For Loop in ReactJS

Day 57/100 08th October 2020

As with most things in web development, there are multiple ways to loop, or iterate, through an array in React using JavaScript.

Some of the iterators we have at our disposal in JavaScript are:

- ForEach
- For-of
- Map (ES6)

Out of the three iterators above, our best option to iterate over an array in React inside of JSX is the **Map** function.

Using The Map Function in React.

Introduced in ES6, the Map array function is by far my most used method of iterating over an array in React because it's incredibly versatile, succinct, and easy to read once you wrap your head around it.

Example:

App.js

```
const names = ['John', 'Paul', 'Grace', 'Maye', 'Melvin'];
    function App() {
        return (
            <div> {names.map(name => (
                        <li>
                            {name}
                        </li>
                    ))}
            </div>
        );
    }
```

In the above code we began by initializing an array called names and populate it with five strings. You might recognize *some* of the names in this array.

```
const names = ['John', 'Paul', 'Grace', 'Maye', 'Melvin'];
```

Then, we define a functional React component named App.

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

You may already be aware of this, since the release React 16.8 &hooks, Class components are no longer necessary. We can add state to functional components.

The functional React component returns a single div element with some JSX inside. As we said in JSX,(DAY) tutorials JSX is absolutely brilliant. It’s like magic and allows us to write JavaScript inside HTML.

Lastly, we simply looped through the names array using the brilliantly simple Map Array function. Again, because we’re using JSX, we can have our JavaScript Map function output HTML.

```
{names.map(name => (
    <li>
        {name}
     </li>
))}
```

To ensure that each HTML element in the React DOM has a unique identifier, we are required to provide each rendered element witha unique key. We will do this by providing a *key* attribute on the HTML element that you are rendering inside of the Map function.

In our our example above, we rendered a set of li tags in our Map function. Therefore, we have to add an additional attribute to the li tag.

Example:

```
<div>
     {names.map((name, index) => (
           <li key={index}>
               {name}
          </li>
     ))}
 </div>
```

Note that, we used the index of the array element for the unique key, this isn't the best way of providing a unique key because the index can change if you add or remove elements from the array during runtime. I would personally recommend using a third party like UUID library to generate unique Ids, and use those as a way of uniquely identifying the elements in the DOM.

Remember to read and document what you have learnt

Best wishes and explore more. Lux Academy.