COMP 3612
Assignment #2: Single-Page App

Version: 1.1 Oct. 27, 2025, changes in yellow
Due Saturday November 22, 2025 at midnightish

Overview

This assignment provides an opportunity for you to demonstrate your ability to work with JavaScript. The application you will be creating is a Single-Page Application to view products.

You can work in pairs or by yourself. Don't delay finding a partner; if you cannot find a partner to work with, **don't ask me to be in a group of 3**, you will have to work alone. The assignment is very doable by one person, but it will be a more pleasant experience sharing the work with another.

Beginning

It is your responsibility to read all the assignment instructions thoroughly and carefully. If you are unclear about something, ask me. But before you do, read the material in question again!

Grading

The grade for this assignment will be broken down as follows:

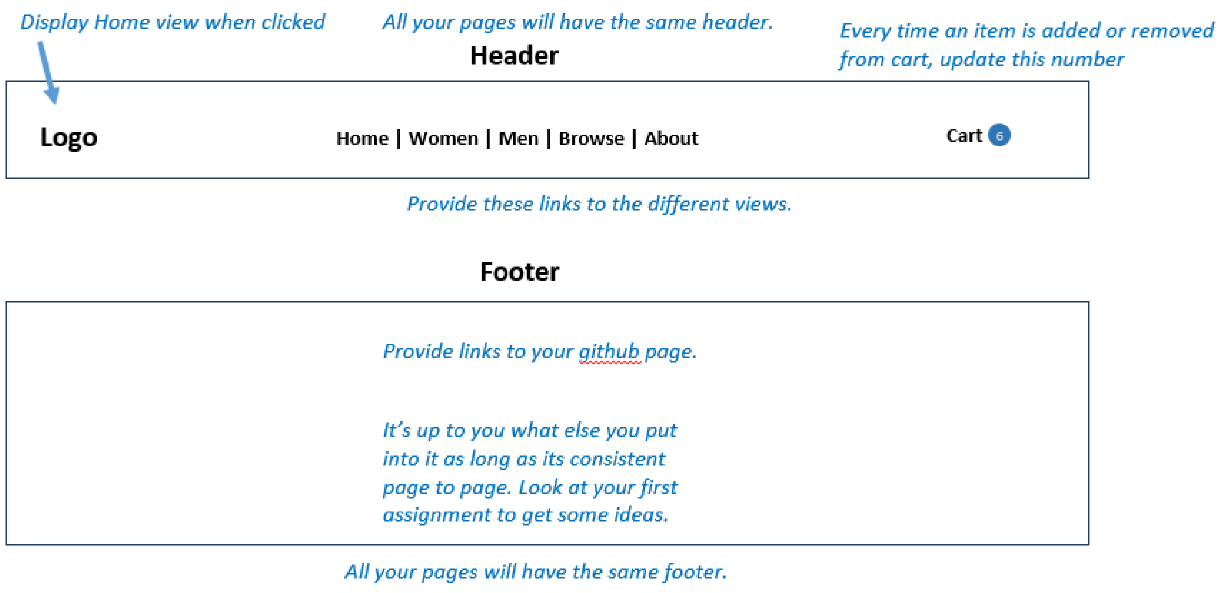
Visual Design	15%
Programming Design and Documentation	10%
Functionality (follows requirements)	75%

Requirements

This assignment consists of a **single** HTML page (hence single-page application or SPA) with the following functionality:

1. **You can use a third-party CSS library/framework (indeed I would encourage you to use one).** Tailwind CSS is a great thing to have on your resume, but it does have a bit of a learning curve (Lab7b in your gumroad package can help you quickly learn its essentials). Be careful though that your CSS library is not using/requiring its own JavaScript library as well. Some popular CSS libraries (e.g., Bootstrap) include their own JavaScript libraries to do fancy things with select lists, modals, etc. **You are not allowed to include these** so be sure you don't have any extra <script> tags from your CSS library!
2. Your assignment should have just a single HTML page which **must be** named `index.html`.
3. The assignment should have five main views (**Home, Men/Womens Entry, Browse, Single Product** and **Shopping Cart**) and one dialog/popup (**About Us**). When the program first starts, display the **Home** view.
4. The source of data is in a provided json file (I have provided a formatted and a compact version; you should use the compact version as it will be faster to load). More detail will be provided below.

