JavaScript

Notes By

Naveen

What is JavaScript?

JavaScript is an interpreted, client-side, event-based, object oriented scripting language that you can use to add dynamic interactivity to your web pages.

You can use JavaScript to achieve any of the following.

1. Create special effects with images that give the impression that a button is either highlighted or depressed whenever the mouse pointer is hovered over it.

2. Validate information that users enter into your web forms.

3. Open pages in new windows, and customise the appearance of those new windows.

4. Detect the capabilities of the user's browser and alter your page's content appropriately.

5. Create custom pages "on the fly" without the need for a server-side language like PYTHON.

6. Client-side validation

7. Dynamic drop-down menus

8. Displaying date and time

9. Displaying pop-up windows and dialog boxes …

JavaScript Example

JavaScript example is easy to code. JavaScript provides 3 places to put the JavaScript code: within body tag, within head tag and external JavaScript file.

1) Code between the body tag

Open a notepad or any html editor and type the below program and save it with any name but the ext must be ".html".

```
<html>
  <body>
    <script>
      /*Function without argument and without return*/
      function show()
      {
        alert('Python With Naveen')
      }
    </script>
      <button type="submit"  onclick="show()">
        Python
      </button>
      <button type="submit" onclick="show()">
        Django

      </button>
  </body>
</html>
```

To Run the Program just double click on the file, It will Run in a web browser.

2) Code between the head tag

Let's see the same example of displaying alert dialog box of JavaScript that is contained inside the head tag.

```
<html>
  <head>
    <script>
      /*Function without argument and without return*/
      function show()
      {
        alert('Python With Naveen')
      }
    </script>
  </head>

  <body>
    <button type="submit"  onclick="show()">
      Python
    </button>
    <button type="submit"  onclick="show()">
      Django
    </button>
  </body>
</html>
```

To Run the Program just double click on the file, It will Run in a web browser.

Note: We can use any browser to the output.

3) External JavaScript (".js") file

Step 1: Open a notepad or any JavaScript file editor, write the below program and save using any name but the file extension must be ".js".

```
function display()
{
   alert("Welcome Python With Naveen")
}
```

Step 2: Using the ".js" file in HTML file.

```
<html>
   <head>
      <script type="text/javascript" src="example.js"></script>
   </head>


   <body bgcolor="red">
      <button type="submit" onclick="display()">1st Button</button>
   </body>
</html>
```

Note: save the ".js" and ".html" in the same folder as per this program.

To Run the Program just double click on the file, It will Run in a web browser.

Note: We can use any browser to the output.

JavaScript Single line Comment

It is represented by double forward slashes (//). It can be used before and after the statement.

Example :

```
// alert("Welcome Python With Naveen")
```

JavaScript Multi line Comment

It can be used to add single as well as multi line comments.

It is represented by forward slash with asterisk (/*) then asterisk with forward slash (*/) .

Example:

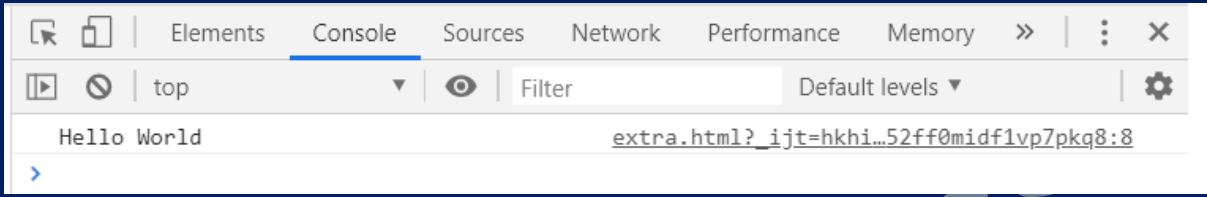
```
/*function display()
{
   alert("Welcome Python With Naveen")
}*/
```

Example on console

```
<html>
<body>
   <script>
      console.log("Hello World")
   </script>
</body>
</html>
```

In the above example, the **console.log()** function prints Hello, World! to the console and returns undefined. This is because console.log() has no explicit *return value*.

Output



Data Types

Primitive Data Types

1. String
2. Number
3. Boolean
4. Null
5. Undefined

Non-primitive Data Type

1. Object
2. Date
3. Array

Variable

Variable is a named memory location to store some data temporally.

