Document Object Model (DOM) Level 3 XPath Specification

Version 1.0

W3C Working Draft 28 March 2002

This version:
http://www.w3.org/TR/2002/WD-DOM-Level-3-XPath-20020328
(PostScript file , PDF file , plain text , ZIP file , single HTML file)
Latest version:
http://www.w3.org/TR/DOM-Level-3-XPath
Previous version:
http://www.w3.org/TR/2002/WD-DOM-Level-3-XPath-20020208

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].

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 is a Last Call Working Draft for review by W3C members and other interested parties. The Last Call review **period ends on May 1, 2002**. Please send reviews before the review period ends to the public mailing list www-dom@w3.org. An archive is available at http://lists.w3.org/Archives/Public/www-dom/.

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.

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 © 1994-2002 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

Editor:
Ray Whitmer, Netscape/AOL

1.1. Introduction

XPath 1.0 [XPath 1.0] is becoming an important part of a variety of many specifications including XForms, XPointer, XSL, XML Query, 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* [p.39] issues have plagued each attempt to get a list of DOM nodes matching specific criteria, as would be expected for an XPath *API* [p.39] . There have also traditionally been *model* [p.39] mismatches between DOM and XPath. This proposal specifies new interfaces and approaches to resolving these issues.

Issue XPath-2:

There should be a function to generate an XPath expression which addresses any arbitrary node within the DOM.

Resolution: Out of scope.

That is out of scope for the current requirements of this module. It could be a quite useful feature but probably not as a part of this module.

1.2. Mapping DOM to XPath

This section considers the differences between the Document Object Model [DOM Level 2 Core] and the XPath 1.0 model [XPath 1.0].

1.2.1. Text Nodes

The XPath model relies on the XML Information Set [XML Information set] ands represents *Character Information Items* in 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` or `CDATASection` node of any logical XPath text will be returned in the node set. Applications using XPath in an environment with fragmented text nodes must manually gather the text of a single logical text node possibly from multiple nodes beginning with the first `Text` node or `CDATASection` node returned by the implementation.

Note: In an attempt to better implement the XML Information Set, DOM Level 3 Core [DOM Level 3 Core] adds the attribute `wholeText` on the `Text` interface for retrieving the whole text for *logically-adjacent Text nodes* [p.39] and the method `replaceWholeText` for replacing those nodes.

1.2.2. Namespace Nodes

The XPath model expects namespace nodes for each in-scope namespace to be attached to each *element* [p.39] . DOM and certain other W3C Information Set 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 returns a new node of type `XPATH_NAMESPACE_NODE`, defined in the `XPathNamespace` [p.23] interface, to properly preserve identity and ordering. This node type is only visible using the XPath evaluation methods.

1.2.3. Document order

The *document order* [p.39] of nodes in the DOM Core has been defined to be compatible with the *XPath document order*. The XPath DOM is extending the document order of the DOM Core to include the `XPathNamespace` [p.23] nodes. Element nodes occur before their children. The attribute nodes and namespace nodes of an element occur before the children of the element. The namespace nodes are defined to occur before the attribute nodes. The relative order of namespace nodes is implementation-dependent. The relative order of attribute nodes is implementation-dependent. The `compareTreePosition` method on the `Node` interface defined in the DOM Core must compare the `XPathNamespace` nodes using this extending document order if the XPath DOM module is supported.

1.3. Interfaces

An implementation is DOM Level 3 XPath conformant if it supports the Core module defined in [DOM Level 2 Core] and the module defined in this specification. An implementation conforms to a DOM module if it supports all the interfaces for that module and the associated semantics.

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 XPath 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 2 Core specification [DOM Level 2 Core].

A DOM implementation must not return `true` to the `hasFeature(feature, version)` method of the DOMImplementation interface for that feature unless the implementation conforms to that module. The version number for the feature used in this document is `"3.0"`.

Exception *XPathException*

A new exception has been created for exceptions specific to these XPath interfaces.

IDL Definition

```
exception XPathException {
  unsigned short   code;
};
// XPathExceptionCode
const unsigned short      INVALID_EXPRESSION_ERR         = 1;
const unsigned short      TYPE_ERR                       = 2;
```

Definition group *XPathExceptionCode*

Defined Constants

`INVALID_EXPRESSION_ERR`

If the expression has a syntax error or otherwise is not a legal expression according to the rules of the specific `XPathEvaluator` [p.11] . If the `XPathEvaluator` was obtained by casting the document, the expression must be XPath 1.0 with no special extension functions.

Issue XPath-4:

A separate exception should be raised if there are problems resolving namespaces.

Resolution: Yes. These now raise DOMException with the code NAMESPACE_ERR.

`TYPE_ERR`

If the expression cannot be converted to return the specified type.

Interface *XPathEvaluator*

The evaluation of XPath expressions is provided by `XPathEvaluator`, which will provide evaluation of XPath 1.0 expressions with no specialized extension functions or variables. It is expected that the `XPathEvaluator` interface will be implemented on the same object which implements the `Document` interface in an implementation which supports the XPath DOM module. `XPathEvaluator` implementations may be available from other sources that may provide support for special extension functions or variables which are not defined in this specification.

Issue XPath-16:

The methods of XPathExpression should be named with more-XPath- specific names because the interface will often be implemented by the same object which implements document.

Resolution: No change.

The point of interfaces is to localize the implementing namespace. This would make the method names unnecessarily long and complex even though there are no conflicts in the interface itself. The new core method getInterface is designed for discovering interfaces of additional modules that may not be directly implemented on the objects to which they are attached. This could be used to implement XPath on a separate object. The user only refers to the separate interfaces and not the proprietary aggregate implementation.

Issue XPath-22:

Should entity refs be supported so that queries can be made on them?

Resolution: No change.

We will not do this now. They are not part of the XPath data model. Note that they may be present in the hierarchy of returned nodes, but may not directly be requested or returned in the node set.

Issue XPath-24:

What does createResult create when one wants to reuse the XPath?

Resolution: It is not useful.

Removed method.

Issue XPath-27:

Should ordering be a separate flag, or a type of result that can be requested. As a type of result, it can be better optimized in implementations.

Resolution: It makes sense as a type of result. Changed.

Removed method.

Issue XPath-37:

Implementing XPathEvaluator on Document can be a problem due to conflicts in the names of the methods.

The working group finds no better solution. GetInterface in Level 3 permits the object to be implemented separately. We should be committed to this. We will leave this issue open to see if we get more feedback on it.

Issue XPath-38:

How does this interface adapt to XPath 2.0 and other query languages.

Resolution: No change.

This interface is not intended to adapt to XPath 2.0 or other languages. The models of these are likely to be incompatible enough to require new APIs.

For alternate implementations that can use this API, it can be obtained from different sources.

Issue XPath-39:

Support for custom variables and functions would be very useful.

Resolution: No change.

It is possible for an implementation to supply alternative sources of an XPathEvaluator that can be customized with a custom variable and function context. We do not specify how this is accomplished. It is too complex to address in this version of the XPath DOM.

IDL Definition

```
interface XPathEvaluator {
  XPathExpression    createExpression(in DOMString expression,
                                      in XPathNSResolver resolver)
                                        raises(XPathException,
                                               DOMException);
  XPathNSResolver    createNSResolver(in Node nodeResolver);
  XPathResult        evaluate(in DOMString expression,
                              in Node contextNode,
                              in XPathNSResolver resolver,
                              in unsigned short type,
                              in XPathResult result)
                                        raises(XPathException,
                                               DOMException);
};
```

Methods

`createExpression`

Creates a parsed XPath expression with resolved namespaces. This is useful when an expression will be reused in an application since it makes it possible to compile the expression string into a more efficient internal form and preresolve all *namespace prefixes* [p.39] which occur within the expression.

Issue XPath-5:

createExpression should not raise exceptions about type coercion.

Resolution: This was already fixed in the public draft.

Parameters

`expression` of type `DOMString`

The XPath expression string to be parsed.

`resolver` of type `XPathNSResolver` [p.17]
The `resolver` permits translation of prefixes within the XPath expression into appropriate *namespace URIs* [p.39] . If this is specified as `null`, any *namespace prefix* [p.39] within the expression will result in `DOMException` being thrown with the code `NAMESPACE_ERR`.

Return Value

`XPathExpression` [p.16]	The compiled form of the XPath expression.

Exceptions

`XPathException` [p.10]	INVALID_EXPRESSION_ERR: Raised if the expression is not legal according to the rules of the `XPathEvaluator`i
`DOMException`	NAMESPACE_ERR: Raised if the expression contains *namespace prefixes* [p.39] which cannot be resolved by the specified `XPathNSResolver` [p.17] .

`createNSResolver`
Adapts any DOM node to resolve namespaces so that an XPath expression can be easily evaluated relative to the context of the node where it appeared within the document. This adapter works by calling the method `lookupNamespacePrefix` on `Node`.
Issue XPath-6:
It should be possible to create an XPathNSResolver that does not rely on a node, but which implements a map of resolutions that can be added to by the application.
Resolution: No change.
The application can easily create this, which was why the interface was designed as it is. The specification will not require a specific factory at this time for application populated maps.
Issue XPath-14:
There should be type restrictions on which types of nodes may be adapted by createNSResolver.
Resolution: No change.
The namespace methods on the Node interface of the Level 3 core may be called without exception on all node types. In some cases no non-null namespace resolution will ever be returned. That is what may also be expected of this adapter.

Parameters

`nodeResolver` of type `Node`
The node to be used as a context for namespace resolution.

Return Value

`XPathNSResolver` [p.17]	`XPathNSResolver` which resolves namespaces with respect to the definitions in scope for a specified node.

No Exceptions

`evaluate`

Evaluates an XPath expression string and returns a result of the specified type if possible.

Issue XPath-17-18:

An exception needs to be raised when an XPath expression is evaluated on a node such as an EntityReference which cannot serve as an XPath context node.

Resolution: Done: NOT_SUPPORTED_ERR.

Issue XPath-19:

A description is needed of what happens when the node passed to the evaluation function is a Text or CDATASection in the DOM case where the text may be fragmented between text nodes.

Resolution: Done.

Issue XPath-20:

Eliminate the evaluate method from XPathEvaluator, forcing everyone to create expressions.

Resolution: No change.

Any implementor can easily implement it by creating an expression. Having it available as a separate routine is a convenience and may be an optimization as well in some cases.

Issue XPath-21:

Revert to multiple evaluateAs methods instead of passing a type code.

Resolution: No change.

This is an alternative which eliminates a method argument while adding methods, but the type code is used to designate the type on returns anyway and using it as an argument to specify any coercion seems natural to many.

Issue XPath-13:

Error exceptions are needed when there is a mismatch between the implementation of XPathEvaluator and the context node being evaluated.

Resolution: Done: WRONG_DOCUMENT_ERR

Issue XPath-29:

Concern that the XPath API should only support natural results of XPath expression, without convenience coercion or alternative representations. Any special thing such as ordering should be added later to result

Resolution: No change.

We have significant use cases for returning alternative types and representations by explicit request in advance.

Issue XPath-30:

Eliminate the reusable result argument.

Resolution: No change.

No. We have use cases for it, and there is already an implementation showing there is nothing wrong with it.

Issue XPath-31:

State that the XPathNSResolver argument may be a function in Javascript.

Resolution: Yes.

Issue XPath-32:

There is an exception when there is a problem parsing the expression, but none when

there is a problem evaluating the expression.
Resolution: No change.
If the expression parsing was OK, then the worst that can happen is an empty result is returned.

Issue XPath-33:
When requesting any type, the implementation should be permitted to return any type of node set, i.e. ordered or unordered, it finds convenient.
Resolution: No change.
The iterator it returns may contain ordered results, but identifying it as such produces undesirable results, because it would create complexity for the user -- requiring checking two types to see if the result was a node set -- or incompatibility caused by assuming it was always the one returned by a particular implementation the developer was using.

Issue XPath-35:
NAMESPACE_ERR description is not appropriate to the way it is being used here.
Resolution: Make the description of NAMESPACE_ERR in the core specification more general.

Issue XPath-36:
Should the INVALID_EXPRESSION_ERR be INVALID_SYNTAX_ERR?
Resolution: No change.
We can improve the description of the error, but the name is appropriate as-is. It covers not only syntax errors but expression errors, such as when the implementation has no custom functions or variables but the expression specifies custom functions or variables.

Parameters

`expression` of type `DOMString`
The XPath expression string to be parsed and evaluated.

`contextNode` of type `Node`
The `context` is context node for the evaluation of this XPath expression. If the XPathEvaluator was obtained by casting the `Document` then this must be owned by the same document and must be a `Document`, `Element`, `Attribute`, `Text`, `CDATASection`, `Comment`, `ProcessingInstruction`, or `XPathNamespace` [p.23] node. If the context node is a `Text` or a `CDATASection`, then the context is interpreted as the whole logical text node as seen by XPath, unless the node is empty in which case it may not serve as the XPath context.

`resolver` of type `XPathNSResolver` [p.17]
The `resolver` permits translation of prefixes within the XPath expression into appropriate *namespace URIs* [p.39] . If this is specified as `null`, any *namespace prefix* [p.39] within the expression will result in `DOMException` being thrown with the code `NAMESPACE_ERR`.

`type` of type `unsigned short`
If a specific `type` is specified, then the result will be coerced to return the specified type relying on *XPath type conversions* and fail if the desired coercion is not possible. This must be one of the type codes of `XPathResult` [p.18] .

`result` of type `XPathResult` [p.18]
The `result` specifies a specific `XPathResult` which may be reused and returned by this method. If this is specified as `null`or the implementation cannot reuse the specified result, a new `XPathResult` will be constructed and returned.

Return Value

`XPathResult` [p.18]	The result of the evaluation of the XPath expression.

Exceptions

`XPathException` [p.10]	INVALID_EXPRESSION_ERR: Raised if the expression is not legal according to the rules of the `XPathEvaluator`i TYPE_ERR: Raised if the result cannot be converted to return the specified type.
`DOMException`	NAMESPACE_ERR: Raised if the expression contains *namespace prefixes* [p.39] which cannot be resolved by the specified `XPathNSResolver` [p.17] . WRONG_DOCUMENT_ERR: The Node is from a document that is not supported by this XPathEvaluator. NOT_SUPPORTED_ERR: The Node is not a type permitted as an XPath context node.

Interface *XPathExpression*

The `XPathExpression` interface represents a parsed and resolved XPath expression.

Issue XPath-8:
The evaluateExpression method should be moved to the XPathExpression interface so you do not have to use / pass two interfaces just to use it.
Resolution: Done.

Issue XPath-15:
XPathExpression should have a public reference to the XPathEvaluator that created it.
Resolution: No change.
Lacks justification.

IDL Definition

```
interface XPathExpression {
  XPathResult        evaluate(in Node contextNode,
                              in unsigned short type,
                              in XPathResult result)
                                        raises(XPathException,
                                               DOMException);
};
```

Methods

`evaluate`

Evaluates this XPath expression and returns a result.

Parameters

`contextNode` of type `Node`

The `context` is context node for the evaluation of this XPath expression.
If the XPathEvaluator was obtained by casting the `Document` then this must be owned by the same document and must be a `Document`, `Element`, `Attribute`, `Text`, `CDATASection`, `Comment`, `ProcessingInstruction`, or `XPathNamespace` [p.23] node.
If the context node is a `Text` or a `CDATASection`, then the context is interpreted as the whole logical text node as seen by XPath, unless the node is empty in which case it may not serve as the XPath context.

`type` of type `unsigned short`

If a specific `type` is specified, then the result will be coerced to return the specified type relying on XPath conversions and fail if the desired coercion is not possible. This must be one of the type codes of `XPathResult` [p.18] .

`result` of type `XPathResult` [p.18]

The `result` specifies a specific `XPathResult` which may be reused and returned by this method. If this is specified as `null`or the implementation cannot reuse the specified result, a new `XPathResult` will be constructed and returned.

Return Value

`XPathResult` [p.18]	The result of the evaluation of the XPath expression.

Exceptions

`XPathException` [p.10]	TYPE_ERR: Raised if the result cannot be converted to return the specified type.
`DOMException`	WRONG_DOCUMENT_ERR: The Node is from a document that is not supported by the XPathExpression that created this `XPathExpression`. NOT_SUPPORTED_ERR: The Node is not a type permitted as an XPath context node.

Interface *XPathNSResolver*

The `XPathNSResolver` interface permit `prefix` strings in the expression to be properly bound to `namespaceURI` strings. `XPathEvaluator` [p.11] can construct an implementation of `XPathNSResolver` from a node, or the interface may be implemented by any application.

IDL Definition

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

Methods

`lookupNamespaceURI`

Look up the *namespace URI* [p.39] associated to the given *namespace prefix* [p.39] . The XPath evaluator must never call this with a `null` or empty argument, because the result of doing this is undefined.

Issue XPath-9:

Null / empty prefix passed to XPathNSResolver should return default namespace.

Resolution: Do not permit `null`to be passed in invocation, allowing the implementation, if shared, to do anything it wants with a passed `null`.

