

# JS Lib

## Return Types

[Array](#) [Boolean](#) [Date](#) [Function](#) [JSON](#) [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\(encodedURI\)](#)  
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\(URI\)](#)  
Encodes a URI by replacing certain characters with escape sequences.

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

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

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

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

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

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

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

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

[String](#) [#uneval\(obj\)](#)  
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**(encodedURI)

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

### Parameters

[String](#) encodedURI

### 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(**URI**)

Encodes a URI by replacing certain characters with escape sequences.

**Parameters**

{**String**} **URI**

**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]
```

encodeURIComponentComponent

**String** encodeURIComponentComponent(**URI**)

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

**Parameters**

{**String**} **URI**

**Returns**

**String**

**Sample**

```
var str = "my code.asp?name=[cool]";
var encoded = encodeURIComponentComponent(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(**expression**)

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

**Parameters**

{**String**} **expression**

**Returns**

**Object**

**Sample**

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

isFinite

**Boolean** isFinite(**n**)

Returns true if the given number is a finite number.

**Parameters**

{**Number**} **n**

**Returns**

**Boolean**

**Sample**

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

isNaN

**void** isNaN(**value**)

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

**Parameters**

{**Object**} **value**

**Returns**

**void**

### Sample

```
isNaN( value )
```

isXMLName

**Boolean** **isXMLName**(name)

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

**Parameters**

{String} name

**Returns**

Boolean

### Sample

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

parseFloat

**Number** **parseFloat**(text)

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

**Parameters**

{String} text

**Returns**

Number

### Sample

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

parseInt

**Number** **parseInt**(text)

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

**Parameters**

{String} text

**Returns**

Number

### Sample

```
parseInt('0774')
```

parseInt

**Number** **parseInt**(text, radix)

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

**Parameters**

{String} text

{Number} radix

**Returns**

Number

### Sample

```
parseInt('0774' , 8)
```

uneval

**String** **uneval**(obj)

Returns the string representation behind a given object.

**Parameters**

{Object} obj

**Returns**

String

**Sample**

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