Kaitlin Hoffmann

Office Hours:

SH 243 Monday 1:15 - 3:15 PM. Tuesday 2:45 - 4:45 PM.

Email: hoffmank4@newpaltz.edu

GIT AND GITHUB

WDP

OBJECTIVES

- Git
- GitHub

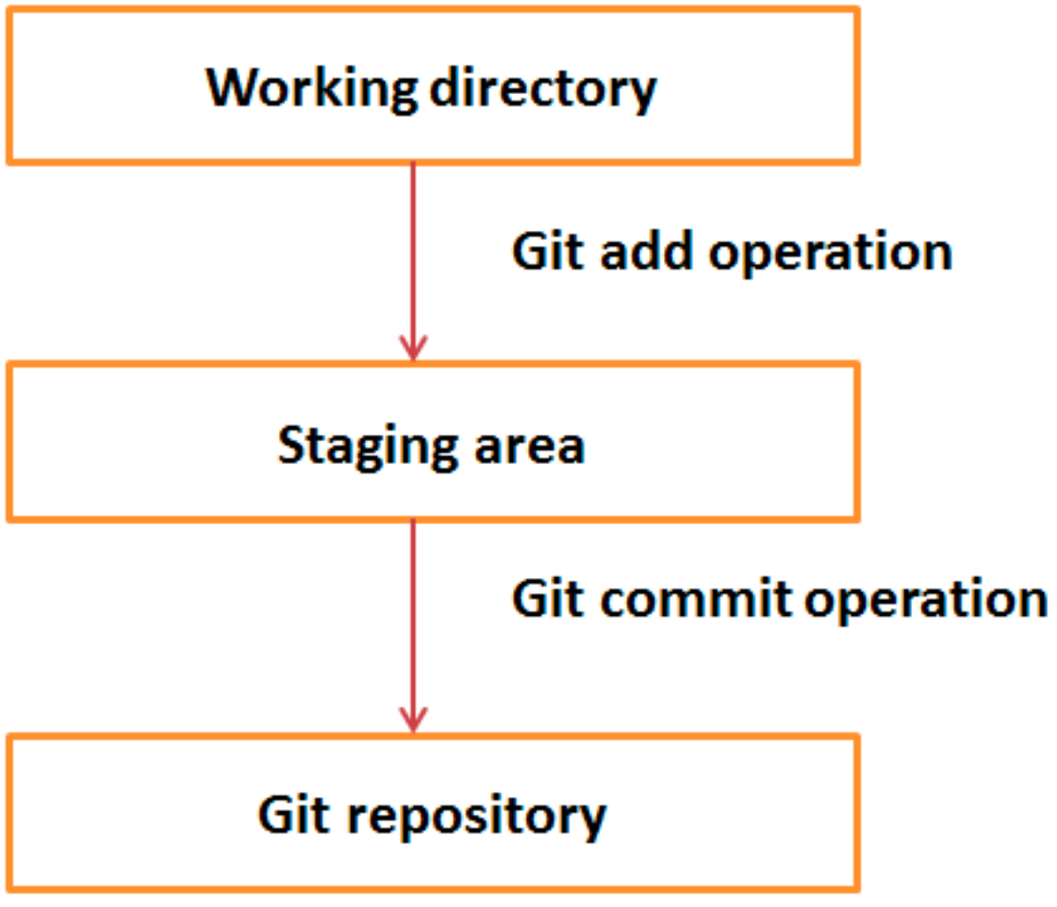
WHAT IS GIT?

- 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. You can also make a copy of your file, make changes to that copy, and then merge these changes to the original copy.

BASIC WORKFLOW

Let us see the basic workflow of Git:

- **Step 1** – You modify a file from the working directory (your working directory could be your project for example).
- **Step 2** – You add these files to the staging area.
- **Step 3** – You perform commit operation that moves the files from the staging area. After push operation, it stores the changes permanently to the Git repository.



DOWNLOAD GIT TO YOUR COMPUTER

- In order to use Git on your computer, you'll have to download it. To verify you downloaded Git, enter the following inside your terminal:

  git – –version

- If a version pops up, you successfully downloaded Git. If you have issues, let me know.

Link to download Git (Follow the steps for the computer you have – Windows, Mac, Linux):

https://git-scm.com/book/en/v2/Getting-Started-Installing-Git

To get to your **terminal** (Mac/Linux) or **command line** (Windows), type in **terminal** or **command line** in your search bar on your computer and click it open.

HOW TO CONFIGURE GIT

- To set your username, type and execute these commands inside your Command Prompt/Terminal (command prompt is Windows, Terminal is Mac/Linux):

  git config --global user.name "YOUR_USERNAME"

  git config --global user.email "YOUR_EMAIL"

- Be sure to replace “YOUR_USERNAME” and “YOUR_EMAIL" with the values you choose.

- You can find your command prompt/terminal by using your search bar. See the next slide to simply use the terminal in VSCode.

```
[kaitlinhoffmann ~ $ git config --global user.name kaitlinchoffmann
[kaitlinhoffmann ~ $ git config --global user.email kaitlinchoffmann@gmail.com
 kaitlinhoffmann ~ $
```

CREATE AND INITIALIZE A PROJECT IN GIT

- Go into the folder you want to initialize using your terminal (in your case, it will be your Project folder). You can do this in Visual Studio Code.
  - To open up your terminal in VSCode, press **Ctrl +** `
- Then type **git init** inside your terminal. Press enter.

STAGING FILES

- When you first initialize a git repo, it's empty. Let's say you want to add files, such as your login, register and style.css pages. We need to first add files to a **staging environment.**
- **Staged** files are files that are ready to be committed to the repository you are working on.
- You can add individual files by using the command: ***git add fileName***

```
kaitlinhoffmann (master #) Random Project $ git add index.html
kaitlinhoffmann (master +) Random Project $
```

- OR you can add all your files using the command: ***git add .***

```
kaitlinhoffmann (master +) Random Project $ git add .
kaitlinhoffmann (master +) Random Project $
```

STAGING FILES

- To check on the **status** of your git repo, use the command ***git status***

```
kaitlinhoffmann (master +) Random Project $ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   index.html
        new file:   style.css

kaitlinhoffmann (master +) Random Project $
```

- All of my files are **staged**. We still need to **commit** our changes! Let's do so now...

