**JavaScript Fetch API Example –**

[Samuel DeSantis](https://samuel-desantis.medium.com/?source=post_page-----57115af5d52e--------------------------------) Feb 27, 2022

Summary: Brief overview of Fetch API.

Fetch is a great tool for retrieving and sending data. Granted, it gets much more technical then this, but on a basic level what you want it to do is retrieve or send data with some action. See [RESTful routes](https://learn.co/lessons/sinatra-restful-routes-readme#:~:text=A%20RESTful%20route%20is%20a,HTTP%20verb%20and%20the%20URL.) for more information on these actions. I also plan on introducing some different concepts such as Lifecycle Methods and state.

Aside: There are many readings scattered throughout the post, so if you are unfamiliar with something being used there should be a link to more in-depth reading.

I am assuming that your environment is setup and you are able to use React, if not, go ahead and do that, then come back for the rest of the post. First, lets go ahead and create our [React](https://reactjs.org/) app with:

npx create-react-app react-fetch-example

You can change the name to whatever you would like… make sure you are in the React app directory, then go ahead and install modules and dependencies and start the server, npm install && npm start . You will then be greeted with the lovely default React page.

For this example I wanted to use a React Class Component. So delete what should be a React Functional App component in src/App.js and refer to Figure 1 to setup the React Class Component. See [Components & Props](https://reactjs.org/docs/components-and-props.html) for more information.

Text

Description automatically generated

Figure 1: React Class Component

Now that we have the Class Component all set up, lets go over what we want to do. First, I want to get data from [Open Notify](http://open-notify.org/Open-Notify-API/People-In-Space/) to see who all is in outer space right now. Then, I want to show that data on the webpage so the user can see who is in space.

Go ahead and create a function before render() and lets call it: getAstronauts = () => {}

See [Arrow Function Expressions](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions/Arrow_functions) if you aren’t familiar with them.

Now that we have our function we are going to add our URL and start building out our fetch request. Add the following to the first line of our function.

const URL = "http://api.open-notify.org/astros.json"

Now, lets take a look at a fetch request… a fetch request takes an argument of a URL, retrieves some data from the URL, extracts [JSON](https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/JSON), then we can do whatever we want with that JSON. Go ahead and add the following lines of code:

fetch(URL) //Fetching data from the URL we added above

.then(response => response.json() //Extracting the JSON

.then(result => console.log(result)) //Logging the resulting JSON

Okay, great, so now if we look at the server console that we have up, we should see… Wait a minute… We aren’t actually calling getAstronauts() yet so no wonder it isn’t showing up. What we are going to use now is called a [Lifecycle Method](https://reactjs.org/docs/react-component.html#commonly-used-lifecycle-methods). We are going to use the Lifecycle method componentDidMount so that when our App component renders it will go ahead and run our getAstronauts() function.

componentDidMount () { this.getAstronauts() }

Go ahead and check the server console again and if everything is working right, you should see something like, {people: Array(10), message: 'success', number: 10} . You can play around with expanding the data to see what we are working too. Next, we are going to use our component state to store the data that we just fetched. Before we move on to state, lets make sure we are looking at the same code.

Text

Description automatically generated

Figure 2: Setting Up Fetch

This next bit may get a little more complicated, but don’t worry, you don’t need to know the in’s and out’s of it. Once you get introduced to the idea you can read more about it and figure it out as you go. Lets go ahead and add [state](https://reactjs.org/docs/state-and-lifecycle.html) to the top of our component.

state = { people: [] } // We can store data in the component state

Currently the state of people is an empty [array](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array), but now lets go ahead and populate it with the people that we are grabbing from our fetch data. We are going to take the .then(result => console.log(result)) code from above and change it to:

.then(result => this.setState({ people: result.people }))

The line above is what is called a ‘shallow merge’ of stateChange into the new state. Basically, we take the data we fetched, use only the people from that data, then store it in our once empty people array.

Now that we have the data stored in our state, lets go ahead and show it to the user. Lets create a function called, showAstronauts = astronauts => {} where we take a bunch of astronauts and show them to the user. Let’s go ahead and game plan this function…

1. Take an array of an of astronauts
2. Look at each astronaut and get their info
3. Display astronaut information

We have already taken care of the first point in the above line of code setting a parameter of astronauts . Next, we are going to update showAstronauts to:

showAstronauts = astronauts => {return astronauts.map(astronaut=>{})}

We are using [map](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/map) here to create a new array and look at each one of our elements in the array of astronauts. Next, we are going to take the astronaut information and format it in a <div> so we can display it in our HTML body.

showAstronauts = astronauts => { // parameter of ‘astronauts’

return astronauts.map(astronaut => { // looking at an astronaut

return <div>{astronaut.craft} : {astronaut.name}</div>})} // show info

Now that the function is created, we just need to call it in the right spot and feed it the astronauts that we stored in our state of people from earlier. Let’s go ahead and replace “App” in our render function with our newly created function.

render () {return <div>{this.showAstronauts(this.state.people)}</div>}

Cool! Now you should be seeing a list of astronauts kinda like this:

A picture containing text, person, screenshot

Description automatically generated

Figure 3: List of Astronauts

If you aren’t seeing this, then go ahead and check the code below to see if we have the same thing!

Text

Description automatically generated

Figure 4: Finished Code

The above code is typed to match what I presented in the post, but you can adjust the spacing and indentation to look prettier. I tried to keep things inline for the post because it looks a bit better on Medium then having multiple lines.

I hope this was helpful and maybe opened up some new things to take a look at for someone who is starting out.

For more information visit: [Using the Fetch API by MDN Web Docs](https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch)