| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/Node.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/org/w3c/dom/NameList.html)   [**NEXT CLASS**](http://docs.google.com/org/w3c/dom/NodeList.html) | [**FRAMES**](http://docs.google.com/index.html?org/w3c/dom/Node.html)    [**NO FRAMES**](http://docs.google.com/Node.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3znysh7) | CONSTR | [METHOD](#2et92p0) | DETAIL: [FIELD](#tyjcwt) | CONSTR | [METHOD](#3whwml4) |

## **org.w3c.dom**

Interface Node

**All Known Subinterfaces:** [Attr](http://docs.google.com/org/w3c/dom/Attr.html), [CDATASection](http://docs.google.com/org/w3c/dom/CDATASection.html), [CharacterData](http://docs.google.com/org/w3c/dom/CharacterData.html), [Comment](http://docs.google.com/org/w3c/dom/Comment.html), [Detail](http://docs.google.com/javax/xml/soap/Detail.html), [DetailEntry](http://docs.google.com/javax/xml/soap/DetailEntry.html), [Document](http://docs.google.com/org/w3c/dom/Document.html), [DocumentFragment](http://docs.google.com/org/w3c/dom/DocumentFragment.html), [DocumentType](http://docs.google.com/org/w3c/dom/DocumentType.html), [Element](http://docs.google.com/org/w3c/dom/Element.html), [Entity](http://docs.google.com/org/w3c/dom/Entity.html), [EntityReference](http://docs.google.com/org/w3c/dom/EntityReference.html), [Node](http://docs.google.com/javax/xml/soap/Node.html), [Notation](http://docs.google.com/org/w3c/dom/Notation.html), [ProcessingInstruction](http://docs.google.com/org/w3c/dom/ProcessingInstruction.html), [SOAPBody](http://docs.google.com/javax/xml/soap/SOAPBody.html), [SOAPBodyElement](http://docs.google.com/javax/xml/soap/SOAPBodyElement.html), [SOAPElement](http://docs.google.com/javax/xml/soap/SOAPElement.html), [SOAPEnvelope](http://docs.google.com/javax/xml/soap/SOAPEnvelope.html), [SOAPFault](http://docs.google.com/javax/xml/soap/SOAPFault.html), [SOAPFaultElement](http://docs.google.com/javax/xml/soap/SOAPFaultElement.html), [SOAPHeader](http://docs.google.com/javax/xml/soap/SOAPHeader.html), [SOAPHeaderElement](http://docs.google.com/javax/xml/soap/SOAPHeaderElement.html), [Text](http://docs.google.com/org/w3c/dom/Text.html), [Text](http://docs.google.com/javax/xml/soap/Text.html) **All Known Implementing Classes:** [IIOMetadataNode](http://docs.google.com/javax/imageio/metadata/IIOMetadataNode.html), [SOAPPart](http://docs.google.com/javax/xml/soap/SOAPPart.html)

public interface **Node**

The Node interface is the primary datatype for the entire Document Object Model. It represents a single node in the document tree. While all objects implementing the Node interface expose methods for dealing with children, not all objects implementing the Node interface may have children. For example, Text nodes may not have children, and adding children to such nodes results in a DOMException being raised.

The attributes nodeName, nodeValue and attributes are included as a mechanism to get at node information without casting down to the specific derived interface. In cases where there is no obvious mapping of these attributes for a specific nodeType (e.g., nodeValue for an Element or attributes for a Comment ), this returns null. Note that the specialized interfaces may contain additional and more convenient mechanisms to get and set the relevant information.

The values of nodeName, nodeValue, and attributes vary according to the node type as follows:

| Interface | nodeName | nodeValue | attributes |
| --- | --- | --- | --- |
| Attr | same as Attr.name | same as Attr.value | null |
| CDATASection | "#cdata-section" | same as CharacterData.data, the content of the CDATA Section | null |
| Comment | "#comment" | same as CharacterData.data, the content of the comment | null |
| Document | "#document" | null | null |
| DocumentFragment | "#document-fragment" | null | null |
| DocumentType | same as DocumentType.name | null | null |
| Element | same as Element.tagName | null | NamedNodeMap |
| Entity | entity name | null | null |
| EntityReference | name of entity referenced | null | null |
| Notation | notation name | null | null |
| ProcessingInstruction | same as ProcessingInstruction.target | same as ProcessingInstruction.data | null |
| Text | "#text" | same as CharacterData.data, the content of the text node | null |

See also the [Document Object Model (DOM) Level 3 Core Specification](http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407).