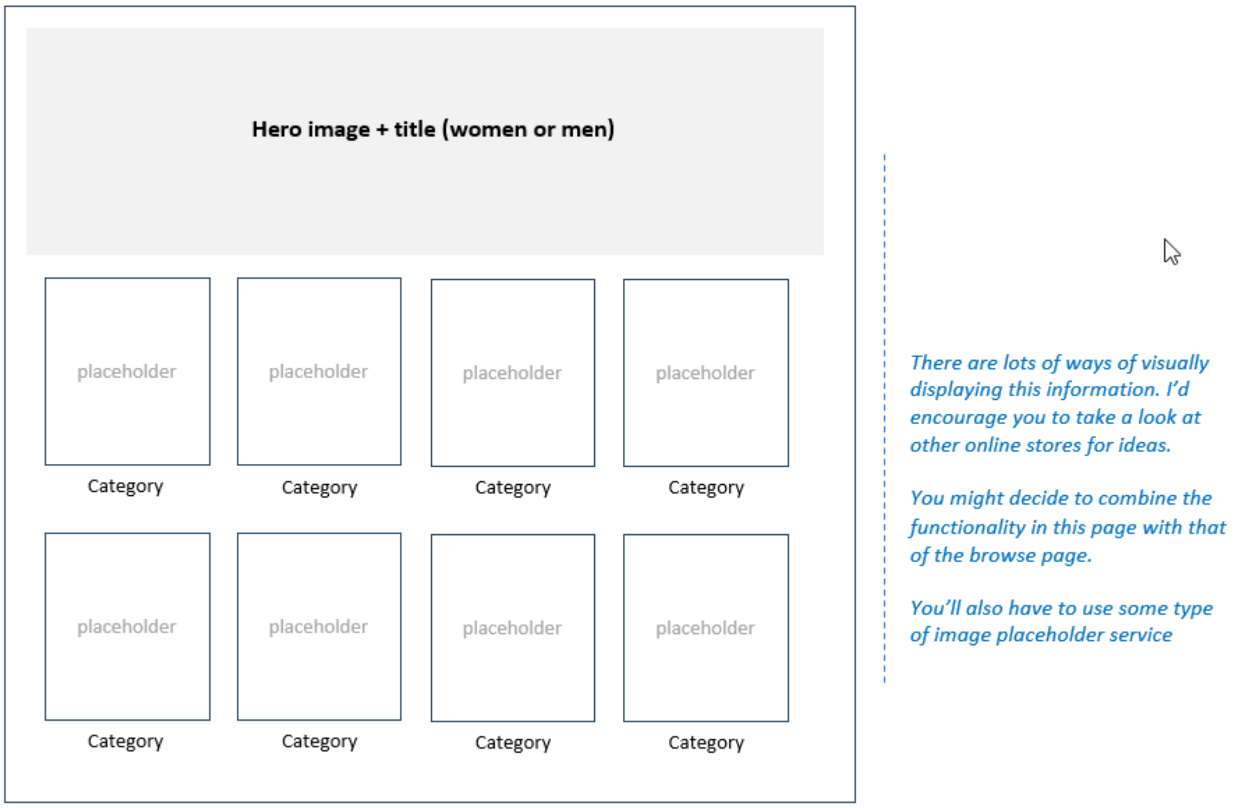
5. You must write your own JavaScript. That is, no jQuery, no Bootstrap, no other third-party JavaScript libraries. You have all the knowledge you need from the lectures and the JavaScript labs (Lab8, Lab9a, Lab9b, and parts of Lab10) to complete this assignment. If you do find yourself, however, making use of some small snippet of code you found online (and I do mean small), then you must provide references (i.e., specify the URL where you found it) via comments within your .js file. Failure to properly credit other people's work in your JavaScript will result in a zero grade. Using antediluvian JavaScript found online will also result in lower marks. There is no need to credit code you found in the labs or lectures.
6. Header / Footer. All your views will contain the same header / footer. Another way of saying this is that you will be changing the content *between* the header and footer. The image below demonstrates the functionality required. The links are not URLs to other html files; instead, they indicate that your SPA must display a different view.


Display Home view when clicked
All your pages will have the same header.
Every time an item is added or removed from cart, update this number
Header
Logo
Home | Women | Men | Browse | About
Cart 6
Provide these links to the different views.
Footer
Provide links to your github page.
It's up to you what else you put into it as long as its consistent page to page. Look at your first assignment to get some ideas.
All your pages will have the same footer.

