Git and Git hub is at the bottom and at the middle part.

1.Ls- list, ls-a list hidden files also.

to create file in terminal: type nul > file_name

2.to create json file: npm init -y this -y will automatically creates json file.

3.how to run scripts: if we add any scripts in json file then we can run it using npm script_name

#creating express app using command: npx express-generator , this command will create whole react and all folder hierarchy for our express application.

4.To install nodemon:

Npm I –save –dev nodemon

Then in package.json in script cut it and write :

```
"nodemon": "nodemon file_name.js"
```

Finally in terminal type : npm run nodemon or if did via dev then do npm run dev

```
Or  "dev": "nodemon src/app.js -e, js, hbs"

    //  this is to use nodemon for js and hbs files -e is for extension
```

Put this in scripts in json file and run npm run dev

Node modules: const fs=require("fs") // including module

Const content=fs.readFileSync("file.name", "utf-8", (a, b)=>{

Console.log();}

)

(a, b) not compulsory, a means error and b is content of file:

Lly,

Const content=fs.writeFileSync("file.name", "utf-8");

Blocking vs non blocking(synchrously vs asynchronously)

Blocking line by line execution

Non blockingline by line not guaranteed , function call can hit though execution of program continues.

Backend: it is all about serving files on request by user

Creating server:

First include http

Then,

const server= http.createServer(“req”, “res”, ()=>{
res.writeHead(200, {‘’Content-type’’, “text/html”});

res.end(filename);

listening to server:

server.listen(80, ‘127.0.0.1’, ()=>{

console.log(“anything”)});

and topics like custom module means custom function which we import from other file, postman and express app which are used to manage request and response of different requests without creating custom backend. Read these things from cwh notes in more detail

Packages : files which are written by someone else and which we can import in our code to use it.

A package is a file or directory that is described by a package. json file.

The package is a simple directory having collections of modules.

Modules : A module in JavaScript is just **a file containing related code**.

Node module is a file contain all dependencies

Version : 1:0:0 means major change: minor change: patch means (bug fixing)

npm init will create nodemodules.json file which contains all information that we have answered..

using npm install module_name we can install modules

when we install any new module it will be updated in nodemodules.json file under dependencies

nodemoon package It gives us the server which automatically gets restart every time.

// if we delete the created modules file then after writing npm install all will come back

If we want some modules for only development but not for production anything then we can use syntax : ***npm intall nodemon --save-dev this is for nodemoon example***
https://www.codewithharry.com/videos/web-development-in-hindi-69/

Postman, and express :

It is app which is used to request response of any url

Routes :

Nothing but app.get(“file.html, function());

Here, file.html is routes.

And when we create form that content of action is also our route

Parser:

npm install body-parser □this will install body parser to store data from server.

API: application programming interface :

It is a set of functions and procedures allowing the creation of applications that access the features or data of an operating system, application, or other service.

API stands for **Application Programming Interface**. In the context of APIs, the word Application refers to any software with a distinct function. Interface can be thought of as a contract of service between two applications. This contract defines how the two communicate with each other using requests and responses.

Eg. Amazon is client want to book tickets from indigo airlines, so here indigo is banglow and API is watchman if api of amazon is correct then watchman can allow to interact with bangle

- App api’s does not neccesarily have api key some api’s are free to use
- 1. Endpoint □starting address in api path, which is most general …
- 2.paths □ after endpoint path is there, eg. We have api of jokes and if we want to specify that we want programming jokes,
- 3.parameter □ after path there is parameter, it starts with ? eg. If we want that jokes should contains keyword debugging, then we can specify this like□ ?contains=”debugging” , we can use more than 1 parameters using & operator…
- 4.authentication □

Kanye.rest API

Use this link to learn more about api’s and parameters in api are in the form of key value pairs so you can create api using key and value pairs using postman app : https://home.openweathermap.org/users/sign_up □ here first go to api key and create one key, which you can use while making any API call.

Go to api on site and then select any api and use it, error will occur because u should put your appid there i.e api key that u created. i.e u got while signing up.

Order of parameters are not important.