It would be confusing to specify more than this since the resolution of namespaces for XPath expressions never requires the default namespace.

Issue XPath-10:

Null returns are problematic.

Resolution: No change.

They should be adequately addressed in core. Some implementations have not properly supported them, but they will be fixed to be compliant. Bindings are still free to choose alternative representations of `null`where required.

Parameters

`prefix` of type `DOMString`

The prefix to look for.

Return Value

`DOMString`	Returns the associated *namespace URI* [p.39] or `null` if none is found.

No Exceptions

Interface *XPathResult*

The `XPathResult` interface represents the result of the evaluation of an XPath expression within the context of a particular node. Since evaluation of an XPath expression can result in various result types, this object makes it possible to discover and manipulate the type and value of the result.

Issue XPath-23:

Should there be a flag on the result to say whether an iteration has become invalid?

Resolution: Yes.

Added the boolean attribute `invalidIteratorState`

Issue XPath-25:

Should there be a reset method on the result in case someone wants to iterate the result multiple times?

It may be more trouble than it is worth, because the user can request a new query. See if there are use cases.

Issue XPath-26:

It might be better to consolidate the interfaces and just move the snapshot and iterator functions to the result object.

Resolution: Yes.

The result of the consolidation looks good and unless there are great objections, this is how it will be.

Issue XPath-28:

There is concern that the result cannot represent multiple strings, which is a possible result of XPath 2.0. on them?

Resolution: No change.

This is not part of the XPath 1.0 data model. We cannot plan well for the XPath 2.0 data model at this point. Most likely a new API will be required for XPath 2.0

IDL Definition

```
interface XPathResult {

  // XPathResultType
  const unsigned short      ANY_TYPE                      = 0;
  const unsigned short      NUMBER_TYPE                   = 1;
  const unsigned short      STRING_TYPE                   = 2;
  const unsigned short      BOOLEAN_TYPE                  = 3;
  const unsigned short      UNORDERED_NODE_ITERATOR_TYPE  = 4;
  const unsigned short      ORDERED_NODE_ITERATOR_TYPE    = 5;
  const unsigned short      UNORDERED_NODE_SNAPSHOT_TYPE  = 6;
  const unsigned short      ORDERED_NODE_SNAPSHOT_TYPE    = 7;
  const unsigned short      ANY_UNORDERED_NODE_TYPE       = 8;
  const unsigned short      FIRST_ORDERED_NODE_TYPE       = 9;

  readonly attribute unsigned short  resultType;
  readonly attribute double          numberValue;
                                        // raises(XPathException) on retrieval

  readonly attribute DOMString       stringValue;
                                        // raises(XPathException) on retrieval

  readonly attribute boolean         booleanValue;
                                        // raises(XPathException) on retrieval

  readonly attribute Node            singleNodeValue;
                                        // raises(XPathException) on retrieval

  readonly attribute boolean         invalidIteratorState;
  readonly attribute unsigned long   snapshotLength;
                                        // raises(XPathException) on retrieval

  Node               iterateNext()
                                        raises(XPathException,
                                               DOMException);
  Node               snapshotItem(in unsigned long index)
                                        raises(XPathException);
};
```

Definition group *XPathResultType*

An integer indicating what type of result this is.

Defined Constants

`ANY_TYPE`
: This code does not represent a specific type. An evaluation of an XPath expression will never produce this type. If this type is requested, then the evaluation returns whatever type naturally results from evaluation of the expression.
If the natural result is a node set when `ANY_TYPE` was requested, then `UNORDERED_NODE_ITERATOR_TYPE` is always the resulting type. Any other representation of a node set must be explicitly requested.

`ANY_UNORDERED_NODE_TYPE`
: The result is a *node set* as defined by [XPath 1.0] and will be accessed as a single node, which may be `null`if the node set is empty. Document modification does not invalidate the node, but may mean that the result node no longer corresponds to the current document. This is a convenience that permits optimization since the implementation can stop once any node in the in the resulting set has been found.
If there are more than one node in the actual result, the single node returned may not be the first in document order.

`BOOLEAN_TYPE`
: The result is a *boolean* as defined by [XPath 1.0]. Document modification does not invalidate the boolean, but may mean that reevaluation would not yield the same boolean.

`FIRST_ORDERED_NODE_TYPE`
: The result is a *node set* as defined by [XPath 1.0] and will be accessed as a single node, which may be `null` if the node set is empty. Document modification does not invalidate the node, but may mean that the result node no longer corresponds to the current document. This is a convenience that permits optimization since the implementation can stop once the first node in document order of the resulting set has been found.
If there are more than one node in the actual result, the single node returned will be the first in document order.

`NUMBER_TYPE`
: The result is a *number* as defined by [XPath 1.0]. Document modification does not invalidate the number, but may mean that reevaluation would not yield the same number.

`ORDERED_NODE_ITERATOR_TYPE`
: The result is a node set as defined by [XPath 1.0] that will be accessed iteratively, which will produce document-ordered nodes. Document modification invalidates the iteration.

`ORDERED_NODE_SNAPSHOT_TYPE`
: The result is a *node set* as defined by [XPath 1.0] that will be accessed as a snapshot list of nodes that will be in original document order. Document modification does not invalidate the snapshot but may mean that reevaluation would not yield the same snapshot and nodes in the snapshot may have been altered, moved, or removed from the document.

`STRING_TYPE`
: The result is a *string* as defined by [XPath 1.0]. Document modification does not invalidate the string, but may mean that the string no longer corresponds to the current document.

`UNORDERED_NODE_ITERATOR_TYPE`
: The result is a *node set* as defined by [XPath 1.0] that will be accessed iteratively, which may not produce nodes in a particular order. Document modification invalidates the iteration.

