

## String

An object representing a series of characters in a string.

<i>Core object</i>	
<i>Implemented in</i>	<p>JavaScript 1.0: Create a String object only by quoting characters.</p> <p>JavaScript 1.1, NES 2.0: added String constructor; added prototype property; added <a href="#">split</a> method; added ability to pass strings among scripts in different windows or frames (in previous releases, you had to add an empty string to another window's string to refer to it).</p> <p>JavaScript 1.2, NES 3.0: added <a href="#">concat</a>, <a href="#">match</a>, <a href="#">replace</a>, <a href="#">search</a>, <a href="#">slice</a>, and <a href="#">substr</a> methods.</p> <p>JavaScript 1.3: added <a href="#">toSource</a> method; changed <a href="#">charCodeAt</a>, <a href="#">fromCharCode</a>, and <a href="#">replace</a> methods.</p>
<i>ECMA version</i>	ECMA-262

### Created by

The String constructor:

```
new String(string)
```

### Parameters

string	Any string.
--------	-------------

## Description

The String object is a wrapper around the string primitive data type. Do not confuse a string literal with the String object. For example, the following code creates the string literal s1 and also the String object s2:

```
s1 = "foo" // creates a string literal value
s2 = new String("foo") // creates a String object
```

You can call any of the methods of the String object on a string literal value-JavaScript automatically converts the string literal to a temporary String object, calls the method, then discards the temporary String object. You can also use the String.length property with a string literal.

You should use string literals unless you specifically need to use a String object, because String objects can have counterintuitive behavior. For example:

```
s1 = "2 + 2" // creates a string literal value
s2 = new String("2 + 2") // creates a String object
eval(s1) // returns the number 4
eval(s2) // returns the string "2 + 2"
```

A string can be represented as a literal enclosed by single or double quotation marks; for example, "Netscape" or 'Netscape'.

You can convert the value of any object into a string using the top-level [String](#) function.

## Property Summary

Property	Description
<a href="#">constructor</a>	Specifies the function that creates an object's prototype.
<a href="#">length</a>	Reflects the length of the string.
<a href="#">prototype</a>	Allows the addition of properties to a String object.

## Method Summary

<b>Method</b>	<b>Description</b>
<a href="#"><u>anchor</u></a>	Creates an HTML anchor that is used as a hypertext target.
<a href="#"><u>big</u></a>	Causes a string to be displayed in a big font as if it were in a BIG tag.
<a href="#"><u>blink</u></a>	Causes a string to blink as if it were in a BLINK tag.
<a href="#"><u>bold</u></a>	Causes a string to be displayed as if it were in a B tag.
<a href="#"><u>charAt</u></a>	Returns the character at the specified index.
<a href="#"><u>charCodeAt</u></a>	Returns a number indicating the Unicode value of the character at the given index.
<a href="#"><u>concat</u></a>	Combines the text of two strings and returns a new string.
<a href="#"><u>fixed</u></a>	Causes a string to be displayed in fixed-pitch font as if it were in a TT tag.
<a href="#"><u>fontcolor</u></a>	Causes a string to be displayed in the specified color as if it were in a <FONT COLOR=color> tag.
<a href="#"><u>fontsize</u></a>	Causes a string to be displayed in the specified font size as if it were in a <FONT SIZE=size> tag.
<a href="#"><u>fromCharCode</u></a>	Returns a string created by using the specified sequence of Unicode values. This is a method on the String class, not on a String instance.

<a href="#"><u>indexOf</u></a>	Returns the index within the calling String object of the first occurrence of the specified value, or -1 if not found.
<a href="#"><u>italics</u></a>	Causes a string to be italic, as if it were in an I tag.
<a href="#"><u>lastIndexOf</u></a>	Returns the index within the calling String object of the last occurrence of the specified value, or -1 if not found.
<a href="#"><u>link</u></a>	Creates an HTML hypertext link that requests another URL.
<a href="#"><u>match</u></a>	Used to match a regular expression against a string.
<a href="#"><u>replace</u></a>	Used to find a match between a regular expression and a string, and to replace the matched substring with a new substring.
<a href="#"><u>search</u></a>	Executes the search for a match between a regular expression and a specified string.
<a href="#"><u>slice</u></a>	Extracts a section of a string and returns a new string.
<a href="#"><u>small</u></a>	Causes a string to be displayed in a small font, as if it were in a SMALL tag.
<a href="#"><u>split</u></a>	Splits a String object into an array of strings by separating the string into substrings.
<a href="#"><u>strike</u></a>	Causes a string to be displayed as struck-out text, as if it were in a STRIKE tag.
<a href="#"><u>sub</u></a>	Causes a string to be displayed as a subscript, as if it were in a SUB tag.

<a href="#"><u>substr</u></a>	Returns the characters in a string beginning at the specified location through the specified number of characters.
<a href="#"><u>substring</u></a>	Returns the characters in a string between two indexes into the string.
<a href="#"><u>sup</u></a>	Causes a string to be displayed as a superscript, as if it were in a SUP tag.
<a href="#"><u>toLowerCase</u></a>	Returns the calling string value converted to lowercase.
<a href="#"><u>toSource</u></a>	Returns an object literal representing the specified object; you can use this value to create a new object. Overrides the <a href="#"><u>Object.toSource</u></a> method.
<a href="#"><u>toString</u></a>	Returns a string representing the specified object. Overrides the <a href="#"><u>Object.toString</u></a> method.
<a href="#"><u>toUpperCase</u></a>	Returns the calling string value converted to uppercase.
<a href="#"><u>valueOf</u></a>	Returns the primitive value of the specified object. Overrides the <a href="#"><u>Object.valueOf</u></a> method.

In addition, this object inherits the [watch](#) and [unwatch](#) methods from [Object](#).

