Top of Form

Search

Submit your search query

Bottom of Form

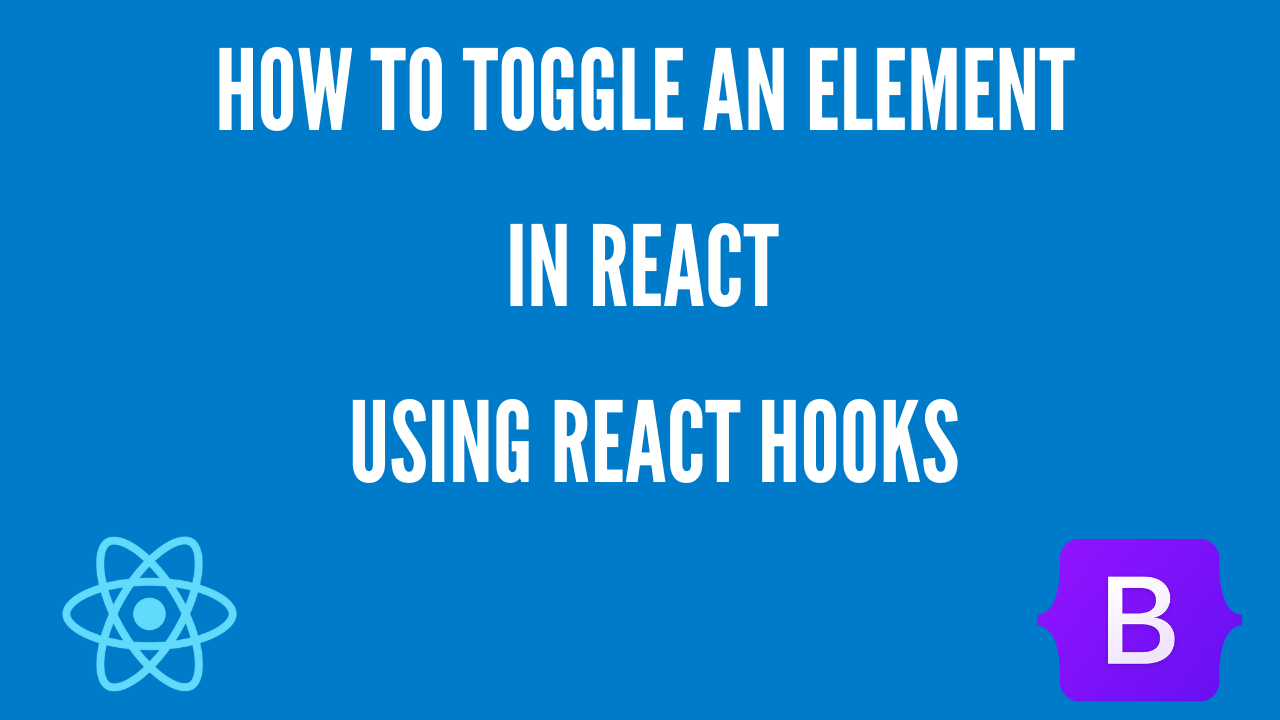
[Forum](https://forum.freecodecamp.org/) [Donate](https://www.freecodecamp.org/donate/)

**[Learn to code — free 3,000-hour curriculum](https://www.freecodecamp.org/" \t "_blank)**

NOVEMBER 7, 2022/[#REACT](https://www.freecodecamp.org/news/tag/react/)

**How to Toggle an Element in React using React Hooks**

[**Deji Adesoga**](https://www.freecodecamp.org/news/author/desoga/)



When building a web application, toggling an element is one of the key features you are likely to come across and may need to implement in your project.

There are various ways you can toggle an element. In this article, we will take a look at how we can implement toggle functionalities in five (5) different ways in React.

