**Git revert:**

git-revert - Revert some existing commits

git revert [--[no-]edit] [-n] [-m parent-number] [-s] [-S[<keyid>]] <commit>…​

git revert (--continue | --skip | --abort | --quit)

Given one or more existing commits, revert the changes that the related patches introduce, and record some new commits that record them. This requires your working tree to be clean (no modifications from the HEAD commit).

Note: git revert is used to record some new commits to reverse the effect of some earlier commits (often only a faulty one). If you want to throw away all uncommitted changes in your working directory, you should see [git-reset[1]](https://git-scm.com/docs/git-reset), particularly the --hard option. If you want to extract specific files as they were in another commit, you should see [git-restore[1]](https://git-scm.com/docs/git-restore), specifically the --source option. Take care with these alternatives as both will discard uncommitted changes in your working directory.

**Git reset vs Git reverse:**

When trying to undo a mistake on git, sometimes we get confused on what exact command helps us get back on safe ground. There are 2 out of many git commands that help us do just that. Choosing the correct command from these 2 depends on the context of your situation. Let’s explore each command and in what situation do you use each.

**Git Reset**

git reset HEAD allows you to go back to a previous commit **and** removes any other commits on its way back. Imagine someone mopping a floor. If someone wants to clean up a hallway, they don’t just clean the end. They mop all of the floor all they way back to the end of the hallway. This type of undoing a mistake ensures you clean up everything on the way back to the commit you want to go back to. On the other hand, this is a destructive option and it doesn’t save any history of what you deleted.

**Git Revert**

git revert creates a new commit with the previous commit that you are reverting back to. Instead of destroying everything back to that commit, you simply make a copy of that and move forward with that commit. No commits are destroyed in the process. Imagine if you wanted to go back 30 commits. If you used git reset all 29 of those commits would be deleted. git revert allows you to save all that history and creates a safe way to proceed forward.

Conclusion

Understanding the difference of these 2 common undoing git commands will help you erase what you need in the right way. Most likely you will need to use both so knowing these 2 commands will help you successfully clean up any mistakes the right way.