## Examples

**Example 1: String literal.** The following statement creates a string literal:

```
var last_name = "Schaefer"
```

**Example 2: String literal properties.** The following statements evaluate to 8, "SCHAEFER," and "schaefer":

```
last_name.length
last_name.toUpperCase()
last_name.toLowerCase()
```

**Example 3: Accessing individual characters in a string.** You can think of a string as an array of characters. In this way, you can access the individual characters in the string by

indexing that array. For example, the following code displays "The first character in the string is H":

```
var myString = "Hello"
myString[0] // returns "H"
```

**Example 4: Pass a string among scripts in different windows or frames.** The following code creates two string variables and opens a second window:

```
var lastName = "Schaefer"
var firstName = "Jesse"
empWindow=window.open('string2.html','window1','width=300,height=300')
```

If the HTML source for the second window (string2.html) creates two string variables, empLastName and empFirstName, the following code in the first window assigns values to the second window's variables:

```
empWindow.empFirstName=firstName
empWindow.empLastName=lastName
```

The following code in the first window displays the values of the second window's variables:

```
alert('empFirstName in empWindow is ' + empWindow.empFirstName)
alert('empLastName in empWindow is ' + empWindow.empLastName)
```

## anchor

Creates an HTML anchor that is used as a hypertext target.

<i>Method of</i>	<a href="#">String</a>
<i>Implemented in</i>	JavaScript 1.0, NES 2.0

### Syntax

`anchor(nameAttribute)`

### Parameters

nameAttribute	
---------------	--

A string.

## Description

Use the anchor method with the document.write or document.writeln methods to programmatically create and display an anchor in a document. Create the anchor with the anchor method, and then call write or writeln to display the anchor in a document. In server-side JavaScript, use the write function to display the anchor.

In the syntax, the text string represents the literal text that you want the user to see. The nameAttribute string represents the NAME attribute of the A tag.

Anchors created with the anchor method become elements in the document.anchors array.

## Examples

The following example opens the msgWindow window and creates an anchor for the table of contents:

```
var myString="Table of Contents"  
msgWindow.document.writeln(myString.anchor("contents_anchor"))
```

The previous example produces the same output as the following HTML:

```
<A NAME="contents_anchor">Table of Contents</A>
```

## See also

[String.link](#)

## big

Causes a string to be displayed in a big font as if it were in a BIG tag.

<i>Method of</i>	<a href="#">String</a>
<i>Implemented in</i>	JavaScript 1.0, NES 2.0

## Syntax

big()

## Parameters

None

## Description

Use the `big` method with the `write` or `writeln` methods to format and display a string in a document. In server-side JavaScript, use the `write` function to display the string.

## Examples

The following example uses string methods to change the size of a string:

```
var worldString="Hello, world"

document.write(worldString.small())
document.write("<P>" + worldString.big())
document.write("<P>" + worldString.fontsize(7))
```

The previous example produces the same output as the following HTML:

```
<SMALL>Hello, world</SMALL>
<P><BIG>Hello, world</BIG>
<P><FONTSIZE=7>Hello, world</FONTSIZE>
```

## See also

[String.fontsize](#), [String.small](#)

## blink

Causes a string to blink as if it were in a BLINK tag.

<i>Method of</i>	<a href="#">String</a>
<i>Implemented in</i>	JavaScript 1.0, NES 2.0

## Syntax

`blink()`

## Parameters

None

## Description

Use the `blink` method with the `write` or `writeln` methods to format and display a string in a document. In server-side JavaScript, use the `write` function to display the string.

## Examples

The following example uses string methods to change the formatting of a string:

```
var worldString="Hello, world"
```

```
document.write(worldString.blink())
document.write("<P>" + worldString.bold())
document.write("<P>" + worldString.italics())
document.write("<P>" + worldString.strike())
```

The previous example produces the same output as the following HTML:

```
<BLINK>Hello, world</BLINK>
<P><B>Hello, world</B>
<P><I>Hello, world</I>
<P><STRIKE>Hello, world</STRIKE>
```

## See also

[String.bold](#), [String.italics](#), [String.strike](#)

## bold

Causes a string to be displayed as bold as if it were in a B tag.

<i>Method of</i>	<a href="#">String</a>
<i>Implemented in</i>	JavaScript 1.0, NES 2.0

## Syntax

`bold()`

## Parameters

None

## Description

Use the bold method with the write or writeln methods to format and display a string in a document. In server-side JavaScript, use the write function to display the string.

## Examples

The following example uses string methods to change the formatting of a string:

```
var worldString="Hello, world"
document.write(worldString.blink())
document.write("<P>" + worldString.bold())
document.write("<P>" + worldString.italics())
document.write("<P>" + worldString.strike())
```

The previous example produces the same output as the following HTML:

```
<BLINK>Hello, world</BLINK>
<P><B>Hello, world</B>
<P><I>Hello, world</I>
<P><STRIKE>Hello, world</STRIKE>
```

## See also

[String.blink](#), [String.italics](#), [String.strike](#)

## charAt

Returns the specified character from the string.

<i>Method of</i>	<a href="#">String</a>
<i>Implemented in</i>	JavaScript 1.0, NES 2.0
<i>ECMA version</i>	ECMA-262

## Syntax

charAt(*index*)

## Parameters

<i>index</i>	An integer between 0 and 1 less than the length of the string.
--------------	--

## Description

Characters in a string are indexed from left to right. The index of the first character is 0, and the index of the last character in a string called `stringName` is `stringName.length - 1`. If the index you supply is out of range, JavaScript returns an empty string.

## Examples

The following example displays characters at different locations in the string "Brave new world":

```
var anyString="Brave new world"
```

```
document.writeln("The character at index 0 is " + anyString.charAt(0))
document.writeln("The character at index 1 is " + anyString.charAt(1))
document.writeln("The character at index 2 is " + anyString.charAt(2))
document.writeln("The character at index 3 is " + anyString.charAt(3))
document.writeln("The character at index 4 is " + anyString.charAt(4))
```

These lines display the following:

The character at index 0 is B  
 The character at index 1 is r  
 The character at index 2 is a  
 The character at index 3 is v  
 The character at index 4 is e

## See also

[String.indexOf](#), [String.lastIndexOf](#), [String.split](#)

## charCodeAt

Returns a number indicating the Unicode value of the character at the given index.

<i>Method of</i>	<a href="#">String</a>
<i>Implemented in</i>	JavaScript 1.2, NES 3.0  JavaScript 1.3: returns a Unicode value rather than an ISO-Latin-1 value.
<i>ECMA version</i>	ECMA-262

**Syntax**`charCodeAt([index])`**Parameters****index**

An integer between 0 and 1 less than the length of the string. The default value is 0.

