AJAX

Springboard

Download Demo Code

Intro

Goals

- Describe what AJAX is
- Compare AJAX requests to non-AJAX requests
- Make GET and POST AJAX requests with axios
- Use async / await to manage asynchronous code with axios
- Describe what JSON is

AJAX

Traditional Requests

Traditional browser requests happen in response to:

- Entering a URL in the browser bar
- Clicking on a link
- Submitting a form

In all cases:

- Browser makes request
- Receives response
- Replaces *entire* resource with result

```
<!-- EXAMPLE 1: SIMPLE GET REQUEST -->

<h2>Simple GET Request</h2>

<a href="/card" class="btn btn-primary">Get Card</a>

<!-- EXAMPLE 2: SIMPLE POST REQUEST -->

<h2>Simple POST Request</h2>

<form action="/borrow" method="POST">
  <input name="amount" placeholder="Amount" />
  <button class="btn btn-warning">Borrow</button>
</form>
```

AJAX

AJAX web request:

- Made from JavaScript in browser
- JavaScript makes request (***GET***, ***POST***, or other)
- You receive a response
- Do whatever you want with result!

AJAX is a technique in Javascript for sending requests and receiving responses from a server *without* having to reload the browser page.

Note: What Does AJAX stand for?

AJAX originally was an acronym for "Asynchronous Javascript and XML". However many people don't send XML over AJAX nowadays; it's more common to send HTML or JSON. The technology is still the same, though, even if the data payload is commonly different. Ultimately, AJAX is a cooler sounding acronym than AJAJ or AJAH.

Regular Request

AJAX Request

Why Use AJAX?

- Don't need to reload entire page if just 1 thing is changing
- Interactive web sites
- Fewer full page loads from server
  - Your JS can talk to other servers directly
- Less info has to go across network

AJAX with Axios

You don't *have* to use Axios for this

- There is an old, clunky built-in tool: *(XMLHttpRequest)*
- Or a newer-but-still-clunky built-in tool: *(fetch)*
- Or lots of other libraries (including *jQuery*)

... but we'll use axios for now! It's featureful & popular

Getting Axios

Can easily include it using a CDN link:

```
<script src="https://unpkg.com/axios/dist/axios.js"></script>
```

Making a Simple Request

```
axios.get(url)
```

Make a ***GET*** request to that URL

Not What We Expected

```
let card = axios.get("/api/card");
console.log(card);
// "Promise {<pending>}"
```

What's A Promise???

- We'll talk about it in more detail when we get to Node.
- For now, all you need to know is that a promise is like a placeholder for a future value.
- We want to wait for the promise to have that value before proceeding.
- But we don't know when the promise will receive its value!

Handling Asynchronous Code

Asynchronicity

AJAX requests are *asynchronous*

- The ***axios.get()*** completes *before* the response is received
- This means that if we want to use the data we get back from our AJAX requests, we need to *wait* until the response has been given to us
- We're going to use two newer keywords in JS to do this: ***async*** and ***await***!

Await

Here's what it looks like:

```
await axios.get('/api/card');

// returns response object, with `.data` as response body
```

The code is asynchronous, but it "waits" for the AJAX request to complete.

Async

When you are using Chrome DevTools in the console, you can just use ***await***. This is great for testing, but normally you will use 'await' in the context of an 'async' function.

To use in a function, you must mark that function as ***async***:

```
async function getCardInfo() {
  let response = await axios.get("/api/card");
  console.log("got", response);
  return response.data;
}
```

When calling async function, you should await it:

```
let card = await getCardInfo();
```

Callbacks Vs Async/Await

- Callbacks are what we've used for event handlers and timers
  - But they're tricky to nest or do other complex things
- ***async*/await** makes it easier to handle chains of requests
  - Modern libraries like Axios return "promises", which you await

Axios API

.get

```
axios.get(url, [config])
```

config is an optional object many Axios methods use

It hold specific configuration for what you need.

demo/templates/index.html

```
<h2>Simple GET Request</h2>

<button class="btn btn-primary"
        id="card-btn"> Get Card </button>

<div id="card" class="box"></div>

<script src="/static/card.js"></script>
```

demo/static/card.js

```
/* show ajax result directly in card box */

async function getCard() {
  let response = await axios.get(
    "/api/card");

  console.log("getCard resp=", response);
  $("#card").html(response.data);
}

$("#card-btn").on("click", getCard);
```

To make request for */resource?a=1&b=2*, can either use:

```
axios.get("/resource?a=1&b=2")
```

or

```
axios.get("/resource", {params: {a: 1, b: 2}})
```

Second form is better: you don't have to worry about how to "url safe quote" characters that aren't normally legal in URLs.

