WD-DOM-Level-2-19990719

Document Object Model (DOM) Level 2 Specification

Version 1.0

W3C Working Draft *19 July, 1999*

This version:	http://www.w3.org/TR/1999/WD-DOM-Level-2-19990719 (PostScript file, PDF file, plain text, ZIP file)
Latest version:	http://www.w3.org/TR/WD-DOM-Level-2
Previous versions:	http://www.w3.org/TR/1999/WD-DOM-Level-2-19990304

WG Chair:	Lauren Wood, *SoftQuad Software Inc.*
Editors:	Vidur Apparao, *Netscape Communications Corporation* Mike Champion, *Arbortext and Software AG* Joe Kesselman, *IBM* Arnaud Le Hors, *W3C* Philippe Le Hégaret, *W3C* Tom Pixley, *Netscape Communications Corporation* Jonathan Robie, *Texcel Research and Software AG* Peter Sharpe, *SoftQuad Software Inc.* Chris Wilson, *Microsoft* Lauren Wood, *SoftQuad Software Inc.*

Status of this document

This document is an early release of the Document Object Model Level 2. It is guaranteed to change; anyone implementing it should realize that we will not allow ourselves to be restricted by experimental implementations of Level 2 when deciding whether to change the specifications.

This is a W3C 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.

This document has been produced as part of the W3C DOM Activity. The authors of this document are the DOM WG members. Different modules of the Document Object Model have different editors.

This document is for public review. Comments on this document should be sent to the public mailing list www-dom@w3.org.

Abstract

This specification defines the Document Object Model Level 2, a platform- and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure and style of documents. The Document Object Model Level 2 builds on the Document Object Model Level 1.

This release of the Document Object Model Level 2 has all of the interfaces that the final version is expected to have. It contains interfaces for creating a document, importing a node from one document to another, supporting XML namespaces, associating stylesheets with a document, the Cascading Style Sheets object model, the Range object model, filters and iterators, and the Events object model. The DOM WG wants to get feedback on these, and especially on the two options presented for XML namespaces, so that final decisions can be made for the DOM Level 2 specification.

Table of contents

Expanded Table of Contents

Copyright Notice

1. Document Object Model (Core) Level 2

Editors
Arnaud Le Hors, W3C

1.1. Overview of the DOM Level 2 Core Interfaces

This section defines a set of extensions to the interfaces defined in the Core section of the DOM Level 1 Recommendation to provide functionalities which are found to be essential but were not addressed in DOM Level 1.

These functionalitites include:

- Creating a `Document` object
- Copying a node from one document to another
- A way to get the element an attribute is attached to

(**ED:** Although new methods and attributes are introduced in this draft through the definition of a set of new interfaces, they are really meant to be added to the DOM Level 1 interfaces. The next version of this document will make this clear.)

1.2. The Core Interfaces

Interface *DocumentType2*

Two new attributes are added to the `DocumentType` interface to provide a way for retrieving the public and system identifiers

IDL Definition

```
interface DocumentType2 : DocumentType {
  readonly attribute DOMString        publicID;
  readonly attribute DOMString        systemID;
};
```

Attributes

`publicID`
The public identifier of the document type.

`systemID`
The system identifier of the document type.

Interface *DOMImplementation2*

The `DOMImplementation` interface is extended with methods for creating an XML document instance.

IDL Definition

```
interface DOMImplementation2 : DOMImplementation {
  DocumentType       createDocumentType(in DOMString name,
                                        in DOMString publicID,
                                        in DOMString systemID)
                                        raises(DOMException);
  Document           createDocument(in DOMString name,
                                    in DocumentType doctype)
                                        raises(DOMException);
};
```

Methods

`createDocumentType`

Creates an empty `DocumentType` node.

Parameters

`name`	The document type name.
`publicID`	The document type public identifier.
`systemID`	The document type system identifier.

Return Value

A new `DocumentType` node with `Node.ownerDocument` set to `null`.

Exceptions

`DOMException`

NOT_SUPPORTED_ERR: Raised if the requested document type is not supported.

`createDocument`

Creates an XML `Document` object of the specified type with its document element. (**ED:** Depending on how namespaces are supported this method may need one more parameter to hold the namespace name.)

Parameters

`name`	The name of document element to be created.
`doctype`	The type of document to be created or `null`. When `doctype` is not `null`, its `Node.ownerDocument` attribute is set to the document being created.

Return Value

A new `Document` object.

Exceptions

`DOMException`

WRONG_DOCUMENT_ERR: Raised if `doctype` has already been used with a different document.

NOT_SUPPORTED_ERR: Raised if the requested document type is not supported.

Interface *Document2*

The `Document` interface is extended with a method for importing nodes from another document.

IDL Definition

```
interface Document2 : Document {
  Node               importNode(in Node importedNode,
                                in boolean deep);
};
```

Methods

`importNode`

Imports a node from another document to this document. The returned node has no parent (`parentNode` is `null`.).

For all nodes, importing a node creates a node object owned by the importing document, with attribute values identical to the source node's `nodeName` and `nodeType`, plus the attributes related to namespaces (prefix and namespaces URI). As in the `Node.cloneNode()` operation, the source node is not altered.

Additional information is copied as appropriate to the `nodeType`, attempting to mirror the behavior expected if a fragment of XML or HTML source was copied from one document to another, recognizing that the two documents may have different DTDs in the XML case. The following list describes the specifics for every type of node.

ELEMENT_NODE

Specified attributes nodes of the source element are imported, and the generated `Attr` nodes are attached to the generated `Element`. Default attributes are *not* copied, though if the document being imported into defines default attributes for this element name, those are assigned. If importNode's "deep" option was set True, the descendents of the the source element will be recursively imported and the resulting nodes reassembled to form the corresponding subtree.

ATTRIBUTE_NODE

The `specified` flag is set `false` on the generated `Attr`. The descendents of the the source `Attr` are recursively imported and the resulting nodes reassembled to form the corresponding subtree.Note that the `deep` parameter does not apply to `Attr` nodes, they always carry their children with them when imported.

TEXT_NODE, CDATA_SECTION_NODE, COMMENT_NODE

These three types of nodes inheriting from `CharacterData` copy their `data` and `length` attributes from those of the source node.

ENTITY_REFERENCE_NODE

Only the `EntityReference` itself is copied, even if a `deep` import was requested, since the source and destination documents might have defined the entity differently. If the document being imported into provides a definition for this entity name, its value is assigned.

ENTITY_NODE

`Entity` nodes cannot be imported.

PROCESSING_INSTRUCTION_NODE

The imported node copies its `target` and `data` values from those of the source node.

DOCUMENT_NODE
`Document` nodes cannot be imported.
DOCUMENT_TYPE_NODE
`DocumentType` nodes cannot be imported.
DOCUMENT_FRAGMENT_NODE
If the `deep` option was set `true`, the descendents of the the source element will be recursively imported and the resulting nodes reassembled to form the corresponding subtree. Otherwise, this simply generates an empty `DocumentFragment`.
NOTATION_NODE
`Notation` nodes cannot be imported.
Parameters

`importedNode`	The node to import.
`deep`	If `true`, recursively import the subtree under the specified node; if `false`, import only the node itself, as explained above. This does not apply to `Attr` and `EntityReference` nodes.

Return Value
The imported node that belongs to this `Document`.
This method raises no exceptions.

Interface *Node2*

The `Node` interface is extended with an additional method to test if it supports a specific feature.
IDL Definition

```
interface Node2 : Node {
  boolean            supports(in DOMString feature,
                              in DOMString version);
};
```

Methods
`supports`
Tests whether the DOM implementation implements a specific feature and that feature is supported by this node.
Parameters

`feature`	The package name of the feature to test. This is the same name as what can be passed to the method `hasFeature` on `DOMImplementation`.
`version`	This is the version number of the package name to test. In Level 2, version 1, this is the string "2.0". If the version is not specified, supporting any version of the feature will cause the method to return `true`.

Return Value

Returns `true` if this node defines a subtree within which the specified feature is supported, `false` otherwise.

This method raises no exceptions.

Interface *Attr2*

The `Attr` interface provides an additional method for accessing the `Element` node the attribute is attached to.

IDL Definition

```
interface Attr2 : Attr {
  readonly attribute Element          ownerElement;
};
```

Attributes

`ownerElement`

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

1.3. The HTML Interfaces

(**ED:** This interface is not actually part of the DOM Core. It is part of HTML DOM Level 2. The next version of this document will make this clear.)

Interface *HTMLDOMImplementation*

The `HTMLDOMImplementation` interface extends the `DOMImplementation` interface with a method for creating an HTML document instance.

IDL Definition

```
interface HTMLDOMImplementation : DOMImplementation {
  HTMLDocument       createHTMLDocument(in DOMString title);
};
```

Methods

`createHTMLDocument`

Creates an `HTMLDocument` object with the minimal tree made of the following elements: `HTML`, `HEAD`, `TITLE`, and `BODY`.

Parameters

`title`	The title of the document to be set as the content of the `TITLE` element, through a child `Text` node.

Return Value

A new `HTMLDocument` object.

This method raises no exceptions.

1.4. Open Issues

1. Should import of an ENTITY_NODE be supported?
2. Should import of an DOCUMENT_NODE be supported?
3. Should import of an DOCUMENT_TYPE_NODE be supported?
4. Should import of an NOTATION_NODE be supported?
5. Should we add a flag to importNode to request for removal of the source node? This would potentially allow implementations to optimize the operation by doing an actual move underneath when possible.

1.4. Open Issues

2. Document Object Model Namespaces

Editors
Arnaud Le Hors, W3C

2.1. Introduction

This section defines two possible solutions to support XML namespaces. The first option consists in augmenting the interfaces defined in the Core section, leaving the semantics of DOM Level 1 as it is. The second option, on the contrary, consists in changing the semantics of DOM Level 1 and only augmenting existing interfaces where strictly necessary.

In any case, support for namespaces is mandatory.
(**ED:** Eventually only one of these two options will remain. But which one is still to be decided.)
(**ED:** This section defines a set of new interfaces but their methods and attributes are actually meant to be added to the corresponding DOM Level 1 interface. The next version of this specification will make that clear.)

2.2. The Namespaces related Interfaces Option #1

Interface *NodeNS*

The `Node` interface is extended to include a set of attributes to access the namespace prefix and namespace name of a node, and the local part of its qualified name (also called "local name" in this document).
IDL Definition

```
interface NodeNS {
  readonly attribute DOMString        namespaceName;
           attribute DOMString        prefix;
                                        // raises(DOMException) on setting

  readonly attribute DOMString        localName;
};
```

Attributes

`namespaceName`
: Returns the namespace name of this node or `null` if it is unspecified.

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

  For nodes created with a DOM Level 1 method, such as `Document.createElement`, this is `null`.

`prefix`
: The namespace prefix of this node or `null` if it is unspecified.

  For nodes created with a DOM Level 1 method, such as `Document.createElement`, this is `null`.

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

Exceptions on setting

`DOMException`

INVALID_CHARACTER_ERR: Raised if the specified prefix contains an invalid character.

`localName`

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

For nodes created with a DOM Level 1 method, such as `Document.createElement`, this is the same as `Node.nodeName`.

Interface *DocumentNS*

The `Document` interface provides two new methods for creating XML elements and attributes with a namespace prefix and namespace name.

IDL Definition

```
interface DocumentNS {
  Element           createElementNS(in DOMString namespaceName,
                                    in DOMString qualifiedName)
                                      raises(DOMException);
  Attr              createAttributeNS(in DOMString namespaceName,
                                      in DOMString qualifiedName)
                                      raises(DOMException);
  NodeList          getElementsByTagNameNS(in DOMString namespaceName,
                                           in DOMString localName);
};
```

Methods

`createElementNS`

Creates an element of the given qualified name and namespace name.

Parameters

`namespaceName`	The namespace name of the element to create.
`qualifiedName`	The qualified name of the element type to instantiate. This can contain a namespace prefix.

Return Value

A new `Element` object with the following attributes:

Attribute	Value
`Node.nodeName`	qualifiedName
`Node.namespaceName`	namespaceName
`Node.prefix`	prefix, extracted from qualifiedName, or null if there is no prefix
`Node.localName`	local part, extracted from qualifiedName
`Element.tagName`	qualifiedName

Exceptions

`DOMException`

INVALID_CHARACTER_ERR: Raised if the specified name contains an invalid character.

`createAttributeNS`

Creates an attribute of the given qualified name and namespace name.

Parameters

`namespaceName`	The namespace name of the attribute to create.
`qualifiedName`	The qualified name of the attribute to instantiate. This can contain a namespace prefix.

Return Value

A new `Attr` object with the following attributes:

Attribute	Value
`Node.nodeName`	qualifiedName
`Node.namespaceName`	namespaceName
`Node.prefix`	prefix, extracted from qualifiedName, or null if there is no prefix
`Node.localName`	local part, extracted from qualifiedName
`Attr.name`	qualifiedName

Exceptions

`DOMException`

INVALID_CHARACTER_ERR: Raised if the specified name contains an invalid character.

`getElementsByTagNameNS`

Returns a `NodeList` of all the `Element`s with a given local name and namespace name in the order in which they would be encountered in a preorder traversal of the `Document` tree.

Parameters

`namespaceName`	The namespace name of the elements to match on. The special value "*" matches all namespaces.
`localName`	The local name of the elements to match on. The special value "*" matches all local names.

Return Value

A new `NodeList` object containing all the matched `Element`s.

This method raises no exceptions.

Interface *ElementNS*

The `Element` interface is extended to provides a set of methods to manipulate attributes with namespaces. ***Note:*** *Both the* `tagName` *attribute from the* `Element` *interface and the* `nodeName` *attribute from the* `Node` *interface return the qualified name.*

IDL Definition

```
interface ElementNS {
  DOMString          getAttributeNS(in DOMString namespaceName,
                                    in DOMString localName);
  void               setAttributeNS(in DOMString namespaceName,
                                    in DOMString localName,
                                    in DOMString value)
                                        raises(DOMException);
  void               removeAttributeNS(in DOMString namespaceName,
                                       in DOMString localName)
                                        raises(DOMException);
  Attr               getAttributeNodeNS(in DOMString namespaceName,
                                        in DOMString localName);
  Attr               setAttributeNodeNS(in Attr newAttr)
                                        raises(DOMException);
  NodeList           getElementsByTagNameNS(in DOMString namespaceName,
                                            in DOMString localName);
};
```

Methods

`getAttributeNS`

Retrieves an attribute value by name and namespace name.

Parameters

`namespaceName`	The namespace name of the attribute to retrieve.
`localName`	The local name of the attribute to retrieve.

Return Value

The `Attr` value as a string, or an empty string if that attribute does not have a specified or default value.

This method raises no exceptions.

`setAttributeNS`

Adds a new attribute. If an attribute with that local name and namespace name is already present in the element, its value is changed to be that of the value parameter. This value is a simple string, it is not parsed as it is being set. So any markup (such as syntax to be recognized as an entity reference) is treated as literal text, and needs to be appropriately escaped by the implementation when it is written out. In order to assign an attribute value that contains entity references, the user must create an `Attr` node plus any `Text` and `EntityReference` nodes, build the appropriate subtree, and use `setAttributeNodeNS` or `setAttributeNode` to assign it as the value of an attribute.

Parameters

`namespaceName`	The namespace name of the attribute to create or alter.
`localName`	The local name of the attribute to create or alter.
`value`	The value to set in string form.

Exceptions

`DOMException`

INVALID_CHARACTER_ERR: Raised if the specified name contains an invalid character.

NO_MODIFICATION_ALLOWED_ERR: Raised if this node is readonly.

This method returns nothing.

`removeAttributeNS`

Removes an attribute by local name and namespace name. If the removed attribute has a default value it is immediately replaced.

Parameters

`namespaceName`	The namespace name of the attribute to remove.
`localName`	The local name of the attribute to remove.

Exceptions

`DOMException`

NO_MODIFICATION_ALLOWED_ERR: Raised if this node is readonly.

This method returns nothing.

`getAttributeNodeNS`

Retrieves an `Attr` node by name and namespace name.

Parameters

`namespaceName`	The namespace name of the attribute to retreive.
`localName`	The local name of the attribute to retrieve.

Return Value

The `Attr` node with the specified attribute local name and namespace name or `null` if there is no such attribute.

This method raises no exceptions.

`setAttributeNodeNS`

Adds a new attribute. If an attribute with that local name and namespace name is already present in the element, it is replaced by the new one.

Parameters

`newAttr`	The `Attr` node to add to the attribute list.

Return Value

If the `newAttr` attribute replaces an existing attribute with the same local name and namespace name, the previously existing `Attr` node is returned, otherwise `null` is returned.

Exceptions

`DOMException`

WRONG_DOCUMENT_ERR: Raised if `newAttr` was created from a different document than the one that created the element.

NO_MODIFICATION_ALLOWED_ERR: Raised if this node is readonly.

INUSE_ATTRIBUTE_ERR: Raised if `newAttr` is already an attribute of another `ElementNS` object. The DOM user must explicitly clone `Attr` nodes to re-use them in other elements.

`getElementsByTagNameNS`

Returns a `NodeList` of all the `Element`s with a given local name and namespace name in the order in which they would be encountered in a preorder traversal of the `Document` tree, starting from this node.

Parameters

`namespaceName`	The namespace name of the elements to match on. The special value "*" matches all namespaces.
`localName`	The local name of the elements to match on. The special value "*" matches all local names.

Return Value

A new `NodeList` object containing all the matched `Element`s.

This method raises no exceptions.

2.3. Further Considerations about Namespaces and Option #1

Special attributes used for declaring XML namespaces are exposed through the DOM and can be manipulated just like any other attribute. Moving a node within a document, using the DOM, in no case results in a change of its namespace prefix or namespace name. Similarly, creating a node with a namespace prefix and namespace name, or changing the namespace prefix of a node, does not result in any addition, removal, or modification of any special attributes for declaring the appropriate XML namespaces. Applications are therefore responsible for declaring every namespace in use when saving a document into XML.

Elements and attributes can still be created using the `createElement` and `createAttribute` methods from the `Document` interface. However, they do not have any namespace prefix or namespace name then.

This option garantees full backwards compatibility with DOM Level 1, however, it introduces a whole set of new interfaces and obsoletes a large swath of the Level 1 API which simply cannot be used by a namespace aware application.

2.4. The Namespaces Support Option #2

The solution described in this section is based on the use of "universal names". Universal names are made of the namespace name and the local name. Although there isn't currently any standard syntax for such names the following has been proposed: *{namespaceName}localName*. Assuming such names exist, supporting Namespaces can then simply be achieved by changing the DOM Level 1 semantics so that wherever an element or attribute name is taken in argument, if it is a universal name, namespace special handling is thrown into gear.

Interface *NodeNS*

The `Node` interface is extended to include a set of attributes to access the namespace prefix and namespace name of a node, and the local part of its qualified name (also called "local name" in this document).
(**ED:** This is the same as in Option #1 with the additional `universalName` attribute.)
IDL Definition

```
interface NodeNS {
  readonly attribute DOMString        universalName;
  readonly attribute DOMString        namespaceName;
           attribute DOMString        prefix;
                                        // raises(DOMException) on setting

  readonly attribute DOMString        localName;
};
```

Attributes

`universalName`

Returns the universal name of this node.

`namespaceName`

Returns the namespace name of this node or `null` if it is unspecified.

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

`prefix`

The namespace prefix of this node or `null` if it is unspecified.

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

Exceptions on setting

`DOMException`

INVALID_CHARACTER_ERR: Raised if the specified prefix contains an invalid character.

`localName`

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

Definition group *Document changes*

The following methods of the `Document` interface are changed.

Methods

`createElement`

Creates an element of the type specified.

Parameters

`universalName`	The universal name of the element type to instantiate. For XML, this is case-sensitive. For HTML, the `universalName` parameter is simply the `tagName` and it may be provided in any case, but it must be mapped to the canonical uppercase form by the DOM implementation. This does not set the prefix which may be defined later through the `Node.prefix` attribute.

Return Value

A new `Element` object with the following attributes:

Attribute	Value
`Node.nodeName`	qualified name, initialized with the local part extracted from universalName
`Node.namespaceName`	namespaceName, extracted from universalName
`Node.prefix`	null
`Node.localName`	local part, extracted from universalName
`Element.tagName`	qualified name, initialized with the local part extracted from universalName

Exceptions

`DOMException`

INVALID_CHARACTER_ERR: Raised if the specified name contains an invalid character.

`createAttribute`

Creates an `Attr` of the given name. The returned object implements the `Attr` interface as well as the `Node` interface. It can then be set on an `Element` using the `setAttributeNode` method. This does not set the prefix which may be defined later through the `Node.prefix` attribute.

Parameters

`universalName`	The universal name of the attribute. For HTML, this is simply the attribute name.

Return Value

A new `Attr` object with the following attributes:

Attribute	Value
`Node.nodeName`	qualified name, initialized with the local part extracted from universalName
`Node.namespaceName`	namespaceName, extracted from universalName
`Node.prefix`	null
`Node.localName`	local part, extracted from universalName
`Attr.name`	qualified name, initialized with the local part extracted from universalName

Exceptions

`DOMException`

INVALID_CHARACTER_ERR: Raised if the specified name contains an invalid character.

`getElementsByTagName`

Returns a `NodeList` of all the `Element`s with a given universal name in the order in which they would be encountered in a preorder traversal of the `Document` tree.

Parameters

`universalName`	The universal name of the elements to match on. The special value "*" can be used to match all namespaces and/or local names.

Return Value

A new `NodeList` object containing all the matched `Element`s.

This method raises no exceptions.

Definition group *Element changes*

The following methods of the `Element` interface are changed.

Methods

`getAttribute`

Retrieves an attribute value by universal name.

Parameters

`universalName`	The universal name of the attribute to retrieve.

Return Value

The `Attr` value as a string, or the empty string if that attribute does not have a specified or default value.

This method raises no exceptions.

`setAttribute`

Adds a new attribute. If an attribute with that universal name is already present in the element, its value is changed to be that of the value parameter. This value is a simple string, it is not parsed as it is being set. So any markup (such as syntax to be recognized as an entity reference) is treated as literal text, and needs to be appropriately escaped by the implementation when it is written out. In order to assign an attribute value that contains entity references, the user must create an `Attr` node plus any `Text` and `EntityReference` nodes, build the appropriate subtree, and use `setAttributeNode` to assign it as the value of an attribute.

Parameters

`universalName`	The universal name of the attribute to create or alter.
`value`	Value to set in string form.

Exceptions

`DOMException`

INVALID_CHARACTER_ERR: Raised if the specified name contains an invalid character.

NO_MODIFICATION_ALLOWED_ERR: Raised if this node is readonly.

This method returns nothing.

`removeAttribute`

Removes an attribute by universal name. If the removed attribute has a default value it is immediately replaced.

Parameters

`universalName`	The universal name of the attribute to remove.

Exceptions

`DOMException`

NO_MODIFICATION_ALLOWED_ERR: Raised if this node is readonly.

This method returns nothing.

`getAttributeNode`

Retrieves an `Attr` node by universal name.

Parameters

`universalName`	The universal name of the attribute to retrieve.

Return Value

The `Attr` node with the specified attribute universal name or `null` if there is no such attribute.

This method raises no exceptions.

`setAttributeNode`

Adds a new attribute. If an attribute with that universal name is already present in the element, it is replaced by the new one.

Parameters

`newAttr`	The `Attr` node to add to the attribute list.

Return Value

If the `newAttr` attribute replaces an existing attribute with the same universal name, the previously existing `Attr` node is returned, otherwise `null` is returned.

Exceptions

`DOMException`

WRONG_DOCUMENT_ERR: Raised if `newAttr` was created from a different document than the one that created the element.

NO_MODIFICATION_ALLOWED_ERR: Raised if this node is readonly.

INUSE_ATTRIBUTE_ERR: Raised if `newAttr` is already an attribute of another `Element` object. The DOM user must explicitly clone `Attr` nodes to re-use them in other elements.

`getElementsByTagName`

Returns a `NodeList` of all the `Element`s with a given universal name in the order in which they would be encountered in a preorder traversal of the `Document` tree, starting from this node.

Parameters

`universalName`	The universal name of the elements to match on. The special value "*" can be used to match all namespaces and/or local names.

Return Value

A new `NodeList` object containing all the matched `Element`s.

This method raises no exceptions.

2.5. Further Considerations about Option #2

The model is the same as in Option #1, the difference only lies in the way we define access to the namespace information. The Option #2 has the obvious advantage of requiring only one new interface and very little change to make an application namespace aware. However, this is done at the cost of introducing some backwards incompatibility. Namely what is considered to be an error in DOM Level 1, now simply throw into gear some special handling of namespaces. In particular, while a DOM Level 1 implementation raises an INVALID_CHARACTER_ERR `DOMException` when `"{myuri}foo"` is passed to `Document.createElement`, a DOM Level 2 implementation would not. A DOM Level 1 application relying on this exception to be raised would therefore fail on a DOM Level 2 implementation.

Here is a specific scenario where changing the behavior of the Level 1 DOM would adversely impact an application.

Consider an editor application authored using the Level 1 DOM that allows a user to construct XML documents. One function of the editor allows the user to create elements in the DOM tree. The user enters the tag name through a UI that accepts the name of the tag, then calls `Document.createElement` to create an element node and then inserts the node into the tree using additional DOM methods (specifics are not req'd here). The editor allows the user to save the document to disk through a menu item in the editor. The save function is implemented using the Level 1 DOM. Basically, the save function walks the tree writing out the XML. For each element, it uses either the `Node.nodeName` or `Element.tagName` property to get the name of the element to output.

Now, consider the same editor running under a Level 2 DOM implementation (note that the editor has not been re-authored for Level 2 DOM). The user agains begin creating elements in the DOM tree, however the user enters a valid universal name into the editor which in turn calls `Document.createElement()`. Since `createElement()` now accepts a universal name in Level 2, this succeeds whereas it would have failed in Level 1. The problems arises when the user tells the editor to save the document. As the save function walks the tree to output the XML, data loss occurs because in Level 1 there was no concept of namespaces. Therefore, `Node.nodeName` or `Element.tagName` return only the localName without the namespace.

A code example for the scenario is:

```
function buildElement(tagName,parent)
{
  parent.appendChild(document.createElement(tagName));
}

function saveTree(root)
{
  switch (root.nodeType)
  {
     case Element:
        print("<" + root.nodeName + ">");
        for (i = 0 ; i < root.childNodes.length ; i++)
          saveTree(root.childNodes.item(i));
        print("</" + root.nodeName + ">");
        break;
        // add more processing for other node types
      ...
  }
}
```

The call sequence for Level 1 DOM would be:

```
  // foo entered by user
  buildElement("foo",document.root);
  saveTree(document.root);
```

The result would be:

```
<root>
   <foo></foo>
</root>
```

The call sequence for Level 2 DOM would be:

```
  // universal name entered by user
  buildElement("{http://somedomain/foonamespace}foo",document.root);
  saveTree(document.root);
```

The result would be:

```
<root>
  <foo></foo>
</root>
```

which is not the desired result.

2.6. Open Issues

1. Which option do we choose?!!
2. Is the name "localName" ok? The namespaces spec uses "localPart" but it doesn’t seem descriptive enough. We could make it "localPartName". We need to sync with XSL.
3. `getElementsByTagname` is a misnommer when used with namespaces, should we use another name?!! If yes, which one? `getElementsByName`?

2.6. Open Issues

3. Document Object Model StyleSheets

Editors
Vidur Apparao, Netscape Communications Corp.
Philippe Le Hégaret, W3C
Chris Wilson, Microsoft

3.1. Introduction

The DOM Level 2 Style Sheet interfaces are base interfaces used to represent any type of style sheet. The expectation is that DOM modules that represent a specific style sheet language may contain interfaces that derive from these interfaces.

3.2. Style Sheet Interfaces

This set of interfaces represents the generic notion of style sheets.

Interface *StyleSheet*

The `StyleSheet` interface is the abstract base interface for any type of style sheet. It represents a single style sheet associated with a structured document. In HTML, the StyleSheet interface represents either an external style sheet, included via the HTML LINK element, or an inline STYLE element. In XML, this interface represents an external style sheet, included via a style sheet processing instruction .

IDL Definition

```
interface StyleSheet {
  readonly attribute DOMString        type;
           attribute boolean          disabled;
  readonly attribute Node             ownerNode;
  readonly attribute StyleSheet       parentStyleSheet;
  readonly attribute DOMString        href;
  readonly attribute DOMString        title;
  readonly attribute MediaList        media;
};
```

Attributes

`type`
: This specifies the style sheet language for this style sheet. The style sheet language is specified as a content type (e.g. "text/css"). The content type is often specified in the `ownerNode`. A list of registered content types can be found at ftp://ftp.isi.edu/in-notes/iana/assignments/media-types/. Also see the type attribute definition for the `LINK` element in HTML 4.0, and the type pseudo-attribute for the XML style sheet processing instruction .

`disabled`
: `false` if the style sheet is applied to the document. `true` if it is not. Modifying this attribute may cause a reresolution of style for the document.

`ownerNode`
: The node that associates this style sheet with the document. For HTML, this may be the corresponding `LINK` or `STYLE` element. For XML, it may be the linking processing instruction. For style sheets that are included by other style sheets, this attribute has a value of null.

`parentStyleSheet`
: For style sheet languages that support the concept of style sheet inclusion, this attribute represents the including style sheet, if one exists. If the style sheet is a top-level style sheet,

or the style sheet language does not support inclusion, the value of the attribute is null.

`href`
: If the style sheet is a linked style sheet, the value of its attribute is its location. For inline style sheets, the value of this attribute is null. See the href attribute definition for the `LINK` element in HTML 4.0, and the href pseudo-attribute for the XML style sheet processing instruction .

`title`
: The advisory title. The title is often specified in the `ownerNode`. See the title attribute definition for the `LINK` element in HTML 4.0, and the title pseudo-attribute for the XML style sheet processing instruction .

`media`
: The intended destination media for style information. The media is often specified in the `ownerNode`. See the media attribute definition for the `LINK` element in HTML 4.0, and the media pseudo-attribute for the XML style sheet processing instruction .

Interface *StyleSheetList*

The `StyleSheetList` interface provides the abstraction of an ordered collection of style sheets.

IDL Definition

```
interface StyleSheetList {
  readonly attribute unsigned long    length;
  StyleSheet         item(in unsigned long index);
};
```

Attributes

`length`
: The number of `StyleSheet` [p.34] in the list. The range of valid child stylesheet indices is `0` to `length-1` inclusive.

Methods

`item`
: Used to retrieve a style sheet by ordinal index.

Parameters

`index`	Index into the collection

Return Value

The style sheet at the `index` position in the `StyleSheetList`, or `null` if that is not a valid index.

This method raises no exceptions.

Interface *MediaList*

The `MediaList` interface provides the abstraction of an ordered collection of media, without defining or constraining how this collection is implemented. All media are lowercase strings.

IDL Definition

```
interface MediaList {
           attribute DOMString        cssText;
                                        // raises(DOMException) on setting

  readonly attribute unsigned long    length;
  DOMString          item(in unsigned long index);
  void               delete(in DOMString oldMedium)
                                        raises(DOMException);
  void               append(in DOMString newMedium)
                                        raises(DOMException);
};
```

Attributes

`cssText`

The parsable textual representation of the media list. This is a comma-separated list of media.

