WEB TECHNOLOGY

UNIT-II

JAVA SCRIPT

K.A.Usaif ahamed
Assistant Professor
Department of CS & IT

Why Learn JavaScript?

- JavaScript is among the most powerful and flexible programming languages of the web. It powers the dynamic behavior on most websites, including this one.

Why to Learn Javascript

- **JavaScript** is for students and working professionals to become a great Software Engineer specially when they are working in Web Development Domain.
- JavaScript, it helps you developing great **front-end** as well as **back-end** software using different JavaScript based frameworks like jQuery, Node.JS etc.
- JavaScript is everywhere, it comes installed on every modern web browser and so to learn JavaScript you really do not need any **special environment setup**. For example Chrome, Mozilla Firefox , and every browser, supports JavaScript.
- Javascript helps you create really **beautiful and crazy fast websites**.
- JavaScript usage has now extended to mobile app development, desktop app development, and game development. This opens many opportunities for you as JavaScript Programmer.

Hello World using Javascript

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

What are Variables?

- Variables are containers for storing data (storing data values).
- In this example, x, y, and z, are variables, declared with the var keyword:

In this example, x, y, and z, are variables, declared with the var keyword:

```
var x = 5;
var y = 6;
var z = x + y;
```

In this example, x, y, and z, are variables, declared with the let keyword:

```
let x = 5;
let y = 6;
let z = x + y;
```

When to Use JavaScript var?

- Always declare JavaScript variables with var,let, or const.
- The var keyword is used in all JavaScript code from 1995 to 2015.
- The let and const keywords were added to JavaScript in 2015.
- If you want your code to run in older browser, you must use var.

Operators

JavaScript Arithmetic Operators

Arithmetic operators are used to perform arithmetic on numbers:

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
**	Exponentiation
/	Division
%	Modulus (Division Remainder)
++	Increment
--	Decrement

JavaScript Assignment Operators

Assignment operators assign values to JavaScript variables.

Operator	Example	Same As
=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y

JavaScript String Operators

- The + operator can also be used to add (concatenate) strings.

Example: 1

- let text1 = “usaif";
- let text2 = “Ahamed";
- let text3 = text1 + " " + text2;

Usaif Ahamed

Example: 2

- let text1 = “Usaif ahamed ";
- text1 += “Assistant Professor";

Usaif Ahamed Assistant Professor

JavaScript Comparison Operators

Operator	Description
==	equal to
===	equal value and equal type
!=	not equal
!==	not equal value or not equal type
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to
?	ternary operator

JavaScript Logical Operators

Operator	Description
&&	logical and
||	logical or
!	logical not

```
<html>
 <body>
  <script type = "text/javascript">
  <!--
   var a = true;
   var b = false;
   var linebreak = "<br />";
     document.write("(a && b) => ");
   result = (a && b);
   document.write(result);
   document.write(linebreak);
        document.write("(a || b) => ");
   result = (a || b);
   document.write(result);
   document.write(linebreak);
        document.write("!(a && b) => ");
      result = (!(a && b));
      document.write(result);
      document.write(linebreak);
     //-->
   </script>
 </body>
</html>
```

```
(a && b) => false
(a || b) => true
!(a && b) => true
```

Conditional Operator (? :)

The conditional operator first evaluates an expression for a true or false value and then executes one of the two given statements depending upon the result of the evaluation.

Sr.No.	Operator and Description
1	**? : (Conditional)** If Condition is true? Then value X : Otherwise value Y

```
<html>
  <body>
    <script type = "text/javascript">
      <!--
        var a = 10;
        var b = 20;
        var linebreak = "<br />";
              document.write ("((a > b) ? 100 : 200) => ");
        result = (a > b) ? 100 : 200;
        document.write(result);
</script>
      </body></html>
```

Conditional statements

- JavaScript supports conditional statements which are used to perform different actions based on different conditions. Here we will explain the **if..else** statement.

JavaScript supports the following forms of **if..else** statement

- **if statement**
- **if...else statement**
- **if...else if... statement.**

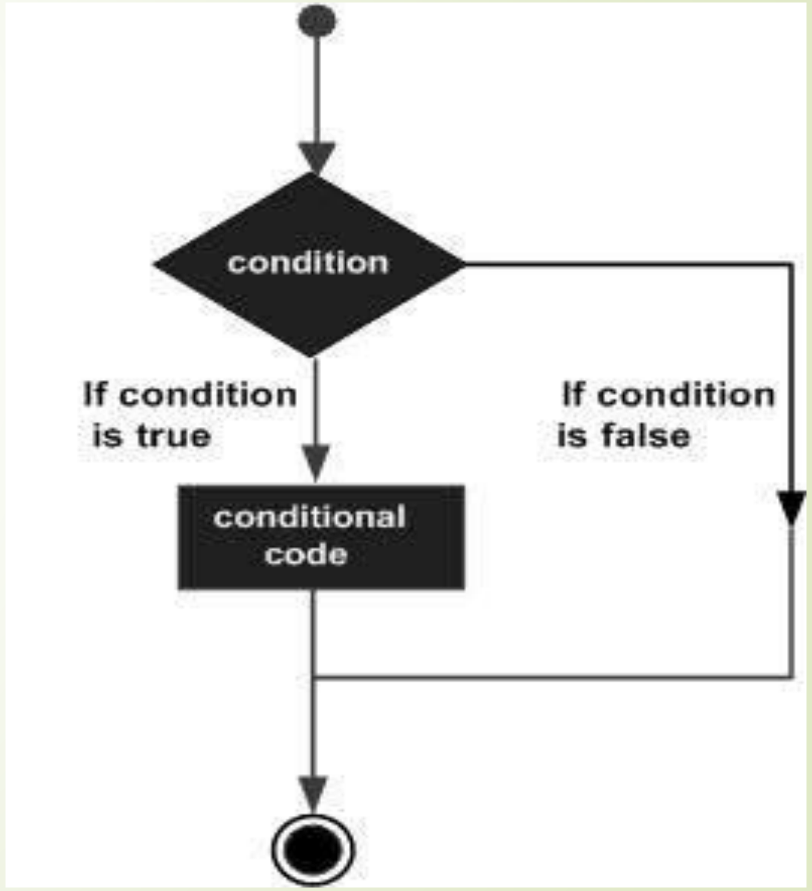
if statement

- The if statement is the fundamental control statement that allows JavaScript to make decisions and execute statements conditionally.

Syntax

- The syntax for a basic if statement is as follows –

1. if (expression) {
2. Statement(s) to be executed if expression is true
3. }



```
<html>
  <body>
    <script type = "text/javascript">
      <!--
        var age = 20;

        if( age > 18 ) {
          document.write("<b>Qualifies for driving</b>");
        }
      //-->
    </script>
    <p>Set the variable to different value and then try...</p>
  </body>
</html>
```

if...else statement

- The 'if...else' statement is the next form of control statement that allows JavaScript to execute statements in a more controlled way.

Syntax

1. if (expression) {
2. Statement(s) to be executed if expression is true
3. } else {
4. Statement(s) to be executed if expression is false
5. }

- Here JavaScript expression is evaluated. If the resulting value is true, the given statement(s) in the ‘if’ block, are executed. If the expression is false, then the given statement(s) in the else block are executed.

```
<html>
   <body>
      <script type = "text/javascript">
        var age = 15;
                    if( age > 18 ) {
              document.write("<b>Qualifies for driving</b>");
           } else {
              document.write("<b>Does not qualify for driving</b>");
           }
      </script>
      <p>Welcome to Jamal Mohamed college</p>
   </body>
</html>
```

if...else if... statement

The if...else if... statement is an advanced form of if…else that allows JavaScript to make a correct decision out of several conditions.

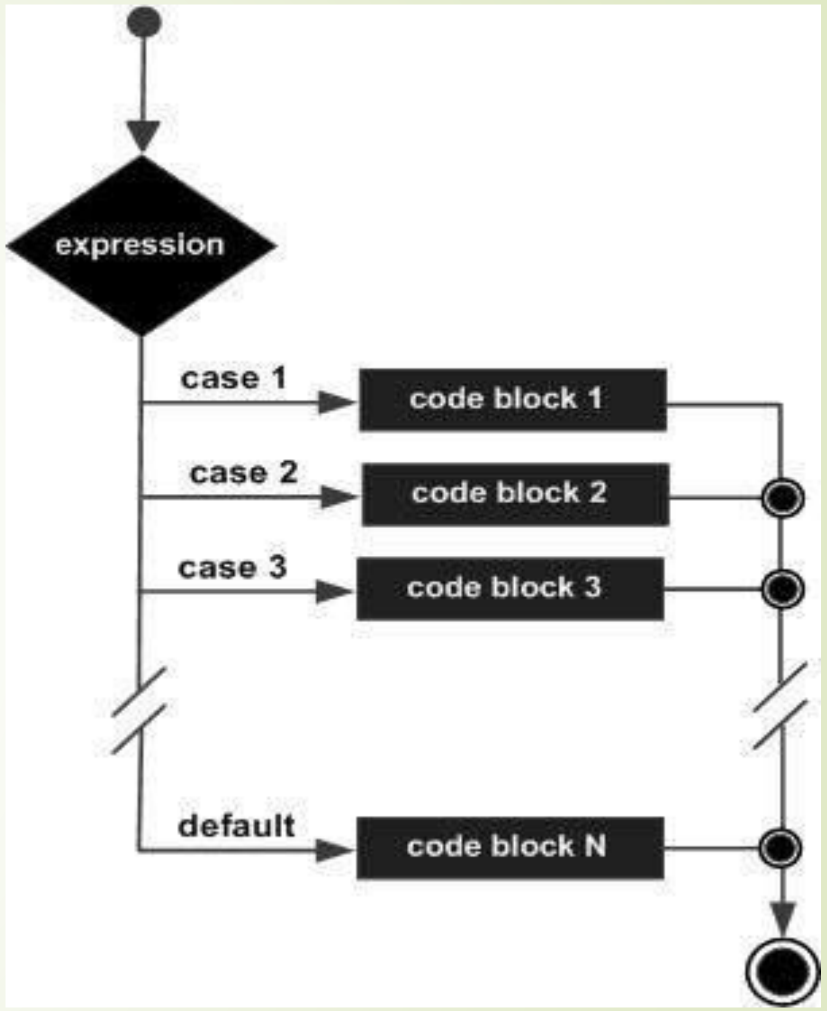
Syntax

```
if (expression 1) {
Statement(s) to be executed if expression 1 is true
} else if (expression 2) {
Statement(s) to be executed if expression 2 is true
} else if (expression 3) {
Statement(s) to be executed if expression 3 is true
} else {
Statement(s) to be executed if no expression is true
}
```

```
<html>
  <body>
    <script type = "text/javascript">
       var NME= "maths";
         if(NME == "history" ) {
           document.write("<b>History Dept</b>");
         } else if(NME == "maths" ) {
           document.write("<b>Maths Dept</b>");
         } else if(NME == "economics" ) {
           document.write("<b>Economics Dept</b>");
         } else {
           document.write("<b>Unknown Dept</b>");
         }
</script>
    <p>All the Best...</p>
  </body><html>
```

Switch Case

- The objective of a **switch** statement is to give an expression to evaluate and several different statements to execute based on the value of the expression. The interpreter checks each **case** against the value of the expression until a match is found. If nothing matches, a **default** condition will be used.



Switch (expression) {

1. case condition 1: statement(s)
2. break;

3. case condition 2: statement(s)
4. break;
5. ...

6. case condition n: statement(s)
7. break;

8. default: statement(s)
9. }

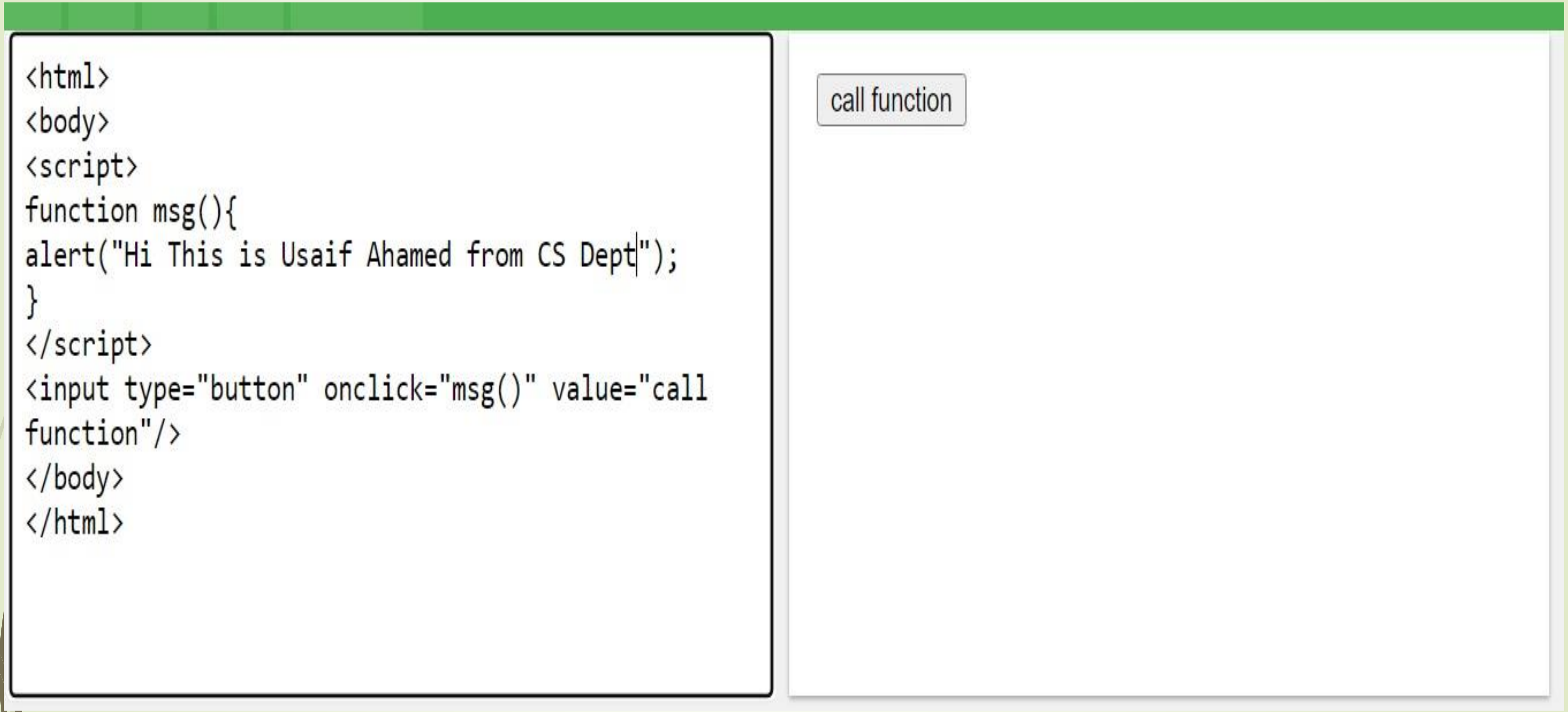
```
<html>
   <body>
      <script type = "text/javascript">
         var grade = 'A';
           document.write("Entering switch block<br />");
           switch (grade) {
             case 'A': document.write("First Class<br />");
             break;

             case 'O': document.write("Out Standing <br />");
             break;

              case 'B': document.write("Second class<br />");
              break;

              case 'C': document.write("Not so good<br />");
              break;

              case 'F': document.write("Failed<br />");
                 break;

               default:  document.write("Unknown grade<br />")
              }
              document.write("Exiting switch block");
        </script>
</body>
</html>
```