COMMITTING FILES

- Adding commits keep track of your progress and changes as you work.
- Git considers each commit change point or "**save point**"– It is a point in the project you can go back to if you find a bug, or want to make a change.
- Always include a message when committing. This will make it easy for you and others to see the changes made and/or components added.
- Command to commit: **git commit -m *"your message"***

```
kaitlinhoffmann (master +) Random Project $ git commit -m "Add index and style files"
[master (root-commit) 9b684c0] Add index and style files
 2 files changed, 16 insertions(+)
 create mode 100644 index.html
 create mode 100644 style.css
kaitlinhoffmann (master) Random Project $
```

GIT COMMIT WITHOUT STAGE

- If you make a small change, you can skip the staging step entirely and commit after making changes to your files. Just add **-a** in your command.

- Let's say I added some text to my index.html file. Instead of staging then committing, I will use the following command to do both in one line:

git commit -a -m *"your message*"

```
kaitlinhoffmann (master) Random Project $ git commit -a -m "Add text to index"
[master 2213a53] Add text to index
 1 file changed, 3 insertions(+)
kaitlinhoffmann (master) Random Project $
```

COMMITTING FILES

- Let’s check the status now…nothing to commit, worked!

```
kaitlinhoffmann (master) Random Project $ git status
On branch master
nothing to commit, working tree clean
kaitlinhoffmann (master) Random Project $
```

SEEING CHANGES

- To see a log of all of your commits, use **git log**

```
kaitlinhoffmann (master) Random Project $ git log
commit 0796e751adb82649f0a86cebcff6a48172977f24 (HEAD -> master)
Author: kaitlinchoffmann <kaitlinchoffmann@gmail.com>
Date:   Sun Feb 27 11:54:38 2022 -0500

    Add content to index, change title

commit 2213a5392e3ea295021a8262489db91e86f761d9
Author: kaitlinchoffmann <kaitlinchoffmann@gmail.com>
Date:   Sun Feb 27 11:43:52 2022 -0500

    Add text to index

commit d7475b54f0c9cf8ca3108927992711 6b38770828
Author: kaitlinchoffmann <kaitlinchoffmann@gmail.com>
Date:   Sun Feb 27 11:37:34 2022 -0500

    Change h1 color
```

SEEING ADDITIONS AND DELETIONS IN A COMMIT

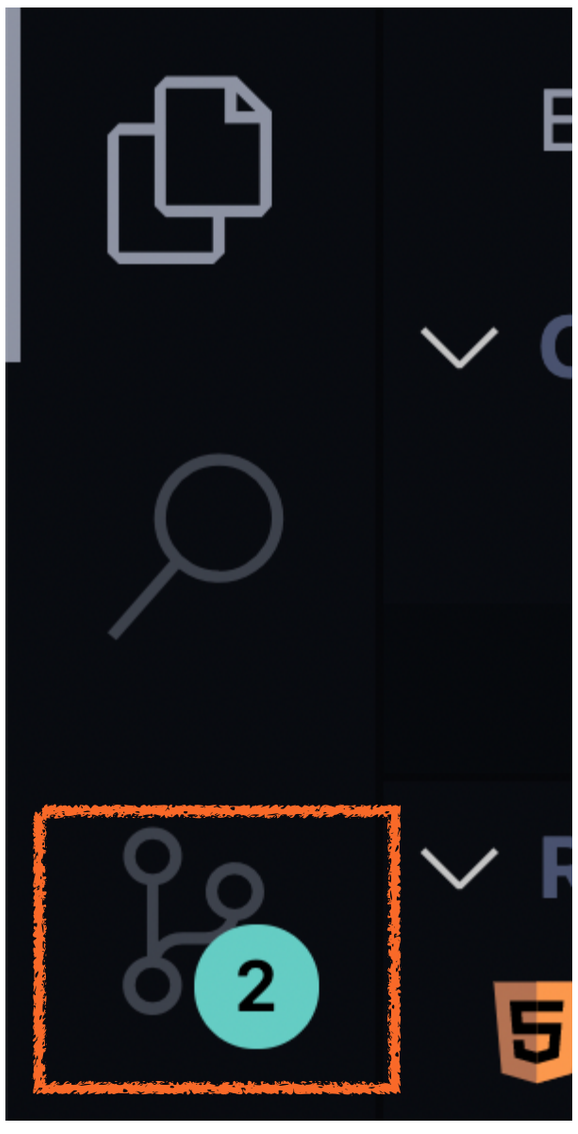
- To see the actual changes (additions and deletions) from a particular commit, grab the commit id from the git log, then use: **git diff *commitID***

```
kaitlinhoffmann (master) Random Project $ git diff 9b684c0002b22be7
df5d4
diff --git a/index.html b/index.html
index bd85870..fd1d8f8 100644
--- a/index.html
+++ b/index.html
@@ -8,6 +8,15 @@
    <title>Home</title>
  </head>
  <body>
-   <h1>Hi.</h1>
+   <h2>My New Title</h2>
```

If something breaks in your program, you can easily see what changes you previously made that could be causing you issues.

- – sign and **red** indicates what was **removed** by you
- + sign and green-yellow indicates what was **added** by you

