Week 2

Tutorial on undoing changes, taken by instructor Francis.

Under **Undoing Changes**, we have:

- Git checkout.
- Git clean.
- Git revert.
- Git reset.
- Git rm.

We'll be starting with Git checkout.

Git checkout

Commonly used to switch branch but this command has various use case which is quite outstanding for a single command.

Note: `git checkout` operates on branches, files and commits.

Basic usage:

```
git checkout <file|commit|branch>
```

When used on a file that's in the staging area, the file is reverted to the current state it is in the HEAD commit.

Git clean

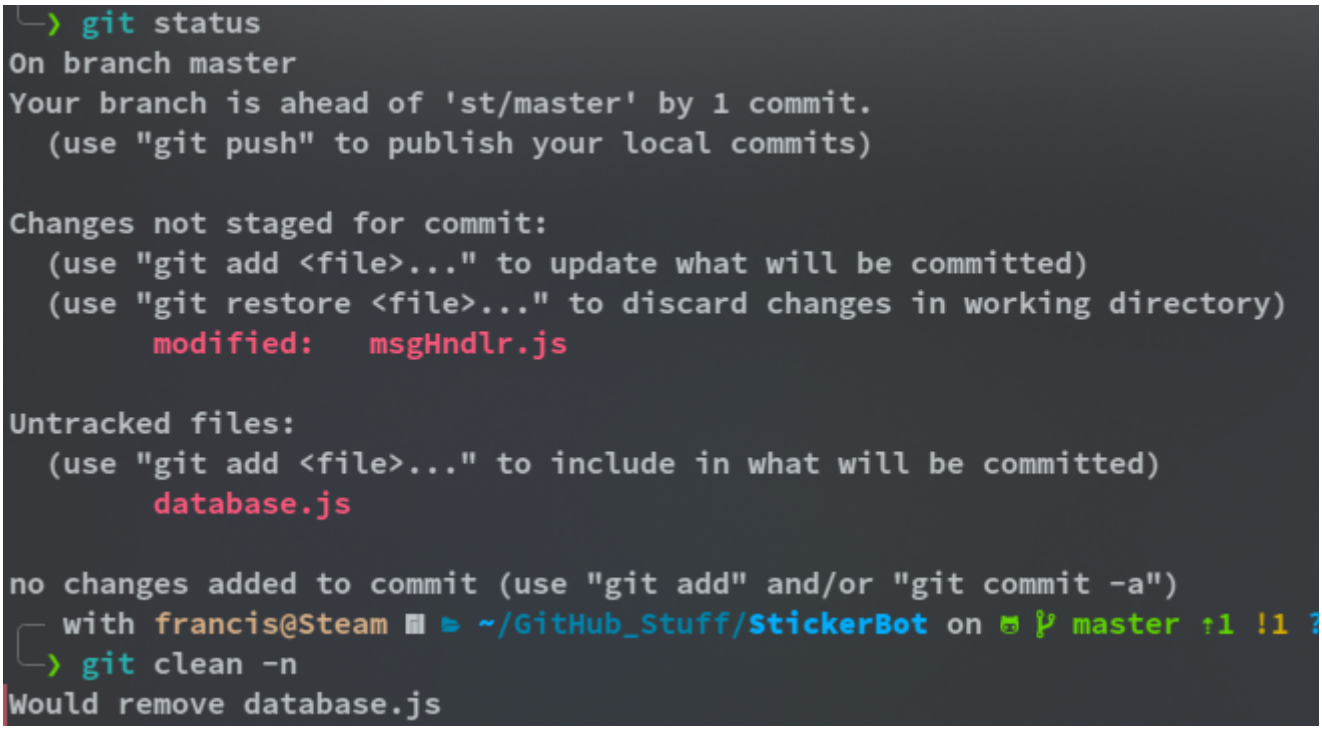
`git clean` is a command that shouldn't be used jokingly or as a prank as it deletes file which hasn't been tracked, in other words, it deletes file which we haven't used `git add <file>` or `git add .` on. As such the default behavior of `git clean` will throw an error, which is shown below

```
→ git clean
fatal: clean.requireForce defaults to true and neither -i, -n, nor -f given; refusing to clean
```

If we are entirely sure of our action, then we use

```
git clean -f
```

Let's say we aren't sure of what `git clean` will do, we run `git clean -n`



Output of git clean -n

We can see `database.js` is an untracked file therefore if `git clean -f` is run, `database.js` will be deleted.

By default directories aren't deleted when `git clean -f` is ran but if we want them gone, we use `git clean -fd`

What if we want files in `.gitignore` gone as well?

We use :

`git clean -xf`

Git revert

In simple words, it reverts(reverse) the commit specified. This command is very helpful in dire situations, like when a bug has been found and it's cause narrowed down to a specific commit, we get the SHA hash of the commit or if it's the topmost commit(HEAD) we just use

`git revert HEAD` or `git revert <SHA Hash>`

This will add a new commit which reverts the commit we specified so in our case we revert the bugged commit and our bug should be gone.

`git revert` can be considered as a commit undo and it doesn't mess with our commit history.

Git reset

Useful in some scenario and deadly in others. It's a command that can reset(undo, remove...) the commit indicated. It basically moves the current HEAD to the specified commit.

Basic usage:

```
git reset <commit hash>
```

Few options

```
git reset --soft <commit hash>
```

```
git reset --hard <commit hash>
```

--soft: This option only resets the commit history.

--hard: The most used option and the most dangerous. It resets the commit history, staging area and working directory.

Git rm

Like the Unix `rm` or `del` on Windows, this command removes files from a git repository.

When `git rm` is used, it removes the specified file then performs a `git add` to add the removed file operation to the staging index.

Basic usage:

```
git rm <file>
```

<file> can be an individual file, a group of files or directory.

Luckily, the effect can be reversed using either `git reset HEAD` or `git restore`.

That's all on Undoing changes.