Chapter 2 – JAVA SCRIPT, BOOTSTRAP, JQUERY

Java Script: Introduction to Javascript, variables, primitive data types, control flow statements, Built-in objects, arrays, functions, event handling, DHTML - Object model.
Bootstrap: Introduction to Bootstrap, Structure of the page, Typography, Forms.
JQuery: Working with JQuery.

INTRODUCTION TO JAVA SCRIPT:

- Web pages are two types
    i. *Static web page:* there is no specific interaction with the client
    ii. *Dynamic web page:* web page which is having interactions with client and as well as validations can be added.
- Script means small piece of Code.
- Scripting Language is a high-level programming language, whose programs are interpreted by another program at run time rather than compiled by the computer processor.
- BY using JavaScript we can create interactive web pages. It is designed to add interactivity to HTML pages.
- Previously JavaScript was known as LiveScript, but later it was changed to JavaScript. As Java was very popular at that time and introducing a new language with the similarity in names would be beneficial they thought.
- Scripting languages are of 2 types.
    - *client-side scripting languages*
    - *servers-side scripting languages*
- In general Client-side scripting is used for performing simple validations at client-side;

Server-side scripting is used for database verifications.

- **Examples:**

  Client-side scripting languages: VBScript, JavaScript and Jscript.

  Server-side scripting languages: ASP, JSP, Servlets and PHP etc.
- Simple HTML code is called static web page, if you add script to HTML page it is called dynamic page.
- Netscape Navigator developed JavaScript and Microsoft□s version of JavaScript is Jscript.

<u>Features of JavaScript:</u>

- JavaScript is a lightweight, interpreted programming language means that scripts execute without preliminary compilation.
- It is an Object-based Scripting Language.
- Java script is case sensitive language
- Complementary to and integrated with Java.
- Open and cross-platform.

<u>Advantages of JavaScript:</u>

1. **Less server interaction:**

   You can validate the user input before sending the page off to the server. This saves server traffic, which means less load on server.

2. **Immediate feedback to the visitors or end-users:**

   If you submit a form if there is any error in form filling immediately visitor get the feedback, because validation performed at client side.

3. Can put dynamic text into an HTML page
4. Used to Validate form input data
5. Java script code can react to user events

6. Can be used to detect the visitor□s browser

Limitations of JavaScript:

- Client-side JavaScript does not allow the reading or writing of files. This has been kept for security reason.
- JavaScript cannot be used for networking applications because there is no such support available.
- JavaScript doesn't have any multithreading or multiprocessor capabilities.

JAVA Vs JAVASCRIPT:

JAVA	JAVASCRIPT
1. Object Oriented Programming Language	2.1 Object based Scripting Language
2. Platform Independent	2.2 Browser Dependant
3. It is both compiled and interpreted	2.3 It is interpreted at runtime
4. It is used to create server side applications and standalone programming	2.4 It is used to make the web pages more interactive
5. Java is a strongly typed language	2.5 JavaScript is not strongly typed(Loosely Typed)
6. Developed by sun Microsystems	2.6 Developed by Netscape
7. Java Programs can be standalone	2.7 JavaScript must be placed inside an HTML document to function

Embedding JavaScript in an HTML Page:

Embed a JavaScript in an HTML document by using <script> and </script> html tags.

Syntax:

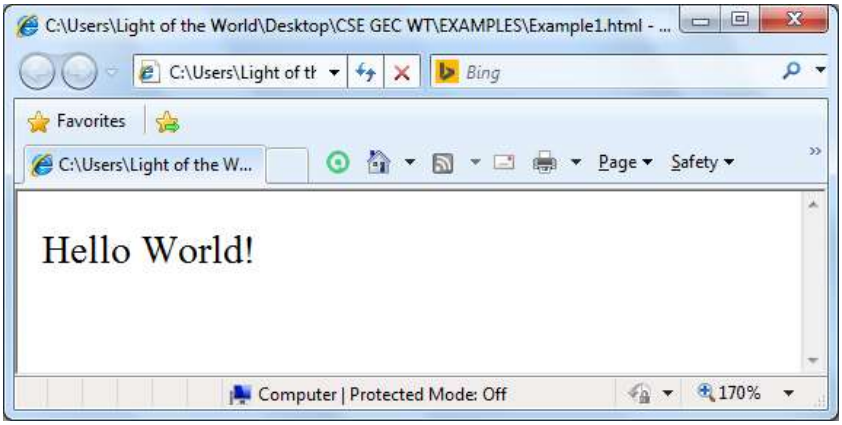
```
<script ...>
        JavaScript code
</script>
```

<script > tag has the following attributes.

Type	Refers to the MIME (Multipurpose Internet Mail Extensions) type of the script.
Language	This attribute specifies what scripting language you are using. Typically, its value will be javascript. Although recent versions of HTML (and XHTML, its successor) have phased out the use of this attribute.

Example:

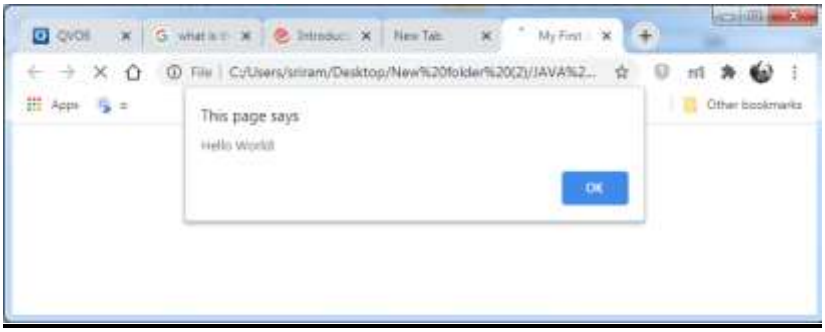
```
<html>
<body>
<script language="javascript" type="text/javascript">
        document.write ("Hello World!")
</script>
</body>
</html>
```



Alert Message Example:

```
<html>
<head>
<title>My First JavaScript code!!!</title>
<script type="text/javascript">
alert("Hello World!");
</script>
</head&gt;
<body>
</body>
</html>
```

Output:



Alert(□Hello World□);

Var d= Confirm(□Do you want to continue?□);

Var i=prompt(□Enter your name:□,□SRGEC□);

Comments in JavaScript:

JavaScript supports both C-style and C++-style comments.

Thus:

- Any text between a // and the end of a line is treated as a comment and is ignored by JavaScript.
- Any text between the characters /* and */ is treated as a comment. This may span multiple lines.

VARIABLES:

- Like any programming language JavaScript has variables.
- Stores data items used in the script.
- Strict rules governing how you name your variables (Much like other languages):

Naming Conventions for Variables:

- Variable names **must** begin with a alphabet([a-z]/[A-Z]) or underscore;
- You can□t use spaces in names
- Names are **case sensitive** so the variables fred, FRED and frEd all refer to **different** variables,
- It is not a good idea to name variables with similar names
- You **can□t use a reserved word** as a variable name, **e.g.** var.

Creating Variables

- Before you use a variable in a JavaScript program, you must declare it. Variables are declared with the **var** keyword as follows.

```
<script type="text/javascript">
        var name;
        var rollno;
</script>
```

- Storing a value in a variable is called **variable initialization**. You can do variable initialization at the time of variable creation or at a later point in time when you need that variable.

```
<script type="text/javascript">
        var name = Aziz;
        var rollno=501;
</script>
```

Scope of Variables in JavaScript:

The scope of a variable is the region of your program in which it is defined and is accessible. JavaScript variables have only two scopes.

- **Global Variables:** A global variable has global scope which means it can be defined and used anywhere in your JavaScript code.

- **Local Variables:** A local variable will be visible only within a function where it is defined. Function parameters are always local to that function.

Automatically Global:

- If you assign a value to a variable that has not been declared, it will automatically become a **GLOBAL** variable.
- This code example will declare a global variable **price**, even if the value is assigned inside a function.

Example:

```
myFunction();
// code here can use price
function myFunction()
{
    price = 250; //has Global scope
}
```

Example:

```
<script language="javascript" type="text/javascript">
var  collegename="GEC college"; //global scope
function function1()
{
var studentname="Anand";//local scope
document.write("<center>"+studentname+"</center><br
>");
document.write("<center>"+collegename+"</center><br>
");//global scope
}
function function2()
{
var branchname="Information Technology";//local
scope
```

```
        document.write("<center>"+branchname+"</center><br
        >");
        document.write("<center>"+collegename+"</center><br>
        ");//global scope
        document.write("<center>"+studentname+"</center>");/
        /not displayed because of local scope
        }
        function1();
        function2();
</script>
```

DATA TYPES:

- JavaScript has only four types of data
  - Numeric
  - String
  - Boolean
  - Null
- **Numeric :**
  - Integers such as 108 or 1120 or 2016
  - Floating point values like 23.42, -56.01 and 2E45.
  - No need to differentiate between.
  - In fact variables can change type within program.
- **String:**
  - A String is a Collection of character.
  - All of the following are strings:

    "Computer", "Digital" , "12345.432".
  - Put quotes around the value to a assign a variable:

    name = "Uttam K.Roy";
- **Boolean:**
  - Variables can hold the values true and false.
  - Used a lot in conditional tests (later).

- **Null:**
  - Used when you don t yet know something.
  - A null value means one that has not yet been decided.
  - It does not mean **nil** or **zero** and **should NOT** be used in that way.

FUNCTIONS:

- A function is a group of reusable code which can be called anywhere in your program.
- This eliminates the need of writing the same code again and again.
- It helps programmers in writing modular codes. Functions allow a programmer to divide a big program into a number of small and manageable functions.
- Like any other advanced programming language, JavaScript also supports all the features necessary to write modular code using functions.
- We were using these functions again and again, but they had been written in core JavaScript only once.
- JavaScript allows us to write our own functions as well.
- **Function Definition**
  - Before we use a function, we need to define it.
  - The most common way to define a function in JavaScript is
  - By using keyword **function**, followed by a unique function name, a list of parameters (that might be empty), and a statement block surrounded by curly braces.

Syntax:

```
<script type="text/javascript">
function functionname(parameter-list)
 {
        statements
 }
</script>
```

Example:

```
<html>
<head>
<title>My First JavaScript code!!!</title>
<script type="text/javascript">
                function sayHello()
                {
                  document.write("Hello Anand How
                  are you...?");
                }
                sayHello();//calling function
</script>
</head>;
<body>
</body>
</html>
```

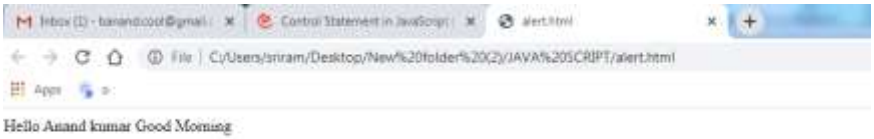
Calling a Function:

To invoke a function somewhere later in the script, you would simply need to write the name of that function as shown in the following code.

```
<html>
<head>
<title>Calling a function</title>
        <style type='text/css'>
        {
        text-align:center;
        }
        </style>
<script type="text/javascript">
                function sayHello()
                {
                var name=form.name.value;
                document.write("Hello "+name+" Good
                Morning");
                }

</script>
</head>
<body>
                    <p>Please enter you name and click
                    the button to get wishes
                    </p></br>
                <form name='form'>
                <input type='text' name='name'
                placeholder='Enter Name'><br><br>
                <input type='button' value='click here'
```

```
                    onclick='sayHello();'>
                    </form>
</body>
</html>
```

Output:



OPERATORS:

JavaScript supports the following types of operators.

- Arithmetic Operators
- Assignment Operators
- Comparison Operators
- Logical (or Relational) Operators
- Conditional (or ternary) Operators

Arithmetic Operators:

- JavaScript supports the following arithmetic operators:
- Assume variable A holds 10 and variable B holds 20, then:

Operator	Descrition	Example
+	Adds two numbers or joins two strings	20+10 returns 30
-	Subtracts two numbers or represents a negative number	20-10 returns 10
*	Multiplies two numbers	20*10 returns 200
/	Divides two numbers evenly and returns the quotient	20/10 returns 2
%	Divides two numbers and returns the remainder	20%10 returns 0
++	Increments the value of a number by 1 • Prefix (Pre-increment) • Suffix (Post-increment)	m = 20 n=++m assigns 21 to n
		m = 20 n=m++ assigns 20 to n
- -	Decrements the value of a number by 1 • Prefix (Pre-Decrement) • Suffix (Post-Decrement)	m = 20 n=--m assigns 19 to n
		m = 20 n=m++ assigns 20 to n

Assignment Operators:

Operator	Description	Example
=	Assigns the value on the right hand side to the variable on left hand side	m=20
+=	Adds the right hand side operand to the left hand side operand and assigns the result to the left hand side operand.	m = 20 n = 10 m+=n assigns 30 to m
-=	Subtracts the right hand side operand from the left hand side operand and assigns the result to the left hand side operand.	m = 20 n = 5 m-=n assigns 15 to m
*=	Multiplies the right hand side operand and the left hand side operand and assigns the result to the left hand side operand.	m = 20 n = 10 m*=n assigns 200 to m
/=	Devides the left hand side operand by the right hand side operand and assigns the quotient to the left hand side operand.	m = 20 n = 10 m/=n assigns 2 to m
%=	Divides the left hand side operand by the right hand side operand and assigns the remainder to the left hand side operand.	m = 20 n = 10 m%=n assigns 0 to m

Comparison Operators:

Operator	Description	Example
==	Returns true if both the operands are equal otherwise returns false	20==10 returns false
!=	Returns true if both the operands are not equal otherwise returns false	20 !=10 returns true
>	Returns true if left hand side operand Is greater than the right hand side operand. otherwise returns false	20 > 10 returns true
>=	Returns true if left hand side operand is greater than or equal to the right hand side operand. otherwise returns false	20 >= 10 returns true
<	Returns true if left hand side operand Is less than the right hand side operand. otherwise returns false	20 < 10 returns false
<=	Returns true if left hand side operand is less than or equal to the right hand side operand. otherwise returns false	20 <= 10 returns false

Logical (or Relational) Operators:

Operator	Descrition	Example
&&	Returns true only if both the operands are true, otherwise returns false	True && True returns True
||	Returns true only if either of the operands are true. It returns false when both the operands are false	True || False returns True
!	Negates the operand	!true returns false

Conditional (or ternary) Operators:

Operator	Description	Example
?:	Returns the second operand if the first operand is true, otherwise returns the third operand.	Result=(20 > 10)? 20 : 10 Here, 20 is assigned to Result

---

CONTROL FLOW STATEMENTS: These statements allow you to control the flow of your program□s execution based upon conditions known only during run time.

In JavaScript we have the following conditional statements:

- Use **if** to specify a block of code to be executed, if a specified condition is true

- Use **else** to specify a block of code to be executed, if the same condition is false
- Use **else if** to specify a new condition to test, if the first condition is false
- Use **switch** to specify many alternative blocks of code to be executed

The if Statement

Syntax

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

The else Statement

Use the **else** statement to specify a block of code to be executed if the condition is false.

```
if (condition)
 {
    block of code to be executed if the condition is true


 }
  else
  {
    block of code to be executed if the condition is false
  }
```

Example:

```
<HTML>
 <HEAD>
                <script>
                 function check()
                 {
                 var age=form.age.value;
                 if(age>=18)
                 {
                 alert("You are eligible for vote");
                 }
                 else
                 {
                 alert("You are not eligible for vote");
                 }
                 }
                </script>
 </HEAD>

 <BODY>
                <form name='form'>
                <p>Enter your age and check whether you are
                eligible for vote or not?</p><br>
                <input type='text' name='age'><br><br>
                <input type='button' value='check eligibility'
                onclick='check();'>
                </form>
 </BODY>
</HTML>
```

Output:

The else if Statement

Use the **else if** statement to specify a new condition if the first condition is false.

Syntax:

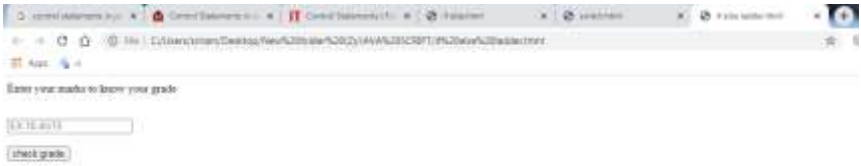
```
if (condition1)
{
    block of code to be executed if condition1 is true
}
else if (condition2)
{
    block of code to be executed if the condition1 is false and
condition2 is true
}
else
 {
    block of code to be executed if the condition1 is false and
condition2 is false
}
```

Example:

```
<HTML>
 <HEAD>
                <script>
                 function check()
                 {
                 var percentage=form.percentage.value;
                 if(percentage>=90&&percentage<=100)
                 {
                 alert("Your grade is A+");
                 }
                 else if(percentage>=75&&percentage<90)
                 {
                 alert("Your grade is A");
                 }
                 else if(percentage>=60&&percentage<75)
                 {
                 alert("Your grade is B");
                 }
                 else if(percentage>=40&&percentage<60)
                 {
                 alert("Your grade is C");
                 }
                 else if(percentage>100)
                 {
                 alert("Wrong details.....");
                 }
                 else
                 {
                 alert("You are failed");
```

```
                }
                }
            </script>
</HEAD>

<BODY>
            <form name='form'>
            <p>Enter your marks to know your
            grade</p><br>
            <input type='text' name='percentage'
            placeholder='EX:70.45/70'><br><br>
            <input type='button' value='check grade'
            onclick='check();'>
            </form>
</BODY>
</HTML>
```

Output:



<u>Switch Statement:</u>

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

Syntax:

```
switch(expression) {
   case 1:
      code block
      break;
   case 2:
      code block
      break;
        .
        .
    case n:
   code block
      break;
   default:
      default code block
}
```

This is how it works:

- The switch expression is evaluated once.
- The value of the expression is compared with the values of each case.
- If there is a match, the associated block of code is executed.

Example:

```
<HTML>
 <HEAD>
            <script>
            function check()
            {
            var category=form.category.value;
            switch(category)
```

```
{
        case "SC":
        alert("50 vanacies");
        break;
        case "OC":
        alert("5 vacanices");
        break;
        case "BC":
        alert("30 vanacies");
        break;
        case "ST":
        alert("45 vanacies");
        break;
        case "OBC":
        alert("20 vanacies");
        break;
        default:
        alert("please enter valid category");
        break;
}
}
</script>
</HEAD>
<BODY>
<form name='form'>
<p>Please enter your category to check no of
vacanices</p><br>
<input type='text' name='category'
placeholder='EX:OC/BC/OBC/SC/ST'><br><br>
```

```
                <input type='button' value='Check Vacancies'
                onclick='check();'>
                </form>
 </BODY>
</HTML>
```

Output:

The While Loop

Syntax:

```
while (condition)
 {
    code block to be executed
}
```

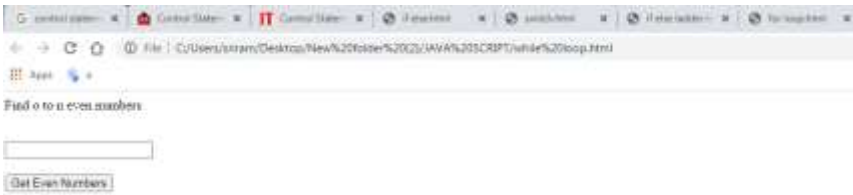
Example:

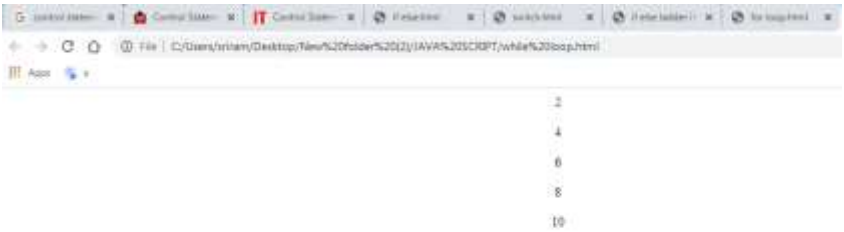
Write a JavaScript code to print 0 to n even numbers using while loop.

```
   <HTML>
 <HEAD>
                <script>
                  function check()
                  {
```

```
                var number=form.number.value;
                var i=1;
                while(i<=number)
                {
                if(i%2==0)
                document.write("<center>"+i+"</center><br>");
                i++;
                }
                }
               </script>
 </HEAD>

 <BODY>
               <form name='form'>
               <p>Find o to n even numbers</p><br>
               <input type='text' name='number'><br><br>
               <input type='button' value='Get Even Numbers'
onclick='check();'>
               </form>
 </BODY>
</HTML>
```

Output:





The Do/While Loop

The do/while loop is a variant of the while loop. This loop will execute the code block once, before checking if the condition is true, then it will repeat the loop as long as the condition is true.

