JAVASCRPT SITE ASSIGNMENT GUIDE

# What

## Background

For this site you will make a website for a coffee shop [[Project File Link](https://uncg.box.com/s/ptn1ytespae4wyk8o3si5rn9vnnijtzw)]. The shop wants to have a menu that users can easily browse from on their phone while they wait to speed ordering times at the counter. They have asked that you code a simple multiple page html site with a menu with buttons that show and hide the different sub menus they have. Your boss has coded up the files and assets included in project files. Your “job” is to write code to make these files pretty and to make

From the included assets in the project file, finish any loose ends in the HTML, stylize the site to meet the spec, and write some JavaScript to get the menu to function as desired.

## Requirements

* Your site must be based on the primary brand colors, but I recommend choosing a new off white rather than F0E9F0, which is a pink-white.

### HTML

* Each page in your site should have your name in the appropriate meta tag
* Remove the color swatches from the main page.
* Link the banner to the homepage.

CSS

* Use a single CSS file to style each of the pages included in the project files.
* Use comments to help you organize your code.
* Be responsive
* The about page and contact us pages have a current class in their menu to provide a breadcrumb for the user, come up with a visually distinct way to represent this within the navigation menu.
* The headings on the contact menu should be smaller than the h1 on other portions of the site.
* The divs containing the different ways to contact should be styled as blocks.
* The first word in each paragraph of the about page should be highlighted and a different color than the rest of the text.
* The menu should have a clear visual hierarchy.
* Each items price should have a dollar sign in front of it. Add this in CSS not HTML.

JavaScript

* Create a menu from the definitional lists with buttons that only shows selected menu
* In addition, make it so that the menu has buttons that show and hide the parts of the menu that the user might want to look at or might be done with.
  + Use JavaScript for this.

### Other

* Create a way to highlight specials either using CSS or JavaScript

## Submission

Take all the project files necessary to make your site work, zip them, and upload the zip file into the course’s canvas shell for the assignment. Use the LASTNAME\_jsSite for the files naming convention.

[Zip Guide [PC]](https://support.microsoft.com/en-us/windows/zip-and-unzip-files-8d28fa72-f2f9-712f-67df-f80cf89fd4e5) – [Zip Guide [Mac]](https://support.apple.com/guide/mac-help/zip-and-unzip-files-and-folders-on-mac-mchlp2528/mac)

# The why

I’m evil. Also, you must learn JavaScript. No really. It’s so important figuring out a basic script like this might be tough, but you will be so happy you did.

# The How

Things are almost ready to go.

1. Look at the color swatches, maybe make a separate document with how you want to use them.
2. Start fixing the HTML
3. Start styling in CSS
4. When things look about rightish, go in and start writing your JavaScript app.
   1. This AG has a very general set of instruction, write specific smaller steps so you know what to code.
   2. Start writing code, declare variables, and write test functions to make sure things are connected.
   3. Then go deeper and write the code you want to see in the world.
5. When you’re done, delete the extraneous files and zip your drive. I’m going to grade it on whether it works or not, so you need to keep visual and code assets, but you can delete instructions and licensing.

**Criteria**

In evaluating project submissions each criteria item is evaluated using the following levels:

1. Subpar or Lacking Proficiency
2. Baseline Proficiency
3. High Proficiency

***Functionality***

| **Metric** | **High Proficiency** | **Baseline Proficiency** | **Subpar or Lacking Proficiency** |
| --- | --- | --- | --- |
| Matching specifications | The application functions exactly as intended. | The application functions mostly as intended. | The application does not function as intended. |
| Javascript Console Errors | No errors. | Less than 3 noted errors per page. | More than 3 noted errors per page. |
| Markup errors (missing/extra closing tags). | No errors. | Less than 3 noted errors per page. | More than 3 noted errors. |
| CSS errors (non-existent properties, incorrect rule syntax). | No errors. | Less than [3] errors. | More than [3] errors. |
| File references (images, css files, linked scripts). | No broken references. | Less than [3] broken references. | More than [3] broken references. |

***Appearance***

| **Metric** | **High Proficiency** | **Baseline Proficiency** | **Subpar or Lacking Proficiency** |
| --- | --- | --- | --- |
| ~~Matching specifications~~ | ~~The application’s front-end matches the designs exactly.~~ | ~~The application’s front-end matches the designs closely, but not exactly.~~ | ~~Less than 50% of the application’s front-end matches the design.~~ |
| Visual errors (unnecessary vertical/horizontal scroll distances, misplaced content) | No visual errors. | Less than 3 errors. | More than 3 errors. |

