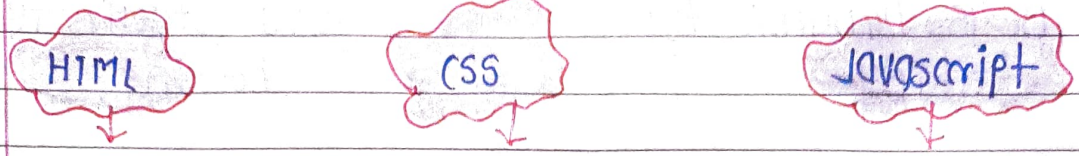
Javascript

Camlin | Page
Date / /



HTML	CSS	Javascript
hyperteaet Markup language	cascading style sheet	dynamic Nature to Webpage.

Javascript

- Javascript is high level programming language (scripting)
- used to make Webpage intractive.
- javascript it is execute on dient machine.
- It is powerfull client side scripting language.
- JS case-sensitive, started in 1995 first version 1.0 (Netscap Navigator 2.0)

Difference

Javascript	Java
1) object orientended scripting language	object oriented progra--ming language.
2) It run only browser	java run on browser & Virtual machine as well
3) In JS code is timer compiler before execution but ~~java compile be~~ before execution	java compile before execution.

In JS Dynamic type checking	java static type checking.

Dom - Document Object Model - (When webpage is loaded)

→ Dom is an api interface provided by browser

→ When Webpage is loaded the browser create a document object Model of the page.

→ With dom java script can access and change all elements of an HTML document.

→ JS can not understand directly HTML. (Interact with HTML using DOM)
↓
Dom create it automatically
↓
at the time of browser loading.

→ JS interacts with HTML using DOM.

→ javascript compatiable with CSS and HTML

Tools -

1) Text Editor - e.g visual studio code.

2) Browser - All browser comes with inbuilt JS engine.

* console.log - used to print out & Mostly for debuging.

→ console.log / document.write.

→ // single line comment , /* multiple line.

Javascript Variable - (let, char, const)

→ Variables where we store the value.

→ which holds the value while the java program executed.

→ variable assigned with data type, name of memory location.

→ let, char, const these are the keywords used to store the variables. (value)

→ Two types of variable local & global.

→ local - local variable declared inside block/fun^n

↓

→ e.g function abc() {

inside the fun. var x= 10; // local

}

→ global - declare outside the function.

↓

var data = 2000 / global.

outside the fun a() {

fun. document.write(data);

}.

	let	Var	const
	↓	↓	↓
1)	block local scoped	out - global side scoped / fun scoped inside Particular fun	local block scoped. declare const vle need to initialize it.
2)	can't Reassing	Reassing theval.	cant chage value
3)	can't Redeclare variable	Redeclare variable.	can't Redeclare.

Keywords -

- Keywords are reserved words that are specific to diff programing language.
- Keywords are predefined words.
- e.g const a = "Hello" here const is a keyword that denotes that a is a constant.

Datatypes -

→ Datatypes - value stored in variable.

→ Two types of datatype primitive and non-primitive.

- primitive → Inbuit

Nonprimitive → Userdefined.

