

# **Computational Fundamentals: D3**

## **Week 2: D3 Foundations 1 (JS/Course Basics)**

**Nicole Cote, Spring 2024**

# What We'll Cover

- A conceptual intro to D3
- The tools we need for this course
- Submitting assignments
- JavaScript Intro/Resource

**What is D3 and why learn it?**

# Setting Up a D3 Project

# Process for Using D3 (for now)

1. Make a folder for each HW assignment (i.e. Week2-HW) in your class folder
2. Copy-in the provided index.html template
3. Using VS Code and Live Server (be sure to open the folder with VS Code):
  1. Double-check the template is referencing the current version of D3 (check <http://unpkg.com/d3> or the d3 website directly)
  2. Write fabulous code in the index.html to create a working vis per the assignment
4. Add a README.md to your folder (see Readme/Markdown/GitHub slides)
5. Include screenshot(s)/gif of your working vis (i.e. figure1.png)
6. Push your homework folder (with all files) to GitHub before the due date.

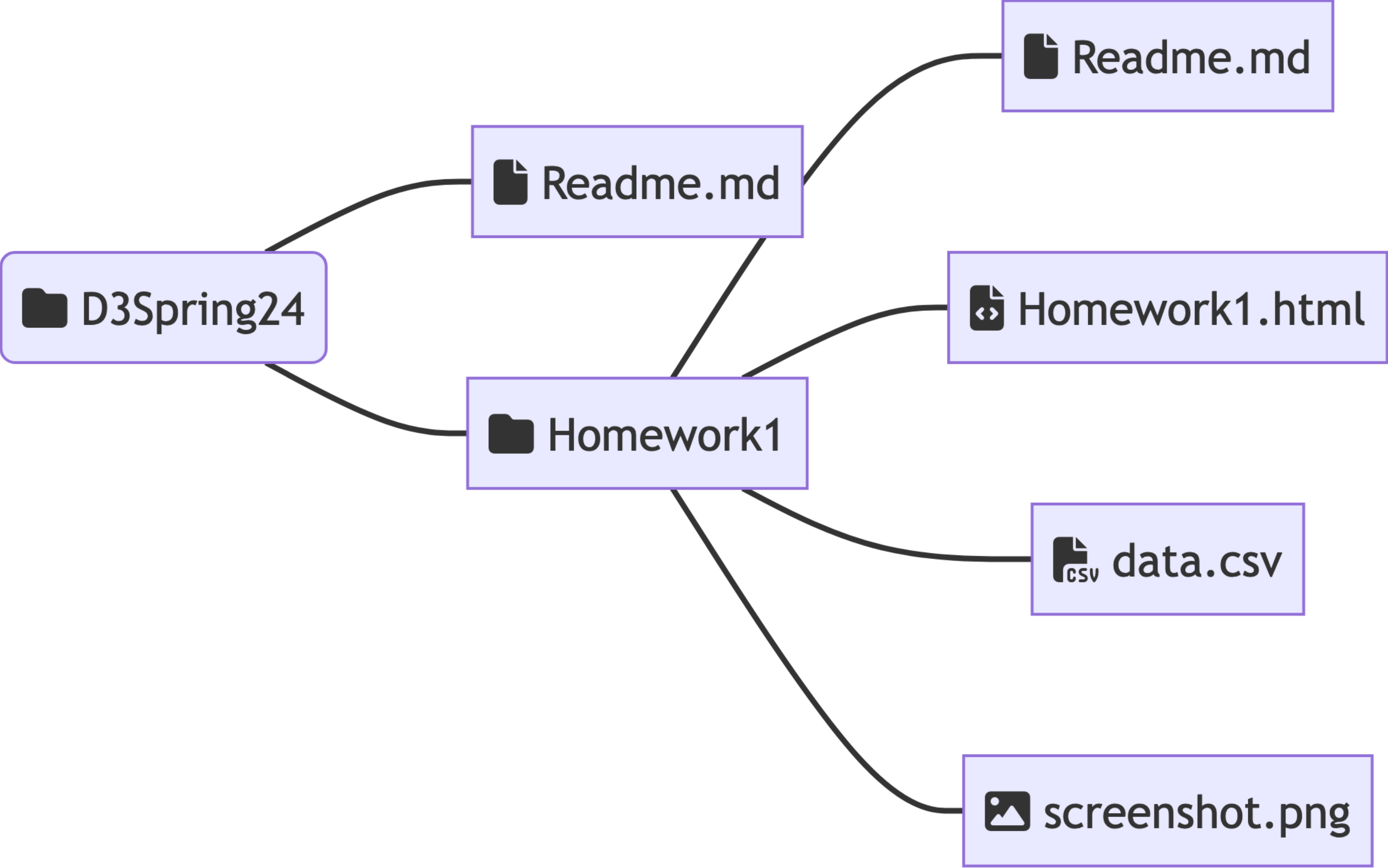
# Some Necessary Tools

# **VS Code & Live Server**

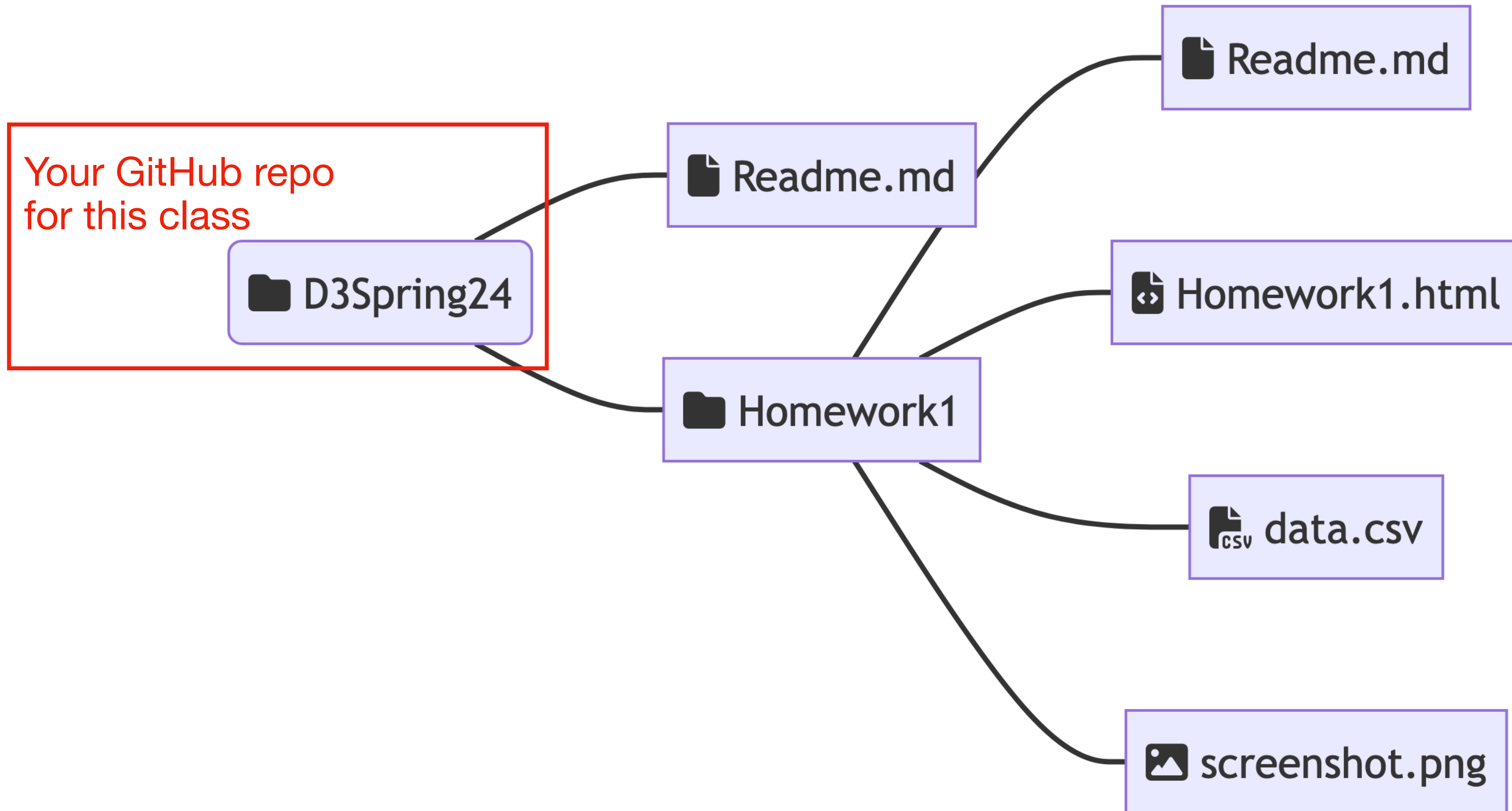
# GitHub



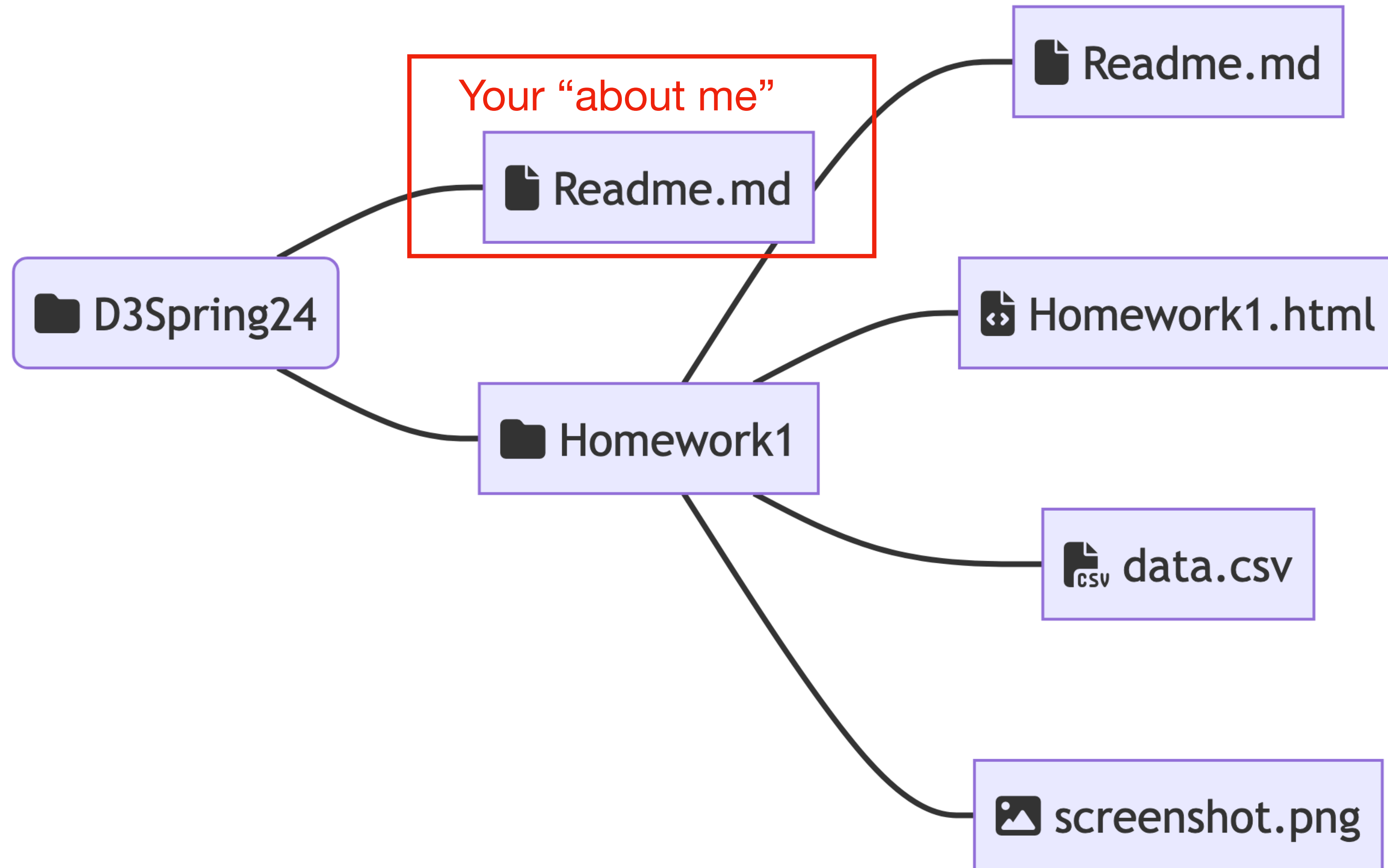
# GitHub Repo Structure- For HW Submission



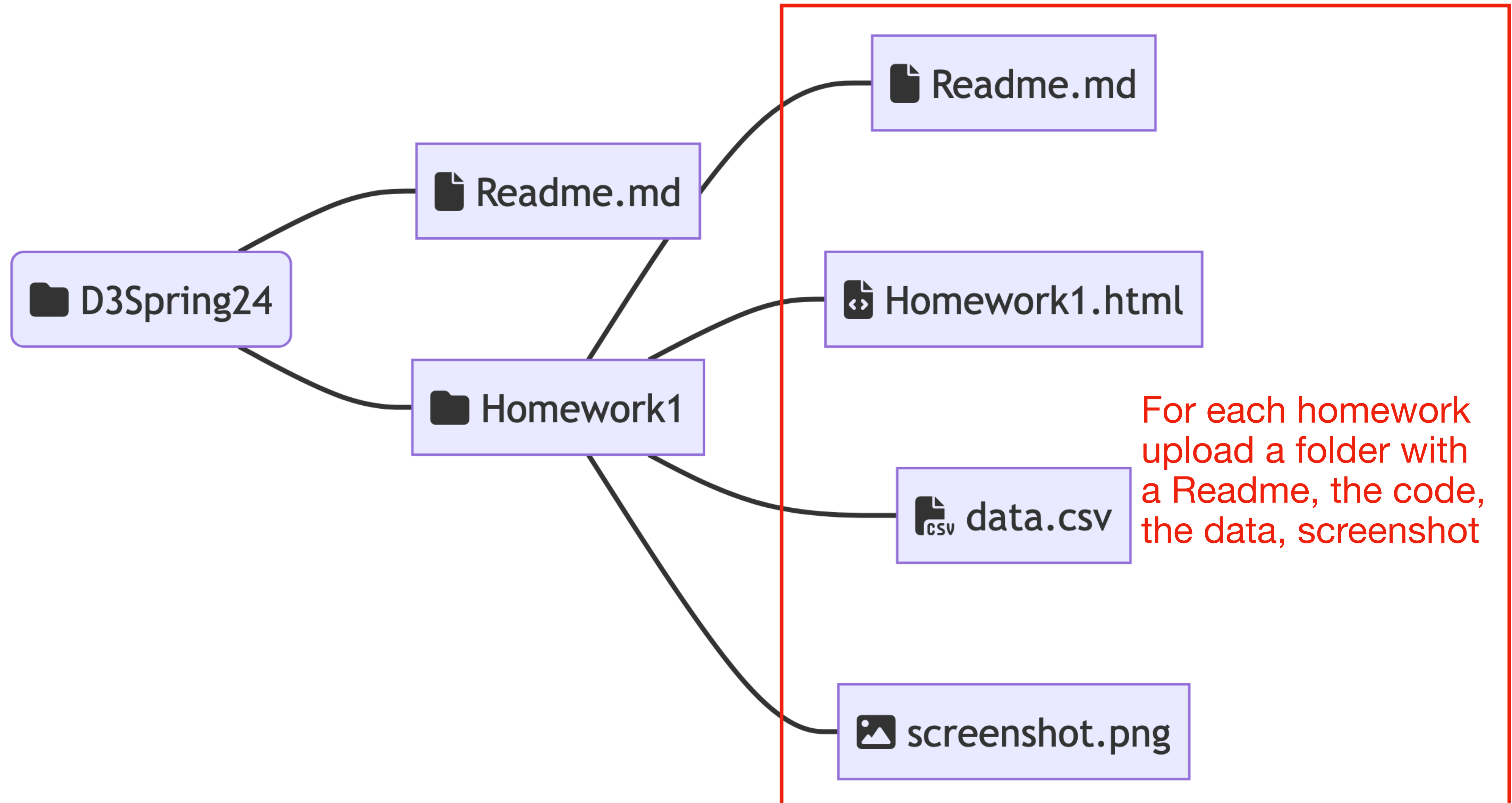
# GitHub Repo Structure - For HW Submission



