Document Object Model (DOM) Level 3 XPath Specification

Version 1.0

W3C Working Draft 18 June 2001

This version:
http://www.w3.org/TR/2001/WD-DOM-Level-3-XPath-20010618
(PostScript file , PDF file , plain text , ZIP file , single HTML file)
Latest version:
http://www.w3.org/TR/DOM-Level-3-XPath

Editor:
Ray Whitmer, *Netscape/AOL*

Abstract

This specification defines the Document Object Model Level 3 XPath. It provides simple functionalities to access a DOM tree using [XPath 1.0]. This module builds on top of the Document Object Model Level 3 Core.

Status of this document

This section describes the status of this document at the time of its publication. Other documents may supersede this document. The latest status of this document series is maintained at the W3C.

This document contains the Document Object Model Level 3 XPath specification.

This is a first public Working Draft for review by W3C members and other interested parties.

It is a draft document and may be updated, replaced or obsoleted by other documents at any time. It is inappropriate to use W3C Working Drafts as reference material or to cite them as other than "work in progress". This is work in progress and does not imply endorsement by, or the consensus of, either W3C or members of the DOM Working Group or members of the XSL Working Group.

Comments on this document are invited and are to be sent to the public mailing list www-dom@w3.org. An archive is available at http://lists.w3.org/Archives/Public/www-dom/.

This document has been produced as part of the W3C DOM Activity. The authors of this document are the DOM Working Group members.

A list of current W3C Recommendations and other technical documents can be found at http://www.w3.org/TR.

Table of contents

Expanded Table of Contents

Copyright Notice

This document is published under the W3C Document Copyright Notice and License [p.5] . The bindings within this document are published under the W3C Software Copyright Notice and License [p.6] . The software license requires "Notice of any changes or modifications to the W3C files, including the date changes were made." Consequently, modified versions of the DOM bindings must document that they do not conform to the W3C standard; in the case of the IDL definitions, the pragma prefix can no longer be 'w3c.org'; in the case of the Java language binding, the package names can no longer be in the 'org.w3c' package.

W3C Document Copyright Notice and License

Note: This section is a copy of the W3C Document Notice and License and could be found at http://www.w3.org/Consortium/Legal/copyright-documents-19990405.

http://www.w3.org/Consortium/Legal/

Public documents on the W3C site are provided by the copyright holders under the following license. The software or Document Type Definitions (DTDs) associated with W3C specifications are governed by the Software Notice. By using and/or copying this document, or the W3C document from which this statement is linked, you (the licensee) agree that you have read, understood, and will comply with the following terms and conditions:

Permission to use, copy, and distribute the contents of this document, or the W3C document from which this statement is linked, in any medium for any purpose and without fee or royalty is hereby granted, provided that you include the following on *ALL* copies of the document, or portions thereof, that you use:

1. A link or URL to the original W3C document.
2. The pre-existing copyright notice of the original author, or if it doesn't exist, a notice of the form: "Copyright © [$date-of-document] World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University). All Rights Reserved. http://www.w3.org/Consortium/Legal/" (Hypertext is preferred, but a textual representation is permitted.)
3. *If it exists*, the STATUS of the W3C document.

When space permits, inclusion of the full text of this **NOTICE** should be provided. We request that authorship attribution be provided in any software, documents, or other items or products that you create pursuant to the implementation of the contents of this document, or any portion thereof.

No right to create modifications or derivatives of W3C documents is granted pursuant to this license. However, if additional requirements (documented in the Copyright FAQ) are satisfied, the right to create modifications or derivatives is sometimes granted by the W3C to individuals complying with those requirements.

THIS DOCUMENT IS PROVIDED "AS IS," AND COPYRIGHT HOLDERS MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR TITLE; THAT THE CONTENTS OF THE DOCUMENT ARE SUITABLE FOR ANY PURPOSE; NOR THAT THE IMPLEMENTATION OF SUCH CONTENTS WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS.

COPYRIGHT HOLDERS WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF ANY USE OF THE DOCUMENT OR THE PERFORMANCE OR IMPLEMENTATION OF THE CONTENTS THEREOF.

The name and trademarks of copyright holders may NOT be used in advertising or publicity pertaining to this document or its contents without specific, written prior permission. Title to copyright in this document will at all times remain with copyright holders.