VISUAL STUDIO CODE MAKES VERSION CONTROL EASY

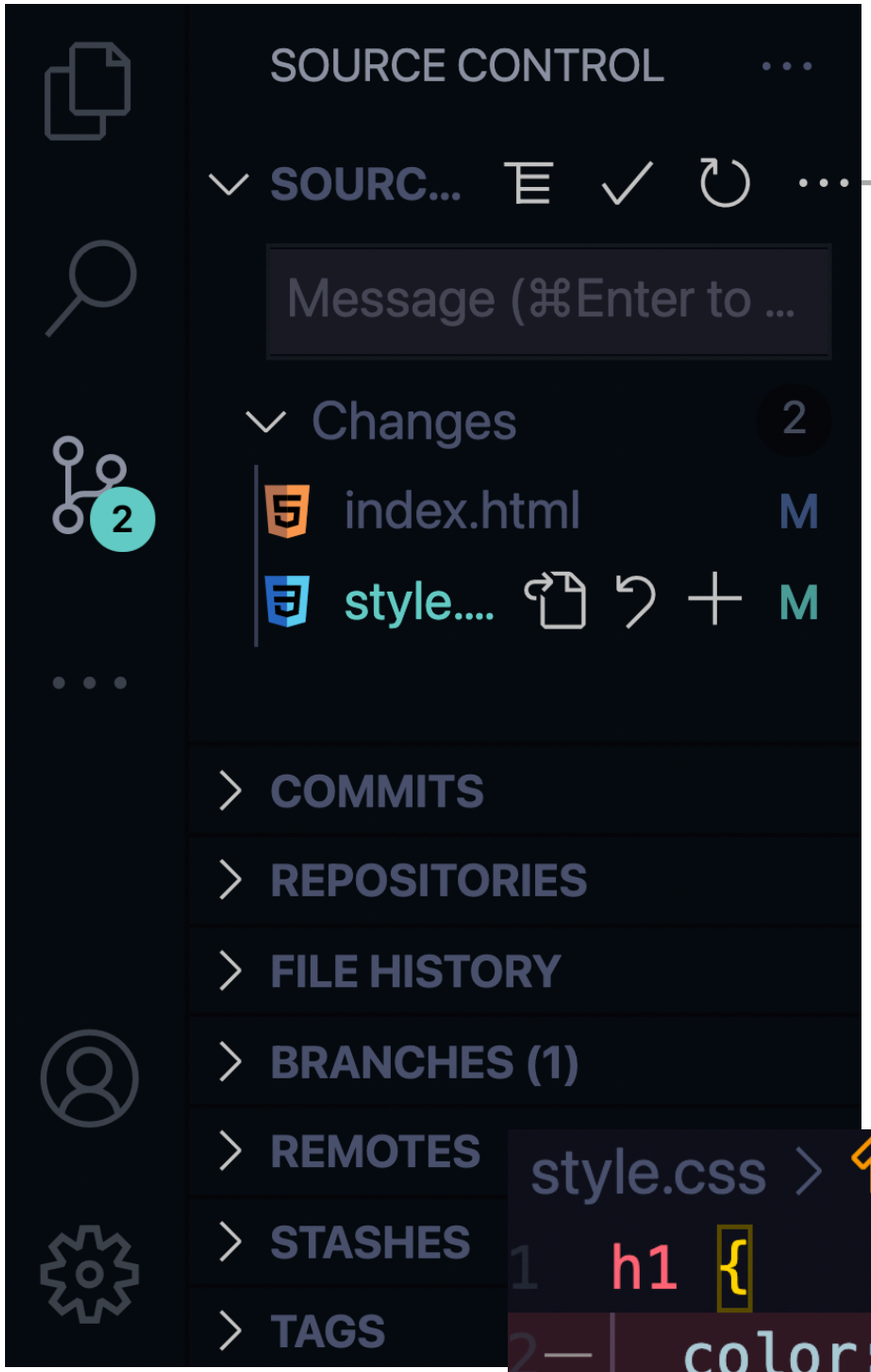


- Instead of using the terminal, you can use the **source control** section in Visual Studio Code.
- Once you have git installed and git initialized in whatever folder you want, you can use VSCode to add all commits.

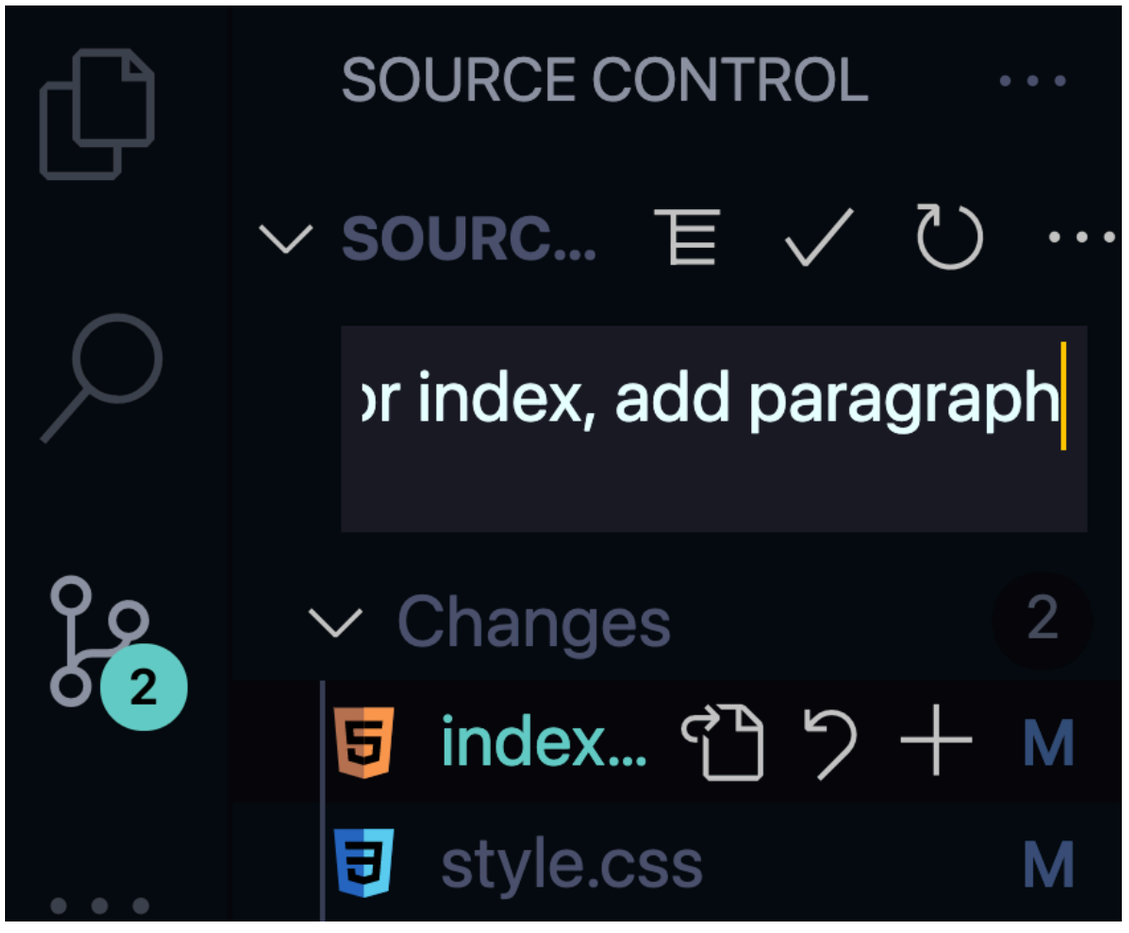
1. First, click on the 3rd button down on left.
    - The **2** means there are 2 files with changes that can be staged and committed.

