Unit – 9 Advance JavaScript

Document Object Model (DOM)

With the HTML DOM, JavaScript can access and change all the elements of an HTML document.

With the object model, JavaScript gets all the power it needs to create dynamic HTML:

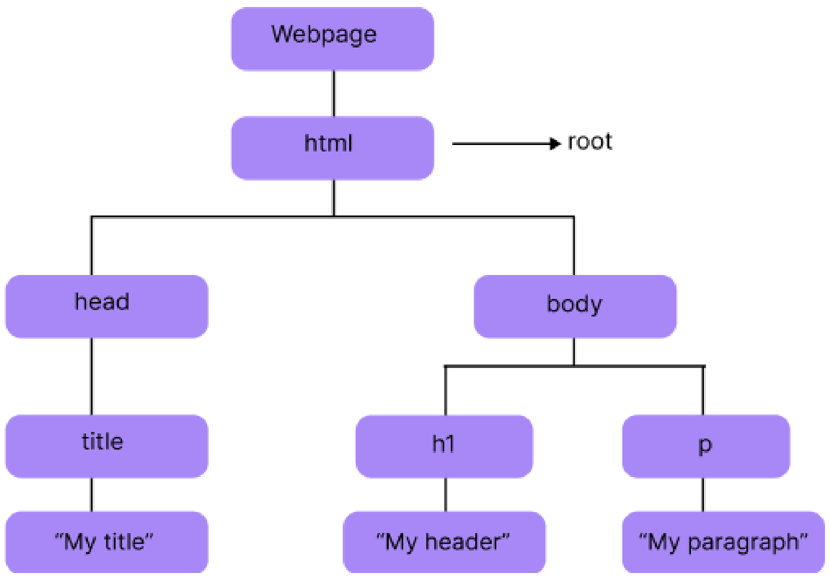
- JavaScript can change all the HTML elements in the page
- JavaScript can change all the HTML attributes in the page
- JavaScript can change all the CSS styles in the page
- JavaScript can remove existing HTML elements and attributes
- JavaScript can add new HTML elements and attributes
- JavaScript can react to all existing HTML events in the page
- JavaScript can create new HTML events in the page

What is the HTML DOM?

The HTML DOM is a standard object model and programming interface for HTML. It defines:

- The HTML elements as objects
- The properties of all HTML elements
- The methods to access all HTML elements
- The events for all HTML elements

In other words: The HTML DOM is a standard for how to get, change, add, or delete HTML elements.



DOM methods: Find HTML Element

To access HTML elements with JavaScript, one has to find the elements first. There are a couple of ways to do this:

- Finding HTML elements by id
- Finding HTML elements by tag name
- Finding HTML elements by class name

To Write in DOM – write()/writeln()

To Find in DOM - getElementById(), getElementsByClassName(), getElementsByTagName()

- ➢ HTML DOM methods are actions you can perform on HTML elements.
- ➢ HTML DOM properties are values that you can set or change of HTML elements.
- ➢ In DOM, all HTML elements are defined as objects.

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

Example to understand write and writeln

```
document.write("string")
document.write("string")
```
Output:

```
stringstring
```

```
document.writeln("string")
document.writeln("string")
```
Output:

```
string string
```

Traversing and Modifying DOM Tree: innerHTML, attribute, setting style Changing

Property	**Description**
element.innerHTML = new html content	Change the inner HTML of an element
element.attribute = new value	Change the attribute value of an HTML element
element.style.property = new style	Change the style of an HTML element
Method	**Description**
element.setAttribute(attribute, value)	Change the attribute value of an HTML element

1. The innerHTML Property : Changing HTML Content

The easiest way to get the content of an element is by using the inner HTML property. The inner HTML property is useful for getting or replacing the content of HTML elements. The inner HTML property can be used to get or change any HTML element, including <html> and <body>.

- **Finding HTML Element by Id :** The getElementById Method

The most common way to access an HTML element is to use the id of the element. In the example above the getElementById method used id="demo" to find the element.

Example:

```
<html>
<body>
<p id="demo"></p>
      //<p id="demo">hi</p>//  //hi will be replaced with "Hello World!"
<script type="text/javascript" language="javascript">
      document.getElementById("demo").innerHTML = "Hello World!";
</script>
</body>
</html>
```

Output:

```
Hello World!
```

Note: <script> should be after <p> tag, since id is defined in <p> tag. If <script> tag is before <p> tag then <p> tag will give output as it is.

2. Style Property: Changing HTML Style

To change the style of an HTML element, use this syntax:

document.getElementById(id).**style.property** = ‘style’

Example

document.getElementById(‘p1’).**style.backgroundColor = "red";**

3. Attribute Property: Changing the Value of an Attribute

To change the value of an HTML attribute, use this syntax:

document.getElementById(*id*).***attribute*** = *new value*

Example

document.getElementById(‘p1’).**title = "demo";**

- ## Finding HTML Elements by Tag Name

This example finds all <p> elements:

Example

```
var x = document.getElementsByTagName("p");
```

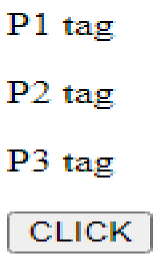
- It will return all p elements of the document in array.
- Use index or for loop to access the return elements.

```
<p></p>
<p></p>
<script>
p_element = document.getElementsByTagName("p");
   for(i=0; i<p_element.length;i++)
   {
      p_element[i].innerHTML = "Hi";  //add/modify “Hi” to all p elements
   }
<script>
                                    Or
<p></p>
<p></p>
<script>
p_element = document.getElementsByTagName("p");
 p_element[0].innerHTML = "Hi"; //add/modify “Hi” to 1st p element.
<script>
```

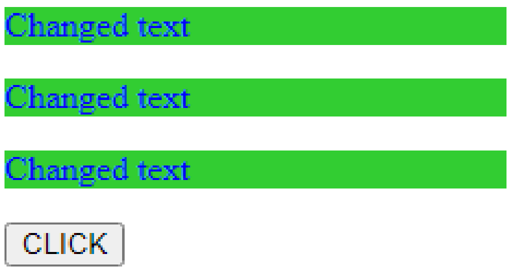
Example: Write Js to change text, color and background color on click using "tagname" method.

```
<html>
<head>
<script type="text/javascript">
    function cls()
   {
          cl = document.getElementsByTagName("p");
          for(i=0; i<cl.length;i++)
        {
          cl[i].innerHTML = "Changed text";
          cl[i].style.color = "blue";
          cl[i].style.backgroundColor = "limegreen";
          }
     }
</script></head>
<body>
          <p>P1 tag</p>
          <p>P2 tag</p>
          <p>P3 tag</p>
          <input type="button" onclick="cls();" value="CLICK"/>
          </body>
</html>
```

Output:



(after clicking "CLICK" button)



- **Finding HTML Elements by Class Name**

If you want to find all HTML elements with the same class name, use **getElementsByClassName()**.

```
var x = document.getElementsByClassName("p1");

Finding elements by class name does not work in Internet Explorer 8 and earlier versions.
```

