305CDE Worksheet 3

About

This week covers more advanced uses of objects and functions in JS.

Objects and functions are extremely powerful. This week explores some of the more advanced ways you can use both of them.

Task List

Aim to complete these in roughly 60-80 minutes of lab time. Use SourceTree git client and Brackets editor, *or other equivalent software that you are familiar with.*

1. Set up a secondary remote repository to store your worksheet solutions.
2. Write a closure to maintain the internal state of a counter.
3. Define a constructor function and modify its prototype.

(For the adventurous!)

4. Create a new JS module encapsulating functionality from task 3.

Resources

- Working with remote git repositories
- MDN guide to JS closures

Step-by-Step

1. Set up a second remote repository

Motivation Now that the code you are writing in lab sessions is more complex, you may wish to keep track of your solutions to the tasks.

Currently you are pulling changes from the remote origin on GitLab, containing the files for 305CDE. However, this repository is *read only* and you cannot push any changes to the repository.

To enable you to work on the repository, keep it up to date, AND keep track of your versioned changes you need another read-write remote repository that you can push your changes to.

In this task you will use the *git shell/terminal* to achieve this.

Fork the 305CDE repo `Forking` allows you to copy an entire repository to your own account.

- Log in to your GitLab account
- Browse to the 305CDE repository
- On the top right of the project screen click "Fork"
- Load your GitLab dashboard and you will see `305CDE` in your list of projects
- View the project page and note that it shows the original repo you forked from, as shown in the screenshot.

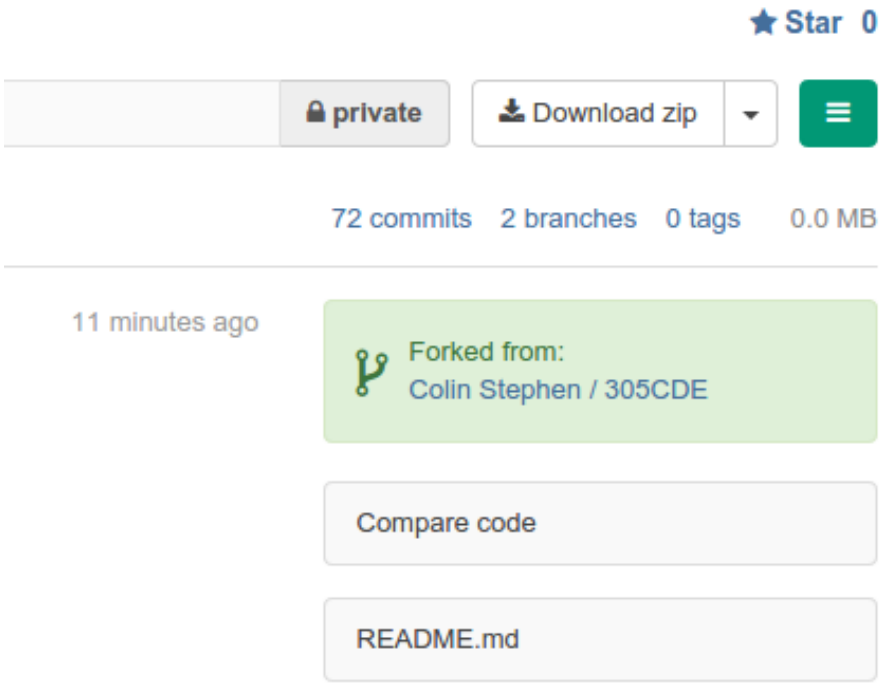

Figure 1: 305CDE Repository forked to user account

- If you wish to keep your new copy private, view settings and set the "Visibility Level" to *Private*.
- Copy the project `https://` URL listed on the project page for the next steps.

Using the git shell Next you will configure your git client to push any changes to your fork of the repository

- First select the git repo you are working on in the left hand column of SourceTree, if you have more than one

- Click on 'Terminal' in the toolbar of SourceTree to load a git shell
  - The terminal provides a text-input interface to git
  - All changes you do are reflected in the GUI later
  - It is much faster to use than the GUI
  - It will become easier to understand and use than the GUI too
- For information, list the current remote origin by typing `git remote -v` in the terminal
  - You will see the 305CDE repo origin listed for pushing and pulling, as shown in the screenshot.

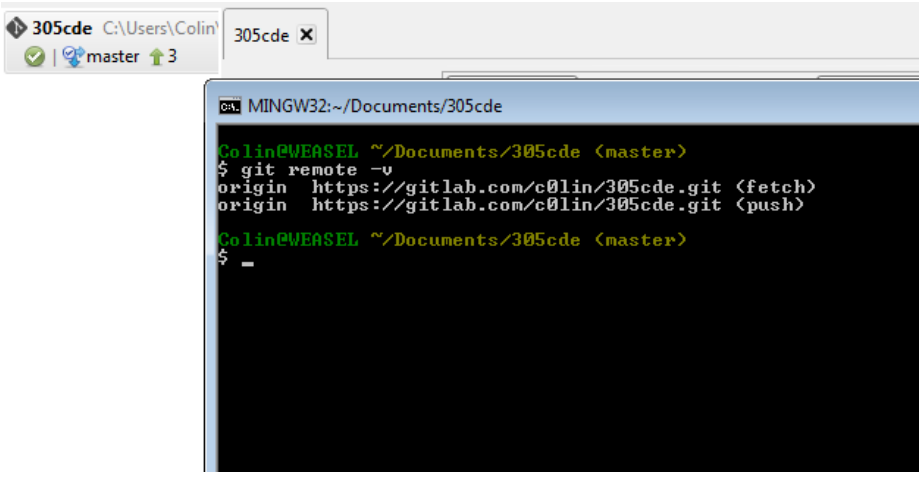

Figure 2: Git remote origin in shell

- To add a new remote corresponding to your fork of the project
  - `git remote add myfork https://your-https-url-for-the-fork`
- Check that it was added correctly with `git remote -v` again
  - This should now show a push and pull URL for `origin` and also for `myfork`

Now you can push any changes you make in labs to your remote `myfork`. The GUI gives you an option to choose which remote to push or pull from. In the terminal, you can push to the fork with the command `git push myfork`.

NOTE: A simple `git push` will default to origin, which will fail. See Working with remote git repositories for much more detail on working effectively with multiple remotes.

2. Use closure to maintain internal state

MDN guide to JS closures

- Load the `function_closure.html` file in Brackets and take a look at the functionality using live preview
- Observe that the `button.onclick` event handler is a function that uses the `count` variable
- Observe also that this function does not specify any value for `count`
- Instead, `count` inside the handler function is given its value *by the enclosing function context*

Since the `window.onload` handler function encloses the `button.onclick` one, the latter has an "environment" of local variables that it can work with. The `onclick` function is therefore able to keep track of the state of a higher-level variable, namely `count`.