7. **Home View**. You will implement a page with similar functionality as that described below. As you can see, I've left it up to you in terms of what content to put on this page.

Make it interesting … typically a home page has an extra-large hero image, and a few featured products featured in a larger rendition.
I'd encourage you to take a look at other online stores for ideas.

8. **Mens/Womens View**. Each gender will have a landing page/view of some sort, with functionality similar to that shown below. Each category should take the user to **Browse View** with the relevant products for that gender and category. Notice that I haven't provided you with any images. If you are using images found on the internet, please provide credit in the About Us dialog.


Hero image + title (women or men)
placeholder
Category
placeholder
Category
placeholder
Category
placeholder
Category
placeholder
Category
placeholder
Category
placeholder
Category
placeholder
Category
There are lots of ways of visually displaying this information. I'd encourage you to take a look at other online stores for ideas.
You might decide to combine the functionality in this page with that of the browse page.
You'll also have to use some type of image placeholder service

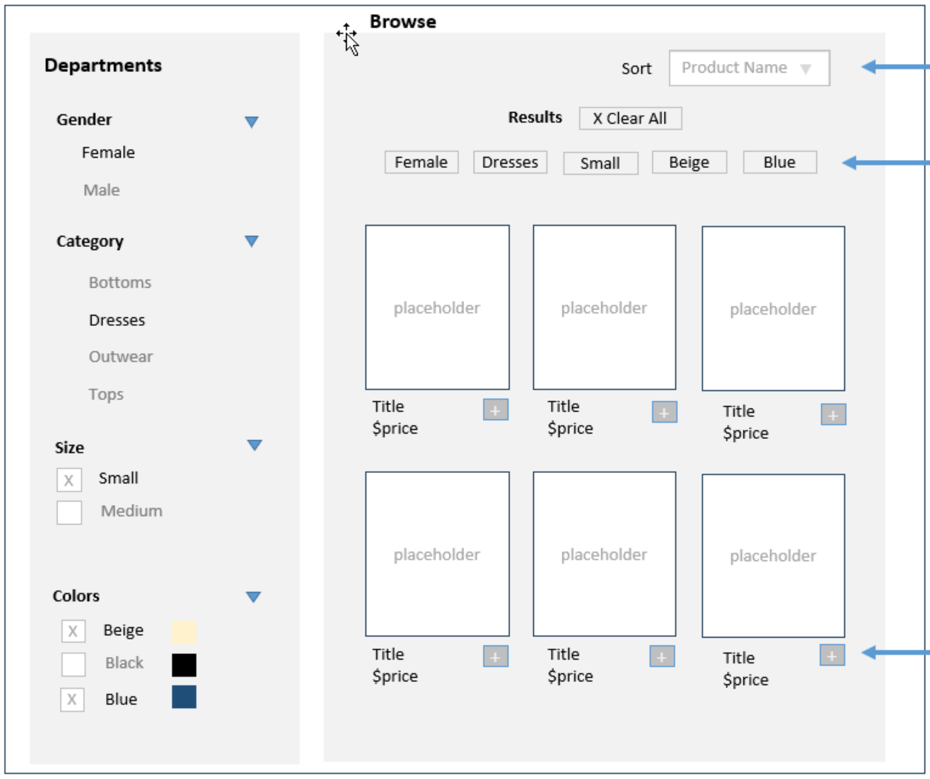
9. **Browse View.** This is the heart of the assignment. Your view must be able to display products that match a variety of different criteria and which has the functionality listed below.

These department links should be additive (ANDs). Be sure to provide visual indication of which ones have been selected. I've indicated two possibilities here (bold and checkboxes).

Need to provide a way to remove a filter.

Every time a new filter is added or removed, update the list of matching products. If there are no matches, tell the user that this is the case.

I'd encourage you to take a look at other online stores for ideas.



Should also be a way to change the sort order between: product name (A-Z), price, and category. The default should be product name.