Syntax

```
do
 {
    code block to be executed
}while (condition);
```

The for Loop: The for loop has the following syntax:

```
for (initialization; condition; iteration)
 {
                code block to be executed
 }
```

Statement 1 is executed before the loop (the code block) starts.

Statement 2 defines the condition for running the loop (the code block).

Statement 3 is executed each time after the loop (the code block) has been executed.

Write a JavaScript code to print 1 to 10 even numbers using

ForDemo - Notepad

File Edit Format View Help

```
<html>
 <body>
 <h2> 1 to 10 Even Numbers </h2>
 <script language="javascript">

  function even()
 {
        for(i=1;i<=10;i++)
        {
          if(i%2==0)
          document.write(i+"<br>");
        }
 }

even();

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

C:\Users\ONLINE\Documents\ForDemo.html - Wind...

C:\Users\ONLINE\D Bing

Favorites C:\Users\ONLINE\Documents\ForDemo.html

1 to 10 Even Numbers

2
4
6
8
10

Computer | Protected Mode: Off 125%

Example 2:

```
<HTML>
 <HEAD>
                <script>
                 function check()
                 {
                 var number=form.number.value;
                 var fact=1;
                 for(var i=1;i<=number;i++)
                 {
                 fact=fact*i;
                 }
                 alert("factorial of "+number+"is "+fact);
                 }
                </script>
 </HEAD>

 <BODY>
                <form name='form'>
                <p>Enter a number to know the
factorial</p><br>
                <input type='text' name='number'><br><br>
                <input type='button' value='Get Factorial'
onclick='check();'>
                </form>
 </BODY>
</HTML>
```

Output:

OBJECTS IN JAVA SCRIPT: (BUILT-IN OBJECTS)

- An Object is a thing.
- There are pre defined objects and user defined objects in Javascript.
- Each object can have properties and methods:
  - A property tells you something about an object.
  - A method performs an action
- The following are some of the Pre defined objects/Built-in Objects.
  - Document
  - Window
  - Browser/Navigator
  - Form
  - String
  - Math
  - Array
  - Date

HTML DOM

The way document content is accessed and modified is called the Document Object Model, or DOM.

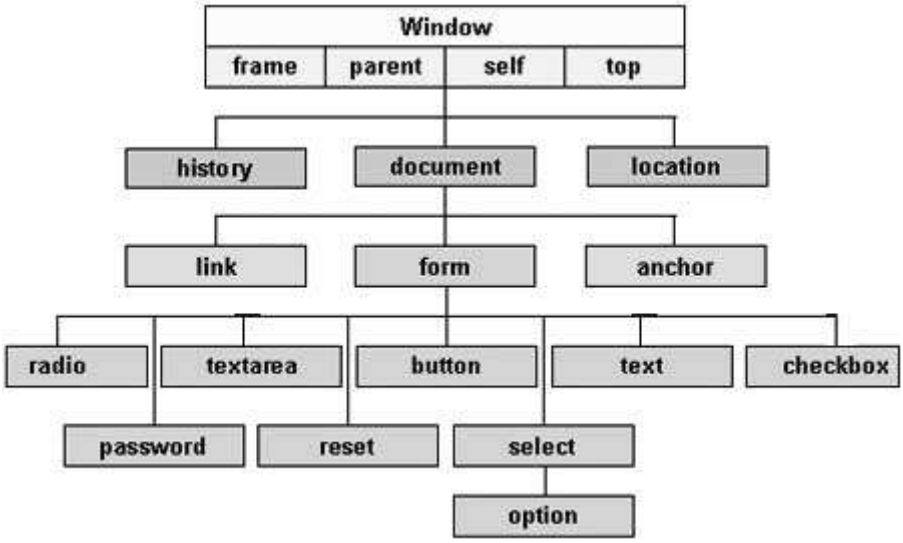
In the HTML DOM (Document Object Model), everything is a **node**:

- The document itself is a document node
- All HTML elements are element nodes
- All HTML attributes are attribute nodes
- Text inside HTML elements are text nodes
- Comments are comment nodes

The Objects are organized in a hierarchy. This hierarchical structure applies to the organization of objects in a Web document.

- **Window object** – Top of the hierarchy. It is the outmost element of the object hierarchy.
- **Document object** – Each HTML document that gets loaded into a window becomes a document object. The document contains the contents of the page.
- **Form object** – Everything enclosed in the <form>...</form> tags sets the form object.
- **Form control elements** – The form object contains all the elements defined for that object such as text fields, buttons, radio buttons, and checkboxes.

Here is a simple hierarchy of a few important objects –



THE DOCUMENT OBJECT

- When an HTML document is loaded into a web browser, it becomes a **document object**.
- The document object is the root node of the HTML document and the "owner" of all other nodes: (element nodes, text nodes, attribute nodes, and comment nodes).
- The document object provides properties and methods to access all node objects, from within JavaScript.
- **Tip:** The document is a part of the Window object and can be accessed as window.document.

Properties

alinkColor -	The color of active links
bgColor -	Sets the background color of the web page. It is set in the <body> tag. The following code sets the background color to white.

Title -	The name of the current document as described between the header TITLE tags.
URL -	The location of the current document.
vlinkColor -	The color of visited links as specified in the <body> tag fgColor -

Methods

getElementById(*id*) -	Find an element by element id
getElementsByTagName(*name*) -	Find elements by tag name
getElementsByClassName(*name*) -	Find elements by class name
write(text) -	Write into the HTML output stream
writeln(text) -	Same as write() but adds a new line at the end of the output

WINDOW OBJECT:

- The **window** object is supported by all browsers. It represents the browser's window.
- All global JavaScript objects, functions, and variables automatically become members of the window object.
- Global variables are properties of the window object.
- Global functions are methods of the window object.
- Even the document object (of the HTML DOM) is a property of the window object:

```
window.document.getElementById("header");
```

is the same as:

```
document.getElementById("header");
```

Properties

- defaultStatus - This is the default message that is loaded into the status bar when the window loads.
- opener The object that caused the window to open.
- status - The status bar is the bar on the lower left side of the browser and is used to display temporary messages
- length - The number of frames that the window contains.

Methods

- alert("message") - The string passed to the alert function is displayed in an alert dialog box.
- open("URLname","Windowname",["options"]) - A new window is opened with the name specified by the second parameter.
- close() - This function will close the current window or the named window.
- confirm("message") The string passed to the confirm function is displayed in the confirm dialog box.
- prompt("message","defaultmessage") - A prompt dialog box is displayed with the message passed as the prompt question or phrase.

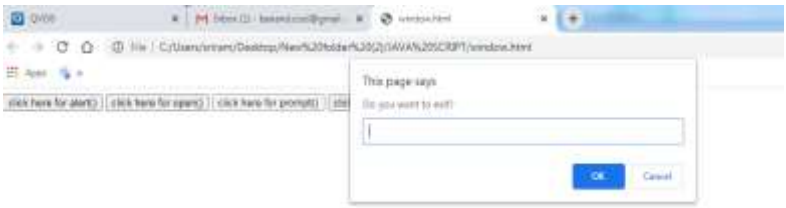
Example:

```
<HTML>
 <HEAD>
            <script>
              function funalert()
              {
              window.alert("Hello be alert....");
              }
              function funopen()
```

```
            {
            window.open("http://www.gmail.com");
            }
            function funprompt()
            {
            window.prompt("Do you want to exit?");
            }
            function funconfirm()
            {
            window.confirm("Do you want to exit?");
            }
            function funclose()
            {
            window.close();
            }
           </script>
</HEAD>
<BODY>
           <form>
           <input type='button' value='click here for alert()'
           onclick='funalert()'>
           <input type='button' value='click here for open()'
           onclick='funopen()'>
           <input type='button' value='click here for
           prompt()' onclick='funprompt()'>
           <input type='button' value='click here for
           cofirm()' onclick='funconfirm()'>
           <input type='button' value='click here for close()'
           onclick='funclose()'>
    </form>
```

```
 </BODY>
</HTML>
```

Output:



FORM OBJECT:

Properties

- action - The action attribute of the Top of Form element
- length - Gives the number of form controls in the form
- method- The method attribute of the Top of Form element
- name - The name attribute of the Top of Form element
- target - The target attribute of the Top of Form element

Methods

- reset()- Resets all form elements to their default values
- submit()- Submits the form

Properties of Form Elements

The following table lists the properties of form elements

- checked - Returns true when checked or false when not
- form - Returns a reference to the form in which it is part of
- length - Number of options in the <select> element.
- name - Accesses the name attribute of the element
- selectedIndex - Returns the index number of the currently selected item
- value - the value attribute of the element or content of a text input

STRING OBJECT:

String The string object allows you to deal with strings of text.

Properties

- length - The number of characters in the string.

Methods:

- charAt(index) - Returns a string containing the character at the specified location.
- indexOf(pattern) - Returns -1 if the value is not found and returns the index of the first character of the first string matching the pattern in the string.
- indexOf(pattern, index) - Returns -1 if the value is not found and returns the index of the first character of the first string matching the pattern in the string. Searching begins at the index value in the string.
- lastIndexOf(pattern) - Returns -1 if the value is not found and returns the index of the first character of the last string matching the pattern in the string.
- lastIndexOf(pattern, index) - Returns -1 if the value is not found and returns the index of the first character of the

last string matching the pattern in the string. Searching begins at the index value in the string.

- split(separator) - Splits a string into substrings based on the separator character.
- substr(start, length) - Returns the string starting at the "start" index of the string Continuing for the specified length of characters unless the end of the string is found first.
- substring(start, end) - Returns the string starting at the "start" index of the string and ending at "end" index location, less one.
- toLowerCase() - Returns a copy of the string with all characters in lower case.
- toUpperCase() - Returns a copy of the string with all characters in upper case.

Example:

```
<HTML>
 <HEAD>
                    <script>

          var txt = ABCDEFGHIJKLMNOPQRSTUVWXYZ";
         var sln = txt.length;
         document.writeln(sln);
          var str = "Please locate where 'locate' occurs!";
         var pos = str.indexOf("locate");
         document.write("<center>"+pos+"</center>");
         document.write("<center>"+str.toUpperCase()+
         "</center>");
```

```
document.write("<center>"+str.toLowerCase()+"</center
>");

document.write("<center>"+str.lastIndexOf("locate")+"</
center>");
 document.write("<center>"+str.split(" ")+"</center>");
document.write("<center>"+str.substr(5,10)+"</center>"
);
                    </script>
 </HEAD>

 <BODY>

 </BODY>
</HTML>
```

Output:

ARRAY OBJECT:

The Array object is used to store multiple values in a single variable.

Properties:

- length - Sets or returns the number of elements in an array

Methods:

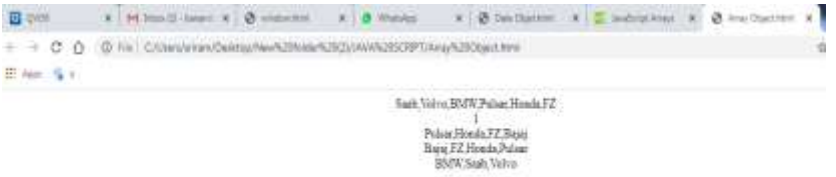
- concat() - Joins two or more arrays, and returns a copy of the joined arrays
- indexOf() - Search the array for an element and returns its position
- join() - Joins all elements of an array into a string
- lastIndexOf() - Search the array for an element, starting at the end, and returns its position
- pop() - Removes the last element of an array, and returns that element
- push()- Adds new elements to the end of an array, and returns the new length
- reverse() - Reverses the order of the elements in an array
- shift() - Removes the first element of an array, and returns that element
- slice() - Selects a part of an array, and returns the new array
- sort() - Sorts the elements of an array
- splice() - Adds/Removes elements from an array
- toString() - Converts an array to a string, and returns the result.

Example:

```
<HTML>
 <HEAD>
    <script>
          var cars = new Array("Saab", "Volvo", "BMW");
          var bikes = new Array("Pulsar", "Honda", "FZ");
          document.write("<center>"+cars.concat(bikes)+
           "</center>");
```

```
                document.write("<center>"+cars.indexOf("Volvo"
                )+"</center>");
              bikes.push("Bajaj");
              document.write("<center>"+bikes+"</center>");
               bikes.reverse();
                document.write("<center>"+bikes+"</center>");
               cars.sort();
                document.write("<center>"+cars+"</center>");
        </script>
</HEAD>
        <BODY>

        </BODY>
</HTML>
```

Output:



<u>BROWSER OBJECT/NAVIGATOR OBJECT:</u>

It is used to obtain information about client browser.

<u>Properties</u>

- appName- Returns Browser Name
- appVersion- Returns Browser Version
- appUserAgent- It Returns User Agent
- plugins- It will display Plugins.

- mimeTypes It will Return Mime type supported by browser

DATE OBJECT:

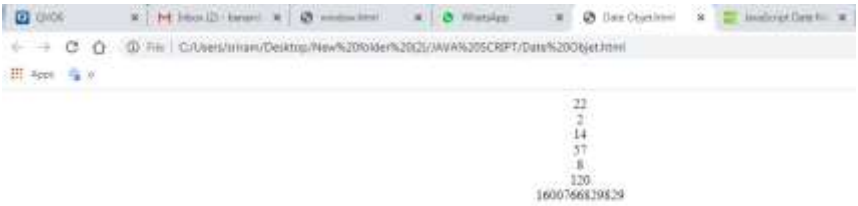
The Date object is used to work with dates and times

- getDate() - Get the day of the month. It is returned as a value between 1 and 31.
- getDay() - Get the day of the week as a value from 0 to 6
- getHours() - The value returned is 0 through 23.
- getMinutes() - The value returned is 0 through 59.
- getMonth() - Returns the month from the date object as a value from 0 through 11.
- getSeconds() - The value returned is 0 through 59.
- getTime() - The number of milliseconds since January 1, 1970.
- getYear() - Returns the numeric four digit value of the year.
- setDate(value) - Set the day of the month in the date object as a value from 1 to 31.
- setHours(value) - Set the hours in the date object with a value of 0 through 59.
- setMinutes(value) - Set the minutes in the date object with a value of 0 through 59.
- setMonth(value) - Set the month in the date object as a value of 0 through 11.
- setSeconds(value) - Set the seconds in the date object with a value of 0 through 59.
- setTime(value) - Sets time on the basis of number of milliseconds since January 1, 1970.
- setYear(value) - Set the year in the date instance as a 4 digit numeric value.

Example:

```
<HTML>
 <HEAD>
                    <script>
                      var d=new Date();

document.write("<center>"+d.getDate()+"</center>");

document.write("<center>"+d.getDay()+"</center>");

document.write("<center>"+d.getHours()+"</center>");

document.write("<center>"+d.getMinutes()+"</center>");

document.write("<center>"+d.getMonth()+"</center>");

document.write("<center>"+d.getYear()+"</center>");

document.write("<center>"+d.getTime()+"</center>");

                    </script>
 </HEAD>
 <BODY>

 </BODY>
</HTML>
```

Output:



EVENT HANDLING:

JavaScript is an Event Driven System

Event:

An Event is □any change that the user makes to the state of the browser□

There are 2 types of events that can be used to trigger script:

1. Window Events
2. User Events

1. Window Events, which occurs when
   - A page loads or unloads
   - Focus is being moved to or away from a window or frame
   - After a period of time has elapsed
2. User Events, which occur when the user interacts with elements in the page using mouse or a keyboard.

Event Handlers:

Event handlers are Javascript functions which you associate with an HTML element as part of its definition in the HTML source code.

Syntax: <element attributes eventAttribute=□handler□>

Attribute	Description
Onblur	The input focus is moved from the object
Onchange	The value of a field in a form has been changes by the user by entering or deleting data
Onclick	Invoked when the user clicked on the object.
Ondblclick	Invoked when the user clicked twice on the object.
Onfocus	Input focus is given to an element
Onkeydown	Invoked when a key was pressed over an element.
Onkeypress	Invoked when a key was pressed over an element then released.
Onkeyup	Invoked when a key was released over an element.
Onload	When a page is loaded by the browser
Onmousedown	The cursor moved over the object and mouse/pointing device was pressed down.
Onmousemove	The cursor moved while hovering over an object.
Onmouseout	The cursor moved off the object
onmouseover	The cursor moved over the object (i.e. user hovers the mouse over the object).
Onmouseup	The mouse/pointing device was released after being pressed down.
Onmove	A window is moved, maximized or restored either by the user or by the script
Onresize	A window is resized by the user or by the script

onmousewheel	Invoked when the mouse wheel is being rotated.
Onreset	When a form is reset
Onselect	Invoked when some or all of the contents of an object is selected. For example, the user selected some text within a text field.
Onsubmit	User submitted a form.
Onunload	User leaves the Page

Examples:

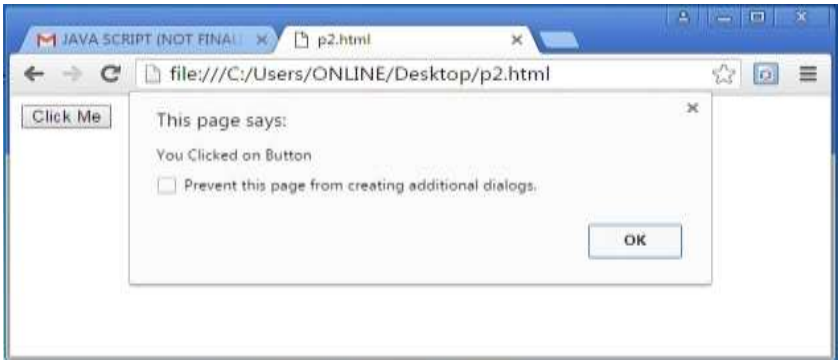
1.
```
<html>
<head>
<script language="javascript">
    function fun()
    {
            alert("Page is Loaded");
    }
</script>
</head>
<body onload="fun()">
</body>
</html>
```

Output:

2.
```
<html>
<head>
```

```
<script language="javascript">
    function fun()
    {
            alert("You Clicked on Button");
    }
</script>
</head>
<body>
    <input type="button" value="Click Me" onClick="fun()">
</body>
</html>
```

Output:




```
<HTML>
 <HEAD>
            <script>
             function check()
             {
             var number=form.number.value;
             var i=1;
             while(i<=number)
             {
             if(i%2==0)
```

```
                document.write("<center>"+i+"</center><br>");
                i++;
                }
                }
              </script>
    </HEAD>

    <BODY>
              <form name='form' onSubmit='check();'>
              <p>Find 1 to n even numbers</p><br>
              <input type='text' name='number'><br><br>
              <input type='submit' value='Get Even Numbers'>
              </form>
    </BODY>
    </HTML>
```

Output:

Math Object:

The **math** object provides you properties and methods for mathematical constants and functions. Unlike other global objects, **Math** is not a constructor. All the properties and methods of **Math** are static and can be called by using Math as an object without creating it.

Thus, you refer to the constant **pi** as **Math.PI** and you call the *sine* function as **Math.sin(x)**, where x is the method's argument.

Math Properties (Constants)

JavaScript provides 8 mathematical constants that can be accessed with the Math object:

Example