W3C Software Copyright Notice and License

Note: This section is a copy of the W3C Software Copyright Notice and License and could be found at http://www.w3.org/Consortium/Legal/copyright-software-19980720

Copyright © 1994-2001 World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University). All Rights Reserved.

http://www.w3.org/Consortium/Legal/

This W3C work (including software, documents, or other related items) is being provided by the copyright holders under the following license. By obtaining, using and/or copying this work, you (the licensee) agree that you have read, understood, and will comply with the following terms and conditions:

Permission to use, copy, and modify this software and its documentation, with or without modification, for any purpose and without fee or royalty is hereby granted, provided that you include the following on ALL copies of the software and documentation or portions thereof, including modifications, that you make:

1. The full text of this NOTICE in a location viewable to users of the redistributed or derivative work.
2. Any pre-existing intellectual property disclaimers. If none exist, then a notice of the following form: "Copyright © [$date-of-software] World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University). All Rights Reserved. http://www.w3.org/Consortium/Legal/."

3. Notice of any changes or modifications to the W3C files, including the date changes were made. (We recommend you provide URIs to the location from which the code is derived.)

THIS SOFTWARE AND DOCUMENTATION IS PROVIDED "AS IS," AND COPYRIGHT HOLDERS MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF THE SOFTWARE OR DOCUMENTATION WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS.

COPYRIGHT HOLDERS WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF ANY USE OF THE SOFTWARE OR DOCUMENTATION.

The name and trademarks of copyright holders may NOT be used in advertising or publicity pertaining to the software without specific, written prior permission. Title to copyright in this software and any associated documentation will at all times remain with copyright holders.

1. Document Object Model XPath

Editors
Ray Whitmer, Netscape/AOL

1.1. Introduction

XPath is becoming an important part of a variety of many specifications including XForms, XPointer, XSL, CSS, and so on. It is also a clear advantage for user applications which use DOM to be able to use XPath expressions to locate nodes automatically and declaratively. But liveness issues have plagued each attempt to get a list of DOM nodes matching specific criteria, as would be expected for an XPath API. There have also traditionally been object model mismatches between DOM and XPath. This proposal specifies new interfaces and approaches to resolving these issues.

1.2. General considerations

This section considers the differences between the Document Object Model and the XPath Model as defined by [XPath 1.0] as well as Ranges.

1.2.1. Text Nodes

The XPath model sees a single logical text node where DOM may have multiple fragmented `Text` nodes due to cdata sections, entity references, etc. Instead of returning multiple nodes where XPath sees a single logical text node, only the first non-empty DOM `Text` node of any logical XPath text will be returned or entered into `Node` set. Applications will have to manually gather the text of a single logical text node from multiple nodes beginning with the first `Text` node identified by the implementation.

Note: DOM Level 3 Core adds the attribute `wholeText` on the `Text` interface for retrieving the whole text for logically-adjacent `Text` nodes.

1.2.2. Namespace Nodes

The XPath model expects namespace nodes for each in-scope namespace attached to each `Element`. DOM and certain other W3C infoset conformant implementations only maintain the declaration of namespaces instead of replicating them on each `Element` where they are in-scope. The DOM implementation of XPath will only return `Attr` nodes of in-scope namespace declarations, which may be attached to the current element or its ancestors.

Issue Namespace-1:
This does not exactly match the description of XPath 1.0. Need input from the XQuery 1.0/XPath 2.0 Data Model.

Issue Namespace-2:
How to represent namespace nodes in the Core?

Issue Namespace-3:
There is no declaration node corresponding to the always-present "`xml`" prefix declaration.
Issue Namespace-4:
The uniqueness and document order of namespace declaration nodes is different from in-scope namespace nodes, producing a different order and number of nodes in the result set where the namespaces nodes of multiple elements are requested.
Issue Namespace-5:
The `ownerElement` of the namespace declaration is different from the parent of in-scope namespace nodes of XPath.

1.2.2.1. Advantages

- Most common use cases would just work.
- No need to manufacture convenience nodes.
- XPath can be used to find and manipulate namespace declarations that are invisible in the strict XPath model.
- XQuery also models the namespace declaration instead of the in-scope namespace.