- It will return all p elements with specified class name of the document in array.
- Use index or for loop to access the return elements.

```
<p class="p1"></p>
<p class="p1"></p>
<script>
p_element = document.getElementsByClassName("p1");
   for(i=0; i<p_element.length;i++)
   {
      p_element[i].innerHTML = "Hi";  //add/modify "Hi" to all p elements with class name
"p1"
   }
<script>
                                        Or

<p></p>
<p class="p1"></p>
<script>
p_element = document.getElementsByClassName("p1");
 p_element[0].innerHTML = "Hi"; //add/modify "Hi" to 1st p element with class name "p1".
<script>
```

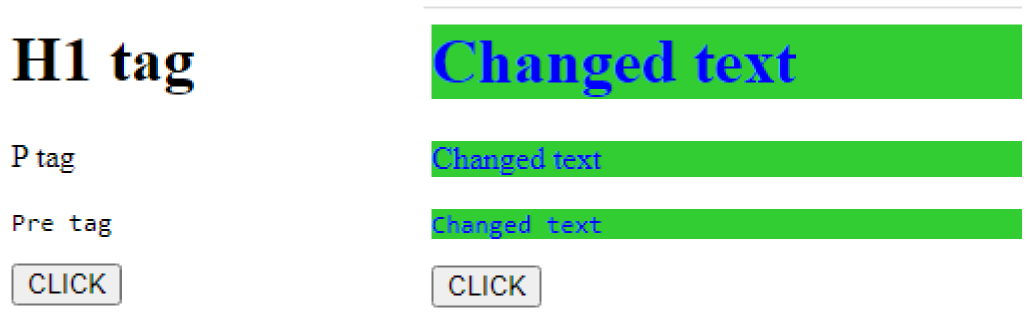
Example: Write Js to change text, color and background color on click using "ClassName" method.

```
<head>
<script type="text/javascript">
  function cls()
  {
        cl = document.getElementsByClassName("same");
        for(i=0; i<cl.length;i++)
        {
          cl[i].innerHTML = "Changed text";
          cl[i].style.color = "blue";
          cl[i].style.backgroundColor = "limegreen";
        }
   }
```

```
</script></head>
<body>
        <h1 class="same">H1 tag</h1>
        <p class="same">P tag</p>
        <pre class="same">Pre tag</pre>
        <input type="button" onclick="cls();" value="CLICK"/>
</body>
```

Output:

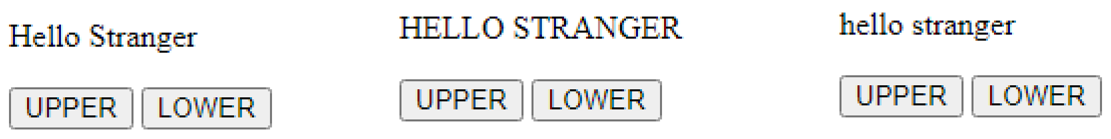
(after clicking "CLICK" button)



Example: Write JS code to Convert Uppercase and Lowercase from written text on click.

```
<html><head>
<script>
function upr()
{
cl = document.getElementsByTagName("p");
   for(i=0; i<cl.length;i++)
   {
      cl[i].style.textTransform = "upperCase";
   }
}
function lwr()
{
cl = document.getElementsByTagName("p")
   for(i=0; i<cl.length;i++)
   {
      cl[i].style.textTransform = "lowerCase";
   }
}
</script></head>
<body>
   <p>Hello Stranger</p>
   <input type="button" onclick="upr()" value="UPPER"/>
   <input type="button" onclick="lwr()" value="LOWER"/>
</body> </html>
```

Output:



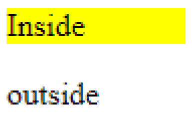
Note:

- **In ClassName and TagName, it returns value in the form of array.** So, for update in it you have to use for loop.
- while assigning css to a tag, we must use camel case for css property. For eg, fontsize should not be written like font-size like normal css. It should be written like fontSize in camelCase.

Just for the understanding.

Below example finds the element with id="main", and then finds all <p> elements inside "main":

```
<div id="main">
   <p>Inside</p>
</div>
<p>outside</p>
   <script>
   const x = document.getElementById("main");
   const y = x.getElementsByTagName("p");
   y[0].style.backgroundColor='yellow';
   </script>
```



```
Inside

outside
```

4. setAttribute(attribute, value)

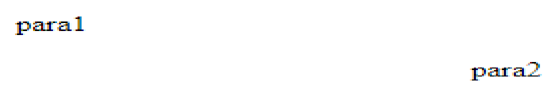
It will use to Change the attribute value of an HTML element

Example: Write JS to set attribute in id having value “i2” where text has to align right.

```
<html>
<body>
    <p id="i1">para1</p>
    <p id="i2">para2</p>
    <script>
           var i1=document.getElementById("i1")
           var i2=document.getElementById("i2")
           i1.align="center"
```

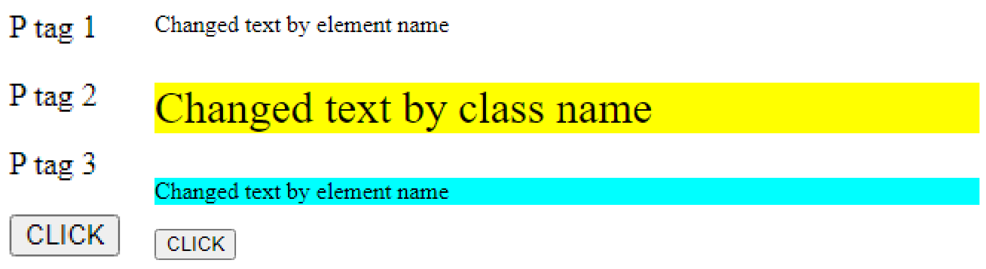
```
        i2.setAttribute("align","right")
    </script>
</body>
</html>
```

Output:


para1

para2

Example: Write a JS to perform as shown in image


P tag 1

P tag 2

P tag 3

CLICK

Changed text by element name

Changed text by class name

Changed text by element name

CLICK

```
<html><head>
<script type="text/javascript">
function demo()
   {
   p_element = document.getElementsByTagName("p");
   p_id = document.getElementById("p1");
   p_class= document.getElementsByClassName("pc")
   for(i=0; i<p_element.length;i++)
   {
      p_element[i].innerHTML = "Changed text by element name";
   }
   for(i=0; i<p_class.length;i++)
   {
      p_class[i].innerHTML = "Changed text by class name";
      p_class[i].style.backgroundColor = "yellow";
      p_class[i].style.fontSize = "30px";
   }
   p_id.style.backgroundColor = "cyan";
   }
</script>
</head>
<body>
<p>P tag 1</p>
<p class="pc">P tag 2</p>
<p id="p1">P tag 3</p>
<input type="button" onclick="demo();" value="CLICK"/>
</body></html>
```

Event Handling with JavaScript

The change in the state of an object is known as an Event. In html, there are various events which represents that some activity is performed by the user or by the browser. When JavaScript code is included in HTML, js react over these events and allow the execution. This process of reacting over the events is called Event Handling. Thus, js handles the HTML events via Event Handlers.