```
Math.E       // returns Euler's number
Math.PI      // returns PI
Math.SQRT2   // returns the square root of 2
Math.SQRT1_2 // returns the square root of 1/2
Math.LN2     // returns the natural logarithm of 2
Math.LN10    // returns the natural logarithm of 10
Math.LOG2E   // returns base 2 logarithm of E
Math.LOG10E  // returns base 10 logarithm of E
```

Math Object Methods

Method	**Description**
abs(x)	Returns the absolute value of x
acos(x)	Returns the arccosine of x, in radians
acosh(x)	Returns the hyperbolic arccosine of x
asin(x)	Returns the arcsine of x, in radians
asinh(x)	Returns the hyperbolic arcsine of x
atan(x)	Returns the arctangent of x as a numeric value between -PI/2 and PI/2 radians
atan2(y, x)	Returns the arctangent of the quotient of its arguments
atanh(x)	Returns the hyperbolic arctangent of x
cbrt(x)	Returns the cubic root of x
ceil(x)	Returns x, rounded upwards to the nearest integer

cos(x)	Returns the cosine of x (x is in radians)
cosh(x)	Returns the hyperbolic cosine of x
exp(x)	Returns the value of E^x
floor(x)	Returns x, rounded downwards to the nearest integer
log(x)	Returns the natural logarithm (base E) of x
max(x, y, z, ..., n)	Returns the number with the highest value
min(x, y, z, ..., n)	Returns the number with the lowest value
pow(x, y)	Returns the value of x to the power of y
random()	Returns a random number between 0 and 1
round(x)	Rounds x to the nearest integer
sin(x)	Returns the sine of x (x is in radians)
sinh(x)	Returns the hyperbolic sine of x
sqrt(x)	Returns the square root of x
tan(x)	Returns the tangent of an angle
tanh(x)	Returns the hyperbolic tangent of a number
trunc(x)	Returns the integer part of a number (x)

Example:

```
<HTML>
 <HEAD>
    <script>
  document.write("<center>"+Math.PI+"</center><br>");

document.write("<center>"+Math.ceil(0.991)+"</center><br>");

document.write("<center>"+Math.floor(0.991)+"</center><br>")
;
```

```
document.write("<center>"+Math.min(12,3,42,55,75,1)+"</cen
ter><br>");

document.write("<center>"+Math.max(12,3,42,55,75,1)+"</ce
nter><br>");

document.write("<center>"+Math.pow(5,3)+"</center><br>");

document.write("<center>"+Math.sqrt(25)+"</center><br>");

document.write("<center>"+Math.random()+"</center><br>");
    </script>
 </HEAD>
 <BODY>
 </BODY>
</HTML>
```

Output:

DHTML WITH JAVASCRIPT:

- It refers to the technique of making web pages dynamic by client-side scripting to manipulate the document content and presentation
- Web pages can be made more lively, dynamic or interactive by DHTML techniques.
- DHTML is **not** a markup language or a software tool.
- DHTML involves the following aspects.
  - HTML - For designing static web pages
  - JAVASCRIPT - For browser scripting
  - CSS (Cascading Style Sheets) - For style and presentation control
  - DOM(Document Object Model) - An API for scripts to access and manipulate the web page as a document.

So, **DHTML = HTML + CSS + JAVASCRIPT + DOM**

HTML Vs DHTML

HTML	DHTML
1. It is used to create static web pages.	1. Used to create dynamic web pages.
2. Consists of simple HTML tags.	2. Made up of HTML tags+CSS+javascript+DOM
3. It is a markup language.	3. It is a technique to make web pages dynamic through client-side programming.
4. Do not allow to alter the text and graphics on the web page unless web page gets changed.	4. DHTML allows you to alter the text and graphics of the web page without changing the entire web page.

5. Creation of HTML web pages is simple.	5. Creation of DHTML web pages is complex.
6. Web pages are less interactive.	6. Web pages are more interactive.
7. HTML sites will be slow upon client-side technologies.	7. DHTML sites will be fast enough upon client-side technologies.

Form Validation:

Form validation normally used to occur at the server, after the client had entered all the necessary data and then pressed the Submit button. If the data entered by a client was incorrect or was simply missing, the server would have to send all the data back to the client and request that the form be resubmitted with correct information. This was really a lengthy process which used to put a lot of burden on the server.

JavaScript provides a way to validate form's data on the client's computer before sending it to the web server. Form validation generally performs two functions.

- **Basic Validation** – First of all, the form must be checked to make sure all the mandatory fields are filled in. It would require just a loop through each field in the form and check for data.
- **Data Format Validation** – Secondly, the data that is entered must be checked for correct form and value. Your code must include appropriate logic to test correctness of data.

Example:

```
<html>
<head>
<script language="javascript" type="text/javascript">
function validate()
{
if(form.name.value==0)
{
alert("Username should not be empty");
form.name.focus();
return false;
}
if(form.password.value==0)
{
alert("password should not be empty");
form.password.focus();
return false;
}
if(form.password.value.length<6)
                    {
```

```
                alert("password length should be greater than 6");
                form.password.focus();
                return false;
                }
                return true;
                }
</script>
</head>
<body>
        <center>
        <form name="form" onsubmit="return validate(this);"
        action="login.jsp">
        <h1>Login Here</h1>
        <table>
        <tr><td>Enter Name</td><td> <input type="text"
        name="name"></td></tr>
        <tr><td>Enter Password</td><td> <input
        type="password" name="password"></td></tr>
        <tr><td colspan='2' align='center'><input type="submit"
        value="Login"></td></tr>
        </table>
        </form>
        </center>
</body>
</html>
```

Output:

Registration page validation Example:

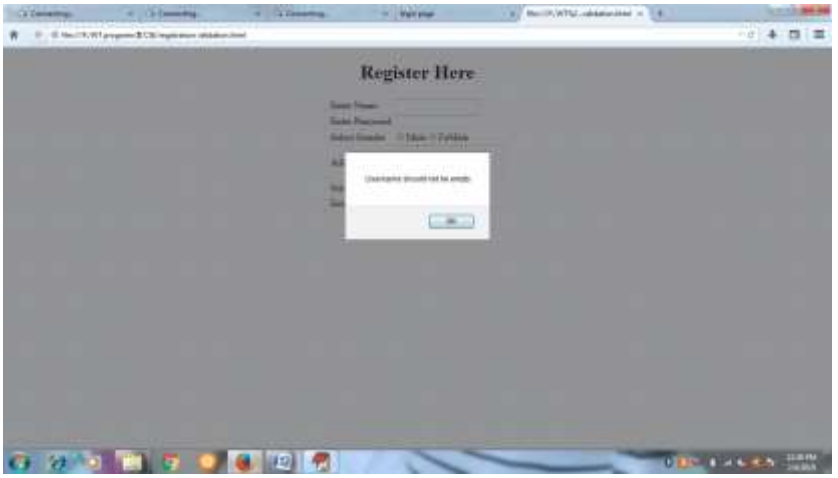
```
<html>
<head>
<script language="javascript" type="text/javascript">
                    function validate()
                    {
                    if(form.name.value==0)
                    {
```

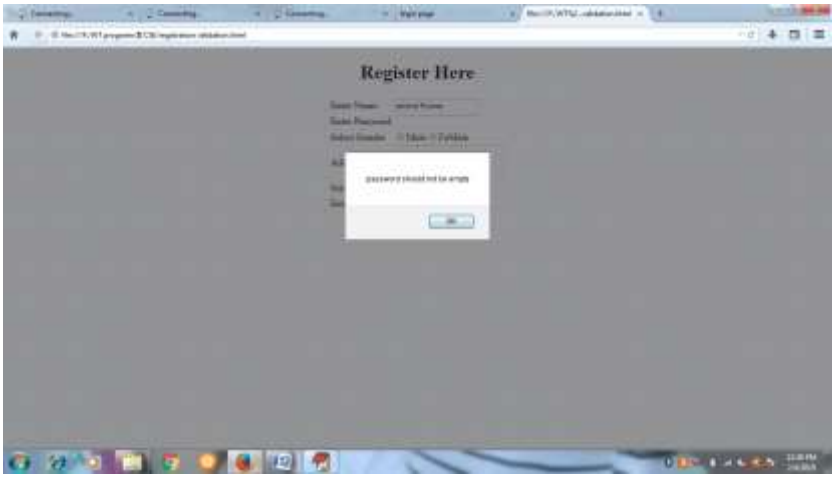
```
alert("Username should not be empty");
form.name.focus();
return false;
}
if(form.name.value.length<8)
{
alert("Username should be minimum 8
characters");
form.name.focus();
return false;
}
if(form.password.value==0)
{
alert("password should not be empty");
form.password.focus();
return false;
}
if(form.password.value.length<6)
{
alert("password length should be greater than 6");
form.password.focus();
return false;
}
if(form.gender.value==0)
{
alert("please select gender");
form.name.focus();
return false;
}
if(form.address.value==0)
```

```
                {
                alert("Address should not be empty");
                form.name.focus();
                return false;
                }
                if(form.mobile.value==0)
                {
                alert("Mobile num should not be empty");
                form.name.focus();
                return false;
                }
                if(form.mobile.value.length<10)
                {
                alert("Mobile num should be 10 digits");
                form.name.focus();
                return false;
                }
                return true;
                }
</script>
</head>
<body>
        <center>
        <form name="form" onsubmit="return validate(this);"
        action="login.jsp">
        <h1>Register Here</h1>
        <table>
        <tr><td>Enter Name</td><td> <input type="text"
        name="name"></td></tr>
```

