
NSXMLNode Class Reference

Data Management: Data Types & Collections



2009-10-19



Apple Inc.
© 2009 Apple Inc.
All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, mechanical, electronic, photocopying, recording, or otherwise, without prior written permission of Apple Inc., with the following exceptions: Any person is hereby authorized to store documentation on a single computer for personal use only and to print copies of documentation for personal use provided that the documentation contains Apple's copyright notice.

The Apple logo is a trademark of Apple Inc.

Use of the "keyboard" Apple logo (Option-Shift-K) for commercial purposes without the prior written consent of Apple may constitute trademark infringement and unfair competition in violation of federal and state laws.

No licenses, express or implied, are granted with respect to any of the technology described in this document. Apple retains all intellectual property rights associated with the technology described in this document. This document is intended to assist application developers to develop applications only for Apple-labeled computers.

Every effort has been made to ensure that the information in this document is accurate. Apple is not responsible for typographical errors.

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

Apple, the Apple logo, Cocoa, Mac, and Mac OS are trademarks of Apple Inc., registered in the United States and other countries.

Numbers is a trademark of Apple Inc.

Simultaneously published in the United States and Canada.

Even though Apple has reviewed this document, APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS DOCUMENT IS PROVIDED "AS IS," AND YOU, THE READER, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.

IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR

CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT OR INACCURACY IN THIS DOCUMENT, even if advised of the possibility of such damages.

THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS OR IMPLIED. No Apple dealer, agent, or employee is authorized to make any modification, extension, or addition to this warranty.

Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.

Contents

NSXMLNode Class Reference 5

Overview	5
Subclassing Notes	5
Adopted Protocols	6
Tasks	6
Creating and Initializing Node Objects	6
Managing XML Node Objects	7
Navigating the Tree of Nodes	8
Emitting Node Content	9
Executing Queries	9
Managing Namespaces	9
Class Methods	9
attributeWithName:stringValue:	9
attributeWithName:URI:stringValue:	10
commentWithStringValue:	11
document	11
documentWithRootElement:	11
DTDNodeWithXMLString:	12
elementWithName:	12
elementWithName:children:attributes:	13
elementWithName:stringValue:	13
elementWithName:URI:	14
localNameForName:	14
namespaceWithName:stringValue:	15
predefinedNamespaceForPrefix:	15
prefixForName:	16
processingInstructionWithName:stringValue:	16
textWithStringValue:	17
Instance Methods	17
canonicalXMLStringPreservingComments:	17
childAtIndex:	18
childCount	19
children	19
description	19
detach	20
index	20
initWithKind:	21
initWithKind:options:	21
kind	22
level	22
localName	23

- name 23
- nextNode 24
- nextSibling 24
- nodesForXPath:error: 25
- objectsForXQuery:constants:error: 25
- objectsForXQuery:error: 26
- objectValue 27
- parent 27
- prefix 27
- previousNode 28
- previousSibling 28
- rootDocument 29
- setName: 29
- setObjectValue: 30
- setStringValue: 30
- setStringValue:resolvingEntities: 31
- setURI: 32
- stringValue 32
- URI 33
- XMLString 33
- XMLStringWithOptions: 34
- XPath 34
- Constants 34
 - Node Kind Constants 34
 - Input and Output Options 36

Document Revision History 41

Index 43

NSXMLNode Class Reference

Inherits from	NSObject
Conforms to	NSCopying NSObject (NSObject)
Framework	/System/Library/Frameworks/Foundation.framework
Availability	Available in Mac OS X v10.4 and later.
Companion guide	Tree-Based XML Programming Guide for Cocoa
Declared in	NSXMLNode.h NSXMLNodeOptions.h
Related sample code	AlbumToSlideshow Core Data HTML Store MovieAssembler TimelineToTC XMLBrowser

Overview

Objects of the `NSXMLNode` class are nodes in the abstract, logical tree structure that represents an XML document. Node objects can be of different kinds, corresponding to the following markup constructs in an XML document: element, attribute, text, processing instruction, namespace, and comment. In addition, a document-node object (specifically, an instance of `NSXMLDocument`) represents an XML document in its entirety. `NSXMLNode` objects can also represent document type declarations as well as declarations in Document Type Definitions (DTDs). Class factory methods of `NSXMLNode` enable you to create nodes of each kind. Only document, element, and DTD nodes may have child nodes.

Among the `NSXML` family of classes—that is, the Foundation classes with the prefix “`NSXML`” (excluding `NSXMLParser`)—the `NSXMLNode` class is the base class. Inheriting from it are the classes `NSXMLElement`, `NSXMLDocument`, `NSXMLDTD`, and `NSXMLDTDNode`. `NSXMLNode` specifies the interface common to all XML node objects and defines common node behavior and attributes, for example hierarchy level, node name and value, tree traversal, and the ability to emit representative XML markup text.

Subclassing Notes

You can subclass `NSXMLNode` if you want nodes of kinds different from the supported ones. You can also create a subclass with more specialized attributes or behavior than `NSXMLNode`.

Methods to Override

To subclass `NSXMLNode` you need to override the primary initializer, `initWithKind:options:` (page 21), and the methods listed below. In most cases, you need only invoke the superclass implementation, adding any subclass-specific code before or after the invocation, as necessary.

<code>kind</code> (page 22)	<code>parent</code> (page 27)
<code>name</code> (page 23)	<code>childAtIndex:</code> (page 18)
<code>setName:</code> (page 29)	<code>childCount</code> (page 19)
<code>objectValue</code> (page 27)	<code>children</code> (page 19)
<code>setObjectValue:</code> (page 30)	<code>detach</code> (page 20)
<code>stringValue</code> (page 32)	<code>localName</code> (page 23)
<code>setStringValue:resolvingEntities:</code> (page 31)	<code>prefix</code> (page 27)
<code>index</code> (page 20)	<code>URI</code> (page 33)

By default `NSXMLNode` implements the `NSObject isEqual:` method to perform a deep comparison: two `NSXMLNode` objects are not considered equal unless they have the same name, same child nodes, same attributes, and so on. The comparison looks at the node and its children, but does not include the node's parent. If you want a different standard of comparison, override `isEqual:`.

Special Considerations

Because of the architecture and data model of NSXML, when it parses and processes a source of XML it cannot know about your subclass unless you override the `NSXMLDocument` class method `replacementClassForClass:` to return your custom class in place of an NSXML class. If your custom class has no direct NSXML counterpart—for example, it is a subclass of `NSXMLNode` that represents CDATA sections—then you can walk the tree after it has been created and insert the new node where appropriate.

Adopted Protocols

`NSCopying`
 - `copyWithZone:`

Tasks

Creating and Initializing Node Objects

- `initWithKind:` (page 21)
 Returns an `NSXMLNode` instance initialized with the constant indicating node kind.

