Git notes

Git removes the need to copy files to and from the class share and your “H” drive as you collaborate.

Git is like using your camera to take a snapshot of your files at a specific point in time that can magically go back to if terrible things happen. Basically git is a checkpoint for your files.

Git exists so you can modify change break and improve your code secure in the knowledge that you can not ruin your work too badly because you created save points along the way.

Git is a collaboration tool that allows different people to work on all parts of a project at the same time.

Git is a tool that protects yourself and others from yourself and others.

The Local Workflow
You can still work on Git without network so if the internet goes down you can still work on it.

Steps:
Open file explorer
create a folder named practice in your “H” drive
type cmd in the address bar a command prompt should open to H:\practice> tell the git to watch this folder by initializing it

Git init creates a repository

There three main states:
Modified - files that are new or have changes not yet saved by Git

Staged - the current version of a file, tagged to be included in the next commit

Committed - files that are safely stored by Git

Open brackets and save an index.html file in your practice folder
Add the basic HTML structure but nothing else and save the file

Switch back to your command prompt window and check the status of your files with Gif

Git is tracking changes to the files after you input (git status)

Now add (git add index.html) to save progress

Git add neatly packs a copy of the specific file changes into a box ready to be stored indefinitely

Now did git status change?
Use (git commit -m ‘description goes here”) to assign it to a box storage and note what goes in there. You need to do this to be able to have checkpoints.

Git commit is physically moving the box of copies into a long term storage. It does not move or remove files in your working directory it makes copies so you can continue to work on it.

Check (git status) in your prompt. What does it say?

Add and commit the change: git add index.html git commit -m “updated title”

Repeat(when updated)
Status, add, commit, status

(git log)
Returns a list of all the commits (boxes) you have saved
The gibberish is the unique name of the box

Add a p tag that says something nice about the person next to you.

Status, add, commit

- Setup a Remote repository
- Learn how to push our local files to the remote server
- Understand branches and how to use them
- Understand how to merge different branches
- Learn what a Merge Conflict is and how to resolve it

A Remote repository is a copy of our project that is stored in the cloud
It is where we Backup our work and Share it with others
It is Accessible where there is an internet connection

Create a github account

Locate your project in file explorer
Initialize your local repository
Add and Commit your existing files

A. git add . (shortcut to add all files)
B. git commit -m “initial commit”

Tell git about the remote repository

- Remote add origin https://<YOUR URL>.git

Push your changes to the remote server

- git push -u <u>origin</u> master

- After the push it can simply be
  i. git push

Git push tells git to upload all your changes to the server. It **does not** need to be done after every commit, because it will upload **all** commits since the last push.

Branches are exactly what they sound like smaller bits extending from a tree trunk.
They represent different versions of our code.

Branches allows us to work on code fixes and features **without breaking** what we already have.
Fixes and new features should <u>always</u> start on a **branch**.

The **master** branch is the "trunk" of your code tree and should only contain **clean code ready for <u>deployment</u>** (use on the web).

Git branch <name> tells git to maintain a new copy of our code with the name given.

Git branch on its own will list the branches available and display an asterisk next to the one we are currently working on.

Git checkout <branch> tells git to switch our working folder to the branch name specified.

1. Create a new branch so we don't break our working code
   a. **git branch <u>mobile</u>**
2. IMPORTANT we have to switch to our new branch
   a. **git checkout <u>mobile</u>**

Open brackets and add a media query line to flex.css
   a. @media screen and (max-width:480px) {}
   b. Save file

Git the **status** of your file
Commit them
Go ahead and **push** them and check your **github** page

Checkout your **master** branch
   a. **git checkout master**

Git push --set-upstream origin mobile

Git merge <branch> combines the file changes in branch we name into our current working branch.

Then what is a merge conflict?

A merge conflict is when a file has a change in both of the branches you are trying to combine and git can't automatically determine what you want to keep.

Create and checkout a new branch named tablet:

```
Git branch tablet
Git checkout tablet
```

Create a tablet size MQ in flex.css"

```
@media screen and (max-width:1024px) {}
```

Note the line number you created this on

Check your flex.css file

Today's lesson helps with corporations because we have to use this to all work on the same project. This helps with being able to not lose work, and being able to work on it all at the same time. We have to collaborate and talk on what we want to change, where to put the content in specific places, and do various things.

From 1-4 it's about a 3 because not completely sure on how to break it anymore, but i can mostly navigate through it a little with some review.

Best part of thanksgiving break is pretty good.