For example, when a user clicks over the browser, add js code, which will execute the task to be performed on the event.

- When a user clicks the mouse
- When a web page has loaded
- When an image has been loaded
- When the mouse moves over an element
- When an input field is changed
- When an HTML form is submitted
- When a user strokes a key

Mouse events

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

- **onclick**

A JavaScript can be executed when an event occurs, like when a user clicks on an HTML element.

To execute code when a user clicks on an element, add JavaScript code to an HTML event attribute:

1. onclick=*JavaScript*

```
<body>
<h1 onclick="this.innerHTML = 'Text Changed!'">Click on this text!</h1>
</body>
```

2. Using function

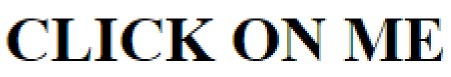
```
<h1 id="dh1" onclick="d1()">Click on this text!</h1>
<script>
   function d1(){
      document.getElementById("dh1").innerHTML="Text Changed!"
   }
</script>
```

Example: (OnClick)

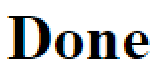
```
<html>
<head>
<script type="text/javascript">
   function change(id)
   {
        id.innerHTML = "Done";
   }
</script>
</head>
<body>
        <h1 onclick="change(this)">CLICK ON ME</h1>
</body>
</html>
```

Output:

(After clicking)



```
CLICK ON ME
```



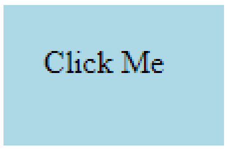
```
Done
```

- <u>**onmousedown, onmouseup**</u>

The **onmousedown, onmouseup** events are all parts of a mouse-click. First when a mouse-button is clicked, the onmousedown event is triggered, then, when the mouse-button is released, the onmouseup event is triggered,

finally, when the mouse-click is completed, the onclick event is triggered (discussed above).

```
<div onmousedown="mdown(this)" onmouseup="mup(this)"
style="background-color:lightblue;width:70px;height:30px;padding:20px;">
Click Me</div>

<script>
function mdown(id) {
  id.style.backgroundColor = "yellow";
  id.innerHTML = "click";
}

function mup(id) {
  id.style.backgroundColor="pink";
  id.innerHTML="Release";
}
</script>
```



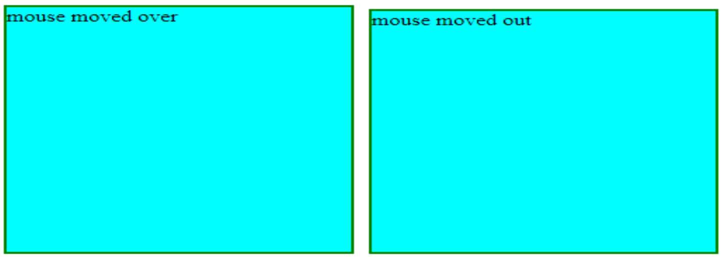
- <u>**onmouseover and onmouseout**</u>

The onmouseover and onmouseout events can be used to trigger a function when the user mouses over, or out of, an HTML element:

Example: (onmouseout, onmouseover)

```
<html>
<body>
<script type="text/javascript">
function fun(id)
{
   id.innerHTML = "mouse moved over";
}
function fun2(id)
{
   id.innerHTML = "mouse moved out";
}
</script>
<div onmouseover="fun(this)" onmouseout="fun2(this)" style="width:50%; height:40%;
background-color:cyan; border:2px solid green;">
</body>
</html>
```

Output:



```
mouse moved over
```

```
mouse moved out
```

MouseEvent button Property

- The button property returns which mouse button is pressed when a mouse event occurs.
- The button property is mostly used with the onmousedown event.
- The button property is read-only.

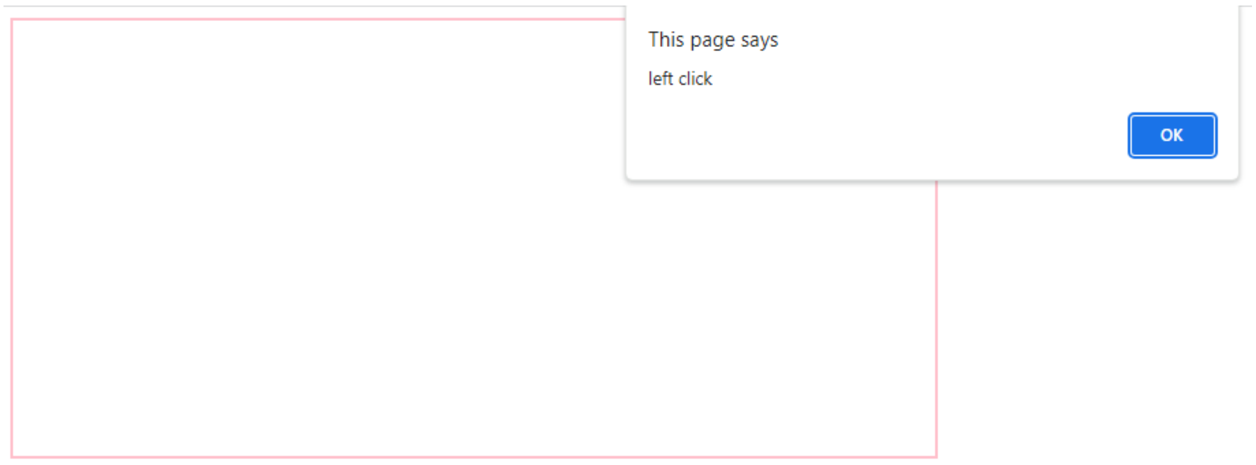
Syntax
event.button
Return Value: A Number.

Which mouse button that was pressed:
0 : Left button
1 : Wheel or middle button (if present)
2 : Right button
For a left-hand configured mouse, the values are reversed.

Example: Write a Js to check which mouse button is clicked (right, left or middle)

```
<html>
<head>
<script type="text/javascript">
function fun(e)
{
   val = e.button;
   switch(val)
   {
      case 0: alert("left click");
      break;
      case 1: alert("middle click");
      break;
      case 2: alert("right click");
      break;
     }
}
</script>
</head>
<body>
       <div style="width:50%;height:50%; border:2px solid pink;"
onmousedown="fun(event)"></div>
</body>
</html>
```

Output:



This page says

left click

Example: Write JS to handle following mouse events

1) If mouse is over heading should turn yellow, If mouse goes out then it should turn black.

2) If find time button is clicked then show date and time information.

