JS JavaScript Class Day 1

#JavaScript Notes

Introduction

JS

What is JavaScript?

- JavaScript was designed to add interactivity to HTML pages
- JavaScript is a scripting language
- A scripting language is a lightweight programming language
- JavaScript is usually embedded directly into HTML pages
- JavaScript is an interpreted language (means that scripts execute without preliminary compilation)
- Everyone can use JavaScript without purchasing a license

What can a JavaScript do?

- JavaScript gives HTML designers a programming tool
- JavaScript can put dynamic text into an HTML page
- JavaScript can react to events
- JavaScript can read and write HTML elements
- JavaScript can be used to validate data
- JavaScript can be used to detect the visitor's browser and it can be used to create cookies

- The HTML <script> tag is used to insert a JavaScript into an HTML page. The example below shows how to add HTML tags to the JavaScript:

```
<html>
        <body>
                <script type="text/javascript">
                document.write("<h1>Hello World!</h1>");
                </script>
        </body>
</html>
```

- So, the <script type="text/javascript"> and </script> tells where the JavaScript starts and ends:
- The **document.write** command is a standard JavaScript command for writing output to a page

Where we can write JavaScript and when it execute

- JavaScripts in a page will be executed immediately while the page loads into the browser. This is not always what we want. Sometimes we want to execute a script when a page loads, other times when a user triggers an event.

- JavaScripts in the body section will be executed WHILE the page loads.

- JavaScripts in the head section will be executed when CALLED.

Scripts in <head>

- Scripts to be executed when they are called, or when an event is triggered, go in the head section.
- If you place a script in the head section, you will ensure that the script is loaded before anyone uses it.

```
<html>
      <head>
            <script type="text/javascript">
                  function message()
                  {
                  alert("This alert box was called with the onload event");
                  }
            </script>
      </head>
      <body onload="message()">
      </body>
</html>
```

Scripts in <body>

- Script to be executive when page load
- If you place a script in the body section, it generates the content of a page.

```
<html>
        <head>
        </head>

        <body>
                <script type="text/javascript">
                        document.write("This message is written by
        JavaScript");
                </script>
        </body>
</html>
```

Insert Special Characters

- The table below lists other special characters that can be added to a text string with the backslash sign:

Code	Outputs
\'	single quote
\"	double quote
\&	ampersand
\\	backslash
\n	new line
\r	carriage return
\t	tab
\b	backspace
\f	form feed

Break up a Code Line

- You can break up a code line **within a text string** with a backslash. The example below will be displayed properly:

```
document.write("Hello \World!");
```

JavaScript is Case Sensitive

- A function named "myfunction" is not the same as "myFunction" and a variable named "myVar" is not the same as "myvar".

- JavaScript is case sensitive - therefore watch your capitalization closely when you create or call variables, objects and functions.

White Space

- JavaScript ignores extra spaces. You can add white space to your script to make it more readable.
- The following lines are equivalent:

  name="Hege"; or name = "Hege";

- JavaScript code (or just JavaScript) is a sequence of JavaScript statements.
- Each statement is executed by the browser in the sequence they are written.

This example will write a heading and two paragraphs to a web page:

```
<script type="text/javascript">
        document.write("<h1>This is a heading</h1>");
        document.write("<p>This is a paragraph.</p>");
        document.write("<p>This is another paragraph.</p>");
</script>
```

JavaScript Blocks

- JavaScript statements can be grouped together in blocks.
- Blocks start with a left curly bracket {, and ends with a right curly bracket }.
- The purpose of a block is to make the sequence of statements execute together.

```
<script type="text/javascript">
{
        document.write("<h1>This is a heading</h1>");
        document.write("<p>This is a paragraph.</p>");
        document.write("<p>This is another paragraph.</p>");
}
</script>
```

- Normally a block is used to group statements together in a function or in a condition (where a group of statements should be executed if a condition is met).

