**Changing The Last Commit**

You've already made plenty of commits with the git commit command. Now with the --amend flag, you can alter the *most-recent* commit.

$ git commit --amend

If your Working Directory is clean (meaning there aren't any uncommitted changes in the repository), then running git commit --amend will let you provide a new commit message. Your code editor will open up and display the original commit message. Just fix a misspelling or completely reword it! Then save it and close the editor to lock in the new commit message.

**Add Forgotten Files To Commit**

Alternatively, git commit --amend will let you include files (or changes to files) you might've forgotten to include. Let's say you've updated the color of all navigation links across your entire website. You committed that change and thought you were done. But then you discovered that a special nav link buried deep on a page doesn't have the new color. You *could* just make a new commit that updates the color for that one link, but that would have two back-to-back commits that do practically the exact same thing (change link colors).

Instead, you can amend the last commit (the one that updated the color of all of the other links) to include this forgotten one. To do get the forgotten link included, just:

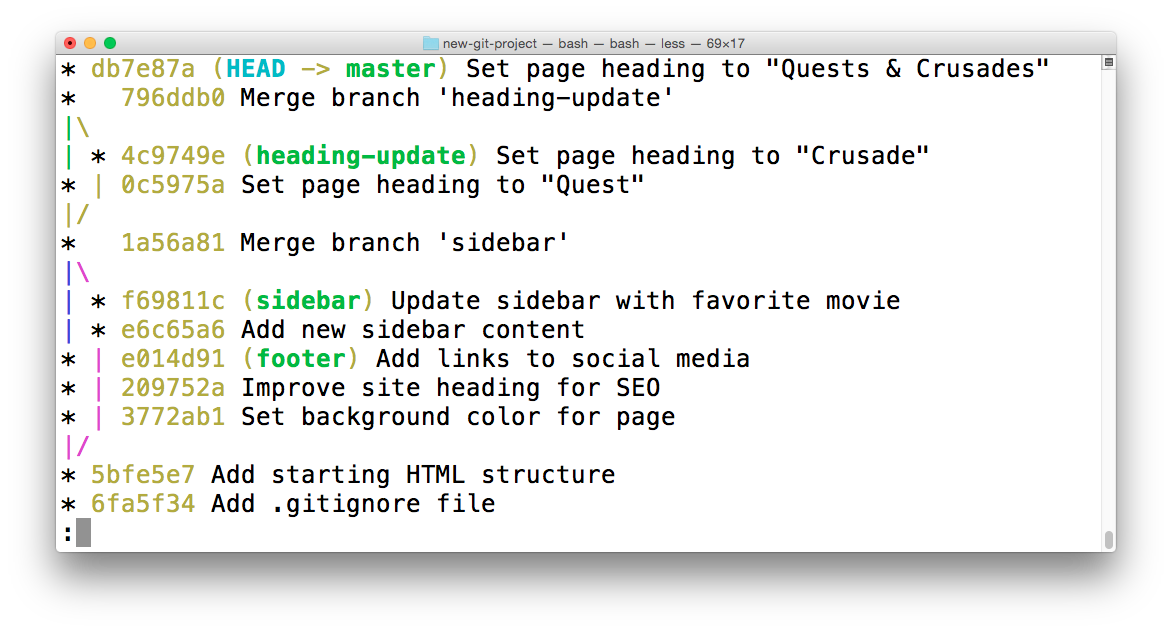
* edit the file(s)
* save the file(s)
* stage the file(s)
* and run git commit --amend

So you'd make changes to the necessary CSS and/or HTML files to get the forgotten link styled correctly, then you'd save all of the files that were modified, then you'd use git add to stage all of the modified files (just as if you were going to make a new commit!), but then you'd run git commit --amend to update the most-recent commit instead of creating a new one.

## What Is A Revert?

When you tell Git to **revert** a specific commit, Git takes the changes that were made in commit and does the exact opposite of them. Let's break that down a bit. If a character is added in commit A, if Git reverts commit A, then Git will make a new commit where that character is deleted. It also works the other way where if a character/line is removed, then reverting that commit will add that content back!

