Expt. No. Date

Javascript notes Page No.

- Scripting language
- By Netscape
- Interpreted language.
- Browser's language
- It can insert dynamic text into HTML.
- It is used for client side scripting.
- also for server side programming.

Javascript is a event driven language.

Events:-

drag, blur, change, click, dblclick, drag, dragend, dragenter, dragleave, dragover. focus, mousedown,

- Mocha was first name of Javascript.
- livescript
- Javascript.

Uses of Javascript in web development.

Menu, Dropdown, Animated sliders, Maps, Multiselect dropdown, pop up windows, audio players, video players, animated gallery, zoom effect, animated calender, form validations etc.

client side Vs server side scripting

Advantages of Javascript

1. Speed
2. Server load
3. Simplicity
4. Rich interfaces

Teacher's Signature :

(7) Weak - Type programing language.

(8) Interpreted language:-
Since javascript is an interpreted language, we can execute Javascript code on any platform like windows, Mac & Linux

(9) Make Web Page interactive.

(10) Provide Rich framework.

Disadvantages of Javascript:

→ (1) Client side security
(2) Browser support

Java interpreter:

Browser contain interpreter that Converts Javascript into machine code your computer understands.

```
<script>
    document.write("Hello");
</script>
```

```
<script src = "/Path To Script file.js"></script>
```

Date

Expt. No. <noscript> Page No.

it is used to provide an alternate content for users that have disabled scripts from in their browser or a browser that doesn't support client side scripts

inside <body>

```
<noscript>
sorry, you have disabled your Java
Bastard
<noscript>
```

Inline javascript:

Adding javascript directly to a Html file.

. between the <head> tag

. between the <body> tag.

External

```
<script src = "myScript.js"> </script>
                myScript2.js
```

Comments:

//

→ document.getElementById("demo").innerHTML = "Javascript";

Teacher's Signature :

Expt. No. Date

Page No.

Reserved keywords :-

Global & local variables

```
<script>
var a = 10;
function fun() { document.write(a); }
</script>
```

Define a global variable inside a function using

window.a = 1001;

- undefined value;

Ex: var str;
document.write(str); //o/p: Undefined

Camel case :-

→ Hyphens, Underscore.

Upper CamelCase, Lower Camel Case.

operators:

Comparison operators:

==, ===

Logical: && || !

Bitwise: AND, OR

Assignment operators: +=, -=, *=, /=, %=

type of

Teacher's Signature :

```
var a = 10;
document.write(typeof(a));
```

→ String operators :-

(+)

String operators :-

```
var Max = (a>b) ? a : b;
```

In Chrome Console :

→ console.log

```
console.warn("this is warning");
console.time("track");
:
console.timeEnd("track");
```

O/P: track: 0.629 sec :

```
debugger; // command used for debugging
```

Let :

ES 2015 introduced two imp keywords; let & const

These two keywords provide Block scope variables (and constants) in Javascript.

Expt. No. Page No. Date

var x = 20;
sop(x)
sop(x)

let x = 20
sop(x)
sop(x)

let x = 100
sop(x) 100

let x = 20
sop(x) 20
sop(x) // 100

let x = 10
let x = 20
error

var x = 10;
var x = 20;

Const (can be reasigned)
rest same as let

Datatypes
string, number, Boolean, array, object, null, Undefined

document.write("
"); // new line

→ string as a Object
var y = new string("hi");
var x = "hi"
x == y
x === y
value is same type is different

Teacher's Signature :

Number in Javascript

→ it has only one type of number.

NaN :

```
var x = 100 / "Apple";
```

Fun" to check **isNaN(x)** ;

```
var x = NaN + 10;
```

Infinity :

```
var myNum = 2
    while (myNum != Infinity) {
        myNum = myNum * myNum;
    }
```

- Division by 0 (zero)

Numbers can be objects :

```
var y = new Number(123);
```

Boolean

```
document.write ( Boolean(10,5));
```

Boolean as objects :

```
var y = new Boolean(false);
```

Arrays : `var cars = ["Saab", "Volvo", "BMW"]`

```
var cars = new Array("  ", "  ", "  ");
```

Expt. No. How to recognise an Array Date

Page No.

```
var arr = [a, b, c];
document.write (Array.isArray(arr));

function isArray(arr){
return arr.constructor.toString().indexOf("Array
}                                          >-1
```

Array Properties & Methods:

```
var x = cars.length;
var y = cars.sort();
```

For Each loop

```
var students = [a, b, c];
students.forEach (myFunc);
function myFunc (value){
    text += value + " ";
}
```

push method in Array:

push return length of array

→ unshift & shift method:

unshift add element at beginning & return new length.

→ shift(): Begin & remove;

Teacher's Signature :

Javascript objects

It is a template based not class based.
Here, we don't create class to get the objects.
But, we direct create objects.
. It is an object based language.

Create objects:

① By object literal
② By creating instance of object directly (using new)
③ By using new.

```
var emp = new Object();
```

Switch

```
var grade = 'A';
switch (grade)
{   case 'A':
      d.w ("Hello A");
    break;
    case 'B':
    .
    default: d.w ("Hello bitch");
```

for IN loop

```
for (var in object) {
      code to be executed
}
```

Expt. No. Page No. Date

<script type="text/javascript">

function showmessage(firstname, lastname){

alert("Hello " + firstname + " " + lastname)

}

</script>

<input type="button" onclick = showmessage('Ram', 'Sita')

value = Show >

* Normal call of the function outside the Script tag will not work.