  This is the default type returned if the result is a node set and `ANY_TYPE` is requested.

`UNORDERED_NODE_SNAPSHOT_TYPE`
: The result is a *node set* as defined by [XPath 1.0] that will be accessed as a snapshot list of nodes that may not be in a particular order. Document modification does not invalidate the snapshot but may mean that reevaluation would not yield the same snapshot and nodes in the snapshot may have been altered, moved, or removed from the document.

Attributes

`booleanValue` of type `boolean`, readonly
: The value of this boolean result.

  Exceptions on retrieval

`XPathException` [p.10]	TYPE_ERR: raised if `resultType` is not `BOOLEAN_TYPE`.

`invalidIteratorState` of type `boolean`, readonly
: Signifies that the iterator has become invalid. True if `resultType` is `UNORDERED_NODE_ITERATOR_TYPE` or `ORDERED_NODE_ITERATOR_TYPE` and the document has been modified since this result was returned.

`numberValue` of type `double`, readonly
: The value of this number result.

  Exceptions on retrieval

`XPathException` [p.10]	TYPE_ERR: raised if `resultType` is not `NUMBER_TYPE`.

`resultType` of type `unsigned short`, readonly
: A code representing the type of this result, as defined by the type constants.

`singleNodeValue` of type `Node`, readonly
: The value of this single node result, which may be `null`.

  Exceptions on retrieval

`XPathException` [p.10]	TYPE_ERR: raised if `resultType` is not `ANY_UNORDERED_NODE_TYPE` or `FIRST_ORDERED_NODE_TYPE`.

`snapshotLength` of type `unsigned long`, readonly
The number of nodes in the result snapshot. Valid values for snapshotItem indices are `0` to `snapshotLength-1` inclusive.
Exceptions on retrieval

`XPathException` [p.10]	TYPE_ERR: raised if `resultType` is not `UNORDERED_NODE_SNAPSHOT_TYPE` or `ORDERED_NODE_SNAPSHOT_TYPE`.

`stringValue` of type `DOMString`, readonly
The value of this string result.
Exceptions on retrieval

`XPathException` [p.10]	TYPE_ERR: raised if `resultType` is not `STRING_TYPE`.

Methods

`iterateNext`
Iterates and returns the next node from the node set or `null`if there are no more nodes.
Return Value

`Node`	Returns the next node.

Exceptions

`XPathException` [p.10]	TYPE_ERR: raised if `resultType` is not `UNORDERED_NODE_ITERATOR_TYPE` or `ORDERED_NODE_ITERATOR_TYPE`.
`DOMException`	INVALID_STATE_ERR: The document has been mutated since the result was returned.

No Parameters

`snapshotItem`
Returns the `index`th item in the snapshot collection. If `index` is greater than or equal to the number of nodes in the list, this method returns `null`. Unlike the iterator result, the snapshot does not become invalid, but may not correspond to the current document if it is mutated.
Parameters
`index` of type `unsigned long`
Index into the snapshot collection.
Return Value

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

Exceptions

`XPathException` [p.10]	TYPE_ERR: raised if `resultType` is not `UNORDERED_NODE_SNAPSHOT_TYPE` or `ORDERED_NODE_SNAPSHOT_TYPE`.

Interface *XPathNamespace*

The `XPathNamespace` interface is returned by `XPathResult` [p.18] interfaces to represent the XPath namespace node type that DOM lacks. There is no public constructor for this node type. Attempts to place it into a hierarchy or a NamedNodeMap result in a `DOMException` with the code `HIERARCHY_REQUEST_ERR`. This node is *read only* [p.40] , so methods or setting of attributes that would mutate the node result in a DOMException with the code `NO_MODIFICATION_ALLOWED_ERR`.

The core specification describes attributes of the `Node` interface that are different for different node node types but does not describe `XPATH_NAMESPACE_NODE`, so here is a description of those attributes for this node type. All attributes of `Node` not described in this section have a `null` or `false` value.

`ownerDocument` matches the `ownerDocument` of the `ownerElement` even if the element is later adopted.

`prefix` is the prefix of the namespace represented by the node.

`nodeName` is the same as `prefix`.

`nodeType` is equal to `XPATH_NAMESPACE_NODE`.

`namespaceURI` is the namespace URI of the namespace represented by the node.

`adoptNode`, `cloneNode`, and `importNode` fail on this node type by raising a `DOMException` with the code `NOT_SUPPORTED_ERR`.

Issue XPath-12:
importNode should also fail on XPathNamespace nodes.
Resolution: This was already fixed in the public draft.

Issue XPath-3:
The Namespace node should be added to DOM Level 3 core and should be available via a read-only NamedNodeMap on element to reduce the confusion of adding a special node type for XPath.
Resolution: No change.
There are no known problems with this add-on node type and uses beyond XPath are not anticipated.

Issue XPath-11:

`Node.namespaceValue` should be identical to Node.namespaceURI and not `null`.

Resolution: No change.

It is not clear why it should be this way since the infoset does not dictate it.

IDL Definition

```
interface XPathNamespace : Node {

  // XPathNodeType
  const unsigned short      XPATH_NAMESPACE_NODE           = 13;

  readonly attribute Element         ownerElement;
};
```

Definition group *XPathNodeType*

An integer indicating which type of node this is.

Note: There is currently only one type of node which is specific to XPath. The numbers in this list must not collide with the values assigned to core node types.

Defined Constants

`XPATH_NAMESPACE_NODE`

The node is a `Namespace`.

Attributes

`ownerElement` of type `Element`, readonly

The `Element` on which the namespace was in scope when it was requested. This does not change on a returned namespace node even if the document changes such that the namespace goes out of scope on that *element* [p.39] and this node is no longer found there by XPath.