Exceptions on setting

`DOMException`

SYNTAX_ERR: Raised if the specified CSS string value has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this media list is readonly.

`length`

The number of media in the list. The range of valid media is `0` to `length-1` inclusive.

Methods

`item`

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

Parameters

`index`	Index into the collection.

Return Value

The medium at the `index`th position in the `MediaList`, or `null` if that is not a valid index.

This method raises no exceptions.

`delete`

Deletes the medium indicated by `oldMedium` from the list.

Parameters

`oldMedium`	The medium to delete in the media list.

Exceptions

`DOMException`

NO_MODIFICATION_ALLOWED_ERR: Raised if this list is readonly.

NOT_FOUND_ERR: Raised if `oldMedium` is not in the list.

This method returns nothing.

`append`

Adds the medium `newMedium` to the end of the list. It the `newMedium` is already used, it is first removed.

Parameters

`newMedium`	The new medium to add.

Exceptions

`DOMException`

NO_MODIFICATION_ALLOWED_ERR: Raised if this list is readonly.

This method returns nothing.

3.3. Document Extensions

Interface *DocumentStyle*

The `DocumentStyle` interface provides a mechanism by which the style sheets embedded a document can be retrieved. The expectation is that an instance of the `DocumentStyle` interface can be obtained by using binding-specific casting methods on an instance of the Level 1 Document interface.

IDL Definition

```
interface DocumentStyle {
  readonly attribute StyleSheetList   styleSheets;
};
```

Attributes

`styleSheets`

A list containing all the style sheets explicitly linked into or embedded in a document. For HTML documents, this includes external style sheets, included via the HTML LINK element, and inline STYLE elements. In XML, this includes external style sheets, included via style sheet processing instructions.

4. Document Object Model CSS

Editors
Vidur Apparao, Netscape Communications Corp.
Philippe Le Hégaret, W3C
Chris Wilson, Microsoft

4.1. Overview of the DOM Level 2 CSS Interfaces

The DOM Level 2 Cascading Style Sheets (CSS) interfaces are designed with the goal of exposing CSS constructs to object model consumers. Cascading Style Sheets is a declarative syntax for defining presentation rules, properties and ancillary constructs used to format and render Web documents. This document specifies a mechanism to programmatically access and modify the rich style and presentation control provided by CSS (specifically CSS level two). This augments CSS by providing a mechanism to dynamically control the inclusion and exclusion of individual style sheets, as well as manipulate CSS rules and properties.

The CSS interfaces are organized in a logical, rather than physical structure. A collection of all style sheets referenced by or embedded in the document is accessible on the document interface. Each item in this collection exposes the properties common to all style sheets referenced or embedded in HTML and XML documents; this interface is described in the Style Sheets chapter of the DOM Level 2. User style sheets are not accessible through this collection, in part due to potential privacy concerns (and certainly read-write issues).

For each CSS style sheet, an additional interface is exposed - the CSSStyleSheet interface. This interface allows access to the collection of rules within a CSS style sheet and methods to modify that collection. Interfaces are provided for each specific type of rule in CSS2 (e.g. style declarations, @import rules, or @font-face rules), as well as a shared generic CSSRule interface.

The most common type of rule is a style declaration. The CSSStyleRule interface that represents this type of rule provides string access to the CSS selector of the rule, and access to the property declarations through the CSSStyleDeclaration interface.

Finally, an optional CSS2Properties interface is described; this interface (if implemented) provides shortcuts to the string values of all the properties in CSS level 2.

4.2. CSS Fundamental Interfaces

The interfaces within this section are considered fundamental, and must be implemented by all conforming applications of this specifcation. These interfaces represent CSS style sheets specifically.

A DOM consumer can use the hasFeature of the DOMImplementation interface to determine whether the CSS module has been implemented by a DOM implementation. The feature string for the fundamental interfaces listed in this section is "CSS".

Interface *CSSStyleSheet*

The `CSSStyleSheet` interface is a concrete interface used to represent a CSS style sheet i.e. a style sheet whose content type is "text/css".
IDL Definition

```
interface CSSStyleSheet : StyleSheet {
  readonly attribute CSSRule          ownerRule;
  readonly attribute CSSRuleList      cssRules;
  unsigned long      insertRule(in DOMString rule,
                                in unsigned long index)
                                        raises(DOMException);
  void               deleteRule(in unsigned long index)
                                        raises(DOMException);
};
```

Attributes

`ownerRule`

If this style sheet comes from an `@import` rule, the `ownerRule` attribute will contain the `CSSImportRule` [p.47] . In that case, the `ownerNode` attribute in the `StyleSheet` [p.34] interface will be `null`. If the style sheet comes from an element or a processing instruction, the `ownerRule` attribute will be `null` and the `ownerNode` attribute will contain the `Node`.

`cssRules`

The list of all CSS rules contained within the style sheet. This includes both rule sets and at-rules.

Methods

`insertRule`

Used to insert a new rule into the style sheet. The new rule now becomes part of the cascade.

Parameters

`rule`	The parsable text representing the rule. For rule sets this contains both the selector and the style declaration. For at-rules, this specifies both the at-identifier and the rule content.
`index`	The index within the style sheet's rule list of the rule before which to insert the specified rule. If the specified index is equal to the length of the style sheet's rule collection, the rule will be added to the end of the style sheet.

Return Value

The index within the style sheet's rule collection of the newly inserted rule.

Exceptions

`DOMException`

HIERARCHY_REQUEST_ERR: Raised if the rule cannot be inserted at the specified index e.g. if an `@import` rule is inserted after a standard rule set or other at-rule.

INDEX_SIZE_ERR: Raised if the specified index is not a valid insertion point.

SYNTAX_ERR: Raised if the specified rule has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this style sheet is readonly.

`deleteRule`

Used to delete a rule from the style sheet.

Parameters

`index`	The index within the style sheet's rule list of the rule to remove.

Exceptions

`DOMException`

INDEX_SIZE_ERR: Raised if the specified index does not correspond to a rule in the style sheet's rule list.

NO_MODIFICATION_ALLOWED_ERR: Raised if this style sheet is readonly.

This method returns nothing.

Interface *CSSRuleList*

The `CSSRuleList` interface provides the abstraction of an ordered collection of CSS rules.

IDL Definition

```
interface CSSRuleList {
  readonly attribute unsigned long    length;
  CSSRule            item(in unsigned long index);
};
```

Attributes

`length`

The number of `CSSRule` [p.43] s in the list. The range of valid child rule indices is `0` to `length-1` inclusive.

Methods

`item`

Used to retrieve a CSS rule by ordinal index. The order in this collection represents the order of the rules in the CSS style sheet.

Parameters

`index`	Index into the collection

Return Value

The style rule at the `index` position in the `CSSRuleList`, or `null` if that is not a valid index.

This method raises no exceptions.

Interface *CSSRule*

The `CSSRule` interface is the abstract base interface for any type of CSS statement. This includes both rule sets and at-rules. An implementation is expected to preserve all rules specified in a CSS style sheet, even if it is not recognized. Unrecognized rules are represented using the `CSSUnknownRule` [p.48] interface.

IDL Definition

```
interface CSSRule {
  // RuleType
  const unsigned short      UNKNOWN_RULE                   = 0;
  const unsigned short      STYLE_RULE                     = 1;
  const unsigned short      CHARSET_RULE                   = 2;
  const unsigned short      IMPORT_RULE                    = 3;
  const unsigned short      MEDIA_RULE                     = 4;
  const unsigned short      FONT_FACE_RULE                 = 5;
  const unsigned short      PAGE_RULE                      = 6;

  readonly attribute unsigned short   type;
           attribute DOMString        cssText;
                                        // raises(DOMException) on setting

  readonly attribute CSSStyleSheet    parentStyleSheet;
  readonly attribute CSSRule          parentRule;
};
```

Definition group *RuleType*

An integer indicating which type of rule this is.

Defined Constants

UNKNOWN_RULE	The rule is a `CSSUnknownRule` [p.48] .
STYLE_RULE	The rule is a `CSSStyleRule` [p.44] .
CHARSET_RULE	The rule is a `CSSCharsetRule` [p.47] .
IMPORT_RULE	The rule is a `CSSImportRule` [p.47] .
MEDIA_RULE	The rule is a `CSSMediaRule` [p.44] .
FONT_FACE_RULE	The rule is a `CSSFontFaceRule` [p.46] .
PAGE_RULE	The rule is a `CSSPageRule` [p.46] .

Attributes

`type`

The type of the rule, as defined above. The expectation is that binding-specific casting methods can be used to cast down from an instance of the `CSSRule` interface to the specific derived interface implied by the `type`.

`cssText`
The parsable textual representation of the rule. This reflects the current state of the rule and not its initial value.
Exceptions on setting
`DOMException`

SYNTAX_ERR: Raised if the specified CSS string value has a syntax error and is unparsable.

HIERARCHY_REQUEST_ERR: Raised if the rule cannot be inserted at this point in the style sheet.

NO_MODIFICATION_ALLOWED_ERR: Raised if this style sheet is readonly.

`parentStyleSheet`
The style sheet that contains this rule.

`parentRule`
If this rule is contained inside another rule (e.g. a style rule inside an @media block), this is the containing rule. If this rule is not nested inside any other rules, this returns `null`.

Interface *CSSStyleRule*

The `CSSStyleRule` interface represents a single rule set in a CSS style sheet.
IDL Definition

```
interface CSSStyleRule : CSSRule {
           attribute DOMString        selectorText;
                                        // raises(DOMException) on setting

  readonly attribute CSSStyleDeclaration  style;
};
```

Attributes

`selectorText`
The textual representation of the selector for the rule set. The implementation may have stripped out insignificant whitespace while parsing the selector.
Exceptions on setting
`DOMException`

SYNTAX_ERR: Raised if the specified CSS string value has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this style sheet is readonly.

`style`
The declaration-block of this rule set.

Interface *CSSMediaRule*

The `CSSMediaRule` interface represents a @media rule in a CSS style sheet. A `@media` rule can be used to delimit style rules for specific media types.

IDL Definition

```
interface CSSMediaRule : CSSRule {
  readonly attribute MediaList        media;
  readonly attribute CSSRuleList      cssRules;
  unsigned long      insertRule(in DOMString rule,
                                in unsigned long index)
                                        raises(DOMException);
  void               deleteRule(in unsigned long index)
                                        raises(DOMException);
};
```

Attributes

`media`

A list of media types for this rule.

`cssRules`

A list of all CSS rules contained within the media block.

Methods

`insertRule`

Used to insert a new rule into the media block.

Parameters

`rule`	The parsable text representing the rule. For rule sets this contains both the selector and the style declaration. For at-rules, this specifies both the at-identifier and the rule content.
`index`	The index within the media block's rule collection of the rule before which to insert the specified rule. If the specified index is equal to the length of the media blocks's rule collection, the rule will be added to the end of the media block.

Return Value

The index within the media block's rule collection of the newly inserted rule.

Exceptions

`DOMException`

HIERARCHY_REQUEST_ERR: Raised if the rule cannot be inserted at the specified index. e.g. if an `@import` rule is inserted after a standard rule set or other at-rule.

INDEX_SIZE_ERR: Raised if the specified index is not a valid insertion point.

SYNTAX_ERR: Raised if the specified rule has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this media rule is readonly.

`deleteRule`

Used to delete a rule from the media block.

Parameters

`index`	The index within the media block's rule collection of the rule to remove.

Exceptions

`DOMException`

INDEX_SIZE_ERR: Raised if the specified index does not correspond to a rule in the media rule list.

NO_MODIFICATION_ALLOWED_ERR: Raised if this media rule is readonly.

This method returns nothing.

Interface *CSSFontFaceRule*

The `CSSFontFaceRule` interface represents a @font-face rule in a CSS style sheet. The `@font-face` rule is used to hold a set of font descriptions.

IDL Definition

```
interface CSSFontFaceRule : CSSRule {
  readonly attribute CSSStyleDeclaration  style;
};
```

Attributes

`style`

The declaration-block of this rule.

Interface *CSSPageRule*

The `CSSPageRule` interface represents a @page rule within a CSS style sheet. The `@page` rule is used to specify the dimensions, orientation, margins, etc. of a page box for paged media.

IDL Definition

```
interface CSSPageRule : CSSRule {
           attribute DOMString        selectorText;
                                        // raises(DOMException) on setting

  readonly attribute CSSStyleDeclaration  style;
};
```

Attributes

`selectorText`

The parsable textual representation of the page selector for the rule.

Exceptions on setting

`DOMException`

SYNTAX_ERR: Raised if the specified CSS string value has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this style sheet is readonly.

`style`
: The declaration-block of this rule.

Interface *CSSImportRule*

The `CSSImportRule` interface represents a @import rule within a CSS style sheet. The `@import` rule is used to import style rules from other style sheets.

IDL Definition

```
interface CSSImportRule : CSSRule {
  readonly attribute DOMString        href;
  readonly attribute MediaList        media;
  readonly attribute CSSStyleSheet    styleSheet;
};
```

Attributes

`href`
: The location of the style sheet to be imported. The attribute will not contain the `"url(...)"` specifier around the URI.

`media`
: A list of media types for which this style sheet may be used.

`styleSheet`
: The style sheet referred to by this rule, if it has been loaded. The value of this attribute is null if the style sheet has not yet been loaded or if it will not be loaded (e.g. if the style sheet is for a media type not supported by the user agent).

Interface *CSSCharsetRule*

The `CSSCharsetRule` interface a @charset rule in a CSS style sheet. A `@charset` rule can be used to define the encoding of the style sheet.

IDL Definition

```
interface CSSCharsetRule : CSSRule {
           attribute DOMString        encoding;
                                        // raises(DOMException) on setting

};
```

Attributes

`encoding`
: The encoding information used in this `@charset` rule.

Exceptions on setting

`DOMException`

SYNTAX_ERR: Raised if the specified encoding value has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this encoding rule is readonly.

Interface *CSSUnknownRule*

The `CSSUnkownRule` interface represents an at-rule not supported by this user agent.

IDL Definition

```
interface CSSUnknownRule : CSSRule {
};
```

Interface *CSSStyleDeclaration*

The `CSSStyleDeclaration` interface represents a single CSS declaration block. This interface may be used to determine the style properties currently set in a block or to set style properties explicitly within the block.

While an implementation may not recognize all CSS properties within a CSS declaration block, it is expected to provide access to all specified properties through the `CSSStyleDeclaration` interface. Furthermore, implementations that support a specific level of CSS should correctly handle CSS shorthand properties for that level. For a further discussion of shorthand properties, see the `CSS2Properties` [p.79] interface.

IDL Definition

```
interface CSSStyleDeclaration {
           attribute DOMString        cssText;
                                        // raises(DOMException) on setting

  DOMString          getPropertyValue(in DOMString propertyName);
  CSSValue           getPropertyCSSValue(in DOMString propertyName);
  DOMString          removeProperty(in DOMString propertyName)
                                        raises(DOMException);
  DOMString          getPropertyPriority(in DOMString propertyName);
  void               setProperty(in DOMString propertyName,
                                 in DOMString value,
                                 in DOMString priority)
                                        raises(DOMException);
  readonly attribute unsigned long    length;
  DOMString          item(in unsigned long index);
  readonly attribute CSSRule          parentRule;
};
```

Attributes

`cssText`

The parsable textual representation of the declaration block (including the surrounding curly braces). Setting this attribute will result in the parsing of the new value and resetting of the properties in the declaration block.

Exceptions on setting

`DOMException`

SYNTAX_ERR: Raised if the specified CSS string value has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this declaration is readonly.

`length`

The number of properties that have been explicitly set in this declaration block.

`parentRule`

The CSS rule that contains this declaration block.

Methods

`getPropertyValue`

Used to retrieve the value of a CSS property if it has been explicitly set within this declaration block.

Parameters

`propertyName`	The name of the CSS property. See the CSS property index.

Return Value

Returns the value of the property if it has been explicitly set for this declaration block. Returns the empty string if the property has not been set.

This method raises no exceptions.

`getPropertyCSSValue`

Used to retrieve the object representation of the value of a CSS property if it has been explicitly set within this declaration block. This method returns null if the property is a shorthand property. Shorthand property values can only be accessed and modified as strings, using the `getPropertyValue` and `setProperty` methods.

Parameters

`propertyName`	The name of the CSS property. See the CSS property index.

Return Value

Returns the value of the property if it has been explicitly set for this declaration block. Returns the `null` if the property has not been set.

This method raises no exceptions.

`removeProperty`

Used to remove a CSS property if it has been explicitly set within this declaration block.

Parameters

`propertyName`	The name of the CSS property. See the CSS property index.

Return Value

Returns the value of the property if it has been explicitly set for this declaration block. Returns the empty string if the property has not been set or the property name does not correspond to a valid CSS2 property.

Exceptions

`DOMException`

NO_MODIFICATION_ALLOWED_ERR: Raised if this declaration is readonly.

`getPropertyPriority`

Used to retrieve the priority of a CSS property (e.g. the `"important"` qualifier) if the property has been explicitly set in this declaration block.

Parameters

`propertyName`	The name of the CSS property. See the CSS property index.

Return Value

A string representing the priority (e.g. `"important"`) if one exists. The empty string if none exists.

This method raises no exceptions.

`setProperty`

Used to set a property value and priority within this declaration block.

Parameters

`propertyName`	The name of the CSS property. See the CSS property index.
`value`	The new value of the property.
`priority`	The new priority of the property (e.g. `"important"`).

Exceptions

`DOMException`

SYNTAX_ERR: Raised if the specified value has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this declaration is readonly.

This method returns nothing.

`item`

Used to retrieve the properties that have been explicitly set in this declaration block. The order of the properties retrieved using this method does not have to be the order in which they were set. This method can be used to iterate over all properties in this declaration block.

Parameters

`index`	Index of the property name to retrieve.

Return Value

The name of the property at this ordinal position. The empty string if no property exists at this position.

This method raises no exceptions.

Interface *CSSValue*

The `CSSValue` interface represents a simple or a complexe value.

IDL Definition

```
interface CSSValue {
  // UnitTypes
  const unsigned short      CSS_PRIMITIVE_VALUE            = 0;
  const unsigned short      CSS_VALUE_LIST                 = 1;
  const unsigned short      CSS_CUSTOM                     = 2;

           attribute DOMString        cssText;
                                        // raises(DOMException) on setting

  readonly attribute unsigned short   valueType;
};
```

Definition group *UnitTypes*

An integer indicating which type of unit applies to the value. ***Note:*** *All CSS2 constants are not supposed to be required by the implementation since all CSS2 interfaces are optionals.*

Defined Constants

CSS_PRIMITIVE_VALUE	The value is a `CSSPrimitiveValue` [p.52] .
CSS_VALUE_LIST	The value is a list `CSSValue`.
CSS_CUSTOM	The value is a custom value.

Attributes

`cssText`

A string representation of the current value.

Exceptions on setting

`DOMException`

SYNTAX_ERR: Raised if the specified CSS string value has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this declaration is readonly.

`valueType`

A code defining the type of the value as defined above.

Interface *CSSPrimitiveValue*

The `CSSPrimitiveValue` interface represents a single CSS value. This interface may be used to determine the value of a specific style property currently set in a block or to set a specific style properties explicitly within the block. An instance of this interface can be obtained from the `getPropertyCSSValue` method of the `CSSStyleDeclaration` [p.48] interface.

IDL Definition

```
interface CSSPrimitiveValue : CSSValue {
  // UnitTypes
  const unsigned short      CSS_UNKNOWN                    = 0;
  const unsigned short      CSS_INHERIT                    = 1;
  const unsigned short      CSS_NUMBER                     = 2;
  const unsigned short      CSS_PERCENTAGE                 = 3;
  const unsigned short      CSS_EMS                        = 4;
  const unsigned short      CSS_EXS                        = 5;
  const unsigned short      CSS_PX                         = 6;
  const unsigned short      CSS_CM                         = 7;
  const unsigned short      CSS_MM                         = 8;
  const unsigned short      CSS_IN                         = 9;
  const unsigned short      CSS_PT                         = 10;
  const unsigned short      CSS_PC                         = 11;
  const unsigned short      CSS_DEG                        = 12;
  const unsigned short      CSS_RAD                        = 13;
  const unsigned short      CSS_GRAD                       = 14;
  const unsigned short      CSS_MS                         = 15;
  const unsigned short      CSS_S                          = 16;
  const unsigned short      CSS_HZ                         = 17;
  const unsigned short      CSS_KHZ                        = 18;
  const unsigned short      CSS_DIMENSION                  = 19;
  const unsigned short      CSS_STRING                     = 20;
  const unsigned short      CSS_URI                        = 21;
  const unsigned short      CSS_IDENT                      = 22;
  const unsigned short      CSS_ATTR                       = 23;
  const unsigned short      CSS_COUNTER                    = 24;
  const unsigned short      CSS_RECT                       = 26;
  const unsigned short      CSS_RGBCOLOR                   = 27;

  readonly attribute unsigned short   primitiveType;
  void               setFloatValue(in unsigned short unitType,
                                   in float floatValue)
                                        raises(DOMException);
  float              getFloatValue(in unsigned short unitType)
                                        raises(DOMException);
  void               setStringValue(in unsigned short stringType,
                                    in DOMString stringValue)
                                        raises(DOMException);
  DOMString          getStringValue()
                                        raises(DOMException);
  Counter            getCounterValue()
                                        raises(DOMException);
  Rect               getRectValue()
                                        raises(DOMException);
  RGBColor           getRGBColorValue()
                                        raises(DOMException);
};
```

Definition group *UnitTypes*

An integer indicating which type of unit applies to the value.
Defined Constants

CSS_UNKNOWN	The value is not a recognized CSS2 value. The value can only be obtained by using the `cssText` attribute.
CSS_INHERIT	The value is the `inherit` identifier. The string representation of this value can be obtained by using the `getStringValue` method.
CSS_NUMBER	The value is a simple number. The value can be obtained by using the `getFloatValue` method.
CSS_PERCENTAGE	The value is a percentage. The value can be obtained by using the `getFloatValue` method.
CSS_EMS	The value is length (ems). The value can be obtained by using the `getFloatValue` method.
CSS_EXS	The value is length (exs). The value can be obtained by using the `getFloatValue` method.
CSS_PX	The value is length (px). The value can be obtained by using the `getFloatValue` method.
CSS_CM	The value is length (cm). The value can be obtained by using the `getFloatValue` method.
CSS_MM	The value is length (mm). The value can be obtained by using the `getFloatValue` method.
CSS_IN	The value is length (in). The value can be obtained by using the `getFloatValue` method.
CSS_PT	The value is length (pt). The value can be obtained by using the `getFloatValue` method.
CSS_PC	The value is a length (pc). The value can be obtained by using the `getFloatValue` method.
CSS_DEG	The value is an angle (deg). The value can be obtained by using the `getFloatValue` method.
CSS_RAD	The value is an angle (rad). The value can be obtained by using the `getFloatValue` method.
CSS_GRAD	The value is an angle (grad). The value can be obtained by using the `getFloatValue` method.

CSS_MS	The value is a time (ms). The value can be obtained by using the `getFloatValue` method.
CSS_S	The value is a time (s). The value can be obtained by using the `getFloatValue` method.
CSS_HZ	The value is a frequency (Hz). The value can be obtained by using the `getFloatValue` method.
CSS_KHZ	The value is a frequency (kHz). The value can be obtained by using the `getFloatValue` method.
CSS_DIMENSION	The value is a number with an unknown dimension. The value can be obtained by using the `getFloatValue` method.
CSS_STRING	The value is a STRING. The value can be obtained by using the `getStringValue` method.
CSS_URI	The value is a URI. The value can be obtained by using the `getStringValue` method.
CSS_IDENT	The value is an identifier. The value can be obtained by using the `getStringValue` method.
CSS_ATTR	The value is a attribute function. The value can be obtained by using the `getStringValue` method.
CSS_COUNTER	The value is a counter or counters function. The value can be obtained by using the `getCounterValue` method.
CSS_RECT	The value is a rect function. The value can be obtained by using the `getRectValue` method.
CSS_RGBCOLOR	The value is a RGB color. The value can be obtained by using the `getRGBColorValue` method.

Attributes

`primitiveType`

The type of the value as defined by the constants specified above.

Methods

`setFloatValue`

A method to set the float value with a specified unit. If the property attached with this value can not accept the specified unit or the float value, the value will be unchanged and a `DOMException` will be raised.

Parameters

`unitType`	A unit code as defined above. The unit code can only be a float unit type (e.g. `NUMBER`, `PERCENTAGE`, `CSS_EMS`, `CSS_EXS`, `CSS_PX`, `CSS_PX`, `CSS_CM`, `CSS_MM`, `CSS_IN`, `CSS_PT`, `CSS_PC`, `CSS_DEG`, `CSS_RAD`, `CSS_GRAD`, `CSS_MS`, `CSS_S`, `CSS_HZ`, `CSS_KHZ`, `CSS_DIMENSION`).
`floatValue`	The new float value.

Exceptions

`DOMException`

INVALID_ACCESS_ERR: Raises if the attached property doesn’t support the float value or the unit type.

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

This method returns nothing.

`getFloatValue`

This method is used to get a float value in a specified unit. If this CSS value doesn’t contain a float value or can’t be converted into the specified unit, a `DOMException` is raised.

Parameters

`unitType`	A unit code to get the float value. The unit code can only be a float unit type (e.g. `CSS_NUMBER`, `CSS_PERCENTAGE`, `CSS_EMS`, `CSS_EXS`, `CSS_PX`, `CSS_PX`, `CSS_CM`, `CSS_MM`, `CSS_IN`, `CSS_PT`, `CSS_PC`, `CSS_DEG`, `CSS_RAD`, `CSS_GRAD`, `CSS_MS`, `CSS_S`, `CSS_HZ`, `CSS_KHZ`, `CSS_DIMENSION`).

Return Value

The float value in the specified unit.

Exceptions

`DOMException`

INVALID_ACCESS_ERR: Raises if the CSS value doesn’t contain a float value or if the float value can’t be converted into the specified unit.

`setStringValue`

A method to set the string value with a specified unit. If the property attached to this value can’t accept the specified unit or the string value, the value will be unchanged and a `DOMException` will be raised.

Parameters

`stringType`	A string code as defined above. The string code can only be a string unit type (e.g. `CSS_URI`, `CSS_IDENT`, `CSS_INHERIT` and `CSS_ATTR`).
`stringValue`	The new string value. If the `stringType` is equal to `CSS_INHERIT`, the `stringValue` should be `inherit`.

Exceptions

`DOMException`

INVALID_ACCESS_ERR: Raises if the CSS value doesn’t contain a string value or if the string value can’t be converted into the specified unit.

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

This method returns nothing.

`getStringValue`

This method is used to get the string value in a specified unit. If the CSS value doesn’t contain a string value, a `DOMException` is raised.

Return Value

The string value in the current unit. The current `valueType` can only be a string unit type (e.g. `CSS_URI`, `CSS_IDENT` and `CSS_ATTR`).

Exceptions

`DOMException`

INVALID_ACCESS_ERR: Raises if the CSS value doesn’t contain a string value.

This method has no parameters.

`getCounterValue`

This method is used to get the Counter value. If this CSS value doesn’t contain a counter value, a `DOMException` is raised. Modification to the corresponding style property can be achieved using the `Counter` [p.58] interface.

Return Value

The Counter value.

Exceptions

`DOMException`

INVALID_ACCESS_ERR: Raises if the CSS value doesn’t contain a Counter value.

This method has no parameters.

`getRectValue`

This method is used to get the Rect value. If this CSS value doesn’t contain a rect value, a `DOMException` is raised. Modification to the corresponding style property can be achieved using the `Rect` [p.58] interface.

Return Value

The Rect value.

Exceptions

`DOMException`

INVALID_ACCESS_ERR: Raises if the CSS value doesn't contain a Rect value.

This method has no parameters.

`getRGBColorValue`

This method is used to get the RGB color. If this CSS value doesn't contain a RGB color value, a `DOMException` is raised. Modification to the corresponding style property can be achieved using the `RGBColor` [p.57] interface.

Return Value

the RGB color value.

Exceptions

`DOMException`

INVALID_ACCESS_ERR: Raises if the attached property can't return a RGB color value.

This method has no parameters.

Interface *CSSValueList*

The `CSSValueList` interface provides the absraction of an ordered collection of CSS values.

IDL Definition

```
interface CSSValueList : CSSValue {
  readonly attribute unsigned long    length;
  CSSValue           item(in unsigned long index);
};
```

Attributes

`length`

The number of `CSSValue` [p.51] s in the list. The range of valid values indices is `0` to `length-1` inclusive.

Methods

`item`

Used to retrieve a CSS rule by ordinal index. The order in this collection represents the order of the values in the CSS style property.

Parameters

`index`	Index into the collection.

Return Value

The style rule at the `index` position in the `CSSValueList`, or `null` if that is not valid index.

This method raises no exceptions.

Interface *RGBColor*

The `RGBColor` interface is used to represent any RGB color value. This interface reflects the values in the underlying style property. Hence, modifications made through this interface modify the style property.

IDL Definition

```
interface RGBColor {
           attribute CSSValue         red;
           attribute CSSValue         green;
           attribute CSSValue         blue;
};
```

Attributes

`red`
: This attribute is used for the red value of the RGB color.

`green`
: This attribute is used for the green value of the RGB color.

`blue`
: This attribute is used for the blue value of the RGB color.

Interface *Rect*

The `Rect` interface is used to represent any rect value. This interface reflects the values in the underlying style property. Hence, modifications made through this interface modify the style property.

IDL Definition

```
interface Rect {
           attribute CSSValue         top;
           attribute CSSValue         right;
           attribute CSSValue         bottom;
           attribute CSSValue         left;
};
```

Attributes

`top`
: This attribute is used for the top of the rect.

`right`
: This attribute is used for the right of the rect.

`bottom`
: This attribute is used for the bottom of the rect.

`left`
: This attribute is used for the left of the rect.

Interface *Counter*

The `Counter` interface is used to represent any counter or counters function value. This interface reflects the values in the underlying style property. Hence, modifications made through this interface modify the style property.

IDL Definition

```
interface Counter {
           attribute DOMString        identifier;
           attribute DOMString        listStyle;
           attribute DOMString        separator;
};
```

Attributes

`identifier`
: This attribute is used for the identifier of the counter.

`listStyle`
: This attribute is used for the style of the list.

`separator`
: This attribute is used for the separator of nested counters.

4.3. CSS Extended Interfaces

The interfaces found within this section are not mandatory. A DOM consumer can use the hasFeature of the DOMImplementation interface to determine whether the CSS2 extended interfaces have been implemented by a DOM implementation. The feature string for all the extended interfaces listed in this section except the `CSS2Properties` [p.79] interface is "CSS2".

The following table specifies the type of `CSSValue` [p.51] used to represent each property that can be specified in a `CSSStyleDeclaration` [p.48] found in a `CSSStyleRule` [p.44] for a CSS Level 2 style sheet. The expectation is that the `CSSValue` [p.51] returned from the `getPropertyCSSValue` method on the `CSSStyleDeclaration` [p.48] interface can be cast down, using binding-specific casting methods, to the specific derived interface.