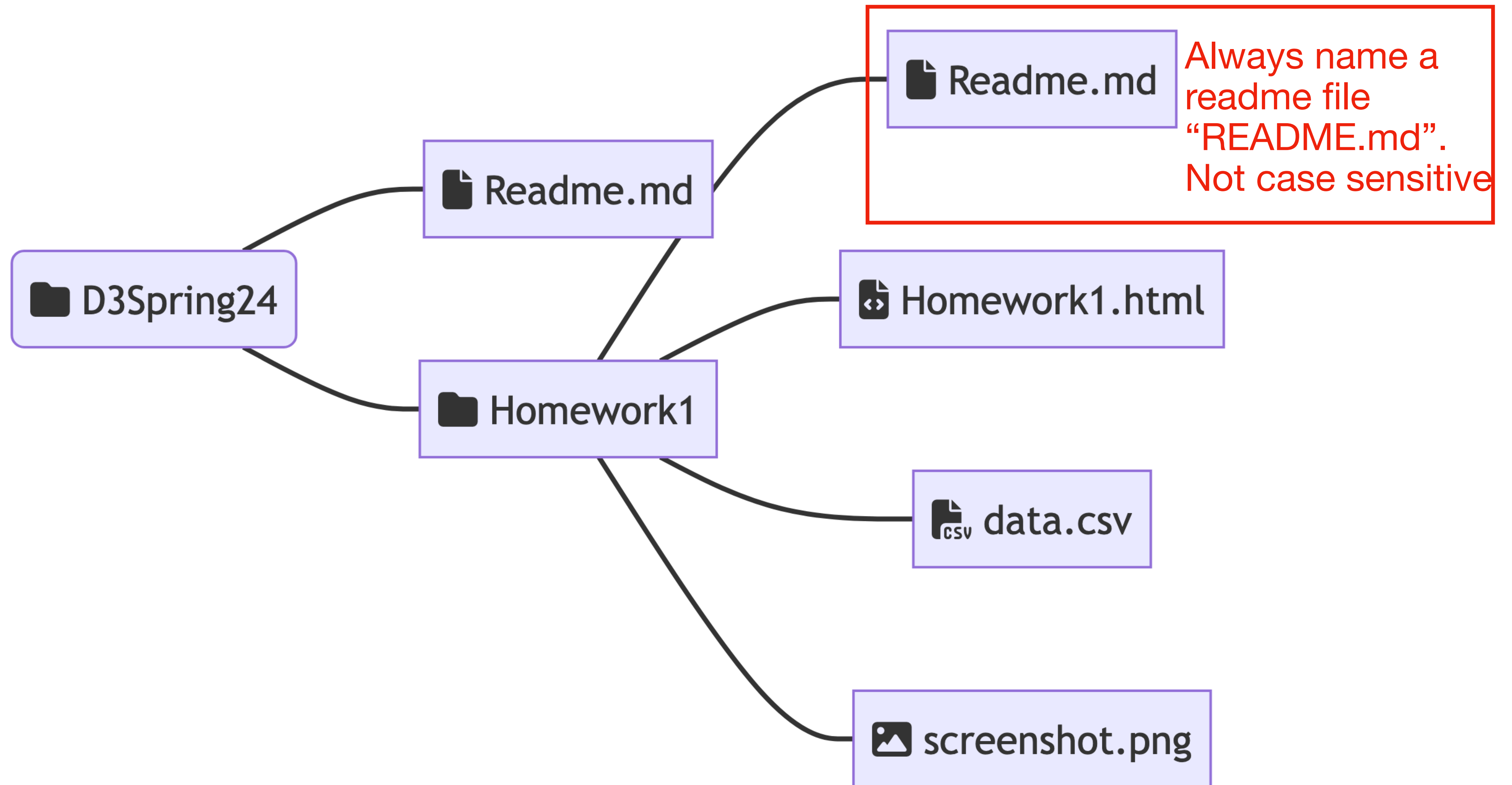
# GitHub Repo Structure - For HW Submission



