Project 1

Computer Science 144 – Prof. Rosario

Due Sunday, Apr 20, 2025 11:59 PM on Gradescope

HTML, CSS and JavaScript

In CS 144 we will study several technologies used to create web applications. We need to start somewhere though, so Project 1 serves as a warmup and assessment of the basic building blocks required for future projects, as well as some newer, more relevant content. **This project will be built locally and will not require Google Cloud (yet).**

The purpose of this assignment is to explore the basics of building a web page using HTML, CSS and JavaScript. You will construct at least one web page. **While this is an assignment, try to have fun with it!**

Purpose

We are aware that many students in the course are quite familiar with HTML, or at least the basics of HTML and CSS. The purpose of this assignment is to

- Refresh your understanding of basic HTML/CSS, in general, by working on this project
- Assess your understanding of more advanced and/or more timely features.
- This year, the requirements focus more on the second bullet point.

Your grade on this assignment will depend only on your choice and use of HTML/CSS/Javascript features, whether or not your submission validates and works properly, etc. We will not grade based on how beautiful your web page is, or the theme you choose but you should try your best to make the page look presentable. If you have time, we do encourage you to make your web page look beautiful, just as a point of learning.

Overview

You will create at least **one** HTML file, one CSS file to use as a style sheet for your page and one Javascript file as well as at least one image. You will use your browser (I strongly recommend Google Chrome, for consistency) to preview your work so you can make changes. You will then submit a zip file containing the HTML file(s), CSS file(s), Javascript, the image(s) and a `MANIFEST` file. If you choose to submit using Github, you should also include a `README`.

Here are some sample themes you may be of interest to you, though you may come up with your own:

1. An "About Me" page. And this one is not just a pedagogical exercise. These pages can be very useful for your job search or to market your skills. Outside of this class, you can then make it beautiful and host the page on Github for free, or anywhere else that makes sense.
2. Where you are from (e.g. city, state, country) and information about it.
3. A city, country, state, place etc. that you want to visit or have visited
4. Your career interests and why (this makes a nice complement to #1)
5. A visual resume' (again, a complement to #1)
6. About a project you are working on, or have worked on.
7. Your favorite sport or athlete.
8. Your favorite singer/band (e.g. Sabrina Carpenter, Harry Styles, Chappelle Roan, Lifehouse, Mighty Mighty Bosstones, Dave Matthews Band whatever)
9. About someone you admire.
10. **Anything else that is appropriate and respectful** 😀

There is no obligation for you to disclose any personal information about yourself or your interests if you prefer not to. You may want to choose a subject that invites the use of interesting HTML features (e.g. a page about a sports team or athlete will be an easy topic to find a table for).

Note that while we will not be grading based on content, you should address the following guidelines in a way that makes sense based on the context so you understand the importance of each concept. Randomly styling random words in your page is not a great way to show this.

While the requirements may seem overwhelming, some of these can be completed in a few seconds and many in minutes. Your page(s) must collectively meet these requirements. You should address an entire requirement in one of the pages. Meaning, a requirement such as #10 which has 5 sub-requirements should all be implemented in one or both pages, not split across the two. Requirements in blue are responsive web design requirements.

At minimum, your page should include and/or demonstrate at least the following:

1. Good web design hygiene such as a `<title>` etc. and using <div> and <span> where necessary. Apply CSS styling that creates a cohesive, visually appealing design using appropriate properties. At this level of class, we will not be prescriptive except in certain bullet points below, but it must look like an attempt was made to use basic CSS aside from the requirements. You should prepare for your page to contain several paragraphs of text – this is not a requirement, but you may need this content to show all of the requirements. **The main page must be named `index.html`.**
2. All of your content must use HTML5 semantic tags except where it does not make sense. In those cases you can use a `<div>`.

