Introduction:

Javascript: It is used to provide the functionality

Javascript is object based Scripting language primarily used for web Development

It enables dynamic content on websites and allowing interactive features like animations, form validation and content updates without reloading the web page.

Javascript code runs in the browser (with the help of Js Engine in every browser) and also in Server side (using Node.js)

To execute Javascript code in the browser side mandatorily we should create HTML file and JS file we should make the connection between HTML and JS file with the help of <script> tag and 'src' attribute and in src attribute we should pass the JS file path.

To execute Javascript code in the Server side we can pass the comment Node_filename.Js

Specifications of JS:

1. Object based Scripting language

In Javascript there is no need of creating the class in order to create the object we can directly create the object and we can store the properties and methods

In JS Engine there is no compiler only interpreter is present that will execute the code line by line and left to right.

2. Dynamic Language :-

In JS we can dynamically change the datatype of the variable

Eg.

```
let a = 10;
    a = 'str';
```

In the above example for the var 'a' we assign the value with 'str' datatype

3. Synchronous in nature :-

It refers to line by line execution it means step by step order and each line of code waits for the previous one to complete before moving on.

It means tasks are executed sequentially and the next task does not start until the current one finishes

4. Interpreted Language :-

In JS Engine we have interpreter that will execute the code line by line and left to right. If any error occurs the code will execute till the previous line and rest of the code will not be executed.

5. Loosely - typed Scripting Language:

JS is loosely typed which means it will not follow strict syntax

for e.g. There is no need of semicolon at the end of the statement and no need to define the datatype when declaring the variable and dynamically we can change the value for the variable

6. Single - Threaded:

Only one task can be performed at a time.

The current version of JS is ES6 (ECMASCRIPT)

Node.Js is a runtime environment which is used to execute the JS in the server side.

Variables in JS

Variables or the containers to store the data or value which can be any datatype

Variable name also one of the identifiers

We can declare the var in 3 ways

Using var, let, const Keywords

	var	let	const
syntax:	var a = 10;	let b = 20;	const c = 30;
Scope:	global	script	script
Redeclaration:	✓	×	×
Reinitialization:	✓	✓	×

We can only declare let and var variables but in case of const we could not able to only declare the const variable and it will throws error 'const' declarations must be initialized.

Undefined is the datatype and not defined is the error

If we declared any var without assigning the value and if we try to access the variable it will return undefined.

If we try to access any variable without variable declaration it will throws error i.e. var is not defined.

Tokens

Tokens are the smallest unit/ building blocks of every programming language

In tokens we have identifiers, literals, keywords and operators

Identifiers:

Identifiers are the Names given to variable, class, function

Function parameters also one of the identifiers

Rules for Identifiers

1. Identifiers should not start with number and it should be unique but it can have number.
2. Keywords are not allowed as an identifier
3. Except $ and _ no other special characters are allowed
4. Spaces are not allowed. so we can use uppercamel case for class name and lower camel case for variables and functions (not mandatory but good practice)

Keywords:

Keywords are the predefined or reserved words and each keywords having different functionalities

There are 50 + keywords in JS

For e.g. let, for, var, const, type of, in, of, do, while, etc

Operators:

Operators are used to perform some operations in JS

The types of operators are.

1. Arithmetic Operator (+, -, *, **, /, %.)
2. Comparison operator (>, <, >=, <=, ==, ===, !=, !==)
3. Assignment Operator
4. Logical Operator
5. Increment / Decrement Operator
6. Typeof operator
7. Ternary Operator
8. Concatination Operator

Note:- '==' operator will check only the value but '===' operator will compare both value and datatype

typeof operator is used to check the datatype of the data

Datatype

Datatype defines what type of data or what kind of data is stored in the variable

We have 2 types

1. Primitive Datatype
2. Non-primitive Datatype

In Primitive datatype we have number, string, boolean, undefined, bigInt, null, symbol and NaN