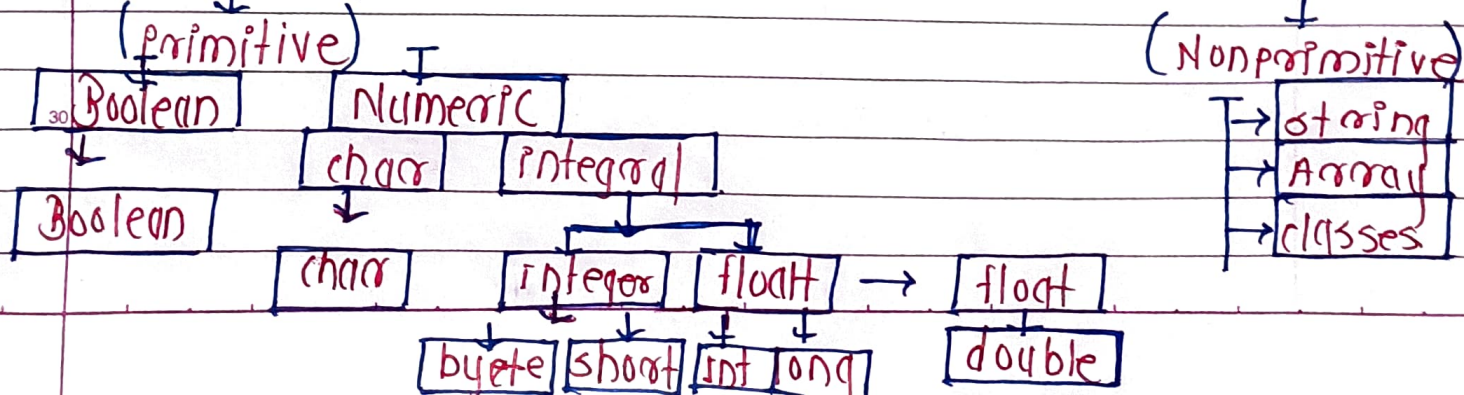
* **javascript primitive -**

Primitive	String →	Represent sequence of char predefined class, act as datatype.
	Number →	Represent Numeric value
	Boolean →	Represent boolean value. gives val. true/ false.
	undefined →	Represent undefined value.
	Null →	Represent Null value

* **javascript Non primitive -**

Non primitive	Object →	Represent instance which can access
	Array →	Represent group of similar value
	RegExp →	Represent regular expression.

* Everything in javascript is object.



Javascript operators -

→ Operators are symbol that are used to perform operations

① Airthmetic - To perform arithmetic operations

mathmatical operations

+	Addition	10 + 20
−	substraction	20 − 10
*	Multiplication	10 * 20
/	Division	20/10
%	Modules	20 % 10
++	Increment	Var a=10; a++, Now a=11
--	Decrement	Var a=10; a--, Now a=9.

② Relational / comparision operators -
To compaires the two oprands.

compare two operands :

==	equal to	10==20	false
===	Identical	10===20	false
!=	Not equal	10!=20	true
!==	Not Identical	20!==20	false
>	greater than	20>10	true
>=	greater than/equlto	20>=10	true
<	less than	20<10	false
<=	less than equalto	20<=10	false.

③ Javascript bitwise and logical operator -

&	AND	(10==20 & 20==33)=false
|	OR	(10==20 | 20==33)=false
^	XOR	(10== ^ 20=33)=false
~ !	NOT	(~10) = -10
<<	left shift	(10<<2) = 40
>>	right shift	(10>>2) = 2
>>>	right shift with zero	(10>>>2) = 2
?	ternary	

(if both cond^n.)

logical -

&&	logical AND	
||	logical OR	
!	logical NOT	

check 1st & 2nd if 1st false.

④ Javascript assignment operators -

=	Assign	10+10=20
+=	Add & assign	var a=10; a+=20; a=30
-=	sub & assign	var a=20; a-=10; a=10
*=	mut & assign	var a=10; a*=20, a=200
/=	Div & assign	var a=10; a/=2; a=5
%=	modulus & assign	var a=10; a%=2 a=0

To assing value.

⑤ special operator -

(?:)	based on cond^n
,	comma operator
delete	
in	if object has given property
instanceof	object is an instance
New	create an instance.
typeof	check type
void	It discards the exp
yield	checks what is returned.

Block -

A block is a group zero / more statements between balanced braces {} can be used anywhere a single statement is allowed.

```
class blockdemo {
    public static void main (String [] args) {
        {
        }
    }
}
```

Control Statement -

→ Java provides three types of control flow statements.

(1) Decision Making statement.
   1) if statments.
   2) switch statement
(2) Loop statements
   1) do while loop
   2) while loop
   3) for loop
   4) for-each loop
(3) Jump statements.
   1) break statement
   2) continue statement.

① Decision-Making-statements-

→ decide which statement to execute and when.

IF & SWITCH.