***Code Quality: HTML***

| **Metric** | **High Proficiency** | **Baseline Proficiency** | **Subpar or Lacking Proficiency** |
| --- | --- | --- | --- |
| Structure | The pages are broken down into logical sections. | The pages are broken down into logical sections, with a few minor exceptions. | There may be some attempt at breaking the pages down into logical sections, but overall, they are not logically structured. |
| Choice of HTML tags | The individual consistently chooses the best (most semantic) element for the job. Tags are nested correctly and logically. | The individual chooses the best (most semantic) element for the job, with a few minor exceptions. 75%+ of tags are nested correctly and logically. | The individual makes some unsuccessful attempts, but overall, fails to choose the best (most semantic) element for the job. Most tags are not nested correctly and logically. |
| Use of classes and IDs | The individual appropriately uses classes on elements with similar purposes and/or similar styling. Class and ID names are intuitive, meaningful, and concise. | The individual uses classes on some elements with similar purposes and/or similar styling. With some minor exceptions, class and ID names are intuitive, meaningful, and concise. | The individual does not apply classes on elements with similar purposes and/or similar styling. Overall, class and ID names are not intuitive, meaningful, or concise. |
| Readability | Code is very easy to read. | The individual indents consistently, and leaves frequent comments. | Code is mostly easy to read. |
| Separation of content | The individual consistently makes good decisions about when to use CSS to achieve visual effects, over HTML. | With a few minor exceptions, the individual uses CSS to achieve visual effects, over HTML, when appropriate. | The individual uses HTML to achieve visual effects, when CSS would be more appropriate. |

**Code Quality CSS**

| **Metric** | **High Proficiency** | **Baseline Proficiency** | **Subpar or Lacking Proficiency** |
| --- | --- | --- | --- |
| Structure | The code is separated into logical sections. | The code is broken down into logical sections, with a few minor exceptions. | There may be some attempt at breaking the code down into logical sections, but overall, it is not logically structured. |
| Use of CSS Properties | The individual demonstrates a command of a wide variety of properties and techniques. He/she successfully accomplishes the desired visual outcomes in standard, accepted ways. | The individual demonstrates ability to implement a variety of properties and techniques. He/she accomplishes the desired visual outcomes, primarily through standard, accepted ways. | The individual implements some properties and techniques. He/she may accomplish the desired visual outcomes, but misuses certain properties. |
| Readability | Code is very easy to read. The individual indents consistently, and leaves frequent comments. | Code is mostly easy to read. The individual indents and leaves comments. | Because the individual does not indent consistently or leave comments, the code is challenging to read. |
| Concision | The individual’s code is very concise. He/she makes clever use of selectors to keep the number of duplicate rules low. | The individual’s code is concise, but there may be a few unnecessary duplicate rules. | The code is not concise. There are several instances where selectors could have been used, but were not. |

**Code Quality Javascript**

| **Metric** | **High Proficiency** | **Baseline Proficiency** | **Subpar or Lacking Proficiency** |
| --- | --- | --- | --- |
| Organization | Procedural code and event handlers are consistently separated from functions. | Procedural code and event handlers are separated from functions, with a few minor exceptions. | Procedural code and event handlers are not separated from functions. |
| Program Logic | The student has very effectively used loops, functions, and conditionals, to make the menu work. | In a few instances, the student did not use loops, functions, and conditionals, where they might more easily accomplished certain goals. | There are several cases in which students could have more easily accomplished certain things, using loops, functions, and conditionals. |
| Readability | The code is well-indented and consistently spaced. Throughout, complex pieces of logic are appropriately commented. | The code is indented and consistently spaced. In most cases, complex pieces of logic are appropriately commented. | While there may be some attempt at organizing the code, it is not indented or spaced to support readability. Further, complex pieces of logic are missing commentary. |
| Efficiency | The code is consistently written in an efficient manner, with an eye for accomplishing as much as possible with concise code. | The code is efficiently written. | Student regularly performs unnecessary operations. |
| Maintainability | Throughout, the code is DRY (free from repetition of blocks of identical or highly similar code). All variables have intuitive names. | The code is DRY (free from repetition of blocks of identical or highly similar code). With a few minor exceptions, variables have intuitive names. | The code includes repetition of blocks of identical or highly similar code. Most variables do not have intuitive names. |