**Table of Contents**

* [How to Install and Setup the React Project](https://www.freecodecamp.org/news/toggle-elements-in-react-using-hooks/#how-to-install-and-setup-the-react-project)
* [How to Toggle an Element Using Logical Operators](https://www.freecodecamp.org/news/toggle-elements-in-react-using-hooks/#how-to-toggle-an-element-using-logical-operators)
* [How to Toggle an Element Using the useToggle Hook](https://www.freecodecamp.org/news/toggle-elements-in-react-using-hooks/#how-to-toggle-an-element-using-the-usetoggle-hook)
* [How to Toggle an Element Using the Ternary Operator](https://www.freecodecamp.org/news/toggle-elements-in-react-using-hooks/#how-to-toggle-an-element-using-the-ternary-operator)
* [How to Toggle an Element Using the If/Else Statement](https://www.freecodecamp.org/news/toggle-elements-in-react-using-hooks/#how-to-toggle-an-element-using-the-if-else-statement)
* [How to Toggle an Element Using CSS Conditional Styling](https://www.freecodecamp.org/news/toggle-elements-in-react-using-hooks/#how-to-toggle-an-element-using-css-conditional-styling)
* [Conclusion](https://www.freecodecamp.org/news/toggle-elements-in-react-using-hooks/#conclusion)

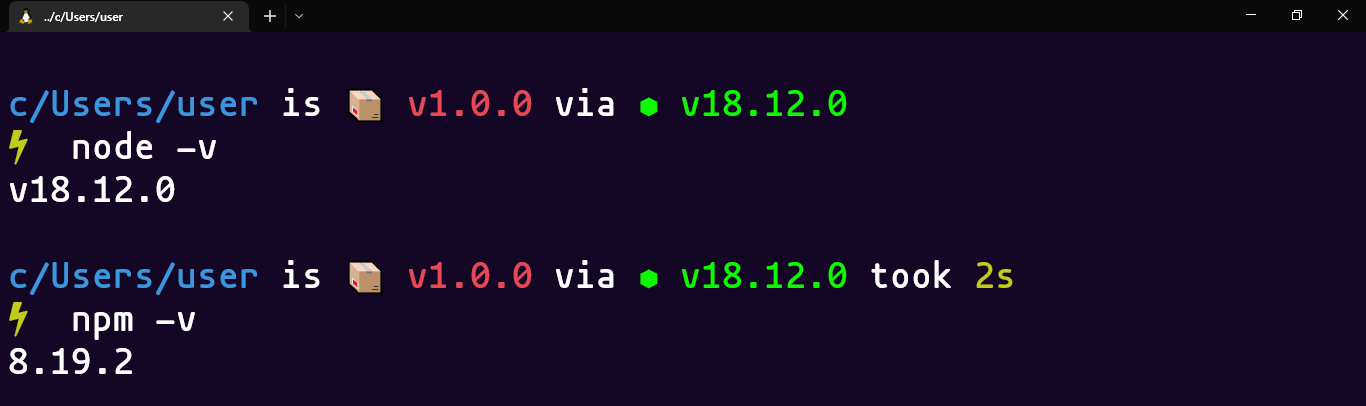
You can also watch the video version of this article below, or on my [YouTube channel](https://www.youtube.com/watch?v=S_mgSHCWCmA&t=15s):

**How to Install and Setup the React Project**

To create a React project, you need to have access to NPM (Node Package Manager). Access to NPM requires that you have Node.js installed. You can install Node by heading to the [official Node.js](https://nodejs.org/en/) website and downloading Node.js.

Node.js official documentation

I advise selecting the "Recommended For Most Users" version. Once the installation is complete, you can open your terminal and run the commands node -v and npm -v. This gives you details on the version of Node and npm you have.

Terminal showing node and npm versions

Still in your terminal, you can now install [Create React App](https://create-react-app.dev/) which is a platform that allows you to create a React project using the command below:

npm i create-react-app

The next step is to create a new React project from the terminal by running the command:

npm init react-app toggle

cd toggle

code .

Above, we created a new project called toggle. Then we navigated into the newly created project directory and opened the project in our code editor. We can now begin the process of implementing the different methods of toggling an element.

**How to Toggle an Element Using Logical Operators**

To make sure the design of our page looks structured, we are going to set up Bootstrap 5 inside of the React project.

