Application
- Presentation layer – HTML, CSS, Javascript
- Business layer – Java, C, C++
- Database layer – SQL, mysql, Nosql.

Difference b/w programming language, Scripting, query language

Programming language:

1.) A programming language is a language used to write set of instruction to perform a task.
2.) programming languages are compiled and runs independently
3.) used in application / software development.
4.) more time required to complete execution

Scripting language:

1.) No need for compilation
2.) interpretation is required
3.) Dependent on some other platform (browser)
4.) used for web development.
5.) lesser time is taken to complete execution.

Query language:

1.) A Query language is a specialized programming language for searching and changing the contents of a database.

what is JavaScript?

1.) Java Script is a Scripting language & high-level language.

2.) It is introduced by bredan Eich in 1995

3.) It became an ECMA Standaird in 1997

4.) Java and Javascript are completely different language, both in design and Concept.

5.) Javascript is world's popular programming language which is used in browser to makes web pages dynamic.

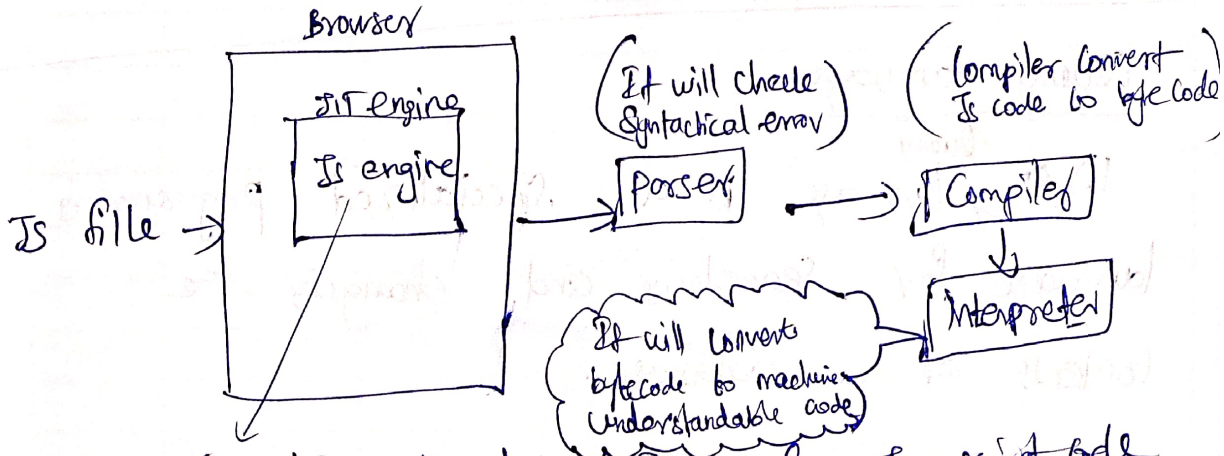
> ECMA means European Computer manufacturer Association

6.) old name: LIVEScript, then Netscape changed to Javascript

Difference between Static & Dynamic web pages?

Static : It have no functionalities

Dynamic : It have some functionalities or even loaded once



JS engine : It will take care of Javascript code which is inside the <script> tag

Java	Javascript
1.) It is a programming language	1.) It is a scripting language
2.) Strictly Typed	2.) It is weekly typed or dynamicly typed
3.) Runs on "JVM (Java virtual machine)	3.) Runs on all the browsers
4.) multi threaded	4.) Single threaded
5.) more memory used	5.) less memory used, light weighted language
6.) Independent language	6.) Executed along with html
7.) Both Compiled and Interpreted	7.) only interpreted (not compiled)
8.) high level language	8.) high level language

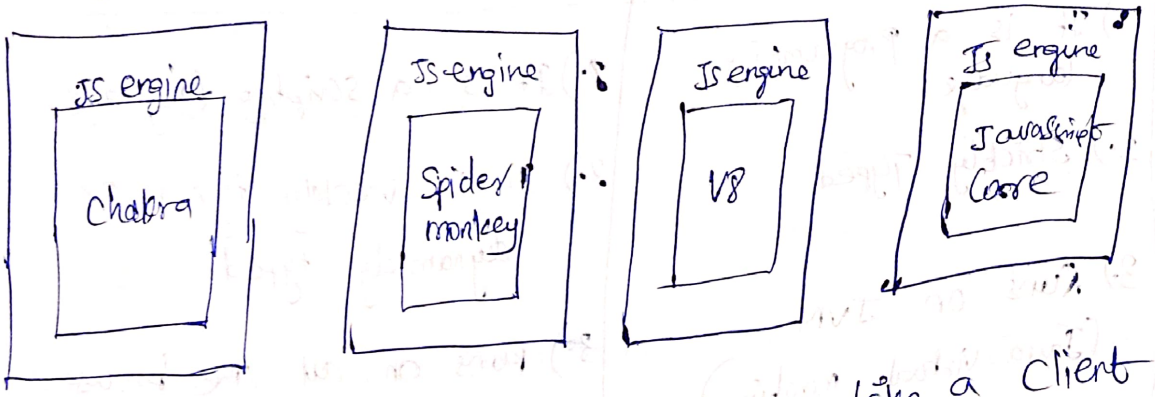
Working of Javascript:

1.) To execute Javascript instructions, Instructions Jscode is mandatory to translate into machine level language. This job is done by Javascript engine.

2.) Every browser have Javascript engine to execute Javascript code. Therefore, browser will become an environment for Javascript.

Edge Firefox Chrome Safari

3.)



4.) Browser is an application which acts like a client

5.) Console is a sub application of browser

Eg:- debugger, windows

---

ways to create Javascript

Internal

↓

Inside body tag

```
<body>
  <script>
  </script>
</body>
```

External

↓

```
<body>
  <script src = "js file">
  </script>
```

Node js

1.) Node.js is a combination of C++ built in methods and bundle of V8 engine.

2.) It serves as an environment to run Javascript outside the browser.

3.) The invention helped JS to gain its popularity in usage as a backend language

Characteristics of Javascript

1.) purely object oriented and based
2.) interpreted language.
3.) Case sensitive
4.) Synchronous (it follows Single threaded architecture which has only one Call Stack).

- **dot operator:**
  It will call function of objects.

Principle of programming language

1.) Variable → Named block of memory
2.) Operator → It is a symbol has predefined function
3.) Block → Set of code we should know how to call function block.
4.) Functions → code reusability.

Tokens

(Smallest unit of programming language).

- Keywords: (Pre-defined word)
- Identifiers (variable name, function name)
- Data literals.
  numbers, boolean, String, null, undefined, Big int

Identifiers :-

1) The name given to components by a developer is called 'identifier'. $ and _ allowed

Keywords :-

× Some identifiers are reserved for some functionality such type of reserved identifiers are keywords.

✓ Predefined reserved words which is engine can understand

Reserved words (63)

Used (47)

removed (16)

→ data types (3) → var, let, const

→ flow control (11) → if, else, switch, case, default, for, do, while, break, continue, return.

→ modifiers (4) → public, private, protected, static,

→ more → Abstract, arguments, await*, boolean, byte, catch, char, class*, debugger, delete, double, enum*, eval, export*, extends*, false, final, finally, float, function, goto, implements, import*, in, instanceof, int, interface, long, native, new, null, package, private, short, super*, synchronized, this, throw, throws, transient, true, try, typeof, void, volatile, with, yield.

Omitted keywords are banned from Javascript 5/6

Data literals

The data which is used in the JS programs.

1.) number

```
var num = 5;
Console.log (typeof num);          -> number
Console.log (typeof typeof num);   -> String.
```

Output

allowed values: $(-2^{53-1}$ to $2^{53-1})$

2.) Boolean

```
var num = true;
Console.log (typeof num);          -> boolean
Console.log (typeof typeof num);   -> String.
```

3.) String

```
var num = "hi";
Console.log (typeof num);          -> String
Console.log (typeof typeof num);   -> String.
```

If string wants new line use Enter

```
Var a = This is my
para;
```

Back ticks also called Template String.

4.) null

```
Var num = null;
Console.log (typeof num);          -> Object
Console.log (typeof typeof null);  -> String.
```

null is an Object in Javascript

null represents absence of object value

5.) undefined

```
Var num = undefined;
Console.log (typeof num);          -> undefined.
Console.log (typeof typeof num);   -> String
```

A variable that has not been assigned a value called undefined

6.) bigint

Note:-
1.) 2**53n → error
2.) 2n**53n → bigint

```
Var num = 1n;
Console.log(typeof num) → bigint
Console.log(typeof typeof num) → String
```

7.) Object

```
Var num = object;
Console.log(typeof num) → function
Console.log(typeof typeof num) → String
```

Mutable (non-primitive) (Array, object except object) all are primitive

If the value can change, the object is mutable

Immutable (primitive) (String)

If the value cannot change, the object is immutable

String, number, etc (primitive)

Primitive types like: String, numbers, boolean, null, undefined are mutable

Value can change.

non-primitive types: Array

immutable, value cannot be change.