Note: The DOM WG should try to ensure compatibility between this approach and XPath 2.0. Removing the parent from XPath 2.0 namespace nodes does not help, because the relationship between the Namespace and Element was maintained, since XPath 2.0 maintains that the namespace nodes of an element still occur before its attribute nodes producing a many-to-one relationship between XPath 2.0 Namespace Nodes and DOM Attr nodes which declare namespaces, making it impossible to sort the results without a separate Namespace node for each occurrence to represent the order correctly. Also, the XQuery 1.0 and XPath 2.0 Data Model specification [XQuery 1.0 and XPath 2.0 Data Model] still maintains that each constructed Namespace node has a unique identity, implying that duplicates would be improperly eliminated in the resultant set. (See XQuery 1.0 and XPath 2.0 Data Model 3.1, 3.2, etc.).

1.2.3. Ranges

Since XPath returns nodes and strings, but never ranges, there is no use for ranges in this API as had been previously suggested. It is conceivable that in the future data types could be added supporting the notion of DOM ranges, especially if the DOM Working Group decided to do the work, at which time a range could become another return type of an XPath expression supported by the evaluator. This would be supportable by adding appropriate additional methods to `XPathEvaluator` [p.11] to evaluate XPath expressions returning ranges.

If sets of ranges were introduced, an appropriate interface for sets of ranges would need to be added to DOM (needed anyway for implementation of multiple selection), and this would be another new return type supported by the XPath evaluator.

1.3. Interfaces

A DOM application may use the `hasFeature(feature, version)` method of the `DOMImplementation` interface with parameter values "XPath" and "3.0" (respectively) to determine whether or not the event module is supported by the implementation. In order to fully support this module,

an implementation must also support the "Core" feature defined in the DOM Level 3 Core specification [DOM Level 3 Core]. Please, refer to additional information about conformance in the DOM Level 3 Core specification [DOM Level 3 Core].

Note: There is a fair amount of complexity in custom extension functions and variables because variables may be of various types and functions have arguments and return types. There are additional issues such as keeping functions from modifying the DOM, what exceptions to allow, and all the fun of callbacks. If the first draft of the specification ignores custom functions and variables, this should not make it more difficult to later add such a factory to the specification or for implementors to provide alternate evaluators that access extension functions and variables.

Definition group *XPathExceptionCode*

Defined Constants

`TYPE_ERR`

If an attempt is made to convert a XPath result into an imcompatible type.

Exception *XPathException*

IDL Definition

```
exception XPathException {
  unsigned short   code;
};
```

Interface *XPathEvaluator*

The evaluation of XPath expressions is available in `XPathEvaluator`. A `Document` which implements the XPath module will be castable using language-specific mechanisms to `XPathEvaluator`, which will provide evaluation of XPath expressions with no special extension functions or variables. `XPathEvaluator` implementations may be available from other sources that provide extension functions or variables.

Issue XPathEvaluator-1:

should we return an Object as the result instead and have one general `evaluateAs` method?

IDL Definition

```
interface XPathEvaluator {
  boolean            evaluateAsBoolean(in DOMString expression,
                                       in Node contextNode,
                                       in NamespaceResolver resolver)
                                        raises(XPathException);
  double             evaluateAsNumber(in DOMString expression,
                                      in Node contextNode,
                                      in NamespaceResolver resolver)
                                        raises(XPathException);
  DOMString          evaluateAsString(in DOMString expression,
                                      in Node contextNode,
                                      in NamespaceResolver resolver)
                                        raises(XPathException);
  Node               evaluateAsNode(in DOMString expression,
                                    in Node contextNode,
                                    in NamespaceResolver resolver)
                                        raises(XPathException);
  ActiveNodeSet      evaluateAsNodeSet(in DOMString expression,
```

```
                                        in Node contextNode,
                                        in NamespaceResolver resolver)
                                         raises(XPathException);
};
```

Methods

`evaluateAsBoolean`

Evaluates an XPath expression and converts the result to a boolean.

Parameters

`expression` of type `DOMString`

The XPath expression to be evaluated.

`contextNode` of type `Node`

The context node for the evaluation of the XPath expression.

`resolver` of type `NamespaceResolver` [p.17]

The `resolver` permits translation of prefixes within the XPath expression into appropriate namespaceURIs.

Return Value

`boolean` The result of the evaluation of the XPath expression.

Exceptions

`XPathException` [p.11]	TYPE_ERR: Raised if the result cannot be converted to a boolean

`evaluateAsNode`

Evaluates an XPath expression and returns the first node of the resulting set, `null` if the resulting set is empty.

