Lab Week 3 - CSS and Agile Intro

- Due Tuesday by 11:59pm
- Points 3
- Available Apr 4 at 3pm - Apr 25 at 11:59pm

Due Date - Tuesday, April 22, 2025 11:59pm on Gradescope

Note - Make sure that you have submitted the previous labs before otherwise your submission would not be graded.

FAQs (https://docs.google.com/document/d/1IcUr5yWcs4d2MnWnkkZBQc7Yd2usp=sharing) for this lab

We will be updating this with common questions as lab hours go on, please check to see if your question has been answered before asking for help. Also remember to join #lab-3 channel by using the 'Browse Channels' option in Slack.

This is to be completed as a solo assignment.

The previous lab assignment had you familiarizing yourself with HTML and your browser's DevTools. In Web Development, the core technologies build off of each other. Now that you have the frame and structure of your GitHub User Page set with HTML, you will be adding an aesthetic layer on top of it with CSS. In this lab you'll get a general overview of the basics of CSS along with how it intersects with HTML. While last week we let you attack the lab in any manner you saw fit, this week we'll be having you incorporate the Agile software engineering model to complete this lab so you can familiarize yourself with the workflow.

Resources

- https://developer.mozilla.org/en-US/docs/Web/CSS (https://developer.mozilla.org/en-US/docs/Web/CSS)
- https://www.w3schools.com/css (https://www.w3schools.com/css)
- (https://github.com/features/project-management) https://github.com/features (https://github.com/features)

Lab Intro and Basic Concepts

Set Up

1. Make a copy of your Lab 2 repository:
   - Create a new public repository called “sp25-cse110-lab3” on GitHub
   - In your local terminal (replace <> with your Lab 2 repo name, without the <>):
     - cd <lab2-repo-name>
     - git push --mirror https://github.com/*your_username*/sp25-cse110-lab3.git **OR if you're authenticating with an SSH key, use** git push --mirror git@github.com:*your_username*/sp25-cse110-lab3.git
     - **** Make sure you don't accidentally push to lab2 ****

       If you have accidentally pushed to lab2, please to revert the pushed commit: **https://stackoverflow.com/questions/22682870/git-undo-pushed-commits/36177806 (https://stackoverflow.com/questions/22682870/git-undo-pushed-commits/36177806)**
   - Clone the new repository to your local machine
2. Publish the new repository through GitHub Pages (**DO NOT** make your lab 2 repo private until you have received your grade back for it)
3. Create a README.md file containing the URL of your new GitHub Pages site. Your GitHub Pages site will be found at **<username>.github.io/sp25-cse110-lab3**

Part 1. Introduction to Agile with GitHub

Give this a quick read:**https://github.com/features (https://github.com/features)** . It introduces project management using different features in GitHub. More specifically, we’ll be practicing **using Issues and Pull Requests (https://docs.github.com/en/github/managing-your-work-on-github/managing-your-work-with-issues-and-pull-requests)** in this part of the lab.

The point of this part of the lab is to introduce you to the general workflow of how your team should operate using GitHub. If done properly: everything is clearly documented for everyone on your team to see, there is a clear list of what needs to be done up next, and (theoretically) any new team member could be onboarded easily by simply looking through your repository.

Issues

https://guides.github.com/features/issues (https://guides.github.com/features/issues)

- Create custom labels
- Create a custom **issue template (https://docs.github.com/en/communities/using-templates-to-encourage-useful-issues-and-pull-requests/configuring-issue-templates-for-your-repository)**
- Go through this week's lab and breakdown tasks. Then create issues for those tasks with the appropriate labels and assignee (yourself).
  - Tasks can be something like: "Create meeting minutes template" or "Style meeting minutes"

Note - Do NOT use random issues, make them relevant to the lab because these are meant to give you practice for when you start working on your project. This is a critical skill in SWE and this portion of the lab is meant to help to reduce the gap between the lab assignments and project.

