XML

Creating XML objects

from a string

```
var myXml = new XML('<test level="1"><test2 level="2">A value</test2></test>');
```

using XML object notation

```
var myXml =
<test level="1">
  <test2 level="2">A value</test2>
</test>;
```

using JavaScript variables

```
var myValue = 'hello, some value!';
var myXml =
<test level="1">
  <test2 level="2">{myValue}</test2>
</test>;
```

Accessing the XML

General

```
var myXml = new XML('<test level="1"><test2 level="2">A value</test2></test>');
application.output(myXml.test2); //outputs 'A value'
```

A specific index on subnodes

```
var myXml = new XML('<test level="1"><test2 level="2">A value</test2><test2 level=2>Another value</test2><
/test>');
application.output(myXml.test2[1]); //outputs 'Another value'
```

XML in Action!

Property Summary

Boolean	#ignoreComments If set to true, then comments in the XML are ignored when constructing new XML objects.
Boolean	#ignoreProcessingInstructions If set to true, then processing instructions are ignored when constructing new XML objects.
Boolean	#ignoreWhitespace If set to true, then whitespace in the XML is ignored when constructing new XML objects.
Boolean	#prettyIndent The amount of positions used when indenting child nodes are relative to their parent if prettyPrinting is enabled.
Boolean	#prettyPrinting If set to true, then toString() and toXMLString() methods will normalize the output to achieve a uniform appearance.

Method Summary

XML	#addNamespace(namespaceToAdd) Takes one argument which can be a string with a namespace URI or a Namespace object and adds the argument to the in scope namespaces of this XML object.
XML	#appendChild(childToAppend) Appends a new child at the end of this XML object's properties, the changed XML object is then returned.

XMLList	#attribute(attributeName) Takes a single argument with the attribute name and returns an XMLList with attributes matching the argument.
XMLList	#attributes() Returns an XMLList with the attributes of this XML object which are in no namespace.
XMLList	#child(propertyName) Returns an XMLList with children matching the property name.
Number	#childIndex() If the XML object has no parent then the special number NaN is returned, otherwise the ordinal position the object has in the context of its parent is returned.
XMLList	#children() Returns an XMLList with the child nodes of this XML object.
XMLList	#comments() Returns an XMLList with the comment nodes which are children of this XML object.
Boolean	#contains(value) Calling xmlObject.
XML	#copy() Returns a deep copy of the XML object it is called on where the internal parent property is set to null
Object	#defaultSettings() Returns an object containing the default XML settings.
XMLList	#descendants([name]) Returns an XMLList with the descendants matching the passed name argument or with all descendants if no argument is passed.
XMLList	#elements([name]) Takes one optional argument, the name of elements you are looking for, and returns an XMLList with all matching child elements.
Boolean	#hasComplexContent() Returns false for XML objects of node kind 'text', 'attribute', 'comment', and 'processing-instruction'.
Boolean	#hasOwnProperty(propertyName) Returns true if the XML object the method is called on has a property of that name.
Boolean	#hasSimpleContent() Returns true for XML objects of node kind text or attribute.
Array	#inScopeNamespaces() Returns an array of Namespace objects representing the namespace that are in scope for this XML object.
XML	#insertChildAfter(childToInserAfter, childToInsert) Takes two arguments, an existing child to insert after and the new child to be inserted.
XML	#insertChildBefore(childToInsertBefore, childToInsert) Takes two arguments, an existing child to insert before and the new child to be inserted.
Number	#length() This always returns 1.
String	#localName() returns the local name part if the XML object has a name.
QName	#name() Returns the qualified name (a QName object) of the XML object it is called
Namespace	#namespace([prefix]) If no argument is passed to the method then it returns the namespace associated with the qualified name of this XML object.
Array	#namespaceDeclarations() Returns an array with the namespace declarations associated with the XML object it is called on.
String	#nodeKind() Returns a string denoting the kind of node this XML object represents.
XML	#normalize() Returns this XML object after normalizing all text content.
XML	#parent() Returns the parent XML object of this XML object or null if there is no parent.
XML	#prependChild(childToPrepend) Iinserts the given value as the first child of the XML object and returns the XML object.
XMLList	#processingInstructions([name]) If no argument is passed in then the method returns an XMLList with all the children of the XML object which are processing instructions.
Boolean	#propertyIsEnumerable(propertyName) Returns true if the property name is '0' and false otherwise.
XML	#removeNamespace(namespace) Removes the namespace from the in scope namespaces of this XML object if the namespace is not used for the qualified name of the object or its attributes.
XML	#replace(propertyName, replacementValue) Takes two arguments, the property name of the property / properties to be replaced, and the value to replace the properties.
XML	#setChildren(value) Replaces all children of the XML object with this value.
void	#setLocalName(name) Changes the local name of this XML object to the name passed in.
void	#setName(name) Replaces the name of this XML object with the name passed in.