For properties that are represented by a custom interface (the `valueType` of the `CSSValue` [p.51] is `CSS_CUSTOM`), the name of the derived interface is specified in the table. For properties that consist of lists of values (the `valueType` of the `CSSValue` [p.51] is `CSS_VALUE_LIST`), the derived interface is `CSSValueList` [p.57] . For all other properties (the `valueType` of the `CSSValue` [p.51] is `CSS_PRIMITIVE_VALUE`), the derived interface is `CSSPrimitiveValue` [p.52] .

Property Name	**Representation**
azimuth	`CSS2Azimuth` [p.63]
background	`null`
background-attachment	ident
background-color	rgbcolor, ident
background-image	uri, ident
background-position	`CSS2BackgroundPosition` [p.65]

background-repeat	ident
border	`null`
border-collapse	ident
border-color	`null`
border-spacing	`CSS2BorderSpacing` [p.67]
border-style	`null`
border-top, border-right, border-bottom, border-left	`null`
border-top-color, border-right-color, border-bottom-color, border-left-color	rgbcolor, ident
border-top-style, border-right-style, border-bottom-style, border-left-style	ident
border-top-width, border-right-width, border-bottom-width, border-left-width	length, ident
border-width	`null`
bottom	length, percentage, ident
caption-side	ident
clear	ident
clip	rect, ident
color	rgbcolor, ident
content	list of string, uri, counter, attr, ident
counter-increment	list of `CSS2CounterIncrement` [p.70]
counter-reset	list of `CSS2CounterReset` [p.70]
cue	`null`
cue-after, cue-before	uri, ident
cursor	`CSS2Cursor` [p.71]
direction	ident
display	ident

elevation	angle, ident
empty-cells	ident
float	ident
font	`null`
font-family	list of strings and idents
font-size	ident, length, percentage
font-size-adjust	number, ident
font-stretch	ident
font-style	ident
font-variant	ident
font-weight	ident
height	length, percentage, ident
left	length, percentage, ident
letter-spacing	ident, length
line-height	ident, length, percentage, number
list-style	`null`
list-style-image	uri, ident
list-style-position	ident
list-style-type	ident
margin	`null`
margin-top, margin-right, margin-bottom, margin-left	length, percentage, ident
marker-offset	length, ident
max-height	length, percentage, ident
max-width	length, percentage, ident
min-height	length, percentage, ident
min-width	length, percentage, ident
orphans	number
outline	`null`

outline-color	rgbcolor, ident
outline-style	ident
outline-width	length, ident
overflow	ident
padding	`null`
padding-top, padding-right, padding-bottom, padding-left	length, percentage
page	ident
page-break-after	ident
page-break-before	ident
page-break-inside	ident
pause	`null`
pause-after, pause-before	time, percentage
pitch	frequency, identifier
pitch-range	number
play-during	`CSS2PlayDuring` [p.72]
position	ident
quotes	list of string or ident
richness	number
right	length, percentage, ident
speak	ident
speak-header	ident
speak-numeral	ident
speak-punctuation	ident
speech-rate	number, ident
stress	number
table-layout	ident
text-align	ident, string

text-decoration	list of ident
text-indent	length, percentage
text-shadow	list of `CSS2TextShadow` [p.73]
text-transform	ident
top	length, percentage, ident
unicode-bidi	ident
vertical-align	ident, percentage, length
visibility	ident
voice-family	list of strings and idents
volume	number, percentage, ident
white-space	ident
widows	number
width	length, percentage, ident
word-spacing	length, ident
z-index	ident, number

Interface *CSS2Azimuth*

The `CSS2Azimuth` interface represents the azimuth CSS Level 2 property.
IDL Definition

```
interface CSS2Azimuth : CSSValue {
  readonly attribute unsigned short   azimuthType;
  readonly attribute DOMString        identifier;
  readonly attribute boolean          behind;
  void               setAngleValue(in unsigned short unitType,
                                   in float floatValue)
                                        raises(DOMException);
  float              getAngleValue(in unsigned short unitType)
                                        raises(DOMException);
  void               setIdentifier(in DOMString identifier,
                                   in boolean behind)
                                        raises(DOMException);
};
```

Attributes

`azimuthType`

A code defining the type of the value as defined in `CSSValue` [p.51] . It would be one of `CSS_DEG`, `CSS_RAD`, `CSS_GRAD` or `CSS_IDENT`.

`identifier`

If `azimuthType` is `CSS_IDENT`, `identifier` contains one of left-side, far-left, left, center-left, center, center-right, right, far-right, right-side, leftwards, rightwards. The empty string if none is set.

`behind`

`behind` indicates whether the behind identifier has been set.

Methods

`setAngleValue`

A method to set the angle value with a specified unit. This method will unset any previously set identifiers values.

Parameters

`unitType`	The unitType could only be one of `CSS_DEG`, `CSS_RAD` or `CSS_GRAD`).
`floatValue`	The new float value of the angle.

Exceptions

`DOMException`

INVALID_ACCESS_ERR: Raised if the unit type is invalid.

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

This method returns nothing.

`getAngleValue`

Used to retrieved the float value of the azimuth property.

Parameters

`unitType`	The unit type can be only an angle unit type (`CSS_DEG`, `CSS_RAD` or `CSS_GRAD`).

Return Value

The float value.

Exceptions

`DOMException`

INVALID_ACCESS_ERR: Raised if the unit type is invalid.

`setIdentifier`

Setting the identifier for the azimuth property will unset any previously set angle value. The value of `azimuthType` is set to `CSS_IDENT`

Parameters

`identifier`	The new identifier. If the identifier is "leftwards" or "rightward", the behind attribute is ignored.
`behind`	The new value for behind.

Exceptions

`DOMException`

SYNTAX_ERR: Raised if the specified `identifier` has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

This method returns nothing.

Interface *CSS2BackgroundPosition*

The `CSS2BackgroundPosition` interface represents the background-position CSS Level 2 property.

IDL Definition

```
interface CSS2BackgroundPosition : CSSValue {
  readonly attribute unsigned short   horizontalType;
  readonly attribute unsigned short   verticalType;
  readonly attribute DOMString        horizontalIdentifier;
  readonly attribute DOMString        verticalIdentifier;
  float              getHorizontalPosition(in float horizontalType)
                                        raises(DOMException);
  float              getVerticalPosition(in float verticalType)
                                        raises(DOMException);
  void               setHorizontalPosition(in unsigned short horizontalType,
                                           in float value)
                                        raises(DOMException);
  void               setVerticalPosition(in unsigned short verticalType,
                                         in float value)
                                        raises(DOMException);
  void               setPositionIdentifier(in DOMString horizontalIdentifier,
                                           in DOMString verticalIdentifier)
                                        raises(DOMException);
};
```

Attributes

`horizontalType`

A code defining the type of the horizontal value. It would be one `CSS_PERCENTAGE`, `CSS_EMS`, `CSS_EXS`, `CSS_PX`, `CSS_CM`, `CSS_MM`, `CSS_IN`, `CSS_PT`, `CSS_PC`, `CSS_IDENT`, `CSS_INHERIT`. If one of horizontal or vertical is `CSS_IDENT` or `CSS_INHERIT`, it’s guaranteed that the other is the same.

`verticalType`

A code defining the type of the horizontal value. The code can be one of the following units : `CSS_PERCENTAGE`, `CSS_EMS`, `CSS_EXS`, `CSS_PX`, `CSS_CM`, `CSS_MM`, `CSS_IN`, `CSS_PT`, `CSS_PC`, `CSS_IDENT`, `CSS_INHERIT`. If one of horizontal or vertical is `CSS_IDENT` or `CSS_INHERIT`, it’s guaranteed that the other is the same.

`horizontalIdentifier`

If `horizontalType` is `CSS_IDENT` or `CSS_INHERIT`, this attribute contains the string representation of the ident, otherwise it contains an empty string.

`verticalIdentifier`

If `verticalType` is `CSS_IDENT` or `CSS_INHERIT`, this attribute contains the string representation of the ident, otherwise it contains an empty string. The value is `"center"` if only the horizontalIdentifier has been set. The value is `"inherit"` if the horizontalIdentifier is `"inherit"`.

Methods

`getHorizontalPosition`

This method is used to get the float value in a specified unit if the `horizontalPosition` represents a length or a percentage. If the float doesn’t contain a float value or can’t be converted into the specified unit, a `DOMException` is raised.

Parameters

`horizontalType`	The specified unit.

Return Value

The float value.

Exceptions

`DOMException`

INVALID_ACCESS_ERR: Raises if the property doesn’t contain a float or the value can’t be converted.

`getVerticalPosition`

This method is used to get the float value in a specified unit if the `verticalPosition` represents a length or a percentage. If the float doesn’t contain a float value or can’t be converted into the specified unit, a `DOMException` is raised. The value is `50%` if only the horizontal value has been specified.

Parameters

`verticalType`	The specified unit.

Return Value

The float value.

Exceptions

`DOMException`

INVALID_ACCESS_ERR: Raises if the property doesn’t contain a float or the value can’t be converted.

`setHorizontalPosition`

This method is used to set the horizontal position with a specified unit. If the vertical value is not a percentage or a length, it sets the vertical position to `50%`.

Parameters

`horizontalType`	The specified unit (a length or a percentage).
`value`	The new value.

Exceptions

`DOMException`

INVALID_ACCESS_ERR: Raises if the specified unit is not a length or a percentage.

NO_MODIFICATION_ALLOWED_ERR: Raises if this property is readonly.

This method returns nothing.

`setVerticalPosition`

This method is used to set the vertical position with a specified unit. If the horizontal value is not a percentage or a length, it sets the vertical position to `50%`.

Parameters

`verticalType`	The specified unit (a length or a percentage).
`value`	The new value.

Exceptions

`DOMException`

INVALID_ACCESS_ERR: Raises if the specified unit is not a length or a percentage.

NO_MODIFICATION_ALLOWED_ERR: Raises if this property is readonly.

This method returns nothing.

`setPositionIdentifier`

Sets the identifiers. If the second identifier is the empty string, the vertical identifier is set to his default value (`"center"`). If the first identfier is `"inherit`, the second identifier is ignored and is set to `"inherit"`.

Parameters

`horizontalIdentifier`	The new horizontal identifier.
`verticalIdentifier`	The new vertical identifier.

Exceptions

`DOMException`

SYNTAX_ERR: Raises if the identifiers have a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raises if this property is readonly.

This method returns nothing.

Interface *CSS2BorderSpacing*

The `CSS2BorderSpacing` interface represents the border-spacing CSS Level 2 property.

IDL Definition

```
interface CSS2BorderSpacing : CSSValue {
  readonly attribute unsigned short   horizontalType;
  readonly attribute unsigned short   verticalType;
  float              getHorizontalSpacing(in float horizontalType)
                                        raises(DOMException);
  float              getVerticalSpacing(in float verticalType)
                                        raises(DOMException);
  void               setHorizontalSpacing(in unsigned short horizontalType,
                                          in float value)
                                        raises(DOMException);
  void               setVerticalSpacing(in unsigned short verticalType,
                                        in float value)
                                        raises(DOMException);
  void               setInherit()();
};
```

Attributes

`horizontalType`

The A code defining the type of the value as defined in `CSSValue` [p.51] . It would be one of `CSS_EMS`, `CSS_EXS`, `CSS_PX`, `CSS_CM`, `CSS_MM`, `CSS_IN`, `CSS_PT`, `CSS_PC` or `CSS_INHERIT`.

`verticalType`

The A code defining the type of the value as defined in `CSSValue` [p.51] . It would be one of `CSS_EMS`, `CSS_EXS`, `CSS_PX`, `CSS_CM`, `CSS_MM`, `CSS_IN`, `CSS_PT`, `CSS_PC` or `CSS_INHERIT`.

Methods

`getHorizontalSpacing`

This method is used to get the float value in a specified unit if the `horizontalSpacing` represents a length. If the float doesn’t contain a float value or can’t be converted into the specified unit, a `DOMException` is raised.

Parameters

`horizontalType`	The specified unit.

Return Value

The float value.

Exceptions

`DOMException`

INVALID_ACCESS_ERR: Raises if the property doesn’t contain a float or the value can’t be converted.

`getVerticalSpacing`

This method is used to get the float value in a specified unit if the `verticalSpacing` represents a length. If the float doesn’t contain a float value or can’t be converted into the specified unit, a `DOMException` is raised. The value is `0` if only the horizontal value has been specified.

Parameters

`verticalType`	The specified unit.

Return Value

The float value.

Exceptions

`DOMException`

INVALID_ACCESS_ERR: Raises if the property doesn't contain a float or the value can't be converted.

`setHorizontalSpacing`

This method is used to set the horizontal spacing with a specified unit. If the vertical value is a length, it sets the vertical spacing to `0`.

Parameters

`horizontalType`	The specified unit.
`value`	The new value.

Exceptions

`DOMException`

INVALID_ACCESS_ERR: Raises if the specified unit is not a length.

NO_MODIFICATION_ALLOWED_ERR: Raises if this property is readonly.

This method returns nothing.

`setVerticalSpacing`

This method is used to set the vertical spacing with a specified unit. If the horizontal value is not a length, it sets the vertical spacing to `0`.

Parameters

`verticalType`	The specified unit.
`value`	The new value.

Exceptions

`DOMException`

INVALID_ACCESS_ERR: Raises if the specified unit is not a length or a percentage.

NO_MODIFICATION_ALLOWED_ERR: Raises if this property is readonly.

This method returns nothing.

`setInherit()`

Set this property as inherit. `horizontalType` and `verticalType` will be inherited.
This method has no parameters.
This method returns nothing.
This method raises no exceptions.

Interface *CSS2CounterReset*

The `CSS2CounterReset` interface represents a simple value for the counter-reset CSS Level 2 property.

IDL Definition

```
interface CSS2CounterReset {
          attribute DOMString        identifier;
                                       // raises(DOMException) on setting

          attribute short            reset;
                                       // raises(DOMException) on setting

};
```

Attributes

`identifier`

The element name.

Exceptions on setting

`DOMException`

SYNTAX_ERR: Raised if the specified identifier has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this identifier is readonly.

`reset`

The reset (default value is 0).

Exceptions on setting

`DOMException`

NO_MODIFICATION_ALLOWED_ERR: Raised if this identifier is readonly.

Interface *CSS2CounterIncrement*

The `CSS2CounterIncrement` interface represents a imple value for the counter-increment CSS Level 2 property.

IDL Definition

```
interface CSS2CounterIncrement {
           attribute DOMString        identifier;
                                        // raises(DOMException) on setting

           attribute short            increment;
                                        // raises(DOMException) on setting

};
```

Attributes

`identifier`

The element name.

Exceptions on setting

`DOMException`

SYNTAX_ERR: Raised if the specified identifier has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this identifier is readonly.

`increment`

The increment (default value is 1).

Exceptions on setting

`DOMException`

NO_MODIFICATION_ALLOWED_ERR: Raised if this identifier is readonly.

Interface *CSS2Cursor*

The `CSS2Cursor` interface represents the cursor CSS Level 2 property.

IDL Definition

```
interface CSS2Cursor : CSSValue {
           attribute unsigned short   cursorType;
  readonly attribute CSSValueList     uris;
           attribute DOMString        predefinedCursor;
                                        // raises(DOMException) on setting

};
```

Attributes

`cursorType`

A code defining the type of the property. It would one of `CSS_UNKNOWN` or `CSS_INHERIT`. If the type is `CSS_UNKNOWN`, then `uris` contains a list of URIs and `predefinedCursor` contains an ident. Setting this attribute from `CSS_INHERIT` to `CSS_UNKNOWN` will set the `predefinedCursor` to `"auto"`.

`uris`

`uris` represents the list of URIs (`CSS_URI`) on the cursor property. The list can be empty.

`predefinedCursor`

This identifier represents a generic cursor name or an empty string.

Exceptions on setting

`DOMException`

SYNTAX_ERR: Raised if the specified CSS string value has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this declaration is readonly.

Interface *CSS2PlayDuring*

The `CSS2PlayDuring` interface represents the play-during CSS Level 2 property.

IDL Definition

```
interface CSS2PlayDuring : CSSValue {
  readonly attribute unsigned short   playDuringType;
           attribute DOMString        playDuringIdentifier;
                                        // raises(DOMException) on setting

           attribute DOMString        uri;
                                        // raises(DOMException) on setting

           attribute boolean          mix;
                                        // raises(DOMException) on setting

           attribute boolean          repeat;
                                        // raises(DOMException) on setting

};
```

Attributes

`playDuringType`

A code defining the type of the value as define in `CSSvalue`. It would be one of `CSS_UNKNOWN`, `CSS_INHERIT`, `CSS_IDENT`.

`playDuringIdentifier`

One of `"inherit"`, `"auto"`, `"none"` or the empty string if the `playDuringType` is `CSS_UNKNOWN`. On setting, it will set the `uri` to the empty string and `mix` and `repeat` to `false`.

Exceptions on setting

`DOMException`

SYNTAX_ERR: Raised if the specified CSS string value has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this declaration is readonly.

`uri`

The sound specified by the `uri`. It will set the `playDuringType` attribute to `CSS_UNKNOWN`.

Exceptions on setting

`DOMException`

SYNTAX_ERR: Raised if the specified CSS string value has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this declaration is readonly.

`mix`

`true` if the sound should be mixed. It will be ignored if the attribute doesn't contain a `uri`.

Exceptions on setting

`DOMException`

NO_MODIFICATION_ALLOWED_ERR: Raised if this declaration is readonly.

`repeat`

`true` if the sound should be repeated. It will be ignored if the attribute doesn't contain a `uri`.

Exceptions on setting

`DOMException`

NO_MODIFICATION_ALLOWED_ERR: Raised if this declaration is readonly.

Interface *CSS2TextShadow*

The `CSS2TextShadow` interface represents a simple value for the text-shadow CSS Level 2 property.

IDL Definition

```
interface CSS2TextShadow {
  readonly attribute CSSValue         color;
  readonly attribute CSSValue         horizontal;
  readonly attribute CSSValue         vertical;
  readonly attribute CSSValue         blur;
};
```

Attributes

`color`

Specified the color of the text shadow. The CSS Value can contain an empty string if no color has been specified.

`horizontal`

The horizontal position of the text shadow. `0` if no length has been specified.

`vertical`

The vertical position of the text shadow. `0` if no length has been specified.

`blur`

The blur radius of the text shadow. `0` if no length has been specified.

The following table specifies the type of `CSSValue` [p.51] used to represent each property that can be specified in a `CSSStyleDeclaration` [p.48] found in a `CSSFontFaceRule` [p.46] for a CSS Level 2 style sheet.

Property Name	**Representation**
font-family	list of strings and idents
font-style	list of idents
font-variant	list of idents
font-weight	list of idents
font-stretch	list of idents
font-size	list of lengths or ident
unicode-range	list of strings
units-per-em	number
src	list of `CSS2FontFaceSrc` [p.74]
panose-1	list of integers
stemv	number
stemh	number
slope	number
cap-height	number
x-height	number
ascent	number
descent	number
widths	list of `CSS2FontFaceWidths` [p.75]
bbox	list of numbers
definition-src	uri
baseline	number
centerline	number
mathline	number
topline	number

Interface *CSS2FontFaceSrc*

The `CSS2Cursor` [p.71] interface represents the src CSS Level 2 descriptor.
IDL Definition

```
interface CSS2FontFaceSrc {
           attribute DOMString        uri;
                                        // raises(DOMException) on setting

  readonly attribute CSSValueList     format;
           attribute DOMString        fontFaceName;
                                        // raises(DOMException) on setting

};
```

Attributes

`uri`
Specifies the source of the font, empty string otherwise.
Exceptions on setting
`DOMException`

SYNTAX_ERR: Raised if the specified CSS string value has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this declaration is readonly.

`format`
This attribute contains a list of strings for the format CSS function.

`fontFaceName`
Specifies the full font name of a locally installed font.
Exceptions on setting
`DOMException`

SYNTAX_ERR: Raised if the specified CSS string value has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this declaration is readonly.

Interface *CSS2FontFaceWidths*

The `CSS2Cursor` [p.71] interface represents a simple value for the widths CSS Level 2 descriptor.
IDL Definition

```
interface CSS2FontFaceWidths {
           attribute DOMString        urange;
                                        // raises(DOMException) on setting

  readonly attribute CSSValueList     numbers;
};
```

Attributes

`urange`
The range for the characters.

Exceptions on setting
`DOMException`

SYNTAX_ERR: Raised if the specified CSS string value has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this declaration is readonly.

`numbers`
A list of numbers representing the glyph widths.

The following table specifies the type of `CSSValue` [p.51] used to represent each property that can be specified in a `CSSStyleDeclaration` [p.48] found in a `CSSPageRule` [p.46] for a CSS Level 2 style sheet.

Property Name	**Representation**
margin	`null`
margin-top, margin,right, margin-bottom, margin-left	length (no CSS_EMS and CSS_EXS), percentage, ident
marks	list of idents
size	`CSS2PageSize` [p.76]

Interface *CSS2PageSize*

The `CSS2Cursor` [p.71] interface represents the size CSS Level 2 descriptor.
IDL Definition

```
interface CSS2PageSize : CSSValue {
  readonly attribute unsigned short   widthType;
  readonly attribute unsigned short   heightType;
  readonly attribute DOMString        identifier;
  float              getWidth(in float widthType)
                                        raises(DOMException);
  float              getHeightSize(in float heightType)
                                        raises(DOMException);
  void               setWidthSize(in unsigned short widthType,
                                  in float value)
                                        raises(DOMException);
  void               setHeightSize(in unsigned short heightType,
                                   in float value)
                                        raises(DOMException);
  void               setIdentifier(in DOMString identifier)
                                        raises(DOMException);
};
```

Attributes
`widthType`
A code defining the type of the width of the page. It would be one of `CSS_EMS`, `CSS_EXS`, `CSS_PX`, `CSS_CM`, `CSS_MM`, `CSS_IN`, `CSS_PT`, `CSS_PC`, `CSS_IDENT`,

`CSS_INHERIT`. If one of width or height is `CSS_IDENT` or `CSS_INHERIT`, it’s guaranteed that the other is the same.

`heightType`

A code defining the type of the height of the page. It would be one of `CSS_EMS`, `CSS_EXS`, `CSS_PX`, `CSS_CM`, `CSS_MM`, `CSS_IN`, `CSS_PT`, `CSS_PC`, `CSS_IDENT`, `CSS_INHERIT`. If one of width or height is `CSS_IDENT` or `CSS_INHERIT`, it’s guaranteed that the other is the same.

`identifier`

If `width` is `CSS_IDENT` or `CSS_INHERIT`, this attribute contains the string representation of the ident, otherwise it contains an empty string.

Methods

`getWidth`

This method is used to get the float value in a specified unit if the `widthType` represents a length. If the float doesn’t contain a float value or can’t be converted into the specified unit, a `DOMException` is raised.

Parameters

`widthType`	The specified unit.

Return Value

The float value.

Exceptions

`DOMException`

INVALID_ACCESS_ERR: Raises if the property doesn’t contain a float or the value can’t be converted.

`getHeightSize`

This method is used to get the float value in a specified unit if the `heightType` represents a length. If the float doesn’t contain a float value or can’t be converted into the specified unit, a `DOMException` is raised. If only the width value has been specified, the height value is the same.

Parameters

`heightType`	The specified unit.

Return Value

The float value.

Exceptions

`DOMException`

INVALID_ACCESS_ERR: Raises if the property doesn’t contain a float or the value can’t be converted.

`setWidthSize`

This method is used to set the width position with a specified unit. If the `heightType` is not a length, it sets the height position to the same value.

Parameters

`widthType`	The specified unit.
`value`	The new value.

Exceptions

`DOMException`

INVALID_ACCESS_ERR: Raises if the specified unit is not a length or a percentage.

NO_MODIFICATION_ALLOWED_ERR: Raises if this property is readonly.

This method returns nothing.

`setHeightSize`

This method is used to set the height position with a specified unit. If the `widthType` is not a length, it sets the width position to the same value.

Parameters

`heightType`	The specified unit.
`value`	The new value.

Exceptions

`DOMException`

INVALID_ACCESS_ERR: Raises if the specified unit is not a length or a percentage.

NO_MODIFICATION_ALLOWED_ERR: Raises if this property is readonly.

This method returns nothing.

`setIdentifier`

Sets the identifier.

Parameters

`identifier`	The new identifier.

Exceptions

`DOMException`

SYNTAX_ERR: Raises if the identifier has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raises if this property is readonly.

This method returns nothing.

The following interface may be implemented by a DOM implementation as a convenience to the DOM script user. A DOM consumer can use the hasFeature of the DOMImplementation interface to determine whether the `CSS2Properties` [p.79] interface has been implemented by the DOM implementation. The festure string for the `CSS2Properties` [p.79] interface is "CSS2Properties".

Interface *CSS2Properties*

The `CSS2Properties` interface represents a convenience mechanism for retrieving and setting properties within a `CSSStyleDeclaration` [p.48] . The attributes of this interface correspond to all the properties specified in CSS2. Getting an attribute of this interface is equivalent to calling the `getPropertyValue` method of the `CSSStyleDeclaration` [p.48] interface. Setting an attribute of this interface is equivalent to calling the `setProperty` method of the `CSSStyleDeclaration` [p.48] interface.

A compliant implementation is not required to implement the `CSS2Properties` interface. If an implementation does implement this interface, the expectation is that language-specific methods can be used to cast from an instance of the `CSSStyleDeclaration` [p.48] interface to the `CSS2Properties` interface.

If an implementation does implement this interface, it is expected to understand the specific syntax of the shorthand properties, and apply their semantics; when the `margin` property is set, for example, the `marginTop`, `marginRight`, `marginBottom` and `marginLeft` properties are actually being set by the underlying implementation.

When dealing with CSS "shorthand" properties, the shorthand properties should be decomposed into their component longhand properties as appropriate, and when querying for their value, the form returned should be the shortest form exactly equivalent to the declarations made in the ruleset. However, if there is no shorthand declaration that could be added to the ruleset without changing in any way the rules already declared in the ruleset (i.e., by adding longhand rules that were previously not declared in the ruleset), then the empty string should be returned for the shorthand property.

For example, querying for the `font` property should not return "normal normal normal 14pt/normal Arial, sans-serif", when "14pt Arial, sans-serif" suffices (the normals are initial values, and are implied by use of the longhand property).

If the values for all the longhand properties that compose a particular string are the initial values, then a string consisting of all the initial values should be returned (e.g. a `border-width` value of "medium" should be returned as such, not as "").

For some shorthand properties that take missing values from other sides, such as the `margin`, `padding`, and `border-[width|style|color]` properties, the minimum number of sides possible should be used, i.e., "0px 10px" will be returned instead of "0px 10px 0px 10px".

If the value of a shorthand property can not be decomposed into its component longhand properties, as is the case for the `font` property with a value of "menu", querying for the values of the component longhand properties should return the empty string.

IDL Definition

```
interface CSS2Properties {
           attribute DOMString        azimuth;
           attribute DOMString        background;
           attribute DOMString        backgroundAttachment;
           attribute DOMString        backgroundColor;
           attribute DOMString        backgroundImage;
           attribute DOMString        backgroundPosition;
           attribute DOMString        backgroundRepeat;
           attribute DOMString        border;
           attribute DOMString        borderCollapse;
           attribute DOMString        borderColor;
           attribute DOMString        borderSpacing;
           attribute DOMString        borderStyle;
           attribute DOMString        borderTop;
           attribute DOMString        borderRight;
           attribute DOMString        borderBottom;
           attribute DOMString        borderLeft;
           attribute DOMString        borderTopColor;
           attribute DOMString        borderRightColor;
           attribute DOMString        borderBottomColor;
           attribute DOMString        borderLeftColor;
           attribute DOMString        borderTopStyle;
           attribute DOMString        borderRightStyle;
           attribute DOMString        borderBottomStyle;
           attribute DOMString        borderLeftStyle;
           attribute DOMString        borderTopWidth;
           attribute DOMString        borderRightWidth;
           attribute DOMString        borderBottomWidth;
           attribute DOMString        borderLeftWidth;
           attribute DOMString        borderWidth;
           attribute DOMString        bottom;
           attribute DOMString        captionSide;
           attribute DOMString        clear;
           attribute DOMString        clip;
           attribute DOMString        color;
           attribute DOMString        content;
           attribute DOMString        counterIncrement;
           attribute DOMString        counterReset;
           attribute DOMString        cue;
           attribute DOMString        cueAfter;
           attribute DOMString        cueBefore;
           attribute DOMString        cursor;
           attribute DOMString        direction;
           attribute DOMString        display;
           attribute DOMString        elevation;
           attribute DOMString        emptyCells;
           attribute DOMString        cssFloat;
           attribute DOMString        font;
           attribute DOMString        fontFamily;
           attribute DOMString        fontSize;
           attribute DOMString        fontSizeAdjust;
           attribute DOMString        fontStretch;
           attribute DOMString        fontStyle;
           attribute DOMString        fontVariant;
           attribute DOMString        fontWeight;
           attribute DOMString        height;
           attribute DOMString        left;
           attribute DOMString        letterSpacing;
           attribute DOMString        lineHeight;
           attribute DOMString        listStyle;
           attribute DOMString        listStyleImage;
           attribute DOMString        listStylePosition;
           attribute DOMString        listStyleType;
           attribute DOMString        margin;
           attribute DOMString        marginTop;
           attribute DOMString        marginRight;
           attribute DOMString        marginBottom;
           attribute DOMString        marginLeft;
           attribute DOMString        markerOffset;
           attribute DOMString        marks;
           attribute DOMString        maxHeight;
           attribute DOMString        maxWidth;
           attribute DOMString        minHeight;
           attribute DOMString        minWidth;
           attribute DOMString        orphans;
           attribute DOMString        outline;
           attribute DOMString        outlineColor;
           attribute DOMString        outlineStyle;
           attribute DOMString        outlineWidth;
           attribute DOMString        overflow;
           attribute DOMString        padding;
           attribute DOMString        paddingTop;
           attribute DOMString        paddingRight;
           attribute DOMString        paddingBottom;
           attribute DOMString        paddingLeft;
           attribute DOMString        page;
           attribute DOMString        pageBreakAfter;
           attribute DOMString        pageBreakBefore;
           attribute DOMString        pageBreakInside;
           attribute DOMString        pause;
           attribute DOMString        pauseAfter;
           attribute DOMString        pauseBefore;
           attribute DOMString        pitch;
           attribute DOMString        pitchRange;
           attribute DOMString        playDuring;
           attribute DOMString        position;
           attribute DOMString        quotes;
           attribute DOMString        richness;
           attribute DOMString        right;
           attribute DOMString        size;
           attribute DOMString        speak;
           attribute DOMString        speakHeader;
           attribute DOMString        speakNumeral;
           attribute DOMString        speakPunctuation;
           attribute DOMString        speechRate;
           attribute DOMString        stress;
           attribute DOMString        tableLayout;
           attribute DOMString        textAlign;
           attribute DOMString        textDecoration;
           attribute DOMString        textIndent;
           attribute DOMString        textShadow;
           attribute DOMString        textTransform;
```

```
           attribute DOMString        top;
           attribute DOMString        unicodeBidi;
           attribute DOMString        verticalAlign;
           attribute DOMString        visibility;
           attribute DOMString        voiceFamily;
           attribute DOMString        volume;
           attribute DOMString        whiteSpace;
           attribute DOMString        widows;
           attribute DOMString        width;
           attribute DOMString        wordSpacing;
           attribute DOMString        zIndex;
};
```

Attributes

`azimuth`
: See the azimuth property definition in CSS2.

