Date / /

Javascript

- Javascript allows client-side scripting to create completely dynamic web applications and website.

⇒ It was designed for making pages alive.

⇒ With Nodejs, Javascript can be used as a backend language.

⇒ Javascript has no link with Java programming language.

Installing VS Code, Extension & Setup

- To download VS Code visit https://code.visualstudio.com/download.
- In VS Code, click on Extensions and download
  i) Live Server
  ii) Javascript ES6 code snippets

Run Javascript in Chrome Console.

- Right click on any page on browser and select inspect Element.
- Select the console option where we can run our Javascript code

- We will see the result of our JS code in Console.

Add JS to HTML

i) <Script> tag : Adding <Script> --- </Script> and writing the code.

ii) External JS : Adding a file with .js extension through link.

<Script src = "js/tut2.js"></Script>

Commands.

i) Console.log()

This is used to print output to the Console. We can put anything inside the log(). It can be array, object, string, boolean etc.

Eg. Console.log(Harry);

↳ Print Harry in Chrome Console

ii) Console.table()

It generate a table inside a Console. The input must be an array or an Object.

Eg. Console.table({name: "Harry",
language: "Javascript", tutorial: 2}
);

Output

Name	"Harry"
language	"Javascript"
tutorial	2

Console.assert()
This method writes a message to the console that the assertion failed and the message we provide as a parameter, but only if an expression evaluate to false.
If expression is true, the nothing will happen.

Eg. Console.assert(0>1, "Expression is false")

Console.warn
- By default, the warning message will be highlighted with Yellow color.

Eg. Console.warn("This is warning");

Date / /

Console.clear()

- It clear the console.

Eg. Console.clear()

Console.time() and Console.timeEnd()

- With them we can find the amount of time spend by a code on execution

Eg.
```
Console.time()
For(i=0; i<100; i++){
    //Code
}
Console.timeend();
```

* Arrays, object, string will be discussed later
* Don't worry about the above code, it will be explained later.

Console.error()

- By default, the error will be highlighted with red color.

Eg. Console.error("This is an Error");

Console.count()

- The console.count() is used to count the number that the function hit by this counting method.

* In JS counting starts from 0

Console.group() and Console.groupEnd()

=) They allows us to group content in a separate block.

Eg.
```
Console.group('Simple');
Console.log('Grouped');
Console.groupEnd('Simple');
Console.log('new section');
```

Two types of Comment in JS

i) Single line comment

// This is a single line JS comment

Shortcut: Ctrl + /

ii) Multiline Comment

```
/* One line
   two line
*/
```

Date / /

Variables

- Containers/memory location to store data

* Datatypes in JS are either variable or constant

* Keyword to declare a variable
  i) var
  ii) let
  iii) const ⇒ (used when you don't want to change value)

```
var num1 = 34;
var num2 = 56;
Console.log(num1 + num2);
```

Output = 90

* In a similar way 'let' is used.

Note: JS variables are case sensitive. For eg. a and A are different variable.

Const: We use const when we are sure a variable will not be redeclared.

Eg.

```
Const age = 20;
Const num = 10;
Const job = 'developer';
```

Datatypes in JS

1. Primitive : undefined, null, number, string, boolean, symbol
2. Reference : Arrays, Objects.

* String

```
Var Str 1 = "This is an apple"
Var Str 2 = "This is a string"
```

→ Console.log(Str1)

↳ It print This is an apple in Console.

* Number

```
var num 1 = 455;
Console.log(num1)
```

* Object

It is reference datatype. It is like other variable but the only diff. is that an object holds multiple values, arrays, function etc. We create an object with figure bracket {...} with an optional list of properties.

Eg.

```
let marks = { ravi : 34
              Harry : 78 }
⇒ Console.log(marks);
```

* Boolean

Represents on of two values: true or False.

Eg.

```
var a = true;
var b = False;
⇒ Console. log(a)
```

↑ Print true

* Undefined

Variable that has been declared but not defined.

Eg.

```
Var und = Undefined;  or var und;
```

Date / /

* Null

Var n = null (No value)

⇒ Console.log(n)

* Arrays

⇒ Arrays are object to store multiple values in a single variable.

