HTML DOM

DOM stands for Document Object Model. When a browser loads a web page, the browser creates a Document Object Model of that page. The HTML DOM is created as a tree of Objects.

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

HTML DOM methods are **actions** you can perform (on HTML Elements).

HTML DOM properties are **values** (of HTML Elements) that you can set or change.

A **property** is a value that you can get or set (like changing the content of an HTML element).

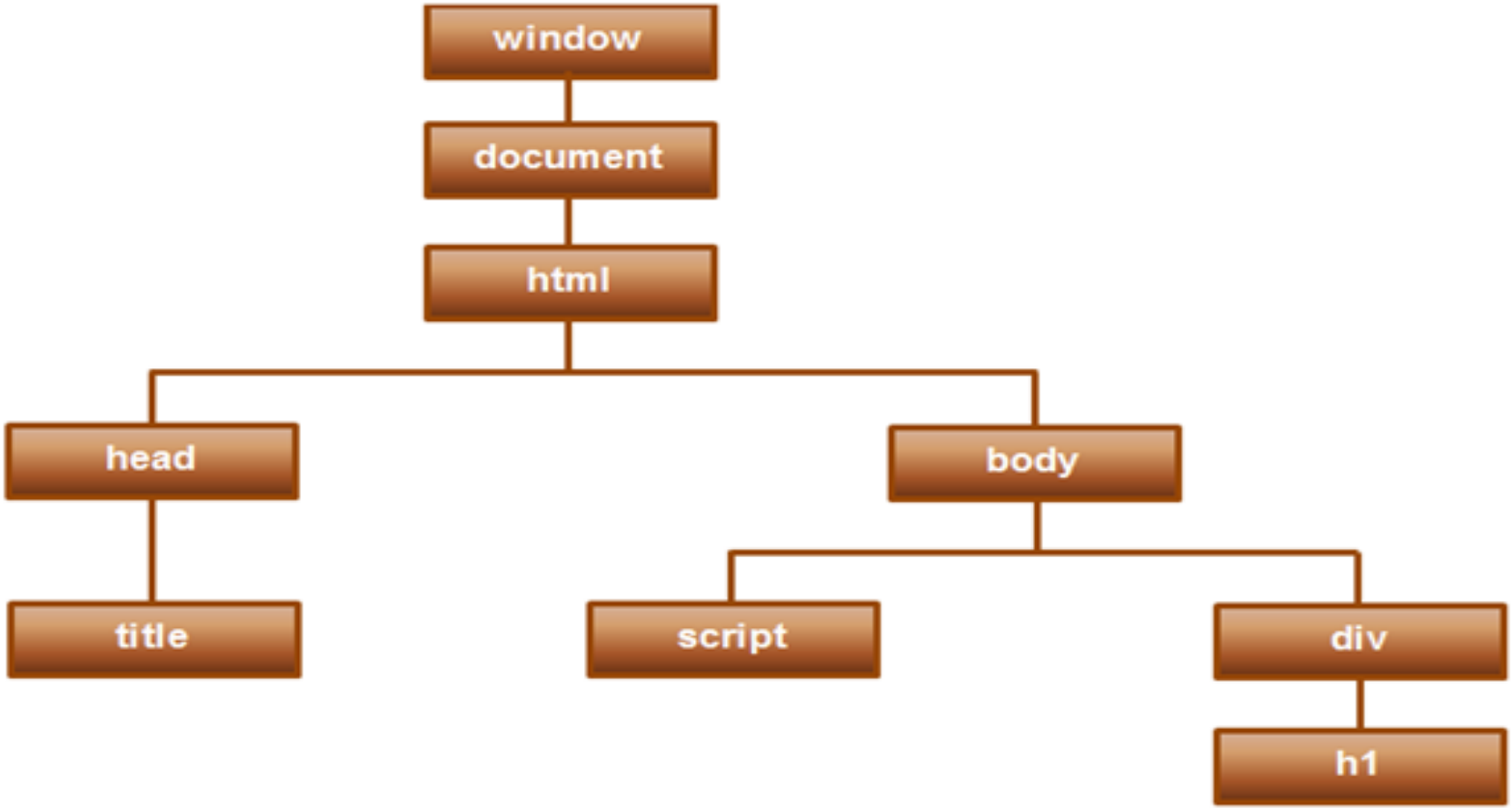
A **method** is an action you can do (like add or deleting an HTML element).

