# JAVASCRIPT - THE W3C DOM

 

This document object model allows access and modification of all document content and is standardized by the World Wide Web Consortium *W3C*. This model is supported by almost all the modern browsers.

The W3C DOM standardizes most of the features of the legacy DOM and adds new ones as well. In addition to supporting forms[ ], images[ ], and other array properties of the Document object, it defines methods that allow scripts to access and manipulate any document element and not just special-purpose elements like forms and images.

## Document Properties in W3C DOM

This model supports all the properties available in Legacy DOM. Additionally, here is a list of document properties which can be accessed using W3C DOM.

| Sr.No | Property & Description |
|---|---|
| 1 | **body**<br><br>A reference to the Element object that represents the <body> tag of this document.<br><br>**Ex** − document.body |
| 2 | **defaultView**<br><br>Its Read-only property and represents the window in which the document is displayed.<br><br>**Ex** − document.defaultView |
| 3 | **documentElement**<br><br>A read-only reference to the <html> tag of the document.<br><br>**Ex** − document.documentElement8/31/2008 |
| 4 | **implementation**<br><br>It is a read-only property and represents the DOMImplementation object that represents the implementation that created this document.<br><br>**Ex** − document.implementation |

## Document Methods in W3C DOM

This model supports all the methods available in Legacy DOM. Additionally, here is a list of methods supported by W3C DOM.

| Sr.No | Property & Description |
|---|---|
| 1 | **createAttribute*name*** |

Returns a newly-created Attr node with the specified name.

**Ex** − document.createAttribute*name*

2

**createComment***text*

Creates and returns a new Comment node containing the specified text.

**Ex** − document.createComment*text*

3

**createDocumentFragment**

Creates and returns an empty DocumentFragment node.

**Ex** − document.createDocumentFragment

4

**createElement***tagName*

Creates and returns a new Element node with the specified tag name.

**Ex** − document.createElement*tagName*

5

**createTextNode***text*

Creates and returns a new Text node that contains the specified text.

**Ex** − document.createTextNode*text*

6

**getElementById***id*

Returns the Element of this document that has the specified value for its id attribute, or null if no such Element exists in the document.

**Ex** − document.getElementById*id*

7

**getElementsByName***name*

Returns an array of nodes of all elements in the document that have a specified value for their name attribute. If no such elements are found, returns a zero-length array.

**Ex** − document.getElementsByName*name*

8

**getElementsByTagName***tagname*

Returns an array of all Element nodes in this document that have the specified tag name. The Element nodes appear in the returned array in the same order they appear in the document source.

**Ex** − document.getElementsByTagName*tagname*

9

**importNode***importedNode, deep*

Creates and returns a copy of a node from some other document that is suitable for insertion into this document. If the deep argument is true, it recursively copies the

children of the node too. Supported in DOM Version 2

**Ex** − document.importNode*importedNode*, *deep*

## Example

This is very easy to manipulate *AccessingandSetting* document element using W3C DOM. You can use any of the methods like **getElementById**, **getElementsByName**, or **getElementsByTagName**.

Here is an example to access document properties using W3C DOM method.

```
<html>

   <head>
      <title> Document Title </title>

      <script type="text/javascript">
         <!--
            function myFunc()
            {
               var ret = document.getElementsByTagName("title");
               alert("Document Title : " + ret[0].text );

               var ret = document.getElementById("heading");
               alert(ret.innerHTML );
            }
         //-->
      </script>

   </head>
   <body>
      <h1 >This is main title</h1>
      <p>Click the following to see the result:</p>

      <form >
         <input type="button" value="Click Me" onclick="myFunc();" />
         <input type="button" value="Cancel">
      </form>

      <form d="form2" name="SecondForm">
         <input type="button" value="Don't ClickMe"/>
      </form>

   </body>
</html>
```

**NOTE** − This example returns objects for forms and elements and we would have to access their values by using those object properties which are not discussed in this tutorial.