In Non-primitive datatype we have function, array and object

1. Number :
   1. We can store the values within the range $(-2^{53}-1)$ to $(2^{53}-1)$ in the number datatype
   2. Number datatype accepts integer values and decimal values also

2. String :
   1. It is used to store bunch of characters which can be any character
   2. We can store the characters within ' ', " " and backticks ` `
   3. String is the index datastructure and each and every character will be stored in respective index position and the index position always starts from 0.
   4. length property is used to find no. of characters present within the string

Q. ☆ what is the advantage of using template literals (backticks) (` `)

→ we can perform interpolation within the string using backticks i.e. we can perform validation operation or we can pass any variable with the backticks and syntax is ${ }

3. Boolean : 1. we can store only true and false in boolean datatype

4. Undefined : 1. If we declare any variable and not assigning any value to the variable and if we try to access it will print undefined

2. The datatype for undefined is undefined

5. BigInt : 1. It is used to store big integer and it should be suffixed with 'n'.

2. The numbers which is less than $(-2^{53}-1)$ and which are greater than $(2^{53}-1)$ we can store in BigInt datatype

6. null : null is used to dereferer the object and the datatype of null is object.

7. Not a Number (NaN): If we try to perform any arithmetic operation between string and one number it will return the value as NaN

The datatype of NaN is number.

TypeCasting: It is the process of converting one datatype into another datatype

Types

1. Implicit Typecasting: In implicit typecasting the data is converted from one datatype into another datatype internally.

2. Explicit Typecasting: Manually we have to convert one datatype into another datatype

ParseInt Method

It converts string into an integer and it will take one argument string reference variable

If ParseInt method could not able to convert the string into number it returns NaN.

ParseFloat Method

It is used to convert string into decimal (floating point number)

Non-Primitive Datatype:

1. Function: Function is a set of instructions which is used to perform some specific task

Syntax: function (keyword) fn name(a, b) { (parameters)

code / body } Function defn / Implementation / body

fn() // function call

We can call the function multiple times

Parameters:

Parameters are the containers which is used to store argument value or data which can be any datatype

We can also set the default value for parameters.

Parameter is also one of the identifier.

We can access the parameter values only inside the function

Arguments:

Arguments are the values which is passed to the parameter.

If we are not passing any argument value then the parameter value will be undefined

The no. of parameters must be equal to the no. of arguments

Return Statement:

Every fn must return some value and only one value can be returned by a fn

Return stmt must be the last statement inside the fn and after return stmt if we define any code it will not get executed

If we return multiple values the last value will be considered as the return value

If the fn is not returning any value then undefined is return

Return Type:

It is the datatype of the data return by a method or fn.

If the method is not returning any value then the return type is void.

Javascript Output Methods

1. Console.log()

It is a printing statement which is used to print some value in the console window

Console is the object and log is the function

2. Document.write/ writeln:

It is used to display some text or content in the browser

Document is an object, write and writeln are the methods

The difference betn write and writeln will give the space after the content or text and write will not give the space

3. Alert:

It is a popup method and it is used to display the alert message to the user in the browser.

The return type is void

4. confirm():

It is also a popup method which is used to confirm the message from the user

It will return boolean value true if the user clicks ok and if the user clicks cancel then it will return false

5. prompt():

It is a popup method which asks the user to enter the input and return in the form of string.

If the user clicks cancel then it will return null

Types of functions:

1. Anonymous fn:

The fn with no fn name

We can pass anonymous fn as an argument or we can store in any variable

We could not able to define anonymous fn alone

Syntax:

```
function () {
    code
};
```

2. Named fn()

The fn with fn name

Syntax:

```
function fn name() {
        code
};
```

3. Function with Expression: Assigning fn as a value to a variable (Anonymous fn)

Syntax:

```
let x = function () {
        code
};
```

4. Arrow Function: This concept was introduced from the version ES6.

Arrow fn will reduce the syntax for fn declaration.