**Description**

Unicode values range from 0 to 65,535. The first 128 Unicode values are a direct match of the ASCII character set. For information on Unicode, see the [Core JavaScript Guide](#).

**Backward Compatibility**

**JavaScript 1.2.** The `charCodeAt` method returns a number indicating the ISO-Latin-1 codeset value of the character at the given index. The ISO-Latin-1 codeset ranges from 0 to 255. The first 0 to 127 are a direct match of the ASCII character set.

**Example**

The following example returns 65, the Unicode value for A.

```
"ABC".charCodeAt(0) // returns 65
```

**concat**

Combines the text of two or more strings and returns a new string.

<i>Method of</i>	<a href="#">String</a>
<i>Implemented in</i>	JavaScript 1.2, NES 3.0

**Syntax**`concat(string2, string3[, ..., stringN])`

## Parameters

string2... stringN	Strings to concatenate to this string.
-----------------------	--

## Description

concat combines the text from one or more strings and returns a new string. Changes to the text in one string do not affect the other string.

## Example

The following example combines two strings into a new string.

```
s1="Oh "
s2="what a beautiful "
s3="mornin'."
s4=s1.concat(s2,s3) // returns "Oh what a beautiful mornin'."
```

## constructor

Specifies the function that creates an object's prototype. Note that the value of this property is a reference to the function itself, not a string containing the function's name.

<i>Property of</i>	<a href="#">String</a>
<i>Implemented in</i>	JavaScript 1.1, NES 2.0
<i>ECMA version</i>	ECMA-262

## Description

See [Object.constructor](#).

## fixed

Causes a string to be displayed in fixed-pitch font as if it were in a TT tag.

<i>Method of</i>	<a href="#">String</a>
<i>Implemented in</i>	JavaScript 1.0, NES 2.0

## Syntax

`fixed()`

## Parameters

None

## Description

Use the fixed method with the write or writeln methods to format and display a string in a document. In server-side JavaScript, use the write function to display the string.

## Examples

The following example uses the fixed method to change the formatting of a string:

```
var worldString="Hello, world"
document.write(worldString.fixed())
```

The previous example produces the same output as the following HTML:

```
<TT>Hello, world</TT>
```

## fontcolor

Causes a string to be displayed in the specified color as if it were in a <FONT COLOR=color> tag.

<i>Method of</i>	<a href="#">String</a>
<i>Implemented in</i>	JavaScript 1.0, NES 2.0

## Syntax

`fontcolor(color)`

## Parameters

color	A string expressing the color as a hexadecimal RGB triplet or as a string literal. String literals for color names are listed in the <a href="#">Core JavaScript Guide</a> .
-------	--

## Description

Use the fontcolor method with the write or writeln methods to format and display a string in a document. In server-side JavaScript, use the write function to display the string.

If you express color as a hexadecimal RGB triplet, you must use the format rrggbb. For example, the hexadecimal RGB values for salmon are red=FA, green=80, and blue=72, so the RGB triplet for salmon is "FA8072".

The fontcolor method overrides a value set in the fgColor property.

## Examples

The following example uses the fontcolor method to change the color of a string:

```
var worldString="Hello, world"

document.write(worldString.fontcolor("maroon") +
  " is maroon in this line")
document.write("<P>" + worldString.fontcolor("salmon") +
  " is salmon in this line")
document.write("<P>" + worldString.fontcolor("red") +
  " is red in this line")

document.write("<P>" + worldString.fontcolor("8000") +
  " is maroon in hexadecimal in this line")
document.write("<P>" + worldString.fontcolor("FA8072") +
  " is salmon in hexadecimal in this line")
document.write("<P>" + worldString.fontcolor("FF00") +
  " is red in hexadecimal in this line")
```

The previous example produces the same output as the following HTML:

```
<FONT COLOR="maroon">Hello, world</FONT> is maroon in this line
<P><FONT COLOR="salmon">Hello, world</FONT> is salmon in this line
<P><FONT COLOR="red">Hello, world</FONT> is red in this line

<FONT COLOR="8000">Hello, world</FONT>
is maroon in hexadecimal in this line
```

<P><FONT COLOR="FA8072">Hello, world</FONT>

is salmon in hexadecimal in this line

<P><FONT COLOR="FF00">Hello, world</FONT>

is red in hexadecimal in this line

## fontsize

Causes a string to be displayed in the specified font size as if it were in a <FONT SIZE=size> tag.

<i>Method of</i>	<a href="#">String</a>
<i>Implemented in</i>	JavaScript 1.0, NES 2.0

### Syntax

`fontsize(size)`

### Parameters

<code>size</code>	An integer between 1 and 7, a string representing a signed integer between 1 and 7.
-------------------	---

### Description

Use the `fontsize` method with the `write` or `writeln` methods to format and display a string in a document. In server-side JavaScript, use the `write` function to display the string.

When you specify `size` as an integer, you set the size of `stringName` to one of the 7 defined sizes. When you specify `size` as a string such as "-2", you adjust the font size of `stringName` relative to the size set in the `BASEFONT` tag.

### Examples

The following example uses string methods to change the size of a string:

```
var worldString="Hello, world"
```

```
document.write(worldString.small())
```

```
document.write("<P>" + worldString.big())
document.write("<P>" + worldString.fontsize(7))
```

The previous example produces the same output as the following HTML:

```
<SMALL>Hello, world</SMALL>
<P><BIG>Hello, world</BIG>
<P><FONTSIZE=7>Hello, world</FONTSIZE>
```

## See also

[String.big](#), [String.small](#)

## fromCharCode

Returns a string created by using the specified sequence of Unicode values.

<i>Method of</i>	<a href="#">String</a>
<i>Static</i>	
<i>Implemented in</i>	JavaScript 1.2, NES 3.0 JavaScript 1.3: uses a Unicode value rather than an ISO-Latin-1 value.
<i>ECMA version</i>	ECMA-262

## Syntax

`fromCharCode(num1, ..., numN)`

## Parameters

<code>num1, ..., numN</code>	A sequence of numbers that are Unicode values.
------------------------------	--

## Description

This method returns a string and not a String object.

