MMC6278 Assignment Instructions

# GitHub Setup

1. Create a public Repository in [GitHub](https://github.com) with the following naming convention:
   * Template: mmc6278-<module#>-<assignment#>-<lastname>-<firstname>
   * Ex: mmc6278-1-1-washington-george
2. You may include a description if you wish, but it is not required.
3. You may choose to include a README or license for the project if you wish, but it is not required.
4. Clone the repository to your computer in whatever directory you usually use to complete your coding assignments.
5. Copy the contents of the “unsolved” folder from the assignment files into your repository. **Do not copy the folder itself, only its contents**. For example, if the unsolved folder has an index.html file, index.js, and style.css file, your repository should then have those same files. Do **NOT** rename any of the files.
6. Make a commit in your repository and push the commit to GitHub.
7. Visit your repository page and inspect the files. You should **not** see an “unsolved” folder. If you do, refer back to steps 5 and 6.
8. After committing your files to GitHub, you will need to deploy your app to GitHub Pages.
   * On your repository page, click the settings tab, then pages.
   * Under Source, select the “master” or “main” branch from the dropdown.
   * To the right of the branch selection, “/root” should be selected by default. Do not change this setting. This points to the root of your repository, or the top-level folder of the repository, where your index.html file should be.
   * Click the “save” button to the right of the branch and directory dropdowns.
9. You should see a URL at the top of the page after clicking save. This is your live site URL.

# Assignment Instructions

1. Open your assignment’s GitHub repository folder with VS Code on your machine.
2. Open the index.html file and right-click, then select “open with live server”.
   * If you do not have the [live server extension](https://marketplace.visualstudio.com/items?itemName=ritwickdey.LiveServer), open the “extensions” menu in VS Code and install it.
3. The instructions and required markup for the assignment are all included in the same index.html file. Follow the instructions displayed on the page to complete the assignment.
   * **Note**: The one exception to this is module 6’s assignment. The instructions for that assignment can be found within the included markdown file instead of the HTML.
4. As you complete the assignment, be sure to push your commits to GitHub and check your live site URL to ensure the site is functioning correctly. You may make as many commits as needed to complete your assignment.
   * Note: it is a good practice to make commits based on features. For example, if completing the header of a website, you may wish to make a commit before proceeding to creating a navigation menu.

# Submission

Please submit your assignment within Canvas. You will need to include **BOTH**the repository URL and the live site URL with your submission.

You may include any comments for your instructor with your submission in the provided comments field.

# Getting Help and Tips for Success

The assignments in this course are intended to challenge you and place you outside of your comfort zone. The lecture material will cover all the necessary concepts you need to master to complete the assignments, but will often not demonstrate exactly how to solve all the problems found within each assignment. Learning to program, especially for the complete beginner, requires considerable effort, patience, and practice. That said, the following tips may help when working on your assignment:

1. **Do not procrastinate.** Do not wait until the day an assignment is due to begin an assignment. It may take multiple sessions spread out over multiple days to complete a single assignment.
2. **Thoroughly read the instructions of each assignment.** Not only is it important to understand exactly how each application should behave, but there may also be helpful hints and strategies included in the instructions.
3. **Break the application down into smaller chunks.** If you need to make a sliding drawer menu for a website, it’s often not helpful to think of the problem as a whole. Think of a small piece of functionality and solve that piece.
   * For example, with a sliding drawer, the user will need to click a button to open and close the menu, press escape to close the menu, or click outside the menu to close it. The menu will also need to be animated. Each of these can be solved individually.
4. **Write pseudocode.** Building on the previous point, once you know the smaller problems you need to solve, you can translate each one pseudocode. After translating to pseudocode, the steps become much easier to program.
   * Returning to the sliding drawer example, the first piece is opening/closing the menu with a button. In pseudocode, we could write:
     1. Make variables to store references to the button and sliding menu HTML elements.
     2. Attach a click event to the button.
     3. Inside the click event, change the menu’s CSS to show/hide the menu.
5. **Test your code.** As you code your solution, make sure to validate the code you are writing is functioning properly. You can both test your code manually on the page and/or with console logs, as well as use the included automated tests to verify specific pieces of functionality.
   * Some students attempt to write their entire JavaScript program at once without validating that their code is working. This can make things much more difficult. It is much easier to debug a few lines of JavaScript at a time than it is 50+ lines.
6. **If your code isn’t working, look in the dev tools console.** The dev tools console often provides helpful error messages and can even link to the specific line of your code that is broken. It’s strongly encouraged that you open dev tools to the console tab when you begin programming anything with JavaScript. This is the first place your instructor will look when you ask for help.
7. **Use Google liberally.** Programmers often look up basic features of programming languages they code in everyday. Do not attempt to memorize programming syntax. Also, if you become stuck while trying to solve an assignment, try to find a solution with Google. MDN, W3 Schools, and Stack Overflow likely contain the answer to your problem. You can find answers more easily by being specific with your search query:
   * Bad: “How do I make a menu”
   * Better: “How do I make a menu with JS”
   * Best: “How can I animate a sliding drawer menu with JS”
8. **Trial and Error is OK.** You may need to experiment with different techniques when attempting to solve a problem. Do not worry about crashing your computer or losing progress when programming with JavaScript. In the worst case scenario, you may have to close your web browser to stop a script from running, but you can simply amed your code and reload the page.
9. **Rubber Ducking.** “Rubber Ducking” is the practice of explaining the problem you are facing to a rubber duck. Although the duck cannot understand your plight, often the act of putting the problem into words is enough to generate new ideas and lead the programmer to a solution. Other things may be used in lieu of a literal rubber duck like pets or significant others (although both may give you funny looks).
   * Often, students will come to a conclusion when stuck merely through writing their question out in a message to their instructor. This usually results in the instructor’s inbox becoming filled with “nevermind, solved it” messages shortly before assignment deadlines.
10. **Take a break.** If you are *truly* stuck, walk away from your computer and do something else for a while. This could be doing chores, taking a walk, watching TV, or even taking a short nap. After some time away from your screen, you may find that you think of a solution in the middle of an unrelated activity or immediately after returning to your computer. Believe it or not, this phenomenon is common among programmers.

If you are unable to find a solution after following the above tips, **you may reach out to your instructor for help**. Please provide links to your repository and live site URL along with a description of the problem or error when asking for help. Also, please avoid taking screenshots of your code as they are generally hard to read and debug.