| **Field Summary** | |
| --- | --- |
| static short | [**ATTRIBUTE\_NODE**](http://docs.google.com/org/w3c/dom/Node.html#ATTRIBUTE_NODE)            The node is an Attr. |
| static short | [**CDATA\_SECTION\_NODE**](http://docs.google.com/org/w3c/dom/Node.html#CDATA_SECTION_NODE)            The node is a CDATASection. |
| static short | [**COMMENT\_NODE**](http://docs.google.com/org/w3c/dom/Node.html#COMMENT_NODE)            The node is a Comment. |
| static short | [**DOCUMENT\_FRAGMENT\_NODE**](http://docs.google.com/org/w3c/dom/Node.html#DOCUMENT_FRAGMENT_NODE)            The node is a DocumentFragment. |
| static short | [**DOCUMENT\_NODE**](http://docs.google.com/org/w3c/dom/Node.html#DOCUMENT_NODE)            The node is a Document. |
| static short | [**DOCUMENT\_POSITION\_CONTAINED\_BY**](http://docs.google.com/org/w3c/dom/Node.html#DOCUMENT_POSITION_CONTAINED_BY)            The node is contained by the reference node. |
| static short | [**DOCUMENT\_POSITION\_CONTAINS**](http://docs.google.com/org/w3c/dom/Node.html#DOCUMENT_POSITION_CONTAINS)            The node contains the reference node. |
| static short | [**DOCUMENT\_POSITION\_DISCONNECTED**](http://docs.google.com/org/w3c/dom/Node.html#DOCUMENT_POSITION_DISCONNECTED)            The two nodes are disconnected. |
| static short | [**DOCUMENT\_POSITION\_FOLLOWING**](http://docs.google.com/org/w3c/dom/Node.html#DOCUMENT_POSITION_FOLLOWING)            The node follows the reference node. |
| static short | [**DOCUMENT\_POSITION\_IMPLEMENTATION\_SPECIFIC**](http://docs.google.com/org/w3c/dom/Node.html#DOCUMENT_POSITION_IMPLEMENTATION_SPECIFIC)            The determination of preceding versus following is implementation-specific. |
| static short | [**DOCUMENT\_POSITION\_PRECEDING**](http://docs.google.com/org/w3c/dom/Node.html#DOCUMENT_POSITION_PRECEDING)            The second node precedes the reference node. |
| static short | [**DOCUMENT\_TYPE\_NODE**](http://docs.google.com/org/w3c/dom/Node.html#DOCUMENT_TYPE_NODE)            The node is a DocumentType. |
| static short | [**ELEMENT\_NODE**](http://docs.google.com/org/w3c/dom/Node.html#ELEMENT_NODE)            The node is an Element. |
| static short | [**ENTITY\_NODE**](http://docs.google.com/org/w3c/dom/Node.html#ENTITY_NODE)            The node is an Entity. |
| static short | [**ENTITY\_REFERENCE\_NODE**](http://docs.google.com/org/w3c/dom/Node.html#ENTITY_REFERENCE_NODE)            The node is an EntityReference. |
| static short | [**NOTATION\_NODE**](http://docs.google.com/org/w3c/dom/Node.html#NOTATION_NODE)            The node is a Notation. |
| static short | [**PROCESSING\_INSTRUCTION\_NODE**](http://docs.google.com/org/w3c/dom/Node.html#PROCESSING_INSTRUCTION_NODE)            The node is a ProcessingInstruction. |
| static short | [**TEXT\_NODE**](http://docs.google.com/org/w3c/dom/Node.html#TEXT_NODE)            The node is a Text node. |

