CI Playground

Nested Spec test: Project 0.5

Due: ~~8pm, Apr 1, 2019~~ **Anytime!** *This is an individual project.*

Spam spam spam.

Introduction

In Computer Science courses, project specifications tend to get really long. The intention for such a long document is good — they usually provide all the information that a student would need to successfully complete the project. However, this can also make it difficult for students to find information quickly — they often get lost in the document, and cannot easily find the information they need on the fly.

Infinite scroll…

In Winter 2019, a student wrote on the EECS 485 Piazza:

> I think it would be very useful to have a table of contents in a sidebar for project specs to make navigating the document easier. I often forget where specific sections are since the page is so long.

This makes sense! Let's solve the problem of long specs by adding a sidebar. We'll do this by writing JavaScript code that can be "plugged in" to the project spec webpages. The JavaScript code we write will then generate a sidebar with a list of all the headings in the webpage.

(You're going to write JavaScript code similar to the one that powers the sidebar on this page[1]!)

Learning Goals

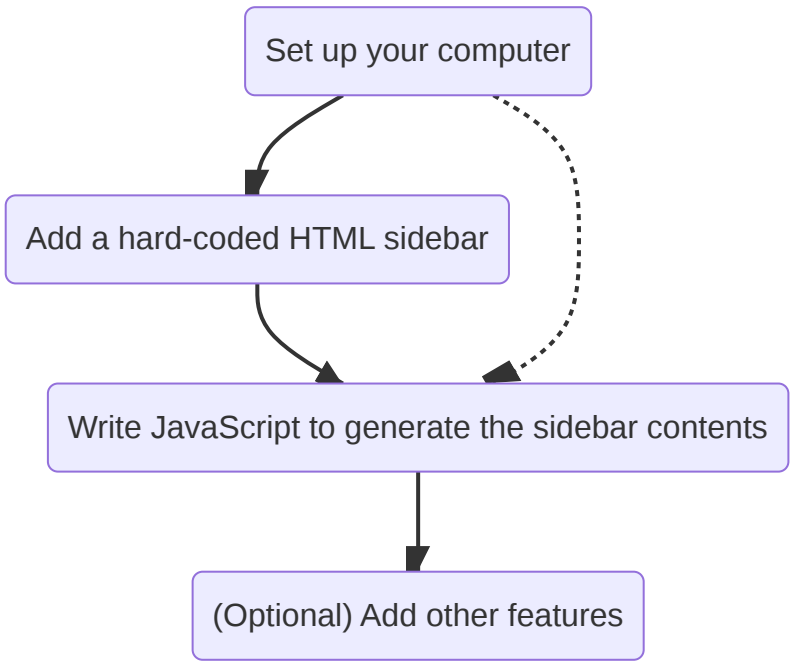
This project is meant to be open-ended — it's not autograded! At the end of this project, you should feel comfortable doing the following:

- [ ] Use HTML and CSS to modify the design of a web page
- [ ] Use JQuery and JavaScript to dynamically add content to the page

If you wish, you can implement the other features of the official Primer Spec, as described in the optional Other Features section.

Contents

This spec will walk you through these parts:



Setup

Follow these steps to get your development environment set up.

Install the dependencies

First make sure that you have Python 3 installed. If you're on MacOS, you may have to `brew install python3` first.

```
$ python3 --version  # NOTE: Your Python version may be different.
Python 3.7.4
```

Download the starter files

Download the starter files and untar them.

```
$ pwd
/users/seshrs
$ wget https://eecs485staff.github.io/primer-spec/demo/starter_files.tar.gz
$ tar -xvzf starter_files.tar.gz
```

