React - Lists

Topics : React JS
Written on January 02, 2024

In React, rendering lists of elements is a common task, and there are several ways to achieve it. Here are some common patterns for working with lists in React:

1. Using `map()` to Render a List:

The `map()` method is commonly used to iterate over an array and render a list of elements.

```
import React from 'react';

const ListComponent = ({ items }) => (
<ul>
{items.map((item, index) => (
<li key={index}>{item}</li>
))}
</ul>
);

export default ListComponent;
```

Ensure that each item in the list has a unique `key` attribute. React uses keys to efficiently update the virtual DOM.

2. Rendering Lists of Components:

You can also render a list of React components.

```
import React from 'react';

const ItemComponent = ({ value }) => <li>{value}</li>;

const ListComponent = ({ items }) => (
<ul>
{items.map((item, index) => (
<ItemComponent key={index} value={item} />
))}
</ul>
);

export default ListComponent;
```

3. Using `map()` with Object Properties:

If you're working with an array of objects, you can use the `map()` method to render a list based on object properties.

```
import React from 'react';

const ObjectListComponent = ({ items }) => (
<ul>
{items.map((item) => (
<li key={item.id}>{item.name}</li>
))}
</ul>
);

export default ObjectListComponent;
```

4. Conditional Rendering in Lists:

You can combine the use of `map()` with conditional rendering.

```
import React from 'react';

const ConditionalListComponent = ({ items }) => (
<ul>
{items.map((item) => (
<li key={item.id}>
{item.completed ? <span style={{ textDecoration: 'line-through' }}>{item.name}</span> :
item.name}
</li>
))}
</ul>
);

export default ConditionalListComponent;
```

5. Keys and Stable IDs:

When rendering a list with React, each item should have a unique and stable identifier (`key`). Using the item index as the key is acceptable for simple lists, but for more dynamic lists with frequent changes, it's better to use a unique identifier provided by your data.

6. Lists with Forms:

When rendering a list of form elements, you may need to handle input changes and update the state accordingly. Ensure that you use unique keys for each form element.

```
import React, { useState } from 'react';

const FormListComponent = ({ items }) => {
const [formData, setFormData] = useState(items);

const handleInputChange = (index, newValue) => {
```

```
const updatedData = [...formData];
updatedData[index] = newValue;
setFormData(updatedData);
};

return (
<ul>
{formData.map((item, index) => (
<li key={index}>
<input
type="text"
value={item}
onChange={(e) => handleInputChange(index, e.target.value)}
/>
</li>
))}
</ul>
);
};

export default FormListComponent;
```