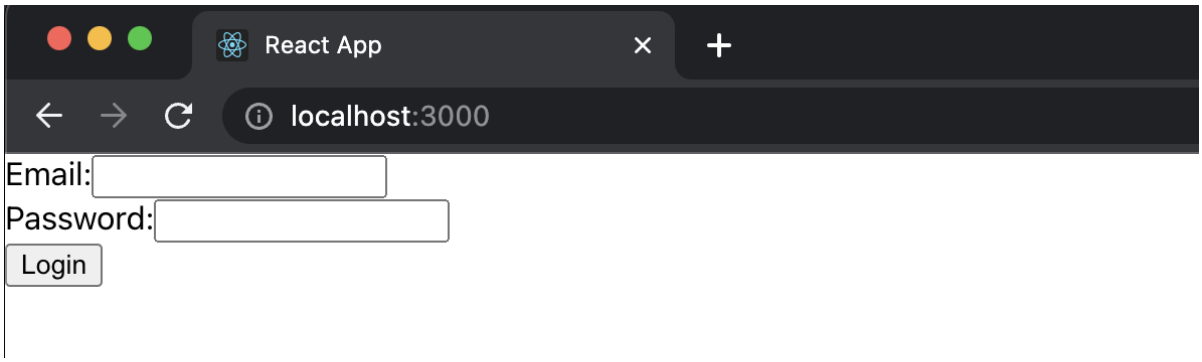
"You are logged in" and a logout button. If the isLoggedIn variable is false, the component will render a login form.

Output: If isLoggedIn is true



You are logged in

Logout



Ternary operator inside JSX element:

The ternary operator can be used within JSX components to conditionally render different elements or components based on certain conditions. The ternary operator is a shorthand version of an if/else statement and has the following syntax:

```javascript
condition ? true : false
```

Here is an example of ternary operator within a JSX component:

```javascript
function MyComponent(props) {
    return (
        <div>
            {props.isLoggedIn ? <p>Welcome back!</p> :
<p>Please log in.</p>}
        </div>
    );
}
```

In this example, the component receives a prop called "isLoggedIn" and uses the ternary operator to determine whether to show a "Welcome back!" message or a "Please log in." message. The condition is the prop isLoggedIn, if it's true, it'll render the first element after the ? sign and if it's false, it'll render the second element after the : sign.

It's worth noting that the ternary operator is more concise than an if/else statement, but it can be less readable when the conditions are complex. In that case, it's recommended to use an if/else statement instead.

Short circuit Operators :

Short circuit operators can be used in JSX to conditionally render elements. It works by evaluating the expressions on either side of the operator. If the first expression is truthy, it will be returned and the second expression will not be evaluated. If the first expression is falsy, the second expression will be evaluated and returned.

Here's an example of using the short circuit operator in JSX:

```javascript
function MyComponent({user}) {
  return (
    <div>
      {user && <p>Welcome, {user.name}</p>}
      {!user && <p>Please log in</p>}
    </div>
  );
}
```

In this example, if the user object is truthy, the first expression user && Welcome, {user.name} will be returned and the second expression will not be evaluated. If the user is falsy, the first expression will evaluate to false, the second expression Please log in will be returned instead.

What are HTML attributes?

HTML attributes are properties that are used to define the characteristics and behaviour of HTML elements.They are added to the opening tag of an HTML element and come in the form of name-value pairs, with the name being the attribute and the value being what it's set to.

Examples of commonly used attributes include “class”, “id”, “src”,”href”,”for” etc and events like “onclick”, “onchange” etc.

In HTML, we use the `class` attribute to specify a class.

```javascript
<div class="my-class"> Hello World </div>
```

But In react, We use the **`className`** attribute instead of **`class`**.

NOTE: In JavaScript class and for are reserved keywords so instead of ‘class’ we use ‘className’ and instead of ‘for’ we use ‘htmlFor’.

Let's take another example of an `onclick` event in HTML.

In HTML, we use “onclick” in all lower case but in react we use it in camel case like “onClick”. Also in HTML we have to put the function in quotes and have to invoke it by putting parentheses but in react we use curly braces and the function name without parenthesis.

Using HTML and JavaScript

```JavaScript
function MyComponent({user}) {
  return (
    <div>
      {user && <p>Welcome, {user.name}</p>}
      {!user && <p>Please log in</p>}
    </div>
  );
}
```

Using ReactJS

```JavaScript
function Welcome() {
      function submit(event){
       console.log("Form Submitted");
      }
  return <button onClick={submit}>Submit</button>;
}
```

In HTML, we use `onchange` event but in react we use in camelcase like `onChange` event.

```JavaScript
<input type="text" id="myInput" onchange="myFunction()">

<script>
  function myFunction() {
    var input = document.getElementById("myInput");
    alert("The value of the input field has changed to: " +
input.value);
  }
</script>
```

In React, we use the `onChange` event instead of `onchange`.

```JavaScript
import React, { useState } from 'react';

function MyInput() {
  const [inputValue, setInputValue] = useState('');

  const handleChange = (event) => {
    setInputValue(event.target.value);
    alert(`The value of the input field has changed to:
${inputValue}`);
  }

  return (
    <>
      <input type="text" value={inputValue}
onChange={handleChange} />
    </>
  );
}

export default MyInput;
```

In HTML, a **"label"** is an element that can be used to provide a text description for a form element, such as a text field or a checkbox. The "for" attribute of the label element is used to associate the label

with a specific form element, typically by specifying the ID of the form element.
In React, a "label" component can be created using JavaScript and JSX (JavaScript syntax extension) to create and style a label element. This component can be reusable and can be used in different parts of the application.

NOTE: As we already know there are certain differences between HTML and JSX and one such difference that we have seen is htmlFor. In normal HTML we have for but we cannot use it in JavaScript since it is a reserved keyword in JavaScript so we have to use htmlFor in JSX.

Example of HTML label:

```JavaScript
<label for="name">Name:</label>
<input type="text" id="name" name="name">
```

And use it other component

JavaScript

```
import Label from './Label';

const Form = () => (
```

```
  <form>
    <Label htmlFor="name" label="Name:" />
    <input type="text" id="name" name="name" />
  </form>
);
```