External Libraries

# Introduction

This document will outline some of the useful libraries that are external from React, however this is by no means an exhaustive list, there are approximately over 800,000 modules available via npm, with this number growing on average by 470 a day.

<http://www.modulecounts.com/>

At the culmination of this document you should be happy working with reactstrap, Font Awesome and Axios.

But most importantly you should be aware that there are literally thousands of other libraries you may find useful, I cannot stress this enough, if you are writing a piece of code to do something specific, you should first think, **‘Has someone written a module that does this already?’.**

# What are they?

These External Libraries or Modules offer certain functionality, there are some with quite obscure uses, or humorous uses and many that are very practical. You should become familiar with reading their documentation and understanding how to pick them up and use them.

# Adding Reactstrap

<https://www.npmjs.com/package/reactstrap>

Npm install bootstrap –save

Npm install –save reactstrap

Import Bootstrap CSS into your src/index.js file

**import** 'bootstrap/dist/css/bootstrap.css';

If you have any issues with this installation process, you should first ensure your package.json has the correct dependancies in it:

"dependencies": {

"bootstrap": "^4.3.1",

"react": "^16.8.6",

"react-dom": "^16.8.6",

"react-scripts": "2.1.8",

"reactstrap": "^8.0.0"

},

Then delete the node\_modules folder, after it has been deleted run *npm install* in the root of your project to recreate the node\_modules folder.

# Creating your first Reactstrap Component

The simplist way to start working with Reactstrap is to import a simple Component, for example a *Button*, and modify the various parameters of the component.

import React, { Component } from "react";

import "./App.css";

import { Button } from "reactstrap";

class App extends Component {

render() {

return (

<div>

<Button color="success">Success</Button>

</div>

);

}

}

export default App;

Now we have created a simple reactstrap component we need to understand:

* what we have done
* how we can extend this into other use cases
* ultimately how this has benefitted us.

# What we have done

Reactstrap is effectively a collection of components that have been designed for particular well known purposes that we can pick up and use in our react applications with minimal fuss. Here the *Button* is just a component that we are using, passing different props to it will make the button appear in different ways.

<https://reactstrap.github.io/components/buttons/>

The documentation at this link describes the component we have used and how we can interact with it:

## Properties

This section describes the various props that we can pass into our Button component in order to alter its properties.

For instance we set out component to have the *outline* and *disabled* properties like so:

<Button color="success" outline={true} disabled={true}>

Success

</Button>

This will render the Button as an outline that cannot be clicked.

## A button should do something when clicked

With an HTML button, element, we can pass a function to the *onClick* event that will be called every time the button is clicked. The reactStrap *Button* component is set up to react in exactly the same way.

<Button

onClick={this.handleClick}

color="success"

outline={true}

>

Success

</Button>

# How we can extend to other use cases

If you study the other components available at this link:

<https://reactstrap.github.io/components/alerts/>

On the right hand side, you will see the wide variety of commonly used/seen components available to you.

Want to include *Breadcrumbs* in your web application to aid the User Journey? there’s a component for that, how about a C*arousel*?

Understanding and using the *Button* component is a nice introduction to other harder to use components, however they have all been built along the same principles. The sample code should work straight out of the box, and you will have the ability to change different attributes by changing the props that you pass to this component.

# How has this benefitted us?

* Very quickly we have been able to use pre-built components
* These components look great
* We can easily introduce uniformity across the components used in our application
* It adds a professional look to our application.

# Font Awesome

To continue the theme of making our applications look good, we will take a look at Font Awesome.

<https://fontawesome.com/>

Font Awesome is a library of Icons that you can use, you can find many attractive symbols and designs, as well as brands and logos. For instance, you may want to create an online portfolio and include the logos of the various technologies you can work with such as Java, Docker or React.

# Installation

<https://fontawesome.com/how-to-use/on-the-web/using-with/react>

npm i --save @fortawesome/fontawesome-svg-core

npm i --save @fortawesome/free-solid-svg-icons

npm i --save @fortawesome/react-fontawesome

Run these 3 commands in the root of your project. Next we import and add the icons that we want to use into a *library.*

import { library } from "@fortawesome/fontawesome-svg-core";

import { FontAwesomeIcon } from "@fortawesome/react-fontawesome";

import { faStroopwafel } from "@fortawesome/free-solid-svg-icons";

library.add(faStroopwafel);

class App extends Component {

render() {

return (

<div>

Favorite Food: <FontAwesomeIcon icon="stroopwafel" />

</div>

);

}

}

In the body of our render method we are using the *FontAwesomeIcon* component to render the icon that we have imported. Follow the example above and then check your browser, you should see that a small icon has been rendered.