We ended the previous lesson with a merge conflict and resolved that conflict by setting the heading to Adventurous Quest. Let's say that there's a commit in your repository that changes the heading now to Quests & Crusades.

[[](https://classroom.udacity.com/nanodegrees/nd104/parts/912d84b6-bf81-4311-8e00-b3b2f63dc965/modules/90163ae6-479b-42ef-a088-e5c5f4c10101/lessons/4d79fbd6-5b43-433d-89ef-bbb5613dd35e/concepts/2e4b1c76-466d-494d-959c-a1f2aef5c7f7)](https://classroom.udacity.com/nanodegrees/nd104/parts/912d84b6-bf81-4311-8e00-b3b2f63dc965/modules/90163ae6-479b-42ef-a088-e5c5f4c10101/lessons/4d79fbd6-5b43-433d-89ef-bbb5613dd35e/concepts/2e4b1c76-466d-494d-959c-a1f2aef5c7f7)

[The Terminal application showing the log of a repository. The most-recent commit changes the heading from "Adventurous Quest" to "Quests & Crusades".](https://classroom.udacity.com/nanodegrees/nd104/parts/912d84b6-bf81-4311-8e00-b3b2f63dc965/modules/90163ae6-479b-42ef-a088-e5c5f4c10101/lessons/4d79fbd6-5b43-433d-89ef-bbb5613dd35e/concepts/2e4b1c76-466d-494d-959c-a1f2aef5c7f7)

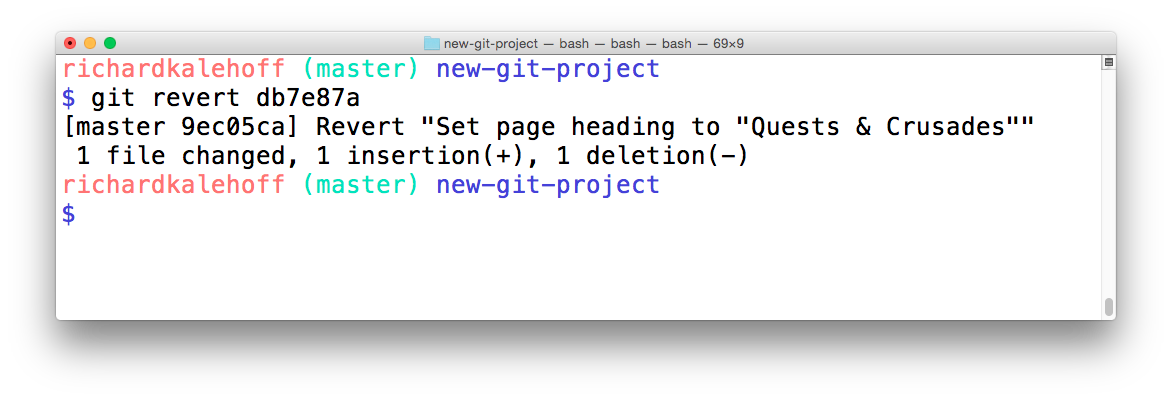
## The git revert Command

Now that I've made a commit with some changes, I can revert it with the git revert command

$ git revert <SHA-of-commit-to-revert>

Since the SHA of the most-recent commit is db7e87a, to revert it: I'll just run git revert db7e87a (this will pop open my code editor to edit/accept the provided commit message)

I'll get the following output:

[[](https://classroom.udacity.com/nanodegrees/nd104/parts/912d84b6-bf81-4311-8e00-b3b2f63dc965/modules/90163ae6-479b-42ef-a088-e5c5f4c10101/lessons/4d79fbd6-5b43-433d-89ef-bbb5613dd35e/concepts/2e4b1c76-466d-494d-959c-a1f2aef5c7f7)](https://classroom.udacity.com/nanodegrees/nd104/parts/912d84b6-bf81-4311-8e00-b3b2f63dc965/modules/90163ae6-479b-42ef-a088-e5c5f4c10101/lessons/4d79fbd6-5b43-433d-89ef-bbb5613dd35e/concepts/2e4b1c76-466d-494d-959c-a1f2aef5c7f7)