Parameters

`expression` of type `DOMString`

The XPath expression to be evaluated.

`contextNode` of type `Node`

The context node for the evaluation of the XPath expression.

`resolver` of type `NamespaceResolver` [p.17]

The `resolver` permits translation of prefixes within the XPath expression into appropriate namespaceURIs.

Return Value

`Node` The result of the evaluation of the XPath expression.

Exceptions

`XPathException` [p.11]	TYPE_ERR: Raised if the result cannot be converted to a node set.

`evaluateAsNodeSet`

Evaluates an XPath expression and returns the result as a node set.

Parameters

`expression` of type `DOMString`

The XPath expression to be evaluated.

`contextNode` of type `Node`

The context node for the evaluation of the XPath expression.

`resolver` of type `NamespaceResolver` [p.17]

The `resolver` permits translation of prefixes within the XPath expression into appropriate namespaceURIs.

Return Value

`ActiveNodeSet` [p.15]	The result of the evaluation of the XPath expression.

Exceptions

`XPathException` [p.11]	TYPE_ERR: Raised if the result cannot be converted to a node set.

`evaluateAsNumber`

Evaluates an XPath expression and converts the result to a number.

Parameters

`expression` of type `DOMString`

The XPath expression to be evaluated.

`contextNode` of type `Node`

The context node for the evaluation of the XPath expression.

`resolver` of type `NamespaceResolver` [p.17]

The `resolver` permits translation of prefixes within the XPath expression into appropriate namespaceURIs.

Return Value

`double`	The result of the evaluation of the XPath expression.

Exceptions

`XPathException` [p.11]	TYPE_ERR: Raised if the result cannot be converted to a number (IEEE double precision floating point value)

`evaluateAsString`

Evaluates an XPath expression and converts the result to a string.

Parameters

`expression` of type `DOMString`

The XPath expression to be evaluated.

`contextNode` of type `Node`
: The context node for the evaluation of the XPath expression.

`resolver` of type `NamespaceResolver` [p.17]
: The `resolver` permits translation of prefixes within the XPath expression into appropriate namespaceURIs.

Return Value

`DOMString`	The result of the evaluation of the XPath expression.

Exceptions

`XPathException` [p.11]	TYPE_ERR: Raised if the result cannot be converted to a `DOMString`

An XPath node set is represented by a `NodeSet` interface which maintains a set of references to nodes. No node is ever duplicated in `NodeSet`. When `ActiveNodeSet` [p.15] is returned from an XPath evaluation, it corresponds to the hierarchy that it was requested on until the hierarchy is mutated in any way (anything that could produce a mutation event) at which time the set becomes invalid and any future call raises an exception. This permits the XPath sets to be computed either all at once or incrementally, however the implementation decides to do it. This also guarantees that any valid `ActiveNodeSet` corresponds to the current hierarchy.

`NodeSet` is not generally ordered in any way because implementations may be parallelized and compound statements may return nodes out of document order even in the simplest serial implementation. XSL and other processing requires XPath sets to be sorted in document order.

Interface *StaticNodeSet*

A `StaticNodeSet` is a collection of `Nodes` evaluated from an XPath expression. It never becomes invalid and never changes the array of node references even if the nodes are no longer are in the hierarchy or no longer match the expression that created the set. The individual nodes of a `StaticNodeSet` may be manipulated in the hierarchy and these changes are seen immediately by users referencing the nodes through the set.

IDL Definition

```
interface StaticNodeSet {
  Node               item(in unsigned long index);
  readonly attribute unsigned long    length;
};
```

Attributes

`length` of type `unsigned long`, readonly
: The number of nodes in the list. The range of valid child node indices is 0 to `length-1` inclusive.

Methods

`item`

Returns the `index`th item in the collection. If `index` is greater than or equal to the number of nodes in the list, this returns `null`.

Parameters

`index` of type `unsigned long`

Index into the collection.

Return Value

`Node` The node at the `index`th position in the `NodeList`, or `null` if that is not a valid index.

No Exceptions

Interface *ActiveNodeSet*

This represents a live node set obtained with the evaluation of an XPath expression.

Issue ActiveNodeSet01:

Do we need a detach() method?