| **Method Summary** | |
| --- | --- |
| [Node](http://docs.google.com/org/w3c/dom/Node.html) | [**appendChild**](http://docs.google.com/org/w3c/dom/Node.html#appendChild(org.w3c.dom.Node))([Node](http://docs.google.com/org/w3c/dom/Node.html) newChild)            Adds the node newChild to the end of the list of children of this node. |
| [Node](http://docs.google.com/org/w3c/dom/Node.html) | [**cloneNode**](http://docs.google.com/org/w3c/dom/Node.html#cloneNode(boolean))(boolean deep)            Returns a duplicate of this node, i.e., serves as a generic copy constructor for nodes. |
| short | [**compareDocumentPosition**](http://docs.google.com/org/w3c/dom/Node.html#compareDocumentPosition(org.w3c.dom.Node))([Node](http://docs.google.com/org/w3c/dom/Node.html) other)            Compares the reference node, i.e. |
| [NamedNodeMap](http://docs.google.com/org/w3c/dom/NamedNodeMap.html) | [**getAttributes**](http://docs.google.com/org/w3c/dom/Node.html#getAttributes())()            A NamedNodeMap containing the attributes of this node (if it is an Element) or null otherwise. |
| [String](http://docs.google.com/java/lang/String.html) | [**getBaseURI**](http://docs.google.com/org/w3c/dom/Node.html#getBaseURI())()            The absolute base URI of this node or null if the implementation wasn't able to obtain an absolute URI. |
| [NodeList](http://docs.google.com/org/w3c/dom/NodeList.html) | [**getChildNodes**](http://docs.google.com/org/w3c/dom/Node.html#getChildNodes())()            A NodeList that contains all children of this node. |
| [Object](http://docs.google.com/java/lang/Object.html) | [**getFeature**](http://docs.google.com/org/w3c/dom/Node.html#getFeature(java.lang.String,%20java.lang.String))([String](http://docs.google.com/java/lang/String.html) feature, [String](http://docs.google.com/java/lang/String.html) version)            This method returns a specialized object which implements the specialized APIs of the specified feature and version, as specified in . |
| [Node](http://docs.google.com/org/w3c/dom/Node.html) | [**getFirstChild**](http://docs.google.com/org/w3c/dom/Node.html#getFirstChild())()            The first child of this node. |
| [Node](http://docs.google.com/org/w3c/dom/Node.html) | [**getLastChild**](http://docs.google.com/org/w3c/dom/Node.html#getLastChild())()            The last child of this node. |
| [String](http://docs.google.com/java/lang/String.html) | [**getLocalName**](http://docs.google.com/org/w3c/dom/Node.html#getLocalName())()            Returns the local part of the qualified name of this node. |
| [String](http://docs.google.com/java/lang/String.html) | [**getNamespaceURI**](http://docs.google.com/org/w3c/dom/Node.html#getNamespaceURI())()            The namespace URI of this node, or null if it is unspecified (see ). |
| [Node](http://docs.google.com/org/w3c/dom/Node.html) | [**getNextSibling**](http://docs.google.com/org/w3c/dom/Node.html#getNextSibling())()            The node immediately following this node. |
| [String](http://docs.google.com/java/lang/String.html) | [**getNodeName**](http://docs.google.com/org/w3c/dom/Node.html#getNodeName())()            The name of this node, depending on its type; see the table above. |
| short | [**getNodeType**](http://docs.google.com/org/w3c/dom/Node.html#getNodeType())()            A code representing the type of the underlying object, as defined above. |
| [String](http://docs.google.com/java/lang/String.html) | [**getNodeValue**](http://docs.google.com/org/w3c/dom/Node.html#getNodeValue())()            The value of this node, depending on its type; see the table above. |
| [Document](http://docs.google.com/org/w3c/dom/Document.html) | [**getOwnerDocument**](http://docs.google.com/org/w3c/dom/Node.html#getOwnerDocument())()            The Document object associated with this node. |
| [Node](http://docs.google.com/org/w3c/dom/Node.html) | [**getParentNode**](http://docs.google.com/org/w3c/dom/Node.html#getParentNode())()            The parent of this node. |
| [String](http://docs.google.com/java/lang/String.html) | [**getPrefix**](http://docs.google.com/org/w3c/dom/Node.html#getPrefix())()            The namespace prefix of this node, or null if it is unspecified. |
| [Node](http://docs.google.com/org/w3c/dom/Node.html) | [**getPreviousSibling**](http://docs.google.com/org/w3c/dom/Node.html#getPreviousSibling())()            The node immediately preceding this node. |
| [String](http://docs.google.com/java/lang/String.html) | [**getTextContent**](http://docs.google.com/org/w3c/dom/Node.html#getTextContent())()            This attribute returns the text content of this node and its descendants. |
| [Object](http://docs.google.com/java/lang/Object.html) | [**getUserData**](http://docs.google.com/org/w3c/dom/Node.html#getUserData(java.lang.String))([String](http://docs.google.com/java/lang/String.html) key)            Retrieves the object associated to a key on a this node. |
| boolean | [**hasAttributes**](http://docs.google.com/org/w3c/dom/Node.html#hasAttributes())()            Returns whether this node (if it is an element) has any attributes. |
| boolean | [**hasChildNodes**](http://docs.google.com/org/w3c/dom/Node.html#hasChildNodes())()            Returns whether this node has any children. |
| [Node](http://docs.google.com/org/w3c/dom/Node.html) | [**insertBefore**](http://docs.google.com/org/w3c/dom/Node.html#insertBefore(org.w3c.dom.Node,%20org.w3c.dom.Node))([Node](http://docs.google.com/org/w3c/dom/Node.html) newChild, [Node](http://docs.google.com/org/w3c/dom/Node.html) refChild)            Inserts the node newChild before the existing child node refChild. |
| boolean | [**isDefaultNamespace**](http://docs.google.com/org/w3c/dom/Node.html#isDefaultNamespace(java.lang.String))([String](http://docs.google.com/java/lang/String.html) namespaceURI)            This method checks if the specified namespaceURI is the default namespace or not. |
| boolean | [**isEqualNode**](http://docs.google.com/org/w3c/dom/Node.html#isEqualNode(org.w3c.dom.Node))([Node](http://docs.google.com/org/w3c/dom/Node.html) arg)            Tests whether two nodes are equal. |
| boolean | [**isSameNode**](http://docs.google.com/org/w3c/dom/Node.html#isSameNode(org.w3c.dom.Node))([Node](http://docs.google.com/org/w3c/dom/Node.html) other)            Returns whether this node is the same node as the given one. |
| boolean | [**isSupported**](http://docs.google.com/org/w3c/dom/Node.html#isSupported(java.lang.String,%20java.lang.String))([String](http://docs.google.com/java/lang/String.html) feature, [String](http://docs.google.com/java/lang/String.html) version)            Tests whether the DOM implementation implements a specific feature and that feature is supported by this node, as specified in . |
| [String](http://docs.google.com/java/lang/String.html) | [**lookupNamespaceURI**](http://docs.google.com/org/w3c/dom/Node.html#lookupNamespaceURI(java.lang.String))([String](http://docs.google.com/java/lang/String.html) prefix)            Look up the namespace URI associated to the given prefix, starting from this node. |
| [String](http://docs.google.com/java/lang/String.html) | [**lookupPrefix**](http://docs.google.com/org/w3c/dom/Node.html#lookupPrefix(java.lang.String))([String](http://docs.google.com/java/lang/String.html) namespaceURI)            Look up the prefix associated to the given namespace URI, starting from this node. |
| void | [**normalize**](http://docs.google.com/org/w3c/dom/Node.html#normalize())()            Puts all Text nodes in the full depth of the sub-tree underneath this Node, including attribute nodes, into a "normal" form where only structure (e.g., elements, comments, processing instructions, CDATA sections, and entity references) separates Text nodes, i.e., there are neither adjacent Text nodes nor empty Text nodes. |
| [Node](http://docs.google.com/org/w3c/dom/Node.html) | [**removeChild**](http://docs.google.com/org/w3c/dom/Node.html#removeChild(org.w3c.dom.Node))([Node](http://docs.google.com/org/w3c/dom/Node.html) oldChild)            Removes the child node indicated by oldChild from the list of children, and returns it. |
| [Node](http://docs.google.com/org/w3c/dom/Node.html) | [**replaceChild**](http://docs.google.com/org/w3c/dom/Node.html#replaceChild(org.w3c.dom.Node,%20org.w3c.dom.Node))([Node](http://docs.google.com/org/w3c/dom/Node.html) newChild, [Node](http://docs.google.com/org/w3c/dom/Node.html) oldChild)            Replaces the child node oldChild with newChild in the list of children, and returns the oldChild node. |
| void | [**setNodeValue**](http://docs.google.com/org/w3c/dom/Node.html#setNodeValue(java.lang.String))([String](http://docs.google.com/java/lang/String.html) nodeValue)            The value of this node, depending on its type; see the table above. |
| void | [**setPrefix**](http://docs.google.com/org/w3c/dom/Node.html#setPrefix(java.lang.String))([String](http://docs.google.com/java/lang/String.html) prefix)            The namespace prefix of this node, or null if it is unspecified. |
| void | [**setTextContent**](http://docs.google.com/org/w3c/dom/Node.html#setTextContent(java.lang.String))([String](http://docs.google.com/java/lang/String.html) textContent)            This attribute returns the text content of this node and its descendants. |
| [Object](http://docs.google.com/java/lang/Object.html) | [**setUserData**](http://docs.google.com/org/w3c/dom/Node.html#setUserData(java.lang.String,%20java.lang.Object,%20org.w3c.dom.UserDataHandler))([String](http://docs.google.com/java/lang/String.html) key, [Object](http://docs.google.com/java/lang/Object.html) data, [UserDataHandler](http://docs.google.com/org/w3c/dom/UserDataHandler.html) handler)            Associate an object to a key on this node. |

| **Field Detail** |
| --- |

### ELEMENT\_NODE

static final short **ELEMENT\_NODE**

The node is an Element.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#org.w3c.dom.Node.ELEMENT_NODE)