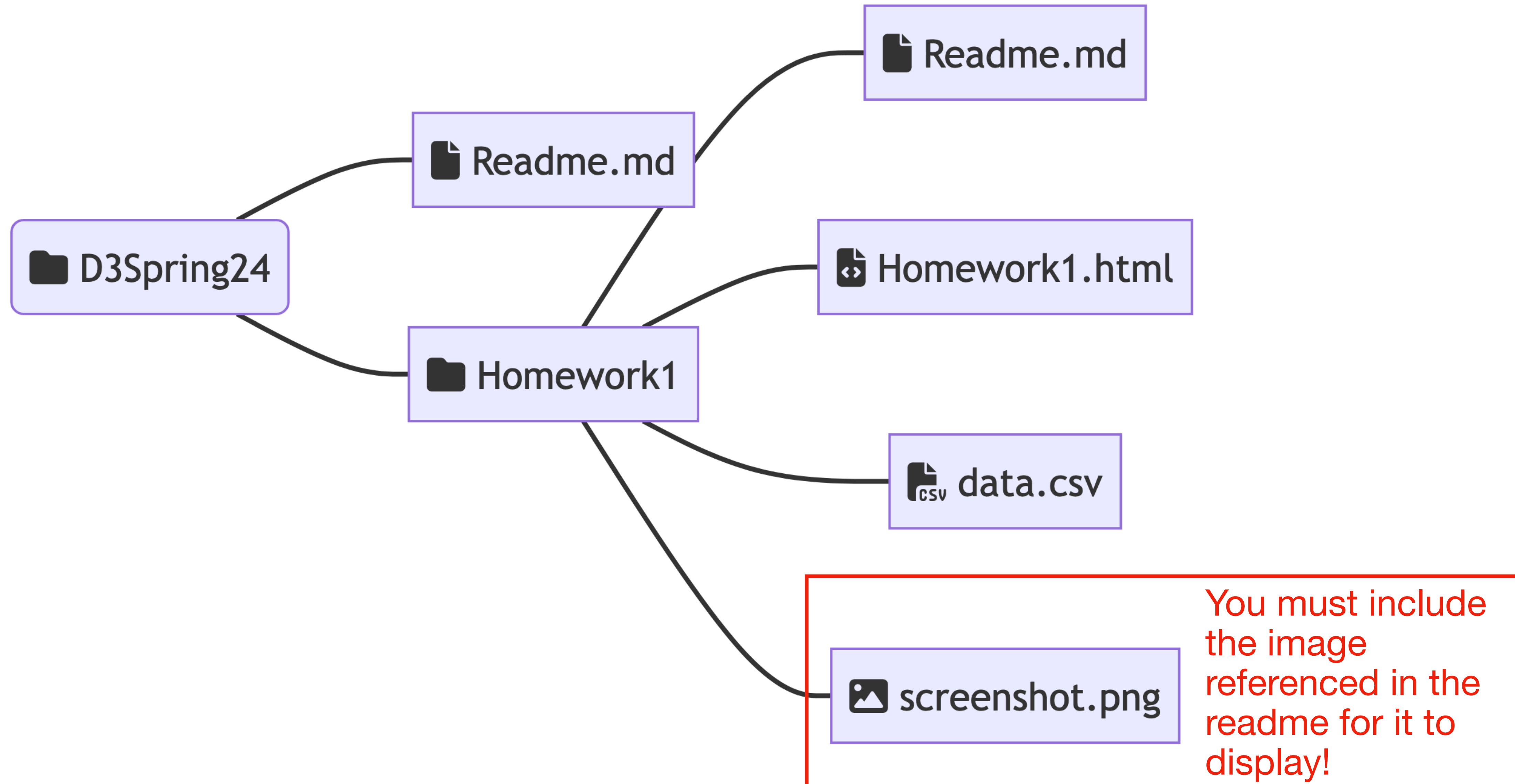
# GitHub Repo Structure - For HW Submission



# GitHub Repo Structure - For HW Submission



# GitHub Repo Structure - For HW Submission



**README.md**

# README Overview

README.md files will display as the cover page in each repo on GitHub. So they'll be used to provide an overview of the repo generally, and each HW. They are written with Markdown.

