**Command Lines**

**Git1**

**Create a new repository on the command line**

Echo “text” >> <filename.extension> / if the file does not exist

Git init

Git add <filename.extemsion> / add file in the local repositoty

**(or add folder)**

Git add <folder name> **(then)** / add folder and the files in it

Git add <folder name>/\*

Git commit -m “text here” / tracked changes and prepares them to be pushed to a remote repository like GitHub or just add a title to the existent local repository once it is pushed

Git remote add origin <URL>

Git remote -v / verifies the new remote repository

**(changing remote repository)**

Git remote set-url origin <URL>

**(create a branch and push the changes using the branch to the repository)**

Git push --set-upstream origin [the name of the branch]

Git checkout master

Git branch

(go to the repository and request a pull)

Git pull

(delete the branch)

Git branch -D [branch name]

Git branch [check the branch again]

**Not recommended (master should not be handled directly)**

Git push -u origin master / pushes commits to a repository on master

**Problems**

**Please enter a commit message to explain why this merge is necessary**

Press “i” / Write your merge message / Press esc / Write “:wq” / Then press enter

**Next.js**

Yarn create next-app / it will ask for the app’s name

Yarn run dev

**Create React**

npx create-react-app <name>

npm start

Using gif

![](namefile)

npx nodemon <file name>

npx install –save pg

**Link & Router**

Npx or npm install react-router-dom

**Git2**

**Push an existing repository from command line**

git remote add origin url

git push -u origin master

**Push a remote repository into your local repository**

Create a folder

cd <folder>

git clone <http>

**To clone a branch from the terminal**

Git clone -b <branch> <remote branch>

**Terminal command lines**

mkdir / creates a new folder

touch / creates a new file

rm / deletes a file

rm -r /deletes directories and files

ls -R | grep <file name> /to find the file

find . <file name> / to find the directory of the file

mv <filename.extension> <target-directory(folder)> / <new-name.extention>

/ move a file to a directory and rename the file

mv <filename.extension> <target-directory(folder)> / <filename.extention>

/ move a file to a directory

cp <filename.extension> <another-filename.extension> /copy a file

cp -r <directory-name> <another-directory-name> /copy the directory recursively

find . -name <filename.extension> / find a file

find . -name <filename\*.extension> / \*find files that match multiple files.

cd ~ /to navigate to home directory

**Git3**

**Merge**

**Create a new branch**

Git branch [yourbranchname]

**Merging master into a branch**

Git checkout [yourbranchname]

Git merge master

**Create a new branch and check it out at the same time**

Git checkout -b [yourbranchname]

Git commit

**Go back to master in the branch**

Git checkout master

Git commit

**Merge branch into master**

Git merge [yourbranchname]

**Git4**

**Rebase**

**Checkout a new branch**

Git checkout -b [yourbranchname]

Git commit

**Go back to master in the branch**

Git checkout master

**Rebase onto master**

Git rebase master

**Rebase onto branch**

Git rebase [yourbranchname]

**Git5**

**Head**

Head is essentially what commit you’re working on top of, head always point to the most

**Head before and after a commit**

Git checkout [branchname]

Git checkout master

Git commit

Git checkout [branchname]

**Detaching head**

Git checkout [branchname]

**Git6**

**Relative commits**

Moving upwards one commit at a time with ^

Moving upwards a number of times with ~<num>

**Commit above master**

Git checkout master ^

**Reference HEAD as a relative ref.**

**This will move HEAD to the top as a reference, depending of the number of pointers to refer, on this example only two.**

Git checkout [yourbranchname]

Git checkout HEAD^

Git checkout HEAD^

**Specifying the number of commits back with ~**

**If there are 4 commits then,**

Git checkout HEAD~4

**Moving or forcing the master branch to three parents behind HEAD**

Git branch -f master HEAD~3

Or Git branch -f [branchname] pointer

**Git7**

**Reset**

**It changes by moving a branch reference backwards in time to an older commit.**

Git reset HEAD~1

**Git Revert**

**Changes and share those reversed changes with others.**

Git revert HEAD

**Visual Studio Code**

“**Shift, option and arrow down”**

copy the highlighted code down

**“fn”**

bring the cursor at the end or beginning of the code line

**“command, return”**

move down the next line at any point in the line of code

**“command, d”**

highlight the same context below and change it all at the same time.

**“option, arrow up or down”**

move the piece of code highlighted up or down

**“.[classname]\*[number]”**

It creates a number of classes at once

Shorthand to create a class and id in html = #[id name].[class name]

**Li{$}\*4**

It will create four automatic points

**Git8**

**Information about commits**

**Git clone** “copy of a repository into your directory”

**Git log** “to get information”

**Git log** **--oneline** “to get information”

**Git log –stat** “get more details about commits”

**Git log -p** “display the actual changes made to a file”

**Git show** <commit number>

**Git add** <file.extension>

**Git add .** “refer to all the files and directories”

echo "# 01-03-2020-day-2-javascript-basics-braucalderon" >> README.md

git init

git add README.md

git commit -m "first commit"

git remote add origin https://github.com/lehman-cuny2x/01-03-2020-day-2-javascript-basics-braucalderon.git

git push -u origin master