Syntax:

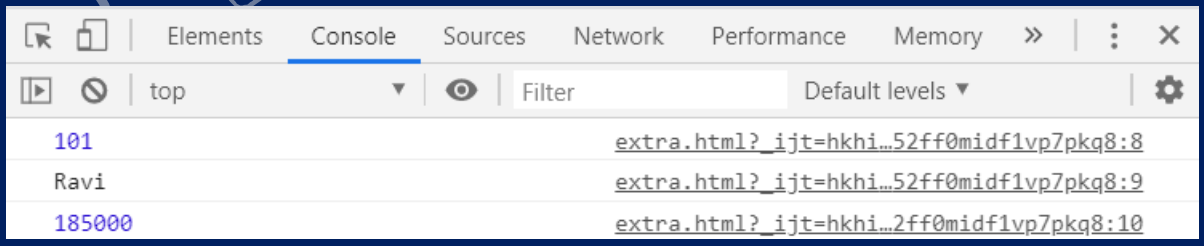
```
var   variable_name   =     value;
```

Note: var is a keyword used to declare a variable.

Example

```
<html>
<body>
  <script>
    var idno = 101;
    var name = "Ravi";
    var salary = 185000.00;
    console.log(idno)
    console.log(name)
    console.log(salary)
  </script>
</body>
</html>
```

Output



In JavaScript we can declare multiple variable in 1 line by separating with comma (,) .

Example

```
<html>
<body>
   <script>
      var idno = 101,name="Ravi"
      console.log(idno)
      console.log(name)
   </script> </body>
</html>
```

Operators

Arithmetic Operators

Operator	Operation	Expression	Result
+	Addition	2 + a	4
-	Subtraction	2 - a	0
*	Multiply	3 * a	6
/	Division	3 / a	1.5
%	Modulus	7 % a	1

Relational Operators

In the table below the variables are: var a = 1 , b = 2.

Operator	Operation	Expression	Result
==	Equal to	a == b	false
!=	Not equal to	a != b	true
<=	Less than or equal to	a <= b	true
>=	Greater than or equal to	a >= b	false
<	Less than	a < b	true
>	Greater than	a > b	false

Assignment Operator

Operator	Operator Name	Description	Example
=	Assignment	It assigns a value from the right-side operand to the left-side operand.	I = 40 It assigns 40 to I

Shorthand Assignment Operator

Operator	Operator Name	Example
+=	Add then assign	I+=J that means I = I + J
-=	Subtract then assign	I-=J that means I = I – J
*=	Multiply the assign	I*=J that means I = I * J
/=	Divide then assign	I/=J that means I = I / J

Logical Operators

In the table below the variables are: var a = 1,b = 2.

Operator	Operation	Expression	Result
&&	Logical and. Returns true only if both its first and second operands are evaluated to true.	a < 3 && b > 5	returns false as b > 5 is false
||	Logical or. Returns true if one of the two operands are evaluated to true, returns false if both are evaluated to true.	a < 3 || b > 5	returns true as a < 3 is true
!	Logical not. Unary operator that simply inverts the Boolean value of its operand.		

Bitwise Operators in Python

For instance, suppose there are two variables, I = 10 and J = 20

And their binary values are:

I = 10 = 00001010

J = 20 = 00010100

Operator	Operator Name	Example
&	Binary AND	I & J ANS = 0000 0000
|	Binary OR	I | J ANS = 0001 1110

^	Binary XOR	I ^ J ANS = 0001 1110
~	Binary Complement	~I ANS = 1111 0101
<<	Binary Left Shift	I << 2 ANS = 40 i.e. 1111 0000
>>	Binary Right Shift	I >> 2 ANS = 15 i.e. 1111

Infinity

In most languages, dividing a number by 0 throws an error and stops program execution.

JavaScript however, returns as *Infinity*.