- [initWithKind:options:](#) (page 21)
Returns an `NSXMLNode` instance initialized with the constant indicating node kind and one or more initialization options.
- + [document](#) (page 11)
Returns an empty document node.
- + [documentWithRootElement:](#) (page 11)
Returns an `NSXMLDocument` object initialized with a given root element.
- + [elementWithName:](#) (page 12)
Returns an `NSXMLElement` object with a given tag identifier, or name
- + [elementWithName:children:attributes:](#) (page 13)
Returns an `NSXMLElement` object with the given tag (name), attributes, and children.
- + [elementWithName:stringValue:](#) (page 13)
Returns an `NSXMLElement` object with a single text-node child containing the specified text.
- + [elementWithName:URI:](#) (page 14)
Returns an element whose fully qualified name is specified.
- + [attributeWithName:stringValue:](#) (page 9)
Returns an `NSXMLNode` object representing an attribute node with a given name and string.
- + [attributeWithName:URI:stringValue:](#) (page 10)
Returns an `NSXMLNode` object representing an attribute node with a given qualified name and string.
- + [textWithStringValue:](#) (page 17)
Returns an `NSXMLNode` object representing a text node with specified content.
- + [commentWithStringValue:](#) (page 11)
Returns an `NSXMLNode` object representing a comment node containing given text.
- + [namespaceWithName:stringValue:](#) (page 15)
Returns an `NSXMLNode` object representing a namespace with a specified name and URI.
- + [DTDNodeWithXMLString:](#) (page 12)
Returns a `NSXMLDTDNode` object representing the DTD declaration for an element, attribute, entity, or notation based on a given string.
- + [predefinedNamespaceForPrefix:](#) (page 15)
Returns an `NSXMLNode` object representing one of the predefined namespaces with the specified prefix.
- + [processingInstructionWithName:stringValue:](#) (page 16)
Returns an `NSXMLNode` object representing a processing instruction with a specified name and value.

Managing XML Node Objects

- [index](#) (page 20)
Returns the index of the receiver identifying its position relative to its sibling nodes.
- [kind](#) (page 22)
Returns the kind of node the receiver is as a constant of type `NSXMLNodeKind` (page 34).
- [level](#) (page 22)
Returns the nesting level of the receiver within the tree hierarchy.
- [setName:](#) (page 29)
Sets the name of the receiver.

- [name](#) (page 23)
Returns the name of the receiver.
- [setObjectValue:](#) (page 30)
Sets the content of the receiver to an object value.
- [objectValue](#) (page 27)
Returns the object value of the receiver.
- [setStringValue:](#) (page 30)
Sets the content of the receiver as a string value.
- [setStringValue:resolvingEntities:](#) (page 31)
Sets the content of the receiver as a string value and, optionally, resolves character references, predefined entities, and user-defined entities as declared in the associated DTD.
- [stringValue](#) (page 32)
Returns the content of the receiver as a string value.
- [setURI:](#) (page 32)
Sets the URI of the receiver.
- [URI](#) (page 33)
Returns the URI associated with the receiver.

Navigating the Tree of Nodes

- [rootDocument](#) (page 29)
Returns the `NSXMLDocument` object containing the root element and representing the XML document as a whole.
- [parent](#) (page 27)
Returns the parent node of the receiver.
- [childAtIndex:](#) (page 18)
Returns the child node of the receiver at the specified location.
- [childCount](#) (page 19)
Returns the number of child nodes the receiver has.
- [children](#) (page 19)
Returns an immutable array containing the child nodes of the receiver (as `NSXMLNode` objects).
- [nextNode](#) (page 24)
Returns the next `NSXMLNode` object in document order.
- [nextSibling](#) (page 24)
Returns the next `NSXMLNode` object that is a sibling node to the receiver.
- [previousNode](#) (page 28)
Returns the previous `NSXMLNode` object in document order.
- [previousSibling](#) (page 28)
Returns the previous `NSXMLNode` object that is a sibling node to the receiver.
- [detach](#) (page 20)
Detaches the receiver from its parent node.

Emitting Node Content

- [XMLString](#) (page 33)
Returns the string representation of the receiver as it would appear in an XML document.
- [XMLStringWithOptions:](#) (page 34)
Returns the string representation of the receiver as it would appear in an XML document, with one or more output options specified.
- [canonicalXMLStringPreservingComments:](#) (page 17)
Returns a string object encapsulating the receiver's XML in canonical form.
- [description](#) (page 19)
Returns a description of the receiver.

Executing Queries

- [nodesForXPath:error:](#) (page 25)
Returns the nodes resulting from executing an XPath query upon the receiver.
- [objectsForXQuery:error:](#) (page 26)
Returns the objects resulting from executing an XQuery query upon the receiver.
- [objectsForXQuery:constants:error:](#) (page 25)
Returns the objects resulting from executing an XQuery query upon the receiver.
- [XPath](#) (page 34)
Returns the XPath expression identifying the receiver's location in the document tree.

Managing Namespaces

- [localName](#) (page 23)
Returns the local name of the receiver.
- + [localNameForName:](#) (page 14)
Returns the local name from the specified qualified name.
- [prefix](#) (page 27)
Returns the prefix of the receiver's name.
- + [prefixForName:](#) (page 16)
Returns the prefix from the specified qualified name.

Class Methods

attributeWithName:stringValue:

Returns an NSXMLNode object representing an attribute node with a given name and string.

```
+ (id)attributeWithName:(NSString *)name stringValue:(NSString *)value
```

Parameters*name*

A string that is the name of an attribute.

value

A string that is the value of an attribute.

Return ValueAn `NSXMLNode` object of kind `NSXMLAttributeKind` or `nil` if the object couldn't be created.**Discussion**For example, in the attribute “`id=12345`”, “`id`” is the attribute name and “`12345`” is the attribute value.**Availability**

Available in Mac OS X v10.4 and later.

Related Sample Code

AlbumToSlideshow

Core Data HTML Store

XMLBrowser

Declared In`NSXMLNode.h`**attributeWithName:URI:stringValue:**Returns an `NSXMLNode` object representing an attribute node with a given qualified name and string.

```
+ (id)attributeWithName:(NSString *)name URI:(NSString *)URI stringValue:(NSString *)value
```

Parameters*name*

A string that is the name of an attribute.

*URI*A URI (Universal Resource Identifier) that qualifies *name*.*value*

A string that is the value of the attribute.

Return ValueAn `NSXMLNode` object of kind `NSXMLAttributeKind` or `nil` if the object couldn't be created.**Discussion**For example, in the attribute “`bst:id=12345`”, “`bst`” is the name qualifier (derived from the URI), “`id`” is the attribute name, and “`12345`” is the attribute value.**Availability**

Available in Mac OS X v10.4 and later.

Declared In`NSXMLNode.h`

commentWithStringValue:

Returns an `NSXMLNode` object representing an comment node containing given text.

```
+ (id)commentWithStringValue:(NSString *)stringValue
```

Parameters

stringValue

A string specifying the text of the comment. You may specify `nil` or an empty string (see Return Value).

Return Value

An `NSXMLNode` object representing an comment node (`NSXMLCommentKind`) containing the text *stringValue* or `nil` if the object couldn't be created. If *stringValue* is `nil` or an empty string, a content-less comment node is returned (`<!-->`).

Availability

Available in Mac OS X v10.4 and later.

Declared In

`NSXMLNode.h`

document

Returns an empty document node.

```
+ (id)document
```

Return Value

An empty document node—that is, an `NSXMLDocument` instance without a root element or XML-declaration information (version, encoding, standalone flag). Returns `nil` if the object couldn't be created.

Availability

Available in Mac OS X v10.4 and later.

