Overview

You are given a file named $journal.html$ that represents a page from Victoria's journal. You are to write a stylesheet called $journal\_layout.css$ that will transform the layout in different ways throughout the following exercises. Finally you will upload your work onto GitHub and share the link to your public repository in a text file.

Exercise 1: Arrange Your Page into Sections

The first task is to organize journal.html by adding ids, classes, spans and divs. Then, add "boxes" around these sections of the website by adding to your journal_layout.css stylesheet.

You are going to match the output image below.

- The **borders** are all 5px thick and solid.
- The **colors** are the intuitive HTML color names, e.g. the red border is the HTML color red.
- *Note:* You should not need define a class or id specifically for the h2s on this page.

Exercise 2: Spacing with Padding and Margins, Backgrounds

You are now going to add padding, margins, and backgrounds to some of the parts you defined in Exercise 1. You should only have to change your journal_layout.css file.

- The box with the **green** border should have a background color of white.
- The boxes with the **blue** borders should have a background color of #E8FBFB.
  It should have a padding of 5px (on all sides) and margin of 10px only on the *top* of the box (the margins for the remaining sides should be left at 0px).
- The **overall page content** area should become centered on the page, should have left and right margins of 10% and a background image using the image provided in the lab resources.

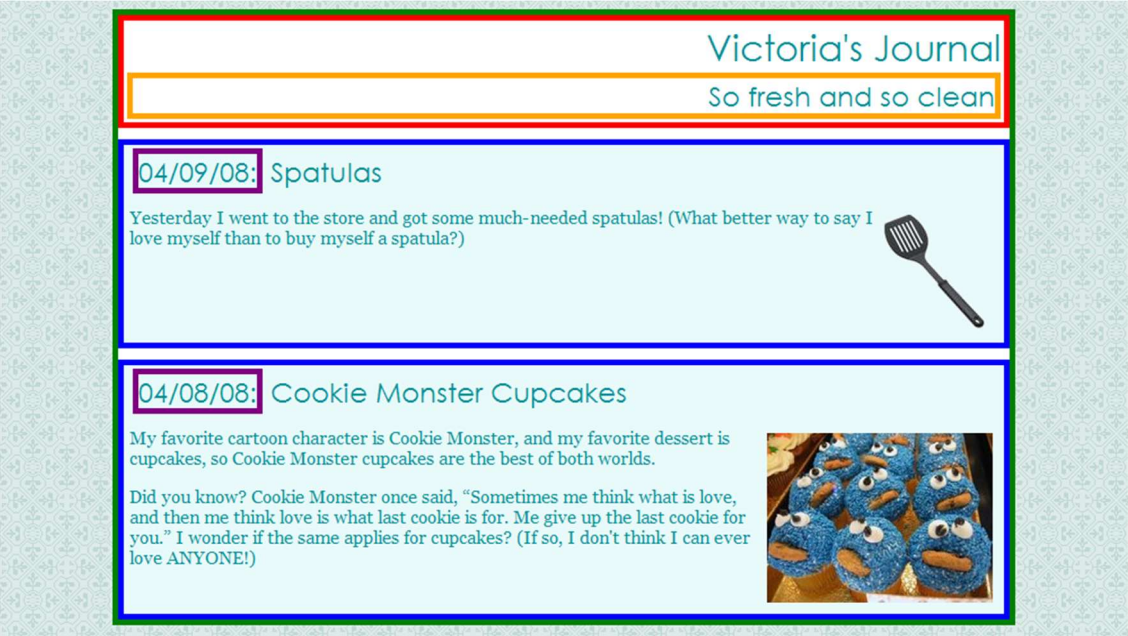
You are going to match the output below:

Exercise 3: Float, Alignment and Clear

Now you're going to practice float, clear, and alignment on the web page. Part of this exercise is understanding the difference between aligning and floating an element. You may have to edit your journal.html code as well as your journal_layout.css stylesheet to get the floats working properly.

- The heading text in the **red** box should appear on the **right** side of that section of the page.
- The journal entry images should hover on the **right** side next to the surrounding text. The image should stay within the bounds of the blue box; that is, it should not *bleed* into the other content below it.

You are going to match the output below:



Exercise 4: Cosmetic Finishing Touches

Finally, we add some finishing touches to make the page look its best.

- Change the border of the box with the **green** border to be a solid, white, 10px-thick border.
- Change the border of the boxes with the **blue** border to have a solid, 4px-thick border, using the hex value #C2E9E9 for its color.
- Change the border of the box with the **purple** border to have *only* a bottom border, and let that bottom border be blue, dashed, and 2px-thick.
- Change the background color of the box with the **red** border to be #A8F0F0 and get rid of its border altogether.
- Change the font size of So fresh and so clean to 14pt and get rid of its border.

You are going to match the output below:

Exercise 6: Add 2nd Column

You should copy and paste the following code into your journal.html:

```
<h1>Friends</h1>
<ul>
          <li><a href="http://www.youtube.com/">190m Buddies</a></li>
          <li><a href="https://web.whatsapp.com/">Big Will</a></li>
          <li><a href="https://www.google.com/" title="Conan the Librarian">Conan the Librarian</a></li>
</ul>
```

Use the appropriate layout-related tags/attributes and CSS to make this list into a second, left-aligned column as shown below. The colors, borders, etc. of the list are not important; the focus is on the layout. **The layout with a second column must still be a liquid layout** -- that is, all parts of it should adjust in size accordingly when the browser size changes.

HINT: When multiple elements float in the same direction, they arrange themselves into columns.

You are going to match the output below:

Exercise 7: Upload Your Page to Github

"Git and GitHub are two technologies that every developer should learn, irrespective of their field"

This lab section will help you understand what Git and version control are, the basic Git commands you need to know, how you can use its features to boost your work efficiency, and how to extend these features using GitHub.

What is Git?

Git is a version control system which lets you track changes you make to your files over time. With Git, you can revert to various states of your files (like a time traveling machine). You can also make a copy of your file, make changes to that copy, and then merge these changes to the original copy.

For example, you could be working on a website's landing page and discover that you do not like the navigation bar. But at the same time, you might not want to start altering its components because it might get worse.

With Git, you can create an identical copy of that file and play around with the navigation bar. Then, when you are satisfied with your changes, you can merge the copy to the original file.

Install Git

In order to use Git, you have to install it on your computer. To do this, you can download the latest version on the official website. You can download for your operating system from the options given.

To verify Git was installed, you can run this command on the command line: git --version. This shows you the current version installed on you PC.

Configure Git

To set your username, type and execute these commands: `git config --global user.name "YOUR_USERNAME"` and `git config --global user.email "YOUR_EMAIL"`. Just make sure to replace `"YOUR_USERNAME"` and `"YOUR_EMAIL"` with the values you choose.

Create and Initialize a Git Repository

(Note: the lab work files you have done previously should be stored in a folder and you've opened this folder using VS Code)

To initialize your project, simply run `git init`. This will tell Git to get ready to start watching your files for every change that occurs. It looks like this:

A repository is just another way to define a project being watched/tracked by Git.

What is GitHub?

GitHub is an online hosting service for Git repositories. Imagine working on a project at home and while you are away, maybe at a friend's place, you suddenly remember the solution to a code error that has kept you restless for days.

You cannot make these changes because your PC is not with you. But if you have your project hosted on GitHub, you can access and download that project with a command on whatever computer you have access to. Then you can make your changes and push the latest version back to GitHub.

In summary, GitHub lets you store your repo on their platform. Another awesome feature that comes with GitHub is the ability to collaborate with other developers from any location.

Now that we have created and initialized our project locally, let's push it to GitHub.

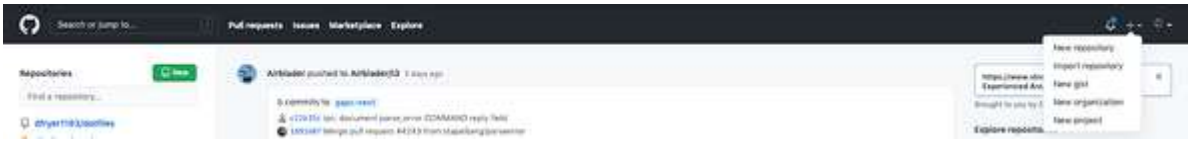
Push a repository to GitHub:

Create a GitHub account

To be able to use GitHub, you will have to create an account first. You can do that on their website.

Create a repository

You can click on the + symbol on the top right corner of the page then choose "New repository". Give your repo a name then scroll down and click on "Create repository".



Add files in Git

When we first initialized our project, the file was not being tracked by Git. To do that, we use this command `git add .` The period or dot that comes after `add` means all the files that exist in the repository. If you want to add a specific file, maybe one named `about.txt`, you use `git add about.txt`.

You will not get a response after this command, but to know what state your file is in, you can run the `git status` command.

Commit files in Git

The next state for a file after the staged state is the committed state. To commit our file, we use the `git commit -m "first commit"` command.

After executing this command, you should get a response similar to this:

```
C:\Users\SamiSh\Desktop\Lab 4>git commit -m "first commit"
[master (root-commit) 3299be6] first commit
 5 files changed, 92 insertions(+)
 create mode 100644 images/background.gif
 create mode 100644 images/cookiemonster.jpg
 create mode 100644 images/spatula.jpg
 create mode 100644 journal.html
 create mode 100644 journal_layout.css
```

Push the repository to GitHub

After you create the repo, you should be redirected to a page that tells you how to create a repo locally or push an existing one.

In our case, the project already exists locally so we will use commands in the "…or push an existing repository from the command line" section. These are the commands:

```
git remote add origin https://github.com/annie/git-and-github-tutorial.git

git branch -M main

git push -u origin main
```

The first command `git remote add origin https://github.com/...` creates a connection between your local repo and the remote repo on Github.

Note: The URL for your remote project should be entirely different from the one above.

The second command `git branch -M main` changes your main branch's name to "main".

The last command `git push -u origin main` pushes your repo from your local device to GitHub.

After adding the new task, run the `git status` command. Try to modify a file in your project and rerun the status command and observe.

Submit

After creating the GitHub repo, copy the URL of your repo and insert it into a text file, and upload it along with your work onto moodle.