React Keys

Topics : React JS
Written on January 03, 2024

In React, keys are special attributes that provide a hint to React about the identity of elements in a list. They help React identify which items have changed, been added, or been removed in a list, facilitating efficient updates and rendering.

When you render a list of elements in React, each element in the array should have a unique `key` prop. The `key` prop is used by React to keep track of the identity of each element in the list. Here's an example:

```
import React from 'react';

function MyListComponent() {
const data = ['Item 1', 'Item 2', 'Item 3'];

return (
<ul>
{data.map((item, index) => (
<li key={index}>{item}</li>
))}
</ul>
);
}

export default MyListComponent;
```

In this example, each `<li>` element has a unique key based on its index in the array. It's important to note that using the array index as a key is suitable only if the list is static and the items don't have a persistent identity. If the list is dynamic and items can be added or removed, it's generally better to use a unique identifier for each item as the key.

For example, if your data contains objects with unique IDs, you should use those IDs as keys:

```
import React from 'react';

function MyListComponent() {
const data = [
{ id: 1, text: 'Item 1' },
{ id: 2, text: 'Item 2' },
{ id: 3, text: 'Item 3' },
];
```

```
return (
<ul>
{data.map((item) => (
<li key={item.id}>{item.text}</li>
))}
</ul>
);
}
```

export default MyListComponent;

Using unique keys is important for efficient updates, especially when dealing with components that have state or when components can be reordered or dynamically added/removed. React uses keys to determine the minimum number of DOM manipulations needed when the list changes, improving performance and avoiding unnecessary re-renders of unchanged components.