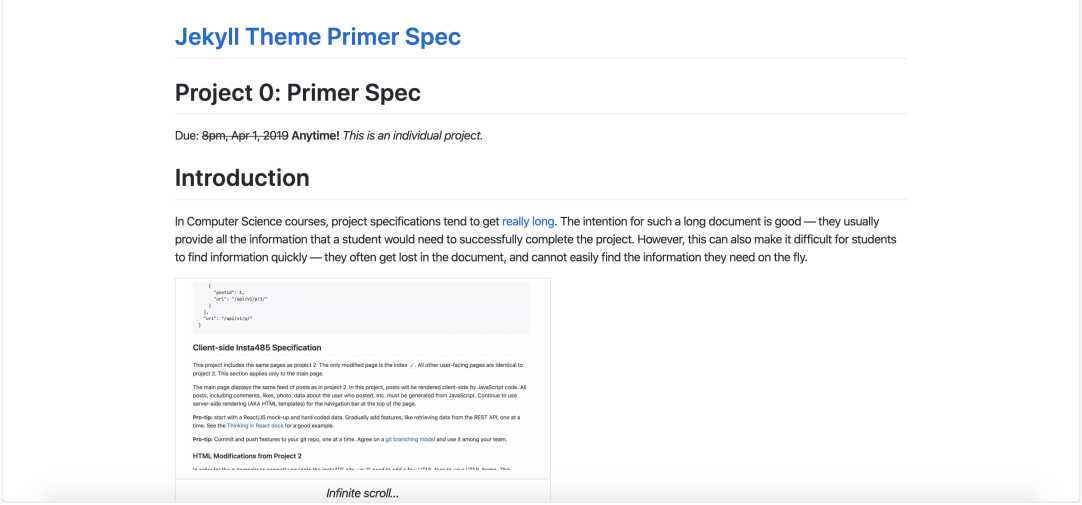
Preview the site on your browser

Have a look at `index.html` by opening the file in your favorite text editor. This file holds all the HTML that represents this project spec. Notice how we've already included some CSS to make the spec look like GitHub's Primer theme.

Use Python's built-in static fileserver to view this HTML file on your browser. Open a console at the directory of the starter files, and then run the Python server.

```
$ pwd
/users/seshrs/primer-spec-spec
$ python3 -m http.server
```

Visit http://localhost:8000 on your web browser. The page should look similar to this screenshot:

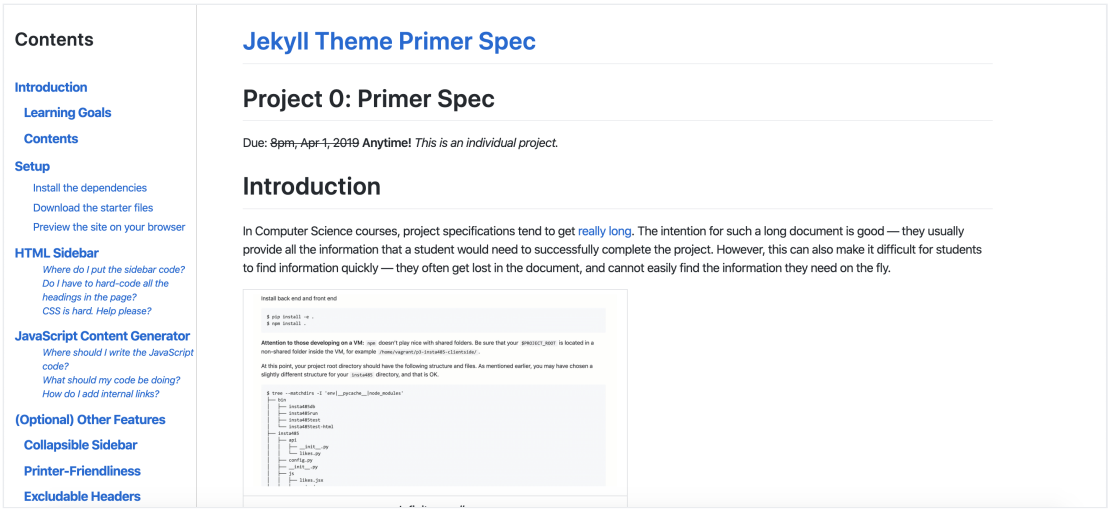


The spec already looks pretty good, but it could certainly be improved with a sidebar.

HTML Sidebar

In this section, you will add a sidebar to the HTML page and hard-code its contents. Although we'll be writing some JavaScript code later in the project to generate the contents of the sidebar, it's still a good idea to decide how it's going to look.

When you're done with this section, your webpage will have a sidebar on the left, something like this screenshot:



Of course, *your* finished webpage doesn't have to look like this. After all, this project isn't autograded! (In fact, feel free to showcase your project spec design with us! Create an issue on our GitHub repository with a screenshot of your design.)

However, if you wish to emulate our design, here are some tips.

