History of Java-Script

-->JavaScript was created in May 1995, by Brendan Eich.
Eich worked at Netscape and implemented JavaScript for their web browser,Netscape Navigator.

-->The idea was that major interactive parts of the client-side web.

-->Initially, JavaScript's name changed several times:

- Its code name was Mocha.
- In the Netscape Navigator 2.0 betas (September 1995), it was called LiveScript.
- In Netscape Navigator 2.0 beta 3 (December 1995), it got its final name, JavaScript

-->The "ECMA" in "ECMAScript" comes from the organization that hosts the primary standard.The original name of that organization was ECMA, an acronym for European Computer Manufacturers Association.

--> NOTE:

JAVA SCRIPT and ECMA SCRIPT both are same.

What is javascript? why js?

--->JavaScript (js) is a light-weight object-based and object-oriented programming language which is used by several websites for scripting the webpages.
--->JavaScript is a client-side scripting language and one of the most efficient, commonly used scripting languages. The term .client-side scripting language means that it runs at the client-side (or on the client machine) inside the web-browser

Note:
---JS is H.L.L (one can easily read)
---Browser:it is an application which provides environment to run JS instructions from HLL to MLL.
---JS code can run both inside and outside the browser.
Inside browser - JS engine. Outside browser - Node.js

---JS engine:
--Node:it is the combination of c++ and V8(Chrome JS engine) with bundle of methods and rules.
--Node provides environment to run JS outside the browser.This invention helps the JS to gain its popularity in usage as a backend language.
--We can run JS both inside and outside the browser.

CHARACTERISTICS OF JAVASCRIPT:

1.Purely object based and object oriented programming language.
2.case sensitive
3.just-in-time compailer
4.Interpreted language
5.Synchronous(coz, it follows SINGLE threaded architecture which has only one stack)

To execute JS on browser:

1.we can execute js instructions directly in the console provided by the browser.
2.we can execute js instructions by embedding it in html page.
types: a.internal, b.external

a.internal:
with the help of script tag, we can embedded js instructions.
syntax:

```
html code ...
<script>

    js code...

</script>
```

---

b.external:
1.we can create a separate file for js with a extension .js
2.link the js file with html page using src attribute of script tag.
syntax:

```
html code ...
<script src="path/filename.js"> (if js file and html file in the
    same folder path is not required)
    js code...
</script>
html code …
```

FUNCTION:
-named block of instructions which is used to perform a specific task.
-Function gets executed only when it is called.
-the main advantage of function is ,we can achieve code reusability.
-MAIN PURPOSE : code reusability, which means-code once declared can be anywhere.

Note:
-In JS function are beautiful, every function is nothing but an object.
Syntax to create a function:
Generally we can create a function in 2 ways:
1.using function declaration statement(function statement)
2.function expression.

1.function declaration statement:

```
function identifier(parameter1, parameter2,...)
{
 statements
}
```

Note:
-function is an object
-name of a function is a variable which holds the reference of function object.
-creating a function using function statement supports function hoisting. therefore we can also a function before function declaration.
ex:

```
console.log('start');
console.log(test);
function test()
{
 console.log("hello")
}
console.log("end");
```

-When we try to log the function name the entire function definition is printed

PARAMETERS:
The variables declared in the function definition/declaration is known as parameter.
The parameters have local scope.Those can be used inside the function body
PARAMETERS are used to hold the values passed by a calling a function.
EX:

```
function sum(a,b)
{
 console.log(a+b);
}
```

here a&b are variables local to the function sum.

ARGUMENTS:

-The values passed in the method call statement is known as arguments.
-An argument can be a literal, variable or an expression which results a value
Ex:
sum(10,20)----- 10,20 are literals used as arguments
-ex2:
sum(-10+3,-20);//-27
-ex3:
let a=20,b=30
sum(a,b);a&b are variables used as arguments

RETURN:

-It is a keyword used as control transfer statement in a function.
-Return will stop the execution of the function and transfers the control along with data to the caller.