To do this, head to the [Bootstrap 5](https://getbootstrap.com/docs/5.0/getting-started/introduction/) website and copy the CSS CDN link tag. Then go to your index.html file in your React project which can be found in the public directory. Paste the CDN link in the head section, you can see in the code below:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8" />

<link rel="icon" href="%PUBLIC\_URL%/favicon.ico" />

<meta name="viewport" content="width=device-width, initial-scale=1" />

<meta name="theme-color" content="#000000" />

<meta name="description" content="Web site created using create-react-app" />

<link rel="apple-touch-icon" href="%PUBLIC\_URL%/logo192.png" />

<link rel="manifest" href="%PUBLIC\_URL%/manifest.json" />

<!-- Bootstrap 5 CDN Link -->

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet"

integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC" crossorigin="anonymous">

<title>React App</title>

</head>

<body>

<noscript>You need to enable JavaScript to run this app.</noscript>

<div id="root"></div>

</body>

</html>

Next, create a new folder called components inside of the src directory. Then create a new file called LogicalNot.js inside the components folder. To implement the **logical not** operator method, we'll implement the code below:

import React, { useState } from 'react'

const LogicalNot = () => {

//Using Inline Function and the The Logical Not (!) to toggle state

const [toggle, setToggle] = useState(true)

return (

<>

<button

onClick={() => setToggle(!toggle)}

class="btn btn-primary mb-5">

Toggle State

</button>

{toggle && (

<ul class="list-group">

<li class="list-group-item">An item</li>

<li class="list-group-item">A second item</li>

<li class="list-group-item">A third item</li>

<li class="list-group-item">A fourth item</li>

<li class="list-group-item">And a fifth one</li>

</ul>

)}

</>

)

}

export default LogicalNot

Inside of the LogicalNot.js file, we start off by:

* Importing the useState hook.
* Then we create two variables called toggle and setToggle, while setting the initial state to **true**.
* Next, inside of the *jsx* section, we create a button that has an onClick event handler. Within this onClick handler, we create an anonymous function by using the setter we declared earlier called setToggle. We then set the argument in the anonymous function to !toggle which creates a false effect when it's clicked.
* Finally, we toggle the element in the ul tag by wrapping it around the toggle variable and then rendering it conditionally on the page using the logical && Operator.

To display the LogicalNot.js file on the browser, head to the App.js file and import the file there as seen below:

import './App.css';

import LogicalNot from './components/LogicalNot';

function App() {

return (

<div className="App mt-5">

<LogicalNot />

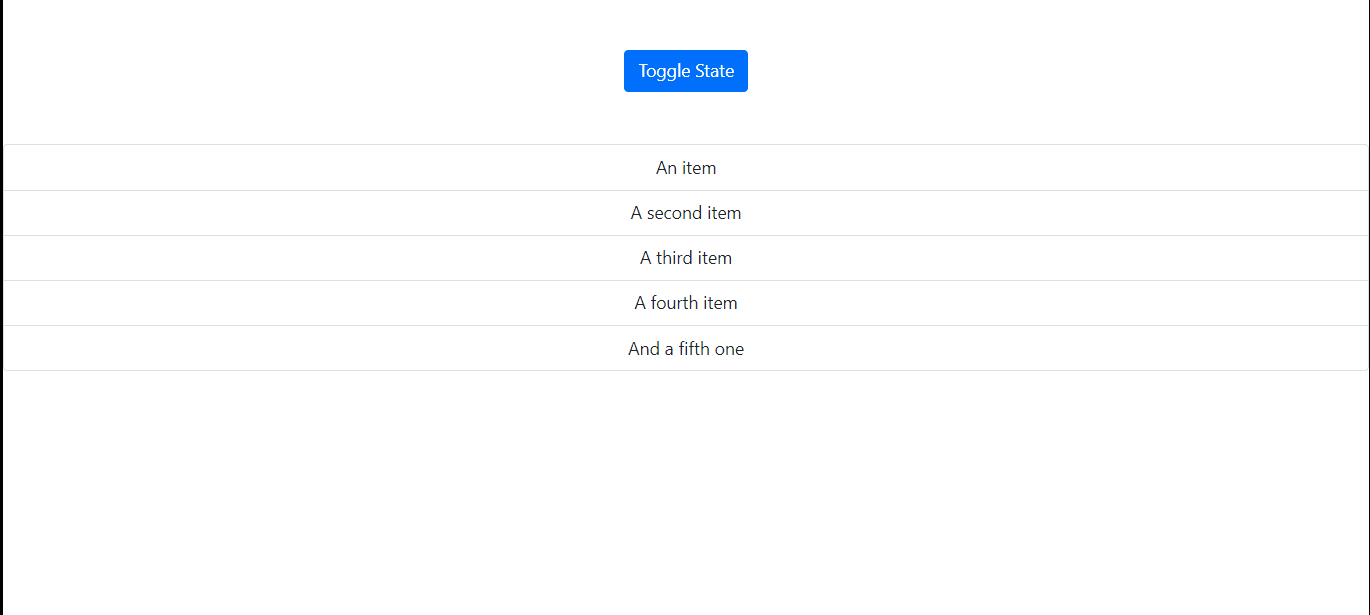
</div>

);

