Conditonal Rendering:

In React, conditional rendering refers to the process of delivering elements and components based on certain conditions.

There’s more than one way to use conditional rendering in React. Like with most things in programming, some things are better suited than others depending on the problem

Conditional rendering in React works the same way conditions work in JavaScript. Use JavaScript operators like if or the conditional operator to create elements representing the current state, and let React update the UI to match them.

In React, you can create distinct components that encapsulate behavior you need. Then, you can render only some of them, depending on the state of your application.

if Statement

We can use the if JavaScript operator to decide which component to render.

We'll use these two components

```
function MissedGoal() {
  return <h1>MISSED!</h1>;
}

function MadeGoal() {
  return <h1>Goal!</h1>;
}
```

example:

```
import React from 'react';
import ReactDOM from 'react-dom/client';

function MissedGoal() {
return <h1>MISSED!</h1>;
}

function MadeGoal() {
return <h1>GOAL!</h1>;
}

function Goal(props) {
const isGoal = props.isGoal;
if (isGoal) {
return <MadeGoal/>;
}
return <MissedGoal/>;
}
```

output:

MISSED!

```
import React from 'react';
import ReactDOM from 'react-dom/client';

function MissedGoal() {
return <h1>MISSED!</h1>;
}

function MadeGoal() {
return <h1>GOAL!</h1>;
}

function Goal(props) {
const isGoal = props.isGoal;
if (isGoal) {
return <MadeGoal/>;
}
return <MissedGoal/>;
}

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<Goal isGoal={true} />);
```

output

GOAL!

Logical && Operator

Another way to conditionally render a React component is by using the && operator.

```
import React from 'react';
import ReactDOM from 'react-dom/client';

function Garage(props) {
const cars = props.cars;
return (
<>
<h1>Garage</h1>
{cars.length > 0 &&
<h2>
I have {cars.length} cars in my garage.
```

```
</h2>
}
</>
);
}

const cars = ['Ford', 'BMW', 'Audi'];
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<Garage cars={cars} />);
```

output:

Garage
I have 3 cars in my garage

```
import React from 'react';
import ReactDOM from 'react-dom/client';

function Garage(props) {
const cars = props.cars;
return (
<>
<h1>Garage</h1>
{cars.length > 0 &&
<h2>
You have {cars.length} cars in your garage.
</h2>
}
</>
);
}

const cars = [];
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<Garage cars={cars} />);
```

output:

Garage

Ternary Operator

Another way to conditionally render elements is by using a ternary operator.

condition ? true : false

```
import React from 'react';
import ReactDOM from 'react-dom/client';

function MissedGoal() {
return <h1>MISSED!</h1>;
}

function MadeGoal() {
return <h1>GOAL!</h1>;
}

function Goal(props) {
const isGoal = props.isGoal;
return (
<>
{ isGoal ? <MadeGoal/> : <MissedGoal/> }
</>
);
}

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<Goal isGoal={false} />);
```

output

MISSED!