In React, rendering lists is a common task, and there are a few approaches you can take. Here's a basic example of how you can render a list of items using React:

```jsx
import React from 'react';

class ListExample extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      items: ['Apple', 'Banana', 'Orange', 'Grapes']
    };
  }

  render() {
    return (
      <div>
        <h1>Fruits List</h1>
        <ul>
          {this.state.items.map((item, index) => (
            <li key={index}>{item}</li>
          ))}
        </ul>
      </div>
    );
  }
}

export default ListExample;
```

In this example:

- We have a component called `ListExample`.
- In the constructor, we initialize the component state with an array of fruits.
- In the `render()` method, we use the `map()` function to iterate over the array of fruits (`this.state.items`).
- For each item in the array, we create a `<li>` element with the item's text content and a unique `key` prop (which is recommended for performance reasons when rendering lists in React).

This is a simple example of rendering a static list. If your list is dynamic and may change over time (e.g., due to user interactions or data fetching), you might want to update the state accordingly, triggering a re-render of the component with the updated list.

Sure, here are five examples of rendering lists in React:

Example 1: Rendering a static list of strings

```jsx
import React from 'react';

const ListExample = () => {
  const items = ['Apple', 'Banana', 'Orange', 'Grapes'];

  return (
    <div>
      <h1>Fruits List</h1>
      <ul>
        {items.map((item, index) => (
          <li key={index}>{item}</li>
        ))}
      </ul>
    </div>
  );
};

export default ListExample;
```

Example 2: Rendering a list of objects with unique keys

```jsx
import React from 'react';

const ListExample = () => {
  const items = [
    { id: 1, name: 'Apple' },
    { id: 2, name: 'Banana' },
    { id: 3, name: 'Orange' },
    { id: 4, name: 'Grapes' }
  ];

  return (
    <div>
      <h1>Fruits List</h1>
      <ul>
        {items.map(item => (
          <li key={item.id}>{item.name}</li>
```

```
      ))}
    </ul>
  </div>
 );
};

export default ListExample;
```

Example 3: Rendering a list of components

```jsx
import React from 'react';
import ListItem from './ListItem'; // Assume ListItem is another component

const ListExample = () => {
 const items = ['Apple', 'Banana', 'Orange', 'Grapes'];

 return (
  <div>
   <h1>Fruits List</h1>
   <ul>
    {items.map((item, index) => (
     <ListItem key={index} text={item} />
    ))}
   </ul>
  </div>
 );
};

export default ListExample;
```

Example 4: Conditional rendering within a list

```jsx
import React from 'react';

const ListExample = () => {
 const items = [
  { id: 1, name: 'Apple', isAvailable: true },
  { id: 2, name: 'Banana', isAvailable: false },
  { id: 3, name: 'Orange', isAvailable: true },
  { id: 4, name: 'Grapes', isAvailable: false }
```

```
  ];

  return (
    <div>
      <h1>Fruits List</h1>
      <ul>
        {items.map(item => (
          <li key={item.id}>
            {item.name} - {item.isAvailable ? 'Available' : 'Not Available'}
          </li>
        ))}
      </ul>
    </div>
  );
};

export default ListExample;
```

Example 5: Handling events within a list

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

const ListExample = () => {
  const [selectedItem, setSelectedItem] = useState(null);
  const items = ['Apple', 'Banana', 'Orange', 'Grapes'];

  const handleClick = (item) => {
    setSelectedItem(item);
  };

  return (
    <div>
      <h1>Fruits List</h1>
      <ul>
        {items.map((item, index) => (
          <li key={index} onClick={() => handleClick(item)}>
            {item} {selectedItem === item && <span>(Selected)</span>}
          </li>
        ))}
      </ul>
    </div>
  );
```

```
};

export default ListExample;
```

These examples demonstrate various scenarios of rendering lists in React, including rendering static lists, lists of objects, lists of components, conditional rendering within lists, and handling events within lists.