6.0 hours
Project Status Passed

Project 2

Pagination & Content Filter

Exceeds Expectations

Your project was reviewed and looks great! Take a look over your grade details to see how you can improve.

Keep Going

- Instructions
- How you'll be graded
- Your Grade

In this project, you'll enhance the usability of a web page that has too much information. To make it easier to find information and read the page, you'll use progressive enhancement to add pagination.

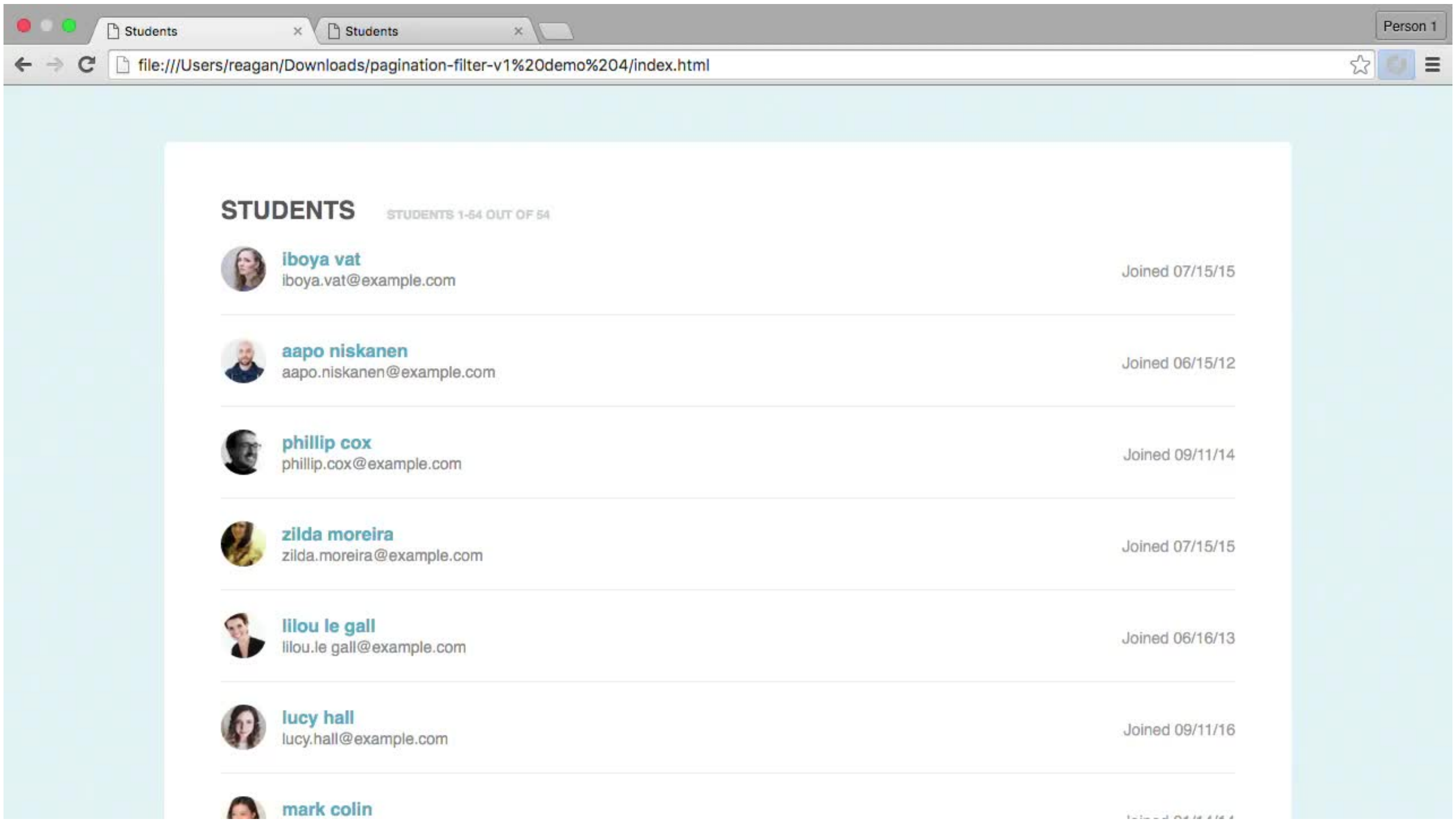
The page includes a list of 54 students. You need to add programming to display 10 students per page, and add buttons to jump between the lists of 10 students.

Your solution should work for any number of student listings, whether there were 5, 25, 55 or 100 student listings.

Optionally, you can add a search bar to let users search the list of students for a particular student or students.

Feel free to use jQuery for this project, but please avoid using plugins to achieve the pagination and/or search functionality.