Empty array syntax

i) let arr = [];

ii) Let arr = new Array();

Eg. var arr = [1, 2, 3, 4, 5]

(indexes: 0, 1, 2, 3, 4)

⇒ Consol.log(arr[4])

↑ Prints

* Numbering starts from 0

---

* Array Methods

i) length of an array : Returns the number of element in an array

Array

• let myarr = ["Fan", "Camera", 34, null, true];

Console.log(myarr.length);

ii) ~~.pop()~~ .push()
⇒ Add an item to end of array.

Eg. let fruits = ["Orange", "Apple"];
let len = fruit.push('mango')

New arr is ["Orange", "Apple", "Mango"]

iii) .pop()
Remove an item from end

iv) .shift()
Remove an item from Beginning of an array

v.) .unshift()
Add an item to the Beginning

vi) .toString()
Convert array to string

vii) splice()
" Remove an item from index position.

Date / /

* Index position is, in short, Numbering. (Generally denoted by i). Other variable can also be used

0 1 2

let fruits = ["Orange", "Apple", "Mango"]
let removed item = Fruits.splice(pos,1)

New array ⇒ ["Orange", "Mango"]

String Methods

Eg. String
let my String = "Harry is a good boy, too good";

i) Length of string
console.log(my string.length)

ii) Index of
console.log(mystring.indexof("good"))

gives index/Numbering of good (First)
⇒ Here it is 11

- last index of

```
Console.log(mystring.lastindex of ("good"))
```

given last 'good' index
Here ⇒ it is 23.

- Slice

```
Console.log(my string.slice(0,3))
```

Print character from 0 to 3

i.e Output ⇒ Har

- Replace

```
mystring.replace("Harry", "Rohan");
Console.log(mystring)
```

Replace 'Harry' with 'Rohan'.

Operators in JS

i) Arithmetic Operators

```
var a = 34;
var b = 56;
Console.log("The value of a+b is", a+b);
```

(Value is operant) i.e 34 & 56

* This gives 90 as an output.

Similarly these commands can be used.

```
Console.log("The value of a-b is", a-b);
Console.log("The value of a*b is", a*b);
Console.log("The value of a/b is", a/b);
```

ii) Assignment

```
var c = b;
Console.log(c);
```

(Here c is assign equal to b)

* To update value of c use,

```
c+=2;
```

↑ This adds '2' to value of c

iii) Comparison

```
var x = 34
var y = 56
Console.log(x==y);
```

↑

Since 34 ≠ 56 it will print False.

Comparison operators

- $x >= y$
- $x < = y$
- $x > y$
- $x < y$

Date / /

iv) Logical

⇒ Logical and (&&)

```
var a = 'Condition1'
var b = 'Condition2'

Console.log(true && true)
Console.log(true && False)
Console.log(False && False)
```

- && means don't print true if/until both are true.

⇒ Logical or (||)

- If any one is true, then it prints true

```
Console.log(true||False)
```

⇒ Logical not

```
Console.log(!False);
```

↑

This means true.

Date / /

Functions

It is a group of reusable code which can be called anywhere in the program. This eliminates the need to rewrite the same code.

Defining a function

- To create a function declaration use function keyword and then its name

Syntax

```
Function name(parameters){
    Statement
}
```

Eg. The below function can be use to find avg of any two number.

```
Function avg(a,b){
    return (a+b)/2; }

C1 = avg(4,6);
C2 = avg(14,16);

Console.log(C1,C2);
```

Number

Console. log (3)

- Print 3
  ↑ in Blue color

String

Console. log ('3')

Print 3
↑ in Black color

Conditionals

```
var age = 34;
```

⇒ If Else

```
if (age > 8) {
        Console. log ("You aren't a Kid");
    }
else {
        Console. log ('You are a Kid);
    }
```

Output: You aren't a Kid.

- 'if' checks the condition, and if it is false then 'else' is printed otherwise 'if' is printed

⇒ If-else ladder

```
if (age > 32) {
        Console.log("You aren't a kid");
    }

Else if (age > 26) {
        Console.log("Not a Kid");
    }

Else { Console.log("You are Kid"); }
```

⇒ It keeps on checking condition & where it is satisfied, then that statement is printed.