Related Sample Code

[XMLBrowser](#)

Declared In

`NSXMLNode.h`

documentWithRootElement:

Returns an `NSXMLDocument` object initialized with a given root element.

```
+ (id)documentWithRootElement:(NSXMLElement *)element
```

Parameters

element

An `NSXMLElement` object representing an element.

Return Value

An `NSXMLDocument` object initialized with the root element *element* or `nil` if the object couldn't be created.

Availability

Available in Mac OS X v10.4 and later.

Declared In

NSXMLNode.h

DTDNodeWithXMLString:

Returns a `NSXMLDTDNode` object representing the DTD declaration for an element, attribute, entity, or notation based on a given string.

```
+ (id)DTDNodeWithXMLString:(NSString *)string
```

Parameters

string

A string that is a DTD declaration. The receiver parses this string to determine the kind of DTD node to create.

Return Value

An `NSXMLDTDNode` object representing the DTD declaration or `nil` if the object couldn't be created.

Discussion

For example, if *string* is the following:

```
<!ENTITY name (#PCDATA)>
```

`NSXMLNode` is able to assign the created node object a kind of `NSXMLEntityDeclarationKind` by parsing "ENTITY".

Note that if an attribute-list declaration (`<!ATTLIST . . . >`) has multiple attributes `NSXMLNode` only creates an `NSXMLDTDNode` object for the last attribute in the declaration.

Availability

Available in Mac OS X v10.4 and later.

Declared In

NSXMLNode.h

elementWithName:

Returns an `NSXMLElement` object with a given tag identifier, or name

```
+ (id)elementWithName:(NSString *)name
```

Parameters

name

A string that is the name (or tag identifier) of an element.

Return Value

An `NSXMLElement` object or `nil` if the object couldn't be created.

Discussion

The equivalent XML markup is `<name></name>`.

Availability

Available in Mac OS X v10.4 and later.

Related Sample Code

AlbumToSlideshow

Core Data HTML Store

Declared In

NSXMLNode.h

elementWithName:children:attributes:

Returns an `NSXMLElement` object with the given tag (name), attributes, and children.

```
+ (id)elementWithName:(NSString *)name children:(NSArray *)children
  attributes:(NSArray *)attributes
```

Parameters

name

A string that is the name (tag identifier) of the element.

children

An array of `NSXMLElement` objects or `NSXMLNode` objects of kinds `NSXMLElementKind`, `NSXMLProcessingInstructionKind`, `NSXMLCommentKind`, and `NSXMLTextKind`. Specify `nil` if there are no children to add to this node object.

attributes

An array of `NSXMLNode` objects of kind `NSXMLAttributeKind`. Specify `nil` if there are no attributes to add to this node object.

Return Value

An `NSXMLElement` object or `nil` if the object couldn't be created.

Availability

Available in Mac OS X v10.4 and later.

Declared In

NSXMLNode.h

elementWithName:stringValue:

Returns an `NSXMLElement` object with a single text-node child containing the specified text.

```
+ (id)elementWithName:(NSString *)name stringValue:(NSString *)string
```

Parameters

name

A string that is the name (tag identifier) of the element.

string

A string that is the content of the receiver's text node.

Return Value

An `NSXMLElement` object with a single text-node child—an `NSXMLNode` object of kind `NSXMLTextKind`—containing the text specified in *string*. Returns `nil` if the object couldn't be created.

Discussion

The equivalent XML markup is `<name>string</name>`.

Availability

Available in Mac OS X v10.4 and later.

Related Sample Code

AlbumToSlideshow

Core Data HTML Store

MovieAssembler

Declared In

NSXMLNode.h

elementWithName:URI:

Returns an element whose fully qualified name is specified.

```
+ (id)elementWithName:(NSString *)name URI:(NSString *)URI
```

Parameters

name

A string that is the name (or tag identifier) of an element.

URI

A URI (Universal Resource Identifier) that qualifies *name*.

Return Value

An `NSXMLElement` object or `nil` if the object cannot be created.

Discussion

The equivalent XML markup is `<URI:name></URI:name>`.

Availability

Available in Mac OS X v10.4 and later.

Declared In

NSXMLNode.h

localNameForName:

Returns the local name from the specified qualified name.

```
+ (NSString *)localNameForName:(NSString *)qName
```

Parameters

qName

A string that is a qualified name.

Discussion

For example, if the qualified name is “`bst:title`”, this method returns “`title`”.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [localName](#) (page 23)
- + [predefinedNamespaceForPrefix:](#) (page 15)
- + [prefixForName:](#) (page 16)

Declared In

NSXMLNode.h

namespaceWithName:stringValue:

Returns an NSXMLNode object representing a namespace with a specified name and URI.

```
+ (id)namespaceWithName:(NSString *)name stringValue:(NSString *)value
```

Parameters*name*

A string that is the name of the namespace. Specify `nil` or an empty string for *name* if this object represents the default namespace.

value

A string that identifies the URI associated with the namespace.

Return Value

An NSXMLNode object of kind `NSXMLNamespaceKind` or `nil` if the object couldn't be created.

Discussion

The equivalent namespace declaration in XML markup is `xmlns:name="value"`.

Note: Applications linked on Mac OS X or later will throw an exception if the *name* parameter is `nil`.

Availability

Available in Mac OS X v10.4 and later.

Declared In

NSXMLNode.h

predefinedNamespaceForPrefix:

Returns an NSXMLNode object representing one of the predefined namespaces with the specified prefix.

```
+ (NSXMLNode *)predefinedNamespaceForPrefix:(NSString *)name
```

Parameters*name*

A string specifying a prefix for a predefined namespace, for example "xml", "xs", or "xsi".

Return Value

An NSXMLNode object of kind `NSXMLNamespaceKind` or `nil` if the object couldn't be created. If something other than a predefined-namespace prefix is specified, the method returns `nil`.

Availability

Available in Mac OS X v10.4 and later.

See Also+ [localNameForName:](#) (page 14)+ [prefixForName:](#) (page 16)**Declared In**

NSXMLNode.h

prefixForName:

Returns the prefix from the specified qualified name.

+ (NSString *)prefixForName:(NSString *)*qName***Parameters***qName*

A string that is a qualified name.

Discussion

For example, if the qualified name is “bst:title,” this method returns “bst.”

Availability

Available in Mac OS X v10.4 and later.

See Also+ [localNameForName:](#) (page 14)- [prefix](#) (page 27)+ [predefinedNamespaceForPrefix:](#) (page 15)**Declared In**

NSXMLNode.h

processingInstructionWithName:stringValue:

Returns an NSXMLNode object representing a processing instruction with a specified name and value.

+ (id)processingInstructionWithName:(NSString *)*name* stringValue:(NSString *)*value***Parameters***name*

A string that is the name of the processing instruction.

value

A string that is the value of the processing instruction.

Return Value

An NSXMLNode object of kind NSXMLProcessingInstructionKind or nil if the object couldn't be created.

DiscussionThe equivalent XML markup is `<?name value?>`.**Availability**

Available in Mac OS X v10.4 and later.

Declared In

NSXMLNode.h

textWithStringValue:

Returns an NSXMLNode object representing a text node with specified content.

+ (id)textWithStringValue:(NSString *)value

Parameters*value*

A string that is the textual content of the node.

Return ValueAn NSXMLNode object of kind NSXMLTextKind initialized with the textual *value* or nil if the object couldn't be created.**Availability**

Available in Mac OS X v10.4 and later.