`background`
: See the background property definition in CSS2.

`backgroundAttachment`
: See the background-attachment property definition in CSS2.

`backgroundColor`
: See the background-color property definition in CSS2.

`backgroundImage`
: See the background-image property definition in CSS2.

`backgroundPosition`
: See the background-position property definition in CSS2.

`backgroundRepeat`
: See the background-repeat property definition in CSS2.

`border`
: See the border property definition in CSS2.

`borderCollapse`
: See the border-collapse property definition in CSS2.

`borderColor`
: See the border-color property definition in CSS2.

`borderSpacing`
: See the border-spacing property definition in CSS2.

`borderStyle`
: See the border-style property definition in CSS2.

`borderTop`
: See the border-top property definition in CSS2.

`borderRight`
: See the border-right property definition in CSS2.

`borderBottom`
: See the border-bottom property definition in CSS2.

`borderLeft`
: See the border-left property definition in CSS2.

`borderTopColor`
: See the border-top-color property definition in CSS2.

`borderRightColor`
: See the border-right-color property definition in CSS2.

`borderBottomColor`
: See the border-bottom-color property definition in CSS2.

`borderLeftColor`
: See the border-left-color property definition in CSS2.

`borderTopStyle`
: See the border-top-style property definition in CSS2.

`borderRightStyle`
: See the border-right-style property definition in CSS2.

`borderBottomStyle`
: See the border-bottom-style property definition in CSS2.

`borderLeftStyle`
: See the border-left-style property definition in CSS2.

`borderTopWidth`
: See the border-top-width property definition in CSS2.

`borderRightWidth`
: See the border-right-width property definition in CSS2.

`borderBottomWidth`
: See the border-bottom-width property definition in CSS2.

`borderLeftWidth`
: See the border-left-width property definition in CSS2.

`borderWidth`
: See the border-width property definition in CSS2.

`bottom`
: See the bottom property definition in CSS2.

`captionSide`
: See the caption-side property definition in CSS2.

`clear`
: See the clear property definition in CSS2.

`clip`
: See the clip property definition in CSS2.

`color`
: See the color property definition in CSS2.

`content`
: See the content property definition in CSS2.

`counterIncrement`
: See the counter-increment property definition in CSS2.

`counterReset`
: See the counter-reset property definition in CSS2.

`cue`
: See the cue property definition in CSS2.

`cueAfter`
: See the cue-after property definition in CSS2.

`cueBefore`
: See the cue-before property definition in CSS2.

`cursor`
: See the cursor property definition in CSS2.

`direction`
: See the direction property definition in CSS2.

`display`
: See the display property definition in CSS2.

`elevation`
: See the elevation property definition in CSS2.

`emptyCells`
: See the empty-cells property definition in CSS2.

`cssFloat`
: See the float property definition in CSS2.

`font`
: See the font property definition in CSS2.

`fontFamily`
: See the font-family property definition in CSS2.

`fontSize`
: See the font-size property definition in CSS2.

`fontSizeAdjust`
: See the font-size-adjust property definition in CSS2.

`fontStretch`
: See the font-stretch property definition in CSS2.

`fontStyle`
: See the font-style property definition in CSS2.

`fontVariant`
: See the font-variant property definition in CSS2.

`fontWeight`
: See the font-weight property definition in CSS2.

`height`
: See the height property definition in CSS2.

`left`
: See the left property definition in CSS2.

`letterSpacing`
: See the letter-spacing property definition in CSS2.

`lineHeight`
: See the line-height property definition in CSS2.

`listStyle`
: See the list-style property definition in CSS2.

`listStyleImage`
: See the list-style-image property definition in CSS2.

`listStylePosition`
: See the list-style-position property definition in CSS2.

`listStyleType`
: See the list-style-type property definition in CSS2.

`margin`
: See the margin property definition in CSS2.

`marginTop`
: See the margin-top property definition in CSS2.

`marginRight`
: See the margin-right property definition in CSS2.

`marginBottom`
: See the margin-bottom property definition in CSS2.

`marginLeft`
: See the margin-left property definition in CSS2.

`markerOffset`
: See the marker-offset property definition in CSS2.

`marks`
: See the marks property definition in CSS2.

`maxHeight`
: See the max-height property definition in CSS2.

`maxWidth`
: See the max-width property definition in CSS2.

`minHeight`
: See the min-height property definition in CSS2.

`minWidth`
: See the min-width property definition in CSS2.

`orphans`
: See the orphans property definition in CSS2.

`outline`
: See the outline property definition in CSS2.

`outlineColor`
: See the outline-color property definition in CSS2.

`outlineStyle`
: See the outline-style property definition in CSS2.

`outlineWidth`
: See the outline-width property definition in CSS2.

`overflow`
: See the overflow property definition in CSS2.

`padding`
: See the padding property definition in CSS2.

`paddingTop`
: See the padding-top property definition in CSS2.

`paddingRight`
: See the padding-right property definition in CSS2.

`paddingBottom`
: See the padding-bottom property definition in CSS2.

`paddingLeft`
: See the padding-left property definition in CSS2.

`page`
: See the page property definition in CSS2.

`pageBreakAfter`
: See the page-break-after property definition in CSS2.

`pageBreakBefore`
: See the page-break-before property definition in CSS2.

`pageBreakInside`
: See the page-break-inside property definition in CSS2.

`pause`
: See the pause property definition in CSS2.

`pauseAfter`
: See the pause-after property definition in CSS2.

`pauseBefore`
: See the pause-before property definition in CSS2.

`pitch`
: See the pitch property definition in CSS2.

`pitchRange`
: See the pitch-range property definition in CSS2.

`playDuring`
: See the play-during property definition in CSS2.

`position`
: See the position property definition in CSS2.

`quotes`
: See the quotes property definition in CSS2.

`richness`
: See the richness property definition in CSS2.

`right`
: See the right property definition in CSS2.

`size`
: See the size property definition in CSS2.

`speak`
: See the speak property definition in CSS2.

`speakHeader`
: See the speak-header property definition in CSS2.

`speakNumeral`
: See the speak-numeral property definition in CSS2.

`speakPunctuation`
: See the speak-punctuation property definition in CSS2.

`speechRate`
: See the speech-rate property definition in CSS2.

`stress`
: See the stress property definition in CSS2.

`tableLayout`
: See the table-layout property definition in CSS2.

`textAlign`
: See the text-align property definition in CSS2.

`textDecoration`
: See the text-decoration property definition in CSS2.

`textIndent`
: See the text-indent property definition in CSS2.

`textShadow`
: See the text-shadow property definition in CSS2.

`textTransform`
: See the text-transform property definition in CSS2.

`top`
: See the top property definition in CSS2.

`unicodeBidi`
: See the unicode-bidi property definition in CSS2.

`verticalAlign`
: See the vertical-align property definition in CSS2.

`visibility`
: See the visibility property definition in CSS2.

`voiceFamily`
: See the voice-family property definition in CSS2.

`volume`
: See the volume property definition in CSS2.

`whiteSpace`
: See the white-space property definition in CSS2.

`widows`
: See the widows property definition in CSS2.

`width`
: See the width property definition in CSS2.

`wordSpacing`
: See the word-spacing property definition in CSS2.

`zIndex`
: See the z-index property definition in CSS2.

4.4. Extensions to Level 1 Interfaces

4.4.1. HTMLElement inline style

Inline style information attached to HTML elements is exposed through the `style` attribute. This represents the contents of the STYLE attribute for HTML elements.

```
interface HTMLElementStyle : HTMLElement {
readonly attribute CSSStyleDeclaration style;
};
```

4.4.2. HTMLStyleElement style sheet

The style sheet associated with an HTML STYLE element is accessible via the styleSheet attribute.

```
interface HTMLStyleElement2 : HTMLStyleElement {
readonly attribute StyleSheet styleSheet;
};
```

4.4.3. HTMLLinkElement style sheet

The styleSheet associated with an HTML LINK element with a REL of "stylesheet" or "alternate stylesheet" is not accessible directly. This is because LINK elements are not used purely as a stylesheet linking mechanism. The `styleSheet` property on LINK elements with other relationships would be incongruous.

4.5. Unresolved Issues

1. The CSS Editorial team is considering a way to represent comments that exist within a CSS style sheet. Our expectation is that absolute position of comments may not be maintained, but relative position (with respect to CSS rules and CSS properties) and the actual contents of the comment will be.
2. The CSS Editorial team is considering a mechanism to allow users to retrieve the cascaded and computed styles for a specific element.We do not intend to provide access to the actual style of specific elements in this level of the CSS DOM. Implementation of the CSS DOM does not require an actual rendering engine for any other reason, and we see that requirement as a limitation on the potential implementations of the CSS DOM.
3. The CSS Editorial team is considering a mechanism to allow users to change the cascaded style for a specific element, or to create rules in an "override" style sheet.
4. The Working Group is still considering whether it should be possible to create style sheets outside the context of a document, abstract from any XML- or HTML-specific embedding or linking of a style sheet.
5. The group is undecided whether to put a cssText attribute on the CSSStyleSheet, which would provide a textual representation of the entire style sheet. Setting this attribute would result in the resetting of all the rules in the style sheet.
6. The group intends to create a CSSException exception that derives from DOMException. This would allow a DOM user to catch CSS-specific exceptions.

5. Document Object Model Events

Editors
Tom Pixley, Netscape Communications Corporation
Chris Wilson, Microsoft Corporation

5.1. Overview of the DOM Level 2 Event Model

The DOM Level 2 Event Model is designed with two main goals. The first goal is the design of a generic event system which allows registration of event handlers, describes event flow through a tree structure, and provides basic contextual information for each event. Additionally, the specification will attempt to provide standard sets of events for user interface control and document mutation notifications, including defined contextual information for each of these event sets.

The second goal of the event model is to provide a common subset of the current event systems used within Microsoft Internet Explorer 4.0 and Netscape Navigator 4.0. This is intended to foster interoperability of existing scripts and content. It is not expected that this goal will be met with full backwards compatibility. However, the specification attempts to achieve this when possible.

5.1.1. Terminology

UI events
: User interface events. These events are generated by user interaction through an external device (mouse, keyboard, etc.)

UI Logical events
: Device independent user interface events such as focus change messages or element triggering notifications.

Mutation events
: Events caused by any action which modifies the structure of the document.

Capturing
: The process by which an event can be handled by one of the event's target's ancestors before being handled by the event's target.

Bubbling
: The process by which an event propagates upward through its ancestors after being handled by the event's target.

Cancellable
: A designation for events which indicates that upon handling the event the client may choose to prevent the DOM implementation from processing any default action associated with the event.

5.1.2. Requirements

The following constitutes the list of requirements for the DOM Level 2 Event Model.
(**ED:** Not all of the requirements below are addressed in the current version of the specification. However, all of the requirements which derive from existing event systems should currently be met.)

Requirements of event flow:

- The model must support multiple event listeners on a single `Node`.
- The model must support the ability to receive events both before and after the DOM implementation has processed the event allowing the action which triggered the event to take place.

Requirements of event listener registration:

- The model must define a programmatic mechanism of specifying event listeners. This mechanism must be rich enough to support custom events, chaining of multiple event listeners, and general event listener registration
- If additional methods of registering event listeners are defined they must be consistent with the programmatic model for event listener registration. Consistent means it is possible to define a sequence of DOM API calls which would have the same result.
- The model must define the interaction between the programmatic event registration mechanism and event listener registration within HTML tags defined in the HTML 4.0 Specification
- The programmatic method of event listener registration should allow the client to specify whether to receive the event before or after it has been processed by the DOM implementation.
- Tag based registration, style based registration, and programmatic registration must all be able to coexist together. The event model must define rules for interaction between them.

Requirements of contextual event information:

- The model must specify a mechanism for providing basic contextual information for any event.
- The model must specify a mechanism to provide UI events with additional UI specific information.

Requirements of event types:

- The model must allow the creation of additional event sets beyond those specified within the DOM Level 2 Event Model specification.
- The model must support UI events.
- The model must define a set of UI logical events to allow reaction to UI input in a device independent way. One use of this is for accessibility.
- The model must define a set of document mutation events which allow notification of any change to the document’s structure.
- The model should define a set of events to allow notification of changes to a document’s style.

5.2. Description of event flow

Event flow is the process through which the an event originates from the DOM implementation and is passed into the Document Object Model. The methods of event capture and event bubbling, along with various event listener registration techniques, allow the event to then be handled in a number of ways. It can be handled locally at the target `Node` level or centrally from a `Node` higher in the document tree.

5.2.1. Basic event flow

Each event has a `Node` toward which the event is directed by the DOM implementation. This `Node` is the event target. When the event reaches the target, any event listeners registered on the `Node` are triggered. Although all `EventListener` [p.94] s on the `Node` are guaranteed to receive the event, no specification is made as to the order in which they will receive the event with regards to the other `EventListener` [p.94] s on the `Node`. If neither event capture or event bubbling are in use for that particular event, the event flow process will complete after all listeners have been triggered. If event capture or event bubbling is in use, the event flow will be modified as described in the sections below.

5.2.2. Event Capture

Event capture is the process by which an ancestor of the event's target can register to intercept events of a given type before they are received by the event's target. Capture operates from the top of the tree downward, making it the symmetrical opposite of bubbling which is described below.

An `EventListener` [p.94] being registered on an `EventTarget` [p.93] may choose to have that `EventListener` [p.94] capture events by specifying the `useCapture` parameter of the `addEventListener` method to be true. Thereafter, when an event of the given type is dispatched toward a descendant of the capturing object, the event will trigger any capturing event listeners of the appropriate type which exist in the direct line between the top of the document and the event's target. This downward propagation continues until either no additional capturing `EventListener` [p.94] s are found or the event's target is reached.

If the capturing `EventListener` [p.94] wishes to prevent further processing of the event it may call the `preventCapture` method of the `Event` [p.95] interface. This will prevent further dispatch of the event to additional `EventTarget`s lower in the tree structure, although additional `EventListener` [p.94] s registered at the same hierarchy level will still receive the event. Only one `EventListeners` is required to call `preventCapture` to stop the propagation of the event If no additional capturers exist and `preventCapture` has not been called, the event triggers the appropriate `EventListener` [p.94] s on the target itself.

Although event capture is similar to the delegation based event model, it is different in two important respects. First, event capture only allows interception of events which are targeted at descendants of the capturing `Node`. It does not allow interception of events targeted to the capturer's ancestors, its siblings, or its sibling's descendants. Secondly, event capture is not specified for a single `Node`, it is specified for a specific type of event. Once specified, event capture intercepts all events of the specified type targeted toward any of the capturer's descendants.

5.2.3. Event bubbling

Events which are designated as bubbling will initially proceed with the same event flow as non-bubbling events. The event is dispatched to its target `Node` and any event listeners found there are triggered. Bubbling events will then trigger any additional event listeners found by following the `Node`'s parent chain upward, checking for any event listeners registered on each successive `Node`. This upward propagation will continue up to and including the `Document`.

Any event handler may choose to prevent continuation of the bubbling process by calling the `preventBubble` method of the `Event` [p.95] interface. If any `EventListener` [p.94] calls this method, all additional `EventListener` [p.94] s on the current `EventTarget` [p.93] will be triggered but bubbling will cease at that level. Only one call to `preventBubble` is required to prevent further bubbling.

5.2.4. Event cancellation

Some events are specified as cancellable. For these events, the DOM implementation generally has a default action associated with the event. Before processing these events, the implementation must check for event listeners registered to receive the event and dispatch the event to those listeners. These listeners then have the option of cancelling the implementation's default action or allowing the default action to proceed. Cancellation is accomplished by calling the `Event` [p.95] 's `preventDefault` method. If one or more `EventListener` [p.94] s call `preventDefault` during any phase of event flow the default action will be cancelled.

5.3. Event listener registration

5.3.1. Event registration interfaces

Interface *EventTarget*

The `EventTarget` interface is implemented by all `Node`s in an implementation which supports the DOM Event Model. The interface allows event listeners to be registered on the node.
IDL Definition

```
interface EventTarget {
  void               addEventListener(in DOMString type,
                                      in EventListener listener,
                                      in boolean useCapture);
  void               removeEventListener(in DOMString type,
                                         in EventListener listener,
                                         in boolean useCapture);
};
```

Methods

`addEventListener`

This method allows the registration of event listeners on the event target.
Parameters

`type`	The event type for which the user is registering
`listener`	The `listener` parameter takes an interface implemented by the user which contains the methods to be called when the event occurs.
`useCapture`	If true, `useCapture` indicates that the user wishes to initiate capture. After initiating capture, all events of the specified type will be dispatched to the registered `EventListener` [p.94] before being dispatched to any `EventTarget`s beneath them in the tree. Events which are bubbling upward through the tree will not trigger an `EventListener` [p.94] designated to use capture.

This method returns nothing.
This method raises no exceptions.

`removeEventListener`

This method allows the removal of event listeners from the event target. If an `EventListener` [p.94] is removed from an `EventTarget` while it is processing an event, it will complete its current actions but will not be triggered again during any later stages of event flow.

Parameters

`type`	Specifies the event type of the `EventListener` [p.94] being removed.
`listener`	The `EventListener` [p.94] parameter indicates the `EventListener` to be removed.
`useCapture`	Specifies whether the `EventListener` [p.94] being removed is a capturing listener or not.

This method returns nothing.
This method raises no exceptions.

Interface *EventListener*

The `EventListener` interface is the primary method for handling events. Users implement the `EventListener` interface and register their listener on a `EventTarget` [p.93] using the `AddEventListener` method. The users should also remove their `EventListener` from its `EventTarget` [p.93] after they have completed using the listener.

IDL Definition

```
interface EventListener {
  void               handleEvent(in Event event);
};
```

Methods

`handleEvent`

This method is called whenever an event occurs of the type for which the `EventListener` interface was registered.

Parameters

`event`	The `Event` [p.95] contains contextual information about the event. It also contains the `returnValue` and `cancelBubble` properties which are used in determining proper event flow.

This method returns nothing.
This method raises no exceptions.

5.3.2. Interaction with HTML 4.0 event listeners

In HTML 4.0, event listeners were specified as properties of an element. As such, registration of a second event listener of the same type would replace the first listener. The DOM Event Model allows registration of multiple event listeners on a single `Node`. To achieve this, event listeners are no longer stored as property values.

In order to achieve compatibility with HTML 4.0, implementors may view the setting of properties which represent event handlers as the creation and registration of an `EventListener` on the `Node`. The value of `useCapture` defaults to false. This `EventListener` [p.94] behaves in the same manner as any other `EventListeners`s which may be registered on the `EventTarget` [p.93] . If the property representing the event listener is changed, this may be viewed as the removal of the previously registered `EventListener` [p.94] and the registration of a new one. No technique is provided to allow HTML 4.0 event listeners access to the context information defined for each event.

5.3.3. Event listener registration issues

The specification currently defines listeners as generic listeners which can be registered for multiple types of events. This solution avails itself readily to extending or creating new events. However, registering the same object for multiple events requires the user to differentiate between the events inside the event listener. The current string based event typing system could make this very inefficient. The DOM Working Group is exploring alternatives to the string based event typing to resolve this issue.

A full solution has not yet been added to meet the suggestion that all listeners be notified of the final resolution of an event. It is possible that use of both pre- and post-processed types of events will achieve this goal but it is not yet clear if this solution will be sufficient.

5.4. Event interfaces

Interface *Event*

The `Event` interface is used to provide contextual information about an event to the handler processing the event. An object which implements the `Event` interface is generally passed as the first parameter to an event handler. More specific context information is passed to event handlers by deriving additional interfaces from `Event` which contain information directly relating to the type of event they accompany. These derived interfaces are also implemented by the object passed to the event listener.

IDL Definition

```
interface Event {
  // PhaseType
  const unsigned short      BUBBLING_PHASE                 = 1;
  const unsigned short      CAPTURING_PHASE                = 2;
  const unsigned short      AT_TARGET                      = 3;

           attribute DOMString        type;
           attribute Node             target;
           attribute Node             currentNode;
           attribute unsigned short   eventPhase;
  void               preventBubble();
  void               preventCapture();
  void               preventDefault();
};
```

Definition group *PhaseType*

An integer indicating which phase of event flow is being processed.
Defined Constants

BUBBLING_PHASE	The current event phase is the bubbling phase.
CAPTURING_PHASE	The current event phase is the capturing phase.
AT_TARGET	The event is currently being evaluated at the target node.

Attributes

`type`
The `type` property represents the event name as a string property.

`target`
The `target` property indicates the `Node` to which the event was originally dispatched.

`currentNode`
The `currentNode` property indicates to which `Node` the event is currently being dispatched. This is particularly useful during capturing and bubbling.

`eventPhase`
The `eventPhase` property indicates which phase of event flow is currently being evaluated.

Methods

`preventBubble`
The `preventBubble` method is used to end the bubbling phase of event flow. If this method is called by any `EventListener` [p.94] s registered on the same `EventTarget` [p.93] during bubbling, the bubbling phase will cease at that level and the event will not be propagated upward within the tree.
This method has no parameters.
This method returns nothing.
This method raises no exceptions.

`preventCapture`
The `preventCapture` method is used to end the capturing phase of event flow. If this method is called by any `EventListener` [p.94] s registered on the same `EventTarget` [p.93] during capturing, the capturing phase will cease at that level and the event will not be propagated any further down.
This method has no parameters.
This method returns nothing.
This method raises no exceptions.

`preventDefault`
If an event is cancellable, the `preventCapture` method is used to signify that the event is to be cancelled. If, during any stage of event flow, the `preventDefault` method is

called the event is cancelled. Any default action associated with the event will not occur. Calling this method for a non-cancellable event has no effect.
This method has no parameters.
This method returns nothing.
This method raises no exceptions.

Interface *UIEvent*

The `UIEvent` interface provides specific contextual information associated with User Interface and Logical events.
(**ED:** The values for the keyCode constants are yet to be determined.)
IDL Definition

```
interface UIEvent : Event {
  const int                 CHAR_UNDEFINED                 = 1;
  const int                 KEY_FIRST                      = 1;
  const int                 KEY_LAST                       = 1;
  const int                 VK_0                           = 1;
  const int                 VK_1                           = 1;
  const int                 VK_2                           = 1;
  const int                 VK_3                           = 1;
  const int                 VK_4                           = 1;
  const int                 VK_5                           = 1;
  const int                 VK_6                           = 1;
  const int                 VK_7                           = 1;
  const int                 VK_8                           = 1;
  const int                 VK_9                           = 1;
  const int                 VK_A                           = 1;
  const int                 VK_ACCEPT                      = 1;
  const int                 VK_ADD                         = 1;
  const int                 VK_AGAIN                       = 1;
  const int                 VK_ALL_CANDIDATES              = 1;
  const int                 VK_ALPHANUMERIC                = 1;
  const int                 VK_ALT                         = 1;
  const int                 VK_ALT_GRAPH                   = 1;
  const int                 VK_AMPERSAND                   = 1;
  const int                 VK_ASTERISK                    = 1;
  const int                 VK_AT                          = 1;
  const int                 VK_B                           = 1;
  const int                 VK_BACK_QUOTE                  = 1;
  const int                 VK_BACK_SLASH                  = 1;
  const int                 VK_BACK_SPACE                  = 1;
  const int                 VK_BRACELEFT                   = 1;
  const int                 VK_BRACERIGHT                  = 1;
  const int                 VK_C                           = 1;
  const int                 VK_CANCEL                      = 1;
  const int                 VK_CAPS_LOCK                   = 1;
  const int                 VK_CIRCUMFLEX                  = 1;
  const int                 VK_CLEAR                       = 1;
  const int                 VK_CLOSE_BRACKET               = 1;
  const int                 VK_CODE_INPUT                  = 1;
  const int                 VK_COLON                       = 1;
  const int                 VK_COMMA                       = 1;
  const int                 VK_COMPOSE                     = 1;
  const int                 VK_CONTROL                     = 1;
  const int                 VK_CONVERT                     = 1;
  const int                 VK_COPY                        = 1;
  const int                 VK_CUT                         = 1;
  const int                 VK_D                           = 1;
  const int                 VK_DEAD_ABOVEDOT               = 1;
  const int                 VK_DEAD_ABOVERING              = 1;
  const int                 VK_DEAD_ACUTE                  = 1;
  const int                 VK_DEAD_BREVE                  = 1;
  const int                 VK_DEAD_CARON                  = 1;
  const int                 VK_DEAD_CEDILLA                = 1;
  const int                 VK_DEAD_CIRCUMFLEX             = 1;
  const int                 VK_DEAD_DIAERESIS              = 1;
  const int                 VK_DEAD_DOUBLEACUTE            = 1;
  const int                 VK_DEAD_GRAVE                  = 1;
  const int                 VK_DEAD_IOTA                   = 1;
  const int                 VK_DEAD_MACRON                 = 1;
  const int                 VK_DEAD_OGONEK                 = 1;
  const int                 VK_DEAD_SEMIVOICED_SOUND       = 1;
  const int                 VK_DEAD_TILDE                  = 1;
  const int                 VK_DEAD_VOICED_SOUND           = 1;
  const int                 VK_DECIMAL                     = 1;
  const int                 VK_DELETE                      = 1;
  const int                 VK_DIVIDE                      = 1;
  const int                 VK_DOLLAR                      = 1;
  const int                 VK_DOWN                        = 1;
  const int                 VK_E                           = 1;
  const int                 VK_END                         = 1;
  const int                 VK_ENTER                       = 1;
  const int                 VK_EQUALS                      = 1;
  const int                 VK_ESCAPE                      = 1;
  const int                 VK_EURO_SIGN                   = 1;
  const int                 VK_EXCLAMATION_MARK            = 1;
  const int                 VK_F                           = 1;
  const int                 VK_F1                          = 1;
  const int                 VK_F10                         = 1;
  const int                 VK_F11                         = 1;
  const int                 VK_F12                         = 1;
  const int                 VK_F13                         = 1;
  const int                 VK_F14                         = 1;
  const int                 VK_F15                         = 1;
  const int                 VK_F16                         = 1;
  const int                 VK_F17                         = 1;
  const int                 VK_F18                         = 1;
  const int                 VK_F19                         = 1;
  const int                 VK_F2                          = 1;
  const int                 VK_F20                         = 1;
  const int                 VK_F21                         = 1;
  const int                 VK_F22                         = 1;
  const int                 VK_F23                         = 1;
  const int                 VK_F24                         = 1;
  const int                 VK_F3                          = 1;
  const int                 VK_F4                          = 1;
  const int                 VK_F5                          = 1;
  const int                 VK_F6                          = 1;
  const int                 VK_F7                          = 1;
  const int                 VK_F8                          = 1;
  const int                 VK_F9                          = 1;
  const int                 VK_FINAL                       = 1;
  const int                 VK_FIND                        = 1;
  const int                 VK_FULL_WIDTH                  = 1;
  const int                 VK_G                           = 1;
  const int                 VK_GREATER                     = 1;
  const int                 VK_H                           = 1;
  const int                 VK_HALF_WIDTH                  = 1;
  const int                 VK_HELP                        = 1;
  const int                 VK_HIRAGANA                    = 1;
  const int                 VK_HOME                        = 1;
  const int                 VK_I                           = 1;
  const int                 VK_INSERT                      = 1;
  const int                 VK_INVERTED_EXCLAMATION_MARK   = 1;
  const int                 VK_J                           = 1;
  const int                 VK_JAPANESE_HIRAGANA           = 1;
  const int                 VK_JAPANESE_KATAKANA           = 1;
  const int                 VK_JAPANESE_ROMAN              = 1;
  const int                 VK_K                           = 1;
  const int                 VK_KANA                        = 1;
  const int                 VK_KANJI                       = 1;
  const int                 VK_KATAKANA                    = 1;
  const int                 VK_KP_DOWN                     = 1;
  const int                 VK_KP_LEFT                     = 1;
  const int                 VK_KP_RIGHT                    = 1;
  const int                 VK_KP_UP                       = 1;
  const int                 VK_L                           = 1;
  const int                 VK_LEFT                        = 1;
  const int                 VK_LEFT_PARENTHESIS            = 1;
  const int                 VK_LESS                        = 1;
  const int                 VK_M                           = 1;
  const int                 VK_META                        = 1;
  const int                 VK_MINUS                       = 1;
  const int                 VK_MODECHANGE                  = 1;
  const int                 VK_MULTIPLY                    = 1;
  const int                 VK_N                           = 1;
  const int                 VK_NONCONVERT                  = 1;
  const int                 VK_NUM_LOCK                    = 1;
  const int                 VK_NUMBER_SIGN                 = 1;
  const int                 VK_NUMPAD0                     = 1;
  const int                 VK_NUMPAD1                     = 1;
  const int                 VK_NUMPAD2                     = 1;
  const int                 VK_NUMPAD3                     = 1;
  const int                 VK_NUMPAD4                     = 1;
  const int                 VK_NUMPAD5                     = 1;
  const int                 VK_NUMPAD6                     = 1;
  const int                 VK_NUMPAD7                     = 1;
  const int                 VK_NUMPAD8                     = 1;
  const int                 VK_NUMPAD9                     = 1;
  const int                 VK_O                           = 1;
  const int                 VK_OPEN_BRACKET                = 1;
  const int                 VK_P                           = 1;
  const int                 VK_PAGE_DOWN                   = 1;
  const int                 VK_PAGE_UP                     = 1;
  const int                 VK_PASTE                       = 1;
  const int                 VK_PAUSE                       = 1;
  const int                 VK_PERIOD                      = 1;
  const int                 VK_PLUS                        = 1;
```

```
  const int                VK_PREVIOUS_CANDIDATE          = 1;
  const int                VK_PRINTSCREEN                 = 1;
  const int                VK_PROPS                       = 1;
  const int                VK_Q                           = 1;
  const int                VK_QUOTE                       = 1;
  const int                VK_QUOTEDBL                    = 1;
  const int                VK_R                           = 1;
  const int                VK_RIGHT                       = 1;
  const int                VK_RIGHT_PARENTHESIS           = 1;
  const int                VK_ROMAN_CHARACTERS            = 1;
  const int                VK_S                           = 1;
  const int                VK_SCROLL_LOCK                 = 1;
  const int                VK_SEMICOLON                   = 1;
  const int                VK_SEPARATER                   = 1;
  const int                VK_SHIFT                       = 1;
  const int                VK_SLASH                       = 1;
  const int                VK_SPACE                       = 1;
  const int                VK_STOP                        = 1;
  const int                VK_SUBTRACT                    = 1;
  const int                VK_T                           = 1;
  const int                VK_TAB                         = 1;
  const int                VK_U                           = 1;
  const int                VK_UNDEFINED                   = 1;
  const int                VK_UNDERSCORE                  = 1;
  const int                VK_UNDO                        = 1;
  const int                VK_UP                          = 1;
  const int                VK_V                           = 1;
  const int                VK_W                           = 1;
  const int                VK_X                           = 1;
  const int                VK_Y                           = 1;
  const int                VK_Z                           = 1;
           attribute long            screenX;
           attribute long            screenY;
           attribute long            clientX;
           attribute long            clientY;
           attribute boolean         ctrlKey;
           attribute boolean         shiftKey;
           attribute boolean         altKey;
           attribute boolean         metaKey;
           attribute unsigned long   keyCode;
           attribute unsigned long   charCode;
           attribute unsigned short  button;
           attribute unsigned short  clickCount;
};
```

Constant *CHAR_UNDEFINED*

KEY_PRESSED and KEY_RELEASED events which do not map to a valid Unicode character use this for the keyChar value.