HTML DOM

Example:

```
<html>
   <head>
      <title>My Page Title</title>
   </head>
   <body>
      <script type="text/javascript">
      </script>
      <div>
         <h1>This is browser DOM</h1>
      </div>
   </body>
</html>
```

For this HTML a **graphical representation of the Document Object Model** is shown below.



The following example changes the content (the innerHTML) of the <p> element with id="demo":

```
<!DOCTYPE html>
<html>
<body>
<h2>My First Page</h2>
<p id="demo"></p>
<script>
document.getElementById("demo").innerHTML = "Hello World!";
</script>
</body>
</html>
```

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.

The innerHTML Property

The easiest way to get the content of an element is by using the **innerHTML** property.

The innerHTML property is useful for getting or replacing the content of HTML elements.

Finding HTML Elements

Method Description :

a. document.getElementById(*id*) Find an element by element id
b. document.getElementsByTagName(*name*) Find elements by tag name
c. document.getElementsByClassName(*name*) Find elements by class name

Changing HTML Elements

Method 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.setAttribute*(attribute, value)* :Change the attribute value of an HTML element

element.style.*property = new style* : Change the style of an HTML element

The **`document`** object

- Many attributes of the current document are available via the **`document`** object:

`Title`	**`Referrer`**
`URL`	**`Images`**
`Forms`	**`Links`**
`Colors`	

`document` Methods

- **`document.write()`** like a print statement – the output goes into the HTML document.
- **`document.writeln()`** adds a newline after printing.

```
document.write("My title is" +
  document.title);
```

The **`window`** Object

- Represents the current window.

- There are possible many objects of type **`Window`**, the predefined object **`window`** represents the current window.
- Access to, and control of, a number of properties including position and size.

window attributes

- **document**
- **name**
- **status** the status line
- **parent**

some **window** methods

```
alert()
close()
prompt()
moveTo()     moveBy()
open()
scroll()     scrollTo()
resizeBy()   resizeTo()
```

Adding and Deleting Elements

Method	Description
document.createElement(*element*)	Create an HTML element
document.removeChild(*element*)	Remove an HTML element
document.appendChild(*element*)	Add an HTML element
document.replaceChild(*element*)	Replace an HTML element
document.write(*text*)	Write into the HTML output stream

Adding Events Handlers

Method	Description
document.getElementById(*id*).onclick = function(){*code*}	Adding event handler code to an onclick event

Finding HTML Objects

document.baseURI	Returns the absolute base URI of the document
document.body	Returns the <body> element
document.cookie	Returns the document's cookie
document.doctype	Returns the document's doctype
document.documentElement	Returns the <html> element

Finding HTML Elements by Class Name

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

<p>Hello World!</p>

<p class="intro">The DOM is very useful.</p>
<p class="intro">This example demonstrates the <b>getElementsByClassName</b>
method.</p>

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

<script>
var x = document.getElementsByClassName("intro");
document.getElementById("demo").innerHTML =
'The first paragraph (index 0) with class="intro": ' + x[0].innerHTML;
</script>

</body>
</html>
```

If you want to find all HTML elements that matches a specified CSS selector (id, class names, types, attributes, values of attributes, etc), use the querySelectorAll() method.

This example returns a list of all <p> elements with class="intro".

```
<!DOCTYPE html>
<html>
<body>
<p>Hello World!</p>
<p class="intro">The DOM is very useful.</p>
<p class="intro">This example demonstrates the <b>querySelectorAll</b> method.</p>
<p id="demo"></p>

<script>
var x = document.querySelectorAll("p.intro");
document.getElementById("demo").innerHTML =
'The first paragraph (index 0) with class="intro": ' + x[0].innerHTML;
</script>
</body>
</html>
```

Changing the Value of an Attribute

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

<img id="image" src="smiley.gif" width="160" height="120">

<script>
document.getElementById("image").src = "landscape.jpg";
</script>

<p>The original image was smiley.gif, but the script changed it to landscape.jpg</p>

</body>
</html>
```