1) Simple if statement -

- most basic statement among all control statement

Simple if cond^n then statements

```
if (condition) {
    statement 1; // execute when condition is true.
}
```

2) if else statement -

is an extension to the if-statement, use another block of code i.e else block.

else block execute when if condition is false

When if not true/execute:

```
if (condition) {
    statement 1; // execute when cond^n true
}
else {
    statement 2; // when false.
}
```

3) if else if -

multiple elseif statements.

multiple elseif

```
if (condition 1) {
    statement 1 // executes when cond^n true
}
else if (condition 2) {
    statement 2 // true.
}
else ~~{condition 3~~ {
    statement 2;
}.
```

Switch ↓

- This are similar to the if-else-statements.
- The switch statements contain multiple block of code.

```
Switch (expression) {
    case value 1:
     statement 1;
      break;
      .
      .
      .
    case valueN:
    statement N;
     break;
     default:
     default statement;
     }
```

(Same as if-else-statement switch statement contain multiple block of code ↓ Transfer program flow)

transfer program flow ↓

---

② looping statement. - loop statement used to execute set of instruction in repeated order.

1) do while loop

↓

do while loop check the condition at the end of the loop after executing the loop statement.

```
do
{
// statements.
} while (condition);
```

(condn at the end.)

condn q at end ↓

2) While loop-

- While loop is used to iterate over the number of statements multiple times

iterate no of statement

while condition then statement

```
while (condition) {
    // looping statement
}
```

3) Javascript for loop-

- check the condn and increment / decrement.
- We use for loop only when exactly known the number of times, we want to execute the block of code.

initialization, condition, increment & decrement statements

```
for (initialization, condition, increment/decrement) {
    // block of statement
}
```

4) for each loop-

- Javascript provides an enhanced for loop to transvers the data structure like array / collection.

enhanced for loop to transver the data structure

```
for (data.type var: array_name/collection_name) {
    // statements.
}
```

ee used to transfer the ,, control

Camlin Page
Date / /

Jump — transfer the control

③ Java ~~break~~ statement - Used ~~break the current flow~~ of program.

① break -

break statement is used to break the current flow of program and transfer the control to the next statement.

- It can be used inside the loop.

to break current flow

```
if (){
    break;
    }
```

② continue-

continue statement doesn't break the loop.

- It skip specific part of the loop and jumps to the next iteration of the loop immediately

doesn't break flow. It skip specific part

Static data Members -

- Static data members are not part of any object as it is part of class instead all the object can use its value
- static data member keep seperatly and hence its value is unique for all

---

Function - Three Types -

function declaration -

```
fun calculator(x,y) {
    console.log(x+y)
    console.log(x+y)
}
calculator(120,40)
```

keyword, funname, parameter, parenthesis, statements, block, call, argument

function expression -

```
let cal2=fun (a,b) {
    console.log(a+b)
    console.log(a-b)
}
cal2(100,101)
```

arrow function -

```
let cal3 = (i,k)=>{
    console.log(i+k)
    console.log(i-k)
}
cal3(12,6)
```

can remove block & return keywords

"fun is an block of code which is used to perform perticular task"

Camlin | Page
Date / /

Function in Javascript -

→ Javascript function are used to perform operation

→ To many times to reuse the code.

→ code reusablity
less coding

Syntax - keyword (function) | fun Name | parenthesis | parameter.

```
function functionName ([arg1, arg2, ... N]) {
    // code — block statements
}  ← curlybrace
```

① function without argument / parameter with return

```
function msg () {
    console.log ("hell! this is msg");
}
```

function without argument with return

```
fun add (a)
{
return ("Hell")
}
console.log (add).
```

② function with argument (parameter) without Return.

```
function getcube (arg) {
    console.log ("number")
}
add (10) // call value.
```

function with argument (parameter) without Return

```
fun add (a,b) return
{
console.log (a+b)
}
add (100, 200);
```

parameter.

③ function without argument without return type.

function functionName() { fun add () {

return "hello"; } console.log("Hello").

console.log(funname); } add();

parameter

④ function with argument with return type.

function add(a,b) { fun add(a,b)

return (a+b); { return (a+b);

} }

console.log(add) let res = add(10,20)

console.log(res)

Need to store that val in variable.

---

* Javascript function object

function constructor to create New fun object

The Pempose of function constructor is to create a new function object.

Syntax - new Function([arg1[, arg2[....argn]], function Body).