Constant *KEY_FIRST*

The first number in the range of ids used for key events.

Constant *KEY_LAST*

The last number in the range of ids used for key events.

Constant *VK_0*

VK_0 thru VK_9 are the same as ASCII ’0’ thru ’9’ (0x30 - 0x39)

Constant *VK_1*
Constant *VK_2*
Constant *VK_3*
Constant *VK_4*
Constant *VK_5*
Constant *VK_6*
Constant *VK_7*
Constant *VK_8*
Constant *VK_9*
Constant *VK_A*

VK_A thru VK_Z are the same as ASCII ’A’ thru ’Z’ (0x41 - 0x5A)

Constant *VK_ACCEPT*
Constant *VK_ADD*
Constant *VK_AGAIN*
Constant *VK_ALL_CANDIDATES*

Constant for the All Candidates function key.

Constant *VK_ALPHANUMERIC*

Constant for the Alphanumeric function key.

Constant *VK_ALT*
Constant *VK_ALT_GRAPH*

Constant for the AltGraph modifier key.

Constant *VK_AMPERSAND*
Constant *VK_ASTERISK*
Constant *VK_AT*

Constant for the "@" key.

Constant *VK_B*
Constant *VK_BACK_QUOTE*
Constant *VK_BACK_SLASH*
Constant *VK_BACK_SPACE*
Constant *VK_BRACELEFT*
Constant *VK_BRACERIGHT*
Constant *VK_C*
Constant *VK_CANCEL*
Constant *VK_CAPS_LOCK*
Constant *VK_CIRCUMFLEX*

Constant for the "^" key.

Constant *VK_CLEAR*

Constant *VK_CLOSE_BRACKET*

Constant *VK_CODE_INPUT*

Constant for the Code Input function key.

Constant *VK_COLON*

Constant for the ":" key.

Constant *VK_COMMA*

Constant *VK_COMPOSE*

Constant for the Compose function key.

Constant *VK_CONTROL*

Constant *VK_CONVERT*

Constant *VK_COPY*

Constant *VK_CUT*

Constant *VK_D*

Constant *VK_DEAD_ABOVEDOT*

Constant *VK_DEAD_ABOVERING*

Constant *VK_DEAD_ACUTE*

Constant *VK_DEAD_BREVE*

Constant *VK_DEAD_CARON*

Constant *VK_DEAD_CEDILLA*

Constant *VK_DEAD_CIRCUMFLEX*

Constant *VK_DEAD_DIAERESIS*

Constant *VK_DEAD_DOUBLEACUTE*

Constant *VK_DEAD_GRAVE*

Constant *VK_DEAD_IOTA*

Constant *VK_DEAD_MACRON*

Constant *VK_DEAD_OGONEK*

Constant *VK_DEAD_SEMIVOICED_SOUND*

Constant *VK_DEAD_TILDE*

Constant *VK_DEAD_VOICED_SOUND*

Constant *VK_DECIMAL*

Constant *VK_DELETE*

Constant *VK_DIVIDE*

Constant *VK_DOLLAR*

Constant for the "$" key.

Constant *VK_DOWN*

Constant *VK_E*

Constant *VK_END*

Constant *VK_ENTER*

Constant *VK_EQUALS*

Constant *VK_ESCAPE*
Constant *VK_EURO_SIGN*

Constant for the Euro currency sign key.
Constant *VK_EXCLAMATION_MARK*

Constant for the "!" key.
Constant *VK_F*
Constant *VK_F1*

Constant for the F1 function key.
Constant *VK_F10*

Constant for the F10 function key.
Constant *VK_F11*

Constant for the F11 function key.
Constant *VK_F12*

Constant for the F12 function key.
Constant *VK_F13*

Constant for the F13 function key.
Constant *VK_F14*

Constant for the F14 function key.
Constant *VK_F15*

Constant for the F15 function key.
Constant *VK_F16*

Constant for the F16 function key.
Constant *VK_F17*

Constant for the F17 function key.
Constant *VK_F18*

Constant for the F18 function key.
Constant *VK_F19*

Constant for the F19 function key.
Constant *VK_F2*

Constant for the F2 function key.
Constant *VK_F20*

Constant for the F20 function key.

Constant *VK_F21*

Constant for the F21 function key.

Constant *VK_F22*

Constant for the F22 function key.

Constant *VK_F23*

Constant for the F23 function key.

Constant *VK_F24*

Constant for the F24 function key.

Constant *VK_F3*

Constant for the F3 function key.

Constant *VK_F4*

Constant for the F4 function key.

Constant *VK_F5*

Constant for the F5 function key.

Constant *VK_F6*

Constant for the F6 function key.

Constant *VK_F7*

Constant for the F7 function key.

Constant *VK_F8*

Constant for the F8 function key.

Constant *VK_F9*

Constant for the F9 function key.

Constant *VK_FINAL*

Constant *VK_FIND*

Constant *VK_FULL_WIDTH*

Constant for the Full-Width Characters function key.

Constant *VK_G*

Constant *VK_GREATER*

Constant *VK_H*

Constant *VK_HALF_WIDTH*

Constant for the Half-Width Characters function key.

Constant *VK_HELP*

Constant *VK_HIRAGANA*

Constant for the Hiragana function key.

Constant *VK_HOME*
Constant *VK_I*
Constant *VK_INSERT*
Constant *VK_INVERTED_EXCLAMATION_MARK*

Constant for the inverted exclamation mark key.

Constant *VK_J*
Constant *VK_JAPANESE_HIRAGANA*

Constant for the Japanese-Hiragana function key.

Constant *VK_JAPANESE_KATAKANA*

Constant for the Japanese-Katakana function key.

Constant *VK_JAPANESE_ROMAN*

Constant for the Japanese-Roman function key.

Constant *VK_K*
Constant *VK_KANA*
Constant *VK_KANJI*
Constant *VK_KATAKANA*

Constant for the Katakana function key.

Constant *VK_KP_DOWN*

for KeyPad cursor arrow keys

Constant *VK_KP_LEFT*

for KeyPad cursor arrow keys

Constant *VK_KP_RIGHT*

for KeyPad cursor arrow keys

Constant *VK_KP_UP*

for KeyPad cursor arrow keys

Constant *VK_L*
Constant *VK_LEFT*
Constant *VK_LEFT_PARENTHESIS*

Constant for the "(" key.

Constant *VK_LESS*
Constant *VK_M*
Constant *VK_META*
Constant *VK_MINUS*
Constant *VK_MODECHANGE*
Constant *VK_MULTIPLY*

Constant *VK_N*
Constant *VK_NONCONVERT*
Constant *VK_NUM_LOCK*
Constant *VK_NUMBER_SIGN*

Constant for the "#" key.
Constant *VK_NUMPAD0*
Constant *VK_NUMPAD1*
Constant *VK_NUMPAD2*
Constant *VK_NUMPAD3*
Constant *VK_NUMPAD4*
Constant *VK_NUMPAD5*
Constant *VK_NUMPAD6*
Constant *VK_NUMPAD7*
Constant *VK_NUMPAD8*
Constant *VK_NUMPAD9*
Constant *VK_O*
Constant *VK_OPEN_BRACKET*
Constant *VK_P*
Constant *VK_PAGE_DOWN*
Constant *VK_PAGE_UP*
Constant *VK_PASTE*
Constant *VK_PAUSE*
Constant *VK_PERIOD*
Constant *VK_PLUS*

Constant for the "+" key.
Constant *VK_PREVIOUS_CANDIDATE*

Constant for the Previous Candidate function key.
Constant *VK_PRINTSCREEN*
Constant *VK_PROPS*
Constant *VK_Q*
Constant *VK_QUOTE*
Constant *VK_QUOTEDBL*
Constant *VK_R*
Constant *VK_RIGHT*
Constant *VK_RIGHT_PARENTHESIS*

Constant for the ")" key.
Constant *VK_ROMAN_CHARACTERS*

Constant for the Roman Characters function key.
Constant *VK_S*
Constant *VK_SCROLL_LOCK*

Constant *VK_SEMICOLON*
Constant *VK_SEPARATER*
Constant *VK_SHIFT*
Constant *VK_SLASH*
Constant *VK_SPACE*
Constant *VK_STOP*
Constant *VK_SUBTRACT*
Constant *VK_T*
Constant *VK_TAB*
Constant *VK_U*
Constant *VK_UNDEFINED*

KEY_TYPED events do not have a keyCode value.
Constant *VK_UNDERSCORE*

Constant for the "_" key.
Constant *VK_UNDO*
Constant *VK_UP*
Constant *VK_V*
Constant *VK_W*
Constant *VK_X*
Constant *VK_Y*
Constant *VK_Z*
Attributes

`screenX`
: `screenX` indicates the horizontal coordinate at which the event occurred in relative to the origin of the screen coordinate system.

`screenY`
: `screenY` indicates the vertical coordinate at which the event occurred relative to the origin of the screen coordinate system.

`clientX`
: `clientX` indicates the horizontal coordinate at which the event occurred relative to the DOM implementation’s client area.

`clientY`
: `clientY` indicates the vertical coordinate at which the event occurred relative to the DOM implementation’s client area.

`ctrlKey`
: `ctrlKey` indicates whether the ’ctrl’ key was depressed during the firing of the event.

`shiftKey`
: `shiftKey` indicates whether the ’shift’ key was depressed during the firing of the event.

`altKey`
: `altKey` indicates whether the ’alt’ key was depressed during the firing of the event. On some platforms this key may map to an alternative key name.

`metaKey`
: `metaKey` indicates whether the ’meta’ key was depressed during the firing of the event. On some platforms this key may map to an alternative key name.

`keyCode`
: The value of `keyCode` holds the virtual key code value of the key which was depressed if the event is a key event. Otherwise, the value is zero.

`charCode`
: `charCode` holds the value of the Unicode character associated with the depressed key if the event is a key event. Otherwise, the value is zero.

`button`
: During mouse events caused by the depression or release of a mouse button, `button` is used to indicate which mouse button changed state.

`clickCount`
: The `clickCount` property indicates the number of times a mouse button has been pressed and released over the same screen location during a user action. The property value is 1 when the user begins this action and increments by 1 for each full sequence of pressing and releasing. If the user moves the mouse between the mousedown and mouseup the value will be set to 0, indicating that no click is occurring.

Interface *MutationEvent*

The `MutationEvent` interface provides specific contextual information associated with Mutation events.

IDL Definition

```
interface MutationEvent : Event {
           attribute Node             relatedNode;
           attribute DOMString        prevValue;
           attribute DOMString        newValue;
           attribute DOMString        attrName;
};
```

Attributes

`relatedNode`
: `relatedNode` is used to identify a secondary node related to a mutation event. For example, if a mutation event is dispatched to a node indicating that its parent has changed, the `relatedNode` is the changed parent. If an event is instead dispatch to a subtree indicating a node was changed within it, the `relatedNode` is the changed node.

`prevValue`
: `prevValue` indicates the previous value of text nodes and attributes in attrModified and charDataModified events.

`newValue`
: `newValue` indicates the new value of text nodes and attributes in attrModified and charDataModified events.

`attrName`
: `attrName` indicates the changed attr in the attrModified event.

5.5. Event set definitions

The DOM Level 2 Event Model allows a DOM implementation to support multiple sets of events. The model has been designed to allow addition of new event sets as is required. The DOM will not attempt to define all possible events. For purposes of interoperability, the DOM will define a set of user interface events, a set of UI logical events, and a set of document mutation events.

5.5.1. User Interface event types

The User Interface event set is composed of events listed in HTML 4.0 and additional events which are supported in both Netscape Navigator 4.0 and Microsoft Internet Explorer 4.0.

click

The click event occurs when the pointing device button is clicked over an element. A click is defined as a mousedown and mouseup over the same screen location. The sequence of these events is:

```
mousedown
mouseup
click
```

If multiple clicks occur at the same screen location, the sequence repeats with the `clickCount` property incrementing with each repetition. This event is valid for most elements.

- Bubbles: Yes
- Cancellable: Yes
- Context Info: screenX, screenY, clientX, clientY, altKey, ctrlKey, shiftKey, button

mousedown

The mousedown event occurs when the pointing device button is pressed over an element. This event is valid for most elements.

- Bubbles: Yes
- Cancellable: Yes
- Context Info: screenX, screenY, clientX, clientY, altKey, ctrlKey, shiftKey, button

mouseup

The mouseup event occurs when the pointing device button is released over an element. This event is valid for most elements.

- Bubbles: Yes
- Cancellable: Yes
- Context Info: screenX, screenY, clientX, clientY, altKey, ctrlKey, shiftKey, button

mouseover

The mouseover event occurs when the pointing device is moved onto an element. This event is valid for most elements.

- Bubbles: Yes
- Cancellable: Yes
- Context Info: screenX, screenY, clientX, clientY, altKey, ctrlKey, shiftKey

mousemove

The mousemove event occurs when the pointing device is moved while it is over an element. This event is valid for most elements.

- Bubbles: Yes
- Cancellable: No
- Context Info: screenX, screenY, clientX, clientY, altKey, ctrlKey, shiftKey

mouseout

The mouseout event occurs when the pointing device is moved away from an element. This event is valid for most elements..

- Bubbles: Yes
- Cancellable: Yes
- Context Info: screenX, screenY, clientX, clientY, altKey, ctrlKey, shiftKey

keypress

The keypress event occurs when a key is pressed. If the key remains depressed, multiple keypresses may be generated. This event maps not to the physical depression of the key but is instead the result of that action, often being the insertion of a character.

- Bubbles: Yes
- Cancellable: Yes
- Context Info: keyCode, charCode

keydown

The keydown event occurs when a key is pressed down.

- Bubbles: Yes
- Cancellable: Yes
- Context Info: keyCode, charCode

keyup

The keyup event occurs when a key is released.

- Bubbles: Yes
- Cancellable: Yes
- Context Info: keyCode, charCode

resize

The resize event occurs when a document is resized.

- Bubbles: Yes
- Cancellable: No
- Context Info: None

scroll

The scroll event occurs when a document is scrolled.

- Bubbles: Yes
- Cancellable: No
- Context Info: None

5.5.2. Mutation event types

The mutation event set is designed to allow notification of any changes to the structure of a document, including attr and text modifications. It may be noted that none of the mutation events listed are designated as cancellable. The reasoning for this stems from the fact that it would be very difficult to make use of existing DOM interfaces which cause document modifications if any change to the document might or might not take place due to cancellation of the related event. Although this is still a desired capability, it was decided that it would be better left until the addition of transactions into the DOM.

subtreeModified

This is a general event for notification of all changes to the document. It can be used instead of the more specific events listed below. Also, the requirement for some sort of batching of mutation events may be accomplished through this event. The target of this event is the lowest common parent of the changes which have taken place. This event is dispatched after any other events caused by the mutation have fired.

- Bubbles: Yes
- Cancellable: No
- Context Info: None

nodeInserted

Fired when a node has been added as a child of another node. This event is dispatched after the insertion has taken place. The target of this event is the node being inserted.

- Bubbles: Yes
- Cancellable: No
- Context Info: relatedNode holds the parent node

nodeRemoved

Fired when a node is being removed from another node. This event is dispatched before the node is removed from the tree. The target of this event is the node being removed.

- Bubbles: Yes
- Cancellable: No
- Context Info: relatedNode holds the parent node

nodeRemovedFromDocument

Fired when a node is being removed from a document, either through direct removal of the Node or removal of a subtree in which it is contained. This event is dispatched before the removal takes place. The target of this event is the node being removed. If the Node is being directly removed the nodeRemoved event will fire before the nodeRemovedFromDocument event.

- Bubbles: No
- Cancellable: No
- Context Info: None

nodeInsertedIntoDocument

Fired when a node is being inserted into a document, either through direct insertion of the Node or insertion of a subtree in which it is contained. This event is dispatched after the insertion has taken place. The target of this event is the node being inserted. If the Node is being directly inserted the nodeInserted event will fire before the nodeInsertedIntoDocument event.

- Bubbles: No
- Cancellable: No
- Context Info: None

attrModified

Fired after an `Attr` has been modified on a node. The target of this event is the node whose `Attr` changed.

- Bubbles: Yes
- Cancellable: No
- Context Info: attrName, prevValue, newValue

characterDataModified

Fired after CharacterData within a node has been modified but the node itself has not been inserted or deleted. The target of this event is the CharacterData node.

- Bubbles: Yes
- Cancellable: No
- Context Info: prevValue, newValue

5.5.3. HTML event types

The HTML event set is composed of events listed in HTML 4.0 and additional events which are supported in both Netscape Navigator 4.0 and Microsoft Internet Explorer 4.0.

load

The load event occurs when the DOM implementation finishes loading all content within a document, all frames within a FRAMESET, or an image.

- Bubbles: No
- Cancellable: No
- Context Info: None

unload

The unload event occurs when the DOM implementation removes a document from a window or frame. This event is valid for BODY and FRAMESET elements.

- Bubbles: No
- Cancellable: No
- Context Info: None

abort

The abort event occurs when page loading is stopped before an image has been allowed to completely load. This event applies to IMG elements.

- Bubbles: Yes
- Cancellable: No
- Context Info: None

error

The error event occurs when an image does not load properly or when an error occurs during script execution. This event is valid for IMG elements, BODY elements, and FRAMESET element.

- Bubbles: Yes
- Cancellable: No
- Context Info: None

select

The select event occurs when a user selects some text in a text field. This event is valid for INPUT and TEXTAREA elements.

- Bubbles: Yes
- Cancellable: No
- Context Info: None

change

The change event occurs when a control loses the input focus and its value has been modified since gaining focus. This event is valid for INPUT, SELECT, and TEXTAREA. element.

- Bubbles: Yes
- Cancellable: No
- Context Info: None

submit

The submit event occurs when a form is submitted. This event only applies to the FORM element.

- Bubbles: Yes
- Cancellable: Yes
- Context Info: None

reset

The reset event occurs when a form is reset. This event only applies to the FORM element.

- Bubbles: Yes
- Cancellable: No
- Context Info: None

focus

The focus event occurs when an element receives focus either via a pointing device or by tabbing navigation. This event is valid for the following elements: LABEL, INPUT, SELECT, TEXTAREA, and BUTTON.

- Bubbles: No
- Cancellable: No
- Context Info: None

blur

The blur event occurs when an element loses focus either by the pointing device or by tabbing navigation. This event is valid for the following elements: LABEL, INPUT, SELECT, TEXTAREA, and BUTTON.

- Bubbles: No
- Cancellable: No
- Context Info: None

5.5.3. HTML event types

6. Document Object Model Filters and Iterators

Editors

Mike Champion, Software AG
Joe Kesselman, IBM
Jonathan Robie, Software AG

6.1. Overview of the DOM Level 2 Iterator, Filter, and TreeWalker Interfaces

The DOM Level 2 Iterator, Filter, and TreeWalker interfaces extend the functionality of the DOM to allow simple and efficient traversal of document subtrees, node lists, or the results of queries.

This proposal contains Iterator, Filter, and TreeWalker interfaces, but no query interfaces. In the future, it is likely that a separate specification will be prepared for query interfaces, which may be query-language independent.

6.1.1. Iterators

An iterator allows the nodes of a data structure to be returned sequentially. When an iterator is first created, calling nextNode() returns the first node. When no more nodes are present, nextNode() returns a null. Since DOM structures may change as a document is loaded, if nextNode() finds no more nodes, it is still quite possible that further nodes may be added in the next instant. An iterator may be active while the data structure it navigates is being edited, so an iterator must behave gracefully in the face of change. Additions and deletions in the underlying data structure do not invalidate an iterator; in fact, an iterator is never invalidated.

Using ordered set semantics, the position of the iterator is determined by the relative position in the ordered set. There is no current node. When an iterator is created for a list, the position is set before the first element:

```
 ->A B C D E F G H I
```

Each call to next() returns a node and advances the position. For instance, if we start with the above position, the first call to next() returns "A" and advances the iterator:

```
 A ->B C D E F G H I
```

The relative position of the iterator remains valid when other nodes are added or deleted. For instance, if "A" is deleted, the position of the iterator is unchanged with respect to the remaining nodes:

```
->B C D E F G H I
```

Similarly, if the node ahead of the iterator is deleted or moved, the iterator "slides forward". Therefore, if "B" is deleted, the position remains unchanged with respect to the remaining nodes:

```
->C D E F G H I
```

For the same reason, moving the "C" node to the end of the set does not change the current position with respect to the remaining nodes:

```
->D E F G H I C
```

When nodes are added as children of the node to the left of the iterator, there is some difference of opinion as to what constitutes the most consistent behavior. Suppose the iterator is advanced past "D" using next(), then a new node is added as a child of "D" in the original tree. Since children of a node occur after the node in document order, one approach is to have the new child appear after the node, but before the current position of the iterator:

```
D a ->E F G H I C
```

The newly inserted node "a" occurs before the iterator, so it will not be encountered when the iterator is moved forward. This is convenient when an iterator is being used to add nodes to the tree, since the programmer does not need to skip over the newly inserted nodes. In this case, if the iterator were moved backward, this new node would be the first one encountered. A second consistent approach is to say that nodes added as children of the node to the left of the iterator appear after the current position of the iterator:

```
D ->a E F G H I C
```

Using this approach, the newly added nodes are encountered as the iterator moves forward. We believe either approach is justifiable, and we have not decided which of the two approaches is best.

Note that the relative position of the iterator is not the same as the absolute position within the set. The position of the iterator is relative to the node before it and the node after it, which is why the position floats gracefully when nodes are deleted or inserted before or after the position of the iterator.

Iterators are created using the createNodeIterator method found in Document. When an iterator is created, flags can be used to determine which nodes will be "visible" and which nodes will be "invisible" while traversing the tree. Nodes that are "invisible" are skipped over by the iterator as though they did not exist. These flags can be combined using OR:

```
NodeIterator iter=document.createNodeIterator(root, SHOW_ELEMENT | SHOW_PROCESSING_INSTRUCTION | SHOW_COMMENT | SHOW_ENTITY_REFERENCE, NULL);
```

6.1.2. Filters

Filters allow the user to "filter out" nodes. Each filter contains a user-written function that looks at a node and determines whether or not it should be filtered out. To use a filter, you create an iterator that uses the filter. The iterator applies the filter to each node, and if the filter rejects the node, the iterator skips over the node as though it were not present in the document. Filters are easy to write, since they need not know how to navigate the structure on which they operate, and they can be reused for different kinds of iterators that operate on different data structures.

Consider a filter that finds the named anchors in an HTML document. In HTML, an HREF can refer to any <A> element that has a NAME attribute. Here is a filter that looks at a node and determines whether it is a named anchor:

```
class NamedAnchorFilter implements NodeFilter
{
    short acceptNode(Node n) {
        if (n instanceof Element) {
            Element e = n;
            if (e.getNodeName() != "A")
```

```
                return FILTER_SKIP;
            if (e.getNodeNameAttributeNode("NAME") != NULL) {
               return FILTER_ACCEPT;
            }
        }
        return FILTER_SKIP;
    }
}
```

To use this filter, the user would create an instance of the filter and create an iterator using it:

```
NamedAnchorFilter myFilter;
NodeIterator iter=document.creatNodeIterator(node, SHOW_ELEMENT, myFilter);
```

If SHOW_ENTITY_REFERENCE is not set, entities are expanded. If SHOW_ENTITY_REFERENCE is set, entity references will be encountered by the iterator. There is no setting that shows both the entity reference and its expansion.

6.1.3. TreeWalker

The TreeWalker interface provides many of the same benefits as the Iterator interface. The main difference between these two interfaces is that the TreeWalker presents a tree-oriented view of the nodes in a subtree, and an Iterator presents a list-oriented view. In other words, an Iterator allows you to move forward or back, but a TreeWalker allows you to move to the parent of a node, to one of its children, or to a sibling.

Using a TreeWalker is quite similar to navigation using the Node directly, and the navigation methods for the two interfaces are analogous. For instance, here is a function that processes the nodes of a subtree in document order using the Node navigation methods:

```
processMe(Node n) {

   doSomething(n);

   if (n.firstChild != null) {
     processMe(n.firstChild);
   }

   if (n.nextSibling != null) {
     processMe(n.nextSibling);
   }
}
```

Here is the code to do the same thing using a TreeWalker:

```
processMe(TreeWalker tw) {

   doSomething(tw.current());

   if (tw.firstChild() != null) {
     processMe(tw);
   }
   if (tw.nextSibling() != null) {
     processMe(tw);
   }

   tw.parent();
}
```

The main difference between these two functions is that the TreeWalker version must take into account the fact that changing the internal position of the TreeWalker will also affect any calling function that continues to use the TreeWalker. Therefore, a function that uses a TreeWalker should be careful about the position after the function is finished.

The advantage of using a TreeWalker instead of direct Node navigation is that the TreeWalker allows the user to choose an appropriate view of the tree. Flags may be used to show or hide comments or processing instructions, entities may be expanded or left as entity references, and sequences of text nodes may be merged into a single virtual text node. In addition, Filters may be used to present a custom view of the tree. Suppose a program needs a view of a document that shows which tables occur in each chapter, listed by chapter. In this view, only the chapter elements and the tables that they contain are seen. The first step is to write an appropriate filter:

```
class TablesInChapters implements NodeFilter {

      short acceptNode(Node n) {
              if (n instanceof Element) {
                      Element e = n;

                      if (e.nodeName == "CHAPTER")
                              return FILTER_ACCEPT;

                      if (e.nodeName == "TABLE")
                              return FILTER_ACCEPT;

                      if (e.nodeName == "SECT1"
                       || e.nodeName == "SECT2"
                       || e.nodeName == "SECT3"
                       || e.nodeName == "SECT4"
                       || e.nodeName == "SECT5"
                       || e.nodeName == "SECT6"
                       || e.nodeName == "SECT7")
                              return FILTER_SKIP;

              }

              return FILTER_REJECT;
      }
}
```

Now the program can create an instance of this Filter, create a TreeWalker that uses it, and pass this TreeWalker to our ProcessMe() function:

```
TablesInChapters tablesInChapters;
TreeWalker tw(root, SHOW_ELEMENT, TablesInChapters);
ProcessMe(tw);
```

Without making any changes to the above ProcessMe() function, it now processes only the <CHAPTER> and <TABLE> elements. The programmer can write other filters or set other flags to choose different sets of nodes; if functions use TreeWalker to navigate, they will support any view of the document defined with a TreeWalker.

6.2. Formal Interface Definition

Interface *NodeIterator*

NodeIterators are used to step through a set of nodes, e.g. the set of nodes in a NodeList, the document subtree governed by a particular node, the results of a query, or any other set of nodes. The set of nodes to be iterated is determined by the factory that creates the iterator.

Any iterator that returns nodes may implement the NodeIterator interface. Users and vendor libraries may also choose to create iterators that implement the NodeIterator interface.
IDL Definition

```
interface NodeIterator {
  readonly attribute long             whatToShow;
  // Constants for whatToShow
  const unsigned long       SHOW_ALL                      = 0xFFFF;
  const unsigned long       SHOW_ELEMENT                  = 0x00000001;
  const unsigned long       SHOW_ATTRIBUTE                = 0x00000002;
  const unsigned long       SHOW_TEXT                     = 0x00000004;
  const unsigned long       SHOW_CDATA_SECTION            = 0x00000008;
  const unsigned long       SHOW_ENTITY_REFERENCE         = 0x00000010;
  const unsigned long       SHOW_ENTITY                   = 0x00000020;
  const unsigned long       SHOW_PROCESSING_INSTRUCTION   = 0x00000040;
  const unsigned long       SHOW_COMMENT                  = 0x00000080;
  const unsigned long       SHOW_DOCUMENT                 = 0x00000100;
  const unsigned long       SHOW_DOCUMENT_TYPE            = 0x00000200;
  const unsigned long       SHOW_DOCUMENT_FRAGMENT        = 0x00000400;
  const unsigned long       SHOW_NOTATION                 = 0x00000800;

  readonly attribute NodeFilter       filter;
  Node               nextNode();
  Node               previousNode();
};
```

Attributes

`whatToShow`

This attribute determines whether entities are expanded, and whether comments, processing instructions, or text are presented via the iterator.

Definition group *Constants for whatToShow*

These are the available values for the whatToShow parameter. They are the same as the set of possible types for Node, and their values are derived by using a bit position corresponding to the value of NodeType for the equivalent node type.
Defined Constants

SHOW_ALL	Show all nodes.
SHOW_ELEMENT	Show element nodes.
SHOW_ATTRIBUTE	Show attribute nodes. This is meaningful only when creating an iterator with an attribute node as its root; in this case, it means that the attribute node will appear in the first position of the iteration. Since attributes are not part of the document tree, they do not appear when iterating over the document tree.
SHOW_TEXT	Show text nodes.
SHOW_CDATA_SECTION	Show CDATASection nodes.
SHOW_ENTITY_REFERENCE	Show Entity Reference nodes.
SHOW_ENTITY	Show Entity nodes. This currently has no effect.
SHOW_PROCESSING_INSTRUCTION	Show ProcessingInstruction nodes.
SHOW_COMMENT	Show Comment nodes.
SHOW_DOCUMENT	Show Document nodes.
SHOW_DOCUMENT_TYPE	Show DocumentType nodes.
SHOW_DOCUMENT_FRAGMENT	Show DocumentFragment nodes.
SHOW_NOTATION	Show Notation nodes. This currently has no effect.

`filter`

The filter used to screen nodes.

Methods

`nextNode`

Returns the next node in the set and advances the position of the iterator in the set. After a NodeIterator is created, the first call to nextNode() returns the first node in the set.

Return Value

The next `Node` in the set being iterated over, or NULL if there are no more members in that set.

This method has no parameters.

This method raises no exceptions.

`previousNode`

Returns the previous node in the set and moves the position of the iterator backwards in the set.

Return Value

The previous `Node` in the set being iterated over, or NULL if there are no more members in that set.

This method has no parameters.

This method raises no exceptions.

Interface *NodeFilter*

Filters are simply objects that know how to "filter out" nodes. If an iterator is given a filter, before it returns the next node, it applies the filter. If the filter says to accept the node, the iterator returns it; otherwise, the iterator looks for the next node and pretends that the node that was rejected was not there.

The DOM does not provide any filters. Filter is just an interface that users can implement to provide their own filters. The introduction to this chapter gives an example of how a user can implement a filter to perform a specific function.

Filters do not need to know how to iterate, nor do they need to know anything about the data structure that is being iterated. This makes it very easy to write filters, since the only thing they have to know how to do is evaluate a single node. One filter may be used with a number of different kinds of iterators, encouraging code reuse.

If a filter is installed for a TreeWalker or Iterator, the system may use that filter for various tasks, especially during fix-up. Filters should make no assumptions about how frequently they will be called.

IDL Definition

```
interface NodeFilter {
  // Constants returned by acceptNode
  const short              FILTER_ACCEPT                  = 1;
  const short              FILTER_REJECT                  = 2;
  const short              FILTER_SKIP                    = 3;

  short              acceptNode(in Node n);
};
```

Definition group *Constants returned by acceptNode*

The following constants are returned by the acceptNode() method:

Defined Constants

FILTER_ACCEPT	Accept the node. Navigation methods defined for Iterator or TreeWalker will return this node.
FILTER_REJECT	Reject the node. Navigation methods defined for Iterator or TreeWalker will not return this node. For TreeWalker, the children of this node will also be rejected. Iterators treat this as a synonym for FILTER_SKIP.
FILTER_SKIP	Reject the node. Navigation methods defined for Iterator or TreeWalker will not return this node. For both Iterator and Treewalker, the children of this node will still be considered.