Changing HTML Style

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

<p id="p1">Hello World!</p>
<p id="p2">Hello World!</p>

<script>
document.getElementById("p2").style.color = "blue";
document.getElementById("p2").style.fontFamily = "Arial";
document.getElementById("p2").style.fontSize = "larger";
</script>

<p>The paragraph above was changed by a script.</p>

</body>
</html>
```

JavaScript can be used to access and modify these DOM objects and their properties. For example, you can add, modify and remove HTML elements and their attributes. Along the same lines, you can use DOM object properties to assign event handlers to events.

Assigning event handlers in JavaScript using DOM object property

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>
<input type="button" value="Click me" id="btn"/>
<script type="text/javascript">
   document.getElementById("btn").onmouseover = changeColorOnMouseOver;
   document.getElementById("btn").onmouseout = changeColorOnMouseOut;

   function changeColorOnMouseOver()
   {
      this.style.background = 'red';
      this.style.color = 'yellow';
   }
```

```
function changeColorOnMouseOut()
   {
      this.style.background = 'black';
      this.style.color = 'white';
   }
</script>
</body>
```

In this case, we are assigning event handlers using the DOM object properties (**onmouseover** & **onmouseout**) instead of using the attributes of the HTML tag. We are using this keyword to reference the current HTML element. In this example "**this**" references the button control.

The following example is same as the last one. In this case we are assigning an anonymous function to **onmouseover** & **onmouseout** properties.

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>
<input type="button" value="Click me" id="btn" />
<script type="text/javascript">
   document.getElementById("btn").onmouseover = function ()
   {
      this.style.background = 'red';
      this.style.color = 'yellow';
   }
   document.getElementById("btn").onmouseout = function ()
   {
      this.style.background = 'black';
      this.style.color = 'white';
   }
</script>
</body>
</html>
```

If an **event handler** is assigned using both, i.e an HTML attribute and DOM object property, the handler that is assigned using the DOM object property overwrites the one assigned using HTML attribute. Here is an example.

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>
<input type="button" value="Click me" id="btn" onclick="clickHandler1()"/>
<script type="text/javascript">
   document.getElementById("btn").onclick = clickHandler2;

   function clickHandler1()
   {
      alert("Handler set using HTML attribute");
   }

   function clickHandler2()
   {
      alert("Handler set using DOM object property");
   }
</script>
</body>
```

Using this approach you can only assign one event handler method to a given event. The handler that is assigned last wins. In the following example, **Handler2**() is assigned after **Handler1**. So **Handler2**() owerites **Handler1**().

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>
<input type="button" value="Click me" id="btn"/>
<script type="text/javascript">
   document.getElementById("btn").onclick = clickHandler1;
   document.getElementById("btn").onclick = clickHandler2;

   function clickHandler1()
   {
      alert("Handler 1");
   }

   function clickHandler2()
   {
      alert("Handler 2");
   }
</script>
</body>
```

addeventlistener and removeeventlistener in JavaScript:

In this topic we will discuss assigning event handlers in JavaScript using the following special methods

addEventListener

removeEventListener

attachEvent

detachEvent

Sytnax for assigning event handler using addEventListener() method

element.addEventListener(event, handler, phase)

Sytnax for removing event handler using removeEventListener() method

element.removeEventListener(event, handler, phase)

Please note : The third parameter phase is usually set to false as it is not used.

Example : In this example, we are passing values for all the 3 parameters including phase.

```
<!DOCTYPE html>
<html>
 <head>
 <title>Event Handler Example</title>
 </head>
<body>
<input type="button" value="Click me" id="btn"/>
<script type="text/javascript">
   btn.addEventListener("mouseover", changeColorOnMouseOver, false);
   btn.addEventListener("mouseout", changeColorOnMouseOut, false);

   function changeColorOnMouseOver()
   {
      this.style.background = 'red';
      this.style.color = 'yellow';
   }

   function changeColorOnMouseOut()
   {
      this.style.background = 'black';
      this.style.color = 'white';
   }
</script>
</body>
 </html>
```

Since the third parameter "phase" is optional you can omit it if you wish.

```
btn.addEventListener("mouseover", changeColorOnMouseOver);
btn.addEventListener("mouseout", changeColorOnMouseOut);
```

Example : This example demonstrates removing event handlers.

