**JavaScript Notes:**

**Array:**

An Array is a type of data structure that allow to store data of same types.

* **Array Method:**
  + Push > add element from end of array
  + Shift > add element from start of array
  + Pop > delete element from end of array
  + unshift > delete element from start of array
  + splice > use to delete element and add oter value on that index.
    - Eg: array.splice(3,1,”newvale”)

**Example:**

1. Let arr = [1,2,3,4,5,6];

Console.log(arr);

1. **Change the element of the arr:**

Let arr = [1,2,3,4,5,6];

Arr[3]=10;

Console.log(arr); //output [1,2,3,10,5,6]

1. **Now to change last element by other method:**

Let arr = [1,2,3,4,5,6];

Arr[arr.length-1]=10;

Console.log(arr); //output [1,2,3,4,5,10]

**Array Methods:**

Combination of data structure and fuction is known as method.

**Push() method**:

Push method is used to add element at the end of array.

**Example:**

let rows = ["Naomi", "Quincy", "CamperChan"];

rows.push("freeCodeCamp");

console.log(rows);

**Output:**

[ 'Naomi', 'Quincy', 'CamperChan', 'freeCodeCamp' ]

**Pop() Method:**

It removes the last element from an array and returns that element.

**Example:**

let rows = ["Naomi", "Quincy", "CamperChan"];

console.log(rows);

let popped = rows.pop();

console.log(popped);

Output: Camperchan

**ForEach Method / Function:**

Is use to execute each element of array.

**Callback function:**

A callback function function passes as an argument to other function.

e.g;

let arr =[1,2,3,4,5,6,7]

arr.forEach((val)=>{

console.log(val);

});

Syns in JS:

**Synchronous:**

Synchronous means the code runs in a particular sequence of instructions given in the program Each instruction waits for the privious instruction to the computer its execution.

**Asychronous:**

Asychronous programing, sometime imp instruction get blocked due to some previous instruction which cause a delay in the UI.Asynchronous code execution allow to execute next instruction immediately and doesn't block the flow.

SerTimeout ():

example;

setTimeout( ( )=>{

console.log("Hello");

},4000); // hellow print after 4 second of execution.

nesting programming:

let age =19;

if (age>=18){

if(age>=60){

console.log("senior");

}

else {

console.log("middle");

}

}

else{

console.log("child");

}

**Callback hell:**

is a problem in calback function in javascript .

Nested callbacks stacked below one another forming a pyramid structure.

This style of programming become difficult to understand & manage.

Example:

function getData ( dataId , getNextData){

setTimeout(( )=>{

console.log("Data",dataId);

if(getNextData){

getNextData();

}

}.2000);

}

getData(1,( )=>{

getData(2,( )=>{

getData(3,( )=>{

getData(4);

});

});

});

**Promises:**

to overcome the problem that is callback hell javascript introduce promises.it is an object in javaScript.

resolve

reject

resolve and reject are acallbacks automatically provided by JS.

**Async-Await:**

to overcome thr problem of promises chain we use Async-Await.

Await pauses the execution of its surrounding async function until the promise isa settled.

await is use in async funter otherwise it will through error.

Example:

function api(){

return new promise((resolve,reject)=>{

setTimeout(()=>{

console.log("weather data");

resolve(200);

},2000);

});

}

async function getWeatherData(){

await api();

}

**Fetch API(Application Programming Interface):**

The Fetch API porvides an interface for fetching (sending/recieving) resources.

it uses request and Response objects.

The Fetch method is used to fetch a resources (data).

**Example:**

const URL = "https://cat-fetch.herokuapp.com/facts";

const factpara = document.querySelector("#fact")

const getFacts = async ( ) =>{

console.log("getting data...");

let response = await fetch(URL);

console.log(response); //JSON format

let data = await response.json( );

factpara.innerText = data[0].text

};

**Map:**

Create a new array with result of some operations.The value its callback return are used to form new array

Map and forEach are similar to each other but the difference is:

Map return a out in new array

forEach perform operation on aray.

Example:

let arr =[1,2,3,4,5,6,7]

let newarr = arr.forEach((val)=>{

console.log(val);

});

**Filter:**

Create a new array that give a true value for a condition.

Example

let arr = [1,2,3,4,5,6,7]

let newarr = arr.filter((val)=>{

return val % 2 === 0;

});

**Find:**

Create a new array and return one value that is targeted.alway find first value return it if we have same values.