VERSION CONTROL IN VSCODE

- Notice we have the files with changes listed. Click on each file to see all of your changes. I clicked on style.css.
- Red highlight and - are deletions, + and green highlight are additions:



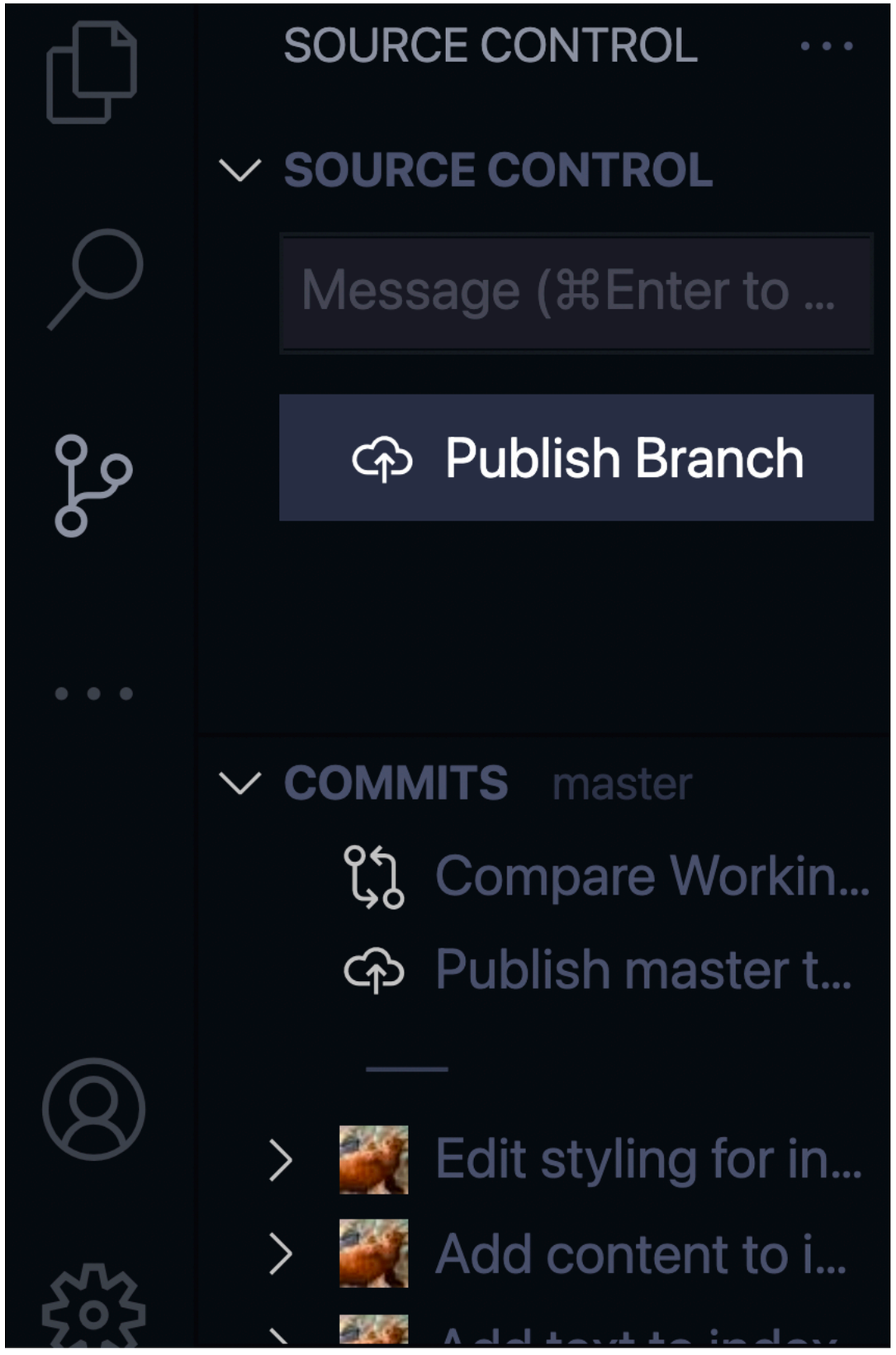
VERSION CONTROL IN VSCODE



2. Click on the text box that says *Message* then add your message for your commit.

3. Press the check mark above to commit.

VERSION CONTROL IN VSCODE



- Nothing there so commit completed!

4. If you want to see your commits, click on the **commits** tab.

- We'll learn about publishing to GitHub in a few slides.

DEMO

- Let's do a quick demo showing how to use git on your computer.
  Quick steps using terminal/command prompt:

  1. Check to see if git installed: **git ––version**
  2. Go to folder that want to use git, then initialize: **git init**
  3. Stage your files –
     a. Individual files: **git add *fileName***
     b. All changes: **git add .**
  4. Commit all changes: **git commit -m *"your message"***

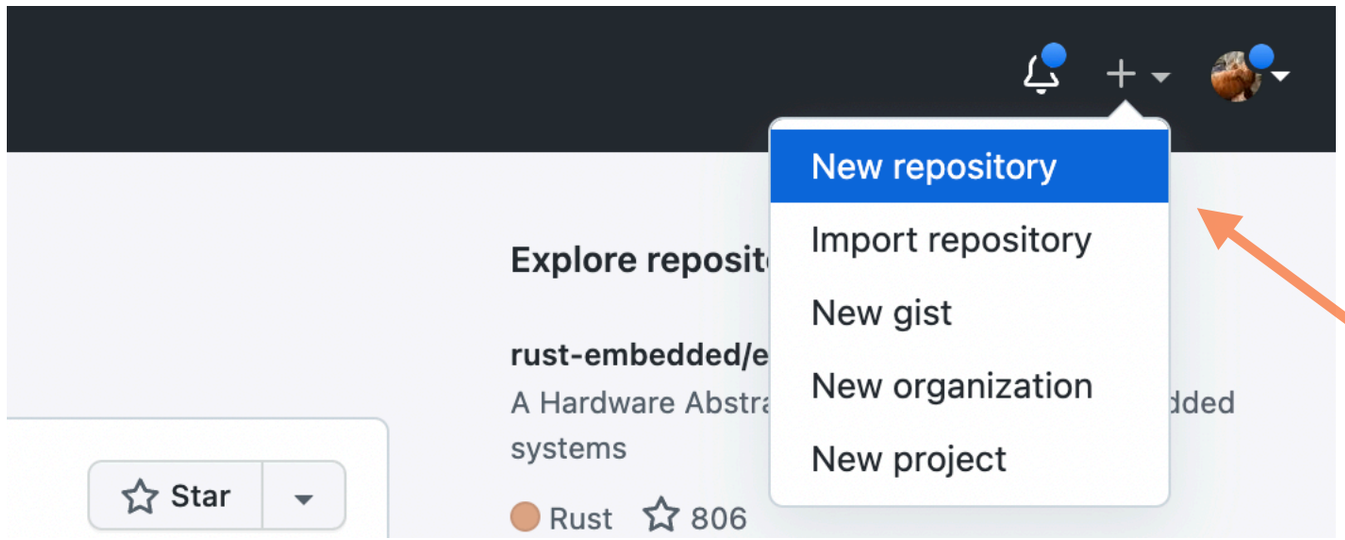
- You can use VSCode or GitHub to see changes, or you can use **git log** and **git diff *commitId*** if want to use terminal/command prompt.