### ATTRIBUTE\_NODE

static final short **ATTRIBUTE\_NODE**

The node is an Attr.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#org.w3c.dom.Node.ATTRIBUTE_NODE)

### TEXT\_NODE

static final short **TEXT\_NODE**

The node is a Text node.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#org.w3c.dom.Node.TEXT_NODE)

### CDATA\_SECTION\_NODE

static final short **CDATA\_SECTION\_NODE**

The node is a CDATASection.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#org.w3c.dom.Node.CDATA_SECTION_NODE)

### ENTITY\_REFERENCE\_NODE

static final short **ENTITY\_REFERENCE\_NODE**

The node is an EntityReference.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#org.w3c.dom.Node.ENTITY_REFERENCE_NODE)

### ENTITY\_NODE

static final short **ENTITY\_NODE**

The node is an Entity.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#org.w3c.dom.Node.ENTITY_NODE)

### PROCESSING\_INSTRUCTION\_NODE

static final short **PROCESSING\_INSTRUCTION\_NODE**

The node is a ProcessingInstruction.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#org.w3c.dom.Node.PROCESSING_INSTRUCTION_NODE)

### COMMENT\_NODE

static final short **COMMENT\_NODE**

The node is a Comment.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#org.w3c.dom.Node.COMMENT_NODE)

### DOCUMENT\_NODE

static final short **DOCUMENT\_NODE**

The node is a Document.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#org.w3c.dom.Node.DOCUMENT_NODE)

### DOCUMENT\_TYPE\_NODE

static final short **DOCUMENT\_TYPE\_NODE**

The node is a DocumentType.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#org.w3c.dom.Node.DOCUMENT_TYPE_NODE)

### DOCUMENT\_FRAGMENT\_NODE

static final short **DOCUMENT\_FRAGMENT\_NODE**

The node is a DocumentFragment.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#org.w3c.dom.Node.DOCUMENT_FRAGMENT_NODE)

### NOTATION\_NODE

static final short **NOTATION\_NODE**

The node is a Notation.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#org.w3c.dom.Node.NOTATION_NODE)

### DOCUMENT\_POSITION\_DISCONNECTED

static final short **DOCUMENT\_POSITION\_DISCONNECTED**

The two nodes are disconnected. Order between disconnected nodes is always implementation-specific.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#org.w3c.dom.Node.DOCUMENT_POSITION_DISCONNECTED)

### DOCUMENT\_POSITION\_PRECEDING

static final short **DOCUMENT\_POSITION\_PRECEDING**

The second node precedes the reference node.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#org.w3c.dom.Node.DOCUMENT_POSITION_PRECEDING)

### DOCUMENT\_POSITION\_FOLLOWING

static final short **DOCUMENT\_POSITION\_FOLLOWING**

The node follows the reference node.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#org.w3c.dom.Node.DOCUMENT_POSITION_FOLLOWING)

### DOCUMENT\_POSITION\_CONTAINS

static final short **DOCUMENT\_POSITION\_CONTAINS**

The node contains the reference node. A node which contains is always preceding, too.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#org.w3c.dom.Node.DOCUMENT_POSITION_CONTAINS)

### DOCUMENT\_POSITION\_CONTAINED\_BY

static final short **DOCUMENT\_POSITION\_CONTAINED\_BY**

The node is contained by the reference node. A node which is contained is always following, too.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#org.w3c.dom.Node.DOCUMENT_POSITION_CONTAINED_BY)

### DOCUMENT\_POSITION\_IMPLEMENTATION\_SPECIFIC

static final short **DOCUMENT\_POSITION\_IMPLEMENTATION\_SPECIFIC**

The determination of preceding versus following is implementation-specific.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#org.w3c.dom.Node.DOCUMENT_POSITION_IMPLEMENTATION_SPECIFIC)

| **Method Detail** |
| --- |

### getNodeName

[String](http://docs.google.com/java/lang/String.html) **getNodeName**()

The name of this node, depending on its type; see the table above.

### getNodeValue

[String](http://docs.google.com/java/lang/String.html) **getNodeValue**()  
 throws [DOMException](http://docs.google.com/org/w3c/dom/DOMException.html)

The value of this node, depending on its type; see the table above. When it is defined to be null, setting it has no effect, including if the node is read-only.

**Throws:** [DOMException](http://docs.google.com/org/w3c/dom/DOMException.html) - DOMSTRING\_SIZE\_ERR: Raised when it would return more characters than fit in a DOMString variable on the implementation platform.

### setNodeValue

void **setNodeValue**([String](http://docs.google.com/java/lang/String.html) nodeValue)  
 throws [DOMException](http://docs.google.com/org/w3c/dom/DOMException.html)

The value of this node, depending on its type; see the table above. When it is defined to be null, setting it has no effect, including if the node is read-only.

**Throws:** [DOMException](http://docs.google.com/org/w3c/dom/DOMException.html) - NO\_MODIFICATION\_ALLOWED\_ERR: Raised when the node is readonly and if it is not defined to be null.

### getNodeType

short **getNodeType**()

A code representing the type of the underlying object, as defined above.

### getParentNode

[Node](http://docs.google.com/org/w3c/dom/Node.html) **getParentNode**()

The parent of this node. All nodes, except Attr, Document, DocumentFragment, Entity, and Notation may have a parent. However, if a node has just been created and not yet added to the tree, or if it has been removed from the tree, this is null.

### getChildNodes

[NodeList](http://docs.google.com/org/w3c/dom/NodeList.html) **getChildNodes**()

A NodeList that contains all children of this node. If there are no children, this is a NodeList containing no nodes.

### getFirstChild

[Node](http://docs.google.com/org/w3c/dom/Node.html) **getFirstChild**()

The first child of this node. If there is no such node, this returns null.

### getLastChild

[Node](http://docs.google.com/org/w3c/dom/Node.html) **getLastChild**()

The last child of this node. If there is no such node, this returns null.

### getPreviousSibling

[Node](http://docs.google.com/org/w3c/dom/Node.html) **getPreviousSibling**()

The node immediately preceding this node. If there is no such node, this returns null.