IDL Definition

```
interface ActiveNodeSet {
  Node               nextNode()
                                        raises(DOMException);
  void               reset()
                                        raises(DOMException);
  ActiveNodeSet      cloneSet()
                                        raises(DOMException);
  ActiveNodeSet      getDocumentOrderedSet()
                                        raises(DOMException);
  StaticNodeSet      getStaticNodeSet()
                                        raises(DOMException);
};
```

Methods

`cloneSet`

Clones the `ActiveNodeSet`.

Issue cloneSet-1:

It was suggested that cloneSet should copy only the remaining set, which might favor incremental implementations and also make it possible to non-destructively pass on the current position. Clones must then be constructed at the appropriate position within the set and complete clones may only be constructed before processing any nodes.

Return Value

`ActiveNodeSet` [p.15] The new `ActiveNodeSet`.

Exceptions

`DOMException`	INVALID_STATE_ERR: The `ActiveNodeSet` is no longer valid.

No Parameters

`getDocumentOrderedSet`

This method may be called as long as the set is not invalid to get a set that is sorted into document order. Nodes which have no defined order with respect to each other (such as `Attr` nodes attached to an `Element`) will not be ordered in any particular order with respect to each other by this method.

Issue getDocumentOrdered-1:

Here are significant differences between DOM's currently proposed document order and the XPath model's document order expected by XSL and other callers of `getDocumentOrderedSet`. Some of the differences, we could just adopt. Others are unclear.

For example, in the fragment `<foo bar1="abc"/><foo bar2="def" bar3="ghi">`, DOM currently says that attribute nodes bar1 and bar2 are unordered with respect to each other, since they each have no parent. XPath and DOM both agree that bar2 and bar3 have no order with respect to each other, but XPath gives them an order with respect to their parent, giving them a definate position in the resulting sorted node set, whereas DOM does not acknowledge the ownerElement as part of the ordering relationship and makes it impossible to meaningfully sort attributes nodes of an element with elements or the attributes of other elements. I think we should adjust DOM's proposed comparison APIs and the result will help, rather than hurting, the DOM specification.

It will be much more difficult to make Attr nodes masquerading as XPath namespace nodes XPath-compatible with respect to their sort order. If it is a requirement to make this work, we have to use real Namespace nodes, IMO.

Return Value

`ActiveNodeSet` [p.15]

Exceptions

`DOMException`	INVALID_STATE_ERR: The `ActiveNodeSet` is no longer valid.

No Parameters

`getStaticNodeSet`

This method may be called as long as the set is not invalid to force complete evaluation and return `StaticNodeSet` [p.14] , which never becomes invalid and never changes the array of node references even if the nodes are no longer are in the hierarchy or no longer match the expression that created the set.

Return Value

`StaticNodeSet` [p.14] The complete evaluation of the XPath expression.

Exceptions

`DOMException` INVALID_STATE_ERR: The `ActiveNodeSet` is no longer valid.

No Parameters

`nextNode`

Returns the next node from the XPath expression evaluation.

Return Value

`Node` Returns the next node.

Exceptions

`DOMException` INVALID_STATE_ERR: The `ActiveNodeSet` is no longer valid.

No Parameters

`reset`

Exceptions

`DOMException` INVALID_STATE_ERR: The `ActiveNodeSet` is no longer valid.

No Parameters

No Return Value

Interface *NamespaceResolver*

The `NamespaceResolver` interface permit `prefix` strings in the expression to be properly bound to `namespaceURI` strings. The expectation is that an instance of the `NamespaceResolver` interface can be obtained by using binding-specific casting methods on an instance of the `Element` interface. This interface may also be user implemented instead of obtaining the implementation from an `Element`.

Issue NamespaceResolver-1:

This interface should be reconciled with the core method for namespace resolution so that it is automatically implemented by any Element

Resolution: The current proposal matches the definition in the Node interface.

IDL Definition

```
interface NamespaceResolver {
  DOMString          lookupNamespaceURI(in DOMString prefix);
};
```

Methods

`lookupNamespaceURI`

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

Parameters

`prefix` of type `DOMString`

The prefix to look for.

Return Value

`DOMString`	Returns the associated namespace URI or `null` if none is found.

No Exceptions

Appendix A: IDL Definitions

This appendix contains the complete OMG IDL [OMGIDL] for the Level 3 Document Object Model XPath definitions.