JavaScript Functions

- **JavaScript functions** are used to perform operations. We can call JavaScript function many times to reuse the code.
- *Advantage of JavaScript function*
- There are mainly two advantages of JavaScript functions.
- **Code reusability**: We can call a function several times so it save coding.
- **Less coding**: It makes our program compact. We don't need to write many lines of code each time to perform a common task.
- The syntax of declaring function is given below.

```
function functionName([arg1, arg2, ...argN])
{
 //code to be executed
}
```

```
<script>
function msg(){
alert("Hi This is Usaif ahamed from CS Dept");
}
</script>
<input type="button" onclick="msg()" value="call function"/>
```



```
<html>
<body>
<script>
function msg(){
alert("Hi This is Usaif Ahamed from CS Dept");
}
</script>
<input type="button" onclick="msg()" value="call
function"/>
</body>
</html>
```

JavaScript Function Arguments

- We can call function by passing arguments. Let's see the example of function that has one argument.

```
<script>
function getcube(number){
alert(number*number*number);
}
</script>
<form>
<input type="button" value="click" onclick="getcube(4)"/>
</form>
```

Window Object

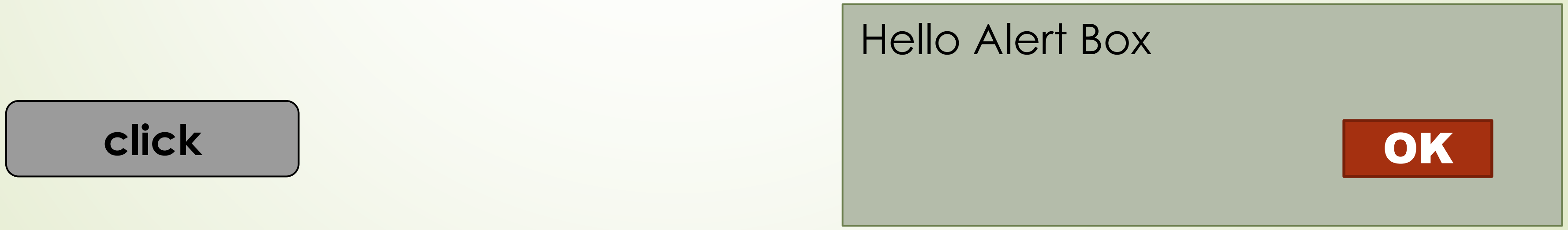
- The **window object** represents a window in browser. An object of window is created automatically by the browser.

Method	Description
alert()	displays the alert box containing message with ok button.
confirm()	displays the confirm dialog box containing message with ok and cancel button.
prompt()	displays a dialog box to get input from the user.
open()	opens the new window.
close()	closes the current window.
setTimeout()	performs action after specified time like calling function, evaluating expressions etc.

alert() in javascript

- It displays alert dialog box. It has message and ok button.

```
<html><body>
<script type="text/javascript">
function msg(){
 alert("Hello Alert Box");
}
</script>
<input type="button" value="click" onclick="msg()"/>
</body></html>
```



Example of confirm() in javascript

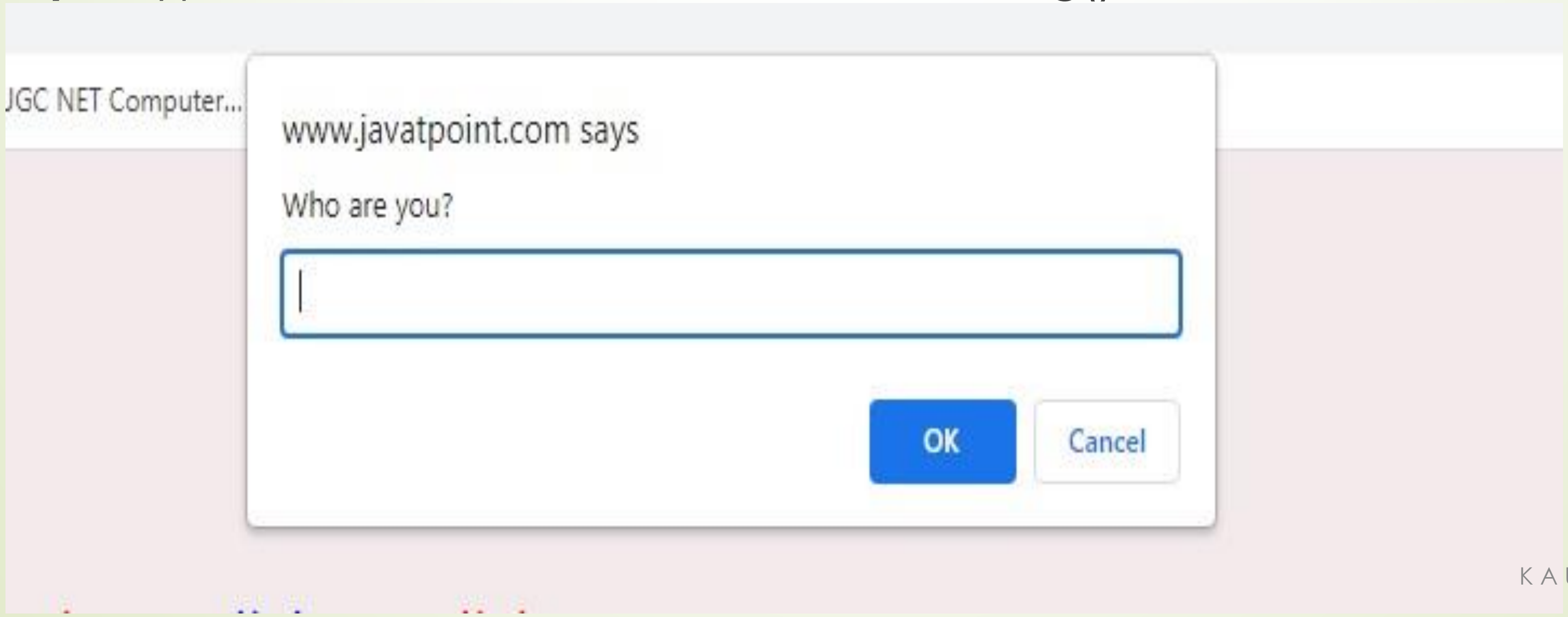
It displays the confirm dialog box. It has message with ok and cancel buttons.

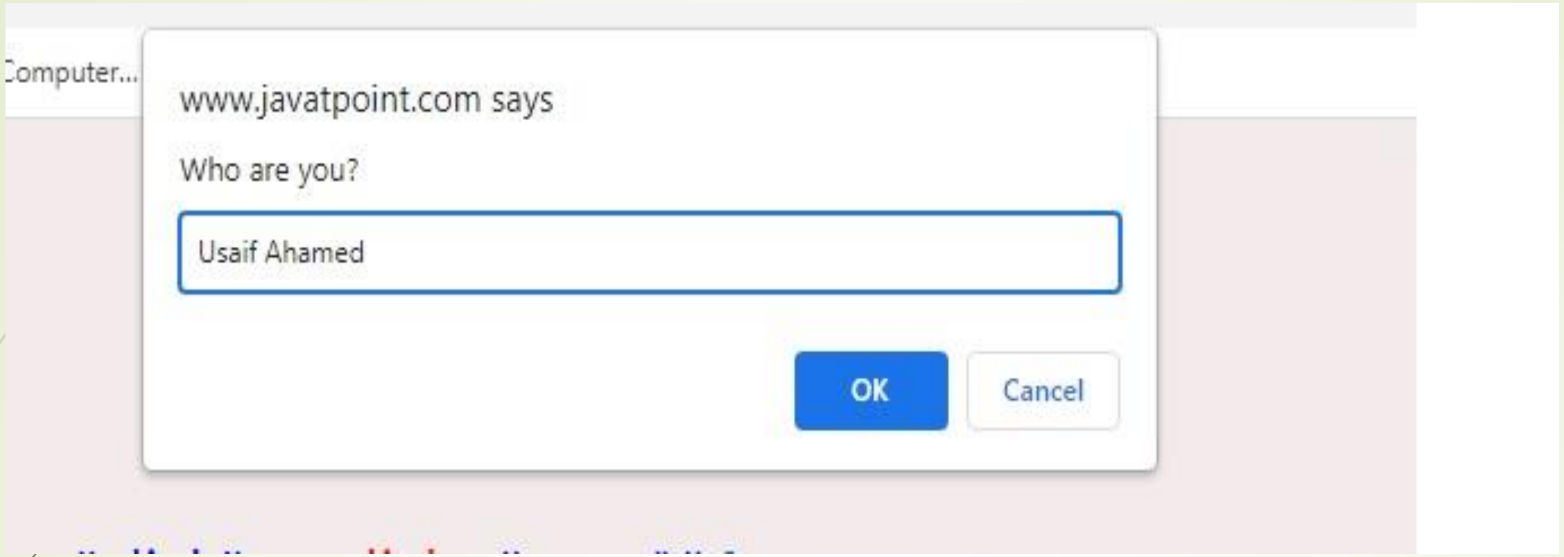
```
<script type="text/javascript">
function msg(){
var v= confirm("Are u sure?");
if(v==true){
alert("ok");
}
else{
alert("cancel");
 }
}
</script>
<input type="button" onclick=msg() value="Clickme">
```

Example of prompt() in javascript

It displays prompt dialog box for input. It has message and textfield.

1. **<script** type="text/javascript"**>**
2. function msg(){
3. var v= prompt("Who are you?");
4. alert("I am "+v);
5. }
6. **</script>**
7. **<input** type="button" value="click" onclick="msg()"**/>**



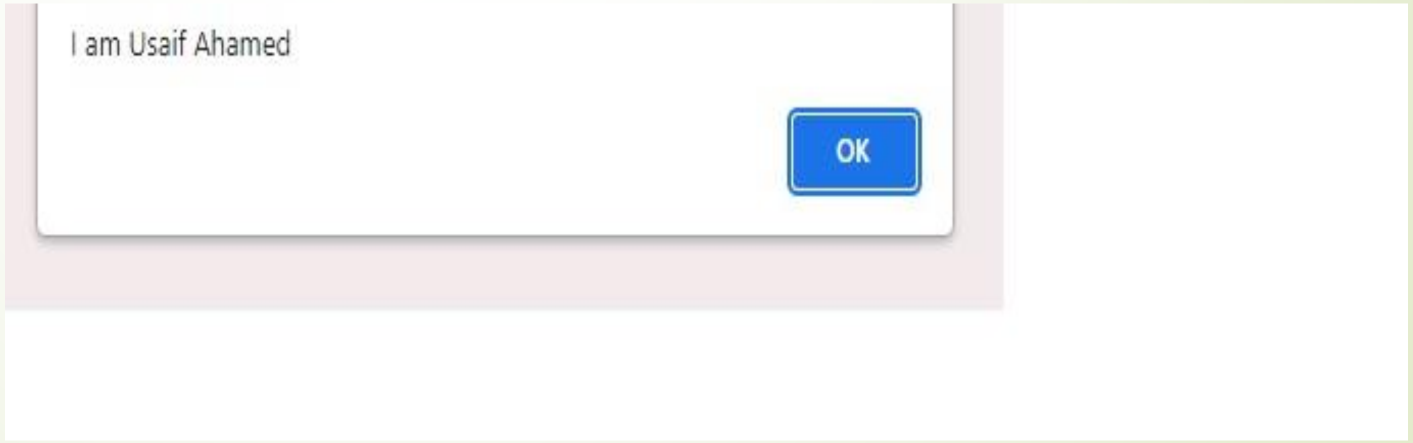


Computer...

www.javatpoint.com says

Who are you?