In Arrow fn function keyword is not needed to create the fn and if we have only one line of code then { } is not mandatory and if only return statement is there then return keyword also not mandatory and if only one parameter is there then parenthesis is not mandatory

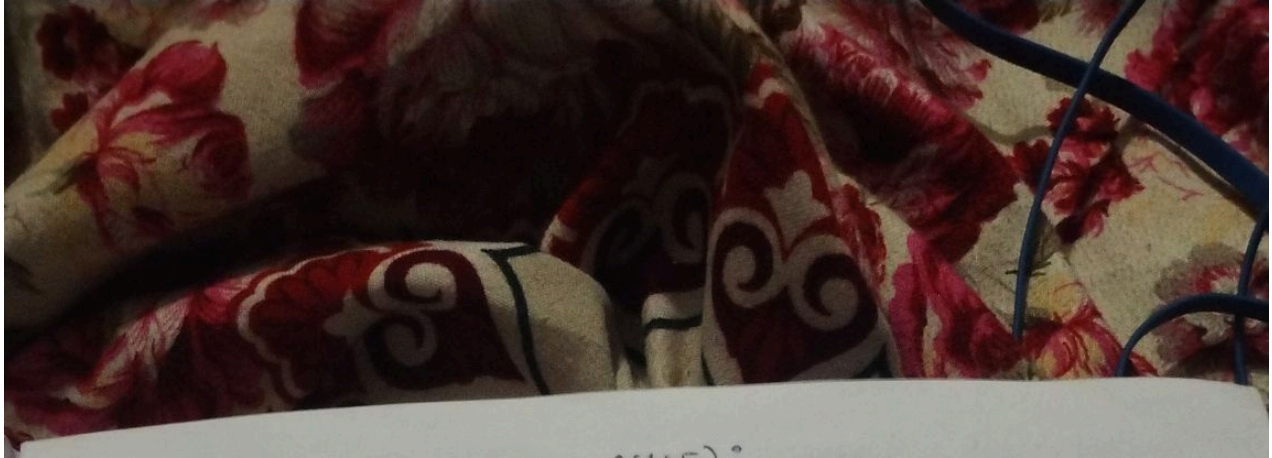
Syntax:

parameter ↑ ↗ Fat arrow

```
const arr = ( ) => {

    code

}
arr();
```



5. Higher Order Function (HOF):
The fn which takes another fn as an argument is called as HOF

6. Call back Fonction:
The fn which we are passing as an argument is called as call back function

HOF Syntax:

```
Function f1 (x, y) {
    x()
    console.log("HOF");
}
```

Callback Syntax:

```
f1( function() { clg("callback") }, 200);
```

Scopes in Javascript

1. Global Scopes: If we declare any variable with var Keyword then it will get stores in the global scope.

2. Script Scope: If we declare any variable with let and const Keyword then it will get stores in the script scope.

3. Local Scope: If we declare any variable with var, let, const within the fn or method then it belongs to local scope.
   Local scope is created only if the function calling statement is present
   Local scope variable cannot be accessed in the global scope.
   Function parameters also belongs to the local scopes.

4. Block: In javascript block is a group of statements enclosed in a `{}`.
   Blocks are used to organize codes and control the scope variables.

5. Block Scope: If we declare any variable with let and const keywords inside the block then it belongs to block scope and if it is declared with var keyword inside the block then it will belongs to the global scope.

Note: global variable can be accessed in the local scope and block scope but local and block scope variable cannot be accessed in global scope.

Debugger: In js debugger is a tool that allows the user to pass the execution of code at a specific point so that you can inspect the current state of the application
It is also allowing to examine variables callstacks and even modify the code etc.

Nested Function: Function which is present inside another function is called as nested function.

Closure: Closure is a scope of memory allocation which gets created in the nested function when we try to assess outer function variable inside the inner function and the closure will gets created for outer function.

Lexical Scoping or Scope Chaining:
It is the ability of a function scope to access the variable till it reaches the global scope.

Callstack:
It is a data structure that Keeps tracks of function call.
It is follows last in first out principle (LIFO).