Rest Parameter in JS (ES6)

→ Indefinite no of arguments.

function sum(...num)

{ for (var i of num)

} sum = sum + i

Argument objects in JS.

function addition(num1, num2)

{ return arguments[0] + arguments[1] + arguments[2];

}

addition(10, 20, 30);

→ passing a function as a parameter:

< button onclick = "show1 (show)" > click here
< / button >

< ~~function~~ button onclick = "show1 (x, show)" > click here (/ bu >

function show (value)
{ d.w (value);
}
fun show1 (x, show) {
alert (show (x));
}

Function expressions in JS:

< body >
< p id = "demo" > < / p >
< script type = "text/javascript" >
var x = function (a,b) return a*b;
docu. getO. byId ('demo'). innerHTML = x(2,3);
< / script >

O/P: 6

Function() constructor in JS:

→ Functions can also be defined with built in JavaScript constructor called Function()

var myf = new function (a,b) return a*b;
var myf = new function ("10","20", "return a*b");

Expt. No. Date

Javascript Hoisting:

Hoisting is Javascript's default behaviour of moving all declarations to the top of current scope. (to the top of current script or current function).

Java Initialised are not Hoisted.

Javascript only hoist declaration not initialisations.

```
x=10;
d.w(x);      O/P:-10
var x;
```

function Hoisting:-

⇒ Hoisting is behaviour to move declarations at top of the current scope.

⇒ Hoisting applies to variables declaration & to function declarations.

→ Because of this, Javascript functions can be called before they are declared.

⇒ function expression can be made "self invoking". starts automatically without being called.

· function expression will execute automatically if the expression is followed by ().

```
(function(){
    d.w("hello");
})();
```

Teacher's Signature :

automatic function => check by using console

(function () { d.w ("hello");
}) ();

Self invoke function :

function can be used as value.

✓ var x = fun(5,10) + 20;

Arrow functions :

Short syntax for writing function expressions.

- You don't need the function keyword, the return keyword and the curly brackets.
- → Arrow functions are not hoisted. They must be defined before they are used.
- Using const is safer than using var, because a function expression is always constant value.
- You can omit the return keyword & the curly brackets if the function is a single statement. Because of this, it might be a good habit to always keep them.

```
const x = (a,b) => a*b ;
or const x = (a,b) => {
    d.w(a)
    d.w(b)
}
```

Expt. No. Invoking a function with constructor

Date

Page No.

```
function myfun(str1, str2){
    this.firstname = str1;
    this.lastname = str2;
}
var x = new myfunc("Chandan", "nelish");
```

~~Anonymous function~~

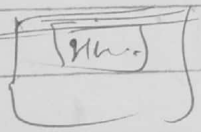
```
var x = function(){ d.w("Hello") };
```

eval()

global func that evaluates a specific string as Javascript code & execute it.

```
eval("alert("Hello How are you")");
```

O/P;



Alert dialog box:

Conformation dialog box:

(two buttons, ok and cancel)

If OK, then confirm() will return true,

else confirm() will return false:

```
function getcofn()
{
    var getval = confirm("Do you want to continue");
    if true, ok
```

Teacher's Signature :

prompt dialogbox:

ok & cancel.

- if ok then retrieve the given value from the textbox,
  else return null;

```
function getValue(){
  var getval = prompt("Enter your name", "default");
  d.w(getname);
}
```

```
<input type="button" value="Click me" onclick="getValue()" />
```

Page Redirection:

```
window.location = "http://helloworld.com";
```

```
setTimeOut(redirect(), 10,000ms);
{ setTimeOut('redirect()', 10,000);
```

Page Printing:

```
<P> ... </P>
<P> </P>
<input type="button" value="Print" onclick="window.print()" />
```

Date

Expt. No. Type conversion Page No.

datatypes: string, number, boolean, object, function

6 types of objects:

object, Date, Array, String, Number, Boolean

Number to String:

```
var a=100
var x= String(a);
a.toString();
var x = String(false);
var x = String(Date());
Date().toString();
var v= new Date();
v.getDate();
var v = "3.12";
Number(v);
```

Automatic type conversion

5 + Null = 5

"4" - "2" = 2

toFixed() method:

```
x.toFixed(3);
```

8.7785

op => 8.778500

decimal ke baad.

x.toPrecision(3) => 8.78

Teacher's Signature :

→ Number () method:

ParseInt() ParseFloat()

str.length , var pos = txt.indexOf("nitish");

txt.slice() (7,12); // accept negative index

"excluded

txt.substring(7,15); // cant accept negative index

substr // substr is similar to slice.

but if substr takes second parameter as the length of the extracted part.

.replace("Hello", "Hi");

toLowerCase ();

.concat (a, b);

trim() removes white spaces from both sides

charAt() // return e, a, b ..

charCodeAt() return unicode of the character at the specified index of a string;

(101, 102)

var arr = str.split(" ");

arr.toString(); // toString

var arr = ['a', 'b', 'c', 'd'];

delete arr[1];

var arr2 = new Array(5);

Push method

students.sort();

students.reverse()

∴ By default sort, the sort values as strings.

This works well for strings ("Apple" comes before "Banana").
However, if numbers are sorted as strings, "25" is bigger than "100", because "2" is bigger than "1".
Because of this, sort() method will produce incorrect result when sorting numbers.
You can fix this you can use a compare function;

students.sort(function(a, b) {return a < b});

Math.max.apply(null, arr);

Array.map()

try{ } catch(e) { };

~~Onerror()~~ : onerror() was first feature to facilitate error handling in Javascript. The error ~~error~~ event is fired on the window object when an ~~expect~~ exception occurs on the page.

It displays 3 info.

1. Error msg
2. URL
3. line number

```
window.onerror = function ( msg, url, line ){
    alert (msg);
    alert ( url).
    alert (line);
}
```

Notfound

// err

< ~~button~~ input type="button" onclick="show()" value="click me" />

<div onmouseover="over()" onmouseout="out()"