| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/Attr.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   [**NEXT CLASS**](http://docs.google.com/org/w3c/dom/CDATASection.html) | [**FRAMES**](http://docs.google.com/index.html?org/w3c/dom/Attr.html)    [**NO FRAMES**](http://docs.google.com/Attr.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | CONSTR | [METHOD](#tyjcwt) | DETAIL: FIELD | CONSTR | [METHOD](#1t3h5sf) |

## **org.w3c.dom**

Interface Attr

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

public interface **Attr**extends [Node](http://docs.google.com/org/w3c/dom/Node.html)

The Attr interface represents an attribute in an Element object. Typically the allowable values for the attribute are defined in a schema associated with the document.

Attr objects inherit the Node interface, but since they are not actually child nodes of the element they describe, the DOM does not consider them part of the document tree. Thus, the Node attributes parentNode, previousSibling, and nextSibling have a null value for Attr objects. The DOM takes the view that attributes are properties of elements rather than having a separate identity from the elements they are associated with; this should make it more efficient to implement such features as default attributes associated with all elements of a given type. Furthermore, Attr nodes may not be immediate children of a DocumentFragment. However, they can be associated with Element nodes contained within a DocumentFragment. In short, users and implementors of the DOM need to be aware that Attr nodes have some things in common with other objects inheriting the Node interface, but they also are quite distinct.

The attribute's effective value is determined as follows: if this attribute has been explicitly assigned any value, that value is the attribute's effective value; otherwise, if there is a declaration for this attribute, and that declaration includes a default value, then that default value is the attribute's effective value; otherwise, the attribute does not exist on this element in the structure model until it has been explicitly added. Note that the Node.nodeValue attribute on the Attr instance can also be used to retrieve the string version of the attribute's value(s).

If the attribute was not explicitly given a value in the instance document but has a default value provided by the schema associated with the document, an attribute node will be created with specified set to false. Removing attribute nodes for which a default value is defined in the schema generates a new attribute node with the default value and specified set to false. If validation occurred while invoking Document.normalizeDocument(), attribute nodes with specified equals to false are recomputed according to the default attribute values provided by the schema. If no default value is associate with this attribute in the schema, the attribute node is discarded.

In XML, where the value of an attribute can contain entity references, the child nodes of the Attr node may be either Text or EntityReference nodes (when these are in use; see the description of EntityReference for discussion).

The DOM Core represents all attribute values as simple strings, even if the DTD or schema associated with the document declares them of some specific type such as tokenized.

The way attribute value normalization is performed by the DOM implementation depends on how much the implementation knows about the schema in use. Typically, the value and nodeValue attributes of an Attr node initially returns the normalized value given by the parser. It is also the case after Document.normalizeDocument() is called (assuming the right options have been set). But this may not be the case after mutation, independently of whether the mutation is performed by setting the string value directly or by changing the Attr child nodes. In particular, this is true when [character references](http://www.w3.org/TR/2004/REC-xml-20040204#dt-charref) are involved, given that they are not represented in the DOM and they impact attribute value normalization. On the other hand, if the implementation knows about the schema in use when the attribute value is changed, and it is of a different type than CDATA, it may normalize it again at that time. This is especially true of specialized DOM implementations, such as SVG DOM implementations, which store attribute values in an internal form different from a string.

The following table gives some examples of the relations between the attribute value in the original document (parsed attribute), the value as exposed in the DOM, and the serialization of the value:

| Examples | Parsed attribute value | Initial Attr.value | Serialized attribute value |
| --- | --- | --- | --- |
| Character reference | "x²=5" | "x�=5" | "x²=5" |
| Built-in character entity | "y<6" | "y<6" | "y<6" |
| Literal newline between | "x=5 y=6" | "x=5 y=6" | "x=5 y=6" |
| Normalized newline between | "x=5   y=6" | "x=5 y=6" | "x=5 y=6" |
| Entity e with literal newline | <!ENTITY e '... ...'> [...]> "x=5&e;y=6" | *Dependent on Implementation and Load Options* | *Dependent on Implementation and Load/Save Options* |