Declared In

NSXMLNode.h

Instance Methods

canonicalXMLStringPreservingComments:

Returns a string object encapsulating the receiver's XML in canonical form.

- (NSString *)canonicalXMLStringPreservingComments:(BOOL)comments

Parameters*comments*

YES to preserve comments, NO otherwise.

Discussion

Be sure to set the input option NSXMLNodePreserveWhitespace for true canonical form. The canonical form of an XML document is defined by the World Wide Web Consortium at <http://www.w3.org/TR/xml-c14n>. Generally, if two documents with varying physical representations have the same canonical form, then they are considered logically equivalent within the given application context. The following list summarizes most key aspects of canonical form as defined by the W3C recommendation:

- Encodes the document in UTF-8.
- Normalizes line breaks to "#xA" on input before parsing.
- Normalizes attribute values in the manner of a validating processor.
- Replaces character and parsed entity references with their character content.
- Replaces CDATA sections with their character content.
- Removes the XML declaration and the document type declaration (DTD).

- Converts empty elements to start-end tag pairs.
- Normalizes whitespace outside of the document element and within start and end tags.
- Retains all whitespace characters in content (excluding characters removed during line-feed normalization).
- Sets attribute value delimiters to quotation marks (double quotes).
- Replaces special characters in attribute values and character content with character references.
- Removes superfluous namespace declarations from each element.
- Adds default attributes to each element.
- Imposes lexicographic order on the namespace declarations and attributes of each element.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [XMLString](#) (page 33)
- [XMLStringWithOptions:](#) (page 34)

Declared In

NSXMLNode.h

childAtIndex:

Returns the child node of the receiver at the specified location.

```
- (NSXMLNode *)childAtIndex:(NSUInteger) index
```

Parameters

index

An integer specifying a node position in the receiver's array of children. If *index* is out of bounds, an exception is raised.

Return Value

An NSXMLNode object or `nil` if the receiver has no children.

Discussion

The receiver should be an NSXMLNode object representing a document, element, or document type declaration. The returned node object can represent an element, comment, text, or processing instruction.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [childCount](#) (page 19)

Related Sample Code

Core Data HTML Store
XMLBrowser

Declared In

NSXMLNode.h

childCount

Returns the number of child nodes the receiver has.

- (NSUInteger)childCount

Discussion

This receiver should be an `NSXMLNode` object representing a document, element, or document type declaration. For performance reasons, use this method instead of getting the count from the array returned by `children` (page 19) (for example, `[[thisNode children] count]`).

Availability

Available in Mac OS X v10.4 and later.

See Also

- [childAtIndex:](#) (page 18)
- [children](#) (page 19)
- [parent](#) (page 27)

Related Sample Code

Core Data HTML Store

Declared In

NSXMLNode.h

children

Returns an immutable array containing the child nodes of the receiver (as `NSXMLNode` objects).

- (NSArray *)children

Availability

Available in Mac OS X v10.4 and later.

See Also

- [childAtIndex:](#) (page 18)
- [childCount](#) (page 19)
- [parent](#) (page 27)

Declared In

NSXMLNode.h

description

Returns a description of the receiver.

- (NSString *)description

Discussion

Use this method for debugging rather than for generating XML output. It could yield more information than [XMLString](#) (page 33) and [XMLStringWithOptions:](#) (page 34).

Availability

Available in Mac OS X v10.4 and later.

See Also

- [XMLString](#) (page 33)
- [XMLStringWithOptions:](#) (page 34)

Declared In

NSXMLNode.h

detach

Detaches the receiver from its parent node.

- (void)detach

Discussion

This method is applicable to `NSXMLNode` objects representing elements, text, comments, processing instructions, attributes, and namespaces. Once the node object is detached, you can add it as a child node of another parent.

Availability

Available in Mac OS X v10.4 and later.

Related Sample Code

MovieAssembler

Declared In

NSXMLNode.h

index

Returns the index of the receiver identifying its position relative to its sibling nodes.

- (NSUInteger)index

Return Value

An integer that is the index of the receiver relative to its sibling nodes.

Discussion

The first child node of a parent has an index of zero.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [childAtIndex:](#) (page 18)

Related Sample Code

MovieAssembler

Declared In

NSXMLNode.h

initWithKind:

Returns an `NSXMLNode` instance initialized with the constant indicating node kind.

```
- (id) initWithKind:(NSXMLNodeKind)kind
```

Parameters

kind

An enum constant of type `NSXMLNodeKind` (page 34) that indicates the type of node. See “Constants” (page 34) for a list of valid `NSXMLNodeKind` constants.

Return Value

An `NSXMLNode` object initialized with *kind* or `nil` if the object couldn't be created. If *kind* is not a valid `NSXMLNodeKind` constant, the method returns an `NSXMLNode` object of kind `NSXMLInvalidKind`.

Discussion

This method invokes `initWithKind:options:` (page 21) with the *options* parameter set to `NSXMLNodeOptionsNone`.

Do not use this initializer for creating instances of `NSXMLDTDNode` for attribute-list declarations. Instead, use the `DTDNodeWithXMLString:` (page 12) class method of this class or the `initWithXMLString:` method of the `NSXMLDTDNode` class.

Availability

Available in Mac OS X v10.4 and later.

Related Sample Code

XMLBrowser

Declared In

`NSXMLNode.h`

initWithKind:options:

Returns an `NSXMLNode` instance initialized with the constant indicating node kind and one or more initialization options.

```
- (id) initWithKind:(NSXMLNodeKind)kind options:(NSUInteger)options
```

Parameters

kind

An enum constant of type `NSXMLNodeKind` (page 34) that indicates the type of node. See “Constants” (page 34) for a list of valid `NSXMLNodeKind` constants.

options

One or more constants that specify initialization options; if there are multiple constants, bit-OR them together. These options request operations on the represented XML related to fidelity (for example, preserving entities), quoting style, handling of empty elements, and other things. See “Constants” (page 34) for a list of valid node-initialization constants.

Return Value

An `NSXMLNode` object initialized with the given *kind* and *options*, or `nil` if the object couldn't be created. If *kind* is not a valid `NSXMLNodeKind` constant, the method returns an `NSXMLNode` object of kind `NSXMLInvalidKind`.

Discussion

Do not use this initializer for creating instances of `NSXMLDTDNode` for attribute-list declarations. Instead, use the `DTDNodeWithXMLString:` (page 12) class method of this class or the `initWithXMLString:` method of the `NSXMLDTDNode` class.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [initWithKind:](#) (page 21)

Related Sample Code

Core Data HTML Store

Declared In

`NSXMLNode.h`

kind

Returns the kind of node the receiver is as a constant of type [NSXMLNodeKind](#) (page 34).

- (NSXMLNodeKind)kind

Discussion

`NSXMLNode` objects can represent documents, elements, attributes, namespaces, text, processing instructions, comments, document type declarations, and specific declarations within DTDs. See [“Constants”](#) (page 34) for a list of valid `NSXMLNodeKind` constants

Availability

Available in Mac OS X v10.4 and later.

See Also

- [initWithKind:](#) (page 21)

Declared In

`NSXMLNode.h`

level

Returns the nesting level of the receiver within the tree hierarchy.

- (NSUInteger)level

Return Value

An integer indicating a nesting level.

Discussion

The root element of a document has a nesting level of one.