JSON: javascript object notation it is a format in which we can get our data back

There are other formats like xml and html.

Learn more about json from weather project that I did.

Now status codes…

100-199 ->informational responses

200-299 ->successful responses

300-399 ->client server error responses

400-499->server error responses

Explore more codes like 401 and so on…

HTTP status dogs are also there , which uses dogs to show status codes.

Version control:

Version control, also known as source control, is **the practice of tracking and managing changes to software code**. Version control systems are software tools that help software teams manage changes to source code over time. Using git we can also go to the previous and previous to previous versions of any file or project or app.

Git is a version control system.

Procedure.

Create folder, then create file inside it. Initialize git init for this directory to uninitialised our folder as git folder do → rm -rf .git . Then add our current file to staging area using git add filename command

Check it using git commit command, then do commit this file using git commit –m “msg” use this command . so every time when changes made given msg will be printed.

git status -> is used to see see untracked files.

git log command is used to see current commited files.

Working directory(where git repo initialized)□ staging area □ local repo

Before commiting to file it is in staging area. If we want we can commit it.

Before staging it is in working directory.

Pushing our commit files to remote repository.

Git add . □ to put all files in current directory to staging area.

repository called as git repository.

Procedure:

First we need to create remote using=> git remote add origin https://address (optional ig not sure)

Here origin is a remote name it depends on us to choose name. , this address u will get from github when u create new repository there.

Use git push –u origin master , here master is the main and remote repository which contain all commit files .

or IMP→

just follow these steps first go to desired folder using vs code and do git init then do git add . and then u can check untracked files using git status after do -> git commit -m “msg” then create new repo on github and finally do git push -u origin branch_name, origin is the default branch name we can also change branch name infor related is given above.

Git ignore : it is created by touch .gitignore in our working directory. File name is constant.

git ignore is a file that we should created in our current working directory.

When we do ls –a we can see .DS_store file which is hidden file.

The purpose of gitignore files is **to ensure that certain files not tracked by Git remain untracked**. To stop tracking a file that is currently tracked, use git rm --cached. i.e deleting from our staging area

We can use .filename.extension in gitignore file , or use *.extension to ignore files.

i.e if we performed operations that will not be performed on them.

Note: in github.com/github/gitignore we can see one swift.extension file which we can see ,copy to our git ignore file.

→to delete file from our local repository: git rm –cached filename

→ delete everything from staged: git restore –staged files_name

to delete folder from our local repository : git rm -r –cached folder_name , here -r is for recursively deleting all files in that folder.

PULL Request:

pulling means if someone or we ourself has done some changes on git hub by going to our repository then if we want to accept those changes then we can do pull

pull command is : git pull origin branch_name

Cloning :

Cloning a repository pulls down a full copy of all the repository data that GitHub.com has at that point in time, including all versions of every file and folder for the project.

- Using git clone we can directly download or clone files from git repository using git clone command i.e git clone "url" , we can get url from github from the repo that we want to clone simply by seeing around download or clone button.
- If we want to run anything that we cloned from git hub for it , we should change first bundle identifier name and team name.

Branching and merging :

Branches **allow you to develop features, fix bugs, or safely experiment with new ideas in a contained area of your repository**.

To create branch:

```
git branch <branch name>
```

or git checkout -b branchname

and to switch to that brach do → git checkout branchname

```
git branch
```

to check in branch we are working in, it is represented by *.

to get into and work into that branch `git checkout <branch name>`

let’s say we did some changes in our branch and want to cancel them and restore our previous version then do git restore file_name

Push large files to github:

brew install git-lfs

git lfs install

git lfs track “*.extension_yours”

git lfs push –all origin master

git add .

git commit -m “msg”

git push -u origin master

Merging Branches. Once you've completed work on your branch, it is time to merge it into the main branch. **Merging takes your branch changes and implements them into the main branch**. Depending on the commit history, Git performs merges two ways: fast-forward and three-way merge.

If we are in our main branch command is : git merge “branch_name”.

if any conflict comes vs code provide us facility to choose one of them i.e which change to choose i.e by master brach or by desired merge branch after that again we need to do git add . and git commit to commit all changes.