```
        <tr><td>Enter Password</td><td> <input
        type="password" name="password"></td></tr>
        <tr><td>Select Gender</td><td><input type="radio"
        name="gender" value="male">Male<input type="radio"
        name="gender" value="female">FeMale</td></tr>
        <tr><td>Address</td><td><textarea
        name="address"></textarea></td></tr>
        <tr><td>Select State</td>
        <td>
        <select name="country">
        <option value="Srilanka">Srilanka
        <option value="India">India
        <option value="Australia">Australia
        </select>
        </td>
        </tr>
        <tr><td>Enter Mobile</td><td> <input type="text"
        name="mobile"></td></tr>
        <tr><td colspan='2' align='center'><input type="submit"
        value="Login"></td></tr>
        </table>
        </form>
        </center>
</body>
</html>
```

Output:





INTRODUCTION TO JQUERY:

- jQuery is a Client-side javascript library Created by John Resig in the year 2006.
- **Definition:**
  - jQuery is a lightweight, "write less, do more", javascript library.
  - Designed to simplify - HTML DOM traversal & manipulation, Event handling, CSS animation and AJAX.
- It is free, open source software.
- JQuery is a scripting language. Unlike traditional programming languages, it is interpreted, not executed.
- The purpose of jQuery is to make it much easier to use JavaScript on your website.
- jQuery's syntax is designed to make it easier to navigate a document, select DOM elements, create animations, handle events, and develop Ajax applications.
- It also provides capabilities for developers to create plug-ins on top of the JavaScript library.
- **Advantages of jQuery:**
  1. Simple and easy to use:
     - jQuery library is built using simpler and shorter codes.
     - It consists of a large number of predefined methods which can be directly used in our applications.

- With its open coding standards and simple syntax, web designers can shorten the time that it takes to deploy a site or application.

2. Compact and light weight library about 19KB in size.
3. Open source library:
   - jQuery is an open source library that is free and supported well across different applications.
   - This means that anyone can use this language in their applications without worrying about any licensing or compatibility issues.

4. Separates JavaScript and HTML:
   - Instead of using HTML attributes to call JavaScript functions for event handling, jQuery can be used to handle events purely in JavaScript. Thus, the HTML tags and JavaScript can be completely separated.
5. Cross-browser compatibility:
   - JavaScript engines of different browsers differ slightly so JavaScript code that works for one browser may not work for another.
   - jQuery handles all these cross browser inconsistencies and provides a consistent interface that works across different browsers.
6. AJAX support:

- Enables a web page to make AJAX requests to a web server to add the data, without reloading the page.

7. Event handling:
   - jQuery is tailor-made to respond to events in an HTML page. In jQuery, most DOM events have an equivalent jQuery method to handle them.
8. Custom animations and effects:
   - jQuery provides a lot of built in methods to add effects like fading and sliding of elements.
   - It also allows developer to add custom animations to web pages.
9. HTML/DOM manipulation:
   - The DOM is a tree structure representation of all the elements of a webpage.
   - The jQuery made it easy to select DOM elements, traverse them and modifying their content.
   - jQuery methods like html(), text(), val() and attr() can be used for this purpose.
10. Extensibility:
    - jQuery makes extending the framework very simple. New events, elements and methods can be easily added and then reused as plug-in.
11. Brevity and clarity: jQuery promotes brevity and clarity with features like chainable functions and shorthand function name.

jQuery Syntax:

- The jQuery syntax is used for selecting HTML elements and performing some action on the element(s).
- Basic syntax is:
  1. **$(*selector*).*action*()**
  2. ***$(selector).action(function(){***

     });

     - A $ sign to define/access jQuery
     - A (*selector*) to "query (or find)" HTML elements
     - A jQuery *action*() to be performed on the element(s)
- Examples:
  - $("p").hide() - hides all <p> elements.
  - $(this).hide() - hides the current element.

- **The Document Ready Event:**
  - To prevent any jQuery code from running before the document is finished loading all jQuery methods are written inside a document ready event:
  - Syntax:

```
$(document).ready(function(){
  // jQuery methods go here...
});
```

EVENTS:

- jQuery is Event-driven respond to events in an HTML page.
- An event represents the precise moment when something happens.
- A program contains necessary block of code known as event handler, to handle an event.
- **Example:**
  - Clicking of a mouse
  - Loading of a web page
  - Pressing a key on a keyboard
  - Submitting a form

Event	Description
blur	Occurs when an element loses focus
change	Occurs when the value of an element has been changed
click	Occurs when an element is clicked
dblclick	Occurs when an element is double-clicked.
focus	Occurs when an element gets focus
hover	When the mouse pointer hovers over the selected elements.
keydown, keypress	Occurs when a keyboard key is pressed down.
keyup	Occurs when a keyboard key is released
load	Occurs when a specified element has been loaded

mousedown	Occurs when the left mouse button is pressed down over the selected element.
mouseenter	Mouse pointer enters the selected element.
mouseleave, mouseout	Mouse pointer leaves the selected element.
mousemove	Mouse pointer moves within the selected element.
mouseover	Mouse pointer is over the selected element.
mouseup	Left mouse button is released over the selected element.
ready	Occurs when the dom (document object model) has been loaded.
resize	Occurs when the browser window changes size.
scroll	Occurs when the user scrolls in the specified element.
select	Occurs when a text is selected in a text area or a text field.
submit	Occurs when a form is submitted.
unload	Occurs when the user navigates away from the page

Example:

```
<html>
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/
3.4.1/jquery.min.js">
</script>
```

```
<script>
    $(document).ready(function()
    {
            $("#p1").click(function()
            {
                    alert("You clicked on the
paragraph");
            });
            $("#p2").dblclick(function()
            {
                    alert("You double clicked on the
paragraph");
            });
            $("#p3").hover(function()
            {
                    alert("mouse moved over the
paragraph");
            });
    });
</script>
</head>
<body>
<p id="p1">Click on this paragraph</p>
<br>
<p id="p2">Double Click on this paragraph</p>
<br>
<p id="p3">Move cursor over this paragraph</p>
</body>
</html>
```

Output:

Keyboard events:

List of keyboard events:

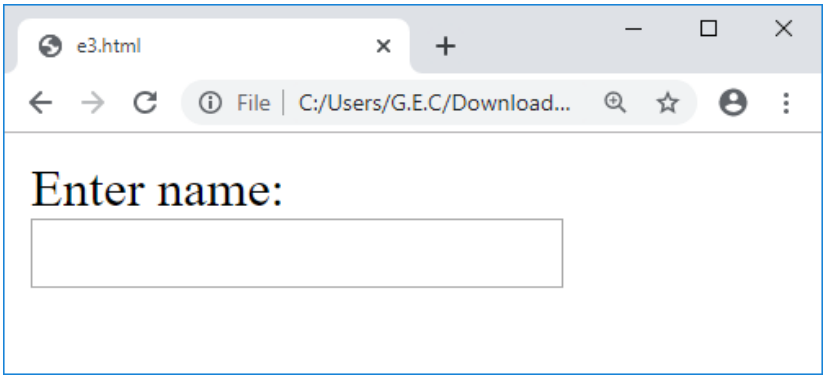
1. keydown
2. keypress
3. keyup

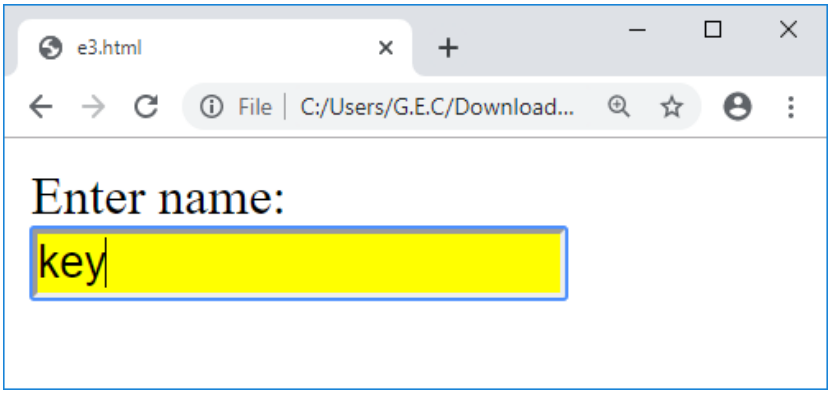
Example:

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

```
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquer
y.min.js">

</script>
<script>
 $(document).ready(function()
 {
        $("input").keydown(function()
        {
                $(this).css("background-
color","yellow");
        });
        $("input").keyup(function()
        {
                $(this).css("background-
color","pink");
        });

 });
</script>
</head>
<body>
<form method="post">
Enter name:<input type="text"
name="t1"><br>
</form>
</body>
</html>
```

Output:





Mouse events:

List of Mouse events:

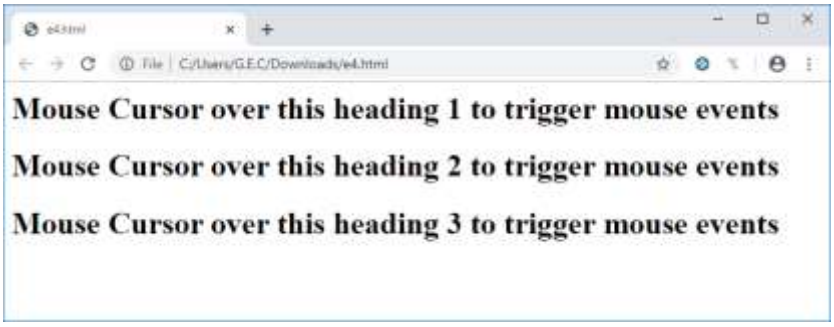
1. mousedown
2. mouseenter
3. mouseleave
4. mousemove
5. mouseout
6. mouseover
7. mouseup

Example:

```
<html>
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquer
y.min.js">
</script>
<script>
        $(document).ready(function()
        {
        $("#i1").mouseenter(function()
        {
                $("#i1").css("background-color","yellow");
        });
        $("#i1").mouseout(function()
        {
                $("#i1").css("background-color","pink");
        });
        $("#i2").mousedown(function()
        {
                $("#i2").css("background-color","blue");
        });
        $("#i2").mouseup(function()
        {
                $("#i2").css("background-color","green");
        });
        $("#i3").mouseover(function()
        {
                alert("cursor is over this heading");
        });
```

```
        });
</script>
</head>
<body>
<h1 id="i1">Mouse Cursor over this heading 1 to trigger mouse events</h1>
<h1 id="i2">Mouse Cursor over this heading 2 to trigger mouse events</h1>
<h1 id="i3">Mouse Cursor over this heading 3 to trigger mouse events</h1>
</body>
</html>
```

Output:



Form events:

List of form events:

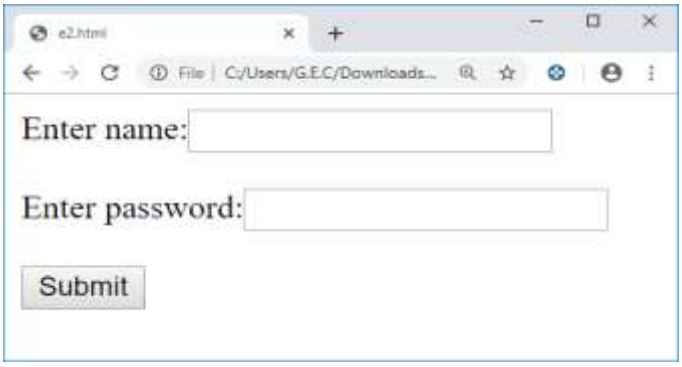
1. Submit
2. Change
3. Focus
4. Blur

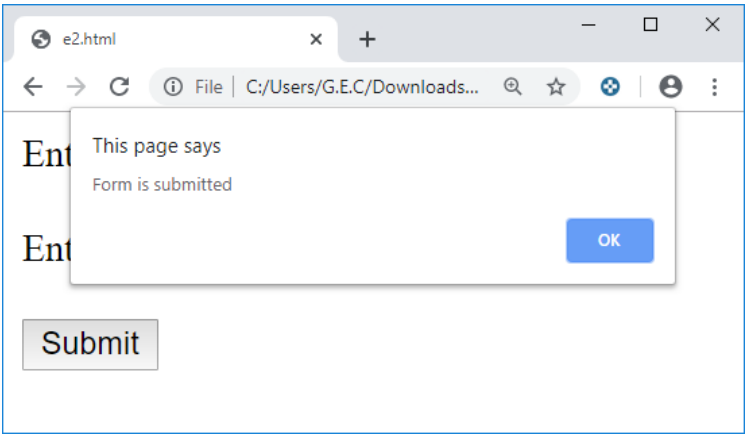
Example:

```
<html>
<head>
<script
                    src="https://ajax.googleapis.com/ajax/libs/j
                    query/3.4.1/jquery.min.js">
</script>
<script>
                    $(document).ready(function()
                    {
                          $("form").submit(function()
                          {
                                    alert("Form is submitted");
                          });
                          $("input").focus(function()
                          {
                                    $(this).css("background-
                    color","yellow");
                          });
                          $("input").blur(function()
                          {
                                    $(this).css("background-
                    color","pink");
                          });
```

```
                $("input").change(function()
                {
                        alert("Text is changed");
                });
                $("input").select(function()
                {
                        alert("Text is selected");
                });
        });
</script>
</head>
<body>
<form method="post">
Enter name:<input type="text" name="t1"><br>
Enter password:<input type="password" name="t2"><br>
<button type="submit">Submit</button>
</form>
</body>
</html>
```

Output:





Document/Window Events:

List of Document/Window Events:

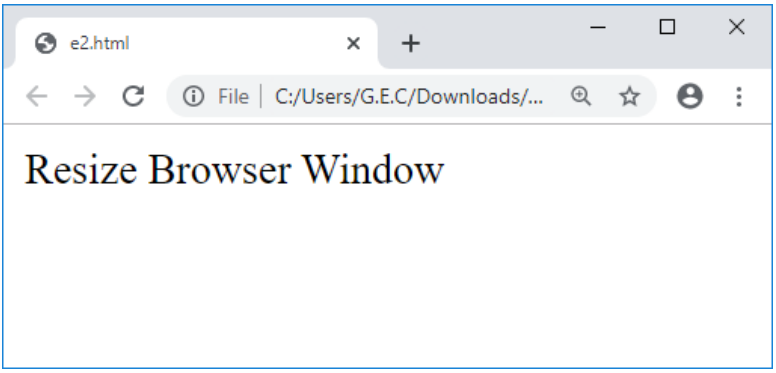
1. Load
2. Resize
3. Scroll
4. Unload

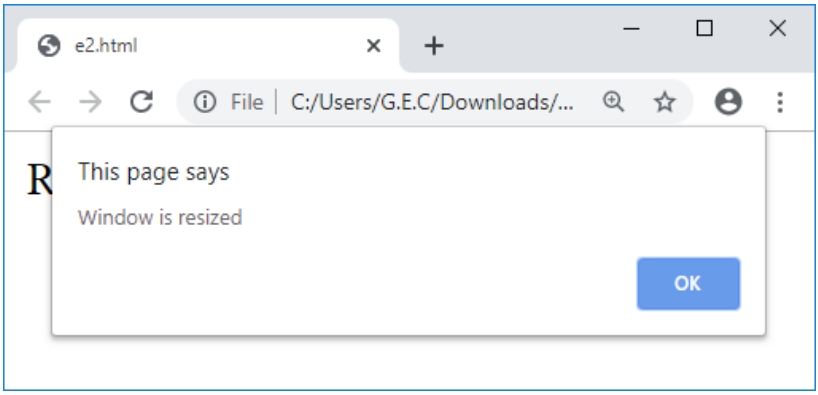
Example:

```
<html>
<head>
<script
        src="https://ajax.googleapis.com/ajax/libs/j
        query/3.4.1/jquery.min.js"></script>
```

```
<script>
$(document).ready(function(){
  $(window).resize(function(){
                        alert("Window is resized");
  });
});
</script>
</head>
<body>
<p>Resize Window</p>
</body>
</html>
```

Output:





EFFECTS:

- The jQuery library provides several techniques for adding animation to a web page.
- It contains various methods to apply simple, standard animations that are frequently used, and also sophisticated custom effects.

Effects	Method	Description
Hide/Show	hide()	Hides the selected elements
	show()	Shows the selected elements
	toggle()	Toggles between the hide() and show() methods
Fading	fadeIn()	Fades in the selected elements
	fadeOut()	Fades out the selected elements
	fadeTo()	Fades in/out the selected elements to a given opacity
	fadeToggle()	Toggles between the fadeIn() and fadeOut() methods
Sliding	slideUp()	Slides-up (hides) the selected elements
	slideDown()	Slides-down (shows) the selected elements
	slideToggle()	Toggles between the slideUp() and slideDown() methods
Animation	animate()	Runs a custom animation on the selected elements

Stop	stop()	Stops the currently running animation for the selected elements
	delay()	Sets a delay for all queued functions on the selected elements. $(selector).delay(speed) Example: $("h1").delay("slow").fadeIn();

Showing and Hiding of elements:

- **hide()** - hide() method hides the selected elements.
  - Syntax:

    $(selector).hide(speed,callback);
  - speed - Specifies the speed of the hide/show effect.

    Possible values: milliseconds, slow, fast.
  - callback - A function to be executed after the method is completed.
- **show()** - shows the hidden, selected elements.
  - Syntax:

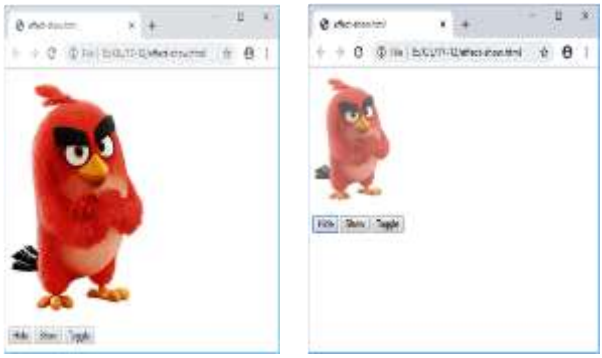
    $(selector).show(speed,callback);
- **toggle()** - toggles between **hide()** and **show()** for the selected elements.
  - show() is run if an element is hidden.
  - hide() is run if an element is visible
  - Syntax:

    $(selector).toggle(speed,callback);

Example:

```
<html>
<head>
<script
src="http://ajax.googleapis.com/ajax/libs/jquery/3
.4.1/jquery.min.js">
</script>
<script>
$(document).ready(function(){
    $("#b1").click(function(){
        $("img").hide(1000);
    });
    $("#b2").click(function(){
        $("img").show("slow");
    });
    $("#b3").click(function(){
        $("img").toggle("fast");
    });
});
</script>
</head>
<body>
<img src="6.png">
<button id="b1"> Hide </button>
<button id="b2"> Show </button>
<button id="b3"> Toggle </button>
</body>
</html>
```

Output:



Fading effects:

With jQuery you can fade an element in and out of visibility. jQuery has the following fade methods:

- **fadeIn() -** used to fade in a hidden element.
  - Syntax:

    $(selector).fadeIn(speed,callback);
- **fadeOut() -** used to fade out a visible element.
  - Syntax:

    $(selector).fadeOut(speed,callback);
- **fadeToggle() –** toggles between the fadeIn() and fadeOut() methods.
  - If the elements are faded out, fadeToggle() will fade them in.
  - If the elements are faded in, fadeToggle() will fade them out.
  - Syntax:

    $(selector).fadeToggle(speed,callback);
- **fadeTo() -** allows fading to a given opacity (value between 0 and 1).

- Syntax:

$(selector).fadeTo(speed,opacity,callback);

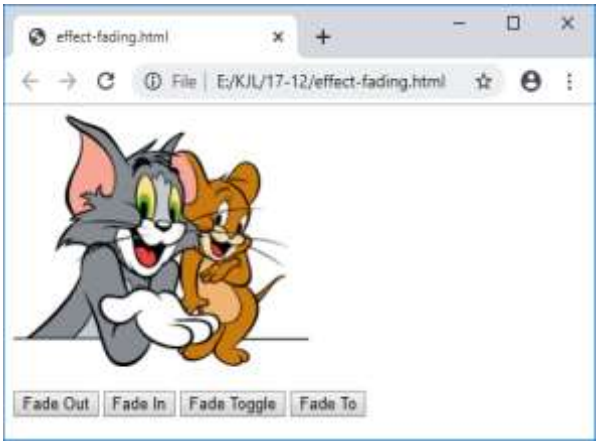
- Opacity: Specifies the opacity to fade to. Must be a number between 0.00 and 1.00.

Example:

```
<html>
<head>
<script
src="http://ajax.googleapis.com/ajax/libs/jquery/3.4.1
/jquery.min.js">
</script>
<script>
$(document).ready(function(){
        $("#b1").click(function(){
                $("img").fadeOut(1000);
        });
        $("#b2").click(function(){
                $("img").fadeIn("slow");
        });
        $("#b3").click(function(){
                $("img").fadeToggle("fast");
        });
        $("#b4").click(function(){
                $("img").fadeTo("slow",0.3);
        });