```
<input type="button" value="Click me" id="btn"/>
<input type="button" value="Remove Event Handlers"
        onclick="removeEventHandlers()" />
<script type="text/javascript">
   btn.addEventListener("mouseover", changeColorOnMouseOver);
   btn.addEventListener("mouseout", changeColorOnMouseOut);


   function changeColorOnMouseOver()
   {
      this.style.background = 'red';
      this.style.color = 'yellow';
   }
```

```
 function changeColorOnMouseOut()
   {
      this.style.background = 'black';
      this.style.color = 'white';
   }


   function removeEventHandlers()
   {
      btn.removeEventListener("mouseover", changeColorOnMouseOver);
      btn.removeEventListener("mouseout", changeColorOnMouseOut);
   }
</script>
```

Using this approach you can assign more than one event handler method to a given event. The order in which handler methods are executed is not defined. In this example, 2 event handler methods (clickHandler1 & clickHandler2) are assigned to click event of the button control. When you click the button both the handler methods are executed.

```
<!DOCTYPE html>
<html>
 <head>
 <title>Event Handler Examples</title>
 </head>
<body>
<input type="button" value="Click me" id="btn"/>
<script type="text/javascript">
    btn.addEventListener("click", clickHandler1);
    btn.addEventListener("click", clickHandler2);

    function clickHandler1()
    {
        alert("Handler 1");
    }

    function clickHandler2()
    {
        alert("Handler 2");
    }
</script>
</body>
 </html>
```

attachEvent() and detachEvent() methods only work in Internet Explorer 8 and earlier versions.

Sytnax for assigning event handler using attachEvent() method

element.attachEvent("on"+event, handler)

Sytnax for removing event handler using detachEvent() method

element.detachEvent("on"+event, handler)

Example : This example will only work in Internet Explorer 8 and earlier versions.

```
<input type="button" value="Click me" id="btn"/>
<script type="text/javascript">
   btn.attachEvent("onclick", clickEventHandler);


   function clickEventHandler()
   {
      alert("Click Handler");
   }
</script>
```

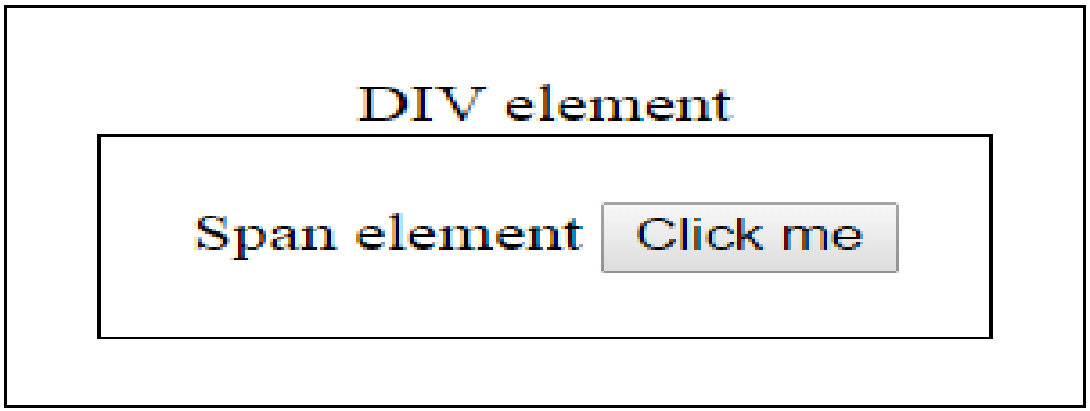
Cross browser solution : For the above example to work in all browsers, modify the script as shown below.

```
<input type="button" value="Click me" id="btn"/>
<script type="text/javascript">


   if (btn.addEventListener)
   {
      btn.addEventListener("click", clickEventHandler);
   }
   else
   {
      btn.attachEvent("onclick", clickEventHandler);
   }


   function clickEventHandler()
   {
      alert("Click Handler");
   }
</script>
```

Event bubbling in JavaScript

What is event bubbling

Let us understand this with an example. HTML elements can be nested inside each other. For example a button element can be nested inside a span element and the span element in turn can be nested inside a div element as shown below.



onclick attribute will be specified for all the 3 elements.

