**Undoing Things**

In this section, we will discuss the available ‘undo’ Git strategies and commands. It is first important to note that Git does not have a traditional ‘undo’ system like those found in a word processing application. It will be beneficial to refrain from mapping Git operations to any traditional ‘undo’ mental model. Additionally, Git has its own nomenclature for ‘undo’ operations that it is best to leverage in a discussion. This nomenclature includes terms like reset, revert, checkout, clean, and more.

A fun metaphor is to think of Git as timeline management utility. Commits are snapshots of a point in time or points of interest along the timeline of a project’s history. Additionally, multiple timelines can be managed through the use of branches. When ‘undoing’ in Git, you are usually moving back in time, or to another timeline where mistakes didn’t happen.

**Things You Should Know**

**git log**

* One of the best utilities for reviewing the history of a Git repository is the *git log* command.
  + By default, *git log* will only show commits for the currently selected branch.

**git checkout**

* The *git checkout* allows you to move to any snapshot/commit in the repo.
  + very safe and will not impact any files in the project
  + changes the local project files to match that commit

**git revert**

* The git revert command can be considered an ‘undo’ type command, however, it is not a traditional undo operation.
* Instead of removing a commit from the project history, it figures out how to invert the changes introduced by the commit and appends a new commit with the resulting inverse content.
  + This prevents Git from losing history, which is important for the integrity of your revision history and for reliable collaboration.
* Reverting should be used when you want to apply the inverse of a commit from your project history.

**git reset**

* The git reset command is a complex and versatile tool for undoing changes.
* It has three primary forms of invocation. These forms correspond to command line arguments –soft, –mixed, –hard.
  + each correspond to Git’s three internal state management mechanism’s, The Commit Tree (HEAD), The Staging Index, and The Working Directory.