Video Player



file:///Users/reagan/Downloads/pagination-filter-v1%20demo%204/index.html

STUDENTS STUDENTS 1-54 OUT OF 54

iboya vat
iboya.vat@example.com
Joined 07/15/15

aapo niskanen
aapo.niskanen@example.com
Joined 06/15/12

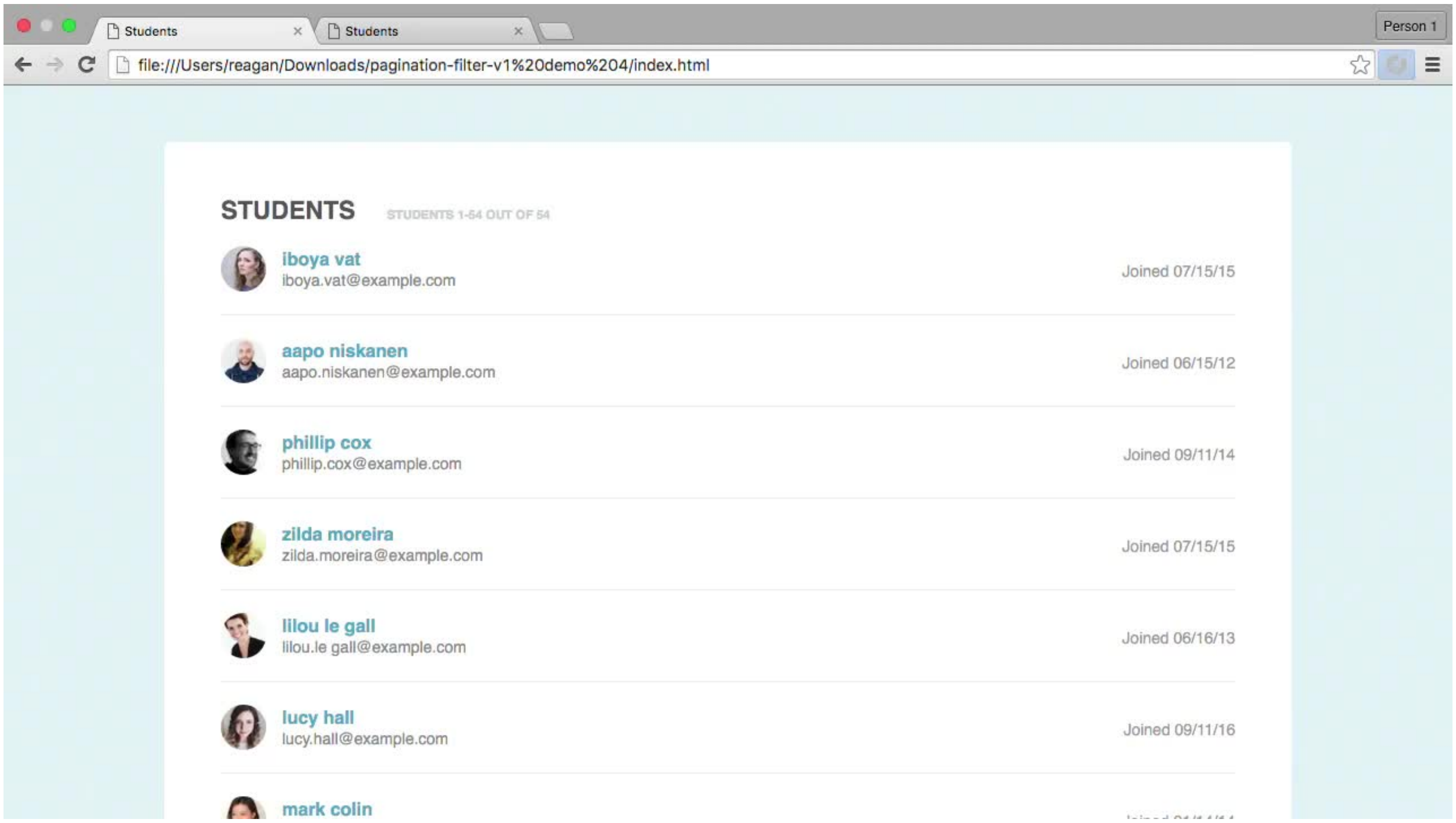
phillip cox
phillip.cox@example.com
Joined 09/11/14

zilda moreira
zilda.moreira@example.com
Joined 07/15/15

lilou le gall
lilou.le gall@example.com
Joined 06/16/13

lucy hall
lucy.hall@example.com
Joined 09/11/16

mark colin


file:///Users/reagan/Downloads/pagination-filter-v1%20demo%204/index.html

STUDENTS STUDENTS 1-54 OUT OF 54

iboya vat
iboya.vat@example.com
Joined 07/15/15

aapo niskanen
aapo.niskanen@example.com
Joined 06/15/12

phillip cox
phillip.cox@example.com
Joined 09/11/14

zilda moreira
zilda.moreira@example.com
Joined 07/15/15

lilou le gall
lilou.le gall@example.com
Joined 06/16/13

lucy hall
lucy.hall@example.com
Joined 09/11/16

mark colin

00:00
00:00
00:00

1. 2x 2x

2. 1.75x 1.75x
3. 1.5x 1.5x
4. 1.25x 1.25x
5. 1.1x 1.1x
6. 1x 1x
7. 0.75x 0.75x
8. 0.5x 0.5x

- ● None
- ○ English

Use Up/Down Arrow keys to increase or decrease volume.

