Fetch Lab

- Due No Due Date
- Points 1
- Submitting a website url

(https://github.com/learn-co-curriculum/phase-1-fetch-lab) (https://github.com/learn-co-curriculum/phase-1-fetch-lab/issues/new)

Learning Goals

- Use `fetch()` to programmatically make a web request

Introduction

In this lab, we'll be using `fetch()` to send web requests to the **Game of Thrones** (https://anapioficeandfire.com/) API. We'll make a web request to the API, and in return we will receive a collection of data, structured like a nested JavaScript `Object` .

What's an API?

An **API**, or application programming interface, is a manner in which companies and organizations, like Twitter or the New York City government, or the super fans behind the Game of Thrones API, expose their data and/or functionality to the public (i.e. talented programmers like yourself) for use. APIs allow us to add important data and functionality to the applications we build. You can think of an API as one way in which data is exposed to us developers for use in our own programs.

Just like we can use JavaScript to send a web request for a web page that is written in HTML, and receive a response that is full of HTML, we can use JavaScript to send a web request to an API and receive a collection of JSON in return.

What's JSON?

JSON is a language-agnostic way of formatting data. If we send a web request to the Game of Thrones API, it will return to us a JSON collection of data. With just one easy line of code, we can tell JavaScript to treat that JSON collection as a nested `Object` . In this way, large and complicated amounts of data can be shared across platforms.