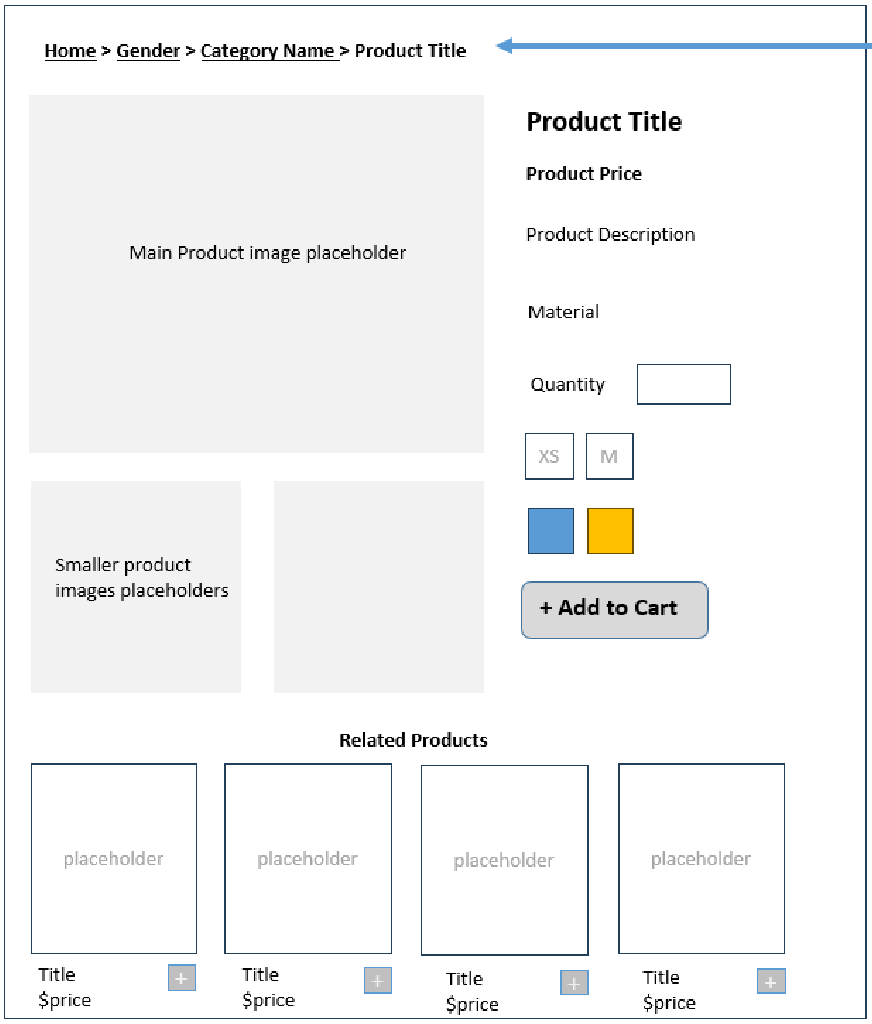
Somewhere you need to show the current filters (and provide easy way to clear them). Here you can see the five filters in place. Perhaps you allow the user to remove these filters selectively here.

There are lots of ways of visually displaying this information. I'd encourage you to take a look at other online stores for ideas.

Each of these images and titles should be links to the single product view

Should also be a way to add product to cart

10. Single Product View. Display details for a single product with the following functionality.



You need to provide breadcrumb navigation, though I don't except the links to work