### getNextSibling

[Node](http://docs.google.com/org/w3c/dom/Node.html) **getNextSibling**()

The node immediately following this node. If there is no such node, this returns null.

### getAttributes

[NamedNodeMap](http://docs.google.com/org/w3c/dom/NamedNodeMap.html) **getAttributes**()

A NamedNodeMap containing the attributes of this node (if it is an Element) or null otherwise.

### getOwnerDocument

[Document](http://docs.google.com/org/w3c/dom/Document.html) **getOwnerDocument**()

The Document object associated with this node. This is also the Document object used to create new nodes. When this node is a Document or a DocumentType which is not used with any Document yet, this is null.

**Since:** DOM Level 2

### insertBefore

[Node](http://docs.google.com/org/w3c/dom/Node.html) **insertBefore**([Node](http://docs.google.com/org/w3c/dom/Node.html) newChild,  
 [Node](http://docs.google.com/org/w3c/dom/Node.html) refChild)  
 throws [DOMException](http://docs.google.com/org/w3c/dom/DOMException.html)

Inserts the node newChild before the existing child node refChild. If refChild is null, insert newChild at the end of the list of children.

If newChild is a DocumentFragment object, all of its children are inserted, in the same order, before refChild. If the newChild is already in the tree, it is first removed.

**Note:** Inserting a node before itself is implementation dependent.

**Parameters:**newChild - The node to insert.refChild - The reference node, i.e., the node before which the new node must be inserted. **Returns:**The node being inserted. **Throws:** [DOMException](http://docs.google.com/org/w3c/dom/DOMException.html) - HIERARCHY\_REQUEST\_ERR: Raised if this node is of a type that does not allow children of the type of the newChild node, or if the node to insert is one of this node's ancestors or this node itself, or if this node is of type Document and the DOM application attempts to insert a second DocumentType or Element node.

WRONG\_DOCUMENT\_ERR: Raised if newChild was created from a different document than the one that created this node.

NO\_MODIFICATION\_ALLOWED\_ERR: Raised if this node is readonly or if the parent of the node being inserted is readonly.

NOT\_FOUND\_ERR: Raised if refChild is not a child of this node.

NOT\_SUPPORTED\_ERR: if this node is of type Document, this exception might be raised if the DOM implementation doesn't support the insertion of a DocumentType or Element node.**Since:** DOM Level 3

### replaceChild

[Node](http://docs.google.com/org/w3c/dom/Node.html) **replaceChild**([Node](http://docs.google.com/org/w3c/dom/Node.html) newChild,  
 [Node](http://docs.google.com/org/w3c/dom/Node.html) oldChild)  
 throws [DOMException](http://docs.google.com/org/w3c/dom/DOMException.html)

Replaces the child node oldChild with newChild in the list of children, and returns the oldChild node.

If newChild is a DocumentFragment object, oldChild is replaced by all of the DocumentFragment children, which are inserted in the same order. If the newChild is already in the tree, it is first removed.

**Note:** Replacing a node with itself is implementation dependent.

**Parameters:**newChild - The new node to put in the child list.oldChild - The node being replaced in the list. **Returns:**The node replaced. **Throws:** [DOMException](http://docs.google.com/org/w3c/dom/DOMException.html) - HIERARCHY\_REQUEST\_ERR: Raised if this node is of a type that does not allow children of the type of the newChild node, or if the node to put in is one of this node's ancestors or this node itself, or if this node is of type Document and the result of the replacement operation would add a second DocumentType or Element on the Document node.

WRONG\_DOCUMENT\_ERR: Raised if newChild was created from a different document than the one that created this node.

NO\_MODIFICATION\_ALLOWED\_ERR: Raised if this node or the parent of the new node is readonly.

NOT\_FOUND\_ERR: Raised if oldChild is not a child of this node.

NOT\_SUPPORTED\_ERR: if this node is of type Document, this exception might be raised if the DOM implementation doesn't support the replacement of the DocumentType child or Element child.**Since:** DOM Level 3

### removeChild

[Node](http://docs.google.com/org/w3c/dom/Node.html) **removeChild**([Node](http://docs.google.com/org/w3c/dom/Node.html) oldChild)  
 throws [DOMException](http://docs.google.com/org/w3c/dom/DOMException.html)

Removes the child node indicated by oldChild from the list of children, and returns it.

**Parameters:**oldChild - The node being removed. **Returns:**The node removed. **Throws:** [DOMException](http://docs.google.com/org/w3c/dom/DOMException.html) - NO\_MODIFICATION\_ALLOWED\_ERR: Raised if this node is readonly.

NOT\_FOUND\_ERR: Raised if oldChild is not a child of this node.

NOT\_SUPPORTED\_ERR: if this node is of type Document, this exception might be raised if the DOM implementation doesn't support the removal of the DocumentType child or the Element child.**Since:** DOM Level 3

### appendChild

[Node](http://docs.google.com/org/w3c/dom/Node.html) **appendChild**([Node](http://docs.google.com/org/w3c/dom/Node.html) newChild)  
 throws [DOMException](http://docs.google.com/org/w3c/dom/DOMException.html)

Adds the node newChild to the end of the list of children of this node. If the newChild is already in the tree, it is first removed.

**Parameters:**newChild - The node to add.If it is a DocumentFragment object, the entire contents of the document fragment are moved into the child list of this node **Returns:**The node added. **Throws:** [DOMException](http://docs.google.com/org/w3c/dom/DOMException.html) - HIERARCHY\_REQUEST\_ERR: Raised if this node is of a type that does not allow children of the type of the newChild node, or if the node to append is one of this node's ancestors or this node itself, or if this node is of type Document and the DOM application attempts to append a second DocumentType or Element node.

WRONG\_DOCUMENT\_ERR: Raised if newChild was created from a different document than the one that created this node.

NO\_MODIFICATION\_ALLOWED\_ERR: Raised if this node is readonly or if the previous parent of the node being inserted is readonly.

NOT\_SUPPORTED\_ERR: if the newChild node is a child of the Document node, this exception might be raised if the DOM implementation doesn't support the removal of the DocumentType child or Element child.**Since:** DOM Level 3

### hasChildNodes

boolean **hasChildNodes**()

Returns whether this node has any children.