Availability

Available in Mac OS X v10.4 and later.

Declared In

NSXMLNode.h

localName

Returns the local name of the receiver.

- (NSString *)localName

Return Value

A string containing the local name of the receiver.

Discussion

The local name is the part of a node name that follows a namespace-qualifying colon or the full name if there is no colon. For example, “chapter” is the local name in the qualified name “acme:chapter”.

Availability

Available in Mac OS X v10.4 and later.

See Also+ [localNameForName:](#) (page 14)**Declared In**

NSXMLNode.h

name

Returns the name of the receiver.

- (NSString *)name

Return ValueReturns a string specifying the name of the receiver. May return `nil` if the receiver is not a valid kind of node (see discussion).**Discussion**

This method is applicable only to `NSXMLNode` objects representing elements, attributes, namespaces, processing instructions, and DTD-declaration nodes. If the receiver is not an object of one of these kinds, this method returns `nil`. For example, in the following construction:

```
<title>Chapter One</title>
```

The returned name for the element is “title”. If the name is associated with a namespace, the qualified name is returned. For example, if you create an element with local name “foo” and URI “http://bar.com” and the namespace “xmlns:baz=‘http://bar.com’” is applied to this node, when you invoke this method on the node you get “baz:foo”.

Availability

Available in Mac OS X v10.4 and later.

See Also- [setName:](#) (page 29)

Related Sample Code

Core Data HTML Store

Declared In

NSXMLNode.h

nextNode

Returns the next NSXMLNode object in document order.

- (NSXMLNode *)nextNode

Discussion

You use this method to “walk” forward through the tree structure representing an XML document or document section. (Use [previousNode](#) (page 28) to traverse the tree in the opposite direction.) Document order is the natural order that XML constructs appear in markup text. If you send this message to the last node in the tree, `nil` is returned. NSXMLNode bypasses namespace and attribute nodes when it traverses a tree in document order.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [nextSibling](#) (page 24)
- [previousSibling](#) (page 28)

Declared In

NSXMLNode.h

nextSibling

Returns the next NSXMLNode object that is a sibling node to the receiver.

- (NSXMLNode *)nextSibling

Discussion

This object will have an [index](#) (page 20) value that is one more than the receiver’s. If there are no more subsequent siblings (that is, other child nodes of the receiver’s parent) the method returns `nil`.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [nextNode](#) (page 24)
- [previousNode](#) (page 28)
- [previousSibling](#) (page 28)

Declared In

NSXMLNode.h

nodesForXPath:error:

Returns the nodes resulting from executing an XPath query upon the receiver.

```
- (NSArray *)nodesForXPath:(NSString *)xpath error:(NSError **)error
```

Parameters

xpath

A string that expresses an XPath query.

error

If query errors occur, indirectly returns an `NSError` object describing the errors.

Return Value

An array of `NSXMLNode` objects that match the query, or an empty array if there are no matches.

Discussion

The receiver acts as the context item for the query ("."). If you have explicitly added adjacent text nodes as children of an element, you should invoke the `NSXMLElement` method `normalizeAdjacentTextNodesPreservingCDATA:` (with an argument of `NO`) on the element before applying any XPath queries to it; this method coalesces these text nodes. The same precaution applies if you have processed a document preserving CDATA sections and these sections are adjacent to text nodes.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [XPath](#) (page 34)

Related Sample Code

CocoaSOAP

Core Data HTML Store

Declared In

`NSXMLNode.h`

objectsForXQuery:constants:error:

Returns the objects resulting from executing an XQuery query upon the receiver.

```
- (NSArray *)objectsForXQuery:(NSString *)xquery constants:(NSDictionary *)constants
error:(NSError **)error
```

Parameters

xquery

A string that expresses an XQuery query.

constants

A dictionary containing externally declared constants where the name of each constant variable is a key.

error

If query errors occur, indirectly returns an `NSError` object describing the errors.

Discussion

The receiver acts as the context item for the query (“.”). If the receiver has been changed after parsing to have multiple adjacent text nodes, you should invoke the `NSXMLElement` method `normalizeAdjacentTextNodesPreservingCDATA:` (with an argument of `NO`) to coalesce the text nodes before querying.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [XPath](#) (page 34)

Related Sample Code

XMLBrowser

Declared In

NSXMLNode.h

objectsForXQuery:error:

Returns the objects resulting from executing an XQuery query upon the receiver.

```
- (NSArray *)objectsForXQuery:(NSString *)xquery error:(NSError **)error
```

Parameters

xquery

A string that expresses an XQuery query.

error

If query errors occur, indirectly returns an `NSError` object describing the errors.

Discussion

The receiver acts as the context item for the query (“.”). If the receiver has been changed after parsing to have multiple adjacent text nodes, you should invoke the `NSXMLElement` method `normalizeAdjacentTextNodesPreservingCDATA:` (with an argument of `NO`) to coalesce the text nodes before querying. This convenience method invokes `objectsForXQuery:constants:error:` (page 25) with `nil` for the `constants` dictionary.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [XPath](#) (page 34)

Related Sample Code

MovieAssembler

TimelineToTC

Declared In

NSXMLNode.h

objectValue

Returns the object value of the receiver.

- (id)objectValue

Return Value

The object value of the receiver, which may be the same as the value returned by [stringValue](#) (page 32). For nodes without content (for example, document nodes), this method returns the string value, or an empty string if there is no string value.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [setObjectValue](#): (page 30)
- [setStringValue](#): (page 30)

Related Sample Code

CocoaSOAP

Core Data HTML Store

Declared In

NSXMLNode.h

parent

Returns the parent node of the receiver.

- (NSXMLNode *)parent

Discussion

Document nodes and standalone nodes (that is, the root of a detached branch of a tree) have no parent, and sending this message to them returns `nil`. A one-to-one relationship does not always exist between a parent and its children; although a namespace or attribute node cannot be a child, it still has a parent element.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [childCount](#) (page 19)
- [children](#) (page 19)

Related Sample Code

Core Data HTML Store

XMLBrowser

Declared In

NSXMLNode.h

prefix

Returns the prefix of the receiver's name.

- (NSString *)prefix

Return Value

A string containing the receiver's prefix. This method returns an empty string if the receiver's name is not qualified by a namespace.

Discussion

The prefix is the part of a namespace-qualified name that precedes the colon. For example, "acme" is the prefix in the qualified name "acme:chapter".

Availability

Available in Mac OS X v10.4 and later.

See Also

+ [prefixForName:](#) (page 16)

Declared In

NSXMLNode.h

previousNode

Returns the previous NSXMLNode object in document order.

- (NSXMLNode *)previousNode

Discussion

You use this method to "walk" backward through the tree structure representing an XML document or document section. (Use [nextNode](#) (page 24) to traverse the tree in the opposite direction.) Document order is the natural order that XML constructs appear in markup text. If you send this message to the first node in the tree (that is, the root element), `nil` is returned. NSXMLNode bypasses namespace and attribute nodes when it traverses a tree in document order.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [nextSibling](#) (page 24)

- [previousSibling](#) (page 28)

Declared In

NSXMLNode.h

previousSibling

Returns the previous NSXMLNode object that is a sibling node to the receiver.