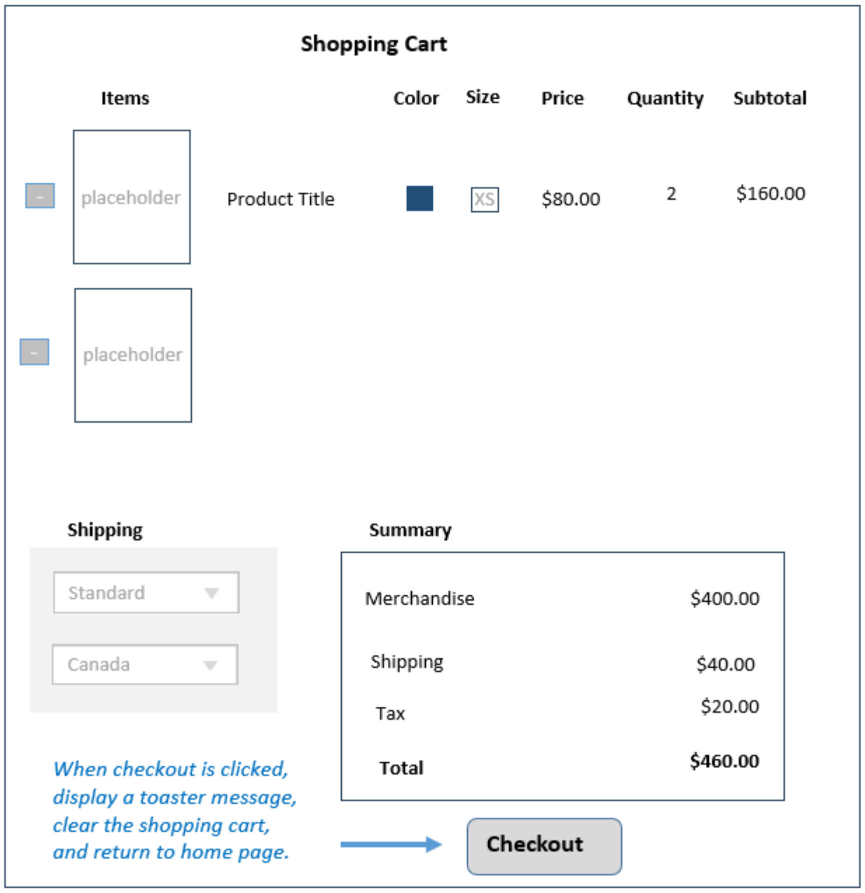
Display the product information in some way (I've shown you possibilities here).

You need the ability for user to enter a quantity (default is 1) for when item is added to shopping cart.

When user enters an item to cart, then use some type of toaster notification (see Lab9b).

This is where some creativity will be needed. What makes an item a related product? Perhaps they have the same category? Similar price? Similar product name?

11. Shopping Cart View. Display current content of shopping cart with the following functionality. The cart must persist; that is, if I close the application and then restart it, the cart will still be there. You must use localStorage (see lab10) to implement this feature.



Shopping Cart

	Items		Color	Size	Price	Quantity	Subtotal
−	placeholder	Product Title	■	XS	$80.00	2	$160.00
−	placeholder						

Shipping

Standard ▾

Canada ▾

Summary

Merchandise	$400.00
Shipping	$40.00
Tax	$20.00
Total	**$460.00**

When checkout is clicked, display a toaster message, clear the shopping cart, and return to home page.

Display all the items in the cart, displaying the info shown here.

If cart is empty, tell the user that this is the case, and disable the Shipping options and the checkout button.

There should also be a way to delete each item in the cart. When this happens the info here should be re-displayed.

Shipping choices: Standard, Express, Priority

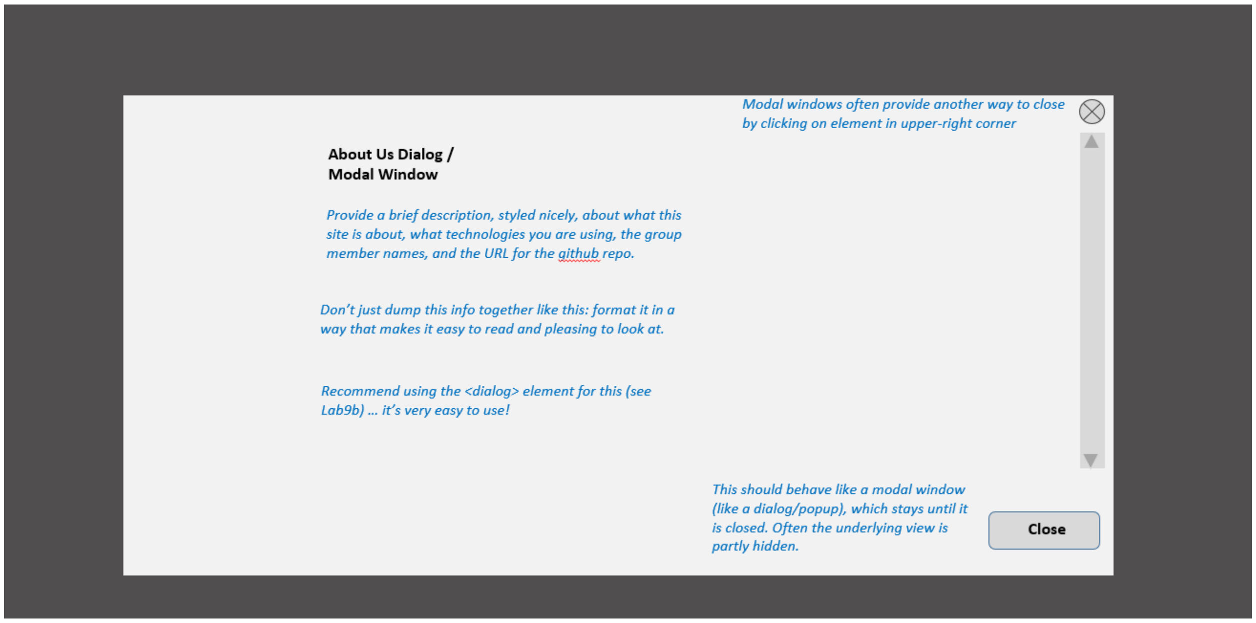
Destination choices: Canada, United States, International.