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** | |
| --- | --- |

| **Fields inherited from interface org.w3c.dom.**[**Node**](http://docs.google.com/org/w3c/dom/Node.html) |
| --- |
| [ATTRIBUTE\_NODE](http://docs.google.com/org/w3c/dom/Node.html#ATTRIBUTE_NODE), [CDATA\_SECTION\_NODE](http://docs.google.com/org/w3c/dom/Node.html#CDATA_SECTION_NODE), [COMMENT\_NODE](http://docs.google.com/org/w3c/dom/Node.html#COMMENT_NODE), [DOCUMENT\_FRAGMENT\_NODE](http://docs.google.com/org/w3c/dom/Node.html#DOCUMENT_FRAGMENT_NODE), [DOCUMENT\_NODE](http://docs.google.com/org/w3c/dom/Node.html#DOCUMENT_NODE), [DOCUMENT\_POSITION\_CONTAINED\_BY](http://docs.google.com/org/w3c/dom/Node.html#DOCUMENT_POSITION_CONTAINED_BY), [DOCUMENT\_POSITION\_CONTAINS](http://docs.google.com/org/w3c/dom/Node.html#DOCUMENT_POSITION_CONTAINS), [DOCUMENT\_POSITION\_DISCONNECTED](http://docs.google.com/org/w3c/dom/Node.html#DOCUMENT_POSITION_DISCONNECTED), [DOCUMENT\_POSITION\_FOLLOWING](http://docs.google.com/org/w3c/dom/Node.html#DOCUMENT_POSITION_FOLLOWING), [DOCUMENT\_POSITION\_IMPLEMENTATION\_SPECIFIC](http://docs.google.com/org/w3c/dom/Node.html#DOCUMENT_POSITION_IMPLEMENTATION_SPECIFIC), [DOCUMENT\_POSITION\_PRECEDING](http://docs.google.com/org/w3c/dom/Node.html#DOCUMENT_POSITION_PRECEDING), [DOCUMENT\_TYPE\_NODE](http://docs.google.com/org/w3c/dom/Node.html#DOCUMENT_TYPE_NODE), [ELEMENT\_NODE](http://docs.google.com/org/w3c/dom/Node.html#ELEMENT_NODE), [ENTITY\_NODE](http://docs.google.com/org/w3c/dom/Node.html#ENTITY_NODE), [ENTITY\_REFERENCE\_NODE](http://docs.google.com/org/w3c/dom/Node.html#ENTITY_REFERENCE_NODE), [NOTATION\_NODE](http://docs.google.com/org/w3c/dom/Node.html#NOTATION_NODE), [PROCESSING\_INSTRUCTION\_NODE](http://docs.google.com/org/w3c/dom/Node.html#PROCESSING_INSTRUCTION_NODE), [TEXT\_NODE](http://docs.google.com/org/w3c/dom/Node.html#TEXT_NODE) |

| **Method Summary** | |
| --- | --- |
| [String](http://docs.google.com/java/lang/String.html) | [**getName**](http://docs.google.com/org/w3c/dom/Attr.html#getName())()            Returns the name of this attribute. |
| [Element](http://docs.google.com/org/w3c/dom/Element.html) | [**getOwnerElement**](http://docs.google.com/org/w3c/dom/Attr.html#getOwnerElement())()            The Element node this attribute is attached to or null if this attribute is not in use. |
| [TypeInfo](http://docs.google.com/org/w3c/dom/TypeInfo.html) | [**getSchemaTypeInfo**](http://docs.google.com/org/w3c/dom/Attr.html#getSchemaTypeInfo())()            The type information associated with this attribute. |
| boolean | [**getSpecified**](http://docs.google.com/org/w3c/dom/Attr.html#getSpecified())()            True if this attribute was explicitly given a value in the instance document, false otherwise. |
| [String](http://docs.google.com/java/lang/String.html) | [**getValue**](http://docs.google.com/org/w3c/dom/Attr.html#getValue())()            On retrieval, the value of the attribute is returned as a string. |
| boolean | [**isId**](http://docs.google.com/org/w3c/dom/Attr.html#isId())()            Returns whether this attribute is known to be of type ID (i.e. |
| void | [**setValue**](http://docs.google.com/org/w3c/dom/Attr.html#setValue(java.lang.String))([String](http://docs.google.com/java/lang/String.html) value)            On retrieval, the value of the attribute is returned as a string. |

