React Lists

Lists are used to display data in an ordered format and mainly used to display menus on websites. In React, Lists can be created in a similar way as we create lists in JavaScript. Let us see how we transform Lists in regular JavaScript.

The map() function is used for traversing the lists. In the below example, the map() function takes an array of numbers and multiply their values with 5. We assign the new array returned by map() to the variable multiplyNums and log it.

Example

```
var numbers = [1, 2, 3, 4, 5];
const multiplyNums = numbers.map((number)=>{
    return (number * 5);
});
console.log(multiplyNums);
```

Output

The above JavaScript code will log the output on the console. The output of the code is given below.

```
[5, 10, 15, 20, 25]
```

Now, let us see how we create a list in React. To do this, we will use the map() function for traversing the list element, and for updates, we enclosed them between **curly braces {}**. Finally, we assign the array elements to listItems. Now, include this new list inside ** ** elements and render it to the DOM.

Example

```
import React from 'react';
import ReactDOM from 'react-dom';

const myList = ['Peter', 'Sachin', 'Kevin', 'Dhoni', 'Alisa'];
const listItems = myList.map((myList)=>{
    return <li>{myList}</li>;
});
ReactDOM.render(
    <ul> {listItems} </ul>,
```

```
    document.getElementById('app')
);
export default App;
```

Output

Rendering Lists inside components

In the previous example, we had directly rendered the list to the DOM. But it is not a good practice to render lists in React. In React, we had already seen that everything is built as individual components. Hence, we would need to render lists inside a component. We can understand it in the following code.

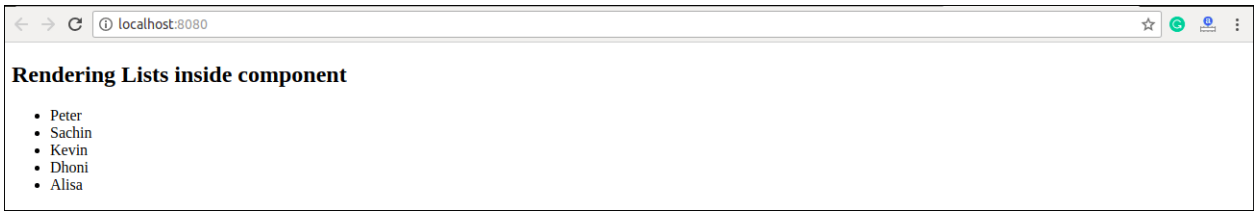
Example

```
import React from 'react';
import ReactDOM from 'react-dom';

function NameList(props) {
  const myLists = props.myLists;
  const listItems = myLists.map((myList) =>
    <li>{myList}</li>
  );
  return (
    <div>
        <h2>Rendering Lists inside component</h2>
            <ul>{listItems}</ul>
    </div>
  );
}
const myLists = ['Peter', 'Sachin', 'Kevin', 'Dhoni', 'Alisa'];
ReactDOM.render(
  <NameList myLists={myLists} />,
  document.getElementById('app')
);
```

```
21. export default App;
```

Output



React Keys

A key is a unique identifier. In React, it is used to identify which items have changed, updated, or deleted from the Lists. It is useful when we dynamically created components or when the users alter the lists. It also helps to determine which components in a collection needs to be re-rendered instead of re-rendering the entire set of components every time.

Keys should be given inside the array to give the elements a stable identity. The best way to pick a key as a string that uniquely identifies the items in the list. It can be understood with the below example.

Example

```
1. const stringLists = [ 'Peter', 'Sachin', 'Kevin', 'Dhoni', 'Alisa' ];
2.
3. const updatedLists = stringLists.map((strList)=>{
4.     <li key={strList.id}> {strList} </li>;
5. });
```

If there are no stable IDs for rendered items, you can assign the item **index** as a key to the lists. It can be shown in the below example.