Method -

call fun ① apply() - used to call fun this val & single argument

create fun ② bind() - used to create new function.

call fun ③ call() - used to call function this val & argument list

return string ④ Tostring() - it returns the result in form of string

Object in Javascript -

- A javascript object is an entity having state and behavior (properties and method).
- javascript is an object-based language. everything is an object in javascript.
- javascript is template based not class based

* ~~Three ways to define~~ Object -

e.g let person ← objname

property name { firstname : "ABC", lastname : "xyz", age : 50, Weight : 60 }; ← value seprate by colon & ends with semicolon. ← value

// Accessing value

Two types • notation & bracket notation

→ Bracket → console.log (person["firstname"])
→ • Notation → console.log (person.age)

// Add New property to object

Adding

person["height"] = 5.5; / person.height = 5.5

console.log (person["height"]);

// update existing property

updating

```
person["weight"]=65;
console.log(person["weight"]);
```

Remove

// Remove property from object

```
delete person["age"]
console.log(person["age"]);
```

// for/in loop looping object

```
for(let x in person)
{
  console.log(x); // print only property name
console.log(person[x]); // print only pro val.
console.log(x + " " +person[x]);
}
```

Method : This. (keyword represent the object)

↓

e.g

```
let employee=
{
empname:"scott",
empid:1023,
job:"Engineer",
basicsal:50000,
bonus:function()
{
return(this.basicsal)*10/100
}
};
console.log(employee.bonus());
```

object

→ ends with semicolon

→ value separate by colon.

Method.

this keyword represent object

Camlin Page
Date / /

Array in Javascript - collection of similar type of element

- array is an object that represents a collection of similar type of elements.
- used to store multiple values in single variable.
- It is also special variable which can hold more value at a time.

e.g let car = ["Indica", "ertica", BMW"]

e.g

```
let car = ["Sab", "ertica", "BMW"]
console.log (cars);
console.log (cars[2]); // accessing element
car[0] = "Opel"; // change the val.
console.log (cars);

let myarray = [100, "Welcome", 10.15, true]
console.log (myarray);

// We can have objects in an array.

let person1 = {
              name: "KK",
              age: 30.
              };
let person2 = {
              name: "Kiran"
              age: 40
              };
```

```
let myarray1 = [person1 , person2 ];

    console.log (myarray1);      // all
    console.log (myarray1[0]);   // 1st
```

length property

```
let fruit = ["Banana", "orange", "Apple",
                                 mango" ];
console.log (fruits.length) : 114.
```

```
// looping elements from array
```

for loop

```
for (let i=0; i<= fruits.length-1; i++)
{
    fruit [i]
}
```

```
// looping elements from array using for/of
                                  loop
```

for of loop

```
for (ele of fruit)
{
console.log (ele);
}
```

Recognize.

```
// Recognize an array - typeof, IsArray

console.log(typeof fruit);   // object  array is an object
console.log (array.IsArray (fruits)  true
                                      ↓
                                  because this is an
                                  array variable
```

Array Method -

- Array is an object it contains method.

e.g

```
let fruits = ["Banana", "Apple", "mango", "orange"]
```

To manupulate	New array	An array index
Add to original.		
push (end)	map (loop)	indexof based on val
unshift (start)	filter using cond	
	filter	findindex. based on cond
Remove from original		
pop (end)	portion of	
shift (start)	slice original.	
splice (any)		
	concat Adding origna.	
others.		An array element.
reverse	flat } filter the element	find.
sort		Based on test cond
fill		
	flatmap	

add

① push() - add element at end of array

② unshift() - add element at begning of array.

Remove

③ pop() - It remove and return last element

④ shift() - It remove and return first element

⑤ ~~splice()~~ - ~~It returns new array part of given array~~ It add/remove elements.

Other

⑤ ~~slice~~sort() - In returns the sorted order

⑥ fill() - fill the elements in array

⑦ reverse() - It reverses the elements of give, array.

return new array

⑧ map() - It return new array

⑨ filter() - It return new array, (filter element)

⑩ slice() - It return a new array (part of given array)

merge

⑪ concat() - return new array object / merged arrays.

new array

⑫ flat() - It create a new array carrying sub array

⑬ flatmap() - It maps all array elements. result new array.

based on value

⑭ indexof() - It searches the specified element in the given array. return index.