Tax is 5% only if Canada

Shipping Cost: If merchandise over $500, then it is free.

Otherwise cost for each depends on destination (CA,US,Int): Standard ($10, $15, $20), Express ($25, $25, $30), Priority ($35,$50,$50)

12. About Us Dialog/Popup. Display an about dialog that has info about the assignment, the technologies you are using, info about yourself, a link to the github repo, etc.



About Us Dialog / Modal Window

Provide a brief description, styled nicely, about what this site is about, what technologies you are using, the group member names, and the URL for the github repo.

Don't just dump this info together like this: format it in a way that makes it easy to read and pleasing to look at.

Recommend using the <dialog> element for this (see Lab9b) … it's very easy to use!

Modal windows often provide another way to close by clicking on element in upper-right corner

This should behave like a modal window (like a dialog/popup), which stays until it is closed. Often the underlying view is partly hidden.

Testing

Every year students lose many easy marks because they didn't read the requirements carefully enough. When your assignment is getting close to done, I would recommend going through each requirement step and carefully evaluate whether your assignment satisfies each specified requirement. After you upload to your host, test again!!!

Handling Data

You will be consuming/fetching data from the following API URL. Your code **must** use fetch.

https://gist.githubusercontent.com/rconnolly/d37a491b50203d66d043c26f33dbd798/raw/37b5b68c527ddbe824eaed12073d266d5455432a/clothing-compact.json

This approach means hosting the application is trivial, in that you can host on any static service (such as github pages).

To see the content of this data, I have provided the data-pretty.json file for you to examine without fetching.

Make sure you are only requesting/fetching the data ONCE! To improve the performance of your assignment (and reduce the number of requests on the server), you must store the data in localstorage after you fetch it from the API (see Exercise 10.11 in Lab 10). Notice that you have to use JSON.stringify to create a JSON string version of the array and then saving that in localstorage; similarly, after you retrieve it from localstorage, you will need to uses JSON.parse() to turn it into an array. Your page should thus check if this data is already saved in local storage: if it is then use local data, otherwise fetch it and store it in local storage. This approach improves initial performance by eliminating this first fetch in future uses of the application. Be sure to test that your application works when local storage is empty before submitting (you can empty local storage in Chrome via the Application tab in DevTools). Failure to implement this functionality will result in a large loss of marks so don't neglect it.

Submitting and Hosting

Your assignment source code must reside on GitHub and reside on a working public server (in this case GitHub Pages).

GitHub Pages works in conjunction with git and github. You push your html/css/images to github repo; and then push to the host. The instructions for doings so can be found at:

`https://docs.github.com/en/pages/getting-started-with-github-pages/creating-a-github-pages-site`

It is up to you whether you want your pages to be public (available to the world) or private (available to only those with access to your github repo).

If you are using a private repo, you must add me as a collaborator!