3. Your entire webpage must be responsive and display properly across different screen sizes:
    a. must look reasonable at 320px width
    b. all content should be legible and properly formatted on mobile (iPhone 14 Pro Max), tablet (iPad Mini), and desktop viewports.
    c. implement at least two media queries to adjust layouts at different breakpoints to satisfy (a). Text should remain readable without horizontal scrolling on small screens
    d. images should scale appropriately to fit different viewport sizes
    e. navigation elements should adapt to smaller screens (possibly changing to a more mobile-friendly format)
    f. test your site at multiple viewport widths to ensure content remains accessible and visually appealing
4. At least two `<meta>` tags (note that one of them is readily apparent, read #3)
5. A favicon. Must be 16px x 16px or 32px x 32px with either 8-bit or 24-bit color.
    a. The image must be converted to an ICO named `favicon.ico`.
    b. Some generators: http://www.favicon.cc/, http://www.dagondesign.com/tools/favicon-generator-tool/
6. At least one heading to make a nice looking title for the page(s).
    a. Change the color of the title to something that is not the default. You can choose any color you want, as long as we can see it.
    b. Change the font family to one that requires linking an external font. You can find cool fonts here, but you will need to "hotlink" them rather than download them.
    c. Change either the font style, or font variant. You can also do both if you wish.
7. Use **two layers** of an unordered list (`<ul>`) or an ordered list (`<ol>`). One layer can be ol and the other ul if you wish, or vice-versa, or they can both be ul or ol. **This means you must have a list within a list.**
    a. The "marker" or "bullet" used for each list item in the innermost list must be changed to something that is not the default. This must be implemented using the newer pseudo-element method.
8. Implement light and dark mode using only CSS. For any colors you use, use CSS variables. You do not need to implement a toggle, but if you do, this will knock out #10.
9. Demonstrate advanced typography:
    a. At least one instance of a more advanced text effect such as text gradients, text-shadow for depth, or multi-line text truncation with ellipsis. Specify what you did in the MANIFEST under 9a.
    b. Responsive typography that scales appropriately across different viewport sizes using relative units and/or the `clamp()` function. Specify what you did under 9b in the MANIFEST.
10. Implement basic JavaScript functionality:
    a. Create at least one interactive element using JavaScript event listeners
    b. Your JavaScript should enhance the user experience in a meaningful way (e.g., form validation, interactive navigation, content filtering, etc.)
    c. Properly separate your JavaScript code in an external .js file

    d. Demonstrate understanding of DOM manipulation by selecting and modifying at least one element
    e. Your JavaScript should follow proper practices (avoid inline JavaScript, use modern ES6+ syntax where appropriate)
    f. Specify what you did in the MANIFEST under JAVASCRIPT.

11. At least one standard size image.
    a. You will need to copy the image file (e,g. JPEG or PNG) into the same directory where your HTML and CSS files are, and link them appropriately. Do not hotlink.
    b. The dimensions of the image should be close to the maximum dimensions needed on the web page (do not use a 1000x1000 image if it will never be displayed larger than 500x500, this is wasteful). You must implement responsive image behavior using CSS:
        i. Images should adjust appropriately to different viewport sizes
        ii. Use properties like `max-width` to prevent overflow on small screens
    c. The image must have alt text for accessibility reasons.
12. Implement advanced visual styling using at least two of the following CSS techniques:
    a. CSS gradients (linear or radial) for creating visually interesting backgrounds
    b. box-shadow and/or text-shadow for creating depth
    c. CSS transforms for visual effects
    d. border styling beyond basic solid borders (e.g., image borders, gradients)
13. At least one responsive layout section using CSS Grid.
    a. the grid must have at least 3 columns and 2 rows using grid-template-columns and grid-template-rows
    b. include at least one element that spans multiple grid cells (using grid-column or grid-row properties)
    c. implement at least one responsive breakpoint using media queries where the grid layout changes (e.g., fewer columns on smaller screens)
    d. the grid should contain meaningful content related to your page theme
    e. style the grid cells with appropriate spacing (gap), borders, or background colors to make the grid structure visually clear
14. Implement at least one flexible layout using CSS Flexbox:
    a. demonstrate understanding of at least three Flexbox properties (such as flex-direction, justify-content, align-items, flex-grow, flex-shrink)
    b. the Flexbox layout should adapt appropriately to different screen sizes
    c. use Flexbox to solve a specific layout challenge (e.g., equal-height columns, vertical centering, or a card layout)
15. At least one element that uses a display property that contradicts its natural category (e.g. a block element that uses inline display, an inline that uses block display)
16. Fancy selectors
    a. at least one that uses >, + or ~.
    b. at least one attribute selector using ~=, *=, |= etc.
17. At least one form that contains
    a. a minimum of three different input types, including at least one HTML5-specific input type (e.g., date, email, range, color, search)
    b. appropriate labels for all form elements