Because fromCharCode is a static method of String, you always use it as `String.fromCharCode()`, rather than as a method of a String object you created.

## Backward Compatibility

**JavaScript 1.2.** The `fromCharCode` method returns a string created by using the specified sequence of ISO-Latin-1 codeset values.

## Examples

The following example returns the string "ABC".

```
String.fromCharCode(65,66,67)
```

## indexOf

Returns the index within the calling String object of the first occurrence of the specified value, starting the search at `fromIndex`, or -1 if the value is not found.

<i>Method of</i>	<a href="#">String</a>
<i>Implemented in</i>	JavaScript 1.0, NES 2.0
<i>ECMA version</i>	ECMA-262

## Syntax

`indexOf(searchValue[, fromIndex])`

## Parameters

searchValue	A string representing the value to search for.
-------------	--

fromIndex	The location within the calling string to start the search from. It can be any integer between 0 and the length of the string. The default value is 0.
-----------	--

## Description

Characters in a string are indexed from left to right. The index of the first character is 0, and the index of the last character of a string called stringName is stringName.length - 1.

```
"Blue Whale".indexOf("Blue") // returns 0
"Blue Whale".indexOf("Blute") // returns -1
"Blue Whale".indexOf("Whale",0) // returns 5
"Blue Whale".indexOf("Whale",5) // returns 5
"Blue Whale".indexOf("",9) // returns 9
"Blue Whale".indexOf("",10) // returns 10
"Blue Whale".indexOf("",11) // returns 10
```

The indexOf method is case sensitive. For example, the following expression returns -1:

```
"Blue Whale".indexOf("blue")
```

## Examples

**Example 1.** The following example uses indexOf and lastIndexOf to locate values in the string "Brave new world."

```
var anyString="Brave new world"

// Displays 8
document.write("<P>The index of the first w from the beginning is " +
anyString.indexOf("w"))
// Displays 10
document.write("<P>The index of the first w from the end is " +
anyString.lastIndexOf("w"))
// Displays 6
document.write("<P>The index of 'new' from the beginning is " +
anyString.indexOf("new"))
// Displays 6
document.write("<P>The index of 'new' from the end is " +
anyString.lastIndexOf("new"))
```

**Example 2.** The following example defines two string variables. The variables contain the same string except that the second string contains uppercase letters. The first writeln method displays 19. But because the indexOf method is case sensitive, the string "cheddar" is not found in myCapString, so the second writeln method displays -1.

```
myString="brie, pepper jack, cheddar"
myCapString="Brie, Pepper Jack, Cheddar"
document.writeln('myString.indexOf("cheddar") is ' +
myString.indexOf("cheddar"))
```

```
document.writeln('<P>myCapString.indexOf("cheddar") is ' +
  myCapString.indexOf("cheddar"))
```

**Example 3.** The following example sets count to the number of occurrences of the letter x in the string str:

```
count = 0;
pos = str.indexOf("x");
while ( pos != -1 ) {
  count++;
  pos = str.indexOf("x",pos+1);
}
```

## See also

[String.charAt](#), [String.lastIndexOf](#), [String.split](#)

## italics

Causes a string to be italic, as if it were in an <I> tag.

<i>Method of</i>	<a href="#">String</a>
<i>Implemented in</i>	JavaScript 1.0, NES 2.0

## Syntax

italics()

## Parameters

None

## Description

Use the `italics` method with the `write` or `writeln` methods to format and display a string in a document. In server-side JavaScript, use the `write` function to display the string.

## Examples

The following example uses string methods to change the formatting of a string:

```
var worldString="Hello, world"
```

```
document.write(worldString.blink())
document.write("<P>" + worldString.bold())
document.write("<P>" + worldString.italics())
```

```
document.write("<P>" + worldString.strike())
```

The previous example produces the same output as the following HTML:

```
<BLINK>Hello, world</BLINK>
<P><B>Hello, world</B>
<P><I>Hello, world</I>
<P><STRIKE>Hello, world</STRIKE>
```

## See also

[String.blink](#), [String.bold](#), [String.strike](#)

## lastIndexOf

Returns the index within the calling String object of the last occurrence of the specified value, or -1 if not found. The calling string is searched backward, starting at fromIndex.

<i>Method of</i>	<a href="#">String</a>
<i>Implemented in</i>	JavaScript 1.0, NES 2.0
<i>ECMA version</i>	ECMA-262

## Syntax

`lastIndexOf(searchValue[, fromIndex])`

## Parameters

searchValue	A string representing the value to search for.
fromIndex	The location within the calling string to start the search from. It can be any integer between 0 and the length of the string. The default value is the length of the string.

## Description

Characters in a string are indexed from left to right. The index of the first character is 0, and the index of the last character is `stringName.length - 1`.

```
"canal".lastIndexOf("a") // returns 3
"canal".lastIndexOf("a",2) // returns 1
"canal".lastIndexOf("a",0) // returns -1
"canal".lastIndexOf("x") // returns -1
```

The `lastIndexOf` method is case sensitive. For example, the following expression returns -1:

```
"Blue Whale, Killer Whale".lastIndexOf("blue")
```

## Examples

The following example uses `indexOf` and `lastIndexOf` to locate values in the string "Brave new world."

```
var anyString="Brave new world"

// Displays 8
document.write("<P>The index of the first w from the beginning is " +
  anyString.indexOf("w"))
// Displays 10
document.write("<P>The index of the first w from the end is " +
  anyString.lastIndexOf("w"))
// Displays 6
document.write("<P>The index of 'new' from the beginning is " +
  anyString.indexOf("new"))
// Displays 6
document.write("<P>The index of 'new' from the end is " +
  anyString.lastIndexOf("new"))
```

## See also

[String.charAt](#), [String.indexOf](#), [String.split](#)

## length

The length of the string.

<i>Property of</i>	<a href="#">String</a>
<i>Read-only</i>	

<i>Implemented in</i>	JavaScript 1.0, NES 2.0
<i>ECMA version</i>	ECMA-262

**Description**

For a null string, length is 0.