[The Terminal application showing the output of reverting a commit. The output provides the commit message of the commit that was reverted. It also creates a new commit to record this change.](https://classroom.udacity.com/nanodegrees/nd104/parts/912d84b6-bf81-4311-8e00-b3b2f63dc965/modules/90163ae6-479b-42ef-a088-e5c5f4c10101/lessons/4d79fbd6-5b43-433d-89ef-bbb5613dd35e/concepts/2e4b1c76-466d-494d-959c-a1f2aef5c7f7)

Did you see how the output of the git revert command tells us what it reverted? It uses the commit message of the commit that I told it to revert. Something that's also important is that it creates a new commit.

## Revert Recap

To recap, the git revert command is used to reverse a previously made commit:

$ git revert <SHA-of-commit-to-revert>

This command:

* will undo the changes that were made by the provided commit
* creates a new commit to record the change

### Further Research

* [git-revert](https://git-scm.com/docs/git-revert) from Git Docs
* [git revert](https://www.atlassian.com/git/tutorials/undoing-changes) Atlassian tutorial

## Reset vs Revert

At first glance, resetting might seem coincidentally close to reverting, but they are actually quite different. Reverting creates a new commit that reverts or undos a previous commit. Resetting, on the other hand, erases commits!

## *⚠️ Resetting Is Dangerous ⚠️*

*You've got to be careful with Git's resetting capabilities. This is one of the few commands that lets you erase commits from the repository. If a commit is no longer in the repository, then its content is gone.*

*To alleviate the stress a bit, Git*does*keep track of everything for about 30 days before it completely erases anything. To access this content, you'll need to use the git reflog command. Check out these links for more info:*

* [*git-reflog*](https://git-scm.com/docs/git-reflog)
* [*Rewriting History*](https://www.atlassian.com/git/tutorials/rewriting-history)
* [*reflog, your safety net*](http://gitready.com/intermediate/2009/02/09/reflog-your-safety-net.html)

## Relative Commit References

You already know that you can reference commits by their SHA, by tags, branches, and the special HEAD pointer. Sometimes that's not enough, though. There will be times when you'll want to reference a commit relative to another commit. For example, there will be times where you'll want to tell Git about the commit that's one before the current commit...or two before the current commit. There are special characters called "Ancestry References" that we can use to tell Git about these relative references. Those characters are:

* ^ – indicates the parent commit
* ~ – indicates the first parent commit

Here's how we can refer to previous commits:

* the parent commit – the following indicate the parent commit of the current commit
  + HEAD^
  + HEAD~
  + HEAD~1
* the grandparent commit – the following indicate the grandparent commit of the current commit
  + HEAD^^
  + HEAD~2
* the great-grandparent commit – the following indicate the great-grandparent commit of the current commit
  + HEAD^^^
  + HEAD~3

The main difference between the ^ and the ~ is when a commit is created from a merge. A merge commit has two parents. With a merge commit, the ^ reference is used to indicate the first parent of the commit while ^2 indicates the second parent. The first parent is the branch you were on when you ran git merge while the second parent is the branch that was merged in.

It's easier if we look at an example. This what my git log currently shows:

\* 9ec05ca (HEAD -> master) Revert "Set page heading to "Quests & Crusades""

\* db7e87a Set page heading to "Quests & Crusades"

\* 796ddb0 Merge branch 'heading-update'

|\

| \* 4c9749e (heading-update) Set page heading to "Crusade"

\* | 0c5975a Set page heading to "Quest"

|/

\* 1a56a81 Merge branch 'sidebar'

|\

| \* f69811c (sidebar) Update sidebar with favorite movie

| \* e6c65a6 Add new sidebar content

\* | e014d91 (footer) Add links to social media

\* | 209752a Improve site heading **for** SEO

\* | 3772ab1 Set background color **for** page

|/

\* 5bfe5e7 Add starting HTML structure

\* 6fa5f34 Add .gitignore file

\* a879849 Add header to blog

\* 94de470 Initial commit

Let's look at how we'd refer to some of the previous commits. Since HEAD points to the 9ec05ca commit:

* HEAD^ is the db7e87a commit
* HEAD~1 is also the db7e87a commit
* HEAD^^ is the 796ddb0 commit
* HEAD~2 is also the 796ddb0 commit
* HEAD^^^ is the 0c5975a commit
* HEAD~3 is also the 0c5975a commit
* HEAD^^^2 is the 4c9749e commit (this is the grandparent's (HEAD^^) second parent (^2))

## Which Commit?

Use this repository to answer the following quiz questions:

\* 9ec05ca (HEAD -> master) Revert "Set page heading to "Quests & Crusades""

\* db7e87a Set page heading to "Quests & Crusades"

\* 796ddb0 Merge branch 'heading-update'

|\

| \* 4c9749e (heading-update) Set page heading to "Crusade"

\* | 0c5975a Set page heading to "Quest"

|/

\* 1a56a81 Merge branch 'sidebar'

|\

| \* f69811c (sidebar) Update sidebar with favorite movie

| \* e6c65a6 Add new sidebar content

\* | e014d91 (footer) Add links to social media

\* | 209752a Improve site heading **for** SEO

\* | 3772ab1 Set background color **for** page

|/

\* 5bfe5e7 Add starting HTML structure

\* 6fa5f34 Add .gitignore file

\* a879849 Add header to blog

\* 94de470 Initial commit

### QUESTION 1 OF 4

Which commit is referenced by HEAD~6?

* 4c9749e
* 0c5975a
* 1a56a81
* f69811c
* e014d91
* 209752a

SUBMIT

You did so well on that last one, why not give this one a go! Using the same repository, which commit is referenced by HEAD~4^2?

SUBMIT

## The git reset Command

The git reset command is used to reset (erase) commits:

$ git reset <reference-to-commit>

It can be used to:

* move the HEAD and current branch pointer to the referenced commit
* erase commits
* move committed changes to the staging index
* unstage committed changes

## Git Reset's Flags

The way that Git determines if it erases, stages previously committed changes, or unstages previously committed changes is by the flag that's used. The flags are:

* --mixed
* --soft
* --hard

It's easier to understand how they work with a little animation.

## *💡 Backup Branch 💡*

*Remember that using the git reset command will*erase*commits from the current branch. So if you want to follow along with all the resetting stuff that's coming up, you'll need to create a branch on the current commit that you can use as a backup.*

*Before I do any resetting, I usually create a backup branch on the most-recent commit so that I can get back to the commits if I make a mistake:*

*$ git branch backup*

## Reset's --mixed Flag

Let's look at each one of these flags.

\* 9ec05ca (HEAD -> master) Revert "Set page heading to "Quests & Crusades""

\* db7e87a Set page heading to "Quests & Crusades"

\* 796ddb0 Merge branch 'heading-update'

Using the sample repo above with HEAD pointing to master on commit 9ec05ca, running git reset --mixed HEAD^ will take the changes made in commit 9ec05ca and move them to the working directory.

[[](https://classroom.udacity.com/nanodegrees/nd104/parts/912d84b6-bf81-4311-8e00-b3b2f63dc965/modules/90163ae6-479b-42ef-a088-e5c5f4c10101/lessons/4d79fbd6-5b43-433d-89ef-bbb5613dd35e/concepts/fed81eb7-49b4-4129-9f6b-8201e0796fd8)](https://classroom.udacity.com/nanodegrees/nd104/parts/912d84b6-bf81-4311-8e00-b3b2f63dc965/modules/90163ae6-479b-42ef-a088-e5c5f4c10101/lessons/4d79fbd6-5b43-433d-89ef-bbb5613dd35e/concepts/fed81eb7-49b4-4129-9f6b-8201e0796fd8)