Eg:- 1 for primitive types can mutable

```
Let x = 10;
Let y = x;
y = y+1;
C.l(x); - 10.
C.l(y); - 11.
```

→ Seperate memory location

Eg: 2 for immutable

```
let a = ["value1", "value2"];
let b = a;
b.pop();
C.l(a); - ["value1"]
C.l(b); - ["value1"]
```

→ Same memory location

Variables

1.) Block of memory which is used to store value is known as variable.

2.) In Javascript, Variables are not strictly types. It is dynamically typed.

3.) In Javascript, Not necessary to specify data type during declaration

4.) In variable, we can store any type of value

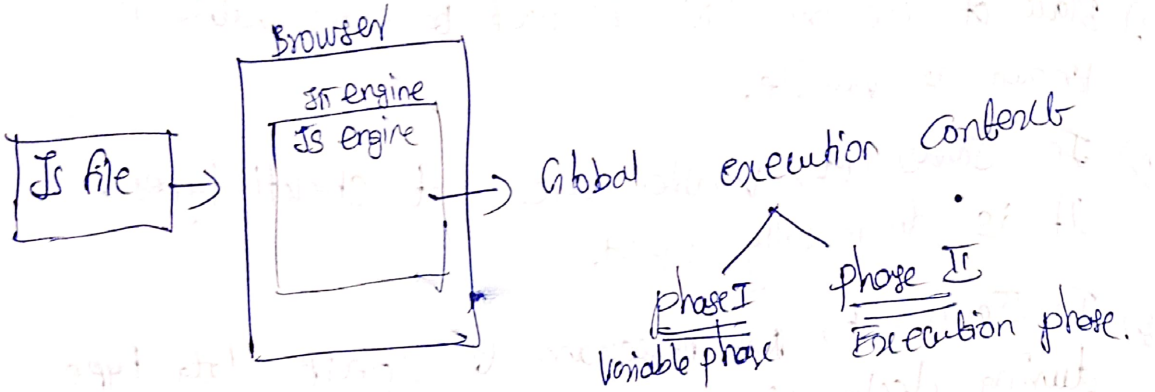
5.) It is mandatory to declare variable before using it.

Difference between var, let, const

var	let	const
dec → var love ;	dec → let love ;	dec → const love ; ✗
init → love = 'priya' ;	init → love = 'priya' ;	init ✗ love = 'Priya' ; ✗
re-init → love = 'manju' ;	re-init → love = 'manju' ; ✓	re-init ✗ love = 'manju' ; ✗
re-dec → var love ;	re-dec ✗ let love ; ✗	re-dec ✗ Const love ;
re-dec, re-init → var love = 'ramya' ;	re-dec, re-init → ✗	re-dec, re-init → Const love = 'ramya' ;
Console.log(love) ;		

	var	let	Const
Decler	✓	✓	✗
initia	✓	✓	✗
Re-inti	✓	✓	✗
Re-dec	✓	✗	✗
Re-dec & Re-init	✓	only once	✓