void	#setNamespace(namespace) Changes the namespace associated with the name of this XML object to the new namespace.
void	#setSettings([settings]) Allows the global XML settings to be adjusted or restored to their default values.
Object	#settings() Returns an object containing the global XML settings.
XMLList	#text() Returns an XMLList with all the children of this XML object that represent text nodes.
String	#toString() Returns a convenient string value of this XML object.
String	#toXMLString() Returns a string with the serialized XML markup for this XML object.
XML	#valueOf() The method simply returns the XML object it is called on.

Property Details

ignoreComments

If set to true, then comments in the XML are ignored when constructing new XML objects.

Returns
Boolean

Sample

```
var element = <foo><!-- my comment --><bar/></foo>;
application.output(element.comments().length());
application.output(element.toXMLString());

XML.ignoreComments = false;

element = <foo><!-- my comment --><bar/></foo>;
application.output(element.comments().length());
application.output(element.toXMLString());
```

ignoreProcessingInstructions

If set to true, then processing instructions are ignored when constructing new XML objects.

Returns
Boolean

Sample

```
XML.ignoreProcessingInstructions=false;
var xmlElement = <publishing><?process author="yes"?><author type="leadership">John C. Maxwell</author><
/publishing>;
application.output(" Element = "+ xmlElement.toXMLString());
```

ignoreWhitespace

If set to true, then whitespace in the XML is ignored when constructing new XML objects.

Returns
Boolean

Sample

```
XML.ignoreWhitespace = false;
 var xmlElement =
 <publishing>
        <author>John C. Maxwell</author>
 </publishing>;
 application.output(xmlElement.toString());
```

prettyIndent

The amount of positions used when indenting child nodes are relative to their parent if prettyPrinting is enabled.

Returns

Boolean

Sample

```
var xmlElement = <publishing><author>Tom DeMarco</author><author>Roger S. Pressman</author></publishing>;
application.output(xmlElement.toXMLString());
XML.prettyPrinting = true;
XML.prettyIndent = 4;
xmlElement = <publishing><author>Tom DeMarco</author><author>Roger S. Pressman</author></publishing>;
application.output(xmlElement.toXMLString());
```

prettyPrinting

If set to true, then toString() and toXMLString() methods will normalize the output to achieve a uniform appearance.

Returns

Boolean

Sample

```
var xmlElement = <publishing><author>Tom DeMarco</author><author>Roger S. Pressman</author></publishing>;
application.output(xmlElement.toXMLString());
XML.prettyPrinting = true;
XML.prettyIndent = 4;
xmlElement = <publishing><author>Tom DeMarco</author><author>Roger S. Pressman</author></publishing>;
application.output(xmlElement.toXMLString());
```

Method Details

addNamespace

XML **addNamespace**(namespaceToAdd)

Takes one argument which can be a string with a namespace URI or a Namespace object and adds the argument to the in scope namespaces of this XML object.

Parameters

{String} namespaceToAdd

Returns

XML

Sample

```
xml.addNamespace(namespaceToAdd)
```

appendChild

XML **appendChild**(childToAppend)

Appends a new child at the end of this XML object's properties, the changed XML object is then returned.

Parameters

{XML} childToAppend

Returns

XML

Sample

```
xml.appendChild(childToAppend)
```

attribute

XMLList **attribute**(attributeName)

Takes a single argument with the attribute name and returns an XMLList with attributes matching the argument.

Parameters

{String} attributeName

Returns

XMLList

Sample

```
xml.attribute(attributeName)
```

attributes

XMLList **attributes**()

Returns an XMLList with the attributes of this XML object which are in no namespace.

