

Conclusion



Gaurav Singhal

- [Introduction](#)
- [What Is a Popover?](#)
- [Popover vs. Tooltip](#)
- [Overlays](#)
- [Set up a Popover](#)
- [Position the Popover Using the Placement Prop](#)
- [Position the Popover Using a Custom Component](#)
- [Conclusion](#)
- [Top ^](#)



# How to Position a React-Bootstrap Popover

Gaurav Singhal

May 8, 2020 • 9 Min read • 5,266 Views

May 8, 2020 • 9 Min read • 5,266 Views

Web Development

Front End Web Dev...

Client-side Framew...

React

## Introduction

React-Bootstrap provides UI elements, such as forms, inputs, buttons, popovers, tooltips, etc., as React components so that they can be imported and used easily without writing any external JavaScript or JQuery. In this guide, you'll learn how to use a popover component from the React-Bootstrap library and learn to position it on the page.

## Conclusion



- [Introduction](#)
- [What Is a Popover?](#)
- [Popover vs. Tooltip](#)
- [Overlays](#)
- [Set up a Popover](#)
- [Position the Popover L](#)
- [Position the Popover L](#)
- [Conclusion](#)
- [Top ^](#)

## What Is a Popover?



Before using a UI element, it's important to understand its use cases to ensure that you're using it in the right place. A popover is a view that lasts for only a short time to show content on a page, usually triggered on a click event. Using popovers enhances the visual effect of your content on large screens, and it helps you divide big chunks of content into a more attractive and presentable view.

For instance, while calculating the price of an inventory item in an e-commerce web app, you could use a popover to show the added taxes or shipping charges. In another scenario, you could use a popover to display the details of a new guitar model that your blogging app is marketing.

## Popover vs. Tooltip

In terms of design interface, both popovers and tooltips are quite similar. They both show content for a particular element when triggered; however, there are some noticeable differences between the two that demarcate their use cases.

## Conclusion



- [Introduction](#)
- [What Is a Popover?](#)
- [Popover vs. Tooltip](#)
- [Overlays](#)
- [Set up a Popover](#)
- [Position the Popover](#)
- [Position the Popover](#)
- [Conclusion](#)
- [Top ^](#)

A tooltip is generally triggered when you hover over an element, whereas a popover is triggered on a click. Additionally, tooltips offer short and precise information about an element to help the user understand what the element says. Popovers, on the other hand, are more verbose and contain a great deal of information, like a title and description of content. This is why tooltips usually find their place on icons, unlike popovers, which are used more indiscriminately.

## Overlays

In order to use the popover and tooltip components, you also need to know what an overlay is and how to use it. An overlay is a component that sits on top of a popover, controlling its position and visibility. Transitions and toggling of the popover component are handled by the overlay component. It can be imagined as a wrapper ensuring proper functioning of the popover.

## Set up a Popover

## Conclusion



- [Introduction](#)
- [What Is a Popover?](#)
- [Popover vs. Tooltip](#)
- [Overlays](#)
- [Set up a Popover](#)
- [Position the Popover](#)
- [Position the Popover](#)
- [Conclusion](#)
- [Top ^](#)

Start by creating a project. Make sure you have `Nodejs` and `npm` installed in your machine (at least version 8 or higher) along with a code editor and a web browser (preferably Chrome or Firefox).



Create a new project using `create-react-app`:

```
1 npx create-react-app react-bootstrap-popover
```

shell

## Install React Bootstrap

First, you should have a bootstrap installed inside your project. To do that, inside the root directory, run the following command to install the React Bootstrap library.

```
1 npm install react-bootstrap bootstrap
```

## Clean Up the Template

For brevity, let's put all the code inside `App.js`. Remove the logo, App.css, and all their imports from `App.js`. Clean out the starter template inside the app component. Your `App.js` should look like this:

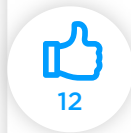
```
1 import React from 'react';  
2
```

jsx

## Conclusion



- [Introduction](#)
- [What Is a Popover?](#)
- [Popover vs. Tooltip](#)
- [Overlays](#)
- [Set up a Popover](#)
- [Position the Popover U](#)
- [Position the Popover U](#)
- [Conclusion](#)
- [Top ^](#)



```
3
4  function App() {
5    return (
6      <div className="App">
7        <h2>Hello</h2>
8      </div>
9    );
10  }
11
12  export default App;
```

## Add the Popover Component

For regular Bootstrap styles to work correctly, import the Bootstrap styles on top.

```
1  import 'bootstrap/dist/css/bootstrap.min.css';
```

jsx

This is equivalent to adding Bootstrap CDN to your `index.html` file. Now import three things: Popover, OverlayTrigger, and Button from `react-bootstrap`.

```
1  import Popover from 'react-bootstrap/Popover';
2  import OverlayTrigger from 'react-bootstrap/OverlayTrigger';
3  import Button from 'react-bootstrap/Button';
```

jsx

## Conclusion



- [Introduction](#)
- [What Is a Popover?](#)
- [Popover vs. Tooltip](#)
- [Overlays](#)
- [Set up a Popover](#)
- [Position the Popover L](#)
- [Position the Popover L](#)
- [Conclusion](#)
- [Top ^](#)

To create a popover inside the `Popover` Component, render the `Popover.Title` component to indicate the title of the popover and `Popover.Content` component to indicate its content. Store this popover in a constant variable and output it inside the JSX template.

```
1      ...
2      const popover = (
3        <Popover id="popover-basic">
4          <Popover.Title as="h3">Popover title</Popover.Title>
5          <Popover.Content>
6            Popover content <strong>some strong content</strong> Normal content
7          </Popover.Content>
8        </Popover>
9      );
10     ...
```

jsx

Output the popover inside an overlay and use the trigger prop to set the type of event to which the overlay listens. The placement prop defines the position of your popover.

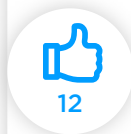
```
1      ....
2      return (
3        <div className="App">
4          <OverlayTrigger trigger="click" placement="right" overlay={popover}>
5            <Button variant="success">Click to trigger popover</Button>
6          </OverlayTrigger>
```

jsx

## Conclusion



- [Introduction](#)
- [What Is a Popover?](#)
- [Popover vs. Tooltip](#)
- [Overlays](#)
- [Set up a Popover](#)
- [Position the Popover](#)
- [Position the Popover](#)
- [Conclusion](#)
- [Top ^](#)



Finally, your `App.js` should look like this.

```
7     </div>
8   );
9   ....
```

jsx

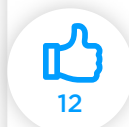
```
1   import React from 'react';
2   import 'bootstrap/dist/css/bootstrap.min.css';
3   import Popover from 'react-bootstrap/Popover';
4   import OverlayTrigger from 'react-bootstrap/OverlayTrigger';
5   import Button from 'react-bootstrap/Button';
6
7
8   function App() {
9     const popover = (
10      <Popover id="popover-basic">
11        <Popover.Title as="h3">Popover title</Popover.Title>
12        <Popover.Content>
13          Popover content <strong>some strong content</strong> Normal content
14        </Popover.Content>
15      </Popover>
16    );
17
18
19    return (
20      <div className="App">
21        <OverlayTrigger trigger="click" placement="right" overlay={popover}>
22          <Button variant="success">Click to trigger popover</Button>
23        </OverlayTrigger>
24      </div>
25    );
```

## Conclusion



- [Introduction](#)
- [What Is a Popover?](#)
- [Popover vs. Tooltip](#)
- [Overlays](#)
- [Set up a Popover](#)
- [Position the Popover L](#)
- [Position the Popover L](#)
- [Conclusion](#)
- [Top ^](#)

```
26      }  
27  
28      export default App;
```



## Test the UI

To see the UI inside the root directory, run this code.

```
1      npm start
```

This will spin up a local development server (usually on port 3000), and you can see the popover button and the popover itself when you click on the button. Remember to style the popover button or its container div to display it at the center of the page.

Now that the popover and overlay are ready, you can position the popover.