Comments

- JavaScript comments can be used to make the code more readable.
- Comments can be added to explain the JavaScript, or to make the code more readable.
- Single line comments start with //.
- Multi line comments start with /* and end with */.
- The following example uses a multi line comment to explain the code:

JavaScript Variables

- As with algebra, JavaScript variables are used to hold values or expressions.
- A variable can have a short name, like x, or a more descriptive name, like car name.
- Rules for JavaScript variable names:
- Variable names are case sensitive (y and Y are two different variables)
- Variable names must begin with a letter or the underscore character

Declaring (Creating) JavaScript Variables

- Creating variables in JavaScript is most often referred to as "declaring" variables.
- You can declare JavaScript variables with the **var statement**

```
var x;
var carname;
```

- After the declaration shown above, the variables are empty (they have no values yet).
- However, you can also assign values to the variables when you declare them:

```
var x=5;
var carname="Volvo";
```

JavaScript Arithmetic Operators

- Arithmetic operators are used to perform arithmetic between variables and/or values.

Given that **y=5**, the table below explains the arithmetic operators:

Operator	Description	Example	Result
+	Addition	x=y+2	x=7
-	Subtraction	x=y-2	x=3
*	Multiplication	x=y*2	x=10
/	Division	x=y/2	x=2.5
%	Modulus (division remainder)	x=y%2	x=1
++	Increment	x=++y	x=6
--	Decrement	x=--y	x=4

JavaScript Assignment Operators

- Assignment operators are used to assign values to JavaScript variables.

  Given that **x=10** and **y=5**, the table below explains the assignment operators:

Operator	Example	Same As	Result
=	x=y	x=y	x=5
+=	x+=y	x=x+y	x=15
-=	x-=y	x=x-y	x=5
*=	x*=y	x=x*y	x=50
/=	x/=y	x=x/y	x=2
%=	x%=y	x=x%y	x=0

The + Operator Used on Strings

- The + operator can also be used to add string variables or text values together.To add two or more string variables together, use the + operator.

```
txt1="What a very";
txt2="nice day";
txt3=txt1+txt2;
```

- After the execution of the statements above, the variable txt3 contains "What a verynice day".
- To add a space between the two strings, insert a space into one of the strings:

```
txt1="What a very ";
txt2="nice day";
txt3=txt1+txt2;
```

or insert a space into the expression:

```
txt1="What a very";
txt2="nice day";
txt3=txt1+" "+txt2;
```

- After the execution of the statements above, the variable txt3 contains:

"What a very nice day"

Adding Strings and Numbers

The rule is:

If you add a number and a string, the result will be a string!

```
<script type="text/javascript">
x=5+5;        // 10
document.write(x);
document.write("<br />");
x="5"+"5";      // 55
document.write(x);
document.write("<br />");
```

```
 x=5+"5";       // 55
document.write(x);

document.write("<br />");

x="5"+5;          // 55

document.write(x);

document.write("<br />");

</script>

<p>The rule is: If you add a number and a string, the result will be a string.</p>
```

Comparison Operators

- Comparison operators are used in logical statements to determine equality or difference between variables or values.

  Given that **x=5**, the table below explains the comparison operators:

Operator	Description	Example
==	is equal to	x==8 is false
===	is exactly equal to (value and type)	x===5 is true x==="5" is false
!=	is not equal	x!=8 is true
>	is greater than	x>8 is false
<	is less than	x<8 is true
>=	is greater than or equal to	x>=8 is false
<=	is less than or equal to	

Logical Operators

- Logical operators are used to determine the logic between variables or values.

Given that **x=6 and y=3**, the table below explains the logical operators:

Operator	Description	Example
&&	And	(x < 10 && y > 1) is true
||	Or	(x==5 || y==5) is false
!	Not	!(x==y) is true

Conditional Operator

- JavaScript also contains a conditional operator that assigns a value to a variable based on some condition.

Syntax

variablename=(condition)?value1:value2

Conditional Statements

- Very often when you write code, you want to perform different actions for different decisions. You can use this in your code to do this.