[The Terminal application showing the result of resetting with the](https://classroom.udacity.com/nanodegrees/nd104/parts/912d84b6-bf81-4311-8e00-b3b2f63dc965/modules/90163ae6-479b-42ef-a088-e5c5f4c10101/lessons/4d79fbd6-5b43-433d-89ef-bbb5613dd35e/concepts/fed81eb7-49b4-4129-9f6b-8201e0796fd8)*[--mixed](https://classroom.udacity.com/nanodegrees/nd104/parts/912d84b6-bf81-4311-8e00-b3b2f63dc965/modules/90163ae6-479b-42ef-a088-e5c5f4c10101/lessons/4d79fbd6-5b43-433d-89ef-bbb5613dd35e/concepts/fed81eb7-49b4-4129-9f6b-8201e0796fd8)*[flag. The changes are unstaged.](https://classroom.udacity.com/nanodegrees/nd104/parts/912d84b6-bf81-4311-8e00-b3b2f63dc965/modules/90163ae6-479b-42ef-a088-e5c5f4c10101/lessons/4d79fbd6-5b43-433d-89ef-bbb5613dd35e/concepts/fed81eb7-49b4-4129-9f6b-8201e0796fd8)

## *💡 Back To Normal 💡*

*If you created the backup branch prior to resetting anything, then you can easily get back to having the master branch point to the same commit as the backup branch. You'll just need to:*

1. *remove the uncommitted changes from the working directory*
2. *merge backup into master (which will cause a Fast-forward merge and move master up to the same point as backup)*

*$ git checkout -- index.html*

*$ git merge backup*

## Reset's --soft Flag

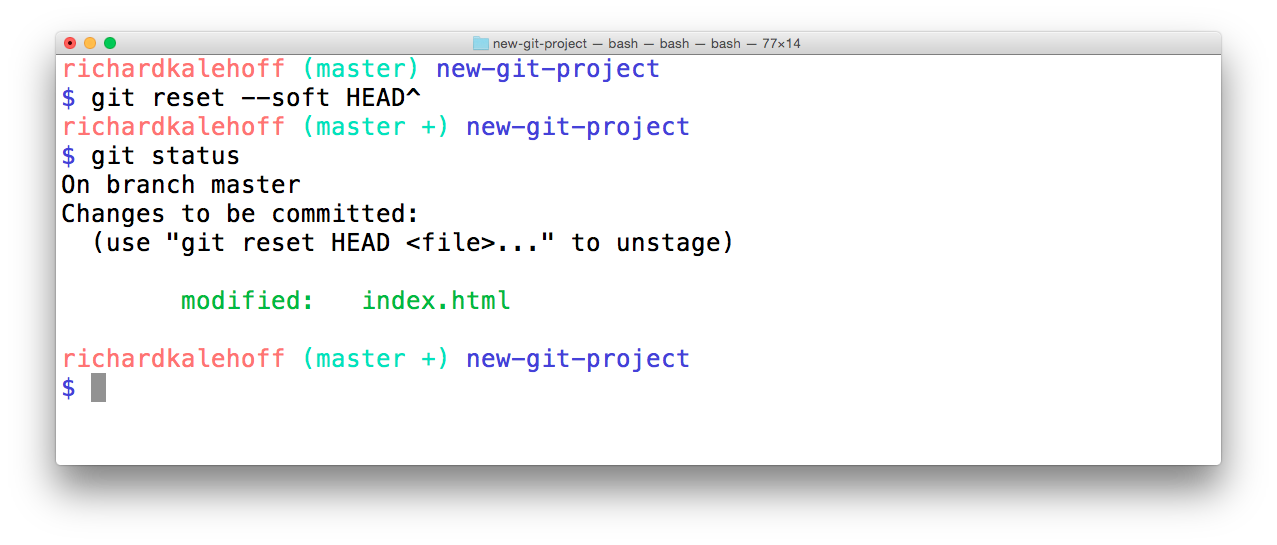
Let's use the same few commits and look at how the --soft flag works:

\* 9ec05ca (HEAD -> master) Revert "Set page heading to "Quests & Crusades""

\* db7e87a Set page heading to "Quests & Crusades"

\* 796ddb0 Merge branch 'heading-update'

Running git reset --soft HEAD^ will take the changes made in commit 9ec05ca and move them directly to the Staging Index.

