In React, the logical AND (`&&`) operator is commonly used for conditional rendering. It allows you to conditionally render components or content based on a certain condition. Here's how you can use it:

```jsx
import React from 'react';

function MyComponent({ isLoggedIn }) {
  return (
    <div>
      {/* Render "Welcome!" only if isLoggedIn is true */}
      {isLoggedIn && <p>Welcome!</p>}
      {/* Render "Please log in" only if isLoggedIn is false */}
      {!isLoggedIn && <p>Please log in</p>}
    </div>
  );
}

export default MyComponent;
```

In the above example, the `&&` operator is used to conditionally render the `<p>` element based on the value of `isLoggedIn`. If `isLoggedIn` is `true`, the first `<p>` element is rendered (`"Welcome!"`). If `isLoggedIn` is `false`, the second `<p>` element is rendered (`"Please log in"`).

This is a common pattern in React for conditional rendering, especially when you want to render certain elements based on the state or props of a component. The `&&` operator ensures that the right-hand side is only evaluated if the left-hand side is `true`, which helps in avoiding unnecessary rendering.

Sure, here are three examples of using the logical AND (`&&`) operator in React:

Example 1: Conditional Rendering

```jsx
import React from 'react';

function Greeting({ isLoggedIn }) {
  return (
    <div>
      {/* Render a greeting message only if isLoggedIn is true */}
      {isLoggedIn && <p>Welcome back!</p>}
    </div>
  );
}

export default Greeting;
```

In this example, the `<p>` element with the message "Welcome back!" will only be rendered if `isLoggedIn` is `true`.

Example 2: Rendering Lists Conditionally

```jsx
import React from 'react';

function List({ items }) {
  return (
    <ul>
      {/* Render list items only if items array is not empty */}
      {items.length > 0 && items.map((item, index) => (
        <li key={index}>{item}</li>
      ))}
    </ul>
  );
}

export default List;
```

Here, the list items are rendered only if the `items` array is not empty. This prevents rendering an empty list if `items` is an empty array.

Example 3: Conditional Styling

```jsx
import React from 'react';

function Button({ isPrimary }) {
  return (
    <button style={{ backgroundColor: isPrimary && 'blue', color: isPrimary && 'white' }}>
      {/* Render different text based on isPrimary */}
      {isPrimary ? 'Primary Button' : 'Regular Button'}
    </button>
  );
}
```

```
export default Button;
```

In this example, the button's background color will be blue and text color will be white if `isPrimary` is `true`, indicating a primary button. Otherwise, the button will have default styling.