```
<!DOCTYPE html>
<html>
<head>
   <style type="text/css">
      .abc
      {
         display: table-cell;
         border: 1px solid black;
         padding: 20px;
         text-align: center;
      }
   </style>
</head>
<body>
   <div class="abc" onclick="alert('Div click handler')">
      DIV element
      <span class="abc" onclick="alert('Span click handler')">
      Span element
         <input type="button" value="Click me" onclick="alert('Button click handler')" />
      </span>
   </div>
</body>
</html>
```

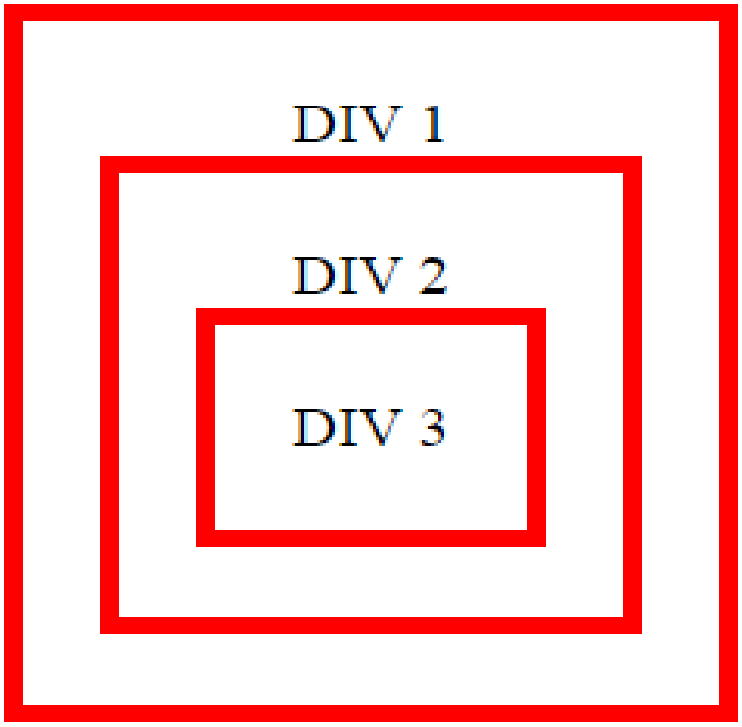
A click on the button, causes a click event to be fired on the button. The button click event handler method handles the event. The click event then bubbles up to the button element parent (span element), which is handled by the span element event handler method. The click event then bubbles up to the span element parent (div element). This is called event bubbling.

Notice that if you click on the element, it's event handler and it's parent( ) element event handler are called. If you click on the element, just the element event handler method is called. So, the event bubbling process starts with the element that triggered the event and then bubbles up to the containing elements in the hierarchy.

The following example is similar to the one above, except we removed the onclick attribute from button and span elements. Because of event bubbling, when you click on the button or the span element, the event gets handled by the div element handler.

```
!DOCTYPE html>
<html>
<head>
   <style type="text/css">
      .abc
      {
         display: table-cell;
         border: 1px solid black;
         padding: 20px;
         text-align: center;
      }
   </style>
</head>
<body>
   <div class="abc"
    onclick="alert('Click event handled by DIV element')">DIV element
      <span class="abc">Span element
         <input type="button" value="Click me"/>
      </span>
   </div>
</body>
</html>
```

When the event gets bubbled, the keyword this references the current element to which the event is bubbled. In the example below, we are using "this" keyword to reference the current div element and change it's border color. When you click on the innermost element, all the 3 elements border get changed due to event bubbling.



```
<!DOCTYPE html>
<html>
<head>
   <style type="text/css">
      .divStyle
      {
         display: table-cell;
         border: 5px solid black;
         padding: 20px;
         text-align: center;
      }
   </style>
</head>
<body>
   <div id="DIV1" class="divStyle">
      DIV 1
      <div id="DIV2" class="divStyle">
         DIV 2
         <div id="DIV3" class="divStyle">
            DIV 3
         </div>
      </div>
   </div>
```

```
<script type="text/javascript">
      var divElements = document.getElementsByTagName('div');

      for (var i = 0; i < divElements.length; i++)
      {
         divElements[i].onclick = function ()
         {
            this.style.borderColor = "red";
            alert(this.getAttribute("id") + " backgound changed");
         }
      }
   </script>
</body>
</html>
```

Stopping event bubbling : If you don't want the event to be bubbled up, you can stop it.

With Internet Explorer 8 and earlier versions

event.cancelBubble = true

With Internet Explorer 9 (and later versions) & all the other browsers

event.stopPropagation();

In the example below we have stopped event bubbling. So, when you click on DIV 3, only DIV 3 border colour is changed.

