Array Function map

One of the best features that JSX offers is the possibility of using the iterator to render elements efficiently.

Let's suppose that we have an array of elements that we want to display in the same way, we can use the map method to do so.

HTML

```
<div id="list"></div>
```

JavaScript

```
let arr = [1, 2, 3]
for (var i = 0; i < arr.length; i++) {
 append('<div>”+arr[i]+”</div>')
}
```

--⟶

JSX (ES6)

```
{
[1, 2, 3].map(currentValue => (
       <div>{currentValue}</div>
))
}
```

After compilation

```
<div>1</div>
<div>2</div>
<div>3</div>
```

The Old Way

Rendering elements with conditions is one of the most vital approaches in creating SPA (single-page applications). Luckily, React gives us the ability perfectly to do so.

Let’s take a look at the example below:

Let's say we want to display a form that says we are open every day of the week except Sunday.

JSX

```
function App() {
 if (isSunday) {
   return <p>We are closed!</p>;
 }
 // the else is implicit (since the first case has a return
 return createForm();
}
```

Note : Keep in mind that there is a better way to do this. We will see it momentarily.

Conditional operators with JSX

Let us recollect and review the logical operator in JavaScript.

JavaScript

```
// AND
true && true // true
true && false // false
false && false // false

// OR
true  || true // true
true  || false // true
false || false// false

// Booleans from other types
!0  // true
```

```
!50 // false, since 50 !== 0

!""   // true

![] // false

!"hello"  // false, since “hello” !== “”
```

Conditional operators: AND/OR

The logical operator `&&` and `||` can be extremely handy when dealing with implicit if. We will use that in JSX to simplify it. Read the code and comments below.

JSX

```
 // Ask the user for a URL
let websiteUrl = prompt("Type in your URL");


location.href = websiteUrl;
// Redirects the user to the websiteUrl
// The problem with this example is that the user can leave the field empty (which me


location.href = websiteUrl || 'http://google.com';
// Redirect the use to Google if it’s empty
// Now, if the user left the field empty, we will redirect him to Google. Otherwise h
// redirected to websiteUrl


// Or we can use the AND operator


websiteUrl && (location.href = websiteUrl);


// If the websiteUrl is not empty, redirect to it. Otherwise
// do nothing
```

Conditional operators: If/else with JSX

The logical operator `&&` and `||` are very useful when it comes to simple `if` statement, but when it comes to an `if else` statement, it’s better to use the ternary operator.

```
// Ask the user for a URL

let websiteUrl = prompt("Type in your URL");


// Check if http exists in the url, add it otherwise

let protocolUsed = (websiteUrl.startsWith("https")) ? "https" : "http";


// Another example

(protocolUsed ==="https") ? alert("You are secure") : alert("You are not secure");
```

Conditional operators: Logical expressions

Let's go back to JSX so we can apply **logical operators** in a real-life scenario.

The form in this example only shows up when the **firstName** is empty.

```
const firstName = prompt("type your first name");


 function App() {
   return (
     <div>
       <p> Hello {firstName || "Anonymous"} </p>
       <p> It looks like you {firstName ? "have" : "don't have"} a name</p>
       {!firstName && (
         <form>
           <p> Type your name here </p>
           <input type="text" />
         </form>
       )}
     </div>
   );
 }
```

‹ Previous next ›