});
</script>
</head>
<body>
```

```
<img src="d1.png"><br><br>
<button id="b1"> Fade Out </button>
<button id="b2"> Fade In </button>
<button id="b3"> Fade Toggle </button>
<button id="b4"> Fade To </button>
</body>
</html>
```

Output:



Sliding effects:

With jQuery you can create a sliding effect on elements. jQuery slide methods slide elements up and down.

- **slideUp()** – used to slide up an element.
  - Syntax:

  $(selector).slideUp(speed,callback);
- **slideDown()** - used to slide down an element.
  - Syntax:

  $(selector).slideDown(speed,callback);
- **slideToggle()** – toggles between the slideDown() and slideUp() methods.
  - If the elements have been slide down, slideToggle() will slide them up.
  - If the elements have been slide up, slideToggle() will slide them down.
  - Syntax:

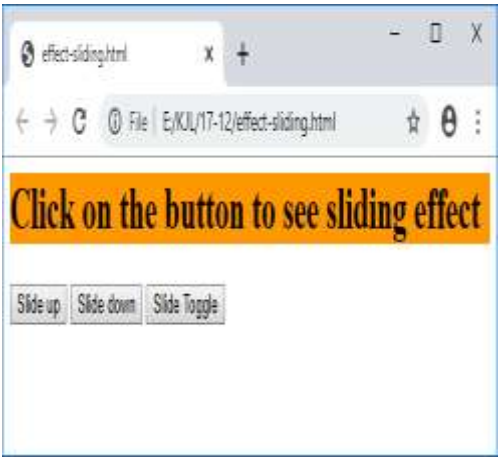
  $(*selector*).slideToggle(*speed,callback*);

Program for sliding effects:

```
<html>
<head>
<script
src="http://ajax.googleapis.com/ajax/libs/jquery/3.4.1/j
query.min.js">
</script>
<script>
$(document).ready(function(){
        $("#b1").click(function(){
            $("h1").slideUp(1000);
        });
```

```
            $("#b2").click(function(){
                $("h1").slideDown("slow");
            });
            $("#b3").click(function(){
                $("h1").slideToggle("fast");
            });
    });
    </script>
    </head>
    <body>
    <h1 style="background-color:orange">Click on the button
    to see sliding effect</h1>
    <button id="b1"> Slide up </button>
    <button id="b2"> Slide down </button>
    <button id="b3"> Slide Toggle </button>
    </body>
    </html>
```

Output:



Animate() and stop():

- **animate()**
  - animate() method is used to create custom animations.
  - multiple (CSS) properties can be animated at the same time using animate()
  - changes an element from one state to another with CSS styles.
  - Syntax:

$(selector).animate({params},speed,callback);

    - **params** - required parameter defines the CSS properties to be animated.

- **Stop() –**
  - The jQuery stop() method is used to stop an animation or effect before it is finished.
  - works for all jQuery effect functions, including sliding, fading and custom animations.
  - Syntax:

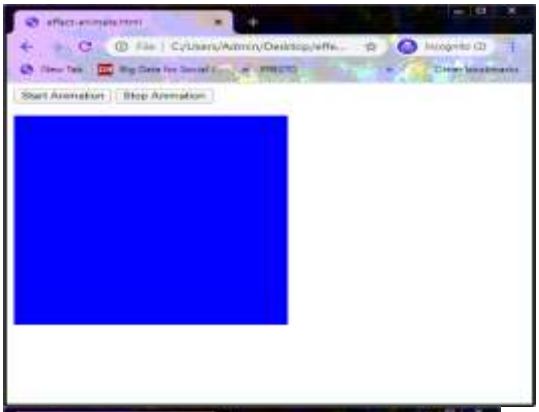
```
$(selector).stop();
```

- kills the current animation being performed on the selected element.

- **Program for animate and stop:**

```
<html>
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/
3.4.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("#b1").click(function(){
    $("div").animate({left:'850px', height:'+=250',
width:'+=250px'},1000);
  });
  $("#b2").click(function(){
    $("div").stop();
  });
});
</script>
</head>
<body>
<button id="b1">Start Animation</button>
<button id="b2">Stop Animation</button>
<br><br>
<div style="background-
color:blue;width:300px;height:300px;position:absolu
te"></div>
</body>
</html>
```

Output:


Start Animation
Stop Animation


Start Animation
Stop Animation

BOOT STRAP:

- Bootstrap is a free front-end framework for faster and easier web development
- Bootstrap includes HTML and CSS based design templates for typography, forms, buttons, tables, navigation, modals, image carousels and many other, as well as optional JavaScript plugins
- Bootstrap also gives you the ability to easily create responsive designs
- Responsive web design is about creating web sites which automatically adjust themselves to look good on all devices, from small phones to large desktops.

There are two ways to start using Bootstrap on your own web site.

You can:

- Download Bootstrap from getbootstrap.com
- Include Bootstrap from a CDN

```
<!-- Latest compiled and minified CSS -->
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">

<!-- jQuery library -->
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>

<!-- Latest compiled JavaScript -->
```

```
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>
```

1. Add the HTML5 doctype

Bootstrap uses HTML elements and CSS properties that require the HTML5 doctype.

Always include the HTML5 doctype at the beginning of the page, along with the lang attribute and the correct character set:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
  </head>
</html>
```

2. Bootstrap 3 is mobile-first

Bootstrap 3 is designed to be responsive to mobile devices. Mobile-first styles are part of the core framework.

To ensure proper rendering and touch zooming, add the following <meta> tag inside the <head> element:

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

The width=device-width part sets the width of the page to follow the screen-width of the device (which will vary depending on the device).

The initial-scale=1 part sets the initial zoom level when the page is first loaded by the browser.

3. Containers

Bootstrap also requires a containing element to wrap site contents.

There are two container classes to choose from:

1. The .container class provides a responsive **fixed width container**
2. The .container-fluid class provides a **full width container**, spanning the entire width of the viewport

Bootstrap's grid system allows up to 12 columns across the page.

If you do not want to use all 12 columns individually, you can group the columns together to create wider columns:

Bootstrap's grid system is responsive, and the columns will re-arrange automatically depending on the screen size.

Grid Classes

The Bootstrap grid system has four classes:

- xs (for phones - screens less than 768px wide)
- sm (for tablets - screens equal to or greater than 768px wide)
- md (for small laptops - screens equal to or greater than 992px wide)
- lg (for laptops and desktops - screens equal to or greater than 1200px wide)

The classes above can be combined to create more dynamic and flexible layouts.

Example:

```
<div class="row">
  <div class="col-sm-4">.col-sm-4</div>
  <div class="col-sm-8">.col-sm-8</div>
</div>
```

Generally, Bootstrap 4 is distributed using the repositories Bower (via Github) and NPM (node package manager). Moreover, you also can create your own distribution and use to the source code that connects/links directly to the website.1 Bootstrap also utilizes the raw files of the cascading stylesheets language SASS□this is a precompiler that translates into CSS (unlike its predecessor, Bootstrap 3, where the primary language was LESS).

You can load bootstrap from a CDN (content delivery network) or locally. The local version can be pulled from Bower, npm, Github, or the Bootstrap website respectively.

```
bootstrap/
 ├──── css/
 |    ├──── bootstrap.css
 |    ├──── bootstrap.css.map
 |    ├──── bootstrap.min.css
|    ├──── bootstrap.min.css.map
 |    ├──── bootstrap-flex.css
 |    ├──── bootstrap-flex.css.map
 |    ├──── bootstrap-flex.min.css
 |    ├──── bootstrap-flex.min.css.map
 |    ├──── bootstrap-grid.css
 |    ├──── bootstrap-grid.css.map
 |    ├──── bootstrap-grid.min.css
 |    ├──── bootstrap-grid.min.css.map
 |    ├──── bootstrap-reboot.css
 |    ├──── bootstrap-reboot.css.map
 |    ├──── bootstrap-rebooot.min.css
 |    └──── bootstrap-rebooot.min.css.map
 ├──── js/
 |    ├──── bootstrap.js
 |    ├──── bootstrap.min.js
Font-Awesome/
 ├──── css/
 |    ├──── fontawesome.css
 |    ├──── fontawesome.css.map
 |    ├──── fontawesome.min.css
 ├──── fonts/
 |    ├──── fontawesome-webfont.eot
 |    ├──── fontawesome-webfont.svg
 |    ├──── fontawesome-webfont.ttf
 |    ├──── fontawesome-webfont.woff
 |    ├──── fontawesome-webfont.woff2
Tether/
 ├──── js/
 |    └──── tether.js
 └──── css/
 |    ├──── tether.css
      └──── tether.min.css
```

Once everything is ready, you can create the first page. This page should provide the basic

layout of the entire application.

```
<!DOCTYPE html>
<html lang="en">
 <head>
 <meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-
scale =1, shrink-to-fit=no">
 <meta http-equiv="x-ua-compatible" content="ie=edge">
 <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/
bootstrap.min.css" crossorigin="anonymous">
 </head>
 <body>
 <h1>Hello Bootstrap 4</h1>
 <script
src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.4/
jquery.min.js"></script>
 <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/
bootstrap.min.js" crossorigin="anonymous"></script>
 </body>
 </html>
```

Typography

Bootstrap's global default font-size is 14px, with a line-height of 1.428.

This is applied to the <body> element and all paragraphs (<p>).

In addition, all <p> elements have a bottom margin that equals half their computed line-height (10px by default).

- **<mark>**
- **<abbr>**
- **<blockquote>**
- **<dl>**
- **<code>**
- **<kbd>**
- **<pre>**
- table
- table-striped
- table-bordered
- table-hover

Forms

Forms are fully supported in Bootstrap 4. Many of the components are mainly used to make the forms responsive and can be used with any screen width

Form elements automatically receive the correct formatting. The main class for controls is *.form-control*. Elements that have controllable horizontal extensions such as <input>, <textarea>, and <select> are set to a width of 100% of the parent container. Using *.formgroup*the labels and inputs are grouped. They arrange themselves depending on the available width either side-by-side or above one another.

```
<form>
<div class="form-group">
<label for="txtMail">eMail</label>
<input type="email" class="form-control" id="txtMail"
placeholder="eMail">
</div>
 <div class="form-group">
<label for="txtPassword">password</label>
<input type="password" class="form-control"
 id="txtPassword" placeholder="Password">
 </div>
<div class="form-group">
 <label for="txtFile">File Selection</label>
 <input type="file" id="txtFile">
 <p class="form-text small">
 Here is the help for uploading. </p>
</div>
 <div class="checkbox">
 <label>
 <input type="checkbox"> Save
 </label>
 </div>
 <button type="submit" class="btn btn-secondary"> Send
</button>
 </form>
```