.post

Similar to ***axios.get***, but uses a ***POST*** request

```
axios.post(url, [data,] [config])

axios.post(url, {a: 1, b: 2})
```

This is passed as JSON to the server

demo/templates/index.html

```
<h2>Simple POST Request</h2>

<input id="amount" placeholder="Amount" />
<button class="btn btn-warning"
        id="borrow-btn"> Borrow </button>

<div id="borrowed" class="box"></div>

<script src="/static/borrow.js"></script>
```

demo/static/borrow.js

```
/* show result of borrowing in box */

function showBorrow(res) {
  $("#borrowed").html(res);
}

async function borrowMoney() {
  let amount = Number($("#amount").val());

  let response = await axios.post(
    "/api/borrow", { amount });

  console.log("borrow resp=", response);
  showBorrow(response.data)
}

$("#borrow-btn").on("click", borrowMoney);
```

JSON

- JSON is a string that looks like a JS object
- Most APIs use JSON to communicate
- By default, Axios recognizes JSON response & turns into JS object
- By default, Axios sends ***POST*** data as JSON

demo/templates/index.html

```
<h2>Getting JSON Responses</h2>

Get <input id="ncards" value="5" /> Cards

<button class="btn btn-primary"
        id="hand-btn">Go!</button>

<div id="hand" class="box"></div>

<script src="/static/hand.js"></script>
```

demo/static/hand.js

```
/* show result of hand in box */

function showHand(hand) {
  let $box = $("#hand");
  $box.empty();

  for (let {rank, suit} of hand) {
    let t = `${rank.| of }${suit}</p>`;
    $box.append($(t));
  }
}

async function getHand() {
  let ncards = Number($("#ncards").val());

  let response = await axios.get(
    "/api/hand", { params: { ncards } });

  console.log("getHand resp=", response);
  showHand(response.data.hand);
}

$("#hand-btn").on("click", getHand);
```

Note: Global JSON object

JavaScript comes with a global ***JSON*** object which can convert strings of JSON into JavaScript objects, and vice versa. These methods are ***JSON.stringify*** (object -> JSON) and ***JSON.parse*** (JSON -> object).

```
JSON.stringify({
  name: "Whiskey",
  favfood: "popcorn",
  birthMonth: 7
});
// '{"name": "Whiskey", "favfood": "popcorn", "birthMonth": 7}'

JSON.parse('{"name": "Whiskey", "favfood": "popcorn", "birthMonth": 7}');
// {name: 'Whiskey', favfood: 'popcorn', birthMonth: 7}
```

Note: "Form Encoded" POST requests

By default, Axios sends POST data as JSON. This is what almost all modern APIs expect.

When web browsers submit POST forms in the traditional way (ie, not using AJAX), they don't send this data in JSON — they send it in an older format, "form-encoded".

It's not common that you'd want Axios to send POST data this way. But you may be working with older APIs that expect data in this format, or you may want to work on switching over an older, non-AJAX application to an AJAX one, and find it helpful for the server to receive traditional form-encoded data. For an example of how to do so, see https://www.npmjs.com/package/axios#browser

Wrap Up

Big Ideas

- Traditional web requests:
  - Made by browser (via link, form, URL bar, etc)
  - Replace *entire page* with thing linked to
- AJAX requests:
  - Made via JS AJAX calls
  - JS get data, JS decides what to do with it
- Axios is the popular AJAX client we'll use
- AJAX calls are asynchronous & return a "promise"
  - You need to ***await*** those to get real results
  - Functions that use ***await*** must be ***async***
- JSON
  - Axios parses JSON responses automatically for us

Axios Docs

https://www.npmjs.com/package/axios