Execution of Javascript code



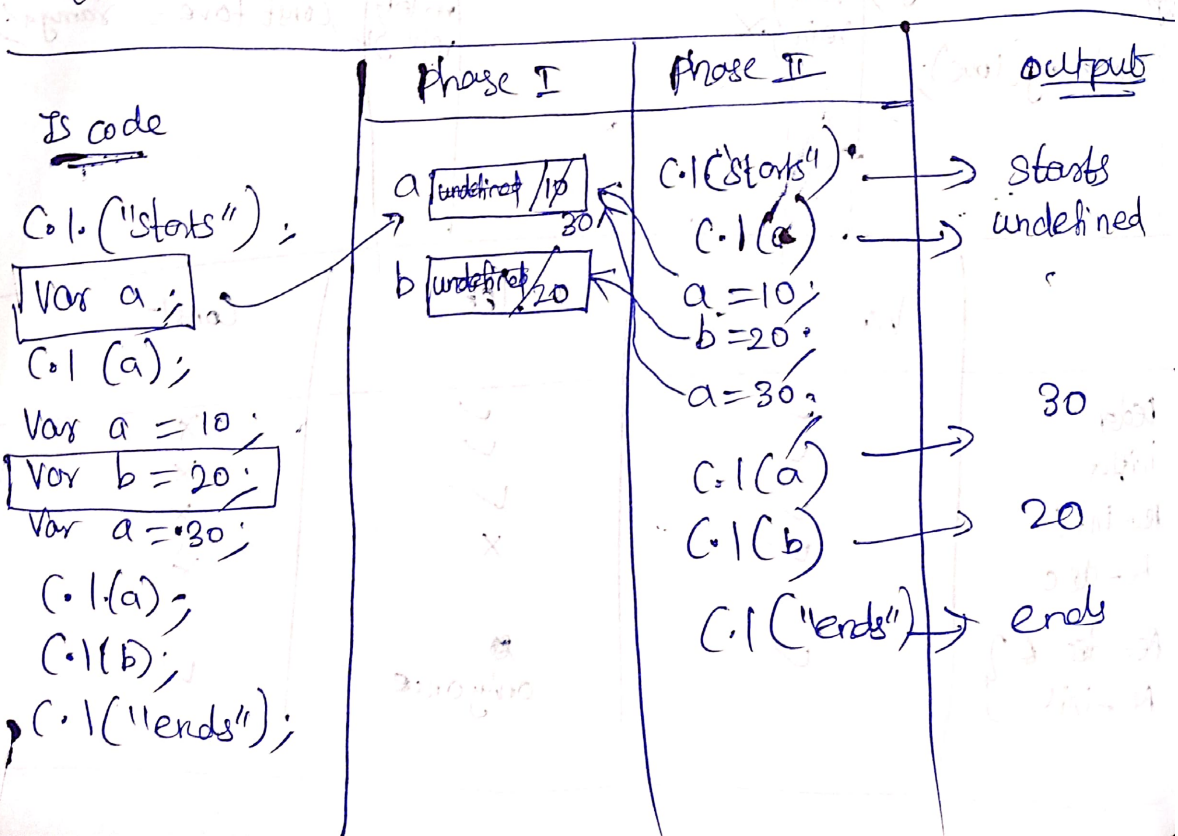
JS codes given to Javascript engine (browser), generally it follows two phase to execute JS code.

phase I - Variable phase:

All the memory allocated for the declaration in top to bottom order and assigned with default value undefined in variable area of Global execution.

phase 2 - Execution phase:

All the instructions get executed in the top to bottom order. In the area of Global execution context.



Global Execution Context

JS code	Phase I	Phase II	Output

C.l.("start");
var a = 20;
C.l(a);
var b = 40;
C.l(b);
var a = 40;
C.l(a);
C.l(a);
C.l(b);
C.l("ends");
	a undefined/20/40
	b undefined/40
		C.l.("starts") → starts
		a = 20
		C.l(a) → 20
		b = 40 → 40
		C.l(b)
		a = 40
		C.l(a) → 40
		C.l(a) → 40
		C.l(b) → 40
		C.l("ends") → ends	

Absolute path:-

It will start from parent node

Relative path:-

It will start from any of the child node.

Various ways to create String:-

1.) Single quote → Eg: 'operation';
2.) double quote → Eg: "This is Saravanan";
3.) Back ticks → Eg:- `how are you?` I am good`;
4.) new keyword → Eg:- var name = String("John");

String methods or String member functions:-

There are 40 member functions in String.

Some are:

1) length;
2) charAt(index no);
3) .concat(" ");
4) .includes(" ");
5) .startsWith
6) .endswith
7) .indexof
8) .repeat
9) .replace
10) .slice
11) .match
12) .split
13) .substring
14) .toUppercase
15) .toLowercase
16) .trim
17) .trimstart
18) .trimend
19) eval
20) eval(num.toS

alert, confirm, prompt, document.write, document.writeln

(1) alert:

alert("Hi, welcome back");

(2) confirm:

confirm("Hi, are you ok");

OK	or	Cancel

(3) prompt:-

var username = prompt("Enter your username"); // Saravanan

C.l(username) // Saravanan.

Everything taken as string in prom

(4) console.log("Hi, this is console"); // Hi, this is console.