}

export default App;

With that, you should have the result below:

Logical not operator sample

**How to Toggle an Element Using the useToggle Hook**

You'll start this step by creating a new file called ToggleHook.js inside the *components* folder. Inside this file, import the useState hook.

import React, { useState } from 'react'

Next, create a variable called useToggle which will hold the logic for the useToggle hook as you can se below:

//Using useToggle Hook

const useToggle = (initialState) => {

const [toggleValue, setToggleValue] = useState(initialState);

const toggler = () => { setToggleValue(!toggleValue) };

return [toggleValue, toggler]

};

* Above, we created a callback function and then gave it a parameter called initialState.
* Next, we used the useState hook to create a getter and a setter called toggleValue and setToggleValue, respectively. The useState hook takes the initialState parameter we created earlier which sets the initial value as false by default.
* Finally we then created a variable called toggler. This variable holds an anonymous function that contains our useState variables and then sets the results to the opposite value when clicked. We then returned both the toggleValue and toggler variables in an array.

With this we can now use the useToggle hook to create a getter and a setter variable as you can see below:

const [toggle, setToggle] = useToggle();

We can now implement the logic of the useToggle hook in the jsx part of our code:

return (

<div>

<button

onClick={setToggle}

class="btn btn-secondary mb-5">

Toggle State

</button>

{toggle && (

<ul class="list-group">

<li class="list-group-item">An item</li>

<li class="list-group-item">A second item</li>

<li class="list-group-item">A third item</li>

<li class="list-group-item">A fourth item</li>

<li class="list-group-item">And a fifth one</li>

</ul>

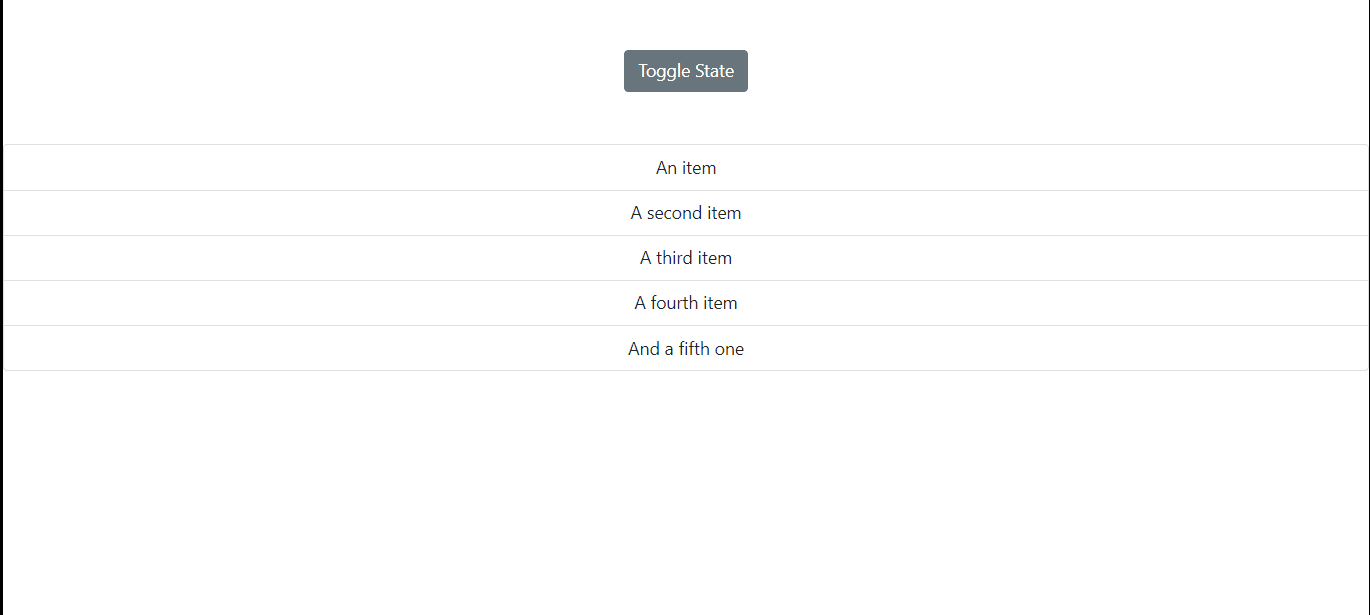
)}

</div>

)

* Above, we created a button that contains an onClick event handler using the setToggle setter previously declared above.
* Then we rendered the elements based on the boolean condition of the toggle variable when it gets clicked.