**The README file for each assignment must include:**

- a few sentences of explanation about your vis and the data (i.e. what it is, what it represents); this should also include where the data is sourced from
- any relevant citations
  - Note: reference the Academic Integrity section of the syllabus for more
- a screenshot of the working vis
- (when we get to interaction): additional screenshots of any interactive component



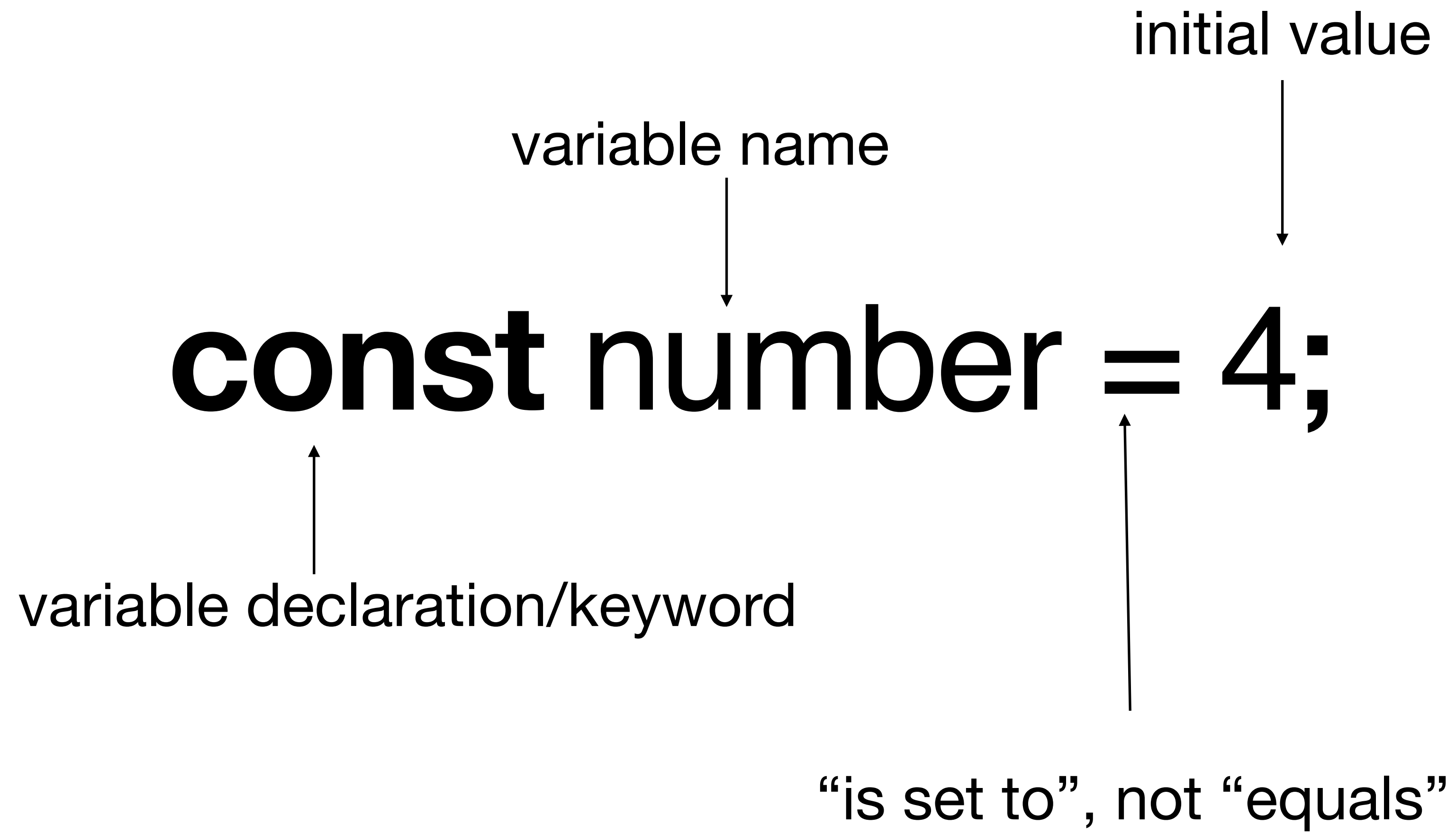
# Markdown Basics

# JS Intro/Reference

# What we'll cover

- Variables
- Data Types (basics): Arrays, Objects, Arrays & Objects, JSON
- Operators
- Conditionals

# Variables



# **const number -vs- const Number**

**\*\*JavaScript is case sensitive!\*\***

# variable types

- numbers (-2, -1.5, 0, 0.53, 10.75, 100)
- string (“text”, “Text”, “9”, “0.5”)
- Boolean (true, false)
- null

**const      let      var**



**const**

**let**

**~~var~~**

# Data Types

# Arrays

Instead of many values stored in single variables:

```
const numberA = 10;  
const numberB = 20;  
const numberC = 30;  
const numberD = 40;
```

You can make an array:

```
const numbers = [10, 20, 30, 40];
```

# Arrays Pt 2

Retrieving values from an array:

```
const numbers = [10, 20, 30, 40];
```

numbers[0]	returns 10
numbers[1]	returns 20
numbers[2]	returns 30
numbers[3]	returns 40