WHAT IS GITHUB?

- Git is not the same as GitHub. GitHub makes tools that use Git. GitHub is the largest host of source code in the world.
- GitHub makes your code available to others online which makes it easy to work with other programmers.
- You will need a GitHub account for this course so I can access your code. To start, make an account:

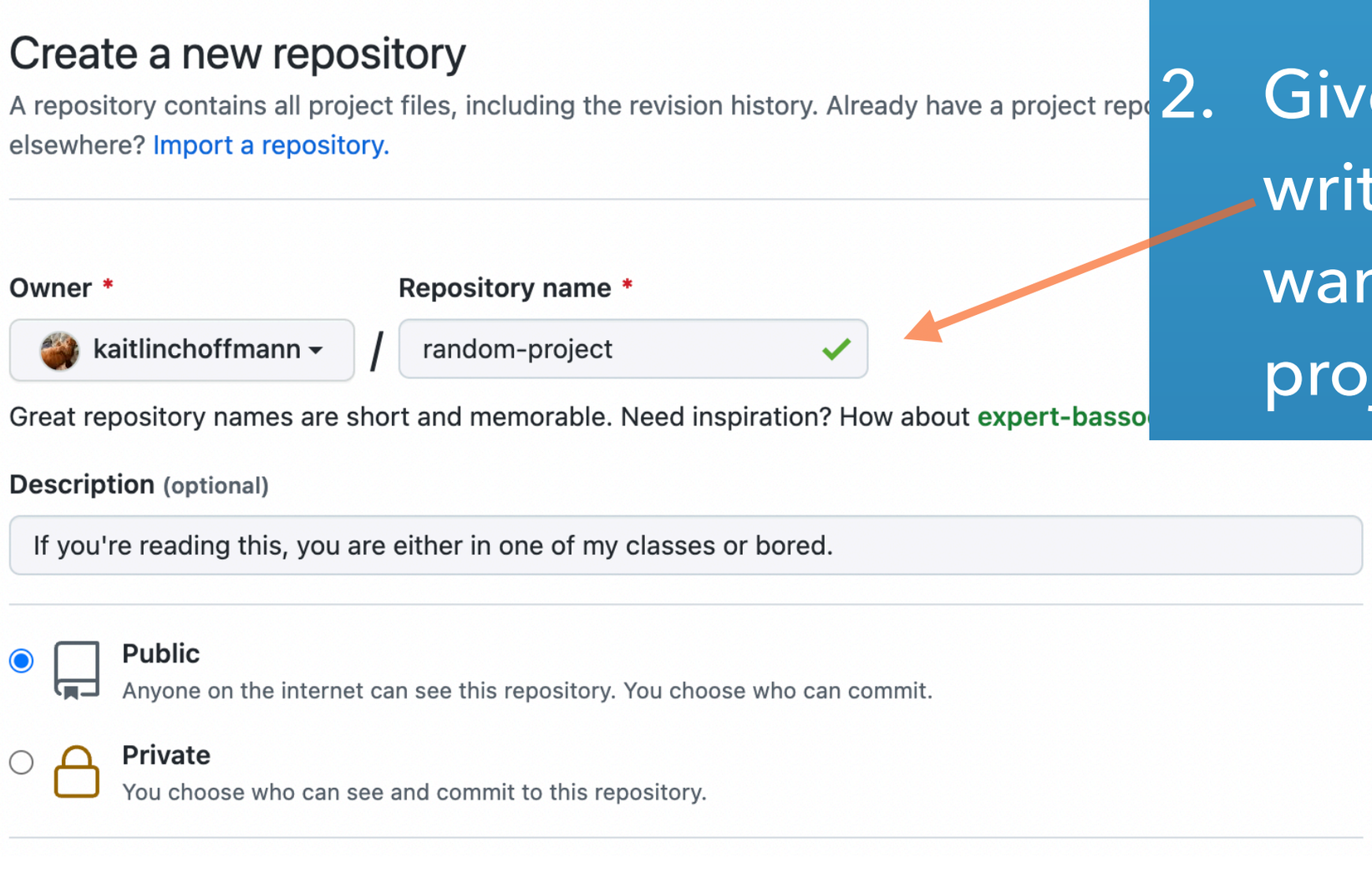
  https://github.com/

ADDING GIT REPO TO GITHUB

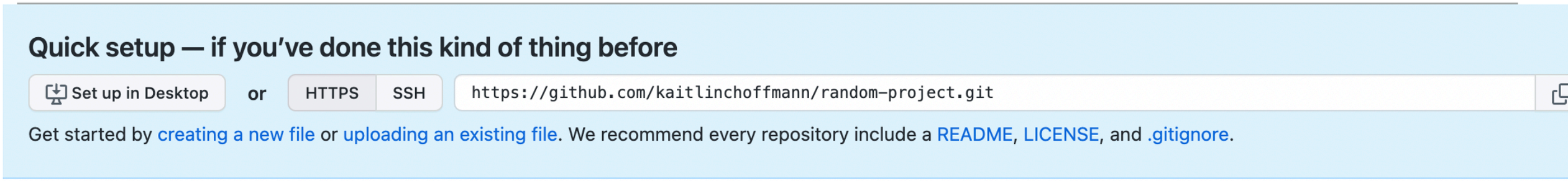


After you create your account and/or sign in:

1. On the right hand corner, press the **+** sign then **New repository**.
2. Give your project a name, write a description if you want, and make your project public or private.