Git Log:

The git log command shows a list of all the commits made to a repository

EJS : embedded javascript templating, it is a template engine for js, As the name suggests, it lets us embed JavaScript code in a template language that is then used to generate HTML.

First npm install ejs

Then create view folder and create file index.ejs in it to serve our html , main thing is in get method there is function res.render(“file_name”, {var_in_ejs : value_for_var , …..so on)

Now we can use this value in our ejs file to send it on server or to perform operatios on it.

Var's should be written <%=var% > and loops or conditional statements except html tags in <% anything %>

Boss level challenges 4:

Challenge 6: In href(hypertext reference)) we write link so on clicking on this link we go to given address in href. To embed js file we write link in src in script tags.

Challenge 11: to push elements or objects into an array from server each time declare that array globally then arr.push(obj); in post method. so each time we will get updated array.

Challenge 12 :

```
res.render("home", {arrhome:arr});
```

here arr can be array.

Challenge 13:

```
<% for(var i=0 ; i<arrhome.length; i++){ %>
    <% console.log(arrhome[i].title);  %>
 <% } %>
```

Object in the form of key value pair so we can access the title element for object which is in array arrhome like above., here titile was the key by doing above I can get value of it.

Challenge 14:

Above code using forEach loop:

```
<% arrhome.forEach(function(arr){ %>
    <% console.log(arr.title);  %>
 <% }) %>
```

Express routing parameters :

```
app.get("/posts/:postNews", function(req, res){
console.log(req.params.postNews);
})
```

If we hit the url localhost:3000/posts/postany

So in out console postany will be printed, this is achieved by above code, any thing which is after :/ will be printed name after /: and params. Should be same i.e compulsory.

Lodash :

Const _=require(“lodash”);

```
const ans=_.lowerCase(req.params.postNews);                    // this
lowerCase is called lodash , it ignores all '-',  spaces or some uppercase and
some lowercase and consider then in one form  i.e UPPer , uppeR, upp-er all
these will be considered as same
```

challenge: 20 :

truncate string to certain characters , i.e limit length of string

string_name.substring(0, desired_length);

Git Commands in Short :

git init : initialise repository.

git status : see all untracked files.

git add . : add all files to staging area.

git reset –staged . : remove all files from staging area.

git reset –staged file_name.extension.   //instead of reset , restore can also be used.

git rm –cached file_name.ext : remove the file from local repository.

git rm -r –cached folder_name : removes the folder from the local repo.

git branch : see all branches.

git branch branch_name : create branch

git checkout branch_name : switch to branch.

git checkout -b branch_name : create and switch to created branch.

git commit -m "ok" : //this -m is optional and if we don't use it here then after executing this command it will open the default editor ig vim for cmd there we have to enter the msg. means msg is mandatory.

git log : We can see the history of changes/commits in the local repository.

git pull origin branch_name : to accept the changes that has done on remote repository.

git merge source_branch_name : below is detailed description.

git fetch origin branch_name : it updates your local repository with changes from a remote repository without merging those changes into your working directory.

Merge Conflicts :

Git conflicts occur when changes from different branches overlap in ways that Git cannot automatically reconcile. These conflicts require manual resolution, where you decide how to integrate the conflicting changes. By carefully editing the conflicted files, staging them, and committing the resolved changes, you can successfully merge branches despite conflicts.

Diff between merge and pull :

Merge : The `merge` command is used to integrate changes from one branch into another branch.

> **Workflow**:
>
> Switch to the branch where you want to merge changes (
>
> **`git checkout <target-branch>`**).
>
> Execute the merge command (**`git merge <source-branch>`**).
>
> Git tries to integrate the changes from `<source-branch>` into `<target-branch>`.

If there are no conflicts, Git performs a fast-forward merge or creates a merge commit.

If there are conflicts, Git pauses the merge process and asks you to resolve conflicts manually before completing the merge with `git commit`.

Pull :

Purpose: The `pull` command is a combination of two actions: fetching changes from a remote repository and merging them into the current branch. it is used when we want to update our current branch with the latest changes in the remove repository and integrate them automatically.