[[](https://classroom.udacity.com/nanodegrees/nd104/parts/912d84b6-bf81-4311-8e00-b3b2f63dc965/modules/90163ae6-479b-42ef-a088-e5c5f4c10101/lessons/4d79fbd6-5b43-433d-89ef-bbb5613dd35e/concepts/fed81eb7-49b4-4129-9f6b-8201e0796fd8)](https://classroom.udacity.com/nanodegrees/nd104/parts/912d84b6-bf81-4311-8e00-b3b2f63dc965/modules/90163ae6-479b-42ef-a088-e5c5f4c10101/lessons/4d79fbd6-5b43-433d-89ef-bbb5613dd35e/concepts/fed81eb7-49b4-4129-9f6b-8201e0796fd8)

[The Terminal application showing the result of resetting with the](https://classroom.udacity.com/nanodegrees/nd104/parts/912d84b6-bf81-4311-8e00-b3b2f63dc965/modules/90163ae6-479b-42ef-a088-e5c5f4c10101/lessons/4d79fbd6-5b43-433d-89ef-bbb5613dd35e/concepts/fed81eb7-49b4-4129-9f6b-8201e0796fd8)*[--soft](https://classroom.udacity.com/nanodegrees/nd104/parts/912d84b6-bf81-4311-8e00-b3b2f63dc965/modules/90163ae6-479b-42ef-a088-e5c5f4c10101/lessons/4d79fbd6-5b43-433d-89ef-bbb5613dd35e/concepts/fed81eb7-49b4-4129-9f6b-8201e0796fd8)*[flag. The changes are moved to the Staging Index.](https://classroom.udacity.com/nanodegrees/nd104/parts/912d84b6-bf81-4311-8e00-b3b2f63dc965/modules/90163ae6-479b-42ef-a088-e5c5f4c10101/lessons/4d79fbd6-5b43-433d-89ef-bbb5613dd35e/concepts/fed81eb7-49b4-4129-9f6b-8201e0796fd8)

## Reset's --hard Flag

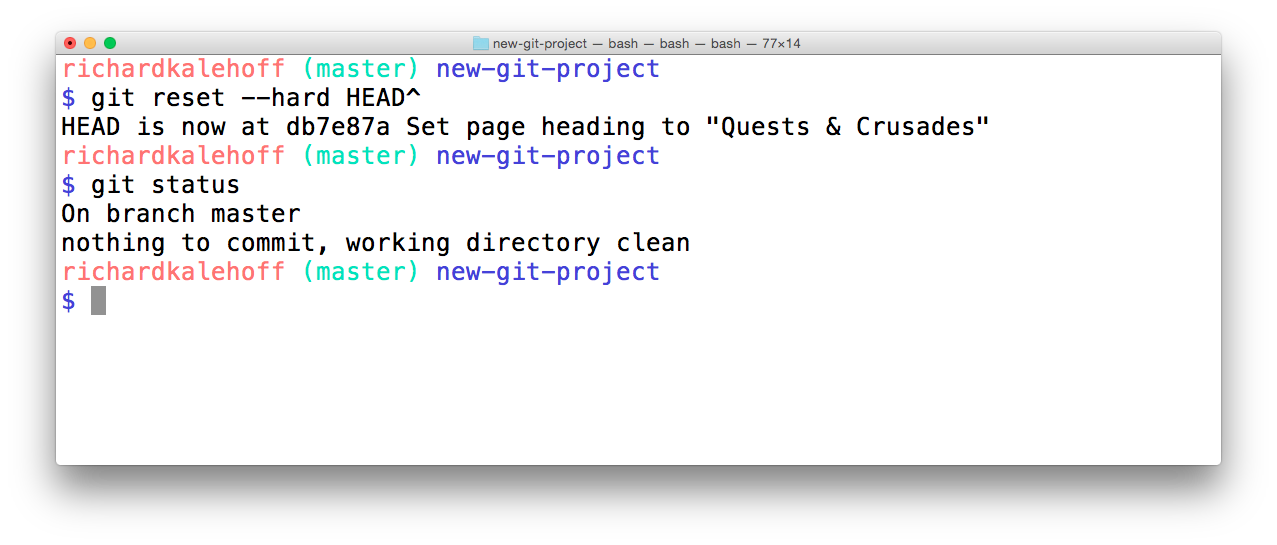
Last but not least, let's look at the --hard flag:

\* 9ec05ca (HEAD -> master) Revert "Set page heading to "Quests & Crusades""

\* db7e87a Set page heading to "Quests & Crusades"

\* 796ddb0 Merge branch 'heading-update'

Running git reset --hard HEAD^ will take the changes made in commit 9ec05ca and erases them.