ADDING GIT REPO TO GITHUB



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

Set up in Desktop or HTTPS SSH https://github.com/kaitlinchoffmann/random-project.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

```
echo "# random-project" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/kaitlinchoffmann/random-project.git
git push -u origin main
```

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

```
git remote add origin https://github.com/kaitlinchoffmann/random-project.git
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

- Pick whichever works for your situation. We already have a an existing repo, so we'll pick the 3rd option.

3. Copy the code in the box and paste it in your terminal.

ADDING GIT REPO TO GITHUB

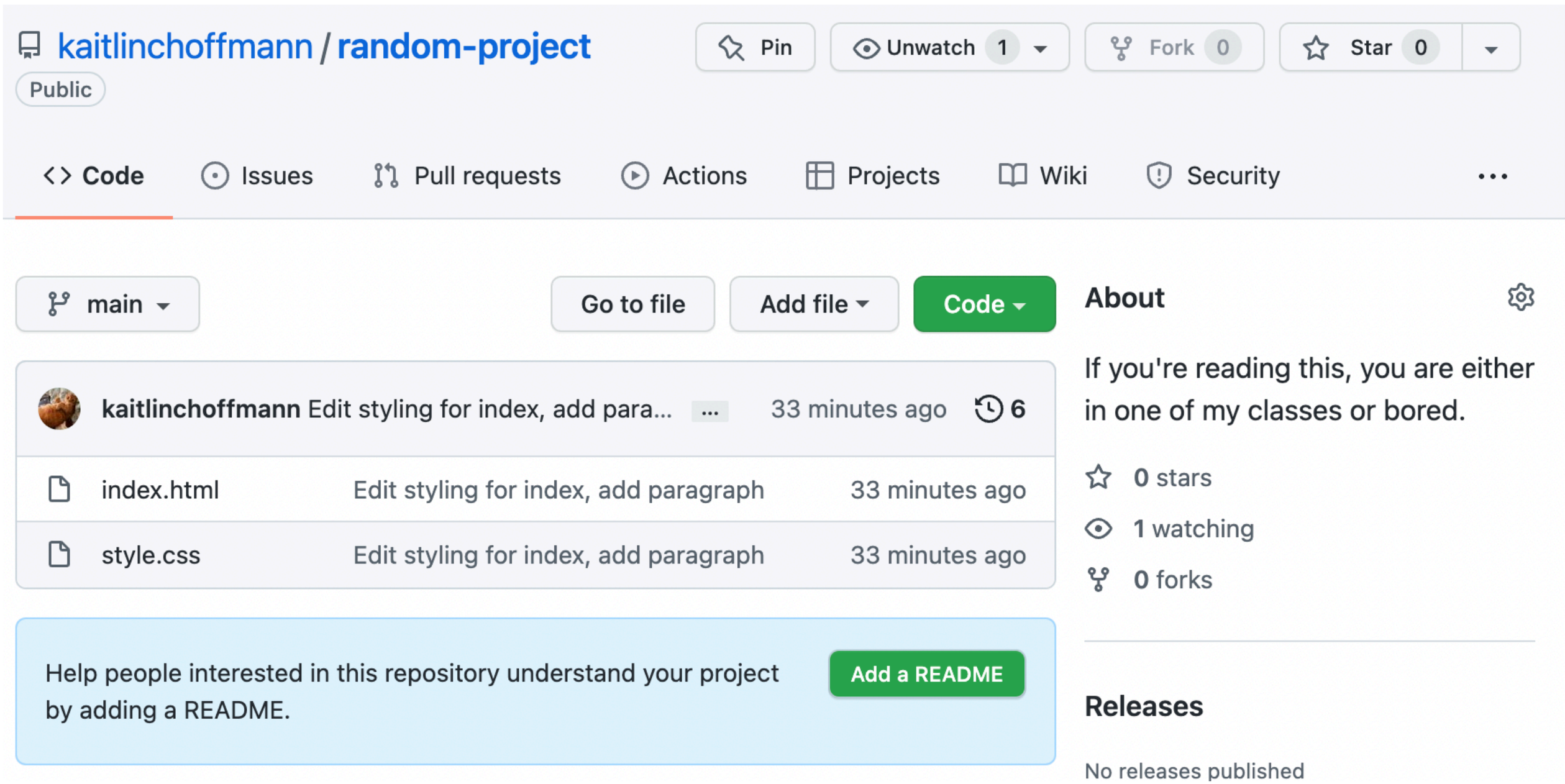
TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE

```
kaitlinhoffmann (main) Random Project $ git remote add origin https:/
/github.com/kaitlinchoffmann/random-project.git
git branch -M main
git push -u origin mainkaitlinhoffmann (main) Random Project $ git br
anch -M main
kaitlinhoffmann (main) Random Project $ git push -u origin main
Enumerating objects: 20, done.
Counting objects: 100% (20/20), done.
Delta compression using up to 8 threads
Compressing objects: 100% (17/17), done.
Writing objects: 100% (20/20), 2.44 KiB | 2.44 MiB/s, done.
Total 20 (delta 3), reused 0 (delta 0)
remote: Resolving deltas: 100% (3/3), done.
To https://github.com/kaitlinchoffmann/random-project.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
kaitlinhoffmann (main) Random Project $
```

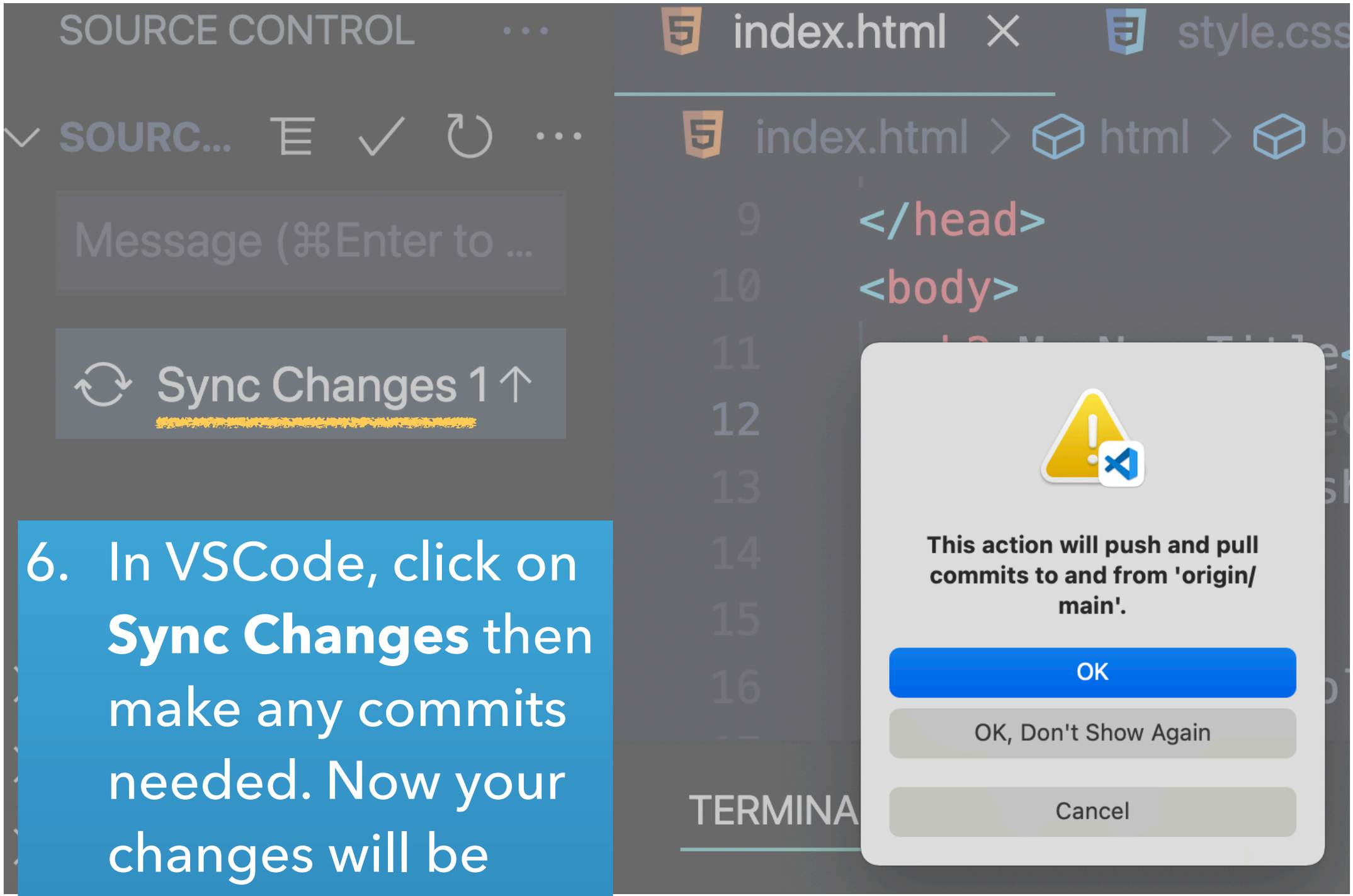
4. Paste the code, then press enter.

ADDING GIT REPO TO GITHUB HTTPS://GITHUB.COM/KAITLINCHOFFMANN/RANDOM-PROJECT

5. Refresh the page on GitHub and success!



ADDING GIT REPO TO GITHUB HTTPS://GITHUB.COM/KAITLINCHOFFMANN/RANDOM-PROJECT



6. In VSCode, click on **Sync Changes** then make any commits needed. Now your changes will be reflected in GitHub!

WHAT IF I WANT TO USE MY TERMINAL TO PUSH CHANGES?

- Instead of syncing, you can use your terminal to **push** and **pull** changes.
- The **push** command pushes all of your commits to your GitHub repo
- The **pull** command pulls all changes from GitHub to your git repo onto your computer. You may need to use this command if you are working with multiple people on the same project.

GIT PUSH

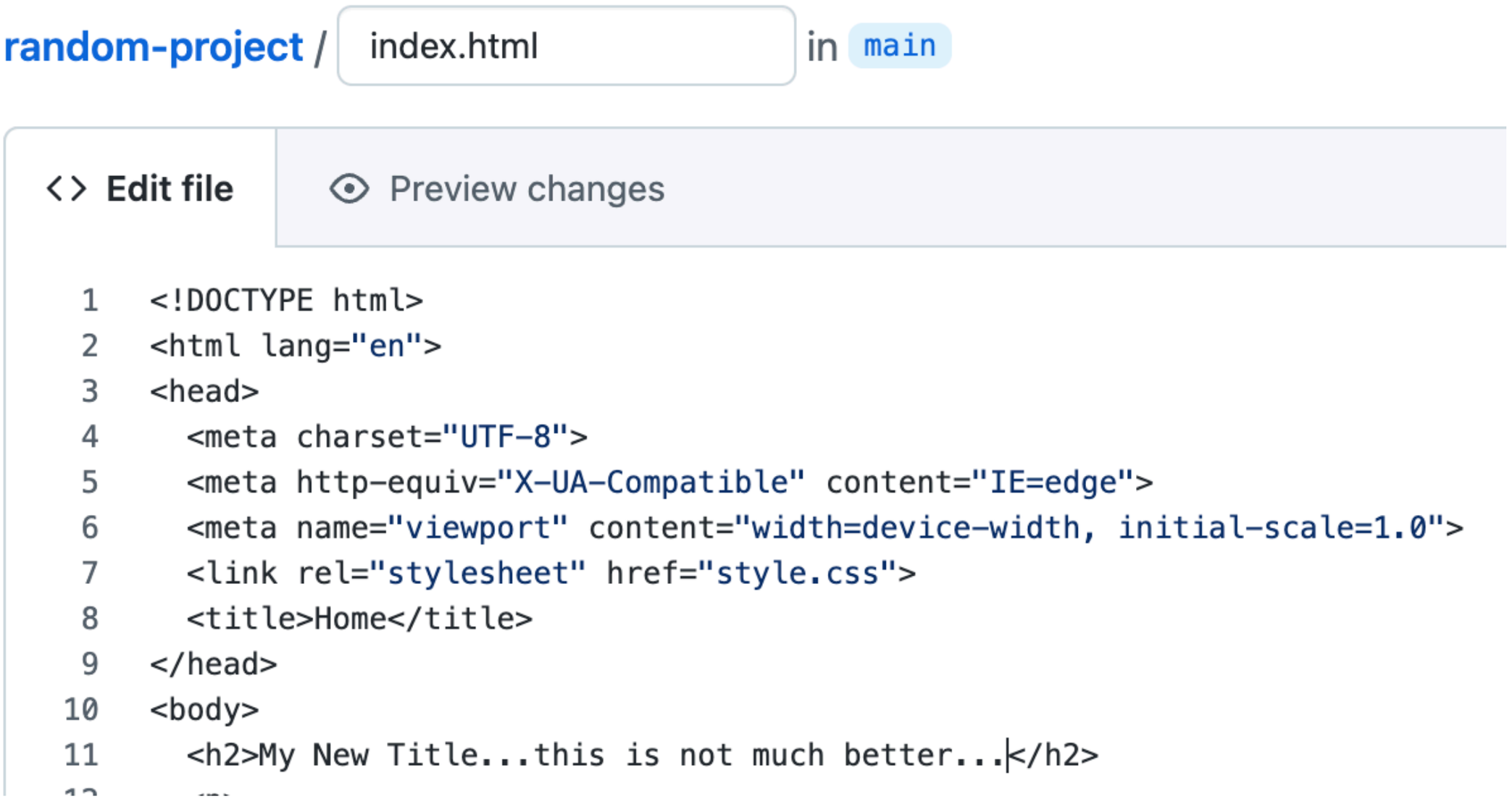
1. Submit your commits by **either** using VSCode or the terminal (git commit -m "*message*")

2. In your terminal, enter **git push origin**. Changes should be reflected on GitHub.