3) If button named “red” is clicked then background color should turn red, and button named “green” is clicked then background color should turn green.

```
<html><head>
<script type="text/javascript">
function fun(id)
{
    id.style.color = "yellow";
}
function fun2(id)
{
    id.style.color = "black";
}
function fun3(id)
{
    d = new Date();
    document.getElementById("demo").innerHTML = d;
}
function fun4()
```

```
{
    id=document.getElementById("bd");
    id.bgColor = "red";
}
function fun5()
{
     id=document.getElementById("bd");
     id.bgColor = "green";
}
</script>
</head>
<body id="bd">
    <h1 onmouseover="fun(this)" onmouseout="fun2(this)">Hello</h1>
    <input type="submit" value="Find Time" onclick="fun3(this)"/>
    <p id="demo"></p>
    <input type="submit" value="red" onclick="fun4()"/>
    <input type="submit" value="green" onclick="fun5()"/>
</body></html>
```

Keyboard events

Event Handler	Description
onkeyup	when the user releases a key on the keyboard
onkeydown	when the user presses a key on the keyboard. (for ctrl, shift, function etc. keys)
onkeypress	when the user presses a key on the keyboard (for letters, numbers, symbols, when some output is there on screen)

KeyboardEvent

key Property

- The key property returns the key that was pressed when the event occured.
- The key property is read-only.

Syntax

event.key

Return Value: A String

The key that was pressed:

- A single character ("A", "a", "4", "+", "$")
- Multiple characters ("F1", "Enter", "HOME", "CAPS LOCK")

Keycode Property

Get the value of the pressed keyboard key:

The keyCode property is deprecated.

Syntax
```
event.keyCode
```

Example (onkeyup, onkeypress, onkeydown)

```
<html>
<head>
<script type="text/javascript">
function fun(id){id.bgColor="blue";}
function fun2(e){alert(e.keyCode);}
function fun3(id){id.bgColor="red";}
</script>
</head>
<body onkeyup="fun(this)" onkeypress="fun2(event)" onkeydown="fun3(this)">
</body>
</html>
```

Output:

Check output after pressing shift key once and any letter key once.

Example: Write JS to handle following key events

1) Give keycode for the key pressed

2) Script should give message "vowel is pressed" on pressing vowel key

```
<html><head>
<script type="text/javascript">
function fun2(e){
c= (e.key);
 if(c=='a' || c=='e' || c=='i' || c=='o' || c=='u' || c=='A' || c=='E' || c=='I' ||
c=='O' || c=='U') {
 document.write("vowel is entered "+e.keyCode+" "+ e.key); }
 else { document.write(e.keyCode+" "+ e.key); }
}
</script>
</head>
<body onkeypress="fun2(event)"></body></html>
```

Form events

Event Handler	Description
onblur	When the focus is away from a form element (clicks out of input field after clicking in input field)
onchange	When the user modifies or changes the value of a form element
onfocus	When the user focuses on an element (clicks in input field)
onsubmit	When the user submits the form

- **onchange**

The onchange event occurs when **the value of an HTML element is changed**.

```
<element onchange="myScript">
```

Tip: This event is similar to the oninput event. The difference is that the oninput event occurs immediately after the value of an element has changed, while onchange occurs when the element loses focus, after the content has been changed. The other difference is that the onchange event also works on <select> elements.

- **onblur**

The onblur event occurs when an **HTML element loses focus**.

```
<element onblur="myScript">
```

The onblur event is often used on input fields.
The onblur event is often used with form validation (when the user leaves a form field).

- **onfocus**

The onfocus event occurs when an element gets focus.
The onfocus event is often used on input fields.

```
<element onfocus="myScript">
```

Remember:
The onBlur event is fired when you have moved away from an object without necessarily having changed its value.
The onChange event is only called when you have changed the value of the field and it loses focus.

- **onsubmit**

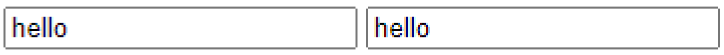
The onsubmit event occurs when a form is submitted.

```
<form action="#" onsubmit="myFunction()">
  Enter name: <input type="text" name="fname" id="fname">
  <input type="submit" value="Submit">
</form>

<script>
function myFunction() {
   a =  document.getElementById("fname");
  alert("The form was submitted! Welcome " +  a.value);
}
</script>
```

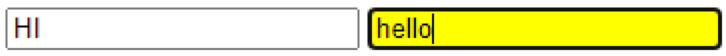
Example: (onchange, onfocus)

```
<html>
<body>
<script type="text/javascript">
    function upper(id)
    {
         id.value=id.value.toUpperCase();
    }
</script>
<!-- press enter after changing text to see uppercase  -->
    <input type="text" value="hello" onchange="upper(this)"/>
    <input type="text" value="hello" onfocus="this.style.backgroundColor='yellow'"/>
</body>
</html>
```

Output:


hello hello

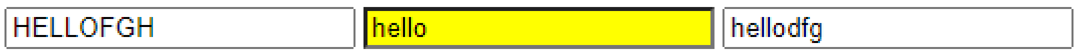
(after change and focus)


HI hello

Example:

Write HTML/Java script to perform below tasks.
1. Add three text fields.
2. In 1st field change text to uppercase only when the value of an HTML element is changed.
3. In 2nd field change field background color to yellow while the field is in focus.
4. In 3rd field change text to lowercase by loosing the focus of the field.

```
<script type="text/javascript">
function uppercase()
{
   a =  document.getElementById("fname");
   a.value = a.value.toUpperCase()
}
function lower(a)
{
a.value=a.value.toLowerCase();
}
</script>
<input type="text" value="hello" id="fname" onchange="uppercase()">
<input type="text" value="hello" onfocus="this.style.backgroundColor='yellow'"/>
<input type="text" value="HELLO" onblur="lower(this)"/>
```



Ways to access form elements

obj = document.forms[“form_name”][“element_name”].value;

or

obj = document. form_name. element_name.value;

or

obj = document.getElementByID(id).value;

Example : Event (onsubmit)

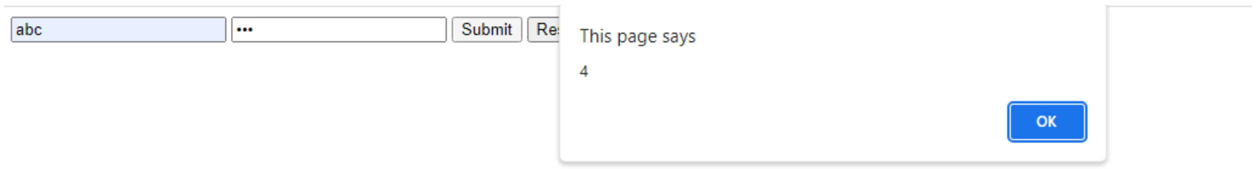
```
<html><head>
<script type="text/javascript">
function fun()
{
                    // METHOD1
    obj2 = document.f1.t1.value;
    document.write(obj2);
                // METHOD2
    // obj3 = document.forms["f1"]["t1"].value;
    // document.write(obj3);}
</script>
</head>
```

```
<body>
<form name="f1" onsubmit="fun()">
   <input type="text" name="t1"/>
   <input type="password"/>
   <input type="submit"/>
   <input type="reset"/>
</form>
</body></html>
```

Example: To find Form Length

```
<html>
<head>
<script type="text/javascript">
function fun()
{
   obj = document.forms["f1"];
   alert(obj.length);
}
</script>
</head>
<body>
<form name="f1" onsubmit="fun()">
   <input type="text" name="t1"/>
   <input type="password"/>
   <input type="submit"/>
   <input type="reset"/>
</form>
</body>
</html>
```

Output: (on clicking submit button.)