# Arrays Pt 3

array.length:

```
const numbers = [10, 20, 30, 40];
```

numbers.length returns 4

# Arrays Pt 4

## Some common methods

Method name	Description
<code>a.toString()</code>	Returns a string with the <code>toString()</code> of each element separated by commas.
<code>a.toLocaleString()</code>	Returns a string with the <code>toLocaleString()</code> of each element separated by commas.
<code>a.concat(item1[, item2[, ...[, itemN]]])</code>	Returns a new array with the items added on to it.
<code>a.join(sep)</code>	Converts the array to a string — with values delimited by the <code>sep</code> param
<code>a.pop()</code>	Removes and returns the last item.
<code>a.push(item1, ..., itemN)</code>	Appends items to the end of the array.
<code>a.shift()</code>	Removes and returns the first item.
<code>a.unshift(item1[, item2[, ...[, itemN]]])</code>	Prepends items to the start of the array.
<code>a.slice(start[, end])</code>	Returns a sub-array.
<code>a.sort([cmpfn])</code>	Takes an optional comparison function.
<code>a.splice(start, delcount[, item1[, ...[, itemN]]])</code>	Lets you modify an array by deleting a section and replacing it with more items.
<code>a.reverse()</code>	Reverses the array.

# Objects

For more complex data:

```
const dog = {  
  kind: "shih tzu",  
  color: "grey",  
  quantity: 1,  
  friendly: true  
};
```

Retrieving values from an object:

dog.kind	returns "shih tzu"
dog.quantity	returns 1
dog.friendly	returns true

# Arrays and Objects

Combining arrays and objects:

```
const dogs = [  
  {kind: "shih tzu",  
    color: "grey",  
    quantity: 1,  
    friendly: true  
  },  
  {kind: "lab",  
    color: "black",  
    quantity: 1,  
    friendly: true  
  },  
  {kind: "poodle",  
    color: "white",  
    quantity: 1,  
    friendly: true  
  }  
];
```

Retrieving values:

dogs[2].kind returns poodle

\*remember: [ ] is array and { } is object



# JSON

Similar to Object Literal, JSON relies on properties (keys and values), but be sure to note the style difference:

```
{“kind”: “shih tzu”,  
“color”: “grey”,  
“quantity”: 1,  
“friendly”: true}
```

JSON can be stored in variables:

```
const dog = {  
“kind”: “shih tzu”,  
“color”: “grey”,  
“quantity”: 1,  
“friendly”: true  
};
```

# Operators

# Mathematical Operators

+ add

- subtract

\* multiply

/ divide

% remainder

# Comparison Operators

`==` equal to

`===` strict equal to (use this one!)

`!=` not equal to

`!==` strict not equal to (use this one!)

`<` less than

`>` greater than

`<=` less than or equal to

`>=` greater than or equal to

# Logical Operators

&& AND

|| OR

! NOT

# Conditionals

# If Statements

## if

```
if (taskIsComplete) {  
  doSomething( );  
}
```

## if/else

```
if (taskIsComplete) {  
  doSomething( );  
}  
else {  
  doSomethingElse( );  
}
```

## if/else if/else

```
if (taskIsComplete) {  
  doSomething( );  
}  
else if (taskIsComplete) {  
  doSomethingElse( );  
}  
else {  
  doAnotherThing( );  
}
```

# Homework



# Assignments

## If Needed

- Download Visual Studio (VS) Code (<https://code.visualstudio.com/>)
- Download the Live Server extension in VS Code (or be able to use a Node server already; Live Server is recommended)
- Download “Learn Markdown” and “Learn Markdown Preview” Extensions in VS Code (optional, but helpful to preview Readme files)
- To use Git locally: download Git (<https://git-scm.com/downloads>); And, if Git is unfamiliar to you watch this tutorial by Derek Banas (<https://bit.ly/3gc8ikl>) (and see Additional Resources slides)
- Create a GitHub account (see Additional Resources slides for help)

## For Everyone

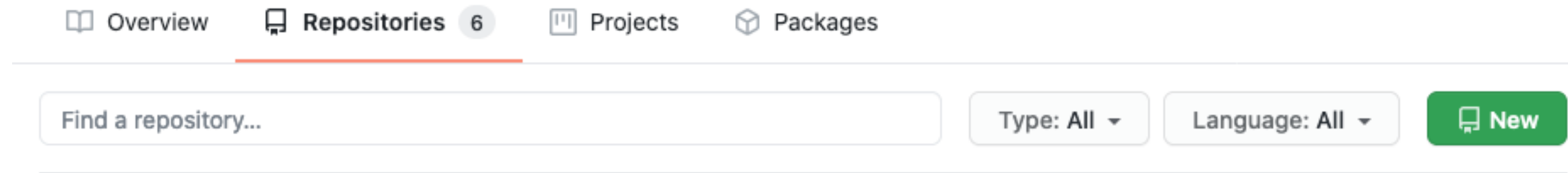
- Create a public GitHub repo for our course and email me the link (called, for example: D3Spring24)
- Add an introduction/“about me” to your GitHub repo as a README.md by the deadline.
  - Note: this must include a representative image of some kind.
  - For Markdown help: see Markdown resources included in today’s lesson
  - Your image can be a stock image. For free stock images one site to use is Unsplash (<https://unsplash.com/>). Ensure you include any citations.
- Check that you can open and edit code files with VS Code and Live Server:
  - 1) open the “wk1code” folder in VS Code. 2) click the “hello-world-d3.html” file and click “Go Live” to run it. It should display in the Browser. 3) in the code file edit the text color for example change “red” to “black”. 4) Save it and run it again—the font should have changed color.
- Optional, for those who need more, read/skim the excellent “Feminist Data Manifest-No” by Cifor et al. (<https://www.manifestno.com/>)