**Examples**

The following example displays 8 in an Alert dialog box:

```
var x="Netscape"
alert("The string length is " + x.length)
```

**link**

Creates an HTML hypertext link that requests another URL.

<i>Method of</i>	<a href="#">String</a>
<i>Implemented in</i>	JavaScript 1.0, NES 2.0

**Syntax**

`link(hrefAttribute)`

**Parameters**

<code>hrefAttribute</code>	Any string that specifies the HREF attribute of the A tag; it should be a valid URL (relative or absolute).
----------------------------	---

**Description**

Use the link method to programmatically create a hypertext link, and then call write or writeln to display the link in a document. In server-side JavaScript, use the write function

to display the link.

Links created with the link method become elements in the links array of the document object. See document.links.

## Examples

The following example displays the word "Netscape" as a hypertext link that returns the user to the Netscape home page:

```
var hotText="Netscape"
var URL="http://home.netscape.com"

document.write("Click to return to " + hotText.link(URL))
```

The previous example produces the same output as the following HTML:

```
Click to return to <A HREF="http://home.netscape.com">Netscape</A>
```

## match

Used to match a regular expression against a string.

<i>Method of</i>	<a href="#">String</a>
<i>Implemented in</i>	JavaScript 1.2
<i>ECMA version</i>	ECMA-262 Edition 3

## Syntax

`match(regexp)`

## Parameters

<code>regexp</code>	Name of the regular expression. It can be a variable name or a literal.
---------------------	---

## Description

If the regular expression does not include the g flag, returns the same result that RegExp.exec would return on the regular expression and string. If the regular expression includes the g flag, returns an array of all the matches of the regular expression in the string.

**Note** If you execute a match simply to find true or false, use [String.search](#) or the regular expression test method.

## Examples

**Example 1.** In the following example, match is used to find 'Chapter' followed by 1 or more numeric characters followed by a decimal point and numeric character 0 or more times. The regular expression includes the i flag so that case will be ignored.

```
<SCRIPT>
str = "For more information, see Chapter 3.4.5.1";
re = /(chapter \d+(\.\d)*)/i;
found = str.match(re);
document.write(found);
</SCRIPT>
```

This returns the array containing Chapter 3.4.5.1, Chapter 3.4.5.1,.1

'Chapter 3.4.5.1' is the first match and the first value remembered from (Chapter \d+ (\.\d)\*).

'.1' is the second value remembered from (\.\d).

**Example 2.** The following example demonstrates the use of the global and ignore case flags with match.

```
<SCRIPT>
str = "abcDdcba";
newArray = str.match(/d/gi);
document.write(newArray);
</SCRIPT>
```

The returned array contains D, d.

## prototype

Represents the prototype for this class. You can use the prototype to add properties or methods to all instances of a class. For information on prototypes, see [Function.prototype](#).

<i>Property of</i>	<a href="#">String</a>
--------------------	------------------------

<i>Implemented in</i>	JavaScript 1.1, NES 3.0
<i>ECMA version</i>	ECMA-262

## replace

Finds a match between a regular expression and a string, and replaces the matched substring with a new substring.

<i>Method of</i>	<a href="#">String</a>
<i>Implemented in</i>	JavaScript 1.2 JavaScript 1.3: added the ability to specify a function as the second parameter.
<i>ECMA version</i>	ECMA-262 Edition 3

### Syntax

`replace(regexp, newSubStr)`  
`replace(regexp, function)`

*Versions prior to JavaScript 1.3:*

`replace(regexp, newSubStr)`

### Parameters

<i>regexp</i>	The name of the regular expression. It can be a variable name or a literal.
<i>newSubStr</i>	The string to put in place of the string found with <i>regexp</i> .

function	A function to be invoked after the match has been performed.

### Description

This method does not change the String object it is called on. It simply returns a new string.

If you want to execute a global search and replace, include the g flag in the regular expression.

**Specifying a string as a parameter.** The replacement string can include the following special replacement patterns:

\$\$	Inserts a '\$'.
\$&	Inserts the matched substring..
\$`	Inserts the portion of the string that precedes the matched substring.
\$`	Inserts the portion of the string that follows the matched substring.
\$n or \$nn	Where <i>n</i> or <i>nn</i> are decimal digits, inserts the <i>n</i> th parenthesized submatch string.

**Specifying a function as a parameter.** When you specify a function as the second parameter, the function is invoked after the match has been performed. (The use of a function in this manner is often called a lambda expression.)

In your function, you can dynamically generate the string that replaces the matched substring. The result of the function call is used as the replacement value.

The nested function can use the matched substrings to determine the new string (newSubStr) that replaces the found substring. You get the matched substrings through

the parameters of your function. The first parameter of your function holds the complete matched substring. The following  $n$  parameters can be used for parenthetical matches, remembered submatch strings, where  $n$  is the number of submatch strings in the regular expression. Finally, the last two parameters are the offset within the string where the match occurred and the string itself. For example, the following replace method returns XX.zzzz - XX , zzzz.

```
"XXzzzz".replace(/(X*)(z*)/,  
    function (str, p1, p2, offset, s) {  
        return str + " - " + p1 + " , " + p2;  
    }  
)
```

## Backward Compatibility

**JavaScript 1.2.** You cannot specify a function to be invoked after the match has been performed.

### Examples

**Example 1.** In the following example, the regular expression includes the global and ignore case flags which permits replace to replace each occurrence of 'apples' in the string with 'oranges.'

```
<SCRIPT>  
re = /apples/gi;  
str = "Apples are round, and apples are juicy.";  
newstr=str.replace(re, "oranges");  
document.write(newstr)  
</SCRIPT>
```

This prints "oranges are round, and oranges are juicy."

**Example 2.** In the following example, the regular expression is defined in replace and includes the ignore case flag.

```
<SCRIPT>  
str = "Twas the night before Xmas...";  
newstr=str.replace(/xmas/i, "Christmas");  
document.write(newstr)  
</SCRIPT>
```

This prints "Twas the night before Christmas..."

**Example 3.** The following script switches the words in the string. For the replacement text, the script uses the \$1 and \$2 replacement patterns.

```
<SCRIPT LANGUAGE="JavaScript1.2">  
re = /(\w+)\s(\w+)/;  
str = "John Smith";  
newstr = str.replace(re, "$2, $1");  
document.write(newstr)  
</SCRIPT>
```

This prints "Smith, John".