Appendix A: IDL Definitions

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

The IDL files are also available as:
http://www.w3.org/TR/2002/WD-DOM-Level-3-XPath-20020328/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;
  typedef dom::Element Element;

  interface XPathNSResolver;
  interface XPathResult;
  interface XPathExpression;

  exception XPathException {
    unsigned short   code;
  };
  // XPathExceptionCode
  const unsigned short      INVALID_EXPRESSION_ERR         = 1;
  const unsigned short      TYPE_ERR                       = 2;


  interface XPathEvaluator {
    XPathExpression    createExpression(in DOMString expression,
                                        in XPathNSResolver resolver)
                                        raises(XPathException,
                                               dom::DOMException);
    XPathNSResolver    createNSResolver(in Node nodeResolver);
    XPathResult        evaluate(in DOMString expression,
                                in Node contextNode,
                                in XPathNSResolver resolver,
                                in unsigned short type,
                                in XPathResult result)
                                        raises(XPathException,
                                               dom::DOMException);
  };

  interface XPathExpression {
    XPathResult        evaluate(in Node contextNode,
```

```
                              in unsigned short type,
                              in XPathResult result)
                                      raises(XPathException,
                                             dom::DOMException);
};

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

interface XPathResult {

  // XPathResultType
  const unsigned short      ANY_TYPE                        = 0;
  const unsigned short      NUMBER_TYPE                     = 1;
  const unsigned short      STRING_TYPE                     = 2;
  const unsigned short      BOOLEAN_TYPE                    = 3;
  const unsigned short      UNORDERED_NODE_ITERATOR_TYPE    = 4;
  const unsigned short      ORDERED_NODE_ITERATOR_TYPE      = 5;
  const unsigned short      UNORDERED_NODE_SNAPSHOT_TYPE    = 6;
  const unsigned short      ORDERED_NODE_SNAPSHOT_TYPE      = 7;
  const unsigned short      ANY_UNORDERED_NODE_TYPE         = 8;
  const unsigned short      FIRST_ORDERED_NODE_TYPE         = 9;

  readonly attribute unsigned short  resultType;
  readonly attribute double          numberValue;
                                      // raises(XPathException) on retrieval

  readonly attribute DOMString       stringValue;
                                      // raises(XPathException) on retrieval

  readonly attribute boolean         booleanValue;
                                      // raises(XPathException) on retrieval

  readonly attribute Node            singleNodeValue;
                                      // raises(XPathException) on retrieval

  readonly attribute boolean         invalidIteratorState;
  readonly attribute unsigned long   snapshotLength;
                                      // raises(XPathException) on retrieval

  Node               iterateNext()
                                      raises(XPathException,
                                             dom::DOMException);
  Node               snapshotItem(in unsigned long index)
                                      raises(XPathException);
};

interface XPathNamespace : Node {

  // XPathNodeType
  const unsigned short      XPATH_NAMESPACE_NODE           = 13;

  readonly attribute Element         ownerElement;
```

```
  };
};

#endif // _XPATH_IDL_
```

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/2002/WD-DOM-Level-3-XPath-20020328/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;
    // XPathExceptionCode
    public static final short INVALID_EXPRESSION_ERR    = 1;
    public static final short TYPE_ERR                  = 2;

}
```

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

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

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

public interface XPathEvaluator {
    public XPathExpression createExpression(String expression,
                                            XPathNSResolver resolver)
                                            throws XPathException, DOMException;

    public XPathNSResolver createNSResolver(Node nodeResolver);

    public XPathResult evaluate(String expression,
                                Node contextNode,
                                XPathNSResolver resolver,
                                short type,
                                XPathResult result)
                                throws XPathException, DOMException;

}
```

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

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

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

public interface XPathExpression {
    public XPathResult evaluate(Node contextNode,
                                short type,
                                XPathResult result)
                                throws XPathException, DOMException;

}
```

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

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

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

}
```

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

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

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

public interface XPathResult {
    // XPathResultType
    public static final short ANY_TYPE                 = 0;
    public static final short NUMBER_TYPE              = 1;
    public static final short STRING_TYPE              = 2;
    public static final short BOOLEAN_TYPE             = 3;
    public static final short UNORDERED_NODE_ITERATOR_TYPE = 4;
    public static final short ORDERED_NODE_ITERATOR_TYPE = 5;
    public static final short UNORDERED_NODE_SNAPSHOT_TYPE = 6;
    public static final short ORDERED_NODE_SNAPSHOT_TYPE = 7;
    public static final short ANY_UNORDERED_NODE_TYPE   = 8;
    public static final short FIRST_ORDERED_NODE_TYPE   = 9;

    public short getResultType();

    public double getNumberValue()
                                    throws XPathException;

    public String getStringValue()
                                    throws XPathException;

    public boolean getBooleanValue()
                                    throws XPathException;
```

```
    public Node getSingleNodeValue()
                                    throws XPathException;

    public boolean getInvalidIteratorState();

    public int getSnapshotLength()
                                    throws XPathException;

    public Node iterateNext()
                            throws XPathException, DOMException;

    public Node snapshotItem(int index)
                             throws XPathException;

}
```

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

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

import org.w3c.dom.Element;
import org.w3c.dom.Node;

public interface XPathNamespace extends Node {
    // XPathNodeType
    public static final short XPATH_NAMESPACE_NODE      = 13;

    public Element getOwnerElement();

