**PRIMEROS COMANDOS DE GIT**

## **GIT**

$ git init //inicializar el repositorio

$ git add nombre\_de\_archivo.extencion //Agregar el archivo al repositorio

$ git commit -m “Mensaje”// Agregamos los cambios para el repositorio

$ git add .// Agregar los cambios de la carpeta en la que nos encontramos agregar todo

$ git status // visualizar cambios

$ git log nombre\_de\_archivos.extencion //historico de cambios con detalles

$ git push //envia a otro repositorio remoto lo que estamos haciendo

$ git pull //traer repositorio remoto

$ ls //listado de carpetas en donde me encuentro es decir dir en windows

$ pwd //ubicacion actual

$ mkdir //make directori nueva carpeta

$ touch archivo.extencion//crear archivo vacio

$ cat archivo.extencion//muestra el contenido del archivo

$ history //historial de comandos utilizados durante esa sesion

$ rm archivo.extencion //eliminacion de archivo

$ comando --help //ayuda sobre el comando

$ checkout //traer cambios realizado (tmb hace q tu código sea como la version q le indicas)

$ git rm --cached archivo.extencion//se usa para devolver el archivo que se tiene en ram cuando escribimos git add lo devuleve a estado natural mientra esta en staging

$ git config --list //muestra la lista de configuracion de git

$ git config --list --show-origin//rutas de acceso a la configuración de git

$ git log archivo.extencion //muestra la historia del archivo

$ git config --global user.email "micorreo@ejm.com" // para cambiar el correo de git y asi decir q tal persona está haciendo las demas versiones

$ eval $(ssh-agent -s) //con esto es para ver si el conector de las llaves entre tu compu y tu github están corriendo para q pueda funcionar

$ ssh-add ~/.ssh/id\_rsa // aqui estamos agregando la llave privada

**OTROS COMANDOS**

ctrl + l para limpiar

$ git checkout master : para regresar a la rama principal, el ultimo forma del proyecto (obs tener cuidado ya q es el master si lo cambias cambias todo el proyecto)

$ git branch //te muestra las ramas que tienes

$ git branch \_(nombre de nueva rama)\_\_ // se crea una nueva rama al poner un nombre al lado (ojo que cuando se usa el git branch te saldrá uno con un asteristco y resaltado de verda, esto quiere decir q estás en dicha rama)

$ git checkout \_(nombre de rama)\_ // te direcciona a la rama que llamaste

$ git merge \_(rama): // te trae el conjunto de commit de la ramma

$ git reset \_\_(commit N° de version)\_ --(modo)

este modo puede ser

hard: regresa a dicha version por completo

solft: regresa a dicha version pero lo que está en stagin sigue en stagin, si hiciste cambios y diste git add eso sigue allí

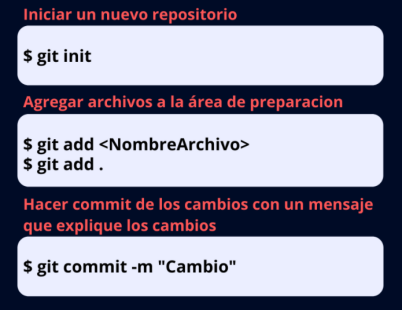
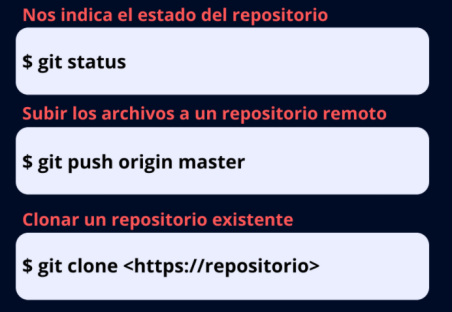
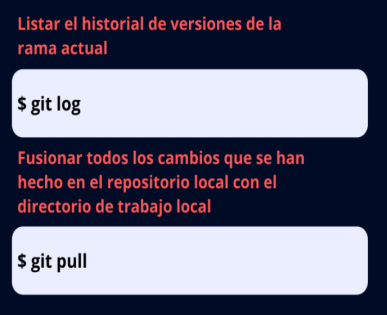
$ git branch //crear una rama o muestra la rama donde uno está

$ git checkout \_(nombre de rama)\_ // ahora lo q hagas está en la rama ya no en el master

$ git commit -am "mensaje"; aqui usamos "a" para agregar los cambios q hemos hecho y "m" par aagregar un mensaje

MERGE: lo que hace es juntar el último versión de tus dos clases q tienes

una será la principal q se llama "master" y la otra le pones un nombre para el caso se llamará "cabecera". Entonces lo que hace es unir las versiones finales de master con el de cabecera .

