GIT

- *Directory*: A folder used for storing multiple files.
- *Repository*: A directory where Git has been initialized to start version controlling your files.
- **`git init`** : tell Git you want to use it in this folder (creates a repository)
- **`git status`** : see what the current state of our project is
  - Tip: It's healthy to run git status often. Sometimes things change and you don't notice it.
- *Staging Area*: A place where we can group files together before we "commit" them to Git.
- *Commit:* A "commit" is a snapshot of our repository. This way if we ever need to look back at the changes we've made (or if someone else does), we will see a nice timeline of all changes.
- *staged*: Files are ready to be committed
- *unstaged*: Files with changes that have not been prepared to be committed
- *untracked*: Files aren't tracked by Git yet. This usually indicates a newly created file
- *deleted*: File has been deleted and is waiting to be removed from Git
- **`git add [filename]`** : Tell Git to stage this file
  - **`git add -a .`** : add all - where the dot stands for the current directory, so everything in and beneath it is added. The -a stands for all and ensures even file deletions are included
- **`git reset [filename]`** : removes a file from the staging area
- **`git log`** : Git's journal - remembers the changes we've committed in the order we committed them
  - Tip: use `git log --summary` to see more information for each commit. You can see where new files were added for the first time or where files were deleted. It's a good overview of the project.
- **`git remote add [alias] [url] :`** add a new remote repository of your project
- *Remote repositories:* versions of your project that are hosted on the Internet or network somewhere
- **`git push [to] [from]`** : tells Git where to put our commits
  - The -u before [to] tells Git to remember the parameters, so that next time we can simply run git push and Git will know what to do. Go ahead and push it
- **`git pull [from] [to]`** : check for changes on our GitHub repository and pull down new changes
- **`git diff [change]`**
  - HEAD by default holds most recent commit, so git diff HEAD shows difference from last commit
  - git diff --staged lets you see the changes you just staged
  - **`git reset [file]`** : unstages the file
- **`git checkout -- [file] :`** get rid of all the changes since the last commit for the file
- **`git clone [url]`** : copy a git repository so you can add to it

HTML

<head> at the top of your html document, contains information about the page </head>

<title> window name </title>

<body> the content of your page </body>

<h1>Big Heading</h1> …. **<h3>**Smaller headering</h3> **<p>**A paragraph</p>

<a href=”http://www.google.ca”>This is a link to Google</a>

<img src=”http://imageshare.com/link_to_your_picture”/>

<img src=”../file_path/to_picture/on_your/computer”/>

```
unordered list                  ordered list
<ul>                            <ol>
       <li>Item1</li>                 <li>Item1</li>
       <li>Item2</li>                 <li>Item2</li>
</ul>                           </ol>
```

put things in a container together:

<div></div> for block sections (new line before and after)

<span></span> for inline (no line break before or after)

CSS

to link to html file, put this in the <head> tag: **<link type="text/css" rel="stylesheet" href="stylesheet.css"/>**

element_name {	e.g. element_name is body
property: value;	e.g. property is **background-color**, value is blue
another_property: another_value;	e.g. property **font-size**, value 18px;
}	e.g. property **font-family**, value serif, sans-serif, cursive

- put a semicolon (;) at the end of each line -- tells CSS that one property-value pair is over and it's time to move on to the next one
- all property-value pairs for a selector are surrounded by curly braces ({})

IDs are great for when you have exactly one element that should receive a certain kind of styling

- `<div id="id_name"> STUFF STUFF </div>`
- in the stylesheet: `#id_name { … }`

Classes are useful when you have a bunch of elements that should all receive the same styling

- `<div class="class_name"> STUFF STUFF </div>`
- `<div class="class_name"> OTHER STUFF </div>`
- in the stylesheet: `.class_name { … }`

Example: make the navbar links change colour on hover

```
#navbar a:hover {
    color: #000099; /* blue */
}
```

we're styling <a> tags inside #navbar (we can make lists of selectors like this) and specifically styling *when the mouse hovers*

RESOURCES

Slides from today:
https://docs.google.com/presentation/d/1rypiu76yaIuUvIjIINWSNIkC-_5sHxPXfIoUOZkxmfQ/edit?usp=sharing

Git Resources

- Tutorial we used (we didn't finish it) https://try.github.io/levels/1/challenges/1
- Reference (reminds you what things do): http://gitref.org/
- OpenHatch mission (practice using git in a short challenge): http://openhatch.org/missions/git
- Chapters 1-3 of this awesome book of tells you most of what you need to know: http://git-scm.com/book/en/v2

HTML/CSS resources

- CSS reference https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Getting_started

Learn more about web development (and other tools that you use with HTML/CSS)

- https://www.udacity.com/course/cs253 Web Development Course - How to Build a Blog