Methods

`acceptNode`

Parameters

`n`	The node to check to see if it passes the filter or not.

Return Value

Returns a constant to determine whether the node is accepted, rejected, or skipped, as defined above [p.122] . ***Note:*** *If an exception is thrown in this method, the results are unspecified.*

This method raises no exceptions.

Interface *TreeWalker*

TreeWalkers are used to navigate a document tree or subtree using the view of the document defined by its whatToShow flags and any filters that are defined for the TreeWalker. Any function which performs navigation using a TreeWalker will automatically support any view defined by a TreeWalker.

IDL Definition

```
interface TreeWalker {
  readonly attribute long             whatToShow;
  // Constants for whatToShow
  const unsigned long       SHOW_ALL                       = 0xFFFF;
  const unsigned long       SHOW_ELEMENT                   = 0x00000001;
  const unsigned long       SHOW_ATTRIBUTE                 = 0x00000002;
  const unsigned long       SHOW_TEXT                      = 0x00000004;
  const unsigned long       SHOW_CDATA_SECTION             = 0x00000008;
  const unsigned long       SHOW_ENTITY_REFERENCE          = 0x00000010;
  const unsigned long       SHOW_ENTITY                    = 0x00000020;
  const unsigned long       SHOW_PROCESSING_INSTRUCTION    = 0x00000040;
  const unsigned long       SHOW_COMMENT                   = 0x00000080;
  const unsigned long       SHOW_DOCUMENT                  = 0x00000100;
  const unsigned long       SHOW_DOCUMENT_TYPE             = 0x00000200;
  const unsigned long       SHOW_DOCUMENT_FRAGMENT         = 0x00000400;
  const unsigned long       SHOW_NOTATION                  = 0x00000800;

  readonly attribute NodeFilter       filter;
  Node                current();
  Node                parentNode();
  Node                firstChild();
  Node                lastChild();
  Node                previousSibling();
  Node                nextSibling();
};
```

Attributes

`whatToShow`

This attribute determines whether entities are expanded, and whether comments, processing instructions, or text are presented via the iterator.

Definition group *Constants for whatToShow*

These are the available values for the whatToShow parameter. They are the same as the set of possible types for Node, and their values are derived by using a bit position corresponding to the value of NodeType for the equivalent node type.

Defined Constants

SHOW_ALL	Show all nodes.
SHOW_ELEMENT	Show element nodes.
SHOW_ATTRIBUTE	Show attribute nodes. This is meaningful only when creating an TreeWalker with an attribute node as its root; in this case, it means that the attribute node will appear in the first position of the iteration. Since attributes are not part of the document tree, they do not appear when iterating over the document tree.
SHOW_TEXT	Show text nodes.
SHOW_CDATA_SECTION	Show CDATASection nodes.
SHOW_ENTITY_REFERENCE	Show Entity Reference nodes.
SHOW_ENTITY	Show Entity nodes. This currently has no effect.
SHOW_PROCESSING_INSTRUCTION	Show ProcessingInstruction nodes.
SHOW_COMMENT	Show Comment nodes.
SHOW_DOCUMENT	Show Document nodes.
SHOW_DOCUMENT_TYPE	Show DocumentType nodes.
SHOW_DOCUMENT_FRAGMENT	Show DocumentFragment nodes.
SHOW_NOTATION	Show Notation nodes. This currently has no effect.

`filter`
: The filter used to screen nodes.

Methods

`current`

Returns the current node without changing position.

Return Value

The current node.

This method has no parameters.

This method raises no exceptions.

`parentNode`

Moves to the parent node. This method will never position beyond the root of the subtree for which the TreeWalker was created.

Return Value

The new node. If the current node is the root of the subtree for which the TreeWalker was created, returns null, and retains the current node.

This method has no parameters.

This method raises no exceptions.

`firstChild`

Moves the TreeWalker to the first child of the current node, and returns the new node. If the current node has no children, returns null, and retains the current node.

Return Value

The new node, or null if the current node has no children.

This method has no parameters.

This method raises no exceptions.

`lastChild`

Moves the TreeWalker to the last child of the current node, and returns the new node. If the current node has no children, returns null, and retains the current node.

Return Value

The new node, or null if the current node has no children.

This method has no parameters.

This method raises no exceptions.

`previousSibling`

Moves the TreeWalker to the previous sibling of the current node, and returns the new node. If the current node has no previous sibling, returns null, and retains the current node.

Return Value

The new node, or null if the current node has no previous sibling.

This method has no parameters.

This method raises no exceptions.

`nextSibling`

Moves the TreeWalker to the next sibling of the current node, and returns the new node. If the current node has no next sibling, returns null, and retains the current node.

Return Value

The new node, or null if the current node has no next sibling.

This method has no parameters.

This method raises no exceptions.

Interface *DocumentIF*

Document contains methods that creates iterators to traverse a node and its children in document order (depth first, pre-order traversal, which is equivalent to the order in which the start tags occur in the text representation of the document).

IDL Definition

```
interface DocumentIF {
  short              createNodeIterator(in Node root,
                                        in short whatToShow,
                                        in NodeFilter filter);
};
```

Methods

`createNodeIterator`

Parameters

`root`	The node which will be iterated together with its children.
`whatToShow`	This flag determines whether entities are expanded, and whether comments, processing instructions, or text are presented via the iterator. See the description of Iterator for the set of possible values. These flags can be combined using OR: `NodeIterator iter=doc.createNodeIterator(root, SHOW_ELEMENT | SHOW_PROCESSING_INSTRUCTION | SHOW_COMMENT | SHOW_ENTITY_REFERENCE, myFilter);` If SHOW_ENTITY_REFERENCE is not set, entities are expanded. If SHOW_ENTITY_REFERENCE is set, entity references will be encountered by the iterator. There is no setting that shows both the entity reference and its expansion. (**ED:** Several people have suggested that the functionality of whatToShow be implemented using filters. We feel that it is better to implement them using iterators, since it makes it possible to provide a more efficient implementation. A filter must examine each node individually; an iterator can make use of internal data structures to examine only those nodes that are desired.)
`filter`	

Return Value

The newly created NodeIterator.

This method raises no exceptions.

7. Document Object Model Range

Editors
Vidur Apparao, Netscape Communications
Peter Sharpe, SoftQuad Software Inc.

7.1. Introduction

A Range identifies a range of content in a Document or DocumentFragment. It is contiguous in the sense that it can be characterized as selecting all of the content between a single pair of end-points. ***Note:*** *In a text editor or a word processor, a user can make a selection by pressing down the mouse at one point in a document, moving the mouse to another point, and releasing the mouse. The resulting selection is contiguous and consists of the content between the two points.*
The term ’selected’ does not mean that every Range corresponds to a selection made by a GUI user; however, such a selection can be returned to a DOM user as a Range.

The Range interface provides methods for accessing and manipulating the document tree at a higher level than similar methods in the Node interface. The expectation is that each of the methods provided by the Range interface for the insertion, deletion and copying of content can be directly mapped to a series of Node editing operations enabled by DOM Level 1. In this sense, the Range operations can be viewed as convenience methods that also enable the implementation to optimize common editing patterns.

This chapter describes the Range interface, including methods to create and move a Range and methods to use Ranges to manipulate content.

7.2. Definitions and Notation

7.2.1. Position

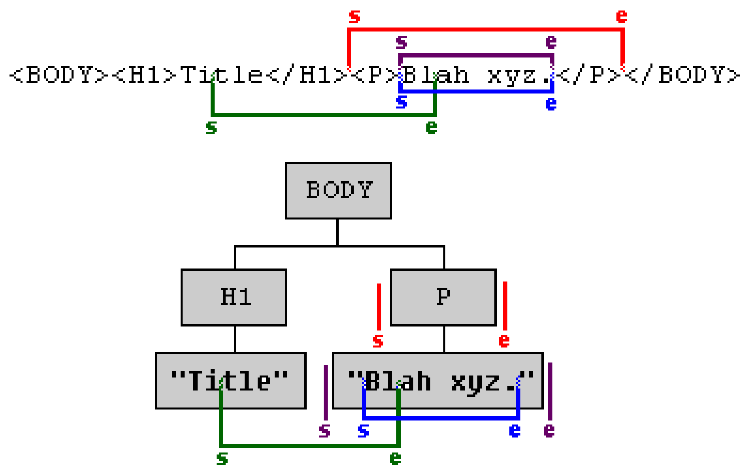
This chapter refers to two different representations of a document - the text or source form that includes the document markup, and the tree representation similar to the one described in the DOM Level 1Introduction.

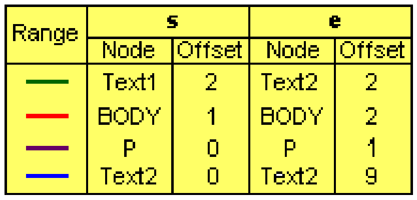
A Range consists of two *end-points* corresponding to the start and the end of the Range. An end-point’s position in a document or document fragment tree can be characterized by a node and an offset. The node is called the *container* of the end-point and of its position. The container and its ancestors are the *ancestor container*s of the end-point and of its position. The offset within the node is called the *offset* of the end-point and its position. If the container is an Attribute, Document, Document Fragment, Element or EntityReference node, the offset is within its child nodes list. If the container is a CharacterData, Comment or Processing Instruction node, the offset is within the 16-bit units contained by it.

The end-points of a Range must have a common ancestor container which is either a Document, DocumentFragment or Attr node. That is, the Range must contain content that is entirely within the subtree rooted by a single Document, DocumentFragment or Attr Node. The container of an end-point of a Range must be an Element, Comment, ProcessingInstruction, EntityReference, CDATASection, Document, DocumentFragment, Attr, or Text node. None of the ancestor containers of the end-point of a Range can be a DocumentType, Entity and Notation node.

Viewed in terms of the text representation of a document, the end-points of a Range can only be on token boundaries. That is, the end-point of the text range cannot be in the middle of a start- or end-tag of an element or within an entity or character reference. A Range locates a contiguous portion of the content of the structure model.

The relationship between locations in a text representation of the document and in the Node tree interface of the DOM is illustrated in the following diagram:





Range	s		e	
	Node	Offset	Node	Offset
🟩	Text1	2	Text2	2
🟥	BODY	1	BODY	2
🟪	P	0	P	1
🟦	Text2	0	Text2	9

Range Example

In this diagram, four different Ranges are illustrated. The end-points of each range are labelled with *s* (the start of the range) and *e* (the end of the range). For the red Range, the start is in the BODY element and is immediately after the H1 element and immediately before the P element, so its position is between the H1 and P children of BODY. The offset of an end-point whose container is not a Text node is 0 if it is before the first child, 1 if between the first and second child, and so on. So, for the start of the red Range, the container is BODY and the offset is 1. The offset of an end-point whose container is a Text node is obtained similarly but using 16-bit unit positions instead. For example, the end-point labelled s of the green Range has a Text node (the one containing "Title") as its container and an offset of 2 since it is between the second and third 16-bit unit.

Notice that the end-points of purple and blue ranges correspond to the same location in the text representation. An important feature of the Range is that an end-point of a Range can unambiguously represent every position within the document tree.

The containers and offsets of the end-points can be obtained through the following read-only Range attributes:

```
  readonly attribute Node startContainer;
  readonly attribute long startOffset;
  readonly attribute Node endContainer;
  readonly attribute long endOffset;
```

If the end-points of a Range have the same containers and offsets, the Range is said to be a *collapsed* Range. (This is often referred to as an insertion point in a user agent.)

7.2.2. Selection and Partial Selection

A node or 16-bit unit is said to be *selected* by a Range if it is between the two end-points of the Range, that is, if the position immediately before the node or 16-bit unit is before the end of the Range and the position immediately after the node or 16-bit unit is after the start of the range. For example, in terms of a text representation of the document, an element would be selected by a Range if its corresponding start-tag was located after the start of the Range and its end-tag was located before the end of the Range. In the examples in the above diagram, the red Range selects the P node and the purple Range selects the text node containing the text "Blah xyz."

A node is said to be *partially selected* by a Range if it is an ancestor container of exactly one end-point of the Range. For example, consider the green Range in the above diagram. H1 is partially selected by that Range since the start of the Range is within one of its children.

7.2.3. Notation

Many of the examples in this chapter are illustrated using a text representation of a document. The end-points of a range are indicated by displaying the characters (be they markup or data characters) between the two end-points in bold, as in

<FOO>A**BC<BAR>DE**F</BAR></FOO>

When both end-points are at the same position, they are indicated with a bold caret ('**^**'), as in

<FOO>A**^**BC<BAR>DEF</BAR></FOO>

And when referring to a single end-point, it will be shown as a bold asterisk ('*****') as in

<FOO>A*****BC<BAR>DEF</BAR></FOO>

7.3. Creating a Range

A range is created by calling a method on the RangeFactory interface. The expectation is that this interface can be obtained from the object implementing the Document using binding-specific casting methods.

```
  interface RangeFactory {
    Range createRange();
  }
```

The initial state of the range returned from this method is such that both of its end-points are positioned at the beginning of the corresponding Document, before any content. In other words, the container of each end-point is the Document node and the offset within that node is 0.

Like some objects created using methods in the Document interface (such as Nodes and DocumentFragments), Ranges created via a particular document instance can select only content associated with that Document, or DocumentFragments and Attrs for which that Document is the `ownerDocument`. This Range can then not be be used with other Document instances. The DOM WG is considering allowing a Range instance to be used with any Document. While the rules associated with common ancestor containers for a Range’s end-points will remain the same, a Range would not be tied to a specific Document instance.

7.4. Changing a Range’s Position

A Range’s position can be specified by setting the container and offset of each end-point with the `setStart` and `setEnd` methods.

```
void setStart(in Node parent, in long offset)
                      raises(RangeException);
void setEnd(in Node parent, in long offset)
              raises(RangeException);
```

If one end-point of a Range is set to be positioned somewhere in a Document, DocumentFragment or Attr node other than the one in which the range is currently positioned, the range is collapsed to the new position. This enforces the restriction that both end-points of a Range must be in the same document or fragment.

The start position is guaranteed to never be after the end position. To enforce this restriction, if the start is set to be at a position after the end, the range is collapsed to that position. The case in which the end is set to be at a position before the start is similarly handled.

It is also possible to set a Range’s position relative to nodes in the tree:

```
void setStartBefore(in Node node);
                            raises(RangeException);
void setStartAfter(in Node node);
                     raises(RangeException);
void setEndBefore(in Node node);
                    raises(RangeException);
void setEndAfter(in Node node);
                   raises(RangeException);
```

The parent of the node becomes the container of the end-point and the Range is subject to the same restrictions as given above in the description of `setStart()` and `setEnd()`.

A Range can be collapsed to either end-point:

```
  void collapse(in boolean toStart);
```

Passing `TRUE` to the parameter toStart will collapse the Range to its start , `FALSE` to its end.

Testing whether a Range is collapsed can be done by examining the `isCollapsed` attribute:

```
  readonly attribute boolean isCollapsed;
```

The following methods can be used to make a range select the contents of a node or the node itself.

```
  void selectNode(in Node n);
  void selectNodeContents(in Node n);
```

The following examples demonstrate the operation of the methods `selectNode` and `selectNodeContents`:

```
Before:
  ^<BAR><FOO>A<MOO>B</MOO>C</FOO></BAR>
After range.selectNodeContents(FOO):
  <BAR><FOO>A<MOO>B</MOO>C</FOO></BAR>
After range.selectNode(FOO):

<BAR><FOO>A<MOO>B</MOO>C</FOO></BAR>
```

7.5. Comparing Range End-Points

It is possible to compare two Ranges by comparing their end-points:

```
  int compareEndPoints(CompareHow how, Range sourceRange);
```

where `CompareHow` is one of four values: `StartToStart`, `StartToEnd`, `EndToEnd` and `EndToStart`. The return value is -1, 0 or 1 depending on whether the corresponding end-point of the Range is before, equal to, or after the corresponding end-point of sourceRange.

The result of comparing two end-points (or positions) is specified below. An informal but incorrect specification is that an end-point is before, equal to, or after another if it corresponds to a location in a text representation before, equal to, or after the other's corresponding location.

Let A and B be two end-points or positions. Then one of the following holds: A is *before* B, A is *equal to* B, or A is *after* B. Which one holds is specified in the following by examining four cases:

In the first case the end-points have the same container. A is *before* B if its offset is less than the offset of B, A is *equal to* B if its offset is equal to the offset of B, and A if *after* B if its offset is greater than the offset of B.

In the second case a child C of the container of A is an ancestor container of B. In this case, A is *before* B if the offset of A is less than or equal to the index of the child C and A is *after* B otherwise.

In the third case a child C of the container of B is an ancestor container of A. In this case, A is *before* B if the index of the child C is less than the offset of B and A is *after* B otherwise.

In the fourth case none of three other cases hold (then the containers of A and B are siblings or descendants of sibling nodes). In this case, A is *before* B if the container of A is before container of B in a pre-order traversal and A is *after* B otherwise.

Note that because the same location in a text representation of the document can correspond to two different positions in the DOM tree, it is possible for two end-points to not compare equal even though they would be equal in the text representation. For this reason, the informal definition above can sometimes be incorrect.

7.6. Deleting Content with a Range

One can delete the contents selected by a Range with:

```
  void deleteContents();
```

`deleteContents()` deletes all nodes and characters selected by the Range. All other nodes and characters remain in the document or document fragment that the Range is in. Some examples:

```
(1) <FOO>AB<MOO>CD</MOO>CD</FOO>  -->
<FOO>A^CD</FOO>

(2) <FOO>A<MOO>BC</MOO>DE</FOO>  -->
<FOO>A<MOO>B</MOO>^E</FOO>

(3) <FOO>XY<BAR>ZW</BAR>Q</FOO>  -->
<FOO>X^<BAR>W</BAR>Q</FOO>

(4)
<FOO><BAR1>AB</BAR1><BAR2/><BAR3>CD</BAR3></FOO>
-->  <FOO><BAR1>A</BAR1>^<BAR3>D</BAR3>
```

After `deleteContents()`, the Range is collapsed. If no node was partially selected by the Range, then it is collapsed to its original start point, as in example (1). If a node was partially selected by the range and was an ancestor container of the start of the range and no ancestor of the node satisfies these two conditions, then the range is collapsed to the position immediately after the node, as in examples (2) and (4). If a node was partially selected by the range and was an ancestor container of the end of the range and no ancestor of the node satisfies these two conditions, then the range is collapsed to the position immediately before the node, as in examples (3) and (4).

7.7. Extracting Content

If the contents of a range should be extracted rather than deleted, the following method may be used:

```
  DocumentFragment extractContents();
```

The `extractContents()` method does what the `deleteContents()` method does and, in addition, places the deleted contents in a new DocumentFragment. The following examples illustrate the contents of the returned document fragment:

```
(1) <FOO>AB<MOO>CD</MOO>CD</FOO>  -->
B<MOO>CD</MOO>

(2) <FOO>A<MOO>BC</MOO>DE</FOO>  -->
<MOO>C<MOO>D

(3) <FOO>XY<BAR>ZW</BAR>Q</FOO>  -->
<BAR>Y</BAR>Z

(4)
<FOO><BAR1>AB</BAR1><BAR2/><BAR3>CD</BAR3></FOO>
-->  <BAR1>B</BAR1><BAR2/><BAR3>C</BAR3>
```

It is important to note that nodes that are partially selected by the range are cloned. Since part of such a node's contents must remain in the original document (or document fragment) and part of the contents must be moved to the new fragment, a clone of the partially selected node is brought along to the new fragment. Note that cloning does not take place for selected elements; these nodes are moved to the new fragment.

7.8. Cloning Content

The contents of a range may be duplicated using the following method:

```
  DocumentFragment cloneContents();
```

This method returns a document fragment that is similar to the one returned by the method `extractContents()`. However, in this case, the original nodes and text content in the range are not deleted from the original document. Instead, all of the nodes and text content within the returned document fragment are cloned.

7.9. Inserting Content

A node may be inserted into a range using the following method:

```
  void insertNode(in Node n);
```

The `insertNode()` method inserts the specified node into the document or document fragment in which the range resides. For this method, the end of the range is ignored and the node is inserted at the start of the range.

The Node passed into this method can be a DocumentFragment. In that case, the contents of the fragment are inserted at the start position of the range, but the fragment itself is not. Note that if the Node represents the root of a sub-tree, the entire sub-tree is inserted.

Note that the same rules that apply to the `insertBefore()` method on the Node interface apply here. Specifically, the Node passed in will be removed from its existing position in the same document or another fragment.

7.10. Surrounding Content

The insertion of a single node to subsume the content selected by range can be performed with:

```
void surroundContents(in Node n);
```

The `surroundContents` member differs from `insertNode()` in that `surroundContents`() causes all of the content selected by the range to become content of `node`, whereas `insertNode()` splices in existing content at the given point in the document.

For example, calling `surroundContents()` with the node FOO yields:

```
Before:
  <BAR>AB<MOO>C</MOO>DE</BAR>
After surroundContents ( FOO ):
```

```
<BAR>A<FOO>B<MOO>C</MOO>D</FOO>E</BAR>
```

Another way of of describing the effect of this member on the document or fragment tree is to decompose it in terms of other operations:

1. Remove the contents selected by the range with a call to `extractContents()`.
2. Insert `node` where the range is now collapsed (after the extraction) with `insertNode()`
3. Insert the entire contents of the extracted contents into `node`.
4. Select `node` and all of its contents with `selectNode()`.

Because inserting a node in such a manner will be a common operation, `surroundContents()` is provided to avoid the overhead of these four steps.

The `surroundContents()` method raises an exception if the range partially selects a non-Text node. An example of a range for which `surroundContents()` raises an exception is:

```
<FOO>AB<BAR>CD</BAR>E</FOO>
```

If `node` has any children, those children are removed before its insertion. Also, if `node` is part of any existing content, it is also removed from that content before insertion.

7.11. Miscellaneous Members

One can clone a Range:

```
Range cloneRange();
```

This creates a new Range which selects exactly the same content of the Range on which it was called. No content is affected by this operation.

Because the end-points of a range do not necessarily have the same containers, use:

```
  readonly attribute Node commonAncestorContainer;
```

to get the deepest node that is an ancestor container of both end-points.

One can get a copy of all the text nodes selected or partially selected by a range with:

```
  DOMString toString();
```

This does nothing more than simply concatenate all the characters selected by the range.

7.12. Range modification under document mutation

As a document is mutated, the Ranges within the document need to be updated. For example, if one end-point of a Range is within a node and that node is removed from the document, then the Range would be invalid unless it is fixed up in some way. This section describes how Ranges are modified under document mutations so that they remain valid.

There are two general principles which apply to Ranges under document mutation: The first is that all Ranges in a document will remain valid after any mutation operation and the second is that, loosely speaking, all Ranges will select the same portion of the document after any mutation operation, where that is possible.

Any mutation of the document tree which affect Ranges can be considered to be a combination of basic delete and insertion operations. In fact, it can be convenient to think of those operations as being accomplished using the `deleteContents()` and `insertNode()` Range methods.

7.12.1. Insertions

An insertion occurs at a single point, the insertion point, in the document. For any Range in the document tree, consider each end-point. The only case in which the end-point will be changed after the insertion is when the end-point and the insertion point have the same container and the offset of the insertion point is strictly less than the offset of the Range’s end-point. In that case the offset of the Range’s end-point will be increased so that it is between the same nodes or characters as it was before the insertion.

Note that when content is inserted at an end-point, it is ambiguous as to where the end-point should be repositioned if its relative position is to be maintained.

This is not the same as the principle, given above, of having the Range select the same content, although often the Range ends up selecting the same content.There are two possibilities: at the start or at the end of the newly inserted content. We have chosen that in this case neither the container nor offset of the end-point is changed. As a result, it will be positioned at the start of the newly inserted content.

Examples:

Suppose the Range selects the following:

<P>Abcd efgh X**Y blah i**jkl</P>

Consider the insertion of the text "*inserted text*" at the following positions:

1. Before the ’X’:

<P>Abcd efgh *inserted text*X**Y blah i**jkl</P>

2. After the ’X’:

<P>Abcd efgh X***inserted text*****Y blah i**jkl</P>

3. After the ’Y’:

<P>Abcd efgh X**Y*inserted text* blah i**jkl</P>

4. After the ’h’ in "Y blah":

<P>Abcd efgh X**Y blah*inserted text* i**jkl</P>

7.12.2. Deletions

Any deletion from the document tree can be considered as a sequence of `deleteContents()` operations applied to a minimal set of disjoint Ranges. To specify how a Range is modified under deletions we need only to consider what happens to a Range only under a single `deleteContents()` operation of another Range. And, in fact, we need only to consider what happens to a single end-point of the Range since both end-points are modified using the same algorithm.

If an end-point is within the content being deleted, then after the deletion it will be at the same position as the one common to the end-points of the Range used to delete the contents.

If an end-point is after the content being deleted then it is not affected by the deletion unless its container is also the container of one of the end-points of the range being deleted. If there is such a common container, then the index of the end-point is modified so that the end-point maintains its position relative to the content of the container.

If an end-point is before the content being deleted then it is not affected by the deletion at all.

Examples:

In these examples, the Range on which `deleteContents()` is invoked is indicated by the underline.

Example 1.

Before:

```
<P>Abcd efgh The Range
ijkl</P>
```

After:

```
<P>Abcd Range ijkl</P>
```

Example 2.

Before:

```
<p>Abcd efgh The Range ijkl</p>
```

After:

```
<p>Abcd ^kl</p>
```

Example 3.

Before:

```
<P>ABCD efgh The<EM>Range</EM> ijkl</P>
```

After:

```
<P>ABCD <EM>ange</EM> ijkl</P>
```

Example 4.

Before:

```
<P>Abcd efgh The Range ijkl</P>
```

After:

```
<P>Abcd he Range ijkl</P>
```

Example 5.

Before:

```
<P>Abcd <EM>efgh The Rangeij</EM>kl</P>
```

After:

```
<P>Abcd ^kl</P>
```

7.13. Formal Description of the Range Interface

To summarize, the complete, formal description of the `Range` [p.141] interface is given below:

Interface *Range*

IDL Definition

```
interface Range {
  readonly attribute Node             startContainer;
  readonly attribute long             startOffset;
  readonly attribute Node             endContainer;
  readonly attribute long             endOffset;
  readonly attribute boolean          isCollapsed;
  readonly attribute Node             commonAncestorContainer;
  void               setStart(in Node node, 
                              in long offset)
                                        raises(RangeException);
  void               setEnd(in Node node, 
                            in long offset)
                                        raises(RangeException);
  void               setStartBefore(in Node node)
                                        raises(RangeException);
  void               setStartAfter(in Node node)
                                        raises(RangeException);
  void               setEndBefore(in Node node)
                                        raises(RangeException);
  void               setEndAfter(in Node node)
                                        raises(RangeException);
  void               collapse(in boolean toStart);
  void               selectNode(in Node node)
                                        raises(RangeException);
  void               selectNodeContents(in Node node)
                                        raises(RangeException);
  typedef enum CompareHow_ {
    StartToStart,
    StartToEnd,
    EndToEnd,
    EndToStart
  } CompareHow;
  short              compareEndPoints(in CompareHow how, 
                                      in Range sourceRange)
                                        raises(DOMException);
  void               deleteContents()
                                        raises(DOMException);
  DocumentFragment   extractContents()
                                        raises(DOMException);
  DocumentFragment   cloneContents()
                                        raises(DOMException);
  void               insertNode(in Node node)
                                        raises(DOMException, RangeException);
  void               surroundContents(in Node node)
                                        raises(DOMException, RangeException);
  Range              cloneRange();
  DOMString          toString();
};
```

Attributes

`startContainer`
: Node within which the range begins

`startOffset`
: Offset within the starting node of the range.

`endContainer`
: Node within which the range ends

`endOffset`
: Offset within the ending node of the range.

`isCollapsed`
: TRUE if the range is collapsed

`commonAncestorContainer`
: The common ancestor container of the range’s two end-points.

Type Definition *CompareHow*

Enumeration *CompareHow_*

Enumerator Values

StartToStart	
StartToEnd	
EndToEnd	
EndToStart	

Methods

`setStart`

Sets the attributes describing the start of therange.

Parameters

`node`	The `startNode` value. Thisparameter must be non-null.
`offset`	The `startOffset` value.

Exceptions

`RangeException` [p.147]

NULL_NODE_ERR: Raised if `node`is null.

INVALID_NODE_TYPE_ERR: Raised if`node` or an ancestor of `node` is anAttr, Entity, Notation, or DocumentType node.

If an offset is out-of-bounds, shouldit just be fixed up or should an exception be raised.

This method returns nothing.

`setEnd`

Sets the attributes describing the end of a range.

Parameters

`node`	The `endNode` value. Thisparameter must be non-null.
`offset`	The `endOffset` value.

Exceptions

`RangeException` [p.147]

NULL_NODE_ERR: Raised if `node`is null.

INVALID_NODE_TYPE_ERR: Raised if`node` or an ancestor of `node` is anAttr, Entity, Notation, or DocumentType node.

This method returns nothing.

`setStartBefore`

Sets the start position to be before a node

Parameters

`node`	Range starts before `node`

Exceptions

`RangeException` [p.147]

INVALID_NODE_TYPE_ERR: Raised if an ancestorof `node` is an Attr, Entity,Notation, or DocumentType node or if `node` is a Document,DocumentFragment, Attr, Entity, or Notation node.

This method returns nothing.

`setStartAfter`

Sets the start position to be after a node

Parameters

`node`	Range starts after `node`

Exceptions

`RangeException` [p.147]

INVALID_NODE_TYPE_ERR: Raised if an ancestorof `node` is an Attr, Entity,Notation, or DocumentType node or if `node` is a Document,DocumentFragment, Attr, Entity, or Notation node.

This method returns nothing.

`setEndBefore`

Sets the end position to be before a node.

Parameters

`node`	Range ends before `node`

Exceptions

`RangeException` [p.147]

INVALID_NODE_TYPE_ERR: Raised if an ancestorof `node` is an Attr, Entity,Notation, or DocumentType node or if `node` is a Document,DocumentFragment, Attr, Entity, or Notation node.

This method returns nothing.

`setEndAfter`

Sets the end of a range to be after a node

Parameters

`node`	Range ends after `node`.

Exceptions

`RangeException` [p.147]

INVALID_NODE_TYPE_ERR: Raised if an ancestorof `node` is an Attr, Entity,Notation or DocumentType node or if `node` is a Document,DocumentFragment, Attr, Entity, or Notation node.

This method returns nothing.

`collapse`

Collapse a range onto one of its end-points

Parameters

`toStart`	If TRUE, collapses the Range onto its start;if FALSE, collapses it onto its end.

This method returns nothing.

This method raises no exceptions.

`selectNode`

Select a node and its contents

Parameters

`node`	The node to select.

Exceptions

`RangeException` [p.147]

INVALID_NODE_TYPE_ERR: Raised if an ancestorof `node` is an Attr, Entity,Notation or DocumentType node or if `node` is a Document,DocumentFragment, Attr, Entity, or Notation node.

This method returns nothing.