2.FUNCTION EXPRESSION:
SYNTAX:
var/let/const identifier = function(){}
in this syntax the function is used as value.
disadvantage:
the function is not hoisted, we can not use the function before declaration.
Reason:function is not hoisted, instead variable is hoisted and assigned with default value
undefined.Therefore typeof a is not a function, it is undefined. Refer below ex,

```
a();//error; 'a' is not a function
var a = function()
 {
clg("hi");
}
clg(a)://function state or body is printed
a();//hi, function calling
a = 10;
```

a();//error, 'a' is not a function, coz, a is changed to number type from function. This we call it as
Dynamic type. I can change variable during run time.

Note:
1.Any member declared inside a function will have local scope

Local scope:
The scope within the function block is known as local scope.
Any member with local scope can not be used outside the function block.

Ex:
1.A parameter of a function will have local scope
ex:function test(a){} ----------> here 'a' is variable whose scope is local
to test and it can be used only
inside test function block.
ex2:function test(){
var a; ----->it is local to test function, it can not be used outside.
}
ex3. function test(){
function insideTest()
{
}
}
Note:insideTest is function, local to test function and can not be called
from outside test()

NOTE:
-Inside a function we can always use the members of global scope
Ex:
var city = 'Bangalore';
function display(name){
console.log(`${name} belongs to ${city}`;
}
display('Sheela');
in the above ex variable name has local scope and var city has global scope.
It is observed inside function body we can use both the variables name &
city.

this:
-It is a keyword used as variable.
-It holds the address of global window object.
-Therefore with the help of this variable, we can use members of global
window object.
Ex: var b = 20
function test(){
var b = 30;
console.log('from test '+b)//30
console.log(this.b);//20
}
test();
-In JS 'this' is a property of every function(every function will have
'this' keyword);
-Generally 'this' contains the address of current execution context to
which the function belongs to.

FUNCTIONAL PROGRAMMING:

Passing a function as an argument to another function is known as functional programming.

Advantage of functional programming:

-Instead of creating a function with a specific task, functional programming helps programmer to generate function which can perform a generic task by accepting a function as a parameter.

ex:

```
function operation(a,b,task){
 let res = task(a,b);
 return res;
}
let res1 = operation(10,20,function(a,b){
 return a+b;
});
let res2 = operation(30,20,function(a,b){
 return a-b;
});
let res3 = operation(10,5,function(a,b){
 return a/b;
});
console.log(res1);
console.log(res2);
console.log(res3);
```

-In the above example the function operation which can accept a function(task) as a parameter is known as higher order function(operation()).

-The function operation which is passed as an argument to another function is known as callback function.

Hihger order function:A function which accepts function as a parameter is called as HO function.

callback function:the function which is passed as an argument is called call-back function

ARROW FUNCTIONS:

-Arrow function was introduced from ES-6 version of javascript.
-main purpose is to reduce the syntax

-Syntax:
(parameter list,...) => {}

-Arrow functions can have 2 types of return:
a)implicit return:using 'return' keyword is not required.
syntax:
(parameters_list....)=>expressions

b)explicit return:using 'return' keyword is mandatory with curly brackets{},
if 'return' keyword is not used 'undefined' is given back.
-If a block is created an arrow function behaves like explicit return
syntax:
(parameters_list....)=>{return expressions}

eg:-

```
// const sum = (a,b) => a+b;
// console.log(sum(10,20))

// const product = (a,b) => a*b;
// console.log(product(10,20))

// const remaider = (a,b) => a % b;
// console.log(remaider(10,20))

// const difference = (a,b) => a-b;
// console.log(difference(10,20))
```

IMMIDIATE INVOKE FUNCTION EXPRESSION:

-When a function is called immediatly as soon as the function object is created it is known as Immidiate invokation.
-onetime use and throw

-Steps to achive IIFE:
a.Treat a function like a expression by declaring inside a pair of brackets.
b.Add another pair of brackets next to it which behaves like a function call statement.

eg:

```
let a = ( () => {console.log("hii")
                }) ();
```

NESTED FUNCTION:

-In JAVASCRIPT we can define a function inside another function.

```
-function outer(){
 function inner(){
 }
}
```

-The outer () is known as parent,and the inner() is known as child

Note:

-The inner() is local to outer function,it can not be accessed from the outside.

-To use inner function outside,the outer() must return the reference of inner().