The IDL files are also available as:
http://www.w3.org/TR/2001/WD-DOM-Level-3-XPath-20010618/idl.zip

xpath.idl:

```
// File: xpath.idl

#ifndef _XPATH_IDL_
#define _XPATH_IDL_

#include "dom.idl"

#pragma prefix "dom.w3c.org"
module xpath
{

  typedef dom::DOMString DOMString;
  typedef dom::Node Node;

  interface NamespaceResolver;
  interface ActiveNodeSet;

  exception XPathException {
    unsigned short   code;
  };


  interface XPathEvaluator {
    boolean            evaluateAsBoolean(in DOMString expression,
                                         in Node contextNode,
                                         in NamespaceResolver resolver)
                                        raises(XPathException);
    double             evaluateAsNumber(in DOMString expression,
                                        in Node contextNode,
                                        in NamespaceResolver resolver)
                                        raises(XPathException);
    DOMString          evaluateAsString(in DOMString expression,
                                        in Node contextNode,
                                        in NamespaceResolver resolver)
                                        raises(XPathException);
    Node               evaluateAsNode(in DOMString expression,
                                      in Node contextNode,
                                      in NamespaceResolver resolver)
                                        raises(XPathException);
    ActiveNodeSet      evaluateAsNodeSet(in DOMString expression,
                                         in Node contextNode,
                                         in NamespaceResolver resolver)
                                        raises(XPathException);
  };
```

```
  interface StaticNodeSet {
    Node               item(in unsigned long index);
    readonly attribute unsigned long    length;
  };

  interface ActiveNodeSet {
    Node               nextNode()
                                        raises(dom::DOMException);
    void               reset()
                                        raises(dom::DOMException);
    ActiveNodeSet      cloneSet()
                                        raises(dom::DOMException);
    ActiveNodeSet      getDocumentOrderedSet()
                                        raises(dom::DOMException);
    StaticNodeSet      getStaticNodeSet()
                                        raises(dom::DOMException);
  };

  interface NamespaceResolver {
    DOMString          lookupNamespaceURI(in DOMString prefix);
  };
};

#endif // _XPATH_IDL_
```

Appendix B: Java Language Binding

This appendix contains the complete Java [Java] bindings for the Level 3 Document Object Model XPath.

The Java files are also available as
http://www.w3.org/TR/2001/WD-DOM-Level-3-XPath-20010618/java-binding.zip

B.1: Other XPath interfaces

org/w3c/dom/xpath/XPathException.java:

```
package org.w3c.dom.xpath;

public class XPathException extends RuntimeException {
    public XPathException(short code, String message) {
       super(message);
       this.code = code;
    }
    public short   code;
};
```

org/w3c/dom/xpath/XPathEvaluator.java:

```
package org.w3c.dom.xpath;

import org.w3c.dom.Node;

public interface XPathEvaluator {
    public boolean evaluateAsBoolean(String expression,
                                     Node contextNode,
                                     NamespaceResolver resolver)
                                     throws XPathException;

    public double evaluateAsNumber(String expression,
                                   Node contextNode,
                                   NamespaceResolver resolver)
                                   throws XPathException;

    public String evaluateAsString(String expression,
                                   Node contextNode,
                                   NamespaceResolver resolver)
                                   throws XPathException;

    public Node evaluateAsNode(String expression,
                               Node contextNode,
                               NamespaceResolver resolver)
                               throws XPathException;

    public ActiveNodeSet evaluateAsNodeSet(String expression,
                                           Node contextNode,
```

```
                                        NamespaceResolver resolver)
                                        throws XPathException;

}
```

org/w3c/dom/xpath/StaticNodeSet.java:

```
package org.w3c.dom.xpath;

import org.w3c.dom.Node;

public interface StaticNodeSet {
    public Node item(int index);

    public int getLength();

}
```

org/w3c/dom/xpath/ActiveNodeSet.java:

```
package org.w3c.dom.xpath;

import org.w3c.dom.Node;
import org.w3c.dom.DOMException;

public interface ActiveNodeSet {
    public Node nextNode()
                        throws DOMException;

    public void reset()
                      throws DOMException;

    public ActiveNodeSet cloneSet()
                                throws DOMException;

    public ActiveNodeSet getDocumentOrderedSet()
                                            throws DOMException;

    public StaticNodeSet getStaticNodeSet()
                                         throws DOMException;

}
```

org/w3c/dom/xpath/NamespaceResolver.java:

```
package org.w3c.dom.xpath;

public interface NamespaceResolver {
    public String lookupNamespaceURI(String prefix);

}
```

Appendix C: ECMA Script Language Binding

This appendix contains the complete ECMA Script [ECMAScript] binding for the Level 3 Document Object Model XPath definitions.