Loop

```
var arr = [1, 2, 3, 4, 5, 6, 7];
Console.log(arr);
```

i) For loop

Now to iterate the above array

↳ Get one value from array

Date / /

Eg.

index

```
For(var i=0; i < arr.length; i++)
{Console.log (arr[i])}
```

Explaination

i) var i=0 : It means began loop from index 0.

ii) i < arr.length; i++: Run the loop till i is less than arr.length by adding '1' to i again & again.

iii) Print the result.

Output : 1
2
3
4
5
6
7

- To iterate array, we can also use function

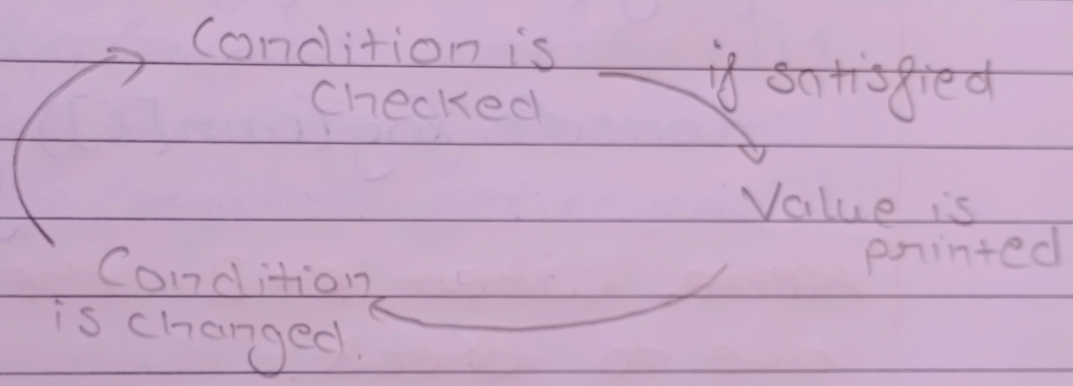
```
arr.Foreach(function(element) {
Console.log(element)
})
```

⇒ Output is same as above.

2. While loop

```
var arr = [1, 2, 3, 4, 5, 6, 7];
let j = 0
while( j < arr.length){
console.log(arr[j]);
    j++;
}
```

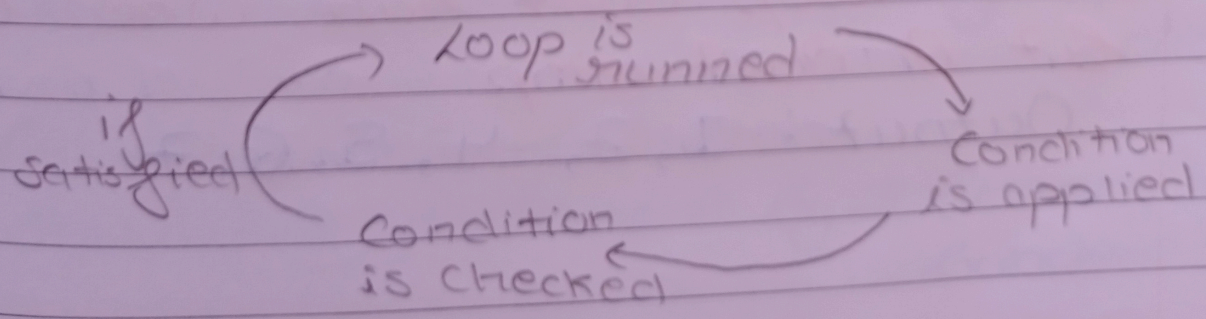
⇒ Condition
⇒ Print
⇒ Change Condition



* This loop continues till the condition get false.

3. Do-while loop

```
Do{ console.log(arr[j]);
        j++
    } while (j < arr.length);
```



Date / /

* Do while loop run one time for sure.

Break and Continue

```
var arr = [1, 2, 3, 4, 5, 6, 7];
For (var i=0; i < arr.length; i++)
  { if (i == 2) { Break; }

    Console.log(arr[i]) }
```

Here, when i = 2 the loop stops due to Break.

Output : 1 , 2

```
var arr = [1, 2, 3, 4, 5, 6, 7];
For (var i=0; i < arr.length; i++)
{ if (i == 2) { Continue; }
    Console.log(arr[i]) }
```

Here, loop don't run when i = 2 but it don't stop.