Window object (Global object)

It is the supermost object in JS and it holds the information about the browser window. Window object having inbuilt properties and methods and directly you can access these properties and methods without object name

Some of the properties and methods of Window object are alert(), confirm(), prompt(), fetch(), document object, local storage, session storage, etc

If we declare any variable with var keyword globally it will get stored in the window object

Globally this keyword refers to the window object

Variable Hoisting

Moving variable declaration to the top of the scope is called as variable hoisting

It is applicable for all the scope

If we access any variable with var keyword before variable declaration and initialization it will return undefined but in case of let and const it will throws error i.e. Uncaught referrence error: cannot access the variable before the initialization

This happens because of Temporal Dead zone (TDZ)

Temporal Dead Zone (TDZ):

It is a time period betn the variable declaration and initialization and during this time period we could not able to access any variable

It is a behaviour in JS where variables declared with let and const cannot be accessed before they are actually declared in the code

Immediate Invoke Function Expression:

It is a technique where we can avoid the global pollution only one time we can call this function.

Syntax:

(Function Declaration)

```
( function() {

});(function call);
```

Use strict:

This statement should be present at the top of the JS code and it is used to follow strict syntax.

For e.g:

1) If we are not declared the variable with var, let and const it will throws error
2) If we are passing duplicate parameter for the fn then also it will throws the error.

Function Hoisting

Moving fn declaration to the top of the scope (every scope)

So that we can call the fn before fn declaration

Function Hoisting is applicable only for named function

isNaN():

It will take one argument and it will return true if the argument is NaN or it will return false Before checking whether it is NaN or not it will convert into number datatype

Number.isNaN():

It behaves exactly like isNaN but only one difference is it will directly checks the argument whether it is NaN or not

Eg: console.log(isNaN("str"));
console.log(Number.isNaN("str"));

Array:

Array is the non-primitive datatype in JS and it is used to store multiple values which can be any datatype

Array length is not fixed

Array is the index data structure and index position always starts from 0

Dupliocate values are allowed in the array

Array is iterable.

There are 2 types of array

1. Homogeneous Array:

   All the array elements having same datatype

Eg: let ar = [10, 20, 30, 40, 50]

2. Heterogeneous Array:

   All the array elements are not having the same datatype

Eg:

```
let arr = [10, 100n, "javascript", null,
           () => "arr", {name:abc},
           [100, 200]]
```

We can add, fetch, delete and update the array elements from the existing array

To delete the array elements we can use delete keyword and it will not affect the length of the array

If we try to fetch any array element where the index position is not present then it will return undefined.

Array.isArray() method:

The datatype of Array is object
To check the data whether it is array or not we can use isArray() method and it will return true if it is an array otherwise it will return false

Length property
It is used to fetch length of the array

Looping Statements
If we have to perform any task again and again based on condition we can use looping statements such as while loop, do-while loop, for loop

for-in loop
It is used to iterate the index of the data which is iterable

for of loop:

It is used to iterate the values of the data which is iterable.

Array Methods

Array Higher Order Function (HOF)

find method:

Find method will return the first element in the array which satisfies the condition

It will take one argument i.e. callback fn and that callback fns will take 3 parameters (value, index, array)

We should pass the condition in the return stmt of the callback fn

find method will iterate each and every array element and once the condition is satisfied it will return the element. If the condition is not satisfied for all array elements it returns undefined

findIndex method:

This method is exactly same like find method but the difference is it will return index position of an array element which satisfies the condition otherwise -1 will be returned

filter method

filter method will return the array of all the elements which are satisfying the condition

It will iterate each and every array element till the last index

If the condition is not satisfied for all the array elements it will return empty array

map method

It is used to perform the same operation to all the array elements without modifying the original array

foreach method

foreach method is used to iterate the array or you can perform same actions for every array element

Return type is void

What is the difference betn map method and foreach method

→ Map method will perform the same operations for all the array elements and it returns new array

Return type of foreach method is void

some method
some method will return true if any one array element satisfies the condition otherwise false

every method
every method returns true if all the array elements satisfies condition otherwise false

reduce method
This method is reduce the array and return the single value and it will take 2 argument
1. Accumulator 1. initial accumulator value
2. callback fn