Some icons are separated out into a different import – Brands

npm i --save @fortawesome/free-brands-svg-icons

import { library } from "@fortawesome/fontawesome-svg-core";

import { FontAwesomeIcon } from "@fortawesome/react-fontawesome";

import { faMugHot } from "@fortawesome/free-solid-svg-icons";

import { faJava } from "@fortawesome/free-brands-svg-icons";

library.add(faMugHot, faJava);

render() {

return (

<div>

Favorite Drink: <FontAwesomeIcon icon="mug-hot" />

Also Coffee : <FontAwesomeIcon icon={["fab", "java"]} />

</div>

)

As you can see above the *FontAwesomeIcon* component for the Java icon looks a little different, this is because by default, the component will look for an icon with the *‘fas’* prefix, i.e. icons from the *FontAwesomeSolid (fas)* library. As the *java* icon is from the *FontAwesomeBrands* *(fab)* library we need to tell the component to look for the prefix *fab* in the style above.

# How do I choose and use an Icon?

<https://fontawesome.com/icons?d=gallery>

Here you can search for a particular icon, for this example lets choose the *React* symbol.

1. Using the *Gallery* above search for the icon you want
2. Make a note of the *Library* it is part of in the top left (fab)
3. Import both the *Core* and *FontAwesome* modules.

import { library } from "@fortawesome/fontawesome-svg-core";

import { FontAwesomeIcon } from "@fortawesome/react-fontawesome";

1. Assuming you have installed the module from earlier, import this icon in your component.

import { faReact } from "@fortawesome/free-brands-svg-icons";

1. Add this icon to your library

library.add(faReact);

1. Create a *FontAwesomeIcon* component and pass in the correct *icon* value

class App extends Component {

render() {

return (

<div>

<FontAwesomeIcon icon={["fab", "react"]} />

</div>

);

}

}

1. You should now be able to see the React logo rendered on the page.

# How has this benefitted us?

* We can nice neat icons to our application
* Make buttons or clickable object more attractive
* Help the User Journey with universal icons.

# Axios

Another very important module you will use is called Axios, it is used for making HTTP Requests to various API’s. In Vanilla JS we have XMLHTTPRequests, however Axios affords some extra functionality that we will find useful in the future.

# Installation

Ensure you are in the root folder of your project.

*Npm install –s axios*

This will allow us to make use of the Axios module in our components.

# Background Knowledge

Before we start looking at the technical nature of using Axios we need to ensure we are familiar with certain concepts.

## URI

*Universal Resource Indentifier –* The parent of URL’s, that is all URL’s are URI’s, but not all URI’s are URL’s.

## HTTP

*HyperText Transfer Protocol –* A protocol, or a contract, that describes how communication can be made over the internet, there are other Protocols that can be used.

## HTTP Request

A request could be described as a message that is sent via HTTP, we will focus on 4 kinds of Requests:

|  |  |
| --- | --- |
| **Request Name** | **What it does** |
| GET | Retrieves information |
| POST | Send Information (used for creating new data) |
| DELETE | Deletes Information |
| PUT | Send Information (Used for updating existing data) |

# Using Axios

First we need to import Axios into our component,

import axios from "axios";

We then need to write a function that will make use of the axios module, this function will make a HTTP request to a specific URI, for this example will be making a simple GET request to the OMDB API.

<http://www.omdbapi.com/>

*(The OMDB API is a really easy external API to start practising with, I recommend you start there)*

makeRequest = () => {

axios

.get("http://www.omdbapi.com/?apikey=cf6d6c63&t=hop")

.then(response => {

this.setState({

data: response.data

});

});

};

*MakeRequest –* The name of the function we are making, made using the Fat Arrow Syntax (ES6) to avoid the needing to use *this.bind(this)*. This function could be called by a button click in the component or using a lifecycle method.

*Axios –* How we imported the axios module at the beginning of the file, via this value we will use the module.

*.get –* This is a function of axios that we are using, it makes a HTTP GET Request for us. It takes in 1 parameter, which is the URI that we want to access, in this example it is the OMDB API, the link above will take you to the documentation for this API.

*.then –* The .get() function returns a *promise*, we are therefore using the *.then()* function to resolve this promise. It returns a *response* object that has various attributes, for example the status code, any headers and the data of the response. In the body of this function we make use of *setState()* to set some state of the component.

# Working with the response

Now that we have made our request and set some state of the component to be the value of response.data, we can access the state to pull out the information that we want.

render() {

const { Title, Year, Rated, Genre, Plot, Poster } = this.state.data;

return (

<div>

<h2>AutoFilm.js</h2>

<h4>{Title}</h4>

<h4>{Year}</h4>

<h4>{Rated}</h4>

<h4>{Genre}</h4>

<h4>{Plot}</h4>

<img src={Poster} />

</div>

);

}

Here we are using a technique called **Destructuring**, we take *this.state.data* and pull out the Title, Year, Rated, Genre, Plot and Poster values.

These values can then be used in the return of the render.

# How has this benefitted us?

* We can easily make HTTP requests to API’s.
* The response comes back as a promise so we can handle it logically.
* This affords us huge amounts of functionality.