    c. client-side validation using HTML5 attributes (required, pattern, min/max, etc.)
    d. proper form structure with fieldset and legend elements
    e. styling for form elements to match your page theme
    f. a submit button (though the form doesn't need to be functional for this project). **Please do not hardcode an email address here**
18. Demonstrate advanced inline text styling using at least one of the following techniques:
    a. Inline SVG for decorative text elements
    b. Sophisticated CSS pseudo-elements (`::before/::after`) for text decoration. If you use some other pseudo-element, tell us in the MANIFEST after marking "b".
19. Implement advanced visual styling using at least two of the following CSS techniques:
    a. CSS gradients (linear or radial) for creating visually interesting backgrounds
    b. Box-shadow and/or text-shadow for creating depth
    c. CSS transforms for visual effects
    d. Border styling beyond basic solid borders (e.g., image borders, gradients)
20. At least three links to other pages or external websites (more are great).
    a. Something about the style of the link must change after the link is clicked on for the first time
    b. something about the style of the link must change when the link is hovered over both in the visited or not-yet-visited state.
21. **There must not be any CSS conflicts, though overriding inherited styles is permitted.**
22. **Your work of art must be fully valid HTML5 and CSS, as demonstrated by the online W3C HTML Validator and the CSS validator.**

Many of these features were covered in class, and some were not. The purpose of this assignment is to sharpen your Web research skills to find the proper information to teach yourself these other concepts. This is how we all learn it, as you probably already know.

Getting Help and Feedback

We are here to help you during office hours, discussion or on Piazza. If there are ambiguities, confusions, or something that does not seem to work in the project spec, please post on Piazza so the instructor can intervene and either clarify or modify the expectations if necessary. Parts of this project may be autograded, but only if it can be tested consistently.

Submitting your Work

Github

Starting with Project 2, all projects will be submitted on a private Github that the TAs and I will manage. For this project, we encourage you to share your work publicly. You are welcomed to use a **public** (or private if you prefer) Github repository for your Project 1, and then submit the Github repo to Gradescope. You can also just submit a Zip file if you wish.

Regardless of which method you choose, you must include a text file in your submission:

MANIFEST

To make sure all of us are on the same page (no pun intended), the `MANIFEST` **must** be submitted. This is an inventory manifest that tells the grader where you implemented each requirement (if using multiple pages) and which features you chose to implement in cases where we let you decide which to implement. Thank you for your cooperation. Please download and use this template.

Zip File

If you submit your work in a Zip file, the file must be called `project1.zip`. For the grader's sanity, please make sure the archive extracts to a directory called `project1`.

```
project1.zip
|- project1
|-- index.html
|-- style.css
|-- image.png/jpg (any name, as long as linked properly from HTML)
|-- favicon.ico
|-- any other files that may be required
|-- MANIFEST
|-- README or README.md for Github only
```

Note that Github does not require this directory structure as we already get a directory from Gradescope.

Github

If you use Github, you should create a coherent README or README.md. We will not grade this.

Frequently Asked Questions

Q: Can I use Tailwind or Sass?
A: Your page needs to meet the spec. If you create two pages, you can absolutely use one of them to meet the basic CSS spec, and the other with Tailwind. Then, you can implement different requirements in each file, just tell us which ones are implemented where in the MANIFEST. If you're only creating one page, this can be problematic unless you somehow mix Tailwind with standard CSS but you still need to show the requirement of the spec.