Usaif Ahamed

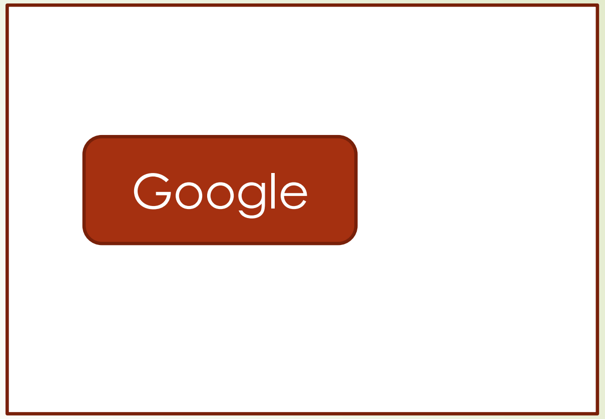


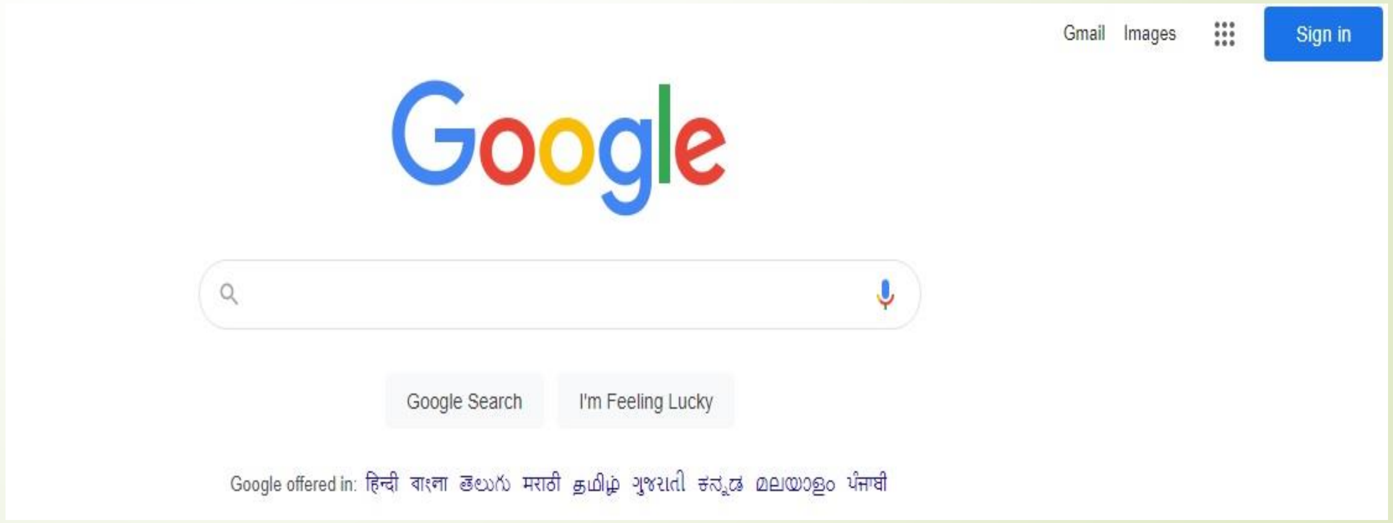
I am Usaif Ahamed

open() in javascript

It displays the content in a new window.

```
<script type="text/javascript">
function msg(){
open("http://www.google.com");
}
</script>
<input type="button" value=" Google " onclick="msg()"/>
```




Gmail
Images
Sign in
Google
Google Search
I'm Feeling Lucky
Google offered in: हिन्दी বাংলা తెలుగు मराठी தமிழ் ગુજરાતી ಕನ್ನಡ മലയാളം ਪੰਜਾਬੀ

The Window Object

Set a background color for a document:

```
document.body.style.backgroundColor = "red";
```

```
<body>
<h1>Hello World!</h1>
<input type="button" onclick="myFunction()“ value=“Set background color”>
<script>
function myFunction() {
  document.body.style.backgroundColor = "red";
}</script>
</body>
```

Set the font for an element to "italic":

```
document.getElementById("demo").style.fontStyle = "italic";
<body>
<p id="demo">This is a paragraph.</p>
<button type="button" onclick="myFunction()“ value=“Set font style”>
<script>
function myFunction() {
  document.getElementById("demo").style.fontStyle = "italic";
}
</script>
</body>
```

Title : The title of the document, as displayed at the top of the browser window.

- Ex: document.title=”This title is displayed now”;

Document methods

Write(): Allows a string of text to be written to the document.

- Document.write (“ <h1> Hello </h1>”);

```
<head>
   <script type="text/javascript">
      document.alinkColor = "green";
      document.linkColor = "red";
      document.vlinkColor = "black";
   </script>
</head>
<body>
   Click on the following hyperlinks to see the default, normal and visited states:
   <br /><br />
   <a href="#first">Hyperlink text</a>
   <br /><br />
   <a href="#second">Hyperlink text</a>
   <br /><br />
   <a href="#third">Hyperlink text</a>
</body>
```

Title : The title of the document, as displayed at the top of the browser window.

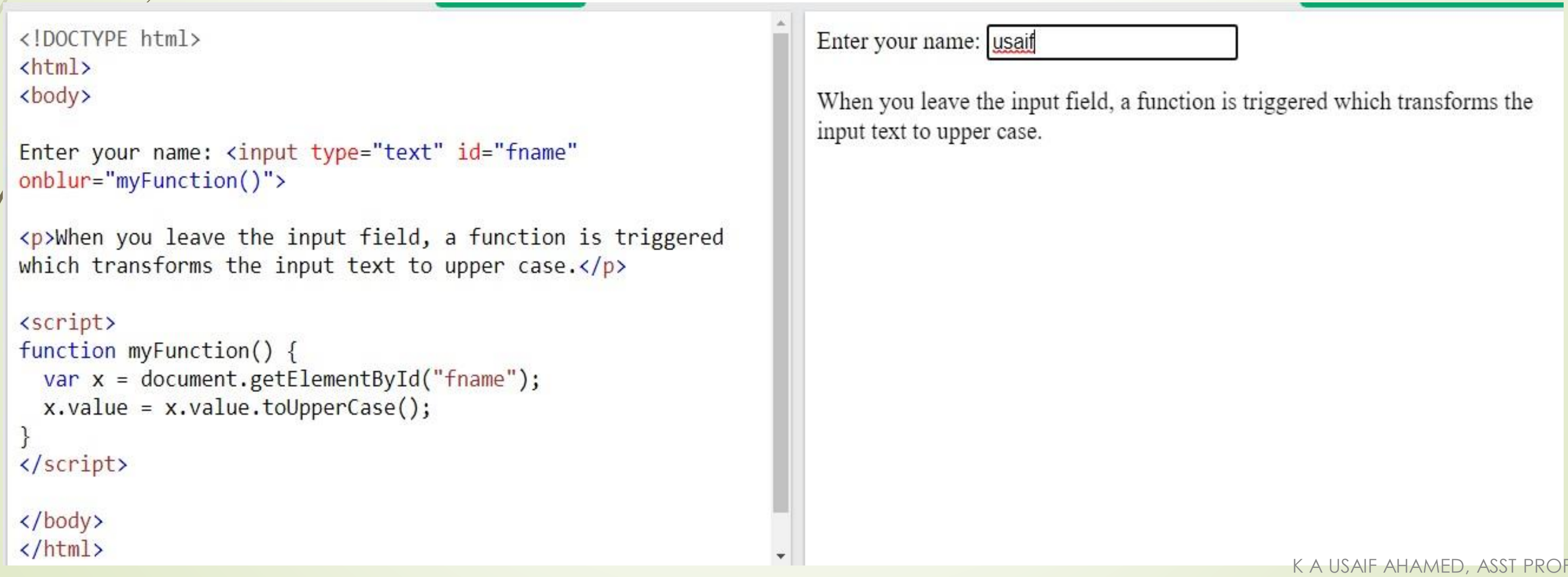
- Ex: document.title="This title is displayed now";

Onblur:

Document methods

Write(): Allows a string of text to be written to the document.

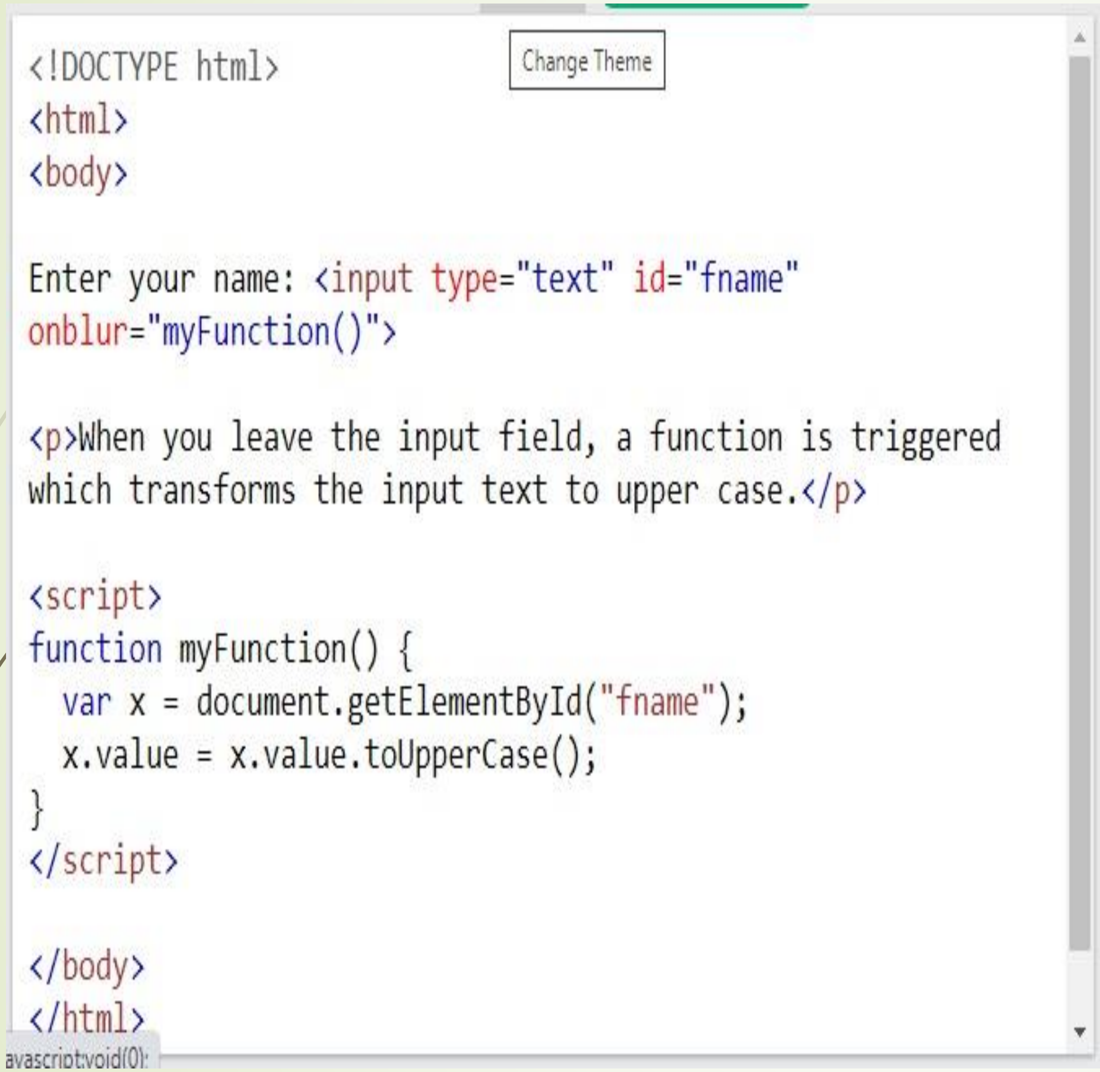
- Document.write (" <h1> Hello </h1>");



```
<!DOCTYPE html>
<html>
<body>

Enter your name: <input type="text" id="fname"
onblur="myFunction()">

<p>When you leave the input field, a function is triggered
which transforms the input text to upper case.</p>

<script>
function myFunction() {
  var x = document.getElementById("fname");
  x.value = x.value.toUpperCase();
}
</script>

</body>
</html>
```

Enter your name: usaif

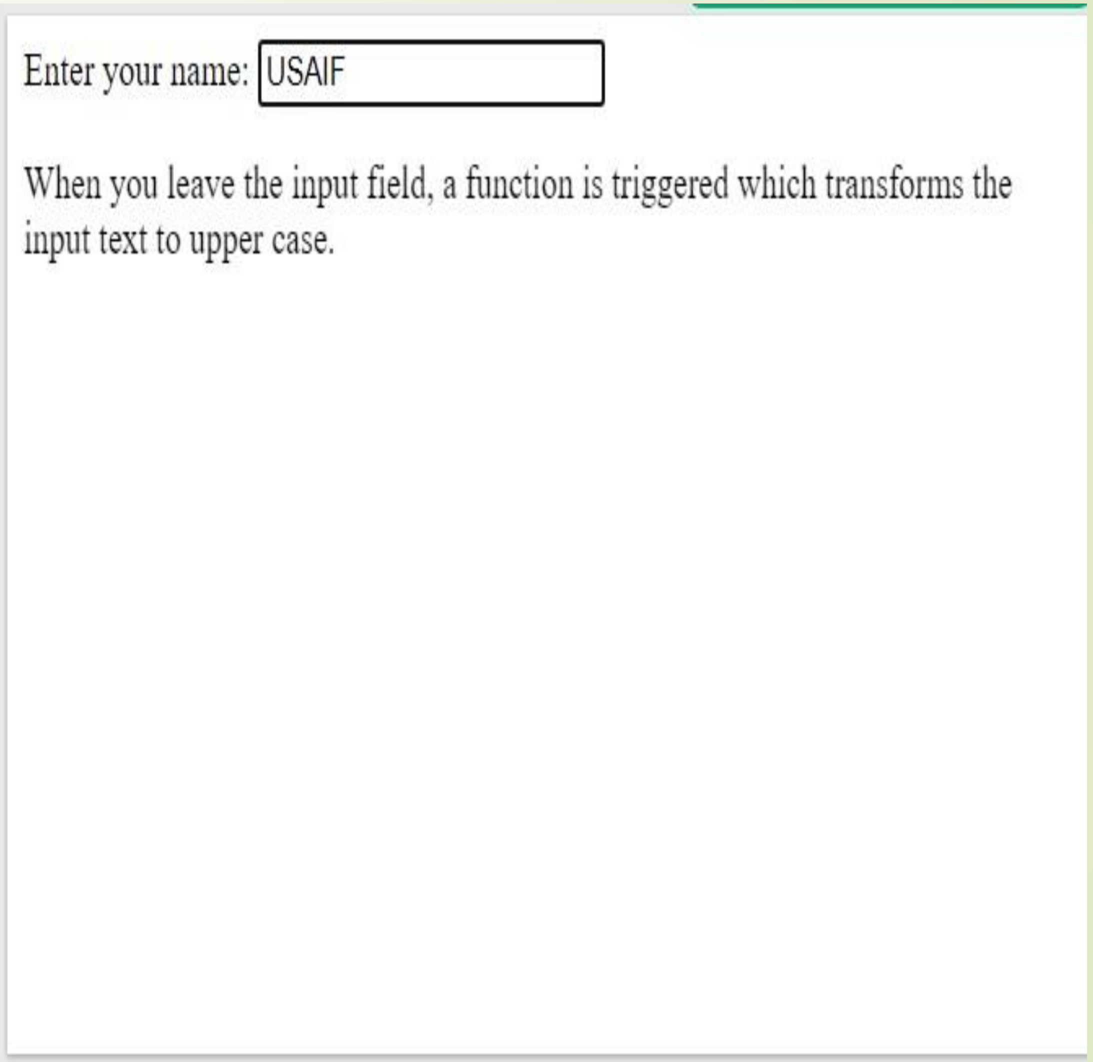
When you leave the input field, a function is triggered which transforms the input text to upper case.