[[](https://classroom.udacity.com/nanodegrees/nd104/parts/912d84b6-bf81-4311-8e00-b3b2f63dc965/modules/90163ae6-479b-42ef-a088-e5c5f4c10101/lessons/4d79fbd6-5b43-433d-89ef-bbb5613dd35e/concepts/fed81eb7-49b4-4129-9f6b-8201e0796fd8)](https://classroom.udacity.com/nanodegrees/nd104/parts/912d84b6-bf81-4311-8e00-b3b2f63dc965/modules/90163ae6-479b-42ef-a088-e5c5f4c10101/lessons/4d79fbd6-5b43-433d-89ef-bbb5613dd35e/concepts/fed81eb7-49b4-4129-9f6b-8201e0796fd8)

[The Terminal application showing the result of resetting with the](https://classroom.udacity.com/nanodegrees/nd104/parts/912d84b6-bf81-4311-8e00-b3b2f63dc965/modules/90163ae6-479b-42ef-a088-e5c5f4c10101/lessons/4d79fbd6-5b43-433d-89ef-bbb5613dd35e/concepts/fed81eb7-49b4-4129-9f6b-8201e0796fd8)*[--hard](https://classroom.udacity.com/nanodegrees/nd104/parts/912d84b6-bf81-4311-8e00-b3b2f63dc965/modules/90163ae6-479b-42ef-a088-e5c5f4c10101/lessons/4d79fbd6-5b43-433d-89ef-bbb5613dd35e/concepts/fed81eb7-49b4-4129-9f6b-8201e0796fd8)*[flag. The changes are moved erased.](https://classroom.udacity.com/nanodegrees/nd104/parts/912d84b6-bf81-4311-8e00-b3b2f63dc965/modules/90163ae6-479b-42ef-a088-e5c5f4c10101/lessons/4d79fbd6-5b43-433d-89ef-bbb5613dd35e/concepts/fed81eb7-49b4-4129-9f6b-8201e0796fd8)

Now it's your turn!

Refer to the following repository:

\* e014d91 (HEAD -> master, footer) Add links to social media

\* 209752a Improve site heading **for** SEO

\* 3772ab1 Set background color **for** page

\* 5bfe5e7 Add starting HTML structure

\* 6fa5f34 Add .gitignore file

\* a879849 Add header to blog

\* 94de470 Initial commit

### QUESTION 3 OF 4

What will happen to the changes from the 3772ab1 commit if git reset --hard HEAD~3 is run? Will the changes be in the Staging Index, in the Working Directory, or complete erased?

* Staging Index
* Working Directory
* erased

SUBMIT

### QUESTION 4 OF 4

What will happen to the changes from the 209752a commit if git reset --soft HEAD^^ is run? Will the changes be in the Staging Index, in the Working Directory, or complete erased?

* Staging Index
* Working Directory
* erased

SUBMIT

## Reset Recap

To recap, the git reset command is used erase commits:

$ git reset <reference-to-commit>

It can be used to:

* move the HEAD and current branch pointer to the referenced commit
* erase commits with the --hard flag
* moves committed changes to the staging index with the --soft flag
* unstages committed changes --mixed flag

Typically, ancestry references are used to indicate previous commits. The ancestry references are:

* ^ – indicates the parent commit
* ~ – indicates the first parent commit

### Further Research

* [git-reset](https://git-scm.com/docs/git-reset) from Git docs
* [Reset Demystified](https://git-scm.com/book/en/v2/Git-Tools-Reset-Demystified) from Git Blog
* [Ancestry References](https://git-scm.com/book/en/v2/Git-Tools-Revision-Selection#Ancestry-References) from Git Book