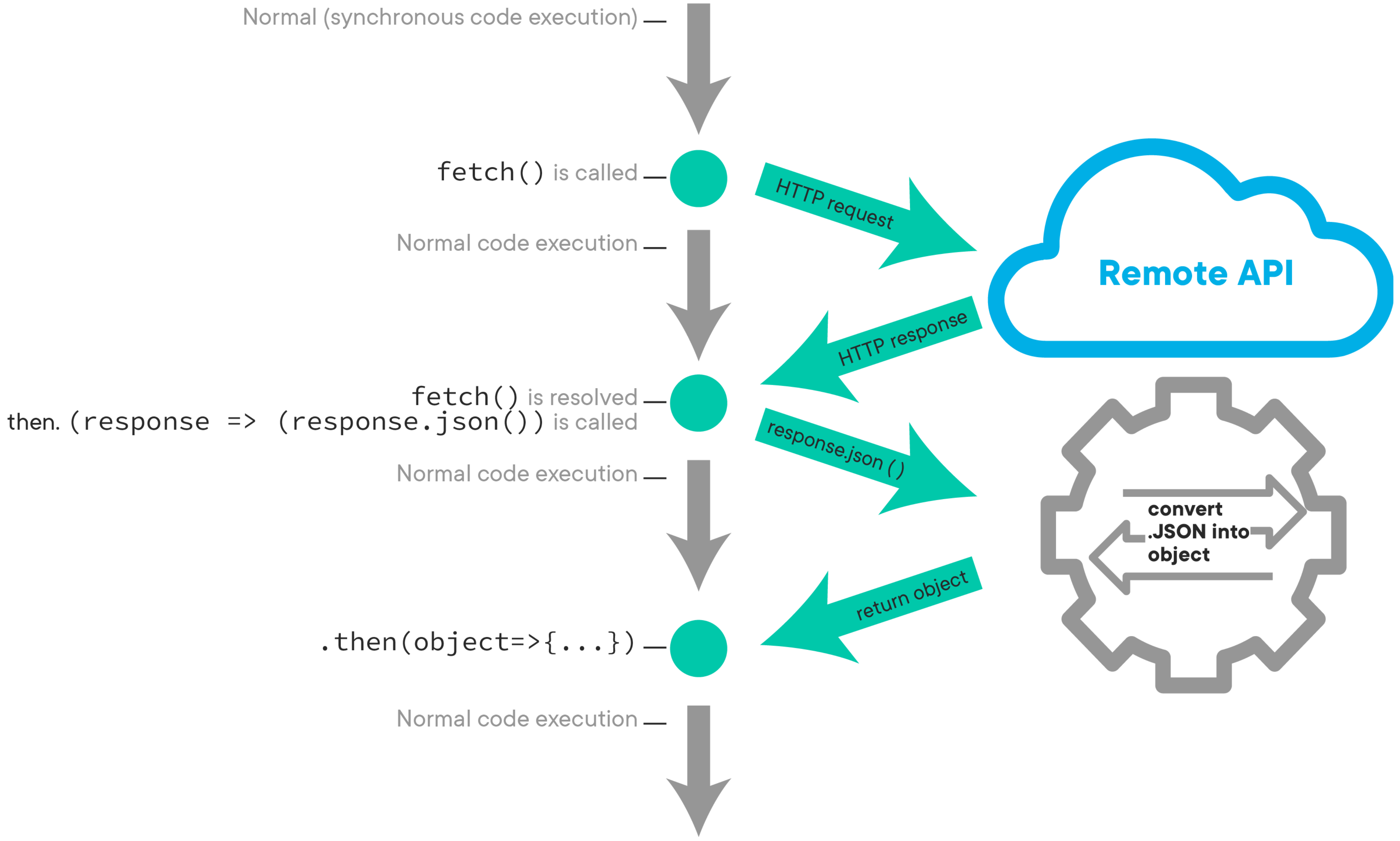
Instructions

Go ahead and navigate to the **Game of Thrones (https://anapioficeandfire.com/)** API in a separate browser tab and open DevTools. Copy the following code into the console:

```
fetch("https://anapioficeandfire.com/api/books")
  .then((resp) => resp.json())
  .then((json) => console.log(json));
```

The first line of code fetches the requested data from the API. In the second line, we use the `json()` **method (https://developer.mozilla.org/en-US/docs/Web/API/Response/json)** of the `Response` **interface (https://developer.mozilla.org/en-US/docs/Web/API/Response)** to render the API's response as plain old JavaScript object (POJO). Because we're using arrow syntax, the object is returned and passed to the *next* `then()` . Finally, in the second `then()` , the `console.log()` prints the JavaScript object to our console.

> **Note** You'll find a lot of great info on using the `fetch` method in the **MDN Using Fetch (https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch)** guide. Bookmark this page for future reference!



The response from the API contains all ten books currently existing in the Game of Thrones series, in a JSON format.

```
> fetch('https://anapioficeandfire.com/api/books')
```

Since we asked for all the books by making a query to the `/books` path, it gave us all the books. APIs have many different variations and can be as customizable as the developer wants them to be. If you're really lucky, there will be robust documentation to go along with the API that gives you a road map to help you figure out how to format your request for information. For now we'll focus on just getting different kinds of information out of the API's `/books` path.

Play around with the logged response. See if you can design a strategy to use the logged object and find the following answers.

1. The 5th book in the series
2. The 1031st character in the series
3. The total number of pages of all the books

Note that you do not need to encode these strategies to pass the lab. It's not uncommon for developers who are integrating with third-party APIs to have to do some exploration of the returned data to find the thing they're looking for. This is an opportunity for you to practice.

Deliverables

In `index.js` , there is an empty function, `fetchBooks()` , that is called when `index.html` is loaded. To pass this lab, this function should include a fetch request to the Game of Thrones API (**https://anapioficeandfire.com/api/books** (https://anapioficeandfire.com/api/books)). The returned response should be converted to JSON. Then, it should call the second function, `renderBooks()` , passing in the JSON-ified data as the argument. To check if you have done this correctly, open up the index.html page of this lab; you should see a list of Game Of Thrones titles on your webpage.

> **NOTE**: The tests in this lab need to access the `fetch()` request you will create inside `fetchBooks()` . In order to give them access, write your solution so that `fetchBooks()` *returns* the `fetch()` . This will not change the behavior of your `fetch()` .

Conclusion

APIs are powerful tools that can help you leverage the power of the available data on the web. Once you feel comfortable using the tools that access the data, the world is your oyster!

Resources

- **Game of Thrones API (https://anapioficeandfire.com/)**
- **MDN: Using Fetch (https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch)**