# **Undo Changes in Git git checkout, git revert, & git reset**

**Undoing Local Changes That Have Not Been Committed**

If you have made changes that you don't like, and they have not been committed yet, do as follows:

1. In your terminal (Terminal, Git Bash, or Windows Command Prompt), navigate to the folder for your Git repo.
2. Run git status and you should see the affected file listed.
3. Run the following command, replacing filename.html with your file path (which you could copy and paste from the git status command):  
   -git restore --staged <file>  
   -git checkout filename.html
4. That file has now been reverted to the way it was at the previous commit (before your changes).

**Undoing a Specific Commit (That Has Been Pushed)**

If you have one specific commit you want to undo, you can revert it as follows:

1. In your terminal (Terminal, Git Bash, or Windows Command Prompt), navigate to the folder for your Git repo.
2. Run git status and make sure you have a clean working tree.
3. Each commit has a unique hash (which looks something like 2f5451f). You need to find the hash for the commit you want to undo. Here are two places you can see the hash for commits:  
   -In the commit history on the GitHub or Bitbucket or website.  
   -In your terminal (Terminal, Git Bash, or Windows Command Prompt) run the command git log --oneline
4. Once you know the hash for the commit you want to undo, run the following command (replacing 2f5451f with your commit's hash):  
   -git revert 2f5451f --no-edit  
   -NOTE: The --no-edit option prevents git from asking you to enter in a commit message. If you don't add that option, you'll end up in the VIM text editor. To exit VIM, press : to enter command mode, then q for quit, and finally hit Return (Mac) or Enter (Windows).
5. This will make a new commit that is the opposite of the existing commit, reverting the file(s) to their previous state as if it was never changed.
6. If working with a remote repo, you can now push those changes:  
   -git push

**Undoing Your Last Commit (That Has Not Been Pushed)**

If you made a mistake on your last commit and have not pushed yet, you can undo it. For example, maybe you added some files and made a commit, and then immediately realized you forgot something. You can undo the commit, and then make a new (correct) commit. This will keep your history cleaner.

1. In your terminal (Terminal, Git Bash, or Windows Command Prompt), navigate to the folder for your Git repo.
2. Run this command:  
   -git reset --soft HEAD~  
   -TIP: Add a number to the end to undo multiple commits. For example, to undo the last 2 commits (assuming both have not been pushed) run git reset --soft HEAD~2  
   -NOTE: git reset --soft HEAD~ is the same as git reset --soft HEAD^ which you may see in Git documentation.
3. Your latest commit will now be undone. Your changes remain in place, and the files go back to being staged (e.g. with git add) so you can make any additional changes or add any missing files. You can then make a new commit.

**Undoing Local Changes That Have Been Committed (But Not Pushed)**

If you have made local commits that you don't like, and they have not been pushed yet you can reset things back to a previous good commit. It will be as if the bad commits never happened. Here's how:

1. In your terminal (Terminal, Git Bash, or Windows Command Prompt), navigate to the folder for your Git repo.
2. Run git status and make sure you have a clean working tree.
3. Each commit has a unique hash (which looks something like 2f5451f). You need to find the hash for the last good commit (the one you want to revert back to). Here are two places you can see the hash for commits:  
   -In the commit history on the GitHub or Bitbucket or website.  
   -In your terminal (Terminal, Git Bash, or Windows Command Prompt) run the command git log --oneline
4. Once you know the hash for the last good commit (the one you want to revert back to), run the following command (replacing 2f5451f with your commit's hash):  
   -git reset 2f5451f  
   -git reset --hard 2f5451f  
   -NOTE: If you do git reset the commits will be removed, but the changes will appear as uncommitted, giving you access to the code. This is the safest option, because maybe you wanted some of that code and you can now make changes and new commits that are good. Often though you'll want to undo the commits and through away the code, which is what git reset --hard does.