Project 2: List Pagination & Filtering Study Guide

Sections of this Guide:

- **How to approach this project** includes detailed guidance to help you think about how to organize your code, project and files.
- **How to succeed at this project** lists the grading requirements for the project, with hints, links to course videos to refresh your memory and helpful resources.

How to Approach This project

When beginning a project like this, and looking at all that needs to be done, it's easy to feel overwhelmed. Instead of thinking about this as one big project, think of it as several smaller parts. When you break down a programming problem into smaller tasks, you can concentrate on getting each task done.

And don't let the title of this project — ***List Pagination and Filtering*** — scare you. Sounds like a complicated process, but really what you are doing is **hiding** and **showing HTML** on a page, using a set of buttons that you add to the page dynamically. In other words, this is **basic DOM Manipulation** — something you've been learning throughout this Unit.

Not so bad, after all, right? OK. Let's break this project down further. The two main parts of this project that you need to complete to get a **Meets Expectations** grade require you to *add buttons to the bottom of the page* and *show different sets of student information when each button is clicked*.

One approach then is to create two functions, one that handles each of those tasks. Here are some ideas for how to create those:

The showPage Function

To hide all students and show only a particular set of ten, we could create a function that takes in parameters for the list (all of the students) it's supposed to work on, and the "page" that's supposed to be shown. A "page" here just means a list of 10 students: so the first 1-10 students would be "page 1", students 11-20 appear on "page 2", and so on.

```
const showPage = (list, page) => {
   /*
   Loop over items in the list parameter
      -- If the index of a list item is >= the index of the first
      item that should be shown on the page
      -- && the list item index is <= the index of the last item
      that should be shown on the page, show it
   */
 }
```

The appendPageLinks Function

Then we could use a function that creates all the pagination buttons, adds them to the DOM, and adds their functionality. So you would see a button with the number 1 which, when clicked, would show the first ten students (the first "page"). When each link is clicked, the showPage function displays the corresponding page (set of ten students), and highlights that page's link. For example, clicking the link to page 2 will display students 11 through 20 and highlight button 2.

```
const appendPageLinks = (list) => {
  /*
      1. Determine how many pages are needed for the list by dividing the
      total number of list items by the max number of items per page
      2. Create a div, give it the “pagination” class, and append it to the .page div
      3. Add a ul to the “pagination” div to store the pagination links
      4. for every page, add li and a tags with the page number text
      5. Add an event listener to each a tag. When they are clicked
          call the showPage function to display the appropriate page
      6. Loop over pagination links to remove active class from all links
      7. Add the active class to the link that was just clicked. You can identify that
        clicked link using event.target
  */
}
```

How to succeed at this project

Here are the things you need to do pass this project. Make sure you complete them ***before*** you turn in your project.

Progressive enhancement & unobtrusive JavaScript

- ❏ **No inline JavaScript. All JavaScript is linked from an external file.**
  - ❏ *Related video:* **Where Does JavaScript Go?**
- ❏ **Use unobtrusive JavaScript to append markup for the pagination links.**
  - ❏ *Related video:* **Understanding Unobtrusive JavaScript**
  - ❏ *Related video:* **Appending Nodes (DOM API)**

Pagination Links

- ❏ **Pagination links are created. If there are 44 students, 5 links should be generated, if there's 66, 7 links should be generated.**
  - ❏ *Related video:* **Listening for Events with addEventListener() (DOM API)**
  - ❏ *Related video:* **Creating New DOM Elements (DOM API)**
  - ❏ *Related video:* **Removing Nodes (DOM API)**
  - ❏ *Related video:* **Practice Manipulating the DOM**

Paging

- ❏ **The first 10 students are shown when the page loads, and each pagination link displays the correct students.**
  - ❏ *Related video:* **Styling Elements (DOM API)**
- ❏ **Clicking on "1" in the pagination links should shows students 1 to 10. Clicking "2" shows 11 to 20. Clicking "5" shows students 41 to 50, and so on.**
  - ❏ *Related instruction:* **The showPage Function**
  - ❏ *Related practice:* **Practice Basic JavaScript Functions**