```
<!DOCTYPE html>
<html>
<head>
   <style type="text/css">
      .divStyle
      {
         display: table-cell;
         border: 5px solid black;
         padding: 20px;
         text-align: center;
      }
   </style>
</head>
```

```
<body>
   <div id="DIV1" class="divStyle">
      DIV 1
      <div id="DIV2" class="divStyle">
         DIV 2
         <div id="DIV3" class="divStyle">
            DIV 3
         </div>
      </div>
   </div>
   <script type="text/javascript">
      var divElements = document.getElementsByTagName('div');

      for (var i = 0; i < divElements.length; i++)
      {
         divElements[i].onclick = function (event)
         {
                                        alert(event.type);
            event = event || window.event;
```

```
 if (event.stopPropagation)
            {
               event.stopPropagation();
            }
  else
            {
               event.cancelBubble = true;
            }

            this.style.borderColor = "red";
            alert(this.getAttribute("id") + " backgound changed");
         }
      }
   </script>
</body>
</html>
```

Image gallery with thumbnails in JavaScript

The **image gallery** should be as shown in the image below. When you click on the image thumbnail, the respective image should be displayed in the main section of the page.

```
<!DOCTYPE html>
<html>
<head>
   <style type="text/css">
      .imgStyle
      {
         width:100px;
         height:100px;
         border:3px solid grey;
      }
   </style>
</head>
<body>
   <img id="mainImage" style="border:3px solid grey"
       src="Images/1.jpg" height="500px" width="540x"/>
   <br />
   <div id="divId" onclick="changeImageOnClick(event)">
      <img class="imgStyle" src="Images/1.jpg" />
      <img class="imgStyle" src="Images/2.jpg" />
      <img class="imgStyle" src="Images/3.jpg" />
      <img class="imgStyle" src="Images/4.jpg" />
      <img class="imgStyle" src="Images/5.jpg" />
   </div>
```

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

    var images = document.getElementById("divId") .getElementsByTagName("img");

    for (var i = 0; i < images.length; i++)
    {
       images[i].onmouseover = function ()
       {
          this.style.cursor = 'hand';
          this.style.borderColor = 'red';
       }
       images[i].onmouseout = function ()
       {
          this.style.cursor = 'pointer';
          this.style.borderColor = 'grey';
       }
    }
```

```
 function changeImageOnClick(event)
      {

         event = event || window.event;
         var targetElement = event.target || event.srcElement;
document.getElementById("mainImage").src = targetElement.getAttribute("src");
      }
   </script>
</body>
</html>
```

```
<!DOCTYPE html>
<html>
<head>
   <style type="text/css">
      .imgStyle
      {
         width:100px;
         height:100px;
         border:3px solid grey;
      }
   </style>
</head>
<body>
   <img id="mainImage" style="border:3px solid grey"
       src="Images/1.jpg" height="500px" width="540x"/> <br />
   <div id="divId" onclick="changeImageOnClick(event)">
      <img class="imgStyle" src="Images/1.jpg" />
      <img class="imgStyle" src="Images/2.jpg" />
      <img class="imgStyle" src="Images/3.jpg" />
      <img class="imgStyle" src="Images/4.jpg" />
      <img class="imgStyle" src="Images/5.jpg" />
   </div>
```

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

    var images = document.getElementById("divId").getElementsByTagName("img");

    for (var i = 0; i < images.length; i++)
    {
       images[i].onmouseover = function ()
       {
          this.style.cursor = 'hand';
          this.style.borderColor = 'red';
       }
       images[i].onmouseout = function ()
       {
          this.style.cursor = 'pointer';
          this.style.borderColor = 'grey';
       }
    }
```

```
function changeImageOnClick(event)
      {
         event = event || window.event;
         var targetElement = event.target || event.srcElement;

         if (targetElement.tagName == "IMG")
         {
            mainImage.src = targetElement.getAttribute("src");
         }
      }
   </script>
</body>
</html>
```