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

Read about the <a id="myAnchor" href="dom_obj_attributes.asp"
target="_blank">Attr object</a>.

<p>Click the button to display the value of the target attribute of the link
above.</p>

<button onclick="myFunction()">Try it</button>

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

<script>
function myFunction() {
  var x = document.getElementById("myAnchor").getAttribute("target");
  document.getElementById("demo").innerHTML = x;
}
</script>

</body>
</html>
```

```html
<!DOCTYPE html>
<html>
<head>
<style>
.democlass {
  color: red;
}
</style>
</head>
<body>

<h1>Hello World</h1>

<p>Click the button to add a class attribute with the value of "democlass" to the h1 element.</p>

<button onclick="myFunction()">Try it</button>

<script>
function myFunction() {
  document.getElementsByTagName("H1")[0].setAttribute("class", "democlass");
}
</script>

</body>
</html>
```

```
<body>

<p>Example list:</p>
<ul id="myList"><li>Coffee</li><li>Tea</li></ul>

<p>Click the button to get the HTML content of the list's first child node.
</p>

<button onclick="myFunction()">Try it</button>

<p><strong>Note:</strong> Whitespace inside elements is considered as text,
and text is considered as nodes.</p>

<p>If you add whitespace before the first LI element, the result will be
"undefined".</p>

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

<script>
function myFunction() {
  var list = document.getElementById("myList").firstChild.innerHTML;
  document.getElementById("demo").innerHTML = list;
}
</script>

</body>
</html>
```

HTML DOM firstChild Property

Element Object

Example

Get the HTML content of the first child node of an <ul> element:

```
var x = document.getElementById("myList").firstChild.innerHTML;
```

HTML DOM focus() Method

‹ Element Object

Example

Give focus to an <a> element:

```
document.getElementById("myAnchor").focus();
```

Try it Yourself »

HTML DOM removeChild() Method

Element Object

Example

Remove the first <li> element from a list:

```
var list = document.getElementById("myList");   // Get the <ul> element with id="myList"
list.removeChild(list.childNodes[0]);           // Remove <ul>'s first child node (index 0)
```

Before removing:

- Coffee
- Tea
- Milk

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

<!-- Note that the <li> elements inside <ul> are not indented (whitespaces).
If they were, the first child node of <ul> would be a text node
-->
<ul id="myList"><li>Coffee</li><li>Tea</li><li>Milk</li></ul>

<p>Click the button to remove the first item from the list.</p>

<button onclick="myFunction()">Try it</button>

<script>
function myFunction() {
  var list = document.getElementById("myList");
  list.removeChild(list.childNodes[0]);
}
</script>

</body>
</html>
```

HTML DOM removeAttribute() Method

‹ Element Object

Example

Remove the class attribute from an <h1> element:

```
document.getElementsByTagName("H1")[0].removeAttribute("class");
```

Try it Yourself »

```
<!DOCTYPE html>
<html>
<head>
<style>
.democlass {
  color: red;
}
</style>
</head>
<body>

<h1 class="democlass">Hello World</h1>

<p id="demo">Click the button to remove the class attribute from the h1
element.</p>

<button onclick="myFunction()">Try it</button>

<script>
function myFunction() {
  document.getElementsByTagName("H1")[0].removeAttribute("class");
}
</script>

</body>
</html>
```

HTML DOM namespaceURI Property

Element Object

Example

Get the URI of the namespace for an XHTML document:

```
var x = document.documentElement.namespaceURI;
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
<body>

<p >Click the button get the URI of the document's namespace.</p>

<button onclick="myFunction()">Try it</button>

<p><strong>Note:</strong> Internet Explorer 8 and earlier does not support the
namespaceURI property.</p>

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

<script>
function myFunction() {
  var x = document.documentElement.namespaceURI;
  document.getElementById("demo").innerHTML = x;
}
</script>

</body>
</html>
```

HTML DOM addEventListener() Method

‹ Element Object

Example

Attach a click event to a <button> element. When the user clicks on the button, output "Hello World" in a <p> element with id="demo":

```
document.getElementById("myBtn").addEventListener("click", function(){
  document.getElementById("demo").innerHTML = "Hello World";
});
```

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

<p>This example uses the addEventListener() method to attach a click event to a
button.</p>

<button id="myBtn">Try it</button>

<p><strong>Note:</strong> The addEventListener() method is not supported in
Internet Explorer 8 and earlier versions.</p>