**“Data can be a check-in, a story, an experience or set of experiences, and a resource to begin and continue dialogue. It can - and should always - resist reduction. Data is a thing, a process, and a relationship we make and put to use. *We can make it and use it differently.*”**

# Appendix: Additional Resources

# GitHub Resources

# Making a New Repo in GitHub



- In GitHub, navigate to Repositories (on the top bar)
- Click New

# Making a New Repo in GitHub Pt 2

- Fill in the details, and click Create Repository

## Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner \*

Repository name \*

 /  ✓

Great repository names are short and memorable. Need inspiration? How about **musical-octo-pancake**?

Description (optional)

This is a repo for Programming Interactive Visualizations Fall 2020



**Public**

Anyone on the internet can see this repository. You choose who can commit.



**Private**

You choose who can see and commit to this repository.

Skip this step if you're importing an existing repository.

☐ **Initialize this repository with a README**

This will let you immediately clone the repository to your computer.

Add .gitignore: None ▼

Add a license: None ▼




Create repository


# Making a New Repo in GitHub Pt 3 (Command Line)

- Follow instructions on the following page:

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

 Set up in Desktop or 

HTTPSSSH




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**

```
echo "# test" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin 

your repo info


git push -u origin main
```




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

```
git remote add origin 

your repo info


git branch -M main
git push -u origin main
```



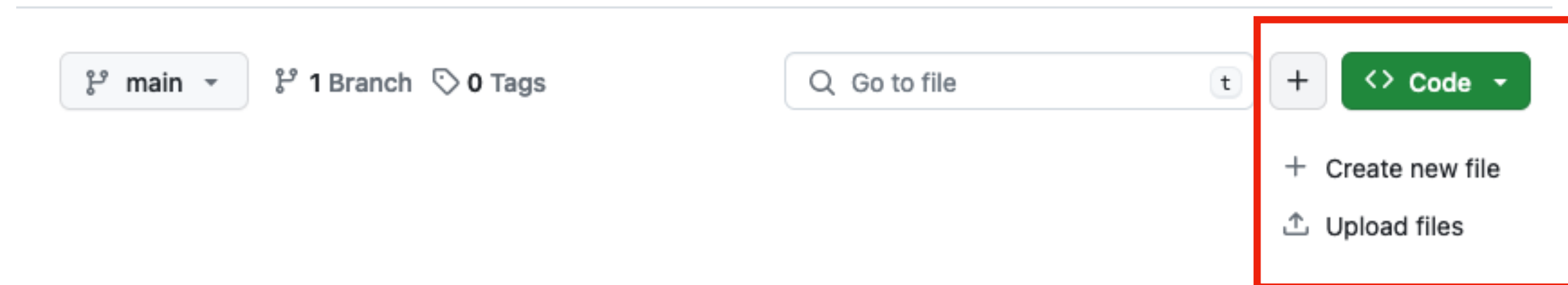
**...or import code from another repository**

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

Import code

# Adding Files to GitHub (without Command Line)

- In your repo click the +



To add your weekly folders and/or individuals files to your GitHub repo without the command line use the “upload files” feature

- Write a commit message <50 characters
- Commit changes to main branch



# Basic Git/GitHub Process (Command Line)

1. code something
2. open the Terminal
3. `cd` to your directory
4. `git status`
5. `git add --all`
6. `git commit -m 'write message'`
7. `git push origin main`

# Common Terminal and Git Commands

- `cd` (change directory)
- `clear` (clear what's visible in Terminal)
- `ls` (list contents of folder)
- `git init` (creates git directory)
- `git add` ("stages" your changes)
- `git commit -m 'Type Message'` (commits your staged changes with a message <50 characters)
- `git status` (shows current status of your repo)
- `git help` (shows the help feature)
- `git diff` (shows changes)
- `git push origin main` (pushes local changes to the main branch of your GitHub repo)
- `pwd` (print working directory, i.e. the folder you are in currently)

# Git Resources

# Git

- Derek Banas “Git Tutorial” video (<https://bit.ly/3gc8ikl>) and Cheat Sheet (<https://bit.ly/31dNyCD>)
- Mini-Videos provided by Git (<https://git-scm.com/videos>)
- Git Cheat Sheet by GitHub (<https://github.github.com/training-kit/downloads/github-git-cheat-sheet.pdf>)

# Markdown Resources

# Markdown

- Markdown Guide <https://www.markdownguide.org/>
- Markdown Guide “Cheat Sheet” <https://www.markdownguide.org/cheat-sheet/>