WEB FRONTEND DEVELOPMENT II

WDD 330

Ponder Team Activity: Working with a remote API

Overview

This activity will introduce you to working with a remote API. We will use Fetch with promises to do this here. These are both new additions to Javascript with ES2015. This could also be done more traditionally with an XMLHTTPRequest and callbacks. The API we will be consuming is SWAPI: Star Wars API. We will pull data in batches, then build a pagination UI to display it.

> FYI. If Star Wars is not your thing you could also complete this activity with the PokeAPI 

Instructions

Complete the following assignment as a team. Designate one team member as the "main driver" and collaborate on their copy of the code. Everyone on the team should be actively engaged in writing the code and contributing to the solution. Once the solution is working, make sure that everyone on the team gets a copy of the code. Each week let someone else be the "main driver" of the coding.

Core Requirements

Take a look at the documentation for the api first. It gives good examples of how to make calls to each endpoint, and shows the structure of the data.

Create a basic application and use **`fetch`** to pull a list of people or ships. Display this list in the browser window.

> Remember that **`fetch`** returns a Promise, **not** the data! Also remember that we will need to tell **`fetch`** what type of data we are expecting: **`.text()`**, **`.json()`**, or **`.blob()`** so it can process it accordingly.
>
> We can resolve(process) a Promise in one of two ways: either with a **`.then()`**, or with **`async/await`**. (MDN: Using Promises)

If you made a request to **`https://swapi.dev/api/people`**, for example, it will tell you that there are 87 different people in the database...but it will only send you 10 of them! It is sending back the data in such a way to facilitate viewing it in small chunks...paginated. It also gives a **`next`** url.

Use the **`next`** and **`prev`** urls to build a UI with next and prev buttons to move back and forth between pages of results.

Style it.

Consider your target...a small mobile screen. What will be the best way to present the items returned? Style your paginated list to work well on a mobile screen.

Stretch Goals

Get the details

When we touch an item in your list (people, ships, etc) it would be good if it showed us the full details for that item. Make it so.

Skip to page

Add a button to your paginated list for each page in the results set that will let you jump straight to that page. For example if the results set had 57 items in it...and you got 10 at a time, you would have 6 pages of results.

Instructors Solution

As a part of this team activity, you are expected to look over a solution from the instructor, to compare your approach to that one. One of the questions on the I-Learn submission will ask you to provide insights from this comparison.

Please DO NOT open the solution until you have worked through this activity as a team for the one hour period. At the end of the hour, if you are still struggling with some of the core requirements, you are welcome to view the instructor's solution and use it to help you complete your own code. Even if you use the instructor's code to help you, you are welcome to report that you finished the core requirements, if you code them up yourself.

After working with your team for the one hour activity, click here for the instructor's solution.

Submission

When you have finished this activity, please fill out the assessment in I-Learn. You are welcome to complete any additional parts of this activity by yourself or with others after your meeting before submitting the assessment.