`selectNodeContents`

Select the contents within a node

Parameters

`node`	Node to select from

Exceptions

`RangeException` [p.147]

INVALID_NODE_TYPE_ERR: Raised if`node` or an ancestor of `node` is anAttr, Entity, Notation or DocumentType node.

This method returns nothing.

`compareEndPoints`

Compare the end-points of two ranges in a document.

Parameters

`how`

`sourceRange`

Return Value

-1, 0 or 1 depending on whether the correspondingend-point of the Range is before, equal to, or after thecorresponding end-point of `sourceRange`.

Exceptions

`DOMException`

WRONG_DOCUMENT_ERR: Raised if the two Rangesare not in the same document or document fragment.

`deleteContents`

Removes the contents of a range from the containingdocument or document fragment without returning a reference to theremoved content.

Exceptions

`DOMException`

NO_MODIFICATION_ALLOWED_ERR: Raised if anyportion of the content of the range is read-only or anyof the nodes that contain any of the content of the range areread-only.

This method has no parameters.

This method returns nothing.

`extractContents`

Moves the contents of a range from the containingdocument or document fragment to a new DocumentFragment.

Return Value

A DocumentFragment containing the extractedcontents.

Exceptions

`DOMException`

NO_MODIFICATION_ALLOWED_ERR: Raised if anyportion of the content of the range is read-only or anyof the nodes which contain any of the content of the range are read-only.

HIERARCHY_REQUEST_ERR: Raised if aDocumentType node would be extracted into the newDocumentFragment.

This method has no parameters.

`cloneContents`

Duplicates the contents of a range

Return Value

A DocumentFragment containing contents equivalentto those of this range.

Exceptions

`DOMException`

HIERARCHY_REQUEST_ERR: Raised if aDocumentType node would be extracted into the newDocumentFragment.

This method has no parameters.

`insertNode`

Inserts a node into the document or document fragmentat the start of the range.

Parameters

`node`	The node to insert at the start of therange

Exceptions

`DOMException`

NO_MODIFICATION_ALLOWED_ERR: Raised if an ancestor container of the start of the range is read-only.

WRONG_DOCUMENT_ERR: Raised if`node` and the container of the start of the Range were not created from the same document.

HIERARCHY_REQUEST_ERR: Raised if the container of the start of the Range is of a type that does not allow children ofthe type of `node` or if `node` is an ancestor of thecontainer.

`RangeException` [p.147]

INVALID_NODE_TYPE_ERR: Raised if`node` is an Attr, Entity, Notation,DocumentFragment, or Document node.

This method returns nothing.

`surroundContents`

Reparents the contents of the range to the given nodeand inserts the node at the position of the start of therange.

Parameters

`node`	The node to surround the contents with.

Exceptions

`DOMException`

NO_MODIFICATION_ALLOWED_ERR: Raised if an ancestor container of either end-point of the range is read-only.

WRONG_DOCUMENT_ERR: Raised if`node` and the container of the start of the Range were not created from the same document.

HIERARCHY_REQUEST_ERR: Raised if the container of the start of the Range is of a type that does not allow children ofthe type of `node` or if `node` is an ancestor of thecontaineror if `node` would end up with a child node of a type not allowedby the type of `node`.

`RangeException` [p.147]

BAD_ENDPOINTS_ERR: Raised if the range partially selects a non-text node.

INVALID_NODE_TYPE_ERR: Raised if`node` is an Attr, Entity, DocumentType, Notation,Document, or DocumentFragment node.

This method returns nothing.

`cloneRange`

Produces a new range whose end-points are equal tothe end-points of the range.

Return Value

The duplicated range.

This method has no parameters.

This method raises no exceptions.

`toString`

Returns the contents of a range as a string.

Return Value

The contents of the range.

This method has no parameters.

This method raises no exceptions.

Exception *RangeException*

The Range object needs additional exception codes to thosein DOM Level 1. These codes will need to be consolidated withother exception codes added to DOM Level 2.

IDL Definition

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

// RangeExceptionCode
const unsigned short      BAD_ENDPOINTS_ERR              = 201;
const unsigned short      INVALID_NODE_TYPE_ERR          = 202;
const unsigned short      NULL_NODE_ERR                  = 203;
```

Definition group *RangeExceptionCode*

An integer indicating the type of error generated.
Defined Constants

BAD_ENDPOINTS_ERR	If the end-points of a range do not meet specific requirements.
INVALID_NODE_TYPE_ERR	If the container of an end-point of a range is being set to either a node ofan invalid type or a node with an ancestor of an invalid type.
NULL_NODE_ERR	If the container of an end-point of a range is being set to null.

Appendix A: Contributors

Members of the DOM Working Group and Interest Group contributing to this specification were:

Lauren Wood, SoftQuad Software Inc., *chair*
Arnaud Le Hors, W3C, *W3C staff contact*
Andy Heninger, IBM
Bill Smith, Sun
Bill Shea, Merrill Lynch
Bob Sutor, IBM
Chris Wilson, Microsoft
David Brownell, Sun
Don Park, Docuverse
Eric Vasilik, Microsoft
Gavin Nicol, INSO
Jared Sorensen, Novell
Joe Kesselman, IBM
Joe Lapp, webMethods
Jonathan Robie, Software AG
Mike Champion, Software AG
Peter Sharpe, SoftQuad Software Inc.
Philippe Le Hégaret, W3C
Ramesh Lekshmynarayanan, Merrill Lynch
Ray Whitmer, iMall
Rich Rollman, Microsoft
Tom Pixley, Netscape
Vidur Apparao, Netscape

Appendix A: Contributors

Appendix B: Glossary

Editors
Robert S. Sutor, IBM Research

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.

ancestor
: An *ancestor* node of any node A is any node above A in a tree model of a document, where "above" means "toward the root."

API
: An *API* is an application programming interface, a set of functions or methods used to access some functionality.

child
: A *child* is an immediate descendant node of a node.

client application
: A [client] application is any software that uses the Document Object Model programming interfaces provided by the hosting implementation to accomplish useful work. Some examples of client applications are scripts within an HTML or XML document.

COM
: *COM* is Microsoft's Component Object Model, a technology for building applications from binary software components.

content model
: The *content model* is a simple grammar governing the allowed types of the child elements and the order in which they appear. See [XML]

context
: A *context* specifies an access pattern (or path): a set of interfaces which give you a way to interact with a model. For example, imagine a model with different colored arcs connecting data nodes. A context might be a sheet of colored acetate that is placed over the model allowing you a partial view of the total information in the model.

convenience
: A *convenience method* is an operation on an object that could be accomplished by a program consisting of more basic operations on the object. Convenience methods are usually provided to make the API easier and simpler to use or to allow specific programs to create more optimized implementations for common operations. A similar definition holds for a *convenience property*.

cooked model
: A model for a document that represents the document after it has been manipulated in some way. For example, any combination of any of the following transformations would create a cooked model:

1. Expansion of internal text entities.
2. Expansion of external entities.
3. Model augmentation with style-specified generated text.
4. Execution of style-specified reordering.
5. Execution of scripts.

A browser might only be able to provide access to a cooked model, while an editor might provide access to a cooked or the initial structure model (also known as the *uncooked model*) for a document.

CORBA
: *CORBA* is the *Common Object Request Broker Architecture* from the OMG. This architecture is a collection of objects and libraries that allow the creation of applications containing objects that make and receive requests and responses in a distributed environment.

cursor
A *cursor* is an object representation of a node. It may possess information about context and the path traversed to reach the node.

data model
A *data model* is a collection of descriptions of data structures and their contained fields, together with the operations or functions that manipulate them.

deprecation
When new releases of specifications are released, some older features may be marked as being *deprecated*. This means that new work should not use the features and that although they are supported in the current release, they may not be supported or available in future releases.

descendant
A *descendant* node of any node A is any node below A in a tree model of a document, where "above" means "toward the root."

ECMAScript
The programming language defined by the ECMA-262 standard. As stated in the standard, the originating technology for ECMAScript was JavaScript. Note that in the ECMAScript binding, the word "property" is used in the same sense as the IDL term "attribute."

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. [XML]

event propagation, also known as event bubbling
This is the idea that an event can affect one object and a set of related objects. Any of the potentially affected objects can block the event or substitute a different one (upward event propagation). The event is broadcast from the node at which it originates to every parent node.

equivalence
Two nodes are *equivalent* if they have the same node type and same node name. Also, if the nodes contain data, that must be the same. Finally, if the nodes have attributes then collection of attribute names must be the same and the attributes corresponding by name must be equivalent as nodes. Two nodes are *deeply equivalent* if they are *equivalent*, the child node lists are equivalent are equivalent as NodeList objects, and the pairs of equivalent attributes must in fact be deeply equivalent. Two NodeList objects are *equivalent* if they have the same length, and the nodes corresponding by index are deeply equivalent. Two NamedNodeMap objects are *equivalent* if they are have the same length, they have same collection of names, and the nodes corresponding by name in the maps are deeply equivalent. Two DocumentType nodes are *equivalent* if they are equivalent as nodes, have the same names, and have equivalent entities and attributes NamedNodeMap objects.

hosting implementation
A [hosting] implementation is a software module that provides an implementation of the DOM interfaces so that a client application can use them. Some examples of hosting implementations are browsers, editors and document repositories.

HTML
The HyperText Markup Language (*HTML*) is a simple markup language used to create hypertext documents that are portable from one platform to another. HTML documents are SGML documents with generic semantics that are appropriate for representing information from a wide range of applications. [HTML 3.2] [HTML4.0]

IDL
: An Interface Definition Language (*IDL*) is used to define the interfaces for accessing and operating upon objects. Examples of IDLs are the Object Management Group’s IDL, Microsoft’s IDL, and Sun’s Java IDL.

implementor
: Companies, organizations, and individuals that claim to support the Document Object Model as an API for their products.

inheritance
: In object-oriented programming, the ability to create new classes (or interfaces) that contain all the methods and properties of another class (or interface), plus additional methods and properties. If class (or interface) D inherits from class (or interface) B, then D is said to be *derived* from B. B is said to be a *base* class (or interface) for D. Some programming languages allow for multiple inheritance, that is, inheritance from more than one class or interface.

initial structure model
: Also known as the *raw structure model* or the *uncooked model*, this represents the document before it has been modified by entity expansions, generated text, style-specified reordering, or the execution of scripts. In some implementations, this might correspond to the "initial parse tree" for the document, if it ever exists. Note that a given implementation might not be able to provide access to the initial structure model for a document, though an editor probably would.

interface
: An *interface* is a declaration of a set of methods with no information given about their implementation. In object systems that support interfaces and inheritance, interfaces can usually inherit from one another.

language binding
: A programming *language binding* for an IDL specification is an implementation of the interfaces in the specification for the given language. For example, a Java language binding for the Document Object Model IDL specification would implement the concrete Java classes that provide the functionality exposed by the interfaces.

method
: A *method* is an operation or function that is associated with an object and is allowed to manipulate the object’s data.

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.

object model
: An *object model* is a collection of descriptions of classes or interfaces, together with their member data, member functions, and class-static operations.

parent
: A *parent* is an immediate ancestor node of a node.

root node
: The *root node* is the unique node that is not a child of any other node. All other nodes are children or other descendents of the root node. [XML]

sibling
: Two nodes are *siblings* if they have the same parent node.

string comparison
: When string matching is required, it is to occur as though the comparison was between 2 sequences of code points from the Unicode 2.0 standard.

tag valid document
: A document is *tag valid* if all begin and end tags are properly balanced and nested.

type valid document
: A document is *type valid* if it conforms to an explicit DTD.

uncooked model
: See initial structure model.

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).

XML
: Extensible Markup Language (*XML*) is an extremely simple dialect of SGML which is completely described in this document. The goal is to enable generic SGML to be served, received, and processed on the Web in the way that is now possible with HTML. XML has been designed for ease of implementation and for interoperability with both SGML and HTML. [XML]

Appendix B: Glossary

Appendix C: IDL Definitions

This appendix contains the complete OMG IDL for the Level 2 Document Object Model definitions. The definitions are divided into Core [p.157] , Namespaces [p.158] , Stylesheets [p.159] , CSS [p.160] , Events [p.168] , Filters and Iterators [p.173] , and Range [p.174] .

The IDL files are also available as: http://www.w3.org/TR/1999/WD-DOM-Level-2-19990719/idl.zip

C.1: Document Object Model Level 2 Core

dom2.idl:

```
// File: dom2.idl
#ifndef _DOM2_IDL_
#define _DOM2_IDL_

#include "dom.idl"

#pragma prefix "dom.w3c.org"
module dom2
{
  typedef dom::DocumentType DocumentType;
  typedef dom::DOMString DOMString;
  typedef dom::DOMImplementation DOMImplementation;
  typedef dom::Document Document;
  typedef dom::Node Node;
  typedef dom::Attr Attr;
  typedef dom::Element Element;
  typedef dom::HTMLDocument HTMLDocument;

  interface DocumentType2 : DocumentType {
    readonly attribute DOMString        publicID;
    readonly attribute DOMString        systemID;
  };

  interface DOMImplementation2 : DOMImplementation {
    DocumentType       createDocumentType(in DOMString name,
                                          in DOMString publicID,
                                          in DOMString systemID)
                                        raises(DOMException);
    Document           createDocument(in DOMString name,
                                      in DocumentType doctype)
                                        raises(DOMException);
  };

  interface Document2 : Document {
    Node               importNode(in Node importedNode,
                                  in boolean deep);
  };

  interface Node2 : Node {
    boolean            supports(in DOMString feature,
                                in DOMString version);
  };

  interface Attr2 : Attr {
    readonly attribute Element          ownerElement;
  };

  interface HTMLDOMImplementation : DOMImplementation {
    HTMLDocument       createHTMLDocument(in DOMString title);
  };

};

#endif // _DOM2_IDL_
```

C.2: Document Object Model Level 2 Namespaces

namespaces.idl:

```
// File: namespaces.idl
#ifndef _NAMESPACES_IDL_
#define _NAMESPACES_IDL_

#include "dom.idl"
#include "dom2.idl"

#pragma prefix "dom.w3c.org"
module namespaces
{
  typedef dom dom2::DOMString DOMString;
  typedef dom dom2::Element Element;
  typedef dom dom2::Attr Attr;
  typedef dom dom2::NodeList NodeList;

  interface NodeNS {
    readonly attribute DOMString        namespaceName;
             attribute DOMString        prefix;
                                        // raises(DOMException) on setting

    readonly attribute DOMString        localName;
  };

  interface DocumentNS {
    Element            createElementNS(in DOMString namespaceName,
                                       in DOMString qualifiedName)
                                        raises(DOMException);
    Attr               createAttributeNS(in DOMString namespaceName,
                                         in DOMString qualifiedName)
                                        raises(DOMException);
    NodeList           getElementsByTagNameNS(in DOMString namespaceName,
                                              in DOMString localName);
  };

  interface ElementNS {
    DOMString          getAttributeNS(in DOMString namespaceName,
                                      in DOMString localName);
    void               setAttributeNS(in DOMString namespaceName,
                                      in DOMString localName,
                                      in DOMString value)
                                        raises(DOMException);
    void               removeAttributeNS(in DOMString namespaceName,
                                         in DOMString localName)
                                        raises(DOMException);
    Attr               getAttributeNodeNS(in DOMString namespaceName,
                                          in DOMString localName);
    Attr               setAttributeNodeNS(in Attr newAttr)
                                        raises(DOMException);
    NodeList           getElementsByTagNameNS(in DOMString namespaceName,
                                              in DOMString localName);
  };

  interface NodeNS {
    readonly attribute DOMString        universalName;
    readonly attribute DOMString        namespaceName;
             attribute DOMString        prefix;
                                        // raises(DOMException) on setting

    readonly attribute DOMString        localName;
  };

};

#endif // _NAMESPACES_IDL_
```

C.3: Document Object Model Level 2 Stylesheets

stylesheets.idl:

```
// File: stylesheets.idl
#ifndef _STYLESHEETS_IDL_
#define _STYLESHEETS_IDL_

#include "dom.idl"

#pragma prefix "dom.w3c.org"
module stylesheets
{
  typedef dom::DOMString DOMString;
  typedef dom::Node Node;

  interface MediaList;

  interface StyleSheet {
    readonly attribute DOMString        type;
             attribute boolean          disabled;
    readonly attribute Node             ownerNode;
    readonly attribute StyleSheet       parentStyleSheet;
    readonly attribute DOMString        href;
    readonly attribute DOMString        title;
    readonly attribute MediaList        media;
  };
  interface StyleSheetList {
    readonly attribute unsigned long    length;
    StyleSheet         item(in unsigned long index);
  };

  interface MediaList {
             attribute DOMString        cssText;
                                        // raises(DOMException) on setting

    readonly attribute unsigned long    length;
    DOMString          item(in unsigned long index);
    void               delete(in DOMString oldMedium)
                                        raises(DOMException);
    void               append(in DOMString newMedium)
                                        raises(DOMException);
  };

  interface DocumentStyle {
    readonly attribute StyleSheetList   styleSheets;
  };

};

#endif // _STYLESHEETS_IDL_
```

C.4: Document Object Model Level 2 CSS

css.idl:

```
// File: css.idl
#ifndef _CSS_IDL_
#define _CSS_IDL_

#include "dom.idl"
#include "stylesheets.idl"

#pragma prefix "dom.w3c.org"
module css
{
  typedef dom stylesheets::DOMString DOMString;
  typedef dom stylesheets::MediaList MediaList;
  typedef dom stylesheets::float float;
  typedef dom stylesheets::StyleSheet StyleSheet;

  interface CSSRule;
  interface CSSStyleSheet;
  interface CSSStyleDeclaration;
  interface CSSValue;
  interface Counter;
  interface Rect;
  interface RGBColor;

  interface CSSRuleList {
    readonly attribute unsigned long    length;
    CSSRule            item(in unsigned long index);
  };

  interface CSSRule {
    // RuleType
    const unsigned short      UNKNOWN_RULE                   = 0;
    const unsigned short      STYLE_RULE                     = 1;
    const unsigned short      CHARSET_RULE                   = 2;
    const unsigned short      IMPORT_RULE                    = 3;
    const unsigned short      MEDIA_RULE                     = 4;
    const unsigned short      FONT_FACE_RULE                 = 5;
    const unsigned short      PAGE_RULE                      = 6;

    readonly attribute unsigned short   type;
             attribute DOMString        cssText;
                                        // raises(DOMException) on setting

    readonly attribute CSSStyleSheet    parentStyleSheet;
    readonly attribute CSSRule          parentRule;
  };

  interface CSSStyleRule : CSSRule {
             attribute DOMString        selectorText;
                                        // raises(DOMException) on setting

    readonly attribute CSSStyleDeclaration  style;
  };

  interface CSSMediaRule : CSSRule {
    readonly attribute MediaList        media;
    readonly attribute CSSRuleList      cssRules;
    unsigned long      insertRule(in DOMString rule,
                                  in unsigned long index)
                                        raises(DOMException);
    void               deleteRule(in unsigned long index)
                                        raises(DOMException);
  };

  interface CSSFontFaceRule : CSSRule {
    readonly attribute CSSStyleDeclaration  style;
  };

  interface CSSPageRule : CSSRule {
             attribute DOMString        selectorText;
                                        // raises(DOMException) on setting

    readonly attribute CSSStyleDeclaration  style;
  };

  interface CSSImportRule : CSSRule {
    readonly attribute DOMString        href;
    readonly attribute MediaList        media;
    readonly attribute CSSStyleSheet    styleSheet;
  };

  interface CSSCharsetRule : CSSRule {
             attribute DOMString        encoding;
```

```
                                    // raises(DOMException) on setting

};

interface CSSUnknownRule : CSSRule {
};

interface CSSStyleDeclaration {
           attribute DOMString        cssText;
                                    // raises(DOMException) on setting

  DOMString          getPropertyValue(in DOMString propertyName);
  CSSValue           getPropertyCSSValue(in DOMString propertyName);
  DOMString          removeProperty(in DOMString propertyName)
                                    raises(DOMException);
  DOMString          getPropertyPriority(in DOMString propertyName);
  void               setProperty(in DOMString propertyName,
                                 in DOMString value,
                                 in DOMString priority)
                                    raises(DOMException);
  readonly attribute unsigned long    length;
  DOMString          item(in unsigned long index);
  readonly attribute CSSRule          parentRule;
};

interface CSSValue {
  // UnitTypes
  const unsigned short      CSS_PRIMITIVE_VALUE            = 0;
  const unsigned short      CSS_VALUE_LIST                 = 1;
  const unsigned short      CSS_CUSTOM                     = 2;

           attribute DOMString        cssText;
                                    // raises(DOMException) on setting

  readonly attribute unsigned short   valueType;
};

interface CSSPrimitiveValue : CSSValue {
  // UnitTypes
  const unsigned short      CSS_UNKNOWN                    = 0;
  const unsigned short      CSS_INHERIT                    = 1;
  const unsigned short      CSS_NUMBER                     = 2;
  const unsigned short      CSS_PERCENTAGE                 = 3;
  const unsigned short      CSS_EMS                        = 4;
  const unsigned short      CSS_EXS                        = 5;
  const unsigned short      CSS_PX                         = 6;
  const unsigned short      CSS_CM                         = 7;
  const unsigned short      CSS_MM                         = 8;
  const unsigned short      CSS_IN                         = 9;
  const unsigned short      CSS_PT                         = 10;
  const unsigned short      CSS_PC                         = 11;
  const unsigned short      CSS_DEG                        = 12;
  const unsigned short      CSS_RAD                        = 13;
  const unsigned short      CSS_GRAD                       = 14;
  const unsigned short      CSS_MS                         = 15;
  const unsigned short      CSS_S                          = 16;
  const unsigned short      CSS_HZ                         = 17;
```

```
    const unsigned short      CSS_KHZ                        = 18;
    const unsigned short      CSS_DIMENSION                  = 19;
    const unsigned short      CSS_STRING                     = 20;
    const unsigned short      CSS_URI                        = 21;
    const unsigned short      CSS_IDENT                      = 22;
    const unsigned short      CSS_ATTR                       = 23;
    const unsigned short      CSS_COUNTER                    = 24;
    const unsigned short      CSS_RECT                       = 26;
    const unsigned short      CSS_RGBCOLOR                   = 27;

    readonly attribute unsigned short   primitiveType;
    void               setFloatValue(in unsigned short unitType,
                                     in float floatValue)
                                        raises(DOMException);
    float              getFloatValue(in unsigned short unitType)
                                        raises(DOMException);
    void               setStringValue(in unsigned short stringType,
                                      in DOMString stringValue)
                                        raises(DOMException);
    DOMString          getStringValue()
                                        raises(DOMException);
    Counter            getCounterValue()
                                        raises(DOMException);
    Rect               getRectValue()
                                        raises(DOMException);
    RGBColor           getRGBColorValue()
                                        raises(DOMException);
  };

  interface CSSValueList : CSSValue {
    readonly attribute unsigned long    length;
    CSSValue           item(in unsigned long index);
  };

  interface RGBColor {
             attribute CSSValue         red;
             attribute CSSValue         green;
             attribute CSSValue         blue;
  };

  interface Rect {
             attribute CSSValue         top;
             attribute CSSValue         right;
             attribute CSSValue         bottom;
             attribute CSSValue         left;
  };

  interface Counter {
             attribute DOMString        identifier;
             attribute DOMString        listStyle;
             attribute DOMString        separator;
  };

  interface CSS2Azimuth : CSSValue {
    readonly attribute unsigned short   azimuthType;
    readonly attribute DOMString        identifier;
    readonly attribute boolean          behind;
    void               setAngleValue(in unsigned short unitType,
                                     in float floatValue)
                                        raises(DOMException);
    float              getAngleValue(in unsigned short unitType)
                                        raises(DOMException);
    void               setIdentifier(in DOMString identifier,
                                     in boolean behind)
                                        raises(DOMException);
  };

  interface CSS2BackgroundPosition : CSSValue {
    readonly attribute unsigned short   horizontalType;
    readonly attribute unsigned short   verticalType;
    readonly attribute DOMString        horizontalIdentifier;
    readonly attribute DOMString        verticalIdentifier;
    float              getHorizontalPosition(in float horizontalType)
                                        raises(DOMException);
    float              getVerticalPosition(in float verticalType)
                                        raises(DOMException);
    void               setHorizontalPosition(in unsigned short horizontalType,
                                             in float value)
                                        raises(DOMException);
    void               setVerticalPosition(in unsigned short verticalType,
                                           in float value)
                                        raises(DOMException);
    void               setPositionIdentifier(in DOMString horizontalIdentifier,
                                             in DOMString verticalIdentifier)
                                        raises(DOMException);
  };

  interface CSS2BorderSpacing : CSSValue {
    readonly attribute unsigned short   horizontalType;
    readonly attribute unsigned short   verticalType;
    float              getHorizontalSpacing(in float horizontalType)
                                        raises(DOMException);
    float              getVerticalSpacing(in float verticalType)
                                        raises(DOMException);
    void               setHorizontalSpacing(in unsigned short horizontalType,
                                            in float value)
                                        raises(DOMException);
    void               setVerticalSpacing(in unsigned short verticalType,
                                          in float value)
                                        raises(DOMException);
    void               setInherit()();
  };

  interface CSS2CounterReset {
             attribute DOMString        identifier;
                                        // raises(DOMException) on setting

             attribute short            reset;
                                        // raises(DOMException) on setting

  };

  interface CSS2CounterIncrement {
             attribute DOMString        identifier;
```

```
                                     // raises(DOMException) on setting

           attribute short            increment;
                                     // raises(DOMException) on setting

};

interface CSS2Cursor : CSSValue {
           attribute unsigned short   cursorType;
  readonly attribute CSSValueList     uris;
           attribute DOMString        predefinedCursor;
                                     // raises(DOMException) on setting

};

interface CSS2PlayDuring : CSSValue {
  readonly attribute unsigned short   playDuringType;
           attribute DOMString        playDuringIdentifier;
                                     // raises(DOMException) on setting

           attribute DOMString        uri;
                                     // raises(DOMException) on setting

           attribute boolean          mix;
                                     // raises(DOMException) on setting

           attribute boolean          repeat;
                                     // raises(DOMException) on setting

};

interface CSS2TextShadow {
  readonly attribute CSSValue         color;
  readonly attribute CSSValue         horizontal;
  readonly attribute CSSValue         vertical;
  readonly attribute CSSValue         blur;
};

interface CSS2FontFaceSrc {
           attribute DOMString        uri;
                                     // raises(DOMException) on setting

  readonly attribute CSSValueList     format;
           attribute DOMString        fontFaceName;
                                     // raises(DOMException) on setting

};

interface CSS2FontFaceWidths {
           attribute DOMString        urange;
                                     // raises(DOMException) on setting

  readonly attribute CSSValueList     numbers;
};

interface CSS2PageSize : CSSValue {
  readonly attribute unsigned short   widthType;
  readonly attribute unsigned short   heightType;
  readonly attribute DOMString        identifier;
  float              getWidth(in float widthType)
                                      raises(DOMException);
  float              getHeightSize(in float heightType)
                                      raises(DOMException);
  void               setWidthSize(in unsigned short widthType,
                                  in float value)
                                      raises(DOMException);
  void               setHeightSize(in unsigned short heightType,
                                   in float value)
                                      raises(DOMException);
  void               setIdentifier(in DOMString identifier)
                                      raises(DOMException);
};

interface CSS2Properties {
           attribute DOMString        azimuth;
           attribute DOMString        background;
           attribute DOMString        backgroundAttachment;
           attribute DOMString        backgroundColor;
           attribute DOMString        backgroundImage;
           attribute DOMString        backgroundPosition;
           attribute DOMString        backgroundRepeat;
           attribute DOMString        border;
           attribute DOMString        borderCollapse;
           attribute DOMString        borderColor;
           attribute DOMString        borderSpacing;
           attribute DOMString        borderStyle;
           attribute DOMString        borderTop;
           attribute DOMString        borderRight;
           attribute DOMString        borderBottom;
           attribute DOMString        borderLeft;
           attribute DOMString        borderTopColor;
           attribute DOMString        borderRightColor;
           attribute DOMString        borderBottomColor;
           attribute DOMString        borderLeftColor;
           attribute DOMString        borderTopStyle;
           attribute DOMString        borderRightStyle;
           attribute DOMString        borderBottomStyle;
           attribute DOMString        borderLeftStyle;
           attribute DOMString        borderTopWidth;
           attribute DOMString        borderRightWidth;
           attribute DOMString        borderBottomWidth;
           attribute DOMString        borderLeftWidth;
           attribute DOMString        borderWidth;
           attribute DOMString        bottom;
           attribute DOMString        captionSide;
           attribute DOMString        clear;
           attribute DOMString        clip;
           attribute DOMString        color;
           attribute DOMString        content;
           attribute DOMString        counterIncrement;
           attribute DOMString        counterReset;
           attribute DOMString        cue;
           attribute DOMString        cueAfter;
           attribute DOMString        cueBefore;
```

```
    attribute DOMString        cursor;
    attribute DOMString        direction;
    attribute DOMString        display;
    attribute DOMString        elevation;
    attribute DOMString        emptyCells;
    attribute DOMString        cssFloat;
    attribute DOMString        font;
    attribute DOMString        fontFamily;
    attribute DOMString        fontSize;
    attribute DOMString        fontSizeAdjust;
    attribute DOMString        fontStretch;
    attribute DOMString        fontStyle;
    attribute DOMString        fontVariant;
    attribute DOMString        fontWeight;
    attribute DOMString        height;
    attribute DOMString        left;
    attribute DOMString        letterSpacing;
    attribute DOMString        lineHeight;
    attribute DOMString        listStyle;
    attribute DOMString        listStyleImage;
    attribute DOMString        listStylePosition;
    attribute DOMString        listStyleType;
    attribute DOMString        margin;
    attribute DOMString        marginTop;
    attribute DOMString        marginRight;
    attribute DOMString        marginBottom;
    attribute DOMString        marginLeft;
    attribute DOMString        markerOffset;
    attribute DOMString        marks;
    attribute DOMString        maxHeight;
    attribute DOMString        maxWidth;
    attribute DOMString        minHeight;
    attribute DOMString        minWidth;
    attribute DOMString        orphans;
    attribute DOMString        outline;
    attribute DOMString        outlineColor;
    attribute DOMString        outlineStyle;
    attribute DOMString        outlineWidth;
    attribute DOMString        overflow;
    attribute DOMString        padding;
    attribute DOMString        paddingTop;
    attribute DOMString        paddingRight;
    attribute DOMString        paddingBottom;
    attribute DOMString        paddingLeft;
    attribute DOMString        page;
    attribute DOMString        pageBreakAfter;
    attribute DOMString        pageBreakBefore;
    attribute DOMString        pageBreakInside;
    attribute DOMString        pause;
    attribute DOMString        pauseAfter;
    attribute DOMString        pauseBefore;
    attribute DOMString        pitch;
    attribute DOMString        pitchRange;
    attribute DOMString        playDuring;
    attribute DOMString        position;
    attribute DOMString        quotes;
    attribute DOMString        richness;
```