Ex:

```
function outer(){
 function inner(){

 }
return inner ;
}

outer()()
```

-We can now call inner() from outside as follows:

-->type 1.

```
let fun = outer();
fun();
```

-->type 2.

```
outer ()();
```

SCOPE CHAIN(LEXICAL SCOPE):

-The ability of the javascript engine to search for a variable in the outer scope when it is not available in local scope is known as Scope chain.
-The scope chain is generally established between,

1.function & global object,
eg:-

```
let a = 10;
function test(){
    ++a
 console.log(a)
 }
```

2.child function & parent
function with the help of closure.

2.child function & parent:(nested function)
ex:

```
let a =10;
function x(){
let b=20;
 function y(){
 console.log(b);
 console.log(a);
 }
 return y;
}
x()()
```

NOTE:
when the function y() is executed console.log(b) encountered,js engine looks for b in the local scope of function y(),since b is not present the y() is child of x() js engine will search for b in the parent function x() scope with the help of CLOSURE.
-We can achive scope chaining with the help of closure.

CLOSURE:
-The closure is a binding of parent function variables with the child function. This enables a JS programmer to use parent functions member inside child function.

DOCUMENT:
-Document is an object created by browser
-The document object is the root node of DOM tree.

DOM:
-The every html element is considered as a node(js object) in DOM.
-DOM allows to modify the document content rendred by the browser without reloading the page.Therefore DOM helps to make a web page dynamic.

NOTE:
-Any modification done using DOM is not updated to the original page. Therefore , once we reload apage all the modification done using DOM will be lost.

-We can write the content on the browsers dynamically with the help of write and writeln method of document object.
EX:
To display a message on the browser page from javascript code:
document.write("hi")

-DOM is an API provided by the browser
-DOM is not javascript,it is built using javascript
-DOM is a object oriented represetation of html file.
-Every time an html document is given to the browser ,it automatically generates dom tree it can be accessed and modified using the root node document in javascript

TO OBTAIN THE ELEMENTS FROM DOM:

methods of dom

1.getElementById():

-for this method we need to pass the ID of an element as a string ,it returns the first element specified ID.

Syntax:

document.getElementById("ID");

2.To fetch an element using 'name' attribute:

-we need to pass the name of an element as a string.

-It returns a nodelist,which is similar to array but not an array

-If there is no an element with the given name,then it returns empty nodelist

EX:(refer below ex in VS code)

```
let e1 = document.getElementsByName("heading")
console.log(e1[0]);
console.log(e1[0].textContent);
e1[0].textContent='welcome';
console.log(e1[0].textContent);
```

3.To fetch an element from DOM using the class name:

getElementsByClassName():

-Pass the class name as a string

-It returns a nodelist,containig all the elements matching the given html collection, which is similar to array but not an array

Syntax:

document.getElementByClass("class");

```
let e3 = document.getElementsByClassName("divstyle1")
console.log(e3[0].textContent);
```

4.To fetch the elements from the DOM using tag name

getElementsByTagName():

-We need to pass tag name as a string

-It returns an html collection containing all the elements matching the given tag name.

To fetch the elements from the DOM using CSS selectors:

-We can select an element from DOM using css selectors with the help of

1.querySelectorAll

2.querySelector

Note:

-We need to pass a css selectors as a string

-Query selector returns the reference of first element found.

-QuerySelectorAll returns a node list of all the elements found in the original order.

PROPORTIES OF DOM OBJECT WHICH RETURNS DOM OBJECT REFERENCE:

1.firstChild
-It is a property, which holds the reference of first child node of the target node.
-syntax:
target_node. firstChild
-firstChild can be anything either a text or a comment or an element

2.firstElementChild
-It is a property ,it gives the reference of the element which is the firstchild of the target node.
-syntax:
target_node.firstElementChild

3.lastChild
-It is a property,it gives the reference of last child object of the target node.
-syntax:
target_node.lastChild
-It can either return text,comment or element object.

4.lastElementChild
-It is a property, it returns the reference of an element which is a last element of the target node.