This callback fn will take 2 parameters
1. Accumulator
2. Current value

Eg :-

```
let ar = [10, 20, 30, 40, 50];
ar.reduce ((acc, cv) => {

    return acc+cv;

},0)
```

reduceRight method
It will reduce the array from right to left.

sort(): This method is used to sort the array based on the ASCII value. but in case of number array we have to pass one function as an argument that is compare function which will compare each & every array elements
This compare fn will take 2 parameters and during every iteration that compare fn will take the array element & store in the parameter - This compare fn will return number If 0 then there will be no change
If +val then b sorted before a If -val then a sorted before b

Array Methods

Non-static methods

1. push(): It is used to add the array element at the end of the array and returns the new length of the array

It will take multiple argument so that we can add multiple array elements

2. unshift(): It behave exactly like push() but it will add the array element at the beginning of the array

3. pop(): It is used to remove the array element at the end of the array and only one array we can remove and it will return the deleted array element

4. shift(): Extracts an item from the beginning.

5. splice(): It is used to add and delete the array element at the particular index position and it will return the array which contains delete array elements as well as make changes in existing array

6. slice(): It is used to extract the part of the array and return in the new array

7. concat(): It is used to merge 2 or more array and it will not affect the original array

8. reverse(): It is used to reverse the array and it will affect the original array

9. fill(): Changes all array elements from start to end index to a static 'value' and returns the modified array

- first parameter: the value
- second parameter: start index
- third parameter: end index -1

10. indexOf(): Returns the index of the first occurrence of a value in an array if it is not present returns -1

Optional second parameter: The array index ~~tat~~ which to begin the search.

11. flat(): to convert multidimensional array into a single dimensional array.

12. join(" "): ~~join the array element by the seperator~~ Convert the string into array based on the seperator which we are passing as an argument

13. lastIndexOf(Search element from Index):
-Returns index of last occurrence of a specified value in an array or -1 if it is not present.

14. includes: Determines whether an array includes a certain element, returning true or false

Ways to create an Array in JS:-

1) Literal way:- var arr = [10,20,30]

2) Using Array Constructor:- const arr = new Array(100,200,300)
clg(arr)

3) Array.of():- const a = Array.of(200,300,400)
clg(a)

Note:- Difference bet^n creating array using array constructor & Array.of():-

① If we pass one argument that is number datatype then it will consider length of the array → array constructor

② In case of array.of method it will consider as a element of the array

Eg:- ① clog(new array(6)) ⇒ length of arr

② clog(Array.of(6)) ⇒ element of array

String Methods:

1) slice(): It is used to extract the part of the string and returns new string
   It will take 2 arguments
   - 1st starting index
   - 2nd is End index and it will not consider the last index character
   - It will not modify the original string
   - We can pass the -val as an argument

2) Substring(): It is used to extract the part of the string and returns new string and it works exactly like slice method but it will not take -val as an argument.

3) Substr(): This works also exactly like slice method but 2nd argument is length of the extractor string

4) includes(): - It will return boolean value
   - It will take 2 arguments 1st search string 2nd is from index
   - It will return true if the search string will present in the main string otherwise false

5) concat(): It is used to merge 2 or more strings and return in the new string.

6) split(): - It is used to convert string into array based on the seperator which we are passing as an argument.

7) replace(): - It will take 2 argument 1st is search ~~string~~ value and 2nd is replace value
   - This will replace the 1st occurence of the substring given if the substring is present
   - It will not affect the original string

8) replaceAll(): - It is exactly like replace method but it replaces all the substring if search value is present.

9) toUpperCase(): It is used to convert all the character into uppercase

10) toLowerCase(): It is used to convert all the characters into lowercase.

11) toString():- It returns the string representation of an object.
-It converts any other datatype into string datatype.

12) indexOf():- -It will take 2 arguments 1st is search string and 2nd is from Index
-It will return the index position of the 1st character of the search string if the search string is present in the main string otherwise it will return -1
-It starts searching from left to right.