Pull Requests

https://guides.github.com/activities/hello-world/#pr (https://guides.github.com/activities/hello-world/#pr)

- As you go through part 2, make incremental pull requests to resolve the issues you've made (ideally 1 pull request per issue)

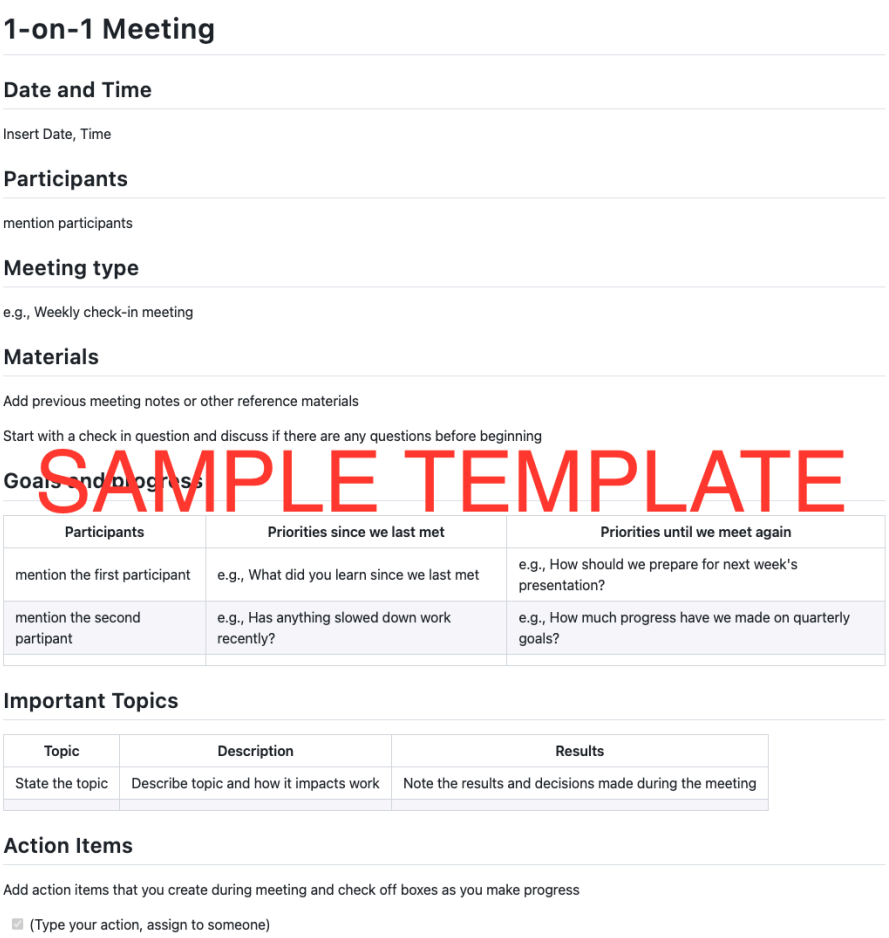
1. Instead of pushing straight to master, create a new branch to push to.
2. Make your changes on this branch.
3. Make a pull request from this branch to merge changes back into the main branch. Before you merge the pull request, **link this pull request to its corresponding issue (https://docs.github.com/en/github/managing-your-work-on-github/linking-a-pull-request-to-an-issue)**.
4. Repeat this process for each issue.

Standup Notes

This Agile standup guide from Atlassian (https://www.atlassian.com/agile/scrum/standups) has a good explanation of what standups are. Essentially, standups are daily meetings where typically, each member gives their status updates so that the team can keep track of sprint progress and resolve blockers. Standup notes are just notes to facilitate this meeting.

- Create a standup notes template in a Markdown file **standup.md**

You can also find a sample standup notes template below. You **DO NOT** have to create the exact same template, this is just for reference and to offset your thinking. Please **do not copy** these templates.



1-on-1 Meeting

Date and Time

Insert Date, Time

Participants

mention participants

Meeting type

e.g., Weekly check-in meeting

Materials

Add previous meeting notes or other reference materials

Start with a check in question and discuss if there are any questions before beginning

Goals and progress

SAMPLE TEMPLATE

Participants	Priorities since we last met	Priorities until we meet again
mention the first participant	e.g., What did you learn since we last met	e.g., How should we prepare for next week's presentation?
mention the second partipant	e.g., Has anything slowed down work recently?	e.g., How much progress have we made on quarterly goals?