Example

```
1. const stringLists = [ 'Peter', 'Sachin', 'Kevin', 'Dhoni', 'Alisa' ];
2.
3. const updatedLists = stringLists.map((strList, index)=>{
```

```
4.      <li key={index}> {strList} </li>;
5.  });
```

Note: It is not recommended to use indexes for keys if the order of the item may change in future. It creates confusion for the developer and may cause issues with the component state.

Using Keys with component

Consider you have created a separate component for **ListItem** and extracting ListItem from that component. In this case, you should have to assign keys on the **<ListItem />** elements in the array, not to the **<li>** elements in the ListItem itself. To avoid mistakes, you have to keep in mind that keys only make sense in the context of the surrounding array. So, anything you are returning from map() function is recommended to be assigned a key.

Example: Incorrect Key usage

```
1.  import React from 'react';
2.  import ReactDOM from 'react-dom';
3.
4.  function ListItem(props) {
5.    const item = props.item;
6.    return (
7.      // Wrong! No need to specify the key here.
8.      <li key={item.toString()}>
9.        {item}
10.     </li>
11.   );
12. }
13. function NameList(props) {
14.   const myLists = props.myLists;
15.   const listItems = myLists.map((strLists) =>
16.     // The key should have been specified here.
17.     <ListItem item={strLists} />
18.   );
19.   return (
20.     <div>
21.         <h2>Incorrect Key Usage Example</h2>
22.             <ol>{listItems}</ol>
```

```
    </div>
  );
}
const myLists = ['Peter', 'Sachin', 'Kevin', 'Dhoni', 'Alisa'];
ReactDOM.render(
  <NameList myLists={myLists}/>,
  document.getElementById('app')
);
export default App;
```

In the given example, the list is rendered successfully. But it is not a good practice that we had not assigned a key to the map() iterator.

Output

Example: Correct Key usage

To correct the above example, we should have to assign key to the map() iterator.

```
import React from 'react';
import ReactDOM from 'react-dom';

function ListItem(props) {
  const item = props.item;
  return (
    // No need to specify the key here.
    <li> {item} </li>
  );
}
function NameList(props) {
  const myLists = props.myLists;
  const listItems = myLists.map((strLists) =>
    // The key should have been specified here.
    <ListItem key={myLists.toString()} item={strLists} />
```

```
16.   );
17.   return (
18.     <div>
19.         <h2>Correct Key Usage Example</h2>
20.             <ol>{listItems}</ol>
21.     </div>
22.   );
23. }
24. const myLists = ['Peter', 'Sachin', 'Kevin', 'Dhoni', 'Alisa'];
25. ReactDOM.render(
26.   <NameList myLists={myLists}/>,
27.   document.getElementById('app')
28. );
29. export default App;
```

Output

Uniqueness of Keys among Siblings

We had discussed that keys assignment in arrays must be unique among their **siblings**. However, it doesn't mean that the keys should be **globally** unique. We can use the same set of keys in producing two different arrays. It can be understood in the below example.

Example

```
1.  import React from 'react';
2.  import ReactDOM from 'react-dom';
3.  function MenuBlog(props) {
4.    const titlebar = (
5.      <ol>
6.        {props.data.map((show) =>
7.          <li key={show.id}>
8.            {show.title}
```

```
9.          </li>
10.       )}
11.     </ol>
12.   );
13.   const content = props.data.map((show) =>
14.     <div key={show.id}>
15.       <h3>{show.title}: {show.content}</h3>
16.     </div>
17.   );
18.   return (
19.     <div>
20.       {titlebar}
21.       <hr />
22.       {content}
23.     </div>
24.   );
25. }
26. const data = [
27.   {id: 1, title: 'First', content: 'Welcome to JavaTpoint!!'},
28.   {id: 2, title: 'Second', content: 'It is the best ReactJS Tutorial!!'},
29.   {id: 3, title: 'Third', content: 'Here, you can learn all the ReactJS topics!!'}
30. ];
31. ReactDOM.render(
32.    <MenuBlog data={data} />,
33.   document.getElementById('app')
34. );
35. export default App;
```

Output