Web Programming- 17th Oct

JavaScript-

Used to write logic which can be executed in.**web browsers.**
Follows some of the syntax of Java language.
Object Oriented programming language.- get pre-defined objects as well as create our own objects, class object, encapsulation
dynamically typed language.

Client Side Scripting:

client=> client side scripting=> Logic which has to be executed in the client machine.
server=> server side scripting=> Logical part which has to be executed on the server.

File downloading:

client=> To store the file into the client's folder structure.
server=> To receive the file and store it on the server.
File Uploading:
client=> to send files on the server.
server=> Receive file and store it on the server.

Client's logic to send file:

File Uploading:

client=> to send the file on the server.
Receive the file and store it on the server.

Post a Comment:

client=> show text field, send that written comment on the server.
server=> to receive the comment and store it on the database.

Signup

Client=> show text field, send that written comment on the server

Server=>receive the data and store it on the database.

Login:

Client=> show login form, form - validation,send the username and password.
Server=> Receive the data and store it in database.

Document:

pre-defined object in JS. Represents your entire web page.
To perform any operation on the webpage, we are going to use document.
We can display some data on the web page.
We can remove some element from our page.
Change the content of the element.
write()- To print data on the webpage.
document.write(“Hello World”);
Console- only for development purposes.
Console object- represents the console of the browser.
console.log(“Hello Console!”);
log()- used to write a message to display.

How to write JavaScript into html:

```
<body>
<script>document.write(“Hello World”);
console.log(“Hello Console!”);
if(condition{
document.write(“inside if block”)
}else{
document.write(“nothing”)
}
</script>
</body>
```

We want to add 2 integer numbers, but we must get those numbers from user input.

If the JS codes are written on the same line, the use of semicolon is mandatory. Otherwise it is not mandatory to use a semicolon.

Ex: When the code is written on the same line.

```
<script>document.write(“Hello World”); console.log(“Hello Console!”);
```

When the code is written on different lines.

```
<script>document.write(“Hello World”)
console.log(“Hello Console!”)
```

java-> compiler,JVM
javaScript-> js(javaScript) engine

In JS, we just have to Write Code-> Run Code.

There is no compiler in JavaScript. JS uses an interpreter to execute the code.

```
a=7;
b=4;
```

For declaring variables=> use var,etc.

Data types in JS:

The data types will be assigned dynamically(at run time) by looking at the value.The JS engine assigns the memory location at the server.

Ex: Numbers.

```
<script>
a=5;
b=8;
c=”Hello”;
d=’a’;
a=”world”;
console.log(a);
console.log(a,b);
s1=”hello”;
s2=”world”;
console.log(s1,s2);
c=true;
d=false;
console.log(c,d);
</script>
```

Operators	**Symbols**
Arithmetic Operators	+,-,*,/,%
Relational Operators	< > <= >= != ==

Logical Operators	|| ! &&
Assignment Operators	= += -= *=
Increment Operators	++
Decrement Operators	–
Conditional/Ternary Operators	v3=condition?v1:v2;
Bitwise Operators	& |

Every text in JS is considered as a string.
In JS, variables can change their data type.
Typescript: It is used for securing the nature of the variable.
The const keyword is used instead of final.
JS is not type safe.

TypeScript= JS+Data Type.

```
<script>
a=5;
b=8;
c=a>b?a:b;
</script>
```

Control Structures
if

If else
If-else-if ladder
nested-if
Switch case
Looping Structures
For Loop
While loop
Do While loop

Example of using for loops.

Links for practicing if-else and loop based programs. www.w3schools.com

Ex: to display even numbers between 1 to 20.

```
for(i=1;i<=20;i++)
{
if(i%2==0)
document.write(i+” ”);

document.write(i+”<br> ”);
console.log(i);
}
```

Arrays: Collection of similar elements. In JS, an array is simply a collection of elements. The elements can either be homogenous or heterogenous.

```
<script>
myList=[10,20,30,”hello”,true,9.4]
console.log(myList);
console.log(myList[0]);// Displaying the element in the first index
console.log(myList[1]);// Displaying the element in the second index
console.log(myList.length);// Getting the length of the array.
</script>
```