}
```

Appendix C: ECMAScript Language Binding

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

Properties of the **XPathException** Constructor function:

XPathException.INVALID_EXPRESSION_ERR

The value of the constant **XPathException.INVALID_EXPRESSION_ERR** is **1**.

XPathException.TYPE_ERR

The value of the constant **XPathException.TYPE_ERR** is **2**.

Objects that implement the **XPathException** interface:

Properties of objects that implement the **XPathException** interface:

code

This property is a **Number**.

Objects that implement the **XPathEvaluator** interface:

Functions of objects that implement the **XPathEvaluator** interface:

createExpression(expression, resolver)

This function returns an object that implements the **XPathExpression** interface.
The **expression** parameter is a **String**.
The **resolver** parameter is an object that implements the **XPathNSResolver** interface.
This function can raise an object that implements the **XPathException** interface or the **DOMException** interface.

createNSResolver(nodeResolver)

This function returns an object that implements the **XPathNSResolver** interface.
The **nodeResolver** parameter is an object that implements the **Node** interface.

evaluate(expression, contextNode, resolver, type, result)

This function returns an object that implements the **XPathResult** interface.
The **expression** parameter is a **String**.
The **contextNode** parameter is an object that implements the **Node** interface.
The **resolver** parameter is an object that implements the **XPathNSResolver** interface.
The **type** parameter is a **Number**.
The **result** parameter is an object that implements the **XPathResult** interface.
This function can raise an object that implements the **XPathException** interface or the **DOMException** interface.

Objects that implement the **XPathExpression** interface:

Functions of objects that implement the **XPathExpression** interface:

evaluate(contextNode, type, result)

This function returns an object that implements the **XPathResult** interface.
The **contextNode** parameter is an object that implements the **Node** interface.
The **type** parameter is a **Number**.
The **result** parameter is an object that implements the **XPathResult** interface.
This function can raise an object that implements the **XPathException** interface or the **DOMException** interface.

Objects that implement the **XPathNSResolver** interface:

Functions of objects that implement the **XPathNSResolver** interface:

lookupNamespaceURI(prefix)

This function returns a **String**.

The **prefix** parameter is a **String**.

Properties of the **XPathResult** Constructor function:

XPathResult.ANY_TYPE

The value of the constant **XPathResult.ANY_TYPE** is **0**.

XPathResult.NUMBER_TYPE

The value of the constant **XPathResult.NUMBER_TYPE** is **1**.

XPathResult.STRING_TYPE

The value of the constant **XPathResult.STRING_TYPE** is **2**.

XPathResult.BOOLEAN_TYPE

The value of the constant **XPathResult.BOOLEAN_TYPE** is **3**.

XPathResult.UNORDERED_NODE_ITERATOR_TYPE

The value of the constant **XPathResult.UNORDERED_NODE_ITERATOR_TYPE** is **4**.

XPathResult.ORDERED_NODE_ITERATOR_TYPE

The value of the constant **XPathResult.ORDERED_NODE_ITERATOR_TYPE** is **5**.

XPathResult.UNORDERED_NODE_SNAPSHOT_TYPE

The value of the constant **XPathResult.UNORDERED_NODE_SNAPSHOT_TYPE** is **6**.

XPathResult.ORDERED_NODE_SNAPSHOT_TYPE

The value of the constant **XPathResult.ORDERED_NODE_SNAPSHOT_TYPE** is **7**.

XPathResult.ANY_UNORDERED_NODE_TYPE

The value of the constant **XPathResult.ANY_UNORDERED_NODE_TYPE** is **8**.

XPathResult.FIRST_ORDERED_NODE_TYPE

The value of the constant **XPathResult.FIRST_ORDERED_NODE_TYPE** is **9**.

Objects that implement the **XPathResult** interface:

Properties of objects that implement the **XPathResult** interface:

resultType

This read-only property is a **Number**.

numberValue

This read-only property is an object that implements the **double** interface and can raise an object that implements the **XPathException** interface on retrieval.

stringValue

This read-only property is a **String** and can raise an object that implements the **XPathException** interface on retrieval.

booleanValue

This read-only property is a **Boolean** and can raise an object that implements the **XPathException** interface on retrieval.

singleNodeValue

This read-only property is an object that implements the **Node** interface and can raise an object that implements the **XPathException** interface on retrieval.

invalidIteratorState

This read-only property is a **Boolean**.

snapshotLength

This read-only property is a **Number** and can raise an object that implements the **XPathException** interface on retrieval.

Functions of objects that implement the **XPathResult** interface:

iterateNext()

This function returns an object that implements the **Node** interface.

This function can raise an object that implements the **XPathException** interface or the **DOMException** interface.

snapshotItem(index)

This function returns an object that implements the **Node** interface.

The **index** parameter is a **Number**.

This function can raise an object that implements the **XPathException** interface.

Properties of the **XPathNamespace** Constructor function:

XPathNamespace.XPATH_NAMESPACE_NODE

The value of the constant **XPathNamespace.XPATH_NAMESPACE_NODE** is **13**.

Objects that implement the **XPathNamespace** interface:

Objects that implement the **XPathNamespace** interface have all properties and functions of the **Node** interface as well as the properties and functions defined below.

Properties of objects that implement the **XPathNamespace** interface:

ownerElement

This read-only property is an object that implements the **Element** interface.

Note: The parameter `resolver` of the method `XPathEvaluator.evaluate` [p.14] is specified as an object that implements the `XPathNSResolver` [p.17] interface. ECMAScript users can also pass to this method a function which returns a `String` and takes a `String` parameter instead of the `resolver` parameter.

Appendix D: Acknowledgements