  In JavaScript we have the following conditional statements:

if statement

use this statement to execute some code only if a specified condition is true

if...else statement

use this statement to execute some code if the condition is true and another code if the condition is false

if...else if....else statement

use this statement to select one of many blocks of code to be executed

switch statement

use this statement to select one of many blocks of code to be executed

If Statement

- Use the if statement to execute some code only if a specified condition is true.

Syntax

```
if (condition)
{
        code to be executed if condition is true
}
```

Example

```
<script type="text/javascript">
        //Write a "Good morning" greeting if
        //the time is less than 10

        var d=new Date();
        var time=d.getHours();

        if (time<10)
        {
                document.write("<b>Good morning</b>");
        }
  </script>
```

Clock in JS

```
<html>
<body>
<script type="text/javascript">
        var t = new Date();
        var hr = t.getHours();
        var min = t.getMinutes();
        var sec = t.getSeconds();
document.write(hr + " :  " + min  + " :" + sec);
</script>
</body>
</html>
```

If...else Statement

- Use the if....else statement to execute some code if a condition is true and another code if the condition is not true.

Syntax

```
if (condition)
{
        code to be executed if condition is true
}
else
{
        code to be executed if condition is not true
}
```

```
<script type="text/javascript">

  //If the time is less than 10, you will get a "Good morning" greeting.
  //Otherwise you will get a "Good day" greeting.
  var d = new Date();
  var time = d.getHours();


  If (time < 10)
  {
                document.write("Good morning!");
  }
  else
  {
                document.write("Good day!");
  }
</script>
```

If...else if...else Statement

- Use the if....else if...else statement to select one of several blocks of code to be executed.

Syntax

```
if (condition1)
{
        code to be executed if condition1 is true
}
else if (condition2)
{
        code to be executed if condition2 is true
}
else
{
        code to be executed if condition1 and condition2 are not true
}
```

```
<script type="text/javascript">
var d = new Date();
var time = d.getHours()
if (time<10){
  document.write("<b>Good morning</b>");
}
else if (time>10 && time<16){
  document.write("<b>Good day</b>");
}
else{
  document.write("<b>Hello World!</b>");
 }
</script>
```

The JavaScript Switch Statement

- Use the switch statement to select one of many blocks of code to be executed.

Syntax:

```
switch(n)
{
  case 1:
        execute code block 1
        break;
  case 2:
        execute code block 2
        break;
  default:
        code to be executed if n is different from case 1 and 2
}
```

```
<script type="text/javascript">
    //You will receive a different greeting based on what day it is. Note that Sunday=0,
  //Monday=1, Tuesday=2, etc.
  var d=new Date();
  theDay=d.getDay();
  switch (theDay)
  {
          case 5:
                  document.write("Finally Friday");
                  break;
          case 6:
                  document.write("Super Saturday");
                  break;
```

```
        case 0:
                document.write("Sleepy Sunday");
                break;
        default:
                document.write("I'm looking forward to this weekend!");
  }
</script>
```

JavaScript Popup Boxes

- JavaScript has three kind of popup boxes: Alert box, Confirm box, and Prompt box.

Alert Box

- An alert box is often used if you want to make sure information comes through to the user. When an alert box pops up, the user will have to click "OK" to proceed.

Syntax

```
alert("sometext");
```

```
<html>
  <head>
        <script type="text/javascript">
              function show_alert()
              {
              alert("I am an alert box!");
              }
        </script>
  </head>
<body>
  <input type="button" onclick="show_alert()" value="Show alert box" />
</body>
</html>
```

Confirm Box

- A confirm box is often used if you want the user to verify or accept something.
- When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed.
- If the user clicks "OK", the box returns true. If the user clicks "Cancel", the box returns false.