Change Theme

```
<!DOCTYPE html>
<html>
<body>

Enter your name: <input type="text" id="fname"
onblur="myFunction()">

<p>When you leave the input field, a function is triggered
which transforms the input text to upper case.</p>

<script>
function myFunction() {
  var x = document.getElementById("fname");
  x.value = x.value.toUpperCase();
}
</script>

</body>
</html>
```

avascript:void(0);

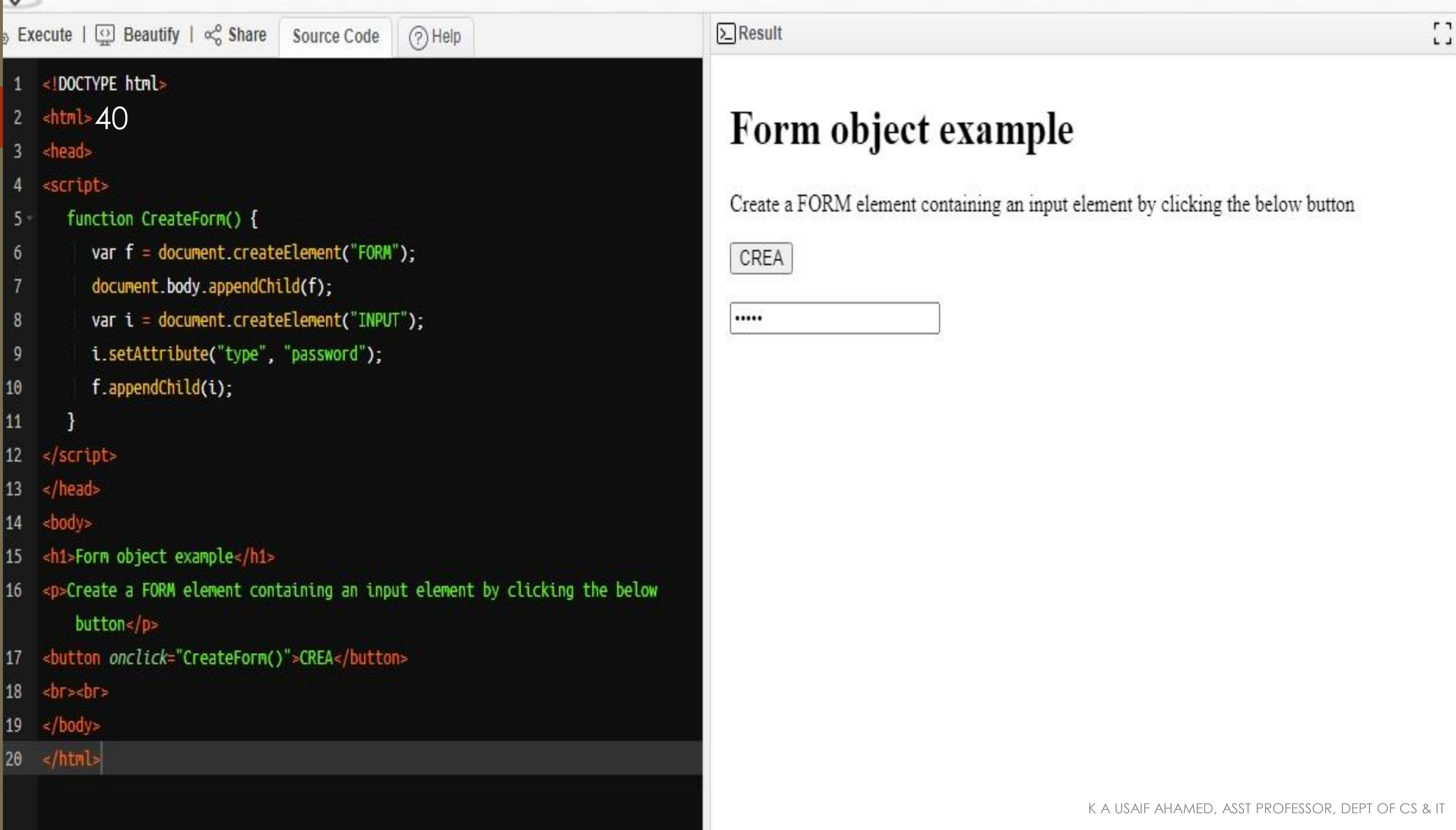


Enter your name: USAIF

When you leave the input field, a function is triggered which transforms the input text to upper case.

Forms Object

- The <form> element has two important attributes: action and method.
- The action attribute specifies a URL that will process the form submission. In this example, the action is the /signup URL.
- The method attribute specifies the HTTP method to submit the form with. Usually, the method is either post or get.
- Generally, you use the get method when you want to retrieve data from the server and the post method when you want to change data on the server.
- The HTMLFormElement element also provides the following useful methods:
- submit() – submit the form.
- reset() – reset the form.



```
<!DOCTYPE html>
<html>
<head>
<script>
  function CreateForm() {
    var f = document.createElement("FORM");
    document.body.appendChild(f);
    var i = document.createElement("INPUT");
    i.setAttribute("type", "password");
    f.appendChild(i);
  }
</script>
</head>
<body>
<h1>Form object example</h1>
<p>Create a FORM element containing an input element by clicking the below
   button</p>
<button onclick="CreateForm()">CREA</button>
<br><br>
</body>
</html>
```

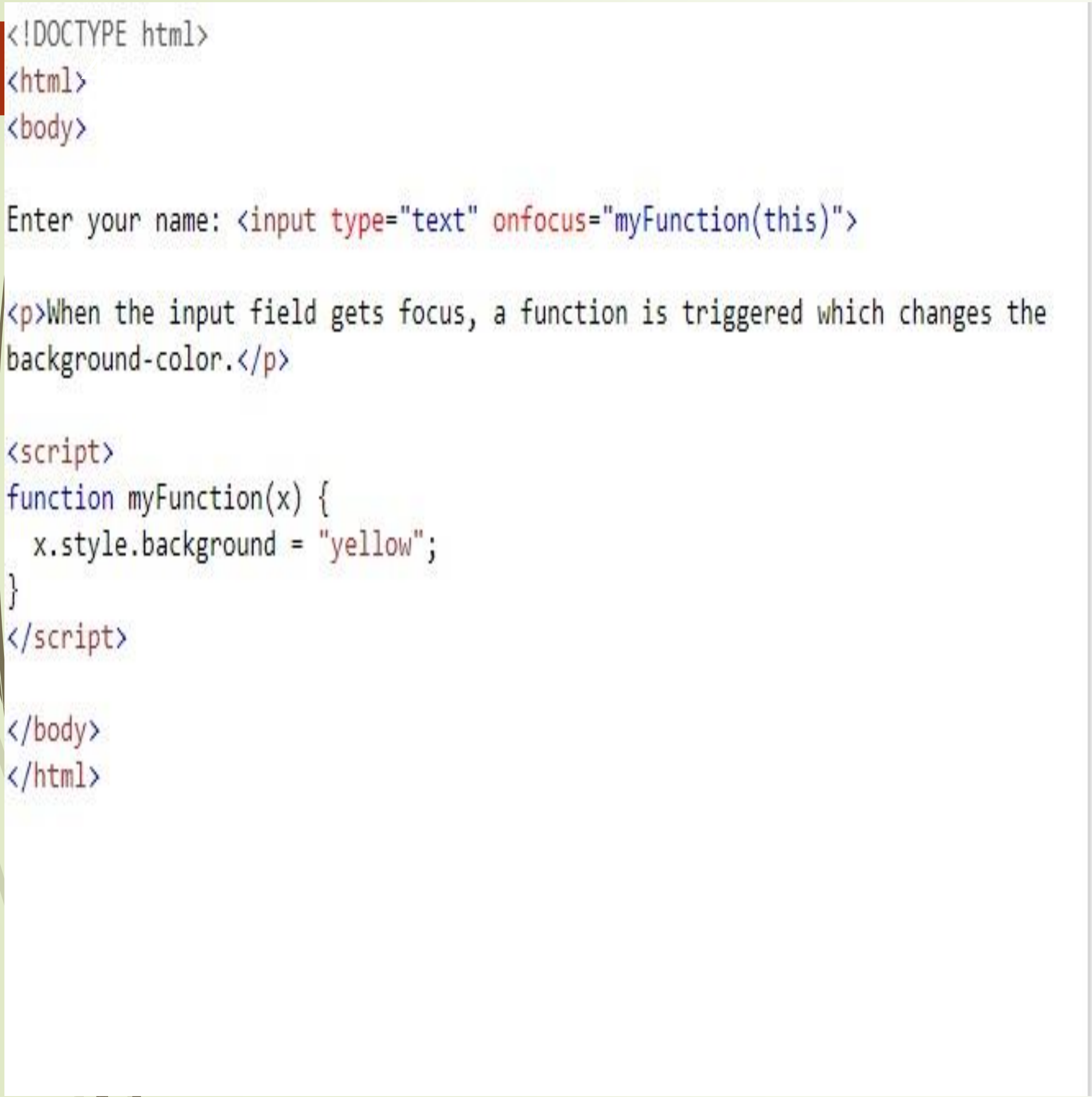
Form object example

Create a FORM element containing an input element by clicking the below button

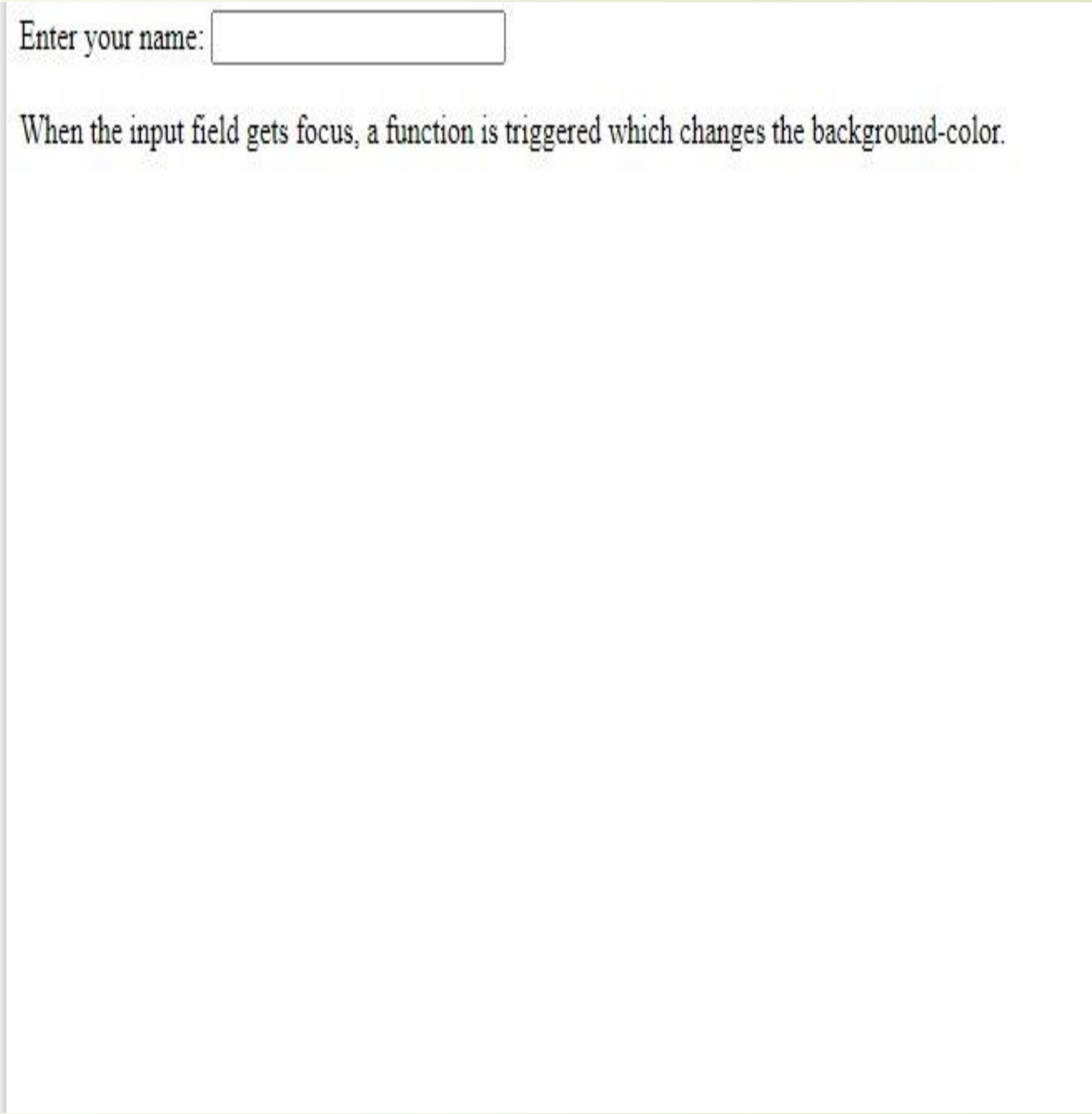
CREA

•••••

TEXTBOXES



```
<!DOCTYPE html>
<html>
<body>

Enter your name: <input type="text" onfocus="myFunction(this)">

<p>When the input field gets focus, a function is triggered which changes the
background-color.</p>

<script>
function myFunction(x) {
  x.style.background = "yellow";
}
</script>

</body>
</html>
```



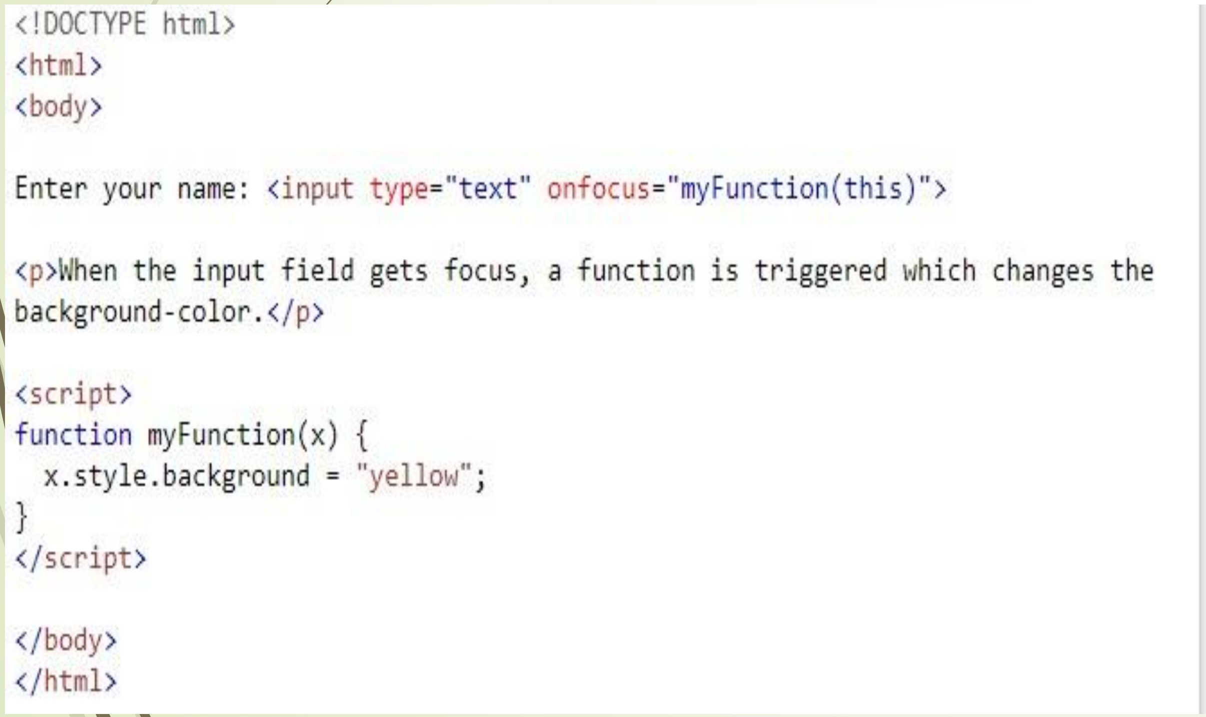
Enter your name:

When the input field gets focus, a function is triggered which changes the background-color.