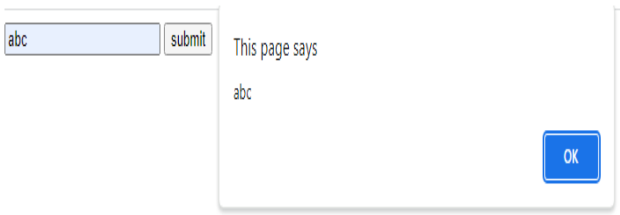
This page says
4

Example: alert will pop up with text **"**Enter Something" if input field is empty on clicking submit button.

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

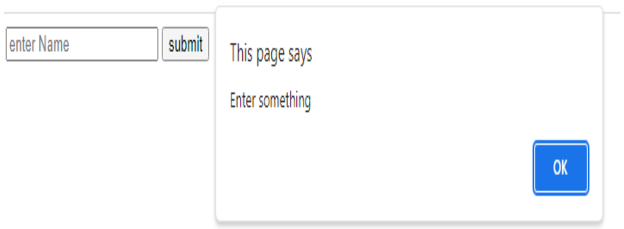
```
function fun(demo)
{
      val=demo.t1.value;
      if(val=="")
      {
           alert("Enter something");
           return false;
     }
     alert(val);
}
</script>
<form action="#" onsubmit="fun(this)">
    <input type="text" maxlength="10" placeholder="enter Name" id="t1"/>
    <input type="submit" value="submit"/>
</form></body></html>
```

Output:


abc

This page says

abc


This page says

Enter something

Example (another method of above example)

```
<html><body>
<script type="text/javascript">
function fun()
{
     val=document.getElementById("t1").value;
     if(val=="")
    {
          alert("Enter something");
          return false;
     }
     alert(val);
  }
</script>
<form action="#">
      <input type="text" maxlength="10" placeholder="enter Name" id="t1"/>
      <input type="submit" value="submit" onclick="fun()"/>
</form>
```

JavaScript Validation

It is important to validate the form submitted by the user because it can have inappropriate values. So, validation is must to authenticate user.

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

Through JavaScript, we can validate name, password, email, date, mobile numbers and more fields.

Regular Expression:

A regular expression is a pattern of characters.

Syntax of ReGex:

/pattern/modifier(s)

test() : This method is called using pattern object and returns true if string is a part of pattern.
syntax:

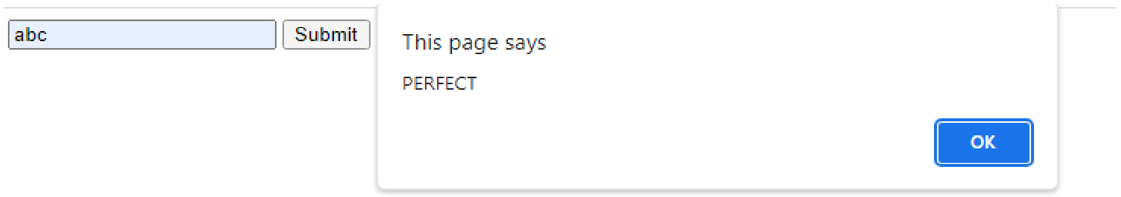
RegExpObject.test(string)

The String is Required. It is to be searched. It returns true if it finds a match, otherwise false.

Example:

```
<html>
<head>
<script type="text/javascript">
function fun()
{
   pat = /abc/;
   u = document.getElementById("t1").value;
   if(pat.test(u))
   {
        document.write("PERFECT"+" &Value is" + u);
   }
}
</script></head>
<body>
   <input type="text" id="t1"/>
   <input type="submit" onclick="fun()"/>
</body>
</html>
```

Output:



This page says

PERFECT

Notations to Create R.E.

1. Modifiers

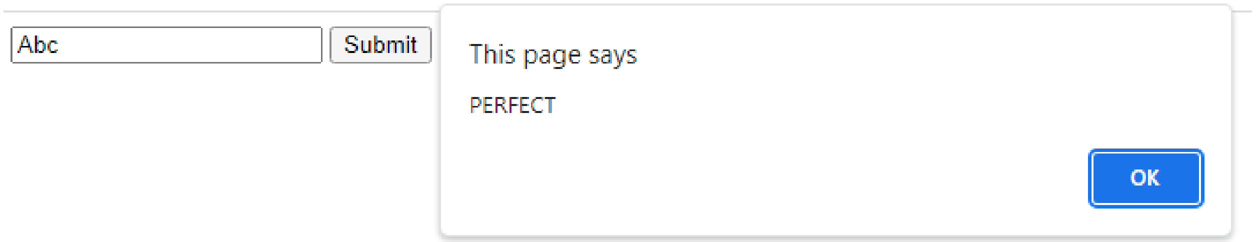
Modifiers are used to perform case-insensitive and global searches.

Modifier	Description
i	Perform case-insensitive matching

Example:

```
<html><head>
<script type="text/javascript">
function fun()
{
   pat = /abc/i;
   u = document.getElementById("t1").value;
   if(pat.test(u))
   {
       alert("PERFECT");
   }
}
</script>
</head>
<body>
   <input type="text" id="t1"/>
   <input type="submit" onclick="fun()"/>
</body></html>
```

Output:



This page says

PERFECT

2. Brackets

Brackets are used to find a range of characters

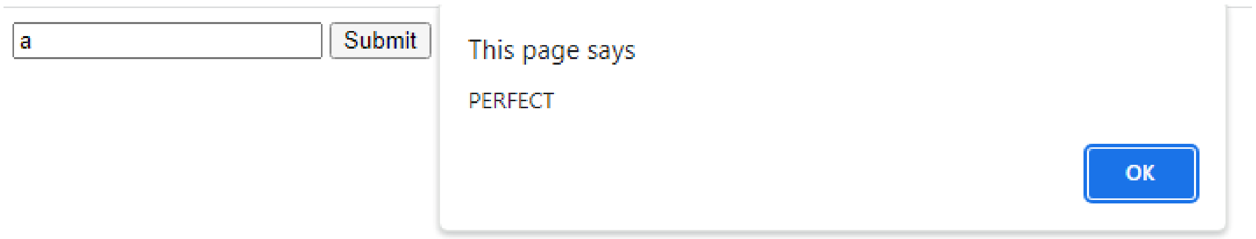
Expressions	**Description**			
[abc]	Find any of the characters inside the brackets			
	Allowed	a12	b	cdef
	Not allowed	C	def	
[^abc]	Find any character, not inside the brackets			
	Allowed	der	de123	abc45
	Not allowed	abc	a	ab
[0-9]	Find any of the digits between the brackets 0 to 9 (any digit)			
	Allowed	A5	A7	654@
	Not allowed	@	ads	
[^0-9]	Find any digit not in between the brackets (any non-digit)			
	Allowed	Ac3	abc@	12321@
	Not allowed	2131		
(x | y)	Find any of the alternatives between x or y separated with |			
	Allowed	x	xz	xyz
	Not allowed	X	abc	