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

<script>
document.getElementById("myBtn").addEventListener("click", function(){
  document.getElementById("demo").innerHTML = "Hello World";
});
</script>

</body>
</html>
```

JavaScript - Document Object Model or DOM

Previous Page Next Page

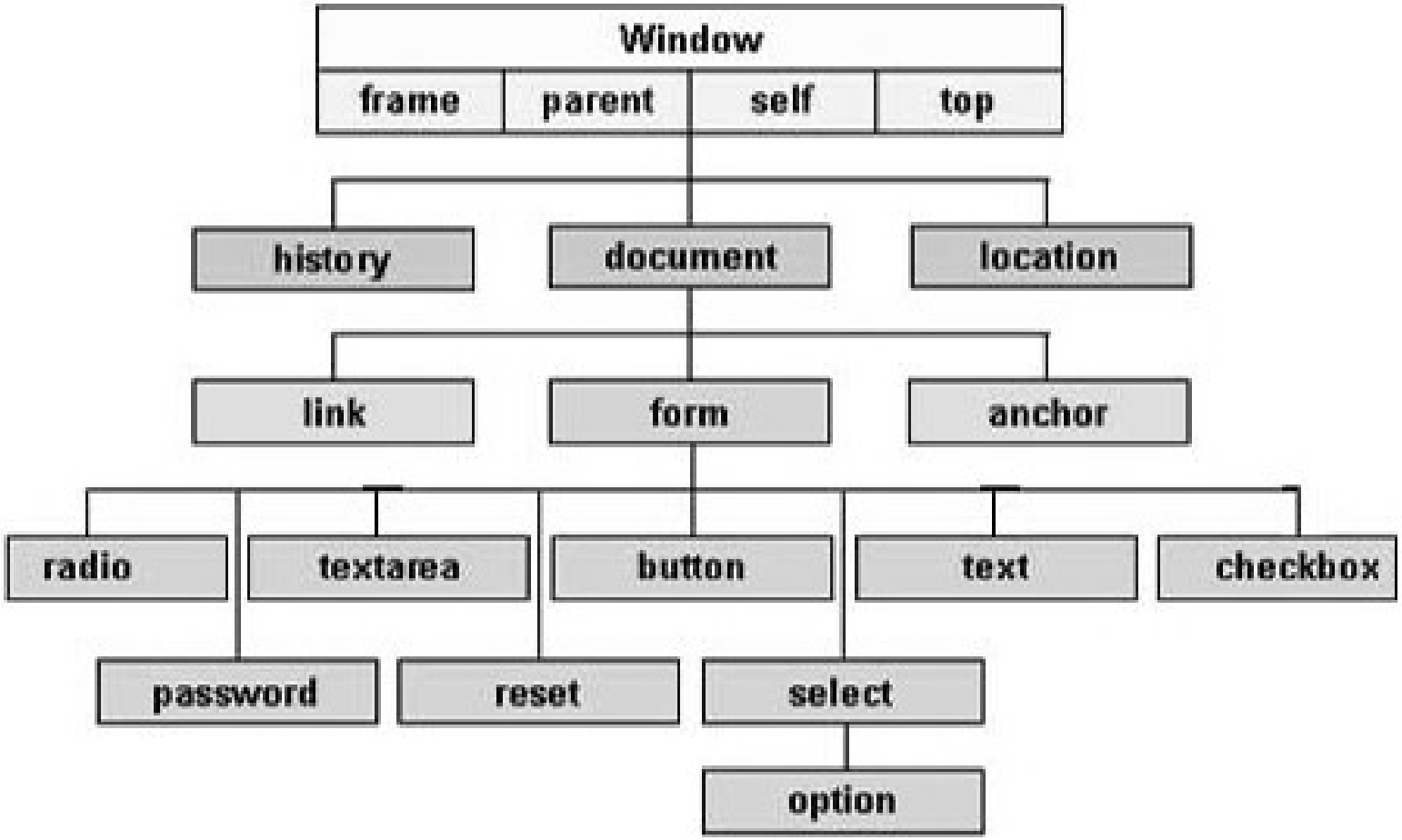
Every web page resides inside a browser window which can be considered as an object.

A Document object represents the HTML document that is displayed in that window. The Document object has various properties that refer to other objects which allow access to and modification of document content.

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 –



4. createElement

This method does exactly what it says: it creates an element and allows you to place that new element anywhere in the DOM structure.

Using the list example from above, we could add a new list item using the following code:

```
var myNewListItem = document.createElement("li");
var myNewLink = document.createElement("a");
```

The above code creates a new `<li>` element and a new anchor tag. But these elements don't exist anywhere except as values inside variables. To add our new elements to the DOM, we can use the next method listed in this article.

7. getAttribute

The `getAttribute` method allows you to access the value of any attribute on any element on your page. For example, going back to our list of links, suppose we had the following code:

```
<ul id="list">
  <li><a href="link1.html" class="link_one">Link Number One</a></li>
  <li><a href="link2.html">Link Number Two</a></li>
  <li><a href="link3.html">Link Number Three</a></li>
  <li><a href="link4.html">Link Number Four</a></li>
  <li><a href="link5.html" id="link_5" rel="external">Link Number Five</a></li
</ul>
```

We've added a new link with an `id` of "link_5" and a `rel` attribute with the value "external".

If we want to read the `rel` attribute of the fifth link in our list, we would use the following code:

```
var myLinkFive = document.getElementById("link_5");
var myLinkAttribute = myLinkFive.getAttribute("rel");
```

8. setAttribute

Writing a new value for a given attribute is another useful method, and can be used in conjunction with `getAttribute`. Let's say we wanted to change the value of the `rel` attribute from the previous example. Here's how that is accomplished:

```
var myLinkFive = document.getElementById("link_5");
myLinkFive.setAttribute("rel", "nofollow");
```

With the above code, the fifth link in our list would now have a `rel` attribute with the value "nofollow", instead of "external".

Keep in mind that an attribute can not only be changed, but it can be added. So, if our fifth link did not have a `rel` attribute, the above code would still work, and would create the `rel` attribute and assign it a value on the fly.

Javascript | eval() Function

eval() is a function property of the global object.

- The argument of the **eval()** function is a string.
- If the string represents an expression, **eval()** evaluates the expression. If the argument represents one or more JavaScript statements, **eval()** evaluates the statements.
- We do not call **eval()** to evaluate an arithmetic expression; JavaScript evaluates arithmetic expressions automatically.

Syntax:

```
eval(string)
Parameters:
String:
A string representing a JavaScript expression, statement, or
sequence of statements. The expression can include variables
and properties of existing objects.

Return Value:
The completion value of evaluating the given code is
returned by using eval().
If the completion value is empty, undefined is returned.
```

Examples:

Examples:

```
Input : eval(new String('2 + 2'));
Output : returns a String object containing "2 + 2"

Input : eval(new String('4 + 4'));
Output : returns a String object containing "4 + 4"
```

Below Programs will illustrate the use of **eval():** function more deeply:

Program 1:

```
<script>
    // JavaScript to illustrate eval() function
    function func() {

        // Original string
        var a = 2;
        var b = 2;

        // Finding the sum
        var value = eval(new String(a + b));
        document.write(value);
    }
// Driver Code
func();
< /script>
```

Output:

```
4
```

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

<p>Click the button to make a BUTTON element with text.</p>

<button onclick="myFunction()">Try it</button>

<script>
function myFunction() {
  var btn = document.createElement("BUTTON");
  var t = document.createTextNode("CLICK ME");
  btn.appendChild(t);
  document.body.appendChild(btn);
}
</script>

</body>
</html>
```

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

<p>Click the button to make a BUTTON element.</p>

<button onclick="myFunction()">Try it</button>

<script>
function myFunction() {
  var btn = document.createElement("BUTTON");
  document.body.appendChild(btn);
}
</script>

</body>
</html>
```

HTML elements often contains text. To create a button *with text* you must also create a Text Node which you append to the <button> element:

Example

Create a button with text:

```
var btn = document.createElement("BUTTON");        // Create a <button> element
var t = document.createTextNode("CLICK ME");       // Create a text node
btn.appendChild(t);                                // Append the text to <button>
document.body.appendChild(btn);                    // Append <button> to <body>
```

Try it Yourself »

```
<head>
<style>
.democlass {
  color: red;
}
</style>
</head>
<body>

<h1 class="democlass">Hello World</h1>

<p>Click the button to display the value of the class attribute of the h1
element.</p>

<button onclick="myFunction()">Try it</button>

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

<script>
function myFunction() {
  var x = document.getElementsByTagName("H1")[0].getAttribute("class");
  document.getElementById("demo").innerHTML = x;
}
</script>

</body>
</html>
```