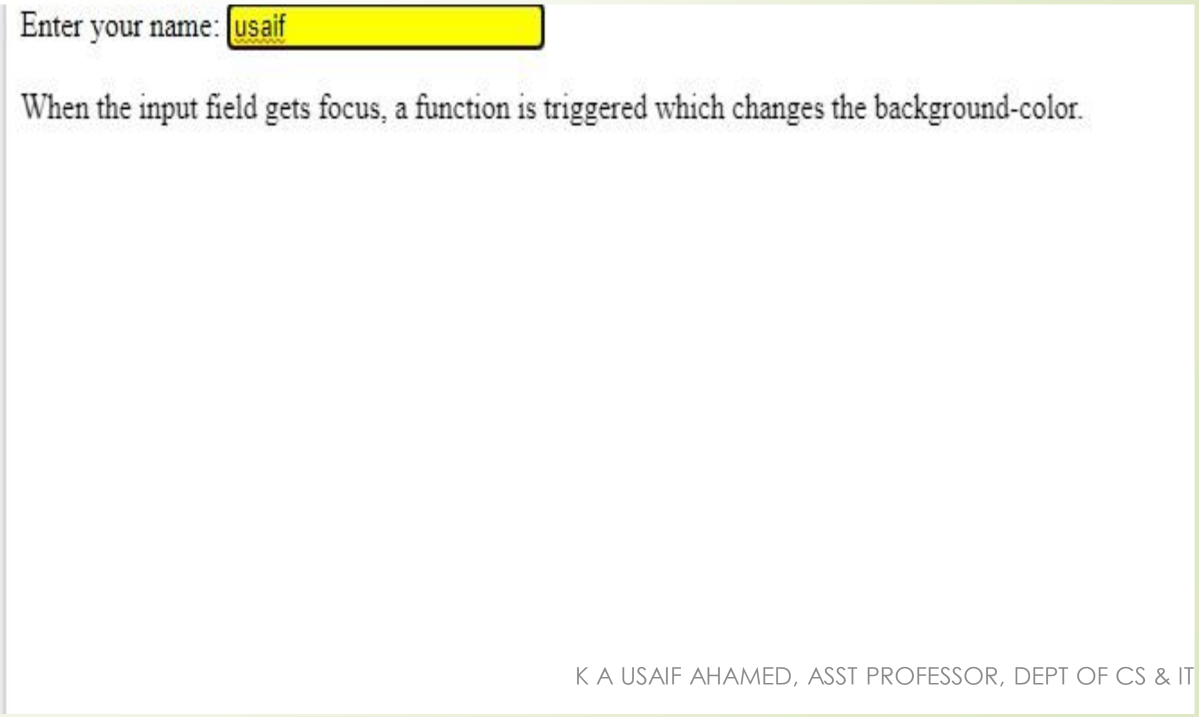
The Button element

- The <button> HTML element is an interactive element activated by a user with a mouse, keyboard, finger, voice command, or other assistive technology. Once activated, it then performs a programmable action, such as submitting a form or opening a dialog.

Methods	Description
blur()	Removes focus away from the button.
click()	Simulates a user clicking on the form button.
focus()	Sets focus on the button.



```
<!DOCTYPE html>
<html>
<body>

Enter your name: <input type="text" onfocus="myFunction(this)">

<p>When the input field gets focus, a function is triggered which changes the
background-color.</p>

<script>
function myFunction(x) {
  x.style.background = "yellow";
}
</script>

</body>
</html>
```

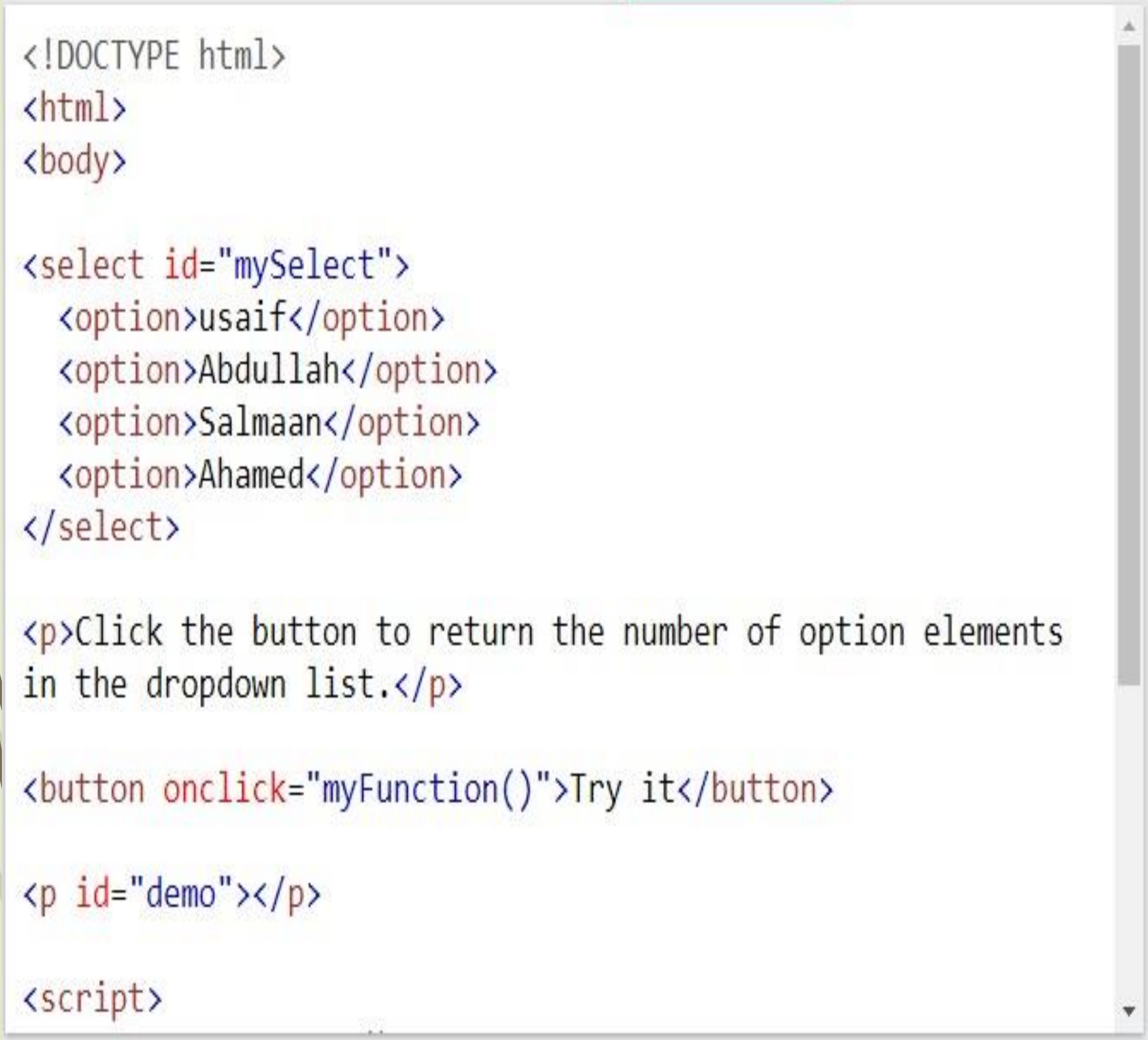


Enter your name: usaif

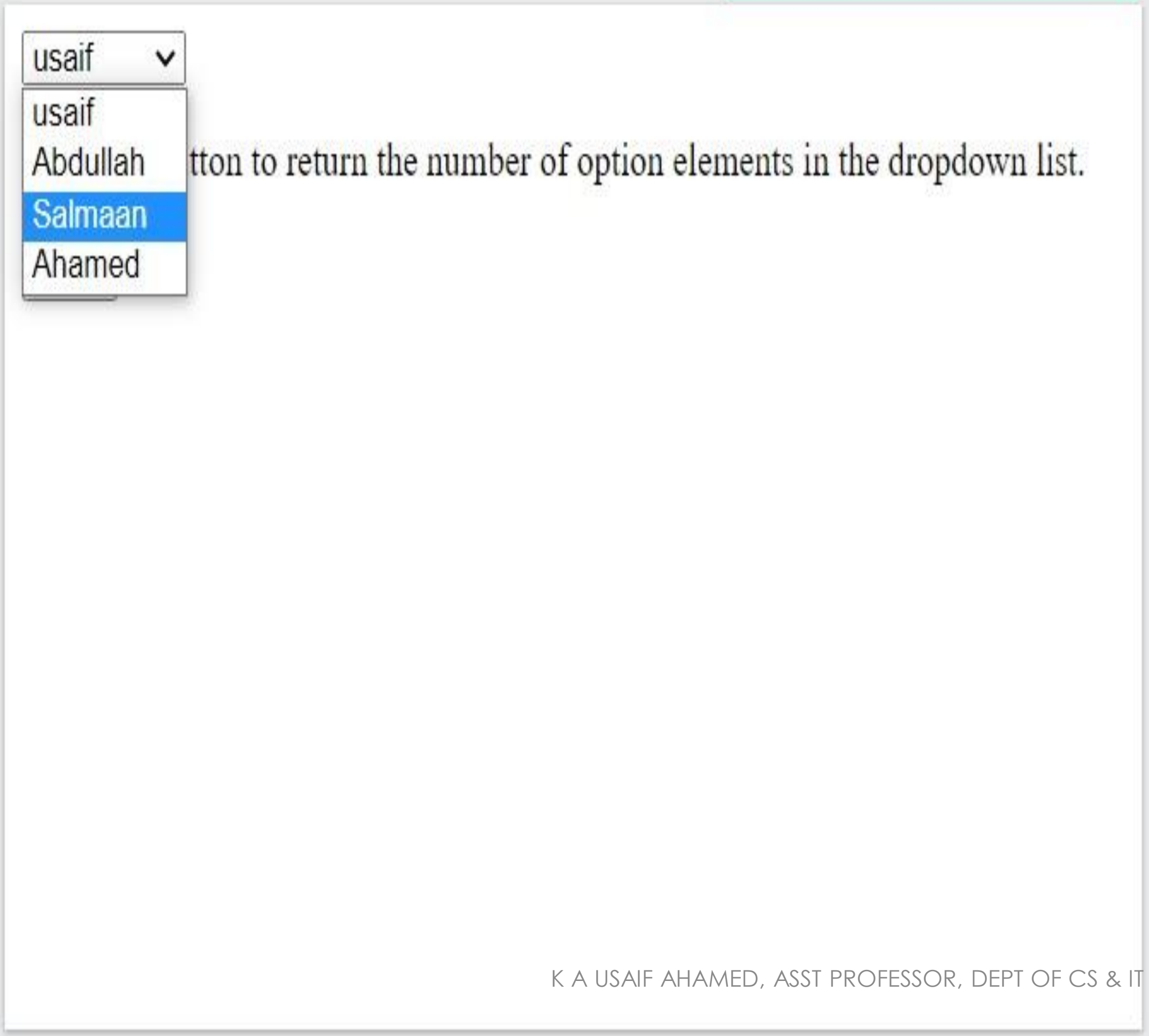
When the input field gets focus, a function is triggered which changes the background-color.

Select

- Form SELECT elements (<select>) within your form can be accessed and manipulated in JavaScript via the corresponding Select object.



```
<!DOCTYPE html>
<html>
<body>

<select id="mySelect">
  <option>usaif</option>
  <option>Abdullah</option>
  <option>Salmaan</option>
  <option>Ahamed</option>
</select>

<p>Click the button to return the number of option elements
in the dropdown list.</p>

<button onclick="myFunction()">Try it</button>

<p id="demo"></p>

<script>
```



```
usaif
usaif
Abdullah
Salmaan
Ahamed
tton to return the number of option elements in the dropdown list.
```

Properties	Description
disabled	Boolean value that sets/ returns whether this SELECT element is disabled.
form	References the form that contains the selection list in question.
length	Returns the number of options in the SELECT element: document.getElementById("mymenu").length //returns number of options
multiple	Boolean that indicates whether this SELECT allows more than one option to be selected simultaneously.
name	Reflects the name of the SELECT element (the name attribute).
options[]	An HTMLCollection of Options objects containing each option within this SELECT element.