Example:

```
<html><head>
<script type="text/javascript">
function fun()
{
                    // pat = /[abc]/;
                     // pat = /[^abc]/;
                    // pat = /[0-9]/;
                    // pat = /[^0-9]/;
                      pat = /(x|y)/;
    u = document.getElementById("t1").value;

   if(pat.test(u))
    {
         alert("PERFECT");
    }
}
</script></head>

<body>
    <input type="text" id="t1"/>
    <input type="submit" onclick="fun()"/>
</body></html>
```

Outputs:

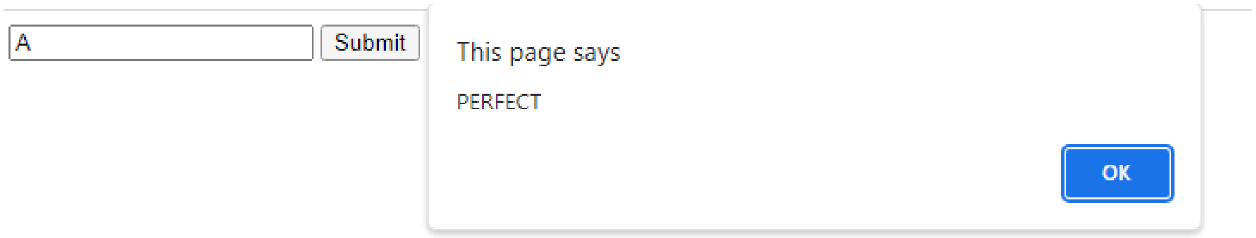
For (pat = /[abc]/;)


a

This page says

PERFECT

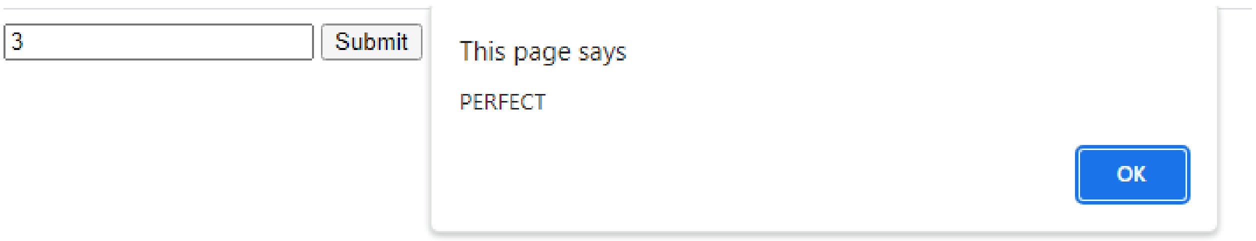
For (pat = /[^abc]/;)


A

This page says

PERFECT

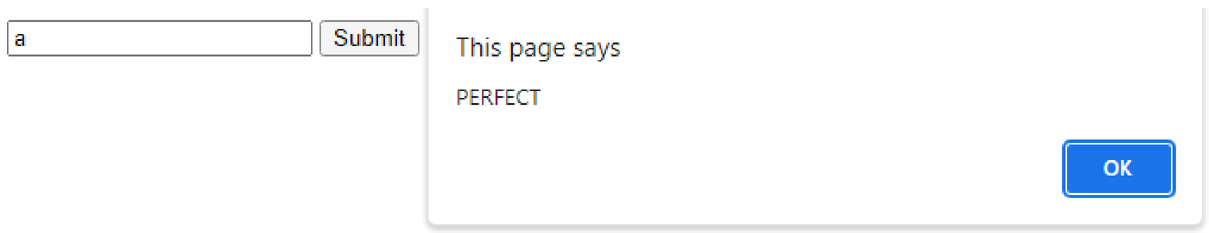
For (pat = /[0-9]/)


3

This page says

PERFECT

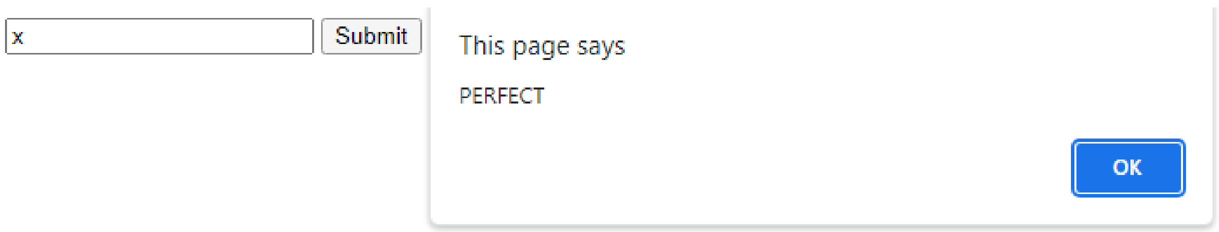
For (pat = /[^0-9]/)


a

This page says

PERFECT

For (pat = /(x|y)/)


x

This page says

PERFECT

Metacharacters

^	String must start with character For eg. /^A/			
	Allowed	Am	A	Apple
	Not allowed	conA	Banana	CAT
*****	Zero or more occurrence For eg. /bo*/			
	Allowed	b	abcd	boo
	Not allowed	xyz	oaz	
$	Matches with end of particular input For eg. /t$/			
	Allowed	Cat	Bat	eat
	Not allowed	Eater	still	
\w	Find the word character i.e. characters from a to z, A to Z, 0 to 9, _			
	Allowed	Apple	_abc	@apple
	Not allowed	@	@!	
\W	Find a non-word character			
	Allowed	@apple	@	123@
	Not allowed	0123	Apple	_abc
\d	Find a digit			
	Allowed	1234	Ab123	
	Not allowed	abcd		
\D	Search non-digit characters i.e all the characters except digits			
	Allowed	Abcd	Ab123	
	Not allowed	1234		
+	One or more occurrence (at least once occurrence)			
?	Zero or one occurrence			

For length:

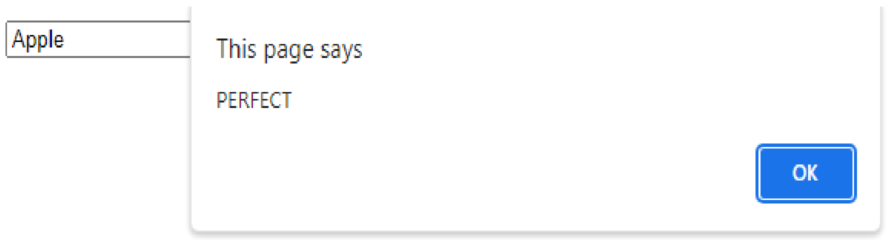
Quantifier	**Minimum**	**Maximum**
{count}	Fix count	Fix count
{min, }	min	Infinity
{min,max}	min	max

