Undoing things with `git-restore`

General Syntax:

```
git restore [<options>] [--source=<tree>] [--staged] [--worktree] [--] <pathspec>…
git restore [<options>] [--source=<tree>] [--staged] [--worktree] --pathspec-from-file=<file> [--pathspec-file-nul]
git restore (-p|--patch) [<options>] [--source=<tree>] [--staged] [--worktree] [--] [<pathspec>…]
```

1. Unstage a staged file

`git restore --staged <file>`

The default `source` if we don't specify one is `HEAD`

`HEAD` is the current commit. Usually, it's the tip of the master branch (the last commit), unless we switch branches. We can view `HEAD` by opening the file in **.git/HEAD**.

2. Unmodify a modified file

`git restore [--worktree] <file>` (default)

The default `source` is `index`.

Index in Git is a file inside the .git directory that represents the project's staging area. We can think of it as a preview of our next commit.

We can unstage a file *and* restore it in the working tree in one command, too:

`git restore --source=HEAD --staged --worktree <file>`

(`HEAD` here is an example, but there must be a source specified for this command to work)

Examples

1.

```
$ git switch master
$ git restore --source master~2 Makefile  (1)
$ rm -f hello.c
$ git restore hello.c                     (2)
```

(1) take a file out of another commit
(2) restore `hello.c` from the index

2. To restore a file in the index to match the version in `HEAD` (this is the same as using git-reset[1])

   ```
   $ git restore --staged hello.c
   ```

3. or you can restore both the index and the working tree (this the same as using git-checkout[1])

   ```
   $ git restore --source=HEAD --staged --worktree hello.c
   ```

   or the short form which is more practical but less readable:

   ```
   $ git restore -s@ -SW hello.c
   ```