JAVASCRIPT - THE LEGACY DOM

This is the model which was introduced in early versions of JavaScript language. It is well supported by all browsers, but allows access only to certain key portions of documents, such as forms, form elements, and images.

This model provides several read-only properties, such as title, URL, and lastModified provide information about the document as a whole. Apart from that, there are various methods provided by this model which can be used to set and get document property values.

Document Properties in Legacy DOM

Here is a list of the document properties which can be accessed using Legacy DOM.

Sr.No	Property & Description
1	**alinkColor** Deprecated − A string that specifies the color of activated links. **Ex** − document.alinkColor
2	**anchors[ ]** An array of Anchor objects, one for each anchor that appears in the document **Ex** − document.anchors[0], document.anchors[1] and so on
3	**applets[ ]** An array of Applet objects, one for each applet that appears in the document **Ex** − document.applets[0], document.applets[1] and so on
4	**bgColor** Deprecated − A string that specifies the background color of the document. **Ex** − document.bgColor
5	**cookie** A string-valued property with special behavior that allows the cookies associated with this document to be queried and set. **Ex** − document.cookie
6	**domain** A string that specifies the Internet domain the document is from. Used for security purpose. **Ex** − document.domain
7	**embeds[ ]** An array of objects that represent data embedded in the document with the <embed>

tag. A synonym for plugins[ ]. Some plugins and ActiveX controls can be controlled with JavaScript code.

Ex − document.embeds[0], document.embeds[1] and so on

8 **fgColor**

Deprecated - A string that specifies the default text color for the document

Ex − document.fgColor

9 **forms[ ]**

An array of Form objects, one for each HTML form that appears in the document.

Ex − document.forms[0], document.forms[1] and so on

10 **images[ ]**

An array of Image objects, one for each image that is embedded in the document with the HTML <img> tag.

Ex − document.images[0], document.images[1] and so on

11 **lastModified**

A read-only string that specifies the date of the most recent change to the document

Ex − document.lastModified

12 **linkColor**

Deprecated − A string that specifies the color of unvisited links

Ex − document.linkColor

13 **links[ ]**

It is a document link array.

Ex − document.links[0], document.links[1] and so on

14 **location**

The URL of the document. Deprecated in favor of the URL property.

Ex − document.location

15 **plugins[ ]**

A synonym for the embeds[ ]

Ex − document.plugins[0], document.plugins[1] and so on

16 **Referrer**

A read-only string that contains the URL of the document, if any, from which the current document was linked.

Ex − document.referrer

17	**Title** The text contents of the <title> tag. **Ex** − document.title
18	**URL** A read-only string that specifies the URL of the document. **Ex** − document.URL
19	**vlinkColor** Deprecated − A string that specifies the color of visited links. **Ex** − document.vlinkColor

Document Methods in Legacy DOM

Here is a list of methods supported by Legacy DOM.

Sr.No	Property & Description
1	**clear** Deprecated − Erases the contents of the document and returns nothing. **Ex** − document.clear
2	**close** Closes a document stream opened with the open method and returns nothing. **Ex** − document.close
3	**open** Deletes existing document content and opens a stream to which new document contents may be written. Returns nothing. **Ex** − document.open
4	**write***value, . . .* Inserts the specified string or strings into the document currently being parsed or appends to document opened with open. Returns nothing. **Ex** − document.write*value, . . .*
4	**writeln***value, . . .* Identical to write, except that it appends a newline character to the output. Returns nothing. **Ex** − document.writeln*value, . . .*

Example

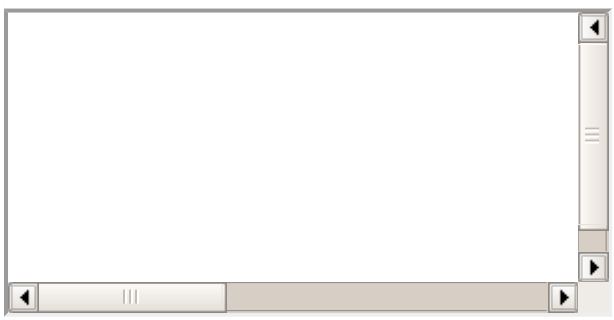
We can locate any HTML element within any HTML document using HTML DOM. For instance, if a web document contains a **form** element then using JavaScript we can refer to it as **document.forms[0]**. If your Web document includes two **form** elements the first form is referred to as document.forms[0] and the second document.forms[1].

Using the hierarchy and properties given above, we can access the first form element using **document.forms[0].elements[0]** and so on.

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

```
<html>

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

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

               var ret = document.URL;
               alert("Document URL : " + ret );

               var ret = document.forms[0];
               alert("Document First Form : " + ret );

               var ret = document.forms[0].elements[1];
               alert("Second element : " + ret );
            }
         //-->
      </script>

   </head>

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

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

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

   </body>
</html>
```

Output



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.