⑮ findindex() - It return index value of 1st element.

⑯ find() - It return the value of the first

Date Constructor in javascript -

e.g.

dateobject -

```
let d= new date();

console.log (d);
console.log (d.getDate());
console.log (d.getMonth()+1);
console.log (d.getfullyear());

console.log (d.getDate()+" "+d.getMonth()+1+" "+d
.getfullyear());
```

String and Numbers in javascript

String - Javascript string is also an object. that represents a sequence of characters.

Sequence of characters

1) By string literal
2) By string object

Types of string Methods.

1) charAT() - It provide char value.
2) charCodeAT() - It provide unicode values.
3) concat() - it provide combination of two / more strings.

4) IndexOf() - It provide the position of a char value present in the given string

5) lastIndexOf() - It provides the position of char value present in the given string by searching character

6) Search() - It searches a specified regular expression in a given string and return position of match.

7) Match() - It searches a specified regular expression in a given string

8) Replace() - It replace a given string with the specified replacement.

9) substr() - It is used to fetch the part of the given string on the basis of the specified starting position & length.

10) substring() - It is used to fetch the part of given string (specified index)

11) slice() - It is used to fetch part of given string It allows us to design positive as well negative index.

12) toLowerCase() - It converts the given string in to lowercase

13) toUpperCase() - It converts the given string into uppercase.

14) **Tostring()** - It provide a string representing the particular object

15) **Valueof()** - It provide the primitive value of string object.

To split

16) **split()** - It split a string into substring array, then return that newly created array.

return new array

17) **trim()** - It trims the ~~white~~ space from left and right side of the string

To trim the space.

If you know these five "fundamentals" you can learn almost anything in programing

Variables - are containers for storing data (values)

Array - used to store multiple values in a single variable.

loops - loops can execute a block of code a number of times

object - object is an entity which have the methods and properties

function - a function is a block of code designed to perform particular task.

function is executed when somethings calls it.

Camlin Page
Date / /

OOPS concept in javascript -

- OOPS - object oriented programming language.
- OOPS is the concept of abstract classes and interfaces.
- In object oriented programing language.

class
object
constructor method.
static Method.
Encapsulation.
Inheritance
polymorphism
Abstraction

} this four are the key principle.

① Class - class is an logical entity which ~~defin~~ contains variables & methods.

- class also contain constructors.
- class is an special type of functions.
- We can define the class like function
  1) declaration and
  2) expression
- The javascript class contains various class. members within a body including method / constructor.

1) class declaration -

- A class keyword is used to declare a class with any particular name.
- name of class always starts with the uppercase

2) class expression -

- Another way to define a class is by using a class expression.
- class expression can be named/ unamed
- The class expression allows us to fetch the class name.

② object - object is an physical entity which contains methods and property.

- object also contain constructor.
- object are required in loops because they can be created to call a non static function
- using new keyword we can declare a the object.

③ Methods- Method is a peice of code with will perform certain task..

- a method takes parameters.
- a method may not take parameters.
- a method returns some value
- a method may not return any value.

```
class test
{
void m1()
// code
}
}
```

④ constructor-

- A constructor is a special function that create and initialize an object instance of a class.
- Javascript, a constructor gets called when an object is created using the new keywords.
- The purpose of constructor is to create a new object and set values.

Constructor overloading - → special kind of method

- Constructor is special kind of method
- Constructor name should be same as class name
- constructor will not return any value
- constructor will be invoked at the time object creation.
- constructor will take parameters just like method
- constructor is used for initialize the value.

two types of constructor → we can create multiple main method but diff parameter.

① default constructor.
② parameterized constructor

~~this~~ ~~Static~~ keyword - specification

→ this keywords used to assing values.
→ which one is local & which one is explicit
→ When we are using same external variable names then we use this keyword

```
void sum (int a, int b)
{
    this.a = a;
    this.b = b;
    System.out.print(a+b);
}
```

→ To specify this variable

same method & constructor name.

