When it comes to working in a group, success and communication are directly correlated.

Understanding the project workflow, and its associated tools, will be crucial towards success in this course.

All necessary software is downloaded / up to date.

Git

- Git is a distributed version control method… this is shown by having local *and* server copies of each file.
- A repository is a location in which all of the commit objects are contained. In other words, it is the storage location for all of the code that I'll write.
  - Repositories should include a REAME that introduce the software project.
- Username, email, and color set in the terminal.
- Git projects have three sections:
  - The working directory, which contains files that can be modified. In other words, this is the space in which you will be doing your work. After modifying files, they move to the next section, which is the (git add)
  - Staging directory, which finally passes your changes off to the (git commit)
  - Commit directory, or the repository.
- Before initializing Git on your machine, navigate to the folder in which you want to store the copy of your project. After doing this, just type: *git init*
- After doing this, type: *git remote add origin <project URL>*
- Finally, you can verify the URL using *git remote -v*
- Use *git status* to see which files are tracked, modified, etc.
- Git workflow:
  - Stage all of your changes at once:
    - *git add .*
  - Take a snapshot of those changes, making sure that the commit message actually describes the changes that were made:
    - *git commit -m <message>*
  - It is typically good practice to pull before you push
    - *git pull origin master*
  - Finally, we push these changes up to the server:
    - *git push origin master*
- *git log –-pretty=oneline* can be used to see the commit history of a project, which is good for the sake of tracking a project's version history.
- If you add something by accident, use *git reset HEAD <filename>* to remove it from the staging area. Use a *git status* to verify that the staging area is correct.
- If you commit something by accident, use *git reset --soft HEAD~1*

    - All this does is restore the status of the project to one commit prior, therefore removing the commit that was added by mistake. You will also have to remember to remove the erroneous file from the staging area.
- If you push something by accident, use *git revert HEAD* to create a commit that is actually a copy of the commit that happened before the push that you didn't want to do. Then be sure to push the reversion commit up to the server.
- *git diff* can be used to check for errors before going through the Git workflow.
- In order to combine the fetch / merge commands, use *git pull origin master*
- If there is a merge conflict, you must first manually decide which sections of the code to keep. After modifying and saving the file, restart and complete the Git workflow (however, avoid doing a pull during this process).
- Branching!
    - The see the names of branches, type *git branch -a*. Note that your current branch is denoted by an asterisk.
    - To create a branch, type *git branch <branch name>*
    - To then switch to this branch, use the *git checkout <branch name>* command.
    - To change the name of a branch, type *git branch -m <current name> <new name>*
    - To delete a branch, type *git branch -d <branch name>*
- In order to merge a branch, first merge onto master, and then use *git merge <branch name>*

Sitepoint Video on Mobile Design

- When designing an application, always consider the smallest, simplest hardware first. In other words, always start with a mobile consideration.
    - This is due to the fact that mobile devices have become part of almost every element of our society, and they aren't going away anytime soon.
- Don't make the user hunt for what they want.
- For mobile design, consider the average reach of the human hand. Interaction should occur near the bottom of the screen.
- Remember, users aren't likely to complain about an app. They'll simply move over to an app that better suits their needs.
- Users don't come to websites to look at ads – keep the content forward, and have the advertisements flow around that content as simply as possible.

Ch. 12 on Object Oriented Design

- As a reminder, the three main concepts of OO design are encapsulation, inheritance, and polymorphism.
    - Encapsulation – only expose essential functionality to the user.
    - Polymorphism – the same process can be used for different objects.

    - Inheritance – taking a preexisting object, inheriting its existing methods, and then adding additional functionality.
- I used this article to track what is and isn’t available within JavaScript, though I’ll still have to do some research as programming needs present themselves.

Article About “This”

The biggest take away for me from this article was that “this” is not bound, and its context is decided at compile time. This is definitely a switch to keep in mind.

Ch. 15 Modular JavaScript

I need to look into this further, but modules seem to be very helpful within the GitHub context.