Conditional Rendering

Showing different UI based on a condition with if, a ternary (? :), and the && operator, and returning different components based on a condition -- with simple examples.

Beginner 9 min

Last updated: August 18, 2026
Read online: https://www.learnforgex.com/en/topics/conditional-rendering

---

Conditional Rendering

In JSX, we briefly saw that curly braces `{ }` can only hold an expression that produces a value, so we used a ternary instead of `if`. This lesson covers showing different UI based on a condition ("conditional rendering") in full, together with state.

Conditional Rendering with if

`if` can't be written directly inside JSX's `{ }` -- but you can use it BEFORE `return`, to decide the value of a regular JavaScript variable:

```
function IfConditionalExample({ isLoggedIn }) {
  // if does NOT PRODUCE A VALUE, so it can't be written directly inside
  // JSX's { }. Instead, we decide on a plain variable BEFORE the return.
  let message;

  if (isLoggedIn) {
    message = "Welcome!";
  } else {
    message = "Please log in.";
  }

  return <p>{message}</p>;
}
```

Here, `if` decides which text gets assigned to the `message` variable; the JSX itself is still just a single `{message}` expression.

Conditional Rendering with a Ternary (? :)

A ternary produces a value, so it can be written directly inside `{ }` -- it collapses the several lines from `IfConditionalExample` into one:

```
function TernaryConditionalExample({ isLoggedIn }) {
  // The ternary (? :) PRODUCES A VALUE, so it can be written directly
  // inside { }. This condenses the four lines from IfConditionalExample
  // into a single line.
  return <p>{isLoggedIn ? "Welcome!" : "Please log in."}</p>;
}
```

When choosing between two options (this OR that), a ternary is shorter than `if` and feels more natural inside JSX.

Conditional Rendering with the && Operator

Sometimes you're not choosing between two options, but between "show something" and "show nothing at all." That's what `&&` is for:

```
function AndOperatorConditionalExample({ hasNewMessage }) {
  // The && operator: renders the right side if the left side is "truthy",
  // and renders nothing if it's "falsy" (false, 0, "", null, undefined).
  // Useful when there's no "else" branch like in a ternary -- i.e. for
  // "either show this or show nothing" situations.
  return (
    <div>
      <p>Inbox</p>
      {hasNewMessage && <p>You have a new message!</p>}
    </div>
  );
}
```

If the left side of `&&` is truthy, the right side gets rendered; if it's falsy (`false`, `0`, `""`, `null`, `undefined`), React renders nothing. **A common mistake:** if the left side is a number that could be `0`, then `0 && <p>...</p>` actually renders the text `0` on screen -- because `0` is falsy, but it's still a renderable value inside JSX. To avoid this, explicitly convert the number to a boolean with a comparison like `hasNewMessage > 0`.

Conditional Components

Sometimes what changes based on a condition isn't a single piece of text, but an entirely different component:

```
function LoadingMessage() {
  return <p>Loading...</p>;
}

function WelcomeMessage({ name }) {
  return <p>Welcome, {name}!</p>;
}

function ConditionalComponentExample({ isLoading, name }) {
  // Instead of just a single piece of text, you can return an entirely
  // DIFFERENT component depending on the condition -- both are meaningful
  // components in their own right.
  if (isLoading) {
    return <LoadingMessage />;
  }

  return <WelcomeMessage name={name} />;
}
```

`ConditionalComponentExample` returns either `<LoadingMessage />` or `<WelcomeMessage />` depending on the condition -- both are meaningful, independent components on their own. This is the "build bigger UIs out of small components" idea from Component Composition, combined with a condition.

Summary and Glossary

Conditional rendering is how you show different content based on a condition. `if` is used outside JSX (before `return`) to decide a variable's value; a ternary is used inside JSX between two options, and `&&` for "show it or show nothing." You can also return entirely different components based on a condition.

Glossary

Conditional Rendering — Rendering different content (text, an element, a component) based on a condition.

Truthy / Falsy — How a value behaves in a logical context like `if` / `&&` in JavaScript; `false`, `0`, `""`, `null`, and `undefined` are falsy, everything else is truthy.

`&&` **Operator** — A JavaScript operator that returns its right side if the left side is truthy, and renders nothing otherwise.