Conditional Rendering

This lesson will teach you how to add Conditional Rendering in your application by using ternary operator.

Conditional rendering is usually introduced early in React applications, though not in our working example. It happens when you decide to render either of two elements, which sometimes is a choice between rendering an element or nothing. The simplest usage of a conditional rendering can be expressed by an if-else statement in JSX.

The `result` object in the local component state is `null` in the beginning. So far, the App component returned no elements when the `result` hasn't arrived from the API. That's already a conditional rendering, because you return earlier from the `render()` lifecycle method for a certain condition. The App component either renders nothing or its elements.

But let's go one step further. It makes more sense to wrap the Table component, which is the only component that depends on `result` in an independent conditional rendering. Everything else should be displayed, even though there is no `result` yet. You can simply use a ternary operator in the JSX:

```
class App extends Component {

  ...

  render() {
    const { searchTerm, result } = this.state;
    return (
      <div className="page">
        <div className="interactions">
          <Search
            value={searchTerm}
            onChange={this.onSearchChange}
          >
            Search
          </Search>
        </div>
        { result
          ? <Table
            list={result.hits}
            pattern={searchTerm}
            onDismiss={this.onDismiss}
          />

          : null
        }
      </div>
    );
  }
}
```

That's your second option to express a conditional rendering. A third option is the logical `&&` operator. In JavaScript a `true && 'Hello World'` always evaluates to 'Hello World'. A `false && 'Hello World'` always evaluates to false.

```
const result = true && 'Hello World';
console.log(result);
// output: Hello World

const result2 = false && 'Hello World';
console.log(result2);
// output: false
```

In React, you can make use of that behavior. If the condition is true, the expression after the logical `&&` operator will be the output. If the condition is false, React ignores the expression. It is applicable in the Table component's conditional rendering case, because it should either return a Table or nothing.

```
{ result &&
  <Table
    list={result.hits}
    pattern={searchTerm}
    onDismiss={this.onDismiss}
  />
}
```

These were a few approaches to use conditional rendering in React. You can read about more alternatives in an exhaustive list of examples. Moreover, you will get to know their different uses and when to apply them.

if you are following along on a local react setup, you should be able to see the fetched data in your application by now. Everything except the Table is displayed when data fetching is pending. Once the request resolves the result and stores it into the local state, the Table is displayed because the `render()`

method runs again, and the condition in the conditional rendering resolves in favor of displaying the Table component.

Further Readings:

- Read about different ways for conditional renderings
- Read about React conditional rendering