Where do I put the sidebar code?

We added our sidebar HTML at the very top of the `body` tag, just before the the main content begins. (The main content is contained in a `div` with class `markdown-body`.)

Our sidebar is `fixed` to the page with a constant width (we used `18em`), height and padding. You may need to push the main content a bit to make room for your sidebar. You should also consider what would happen if there were a lot of items in the sidebar.

We recommend placing any CSS style definitions you need in `spec.css`.

Do I have to hard-code *all* the headings in the page?

No, but it's a good idea to hard-code at least some of them, and randomly hard-code some other items in the sidebar. In particular, make sure to have a mix of heading indentations in your sidebar — consider styling each header level differently.

When you style the various header levels, remember that you'll be writing JavaScript code soon that will generate all of this HTML inside the sidebar. (Hint: Use CSS classes to style the header levels differently. Consider using the *name* of the HTML header tag in the class name.)

CSS is hard. Help please?

Yeah, it has a bit of a learning curve. Google, StackOverflow and MDN are your best friends. Experiment with different CSS style attributes to see what works best.

Pro tip: *Use* `div` *(unstylyed containers) and classes liberally. They make it easy to apply styles to collections of elements.*

Pro tip: *Use your browser's "developer tools" to modify CSS styles on your page to dynamically see the effects of your changes.*

`JavaScript` Content Generator

The spec page looks great! But if you hard-coded all the headings on the page, you *know* how much time it takes to hard-code the sidebar contents. Imagine having to do that for several different project specs in a course website! It would be nice to *automatically* generate the sidebar contents for any web page. Let's use JavaScript to write this content generator.

(If you didn't hard-code all the headings on the page, then your sidebar is still incomplete, so you need to write this JavaScript generator anyway.)

Your generator will vary depending on your HTML sidebar design. Simple sidebar designs (for instance, just a list of all the headers with no differentiation) will require simpler generators. The sidebar we use in the official Primer Spec is more complex because we group "child sections" within divs to add further group styles.

Here are some suggestions for getting started.

Where should I write the `JavaScript` code?

Write your code in `spec.js`. This code is loaded and executed at the very end, after all other content and scripts are loaded and executed. As a result, your code in `spec.js` will have access to JQuery APIs.

JQuery APIs let you find HTML elements using CSS selectors and easily manipulate them. For example:

```
1  $('primer-spec-sidebar').append('<h2>Contents</h2>');
2  $('primer-spec-topbar').toggleClass('primer-spec-sidebar-shown');
```

What should my code be doing?

The end goal is to generate a list of headings on the page, and place that in the sidebar. At its simplest:

1. Find all "heading HTML elements" on the page. Remember to look for all the different header levels, from `h1` through `h6`.

2. Create a new HTML element for each “heading” you find. This HTML element could be a list item, or simply a text paragraph.
3. Find the sidebar (or the place where you want to display the “table of contents”). Insert the HTML elements that were created in step 2.

How do I add internal links?

While you’re traversing the list of “heading elements” on the page, notice how they all have an `id` attribute. If you were to create an HTML anchor (`a`) with `href` set to `#{insert-id-here}`, the anchor would become an internal link. (Clicking the internal link will scroll the page to that section.)

(Optional) Other Features

If you’ve completed the rest of this spec, congratulations! You’ve just built the core foundation of the official Primer Spec.

Primer Spec has several other features that you could consider implementing in your own project. These reach goals are of varying difficulty, but could still be fun to work on. Choose your own challenge!

Here are some of the optional features you could implement, in somewhat increasing order of difficulty:

- Collapsible Sidebar
- Printer-Friendliness
- Excludable Headers
- Currently-Viewing Section
- Mobile-Friendliness
- LaTeX Support

Collapsible Sidebar

For some viewers, the sidebar can be visually distracting. It would be nice to allow viewers to hide the sidebar. This can be achieved using something as simple as a toggle button. However, consider emulating a better user experience similar to that of the official Primer Spec theme — using a hamburger icon on the top-left corner.

One other thing to consider: Is simply hiding the sidebar enough? While styling the sidebar, if you added CSS styles to push the main content to make room for the sidebar, then the content will look off-centered if the sidebar is simply hidden.

Printer-Friendliness

Many students like to print project specifications for easy offline reference. Try printing your project web page — how do you like its look?