Many people contributed to the DOM specifications (Level 1, 2 or 3), 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), Angel Diaz (IBM), Arnaud Le Hors (W3C and IBM), Ashok Malhotra (IBM and Microsoft), Ben Chang (Oracle), Bill Smith (Sun), Bill Shea (Merrill Lynch), Bob Sutor (IBM), Chris Lovett (Microsoft), Chris Wilson (Microsoft), David Brownell (Sun), David Ezell (Hewlett Packard Company), David Singer (IBM), Dimitris Dimitriadis (Improve AB), Don Park (invited), Elena Litani (IBM), Eric Vasilik (Microsoft), Gavin Nicol (INSO), Ian Jacobs (W3C), James Clark (invited), James Davidson (Sun), Jared Sorensen (Novell), Jeroen van Rotterdam (X-Hive Corporation), Joe Kesselman (IBM), Joe Lapp (webMethods), Joe Marini (Macromedia), Johnny Stenback (Netscape/AOL), Jon Ferraiolo (Adobe), 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), Mary Brady (NIST), 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), Rezaur Rahman (Intel), Rich Rollman (Microsoft), Rick Gessner (Netscape), Rick Jelliffe (invited), Rob Relyea (Microsoft), Scott Isaacs (Microsoft), Sharon Adler (INSO), Steve Byrne (JavaSoft), Tim Bray (invited), Tim Yu (Oracle), Tom Pixley (Netscape/AOL), Vidur Apparao (Netscape), Vinod Anupam (Lucent).

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 ECMAScript 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.

API
An *API* is an Application Programming Interface, a set of functions or methods used to access some functionality.

document element
There is only one document element in a `Document`. This element node is a child of the `Document` node. See *Well-Formed XML Documents* in XML [XML 1.0].

document order
There is an ordering, *document order*, defined on all the nodes in the document corresponding to the order in which the first character of the XML representation of each node occurs in the XML representation of the document after expansion of general entities. Thus, the *document element* [p.39] node will be the first node. Element nodes occur before their children. Thus, document order orders element nodes in order of the occurrence of their start-tag in the XML (after expansion of entities). The attribute nodes of an element occur after the element and before its children. The relative order of attribute nodes is implementation-dependent.

element
Each document contains one or more elements, the boundaries of which are either delimited by start-tags and end-tags, or, for empty elements by an empty-element tag. Each element has a type, identified by name, and may have a set of attributes. Each attribute has a name and a value. See *Logical Structures* in XML [XML 1.0].

logically-adjacent text nodes
Logically-adjacent text nodes are `Text` or `CDataSection` nodes that may be visited sequentially in *document order* [p.39] without entering, exiting, or passing over `Element`, `Comment`, or `ProcessingInstruction` nodes.

live
An object is *live* if any change to the underlying document structure is reflected in the object.

model
A *model* is the actual data representation for the information at hand. Examples are the structural model and the style model representing the parse structure and the style information associated with a document. The model might be a tree, or a directed graph, or something else.

namespace prefix
A *namespace prefix* is a string that associates an element or attribute name with a *namespace URI* in XML. See namespace prefix in Namespaces in XML [XML Namespaces].

namespace URI
A *namespace URI* is a URI that identifies an XML namespace. This is called the namespace name in Namespaces in XML [XML Namespaces].

read only node

A *read only node* is a node that is immutable. This means its list of children, its content, and its attributes, when it is an element, cannot be changed in any way. However, a read only node can possibly be moved, when it is not itself contained in a read only node.

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 2 Core]
Document Object Model Level 2 Core Specification, A. Le Hors, et al., Editors. World Wide Web Consortium, 13 November 2000. This version of the DOM Level 2 Core Recommendation is http://www.w3.org/TR/2000/REC-DOM-Level-2-Core-20001113. The latest version of DOM Level 2 Core is available at http://www.w3.org/TR/DOM-Level-2-Core.

[ECMAScript]
ECMAScript Language Specification, Third Edition. European Computer Manufacturers Association, December 1999. This version of the ECMAScript Language is available at http://www.ecma.ch/ecma1/STAND/ECMA-262.HTM.

[Java]
The Java Language Specification, J. Gosling, B. Joy, and G. Steele, Authors. Addison-Wesley, September 1996. Available at http://java.sun.com/docs/books/jls

[OMG IDL]
"OMG IDL Syntax and Semantics" defined in *The Common Object Request Broker: Architecture and Specification, version 2*, Object Management Group. The latest version of CORBA version 2.0 is available at http://www.omg.org/technology/documents/formal/corba_2.htm.

[XML Information set]
XML Information Set, J. Cowan and R. Tobin, Editors. World Wide Web Consortium, 24 October 2001. This version of the XML Information Set Recommendation is http://www.w3.org/TR/2001/REC-xml-infoset-20011024. The latest version of XML Information Set is available at http://www.w3.org/TR/xml-infoset.

[XPath 1.0]
XML Path Language (XPath) Version 1.0, J. Clark and S. DeRose, Editors. World Wide Web Consortium, 16 November 1999. This version of the XPath 1.0 Recommendation is http://www.w3.org/TR/1999/REC-xpath-19991116. The latest version of XPath 1.0 is available at http://www.w3.org/TR/xpath.

F.2: Informative references

[DOM Level 3 Core]
Document Object Model Level 3 Core Specification, A. Le Hors, et al., Editors. World Wide Web Consortium, January 2002. This version of the Document Object Model Level 3 Core Specification is http://www.w3.org/TR/2002/WD-DOM-Level-3-Core-20020114. The latest version of DOM Level 3 Core is available at http://www.w3.org/TR/DOM-Level-3-Core.

[XML 1.0]
Extensible Markup Language (XML) 1.0 (Second Edition), T. Bray, J. Paoli, C. M. Sperberg-McQueen, and E. Maler, Editors. World Wide Web Consortium, 10 February 1998, revised 6 October 2000. This version of the XML 1.0 Recommendation is

http://www.w3.org/TR/2000/REC-xml-20001006. The latest version of XML 1.0 is available at http://www.w3.org/TR/REC-xml.

[XML Namespaces]

Namespaces in XML, T. Bray, D. Hollander, and A. Layman, Editors. World Wide Web Consortium, 14 January 1999. This version of the XML Information Set Recommendation is http://www.w3.org/TR/1999/REC-xml-names-19990114. The latest version of Namespaces in XML is available at http://www.w3.org/TR/REC-xml-names.

Index