1. 10 / 0; // => Infinity
2. -10 / 0; // => -Infinity
3. 2 / +0; // => +Infinity
4. 2 / -0; // => -Infinity

NaN

In JavaScript when an arithmetic operation fails it does not throw an error, instead it returns a special numeric value, called *NaN* (Not a Number)

1. Infinity / Infinity; // => NaN.
2. 0 / 0; // => NaN.
3. "Hello" / 1; // => NaN.

"Hello" cannot be converted to number

4. Math.sqrt(-1); // => NaN.
5. NaN / 0; // => NaN.

Any operation with NaN results in NaN

++ : Increment Operator:

It will increment a variable value by 1

Post Increment : variableName ++

Example

```
<html>
<body>
   <script>
      var a = 10;
      console.log("a=",a); //a=10
      console.log("a=",a++); //a=10
      console.log("a=",a); //a=11
   </script>
</body>
</html>
```

Pre Increment : ++ variableName

Example

```
<html>
<body>
   <script>
      var a = 10;
      console.log("a=",a); //a=10
```

```
    console.log("a=",++a); //a=11
    console.log("a=",a); //a=11     </script> </body> </html>
```

-- : Decrement Operator:

It will decrement a variable value by 1

Post Decrement: variableName --

Example

```
<html>
<body>
  <script>
    var a = 10;
    console.log("a=",a); //a=10
    console.log("a=",a--); //a=10
    console.log("a=",a); //a=9
  </script>
</body>
</html>
```

Pre Decrement : -- variableName

Example

```
<html>
<body>
  <script>
    var a = 10;
    console.log("a=",a); //a=10
    console.log("a=",--a); //a=9
    console.log("a=",a); //a=9
```

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

What is the difference between === and ==

1. **var** x = 2;
2. **var** y = “2”;

In JavaScript x and y will behave same type. This becomes an issue when you want to compare two variables of two different *types*, e.g. an integer and a string. That’s why we use the **===** operator to **insure** both type AND value are the same.

Example 1

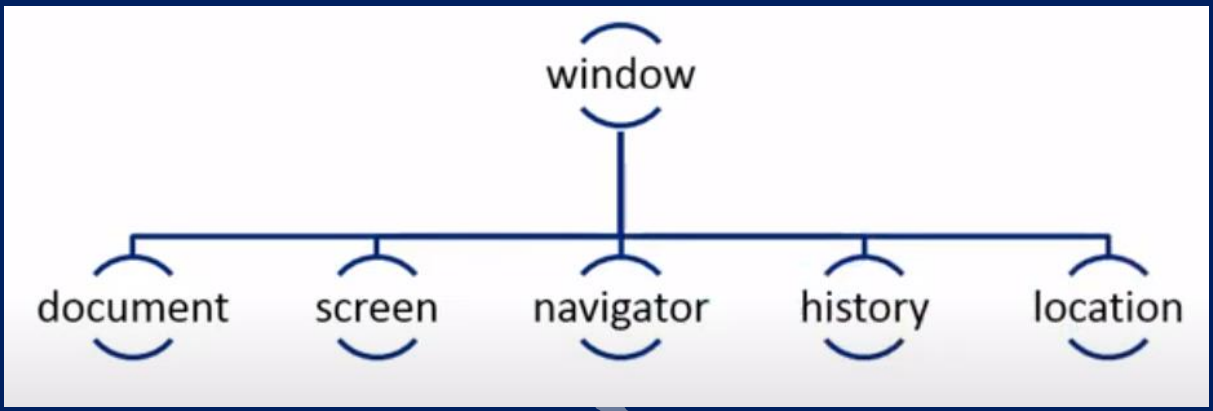
```
<html>
<body>
  <script>
      var x = 2;    // integer
      var y = "2";  // string
      console.log(x == y);
      console.log(x === y);
  </script>
</body>
</html>
```

1. **(x == y) -> true**
2. **if (x === y) -> false**

Browser Object Model

The **Browser Object Model** (BOM) is used to interact with the browser.

The default object of browser is **"window"**.



Example

```
window.alert("Python with Naveen");
```

(or)

```
alert("Python With Naveen");
```

In the above example both are same because **"window"** is the default.

Window Object

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

Methods of window object

The important methods of window object are as follows:

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.

Example

```
<html>
<body>
   <script>
      alert("Python With Naveen");

      var v= confirm("Hello Buddy are you attending class");
      alert(v)

      var v= prompt("Please Enter Your Name ");
      alert("Welcome Mr/Miss : "+v);
```

```
        open("Demo4.html", "PopUp", "toolbar=no, width=500,
height=300");

        setTimeout(alert("Python With Naveen"),2000);
    </script>
</body>
</html>
```

Document Object Model

The **document object** represents the whole html document.