**Returns:**Returns true if this node has any children, false otherwise.

### cloneNode

[Node](http://docs.google.com/org/w3c/dom/Node.html) **cloneNode**(boolean deep)

Returns a duplicate of this node, i.e., serves as a generic copy constructor for nodes. The duplicate node has no parent ( parentNode is null) and no user data. User data associated to the imported node is not carried over. However, if any UserDataHandlers has been specified along with the associated data these handlers will be called with the appropriate parameters before this method returns.

Cloning an Element copies all attributes and their values, including those generated by the XML processor to represent defaulted attributes, but this method does not copy any children it contains unless it is a deep clone. This includes text contained in an the Element since the text is contained in a child Text node. Cloning an Attr directly, as opposed to be cloned as part of an Element cloning operation, returns a specified attribute (specified is true). Cloning an Attr always clones its children, since they represent its value, no matter whether this is a deep clone or not. Cloning an EntityReference automatically constructs its subtree if a corresponding Entity is available, no matter whether this is a deep clone or not. Cloning any other type of node simply returns a copy of this node.

Note that cloning an immutable subtree results in a mutable copy, but the children of an EntityReference clone are readonly . In addition, clones of unspecified Attr nodes are specified. And, cloning Document, DocumentType, Entity, and Notation nodes is implementation dependent.

**Parameters:**deep - If true, recursively clone the subtree under the specified node; if false, clone only the node itself (and its attributes, if it is an Element). **Returns:**The duplicate node.

### normalize

void **normalize**()

Puts all Text nodes in the full depth of the sub-tree underneath this Node, including attribute nodes, into a "normal" form where only structure (e.g., elements, comments, processing instructions, CDATA sections, and entity references) separates Text nodes, i.e., there are neither adjacent Text nodes nor empty Text nodes. This can be used to ensure that the DOM view of a document is the same as if it were saved and re-loaded, and is useful when operations (such as XPointer [[XPointer](http://www.w3.org/TR/2003/REC-xptr-framework-20030325/)] lookups) that depend on a particular document tree structure are to be used. If the parameter "normalize-characters" of the DOMConfiguration object attached to the Node.ownerDocument is true, this method will also fully normalize the characters of the Text nodes.

**Note:** In cases where the document contains CDATASections, the normalize operation alone may not be sufficient, since XPointers do not differentiate between Text nodes and CDATASection nodes.

**Since:** DOM Level 3

### isSupported

boolean **isSupported**([String](http://docs.google.com/java/lang/String.html) feature,  
 [String](http://docs.google.com/java/lang/String.html) version)

Tests whether the DOM implementation implements a specific feature and that feature is supported by this node, as specified in .

**Parameters:**feature - The name of the feature to test.version - This is the version number of the feature to test. **Returns:**Returns true if the specified feature is supported on this node, false otherwise.**Since:** DOM Level 2

### getNamespaceURI

[String](http://docs.google.com/java/lang/String.html) **getNamespaceURI**()

The namespace URI of this node, or null if it is unspecified (see ).

This is not a computed value that is the result of a namespace lookup based on an examination of the namespace declarations in scope. It is merely the namespace URI given at creation time.

For nodes of any type other than ELEMENT\_NODE and ATTRIBUTE\_NODE and nodes created with a DOM Level 1 method, such as Document.createElement(), this is always null.

**Note:** Per the *Namespaces in XML* Specification [[XML Namespaces](http://www.w3.org/TR/1999/REC-xml-names-19990114/)] an attribute does not inherit its namespace from the element it is attached to. If an attribute is not explicitly given a namespace, it simply has no namespace.

**Since:** DOM Level 2

### getPrefix

[String](http://docs.google.com/java/lang/String.html) **getPrefix**()

The namespace prefix of this node, or null if it is unspecified. When it is defined to be null, setting it has no effect, including if the node is read-only.

Note that setting this attribute, when permitted, changes the nodeName attribute, which holds the qualified name, as well as the tagName and name attributes of the Element and Attr interfaces, when applicable.

Setting the prefix to null makes it unspecified, setting it to an empty string is implementation dependent.

Note also that changing the prefix of an attribute that is known to have a default value, does not make a new attribute with the default value and the original prefix appear, since the namespaceURI and localName do not change.

For nodes of any type other than ELEMENT\_NODE and ATTRIBUTE\_NODE and nodes created with a DOM Level 1 method, such as createElement from the Document interface, this is always null.

**Since:** DOM Level 2

### setPrefix

void **setPrefix**([String](http://docs.google.com/java/lang/String.html) prefix)  
 throws [DOMException](http://docs.google.com/org/w3c/dom/DOMException.html)

The namespace prefix of this node, or null if it is unspecified. When it is defined to be null, setting it has no effect, including if the node is read-only.

Note that setting this attribute, when permitted, changes the nodeName attribute, which holds the qualified name, as well as the tagName and name attributes of the Element and Attr interfaces, when applicable.

Setting the prefix to null makes it unspecified, setting it to an empty string is implementation dependent.

Note also that changing the prefix of an attribute that is known to have a default value, does not make a new attribute with the default value and the original prefix appear, since the namespaceURI and localName do not change.

For nodes of any type other than ELEMENT\_NODE and ATTRIBUTE\_NODE and nodes created with a DOM Level 1 method, such as createElement from the Document interface, this is always null.

**Throws:** [DOMException](http://docs.google.com/org/w3c/dom/DOMException.html) - INVALID\_CHARACTER\_ERR: Raised if the specified prefix contains an illegal character according to the XML version in use specified in the Document.xmlVersion attribute.

NO\_MODIFICATION\_ALLOWED\_ERR: Raised if this node is readonly.

NAMESPACE\_ERR: Raised if the specified prefix is malformed per the Namespaces in XML specification, if the namespaceURI of this node is null, if the specified prefix is "xml" and the namespaceURI of this node is different from "<http://www.w3.org/XML/1998/namespace>", if this node is an attribute and the specified prefix is "xmlns" and the namespaceURI of this node is different from "<http://www.w3.org/2000/xmlns/>", or if this node is an attribute and the qualifiedName of this node is "xmlns" [[XML Namespaces](http://www.w3.org/TR/1999/REC-xml-names-19990114/)] .**Since:** DOM Level 2

### getLocalName

[String](http://docs.google.com/java/lang/String.html) **getLocalName**()

Returns the local part of the qualified name of this node.