Example:

```
<html>
<head>
<script type="text/javascript">
function fun()
   {
                    // pat = /^A/;
                    // pat = /bo*/;
                    // pat = /t$/;
                    // pat = /\d/;
                   // pat = /\D/;
                   // pat = /\w/;
                      pat = /\W/;

       u = document.getElementById("t1").value;

       if(pat.test(u))
       {
              alert("PERFECT");
       }

 }
</script>
</head>

<body>
     <input type="text" id="t1"/>
     <input type="submit" onclick="fun()"/>
</body>
</html>
```

Output:

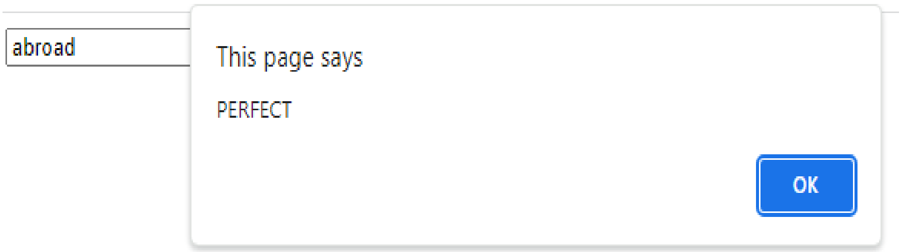
For (pat = /^A/)


Apple

This page says

PERFECT

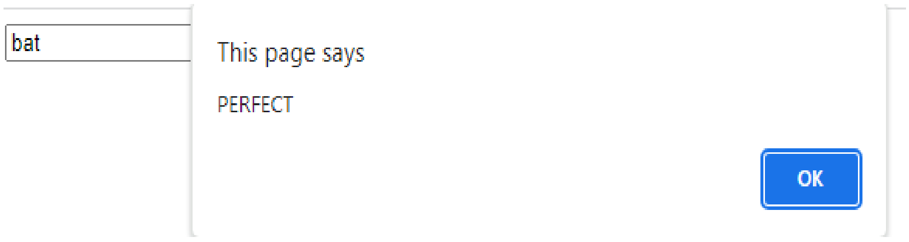
For (pat = /bo*/)


abroad

This page says

PERFECT

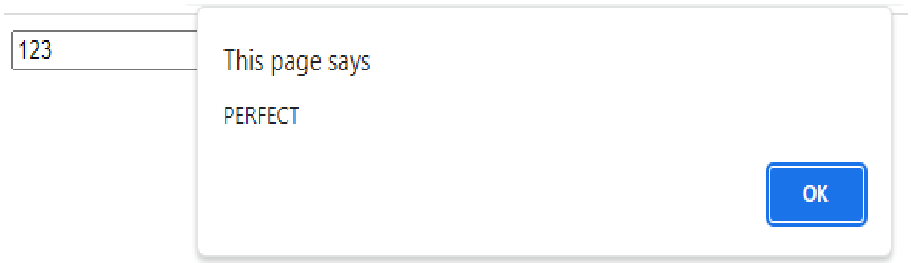
For (pat = /t$/)


bat

This page says

PERFECT

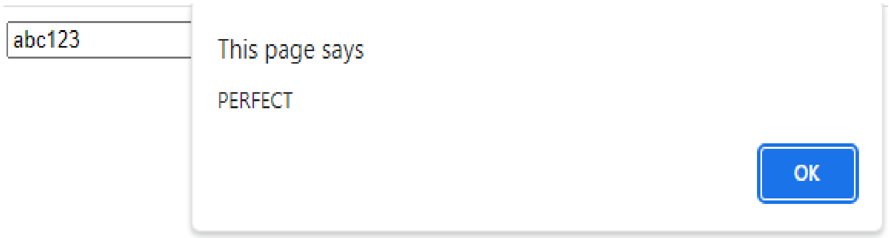
For (pat = /\d/)


123

This page says

PERFECT

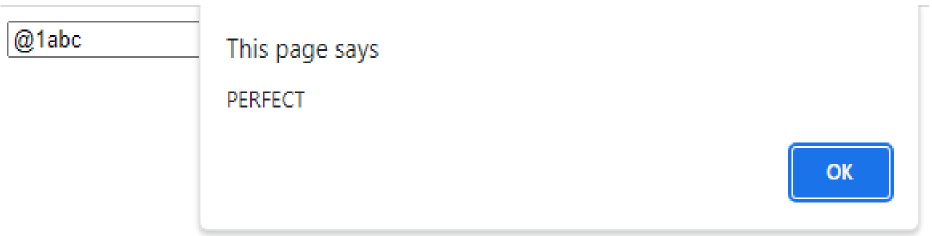
For (pat = /\D/)


abc123

This page says

PERFECT

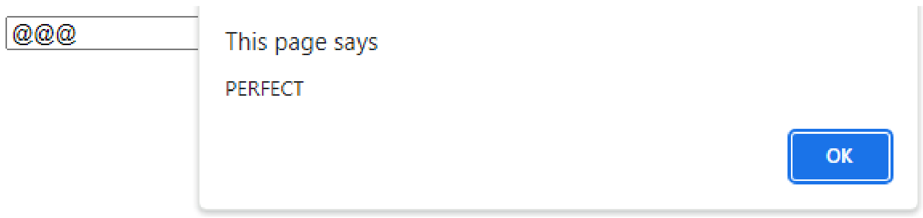
For (pat = /\w/)


@1abc

This page says

PERFECT

For (pat = /\W/)



```
@@@
```

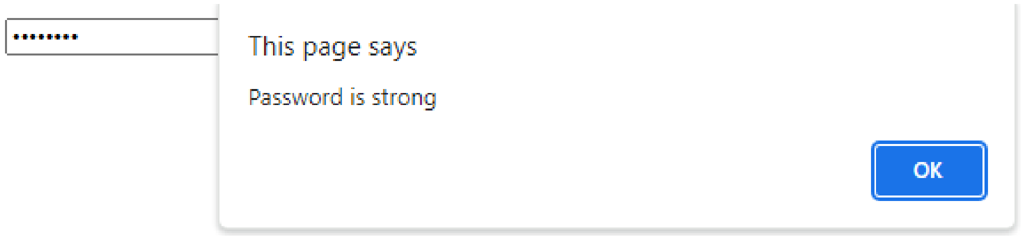
This page says

PERFECT

Example: Design a login form using JS. Following validation in password field, Minimum length of password must be of 8 letters and it must have some special characters.

```
<html>
<head>
<script type="text/javascript">
function fun()
{
   pat=/\W/;
   u = document.getElementById("p1").value;
   document.write(u.length);
   if(u.length<8)
   {
      alert("Password must be strong...length>=8");
   }
   else if(pat.test(u)==0)
   {
      alert("Password must be strong...add one sp.char");
   }
   else
   {
      alert("Password is strong");
   }
}
</script>
</head>
<body>
   <input type="password" id="p1"/>
   <input type="submit" onclick="fun()"/>
</body>
</html>
```

Output:



```
••••••••
This page says
Password is strong
```

Example: **Write a JS to validate username and password.**

Password: Length must be of 6 to 12 characters.

