HTML, CSS

More on Git

- 'Branching' is a feature of Git that allows a project to move in multiple different directions simultaneously. There is one `master` branch that is always usable, but any number of new branches can be created to develop new features. Once ready, these branches can then be merged back into `master`.
- When working in a Git repository, `HEAD` refers to the current branch being worked on. When a different branch is 'checked out', the `HEAD` changes to indicate the new working branch.
- When merging a branch back into `master`, there is the possibility for merge conflicts to arise. These can be resolved in the same way discussed in Lecture 0.
- Some Git commands related to branching:
  - `git branch` : list all the branches currently in a repository
  - `git branch <name>` : create a new branch called `name`
  - `git checkout <name>` : switch current working branch to `name`
  - `git merge <name>` : merge branch `name` into current working branch (normally `master`)
- Any version of a repository that is not stored locally on a device is called a 'remote'. 'Origin' is used to refer to the remote from which the local repository was originally downloaded from.
- Some Git commands related to remotes:
  - `git fetch` : download all of the latest commits from a remote to a local device
  - `git merge origin/master` : merge `origin/master`, which is the remote version of a repository normally downloaded with `git fetch`, into the local, preexesiting `master` branch
    - Note that `git pull` is equivalent to running `git fetch` and then `git merge origin/master`
- A 'fork' of a repository is an entirely separate repository which is copy of the original repository. A forked repository can be managed and modified like any other, all without affecting the original copy.
- Open source projects are often developed using forks. There will be one central version of the software which contributors will fork and improve on, and when they want these changes to be merged into the central repository, they submit a 'pull request'.
- A pull request can be made to merge a branch of a repository with another branch of the same repository or even a different repository. Pull requests are a good way to get feedback on changes from collaborators on the same project.
- Note that forks and pull requests are both GitHub specific features.

More on HTML

- More useful HTML tags:
  - `<a href="path/to/hello.html">Click here!</a>` : link to `hello.html`, some URL, or some other content marked by `id` by passing `#id` to `href`
  - `<input type="radio"> Option 1` : radio-button option for a form, where only 1 out of all the options may be selected ``` html
- There are lots of new useful tags with HTML5, but not all browsers, especially older browsers, will support these new features. Nonetheless, these new features can be used with increasing confidence that they will be rendered appropriately for a significant portion of users.

More on CSS

- CSS selectors are used to select different parts of a website to style in particular ways.
- Some common CSS selectors:
- Select `h1` and `h2`

```
h1, h2 {  color: red;  }
```

- Select all `li` that are descendants of `ol` (not necessarily immediate descendants

```
ol li {  color: red;  }
```

- Select all `li` that are immediate children of `ol`

```
ol > li {  color: red;  }
```

- Select all `input` fields with the attribute `type=text`

```
input[type=text] {  background-color: red;  }
```

- Select all `button`s with the pseudoclass `hover`

```
button:hover {  background-color: orange;  }
```

  - A 'pseudoclass' is a special state of an HTML element. In this example, the state is whether or not the cursor is hovering over a button.
- Select all `before` pseudoelements of the element `a`

```
a::before {  content: "\21d2 Click here: ";  font-weight: bold;  }
```

  - A 'pseudoelement' is a way to affect certain parts of an HTML element. In this example, the `before` selector applies `content` with its included styling before the contents of all `a` elements.
  - `\21d2` is a hexadecimal value for a Unicode icon, which can represent symbols like emoji.
- Select all `selection` pseudoelements of the element `p`

```
p::selection {  color: red;  background-color: yellow;
}
```

Responsive Design

- Responsive design is the idea that a website should look good regardless of the platform its viewed from.
- One way we can do this is by using a 'media query':

```
  <style>       @media print {           .screen-only {
display: none;           }       }   </style>   <body>       <p
class="screen-only">This will not appear when printed</p>   </body>
```

  - `@media` is a media query, which means the following CSS will be applied only in certain situations, namely, when the webpage is being printed. `.screen-only` is a class selector which identifies what content we want to be print only

```
  @media (min-width: 500px) {   body {       background-color: red;
}   }   @media (max-width: 499px) {   body {       background-color:
yellow;   }   }
```

