MAKE SCHOOL

# HTML Forms + Version Control

WEB 1.0

# Agenda

- Learning Outcomes
- Your File System
- GitHub
- **BREAK**
- Forms & Input
- Forms Activity
- Lab Time

# Learning Outcomes

By the end of today, you should be able to...

1. **Describe** the structure of your computer's file system.
2. **Use** Git + GitHub to clone a repo, create a new repo, and push changes.
3. **Identify & use** the most common input tags to construct HTML forms.

# Warm-Up: Code Reviews (10 minutes)

Break out into pairs and choose who will be the **reviewer** and **reviewee** for your finished portfolio assignment.

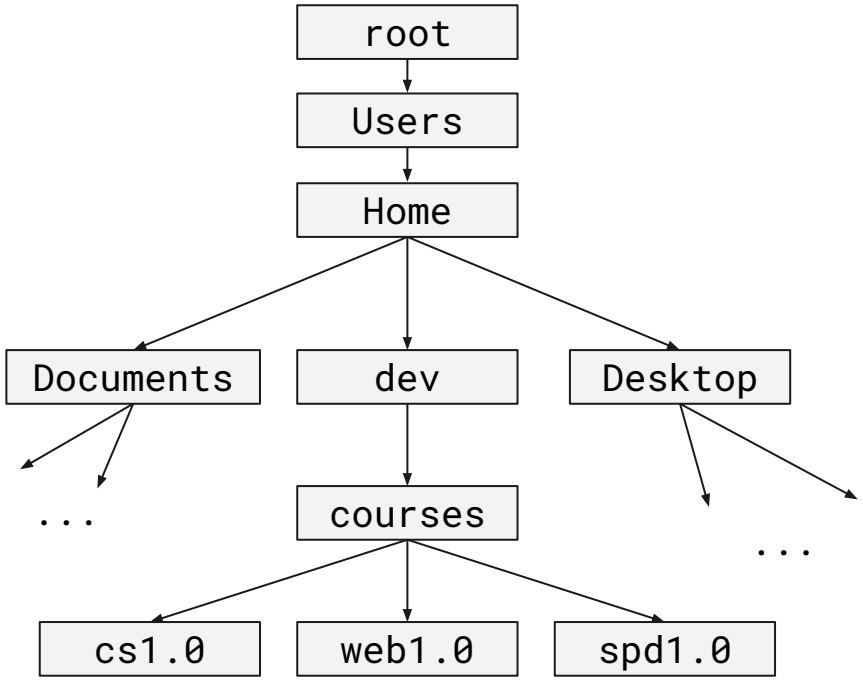
- **Reviewee**: Share your screen and explain what your code does from top to bottom.
- **Reviewer**: Listen, ask questions, and make suggestions for improvement. Pay attention to syntax & semantic usage of tags.

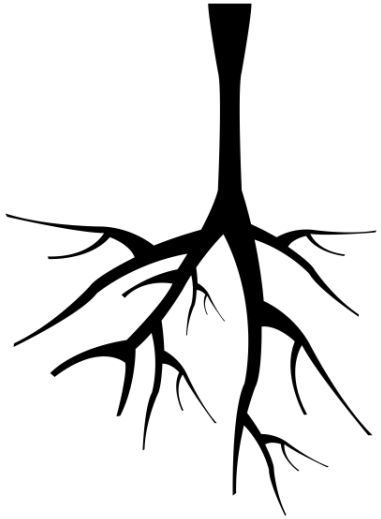
After 5 minutes, switch roles.

# Your File System

# File System Structure

The file system's structure is shaped like an upside-down tree. We sometimes call it the "file tree".



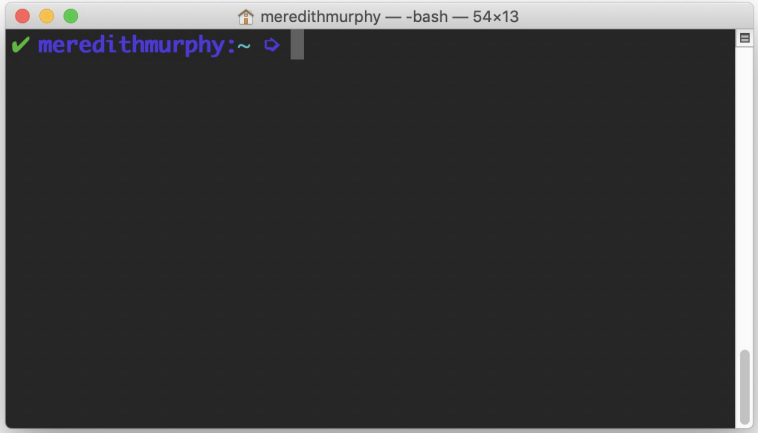



# What is Terminal?

Terminal is a program for your Mac that allows you to:

- Navigate your file system
- Execute commands
- Run programs
- Install programs/packages

using text-based commands.



