Adding Files to the Repository

1. Covers
   1. Text Editors
      1. Text editor needed
         1. Windows: Notepad ++
         2. Mac: TextEdit
            1. SublimeText
            2. Atom
   2. Markdown
      1. Very simple markup based
      2. Text-based, easy to read
      3. No style information (unlike HTML)
      4. GitHub can display nicely
         1. GitHub assumes GitHub-flavored Markdown
      5. GitHub used for course
         1. But you don’t need to know it
   3. Useful Git Commands
      1. git status
         1. Provides useful information
      2. git add
         1. Moves one or more files from unstaged to staged
      3. git commit
         1. Commits any staged files
      4. git push
         1. Uploads committed files to GitHub

Unstaged is at the beginning of the process

1. Making Changes
   1. Covers
      1. What happens when a change is made
         1. File is automatically unstaged
         2. Use git add to stage
         3. There is still a chance to change before committing
            1. git add needs to be used again to stage
         4. git commit to commit
         5. git push to upload to GitHub
            1. Commands can be used for multiple files
      2. How to stage, commit, and push that change
2. Renaming and Deleting Files
   1. Covers
      1. How to rename a file using git
         1. Renaming is considered a change, so file is unstaged
         2. Need git add, git commit, and git push to upload to GitHub
            1. Alternative is to use git mv

Combines renaming and git add .

* + 1. How to delete a file using git
       1. Deletion is considered a change so file is unstaged
       2. Need git add . To stage
          1. Period means add everything from current directory
       3. git commit to commit
          1. Git commit –m “message in quotes”
       4. git push to upload to GitHub

1. Why is Git designed this way?
   1. Covers
      1. When would you make changes but not stage them?
         1. You don’t stage changes until they are at a point where you expect to commit them
         2. You don’t know yet that you are going to keep these changes
            1. Project manager wants total reorganization to see if it “works”
            2. Decide to delete all changes and go back to old organization
      2. When would you stage changes but not commit them?
         1. Staging gives final check to make sure changes are correct
      3. When would you commit changes but not push them?
         1. Until they are pushed, they are private
         2. May not be ready to share
2. Going back in time
   1. Covers
      1. The power of version control
         1. GitHub can show previous commits through UI
         2. Allows user to easily look back at previous work
      2. How to view previous commits in GitHub
      3. The HEAD commit
         1. Git labels a special commit as HEAD (all capitals)
         2. It means the commit currently being worked on
            1. Usually the most recent commit

Technically the most recent on a branch

* + - 1. HEAD can be moved to different commit
    1. How to work with previous commits in Git
       1. git log
          1. Shows history
       2. git log --oneline
          1. Presents a compact version
       3. git checkout
          1. Sets the commit being worked on
          2. Can set current version to an old commit
       4. git checkout master
          1. Set the current version to the most recent (on the master branch)

Can use git log --oneline command to get version number of old commit to move HEAD with git checkout command (current to old commit)

Git checkout master command moves HEAD to current version/commit

\*\*Only use for information, changes on old commits don’t work

* + 1. How Git works under the covers
       1. Changes saved in .git file
    2. Best practices for commits
       1. Commit often
       2. Ideally each commit for changes contain just one change
          1. Eg. Added a section, removed a section, changed the product name, etc.
       3. Clear, concise message for each commit
          1. Git commit -m “message in quotes”