Creative Project 2 Specification

Due: Friday, April 30th 2AM PST (**morning**)

New Submission Process (CP2 Onward)

We have been using CodePost (codepost.io) this term to submit your code assignments for grading and annotated feedback. Unfortunately, CodePost has some limitations unique to web development assignments, including corrupting files unexpectedly and limiting file sizes (you should still try to compress your images if possible to avoid large submissions).

Hopefully you've realized the potential of web programming to implement a variety of useful tools - our TA Gavin has done just that with a new interface system to minimize submission issues and streamline the student submission process. **Students are required to use the submission process described here for assignments CP2 and onward to ensure we receive all of your uncorrupted files** (you can still submit HW2 Part A and HW1 reworks on CodePost). If you have any questions, please ask on Discord.

Overview

For your second Creative Project, you will get a bit more practice with responsive flexbox layout (CSS) in addition to what we're learning about JavaScript (JS) to add interactivity to a web page using the DOM and event handlers. As a creative project, you have freedom to have more ownership in your work, as long as you meet the requirements listed below.

We encourage you to explore the new material covered in class, as well as related (but optional) content we may link to along the way, as long as you follow the CS 101 Code Quality guidelines. This is your chance to:

1. Continue to build a website that you can link to on your resume or code portfolio (CPs can be public, most HWs cannot be).
2. Ask Melissa and/or TAs about features you want to learn how to implement (we can provide more support for CPs than HWs) and ask for feedback/ideas on Discord.
3. Apply what you're learning in CS 101 to projects you are personally interested in and may use outside of just a CS 101 assignment.
4. Get feedback on code quality when learning new technologies in CS 101 to implement for the corresponding HW, which will be worth more points.

You may choose to do a new website for each CP, or build on a single project, as long as you meet each CP requirements (note that you do not need to meet any CP1 requirements in later CPs).

Ideas for CP2

As long as you meet the requirements outlined below, you have freedom in what kind of website you create. Here are some ideas (the staff is more than happy to help discuss more ideas on Discord or during office hours!):

- Continue to extend your portfolio page to add interactivity in some way.
- Write a website to implement drag/drop features.
- Write a list manager or something similar where you can add or remove items.
- Write something where you add or remove paragraphs or images to/from your page.
- Build a form that adds new features/options based on user input.
- Build a "create your own adventure" game that responds to the users choices.
- Implement a timeout/interval feature, etc.
- Write a website to visualize data structures like arrays, lists, or maps.
- Write a page implementing a small web-based game or puzzles.
- Write a website to help solve math/science/etc. formulae.

Getting Started

This Creative Project pieces together the languages you've learned in Module 1 as well as the basics of JavaScript you're learning about in Module 2 (responding to events, DOM manipulation, etc.). Knowing where to start with JS for the first time can be difficult (but exciting!) so this is how we suggest breaking down this assignment:

1. **Start with the HTML/CSS**: Write the HTML/CSS for your page first (recommended with a wireframe first). This gives you a front end to work with and reference before implementing any JavaScript code. Keep the DOM tree in mind as you are structuring your HTML. This will be useful as you move into JS implementation. You should aim to invest about 25-30% of the time you dedicate to CP2 on this part.
    - Tip: When supporting user interactivity to a webpage, you may find it useful to think about different UI elements (like buttons or input boxes) and can find a good list of common ones here for some inspiration.
2. **Break Down the UI**: This requires no coding! Remember, JavaScript is *event driven* meaning that everything occurs in response to an event such as a "click" of a button, a "change" of a radio button, a drag/drop of the mouse, a "keypress", etc. Spend the time thinking about what you want to implement and how you will do that. Something that you may find helpful is filling out a table like this. An example for a hypothetical page has been filled out for you:

event	element listening to event	response/elements changed
click	img with id #pet	img changes from cat image to dog image
...	...	...

The event in this case is a "click" event that takes place on an image with id #pet. The response functionality to the event is a change in the image from a cat to a dog.

Invest about 40% of the total time you expect to spend on this assignment on this part.

3. **Write the JavaScript**: Finally you are ready to write the JavaScript. Take it piece by piece and refer back to the brainstorming you did. If you need to reference proper syntax or forgot how to do something shown in lecture refer back to the slides/examples. Also, as you are coding, remember to use the console as demonstrated in lecture. It is a great way for you to experiment with JavaScript and get instant feedback. Aim to invest about 30-35% of the total time you expect to spend on the assignment on this part.

External Requirements

Overall External Requirements

- Your project must include the following three files at a minimum:
    - index.html - main page of your website
    - styles.css - file to style your .html file
    - *.js - a JS file (named appropriately of your choosing) containing your JavaScript code
- Similar to CP1, **All file names and links in your project must be lowercased without spaces** (e.g. img/puppy.jpg but not img/puppy.JPG or img/Puppy.jpg). This is enforced to avoid broken links commonly occurring in CP/HW submissions due to case-insensitivity of file names on Windows machines.
- Your page should include appropriate content and copyrights and citations. If we find plagiarism in CPs or inappropriate content, **you will be ineligible for any points on the CP**. Ask if you're unsure if your work is appropriate.

HTML Requirements

- Your website must contain at least one HTML page linked to your JS file to respond to page events as we've discussed in class.

CSS Requirements

- You must set a rule in your styles.css so that *at least* one element is a flex container.
- Additionally, you must use at least one of the flex properties (such as `justify-content`, `align-items`, `flex-wrap`) to create a more responsive layout.