Output : 1, 2, 4, 5, 6, 7

Dates

```
let mydate = new date();
    Console.log(mydate)
```
↳ Print date

Check out -

i) Console.log(mydate.gettime());
ii) Console.log(mydate.getfullyear());
iii) Console.log(mydate.getday());
iv) Console.log(mydate.getminutes());
v.) Console.log(mydate.get Hours());

DOM manipulation

=) We can edit HTML & CSS of a page dynamically using JS.

```
<button id='Harry' Clickme</button>
```

Click me

- To click on a button we can use
  document.get element by Id('Harry').click()

- Changing CSS of button
  document.getelement by Id('Harry').style
  border='blue'

- To find element in complex program

```
let elem = document.getElementById('Harry');
console.log(elem)
```

↑ This print element having id = 'Harry'

2.) By Class name

i) `let elemclass = doc.getelementbyclassname('Container')`

```
console.log(elemclass);
```

↑ Print element having Class = 'Container'

ii) `elemclass[0].style.background = "Yellow"`

↑ Target: First element in Class.

iii) `console.log(elemclass[0].innerHTML);`

↑ It gives HTML (text with tag) of first element in class

iv) `console.log(elemclass[0].innertext);`

↳ gives text

3. By Tag name(tn)

```
tn = document.getelementbytagname('div')
```

- `console.log(tn)`
  ↳ Gives 'div'

To add an element in div we use,

```
createdelement = document.create element('p');
createdelement.innertext = "This is para";
```

↑ Add text in created element

To add :

```
tn[0].appendchild(created element);
```

⇒ To replace created element

```
createdElement2 = document.createElement('b');
createdElement2.innertext = "This is bold";

tn[0].replacechild(createdElement2, createdElement);
```

Check out –
document.title , document.script,
document.url, document.domain,
document.links

Date / /

Selecting using Query

```
sel = document.queryselector('.container')
    Console.log(sel)
```

↳ Select first element having Class = 'Container'

```
sel = document.queryselectorall('.container')
    Console.log(sel)
```

↑ All element having Class = 'Container'

Events in Javascript

i) On Clicked

Eg. Create an HTML tag like.

```
<button id="Harry" OnClick="click()"
    > Clickme </button
```

On Click, invoke the click() function.

Create Click() func.

```
function Click()
    Console.log('Button was clicked')
```

This will be printed in console after you clicked button.

2. On load

```
window.onload = function() {
    console.log('The doc was loaded')
```

3.) Add Event listener

Create an HTML element with id = 'First container'.

```
Firstcontainer.addeventlistener
    ('Click', function() {
    console.log("Clicked on container)})
```

Here, addeventlistener allows us to fire the second argument whenever the described event is invoked.

Eg. 2

```
Firstcontainer.addeventlistener('Click',
function() {
    document.queryselectorall('.container')
    [1].innerHTML = "<b> we have click
                                    </b>"

console.log("Clicked on container")
})
```

→ Described event

* On clicking, the inner HTML of container changes with 'We have Click' text.
↳ Second argument

* Arrow function
↳ Another way of writing a function

• Arrow	• Traditional
sum = (a, b) => {	Function sum(a, b) {
return a+b;	return a+b
}	}

=> Both function serve the same purpose

• SetTimeout and SetInterval

```
doLog = () => {
  document.querySelectorall('.Container')
[1].innerHTML = "<b> Set fired </b>"

  Console.log("I am your log")
}
```

⇒ clr = set timeout(do log; 2000);

It sets time ↵
i.e after 2sec the function will be invoked.

⇒ Clear timeout(clr)

↳ It stop the above shedule of running after 2second.

Set interval

⇒ clr = set interval (do log, 2000);

After every ↵
two second, the function is invoked.

⇒ It will continue to run.

To stop use:

Clear interval (clr).

Local Storage

⇒ It allows to save key-value pairs in a web browser with no expiration date.

⇒ Every domain has local Storage.

- localstorage.setitem('name', 'harry')

To clear the storage.

⇒ localstorage.clear()

JSON

⇒ It is a lightweight format for storing and transporting data. JSON is often used when data is sent from a server to a webpage.

Eg.

```
Obj = {name: "harry", length: 1,
       a: {this: that}}

jso = JSON.stringify(obj);
```

↳ It converts js object to a JSON string.

```
Console.log(jso)
```

↳ String will be printed in Console.