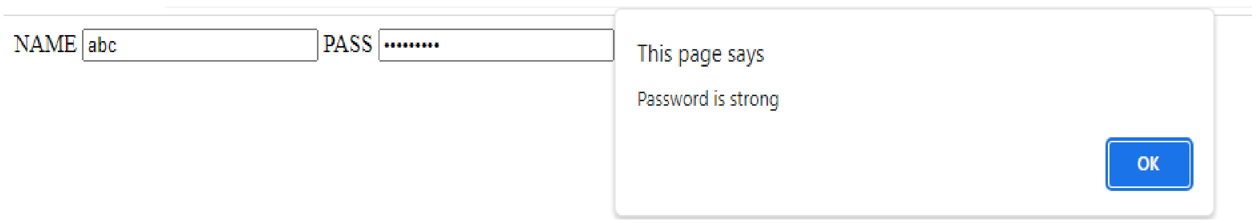
Username: Should not start with _, @ and any number.

Both must not be blank

```
<html><head><script type="text/javascript">
function fun()
{
   pat=/^_|^@|^\d/;
   u = document.getElementById("t1").value;
   v = document.getElementById("p1").value;
        //v==" " || u==" "
   if(v.length==0 || u.length==0)
   {
      alert("Both must not be blank");
   }
   else if(v.length<6 || v.length>12)
   {
      alert("Password length must be between 6 to 12");
   }
   else
   {
      alert("Password is strong");
   }
    if(pat.test(u))
   {
      alert("Enter valid username");
   }
}
```

```
</script>
</head>
<body>
   NAME <input type="text" id="t1"/>
   PASS <input type="password" id="p1"/>
   <input type="submit" onclick="fun()"/>
</body></html>
```

Output:



NAME abc PASS ••••••••

This page says

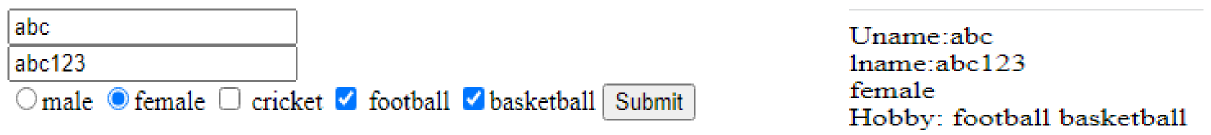
Password is strong

Example: Write a JavaScript program to display form elements value; first name, last name, gender, choose game.

```
<html> <head><script>
         function f1()
         {
             uname=document.form1.uname.value;
          lname=document.form1.lname.value;
          rad=document.getElementsByName("r1");
          for(i=0;i<rad.length;i++)
          {
            if(rad[i].checked==true)
            {
                  gender= rad[i].value;
            }
          }
          hobby=document.getElementsByName("ch1");
          var hob="";
          for(j=0;j<hobby.length;j++)
          {
            if(hobby[j].checked)
            {
                  hob = hob + " " + hobby[j].value ;
                }
          }
          document.write("Uname:"+uname);
          document.write("<br>lname:"+lname);
          document.write("<br>"+ gender);
          document.write( "<br>Hobby:" + hob );
         }
```

```
        </script>    </head>   <body>
    <form name="form1" method="get">
   <input type="text" id="uname" name="uname"/></br>
   <input type="text" id="lname" name="lname"/></br>
   <input type="radio" name="r1" value="male"/>male
   <input type="radio" name="r1" value="female"/>female
   <input type="checkbox" name="ch1" value="cricket"/> cricket
   <input type="checkbox" name="ch1" value="football"/> football
   <input type="checkbox" name="ch1" value="basketball"/>basketball
   <input type="submit" onclick="f1()">
      </form> </body></html>
```

Output:



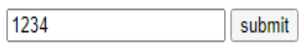
```
abc
abc123
○ male  ◉ female  ☐ cricket  ☑ football  ☑ basketball  [Submit]
```

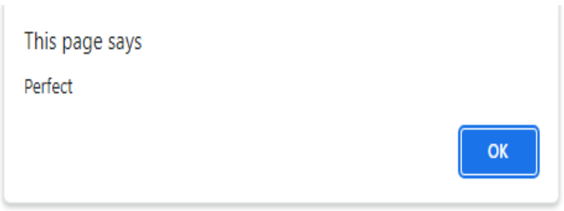
```
Uname:abc
lname:abc123
female
Hobby: football basketball
```

Example: Write HTML form accepting an integer having four digits only. Input should not accept characters of letters and special symbols.

```
<html>
<head>
   <script>
      function test(){
         var u=document.getElementById("no").value;
         if(isNaN(u))              //isNaN()  method returns true if a value is Not-a-Number.//
{
              alert("Only digits are allowed")
         }
         else if(u.length !="4")
         {
               alert("Enter four digit no")
         }
         else{
               alert("Perfect")
         }
      }
   </script>
</head>
<body>
     <input type="text" id="no">
     <input type="button" onclick="test()" value="submit">
</body>
</html>
```

Output:





This page says

Perfect

Example: Write a JS to print characters of a string at odd positions. (For eg. For the string INDIA; I, D and A should get printed.)

```
<html>
<head>
   <script>
      var str="INDIA"
      for(i=0;i<=str.length;i++){
         if(i%2!=0)
         {
            document.write(str.charAt(i-1))
         }
      }
   </script>
</head>
</html>
```

Example: Show validation using JS on fields like name, phone number and email id

RE for Username: /^[A-z]+$/;

RE for Phone number: /^\d{10}$/;

RE for email: /^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$/

1. The two forward-slashes /.../ contains a regex.
2. The leading ^ and trailing $ match the beginning and the ending of the input string, respectively. That is, the entire input string shall match with this regex, instead of a part of the input string.
3. \w+ matches 1 or more word characters (a-z, A-Z, 0-9 and underscore).
4. [.-] matches character . or -. We need to use . to represent . as . has special meaning in regex. The \ is known as the escape code, which restore the original literal meaning of the following character.
5. [.-]? matches 0 or 1 occurrence of [.-].
6. Again, \w+ matches 1 or more word characters.
7. ([.-]?\w+)* matches 0 or more occurrences of [.-]?\w+.
8. The sub-expression \w+([.-]?\w+)* is used to match the username in the email, before the @ sign. It begins with at least one word character (a-z, A-Z, 0-9 and underscore), followed by more word characters or . or -. However, a . or - must follow by a word character (a-z, A-Z, 0-9 and underscore). That is, the string cannot contain "..", "--", ".-" or "-.". Example of valid string are "a.1-2-3"
9. The @ matches itself
10. Again, the sub-expression \w+([.-]?\w+)* is used to match the email domain name, with the same pattern as the username described above.
11. The sub-expression .\w{2,3} matches a . followed by two or three word characters, e.g., ".com", ".edu", ".us", ".uk", ".co"
12. (.\w{2,3})+ specifies that the above sub-expression shall occur one or more times, e.g., ".com", ".co.uk", ".edu.sg" etc.