Before you start

To prepare for this project you'll need to make sure you complete and understand these steps.

4 steps

- **Have a GitHub account and know how to create a new repository and upload files to it. As with the previous projects, you'll submit your finished working using GitHub.**

- **If you need a reminder on how to use GitHub and GitHub desktop to create a new repository check out the workshop 'Share Your Projects wIth GitHub Desktop' in the Project Resources.**

- **Download the project files. We've supplied several files for you to use:**

    - 2 style sheets, a reset.css that sets some basic CSS styles for cross-browser support and design.css that contains CSS specific to this project.
    - 3 HTML files, index.html, example-meets.html and example-exceeds.html. The index.html file contains the initial HTML page — you'll use JavaScript to modify the markup on this page to meet the goals of this project (as described in the instructions below) The two example files demonstrate what the page should look with the pagination in place and with the optional search box in place.
    - A `student-list-examples` folder which contains two other pages with differing amounts of listings. You don't need to use these file but they can help you test that your programming works no matter how many students listings are on the page.
    - You will need to create your own external JavaScript file.

- **Have a basic understanding of Progressive Enhancement. See the Resources links on this page to find out more about Progressive Enhancement. For this project, there are two important Progressive Enhancement principles to keep in mind:**

- **Basic content should be accessible to browsers that don't support JavaScript.** In other words, the `index.html` file should display the entire list of students for browsers that don't support JavaScript.
- **Unobstrusive JavaScript:** this means that any content or functionality related to JavaScript should be added programmatically by JavaScript. For this project, that means, the pagination buttons and the (optional) search shouldn't be added directly to the `index.html` file. You need to use JavaScript to add them.

Project Instructions

To complete this project, follow the instructions below. If you get stuck, ask a question in the community.

7 steps

- **When the page loads, your program should hide all but the first 10 students in the list.**

- **Look at the HTML in the `example-meets.html` on lines 119-137 -- this is an example of the markup you'll need to add dynamically to the index.html page to create pagination links.**

- **Since only 10 students should be shown at a time, your programming needs to calculate the number of pages needed and add the appropriate number of links to the bottom of the page.**

- **When a user clicks on “2” in the pagination, students 11 through 20 are shown. When a user clicks “3”, students 21 through 30 are shown. And so on. When “6” is clicked 51 through 55 should be shown.**

- **Your program should work for any number of students. There are 54 students in index.html, but you can test your code by adding the JavaScript file your write to the other lists of students we’ve provided in the `student-list-examples` folder.**

- **If you would like some extra guidance on how to do this project, read Pagination Project Guide listed in the project resources for some helpful hints!**

- **Before you submit your project for review, make sure you can check off all of the items on the Student Project Submission Checklist. The checklist is designed to help you make sure you’ve met the grading requirements and that your project is complete and ready to be submitted!**

NOTE: It’s good practice to check your project for cross browser compatibility. We recommend making sure your project looks and functions as expected in at least 3 different browsers.

Extra Credit

To get an "exceeds" rating, you can expand on the project in the following ways:

3 steps

- **Include a search component so that a user could search for a particular student or students. See the file `example-exceeds.html` and lines 16-19 for what the markup for the search component should look like.**

- **When the "Search" button is clicked, the list of students is filtered to match the search. For example if the name `Phillip` is typed into the box list all students whose name or email includes Phillip.**

- **If no matches are found by the search, include a message in the HTML to tell the user there are no matches.**

NOTE:

- To get an "Exceeds Expectations" grade for this project, you'll need to complete **each** of the items in this section. See the rubric in the "**How You'll Be Graded**" tab above for details on how you'll be graded.
- If you're shooting for the "Exceeds Expectations" grade, it is recommended that you mention so in your submission notes.
- Passing grades are final. If you try for the "Exceeds Expectations" grade, but miss an item and receive a "Meets Expectations" grade, you won't get a second chance. Exceptions can be made for items that have been misgraded in review.

Download files

Zip file

Project Resources

External Link

What is Progressive Enhancement and Why Should You Care?

External Link

Core Principles of Progressive Enhancement

Video

What is Unobtrusive JavaScript?

External Link

Unobtrusive JavaScript

External Link

Pagination Project Guide

Need Help?

Have questions about this project? Start a discussion with the community and Treehouse staff.

Get Help