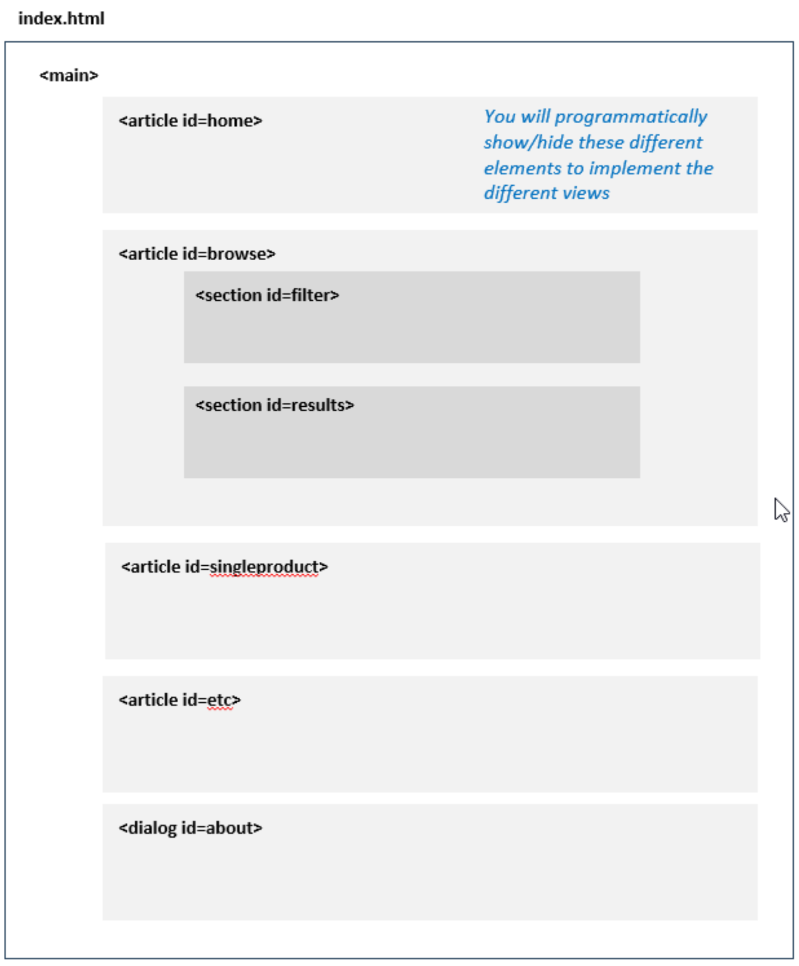
I would strongly recommend getting your hosting to work a few days before the due date. It's okay if your assignment is still not complete at that point: the idea here is to make sure hosting works ahead of time!

When your hosting is working and the assignment is ready to be marked, then send me an email with the following information:

- The URL of the home page of the site on github pages.
- The URL of the github repo so that I can mark the source code. If your repo is private, then add me as a collaborator.

Hints

1. This assignment requires swapping different views. You should implement each view in its own container, e.g.,



2. Remember that JSON and JavaScript are case sensitive. For instance, it is easy to type in `Category` and it won't work because the JSON field is actually `category`.

3. Your `index.html` file will include the markup for all your views. Your JavaScript code will programmatically hide/unhide (i.e., change the `display` value) the relevant markup container for these views.

4. I would recommend implementing the **Single Product View** first and then the **Shopping Cart View** second. The **Browse View** is the most complex so maybe do that third (though some programmers prefer to tackle the most complex use case first).

5. Most of your visual design mark will be determined by how much effort you took to make the views look well designed. Simply throwing up the data with basic formatting will earn you minimal design marks.

6. Most years, students tend to get low marks on the design side of the assignment. I would recommend looking at other sites on the internet and examine how they present data. Notice the use of contrast (weight, color, etc) and spacing.

7. Most of your programming design mark will be based on my assessment of your code using typical code review criteria. For instance, did you modularize your code using functions? Are your functions too long? Is the code documented? Do you have a lot of code duplication (you shouldn't … if you are copy+pasting code, that should tell you that you are doing things wrong from a design standpoint)? Did you make use of object-oriented techniques? Are variables and functions sensibly named? Is your code inefficient (e.g., fetching the same data repeatedly)? Are you using outdated JavaScript techniques (e.g., inline coding, var, `XmlHttpRequest`, etc)? Is the code excessively reliant on found code from Stack Overflow, etc?

8. You will need to find some way of handling routes. For instance, how will your JavaScript "know" whether to display single product view or shopping cart view? Ultimately, you will be listening for button / link / list clicks. But on any given page, there will be a lot of them! When I assess the design mark, I will be looking at how you've handled this problem. See if you can come up with more generalized solutions to this problem. BTW, one of the advantages of large JavaScript frameworks such as React or Next is that they provide some type of easy way of managing and implementing routes.