UI: User Interface - whatever user seeing in webpages is user interface

(5) document.write("Hi") // it will display in webpage Hi

(6) document.write(<h1>"Hi"</h1>) // It will shown in h1 size

(7) document.writeln("Hi"); // Hi Hi

(1.) Read a text from the browser using a prompt and display no. of characters present in a text.

```
Var name = prompt("Enter your name");
Var Result = name.length;   Spiderman
console.log(Result) // 9
```

(2.) Read a text from the browser using prompt and print back the text on the console in uppercase.

```
Var username = prompt("Enter username");   Saravanan
Console.log(username.toUpperCase);
```

(3.) merge two text:

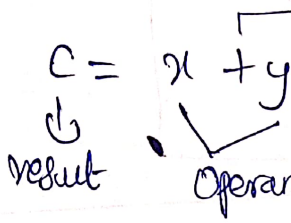
```
Var fname = prompt("Enter your First name");
Var lname = prompt("Enter your last name");
Console.log(`Full name : ${fname} ${lname}`);
```

(4.) write a JS code to store following details
1.) name, 2.) age, 3) gender, 4.) degree. and display in console.

```
Var name = "Saravanan";
Var age = 25;
Var gender = "male";
Var degree = "B.E";
Console.log(`Name : ${name}
             age : ${age}
             gender : ${gender}
             degree : ${degree}`);
```

Operator.

Operators are the predefined Symbol which are performers some specific task



Types of operator

1.) Aithematic → performs arithematic operation
2.) Assignment → Assign value to variable.
3.) Relational or Comparision → Compare value and/or type
4.) logical → Combines expressions & makes desis.
5.) Conditional → Evaluate true/false based on condita

① Aithematic operator.

operator:	description	example	result
(+)	Addition	x=5 x+5	10
(-)	Subtraction	x=5 x-5	0
(*)	Multiplication	x=5 x*5	25
(/)	Division	x=5 x/5	1
(%)	modulus	5%2	1
++	Increment	x=5 x++	6
--	Decrement	x=5 x--	4

(2) Assignment Operator:

Operator	Example	Expression m=15	Result
+=	m+=10	m=m+10	25
-=	m-=10	m=m-10	5
/=	m/=10	m=m/10	1
%=	m%=10	m=m%10	5
*=	m*=10	m=m*10	150

(3) Relational or Comparision Operator:

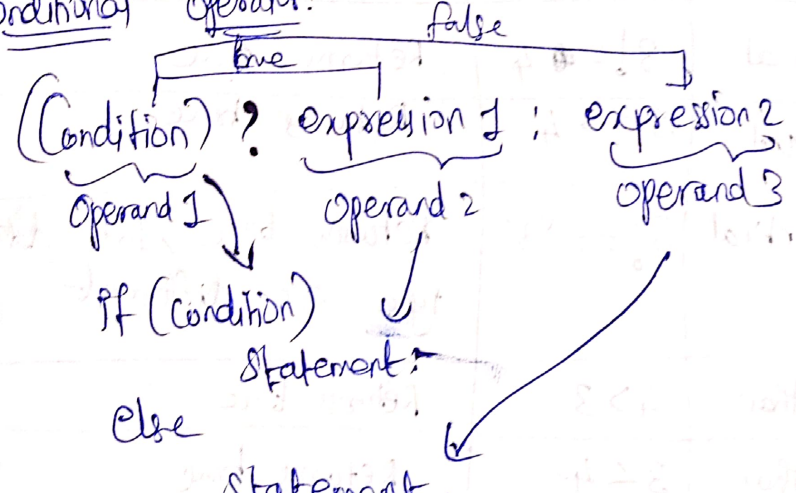
Operator	Name	Example	Result
==	Equal	$x==$y 3==3 3=="3"	Returns true Returns true
===	Identical	3 == 3 3 == "3"	Returns true Returns false, both are different data type
!=	Not equal	3!=4	Returns true
<>	Not Equal	3<>4	Returning true
!==	Not Identical	3!=="3"	Returns true, both data types are different
>	Greater than	4>3	Returns true
<	Lesser than	3<4	Returns true
>=	Greater than or Equal to	4>=4	Returns true
<=	Lesser than or Equal to	5<=5	Returns true.

④ Logical operators

Operator	Description	Examples
&&	and	x=6 y=3 (x<10 && y>1) returns true
||	or	x=6 y=3 (x==5 || y==5) returns false
!	not	x=6 y=3 !(x==y) Returns true

A	B	A && B	A || B	!A
true	true	true	true	false
false	true	false	true	true
true	false	false	true	false
false	false	false	false	true

⑤ Conditional operator:



(6.) Typeof operator:-

The typeof operator is used to get the data type (returns a String) of its operand. The operand can be either a literal or a data structure such as a variable, a function, or an object. The operator returns the data type. S.

Example	Return
typeof "hello"	String
typeof true	boolean
typeof false	boolean
typeof 100	number
typeof function() {}	function.
typeof null	object
typeof {}	Object
typeof []	object / Array
var a; typeof a	undefined

Scope

It is a visibility of a member function.