**Example 4.** The following example replaces a Fahrenheit degree with its equivalent Celsius degree. The Fahrenheit degree should be a number ending with F. The function returns the Celsius number ending with C. For example, if the input number is 212F, the function returns 100C. If the number is 0F, the function returns -17.7777777777778C.

The regular expression test checks for any number that ends with F. The number of Fahrenheit degree is accessible to your function through the parameter \$1. The function sets the Celsius number based on the Fahrenheit degree passed in a string to the f2c function. f2c then returns the Celsius number. This function approximates Perl's s///e flag.

```
function f2c(x) {
  var s = String(x)
  var test = /(\d+(?:\.\d*)?)F\b/g
  return s.replace
    (test,
      function (str,p1,offset,s) {
        return ((p1-32) * 5/9) + "C";
      }
    )
}
```

## search

Executes the search for a match between a regular expression and this String object.

<i>Method of</i>	<a href="#">String</a>
<i>Implemented in</i>	JavaScript 1.2
<i>ECMA version</i>	ECMA-262 Edition 3

## Syntax

`search(regexp)`

## Parameters

regexp	Name of the regular expression. It can be a variable name or a literal.
--------	---

## Description

If successful, search returns the index of the regular expression inside the string. Otherwise, it returns -1.

When you want to know whether a pattern is found in a string use search (similar to the regular expression test method); for more information (but slower execution) use [match](#) (similar to the regular expression exec method).

## Example

The following example prints a message which depends on the success of the test.

```
function testinput(re, str){
  if (str.search(re) != -1)
    midstring = " contains ";
  else
    midstring = " does not contain ";
  document.write (str + midstring + re.source);
}
```

## slice

Extracts a section of a string and returns a new string.

<i>Method of</i>	<a href="#">String</a>
<i>Implemented in</i>	JavaScript 1.0, NES 2.0
<i>ECMA version</i>	ECMA-262 Edition 3

## Syntax

`slice(beginSlice[, endSlice])`

## Parameters

<code>beginSlice</code>	The zero-based index at which to begin extraction.
<code>endSlice</code>	The zero-based index at which to end extraction. If omitted, slice extracts to the end of the string.

**Description**

`slice` extracts the text from one string and returns a new string. Changes to the text in one string do not affect the other string.

`slice` extracts up to but not including `endSlice`. `string.slice(1,4)` extracts the second character through the fourth character (characters indexed 1, 2, and 3).

As a negative index, `endSlice` indicates an offset from the end of the string. `string.slice(2,-1)` extracts the third character through the second to last character in the string.

**Example**

The following example uses `slice` to create a new string.

```
<SCRIPT>
str1="The morning is upon us. "
str2=str1.slice(3,-5)
document.write(str2)
</SCRIPT>
```

This writes:

morning is upon

**small**

Causes a string to be displayed in a small font, as if it were in a `<SMALL>` tag.

<i>Method of</i>	<a href="#"><u>String</u></a>
<i>Implemented in</i>	JavaScript 1.0, NES 2.0

## Syntax

`small()`

## Parameters

None

## Description

Use the `small` method with the `write` or `writeln` methods to format and display a string in a document. In server-side JavaScript, use the `write` function to display the string.

## Examples

The following example uses string methods to change the size of a string:

```
var worldString="Hello, world"

document.write(worldString.small())
document.write("<P>" + worldString.big())
document.write("<P>" + worldString.fontsize(7))
```

The previous example produces the same output as the following HTML:

```
<SMALL>Hello, world</SMALL>
<P><BIG>Hello, world</BIG>
<P><FONTSIZE=7>Hello, world</FONTSIZE>
```

## See also

[String.big](#), [String.fontsize](#)

## split

Splits a String object into an array of strings by separating the string into substrings.

<i>Method of</i>	<a href="#">String</a>
<i>Implemented in</i>	JavaScript 1.1, NES 2.0
<i>ECMA version</i>	ECMA-262 (if separator is a string) ECMA-262 Edition 3 (if separator is a regular expression)

## Syntax

`split([separator][, limit])`

## Parameters

separator	Specifies the character to use for separating the string. The separator is treated as a string or a regular expression. If separator is omitted, the array returned contains one element consisting of the entire string.
limit	Integer specifying a limit on the number of splits to be found.

## Description

The `split` method returns the new array.

When found, separator is removed from the string and the substrings are returned in an array. If separator is omitted, the array contains one element consisting of the entire string.

In JavaScript 1.2 or later, `split` has the following additions:

- It can take a regular expression argument, as well as a fixed string, by which to split the object string. If separator is a regular expression, any included parenthesis cause submatches to be included in the returned array.
- It can take a limit count so that the resulting array does not include trailing empty elements.
- If you specify LANGUAGE="JavaScript1.2" in the SCRIPT tag, `string.split(" ")` splits on any run of 1 or more white space characters including spaces, tabs, line feeds, and carriage returns. For this behavior, LANGUAGE="JavaScript1.2" must be specified in the <SCRIPT> tag.

## Examples

**Example 1.** The following example defines a function that splits a string into an array of strings using the specified separator. After splitting the string, the function displays messages indicating the original string (before the split), the separator used, the number of elements in the array, and the individual array elements.

```
function splitString (stringToSplit,separator) {
    arrayOfStrings = stringToSplit.split(separator)
    document.write ('<P>The original string is: "' + stringToSplit + '"')
    document.write ('<BR>The separator is: "' + separator + '"')
```

```

document.write ("<BR>The array has " + arrayOfStrings.length + " elements: ")

for (var i=0; i < arrayOfStrings.length; i++) {
    document.write (arrayOfStrings[i] + " / ")
}

var tempestString="Oh brave new world that has such people in it."
var monthString="Jan,Feb,Mar,Apr,May,Jun,Jul,Aug,Sep,Oct,Nov,Dec"

var space=" "
var comma=","

splitString(tempestString,space)
splitString(tempestString)
splitString(monthString,comma)

```

This example produces the following output:

The original string is: "Oh brave new world that has such people in it."  
 The separator is: " "  
 The array has 10 elements: Oh / brave / new / world / that / has / such / people / in / it. /

The original string is: "Oh brave new world that has such people in it."  
 The separator is: "undefined"  
 The array has 1 elements: Oh brave new world that has such people in it. /  
  
 The original string is: "Jan,Feb,Mar,Apr,May,Jun,Jul,Aug,Sep,Oct,Nov,Dec"  
 The separator is: ","
 The array has 12 elements: Jan / Feb / Mar / Apr / May / Jun / Jul / Aug / Sep / Oct / Nov / Dec /

**Example 2.** Consider the following script:

```

<SCRIPT LANGUAGE="JavaScript1.2">
str="She sells seashells \nby the\n seashore"
document.write(str + "<BR>")
a=str.split(" ")
document.write(a)
</SCRIPT>

```

Using LANGUAGE="JavaScript1.2", this script produces

"She", "sells", "seashells", "by", "the", "seashore"

Without LANGUAGE="JavaScript1.2", this script splits only on single space characters, producing

"She", "sells", , , , "seashells", "by", , , "the", "seashore"

**Example 3.** In the following example, split looks for 0 or more spaces followed by a semicolon followed by 0 or more spaces and, when found, removes the spaces from the string. nameList is the array returned as a result of split.

```

<SCRIPT>
names = "Harry Trump ;Fred Barney; Helen Rigby ; Bill Abel ;Chris Hand ";

```

```
document.write (names + "<BR>" + "<BR>");
re = /\s*;\s*/;
nameList = names.split (re);
document.write(nameList);
</SCRIPT>
```

This prints two lines; the first line prints the original string, and the second line prints the resulting array.

```
Harry Trump ;Fred Barney; Helen Rigby ; Bill Abel ;Chris Hand
Harry Trump,Fred Barney,Helen Rigby,Bill Abel,Chris Hand
```

**Example 4.** In the following example, split looks for 0 or more spaces in a string and returns the first 3 splits that it finds.

```
<SCRIPT LANGUAGE="JavaScript1.2">
myVar = " Hello World. How are you doing? ";
splits = myVar.split(" ", 3);
document.write(splits)
</SCRIPT>
```

This script displays the following:

```
["Hello", "World.", "How"]
```

## See also

[String.charAt](#), [String.indexOf](#), [String.lastIndexOf](#)

## strike

Causes a string to be displayed as struck-out text, as if it were in a <STRIKE> tag.

<i>Method of</i>	<a href="#">String</a>
<i>Implemented in</i>	JavaScript 1.0, NES 2.0

## Syntax

```
strike()
```

## Parameters

None

## Description

Use the `strike` method with the `write` or `writeln` methods to format and display a string in a document. In server-side JavaScript, use the `write` function to display the string.

## Examples

The following example uses string methods to change the formatting of a string:

```
var worldString="Hello, world"

document.write(worldString.blink())
document.write("<P>" + worldString.bold())
document.write("<P>" + worldString.italics())
document.write("<P>" + worldString.strike())
```

The previous example produces the same output as the following HTML:

```
<BLINK>Hello, world</BLINK>
<P><B>Hello, world</B>
<P><I>Hello, world</I>
<P><STRIKE>Hello, world</STRIKE>
```

## See also

[String.blink](#), [String.bold](#), [String.italics](#)

## sub

Causes a string to be displayed as a subscript, as if it were in a `<SUB>` tag.

<i>Method of</i>	<a href="#">String</a>
<i>Implemented in</i>	JavaScript 1.0, NES 2.0

## Syntax

`sub()`

## Parameters

None

## Description

Use the `sub` method with the `write` or `writeln` methods to format and display a string in a document. In server-side JavaScript, use the `write` function to generate the HTML.

## Examples

The following example uses the sub and sup methods to format a string:

```
var superText="superscript"
var subText="subscript"

document.write("This is what a " + superText.sup() + " looks like.")
document.write("<P>This is what a " + subText.sub() + " looks like.")
```

The previous example produces the same output as the following HTML:

```
This is what a <SUP>superscript</SUP> looks like.
<P>This is what a <SUB>subscript</SUB> looks like.
```

## See also

[String.sup](#)

## substr

Returns the characters in a string beginning at the specified location through the specified number of characters.

<i>Method of</i>	<a href="#">String</a>
<i>Implemented in</i>	JavaScript 1.0, NES 2.0

## Syntax

`substr(start[, length])`

## Parameters

<i>start</i>	Location at which to begin extracting characters.
<i>length</i>	The number of characters to extract.

## Description

start is a character index. The index of the first character is 0, and the index of the last character is 1 less than the length of the string. substr begins extracting characters at start and collects length number of characters.

If start is positive and is the length of the string or longer, substr returns no characters.

If start is negative, substr uses it as a character index from the end of the string. If start is negative and abs(start) is larger than the length of the string, substr uses 0 is the start index.

If length is 0 or negative, substr returns no characters. If length is omitted, start extracts characters to the end of the string.

## Example

Consider the following script:

```
<SCRIPT LANGUAGE="JavaScript1.2">

str = "abcdefghijklm"
document.writeln("(1,2): ", str.substr(1,2))
document.writeln("(-2,2): ", str.substr(-2,2))
document.writeln("(1): ", str.substr(1))
document.writeln("(-20, 2): ", str.substr(1,20))
document.writeln("(20, 2): ", str.substr(20,2))

</SCRIPT>
```

This script displays:

```
(1,2): bc
(-2,2): ij
(1): bcdefghij
(-20, 2): bcdefghij
(20, 2):
```

## See also

[substring](#)

## substring

Returns a subset of a String object.

*Method of*

[String](#)

<i>Implemented in</i>	JavaScript 1.0, NES 2.0
<i>ECMA version</i>	ECMA-262

## Syntax

`substring(indexA, indexB)`

## Parameters

<code>indexA</code>	An integer between 0 and 1 less than the length of the string.
<code>indexB</code>	An integer between 0 and 1 less than the length of the string.