13) lastIndexOf(): It is exactly the indexOf method but difference is it starts searching from right to left

14) trim(): It is used to remove the white space at both the side of the string

15) trimStart():- It is used to remove the white spaces from the start of the string.

16) trimEnd(): It is used to remove the white spaces from the end of the string

Join():-[

Object : Object is a non-primitive datatype in JS and it is used to store the properties and methods in the form of key and value pair.

Keys should be uniaue.

Object are not iterable because it is not having index data structure

Internally the keys in an object will be stored in the form of string

If we are passing any property with the same key the value will gets updated

we can also access the values from the object by 2 ways

1) . notation

2) () notation

If we are using () - notation then keys should be within the string

we can also add, delete and update the object values

To delete the object key and value we can use delete keyword

We can create object in 4 ways

1) Literal way : Directly we are passing properties and methods in the form of key and value pair within the {} bracked

Eg :- let std = {

name = "abc";

id = 121;

}

2) Using class and Constructor : It is a blueprint to create multiple objects

Inside the class we can pass properties and methods that can be static or non-static members

Inside class only constructor should present.

Static members can be accessed by classname and non-static members can be accessed by object reference

Constructor : It is a special type of method and it is used to assign the values for the keys within the object.

Constructor can be called with the help of new keyword

It will take parameters and arguments

If we call the constructor then object for the class will be created.

Constructor will not return any value

this keyword inside constructor refers to current creating object.

If we are going to create multiple objects with same key then we can use class constructor

3) Using Constructor C) fn :

Constructor C) fn must be the named fn

Constructor C) fn exactly behaves same as class constructor but the only diff^n is syntax.

Syntax :

```
function functionname (val1, val2)
{
    this.key1 = value1;
    this.key2 = value2;
}
```

this constructor fn C) also invoked using new Keyword.

4) Using Object Constructor

We can also create the obj by invoking a object class Constructor

Eg :

```
let a = new Object();
```

It will create empty object.

Static methods in Object

1) Object.freeze() : If we are using freeze() m for an object we could not a to add, delete, update any from the object.

2. Object.isFrozen(): This method will return boolean value if the object is in freeze state otherwise false.

3. Object.seal(): Here we can able to update the values in the object but we could not able to add or delete any keys and values.

4. Object.isSealed(): This method will return boolean value true if the object is in sealed state otherwise false.

5. Object.keys(): It will return all the keys from the object in the form of arrays.

6. Object.values(): It will return all the values from the object in the form of array.

7. Object.entries(): It will return both keys and values from the object in 2d array.

8. Object.assign(): This method is used to merge 2 or multiple objects and it will take 2 or multiple arguments
   - First one is target
   - Rest of the arg are source

This method after merging will returns the targeted object and this method will make changes in the targeted object.

Rest parameter : It is used to store multiple arguments in single parameter
It must be present at the last in the parameter list
It should be prefixed with (...)
It will return rest of the arguments in the form of array

Spread Operator : It is used to spread the values for those datatype which are iterable

Imp We could not be able to spread the object because it is not iterable

Destructuring :

1) Array Destructing : It is used to pass the unique identifies for the array elements.

In array destructuring it follows order or sequence i.e. first element will get stored in the first identifier

We can also store multiple array elements in a single identifier by using rest element

Eg

2) C

Eg :-

```
let arr = [10, 20, 30, 40, 50]

let [a, b, ...c]
        ↳ rest element
```

2) Object Destructuring : This is used to give the unique identifiers for the object elements

In object destructuring it will not follows order or sequence i.e. we can able to destructure last object element at first

In object destructuring the identifier must be equal to the key

we can also change the identifier by using colon [:]

Eg :

```
let obj = {
    name: "abc",
    id : 121,
    skills : ["Java"],
    percentage : 70,
};
```

```
let { id:eid, skills,
    ...x } = obj;

clg (eid); //121
clg (skills); //["Java"]
clg (x); {name:"abc",
          percentage:70}
```