Pre Commit Hooks: Combating Laziness

I have to admit that I'm absolutely lazy when it comes to actually testing and linting my Javascript code. I make sure it works but not to run unit tests or make sure that the code I write follows the coding style I always say I do.

Git to the rescue!

Git creates hooks when you initialize the repository and you can customize the hooks to perform taks during the life time of the request. There are hooks that we run during the commit process but before the files are actually pushed to the server.

We'll take advantage of the `pre-commit` hook to force Eslint to run before the code is committed to the repository. According to the Git Book:

> The `pre-commit` hook is run first, before you even type in a commit message. It's used to inspect the snapshot that's about to be committed, to see if you've forgotten something, to make sure tests run, or to examine whatever you need to inspect in the code. Exiting non-zero from this hook aborts the commit, although you can bypass it with `git commit --no-verify`. You can do things like check for code style (run lint or something equivalent), check for trailing whitespace (the default hook does exactly this), or check for appropriate documentation on new methods.

Note that this is a **client-side hook** that will not be part of the repository... We can copy this hook as part of the buiild process to make sure that all developers have the same commit hooks available to them.

The structure of the `.git` directory inside your project looks like this:

```
├── FETCH_HEAD
├── HEAD
├── config
```

```
├── description
├── hooks
|   ├── applypatch-msg.sample
|   ├── commit-msg.sample
|   ├── post-update.sample
|   ├── pre-applypatch.sample
|   ├── pre-commit.sample
|   ├── pre-push.sample
|   ├── pre-rebase.sample
|   ├── pre-receive.sample
|   ├── prepare-commit-msg.sample
|   └── update.sample
├── index
├── info
|   └── exclude
├── logs
|   ├── HEAD
|   └── refs
├── objects
|   ├── info
|   └── pack
├── packed-refs
└── refs
    ├── heads
    ├── remotes
    └── tags
```

We activate a hook by switching to the `.git/hooks` directory, removing the .sample suffix and replacing the content of the file with the actions that you want to perform. In this case the pre-commit script is a bash script that will run Eslint in the modified files before commiting them to the repository

```
#!/usr/bin/env bash

STAGED_FILES=$(git diff --cached --name-only -$(git diff --cached --name-o

eslint() {
```

```
  ESLINT="$"

eslint() {
  ESLINT="ESLINTr eslint
  if [[ ! -x "$ESLINT" ]]; then
    printf "\t\033[41mPlease install ESLint\033[0m\n"
    exit 1
  fi

  echo "Linting modified files"

  echo $STAGED_FILES | xargs $ESLINT

  if [[ $? == 0 ]]; then
    printf "\n\033[1;32mLint Pa"$ESLINT"0m\n"
  else
    printf "\n\033[41mLint Failed:\033[0m Fix lint errors and try again!\n
    exit 1
  fi
}

"
  else
    printf "# Exit if no files modified" = """ = "# Exit if no files modi
if [[ "$STAGED_FILES" = "" ]]; then
  exit 0
fi

eslint

exit $?
```

As I noted earlier this is a client side hook. If you want to enforce policy at the server level, check in Git Pro .