Gaurav Singhal

How to Capture the Value of Dropdown Lists with React-Bootstrap

Gaurav Singhal

Sep 14, 2020 • 8 Min read • 28,845 Views

Sep 14, 2020 • 8 Min read • 28,845 Views

Web Development Front End Web Dev... Client-side Framew... React

Introduction

43

Introduction

React's DOM handling ability, when intertwined with a useful UI, is always pleasing for end users. Courtesy of a large number of open-source packages, React developers can now easily integrate any UI/UX library like Bootstrap (one of the most popular CSS frameworks) with a single-page React app. As an improvisation, Bootstrap's component-based library is more preferred while working with these frontend frameworks. Attributed to its component architecture, every UI component is taken up as a React component itself. Every UI component, such as forms, inputs, tooltips, dropdowns, modals, etc., are all separate React based components in React Bootstrap. This guide will show you how to use React Bootstrap to build a dropdown list for your forms and capture their values on the front end.

onSelect Event

Just like the `onChange` event watches for changes in an input field, the `onSelect` event occurs after some value is selected in an element. A dropdown list can be drawn closer to a regular input field since, under the shell, they both aim to get some value from the user. To listen to those selected values when the user dynamically changes them, the `onSelect` event comes in handy.

onSelect Event Handler

After the `onSelect` event has been set to watch for a selection of value, the next step is to store that dynamic data somewhere. To do so, an `event handler` or a simple JavaScript function is invoked every time the event is triggered, and the data is extracted using the `event object`. In this case, it becomes even simpler using props.

Implementation

Setup

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:

shell

```shell
npx create-react-app react-bootstrap-dropdown
```

Installing React Bootstrap

Inside the root directory, run the following command to install the React Bootstrap library.

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

This will install both Bootstrap and React Bootstrap inside the project.

Cleaning Up Template

Typically, a separate form component should handle everything, but for brevity purposes, 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:

jsx

```jsx
import React from 'react';

function App() {
  return (
    <div className="App">
      <h2>Hello</h2>
    </div>
  );
}

export default App;
```

Adding the Dropdown Component

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

jsx

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

The above is equivalent to adding Bootstrap CDN in your `index.html` file. Now import the **DropdownButton** and **Dropdown** from `react-bootstrap` :

jsx

```
import DropdownButton from 'react-bootstrap/DropdownButton';
import Dropdown from 'react-bootstrap/Dropdown'
```

Render them on the DOM inside `App.js` :

jsx

```
....
     <DropdownButton
      alignRight
      title="Dropdown right"
      id="dropdown-menu-align-right"

        >
              <Dropdown.Item eventKey="option-1">option-1</Dropdown.Item
              <Dropdown.Item eventKey="option-2">option-2</Dropdown.Item
              <Dropdown.Item eventKey="option-3">option 3</Dropdown.Item
              <Dropdown.Divider />
              <Dropdown.Item eventKey="some link">some link</Dropdown.Ite
      </DropdownButton>
  ....
```

Testing the UI

To see the UI inside the root directory, run:

```
npm start
```

This will spin up a local development server (usually on port 3000) and you can see the **dropdown button** along with the **dropdown fields**.

Capturing the Value from the Dropdown

Next, attach a `handleSelect` function that fires when the `onSelect` function is triggered. Place this event as a `prop` inside the `DropdownButton` component and create a `handleSelect` function, which takes in the event object and logs it to the console.

jsx

```jsx
  ....
  const handleSelect=(e)=>{
     console.log(e);
   }
   ....
    <DropdownButton
```

```
          alignRight
          title="Dropdown right"
          id="dropdown-menu-align-right"
          onSelect={handleSelect}
            >
        ....
```

Click on the fields to see their values appear on the console. Great! You have successfully captured the value of a dropdown list in a React Boostrap Dropdown component.

Storing the Captured Value Inside the State

For a more practical use case, you might want to do something with this captured value. Your component n the front end should store it somewhere so that it can be sent to the database or some back end. Set up a state for the app component and store the selected value from the dropdown in the state variable. To avoid changing already existing code, use *hooks* for storing stateful data in a functional component.

jsx

```jsx
import React,{useState} from 'react';
....


function App() {
  const [value,setValue]=useState('');
  const handleSelect=(e)=>{
    console.log(e);
    setValue(e)
  }

  return (
    ...
      <h4>You selected {value}</h4>
    </div>
  );
}

export default App;
```

`value` is a *state variable*, and `setValue` is the asynchronous function that sets this state variable. To achieve this, import the `useState` hook from React. Call `setValue` inside the `handleSelect` function to set the state variable to the value selected from the dropdown. Finally, output the current state value on the DOM inside your JSX.

Try selecting another value and notice how the DOM responds to the state changes to dynamically show the selected value.

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

jsx

```
import React,{useState} from 'react';
import 'bootstrap/dist/css/bootstrap.min.css';
import DropdownButton from 'react-bootstrap/DropdownButton';
import Dropdown from 'react-bootstrap/Dropdown'
```

```


function App() {
  const [value,setValue]=useState('');
  const handleSelect=(e)=>{
    console.log(e);
    setValue(e)
  }
  return (
    <div className="App container">

      <DropdownButton
      alignRight
      title="Dropdown right"
      id="dropdown-menu-align-right"
      onSelect={handleSelect}
        >
              <Dropdown.Item eventKey="option-1">option-1</Dropdown.Item
              <Dropdown.Item eventKey="option-2">option-2</Dropdown.Item
              <Dropdown.Item eventKey="option-3">option 3</Dropdown.Item
              <Dropdown.Divider />
              <Dropdown.Item eventKey="some link">some link</Dropdown.Ite
      </DropdownButton>
      <h4>You selected {value}</h4>
    </div>
  );
}

export default App;
```

In case 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

```json
....
"bootstrap": "^4.4.1",
"react-bootstrap": "^1.0.1",
....
```

Conclusion

For novice developers, component-based libraries often seem daunting and intimidating, but in practical use cases they enhance readability and reduce unnecessary lines of code. Controlled form components can be built conveniently using React-Bootstrap, especially when form fields require validations or have to be sent to a database on submitting them. Also, using UI components in your app will definitely help you code your features faster because the built-in components handle a lot of typical JavaScript under the hood using props.