Regular Expressions and RegExp Object

Advertisements

[Previous Page](http://www.tutorialspoint.com/javascript/javascript_math_object.htm)

[Next Page](http://www.tutorialspoint.com/javascript/javascript_html_dom.htm)

A regular expression is an object that describes a pattern of characters.

The JavaScript **RegExp** class represents regular expressions, and both String and **RegExp**define methods that use regular expressions to perform powerful pattern-matching and search-and-replace functions on text.

Syntax

A regular expression could be defined with the **RegExp ()** constructor, as follows −

var pattern = new RegExp(pattern, attributes);

or simply

var pattern = /pattern/attributes;

Here is the description of the parameters −

* **pattern** − A string that specifies the pattern of the regular expression or another regular expression.
* **attributes** − An optional string containing any of the "g", "i", and "m" attributes that specify global, case-insensitive, and multiline matches, respectively.

Brackets

Brackets ([]) have a special meaning when used in the context of regular expressions. They are used to find a range of characters.

|  |  |
| --- | --- |
| **Expression** | **Description** |
| [...] | Any one character between the brackets. |
| [^...] | Any one character not between the brackets. |
| [0-9] | It matches any decimal digit from 0 through 9. |
| [a-z] | It matches any character from lowercase a through lowercase z. |
| [A-Z] | It matches any character from uppercase A through uppercase Z. |
| [a-Z] | It matches any character from lowercase a through uppercase Z. |

The ranges shown above are general; you could also use the range [0-3] to match any decimal digit ranging from 0 through 3, or the range [b-v] to match any lowercase character ranging from **b** through **v**.

Quantifiers

The frequency or position of bracketed character sequences and single characters can be denoted by a special character. Each special character has a specific connotation. The +, \*, ?, and $ flags all follow a character sequence.

|  |  |
| --- | --- |
| **Expression** | **Description** |
| p+ | It matches any string containing at least one p. |
| p\* | It matches any string containing zero or more p's. |
| p? | It matches any string containing one or more p's. |
| p{**N**} | It matches any string containing a sequence of **N** p's |
| p{2,3} | It matches any string containing a sequence of two or three p's. |
| p{2, } | It matches any string containing a sequence of at least two p's. |
| p$ | It matches any string with p at the end of it. |
| **^**p | It matches any string with p at the beginning of it. |

Examples

Following examples explain more about matching characters.

|  |  |
| --- | --- |
| **Expression** | **Description** |
| [^a-zA-Z] | It matches any string not containing any of the characters ranging from **a**through **z** and **A** through Z. |
| p.p | It matches any string containing **p,** followed by any character, in turn followed by another **p**. |
| ^.{2}$ | It matches any string containing exactly two characters. |
| <b>(.\*)</b> | It matches any string enclosed within <b> and </b>. |
| p(hp)\* | It matches any string containing a **p** followed by zero or more instances of the sequence **hp**. |

Literal characters

|  |  |
| --- | --- |
| **Character** | **Description** |
| Alphanumeric | Itself |
| \0 | The NUL character (\u0000) |
| \t | Tab (\u0009) |
| \n | Newline (\u000A) |
| \v | Vertical tab (\u000B) |
| \f | Form feed (\u000C) |
| \r | Carriage return (\u000D) |
| \xnn | The Latin character specified by the hexadecimal number nn; for example, \x0A is the same as \n |
| \uxxxx | The Unicode character specified by the hexadecimal number xxxx; for example, \u0009 is the same as \t |
| \cX | The control character ^X; for example, \cJ is equivalent to the newline character \n |

Metacharacters

A metacharacter is simply an alphabetical character preceded by a backslash that acts to give the combination a special meaning.

For instance, you can search for a large sum of money using the '\d' metacharacter:**/([\d]+)000/**, Here **\d** will search for any string of numerical character.

The following table lists a set of metacharacters which can be used in PERL Style Regular Expressions.

|  |  |
| --- | --- |
| **Character** | **Description** |
| . | a single character |
| \s | a whitespace character (space, tab, newline) |
| \S | non-whitespace character |
| \d | a digit (0-9) |
| \D | a non-digit |
| \w | a word character (a-z, A-Z, 0-9, \_) |
| \W | a non-word character |
| [\b] | a literal backspace (special case). |
| [aeiou] | matches a single character in the given set |
| [^aeiou] | matches a single character outside the given set |
| (foo|bar|baz) | matches any of the alternatives specified |

Modifiers

Several modifiers are available that can simplify the way you work with **regexps,** like case sensitivity, searching in multiple lines, etc.

|  |  |
| --- | --- |
| **Modifier** | **Description** |
| i | Perform case-insensitive matching. |
| m | Specifies that if the string has newline or carriage return characters, the ^ and $ operators will now match against a newline boundary, instead of a string boundary |
| g | Performs a global matchthat is, find all matches rather than stopping after the first match. |

RegExp Properties

Here is a list of the properties associated with RegExp and their description.

|  |  |
| --- | --- |
| **Property** | **Description** |
| [**constructor**](http://www.tutorialspoint.com/javascript/regexp_constructor.htm) | Specifies the function that creates an object's prototype. |
| [**global**](http://www.tutorialspoint.com/javascript/regexp_global.htm) | Specifies if the "g" modifier is set. |
| [**ignoreCase**](http://www.tutorialspoint.com/javascript/regexp_ignorecase.htm) | Specifies if the "i" modifier is set. |
| [**lastIndex**](http://www.tutorialspoint.com/javascript/regexp_lastindex.htm) | The index at which to start the next match. |
| [**multiline**](http://www.tutorialspoint.com/javascript/regexp_multiline.htm) | Specifies if the "m" modifier is set. |
| [**source**](http://www.tutorialspoint.com/javascript/regexp_source.htm) | The text of the pattern. |

In the following sections, we will have a few examples to demonstrate the usage of RegExp properties.