COMANDOS DE GIT CON GITHUB

git log --oneline - Te muestra el id commit y el título del commit.

git log --decorate- Te muestra donde se encuentra el head point en el log.

git log --stat - Explica el número de líneas que se cambiaron brevemente.

git log -p- Explica el número de líneas que se cambiaron y te muestra que se cambió en el contenido.

git shortlog - Indica que commits ha realizado un usuario, mostrando el usuario y el titulo de sus commits.

git log --graph --oneline --decorate y

git log --pretty=format:"%cn hizo un commit %h el dia %cd" - Muestra mensajes personalizados de los commits.

git log -3 - Limitamos el número de commits.

git log --after=“2018-1-2” ,

git log --after=“today” y

git log --after=“2018-1-2” --before=“today” - Commits para localizar por fechas.

git log --author=“Name Author” - Commits realizados por autor que cumplan exactamente con el nombre.

git log --grep=“INVIE” - Busca los commits que cumplan tal cual está escrito entre las comillas.

git log --grep=“INVIE” –i- Busca los commits que cumplan sin importar mayúsculas o minúsculas.

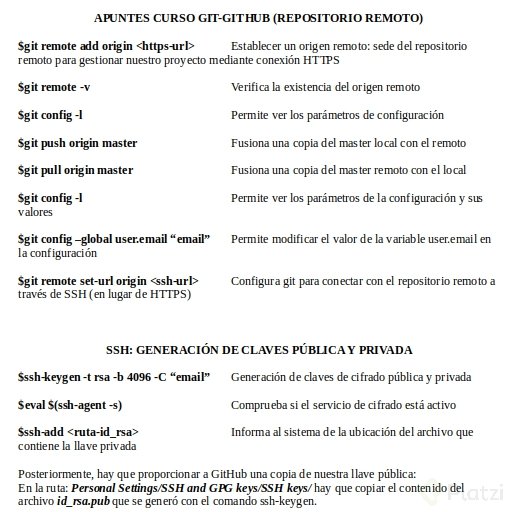
git log – index.html- Busca los commits en un archivo en específico.

git log -S “Por contenido”- Buscar los commits con el contenido dentro del archivo.

git log > log.txt - guardar los logs en un archivo txt

git pull origin master --allow-unrelated-histories //permite fusionar lo que hay en virutal con lo rama local

en lugar de usar "ymaster" usar "main"



**CSS GRID LAYOUT**

ENLACES DE APOYO

Les comparto unas guías:

* [https://css-tricks.com/snippets/css/complete-guide-grid/](https://css-tricks.com/snippets/css/complete-guide-grid/" \t "https://platzi.com/clases/2222-css-grid-layout/36026-que-significa-grid-para-css/_blank)
* [https://cssreference.io/](https://cssreference.io/" \t "https://platzi.com/clases/2222-css-grid-layout/36026-que-significa-grid-para-css/_blank)
* [https://learncssgrid.com/](https://learncssgrid.com/" \t "https://platzi.com/clases/2222-css-grid-layout/36026-que-significa-grid-para-css/_blank)
* [https://css3generator.com/](https://css3generator.com/" \t "https://platzi.com/clases/2222-css-grid-layout/36026-que-significa-grid-para-css/_blank)
* [https://cssgridgarden.com/#es](https://cssgridgarden.com/" \l "es" \t "https://platzi.com/clases/2222-css-grid-layout/36026-que-significa-grid-para-css/_blank)

COMANDOS

**Text-align : left , right, center**

**Vertical-align: top, middle, bottom**

With this you can organize your text box in a big box, for example, guest that we have a big blue box and there is an little box inside, so how can we organize it, using this comands, text-align use to move la little box in left, center or right and vertical-align is to put it the top, middle or bottom

In other words, text-align is to organize horizontal way and vertical, just vertical :v

Justify : this one mean that the movement is horizontal (starts from la left)

Align :this mean that the movement is vertical (starts from the top)

**ITEMS : IT WORKS FOR THE CONTENT OF THE BLOCKS (INSIDE)**

Align-items

Justify-items is the same but here is for what is inside, text, images, etc

Place-items

**CONTENT: IT WORKS FOR WHAT IS AROUND THE CONTAINER**

**align-content: star/ end/ center/ stretch**

--> with this comand we can ubicate in vertical way la content of a “main” o whatever

If is start the content stay upsaide

If is end the content star down

If is center the content keeps in the middle

If is stretch the content adaptate the form of the father box

Also there are we have:

Align-content: space-around / space-between / space-evenly

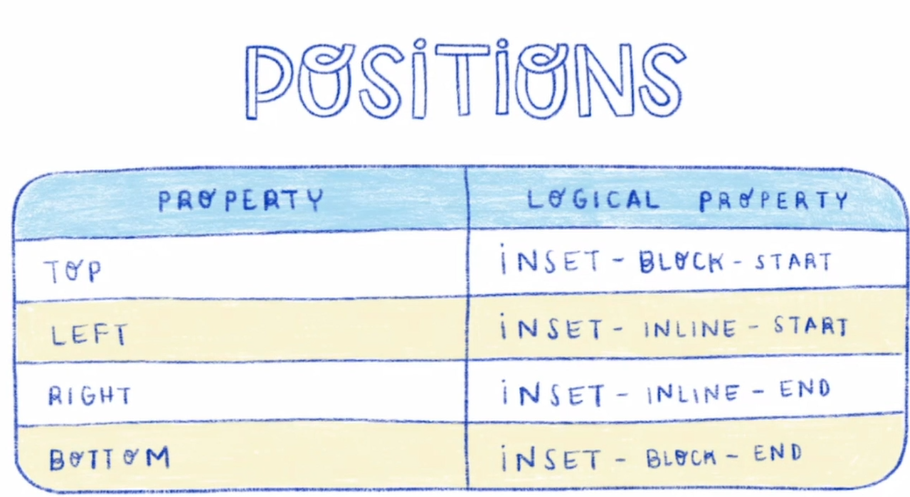
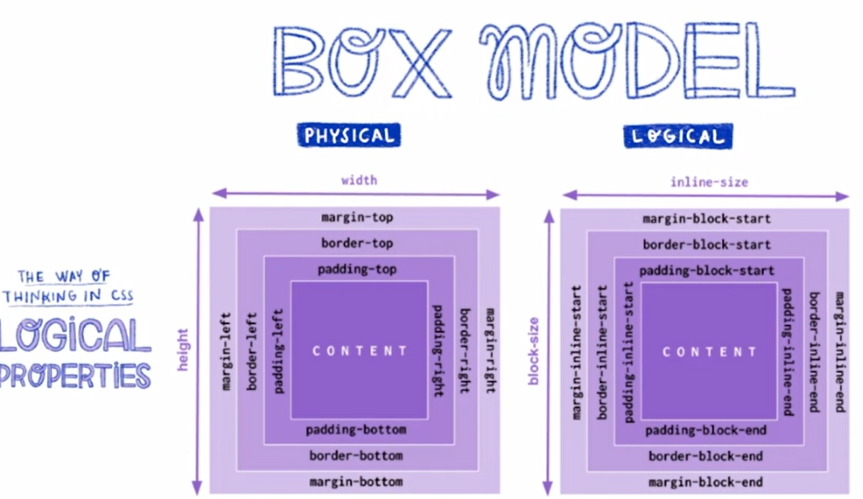
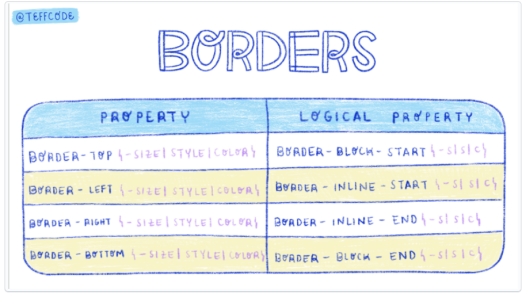
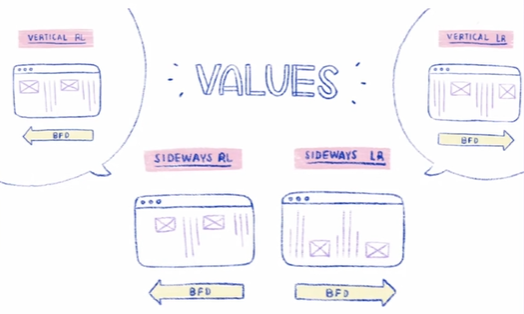
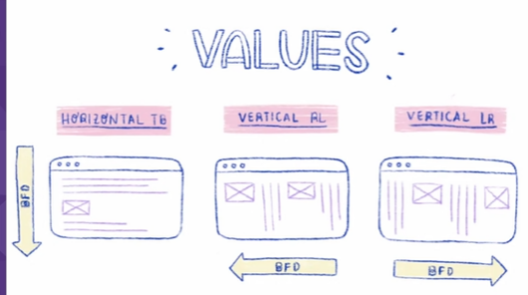
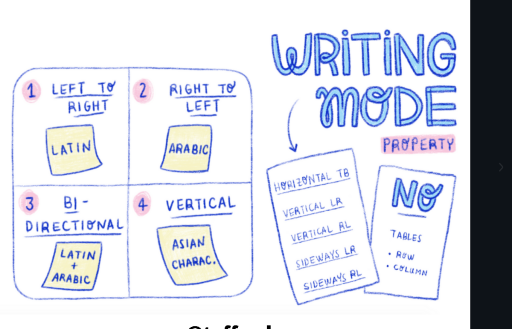
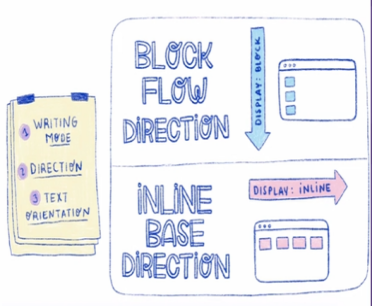
Space-around: all around of each one block appears a space

Space-between: there is a space, extreme to extreme each block stay

Space-evenly: there is a space pretty similar to around but in this case, this spaces is correctly distributed

**Justify-content:**

**Place-content**



**Notes**