With that, we should have the result below:

useToggle sample

**How to Toggle an Element Using the Ternary Operator**

The ternary operator is a JavaScript operator that takes in three different operations, which are:

* A condition
* A question mark (?) to execute a condition if true
* A colon (:) to execute a condition if false

To implement this method, you'll start by importing the useState hook:

import React, { useState } from 'react'

Then you need to create two variables using the useState hook and setting the default value to true:

const [toggle, setToggle] = useState(true);

Next, create a variable called handleClick that holds a callback function. Within this function, call the setToggle setter and then pass in !toggle to return an opposite value when clicked, as you can see below:

const handleClick = () => {

setToggle(!toggle);

};

Finally, you can now render the logic of the variables you created in your jsx:

return (

<div>

<button

onClick={handleClick}

class="btn btn-info mb-5">

Toggle State

</button>

{toggle ?

<ul class="list-group">

<li class="list-group-item">An item</li>

<li class="list-group-item">A second item</li>

<li class="list-group-item">A third item</li>

<li class="list-group-item">A fourth item</li>

<li class="list-group-item">And a fifth one</li>

</ul>

:

<></>

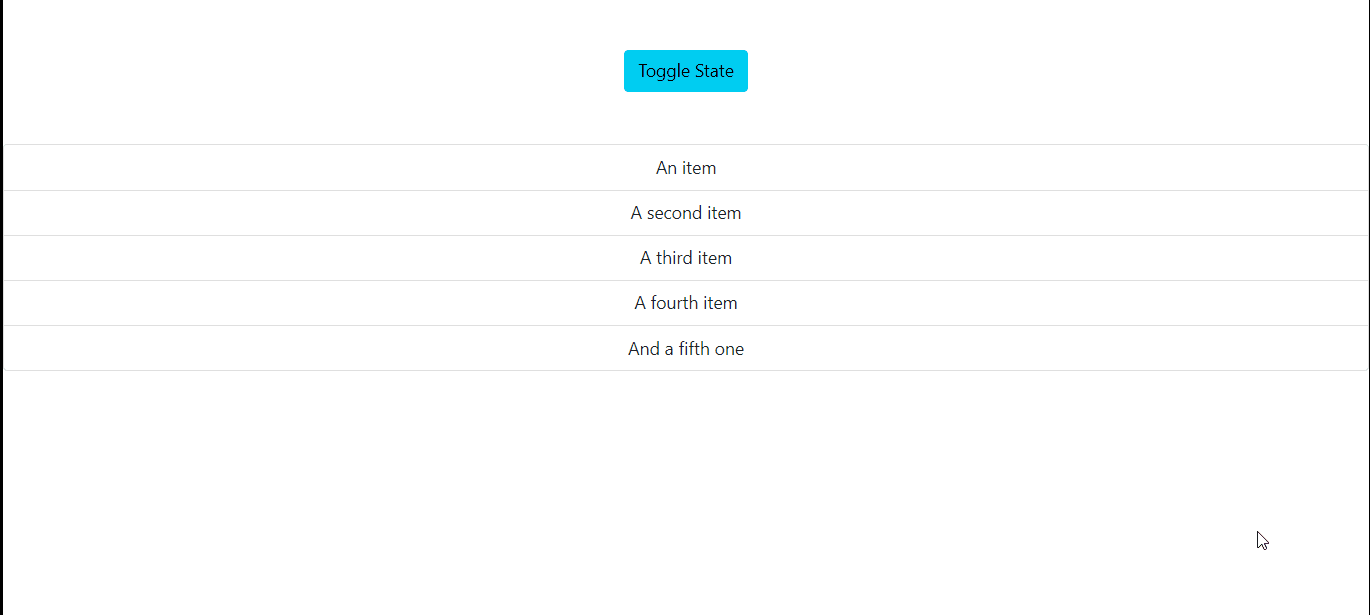
}

</div>

)

* Above we created a button that uses an onClick event handler to reference the handleClick variable we created earlier.
* Then we can render the elements by using the toogle variable condition, as well as the question mark (?) which displays elements if the toggle variable is set to true, or displays an empty *jsx* fragment by using the colon (:) if the toggle variable is set to false.