Ex:

```
var a;
a=10;
{
    let b = 20;
}
```

var a; — global scope

let b = 20; → block scope

block Scope:

It consists of statements & expression.

Eg 1:-

```
var a;
a=20;
{
    c.l (a);  ⟶ 20
    var b = 30;
    c.l (b);  ⟶ 30
}
c.l(b);  ⟶ 30
```

var accessible inside block

var accessible outside of block so it is called global scope

Eg 2):-

```
let a;
a = 20;
{
    c.l (a);  ⟶ 20
    let b=30;
    c.l(b);  ⟶ 30
}
c.l (b);  ⟶ no output
```

let accessible inside block

Let cannot accessible outside of block, so it is called block scope

Eg 8:-

```
const a = 20;
{
    console.log(a);   ---> 20
    const b = 30;
    c.l.(b);          ---> 30
}
c.l(b);               ---> no output
```

Const accessible inside block.

Const cannot accessible outside of block, so it is called block scope.

Note:-

Var - global scope | let, const } block scope

Note:-

```
Var a;
a = 10;
```

The time b/w declaration & initialization is Temporal dead zone

Note:-

```
Var a;
console.log(a);
```

→ accessing variable before declaring value is hoisting.

Global Scope:-

1.) It has the highest visibity to access anywhere

Block Scope:

1.) Visibility of member is only within the block where it is declared.

2.) member with block scope, we can use only inside the block, where it is declared. It cannot be used outside.

Type Coersion / Type Casting

The proces of converting one datatypes to another data type is called type coersion or type Casting.

It has two types:

1.) Implicit type Casting:

The proces of converting one datatype to another data type by using Javascript engine is called Implicit type casting.

Eg:-

(1) C.l(5-"4") // 1 → string converted to number implicitly by JS engine.

(2) C.l((5)+ "4") // → 54

number is converted into string Implicitly — concatination happening

(3) C.l(5-"a"); — (Hw.)

2.) Explicit type casting:

The proces of converting one type of value to another type of value is known as Explicit type Casting.

Eg:-

```
Var username = prompt("Enter username");
```

75621

```
Var name = Number(username);
C.l (name); // 75621
```

explicit type casting.

Implicit type casting:-

```
Var e = 10 + "20";
C.l (e, typeof e); 1020 string
```

```
Var e = "10" + 20;
C.l (e, typeof e); 1020 string
```

```
Var e = 10 + 'a';
C.l (e, typeof e); 10a string
```

```
Var e = 'a' + 10;
C.l (e, typeof e); a10 string
```

```
Var e = 'a' + true;
C.l (e, typeof e); atrue string
```

```
Var e = 1 + false;
C.l (e, typeof e); 1 number
```

write a program to get a character from user using prompt and the character from the user slice the word character

Start value is 4, end value is 7.

```
Var name = prompt("Enter Saravanan's Username");
Console.log(name.slice(4,7));
```

o/p = Van

---

write a program to get a user information from the prompt. Convert it to lower case and uppercase.

Saravanan.

```
Var username = prompt("Enter your name");
Var Result = username.toLowerCase();
console.log(Result);
```

o/p = saravanan,

---

write a program to Concat two string by using promt.

Tamil

```
Var Firstname = Prompt("Enter first name");
Var lastname = prompt("Enter Selvan's last name");
Console.log(Firstname.Concat(lastname));
```

o/p = Tamil/selvan.

0 1 2 3 4 5 6 7 8 9
good night

Substring(1,5) ood-

Substr(1,5) ood-n

Javascript

5-4-3-2-1

Slice(-3)

Slice(5)

ipt

Note:-

Substring() is similar to slice()

The difference is that Substring() cannot accept negative indexes

Substr() is similar to slice() and substr() is depreciated and will not accept negative values.

Write a Javascript code to log Sum of two numbers on the Console by reading the numbers from the browser prompt.

```
var num1 = Number(prompt("Enter first number"));   // 50
var num2 = Number(prompt("ENter second number"));  // 50
var result = Number(num1) + Number(num2);
Console.log(result); //100.
```

(2) Read a name of the fruit from the prompt, then read single character from user. / check the the character is present in fruit or not.

Apple

```
var fruitname = prompt("Enter fruit name");
var char = prompt("Enter character");

var Result = fruitname.Includes("e");
```

~~Consolelog((Result == true)?~~ ~~char "is present" : char is not present~~

```
Console.log((result == true)? char + "is present":
                              char + "is not present"));
```

String method :-

1.) In Javascript, String is a object (it is immutable object).

