Comparison Chart: ‘git reset’ Soft, Mixed, and Hard

This chart will help you compare the three different flavors of ‘git reset’: soft, mixed, and hard. Commands you would type at the command line are shaded blue. These commands should be typed in the software project's folder on your computer.

IMPORTANT NOTE: Do not use ‘git reset’ on a commit that was already pushed to a remote repository.

	git reset --soft	**git reset --mixed**	**git reset --hard**
Goal of Command	Roll back from committed to staged changes. Rolled-back changes still exist in the staging area, but will need to be re-committed before they can be pushed to a remote repository.	Roll back from committed to **un**staged changes. Rolled-back changes still exist in the working directory, but will need to be re-added and re-committed before they can be pushed to a remote repository.	Roll back to a previous commit with no record of the changes you are discarding. **Note: Can be dangerous! Do not use this command unless you are sure it is what you want.**
Example Reasons for Use	To alter a commit (for example, to change the commit message or include additional changes in the commit).	To make additional changes to your software project in the working directory and then re-stage and re-commit all changes as a group.	To remove unwanted changes that were committed.
Command syntax to roll back one commit	`git reset --soft HEAD^`	`git reset --mixed HEAD^`	`git reset --hard HEAD^`

	git reset --soft	git reset --mixed	git reset --hard
Command syntax to roll back multiple commits	git reset --soft HEAD~# where # is the number of commits to roll back	git reset --mixed HEAD~# where # is the number of commits to roll back	git reset --hard HEAD~# where # is the number of commits to roll back
Alternate command syntax	git reset --soft ####### where ####### is the commit ID (hash) to roll back to	git reset --mixed ####### where ####### is the commit ID (hash) to roll back to	git reset --hard ####### where ####### is the commit ID (hash) to roll back to