```
kaitlinhoffmann (main *) Random Project $ git push origin
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 312 bytes | 312.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/kaitlinchoffmann/random-project.git
   f7ad9e5..3b1e732  main -> main
kaitlinhoffmann (main) Random Project $
```

GIT PULL

1. Let's say I changed my files directly on GitHub. To do this, click on the file, then click on edit on the top right hand corner...I never do this, but maybe you need to do a quick fix.



GIT PULL

2. In your terminal, enter **git pull origin** to have all changes reflected in your Git repo on your computer.

```
kaitlinhoffmann (main) Random Project $ git pull origin
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/kaitlinchoffmann/random-project
   3b1e732..de6409d  main       -> origin/main
Updating 3b1e732..de6409d
Fast-forward
 index.html | 4 ++--
 1 file changed, 2 insertions(+), 2 deletions(-)
kaitlinhoffmann (main) Random Project $
```

GITHUB DEMO

- Let's do a demo publishing our project to GitHub. **Quick Steps:**

    1. Log into GitHub
    2. Click on **+** sign and **New Repository**
    3. Pick whichever step works for you (I usually just choose the 3rd option – copy code)
    4. Go to project in terminal, paste code. Press enter.
    5. Push your changes to GitHub using either the sync button in VSCode or by the command **git push origin**

GIT BRANCH

- We have been working in the main branch and that's all we will need to use in this course.
- We are not learning about how to create and utilize multiple branches in this course due to time and because this is a solo project. However, you can check out the Git/GitHub tutorial on W3Schools if interested:

  https://www.w3schools.com/git/git_branch.asp?remote=github

README.MD

- A **README** is a text file that introduces and explains a project. It contains information that is commonly required to understand what the project is about.
- It's an easy way to answer questions that your audience will likely have regarding how to install and use your project and also how to collaborate with you.
- Your README should be placed in the top level directory of the project. This is where someone who is new to your project will start out.

README.MD

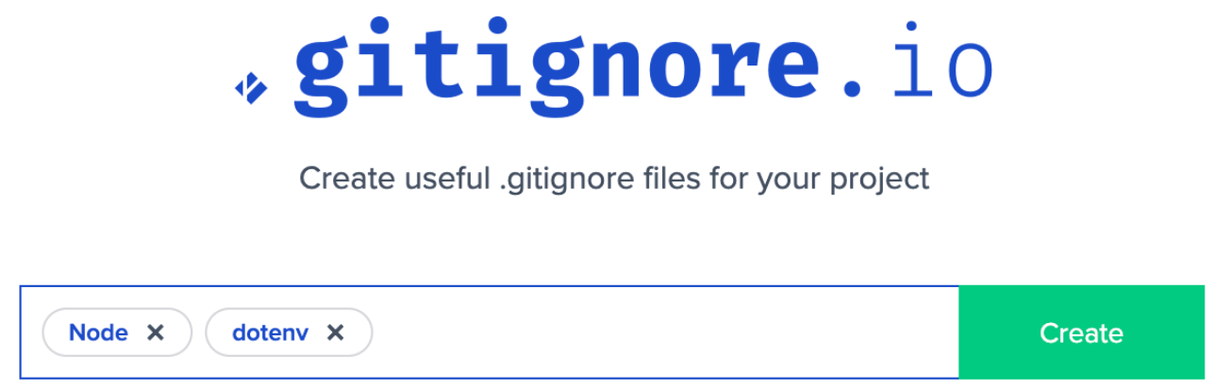
- READMEs can be written in any text file format, the most common one that is used nowadays is Markdown. This uses the extension **md**. Your file must be called ***README.md***
- Check out the following site to learn how easy it is to use Markdown: https://commonmark.org/help/
- For your next assignment, you'll need to add a ReadMe file.

.GITIGNORE

- A **gitignore** file specifies intentionally untracked files that Git should ignore. Files already tracked by Git are not affected.
- An example of a file we will want to ignore in the future is a **dotenv** file. This file contains environment variables which contain sensitive information, such as passwords and access keys. We'll be using a dotenv file in the future.

.GITIGNORE – EASY WAY TO SET UP FILE

1. First, go to https://www.toptal.com/developers/gitignore and enter node and dotenv. Click on **create**:



2. Copy the code and create a file called **.gitignore** in your project file. Paste the code inside of it and save it

3. If Node Modules is already being tracked, you will want to remove it from git control. Enter the following command in your terminal: ***git rm -r --cached .env***

SET EVERYTHING UP NOW

- You will need to submit your GitHub link to me from now on whenever you add to your project, so I highly suggest setting everything up with Git and GitHub now while it's all fresh in your head.
- If you want to make your repo private on GitHub, that is fine, you just need to give me access. (Go to settings while in your repo, click on add collaborators). My username is **kaitlinchoffmann**