2.) Therefore, String objects will have State (variable) and behaviours (function).

3.) We can use this variable and function of the object with the help of dot operator.

4.) Ex.) To find length of string, we have length function (method) which provides length of the string.

```
var Str = "hello";
var result = Str.length;
Consol.log(`Str have ${result} characters`);
```

write a program to check your eligible for going Pub

```
var age = Number(prompt("Enter your age"));
if (age > 18)
{
    console.log("go to pub");
}
else
{
    console.log("stay in home");
}
```

---

write a program you are about to go for department store if the rice is more than 20 Buy rice & Sugar is more than 40 avoid Sugar - if both of them is not present in departmental store go for next shop.

```
var rice = 30;
if (rice > 20)
{
    console.log("buy rice");
}
else if (sugar > 40)
{
    console.log("avoid sugar");
}
else
{
    console.log("go for next Shop");
}
```

```
if (Rice >20 && Sugar <40)
{
    Console.log ("Buy Rice")
}
else if (Rice > 20 && Sugar > 40)
{
    Consol.log ("avoid Sugar");
}
else
{
    Consol.log ("go to next shop");
}
```

Write a javascript program to check whether the integer is greater than 100 or less than 100 or equal to 100.

```
Var num = Number(Prompt("Enter integer number"));
if (num > 100)
{
    Console.log("greater than 100");
}
else if (num < 100)
{
    Console.log("less than 100");
}
else if (num == 100)
{
    Consolo.log("equal to 100");
}
else
{
    Console.log("Enter number properly");
}
```

Switch

```
Var fruits = prompt("Enter fruit name");
Switch (fruits)
{
  Case "Apple" : {Console.log("Apple is 300Rs");}
                 break;
  Case "orange" : {Console.log("orange is 200Rs");}
                  break;
  default : {Console.log("go for another Shop");}
}
```

Write a Javascript program to find the grade of a Student using Switch Case

```
Var marks = Number(prompt("Enter your marks"));
Switch (marks)
{
  Case (marks > 90) : { Console.log("A grade")
                      } break;
  Case (marks > 80) : { Console.log("B grade")
                      } break;
  Case (marks > 70) : {Console.log("C grade");
                      } break;
  Case (marks > 60) : { Console.log("D grade")
                      } break;
  default : { Console.log("failed")
            }.
}
```

Decision Statements

It helps to skip a block of instruction when we dont have favouring situation

decision statements of JS.

1.) if
2.) if else
3.) else if
4.) Switch

(1) if

```
if(condition)
{
  ==
}
```

(2)

```
if (condition)
{
  ==
}
else
{
  ==
}
```

(3)

```
if (condition)
{
  ==
}
else if
{
  ==
}
else
{
  ==
}
```

(4)

```
Switch (expression)
{
  Case 1 : { == }
           break;
  Case 2 : { == }
           break;
  default : { == }
}
```

Switch

1.) A Case block gets executed with the value positive Switch matches the value present in the case

2.) When Case is favourable Case block gets executed as well as all the block present in the Switch gets executed

Break.

(1) It is a control transfer statement

(2) It can be used either switch block or loop block only. when the break statement is executed or when the break statement is encountered then the control is transfer outside that particular block. it will terminate the loop and

---

(1) write a program to get a number from user and print all the numbers line by line

```
var start = 0;
Var end = Number(prompt("Enter the number"));

while (start < end)
{
    start = start+1;
    Console.log (start);
}
```

Iteration	variable	condition	Output	break
(1)	start = 0 end = 10	start < end 0 < 10	start = start + 1 0	1
(2)	start = 1 end = 10	1 < 10	start = 1 + 1	2
(3)	start = 2	2 < 10	start = 2 + 1	3
(4)	start = 3	3 < 10	start = 3 + 1	4
(5)	start = 4	4 < 10	start = 4 + 1	5
(6)	start = 5	5 < 10	start = 5 + 1	6
(7)	start = 6	6 < 10	start = 6 + 1	7
(8)	start = 7	7 < 10	start = 7 + 1	8
(9)	start = 8	8 < 10	start = 8 + 1	9
(10)	start = 9	9 < 10	start = 9 + 1	10
(11)	start = 10	10 < 10 false		

do-while

```
do {
    start = start + 1;
}
while (start < end).
```

iteration	variable	Output	Condition.	C.I.
(1)	start = 0. end = 10		start = start+1 start = 0+1	1

