**DOM (Document Object Model)**

When a web page is loaded, the browser creates a **D**ocument **O**bject **M**odel of the page. Every web page resides inside a browser window which can be considered as an object.

* The **HTML DOM** model is constructed as a tree of **Objects**.
* The HTML DOM Tree of Objects



DOM is a "tree structure" representation created by the browser that enables the HTML structure to be easily accessed by programming languages — for example the browser itself uses it to apply styling and other information to the correct elements as it renders a page, and developers like you can manipulate the DOM with JavaScript after the page has been rendered.

The way a document content is accessed and modified is called the **Document Object Model**, or **DOM**. 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 **HTML DOM** is an **API** (Programming Interface) for **JavaScript**:

* JavaScript can add/change/remove HTML elements
* JavaScript can add/change/remove HTML attributes
* JavaScript can add/change/remove CSS styles
* JavaScript can react to HTML events
* JavaScript can add/change/remove HTML events

The Document object has various properties that refer to other objects which allow access to and modification of document content.

## Finding HTML Elements

**When you want to access HTML elements with JavaScript, you have 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

|  |  |
| --- | --- |
| **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. |
| getElementsByTagName() | returns all the elements having the given tag name. |
| getElementsByClassName() | returns all the elements having the given class name. |

## Example to understand write and writeln

document.write("string")

document.write("string")

**Output:**



document.writeln("string")

document.writeln("string")

**Output:**

## 

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

|  |  |
| --- | --- |
| **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 |

## 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";**

The following example changes the style of a <p> element:

## Changing HTML Content

The easiest way to modify the content of an HTML element is by using the innerHTML property.

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

document.getElementById(*id*).innerHTML =*‘new HTML’*

**Example**

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

## 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";**

## [Modifying Attributes](https://www.digitalocean.com/community/tutorials/how-to-modify-attributes-classes-and-styles-in-the-dom#modifying-attributes)

Attributes are values that contain additional information about HTML elements. They usually come in **name/value** pairs, and may be essential depending on the element.

Some of the most common HTML attributes are the src attribute of an img tag, the href of an a tag, and the class, id, and style attributes.

In JavaScript, we have four methods for modifying element attributes:

| **Method** | **Description** | **Example** |
| --- | --- | --- |
| hasAttribute() | Returns a true or false boolean | element.hasAttribute('href'); |
| getAttribute() | Returns the value of a specified attribute or null | element.getAttribute('href'); |
| setAttribute() | Adds or updates value of a specified attribute | element.setAttribute('href', 'index.html'); |
| removeAttribute() | Removes an attribute from an element | element.removeAttribute('href'); |

**Example:**

document.getElementById(‘p1’)..setAttribute("title","demo");

## Finding HTML Element by Id

* The easiest way to find an HTML element in the DOM, is by using the element id.
* This example finds the element with id="intro":

### Example

 <p id="p1"></p>

    <script>

        var e = **document.getElementById("p1")**;

        e.innerHTML = "Hello";

    </script>

* If the element is found with specified ID, the method will return the element as an **object**
* If the element is not found, then it will contain **null**.

## 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>**

**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>



## 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:**

<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>

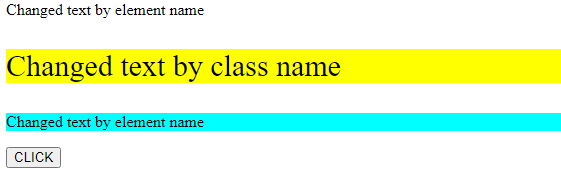
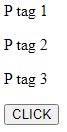
**<p class="pc">P tag 2</p>**

**<p id="p1">P tag 3</p>**

<input type="button" onclick="demo();" value="CLICK"/>

</body>

</html>

****

**Example:**

<html>

<head>

<script type="text/javascript">

function upr()

{

cl = document.getElementsByTagName("h1");

for(i=0; i<cl.length;i++)

{

cl[i].style.textTransform = "upperCase";

}

}

function lwr()

{

cl = document.getElementsByTagName("h1");

for(i=0; i<cl.length;i++)

{

cl[i].style.textTransform = "lowerCase";

}

}

</script>

</head>

<body>

<h1>Good Morning!</h1>

<input type="button" onclick="upr();" value="UPPER"/>

<input type="button" onclick="lwr();" value="LOWER"/>

</body>

</html>



**Event Handling with Javascript**

Examples of HTML events:

* 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

**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:

**onclick=*JavaScript***

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

Using function

<h1 id="dh1" onclick="d1()">Click on this text!</h1>

<script>

    function d1(){

        document.getElementById("dh1").innerHTML="Text Changed!"

    }

</script>

## onmousedown, onmouseup

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(e) {

e.style.backgroundColor = "yellow";

e.innerHTML = "click";

}

function mup(e) {

e.style.backgroundColor="pink";

e.innerHTML="Release";

}

</script>



## onmouseover and onmouseout

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

<div onmouseover="mover(this)" onmouseout="mout(this)" >

Mouse Over Me</div>

<script>

function mover(e) {

e.innerHTML = "Thank You";

e.style.color="red"

}

function mout(e) {

e.innerHTML = "Mouse Over Me"

e.style.color="blue"

}

</script>

## Form events:

**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](https://www.w3schools.com/jsref/event_oninput.asp) 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:**

<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)"/>**



**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>

## Keyboard Events

|  |  |
| --- | --- |
| **Event** | **Occurs When** |
| [onkeydown](https://www.w3schools.com/jsref/event_onkeydown.asp) | A user presses a key |
| [onkeypress](https://www.w3schools.com/jsref/event_onkeypress.asp) | A user presses a key |
| [onkeyup](https://www.w3schools.com/jsref/event_onkeyup.asp) | A user releases a key |

**Example:**

<head>

<script type="text/javascript">

function fun1(id)

{

id.bgColor="blue";

}

function fun2(id)

{

id.bgColor="red";

}

</script>

</head>

<body onkeyup="fun1(this)"  onkeydown="fun2(this)">

</body>

**Write script to display Unicode on key press.**

<script >

function fun(e)

{ alert(e.keyCode); }

</script>

<body onkeypress="fun(event)" ></body>

**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>