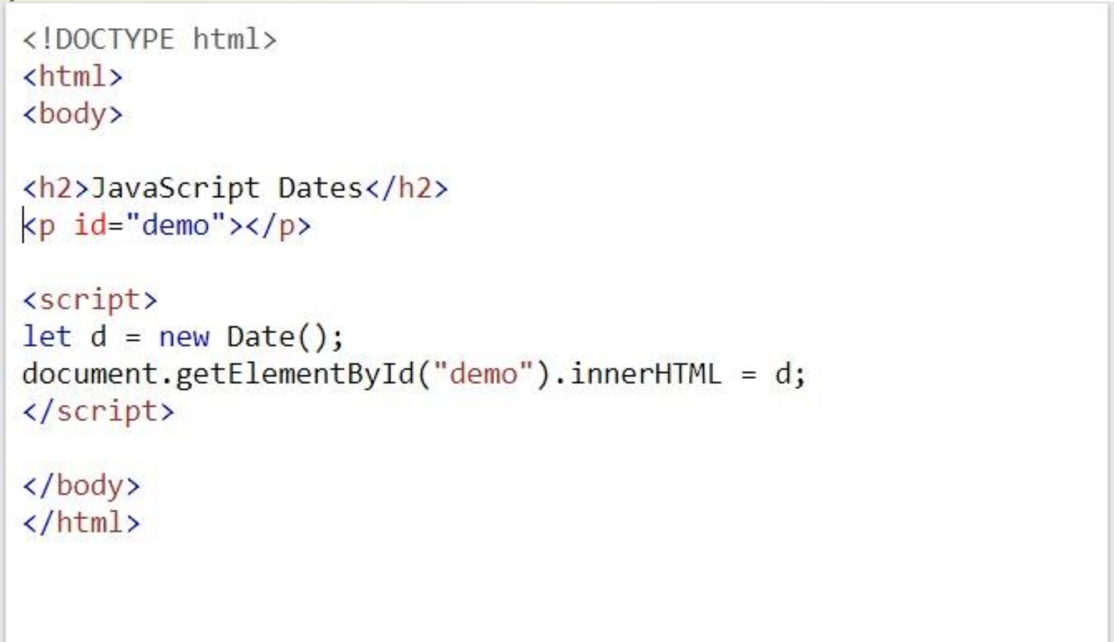
Date Object

- The **JavaScript date** object can be used to get year, month and day. You can display a timer on the webpage by the help of JavaScript date object.
- You can use different Date constructors to create date object. It provides methods to get and set day, month, year, hour, minute and seconds.

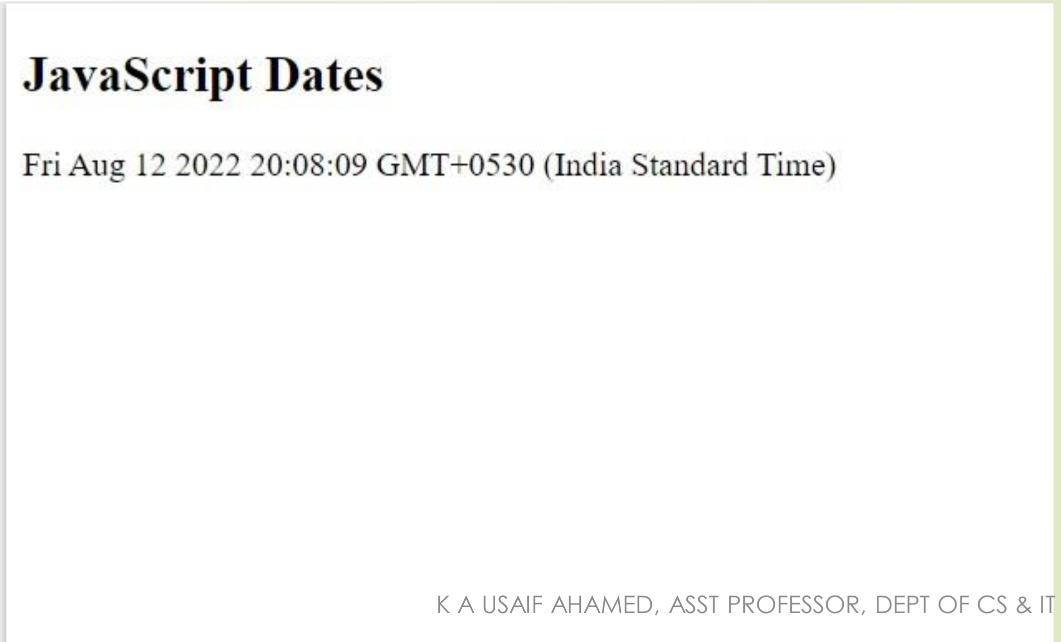
Constructor

You can use 4 variant of Date constructor to create date object.

1. Date()
2. Date(milliseconds)
3. Date(dateString)
4. Date(year, month, day, hours, minutes, seconds, milliseconds)



```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Dates</h2>
<p id="demo"></p>

<script>
let d = new Date();
document.getElementById("demo").innerHTML = d;
</script>

</body>
</html>
```



JavaScript Dates

```
Fri Aug 12 2022 20:08:09 GMT+0530 (India Standard Time)
```

Methods	Description
getDate()	It returns the integer value between 1 and 31 that represents the day for the specified date on the basis of local time.
getDay()	It returns the integer value between 0 and 6 that represents the day of the week on the basis of local time.
getFullYears()	It returns the integer value that represents the year on the basis of local time.
getHours()	It returns the integer value between 0 and 23 that represents the hours on the basis of local time.
getMilliseconds()	It returns the integer value between 0 and 999 that represents the milliseconds on the basis of local time.
getMinutes()	It returns the integer value between 0 and 59 that represents the minutes on the basis of local time.
getMonth()	It returns the integer value between 0 and 11 that represents the month on the basis of local time.
getSeconds()	It returns the integer value between 0 and 60 that represents the seconds on the basis of local time.

Mathematical Operator

- The **JavaScript math** object provides several constants and methods to perform mathematical operation.

Methods	Description
cos()	It returns the cosine of the given number.
exp()	It returns the exponential form of the given number.
log()	It returns natural logarithm of a number.
max()	It returns maximum value of the given numbers.
min()	It returns minimum value of the given numbers.
pow()	It returns value of base to the power of exponent.
random()	It returns random number between 0 (inclusive) and 1 (exclusive).
round()	It returns closest integer value of the given number.
sin()	It returns the sine of the given number.
sqrt()	It returns the square root of the given number

```
<html>
<body>
Square Root of 4 is: <span id="p1"></span>
<script>
document.getElementById('p1').innerHTML=Math.sqrt(4);
</script>
</body>
</html>
```

Outpt: Square Root of 4 is: 2

- The **JavaScript string** is an object that represents a sequence of characters.

Methods	Description
charAt()	It provides the char value present at the specified index.
concat()	It provides a combination of two or more strings.
search()	It searches a specified regular expression in a given string and returns its position if a match occurs.
match()	It searches a specified regular expression in a given string and returns that regular expression if a match occurs.
replace()	It replaces a given string with the specified replacement.
slice()	It is used to fetch the part of the given string. It allows us to assign positive as well negative index.
toLowerCase()	It converts the given string into lowercase letter.
toUpperCase()	It converts the given string into uppercase letter.
split()	It splits a string into substring array, then returns that newly created array.
trim()	It trims the white space from the left and right side of the string.

- The JavaScript String charAt() method returns the character at the given index.

```
<script>
var str="javascript";
document.write(str.charAt(2));
</script>
```

Output: v

Regular Expressions

- The JavaScript RegExp class represents regular expressions, and both String and RegExp define methods that use regular expressions to perform powerful pattern-matching and search-and-replace functions on text.

Syntax

- A regular expression could be defined with the RegExp () constructor, as follows −

var pattern = new RegExp(pattern, attributes);

or simply

var pattern = /pattern/attributes;

Here is the description of the parameters −

- pattern − A string that specifies the pattern of the regular expression or another regular expression.
- attributes − An optional string containing any of the "g", "i", and "m" attributes that specify global, case-insensitive, and multi-line matches, respectively.

Brackets

- Brackets ([]) have a special meaning when used in the context of regular expressions. They are used to find a range of characters.

Sr.No.	Expression & Description
1	[...] Any one character between the brackets.
2	[^...] Any one character not between the brackets.
3	[0-9] It matches any decimal digit from 0 through 9.
4	[a-z] It matches any character from lowercase a through lowercase z.
5	[A-Z] It matches any character from uppercase A through uppercase Z.
6	[a-Z] It matches any character from lowercase a through uppercase Z.

The ranges shown above are general; you could also use the range [0-3] to match any decimal digit ranging from 0 through 3, or the range [b-v] to match any lowercase character ranging from **b** through **v**.

Sr.No.	Expression & Description
1	p+ It matches any string containing one or more p's.
2	p* It matches any string containing zero or more p's.
3	p? It matches any string containing at most one p.
4	p{N} It matches any string containing a sequence of N p's
5	p{2,3} It matches any string containing a sequence of two or three p's.
6	p{2, } It matches any string containing a sequence of at least two p's.
7	p$ It matches any string with p at the end of it.
8	^p It matches any string with p at the beginning of it.

Quantifiers

The frequency or position of bracketed character sequences and single characters can be denoted by a special character. Each special character has a specific connotation. The +, *, ?, and $ flags all follow a character sequence.

JavaScript Array

- **JavaScript array** is an object that represents a collection of similar type of elements.

There are 3 ways to construct array in JavaScript

1. By array literal
2. By creating instance of Array directly (using new keyword)
3. By using an Array constructor (using new keyword)

1.JavaScript array literal

The syntax of creating array using array literal is given below:

```
var arrayname=[value1,value2.....valueN];
```

- **Ex:** var emp=["usaif","salmaan","Abdullah"];

2. JavaScript Array directly (new keyword)

The syntax of creating array directly is given below:

```
var arrayname=new Array();
```

Here, **new keyword** is used to create instance of array.

EX:

```
var emp = new Array();
emp[0] = "Abdullah";
emp[1] = "Usaif";
emp[2] = "Salmaan";
```

3. JavaScript array constructor (new keyword)

Here, you need to create instance of array by passing arguments in constructor so that we don't have to provide value explicitly.

```
<script>
var emp=new Array("Jai","Vijay","Smith");
for (i=0;i<emp.length;i++){
document.write(emp[i] + "<br>");
}
</script>
```

Methods	Description
concat()	It returns a new array object that contains two or more merged arrays.
find()	It returns the value of the first element in the given array that satisfies the specified condition.
indexOf()	It searches the specified element in the given array and returns the index of the first match.
isArray()	It tests if the passed value ia an array.
join()	It joins the elements of an array as a string.
pop()	It removes and returns the last element of an array.
push()	It adds one or more elements to the end of an array.
reverse()	It reverses the elements of given array.
sort()	It returns the element of the given array in a sorted order.

WEB TECHNOLOGY

UNIT-III

VB SCRIPT

K A USAIF AHAMED

ASSISTANT PROFESSOR OF CS & IT

JAMAL MOHAMED COLLEGE

What is VBScript?

- VBScript is a scripting language
- A scripting language is a lightweight programming language
- VBScript is a light version of Microsoft's programming language Visual Basic

How Does it Work?

When a VBScript is inserted into a HTML document, the Internet browser will read the HTML and interpret the VBScript. The VBScript can be executed immediately, or at a later event.

```
<html>
<head>
</head>
<body>
<script type="text/vbscript">
document.write("Hello from VBScript!")
</script>
</body>
</html>
```

Where to Put the VBScript

- Scripts 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.

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

```
<html>
<head>
    <script type="text/vbscript">
        some statements
   </script>
</head>
```

- **Scripts in the body section:** Scripts to be executed when the page loads go in the body section. When you place a script in the body section it generates the content of the page:

```
<head></head>
<body>
<script type="text/vbscript">
  some statements
</script></body>
```

What is a Variable?

- A variable is a "container" for information you want to store. A variable's value can change during the script. You can refer to a variable by name to see its value or to change its value. In VBScript, all variables are of type *variant*, that can store different types of data.

Rules for Variable Names:

- Must begin with a letter
- Cannot contain a period (.)
- Cannot exceed 255 characters

Declaring Variables

- You can declare variables with the Dim, Public or the Private statement. Like this:

```
dim name

name=some value
```

Assigning Values to Variables

- You assign a value to a variable like this:

```
name="USAIF"
i=200
```

Array Variables

- Sometimes you want to assign more than one value to a single variable. Then you can create a variable that can contain a series of values. This is called an array variable. The declaration of an array variable uses parentheses ( ) following the variable name. In the following example, an array containing 3 elements is declared:

```
dim names(2)

names(0)=“USAIF"
names(1)=“AHAMED"
names(2)=“SALMAN“
```

- The number shown in the parentheses is 2. We start at zero so this array contains 3 elements. This is a fixed-size array. You assign data to each of the elements of the array like this:

What is an operator?

- Let's take an expression 4 + 5 is equal to 9. Here, 4 and 5 are called operands and + is called the operator. VBScript language supports following types of operators
- Arithmetic Operators
- Comparison Operators
- Logical (or Relational) Operators
- Concatenation Operators

1.The Arithmetic Operators

Assume variable A holds 5 and variable B holds 10, then

Operator	Description	Example
+	Adds two operands	A + B will give 15
-	Subtracts second operand from the first	A - B will give -5
*	Multiply both operands	A * B will give 50
/	Divide numerator by denumerator	B / A will give 2
%	Modulus Operator and remainder of after an integer division	B MOD A will give 0
^	Exponentiation Operator	B ^ A will give 100000

The Comparison Operators

- There are following comparison operators supported by VBScript language
- Assume variable A holds 10 and variable B holds 20

Operator	Description	Example
=	Checks if the value of two operands are equal or not, if yes then condition becomes true.	(A == B) is False.
<>	Checks if the value of two operands are equal or not, if values are not equal then condition becomes true.	(A <> B) is True.
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(A > B) is False.
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(A < B) is True.
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(A >= B) is False.
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(A <= B) is True.

The Logical Operators

- There are following logical operators supported by VBScript language
- Assume variable A holds 10 and variable B holds 0, then

Operator	Description	Example
AND	Called Logical AND operator. If both the conditions are True, then Expression becomes True.	a<>0 AND b<>0 is False.
OR	Called Logical OR Operator. If any of the two conditions is True, then condition becomes True.	a<>0 OR b<>0 is true.
NOT	Called Logical NOT Operator. It reverses the logical state of its operand. If a condition is True, then the Logical NOT operator will make it False.	NOT(a<>0 OR b<>0) is false.
XOR	Called Logical Exclusion. It is the combination of NOT and OR Operator. If one, and only one, of the expressions evaluates to True, result is True.	(a<>0 XOR b<>0) is true.