# File System Navigation

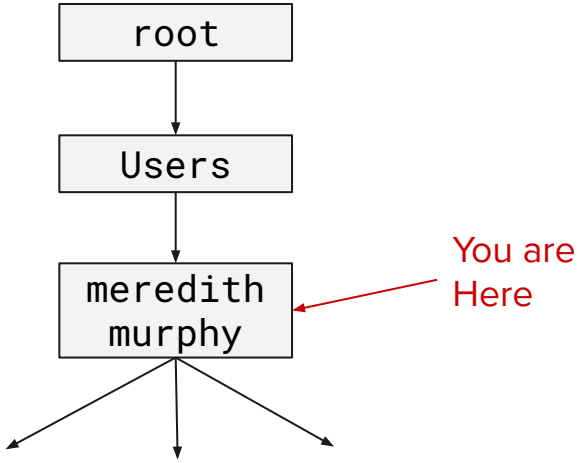
To navigate your file system using the Terminal, we'll use the following 3 commands:

- `pwd` - "Print Working Directory"
- `ls` - "List Files"
- `cd name_of_directory` - "Change Directory"

**Please remember** these 3 commands and practice their use!

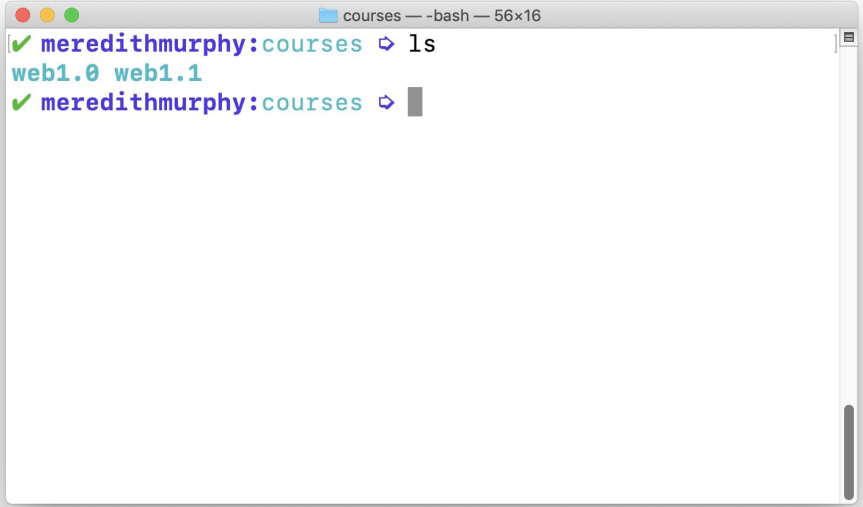
# File System Navigation

When you open your Terminal program, it will always start you off in the "**Home**" directory, which is named according to your username. We can use `pwd` to see this.

# File System Navigation

To see what files are inside the current file, we can use the command `ls`.



# File System Navigation

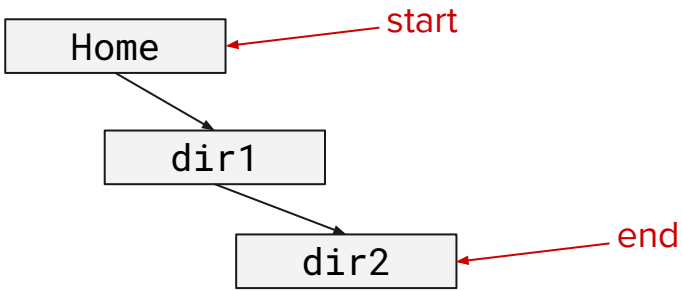
To navigate *up* or *down* in the tree, we can use **cd** followed by the name of the directory we want to move to.

```
web1.0 — -bash — 56×16
✔ meredithmurphy:courses ➪ pwd
/Users/meredithmurphy/dev/courses
✔ meredithmurphy:courses ➪ ls
web1.0 web1.1
✔ meredithmurphy:courses ➪ cd web1.0
✔ meredithmurphy:web1.0 ➪ pwd
/Users/meredithmurphy/dev/courses/web1.0
✔ meredithmurphy:web1.0 ➪
```