If we add a value greater than that of the index value present in an array.

```
console.log(myList[6]);// undefined.
```

To determine the array length:

```
console.log(myList.length);
```

To get each element of the array one by one and print them.

```
<script>
myList=[10,20,30,”hello”,true,9.4]
for(i=0;i<mylist.length;i++){
console.log(myList[i]);
}
</script>
```

OR

```
for(i=0;i<=mylist.length-1;i++){
console.log(myList[i]);
}
```

To determine whether a number is even or not.

```
<script>
myList=[10,20,30,”hello”,true,9.4]
for(i=0;i<mylist.length;i++){
if(myList[i]%2==0){
console.log(myList[i]);
}
}
</script>
```

To determine the maximum value present in an array.

Steps:

Assume that the first element is maximum.
Cross checking our assumption.
If the first element is maximum, then all the other elements must be smaller (ideal scenario).
Check if all the elements are less than the first element or not.
if(1st element<remaining elements)

Let arr=[9 10 87 3 5]

9 is max
max=arr[0]
9<10 => Maximum value changed
10<87=> Maximum value changed
87<3 => Maximum value not changed
87<5 => Maximum value not changed

Therefore Max: 87

Code:

```
<script>
Let arr=[11, 20, 8, 90 ,4 ,6 ,7, 3]
max=arr[0];
for(i=1;i<arr.length;i++{
if(max<arr[i]){
max=arr[i];
}
}
console.log(max);
</script>
```

Two Dimensional (2-D) arrays: Collection of 1-D arrays

Syntax for 2-D arrays:

```
arr=[[10,20,30][40,50,60],[70,80,90]]
console.log(arr[0]);
console.log(arr[0][0]);// Accessing the first element of the first array.
console.log(arr[0][1]);// Accessing the second element of the first array.
console.log(arr[0][2]);// Accessing the third element of the first array.
```

Output:

```
[10,20,30]
```

```
arr=[[10,20,30][40,50,60],[70,80,90]]
for(i=0;i<arr.length;i++){
        for(j=0;j<arr[i].length;j++){
        document.write(arr[][]+” ”);
        }
}
```

To display the diagonal element:

```
arr=[[10,20,30][40,50,60],[70,80,90]]
for(i=0;i<arr.length;i++){
          for(j=0;j<arr[i].length;j++){
                    if(i==j)
          document.write(arr[][]+” ”);
          }
}
```

To display the opposite diagonal element.

```
arr=[[10,20,30][40,50,60],[70,80,90]]
for(i=0;i<arr.length;i++){
          for(j=0;j<arr[i].length;j++){
                    if(i+j==2)
          document.write(arr[][]+” ”);
          }
}
```

push() :

Use of push() function- Insert an element into an array.

Ex;

```
arr=[10,20,30,40]
arr.push(50);
console.log(arr);
```

sort(): Sort the elements into an array.

Ex:

```
arr=[70,20,60,10]
arr.sort();
console.log(arr);
```

join() : Join all the elements of an array to form a single string.

Ex:

```
arr=[70,20,60,10]
str=arr.join();
console.log(str);
```

OR

Ex:

```
arr=[70,20,60,10]
arr.join();
console.log(arr.join());
```

charAt() : Takes out a character value at a specified index.

```
<script>
str=”hello”;
str.charAt(1);
console.log(str.charAt(1));
</script>
```

OR

```
<script>
str=”hello”;
ch=str.charAt(1);
console.log(ch);
</script>
```

substring() - Cut a section of a string from a complete string.

Ex:

```
<script>
str=”bahubali”;
newstr=str.substring(0,4);
console.log(newstr);
</script>
```

indexOf(): Returns index value of the character present in the string.

Ex:

```
<script>
str=”bahubali”;
index=str.indexOf(‘l’);
console.log(index);
</script>
```

toUpperCase(): Converts lowercase characters to uppercase characters.

Ex:

```
<script>
str=”bahubali”;
newstr=str.toUpperCase();
console.log(index);
</script>
```

Reversing a String: - Use your own logic.

Ref Link: www.w3schools.com/JSREF/jsref obj spring.asp