local repo

un tracked files tracked files

insert git add filename

update or

delete git add .

git commit -m "message"

to send files to GitHub

->folder in GitHub(repository)

->git remote add origin http-path

->git push -u origin main/master

(u-> unstream) byte by byte

step 1: create account in git-hub

step 2: create repo in git-hub

https://github.com/soumyareddygurrala/FSD.git

step 3: open vs code and open terminal in vs code

step 4: initialize git in terminal

git init

step 5: configure user-name and email to GitHub

git config --global user.name "soumyareddygurrala"

git config --global user.email “[soumyareddygurrala214@gmail.com](mailto:soumyareddygurrala214@gmail.com)”

step 6: add remote url to local repo

git remote add origin <https://github.com/soumyareddygurrala/FSD.git>

convert untracked files to tracked files

git add .

git commit -m “my new project”

git pull origin “branch”

to push to github

git push -u origin master

merging more than one branch

create a branch:

* git branch branch-name
* to check whether the branch is created or not : **git branch**
* **git checkout btanch-name** shifted from one branch to another branch
* **git add .**
* **git commit -m “”**
* **git push -u origin branch-name(dev1)**

while merging the branch always be in master branch 🡪console 🡪object

git checkout branch-name (master) 🡪log 🡪method

command for merging 🡪var global scope

git merging branch-name (dev1) 🡪let,const script or local scope

java script

What is java script?

It is a object based programming language(we consider entire thing as a single object) used to convert static page to dynamic page

web pages with html , css is static page (no funtionality) to convert static to dynamic java script is used.

Java script is used in two ways : 1)internal and 2)external

Inside the body tag if we take script tag and inside script tag if we write logic then it is internal type.

External js -> we create separate file with .js extention and link it to the html file with the help of script tag

**Datatypes in js:**

It is a value used in a programming language

**Primitive datatypes**:

Number, Sring ,Boolean, Undefined, Null, Big – int

**Non primitive:**

Class , object, array, funtion, map, sets

Scopes:

>gloval scope

Local scope/script scope

Block scope

3 – types

Var, let, const

Ternary 3 operands : codition , true statement, False statement

Operators

It is a predefined symbol used to perform specific operation

Arithmetic operator : +,-,\*,/,%,++,--

Assignment operator : +=,-=,\*=,/=,%= (short-hand)

Ex😊:let m =10;

Console.log( m = m+10);

Console.log(m+=10);

Increment:

Post increment(first value will be assigned and then we will increment the value)

Ex:

let a=10;

console.log(a);//10

console.log(a++);//10

console.log(a++);//11

pre increment :

let b=20;

console.log(b);//20

console.log(++b);//21

Decrement:

Pre decrement :

let c=10;

console.log(d);//10

console.log(--d);//9

post decrement:

let c=10;

console.log(c);//10

console.log(c--);//10

console.log(c--);//9

Logical operator :

AND (&&)

OR (||)

NOT (!)

Relational operator : <, <=, >, >=, ==, ===, != ( **difference b/w ==(equals to) and ===(strictly equals to))**

== checks values not datatypes

Console.log(5 == 5); //true

Console.log(5 == “5”); //true

=== checks values and datatypes

Console.log(5 === 5); //true

Console.log(5 === “5”); //false

Ternary operator :

(condition) ? true – statement : False-statement

Conditional statements : simple – if, if else, elseif ladder

If(5<20)

{

Console.log(“Hello World”);

}

if else

If (5<10 ){

Console.log(“Hello”);

} else {

Console.log(“bye”);

}

Else if ladder

Let a =90;

If (a>=90 && a<100){

Console.log(“a+”);

}Else if (a>=80 && a<=90){

Console.log(“a”);

}else if (a>=70 && a<=80){

Console.log(“b+”);

} else if (a>=60 && a<=70){

Console.log(“b”);

}else{

Console.log(“c”);

}

Switch condition :

Let n = 1;

Switch(n){

Case 1:

{

Console.log(“Sunday”);

}

Break;

Case 2:

{

Console.log(“Monday”);

}

Break;

Case 3:

{

Console.log(“tuesday”);

}

Break;

Default {

Console.log(“invalid”);

}

Looping statement : while, do while, for, for-each

For loop :-

Syntax :=

For(initialization;condition;increment/decrement){

Statement

}

Eg :

For(let 1=0;i<=5;i++){

Console.log(i);

}

While loop : it will iterate the statement until the condition becomes the false

While (condition){

Statement

}

Let n =1;

While(n<=5){

Console.log(n);

N++;

}

Do while loop : it iterates set of instructions and then it will checks the condition

Do{

statement

}while(condition)

Let m = 1;

Do{

Console.log(m)

M++

}while(m<=5)

Functions in javascript

Function are 1st citizens in javascript

Functions are objects in javascript

Types of functions in javascript

1. Function declaration statement or pure fuction or named function
2. Function expression
3. Higher order function , callback function
4. Arrow function
5. Nested function (closure ans lexical scope)
6. IIFE (Immediate Invoking Function Expression)

**Function Decleration Statement :**

**🡪**set of instruction or block of code used to perform specific task

Syntax :

Function function-name(parameters,-----)

{

}

Function-name(arguments,---------)

Return keyword

->Return keyword will stops the execution of the function and controll will be given to caller(programmer)

🡪hoisting is possible(we can call the function before the function declaration)

Types of errors in js :

1. Syntax error
2. Refernce error
3. Range error
4. Uncought error

Advantages of Function :

1. Code reusability (write once call multiple times)

The prompt is present in the window window.prompt

Implict type casting : converting of one datatype to another type by javascript engine

explict type casting : converting of one datatype to another type by user(programmer)

**Function expression :**

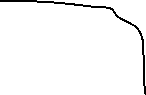
Assigning function as a value to one variable is called function expression

It is used to perform specific task

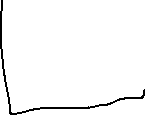
**A function without a name is called anonymous function**

**function ()**

**{ 🡪treat it as a variable**



**console.log(“hello”)**



**}**

**Hoisting is not possible in function expression**

**Arguments object contains all the passed to the function in array format**

**Function programmimg :**

**Hidher order function and callback function**

It is used to perform **generic task** (multiple task)

It can be created using function declaration statement or function expression or arrow function

**Higher Order Function(HOF)**

A Function which accepts another function as a argumnet is called as a Higher Order function

**Callback function 🡪callback function should be anonymous function**

A function which passed as an argument to Higher Order Function is called as callback function

**Arrow function :**

It was introduced in ES6 version in 2015 , ES🡪ECMA – SCRIPT (European Computer Manufacturer Assosiation) organizarion by netscape comapny

Live script ,mocha, ECMA script

It is used to redyce the number of lines in a code.

Two types of return

* Implicit return arrow function
* Explicit return arrow function

Implicit

* No need of using return keyword

Explicit

* Return is mandatory
* Block ic mandatory {}

Nested function:

Function parent()

{

Function child()

{

Console.log(“iam child fun”);

}

Child()

}

Parent()

Eg: lexical scope/scope chains:

The ability of js engine search for variable in local scope if not available it will()

Start search in global scope

var a = 10

let b = 4

function parent1()

{

var a = "hello"

let b = "hii"

console.log(a);

console.log(b);

console.log(this.a);

}

parent1()

eg:

function main()

{

let a = 60;

function child()

{

console.log(a);

}

child()

}

main()

closure: the binding of child function to parent function is called as a closure /or/

the binding of child function to the lexical function to the parent function is called as closure

Eg:

function parent2()

{

let a = 10;

function child()

{

console.log(a);

}

return child

}

parent2()()

when ever we called child fun multiple times multiple, times closure will be created in epria it leads to the memory wastage

eg:

function parent2()

{

function child1()

{

console.log("i am child1");

}

function child2()

{

console.log("i am child2");

}

return [child1,child2]

}

parent2()[0]()

parent2()[1]()