- (NSXMLNode *)previousSibling

Discussion

This object will have an [index](#) (page 20) value that is one less than the receiver's. If there are no more previous siblings (that is, other child nodes of the receiver's parent) the method returns `nil`.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [nextNode](#) (page 24)
- [nextSibling](#) (page 24)
- [previousNode](#) (page 28)

Declared In

NSXMLNode.h

rootDocument

Returns the `NSXMLDocument` object containing the root element and representing the XML document as a whole.

```
- (NSXMLDocument *)rootDocument
```

Discussion

If the receiver is a standalone node (that is, a node at the head of a detached branch of the tree), this method returns `nil`.

Availability

Available in Mac OS X v10.4 and later.

Declared In

NSXMLNode.h

setName:

Sets the name of the receiver.

```
- (void)setName:(NSString *)name
```

Parameters

name

A string that is the name to assign to the receiver.

Discussion

This method is effective for the following node kinds: element, attribute, namespace, processing-instruction, document type declaration, element declaration, attribute declaration, entity declaration, and notation declaration. If an `NSXMLNode` object that requires a name doesn't have one, it cannot be written out as an XML string.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [name](#) (page 23)

Declared In

NSXMLNode.h

setObjectValue:

Sets the content of the receiver to an object value.

```
- (void)setObjectValue:(id) value
```

Parameters

value

An object to assign as the value to the receiver.

Discussion

This method can only be invoked on `NSXMLNode` objects that may have content, specifically elements, attributes, namespaces, processing instructions, text, and DTD-declaration nodes. The given object is usually a Foundation equivalent to one of the atomic types in the XQuery data model: `NSNumber` (integer, decimal, float, double, Boolean), `NSString` (string), `NSDate` (date), `NSData` (base64Binary and hexBinary), `NSURL` (URI), and `NSArray` (NMTOKENS, IDREFS, ENTITIES). However, you can also set the object value to be a custom value and register a value transformer (that is, an instance of `NSValueTransformer`) to convert the object value to an XML string representation when the node is asked for its string value.

Setting a node's object value removes all existing children, including processing instructions and comments. Setting an element node's object value creates a text node as the sole child. When an `NSXMLNode` object emits its object-value contents as a string, and it can determine the type of the value, it ensures that it the string is in a canonical form as defined by the W3C XML Schema Data Types specification.

Note: Prior to Mac OS X v 10.6 `setObjectValue:` would improperly and inconsistently format objects that were `NSNumber` instances. Applications linked on Mac OS X 10.6 or later will use correct scientific notation for all `NSNumber`s passed to `setObjectValue:`.

If you require a particular format for any value in your XML document, you should format the data yourself as a string and then use `setStringValue:` (page 30) to set the value. This guarantees that the text generated is in a format you control directly.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [objectValue](#) (page 27)
- [setStringValue:resolvingEntities:](#) (page 31)

Related Sample Code

Core Data HTML Store

Declared In

`NSXMLNode.h`

setStringValue:

Sets the content of the receiver as a string value.

```
- (void)setStringValue:(NSString *) string
```

Parameters*string*

A string to assign as the value of the receiver.

Discussion

This method invokes [setStringValue:resolvingEntities:](#) (page 31), passing in an argument of `NO` for the second parameter. This method can only be invoked on `NSXMLNode` objects that may have content, specifically elements, attributes, namespaces, processing instructions, text, and DTD-declaration nodes. Setting the string value of a node object removes all existing children, including processing instructions and comments. Setting the string value of an element-node object creates a text node as the sole child.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [setObjectValue:](#) (page 30)
- [stringValue](#) (page 32)

Related Sample Code

Core Data HTML Store

Declared In

`NSXMLNode.h`

setStringValue:resolvingEntities:

Sets the content of the receiver as a string value and, optionally, resolves character references, predefined entities, and user-defined entities as declared in the associated DTD.

```
- (void)setStringValue:(NSString *)string resolvingEntities:(BOOL)resolve
```

Parameters*string*

A string to assign as the value of the receiver.

resolve

YES to resolve character references, predefined entities, and user-defined entities as declared in the associated DTD; NO otherwise. Namespace and processing-instruction nodes have their entities resolved even if *resolve* is NO.

Discussion

User-defined entities not declared in the DTD remain in their unresolved form. This method can only be invoked on `NSXMLNode` objects that may have content, specifically elements, attributes, namespaces, processing instructions, text, and DTD-declaration nodes. Setting the string value of a node object removes all existing children, including processing instructions and comments. Setting the string value of an element-node object creates a text node as the sole child.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [setObjectValue:](#) (page 30)
- [setStringValue:](#) (page 30)
- [stringValue](#) (page 32)

Declared In

NSXMLNode.h

setURI:

Sets the URI of the receiver.

- (void)setURI:(NSString *)URI

Parameters*URI*

The URI to associate with the receiver. A URI (Universal Resource Identifier) is a scheme such as “http” or “ftp” followed by a colon character, and then a scheme-specific part.

Discussion

The receiver must be an `NSXMLElement` or `NSXMLDocument` document, or an attribute (that is, an `NSXMLNode` object of type `NSXMLAttributeKind`). For documents it is the URI of document origin.

Availability

Available in Mac OS X v10.4 and later.

Related Sample Code

XMLBrowser

Declared In

NSXMLNode.h

stringValue

Returns the content of the receiver as a string value.

- (NSString *)stringValue

Discussion

If the receiver is a node object of element kind, the content is that of any text-node children. This method recursively visits elements nodes and concatenates their text nodes in document order with no intervening spaces. If the receiver’s content is set as an object value, this method returns the string value representing the object. If the object value is one of the standard, built-in ones (`NSNumber`, `NSDate`, and so on), the string value is in canonical format as defined by the W3C XML Schema Data Types specification. If the object value is not represented by one of these classes (or if the default value transformer for a class has been overridden), the string value is generated by the `NSValueTransformer` registered for that object type.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [objectValue](#) (page 27)
- [setStringValue:](#) (page 30)
- [setStringValue:resolvingEntities:](#) (page 31)

Related Sample Code

Core Data HTML Store

MovieAssembler

TimelineToTC
XMLBrowser

Declared In
NSXMLNode.h

URI

Returns the URI associated with the receiver.

- (NSString *)URI

Discussion

A node's URI is derived from its namespace or a document's URI; for documents, the URI comes either from the parsed XML or is explicitly set. You cannot change the URI for a particular node other than a namespace or document node.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [setURI:](#) (NSXMLDocument)

Declared In
NSXMLNode.h

XMLString

Returns the string representation of the receiver as it would appear in an XML document.

- (NSString *)XMLString

Discussion

The returned string includes the string representations of all children. This method invokes [XMLStringWithOptions:](#) (page 34) with an *options* argument of `NSXMLNodeOptionsNone`.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [canonicalXMLStringPreservingComments:](#) (page 17)
- [description](#) (page 19)

Related Sample Code

Core Data HTML Store

Declared In
NSXMLNode.h

XMLStringWithOptions:

Returns the string representation of the receiver as it would appear in an XML document, with one or more output options specified.

```
- (NSString *)XMLStringWithOptions:(NSUInteger)options
```

Parameters

options

One or more `enum` constants identifying an output option; bit-OR multiple constants together. See [“Constants”](#) (page 34) for a list of valid constants for specifying output options.

Discussion

The returned string includes the string representations of all children.

Availability

Available in Mac OS X v10.4 and later.

