Comparison Chart: ‘git diff’ for Different Use Cases

This chart will help you compare various uses of the ‘git diff’ command for comparing changes, commits, and branches. Commands you could type at the command line are shaded blue. These commands should be typed in the software project's folder on your computer, if you use them.

Note: ‘git diff’ cannot identify changes to untracked (unstaged) files. It will only recognize changes to files that have already been staged (added) with the ‘git add’ command.

	Compare Changes	Compare Commits	Compare Branches
Goal of Command	Compare the differences between two versions of a software project in a local repository, ***if one version is not yet committed***.	Compare the differences between two commits in a local repository.	Compare the differences between two branches in a local repository.
Command syntax	git diff to compare the staged (indexed) version to the working directory version; git diff HEAD to compare the most recent committed version to the working directory version; git diff --cached to compare the most recent committed version to the staged version.	git diff ####### ####### where each ####### is a commit ID (hash). Note that placing the older commit ID first and the newer commit ID second will tend to produce more intuitive results.	git diff oldbranch newbranch where oldbranch is the name of the older branch and newbranch is the name of the newer branch. For example, ‘git diff master devbranch’ would compare the master branch to a newer development branch called devbranch.