For nodes of any type other than ELEMENT\_NODE and ATTRIBUTE\_NODE and nodes created with a DOM Level 1 method, such as Document.createElement(), this is always null.

**Since:** DOM Level 2

### hasAttributes

boolean **hasAttributes**()

Returns whether this node (if it is an element) has any attributes.

**Returns:**Returns true if this node has any attributes, false otherwise.**Since:** DOM Level 2

### getBaseURI

[String](http://docs.google.com/java/lang/String.html) **getBaseURI**()

The absolute base URI of this node or null if the implementation wasn't able to obtain an absolute URI. This value is computed as described in . However, when the Document supports the feature "HTML" [[DOM Level 2 HTML](http://www.w3.org/TR/2003/REC-DOM-Level-2-HTML-20030109)] , the base URI is computed using first the value of the href attribute of the HTML BASE element if any, and the value of the documentURI attribute from the Document interface otherwise.

**Since:** DOM Level 3

### compareDocumentPosition

short **compareDocumentPosition**([Node](http://docs.google.com/org/w3c/dom/Node.html) other)  
 throws [DOMException](http://docs.google.com/org/w3c/dom/DOMException.html)

Compares the reference node, i.e. the node on which this method is being called, with a node, i.e. the one passed as a parameter, with regard to their position in the document and according to the document order.

**Parameters:**other - The node to compare against the reference node. **Returns:**Returns how the node is positioned relatively to the reference node. **Throws:** [DOMException](http://docs.google.com/org/w3c/dom/DOMException.html) - NOT\_SUPPORTED\_ERR: when the compared nodes are from different DOM implementations that do not coordinate to return consistent implementation-specific results.**Since:** DOM Level 3

### getTextContent

[String](http://docs.google.com/java/lang/String.html) **getTextContent**()  
 throws [DOMException](http://docs.google.com/org/w3c/dom/DOMException.html)

This attribute returns the text content of this node and its descendants. When it is defined to be null, setting it has no effect. On setting, any possible children this node may have are removed and, if it the new string is not empty or null, replaced by a single Text node containing the string this attribute is set to.

On getting, no serialization is performed, the returned string does not contain any markup. No whitespace normalization is performed and the returned string does not contain the white spaces in element content (see the attribute Text.isElementContentWhitespace). Similarly, on setting, no parsing is performed either, the input string is taken as pure textual content.

The string returned is made of the text content of this node depending on its type, as defined below:

| Node type | Content |
| --- | --- |
| ELEMENT\_NODE, ATTRIBUTE\_NODE, ENTITY\_NODE, ENTITY\_REFERENCE\_NODE, DOCUMENT\_FRAGMENT\_NODE | concatenation of the textContent attribute value of every child node, excluding COMMENT\_NODE and PROCESSING\_INSTRUCTION\_NODE nodes. This is the empty string if the node has no children. |
| TEXT\_NODE, CDATA\_SECTION\_NODE, COMMENT\_NODE, PROCESSING\_INSTRUCTION\_NODE | nodeValue |
| DOCUMENT\_NODE, DOCUMENT\_TYPE\_NODE, NOTATION\_NODE | *null* |

**Throws:** [DOMException](http://docs.google.com/org/w3c/dom/DOMException.html) - DOMSTRING\_SIZE\_ERR: Raised when it would return more characters than fit in a DOMString variable on the implementation platform.**Since:** DOM Level 3

### setTextContent

void **setTextContent**([String](http://docs.google.com/java/lang/String.html) textContent)  
 throws [DOMException](http://docs.google.com/org/w3c/dom/DOMException.html)

This attribute returns the text content of this node and its descendants. When it is defined to be null, setting it has no effect. On setting, any possible children this node may have are removed and, if it the new string is not empty or null, replaced by a single Text node containing the string this attribute is set to.

On getting, no serialization is performed, the returned string does not contain any markup. No whitespace normalization is performed and the returned string does not contain the white spaces in element content (see the attribute Text.isElementContentWhitespace). Similarly, on setting, no parsing is performed either, the input string is taken as pure textual content.

The string returned is made of the text content of this node depending on its type, as defined below:

| Node type | Content |
| --- | --- |
| ELEMENT\_NODE, ATTRIBUTE\_NODE, ENTITY\_NODE, ENTITY\_REFERENCE\_NODE, DOCUMENT\_FRAGMENT\_NODE | concatenation of the textContent attribute value of every child node, excluding COMMENT\_NODE and PROCESSING\_INSTRUCTION\_NODE nodes. This is the empty string if the node has no children. |
| TEXT\_NODE, CDATA\_SECTION\_NODE, COMMENT\_NODE, PROCESSING\_INSTRUCTION\_NODE | nodeValue |
| DOCUMENT\_NODE, DOCUMENT\_TYPE\_NODE, NOTATION\_NODE | *null* |

**Throws:** [DOMException](http://docs.google.com/org/w3c/dom/DOMException.html) - NO\_MODIFICATION\_ALLOWED\_ERR: Raised when the node is readonly.**Since:** DOM Level 3

### isSameNode

boolean **isSameNode**([Node](http://docs.google.com/org/w3c/dom/Node.html) other)

Returns whether this node is the same node as the given one.

This method provides a way to determine whether two Node references returned by the implementation reference the same object. When two Node references are references to the same object, even if through a proxy, the references may be used completely interchangeably, such that all attributes have the same values and calling the same DOM method on either reference always has exactly the same effect.

**Parameters:**other - The node to test against. **Returns:**Returns true if the nodes are the same, false otherwise.**Since:** DOM Level 3

### lookupPrefix

[String](http://docs.google.com/java/lang/String.html) **lookupPrefix**([String](http://docs.google.com/java/lang/String.html) namespaceURI)

Look up the prefix associated to the given namespace URI, starting from this node. The default namespace declarations are ignored by this method.