Returns

XMLList

Sample

```
xml.attributes()
```

child

XMLList **child**(propertyName)

Returns an XMLList with children matching the property name.

Parameters

{String} propertyName

Returns

XMLList

Sample

```
xml.child(childPropertyName)
```

childIndex

Number **childIndex**()

If the XML object has no parent then the special number NaN is returned, otherwise the ordinal position the object has in the context of its parent is returned.

Returns

Number

Sample

```
xml.childIndex()
```

children

XMLList **children**()

Returns an XMLList with the child nodes of this XML object.

Returns

XMLList

Sample

```
xml.children()
```

comments

XMLList **comments**()

Returns an XMLList with the comment nodes which are children of this XML object.

Returns

XMLList

Sample

```
xml.comments()
```

contains

Boolean **contains**(value)

Calling xmlObject.contains(value) yields the same result as the equality comparison xmlObject == value

Parameters

{Object} value

Returns

Boolean

Sample

```
xml.contains(value)
```

copy

XML **copy**()

Returns a deep copy of the XML object it is called on where the internal parent property is set to null

Returns

XML

Sample

```
xml.copy()
```

defaultSettings

Object **defaultSettings**()

Returns an object containing the default XML settings.

Returns

Object

Sample

```
xml.defaultSettings()
```

descendants

XMLList **descendants**([name])

Returns an XMLList with the descendants matching the passed name argument or with all descendants if no argument is passed.

Parameters

{String} [name]

Returns

XMLList

Sample

```
xml.descendants([name])
```

elements

XMLList **elements**([name])

Takes one optional argument, the name of elements you are looking for, and returns an XMLList with all matching child elements.

Parameters

{String} [name]

Returns

XMLList

Sample

```
xml.elements([name])
```

hasComplexContent

Boolean **hasComplexContent**()

Returns false for XML objects of node kind 'text', 'attribute', 'comment', and 'processing-instruction'. For objects of kind 'element' it checks whether the element has at least one child element.

Returns

Boolean

Sample

```
xml.hasComplexContent()
```

hasOwnProperty

Boolean **hasOwnProperty**(propertyName)

Returns true if the XML object the method is called on has a property of that name.

Parameters
{String} propertyName
Returns
Boolean
Sample

```
xml.hasOwnProperty(propertyName)
```

hasSimpleContent

Boolean **hasSimpleContent**()

Returns true for XML objects of node kind text or attribute. For XML objects of node kind element it returns true if the element has no child elements and false otherwise.
For other node kinds (comment, processing instruction) the method always returns false.

Returns
Boolean
Sample

```
xml.hasSimpleContent()
```

inScopeNamespaces

Array **inScopeNamespaces**()

Returns an array of Namespace objects representing the namespace that are in scope for this XML object.

Returns
Array
Sample

```
xml.inScopeNamespaces()
```

insertChildAfter

XML **insertChildAfter**(childToInserAfter, childToInsert)

Takes two arguments, an existing child to insert after and the new child to be inserted.
If the first argument is null then the second argument is inserted as the first child of this XML.

Parameters
{XML} childToInserAfter
{XML} childToInsert
Returns
XML
Sample

```
xml.insertChildAfter(childToInsertAfter, childToInsert)
```

insertChildBefore

XML **insertChildBefore**(childToInsertBefore, childToInsert)

Takes two arguments, an existing child to insert before and the new child to be inserted.
If the first argument is null then the child is inserted as the last child.

Parameters
{XML} childToInsertBefore
{XML} childToInsert
Returns
XML
Sample

```
xml.insertChildBefore(childToInsertBefore, childToInsert)
```

length

Number **length**()

This always returns 1. This is done to blur the distinction between an XML object and an XMLList containing exactly one value.

Returns
Number

Sample

```
xml.length()
```

localName

String **localName**()

returns the local name part if the XML object has a name.

Returns

String

Sample

```
xml.localName()
```

name

QName **name**()

Returns the qualified name (a QName object) of the XML object it is called

Returns

QName

Sample

```
xml.name()
```

namespace

Namespace **namespace**([prefix])

If no argument is passed to the method then it returns the namespace associated with the qualified name of this XML object. If a prefix is passed to the method then it looks for a matching namespace in the in scope namespace of this XML object and returns it when found, otherwise undefined is returned.

