Working with Data: the Fetch API

Working with asynchronous functions

Announcements

1. Midpoint project deliverable due tonight
2. We will be presenting in small groups on Wednesday during class.
    a. Attendance mandatory to get credit for the presentation portion of the assignment.
3. Next up:
    a. Friday's Tutorial: getting started on HW7 – graded on attendance / participation.
    b. Wednesday: Presentations + Quiz 3 / Final Exam review (start studying now)

Today’s Agenda

1. Finish the exercises we started last class (with the UNCA course search API).
2. Introduce some of the syntax / concepts of JavaScript’s built-in **fetch** function.

Today’s Agenda

1. **Finish the exercises we started last class** (with the UNCA course search API).
2. Introduce some of the syntax / concepts of JavaScript’s built-in **fetch** function.

Today’ Activity: https://t.ly/4IQU

1. Output the title and the instructor of the first course to the #results section (did on Wednesday).
2. Output ALL of the course titles to the console.
3. Output ALL of the course titles to the #results section.
4. Output ALL of the diversity intensive courses to the results section.
5. Questions to ponder:
    a. How could you make this interface more useful and engaging?
    b. How could you allow your user to…
        i. Select which department they want to view?
        ii. Select which term they want to view?
        iii. Only view classes that meet on Tuesdays and Thursdays?

Quiz

1. Looping through a list of data and displaying every element on the screen.
2. Using a template to display a combination of HTML tags and expressions (data fields).
3. Using an if statement (condition) to only show some of the data (not all of it).
4. Reaching into the DOM and getting something that a user typed into the textbox.

Today’s Agenda

1. Finish the exercises we started last class (with the UNCA course search API).
2. **Intro to fetch**

Intro to AJAX

- **AJAX:** Stands for Asynchronous JavaScript and XML
- Enables JavaScript to make server requests and (optionally) update the current screen
- Not easy to tell that information is even being transmitted to/from a server
- Came on the scene ~2004 (made popular w/Google Mail and Google Maps)

Intro to Fetch

JavaScript's Fetch API

- The Fetch API is a newer instantiation of **asynchronous** server-client web communication
  - Because you are making a request over the network, the code has to wait until it gets a response back from the server before it continues its execution flow.
- Provides an interface for fetching resources (including across the network). The new API provides a more powerful and flexible feature set (improving upon AJAX)

Source: https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API

In other words...

If you want to get data from a server and use it to build part of your web page, use **fetch**.

Fetch Example

```
async function fetchCourses() {
   const url = `https://some-endpoint.com`;
   const data = await fetch(url).then(response => response.json());
   displayResults(data);
}
```

Notes:

- **async / await** keywords go together.
- **await** is needed because you don't want to execute the displayResults() function before the data comes back from the server.
- **displayResults(data)** is responsible for displaying the data to the screen in an interesting way.