# File System Navigation

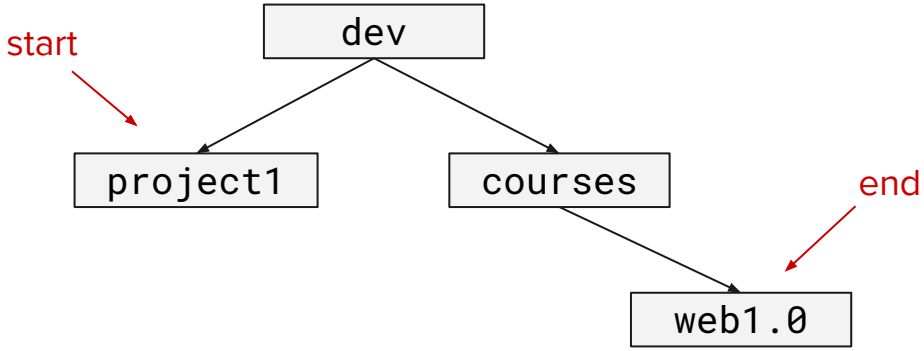
We can use **`cd`** in the following ways:

- **`cd`** (without a directory name) will take you back to your home directory
- **`cd ..`** will take you back *up* to the parent folder
- **`cd dir1/dir2`** will take you into *dir1*, then into *dir2*



# Check for Understanding

How can we go from `project1` to `web1.0` using only one command?




```
cd ../courses/web1.0
```

# Activity (5 minutes)

Open up a Terminal and navigate to your class folder!

# Git & GitHub

# What is Git?

**Git** is a **versioning tool** that allows us to track our code changes over time.

This allows us to:

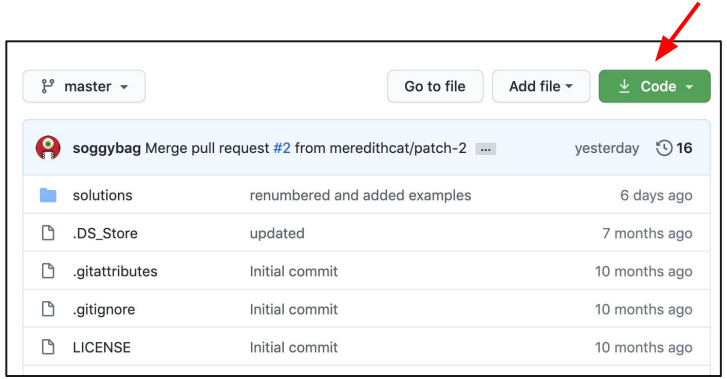
- **Roll back** to a previous change if things break
- See **which person** submitted which change
- **Merge** together conflicting changes.

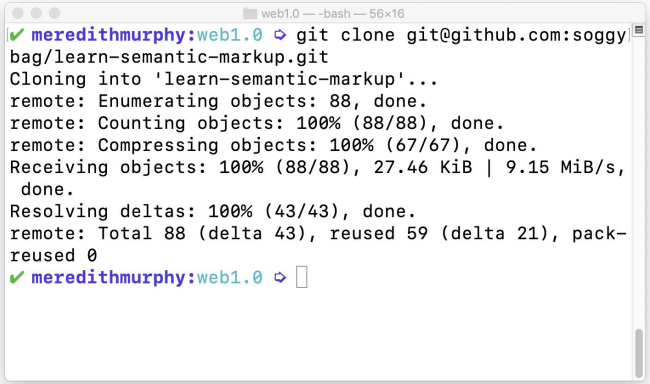
**GitHub** is a cloud-based storage system for your Git repositories.

# git clone

**Cloning** a repository means downloading a copy onto your local computer. You won't just get the files - you'll also get the ***version history*** of who changed what.