By the help of document object, we can add dynamic content to our web page.

window.document

Is same as

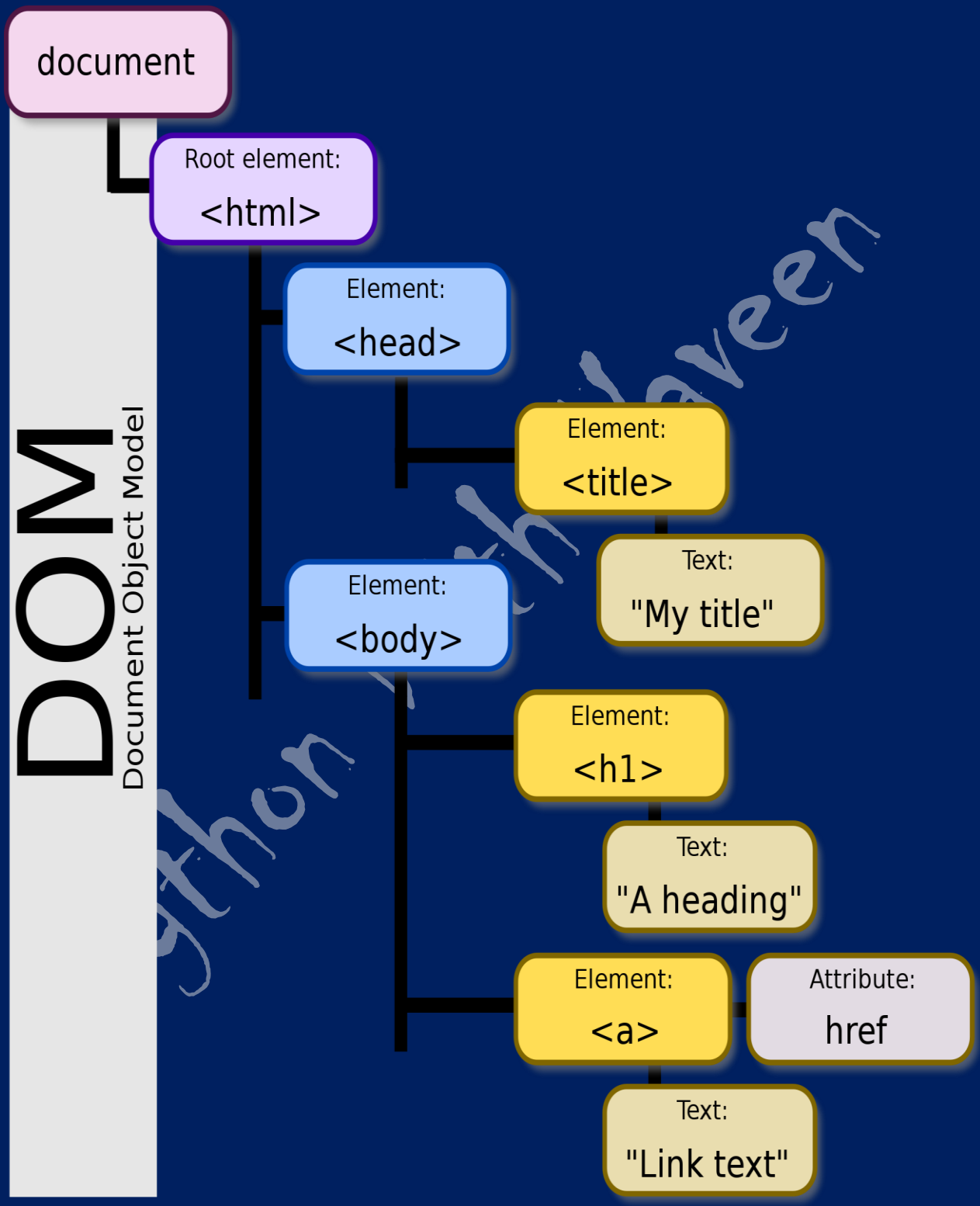
document

Methods of document object

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.