## Description

`substring` extracts characters from `indexA` up to but not including `indexB`. In particular:

- If `indexA` is less than 0, `indexA` is treated as if it were 0.
- If `indexB` is greater than `stringName.length`, `indexB` is treated as if it were `stringName.length`.
- If `indexA` equals `indexB`, `substring` returns an empty string.
- If `indexB` is omitted, `indexA` extracts characters to the end of the string.

In JavaScript 1.2, using `LANGUAGE="JavaScript1.2"` in the `SCRIPT` tag,

- If `indexA` is greater than `indexB`, JavaScript produces a runtime error (out of memory).

In JavaScript 1.2, without `LANGUAGE="JavaScript1.2"` in the `SCRIPT` tag,

- If `indexA` is greater than `indexB`, JavaScript returns a substring beginning with `indexB` and ending with `indexA - 1`.

## Examples

**Example 1.** The following example uses substring to display characters from the string "Netscape":

```
var anyString="Netscape"

// Displays "Net"
document.write(anyString.substring(0,3))
document.write(anyString.substring(3,0))
// Displays "cap"
document.write(anyString.substring(4,7))
document.write(anyString.substring(7,4))
// Displays "Netscap"
document.write(anyString.substring(0,7))
// Displays "Netscape"
document.write(anyString.substring(0,8))
document.write(anyString.substring(0,10))
```

**Example 2.** The following example replaces a substring within a string. It will replace both individual characters and substrings. The function call at the end of the example changes the string "Brave New World" into "Brave New Web".

```
function replaceString(oldS,newS,fullS) {
// Replaces oldS with newS in the string fullS
    for (var i=0; i<fullS.length; i++) {
        if (fullS.substring(i,i+oldS.length) == oldS) {
            fullS = fullS.substring(0,i)+newS+fullS.substring(i+oldS.length,fullS.length)
        }
    }
    return fullS
}

replaceString("World","Web","Brave New World")
```

**Example 3.** In JavaScript 1.2, using LANGUAGE="JavaScript1.2", the following script produces a runtime error (out of memory).

```
<SCRIPT LANGUAGE="JavaScript1.2">
str="Netscape"
document.write(str.substring(0,3));
document.write(str.substring(3,0));
</SCRIPT>
```

Without LANGUAGE="JavaScript1.2", the above script prints the following:

Net Net

In the second write, the index numbers are swapped.

## See also

[substr](#)

**sup**

Causes a string to be displayed as a superscript, as if it were in a <SUP> tag.

<i>Method of</i>	<a href="#">String</a>
<i>Implemented in</i>	JavaScript 1.0, NES 2.0

## Syntax

`sup()`

## Parameters

None

## Description

Use the sup method with the write or writeln methods to format and display a string in a document. In server-side JavaScript, use the write function to generate the HTML.

## Examples

The following example uses the sub and sup methods to format a string:

```
var superText="superscript"
var subText="subscript"

document.write("This is what a " + superText.sup() + " looks like.")
document.write("<P>This is what a " + subText.sub() + " looks like.")
```

The previous example produces the same output as the following HTML:

```
This is what a <SUP>superscript</SUP> looks like.
<P>This is what a <SUB>subscript</SUB> looks like.
```

## See also

[String.sub](#)

## toLowerCase

Returns the calling string value converted to lowercase.

<i>Method of</i>	<a href="#">String</a>
<i>Implemented in</i>	JavaScript 1.0, NES 2.0
<i>ECMA version</i>	ECMA-262

**Syntax**`toLowerCase()`**Parameters**

None

**Description**

The `toLowerCase` method returns the value of the string converted to lowercase. `toLowerCase` does not affect the value of the string itself.

**Examples**

The following example displays the lowercase string "alphabet":

```
var upperText="ALPHABET"
document.write(upperText.toLowerCase())
```

**See also**[String.toUpperCase](#)**toSource**

Returns a string representing the source code of the object.

<i>Method of</i>	<a href="#">String</a>
<i>Implemented in</i>	JavaScript 1.3

**Syntax**`toSource()`

**Parameters**

None

**Description**

The `toSource` method returns the following values:

- For the built-in `String` object, `toSource` returns the following string indicating that the source code is not available:

```
function String() {
    [native code]
}
```

- For instances of `String` or string literals, `toSource` returns a string representing the source code.

This method is usually called internally by JavaScript and not explicitly in code.

**toString**

Returns a string representing the specified object.

<i>Method of</i>	<a href="#">String</a>
<i>Implemented in</i>	JavaScript 1.1, NES 2.0
<i>ECMA version</i>	ECMA-262

**Syntax**

`toString()`

**Parameters**

None.

**Description**

The [String](#) object overrides the `toString` method of the [Object](#) object; it does not inherit [Object.toString](#). For `String` objects, the `toString` method returns a string representation of the object.

## Examples

The following example displays the string value of a String object:

```
x = new String("Hello world");
alert(x.toString()) // Displays "Hello world"
```

## See also

[Object.toString](#)

## toUpperCase

Returns the calling string value converted to uppercase.

<i>Method of</i>	<a href="#">String</a>
<i>Implemented in</i>	JavaScript 1.0, NES 2.0
<i>ECMA version</i>	ECMA-262

## Syntax

`toUpperCase()`

## Parameters

None

## Description

The `toUpperCase` method returns the value of the string converted to uppercase. `toUpperCase` does not affect the value of the string itself.

## Examples

The following example displays the string "ALPHABET":

```
var lowerText="alphabet"
document.write(lowerText.toUpperCase())
```

## See also

[String.toLowerCase](#)

**valueOf**

Returns the primitive value of a String object.

<i>Method of</i>	<a href="#">String</a>
<i>Implemented in</i>	JavaScript 1.1
<i>ECMA version</i>	ECMA-262

**Syntax**

`valueOf()`

**Parameters**

None

**Description**

The `valueOf` method of [String](#) returns the primitive value of a String object as a string data type. This value is equivalent to [String.toString](#).

This method is usually called internally by JavaScript and not explicitly in code.

**Examples**

```
x = new String("Hello world");
alert(x.valueOf())      // Displays "Hello world"
```

**See also**

[String.toString](#), [Object.valueOf](#)

[Previous](#) [Contents](#) [Index](#) [Next](#)

---

Copyright © 2000 [Netscape Communications Corp](#). All rights reserved.

Last Updated **September 28, 2000**