See for details on the algorithm used by this method.

**Parameters:**namespaceURI - The namespace URI to look for. **Returns:**Returns an associated namespace prefix if found or null if none is found. If more than one prefix are associated to the namespace prefix, the returned namespace prefix is implementation dependent.**Since:** DOM Level 3

### isDefaultNamespace

boolean **isDefaultNamespace**([String](http://docs.google.com/java/lang/String.html) namespaceURI)

This method checks if the specified namespaceURI is the default namespace or not.

**Parameters:**namespaceURI - The namespace URI to look for. **Returns:**Returns true if the specified namespaceURI is the default namespace, false otherwise.**Since:** DOM Level 3

### lookupNamespaceURI

[String](http://docs.google.com/java/lang/String.html) **lookupNamespaceURI**([String](http://docs.google.com/java/lang/String.html) prefix)

Look up the namespace URI associated to the given prefix, starting from this node.

See for details on the algorithm used by this method.

**Parameters:**prefix - The prefix to look for. If this parameter is null, the method will return the default namespace URI if any. **Returns:**Returns the associated namespace URI or null if none is found.**Since:** DOM Level 3

### isEqualNode

boolean **isEqualNode**([Node](http://docs.google.com/org/w3c/dom/Node.html) arg)

Tests whether two nodes are equal.

This method tests for equality of nodes, not sameness (i.e., whether the two nodes are references to the same object) which can be tested with Node.isSameNode(). All nodes that are the same will also be equal, though the reverse may not be true.

Two nodes are equal if and only if the following conditions are satisfied:

* The two nodes are of the same type.
* The following string attributes are equal: nodeName, localName, namespaceURI, prefix, nodeValue . This is: they are both null, or they have the same length and are character for character identical.
* The attributes NamedNodeMaps are equal. This is: they are both null, or they have the same length and for each node that exists in one map there is a node that exists in the other map and is equal, although not necessarily at the same index.
* The childNodes NodeLists are equal. This is: they are both null, or they have the same length and contain equal nodes at the same index. Note that normalization can affect equality; to avoid this, nodes should be normalized before being compared.

For two DocumentType nodes to be equal, the following conditions must also be satisfied:

* The following string attributes are equal: publicId, systemId, internalSubset.
* The entities NamedNodeMaps are equal.
* The notations NamedNodeMaps are equal.

On the other hand, the following do not affect equality: the ownerDocument, baseURI, and parentNode attributes, the specified attribute for Attr nodes, the schemaTypeInfo attribute for Attr and Element nodes, the Text.isElementContentWhitespace attribute for Text nodes, as well as any user data or event listeners registered on the nodes.

**Note:** As a general rule, anything not mentioned in the description above is not significant in consideration of equality checking. Note that future versions of this specification may take into account more attributes and implementations conform to this specification are expected to be updated accordingly.

**Parameters:**arg - The node to compare equality with. **Returns:**Returns true if the nodes are equal, false otherwise.**Since:** DOM Level 3

### getFeature

[Object](http://docs.google.com/java/lang/Object.html) **getFeature**([String](http://docs.google.com/java/lang/String.html) feature,  
 [String](http://docs.google.com/java/lang/String.html) version)

This method returns a specialized object which implements the specialized APIs of the specified feature and version, as specified in . The specialized object may also be obtained by using binding-specific casting methods but is not necessarily expected to, as discussed in . This method also allow the implementation to provide specialized objects which do not support the Node interface.

**Parameters:**feature - The name of the feature requested. Note that any plus sign "+" prepended to the name of the feature will be ignored since it is not significant in the context of this method.version - This is the version number of the feature to test. **Returns:**Returns an object which implements the specialized APIs of the specified feature and version, if any, or null if there is no object which implements interfaces associated with that feature. If the DOMObject returned by this method implements the Node interface, it must delegate to the primary core Node and not return results inconsistent with the primary core Node such as attributes, childNodes, etc.**Since:** DOM Level 3

### setUserData

[Object](http://docs.google.com/java/lang/Object.html) **setUserData**([String](http://docs.google.com/java/lang/String.html) key,  
 [Object](http://docs.google.com/java/lang/Object.html) data,  
 [UserDataHandler](http://docs.google.com/org/w3c/dom/UserDataHandler.html) handler)

Associate an object to a key on this node. The object can later be retrieved from this node by calling getUserData with the same key.

**Parameters:**key - The key to associate the object to.data - The object to associate to the given key, or null to remove any existing association to that key.handler - The handler to associate to that key, or null. **Returns:**Returns the DOMUserData previously associated to the given key on this node, or null if there was none.**Since:** DOM Level 3

### getUserData

[Object](http://docs.google.com/java/lang/Object.html) **getUserData**([String](http://docs.google.com/java/lang/String.html) key)

Retrieves the object associated to a key on a this node. The object must first have been set to this node by calling setUserData with the same key.

**Parameters:**key - The key the object is associated to. **Returns:**Returns the DOMUserData associated to the given key on this node, or null if there was none.**Since:** DOM Level 3

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/Node.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/org/w3c/dom/NameList.html)   [**NEXT CLASS**](http://docs.google.com/org/w3c/dom/NodeList.html) | [**FRAMES**](http://docs.google.com/index.html?org/w3c/dom/Node.html)    [**NO FRAMES**](http://docs.google.com/Node.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3znysh7) | CONSTR | [METHOD](#2et92p0) | DETAIL: [FIELD](#tyjcwt) | CONSTR | [METHOD](#3whwml4) |

[Submit a bug or feature](http://bugs.sun.com/services/bugreport/index.jsp)

For further API reference and developer documentation, see [Java SE Developer Documentation](http://docs.google.com/webnotes/devdocs-vs-specs.html). That documentation contains more detailed, developer-targeted descriptions, with conceptual overviews, definitions of terms, workarounds, and working code examples.

Copyright 2006 Sun Microsystems, Inc. All rights reserved. Use is subject to [license terms](http://docs.google.com/legal/license.html). Also see the [documentation redistribution policy](http://java.sun.com/docs/redist.html).