Important Topics

Topic	Description	Results
State the topic	Describe topic and how it impacts work	Note the results and decisions made during the meeting

Action Items

Add action items that you create during meeting and check off boxes as you make progress

- [ ] (Type your action, assign to someone)

Part 2. Adding Cascading Style Sheets (CSS) to your Meeting Minutes

Remember how your minutes from Lab2 was looking a little drab? In this lab, we're going to use Cascading Style Sheets (CSS) to give them a little flair. From the **Mozilla Web Docs (https://developer.mozilla.org/en-US/docs/Web/CSS/Reference)** :

Cascading Style Sheets (CSS) is a stylesheet language used to describe the presentation of a document written in HTML or XML (including XML dialects such as SVG, MathML or XHTML). CSS describes how elements should be rendered on screen, on paper, in speech, or on other media."

3 ways to use CSS (https://www.w3schools.com/css/css_howto.asp) :

- External CSS stylesheet (preferred method for better code readability)
- Inline CSS (helpful for debugging)
- Internal CSS with <style> (convenient for quick experiments)

The bulk of your CSS should be included in an external CSS stylesheet, but do include at least one example of inline and using the <style> element for styling.

Quick CSS Syntax Overview -- CSS styles are applied by selecting an element, and identifying a value (i.e. color, measurement, etc.) to apply to a style property (i.e. width, background-color, etc.). The format looks a little something like this

```
selector {
    property1: value1;
    property2: value2;
}
```

for clarity in this lab writeup:

- selectors will be highlighted in **green**
- properties will be highlighted in **purple**
- values will be highlighted in **orange**

Instructions

Use the bulleted lists below as a checklist for your webpage. **Include every item from every bullet point in the checklist at least once.** If you are unsure of what a specific CSS property does or how it works, feel free to use the **Mozilla CSS reference (https://developer.mozilla.org/en-US/docs/Web/CSS/Reference)** and other resources on the internet to your advantage. This lab will not be graded on visual execution as this is not a visual arts course; that being said we will still be looking to make sure you put in some effort. **Feel free to make any changes you want to your html file from the last lab in order to complete this lab.**

CSS Checklist

1. General **CSS Topics:**

- Comment /* write down comments to make your css easier to read */

- Color /* apply colors to your HTML elements */
  - rgb(r, g, b) or rgba(r, g, b, a) /* red, green, blue, alpha values */
  - #FFF or #FFFFFF /* hex codes */
  - hsl(h, s, l) or hsla(h, s, l, a) /* hue, saturation, lightness, alpha values */
  - Color name (i.e 'orange')
  - Wider-gamut color:
    - color(colorspace c1 c2 c3[/ A]) /* predefined color space, values for color space*/
    - color-mix(method, color1[p1], color2[p2]) /*method to mix colors, values & percentages of color*/
- CSS Variables & Fallbacks
  - Show at least one example of using a CSS variable that also has a fallback.

- **Background** (https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/Backgrounds_and_borders#styling_backgrounds_in_css) /* apply background styles to your elements */
  - background-color

- **Unit** (https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/Values_and_units) /* units of measurement for sizing and spacing your elements */
  - Use 3 unique relative units total
  - Use 3 unique absolute units total

- **Box Model** (https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/The_box_model) /* configure the containers that holds your HTML content */
  ("long" and "short" refer to longhand and shorthand syntax and should give the same results. They're simply different ways to declare your style rules, use at least one of each syntax. Y**ou must use both long and short hand notations for each of the following: margin, padding, border**)
  - Margin /* spacing between html elements */
    - Long (margin-top, margin-bottom, margin-left, margin-right)
    - Short (margin: <top> <right> <bottom> <left>)
    - Auto margins: margin: auto
  - Padding /* spacing within html elements */
    - Long (padding-top, padding-bottom, padding-left, padding-right)
    - Short (padding: <top> <right> <bottom> <left>)
  - **Borders** (https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/Backgrounds_and_borders#borders) /* borders around html elements, hint: apply borders before testing out padding and margin to better understand the difference between the two */
    - border-style
    - border-color
    - border-width
    - border-radius
- Text /* style your text */
  - color
  - text-decoration
  - text-align
- **Display** (https://developer.mozilla.org/en-US/docs/Web/CSS/display)