| **Methods inherited from interface org.w3c.dom.**[**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)), [cloneNode](http://docs.google.com/org/w3c/dom/Node.html#cloneNode(boolean)), [compareDocumentPosition](http://docs.google.com/org/w3c/dom/Node.html#compareDocumentPosition(org.w3c.dom.Node)), [getAttributes](http://docs.google.com/org/w3c/dom/Node.html#getAttributes()), [getBaseURI](http://docs.google.com/org/w3c/dom/Node.html#getBaseURI()), [getChildNodes](http://docs.google.com/org/w3c/dom/Node.html#getChildNodes()), [getFeature](http://docs.google.com/org/w3c/dom/Node.html#getFeature(java.lang.String,%20java.lang.String)), [getFirstChild](http://docs.google.com/org/w3c/dom/Node.html#getFirstChild()), [getLastChild](http://docs.google.com/org/w3c/dom/Node.html#getLastChild()), [getLocalName](http://docs.google.com/org/w3c/dom/Node.html#getLocalName()), [getNamespaceURI](http://docs.google.com/org/w3c/dom/Node.html#getNamespaceURI()), [getNextSibling](http://docs.google.com/org/w3c/dom/Node.html#getNextSibling()), [getNodeName](http://docs.google.com/org/w3c/dom/Node.html#getNodeName()), [getNodeType](http://docs.google.com/org/w3c/dom/Node.html#getNodeType()), [getNodeValue](http://docs.google.com/org/w3c/dom/Node.html#getNodeValue()), [getOwnerDocument](http://docs.google.com/org/w3c/dom/Node.html#getOwnerDocument()), [getParentNode](http://docs.google.com/org/w3c/dom/Node.html#getParentNode()), [getPrefix](http://docs.google.com/org/w3c/dom/Node.html#getPrefix()), [getPreviousSibling](http://docs.google.com/org/w3c/dom/Node.html#getPreviousSibling()), [getTextContent](http://docs.google.com/org/w3c/dom/Node.html#getTextContent()), [getUserData](http://docs.google.com/org/w3c/dom/Node.html#getUserData(java.lang.String)), [hasAttributes](http://docs.google.com/org/w3c/dom/Node.html#hasAttributes()), [hasChildNodes](http://docs.google.com/org/w3c/dom/Node.html#hasChildNodes()), [insertBefore](http://docs.google.com/org/w3c/dom/Node.html#insertBefore(org.w3c.dom.Node,%20org.w3c.dom.Node)), [isDefaultNamespace](http://docs.google.com/org/w3c/dom/Node.html#isDefaultNamespace(java.lang.String)), [isEqualNode](http://docs.google.com/org/w3c/dom/Node.html#isEqualNode(org.w3c.dom.Node)), [isSameNode](http://docs.google.com/org/w3c/dom/Node.html#isSameNode(org.w3c.dom.Node)), [isSupported](http://docs.google.com/org/w3c/dom/Node.html#isSupported(java.lang.String,%20java.lang.String)), [lookupNamespaceURI](http://docs.google.com/org/w3c/dom/Node.html#lookupNamespaceURI(java.lang.String)), [lookupPrefix](http://docs.google.com/org/w3c/dom/Node.html#lookupPrefix(java.lang.String)), [normalize](http://docs.google.com/org/w3c/dom/Node.html#normalize()), [removeChild](http://docs.google.com/org/w3c/dom/Node.html#removeChild(org.w3c.dom.Node)), [replaceChild](http://docs.google.com/org/w3c/dom/Node.html#replaceChild(org.w3c.dom.Node,%20org.w3c.dom.Node)), [setNodeValue](http://docs.google.com/org/w3c/dom/Node.html#setNodeValue(java.lang.String)), [setPrefix](http://docs.google.com/org/w3c/dom/Node.html#setPrefix(java.lang.String)), [setTextContent](http://docs.google.com/org/w3c/dom/Node.html#setTextContent(java.lang.String)), [setUserData](http://docs.google.com/org/w3c/dom/Node.html#setUserData(java.lang.String,%20java.lang.Object,%20org.w3c.dom.UserDataHandler)) |

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

### getName

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

Returns the name of this attribute. If Node.localName is different from null, this attribute is a qualified name.