With that, we should have the result below:

Ternary operator sample

**How to Toggle an Element Using the If/Else Statement**

The If/Else statement is a conditional statement used to perform different actions based on certain parameters. The if statement executes a certain condition if it is true, and the else statement executes when the condition is false.

To see the if/else statement in action, let's begin by importing the useState hook:

import React, { useState } from 'react'

Next, create the getter and setter variables and then set the default value to true:

const [toggle, setToggle] = useState(true);

Next, create a variable called handleClick that holds the callback function. Within this function, call the setToggle setter and then pass in !toggle to return an opposite value when clicked, as you can see below:

const handleClick = () => {

setToggle(!toggle)

};

You can now display the elements by using the toggle getter in the jsx as you can see below:

if (toggle) {

return (

<div>

<button onClick={handleClick} class="btn btn-dark mb-5">Toggle State</button>

<ul class="list-group">

<li class="list-group-item">An item</li>

<li class="list-group-item">A second item</li>

<li class="list-group-item">A third item</li>

<li class="list-group-item">A fourth item</li>

<li class="list-group-item">And a fifth one</li>

</ul>

</div>

)

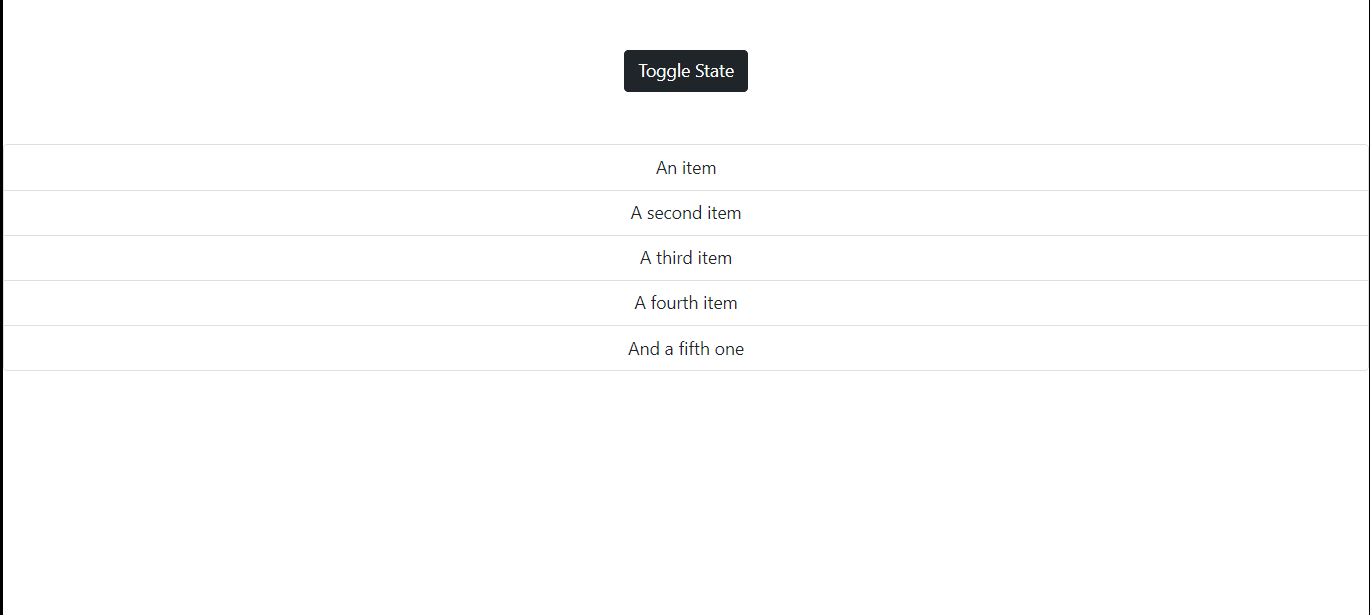
} else {

return <button onClick={handleClick} class="btn btn-dark mb-5">Toggle State</button>

}

* In the jsx, we wrapped the entire element around the if/else statement.
* Within the if statement, we rendered the elements that contain the list items on the page when the toggle is set as true.
* In the else block, however, when the toggle is set to false, only the button element gets returned.