All Box in One

```
<html>
<head>
<script type="text/javascript">
function confirmbox()
{
confirm("This is Confirm Box");
}
</script>
<script type="text/javascript">
function promptbox()
{
        prompt("This is Confirm Box");
}
</script>
```

```
<script type="text/javascript">
function alertbox()
{
    alert("This is Alert Box");
}
</script>
</head>
<body>
<input type="button“ value="ConfirmBox" onClick=confirmbox()>
<input type="button" value="PromptBox" onClick=promptbox() >
<input type="button" value="AlertBox" onClick=alertbox() >
</body>
</html>
```

Example

Syntax :-

confirm("sometext");

```
<html>
  <head>
        <script type="text/javascript">
        function show_confirm(){
                var r=confirm("Press a button");
                if (r==true)
                {
                        document.write("You pressed OK!");
                }
```

```
            else{
            document.write("You pressed Cancel!");
        }
    }
    </script>
  </head>
<body>
  <input type="button" onclick="show_confirm()" value="Show confirm box" />
</body>
</html>
```

Prompt Box

- A prompt box is often used if you want the user to input a value before entering a page.

- When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value.

- If the user clicks "OK" the box returns the input value. If the user clicks "Cancel" the box returns null.

Syntax

```
prompt("sometext","defaultvalue");
```

```
<html>
  <head>
  <script type="text/javascript">
  function show_prompt()
  {
                    var name=prompt("Please enter your name","Amit");
                    if (name!=null && name!="")
                    {
                    document.write("Hello " + name + "! How are you today?");
                    }
  }
  </script>
</head>
  <body>
        <input type="button" onclick="show_prompt()" value="Show prompt box" />
  </body>
</html>
```

JavaScript Functions

- To keep the browser from executing a script when the page loads, you can put your script into a function.

  A function contains code that will be executed by an event or by a call to the function.

Syntax (Define a Function)

```
function functionname(var1,var2,...,varX)
{
        some code
}
```

The parameters var1, var2, etc. are variables or values passed into the function. The { and the } defines the start and end of the function.

Note: A function with no parameters must include the parentheses () after the function name.

The return Statement

- The return statement is used to specify the value that is returned from the function.
- So, functions that are going to return a value must use the return statement.

The example below returns the product of two numbers (a and b):

```
<html>
	<head>
		<script type="text/javascript">
			function product(a,b)
			{
				return a*b;
			}
		</script>
	</head>
	<body>
		<script type="text/javascript">
		document.write(product(4,3));
		</script>
	</body>
</html>
```

The Lifetime of JavaScript Variables

- If you declare a variable within a function, the variable can only be accessed within that function.

- When you exit the function, the variable is destroyed. These variables are called local variables.

- You can have local variables with the same name in different functions, because each is recognized only by the function in which it is declared.

If you declare a variable outside a function, all the functions on your page can access it. The lifetime of these variables starts when they are declared, and ends when the page is closed.

JavaScript Loops

- Often when you write code, you want the same block of code to run over and over again in a row.
- Instead of adding several almost equal lines in a script we can use loops to perform a task like this.

In JavaScript, there are **two** different kind of **loops**:

for - loops through a block of code a specified number of times

while - loops through a block of code while a specified condition is true

do while - loops through a block of code while a specified condition is true

The for Loop

- The for loop is used when you know in advance how many times the script should run.

Syntax :-

```
for (var=startvalue;var<=endvalue;var=var+increment)
{
        code to be executed
}
```

```
<html>
<body>
        <script type="text/javascript">
                var i=0;
                for (i=0;i<=5;i++)
                {
                        document.write("The number is " + i);
                        document.write("<br />");
                }
        </script>

</body>

</html>
```

The while Loop

- The while loop loops through a block of code while a specified condition is true.

