**TRAINING**

***2/1/25***

* Functions
* DOM
* Events
* #mini projects
* Major projects
* Html, css, js, React js
* Java, Spring boot, My sql
* Web applications
* Front-end
* Database, java

Git – it is a version controller , it is used to keep track on code

Git-hub – used to store and manage code (cloud storage)

Git-bash –

***3/1/25***

Git is divided into to parts – 1) local repo and cloud

Local repo is divided into to parts – 1) un-tracked files 2) tracked files

Untracked files - create ,update , delete (these will be in local repo).

Command to convert untracked to Tracked files – git add file-name or git add . and git commit -m

Git remote add origin http-path (origin = github) --- It to create

Git push-u origin master/main (u = upstream which means sending byte by byte)

***6/1/25***

**Variables:** are used to store data

1. Static typed and dynamic type:

Case sensitive

Start with letters , \_ , $

Cant start with numbers

Reserved words are not allowed

**Declaration of variables:**

1. Var
2. Let
3. Const

***7/1/25***

Datatypes:

1)Dynamic type language

Two types:

1)Primitive data type

2)Non – primitive data types: Objects, Arrays

**To check the data type -TypeOf:**

**Primitive data type**: Predefined; we can only store the single values

**1)Numbers:** integers, floats

**2) String:** stream of characters enclosed in quotes --- single, double, backtick quotes. Backtick introduced in ES6 version

**3)Boolean:** true or false.

**4)Null:** empty value or no value.

**5) Undefined:** declared variable with no value.

6) **condition:** An expression that evaluates the result (true or false)

7)**conditional Statements:** to make a set of instructions that execute when the condition is true.

**Indentation:** {

//block of code

}

If : only one condition is possible.

***8/1/25***

***JavaScript operators:***

**1)Arithematic operators : + , - , \*,**

Divison - / (Q)

Modulus - % (R)

Exponential - \*\*

**2) Logical operators: && , ||**

**3) Relational Operators: > ,< , <= , >= , == (it checks value), === (it checks datatype) ,!== , !===**

**4) Conditional operator ; Ternary operator(?)**

**Window methods:**

**Prompt:** is used to take input from user. And the default datatype is string

***9/10/25***

**Conditional statements:**

To make the set of instructions execute only when the condition is true.

**Block of code:** set of instructions.

It will execute when the specific condition is true.

Condition: An expression that evaluates a result (True or False)

Ex: Console.log (5>6)

Switch :

Syntax :

Switch(variable/expression){

Case value 1 :

//block of code for case 1

Break;

Case value 2:

//block of code for case 1

Break;

Default:

//body of default

}

No of students = 10

I student = 9

10 \* 9 = 90

***20/01/25***

Loops: Block of code several tyms

1. For loop
2. For in
3. For of
4. While
5. Do while

For loop:

Syntax: for(initialization; condition ; update—expression){

//block of code

}

***21/01/25***

For in loop

Const a = ‘ramya’

For(let I in a){

//block of code

}

While loop

We use while loop when we doesn’t know the no of iterations in advance

Syntax of while loop

While(condition){

//block of code

}

1. Cond inside the ()
2. {}
3. False

Do while syntax:

Do{

//block of code

}while(condition)

Break statement: if it matches with the condition it will stop executing.

Continue statement: if it matches with the condition it will skip that step and execute the remaining program

Nested loops: A loop inside the another loop

The inner loop will execute one time for each iteration of outer loop

An inner loop within the repeating block of outer loop

***23/1/25***

**Functions:**

A function is reusable block of code

It can be called anywhere in the program

You can use the same code with different arguments to get the results

JS:

Keyword = Function

() parameters are the values(variables) which are declared inside the () while defining the function

Arguments: values which are passed through the parameters

**Anonymous function:**

A function without name , After we create a function without name and we assign it to a variable.

Example:

Sum = function(n1,n2){

Let r = n1+n2;

Return r

}

Console.log(sum(8,9))

***24/01/25***

***Arrow functions:***

Es6 version

More readable and more structure

Anonymous functions(lambda functions)

Without function name but they are assigned to a variable

***Synchronous functions and Asynchronous functions***

synchronous programming: operations are performed one after the other, in sequence. So, basically each line of code waits for the previous one to finish before proceeding to the next

Asynchronous functions :Asynchronous programming, on the other hand, allows multiple tasks to run independently of each other.

Promise:

Eventual completion

Three states

1)pending : still not completed---continuing

2)fulfilled == task completed

3)Rejected == The operation is failed

***25/01/25***

***Callbacks:***

A call back function is a function that is passed argument to another function;

Execution: The function receiving the callback will execute the callback at some point during execution (often at the end or aften an asynchronous operation is done)

Example:

function mainFunction (callback){

            console.log("Executing main function: ");

            callback()

         }

         function callbackFunction(){

            console.log("Callback function executed");

         }

**Explanation:**

* mainFunction(callbackFunction);
* mainfunction takes a parameter callback
* callbackFunction is passed as an argument to mainfunction inside the mainfunction, the callback function

**output:**

* mainfunction(callbackFunction);
* it will execute the code inside the mainfunction
* it will print (Executing main function:)
* then it reaches to callback() where it calls the function which is passed to the mainfunction(callbackFunction)

React js:

Create a element :

React.createElement(‘h1’)

Syntax:

React.ceateElement(type,prop)

Type = tag name(div, h1, p)

Props = className, id , onClick

--to display the element

---ReactDom.createRoot()

--ReactDom.render() –to display the element

Syntax:

ReactElement – what to render

Container = where to render

Introduction to JSX

It is the javascript XML is sysntax extension for js in react js ; it allows us to write html code in react

Make us easier to write Html in react

JSX: JSX code gets compiled into JS

Babel : a tool converts translates into JS

With JSX:

<div id ="app"></div>

    <script type ="text/babel">

        const element = <h1 className ="a"> Hello JSX! </h1>

        const root = ReactDOM.createRoot(document.getElementById('app'));

        root.render(element);

    </script>

Expressions in JSX:

With jsx you can write the expressions inside the { }

This allows us to embed javascript expressions directly ;