With that, we get the result below:

If/else statement sample

**How to Toggle an Element Using CSS Conditional Styling**

Conditional styling is one of the ways you can use to manipulate DOM elements in React based on a specific condition. As we've done previously, let's start by importing the useState hook in React:

import React, { useState } from 'react'

Next, set up your useState hook creating the required variables:

const [toggle, setToggle] = useState(true);

Then create the function that helps set the value of your opposite state when clicked:

const handleClick = () => {

setToggle(!toggle);

};

With this you can now configure the conditional styling in the jsx section of your code:

return (

<div>

<button onClick={handleClick} class="btn btn-warning mb-5">Toggle State</button>

<ul class="list-group" style={{ display: toggle ? 'block' : 'none' }}>

<li class="list-group-item">An item</li>

<li class="list-group-item">A second item</li>

<li class="list-group-item">A third item</li>

<li class="list-group-item">A fourth item</li>

<li class="list-group-item">And a fifth one</li>

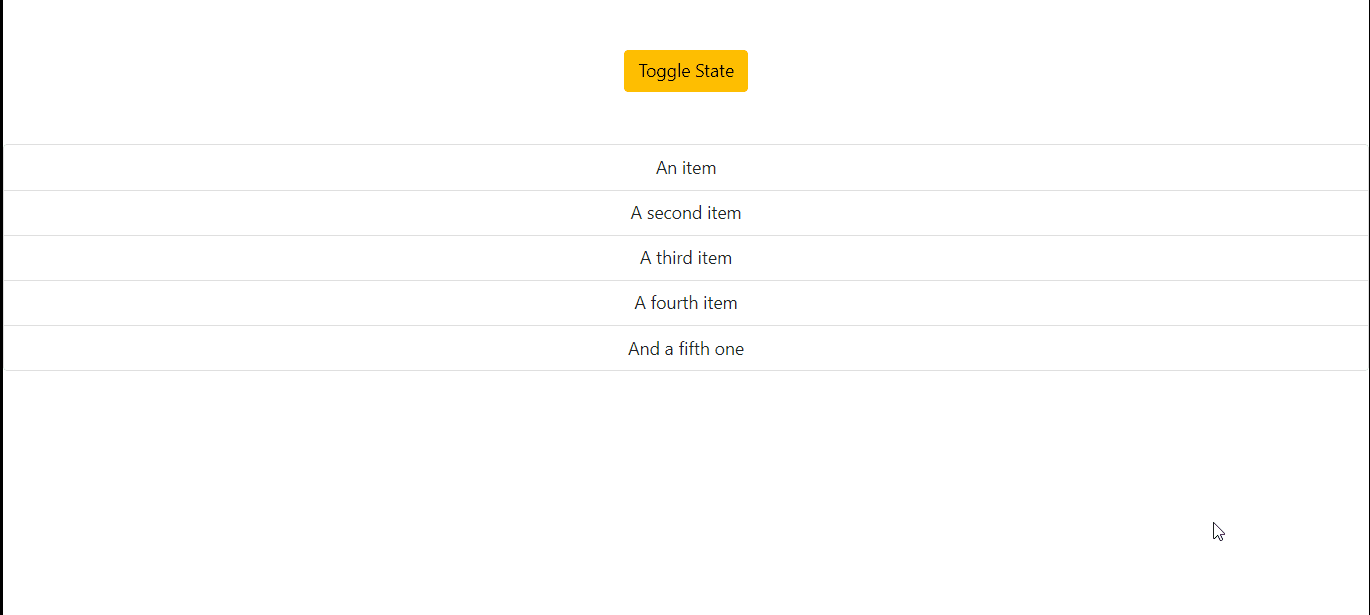
</ul>

</div>

)

* Above, we started by creating the button that contains the onClick event handler called handleClick as created earlier.
* Then we used the style attribute in the ul tag to conditionally set the display to block when the toggle variable is true. If the toggle variable is false, we set the display to none. This is possible through the ternary operator.

The result looks like this:

Conditional styling sample