5.children
-It is a property,it contains list of all the elements which are children of target_node

6.childNodes
-It will give all the child nodes including text and comments

7.nextSibling
-It gives the reference of node which is immediate right sibling of DOM

8.nextElementSibling
-It gives the reference of the node which is next to target_node to its right.

9.parentElement
-It will just give the parent element of the current target_node

manipulation of dom

-->modifying and updating dom tree

a)adding elements to the dom
--->we can add elements in the dom in the following ways:
1)using the property in a **InnerHTML.**
2)by creating an element using **CreateElement()** method of dom object.

---

1)InnerHTML
--->innerhtml is a property of an dom element.
--->the innerhtml conatins everything between the tag as a string
--->we can update property innerhtml

synatx to update:-

target_elemet.innerHTML = "STRING"

target_element.InnerHTML += "string/with htmltags" (gives previous and updated tags)

disadvantage:-
1)security issues :-(if i pass " " without giving += so, the white-space will be updated)
2)it will reduces the efficiency of browser, because the dom is recreated by every time innerHTML is modified.

---

2)using createElement() method
we have two steps :-
step 1 :- create an element object
syntax:-
document.createElement ('tagname');
step2 :- insert the element into the dom.
syntax:-
target_node.appendchild(reference_of_ele_object_to_be_added)

ASYNCHRONOUS

-The behaviour of making way for others is known as Asynchronous.

Explanation:
-Let us consider 2 functionalities f1&f2 purely independent from each other. Let us assume f1 takes 10 minutes to complete the execution,f2 takes just a minute to complete the execution.Therefore,if f1() is called first and then f2().

-f2() should wait for 10 minutes to get execution started. This behavious is synchronous.

-To overcome this we can design our application in such a way that f1() gives way for f2() to complete its execution & then f1() can complete its execution.Such a design is known as Asynchronous.

-We can Asynchronous behaviour with the help of setTimeOut() method.

setTimeOut():
-It is a method of window object
-It accepts 2 arguments, argument 1 --> A callback function
argument 2 --> delay time in miliseconds
-It will register the given callback function and starts the browser timer for the given miliseconds when it is called.
-Once the set time is completed a callback function is moved to callback que.
-The 'event loop' loads the callback function from the callback que into the stack when the stack/callStack/mainStack is 'idol'
-Event loop waits untill callStack/mainStack become idol.
-Callstack becomes idol only after all the instructions of Global Execution Context is completed.

PROMISE:
Promise is an object
-Promise object keeps an eye on the asynchronous task given .
-If the asynchronous task not yet completed,the promise is considered as 'pending'.
-If the asynchronous task is successfully completed,then the promise is considered as 'resolved'.
-If the asynchronous task is completed but not successfull,then it is considered as 'reject'

SYNTAX TO CREATE A PROMISE OBJECT:

```
new Promise((resolve,reject)=>{

 asynchronous_task
} )
```

1.then():
-It can accept a callback function
-The callback function passed(given) to the then() methods gets executed only when the promise returns 'resolved'.
2.catch():
-It can accept a callback function
-The callback function given to catch() gets executed only when the promise retuns 'reject'

The JavaScript string is a sequence of characters.

There are 2 ways to create string in JavaScript

By string literal
By string object (using new keyword)

1) By string literal
The string literal is created using double quotes.

```
var stringname="string value";
```

2) By string object (using new keyword)
The syntax of creating string object using new keyword is given below:

```
var stringname=new String("string literal");
```
Here, new keyword is used to create instance of string.

ARRAYS:

-Array is a huge block of memory which is used to store multiple values of different type.
--Arrays in JavaScript are heterogeneous. You can store any kind of data in the array.
--eg:-var arr = [2, "Hello", 123.5, true, person];

-To create an array :
a.(in javascript array is an object)
b.We can create array object in 2 different ways :

i)using array literal []
ii)by creating an instance of array, using new operator.

i)using array literal []:
Syntax:
let arr = [value1,value2,value3,.....] --->here 'arr' is the variable which stores the address of array object
EX:
let arr = [10,20,30];
console.log(arr)
---
ii)by creating an instance of array, using new operator.

