Homework 2:
Stylish Scripting

CSCI 4131: Internet Programming (Fall 2022)

Deadline

The deadline for this assignment is October 7th at 11:55pm. It can be turned up to 24 hours late for a 10% deduction.

1 Change Log

- V1.0 (2022-09-23) Initial Version.
- V1.1 (2022-09-29) Updated Example in the timeUntil funciton in the widget page.

2 FAQS

- None yet.

3 Introduction

While HTML (seen in HW1) is the back-bone of a website, a website is rather bland if it's *only* HTML. To make a fully functional front-end experience requires HTML working in concert with CSS (to make it look good) and JS (to make it behave interestingly) This assignment will build upon the HTML you wrote in HW1 in three key ways:

- Minor HTML improvements (to set the stage for...)
- A complete visual overhaul using CSS
- A small amount of interactive behavior using javascript.

While we will primarily list javascript and css requirement changes be aware that this might require HTML changes to enable them. For example, you may need to restructure something from a list, to a series of divs, or you may need to add classes, or ids to your HTML in order to make it style-able.

3.1 Learning Goals

This assignment is designed with a few learning goals in mind. By doing this assignment you will:

- Practice applying a wide-range of stylistic changes in css, including both low-level "color and font style" style changes, as well as larger layout and "look/feel" changes.
- See how *sometimes* small HTML changes can be needed to make use of advanced CSS layouts.
- Build confidence working with CSS on a more realistically sized website.
- Apply some open-ended style led only by your *design intuition*
- Use javascript to perform some basic computations
- Develop your own javascript widgets to build a standard web-widget.

4 Requirements

This assignment builds upon your HW1 submission. While there will be some overlapping requirements (I.E. valid HTML and valid CSS) *most* of the requirements remain independent. If you lost points on HW1 and worry it will cause you to lose points on HW2 please reach out to course staff so we can help you resolve those issues before you undertake the changes for HW2.

The requirements for HW2 are broken into 4 parts:

1. General style overhaul
2. About me page specific style overhaul
3. Contacts page improvements
4. Widgets page additions.

You can do these in any particular order, but you may find this order matches the order we've covered material particularly well.

4.1 General Style Overhaul

These requirements reflect CSS changes that should be applied to all pages in your website.

- You should create a file `myStyle.css` which you load from all three CSS pages. This should replace any embedded or inline CSS you used to use.
- You should not have any other CSS (embedded, inline, or linked) than this one file (except on the widget page to enable various widgets. More on that later). You are **Expressly forbidden** from using any CSS libraries or CSS written by other people (except for those provided for the widgets)
- The `myStyle.css` page must create a consistent visual look for your webpage. This must include:
  - A background color for the whole page
  - A separate visually distinct background color for the navigation menu of links at the top of the page.
  - (If you have any footers, or sidebars on your webpage, these too should be visually distinct)
- The links at the top of each page should be updated and restyled
  - The nav bar (containing the links – see the images below) should be full width and sit at the top of your webpage.
  - The nav bar no longer needs to have an outline. Instead it should be *visually distinct* from the background-color of your webpage. You're allowed some creativity here, but at a minimum we're looking for the nav bar to have a distinct color
  - The links in the nav bar should be spaced out as seen in the screenshots below. This should generally respond to window sizes – I.E. this cannot be done by simply hard-coding large margins, but must be done with a proper layout tool
  - The links in the nav bar should additionally be styled. This doesn't need to match my styles/shapes in the images below, but they should be more than just floating text (the default for the a tag.)
  - **Extra credit** When you mouse-over a navigation link, it's background color should change to subtly indicate that you've moved your mouse over a link.

