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. So try not to see the whole project all at once. Try not to get too hung up on all that needs to be done. Instead, break the project down into small digestible chunks and individual tasks.

From a programming perspective, there are many ways to approach this project. You could use a very stateful approach and just create a bunch of global statements that change the DOM as needed. But this type of programming is widely discouraged as it tends to be harder to read, less maintainable and more error prone than alternatives.

You could use an OOP (Object Oriented Programming) approach, and create objects with properties and custom methods. But this project is more about functionality than managing a bunch of complex state, so OOP is probably not the best bet here.

Given that our JavaScript is meant to be unobtrusive, utilize the concepts of progressive enhancement, and work with any list of any size, a pure, stateless, functional approach seems like the way to go.

This means that instead of creating a bunch of statements that interact directly with this particular DOM, we want to create pure, stateless functions that take in parameters, perform operations on those parameters, and then have the actual elements passed in as arguments when we call or "invoke" these functions.

Already Read

Need help? Visit the [Project 2 Slack Channel]

The showPage Function

I have done this section

For example, to hide all students and show only a particular set of ten, we could create a function that takes in parameters for the list it's supposed to work on and the page that's supposed to be shown.

```
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
      // else hide it
}
```

The appendPageLinks Function

Then we could use a function that creates all the pagination links, adds them to the DOM, and adds their functionality. When each link is clicked, we'll use the showPage function to display the corresponding page, and mark the active link. For example, clicking the link to page 2 will display students 11 through 20.

```
const appendPageLinks = (list) => {
   // if pagination already exists, remove it    ???
   // 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    done
   // create a div, give it the "pagination" class, and append it to the .page div    done
   // add a ul to the "pagination" div    done
   // for every page
      // add li and a tags with the page number text
   // add an event listener to each a tag, or add an event listener to the pagination div, and use event delegation to target the a tags to define what happens they are clicked
      // calls the showPage function to display the appropriate page
      // loop over pagination links to remove active class from all
      // add the active class to the link that was just clicked, otherwise known as the event.target
}
```

Need help? Visit the Project 2 Slack Channel

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)**
  - ❏ *Related video:* **Adding New Elements to the DOM (using jQuery)**

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 instruction:* **The appendPageLinks Function**
  - ❏ *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:* **Adding New Elements to the DOM (jQuery)**
  - ❏ *Related video:* **Using on() for Event Handling (jQuery)**
  - ❏ *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)**
  - ❏ *Related video:* **Animating Elements with jQuery (jQuery)**
- ❏ **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**

Need help? Visit the Project 2 Slack Channel