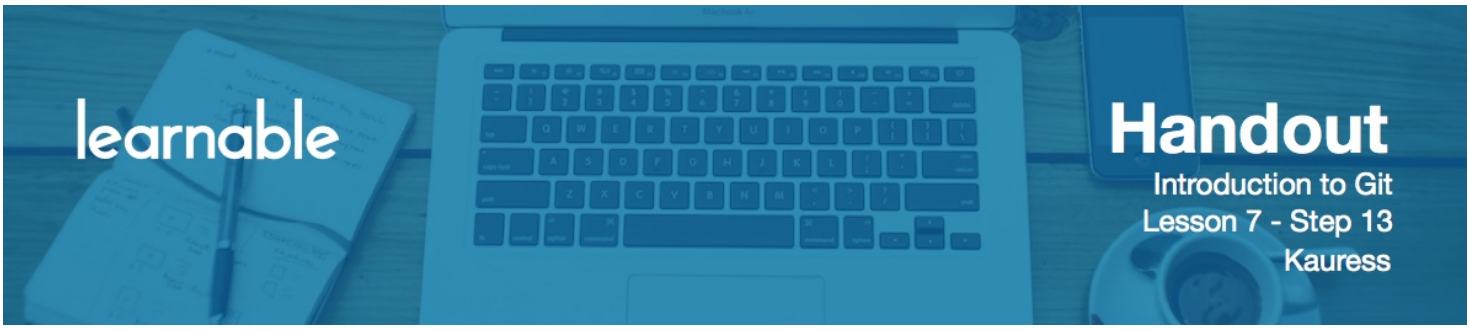

Rebasing in Git

As an extra pointer I will briefly introduce the `git rebase` command before we move on.

Both `git merge` and `git rebase` integrate changes from one branch into another, but the way they go about doing that is different.

`git merge` will create a merge commit to represent the integration of one branch into another. For example, when we merged our `featureA` branch into `master`, a new merge commit was generated on the `master` branch to represent the merge.

`git rebase` moves a branch up to a new base commit. `git rebase` takes the base name as an argument. This means "rebase the branch you're currently checked out on to the base branch". So it will merge another branch into the branch that you are checked out on, and will move all the local commits off the branch that you're checked out onto the tip of the base branch. It might seem that the branch commits are being moved up from one branch to another, however, internally, new commits are created and applied to the base branch, so the history of the branch is rewritten. `git rebase` command constructs a linear commit history.

Because you are rewriting the history of the branch, it's not considered a good practice to rebase on repositories that are being shared with others, as you don't want to rewrite history when working with other people.