Iteration	variable.	Condition.	C.I	while
(1)	start = 0 end = 10.	start = star		

write a Javascript to find the sum of positive numbers only. to

```
Var num = Number (Prompt ("Enter the number"));
Var start = 0;
if(num > 0)
{
    while (start < num)
    {
        start = start + 1;
        Console.log (start);
    }
}
```

Sum of numbers

```
Var num = Number ( prompt("Enter the number "));
Var start = 0;
Var Sum = 0;
if (num > 0)
{
    while (start < num)
    {
        Start = Start + 1;
        Sum = Sum + Start;
    }
    Console.log (Start);
}
```

H/w find

① JavaScript Version — Ecmascript 2022 (ES13)

After ② Sum — prime numbers of a upto a number

Program 1 Print ① to 10

```
Var num = 10;
Var Start = 1;
while (Start <= num)
{
    Console.log ("iteration:" + start);
    Start++;
}
```

Program 2 Breake on 3rd iteration.

```
Var num = 10;
Var Start = 1;
while (start <= num)
{
    console.log ("iteration:" + Start);
    if (Start == 3)
        Start++; Console.log ("Iteration" + Start); breake;
}
```

```
var num = 10;
var start = 1;
var iter = 1
while (start < num)
{ if (start == 3 { break; }
    console.log ("iteration :" + start);
} start++;
```

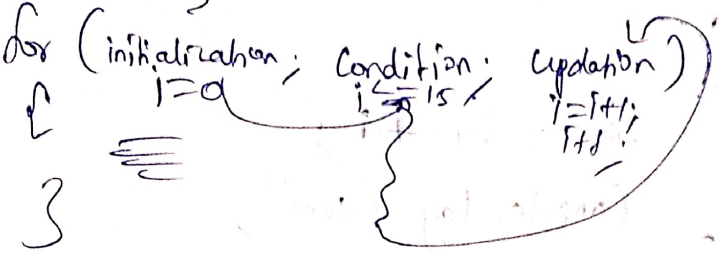
iteration	Variable initialization	Condition	Output	Increment
1	start = 1 num = 10	start < num 1 < 10	iteration : 1	start++ 2.
2.	start = 2 num = 10	2 < 10	iteration : 2	start++ 3

using for loop

Write a program to get a number from prompt and iterate value which is given by the user.

Syntax:

```
for (initialization; Condition; updation)
{
}
```



Program (1)

```
Var num = Number(prompt("Enter the number"));
for (i=0; i<=10; i=i+1)
{
    Console.log(i);
}
```

15

Program (2)

```
Var num = Number(prompt("Enter the number"));
Var start = 1;
for (i=num; num>[illegible]; i--)
{
    Console.log(i);
}
```

```
var num = 10;
var sum = 0;

for (i=1; i<=num; i++)
{
    sum = sum + i;
    console.log (sum);
}
```

```
var name = "tamil";

for (i=0; i<=(name.length); i++)
{
    console.log (name.charAt(i));
}
```

```
while ( )
{
    while ( )
    {
        if { break }
    }
}
```

Function

Prototypes

Function Statement

Function Expression.

Write a Function Sum of 2 digits number :-

```
Var num1 = Number(prompt("Enter 1st number"));
Var num2 = Number(prompt("Enter 2nd number"));

function Sum (a, b) {

    Var c = a + b;
    Console.log(c);
}

Sum (10, 20);
```

Parameter → It holds the value of argument

argument = It holds data literals

Function.

1.) Function is a block of Instruction which is used to perform some specific task.

2.) Advantage.

When function is executed only when the function is called function()

3.) Function helps the developer to achieve code reusability

Note:-

In Javascript functions are beautiful. every function is nothing but an object.

Syntax to create function:

(1) function statements or declaration.

```
function Identifier (parameter1, parameter2)
{
    Statements
}
Identifier (argument1, argument2);
```

(2) function Expression

```
Var add = function Identifier (para1, para2) {

}
add (arg1, arg2);
```

There are two ways to create function.

1.) function declaration or Statement

2.) function expression.

* Function is a object. name of the function is variable. which holds the reference of object.

Parameter :

1.) Variable declared in the function definition is known as parameter

2.) Parameter have local scope.

3.) The parameter are used to hold the values passed while using a function.

argument :

1.) Values passed in a method call the Call statement is known as argument.

Note :-

An argument can be a literal or variable. or expression, which gives result.

Eg:

```
add (a, b); — variables (or) expression.
add (10, 20); — literals
```