**GIT**

-version control system => tracking changes in computer files

-many developers can work on the same project whithout having to be on the same network (distributer version control)

-tracks all the changes between multiple developers

-you often work on a project and have it in your local repository and then post in on a remote repository like GitHub, BitBucket

**Concept:**

-keeps track of code history

-takes **snapshot** of your files by making a **commit** (with command) =>

-you can always look at the pervious snapshots(your perivous code) and revert

-your code is safe with GIT

**Commands**

-$ git init = Initialize Local Git Repository (it will create a .GIT folder, location is based on the command)

-$ git add <file> = adds files that you specify to index

-$ git rm --cached <file> = removes files that you specify to index

-$ git status = check status of working tree

-$ git commit = commit changes in index (puts files from index to local repository)

//For remote repository

-$ git push = push to remote repository

-$ git pull = pull latest from remote respository

-$ git clone = clone remote repository into current folder

//Other commands

-$ git --version = checks the version

-$ touch index.html = creates a certain file

**-$ git config --global user.name ‘Julian Xu‘ (add a your name to GIT)**

**-$ git config --global user.email ‘Julian Xu‘ (add a your email to GIT)**

**-**$ git add . = adds everything

-**$** **git commit** => press I for insert mode => write Inital commit => press ESC => type :wq

=> or use **$ git commit –m ‚changed git.js‘**

**-$ git status** => if there is a **green ‚changes to be commited‘ files** => commit it

If there is a **red ‚untracked files‘** **files** => add the files into the repository (by adding them you add them into the **.git folder**

**File ignoring**

-create a file called only **.gitignore** (is a specific file of files that wont be added to the storage)

-wont be added even after $git add . (everything) => the .gitignore will be added though

**Branches**

-You are a developer who is working on a project with a bunch of people => you are assigned a task of adding a **login system**

-You dont want to start making changes and and saving it inside the repository without it being in the final state

-You should create a **login branch** **(side branch)** and work on that => you can still commit it but not inside the **main branch** **(master branch)**

**-$ git branch login** **(the side branch)** => you created a your **own branch**, which you can manipulate with without modifying the **master branch**

-**$ git checkout login (enter the login branch)** => we added a **new file (login.html)** into the login branch using the command **$ touch login.html** and we added some text **‚login form‘** inside the **index.html (from the master branch)**

-if we use **$ git checkout master (go back to the master branch)** the **login.html** and the **‚login form‘ inside index.html will be gone**

-when we are ready to **merge the changes with the master branch** => in master branch type **$ git merge login (merges the changes)** => you have to add a comment on why you are merging

**Remote repository**

-**$ git remote add origin ‚github link‘** => adds an folder inside the GitHub

-**$ git push -u origin master** => pushes the files inside the folder

-it will require a **login username and password**

-whenever something is changes => you only have to type **$ git push** (pushes it onto the remote repository)

**Downloading and clonning files from GitHub**

-open GIT bash somewhere and type **$ git clone** **‚github link‘** (downloads the files)

-or download the file in a ZIP file

**Pulling**

-if anyone made any changes => use **$ git pull** (you download all the newly changes files)