RegExp Methods

Here is a list of the methods associated with RegExp along with their description.

|  |  |
| --- | --- |
| **Method** | **Description** |
| [**exec()**](http://www.tutorialspoint.com/javascript/regexp_exec.htm) | Executes a search for a match in its string parameter. |
| [**test()**](http://www.tutorialspoint.com/javascript/regexp_test.htm) | Tests for a match in its string parameter. |
| [**toSource()**](http://www.tutorialspoint.com/javascript/regexp_tosource.htm) | Returns an object literal representing the specified object; you can use this value to create a new object. |
| [**toString()**](http://www.tutorialspoint.com/javascript/regexp_tostring.htm) | Returns a string representing the specified object. |

In the following sections, we will have a few examples to demonstrate the usage of RegExp methods.

[Previous Page](http://www.tutorialspoint.com/javascript/javascript_math_object.htm)

[Print](http://www.tutorialspoint.com/cgi-bin/printpage.cgi" \t "_blank)

[PDF](http://www.tutorialspoint.com/javascript/pdf/javascript_regexp_object.pdf" \o "JavaScript The Regular Expressions and RegExp Object" \t "_blank)

[Next Page](http://www.tutorialspoint.com/javascript/javascript_html_dom.htm)

Advertisements

# JavaScript - Document Object Model or DOM

Advertisements

[Previous Page](http://www.tutorialspoint.com/javascript/javascript_regexp_object.htm)

[Next Page](http://www.tutorialspoint.com/javascript/javascript_error_handling.htm)

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 −



There are several DOMs in existence. The following sections explain each of these DOMs in detail and describe how you can use them to access and modify document content.

* [The Legacy DOM](http://www.tutorialspoint.com/javascript/javascript_legacy_dom.htm) − 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.
* [The W3C DOM](http://www.tutorialspoint.com/javascript/javascript_w3c_dom.htm) − 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 IE4 DOM](http://www.tutorialspoint.com/javascript/javascript_ie4_dom.htm) − This document object model was introduced in Version 4 of Microsoft's Internet Explorer browser. IE 5 and later versions include support for most basic W3C DOM features.

## DOM compatibility

If you want to write a script with the flexibility to use either W3C DOM or IE 4 DOM depending on their availability, then you can use a capability-testing approach that first checks for the existence of a method or property to determine whether the browser has the capability you desire. For example −

if (document.getElementById) {

// If the W3C method exists, use it

}

else if (document.all) {

// If the all[] array exists, use it

}

else {

// Otherwise use the legacy DOM

}

[Previous Page](http://www.tutorialspoint.com/javascript/javascript_regexp_object.htm)

[Print](http://www.tutorialspoint.com/cgi-bin/printpage.cgi" \t "_blank)

[PDF](http://www.tutorialspoint.com/javascript/pdf/javascript_html_dom.pdf" \o "JavaScript Document Object Model or DOM" \t "_blank)

[Next Page](http://www.tutorialspoint.com/javascript/javascript_error_handling.htm)

# JavaScript - Document Object Model or DOM

Advertisements

[Previous Page](http://www.tutorialspoint.com/javascript/javascript_regexp_object.htm)

[Next Page](http://www.tutorialspoint.com/javascript/javascript_error_handling.htm)

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 −



There are several DOMs in existence. The following sections explain each of these DOMs in detail and describe how you can use them to access and modify document content.

* [The Legacy DOM](http://www.tutorialspoint.com/javascript/javascript_legacy_dom.htm) − 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.
* [The W3C DOM](http://www.tutorialspoint.com/javascript/javascript_w3c_dom.htm) − 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 IE4 DOM](http://www.tutorialspoint.com/javascript/javascript_ie4_dom.htm) − This document object model was introduced in Version 4 of Microsoft's Internet Explorer browser. IE 5 and later versions include support for most basic W3C DOM features.

## DOM compatibility

If you want to write a script with the flexibility to use either W3C DOM or IE 4 DOM depending on their availability, then you can use a capability-testing approach that first checks for the existence of a method or property to determine whether the browser has the capability you desire. For example −

if (document.getElementById) {

// If the W3C method exists, use it

}

else if (document.all) {

// If the all[] array exists, use it

}

else {

// Otherwise use the legacy DOM

}

[Previous Page](http://www.tutorialspoint.com/javascript/javascript_regexp_object.htm)

[Print](http://www.tutorialspoint.com/cgi-bin/printpage.cgi" \t "_blank)

[PDF](http://www.tutorialspoint.com/javascript/pdf/javascript_html_dom.pdf" \o "JavaScript Document Object Model or DOM" \t "_blank)

[Next Page](http://www.tutorialspoint.com/javascript/javascript_error_handling.htm)