### getSpecified

boolean **getSpecified**()

True if this attribute was explicitly given a value in the instance document, false otherwise. If the application changed the value of this attribute node (even if it ends up having the same value as the default value) then it is set to true. The implementation may handle attributes with default values from other schemas similarly but applications should use Document.normalizeDocument() to guarantee this information is up-to-date.

### getValue

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

On retrieval, the value of the attribute is returned as a string. Character and general entity references are replaced with their values. See also the method getAttribute on the Element interface.

On setting, this creates a Text node with the unparsed contents of the string, i.e. any characters that an XML processor would recognize as markup are instead treated as literal text. See also the method Element.setAttribute().

Some specialized implementations, such as some [[SVG 1.1](http://www.w3.org/TR/2003/REC-SVG11-20030114/)] implementations, may do normalization automatically, even after mutation; in such case, the value on retrieval may differ from the value on setting.

### setValue

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

On retrieval, the value of the attribute is returned as a string. Character and general entity references are replaced with their values. See also the method getAttribute on the Element interface.

On setting, this creates a Text node with the unparsed contents of the string, i.e. any characters that an XML processor would recognize as markup are instead treated as literal text. See also the method Element.setAttribute().

Some specialized implementations, such as some [[SVG 1.1](http://www.w3.org/TR/2003/REC-SVG11-20030114/)] implementations, may do normalization automatically, even after mutation; in such case, the value on retrieval may differ from the value on setting.

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

### getOwnerElement

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

The Element node this attribute is attached to or null if this attribute is not in use.

**Since:** DOM Level 2

### getSchemaTypeInfo

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

The type information associated with this attribute. While the type information contained in this attribute is guarantee to be correct after loading the document or invoking Document.normalizeDocument(), schemaTypeInfo may not be reliable if the node was moved.

**Since:** DOM Level 3

### isId

boolean **isId**()

Returns whether this attribute is known to be of type ID (i.e. to contain an identifier for its owner element) or not. When it is and its value is unique, the ownerElement of this attribute can be retrieved using the method Document.getElementById . The implementation could use several ways to determine if an attribute node is known to contain an identifier:

* If validation occurred using an XML Schema [[XML Schema Part 1](http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/)] while loading the document or while invoking Document.normalizeDocument(), the post-schema-validation infoset contributions (PSVI contributions) values are used to determine if this attribute is a schema-determined ID attribute using the  [schema-determined ID](http://www.w3.org/TR/2003/REC-xptr-framework-20030325/#term-sdi) definition in [[XPointer](http://www.w3.org/TR/2003/REC-xptr-framework-20030325/)] .
* If validation occurred using a DTD while loading the document or while invoking Document.normalizeDocument(), the infoset **[type definition]** value is used to determine if this attribute is a DTD-determined ID attribute using the  [DTD-determined ID](http://www.w3.org/TR/2003/REC-xptr-framework-20030325/#term-ddi) definition in [[XPointer](http://www.w3.org/TR/2003/REC-xptr-framework-20030325/)] .
* from the use of the methods Element.setIdAttribute(), Element.setIdAttributeNS(), or Element.setIdAttributeNode(), i.e. it is an user-determined ID attribute;  
  **Note:** XPointer framework (see section 3.2 in [[XPointer](http://www.w3.org/TR/2003/REC-xptr-framework-20030325/)] ) consider the DOM user-determined ID attribute as being part of the XPointer externally-determined ID definition.
* using mechanisms that are outside the scope of this specification, it is then an externally-determined ID attribute. This includes using schema languages different from XML schema and DTD.

If validation occurred while invoking Document.normalizeDocument(), all user-determined ID attributes are reset and all attribute nodes ID information are then reevaluated in accordance to the schema used. As a consequence, if the Attr.schemaTypeInfo attribute contains an ID type, isId will always return true.

**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/Attr.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   [**NEXT CLASS**](http://docs.google.com/org/w3c/dom/CDATASection.html) | [**FRAMES**](http://docs.google.com/index.html?org/w3c/dom/Attr.html)    [**NO FRAMES**](http://docs.google.com/Attr.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | CONSTR | [METHOD](#tyjcwt) | DETAIL: FIELD | CONSTR | [METHOD](#1t3h5sf) |

[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).