**Example:**

let arr =[1,2,3,4,5];

let newarr = arr.find((val)=>{

if(val ===4){

return val;

}

});

console.log(newarr);

**Reduce:**

Perform some operation and reduce the array to a single value.It return the single value.

**Example:**

let arr = [1,2,3,4,5,6,7,8,9]

const Output = arr.reduce((Curval , result)=>{

    return Curval + result/2;

});

alert(Output);

**Object:**

Collection of different variables.

**e.g:**

const Student ={

Name : “Uzair Khan”,

Roll No : 302-20147,

Cgpa : 3.26,

Ispass: true,

};

Console.log(Student.age);

To change the value of the object.

Student[“age”] = Student[“age”] + 1 ;

We can change the const variable value because it will throght the error. but we can change the key of const object.

**Loops:**

When you have to perform a task repeatedly until a condition is met, you will use a loop. There are many ways to write a loop.

**For Loop:**

**Example:**

const count = 8;

for (let i = 0; i < count; i = i + 1) {

console.log(i)

}

Example 2:

Push i in a rows array.

const character = "#";

const count = 8;

const rows = [];

for (let i = 0; i < count; i = i + 1) {

  rows.push(i);

}

**For of loop:**

would add the string  to the existing string stored in the  variable. This is called concatenation.

Example to create pyramid:

const character = "#";

const count = 8;

const rows = [];

for (let i = 0; i < count; i = i + 1) {

  rows.push(character.repeat(i + 1))

}

let result = ""

for (const row of rows) {

  result = result + "\n" + row;

//Concatenate code // \n is for new line

}

console.log(result);

**Conditional Statement:**

* If
* Else if
* Else

Example:

let age = prompt("Enter your age");

if (age >= 18 ) {

    console.log("Eligible for licence");

}

else {

    console.log("Not Eligible for licence");

}

//Example 2

let num = prompt("Enter your num");

if (num % 2 === 0 ) {

    console.log("Even number");

}

else {

    console.log("Odd Number");

}

//Example 3

let age = prompt("enter your age");

let result = age >= 18 ? "adult" : "not adult";

console.log(result);

**Function:**

A function is a block of code that can be reused throughout your application.

In javaScript Functions are basically object.

Function Syntax:

**function name(fullname) {**

**console.log(fullname);**

**}**

**name(“uzair”);**

The function keyword tells JavaScript that the name variable is going to be a function. parameter is a variable that represents a value that is passed into the function when it is used. A function may have as many, or as few, parameters as you'd like. Like a for loop, the space between the curly braces is the function body.

**Parameter/arguments:**

* Parameter and argument both r same.
* Any input to the fuction is pass as parameter.
* Parameter is pass in function definition.
* Parameter are like local variables of function which mean they are block scope.

**Argument:**

* Argument is passes in fuction call

**Example:**

Function sum(x,y) { //x and y is a parameter

S = x + y;

Return s;

}

Let val = sum(5,4); // 5 and 4 is an argument

Console.log(val);

**Return:**

An important thing to know about the return **keyword** is that it does not just define a value to be returned from your function, it also stops the execution of your code inside a function or a block statement. This means any code after a return statement will not run.

Return keyword return only one value.

The single wale may be an array.

**Arrow Function:**

Compact way to write the function.

e.g:

const sum = (x,y)=>{

return x + y

}

Sum(2,4);

**DOM:**

When a web page is loaded. The browser create a Dom of page

The follow is dom;

<!-- This Structure is called DOM (Document object Model) -->

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script src="practice.js">

    </script>

</body>

</html>

* **Inspect:**
  + Element(conatin all the html code)
  + Console(is a browser)
    - Window Object:

The window object represent the open window in a browser. It is browser object not (javascript object) and created by the browser automatically.

It is aglobal object with alots of methods and property.

**Event in js :**

The change in the State of an object is Known as an event.

These can Arise from user interactions such as using mouse or resizing a window.

some inline event are listed below:

**1) Onclick**

**e.g:**

<button onclick="alert('i`m Clicked...!')"> click me</button>

<button onmouseover="alert('i`m Clicked...!')"> click me</button>

Event handling in java script:

if we handle event in inline handle and javascript handle the priority is given to the javascript event handle.

**Event Object:**

it is the special object that has details about the event.

all event handlers have acess to the event objects property and method.

event Listeners:

event listner is better that inline handling and javaScript handling.

**back end:**

//for json file

npm init

**In JSON File:**

under main write

"type":"module",

in script remove test and write

"start": "node app.js",

"dev": "nodemon app.js"

**install**

npm i express cors dotenv