Prototype Object **XPathExceptionCode**

The **XPathExceptionCode** class has the following constants:

XPathExceptionCode.TYPE_ERR

This constant is of type **Number** and its value is **1**.

Object **XPathExceptionCode**

Object **XPathException**

The **XPathException** object has the following properties:

code

This property is of type **Number**.

Object **XPathEvaluator**

The **XPathEvaluator** object has the following methods:

evaluateAsBoolean(expression, contextNode, resolver)

This method returns a **Boolean**.
The **expression** parameter is of type **String**.
The **contextNode** parameter is a **Node** object.
The **resolver** parameter is a **NamespaceResolver** object.
This method can raise a **XPathException** object.

evaluateAsNumber(expression, contextNode, resolver)

This method returns a **double** object.
The **expression** parameter is of type **String**.
The **contextNode** parameter is a **Node** object.
The **resolver** parameter is a **NamespaceResolver** object.
This method can raise a **XPathException** object.

evaluateAsString(expression, contextNode, resolver)

This method returns a **String**.
The **expression** parameter is of type **String**.
The **contextNode** parameter is a **Node** object.
The **resolver** parameter is a **NamespaceResolver** object.
This method can raise a **XPathException** object.

evaluateAsNode(expression, contextNode, resolver)

This method returns a **Node** object.
The **expression** parameter is of type **String**.
The **contextNode** parameter is a **Node** object.
The **resolver** parameter is a **NamespaceResolver** object.
This method can raise a **XPathException** object.

evaluateAsNodeSet(expression, contextNode, resolver)

This method returns a **ActiveNodeSet** object.
The **expression** parameter is of type **String**.
The **contextNode** parameter is a **Node** object.
The **resolver** parameter is a **NamespaceResolver** object.
This method can raise a **XPathException** object.

Object **StaticNodeSet**

The **StaticNodeSet** object has the following properties:

length

This read-only property is of type **Number**.

The **StaticNodeSet** object has the following methods:

item(index)

This method returns a **Node** object.

The **index** parameter is of type **Number**.

Note: This object can also be dereferenced using square bracket notation (e.g. obj[1]). Dereferencing with an integer **index** is equivalent to invoking the **item** method with that index.

Object **ActiveNodeSet**

The **ActiveNodeSet** object has the following methods:

nextNode()

This method returns a **Node** object.

This method can raise a **DOMException** object.

reset()

This method has no return value.

This method can raise a **DOMException** object.

cloneSet()

This method returns a **ActiveNodeSet** object.

This method can raise a **DOMException** object.

getDocumentOrderedSet()

This method returns a **ActiveNodeSet** object.

This method can raise a **DOMException** object.

getStaticNodeSet()

This method returns a **StaticNodeSet** object.

This method can raise a **DOMException** object.

Object **NamespaceResolver**

The **NamespaceResolver** object has the following methods:

lookupNamespaceURI(prefix)

This method returns a **String**.

The **prefix** parameter is of type **String**.

Appendix D: Acknowledgements

Many people contributed to this specification, including members of the DOM Working Group and the DOM Interest Group. We especially thank the following:

Andrew Watson (Object Management Group), Andy Heninger (IBM), Arnaud Le Hors (W3C and IBM), Ben Chang (Oracle), Bill Smith (Sun), Bill Shea (Merrill Lynch), Bob Sutor (IBM), Chris Lovett (Microsoft), Chris Wilson (Microsoft), David Brownell (Sun), David Singer (IBM), Don Park (invited), Eric Vasilik (Microsoft), Gavin Nicol (INSO), Ian Jacobs (W3C), James Clark (invited), James Davidson (Sun), Jared Sorensen (Novell), Joe Kesselman (IBM), Joe Lapp (webMethods), Joe Marini (Macromedia), Johnny Stenback (Netscape), Jonathan Marsh (Microsoft), Jonathan Robie (Texcel Research and Software AG), Kim Adamson-Sharpe (SoftQuad Software Inc.), Lauren Wood (SoftQuad Software Inc., *former chair*), Laurence Cable (Sun), Mark Davis (IBM), Mark Scardina (Oracle), Martin Dürst (W3C), Mick Goulish (Software AG), Mike Champion (Arbortext and Software AG), Miles Sabin (Cromwell Media), Patti Lutsky (Arbortext), Paul Grosso (Arbortext), Peter Sharpe (SoftQuad Software Inc.), Phil Karlton (Netscape), Philippe Le Hégaret (W3C, *W3C team contact and Chair*), Ramesh Lekshmynarayanan (Merrill Lynch), Ray Whitmer (iMall, Excite@Home and Netscape/AOL), Rich Rollman (Microsoft), Rick Gessner (Netscape), Scott Isaacs (Microsoft), Sharon Adler (INSO), Steve Byrne (JavaSoft), Tim Bray (invited), Tom Pixley (Netscape), Vidur Apparao (Netscape), Vinod Anupam (Lucent), Johnny Stenback (Netscape/AOL), Jeroen van Rotterdam (X-Hive Corporation), Rezaur Rahman (Intel), Rob Relyea (Microsoft), Tim Yu (Oracle), Angel Diaz (Oracle), Jon Ferraiolo (Adobe), David Ezell (Hewlett Packard Company), Ashok Malhotra, (IBM and Microsoft), Rick Jelliffe (invited), Elena Litani (IBM), Mary Brady (NIST), Dimitris Dimitriadis (Improve AB).

Thanks to all those who have helped to improve this specification by sending suggestions and corrections (Please, keep bugging us with your issues!).

D.1: Production Systems

This specification was written in XML. The HTML, OMG IDL, Java and ECMA Script bindings were all produced automatically.

Thanks to Joe English, author of cost, which was used as the basis for producing DOM Level 1. Thanks also to Gavin Nicol, who wrote the scripts which run on top of cost. Arnaud Le Hors and Philippe Le Hégaret maintained the scripts.

After DOM Level 1, we used Xerces as the basis DOM implementation and wish to thank the authors. Philippe Le Hégaret and Arnaud Le Hors wrote the Java programs which are the DOM application.

Thanks also to Jan Kärrman, author of html2ps, which we use in creating the PostScript version of the specification.

Glossary

Editors
Arnaud Le Hors, W3C
Robert S. Sutor, IBM Research (for DOM Level 1)

Several of the following term definitions have been borrowed or modified from similar definitions in other W3C or standards documents. See the links within the definitions for more information.

tokenized
The description given to various information items (for example, attribute values of various types, but not including the StringType CDATA) after having been processed by the XML processor. The process includes stripping leading and trailing white space, and replacing multiple space characters by one. See the definition of tokenized type.

well-formed document
A document is *well-formed* if it is tag valid and entities are limited to single elements (i.e., single sub-trees).

References

For the latest version of any W3C specification please consult the list of W3C Technical Reports available at http://www.w3.org/TR.

F.1: Normative references

DOM Level 3 Core

W3C (World Wide Web Consortium) Document Object Model Level 3 Core Specification, June 2001. Available at http://www.w3.org/TR/2001/WD-DOM-Level-3-Core-20010605

ECMAScript

ISO (International Organization for Standardization). ISO/IEC 16262:1998. ECMAScript Language Specification. Available from ECMA (European Computer Manufacturers Association) at http://www.ecma.ch/ecma1/STAND/ECMA-262.HTM

Java

Sun Microsystems Inc. The Java Language Specification, James Gosling, Bill Joy, and Guy Steele, September 1996. Available at http://java.sun.com/docs/books/jls

OMGIDL

OMG (Object Management Group) IDL (Interface Definition Language) defined in The Common Object Request Broker: Architecture and Specification, version 2.3.1, October 1999. Available from http://www.omg.org

XPath 1.0

W3C (World Wide Web Consortium) XML Path Language (XPath) Version 1.0, November 1999. Available at http://www.w3.org/TR/1999/REC-xpath-19991116.

F.2: Informative references

XQuery 1.0 and XPath 2.0 Data Model

W3C (World Wide Web Consortium) XQuery 1.0 and XML Path 2.0 Data Model, June 2001. Available at http://www.w3.org/TR/query-datamodel.

F.2: Informative references

Index