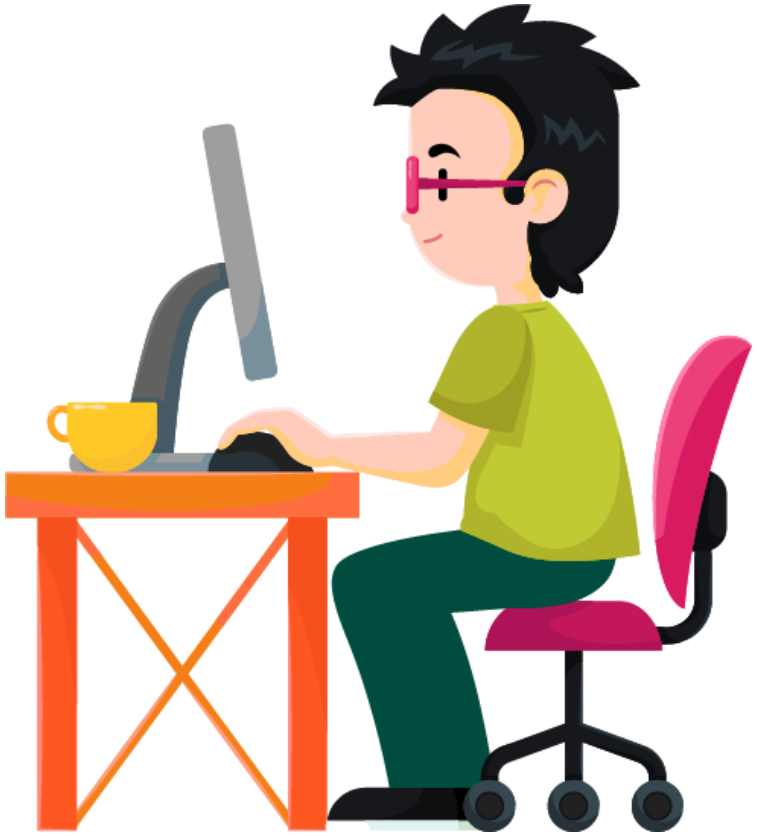
Syntax

```
while (var<=endvalue)
{
        code to be executed
}
```

```
<html>
<body>
	<script type="text/javascript">
		var i=0;
		while (i<=5)
		{
			document.write("The number is " + i);
			document.write("<br />");
			i++;
		}
	</script>
</body>

</html>
```

Get Exclusive Video Tutorials



www.aptutorials.com

https://www.youtube.com/user/Akashtips

Get More Details

www.akashsir.com

If You Liked It !

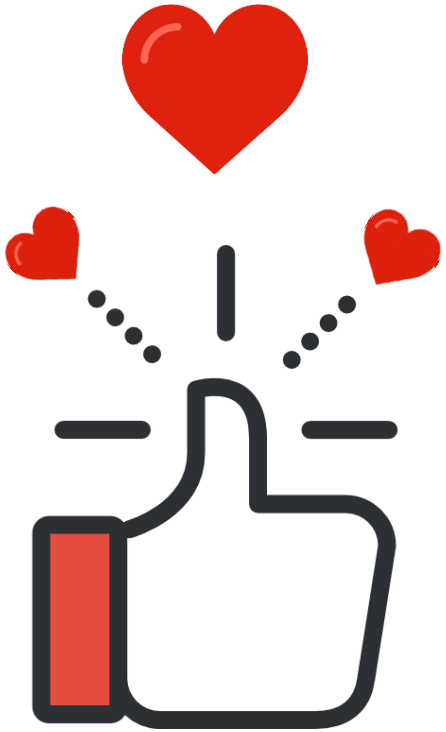
Rating Us Now

Just Dial

https://www.justdial.com/Ahmedabad/Akash-Technolabs-Navrangpura-Bus-Stop-Navrangpura/079PXX79-XX79-170615221520-S5C4_BZDET

Sulekha

https://www.sulekha.com/akash-technolabs-navrangpura-ahmedabad-contact-address/ahmedabad



Connect With Me

Akash Padhiyar
#AkashSir

www.akashsir.com
www.akashtechnolabs.com
www.akashpadhiyar.com
www.aptutorials.com

Social Info

Akash.padhiyar

Akashpadhiyar

Akash_padhiyar

+91 99786-21654

#Akashpadhiyar
#aptutorials