Overloading - Same Method, Same Constructor na time

method overloading - a class contain ~~same~~ more than o method with Same name is called as Method overloading

Constructor Overloading - a class contain more than one constructor with Same name is called as constructor overloading

Method - overloading -

```
class Test                    ← class
{
void m1(int a, int b)         ← method
 {
 // code
 }
void m1(int a, int b)         ← Method
                  double
 {
 // code
 }
}
Test t = new test();          ← object
 t.m1();
```

- Method overloadin
  - number of param
  - oder of paramet
  - data type of para
  - meter
  - declare metho using this & call thos methods using object

- object always create in main method.

Camlin Page
Date / /

Static keyword -

- When we say static the variable and methods we can access directly with class without object.

```
static int a;
static void m1()
{
  // code
}
```

- static method can access only static stuff.
- static method can also access nonstatic. static. stuff but through object
- static method can access everything static & nonstatic direct access

final keyword -

→ final is a modifier which provide restriction

Camlin Page
Date / /

Encapsulation - Wrapping up of data und
single unit ↓

- The javascript encapsulation is a proce
binding the data with functions acting o
that data.

- It allows us to control the data and val

- use var keywords to make data members
private

- use setter method to set the data.
getter methods to get data.

- encapsulation allows us to handle an object
using

Read/ Write - setter Methods to write the data.
getter Method to read the data.

Readonly - In this case, we use gettermethod
Writeonly - In this case, we use settermethod

Inheritance - one class (methods & variables we
properties & behaviours
can use those in another class)

- The javascript Inheritance is a mechanism
that allows us to create new class on
the basis of already existing class.

- The javascript extend keyword is used to
create a child class on the basis of parent
class.

It maintain an Is-A Relationship.

Polymorphism

- **polymorphism** - Poly - many, morph - form. many forms.

- The polymorphism is a core concept an object - oriented paradigm that provides a way to perform a single action in different form.

- It provides an ability to call the same method on different Javascript object.

It is an ability to create a variable, fun, obj that has more than one form.

- **abstraction** -

- an abstraction is a way of hiding the implementation details and showing only the functionality to the users in other in other words.

- We cannot create an instance of abstract class.
- It reduces the duplication of code.

Inheritance

↓

Reusability

① single level inheritance
② multi level inheritance
③ ~~hierac~~ hierachy inheritance

Camlin Page
Date / /

* Overriding

- What ever methods we created in parrent class.
  recreate those in child class is called. Overriding.
- just modified parrent ⌒ child.

* prototype -

→ prototype is an object that is associated with every function & object by default

→ If we want to add New properties at later stage to a function / class we can take help from prototype.

Camlin Page
Date / /

Asynchronouse -

→ Asynchronouse simply it means of parallel programing in which a unit of work

→ Data may take long time to submit database.

→ With asynchronouse programing, the user can move to another screen while the function continues to execute.

e.g when photo is loaded & sent on instragram the user does not have to stay on the same screen waiting for the photo to finish loading.

(Settimeout function)

Synchronouse -

Synchronouse operation is performed only at a time & only when one is completed.

e.g In personal meeting
photo call
video conferance
coffee break conversation.

Javascript is always synchronouse & single threaded.

javascript promises -

→ promises are used to handle asynchronouse operation in javascript.

→ they are easy to manage when dealing with multiple asynchronouse. operations.

→ promises are the ideal choice for handling asynchrous operation in the simplest manner

→ Benifit -

Improves code readability
Better handing of asynchrouse operations.
Better error handling.

→ fulfilled } action related to the promise succeded
rejected }
pending } promise is still pending
settled } promise has fullfiled / rejected.

→ promise can be consumed by registering function . then method.

→ .then() is invoked when a promise is either resolved / rejected.

```
.then (function (result) {
    // handle success.
}, function (error) {
    // handle error
})
```

Camlin | Page
Date / /

JavaScript Promises - resolve() Method.

→ The promise is an object that represents either completion / failure of user task.

→ promise.resolve() method in JS returns a promise object that resolved with a given value.

```
promise.resolve(value);
```

```
<script>

var promise = promise.resolve(123);
  promise.then(function(val){
    console.log(var);
  });
```

o/p - 123.

Read JSON Data using Javascript

JSON is a data formate

first we need to convert JSON formate into javascript object
and from that object we can extract the data