The solution is to simply add a JavaScript event handler when the page is printed. This handler should collapse the sidebar and recenter the main content, as described in the Collapsible Sidebar section. For a better user experience, restore the sidebar to its original state after the page has been printed.

Excludable Headers

Sometimes, it is useful to exclude certain headers from the sidebar for aesthetic reasons. For example, on an official Primer Spec page:

This header will show in the sidebar

But this header will not

The above header element is excluded from the sidebar because it has the class `primer-spec-toc-ignore`. Modify your JavaScript Content Generator to skip adding a section to the sidebar if the corresponding header element has this class attribute.

Currently-Viewing Section

On an official Primer Spec page, viewers can see the section to which they have currently scrolled — the header in the sidebar is highlighted.

To implement this feature, you will need to add a CSS class to indicate a “highlighted” section in the sidebar, and will then have to write JavaScript to apply the “highlighted” class to the correct section in the sidebar. Here’s a high level description of what the JavaScript code should do:

- Add an event listener to scroll events. (After all, the “highlighted” section changes only when the viewer scrolls the page.)
- Use this algorithm to find the currently highlighted section.
  - Iterate over all the headings on the page.
  - First reset the sidebar by removing any occurrences of the “highlighted” class.
  - If the “top” of the heading element is above the “distance scrolled by the user”, then “highlight” the corresponding sidebar section.
  - Stop iterating when remaining header elements are “below” the “distance scrolled by the user”.

It might be easier to implement this using JQuery APIs than plain JavaScript.

Pro tip: *If you implemented the "Exclude Section" feature above, you'll have to take that into accound when highlighting the sidebar sections.*

Mobile-Friendliness

Sometimes, students and staff access project specifications from their mobile devices for quick reference. Try accessing your site from a mobile device. (One way to do this is to use your browser's "developer tools" to simulate visiting your project webpage from a mobile device.)

Their are two primary problems with the current mobile user experience:

1. The sidebar pushes the main content into a long column of single characters.
2. To collapse the sidebar, the viewer must scroll to the toggle button.

There are multiple ways to solve these problems. The official Primer Spec theme addresses them by doing the following *only on mobile devices*:

1. The sidebar is overlaid *on top* of the content, instead of pushing it.
2. The sidebar is collapsed by default. A toggle button is always shown on top. Clicking a sidebar link, or clicking outside the sidebar, collapses the sidebar immediately.

As you go along, you may find that other changes are needed to improve the mobile experience. For instance, since the sidebar collapses whenever a link is clicked, users no longer expect these internal links to accumulate browser history. (Clicking the "back button" should not just take them to a different section.) Continue exploring your project page from a mobile device to discover other user experience issues.

LaTeX Support

Some project specs need to show Math formulae and equations. It's possible to include these as inline images, but it's hard to maintain. Consider adding support for LaTeX in specs using MathJax.

LaTeX can be inserted in the middle of paragraphs ($\forall x \in R$), or as separate Math blocks:

$$\phi(x,y) = \phi\left(\sum_{i=1}^{n} x_i e_i, \sum_{j=1}^{n} y_j e_j\right) = \sum_{i=1}^{n}\sum_{j=1}^{n} x_i y_j \phi(e_i, e_j) =$$
$$(x_1, \ldots, x_n) \begin{pmatrix} \phi(e_1, e_1) & \cdots & \phi(e_1, e_n) \\ \vdots & \ddots & \vdots \\ \phi(e_n, e_1) & \cdots & \phi(e_n, e_n) \end{pmatrix} \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$$

Feedback

If you have suggestions for improving this spec/demo, please create an issue on the Primer Spec GitHub repository. Alternatively, if you know exactly what you'd like to change, fork the repository and create a Pull Request!

Acknowledgmenets

This project spec was originally written to demonstrate the Primer Spec theme by Sesh Sadasivam in 2019.

The structure and style of the project was strongly influenced by EECS 485 project specifications written by Andrew DeOrio.

---

1. This "project 0" spec was written in 2019, and Primer Spec has evolved *a lot* since then. The JavaScript code powering the sidebar actually looks very different than what the starter files indicate.[2] ↩

2. Did you know that you can write *footnotes* in Markdown?! (Documentation) ↩