|  |  |
| --- | --- |
| **ch22.ReactJS Keys** | **Date: 22-02-2022** |

**Topics**

React Props Validation,

# React Props Validation

# 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' ];
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' ];
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.

Skip Ad

### **Example: Incorrect Key usage**

1. **import** React from 'react';
2. **import** ReactDOM from 'react-dom';
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>
23. </div>
24. );
25. }
26. **const** myLists = ['Peter', 'Sachin', 'Kevin', 'Dhoni', 'Alisa'];
27. ReactDOM.render(
28. <NameList myLists={myLists}/>,
29. document.getElementById('app')
30. );
31. 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**

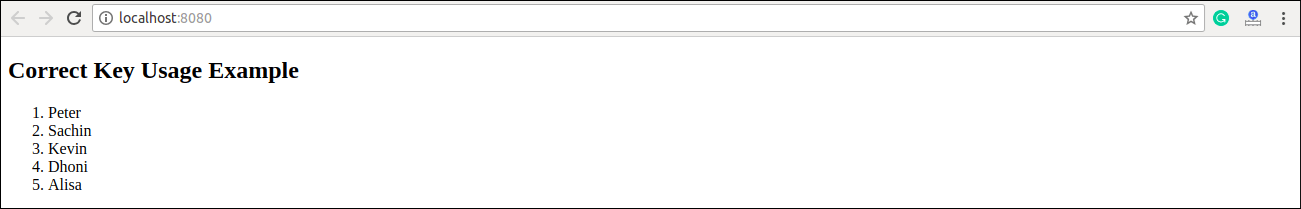


### **Example: Correct Key usage**

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

1. **import** React from 'react';
2. **import** ReactDOM from 'react-dom';
4. function ListItem(props) {
5. **const** item = props.item;
6. **return** (
7. // No need to specify the key here.
8. <li> {item} </li>
9. );
10. }
11. function NameList(props) {
12. **const** myLists = props.myLists;
13. **const** listItems = myLists.map((strLists) =>
14. // The key should have been specified here.
15. <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**



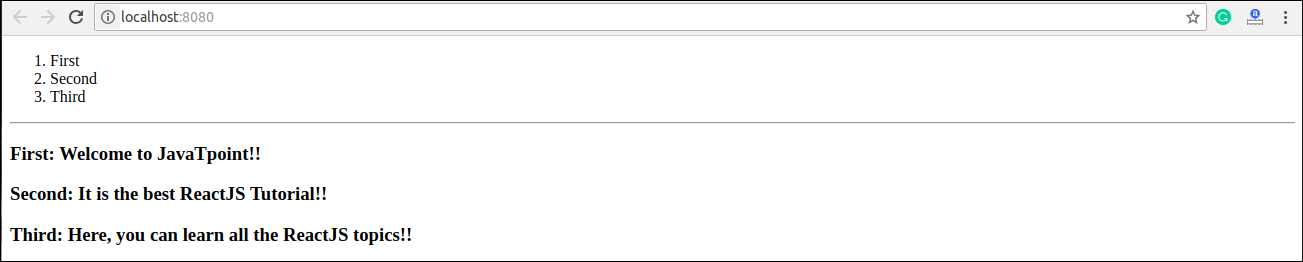
## **Uniqueness of Keys among Siblings**

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**



Next Topic[React Refs](https://www.javatpoint.com/react-refs)