

JS Lib

Return Types

[Array](#) [Boolean](#) [Date](#) [Function](#) [Math](#) [Namespace](#) [Number](#) [Object](#) [QName](#) [RegExp](#) [Special Operators](#) [Statements](#) [String](#) [XML](#) [XMLList](#)

Property Summary

[Number](#) [#Infinity](#)
Numeric value representing infinity.

[Number](#) [#NaN](#)
Value representing Not-a-Number.

[Object](#) [#undefined](#)
The value undefined.

Method Summary

[String](#) [#decodeURI\(\)](#)
Decodes a URI previously encoded with encodeURI or another similar routine.

[String](#) [#decodeURIComponent\(encodedURI\)](#)
Decodes a URI component previously created by encodeURIComponent or by a similar routine.

[String](#) [#encodeURI\(\)](#)
Encodes a URI by replacing certain characters with escape sequences.

[String](#) [#encodeURIComponent\(\)](#)
Encodes a URI component by replacing all special characters with their corresponding UTF-8 escape sequences.

[Object](#) [#eval\(\)](#)
Evaluates JavaScript code passed as a string.

[Boolean](#) [#isFinite\(\)](#)
Returns true if the given number is a finite number.

[void](#) [#isNaN\(\)](#)
The NaN property indicates that a value is 'Not a Number'.

[Boolean](#) [#isXMLName\(\)](#)
Returns true if the given name can be used as a valid name for an XML element or attribute.

[Number](#) [#parseFloat\(\)](#)
Makes a floating point number from the starting numbers in a given string.

[Number](#) [#parseInt\(\)](#)
Makes an integer from the starting numbers in a given string in the base specified.

[String](#) [#uneval\(\)](#)
Returns the string representation behind a given object.

Property Details

Infinity

Numeric value representing infinity.

Returns

[Number](#)

Sample

```
Infinity
```

NaN

Value representing Not-a-Number.

Returns

[Number](#)

Sample

```
NaN
```

undefined

The value undefined.

Returns

[Object](#)

Sample

```
undefined
```

Method Details

[decodeURI](#)

[String](#) **decodeURI()**

Decodes a URI previously encoded with [encodeURIComponent](#) or another similar routine.

Returns

[String](#)

Sample

```
var str = "http://www.mysite.com/my code.asp?name=[cool]";
var encoded = encodeURIComponent(str);
var decoded = decodeURI(encoded);
application.output(encoded); //http://www.mysite.com/my%20code.asp?name=%5bcool%5d
application.output(decoded); //http://www.mysite.com/my code.asp?name=[cool]
```

[decodeURIComponent](#)

[String](#) **decodeURIComponent(encodedURI)**

Decodes a URI component previously created by [encodeURIComponent](#) or by a similar routine.

Parameters

{[String](#)} encodedURI

Returns

[String](#)

Sample

```
var str = "my code.asp?name=[cool]";
var encoded = encodeURIComponent(str);
var decoded = decodeURIComponent(encoded);
application.output(encoded); //my%20code.asp%3fname%3d%5bcool%5d
application.output(decoded); //my code.asp?name=[cool]
```

[encodeURIComponent](#)

[String](#) **encodeURIComponent()**

Encodes a URI by replacing certain characters with escape sequences.

Returns

[String](#)

Sample

```
var str = "http://www.mysite.com/my code.asp?name=[cool]";
var encoded = encodeURI(str);
var decoded = decodeURI(encoded);
application.output(encoded); //http://www.mysite.com/my%20code.asp?name=%5bcool%5d
application.output(decoded); //http://www.mysite.com/my code.asp?name=[cool]
```

[encodeURIComponent](#)

[String](#) **encodeURIComponent()**

Encodes a URI component by replacing all special characters with their corresponding UTF-8 escape sequences.

Returns

[String](#)

Sample

```
var str = "my code.asp?name=[cool]";
var encoded = encodeURIComponent(str);
var decoded = decodeURIComponent(encoded);
application.output(encoded); //my%20code.asp%3fname%3d%5bcool%5d
application.output(decoded); //my code.asp?name=[cool]
```

[eval](#)

[Object](#) **eval()**

Evaluates JavaScript code passed as a string. Returns the value returned by the evaluated code.

Returns

[Object](#)

Sample

```
eval("var x = 2 + 3;");
application.output(x); // prints: 5.0
```

[isFinite](#)

[Boolean](#) **isFinite()**

Returns true if the given number is a finite number.

Returns

[Boolean](#)

Sample

```
application.output(isFinite(1)); // prints: true
application.output(isFinite(Infinity)); // prints: false
application.output(isFinite(isNaN)); // prints: false
```

[isNaN](#)

void **isNaN()**

The NaN property indicates that a value is 'Not a Number'.

Returns

void

Sample

```
isNaN( value )
```

[isXMLName](#)

[Boolean](#) **isXMLName()**

Returns true if the given name can be used as a valid name for an XML element or attribute.

Returns

[Boolean](#)

Sample

```
application.output(isXMLName("good_name")); // prints: true
application.output(isXMLName("bad name")); // because of the space, prints: false
```

parseFloat

[Number](#) **parseFloat()**

Makes a floating point number from the starting numbers in a given string.

Returns

[Number](#)

Sample

```
parseFloat('string')
```

parseInt

[Number](#) **parseInt()**

Makes an integer from the starting numbers in a given string in the base specified.

Returns

[Number](#)

Sample

```
parseInt( 'string' [, base] )
```

uneval

[String](#) **uneval()**

Returns the string representation behind a given object.

Returns

[String](#)

Sample

```
application.output(uneval(isNaN)); // prints something like: function isNaN() { [native code for isNaN,
arity=1] }
```