Related Sample Code

XMLBrowser

Declared In

NSXMLNode.h

XPath

Returns the XPath expression identifying the receiver’s location in the document tree.

```
- (NSString *)XPath
```

Discussion

For example, this method might return a string such as “foo/bar[2]/baz”. The result of this method can be used directly in the [nodesForXPath:error:](#) (page 25) and [objectsForXQuery:constants:error:](#) (page 25) methods.

Availability

Available in Mac OS X v10.4 and later.

Declared In

NSXMLNode.h

Constants

Node Kind Constants

NSXMLNode declares the following constants of type `NSXMLNodeKind` for specifying a node’s kind in the initializer methods [initWithKind:](#) (page 21) and [initWithKind:options:](#) (page 21):

```
enum {
    NSXMLInvalidKind = 0,
    NSXMLDocumentKind,
    NSXMLElementKind,
    NSXMLAttributeKind,
    NSXMLNamespaceKind,
    NSXMLProcessingInstructionKind,
    NSXMLCommentKind,
    NSXMLTextKind,
    NSXMLDTDKind,
    NSXMLEntityDeclarationKind,
    NSXMLAttributeDeclarationKind,
    NSXMLElementDeclarationKind,
    NSXMLNotationDeclarationKind
};
typedef NSUInteger NSXMLNodeKind;
```

Constants

NSXMLInvalidKind

Indicates a node object created without a valid kind being specified (as returned by the [kind](#) (page 22) method).

Available in Mac OS X v10.4 and later.

Declared in NSXMLNode.h.

NSXMLDocumentKind

Specifies a document node.

Available in Mac OS X v10.4 and later.

Declared in NSXMLNode.h.

NSXMLElementKind

Specifies an element node.

Available in Mac OS X v10.4 and later.

Declared in NSXMLNode.h.

NSXMLAttributeKind

Specifies an attribute node

Available in Mac OS X v10.4 and later.

Declared in NSXMLNode.h.

NSXMLNamespaceKind

Specifies a namespace node.

Available in Mac OS X v10.4 and later.

Declared in NSXMLNode.h.

NSXMLProcessingInstructionKind

Specifies a processing-instruction node.

Available in Mac OS X v10.4 and later.

Declared in NSXMLNode.h.

NSXMLCommentKind

Specifies a comment node.

Available in Mac OS X v10.4 and later.

Declared in NSXMLNode.h.

NSXMLTextKind

Specifies a text node.

Available in Mac OS X v10.4 and later.

Declared in NSXMLNode.h.

NSXMLDTDKind

Specifies a document-type declaration (DTD) node.

Available in Mac OS X v10.4 and later.

Declared in NSXMLNode.h.

NSXMLEntityDeclarationKind

Specifies an entity-declaration node.

Available in Mac OS X v10.4 and later.

Declared in NSXMLNode.h.

NSXMLAttributeDeclarationKind

Specifies an attribute-list declaration node.

Available in Mac OS X v10.4 and later.

Declared in NSXMLNode.h.

NSXMLElementDeclarationKind

Specifies an element declaration node.

Available in Mac OS X v10.4 and later.

Declared in NSXMLNode.h.

NSXMLNotationDeclarationKind

Specifies a notation declaration node.

Available in Mac OS X v10.4 and later.

Declared in NSXMLNode.h.

Declared In

NSXMLNode.h

Input and Output Options

These constants are input and output options for all NSXMLNode objects (unless otherwise indicated), including NSXMLDocument objects. You can specify these options (OR'ing multiple options) in the NSXMLNode methods [initWithKind:options:](#) (page 21) and [XMLStringWithOptions:](#) (page 34).

```

enum {
    NSXMLNodeOptionsNone = 0,
    NSXMLNodeIsCDATA = 1 << 0,
    NSXMLNodeExpandEmptyElement = 1 << 1, // <a></a>
    NSXMLNodeCompactEmptyElement = 1 << 2, // <a/>
    NSXMLNodeUseSingleQuotes = 1 << 3,
    NSXMLNodeUseDoubleQuotes = 1 << 4,
    NSXMLDocumentTidyHTML = 1 << 9,
    NSXMLDocumentTidyXML = 1 << 10,
    NSXMLDocumentValidate = 1 << 13,
    NSXMLDocumentXInclude = 1 << 16,
    NSXMLNodePrettyPrint = 1 << 17,
    NSXMLDocumentIncludeContentTypeDeclaration = 1 << 18,
    NSXMLNodePreserveNamespaceOrder = 1 << 20,
    NSXMLNodePreserveAttributeOrder = 1 << 21,
    NSXMLNodePreserveEntities = 1 << 22,
    NSXMLNodePreservePrefixes = 1 << 23,
    NSXMLNodePreserveCDATA = 1 << 24,
    NSXMLNodePreserveWhitespace = 1 << 25,
    NSXMLNodePreserveDTD = 1 << 26,
    NSXMLNodePreserveCharacterReferences = 1 << 27,
    NSXMLNodePreserveEmptyElements =
        (NSXMLNodeExpandEmptyElement | NSXMLNodeCompactEmptyElement),
    NSXMLNodePreserveQuotes =
        (NSXMLNodeUseSingleQuotes | NSXMLNodeUseDoubleQuotes),
    NSXMLNodePreserveAll = (
        NSXMLNodePreserveNamespaceOrder |
        NSXMLNodePreserveAttributeOrder |
        NSXMLNodePreserveEntities |
        NSXMLNodePreservePrefixes |
        NSXMLNodePreserveCDATA |
        NSXMLNodePreserveEmptyElements |
        NSXMLNodePreserveQuotes |
        NSXMLNodePreserveWhitespace |
        NSXMLNodePreserveDTD |
        NSXMLNodePreserveCharacterReferences |
        0xFFF00000) // high 12 bits
};

```

Constants

NSXMLNodeOptionsNone

No options are requested for this input or output action.

Available in Mac OS X v10.4 and later.

Declared in NSXMLNodeOptions.h.

NSXMLNodeIsCDATA

Specifies that a text node contains and is written out as a CDATA section.

Available in Mac OS X v10.4 and later.

Declared in NSXMLNodeOptions.h.

NSXMLNodeExpandEmptyElement

Requests that an element should be expanded when empty; for example, <flag></flag>. This is the default.

Available in Mac OS X v10.4 and later.

Declared in NSXMLNodeOptions.h.

`NSXMLNodeCompactEmptyElement`

Requests that an element should be contracted when empty; for example, `<flag/>`.

Available in Mac OS X v10.4 and later.

Declared in `NSXMLNodeOptions.h`.

`NSXMLNodeUseSingleQuotes`

Requests that NSXML use single quotes for the value of an attribute or namespace node.

Available in Mac OS X v10.4 and later.

Declared in `NSXMLNodeOptions.h`.

`NSXMLNodeUseDoubleQuotes`

Requests that NSXML use double quotes for the value of an attribute or namespace node. This is the default.

Available in Mac OS X v10.4 and later.

Declared in `NSXMLNodeOptions.h`.

`NSXMLNodePrettyPrint`

Print this node with extra space for readability. (Output)

Available in Mac OS X v10.4 and later.

Declared in `NSXMLNodeOptions.h`.

`NSXMLNodePreserveNamespaceOrder`

Requests NSXML to preserve the order of namespace URI definitions as in the source XML.