The Concatenation Operators

- There are following Concatenation operators supported by VBScript language −
- Assume variable A holds 5 and variable B holds 10 then

Operator	Description	Example
+	Adds two Values as Variable Values are Numeric	A + B will give **15**
&	Concatenates two Values	**A & B** will give **510**

- Assume variable A = “USAIF" and variable B=“AHAEMD", then

Operator	Description	Example
+	Concatenates two Values	A + B will give USAIFAHAMED
&	Concatenates two Values	A & B will give USAIFAHAMED

```
<html>
  <body>
    <script language = "vbscript" type = "text/vbscript">
      Dim a = 5
      Dim b = 10
      Dim c

      c = a+b
      Document.write ("Addition Result is " &c)
      Document.write ("<br></br>")

      c = a-b
      Document.write ("Subtraction Result is " &c)
      Document.write ("<br></br>")

      c = a*b
      Document.write ("Multiplication Result is " &c)
      Document.write ("<br></br>")

      c = b/a
      Document.write ("Division Result is " &c)
      Document.write ("<br></br>")

      c = b MOD a
      Document.write ("Modulus Result is " &c)
      Document.write ("<br></br>")

      c = b^a
      Document.write ("Exponentiation Result is " &c)
      Document.write ("<br></br>")
    </script>
  </body>
</html>
```

```
Addition Result is 15

Subtraction Result is -5

Multiplication Result is 50

Division Result is 2

Modulus Result is 0

Exponentiation Result is 100000
```

```
<html>
   <body>
      <script language="vbscript" type="text/vbscript">
      Dim a = 10,b = 20

      If a==b Then
         Document.write ("Operator Line 1 : True")
       Else
         Document.write ("Operator Line 1 : False")
      End If

      If a>=b Then
         Document.write ("Operator Line 2 : True")
      Else
         Document.write ("Operator Line 2 : False")
      End If

      If a<=b Then
         Document.write ("Operator Line 3 : True")
      Else
         Document.write ("Operator Line 3 : False")
      End If
      </script>
</body>
</html>
```

What is a Function?

- A function is a group of reusable code which can be called anywhere in your program. This eliminates the need of writing same code over and over again. This will enable programmers to divide a big program into a number of small and manageable functions. Apart from inbuilt Functions, VBScript allows us to write user-defined functions as well.

Function Definition

A Function procedure:

- Is a series of statements, enclosed by the Function and End Function statements.
- It can perform actions and **can return** a value.
- It can take arguments that are passed to it by a calling procedure.
- without arguments, must include an empty set of parentheses ().
- returns a value by assigning a value to its name.

```
<html>
<body>
<script language = "vbscript" type = "text/vbscript">
         Function Functionname(parameter-list)
             statement 1
             statement 2
             statement 3
               .......
              statement n
        End Function

</script>
</body>
</html>
```

A Sub procedure:

- Is a series of statements, enclosed by the Sub and End Sub statements
- It can perform actions, but **does not return** a value
- It can take arguments that are passed to it by a calling procedure
- without arguments, must include an empty set of parentheses ()

```
Sub mysub()
 some statements
End Sub
```

or

```
Sub mysub(argument1,argument2)
 some statements
End Sub
```

```
<html>
 <body>
 <script language = "vbscript" type = "text/vbscript">
        Sub sayHello()
           msgbox("Hello there")
        End Sub

     </script>
   </body>
</html>
```

Calling Procedures

- To invoke a Procedure somewhere later in the script, you would simply need to write the name of that procedure with or without the **Call** keyword.

```
<html>
 <body>
   <script language = "vbscript" type = "text/vbscript">
     Sub display()
       msgbox(“Welcome to Jamal")
         End Sub
              display()
      </script>
   </body>
</html>
```

```
<script language = "vbscript" type = "text/vbscript">
         Function show()
            msgbox(“Welcome to III-BSC")
         End Function
      </script>
```

Calling a Function

- To invoke a function somewhere later in the script, you would simple need to write the name of that function with the **Call** keyword

```
<html>
   <body>
      <script language = "vbscript" type = "text/vbscript">
         Function sayHello()
            msgbox("Hello there")
         End Function
         Call sayHello()
      </script>
   </body>
</html>
```

Function Parameters

The function without a parameter, but there is a facility to pass different parameters while calling a function. These passed parameters can be captured inside the function and any manipulation can be done over those parameters. The Functions are called using the Call Keyword.

```
<!DOCTYPE html>
<html>
   <body>
      <script language = "vbscript" type = "text/vbscript">
         Function display(name, age)
            msgbox( name & " is " & age & " years old.")
         End Function
         Call display(“usaif", 27)
      </script>
   </body>
</html>
```

VBScript we have four conditional statements:

- **if statement -** use this statement if you want to execute a set of code when a condition is true
- **if...then...else statement -** use this statement if you want to select one of two sets of lines to execute
- **if...then...elseif statement -** use this statement if you want to select one of many sets of lines to execute
- **select case statement -** use this statement if you want to select one of many sets of lines to execute

- **If....Then.....Else**

You should use the If...Then...Else statement if you want to

- execute some code if a condition is true
- select one of two blocks of code to execute

- If you want to execute only **one** statement when a condition is true, you can write the code on one line

```
1. if i=10 then
2.    msgbox "Hello"
3. else
4.    msgbox "Goodbye"
5. end If
```

If....Then.....Elseif

```
if payment="Cash" then
   msgbox "You are going to pay cash!"
 elseif payment="Phone Pay" then
   msgbox "You are going to pay with Phone Pay."
 elseif payment="google pay" then
   msgbox "You are going to pay with Google Pay."
 else
   msgbox "Unknown method of payment."
end If
```

Select Case

- You can also use the SELECT statement if you want to select one of many blocks of code to execute:

```
select case payment
 case "Cash"
   msgbox "You are going to pay cash"
 case “Gpay"
   msgbox "You are going to pay with Gpay"
 case “PhonePay"
   msgbox "You are going to pay with Phone Pay"
 case Else
   msgbox "Unknown method of payment"
end select
```

Looping Statements

- Very often when you write code, you want to allow the same block of code to run a number of times. You can use looping statements in your code to do this.
- In VBScript we have four looping statements:

1. **For...Next statement** - runs statements a specified number of times.
2. **For Each...Next statement** - runs statements for each item in a collection or each element of an array
3. **Do...Loop statement** - loops while or until a condition is true
4. **While...Wend statement** - Do not use it - use the Do...Loop statement instead

1.For...Next Loop

- You can use a **For...Next** statement to run a block of code, when you know how many repetitions you want.
- You can use a counter variable that increases or decreases with each repetition of the loop, like this:

```
For i=1 to 10
  some code
Next
```

2. For Each...Next Loop

A **For Each...Next** loop repeats a block of code for each item in a collection, or for each element of an array.

```
dim fru(2)
fru(0)=“Mango”
fru(1)=“apple”
fru(2)=“Banana“
For Each x in fru
   document.write(x & "<br />")
Next
```

- The **For** step is executed first. This step allows you to initialize any loop control variables and increment the step counter variable.
- Secondly, the **condition** is evaluated. If it is true, the body of the loop is executed. If it is false, the body of the loop does not execute and flow of control jumps to the next statement just after the For Loop.
- After the body of the for loop executes, the flow of control jumps to the **Next** statement. This statement allows you to update any loop control variables. It is updated based on the step counter value.
- The condition is now evaluated again. If it is true, the loop executes and the process repeats itself (body of loop, then increment step, and then again condition). After the condition becomes false, the For Loop terminates.

```
<html>  <body>
 <script language = "vbscript" type = "text/vbscript">
  Dim  a = 10
 For i = 0 to a Step 2
  document.write("The value is i is : " & i)
 document.write("<br></br>")
   Next
    </script>
 </body></html>
```

```
The value is i is : 0

The value is i is : 2

The value is i is : 4

The value is i is : 6

The value is i is : 8

The value is i is : 10
```

3.Do…Loop

- You can use Do...Loop statements to run a block of code when you do not know how many repetitions you want. The block of code is repeated while a condition is true or until a condition becomes true.
- use the While keyword to check a condition in a Do...Loop statement

Example

```
Do While i>10
   some code
Loop
```

Repeating Code Until a Condition Becomes True

- You use the Until keyword to check a condition in a Do...Loop statement.

```
Do Until i=10
    some code
Loop
```

Exit a Do...Loop

You can exit a Do...Loop statement with the Exit Do keyword.

```
Do Until i=10
  i=i-1
 If i<10 Then Exit Do
Loop
```

4. While..Wend

The loop, if the condition is True, all statements are executed until **Wend** keyword is encountered.

- If the condition is false, the loop is exited and the control jumps to very next statement after **Wend** keyword.
- The syntax of a **While..Wend** loop in VBScript is

```
While condition(s)
    [statements 1]
    [statements 2]
     ...
    [statements n]
Wend
```

```
<script language = "vbscript" type = "text/vbscript">
      Dim Counter :  Counter = 10
   While Counter < 15
  Counter = Counter + 1
      document.write("The Current Value of the Counter is : " & Counter)
       document.write("<br></br>")
  Wend
      </script>
```

```
The Current Value of the Counter is : 11
The Current Value of the Counter is : 12
The Current Value of the Counter is : 13
The Current Value of the Counter is : 14
The Current Value of the Counter is : 15
```

Using Objects

- You include an object using the <OBJECT> tags and set its initial property values using <PARAM> tags. If you're a Visual Basic programmer, you'll recognize that using the <PARAM> tags is just like setting initial properties for a control on a form.

- For example, the following set of <OBJECT> and <PARAM> tags adds an ActiveX control, called Label, to a page.

```
<OBJECT
   classid="clsid:99B42120-6EC7-11CF-A6C7-00AA00A47DD2"
   id=lblActiveLbl
   width=250
   height=250
   align=left
   hspace=20
   vspace=0
>
<PARAM NAME="Alignment" VALUE="4">
<PARAM NAME="BackStyle" VALUE="0">
<PARAM NAME="Caption" VALUE="A Simple Label">
<PARAM NAME="FontName" VALUE="Verdana, Arial, Helvetica">
<PARAM NAME="FontSize" VALUE="20">
<PARAM NAME="FontBold" VALUE="1">
<PARAM NAME="FrColor" VALUE="0">
</OBJECT>
<FORM NAME="LabelControls">
<INPUT TYPE="TEXT" NAME="txtNewText" SIZE=25>
<INPUT TYPE="BUTTON" NAME="cmdChangeIt" VALUE="Change Text">
<INPUT TYPE="BUTTON" NAME="cmdRotate" VALUE="Rotate Label">
</FORM>

<SCRIPT LANGUAGE="VBScript">
<!--
Sub cmdChangeIt_onClick
   Dim TheForm
   Set TheForm = Document.LabelControls
   lblActiveLbl.Caption = TheForm.txtNewText.Value
End Sub
-->
</SCRIPT>
```

What are Cookies?

- Web Browsers and Servers use HTTP protocol to communicate and HTTP is a stateless protocol. But for a commercial website, it is required to maintain session information among different pages. For example, one user registration ends after completing many pages. But how to maintain user's session information across all the web pages. In many situations, using cookies is the most efficient method of remembering and tracking preferences, purchases, commissions and other information required for better visitor experience or site statistics.

Cookies are a plain text data record of 5 variable-length fields

1. **Expires** − The date the cookie will expire. If this is blank, the cookie will expire when the visitor quits the browser.
2. **Domain** − The domain name of your site.
3. **Path** − The path to the directory or web page that set the cookie. This may be blank if you want to retrieve the cookie from any directory or page.
4. **Secure** − If this field contains the word "secure", then the cookie may only be retrieved with a secure server. If this field is blank, no such restriction exists.
5. **Name=Value** − Cookies are set and retrieved in the form of key and value pairs.

- Cookies were originally designed for CGI programming and cookies' data is automatically transmitted between the web browser and web server, so CGI scripts on the server can read and write cookie values that are stored on the client.
- VBScript can also manipulate cookies using the cookie property of the *Document* object. VBScript can read, create, modify and delete the cookie or cookies that apply to the current web page.

Storing Cookies

The simplest way to create a cookie is to assign a string value to the *document.cookie* object,

which looks like this:

Syntax

```
document.cookie = "key1 = value1;key2 = value2;expires = date“
```

Here *expires* attribute is optional. If you provide this attribute with a valid date or time,

then cookie will expire at the given date or time and after that cookies' value will not be accessible.

```
<html>  <head>
 <script type = "text/vbscript">
  Function WriteCookie
     If document.myform.customer.value = "" Then
         msgbox "Enter some value!"
     Else
   cookievalue = (document.myform.customer.value)
   document.cookie = "name = " + cookievalue
          msgbox "Setting Cookies : " & "name = " & cookievalue
     End If
   End Function
  </script>  </head>
 <body>
<form name = "myform" action = "">
 Enter name: <input type = "text" name = "customer"/>
  <input type = "button" value = "Set Cookie" onclick = "WriteCookie()"/>
 </form>
</body></html>
```

WEB TECHNOLOGY

UNIT-IV

K.A.USAIF AHAMED

ASSISTANT PROFESSOR

DEPARTMENT OF CS & IT

DHTML stands for Dynamic Hypertext Markup language

Dynamic HTML

- Dynamic HTML is not a markup or programming language but it is a term that combines the features of various web development technologies for creating the web pages dynamic and interactive.
- The DHTML application was introduced by Microsoft with the release of the 4th version of IE (Internet Explorer) in 1997.
- DHTML Technologies

HTML 4

- HTML supports JavaScript
- HTML supports the Document Object Model (**DOM**)
- HTML supports HTML Events
- HTML supports Cascading Style Sheets (**CSS**)

JavaScript

- JavaScript is the scripting standard for HTML.
- DHTML is about using JavaScript to control, access and manipulate HTML elements.

HTML DOM

- The HTML DOM is the standard **Document Object Model** for HTML.
- DHTML is about using the DOM to access and manipulate HTML elements.

CSS

- CSS is the W3C standard style and layout model for HTML.
- CSS allows web developers to control the style and layout of web pages.
- HTML 4 allows dynamic changes to CSS.
- DHTML is about using JavaScript and DOM to change the style and positioning of HTML elements.

What is CSS

1. CSS stands for Cascading Style Sheets. It is a style sheet language which is used to describe the look and formatting of a document written in markup language.
2. It provides an additional feature to HTML. It is generally used with HTML to change the style of web pages and user interfaces. It can also be used with any kind of XML documents.
3. CSS is used along with HTML and JavaScript in most websites to create user interfaces for web applications and user interfaces for many mobile applications.

Why use CSS

These are the three major benefits of CSS:

1) Solves a big problem

Before CSS, tags like font, color, background style, element alignments, border and size had to be repeated on every web page. This was a very long process. For example: If you are developing a large website where fonts and color information are added on every single page, it will be become a long and expensive process. CSS was created to solve this problem.

2) Saves a lot of time

CSS style definitions are saved in external CSS files so it is possible to change the entire website by changing just one file.

3) Provide more attributes

CSS provides more detailed attributes than plain HTML to define the look and feel of the website.

