## The GET request with browsers


Making an HTTP GET request is easy. Let's say you want to visit gomycode.co in your browser. All you need to do is launch your browser and enter the address: https://www.gomycode.co

## What's fetch?

To fetch data from an API, we can use the JavaScript fetch method. We can also use an external library like axios, request, superagent, supertest and many others.
In this Super Skill, we are going to use the fetch method.

The fetch method is provided by web APIs and it's supported by almost all the new browser versions.

PS. There is no need to install anything.

In this part, we are going to get the list of courses from algolia API on Redux.

```
"https://hn.algolia.com/api/v1/search?query=redux"
```

## The GET request with React.

In this step, we are going to create our React project in the App.js. What we are going to need is:

1. A state to set the returned values from the API.

2. A state to handle the error (if there is any).
3. useEffect hook on the API after the component mounts.

```
import React, { useEffect, useState } from "react";

const App = () => {
 const [data, setData] = useState();// where to store the returned data
 const [error, setError] = useState(null);// where to store the coming errors
 useEffect(() => {
   function fetchData() {// the function to fetch data from the api
     fetch("https://hn.algolia.com/api/v1/search?query=redux")
       .then(res => res.json())
       .then(res => setData(res))
       .catch(err => setError(err));
   }

   fetchData();
 }, []);
 return <div />;
};
export default App;
```

## The GET request with React.

Well let's explain some concepts first. When we send a request to an API, it will take some time to find the needed data from its resources and return exactly what we want. Therefore, JavaScript will not block the rest of the operation but it will continue executing the rest of the program and when the API gives us a result of our request, JavaScript will handle it then. This is the asynchronous approach in JavaScript, we will see it in more detail in other super skills.

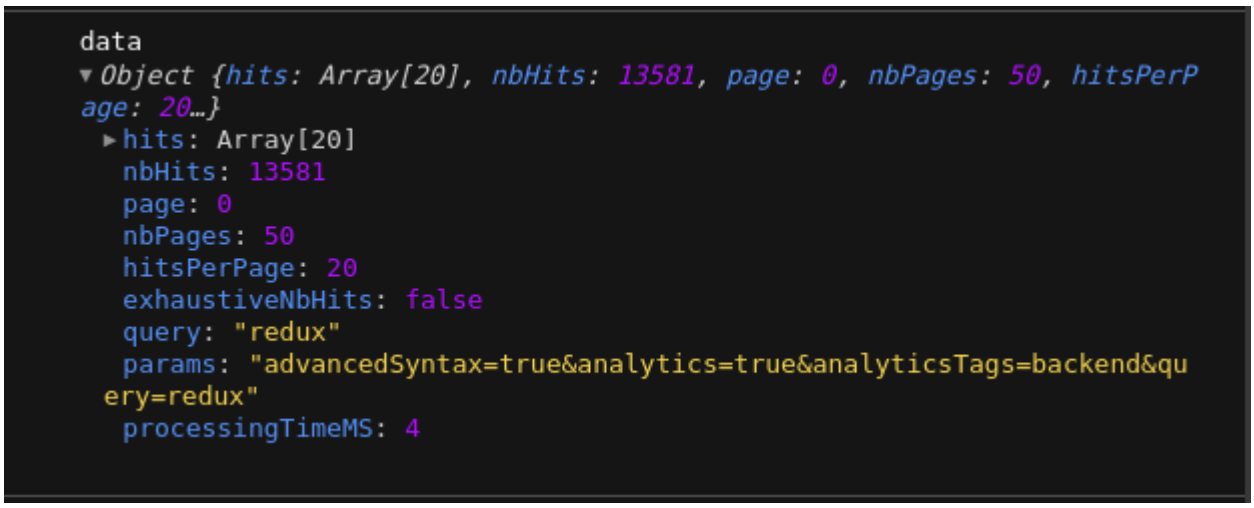
For the time being, JavaScript gives us the `.then()` `.catch()` methods to deal with asynchronous functions.