  - When the width of the screen is at least 500px, the background color of `body` will be red, while if it is less than 499px, the background color of `body` will be yellow.
  - In order to interact with the screen size, the following must be included in `head`: `<meta name="viewport" content="width=device-width, initial-scale=1.0">`
    - `viewport` is the visible area on which the screen is being displayed. `content` refers to the entire webpage the `width` of which is being set to `device-width`.
- Another tool is 'flexbox'. Flexbox allows for the reorganization of content based on the size of the viewport.

```
  .container {       display: flex;       flex-wrap: wrap;   }
```

  - By setting `display: flex` and `flex-wrap: wrap`, content will wrap vertically if necessary, so no content is lost when the width of the screen is shrunk.
- A grid of content can be achieved in a similar fashion.

```
  .grid {       display: grid;       grid-column-gap: 20px;
grid-row-gap: 10px;       grid-template-columns: 200px 200px auto;   }
```

  - By setting `display: grid`, all the different characteristics of a grid layout can be used to format content. In particular, when defining `grid-template-columnns`, the final column can be set to `auto`, filling up however much screen space may be left. If multiple columns are set to `auto`, they will equally share the remaining space.

Bootstrap

- Bootstrap is a CSS library written to help make clean, responsive, and nice-looking websites without having to remember the gritty details about flexboxes or grids everytime a layout needs to be set up.
- The only thing needed to use Bootstrap is by adding a single line which links Bootstrap's CSS stylesheet: `<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.1/css/bootstrap.min.css" integrity="sha384-WskhaSGFgHYWDcbwN70/dfYBj47jz9qbsMId/iRN3ewGhXQFZCSftd1LZCfmhktB" crossorigin="anonymous">`.
- Bootstrap's CSS will make everything look a little cleaner and more modern, but its real power comes with its layout system. Bootstrap uses a column-based model where every row in a website is divided into 12 individual columns, and different elements can be alloted a different number of columns to fill.
- Bootstrap's columns and rows are referenced in HTML with `class="row"` and `class="col-3"` attributes, where the number after `col-` is the number of columns the element should use.
    - Elements can take up a different number of columns based on the size of the screen with attributes like `class="col-lg-3 col-sm-6`. On a small screen, 6 columns will be used, but in a large screen, 3 columns will be used. If another row has to be added, Bootstrap will do so automatically. This is a much easier alternative to something like flexbox (Bootstrap does so behind the scenes).
- Bootstrap has a whole host of other pretty components which can easily be applied by simply adding the appropriate `class` attribute to an element. See Bootstrap's documentation for an extensive list.

Sass

- Sass is an entirely new language built on top of CSS which gives it a little more power and flexibility when designing CSS stylesheets and allows for the generation of stylesheets in a programmatic way. Ultimately, Sass just makes writing CSS easier.
- In order to use Sass, it must first be installed. Once installed, we can execute `sass style.scss style.css` to compile our Sass file `style.scss` into `sass.css`, which can actually be linked to and interpreted by an HTML file.
  - If recompiling gets annoying, `sass --watch style.scss:style.css` to automatically recompile `style.scss` as `style.css` whenever `style.scss` is modified. Additionally, many website deployment systems, like GitHub Pages, have built in support for Sass. For example, if an `.scss` file is pushed to GitHub, GitHub Pages will compile it automatically.
- One feature of Sass is variables, which are defined as so: `$color: red;`. Anywhere `$color` is passed as a value for a CSS property, e.g. `color: $color`, `red` will be used.
- Another feature is nesting, which is a more concise way to style elements which are related to other elements in a certain way.

```
  div {       font-size: 18px;       p {           color: blue;
}       ul {           color: green;       }   }
```

  - In this example, all `p`s inside `div`s will be have `color: blue`, but also `font-size: 18px`, while `ul`s inside `div`s will have `color: green` instead, but still also `font-size: 18px`.
- One more useful feature is inheritance, which is similar to the object-oriented concept. Sass's inheritance allows for slight tweaking of a general style for different components.

```
  %message {       font-family: sans-serif;       font-size: 18px;
font-weight: bold;   }    .specificMessage {       @extend %message;
background-color: green;   }
```

  - `%message` defines a general pattern that can be inherited in other style definitions using the `@extend %message` syntax. In addition, other style properties can be added.