SE3040 – Application Frameworks
BSc (Hons) in Information Technology
Software Engineering Specialization
3rd Year
Faculty of Computing
SLIIT
2023 - Practical
Lab 02

Lab session 2 – Asynchronous JavaScript & basics of Git

Objective: Teach a set of basic concepts in the JavaScript programming language and Git.

Prerequisites: Students should have basic JavaScript knowledge.

Asynchronous JavaScript

a. JavaScript Callbacks - Write a simple callback function for following scenario

> "The fetchData function takes in a URL and a callback function as parameters. It makes a request to the specified URL using the XMLHttpRequest API, and then calls the callback function with the response data (or an error) when the request is complete.
>
> The fetchData function is used with a callback function that logs the response data to the console if there are no errors, or logs the error to the console if there is one."

b. JavaScript Promises - Write a simple promise for following scenario

> "The fetchData function returns a Promise object that wraps an XMLHttpRequest.
>
> The Promise object is either resolved with the response data or rejected with an error message, depending on the result of the request.
>
> The fetchData function is used with a then method that logs the response data to the console if the Promise is resolved,
>
> or a catch method that logs the error to the console if the Promise is rejected."

c. JavaScript async & await - Write the code that explains following scenario

> "The fetchDataAsync function is an asynchronous function that uses the await keyword to wait for the Promise returned by fetchData to be resolved or rejected.

The try...catch block is used to handle any errors that occur during the execution of the function.

The fetchDataAsync function is used to fetch data from a URL and log it to the console if there are no errors,
or log the error to the console if there is one."

Basics of Git

1. Getting start with Git

   d. Download and install Git via - https://git-scm.com/downloads

   e. Check Git installed properly by open command prompt and type following command

   ```
   git --version
   ```

   f. That makes an output current version of Git.

   g. Setting your username in Git by – open git bash and execute the following code.
   Note: "Mona Lisa" can be replaced by your required username.

   ```
   git config --global user.name "Mona Lisa"
   ```

   h. Confirm that you have set the Git username correctly by using following command in git bash

   ```
   git config --global user.name
   ```

   The command will output the user's name that you entered.

   i. Set your email address in Git by executing following command.
   Note: "YOUR_EMAIL" can be replaced by your required email address.

   ```
   git config user.email "YOUR_EMAIL"
   ```

   $ git config --global user.email "user@gmail.com"

   j. Confirm that you have set the Git email address correctly by using following command in git bash

   ```
   git config user.email
   ```

   The command will output the email address that you entered.

   k. Add the email address to your account on GitHub. (more info - https://docs.github.com/en/account-and-profile/setting-up-and-managing-your-personal-

account-on-github/managing-email-preferences/adding-an-email-address-to-your-github-account)

2. Clone a repository

a. If you already have a repository on GitHub, you can clone the repository to your local machine. Use the following syntax.

```
git clone [repository_url]
```

Note: [repository_url] is the web url (HTTPS) of the repository.

3. Basic operations with Git

a. Initialize - The 'git init' command creates an empty .git repository or reinitializes an existing one.

```
git init
```

b. Status - The 'git status' command lists all the modified files ready to be added to the local repository.

```
git status
```

c. Add - The 'git add' command updates the index with the content in the working tree and prepares the content in the staging area for commit.

```
git add [directory] | [file]
```

Note: 'git add .' command to add all the files that haven't been added to the index yet.

```
git add .
```

d. Commit - After adding files to the staging environment, Git can package the files into a commit via the git commit command. The 'git commit' command instructs Git to store that file version.

```
git commit -m "Commit notes"
```

Note: You can replace "Commit notes" with your commit message.

-m option in the git commit command is used to include a commit message directly in the command line, without opening a text editor for you to enter a multi-line commit message

e. Pull - The ‘git pull’ command fetches changes from a remote repository to your local repository.
Before running the ‘git pull’ command, make sure that your central repo is set as origin.
Run following command.

```
git remote add origin [your-repository-link]
```

Then run the pull command.

```
git pull origin master
```

f. Push – The ‘git push’ command sends files from the local repository to the remote repository.

```
git push origin master
```

Other Important Commands

a. If you want to navigate through directories you can use ‘cd <path>’ in git bash.

b. If you want to create a branch via Git, use the following command.

git branch -M main

```
git branch [branch-name]
```

In Git, the git branch -M command is used to rename the current branch. The -M option stands for "move" or "rename.".

c. If you want to change the branch via Git, use the following command.

```
git checkout [branch]
```

d. The ‘git merge’ command allows you to merge two branches together.

```
git merge [branch-name]
```