## Position the Popover Using the Placement Prop

Since the overlay component handles the utility classes of the popover component, a convenient way to set up the popover's position is by playing around with the placement prop. It has been



## Conclusion



- [Introduction](#)
- [What Is a Popover?](#)
- [Popover vs. Tooltip](#)
- [Overlays](#)
- [Set up a Popover](#)
- [Position the Popover Left](#)
- [Position the Popover Top](#)
- [Conclusion](#)
- [Top ^](#)

set to `right`; hence the popover comes to the right of the button.  
You can set it to any of the four positions: top, left, bottom, and



down.

### Position the Popover to the Left

Pass in `left` inside the `placement` prop to position the popover to the left.

```
1      ....
2      <OverlayTrigger trigger="click" placement="left" overlay={popover}>
3          <Button variant="success">Click to trigger popover</Button>
4      </OverlayTrigger>
5      ....
```

jsx

### Position the Popover at the Top

Pass in `top` inside the `placement` prop to position the popover on the top of the button

```
1      ...
2      <OverlayTrigger trigger="click" placement="top" overlay={popover}>
3          <Button variant="success">Click to trigger popover</Button>
4      </OverlayTrigger>
5      ...
```

jsx

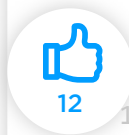
### Position the Popover at the Bottom

## Conclusion



- [Introduction](#)
- [What Is a Popover?](#)
- [Popover vs. Tooltip](#)
- [Overlays](#)
- [Set up a Popover](#)
- [Position the Popover](#)
- [Position the Popover](#)
- [Conclusion](#)
- [Top ^](#)

Pass in `bottom` inside the `placement` prop to position the popover on the bottom of the button



```
1    ...
2    <OverlayTrigger trigger="click" placement="bottom" overlay={popover}>
3        <Button variant="success">Click to trigger popover</Button>
4    </OverlayTrigger>
5    ...
```

jsx

## Position the Popover Using a Custom Component

For precise positions, you can build a custom popover component that calculates its position and then passes that position to the `placement` prop. The `positionLeft` and `positionTop` components can be used to pass the exactly calculated position to the `Overlay` component.

Create a new component `CustomPopover` inside the root directory and add the following code:

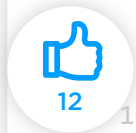
```
1    import React from 'react';
2    import Popover from 'react-bootstrap/Popover';
3    import 'bootstrap/dist/css/bootstrap.min.css';
4
```

jsx

## Conclusion



- [Introduction](#)
- [What Is a Popover?](#)
- [Popover vs. Tooltip](#)
- [Overlays](#)
- [Set up a Popover](#)
- [Position the Popover](#)
- [Position the Popover](#)
- [Conclusion](#)
- [Top ^](#)



```
5     const CustomPopover=({props})=> {
6         const position=calculatePosition;
7         const calculatePosition=()=>{
8             //set position here
9         }
10
11     return (
12         <Popover {...props} positionLeft={position}>
13             {props.children }
14         </Popover>
15     )
16
17 }
18
19 export default CustomPopover;
```

You can use a simple function to calculate the precise position for your popover. Then you can use it inside your `Overlay` component by only rendering this `CustomPopover` component as a child.

If you get any depreciated warnings or errors, you can use the exact version of react-bootstrap used in this guide by updating your `package.json` file and running the command `npm i` .

json

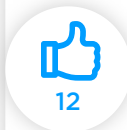
```
1     ....
2     "bootstrap": "^4.4.1",
3     "react-bootstrap": "^1.0.1",
4     ....
```

## Conclusion



- [Introduction](#)
- [What Is a Popover?](#)
- [Popover vs. Tooltip](#)
- [Overlays](#)
- [Set up a Popover](#)
- [Position the Popover Using a Custom Component](#)
- [Position the Popover Using a Custom Component](#)
- [Conclusion](#)
- [Top ^](#)

## Conclusion



Good design has just the right amount of detail and precision. A popover's position should be judged appropriately in terms of the type and length of content it displays. The position of the parent container and the triggering element are defined by the overlay component. If you need a more customized position, revise the design with emphasis on the triggering element, as the popover's position is always relative to its triggering element.