Parameters

{String} [prefix]

Returns

Namespace

Sample

```
xml.namespace([prefix])
```

namespaceDeclarations

Array **namespaceDeclarations**()

Returns an array with the namespace declarations associated with the XML object it is called on.

Returns

Array

Sample

```
xml.namespaceDeclarations()
```

nodeKind

String **nodeKind**()

Returns a string denoting the kind of node this XML object represents. Possible values: 'element', 'attribute', 'text', 'comment', 'processing-instruction'.

Returns

String

Sample

```
xml.nodeKind()
```

normalize

XML **normalize**()

Returns this XML object after normalizing all text content.

Returns

XML

Sample

```
xml.normalize()
```

parent

XML **parent**()

Returns the parent XML object of this XML object or null if there is no parent.

Returns

XML

Sample

```
xml.parent()
```

prependChild

XML **prependChild**(childToPrepend)

Iinserts the given value as the first child of the XML object and returns the XML object.

Parameters

{XML} childToPrepend

Returns

XML

Sample

```
xml.prependChild(childToPrepend)
```

processingInstructions

XMLList **processingInstructions**([name])

If no argument is passed in then the method returns an XMLList with all the children of the XML object which are processing instructions. If an argument is passed in then the method returns an XMLList with all children of the XML object which are processing instructions where the name matches the argument.

Parameters

{String} [name]

Returns

XMLList

Sample

```
xml.processingInstructions([name])
```

propertyIsEnumerable

Boolean **propertyIsEnumerable**(propertyName)

Returns true if the property name is '0' and false otherwise.

Parameters

{String} propertyName

Returns

Boolean

Sample

```
xml.propertyIsEnumerable(propertyName)
```

removeNamespace

XML **removeNamespace**(namespace)

Removes the namespace from the in scope namespaces of this XML object if the namespace is not used for the qualified name of the object or its attributes.

Parameters

{Namespace} namespace

Returns

XML

Sample

```
xml.removeNamespace(namespace)
```

replace

XML **replace**(propertyName, replacementValue)

Takes two arguments, the property name of the property / properties to be replaced, and the value to replace the properties.

Parameters

{String} propertyName
{XML} replacementValue

Returns

XML

Sample

```
xml.replace(propertyName, replacementValue)
```

setChildren

XML **setChildren**(value)

Replaces all children of the XML object with this value. The method returns the XML object it is called on.

Parameters

{Object} value

Returns

XML

Sample

```
xml.setChildren(value)
```

setLocalName

void **setLocalName**(name)

Changes the local name of this XML object to the name passed in.

Parameters

{String} name

Returns

void

Sample

```
xml.setLocalName(name)
```

setName

void **setName**(name)

Replaces the name of this XML object with the name passed in.

Parameters

{String} name

Returns

void

Sample

```
xml.setName(name)
```

setNamespace

void **setNamespace**(namespace)

Changes the namespace associated with the name of this XML object to the new namespace.

Parameters

{Namespace} namespace

Returns

void

Sample

```
xml.setNamespace(namespace)
```

setSettings

void **setSettings**([settings])

Allows the global XML settings to be adjusted or restored to their default values.

Parameters

{Object} [settings] – The new settings that should be applied globally to the XML object.

Returns

void

Sample

```
xml.setSettings(settings)
```

settings

Object **settings**()

Returns an object containing the global XML settings.

Returns

Object

Sample

```
xml.settings()
```

text

XMLList **text**()

Returns an XMLList with all the children of this XML object that represent text nodes.

Returns

XMLList

Sample

```
xml.text()
```

toString

String **toString**()

Returns a convenient string value of this XML object.

Returns

String

Sample

```
xml.toString()
```

toXMLString

String **toXMLString**()

Returns a string with the serialized XML markup for this XML object. XML.prettyPrinting and XML.prettyIndent settings affect the returned string.

Returns

String

Sample

```
xml.toXMLString()
```

valueOf

XML **valueOf**()

The method simply returns the XML object it is called on.

Returns

XML

Sample

```
xml.valueOf()
```