JS Requirements

- Your JS program must respond to at least **2 different** events, including one on an HTML5 UI element that is not a `<button>` (though you may additionally add buttons to your page). Consider changes to a dropdown option, radio button or checkbox selection, etc. You can refer to the UI Element Reference slide deck for some common UI elements, but MDN has a good overview of more. For mouse/keyboard events, you can refer to **Keyboard Events** and **Mouse Events** here). If you would like to explore an event we don't cover in CS 101 on your project but don't know where to get started, feel free to ask.
- You should implement *at minimum* two named functions in your .js program. **All code must be your own work** (you may use the `id`, `qs`, `qsa`, and `gen` functions shown in lecture though; these do not count as the two function requirement).
- At least one of your callback (event handler) functions should change the document (DOM) in some way using `element.appendChild()`, `element.removeChild()`, or `element.replaceChild()`
- Callback functions should be non-trivial, meaning it must be possible they change the page or program "state" in response to the event (i.e. a `console.log` statement or unused variable assignment is trivial)
- You must use JS to dynamically modify at least one `classList` of an element (add/remove/toggle a class that is defined in a linked CSS file). This is a good chance to practice for HW2. Do not change `.style` properties unless they are for dynamically-computed styles (e.g. random hex color strings that are not easily defined in a CSS file).

Tips:

- Make sure to test your webpage UI so that it works properly when a user interacts with page elements - you aren't expected to handle *all* possible error cases, but part of your grade will come from being able to respond to an user event without an error.
- You can find a list of some different event types your page can listen for here (not comprehensive) and post on Discord or go to Office Hours if you want to explore those not covered in class!

Internal Requirements

- Your HTML, CSS, and JavaScript should demonstrate good code quality as demonstrated in class and detailed in the CS 101 Code Quality Guidelines. Part of your grade will come from using consistent indentation, proper naming conventions, curly brace locations, etc. Lecture/section examples demonstrate JS naming/whitespace conventions in this course, but there are more examples in the guide.
- When adding interactivity to your website, you should handle any events (like a mouse event, keyboard event, timer, etc.) by responding to them using a JavaScript function(s) in your .js file. You should not have any JavaScript code in your HTML and you should not have any HTML tags as strings in your JavaScript code (e.g. el.innerHTML = "<p>Foo</p>";).
- Your index.js file should be linked to your index.html or other .html files using <script defer src="..."> in the HTML <head>.
- Links to your .html, .css and .js files should be **relative links**, not absolute.
- Minimize styling in JS (e.g. changing the .style property of elements) - prefer adding/removing classes to DOM elements instead, and style the classes in your CSS. Remember that there is an exception when dynamically generating values for styles or positions that are not reasonably factored out in CSS.
- Any .js code you use must use the module-global pattern (recall the module-global template) and `"use strict"`;.
- Decompose your JS by writing smaller, more generic functions that complete one task rather than a few larger "do-everything" functions.
- Localize your variables as much as possible. You should not use any global variables (outside the module pattern) - see Code Quality Guide for reference.
- Only use module-global variables whenever absolutely necessary. Program constants are also good to remember to use (refer to the Skittles Case Study from Week 4)
- Use const with UPPER_CASED naming conventions (instead of let) for program constants (e.g. a file path to your images if you are working with many images in your JS).
- HTML and CSS files must be well-formed and pass W3C validation. This also includes any generated HTML as a result of DOM manipulation in JS (e.g. remember to include descriptive `alt` attributes when creating `img` elements dynamically).
- Your JS code must pass ESLint with no errors.
- If you want to explore other JS features beyond what we've taught in class, you must cite what resources you used to learn them in order to be eligible for credit. We strongly encourage students to ask the staff for resources instead of finding online tutorials on their own (some are better than others).

While note required, you are encouraged to implement a feature to execute a callback with after a delay (using `setTimeout`) or repeatedly (using `setInterval`). You will need to use these functions in the last part of HW2, so this is a good chance to practice!

Documentation

Place a comment header in each file with your name, section, a brief description of the assignment, and the files contents. Examples:

HTML File:

```
 <!--
     Author: Lorem Hovik
     CS 101 Spring 2021
     Date: April 1st, 2021

     This is the index.html page for my portfolio of web development work.
     It includes links to side projects I have done during CS 101,
     including an AboutMe page, a blog template, and a cryptogram generator.
-->
```

CSS File:

```
 /*
    Author: Lorem Hovik
    CS 101 Spring 2021
    Date: April 1st, 2021

    This is the styles.css page for my portfolio of web development work.
    It is used by all pages in my portfolio to give the site a consistent
    look and feel.
*/
```

Use JSDoc to document *all* of your JS functions with @param, @returns as discussed in the Code Quality Guide.

```
/**
 * Brief description of the function (including requirements, important
 * exceptional cases, and avoiding implementation details).
 * @param {datatype} parameterName1 - parameter description
 * @param {datatype} parameterName2 - parameter description
 * @return {datatype} Description of the return value (if applicable)
 */
 function functionName(parameterName1, parameterName2) {
   ...
 }
```

Your JS file should also have your student information and an overview of the program in a file comment.

Grading

Our goal is to give you feedback, particularly on the internal requirements and style and documentation, so you can incorporate this feedback in your homework assignments which will be worth more towards your final grade.

This CP will be out of 9 points and will be distributed as:

- External Correctness (4 pts)
- Internal Correctness (4 pts)
- Documentation (1 pt)