In simpler terms, when calling an API, we tell the browser that the response will take some time so it can go continuing its work and it will notify us when the response gets back.

```
function fetchData() {// the function to fetch data from the api
    fetch("https://hn.algolia.com/api/v1/search?query=redux")
      .then(res => res.json())// when the result comes back with success here is wha
      .then(res => setData(res))
      .catch(err => setError(err));// if there is an error here what you have to do.
  }
```

## The GET request with React.

If we perform a console.log to the data state, we will receive the following result:




It contains an array of objects named hits. This is what we are going to show to users.

## The GET request with React.

If we perform a console.log to the data state, we will receive the following result:

```
import React, { useEffect, useState } from "react";


const App = () => {
 const [data, setData] = useState([]); // where to store the returned data
 const [error, setError] = useState(null); // where to store the coming errors
 useEffect(() => {
   function fetchData() {
     // the function to fetch data from the api
     fetch("https://hn.algolia.com/api/v1/search?query=redux")
       .then(res => res.json()) // when the result comes back with success here is wh
       .then(res => setData(res.hits))
```

```
        .catch(err => setError(err)); // if there is an error here what you have to d
    }

    fetchData();
  }, []);

  return (
    <div>
      <ul>
        {data.map(course => (
          <li>
            <a href={course.url}> {course.title}</a>
          </li>
        ))}
      </ul>
    </div>
  );
};
export default App;
```

Output

- Guerrilla Public Service Redux (2017)
- Build Yourself a Redux
- Things to learn in React before using Redux
- Show HN: ORY Editor – A rich editor for the browser, built with React and Redux
- A SoundCloud client in React and Redux
- Systemd redux: The end of Linux
- Redux – Not Dead Yet
- Golang SSH Redux
- Redux vs. The React Context API
- Why React/Redux is inferior as a paradigm
- Redux: Atomic Flux with Hot Reloading
- Mern: Build JavaScript apps using React and Redux
- Django React/Redux Base Project
- Show HN: Redux Offline – Build Offline-First Apps for Web and React Native
- Katana – Modern Swift framework for creating iOS apps, inspired by React/Redux
- Relax – A New Generation CMS on Top of React, Redux, and GraphQL
- Call me maybe: Redis redux
- JavaScript Power Tools: Real-World Redux-Saga Patterns
- Step by Step Guide to Building React Redux Apps
- Modeling Redux with TLA+

## The GET request with React: JSON

If we paste the URL of the API in the browser, we will get a result like this one:

```
// 20200701122405
// https://hn.algolia.com/api/v1/search?query=redux

{
  "hits": [↔],
  "nbHits": 13581,
  "page": 0,
  "nbPages": 50,
  "hitsPerPage": 20,
  "exhaustiveNbHits": false,
  "query": "redux",
  "params": "advancedSyntax=true&analytics=true&analyticsTags=backend&query=redux",
  "processingTimeMS": 3
}
```


This representation is called a JSON format. That's the result the API is sending to our application.

## What's JSON?

JSON (JavaScript Object Notation) is the most widely used data format for data interchange on the web. This data interchange can happen between two computer applications at different geographical locations or running within the same machine.

The good thing is that JSON is a format that's understandable by both humans and machines. So while applications/libraries can parse the JSON data – humans can also look at data and derive meaning from it.

A JSON document may contain text, curly braces, square brackets, colons, commas, double quotes, and maybe a few other characters.

Primarily, JSON is built on two structures:

1. A collection of name/value pairs. In various languages, this is realized as an object, record, structure, dictionary, hash table, keyed list, or an associative array.

2. An ordered list of values. In most languages, this is realized as an array, vector, list, or sequence.

```
// 20200701122405
// https://hn.algolia.com/api/v1/search?query=redux

{
  "hits": [↔],
  "nbHits": 13581,
  "page": 0,
  "nbPages": 50,
  "hitsPerPage": 20,
  "exhaustiveNbHits": false,
  "query": "redux",
  "params": "advancedSyntax=true&analytics=true&analyticsTags=backend&query=redux",
  "processingTimeMS": 3
}
```


‹ Previous next ›