Properties of document object



Example on document.getElementsByName()

```
<html>
  <head>
    <title>Python With Naveen</title>
  </head>
  <body>
    <script>
      function totalelements()
      {
        var allgenders=document.getElementsByName("gender");
        alert("Total Genders:"+allgenders.length);
      }
    </script>
    <form>
      Male:<input type="radio" name="gender" value="male">
      Female:<input type="radio" name="gender"
value="female">
      <button type="submit"
onclick="totalelements()">Count</button>
    </form>
  </body>
</html>
```

Example on document.getElementsByTagName()

```
<html>
  <head>
    <title>Python With Naveen</title>
  </head>
  <body>
    <script>
      function totalelements()
      {
        var allinputs=document.getElementsByTagName("input");
        alert("Total Genders:"+allinputs.length);
      }
    </script>
    <form>
      Male:<input type="radio" name="gender" value="male">
      Female:<input type="radio" name="gender"
value="female">
      Others<input type="radio" name="gender" value="others">
      <button type="submit"
onclick="totalelements()">Count</button>
    </form>
  </body>
</html>
```

innerHTML

The **innerHTML** property can be used to write the dynamic html on the html document.

document.getElementById('d1').innerHTML="<h1> python with Naveen</h1>";

Example

```
<html>
   <head>
      <title>Python with Naveen</title>
      <script>
         function show()
         {
            document.getElementById("d1").innerHTML =
"<h1>Python With Naveen</h1>";
         }
      </script>
   </head>
   <body>
      <button type="submit" onclick="show()">Click</button>
      <div id="d1">..</div>
   </body>
</html>
```

innerText

The **innerText** property can be used to write the dynamic text on the html document. Here, text will not be interpreted as html text but a normal text.

document.getElementById('t1').innerText="Python With Naveen";

Example

```
<html>
   <head>
      <title>Python with Naveen</title>
      <script>
         function show()
         {
            document.getElementById("d1").innerText = "Python With Naveen";
         }
      </script>
   </head>
   <body>
         <button type="submit" onclick="show()">Click</button>
         <div id="d1">..</div>
   </body>
</html>
```

Form Validation

JavaScript provides facility to validate the form on the client-side so data processing will be faster than server-side validation.

Example

```
<html>
   <head><script>
      function validate()
      {
         var name = document.getElementById("t1").value.trim();
         if(name == "")
         {
            document.getElementById("s1").innerText = "Empty"
            return false
         }
         else
         {
            return true
         }
      }
   </script></head>
   <body>
   <form action="http://www.facebook.com" onsubmit = " return validate()">
      <input type="text" placeholder="name" id="t1">

       <span id="s1">*</span>
      <button type="submit">Submit</button>
   </form>   </body>   </html>
```

Example program to validate username and password empty or not.

```
<html>
  <head>
    <script>
      function validate()
      {
        var uname = document.getElementById("t1").value.trim();
        var upass = document.getElementById("t2").value.trim();
        var status = true
        if(uname == "")
        {
          document.getElementById("s1").innerText = "Empty";
          status = false
        }
        else
        {
          document.getElementById("s1").innerText = ""
            if(upass == "")
            {
            document.getElementById("s2").innerText = "Empty";
              status = false
            }
            else
            {
              document.getElementById("s2").innerText = ""
              console.log("OK")
            }
        }
```

```
        return status
      }
    </script>
  </head>
  <body bgcolor="#f5deb3">
  <form action="Demo3.html" onsubmit="return validate()">
    <table align="center" border="2">
      <tr><th colspan="2">Login Page</th></tr>
      <tr>
        <th>Username</th>
        <th><input type="text" id="t1"></th>
        <th>
          <span id="s1">*</span>
        </th>
      </tr>
      <tr>
        <th>Password</th>
        <th><input type="password" id="t2"></th>
        <th>
          <span id="s2">*</span>
        </th>
      </tr>
      <tr>
        <th colspan="2">
          <button type="submit">Login</button>
        </th>
      </tr>
    </table>
  </form>    </body>  </html>
```

Events

Event is a flow of action, Occurrence happening at a determinable time and place, with or without the participation of human or thing.