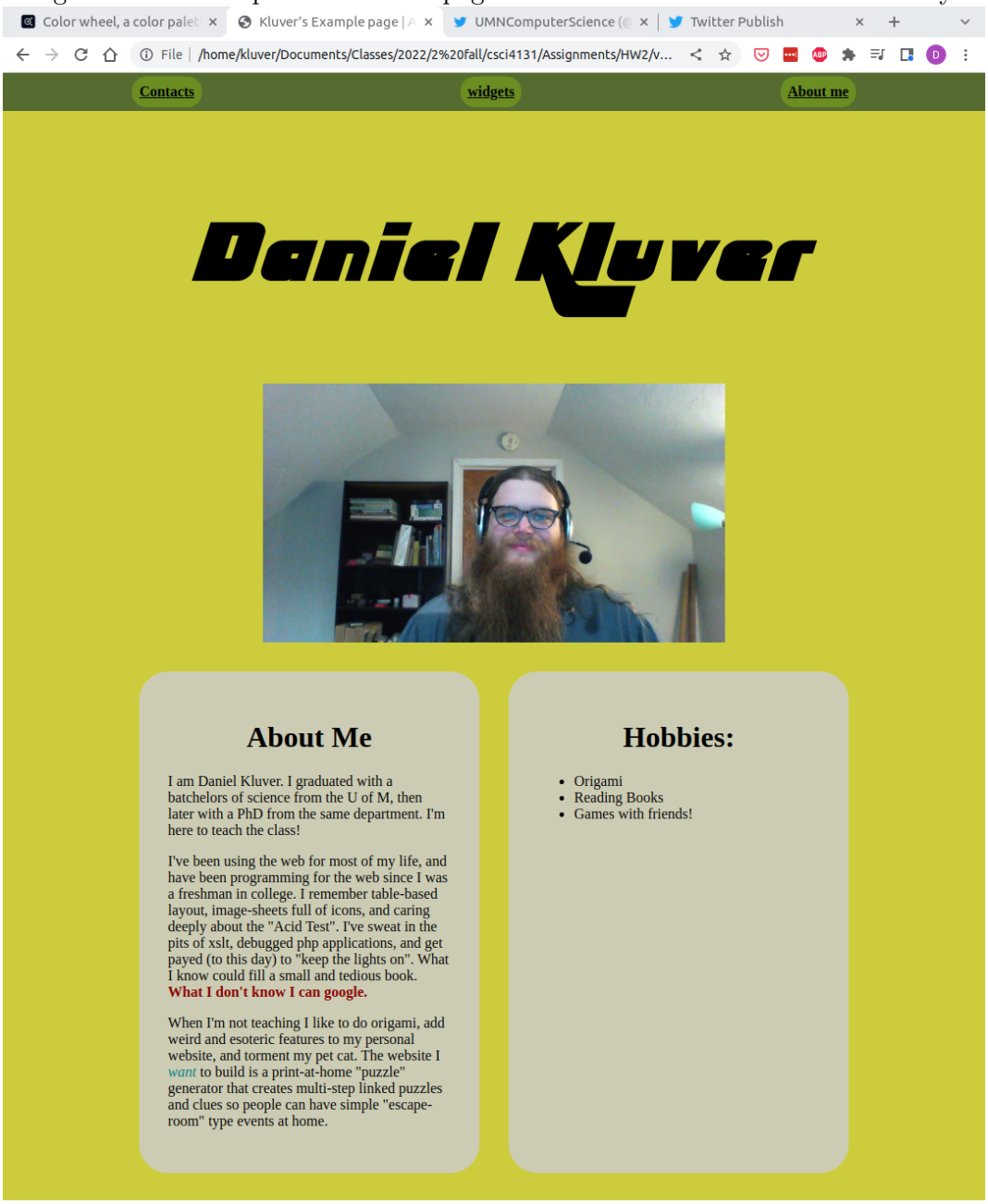
Some hints and guidelines:

- Most browsers add default margin to the body tag. This often makes it hard to do full-page layouts that look right. You may find you need to set the margin on the body tag to 0 to get full-page styles.
- All pages should generally look correct on a screen 1920 pixels wide. It's OK if your monitor is not this wide (we can always test on a *Smaller* window if we need to) but your layouts must not require a screen larger than 1920 pixels to be viewed correctly. As much as possible please support a wide range of screen sizes by using relative-units rather than hardcoded pixel values.
- Across the various pages of this assignment you will have some creative control over colors, fonts, etc used. **Use good judgment here**. Remember that we're going to have too look at these, and styles that are visually unpleasant and interfere with the grading process may lead to points being taken off. The most common way to do this would be to choose background colors that do not have a good contrast with your text, making your text hard to read. When in doubt – be cautious to make the website easy-to-use even if it means compromising your artistic vision.