To clone a repository, go to its GitHub page and click the "Code" button to copy the URL. Then, navigate to the folder where you want to put it and type in `git clone <URL>`.



# git remote

A Git repository will usually have one or more "**remotes**". A **remote** is just a destination for where to "**push**" code changes. By default, a cloned repo will have a remote called **origin**.

You can see all of your repository's remotes with the command `git remote -v`.

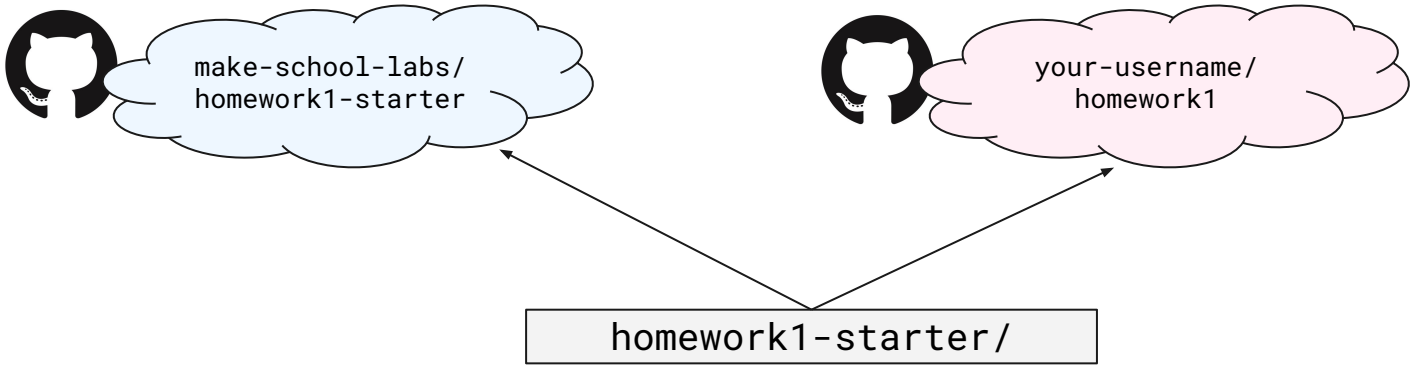
```
✔ meredithmurphy:learn-semantic-markup [ master {origin/
master} | ✔  ] ➪ git remote -v
origin  git@github.com:soggybag/learn-semantic-markup.gi
t (fetch)
origin  git@github.com:soggybag/learn-semantic-markup.gi
t (push)
✔ meredithmurphy:learn-semantic-markup [ master {origin/
master} | ✔  ] ➪
```

# git remote

If you clone someone else's code and want to upload your own copy to GitHub, you'll need to **change** the remote (or add a new one) and then **push** your changes. You can change "origin" to a new URL using the commands:

```
git remote set-url origin git@github.com:your-username/your-repo-name.git
```

```
git push -u origin master
```



# git commit & git push

If you make changes to your code and want them to be reflected in your GitHub repository, you'll need to make a **commit**:

`git add .` # Add all changed files to the commit

`git commit -m "Changed the index.html file"` # Make the commit

`git push origin master` # Push all commits to the remote "origin" and branch "master"

# git commit & git push

Making a commit is like using an online shopping cart.

`git add` is like adding something to your cart.

`git commit` is like checking out and making a transaction.

`git push` is like shipping the goods to your doorstep (in this case, we're "shipping" to GitHub.com).

`git status` lets you view the contents of your shopping cart (files you've changed but not committed).

# git remote / push

Whenever you create a new repository on GitHub, it'll remind you of all of these steps.

**Quick setup — if you've done this kind of thing before**

Set up in Desktop or HTTPS SSH `git@github.com:meredithcat/My-Repo-Name.git`

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

**...or create a new repository on the command line**

New repository

```
echo "# My-Repo-Name" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M master
git remote add origin git@github.com:meredithcat/My-Repo-Name.git
git push -u origin master
```

Cloned repository

**...or push an existing repository from the command line**

```
git remote add origin git@github.com:meredithcat/My-Repo-Name.git
git branch -M master
git push -u origin master
```

# Activity (10 minutes)

If you haven't yet, **create a GitHub repository** for your portfolio project and **push your changes**:

- Go to GitHub.com and create a new repository. (**NOTE**: Do **not** check the box that says "Initialize this repository with a README".)
- Follow the steps for "Create a new repository on the command line":
  - git init
  - git add .
  - git commit -m "First commit"
  - git remote add origin git@github.com:*your-username*/*your-repo-name*.git
  - git push -u origin master
- Refresh the page on your repository and verify that your changes appear!