CSS Syntax

- A CSS rule set contains a selector and a declaration block.

1. Selector: Selector indicates the HTML element you want to style. It could be any

tag like <h1>, <title> etc

2. Declaration Block: The declaration block can contain one or more declarations separated by a semicolon.

color: yellow;

font-size: 11 px;

Each declaration contains a property name and value, separated by a colon.

- **CSS selectors** are used *to select the content you want to style*. Selectors are the part of CSS rule set. CSS selectors select HTML elements according to its id, class, type, attribute etc.

There are several different types of selectors in CSS.

1. **CSS Element Selector**
2. **CSS Id Selector**
3. **CSS Class Selector**
4. **CSS Universal Selector**
5. **CSS Group Selector**

1) CSS Element Selector

The element selector selects the HTML element by name.

```
<head> <style>
p{
    text-align: center;
    color: blue;
}
</style>  </head>
<body>
<p>Usaif Ahamed<p>
</body>
</html>
```

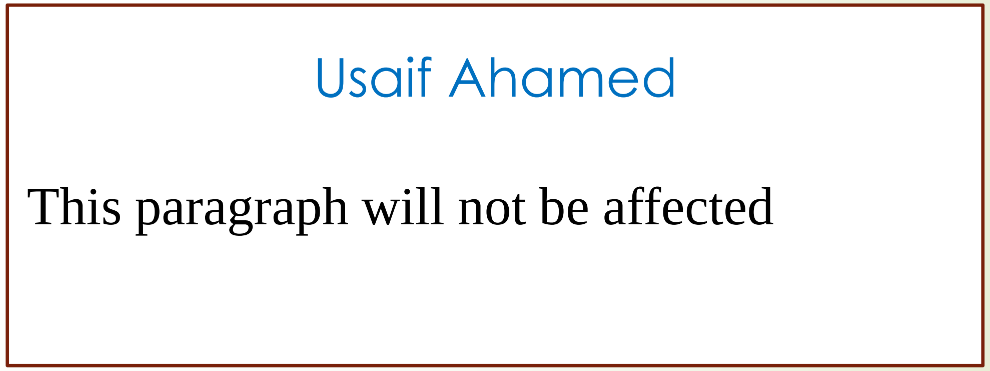
```
Usaif Ahamed
```

2) CSS Id Selector

- The id selector selects the id attribute of an HTML element to select a specific element. An id is always unique within the page so it is chosen to select a single, unique element.

- It is written with the hash character (#), followed by the id of the element.

```
<head> <style>
#para1 {
    text-align: center;
    color: blue;
}
</style>  </head>
<body>
<p id="para1">Usaif Ahamed</p>
<p>This paragraph will not be affected.</p>
```



Usaif Ahamed

This paragraph will not be affected

3) CSS Class Selector

- The class selector selects HTML elements with a specific class attribute. It is used with a period character . (full stop symbol) followed by the class name.
- *Note: A class name should not be started with a number.*

```
<style>
.center {
    text-align: center;
    color: blue;
}
</style>
</head>
<body>
<h1 class="center">This heading is blue and center-aligned.</h1>
```

This heading is blue and center-aligned

CSS Class Selector for specific element

- If you want to specify that only one specific HTML element should be affected then you should use the element name with class selector.

```
<style>
p.center {
    text-align: center;
    color: blue;
}
</style>
</head>
<body>
<h1 class="center">This heading is not affected</h1>
<p class="center">This paragraph is blue and center-aligned.</p>
```

```
This heading is not affected
        This heading is blue and center-aligned
```

4) CSS Universal Selector

- The universal selector is used as a wildcard character. It selects all the elements on the pages.

```
<style>
* {
   color: green;
   font-size: 20px;
}
</style>
</head>
<body>
<h2>This is heading</h2>
<p>This style will be applied on every paragraph.</p>
```

This is heading
This style will be applied on every paragraph

5) CSS Group Selector

- The grouping selector is used to select all the elements with the same style definitions.
- Grouping selector is used to minimize the code. Commas are used to separate each selector in grouping.

- Let's see the CSS code without group selector.

```
h1 {
    text-align: center;
    color: blue;
}
h2 {
    text-align: center;
    color: blue;
}
p {
    text-align: center;
    color: blue;
}
```

There are three ways to insert CSS in HTML documents.

1. Inline CSS
2. Internal CSS
3. External CSS

1) **Inline** CSS is used to apply CSS on a single line or element.

For example: `<p style="color:blue">Hello CSS</p>`

2) **Internal** CSS is used to apply CSS on a single document or page. It can affect all the elements of the page. It is written inside the style tag within head section of html.

For example:

```
        <style>
    p{color:blue}
    </style>
```

3) **External** CSS is used to apply CSS on multiple pages or all pages. Here, we write all the CSS code in a css file

For example:

p{color:blue}

You need to link this style.css file to your html pages like this:

```
<link rel="stylesheet" type="text/css" href="style.css">
```

Inline CSS

1. We can apply CSS in a single element by inline CSS technique.
2. The inline CSS is also a method to insert style sheets in HTML document. This method have some advantages of style sheets so it is advised to use this method carefully.
3. If you want to use inline CSS, you should use the style attribute to the relevant tag.

- Syntax:
- <**htmltag** style="cssproperty1:value; cssproperty2:value;"> </**htmltag**>

- Example:
- <**h2** style="color:red;margin-left:40px;">Inline CSS is applied on this heading.</**h2**>
- <**p**>This paragraph is not affected.</**p**>

```
Output:
Inline CSS is applied on this heading.
This paragraph is not affected.
```

Internal CSS

- The internal style sheet is used to add a unique style for a single document. It is defined in <head> section of the HTML page inside the <style> tag.

Example:

```
<style>
body {
    background-color: blue;
}
h1 {
    color: red;
    margin-left: 80px;
}
</style>
</head>
<body>
<h1>The internal style sheet is applied on this heading.</h1>
<p>This paragraph will not be affected.</p>
</body>
```

External CSS

- The external style sheet is generally used when you want to make changes on multiple pages. It is ideal for this condition because it facilitates you to change the look of the entire web site by changing just one file.

- It uses the <link> tag on every pages and the <link> tag should be put inside the head section.

Example:

```
1. <head>
2. <link rel="stylesheet" type="text/css" href="mystyle.css">
3. </head>
```

- The external style sheet may be written in any text editor but must be saved with a .css extension. This file should not contain HTML elements.

CSS Position

- The **CSS position property** is used *to set position for an element*. it is also used to place an element behind another and also useful for scripted animation effect.

- You can position an element using the top, bottom, left and right properties. These properties can be used only after position property is set first. A position element's computed position property is relative, absolute, fixed or sticky.

1. **CSS Static Positioning**
2. **CSS Fixed Positioning**
3. **CSS Relative Positioning**
4. **CSS Absolute Positioning**

1) **CSS Static Positioning**

This is a by default position for HTML elements. It always positions an element according to the normal flow of the page. It is not affected by the top, bottom, left and right properties.

2. CSS Fixed Positioning

- The fixed positioning property helps to put the text fixed on the browser. This fixed test is positioned relative to the browser window, and doesn't move even you scroll the window.

```
<html>
<head>
<style>
p.pos_fixed {
    position: fixed;
    top: 50px;
    right: 5px;
    color: blue;
}
</style>
</head>
<body>
```

3) CSS Relative Positioning

The relative positioning property is used to set the element relative to its normal position.

```
<style>
h2.left {
   position: relative;
   left: -30px;
}
h2.right {
   position: relative;
   left: 30px;
}
</style>
</head>
<body>
<h2>This is a heading with no position</h2>
<h2 class="left">This heading is positioned left to its normal position</h2>
<h2 class="right">This heading is positioned right  to its normal position</h2>
</body>
```

4) CSS Absolute Positioning

The absolute positioning is used to position an element relative to the first parent element that has a position other than static. If no such element is found, the containing block is HTML.

- With the absolute positioning, you can place an element anywhere on a page.

```
<style>
h2 {
    position: absolute;
    left: 150px;
    top: 250px;
}
</style>
</head>
<body>
<h2>This heading has an absolute position</h2>
<p> The heading below is placed 150px from the left and 250px from the top of the page.</p>
</body>
```

CSS Background

- CSS background property is used to define the background effects on element. There are 5 CSS background properties that affects the HTML elements:

1. background-color
2. background-image
3. background-repeat
4. background-attachment
5. background-position

1) CSS background-color

- The background-color property is used to specify the background color of the element.

2) CSS background-image

- The background-image property is used to set an image as a background of an element. By default the image covers the entire element.

```
<style>
body {
background-image: url("paper1.gif");
margin-left:100px;
}
</style>
```

3) CSS background-repeat

- By default, the background-image property repeats the background image horizontally and vertically. Some images are repeated only horizontally or vertically.
- The background looks better if the image repeated horizontally only

```
<style>
body {
    background-image: url("gradient_bg.png");
    background-repeat: repeat-x;
}
</style>
```

Document Object Model

- The **document object** represents the whole html document.
- When html document is loaded in the browser, it becomes a document object. It is the **root element** that represents the html document. It has properties and methods. By the help of document object, we can add dynamic content to our web page.

Window.document

- Properties of document object
- Let's see the properties of document object that can be accessed and modified by the document object.
- Methods of document object We can access and change the contents of document by its methods.

The important methods of document object are as follows:

Method	Description
write("string")	writes the given string on the doucment.
writeln("string")	writes the given string on the doucment with newline character at the end.
getElementById()	returns the element having the given id value.
getElementsByName()	returns all the elements having the given name value.
getElementsByTagName()	returns all the elements having the given tag name.
getElementsByClassName()	returns all the elements having the given class name.

```
<script type="text/javascript">
function printvalue(){
var name=document.form1.name.value;
alert("Welcome: "+name);
}
</script>

<form name="form1">
Enter Name:<input type="text" name="name"/>
<input type="button" onclick="printvalue()" value="print name"/>
</form>
```

JavaScript Events

- The change in the state of an object is known as an **Event**. In html, there are various events which represents that some activity is performed by the user or by the browser. When JavaScript code is included in HTML, js react over these events and allow the execution. This process of reacting over the events is called **Event Handling**. Thus, js handles the HTML events via **Event Handlers**.
- **For example**, when a user clicks over the browser, add js code, which will execute the task to be performed on the event.
- Some of the HTML events and their event handlers are:

Event Performed	Event Handler	Description
click	onclick	When mouse click on an element
mouseover	onmouseover	When the cursor of the mouse comes over the element
mouseout	onmouseout	When the cursor of the mouse leaves an element
mousedown	onmousedown	When the mouse button is pressed over the element
mouseup	onmouseup	When the mouse button is released over the element
mousemove	onmousemove	When the mouse movement takes place.

Form events

Event Performed	Event Handler	Description
focus	onfocus	When the user focuses on an element
submit	onsubmit	When the user submits the form
blur	onblur	When the focus is away from a form element
change	onchange	When the user modifies or changes the value of a form element

Window Events

Event Performed	Event Handler	Description
load	onload	When the browser finishes the loading of the page
unload	onunload	When the visitor leaves the current webpage, the browser unloads it
resize	onresize	When the visitor resizes the window of the browser

```
<body>
<script language="Javascript" type="text/Javascript">
    <!--
    function clickevent()
    {
        document.write("This is Java Script");
    }
    //-->
</script>
<form>
<input type="button" onclick="clickevent()" value="Who's this?"/>
</form>
</body>
```

Focus Event

```
<html>
<head> Javascript Events</head>
<body>
<h2> Enter something here</h2>
<input type="text" id="input1" onfocus="focusevent()"/>
<script>
<!--
    function focusevent()
    {
        document.getElementById("input1").style.background=" aqua";
    }
//-->
</script>
</body>
</html>
```

- Event Bubbling JavaScript
- In JavaScript, propagation of events is done, which is known as 'Event Flow'. Event Flow is the sequence or order in which the particular web page receives the event.
- **Event Bubbling**
- While developing a webpage or a website via JavaScript, the concept of event bubbling is used where the event handlers are invoked when one element is nested on to the other element and are part of the same event. This technique or method is known as Event Bubbling. Thus, while performing event flow for a web page, event bubbling is used.

```
<body>
  <div id="p1">
    <button id="c1">I am child button</button>
  </div>
      <script>
    var parent = document.querySelector('#p1');
      parent.addEventListener('click', function(){
        console.log("Parent is invoked");
      });
     var child = document.querySelector('#c1');
      child.addEventListener('click', function(){
        console.log("Child is invoked");
      });
  </script>
</body>
```

- **Explanation of Code:**

1. The above code is a HTML and JavaScript based code.
2. We have used a div tag having div id = p1 and within div we have nested a button having button id = c1.
3. Now, within the JavaScript section, we have assigned the html elements (p1 and c1) using the querySelector () function to the variable parent and child.
4. After that, we have created and included an event which is the click event to both div element and child button. Also created two functions that will help us to know the sequence order of the execution of the parent and child. It means if the child event is invoked first, "child is invoked" will be printed otherwise "parent is invoked" will get printed.
5. Thus, when the button is clicked, it will first print "child is invoked" which means that the function within the child event handler executes first. Then it moves to the invocation of the div parent function.

- **CSS filter**
- CSS filters are used to set visual effects to text, images, and other aspects of a webpage. The CSS **filter** property allows us to access the effects such as color or blur, shifting on the rendering of an element before the element gets displayed.
- The syntax of CSS filter property is given below.
- filter:

1. invert()
2. drop-shadow()
3. brightness()
4. saturate()
5. blur()

- brightness()
- As its name implies, it is used to set the brightness of an element. If the brightness is 0%, then it represents completely black, whereas 100% brightness represents the original one. It can also accept values above 100% that provide brighter results.
- We can understand it by using the following illustration.
- **Example**

blur()

```
<!DOCTYPE html>
<html>
<head>
   <title>CSS filter property</title>
   <style>
   body{
   text-align:center;
   }
      #img1 {
         filter: brightness(130%);
      }
   </style>
</head>
<body>
  <img src =  "tiger.png" > <h2>Original
 Image    </h2>
   <img src =  "tiger.png" id = "img1"> <
h2>brightness(130%)</h2>
</body>
</html>
```

```
   <style>
   body{
   text-align:center;
   }
      #img1 {
        filter: blur(2px);
      }
   </style>
</head>
<body>
 <img src =  "tiger.png" > <h2>Origin
al Image    </h2>
  <img src =  "tiger.png" id = "img1">
<h2>blur(2px)</h2>
</body>
```