4.2 About Me Page Overhauls

For HW2 you must update your about Me page. This will involve minor HTML edits, and content generation, and a large amount of CSS improvement. To begin, take a look at the example page below:

Figure 1: An example "about me" page. You don't have to match this exactly.



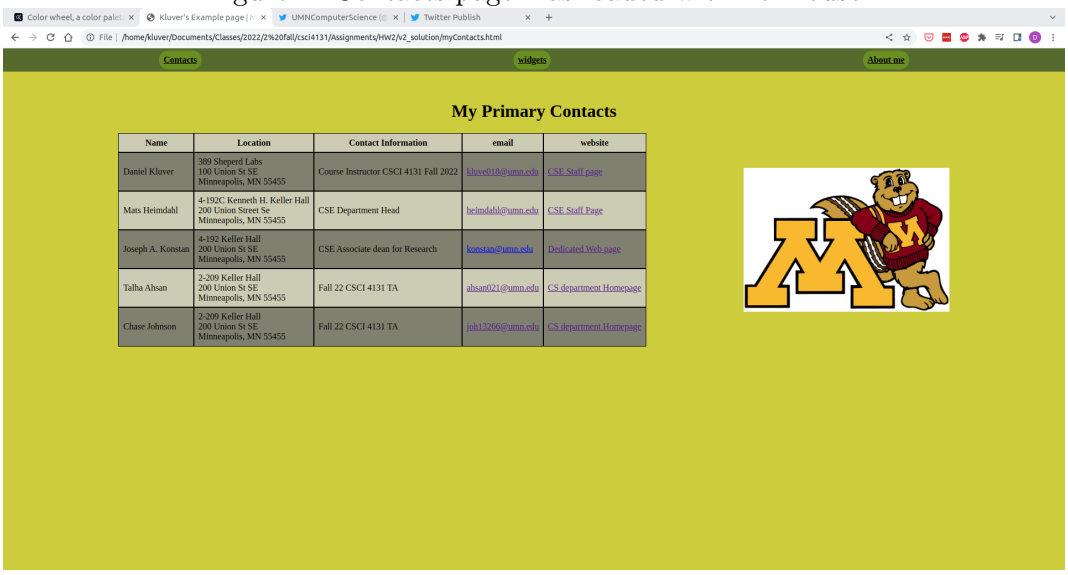
For the AboutMe page, you must add CSS to the `myStyle.css` page to accomplish the following:

- The header with your name should be much larger than normal text, should use a different font than the rest of the text, and should be centered in the window.
  - Use your judgment on fonts and sizes here. Remember that we want things to look decent on a screen around 1920 pixels wide, and we should be able to actually read whatever font you're using.
  - Your name must remain centered across a range of screen sizes. (I.E. this must be properly centered, you can't just hardcode large margins)
  - For credit you do not need to use a custom font as seen in the example image – any "standard" font will do so long as it's different from the font used elsewhere on the page.
  - (Hint – remember you can google how to use the css font properties – there are many guides available here)
  - For *extra credit* use a custom font (I.E. one that has to be loaded from a font-file). There are many websites that will help you find a suitable font, and can also help you craft the CSS to load it.
- The image should be half as wide as the screen (this should remain true at all screen sizes – not only 1920px) And must be centered.
- Under the image you should have a two-column layout as seen in the example:
  - Each column should be visually distinct with a different background color than the overall page background.
  - Each column should have rounded corners and should have some space both between columns and between the columns and the page-side
  - The columns should have the same height. This cannot be a hard-coded height value – instead you should use CSS to ensure that both columns have the same height (the height of the larger column). You will lose points if you hard-code a height for this rather than using a proper layout tool.
  - Both columns should have a centered header using an appropriate HTML tag.
  - The left column should have the "three paragraphs" of autobiographical text you produced for HW1.
  - The right column should have some list of information about you: I went with "hobbies", but you can also list places you've visited, your favorite movies or music, or just today's grocery list. We aren't grading your content here.
  - The columns should both have left-aligned content (other than the header)
- Finally, the bold text, and the italic text in your self-description should be colored to provide appropriate contrast. While this is subjective (and won't be directly graded) try to make the bold provide a stronger contrast (more emphasis) than the italic text.

4.3 Contacts Page improvements

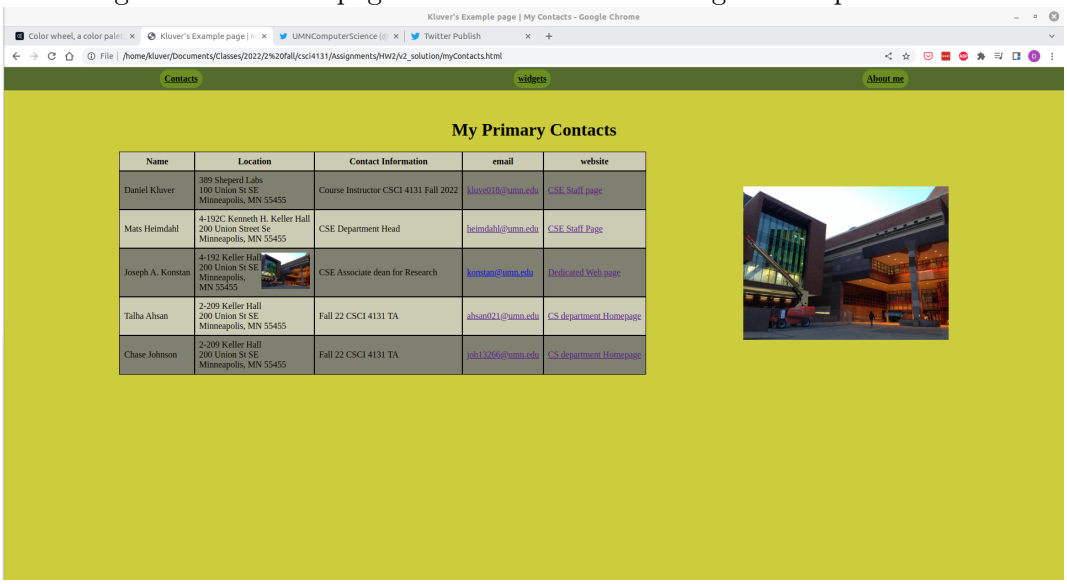
The contacts page will involve both CSS and Javascript changes. The new basic layout of the contacts page can be seen in the image below

Figure 2: Contacts page – as loaded with no mouse



Importantly the new contacts page will have dynamic behavior as well, which responds to your mouse movement. In the below example, my mouse cursor (which doesn't show up in screenshots) was over the middle-row of my table.

Figure 3: Contacts page With the mouse hovering over a specific row.



4.3.1 CSS Changes

Update `myStyle.css` to accomplish the following:

- The rows of your contacts table should be displayed in alternating colors. These colors should be chosen to enhance readability, therefore they should be distinct from both the background color, and each other, and have reasonable contrast with the font.
- The layout should be modified so that the contacts table is on the left, and (when the page loads) the provided placeholder image is on the right. These two elements should be sized to fit a screen 1920 pixels or wider.

4.3.2 Behavior 1

When you hover over a row in the contacts table with a pointing device (e.g. a mouse or finger on touch-screen) a small (thumbnail) image of the corresponding location should be displayed in the "Location" column of the table.

- Use your judgment for how big/small of a "thumbnail" should be used here.
- You can use any reasonable image so long as you're obeying copyright law. You can also use images you've taken, or images from the provided image pack so long as they reasonably reflect the location chosen.
- It's OK if the table layout adjusts a bit when the thumbnail appears.

This can be done in CSS using careful selectors and meta selectors, or JavaScript. We do not require a specific approach to achieving this behavior.

4.3.3 Behavior 2

Additionally, when you hover over a row in the contacts table, the placeholder image should be replaced with a larger version of the image you've chosen to correspond to the location for that contact.

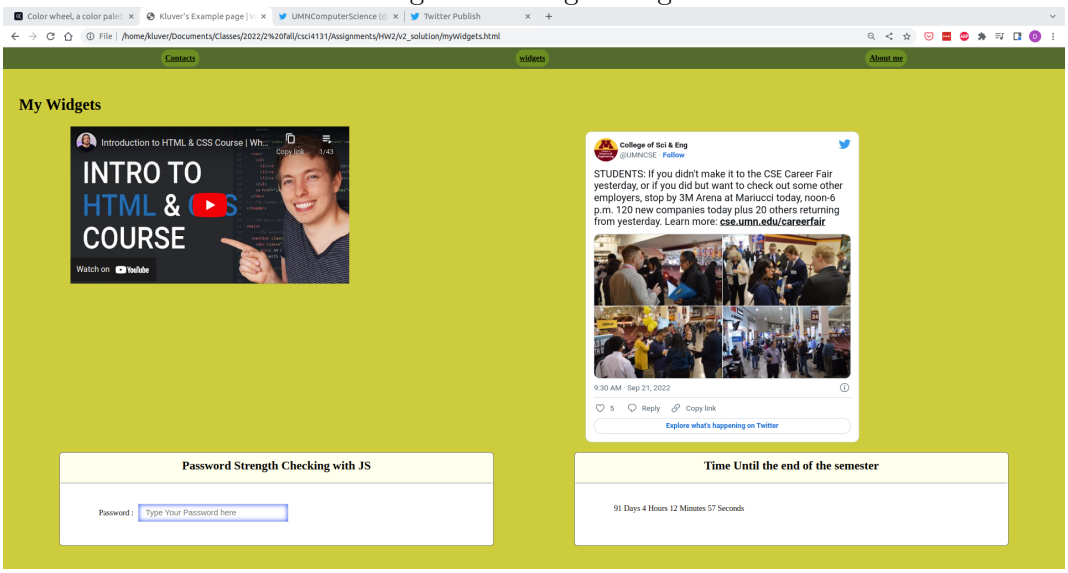
- This placeholder image should be updated each time you change what you're hovering over – so if you move your mouse up/down the table, it should swap from one location-image to another.
- As the mouse moves from row to row, the image should retain width and general size (It's OK if height changes slightly in response to image changes if images do not have the same aspect ratio).
- If the mouse moves off the table you should have the image revert to the placeholder image.

This requirement must be done using Javascript. We recommend looking into the mouseenter and mouseleave events. You may add this javascript either as embedded javascript, or in an external file.

4.4 Widget Page Additions

The widgets page will get a slight layout change, followed by some new javascripty widgets.

Figure 4: Widgets Page



4.4.1 Layout Changes

You must modify the layout of the widget page based on the image above.

- You should modify the page to no longer use an html list.
- Your new layout should be generally two-columned, with space between elements both horizontally and vertically.
- The widget order/position should match what is seen in the image above, youtube, twitter, password, countdown (left-to-right, top-to-bottom).
- You should do this in a way that allows flexibly adding new widgets, and does not require an even number of widgets.
- You will lose points for using an HTML table to accomplish this. You are required to use a CSS layout tool to make this happen.

4.4.2 Password Strength Checker Widget

You will be adding a password strength checking widget. To make this easier (And more consistent) we've provided 3 files, a javascript starter-file, a css file, and an html file.

- You should start by downloading the password widget files. Then follow the instructions in passwordWidget.html to add the widget html, css, and javascript to your code. You should not need to modify the css or html at this point.
- Modify the provided javascript. All you should modify is the checkStrength function. Instructions for the expected behavior of this function are provided in the passwordWidget.js file. This function implements a series of checks to measure how "strong" a password is, and returns a javascript object with a className value (which sets the css class for the password strength indicator) and a message value (which sets the text of the password strength indicator). Specific expectations for this are provided in the passwordWidget.js file.

- You should not have to modify the rest of the provided javascript file, but you may find it interesting to review it so you can see how it works.

4.4.3 Time Left Widget

Your second widget will be a constantly-updating indicator of how many days, hours, minutes, and seconds are left in the Fall 2022 semester.

- You must make this widget "from scratch" – you are allowed to re-use any part of the provided widget HTML/CSS/Javascript to accomplish this goal.
- This widget should be visually similar to the password widget. You will need to copy, rewrite, or improve the CSS provided for the password widget to accomplish this. (This is formally required. That CSS file uses ID selectors to apply styles, and html IDs are expected to be unique. Therefore you can't just re-use it directly.)
- You must create a file timeWidget.js to store your javacsript for this widget
- Your file must contain a constant variable "endOfSem" which is a date object representing December 22nd, 2022 (the end of the semester in the official calendar)
- Your file must contain a function named "timeUntil". This function takes one parameter – a javascript date object – and returns a javascript object with properties days, hours, minutes, and seconds. These values should be integers representing how many days, hours, minutes, and seconds until the end of the semester from the date provided. These values should be rounded towards 0 and within the appropriate range for the unit. As an example, if given 2:21:12PM on September 22nd, the return value should be

```
timeUntil(new Date(2022, 08, 22, 2, 21, 12))
{days: 90, hours: 22, minutes: 38, seconds: 48}
```

- You can ignore any possible effects of leap-seconds or other mathematically inconvenient chronological discontinuities.
- You will need to write another function which calls your timeUntil function with a date object representing the current time, and then displays the results into your widget's HTML. You can see the code we used in the provided code for password checking for an example of how to store the results of a javascript computation into an html element by ID. Note – the formatting must match the example "90 days 22 hours 38 minutes 48 seconds". (You can name this function whatever you want – we won't be *directly* testing it.)
- Finally, use the setInterval function (you can read about this in zybooks 7.5 or online through resources like w3schools and mdn.) to make this final function run once per second. This should start automatically as the page loads, and run until the page is closed

5 Grading information and requirements recap

The assignment will (tentatively) be graded out of 100 total points.

- Overall (20 points)
  - Submission contains one main folder with all content, 3 html (correctly named) files, myStyle.css, as well as javascript and css files needed for the widgets, and finally, and one "resources" folder with images in it as needed.
  - (3 points) Each page passes HTML validation without errors
  - (6 points) Each page passes CSS validation without errors
  - (5 points) HTML, CSS, and Javscript code is readable, with appropriate tabbing etc.
  - Only myStyle.css is used in all places (other than widget-specific CSS libraries)
  - (6 points) background and navigation bar style upgrades.
  - (2 points EC) for navigation link interactive behavior
- About me (20 points)
  - (3 points) header is large centered, and has an updated font.
  - (2 points) image is resized and centered
  - (3 points EC) header is in a custom web font.
  - (10 points) Two column layout with headers for both columns
  - (2 points) bold and italic content is recolored
  - (3 points) content upgrades (list with a few items, any other content updates you need to make)
- Contacts (30 points)
  - (4 points) Table has alternating row-colors and layout has been updated
  - (6 points) "behavior 1" – when the mouse is over a row the thumbnail shows up in the location field.
  - (15 points) "behavior 2" – when the mouse is over a row the preview image is updated.
  - (5 points) When the mouse leaves the table – the preview image reverts to goldie.
- Widgets (30 points)
  - (3 points) Widget page layout has been updated
  - (2 points) Password widget is inserted correctly
  - (10 points) Password widget behaves correctly and checkStrength has been implemented correctly.
  - (3 points) Time left widget exists, and is styled the same as the password widget.
  - (2 points) timeWidget.js contains a constant variable endOfSem that correctly indicates December 22nd 2022.
  - (5 points) timeWidget.js contains a function timeUntil that behaves as described.
  - (5 points) The time widget automatically runs every second displaying the time until the end of the semester in the widget

6 Submission information.

Your process for submitting should be as follows:

1. Check your work against the above requirement checklist one last time.
2. Create a zip file of the folder containing your assignment.
3. Upload the zip file to canvas.
4. Download your submission from canvas
5. Unzip it (where it is, usually in your downloads folder)
6. Confirm that the site still works (so all resources are included and link correctly even when the folder is in a different place on your machine)