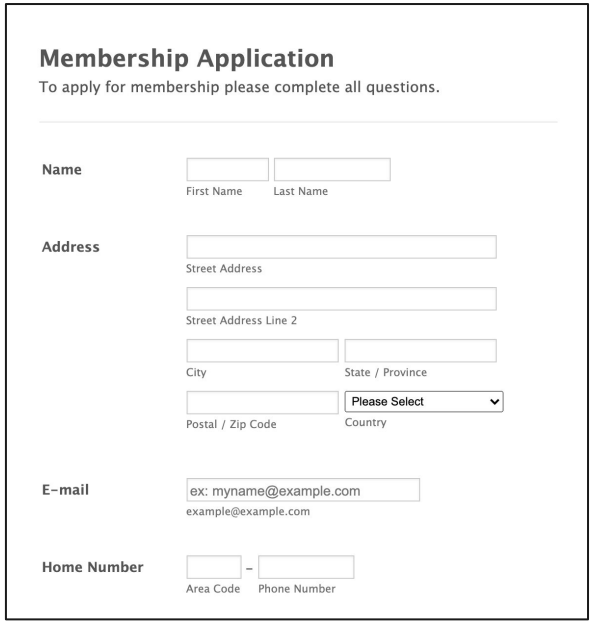
# Break - 10 min

*“Take a 10 minute break and wrap a tag around everything you see.”*

# Forms & Input

# Forms

Forms are used to **collect user information**. If you've ever signed up for an online account or ordered takeout online, you've used a form!



# Forms

To create a form in HTML, we use the `form` tag to surround all of the inputs.

Usually, a form will have at least one input element and a submit button.

```
<form>
   <p>
       <label>What is your name?
         <input type="text" name="firstname">
       </label>
   </p>
   <input type="submit" value="Submit!">
 </form>
```

What is your name?

Submit!

What it looks like

# Input

The **input tag** is used to create a form input. The attributes we need to specify are the **type** (what kind of data are we collecting?) and the **name** (what label are we giving that data?).

```
<input type="text" name="query">
```

We're collecting data of **type** text

We're giving this data the **name** of "query"

# Select

Some form inputs look different, such as the "select" and "option" elements, which form a drop-down.

However, note that they still have the "name" attribute. The "value" attribute is what is sent to the server.

```
<select name="fave-beverage">
   <option value="coffee">Coffee</option>
   <option value="coffee">Tea</option>
</select>
```

What is your favorite beverage?

Coffee

# Radio Buttons

Radio buttons allow you to choose one option among many. It also has a "value" attribute.

```
<input type="radio" name="fave-animal" value="cat">
```

Which is your favorite? Select only one.

- Dogs
- Cats

# Label

Each input needs a label element to go with it. Usually, we use the <label> tags to surround the input element.

```
<label>
   What is your name?
   <input type="text" name="firstname">
</label>
```

This means that, semantically, the text "What is your name?" goes along with the input element.

# Form, Fieldset, Legend

The "form" element is used to enclose all of the inputs & labels for one particular form.

The "fieldset" & "legend" elements are used to give a header to the form.

```
<form>
   <fieldset>
       <legend>Please enter your details:</legend>

       <!-- All form inputs go here! -->
   </fieldset>
</form>
```

Please enter your details:

Name:

Submit!

# Activity (10 minutes)

Open the Forms Practice Repl.It and explore the input elements used.

Then, add more input elements (can be same types) to enhance the form & collect more information from the user.

# Activity

Complete the Forms Practice Repl.It with a partner. If you don't finish or get stuck, take a look at the `solutions.html` file.

Remember to practice pair programming:

- Driver: Share your screen & listen to navigator
- Navigator: Tell driver what to do next

# Lab Time

# Homework

MAKE SCHOOL

**Portfolio & Learn Semantic Markup: Due tonight**

**Markup Level 2: Due Tuesday**

Stay in the main Zoom room if you'd like to stay for more Q&A, homework help, etc.

Go to your individual breakout room if you'd prefer to work with a partner or have quiet time!