  - Experiment with these values: none, block, inline-block, inline. Include at least two of them in your page.
  - Apply these values to the display property
- Sizing /* set the height and width for an element */
  - height
  - width
  - max-width
  - min-width
- **Position** (https://developer.mozilla.org/en-US/docs/Web/CSS/position) /* element positioning on the page */
  - 2 of the following values: static, relative, fixed, absolute, sticky
    - Apply these values to the position property
- **Pseudo-class** (https://developer.mozilla.org/en-US/docs/Web/CSS/Pseudo-classes) /* elements that exist in your document conditionally */
  - :hover
  - :active
- Layouts
  - **Flexbox** (https://css-tricks.com/snippets/css/a-guide-to-flexbox) /* allow your elements to lay themselves out automatically */
    - apply flex to the display property
    - Must have more than two children within the element that is using flexbox. Must use minimum three of the flexbox related attributes
  - **Grid** (https://css-tricks.com/snippets/css/complete-guide-grid) /* instantiate a grid for your layouts */
    - apply grid to the display property
    - Must have more than two children within the element that is using the grid. Must use a minimum of three of the grid related attributes
- Responsiveness /* make your website friendly for multiple devices */
  - At least one query based on the screen width
    - **Media Query** (https://developer.mozilla.org/en-US/docs/Web/CSS/Media_Queries/Using_media_queries)
  - Check the DevTools in your browser! Make sure your site works and looks fine on the three main types of form factors (Very small screen (phone), tablet or smaller laptop, and desktop).
  - The layout of your page should automatically reflow when the size changes, meaning, we shouldn't have to pinch and zoom in a lot to read text, click buttons, etc.
- **Fonts** (https://developer.mozilla.org/en-US/docs/Learn/CSS/Styling_text/Web_fonts) /* pick varying font styles to make your text fun to read */
  - Include and use a 3rd party font (**https://fonts.google.com/** (https://fonts.google.com/)). You can load the font in either your HTML or your CSS

2. CSS Selectors (https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/Selectors)

CSS selectors allow you to select the HTML element you want to style. Each type of selector targets a different identifier on your HTML element. For this lab you must use at least one of every bulleted selector method.

- Class Selector (.class)
- ID Selector (#id)
- Universal Selector (*)
- Element Selector (element)
- Attribute Selector (e.g. [attribute=foo])
- Pseudo-class Selector (e.g. p:hover)
- Selector List (element, element) /* select multiple elements */
- Combinators (you must use one of each) /* specify selections based on element positioning in the DOM tree */
  - Descendant Combinator (element element)
  - Child Combinator (element > element)
  - General sibling combinator (element ~ element)
  - Adjacent sibling combinator (element + element)
  - Combining Two Selectors (element.class)
- New Selectors /*Adopted in December 2023!*/
  - :has() /*You must use this selector. Click **here** (https://developer.mozilla.org/en-US/docs/Web/CSS/:has) for how to use. This is a new selector that was widely adopted in 2023.*/
  - Nested Selectors
    - Nested selectors are another type of selector that have recently been adopted in 2023. **Here** (https://developer.mozilla.org/en-US/docs/Web/CSS/Nesting_selector) is some more information about this selector. Please also include at least one type of this selector in your CSS.

3. CSS Validation

Similar to Lab 2, where we validated our HTML, we can also validate our CSS. Validate your CSS through a **validator** (https://jigsaw.w3.org/css-validator/) and take a screenshot. Add this screenshot to your repo! You might notice that the validator will throw an error for some of the newer selectors you used. If this is the case, that's ok!

However, a good resource to know about is **caniuse.com** (https://caniuse.com/) . Try inputting anything HTML or CSS related into the search bar on this site, such as an HTML tag or a CSS selector and see which browsers support it!

THINK: We just used :has() and nested selectors. Based off caniuse.com, why might these newer selectors be a concern to be using? Remember to think about users and how not everyone may have the latest version of everything.

OPTIONAL: If you find yourself wanting to go above and beyond, here are some things you can do:

- CSS Transformations
- CSS Animations

Canvas Submission:

- You will be submitting **the link to your Lab3 repository (not the link to your Github Pages site)**. Your repo should include:
  - **README.md** file with the link to your hosted GitHub Pages site, found at <username>.github.io/sp25-cse110-lab3 (make sure your site link works)
  - Standup notes template in **standup.md**
  - Relevant HTML (from Lab 2) and CSS files (from Lab 3)
  - Screenshot of CSS validation