Clase 1

Configuración del Git SCM.

PS E:\Facultad\Curso desarrollo web fullstack\Servidores Node JS 03> git config --global user.mail "hermsale@gmail.com"

PS E:\Facultad\Curso desarrollo web fullstack\Servidores Node JS 03> git config user.email

hermsale@gmail.com

PS E:\Facultad\Curso desarrollo web fullstack\Servidores Node JS 03> git config --global user.password H9i4rcjTZ4qTrK5

PS E:\Facultad\Curso desarrollo web fullstack\Servidores Node JS 03> git config user.name hermsale

fatal: not in a git directory

PS E:\Facultad\Curso desarrollo web fullstack\Servidores Node JS 03> git config user.name

Hermsale

Nombre, mail y contraseña. Lo minimo de configuración.

Crear repositorio:

GIT init

Las cosas que vas colocando nuevas, aparecen con una U. esto significa que esta en estado **Untracked**

Para pasar a estado ADD. Usamos el comando: git add “archivo” . para un solo archivo.

Git ADD . 🡨 para todos los archivos

Realizamos el commit para subirlo al github: Git commit –m “commit inicial”

Inicio:

Git init – git add – git commit

Si modificamos usamos: git add – git commit

Cargar o crear un nuevo archivo en github

Vamos a la pagina de git hub y creamos el repositorio. Una vez creado, el Github nos va a dar los códigos para poner en nuestro código VSC Terminal.

Ejemplo:

git remote add origin https://github.com/hermsale/clase1Servidores.git

git branch -M main

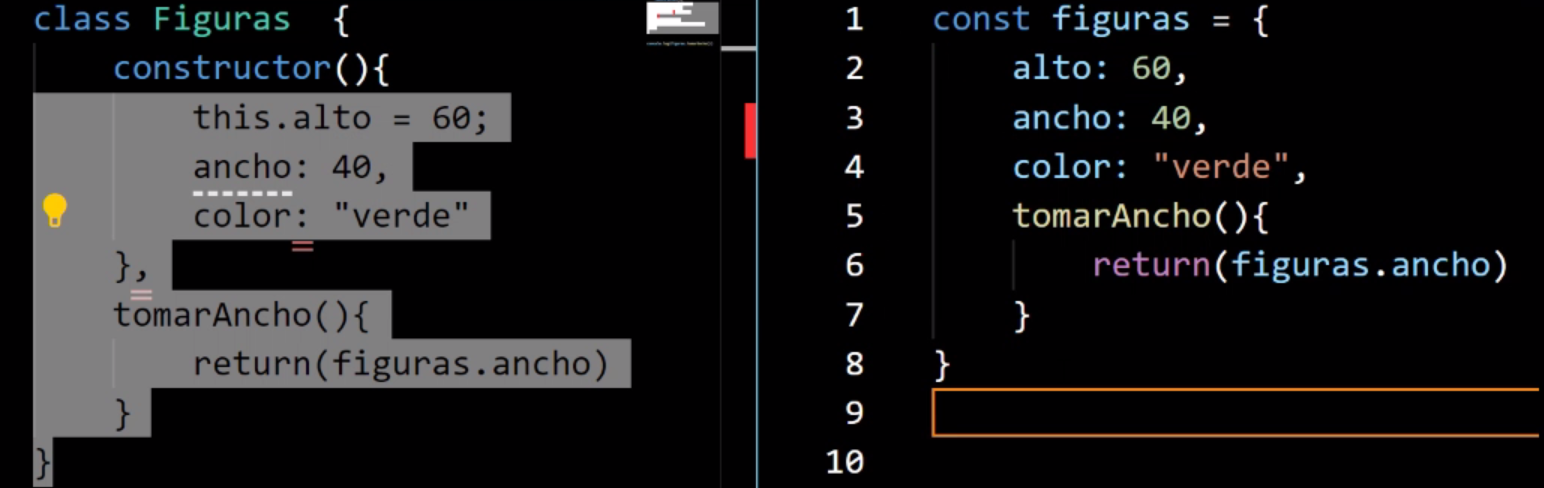
git push -u origin main

crear archivo readme.md.

este archive es la carta de presentación, es donde vamos a escribir las cosas que hicimos en el proyecto, quienes somos.

Programacion OO

Paso de objetos a clases.



Las Clases CLASS, por buenas practicas se ponen primera letra en mayúscula.

El método constructor, toda clase requiere de un constructor.

Esto nos va a permitir instanciar objetos.

Constructor(){

This.alto = 60; // se usa this porque es del mismo objeto. Se usa ; y =

}

Las funciones siempre pueden retornar algo.

El método constructor se limita solo a los atributos.

Los métodos van por fuera del constructor.

GET resolver lógicas y pasar datos: get duplicarAncho(){ }

En el constructor pasamos los parámetros.

Constructor (alto, ancho, color){

This.alto = alto;

This.ancho = ancho;

}

Se les puede pasar un valor por defecto cuando no toma un valor de la instancia (el famoso NaN).

Constructor (alto=0, ancho=0, color=”negro”){

This.alto = alto;

This.ancho = ancho;

}

# Instanciamiento de Objetos

// tenemos que instanciar al objeto para usarlo

const figuras = new Figuras(60,40,"azul");

// una vez que se instancea, se pide cosas

function tomarAncho(){

    console.log(figuras.ancho);

}

console.log(figuras.tomarAncho());

console.log(figuras.tomarColor());

/////// creo otra instancia de Figuras

const rectangulo = new Figuras(10,50,"verde");

console.log(rectangulo.tomarAncho());

console.log(rectangulo.tomarColor());

Figuras sería el molde de objeto, y figuras o rectángulo es una instancia de Figuras.

A los métodos se les puede pasar parámetros

Tomarancho (base);

# Herencia

Las herencias se producen cuando tenemos un objeto, con parámetros similares a otro, pero donde cambia algunas cosas.

Creamos un objeto Class.

Super() es pasar una referencia, de la clase que esta siendo heredada. En el caso de Figuras, trae las propiedades de ancho, alto y color.