Syntax of a Event :-

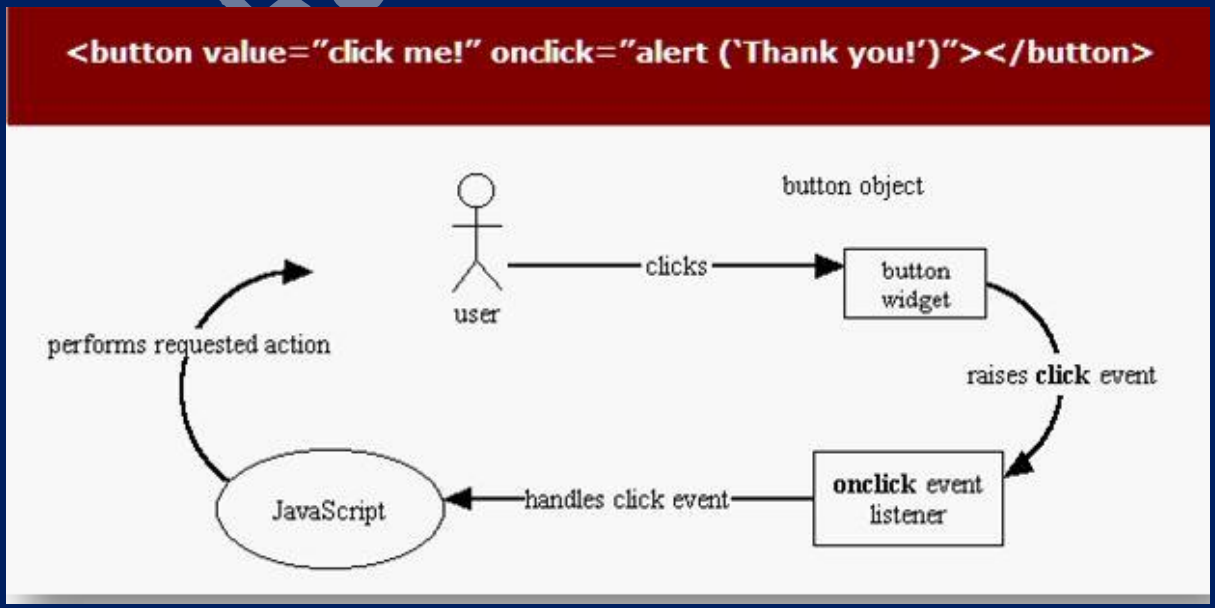
<button value="click me!" onclick="alert ('Thank you!')"></button>

Event Handler :

Using event handler, When action performed in DOM will calls a function in script. If function calling is written in DOM(HTML) that is called **Event Handler.**

Event Listener :

Using event handler, When action performed in DOM will calls a function in script.If function calling is written in script that is called **Event Listener**. While using Event Listener , you no need touch html – It means you can manage everything form javascript only without help of html.



JavaScript Events List

onload:- The browser has finished loading the page

unload:- The document or a dependent resource is being unloaded.

online:-The browser has gained access to the network.

offline:- The browser has lost access to the network.

focus:- An element has received focus (does not bubble).

blur:- An element has lost focus (does not bubble).

scroll:- The document view or an element has been scrolled.

keydown:- ANY key is pressed

keypress:- ANY key except Shift, Fn, CapsLock is in pressed position. (Fired continously.)

keyup:- ANY key is released

mouse enter:- A pointing device is moved onto the element that has the listener attached.

mouserleave:-A pointing device is moved off the element that has the listener attached.

mouserover:- A pointing device is moved onto the element that has the listener attached or onto one of its children.

mousemove:- A pointing device is moved over an element. (Fired continuously as the mouse moves.)

mousedown:- A pointing device button is pressed on an element.

mouseup:- A pointing device button is released over an element.

mouseout:- A pointing device is moved off the element that has the listener attached or off one of its children.

onwheel:- A wheel button of a pointing device is rotated in any direction.

onclick:- A pointing device button (ANY button; soon to be primary button only) has been pressed and released on an element.

dblclick:- A pointing device button is clicked twice on an element.

select:- Some text is being selected.

dragestart:- The user starts dragging an element or text selection.

drage:- An element or text selection is being dragged (Fired continuously every 350ms).

dragenter:- A dragged element or text selection enters a valid drop target.

dragend:- A drag operation is being ended (by releasing a mouse button or hitting the escape key).

drageover:- An element or text selection is being dragged over a valid drop target. (Fired continuously every 350ms.)

drageleave:- A dragged element or text selection leaves a valid drop target.

drop:-An element is dropped on a valid drop target.