Syntax:
let arr2 = new Array(); ---here '()' is function call
---> 'Array()' array function constructor
Array ----> anything starts with capital A it is class
() ---->constructor / constructor function which is having same name as class.
---> 'new' is keyword,which creates new object

note:an array object is creted but no data is

JavaScript array constructor (new keyword)

eg:
let arr3 = new Array(10,20,30);
creating a array objects and passing data to array and address of array of object is given to arr3.

To access the array elements :
-we can access array elements with the help of array object reference,array operator & index
syntax:
array_obj_ref[index]
Index: it is a number starts with zero and ends with the (length of array object - 1)
Ex:
let hobbies = ['reading','writing','playing cricket']
console.log(hobbies[1]);//writing

ADDING ELEMENTS INTO THE ARRAY OBJECT:

1.To add the element at the tail(last)

syntax:
array_ref[array_ref.length]=element;
ex:
let a = [10,20,30];
a[a.length]=40;//40 is added in the last
console.log(a);//[10,20,30,40]

Array.push method:

-it is a built in function of array object
-push function/method adds given arguments into the array object at last.

eg:let a = [10,20,30];
a.push(5);
clg(a);//[5,10,20,30]

note:
we can add an element into array using
syntax:
array_ref[index] = element;

eg:let a = [10,20,30];
a[3]=50;
clg(a);//[10,20,30,50]

---

to remove an elemnt from an array

pop(): to remove an element from tail of the array object

eg:
```
  let a = [10,20,30];
clg(a.pop());//30
```

---

array.shift()
The shift() method removes the first array element
eg:
```
   const a = [10,20,30,40,50];
   a.shift();//[20,30,40,50]
```

---

array.unshift()
The unshift() method adds a new element to an array (at the beginning)
or first index.

eg:
```
 const a = [10,20,30,40,50];
   a.unshift(1,5);//[1,5,10,20,30,40,50]
```

---

indexof()
the indexof() method it returns the index of particular elemet which is passed through it.

```
var arr = [10,210,30]
arr.indexof(20);//2
```

---
includes()

checks if an array contains a specified element

eg:
```
let languages = ["JavaScript", "Java", "C", "C++", "Python", "Lua"];

let check = languages.includes("Java");
console.log(check); // true
```

Array splice()

The splice() method returns an array by changing (adding/removing) its elements in place.

syntax:
arr.splice(start, deleteCount, item1, ..., itemN)

eg:

```
   let prime_numbers = [2, 3, 5, 7, 9, 11];
   let removedElement = prime_numbers.splice(4, 1, 13);
   console.log(removedElement);

  //Output: [ 9 ]
//         [ 2, 3, 5, 7, 13, 11 ]
```

Array slice()

---The slice() method returns selected elements in an array, as a new array.
--- creates a new array

syntax:

```
      slice(start index, end index)
```

eg:

```
 var num = [10,20,30,40]
 num.slice(2,3);//[30]
```

OBJECT:

-Any substance which has its existence in the real world is known as OBJECT.
-Every object will have properties(attributes of feeds which describe them).
-Every object will have actions(behaviours)
EX:
1.car:
properties of car: model,color,price
actions of car:accelarate,brake,start,stop

2.webpage:
properties of webpage:url,data/content on webpage.
actions of webpage:scroll,hover,click

Note:
-In programming an object is a block of memory which represents a real world.
-The properties of real world object are represented using variables
-The actions of real world object are represented using METHODS(FUNCTIONS).
-In javascript object is a collection of attributes and actions in the form
of key&value pairs enclosed
within curly braces{}
-Key represents properties/attributes of an object
-Value represents state of an object.

-In javascript,we can create js object in 3 different ways:
1.Object literal{}
2.Using a function

Note:
-If the key is not present in the object then we get 'undefined'
as the value.
-The objects in javascript are mutable(state of the object can be modified)
in nature.

task on object:

Task:

1.Let us consider a real world object marker with the following attributes color, brand,price their states of black, camlin,25.

a) create javascript object to store.

b)Display the entire object.

c)log price

d)change black to blue and log it

e)add extra attribute shape into the marker object with value as cylinder

f)remove the attribute brand g)display the final object

task on array:

1.create an array with following elements: 10,30,40 & display the elements in the reverse order

2.Write an array to store names of at least 3 cities, display all the cities whose length is even