Available in Mac OS X v10.4 and later.

Declared in `NSXMLNodeOptions.h`.

`NSXMLNodePreserveAttributeOrder`

Requests that NSXMLNode preserve the order of attributes as in the source XML.

Available in Mac OS X v10.4 and later.

Declared in `NSXMLNodeOptions.h`.

`NSXMLNodePreserveEntities`

Specifies that entities (`&xyz;`) should not be resolved for XML output of this node.

Available in Mac OS X v10.4 and later.

Declared in `NSXMLNodeOptions.h`.

`NSXMLNodePreserveCharacterReferences`

Specifies that character references (`&#nnn;`) should not be resolved for XML output of this node.

Available in Mac OS X v10.4 and later.

Declared in `NSXMLNodeOptions.h`.

`NSXMLNodePreservePrefixes`

Requests NSXMLNode not to choose prefixes based on the closest namespace URI definition.

Available in Mac OS X v10.4 and later.

Declared in `NSXMLNodeOptions.h`.

`NSXMLNodePreserveCDATA`

Requests that NSXMLNode preserve CDATA blocks where defined in the input XML.

Available in Mac OS X v10.4 and later.

Declared in `NSXMLNodeOptions.h`.

NSXMLNodePreserveWhitespace

Requests NSXMLNode to preserve whitespace characters (such as tabs and carriage returns) in the XML source that are not part of node content.

Available in Mac OS X v10.4 and later.

Declared in NSXMLNodeOptions.h.

NSXMLNodePreserveEmptyElements

Specifies that empty elements in the input XML be preserved in their contracted or expanded form.

Available in Mac OS X v10.4 and later.

Declared in NSXMLNodeOptions.h.

NSXMLNodePreserveQuotes

Specifies that the quoting style used in the input XML (single or double quotes) be preserved.

Available in Mac OS X v10.4 and later.

Declared in NSXMLNodeOptions.h.

NSXMLNodePreserveDTD

Specifies that declarations in a DTD should be preserved until it the DTD is modified. For example, parameter entities are by default expanded; with this option, they are written out as they originally occur in the DTD.

Available in Mac OS X v10.4 and later.

Declared in NSXMLNodeOptions.h.

NSXMLNodePreserveAll

Turns on all preservation options: attribute and namespace order, entities, prefixes, CDATA, whitespace, quotes, and empty elements. You should try to turn on preservation options selectively because turning on all preservation options significantly affects performance.

Available in Mac OS X v10.4 and later.

Declared in NSXMLNodeOptions.h.

Discussion

The options with “Preserve” in their names are applicable only when external sources of XML are parsed; they have no effect on node objects that are programmatically created. Other options are used in initialization and output methods of NSXMLDocument; see the NSXMLDocument reference documentation for details.

Declared In

NSXMLNodeOptions.h

Document Revision History

This table describes the changes to *NSXMLNode Class Reference*.

Date	Notes
2009-10-19	Added exception warning for <code>namespaceWithName:stringValue:</code> . Added information to <code>setObjectValue:</code> about inconsistent behavior present before Mac OS X v 10.6.
2007-02-27	Documented <code>attributeWithLocalName:URI:stringValue:</code> , <code>elementWithName:URI:</code> , <code>setURI</code> methods and <code>NSXMLInvalidKind</code> constant.
2006-11-07	Corrected typo.
2006-05-23	First publication of this content as a separate document.

REVISION HISTORY

Document Revision History

Index

A

`attributeWithName:stringValue:` class method 9
`attributeWithName:URI:stringValue:` class method 10

C

`canonicalXMLStringPreservingComments:` instance method 17
`childAtIndex:` instance method 18
`childCount` instance method 19
`children` instance method 19
`commentWithStringValue:` class method 11

D

`description` instance method 19
`detach` instance method 20
`document` class method 11
`documentWithRootElement:` class method 11
`DTDNodeWithXMLString:` class method 12

E

`elementWithName:` class method 12
`elementWithName:children:attributes:` class method 13
`elementWithName:stringValue:` class method 13
`elementWithName:URI:` class method 14

I

`index` instance method 20
`initWithKind:` instance method 21

`initWithKind:options:` instance method 21

Input and Output Options 36

K

`kind` instance method 22

L

`level` instance method 22
`localName` instance method 23
`localNameForName:` class method 14

N

`name` instance method 23
`namespaceWithName:stringValue:` class method 15
`nextNode` instance method 24
`nextSibling` instance method 24
Node Kind Constants 34
`nodesForXPath:error:` instance method 25
`NSXMLAttributeDeclarationKind` constant 36
`NSXMLAttributeKind` constant 35
`NSXMLCommentKind` constant 35
`NSXMLDocumentKind` constant 35
`NSXMLDTDKind` constant 36
`NSXMLElementDeclarationKind` constant 36
`NSXMLElementKind` constant 35
`NSXMLEntityDeclarationKind` constant 36
`NSXMLInvalidKind` constant 35
`NSXMLNamespaceKind` constant 35
`NSXMLNodeCompactEmptyElement` constant 38
`NSXMLNodeExpandEmptyElement` constant 37
`NSXMLNodeIsCDATA` constant 37
`NSXMLNodeOptionsNone` constant 37
`NSXMLNodePreserveAll` constant 39
`NSXMLNodePreserveAttributeOrder` constant 38
`NSXMLNodePreserveCDATA` constant 38

NSXMLNodePreserveCharacterReferences **constant** [38](#)
 NSXMLNodePreserveDTD **constant** [39](#)
 NSXMLNodePreserveEmptyElements **constant** [39](#)
 NSXMLNodePreserveEntities **constant** [38](#)
 NSXMLNodePreserveNamespaceOrder **constant** [38](#)
 NSXMLNodePreservePrefixes **constant** [38](#)
 NSXMLNodePreserveQuotes **constant** [39](#)
 NSXMLNodePreserveWhitespace **constant** [39](#)
 NSXMLNodePrettyPrint **constant** [38](#)
 NSXMLNodeUseDoubleQuotes **constant** [38](#)
 NSXMLNodeUseSingleQuotes **constant** [38](#)
 NSXMLNotationDeclarationKind **constant** [36](#)
 NSXMLProcessingInstructionKind **constant** [35](#)
 NSXMLTextKind **constant** [36](#)

O

objectsForXQuery:constants:error: **instance method** [25](#)
 objectsForXQuery:error: **instance method** [26](#)
 objectValue **instance method** [27](#)

P

parent **instance method** [27](#)
 predefinedNamespaceForPrefix: **class method** [15](#)
 prefix **instance method** [27](#)
 prefixForName: **class method** [16](#)
 previousNode **instance method** [28](#)
 previousSibling **instance method** [28](#)
 processingInstructionWithName:stringValue: **class method** [16](#)

R

rootDocument **instance method** [29](#)

S

setName: **instance method** [29](#)
 setObjectValue: **instance method** [30](#)
 setStringValue: **instance method** [30](#)
 setStringValue:resolvingEntities: **instance method** [31](#)
 setURI: **instance method** [32](#)
 stringValue **instance method** [32](#)

T

textWithStringValue: **class method** [17](#)

U

URI **instance method** [33](#)

X

XMLString **instance method** [33](#)
 XMLStringWithOptions: **instance method** [34](#)
 XPath **instance method** [34](#)