```
           attribute DOMString        right;
           attribute DOMString        size;
           attribute DOMString        speak;
           attribute DOMString        speakHeader;
           attribute DOMString        speakNumeral;
           attribute DOMString        speakPunctuation;
           attribute DOMString        speechRate;
           attribute DOMString        stress;
           attribute DOMString        tableLayout;
           attribute DOMString        textAlign;
           attribute DOMString        textDecoration;
           attribute DOMString        textIndent;
           attribute DOMString        textShadow;
           attribute DOMString        textTransform;
           attribute DOMString        top;
           attribute DOMString        unicodeBidi;
           attribute DOMString        verticalAlign;
           attribute DOMString        visibility;
           attribute DOMString        voiceFamily;
           attribute DOMString        volume;
           attribute DOMString        whiteSpace;
           attribute DOMString        widows;
           attribute DOMString        width;
           attribute DOMString        wordSpacing;
           attribute DOMString        zIndex;
  };

  interface CSSStyleSheet : StyleSheet {
    readonly attribute CSSRule          ownerRule;
    readonly attribute CSSRuleList      cssRules;
    unsigned long      insertRule(in DOMString rule,
                                  in unsigned long index)
                                        raises(DOMException);
    void               deleteRule(in unsigned long index)
                                        raises(DOMException);
  };

};

#endif // _CSS_IDL_
```

C.5: Document Object Model Level 2 Events

events.idl:

```
// File: events.idl
#ifndef _EVENTS_IDL_
#define _EVENTS_IDL_

#include "dom.idl"

#pragma prefix "dom.w3c.org"
module events
{
  typedef dom::DOMString DOMString;
  typedef dom::Node Node;

  interface EventListener;
  interface Event;

  interface EventTarget {
    void               addEventListener(in DOMString type,
                                        in EventListener listener,
                                        in boolean useCapture);
    void               removeEventListener(in DOMString type,
                                           in EventListener listener,
                                           in boolean useCapture);
  };

  interface EventListener {
    void               handleEvent(in Event event);
  };

  interface Event {
    // PhaseType
    const unsigned short      BUBBLING_PHASE                 = 1;
    const unsigned short      CAPTURING_PHASE                = 2;
    const unsigned short      AT_TARGET                      = 3;

             attribute DOMString        type;
             attribute Node             target;
             attribute Node             currentNode;
             attribute unsigned short   eventPhase;
    void               preventBubble();
    void               preventCapture();
    void               preventDefault();
  };

  interface UIEvent : Event {
    const int                 CHAR_UNDEFINED                 = 1;
    const int                 KEY_FIRST                      = 1;
    const int                 KEY_LAST                       = 1;
    const int                 VK_0                           = 1;
    const int                 VK_1                           = 1;
    const int                 VK_2                           = 1;
    const int                 VK_3                           = 1;
    const int                 VK_4                           = 1;
    const int                 VK_5                           = 1;
    const int                 VK_6                           = 1;
    const int                 VK_7                           = 1;
    const int                 VK_8                           = 1;
    const int                 VK_9                           = 1;
    const int                 VK_A                           = 1;
    const int                 VK_ACCEPT                      = 1;
    const int                 VK_ADD                         = 1;
    const int                 VK_AGAIN                       = 1;
    const int                 VK_ALL_CANDIDATES              = 1;
    const int                 VK_ALPHANUMERIC                = 1;
    const int                 VK_ALT                         = 1;
    const int                 VK_ALT_GRAPH                   = 1;
    const int                 VK_AMPERSAND                   = 1;
    const int                 VK_ASTERISK                    = 1;
    const int                 VK_AT                          = 1;
    const int                 VK_B                           = 1;
    const int                 VK_BACK_QUOTE                  = 1;
    const int                 VK_BACK_SLASH                  = 1;
    const int                 VK_BACK_SPACE                  = 1;
    const int                 VK_BRACELEFT                   = 1;
    const int                 VK_BRACERIGHT                  = 1;
    const int                 VK_C                           = 1;
    const int                 VK_CANCEL                      = 1;
    const int                 VK_CAPS_LOCK                   = 1;
    const int                 VK_CIRCUMFLEX                  = 1;
    const int                 VK_CLEAR                       = 1;
    const int                 VK_CLOSE_BRACKET               = 1;
    const int                 VK_CODE_INPUT                  = 1;
    const int                 VK_COLON                       = 1;
    const int                 VK_COMMA                       = 1;
    const int                 VK_COMPOSE                     = 1;
    const int                 VK_CONTROL                     = 1;
    const int                 VK_CONVERT                     = 1;
    const int                 VK_COPY                        = 1;
    const int                 VK_CUT                         = 1;
    const int                 VK_D                           = 1;
    const int                 VK_DEAD_ABOVEDOT               = 1;
    const int                 VK_DEAD_ABOVERING              = 1;
    const int                 VK_DEAD_ACUTE                  = 1;
    const int                 VK_DEAD_BREVE                  = 1;
    const int                 VK_DEAD_CARON                  = 1;
    const int                 VK_DEAD_CEDILLA                = 1;
    const int                 VK_DEAD_CIRCUMFLEX             = 1;
    const int                 VK_DEAD_DIAERESIS              = 1;
    const int                 VK_DEAD_DOUBLEACUTE            = 1;
    const int                 VK_DEAD_GRAVE                  = 1;
    const int                 VK_DEAD_IOTA                   = 1;
    const int                 VK_DEAD_MACRON                 = 1;
    const int                 VK_DEAD_OGONEK                 = 1;
    const int                 VK_DEAD_SEMIVOICED_SOUND       = 1;
    const int                 VK_DEAD_TILDE                  = 1;
    const int                 VK_DEAD_VOICED_SOUND           = 1;
    const int                 VK_DECIMAL                     = 1;
    const int                 VK_DELETE                      = 1;
    const int                 VK_DIVIDE                      = 1;
    const int                 VK_DOLLAR                      = 1;
    const int                 VK_DOWN                        = 1;
    const int                 VK_E                           = 1;
    const int                 VK_END                         = 1;
    const int                 VK_ENTER                       = 1;
    const int                 VK_EQUALS                      = 1;
    const int                 VK_ESCAPE                      = 1;
    const int                 VK_EURO_SIGN                   = 1;
    const int                 VK_EXCLAMATION_MARK            = 1;
    const int                 VK_F                           = 1;
    const int                 VK_F1                          = 1;
    const int                 VK_F10                         = 1;
    const int                 VK_F11                         = 1;
    const int                 VK_F12                         = 1;
    const int                 VK_F13                         = 1;
    const int                 VK_F14                         = 1;
    const int                VK_F15                         = 1;
    const int                VK_F16                         = 1;
    const int                VK_F17                         = 1;
    const int                VK_F18                         = 1;
    const int                VK_F19                         = 1;
    const int                VK_F2                          = 1;
    const int                VK_F20                         = 1;
    const int                VK_F21                         = 1;
    const int                VK_F22                         = 1;
    const int                VK_F23                         = 1;
    const int                VK_F24                         = 1;
    const int                VK_F3                          = 1;
    const int                VK_F4                          = 1;
    const int                VK_F5                          = 1;
    const int                VK_F6                          = 1;
    const int                VK_F7                          = 1;
    const int                VK_F8                          = 1;
    const int                VK_F9                          = 1;
    const int                VK_FINAL                       = 1;
    const int                VK_FIND                        = 1;
    const int                VK_FULL_WIDTH                  = 1;
    const int                VK_G                           = 1;
    const int                VK_GREATER                     = 1;
    const int                VK_H                           = 1;
    const int                VK_HALF_WIDTH                  = 1;
    const int                VK_HELP                        = 1;
    const int                VK_HIRAGANA                    = 1;
    const int                VK_HOME                        = 1;
    const int                VK_I                           = 1;
    const int                VK_INSERT                      = 1;
    const int                VK_INVERTED_EXCLAMATION_MARK   = 1;
    const int                VK_J                           = 1;
    const int                VK_JAPANESE_HIRAGANA           = 1;
    const int                VK_JAPANESE_KATAKANA           = 1;
    const int                VK_JAPANESE_ROMAN              = 1;
    const int                VK_K                           = 1;
    const int                VK_KANA                        = 1;
    const int                VK_KANJI                       = 1;
    const int                VK_KATAKANA                    = 1;
    const int                VK_KP_DOWN                     = 1;
    const int                VK_KP_LEFT                     = 1;
    const int                VK_KP_RIGHT                    = 1;
    const int                VK_KP_UP                       = 1;
    const int                VK_L                           = 1;
    const int                VK_LEFT                        = 1;
    const int                VK_LEFT_PARENTHESIS            = 1;
    const int                VK_LESS                        = 1;
    const int                VK_M                           = 1;
    const int                VK_META                        = 1;
    const int                VK_MINUS                       = 1;
    const int                VK_MODECHANGE                  = 1;
    const int                VK_MULTIPLY                    = 1;
    const int                VK_N                           = 1;
    const int                VK_NONCONVERT                  = 1;
    const int                VK_NUM_LOCK                    = 1;
    const int                VK_NUMBER_SIGN                 = 1;
    const int                VK_NUMPAD0                     = 1;
    const int                  VK_NUMPAD1                    = 1;
    const int                  VK_NUMPAD2                    = 1;
    const int                  VK_NUMPAD3                    = 1;
    const int                  VK_NUMPAD4                    = 1;
    const int                  VK_NUMPAD5                    = 1;
    const int                  VK_NUMPAD6                    = 1;
    const int                  VK_NUMPAD7                    = 1;
    const int                  VK_NUMPAD8                    = 1;
    const int                  VK_NUMPAD9                    = 1;
    const int                  VK_O                          = 1;
    const int                  VK_OPEN_BRACKET               = 1;
    const int                  VK_P                          = 1;
    const int                  VK_PAGE_DOWN                  = 1;
    const int                  VK_PAGE_UP                    = 1;
    const int                  VK_PASTE                      = 1;
    const int                  VK_PAUSE                      = 1;
    const int                  VK_PERIOD                     = 1;
    const int                  VK_PLUS                       = 1;
    const int                  VK_PREVIOUS_CANDIDATE         = 1;
    const int                  VK_PRINTSCREEN                = 1;
    const int                  VK_PROPS                      = 1;
    const int                  VK_Q                          = 1;
    const int                  VK_QUOTE                      = 1;
    const int                  VK_QUOTEDBL                   = 1;
    const int                  VK_R                          = 1;
    const int                  VK_RIGHT                      = 1;
    const int                  VK_RIGHT_PARENTHESIS          = 1;
    const int                  VK_ROMAN_CHARACTERS           = 1;
    const int                  VK_S                          = 1;
    const int                  VK_SCROLL_LOCK                = 1;
    const int                  VK_SEMICOLON                  = 1;
    const int                  VK_SEPARATER                  = 1;
    const int                  VK_SHIFT                      = 1;
    const int                  VK_SLASH                      = 1;
    const int                  VK_SPACE                      = 1;
    const int                  VK_STOP                       = 1;
    const int                  VK_SUBTRACT                   = 1;
    const int                  VK_T                          = 1;
    const int                  VK_TAB                        = 1;
    const int                  VK_U                          = 1;
    const int                  VK_UNDEFINED                  = 1;
    const int                  VK_UNDERSCORE                 = 1;
    const int                  VK_UNDO                       = 1;
    const int                  VK_UP                         = 1;
    const int                  VK_V                          = 1;
    const int                  VK_W                          = 1;
    const int                  VK_X                          = 1;
    const int                  VK_Y                          = 1;
    const int                  VK_Z                          = 1;
             attribute long             screenX;
             attribute long             screenY;
             attribute long             clientX;
             attribute long             clientY;
             attribute boolean          ctrlKey;
             attribute boolean          shiftKey;
             attribute boolean          altKey;
             attribute boolean          metaKey;
```

```
            attribute unsigned long    keyCode;
            attribute unsigned long    charCode;
            attribute unsigned short   button;
            attribute unsigned short   clickCount;
  };

  interface MutationEvent : Event {
            attribute Node             relatedNode;
            attribute DOMString        prevValue;
            attribute DOMString        newValue;
            attribute DOMString        attrName;
  };

};

#endif // _EVENTS_IDL_
```

C.6: Document Object Model Level 2 Filters and Iterators

fi.idl:

```
// File: fi.idl
#ifndef _FI_IDL_
#define _FI_IDL_

#include "dom.idl"

#pragma prefix "dom.w3c.org"
module fi
{
  typedef dom::Node Node;

  interface NodeFilter;

  interface NodeIterator {
    readonly attribute long             whatToShow;
    // Constants for whatToShow
    const unsigned long       SHOW_ALL                       = 0xFFFF;
    const unsigned long       SHOW_ELEMENT                   = 0x00000001;
    const unsigned long       SHOW_ATTRIBUTE                 = 0x00000002;
    const unsigned long       SHOW_TEXT                      = 0x00000004;
    const unsigned long       SHOW_CDATA_SECTION             = 0x00000008;
    const unsigned long       SHOW_ENTITY_REFERENCE          = 0x00000010;
    const unsigned long       SHOW_ENTITY                    = 0x00000020;
    const unsigned long       SHOW_PROCESSING_INSTRUCTION    = 0x00000040;
    const unsigned long       SHOW_COMMENT                   = 0x00000080;
    const unsigned long       SHOW_DOCUMENT                  = 0x00000100;
    const unsigned long       SHOW_DOCUMENT_TYPE             = 0x00000200;
    const unsigned long       SHOW_DOCUMENT_FRAGMENT         = 0x00000400;
    const unsigned long       SHOW_NOTATION                  = 0x00000800;

    readonly attribute NodeFilter       filter;
    Node               nextNode();
    Node               previousNode();
  };
  interface NodeFilter {
    // Constants returned by acceptNode
    const short               FILTER_ACCEPT                  = 1;
    const short               FILTER_REJECT                  = 2;
    const short               FILTER_SKIP                    = 3;

    short              acceptNode(in Node n);
  };

  interface TreeWalker {
    readonly attribute long            whatToShow;
    // Constants for whatToShow
    const unsigned long       SHOW_ALL                       = 0xFFFF;
    const unsigned long       SHOW_ELEMENT                   = 0x00000001;
    const unsigned long       SHOW_ATTRIBUTE                 = 0x00000002;
    const unsigned long       SHOW_TEXT                      = 0x00000004;
    const unsigned long       SHOW_CDATA_SECTION             = 0x00000008;
    const unsigned long       SHOW_ENTITY_REFERENCE          = 0x00000010;
    const unsigned long       SHOW_ENTITY                    = 0x00000020;
    const unsigned long       SHOW_PROCESSING_INSTRUCTION    = 0x00000040;
    const unsigned long       SHOW_COMMENT                   = 0x00000080;
    const unsigned long       SHOW_DOCUMENT                  = 0x00000100;
    const unsigned long       SHOW_DOCUMENT_TYPE             = 0x00000200;
    const unsigned long       SHOW_DOCUMENT_FRAGMENT         = 0x00000400;
    const unsigned long       SHOW_NOTATION                  = 0x00000800;

    readonly attribute NodeFilter      filter;
    Node               current();
    Node               parentNode();
    Node               firstChild();
    Node               lastChild();
    Node               previousSibling();
    Node               nextSibling();
  };

  interface DocumentIF {
    short              createNodeIterator(in Node root,
                                          in short whatToShow,
                                          in NodeFilter filter);
  };

};

#endif // _FI_IDL_
```

C.7: Document Object Model Level 2 Range

range.idl:

```
// File: range.idl
#ifndef _RANGE_IDL_
#define _RANGE_IDL_

#include "dom.idl"
```

```
#pragma prefix "dom.w3c.org"
module range
{
  typedef dom::Node Node;
  typedef dom::DocumentFragment DocumentFragment;
  typedef dom::DOMString DOMString;

  exception RangeException {
    unsigned short   code;
  };

  // RangeExceptionCode
  const unsigned short      BAD_ENDPOINTS_ERR              = 201;
  const unsigned short      INVALID_NODE_TYPE_ERR          = 202;
  const unsigned short      NULL_NODE_ERR                  = 203;


  interface Range {
    readonly attribute Node             startContainer;
    readonly attribute long             startOffset;
    readonly attribute Node             endContainer;
    readonly attribute long             endOffset;
    readonly attribute boolean          isCollapsed;
    readonly attribute Node             commonAncestorContainer;
    void               setStart(in Node node,
                                 in long offset)
                                        raises(RangeException);
    void               setEnd(in Node node,
                               in long offset)
                                        raises(RangeException);
    void               setStartBefore(in Node node)
                                        raises(RangeException);
    void               setStartAfter(in Node node)
                                        raises(RangeException);
    void               setEndBefore(in Node node)
                                        raises(RangeException);
    void               setEndAfter(in Node node)
                                        raises(RangeException);
    void               collapse(in boolean toStart);
    void               selectNode(in Node node)
                                        raises(RangeException);
    void               selectNodeContents(in Node node)
                                        raises(RangeException);
    typedef enum CompareHow_ {
      StartToStart,
      StartToEnd,
      EndToEnd,
      EndToStart
    } CompareHow;
    short              compareEndPoints(in CompareHow how,
                                        in Range sourceRange)
                                        raises(DOMException);
    void               deleteContents()
                                        raises(DOMException);
    DocumentFragment   extractContents()
                                        raises(DOMException);
    DocumentFragment   cloneContents()
                                raises(DOMException);
    void               insertNode(in Node node)
                                raises(DOMException, RangeException);
    void               surroundContents(in Node node)
                                raises(DOMException, RangeException);
    Range              cloneRange();
    DOMString          toString();
  };

};

#endif // _RANGE_IDL_
```

Appendix D: Java Language Binding

This appendix contains the complete Java bindings for the Level 2 Document Object Model. The definitions are divided into Core [p.177] , Namespaces [p.178] , Stylesheets [p.179] , CSS [p.180] , Events [p.194] , Filters and Iterators [p.200] , and Range [p.201] .

The Java files are also available as
http://www.w3.org/TR/1999/WD-DOM-Level-2-19990719/java-binding.zip

D.1: Document Object Model Level 2 Core

org/w3c/dom/DocumentType2.java:

```
package org.w3c.dom;

public interface DocumentType2 extends DocumentType {
  public String             getPublicID();
  public String             getSystemID();
}
```

org/w3c/dom/DOMImplementation2.java:

```
package org.w3c.dom;

public interface DOMImplementation2 extends DOMImplementation {
  public DocumentType       createDocumentType(String name, 
                                                   String publicID, 
                                                   String systemID)
                                                   throws DOMException;
  public Document           createDocument(String name, 
                                               DocumentType doctype)
                                               throws DOMException;
}
```

org/w3c/dom/Document2.java:

```
package org.w3c.dom;

public interface Document2 extends Document {
  public Node               importNode(Node importedNode, 
                                           boolean deep);
}
```

org/w3c/dom/Node2.java:

```
package org.w3c.dom;

public interface Node2 extends Node {
  public boolean            supports(String feature, 
                                         String version);
}
```

org/w3c/dom/Attr2.java:

```
package org.w3c.dom;

public interface Attr2 extends Attr {
  public Element            getOwnerElement();
}
```

org/w3c/dom/HTMLDOMImplementation.java:

```
package org.w3c.dom;

public interface HTMLDOMImplementation extends DOMImplementation {
  public HTMLDocument       createHTMLDocument(String title);
}
```

D.2: Document Object Model Level 2 Namespaces

org/w3c/dom/namespaces/NodeNS.java:

```
package org.w3c.dom.namespaces;

import org.w3c.dom.*;

public interface NodeNS {
  public String             getNamespaceName();
  public String             getPrefix();
  public void               setPrefix(String prefix)
                                                   throws DOMException;
  public String             getLocalName();
}
```

org/w3c/dom/namespaces/DocumentNS.java:

```
package org.w3c.dom.namespaces;

import org.w3c.dom.*;

public interface DocumentNS {
  public Element            createElementNS(String namespaceName,
                                            String qualifiedName)
                                            throws DOMException;
  public Attr               createAttributeNS(String namespaceName,
                                              String qualifiedName)
                                              throws DOMException;
  public NodeList           getElementsByTagNameNS(String namespaceName,
                                                   String localName);
}
```

org/w3c/dom/namespaces/ElementNS.java:

```
package org.w3c.dom.namespaces;

import org.w3c.dom.*;

public interface ElementNS {
  public String             getAttributeNS(String namespaceName,
                                           String localName);
  public void               setAttributeNS(String namespaceName,
                                           String localName,
                                           String value)
                                           throws DOMException;
  public void               removeAttributeNS(String namespaceName,
                                              String localName)
                                              throws DOMException;
  public Attr               getAttributeNodeNS(String namespaceName,
                                               String localName);
  public Attr               setAttributeNodeNS(Attr newAttr)
                                               throws DOMException;
  public NodeList           getElementsByTagNameNS(String namespaceName,
                                                   String localName);
}
```

org/w3c/dom/namespaces/NodeNS.java:

```
package org.w3c.dom.namespaces;

import org.w3c.dom.*;

public interface NodeNS {
  public String             getUniversalName();
  public String             getNamespaceName();
  public String             getPrefix();
  public void               setPrefix(String prefix)
                                                   throws DOMException;
  public String             getLocalName();
}
```

D.3: Document Object Model Level 2 Stylesheets

org/w3c/dom/stylesheets/StyleSheet.java:

```
package org.w3c.dom.stylesheets;

import org.w3c.dom.*;

public interface StyleSheet {
  public String             getType();
  public boolean            getDisabled();
  public void               setDisabled(boolean disabled);
  public Node               getOwnerNode();
  public StyleSheet         getParentStyleSheet();
  public String             getHref();
  public String             getTitle();
  public MediaList          getMedia();
}
```

org/w3c/dom/stylesheets/StyleSheetList.java:

```
package org.w3c.dom.stylesheets;

import org.w3c.dom.*;

public interface StyleSheetList {
  public int                getLength();
  public StyleSheet         item(int index);
}
```

org/w3c/dom/stylesheets/MediaList.java:

```
package org.w3c.dom.stylesheets;

import org.w3c.dom.*;

public interface MediaList {
  public String             getCssText();
  public void               setCssText(String cssText)
                                 throws DOMException;
  public int                getLength();
  public String             item(int index);
  public void               delete(String oldMedium)
                                   throws DOMException;
  public void               append(String newMedium)
                                   throws DOMException;
}
```

org/w3c/dom/stylesheets/DocumentStyle.java:

```
package org.w3c.dom.stylesheets;

import org.w3c.dom.*;

public interface DocumentStyle {
  public StyleSheetList     getStyleSheets();
}
```

D.4: Document Object Model Level 2 CSS

org/w3c/dom/css/CSSStyleSheet.java:

```
package org.w3c.dom.css;

import org.w3c.dom.*;
import org.w3c.dom.stylesheets.*;
```

```
public interface CSSStyleSheet extends StyleSheet {
  public CSSRule            getOwnerRule();
  public CSSRuleList        getCssRules();
  public int                insertRule(String rule,
                                       int index)
                                       throws DOMException;
  public void               deleteRule(int index)
                                       throws DOMException;
}
```

org/w3c/dom/css/CSSRuleList.java:

```
package org.w3c.dom.css;

import org.w3c.dom.*;
import org.w3c.dom.stylesheets.*;

public interface CSSRuleList {
  public int                getLength();
  public CSSRule            item(int index);
}
```

org/w3c/dom/css/CSSRule.java:

```
package org.w3c.dom.css;

import org.w3c.dom.*;
import org.w3c.dom.stylesheets.*;

public interface CSSRule {
  // RuleType
  public static final short           UNKNOWN_RULE         = 0;
  public static final short           STYLE_RULE           = 1;
  public static final short           CHARSET_RULE         = 2;
  public static final short           IMPORT_RULE          = 3;
  public static final short           MEDIA_RULE           = 4;
  public static final short           FONT_FACE_RULE       = 5;
  public static final short           PAGE_RULE            = 6;

  public short              getType();
  public String             getCssText();
  public void               setCssText(String cssText)
                                 throws DOMException;
  public CSSStyleSheet      getParentStyleSheet();
  public CSSRule            getParentRule();
}
```

org/w3c/dom/css/CSSStyleRule.java:

```
package org.w3c.dom.css;

import org.w3c.dom.*;
import org.w3c.dom.stylesheets.*;

public interface CSSStyleRule extends CSSRule {
  public String             getSelectorText();
  public void               setSelectorText(String selectorText)
                                 throws DOMException;
  public CSSStyleDeclaration getStyle();
}
```

org/w3c/dom/css/CSSMediaRule.java:

```
package org.w3c.dom.css;

import org.w3c.dom.*;
import org.w3c.dom.stylesheets.*;

public interface CSSMediaRule extends CSSRule {
  public MediaList          getMedia();
  public CSSRuleList        getCssRules();
  public int                insertRule(String rule,
                                       int index)
                                       throws DOMException;
  public void               deleteRule(int index)
                                       throws DOMException;
}
```

org/w3c/dom/css/CSSFontFaceRule.java:

```
package org.w3c.dom.css;

import org.w3c.dom.*;
import org.w3c.dom.stylesheets.*;

public interface CSSFontFaceRule extends CSSRule {
  public CSSStyleDeclaration getStyle();
}
```

org/w3c/dom/css/CSSPageRule.java:

```
package org.w3c.dom.css;

import org.w3c.dom.*;
import org.w3c.dom.stylesheets.*;

public interface CSSPageRule extends CSSRule {
  public String             getSelectorText();
  public void               setSelectorText(String selectorText)
                                       throws DOMException;
  public CSSStyleDeclaration getStyle();
}
```

org/w3c/dom/css/CSSImportRule.java:

```
package org.w3c.dom.css;

import org.w3c.dom.*;
import org.w3c.dom.stylesheets.*;
```

```
public interface CSSImportRule extends CSSRule {
  public String             getHref();
  public MediaList          getMedia();
  public CSSStyleSheet      getStyleSheet();
}
```

org/w3c/dom/css/CSSCharsetRule.java:

```
package org.w3c.dom.css;

import org.w3c.dom.*;
import org.w3c.dom.stylesheets.*;

public interface CSSCharsetRule extends CSSRule {
  public String             getEncoding();
  public void               setEncoding(String encoding)
                                       throws DOMException;
}
```

org/w3c/dom/css/CSSUnknownRule.java:

```
package org.w3c.dom.css;

import org.w3c.dom.*;
import org.w3c.dom.stylesheets.*;

public interface CSSUnknownRule extends CSSRule {
}
```

org/w3c/dom/css/CSSStyleDeclaration.java:

```
package org.w3c.dom.css;

import org.w3c.dom.*;
import org.w3c.dom.stylesheets.*;

public interface CSSStyleDeclaration {
  public String             getCssText();
  public void               setCssText(String cssText)
                                       throws DOMException;
  public String             getPropertyValue(String propertyName);
  public CSSValue           getPropertyCSSValue(String propertyName);
  public String             removeProperty(String propertyName)
                                           throws DOMException;
  public String             getPropertyPriority(String propertyName);
  public void               setProperty(String propertyName,
                                        String value,
                                        String priority)
                                        throws DOMException;
  public int                getLength();
  public String             item(int index);
  public CSSRule            getParentRule();
}
```

org/w3c/dom/css/CSSValue.java:

```
package org.w3c.dom.css;

import org.w3c.dom.*;
import org.w3c.dom.stylesheets.*;

public interface CSSValue {
  // UnitTypes
  public static final short           CSS_PRIMITIVE_VALUE  = 0;
  public static final short           CSS_VALUE_LIST       = 1;
  public static final short           CSS_CUSTOM           = 2;

  public String             getCssText();
  public void               setCssText(String cssText)
                                 throws DOMException;
  public short              getValueType();
}
```

org/w3c/dom/css/CSSPrimitiveValue.java:

```
package org.w3c.dom.css;

import org.w3c.dom.*;
import org.w3c.dom.stylesheets.*;

public interface CSSPrimitiveValue extends CSSValue {
  // UnitTypes
  public static final short           CSS_UNKNOWN          = 0;
  public static final short           CSS_INHERIT          = 1;
  public static final short           CSS_NUMBER           = 2;
  public static final short           CSS_PERCENTAGE       = 3;
  public static final short           CSS_EMS              = 4;
  public static final short           CSS_EXS              = 5;
  public static final short           CSS_PX               = 6;
  public static final short           CSS_CM               = 7;
  public static final short           CSS_MM               = 8;
  public static final short           CSS_IN               = 9;
  public static final short           CSS_PT               = 10;
  public static final short           CSS_PC               = 11;
  public static final short           CSS_DEG              = 12;
  public static final short           CSS_RAD              = 13;
  public static final short           CSS_GRAD             = 14;
  public static final short           CSS_MS               = 15;
  public static final short           CSS_S                = 16;
  public static final short           CSS_HZ               = 17;
  public static final short           CSS_KHZ              = 18;
  public static final short           CSS_DIMENSION        = 19;
  public static final short           CSS_STRING           = 20;
  public static final short           CSS_URI              = 21;
  public static final short           CSS_IDENT            = 22;
  public static final short           CSS_ATTR             = 23;
  public static final short           CSS_COUNTER          = 24;
  public static final short           CSS_RECT             = 26;
  public static final short           CSS_RGBCOLOR         = 27;
  public short              getPrimitiveType();
  public void               setFloatValue(short unitType,
                                          float floatValue)
                                          throws DOMException;
  public float              getFloatValue(short unitType)
                                          throws DOMException;
  public void               setStringValue(short stringType,
                                           String stringValue)
                                           throws DOMException;
  public String             getStringValue()
                                           throws DOMException;
  public Counter            getCounterValue()
                                            throws DOMException;
  public Rect               getRectValue()
                                         throws DOMException;
  public RGBColor           getRGBColorValue()
                                             throws DOMException;
}
```

org/w3c/dom/css/CSSValueList.java:

```
package org.w3c.dom.css;

import org.w3c.dom.*;
import org.w3c.dom.stylesheets.*;

public interface CSSValueList extends CSSValue {
  public int                getLength();
  public CSSValue           item(int index);
}
```

org/w3c/dom/css/RGBColor.java:

```
package org.w3c.dom.css;

import org.w3c.dom.*;
import org.w3c.dom.stylesheets.*;

public interface RGBColor {
  public CSSValue           getRed();
  public void               setRed(CSSValue red);
  public CSSValue           getGreen();
  public void               setGreen(CSSValue green);
  public CSSValue           getBlue();
  public void               setBlue(CSSValue blue);
}
```

org/w3c/dom/css/Rect.java:

```
package org.w3c.dom.css;

import org.w3c.dom.*;
import org.w3c.dom.stylesheets.*;

public interface Rect {
  public CSSValue           getTop();
  public void               setTop(CSSValue top);
  public CSSValue           getRight();
  public void               setRight(CSSValue right);
  public CSSValue           getBottom();
  public void               setBottom(CSSValue bottom);
  public CSSValue           getLeft();
  public void               setLeft(CSSValue left);
}
```

org/w3c/dom/css/Counter.java:

```
package org.w3c.dom.css;

import org.w3c.dom.*;
import org.w3c.dom.stylesheets.*;

public interface Counter {
  public String             getIdentifier();
  public void               setIdentifier(String identifier);
  public String             getListStyle();
  public void               setListStyle(String listStyle);
  public String             getSeparator();
  public void               setSeparator(String separator);
}
```

org/w3c/dom/css/CSS2Azimuth.java:

```
package org.w3c.dom.css;

import org.w3c.dom.*;
import org.w3c.dom.stylesheets.*;

public interface CSS2Azimuth extends CSSValue {
  public short              getAzimuthType();
  public String             getIdentifier();
  public boolean            getBehind();
  public void               setAngleValue(short unitType,
                                          float floatValue)
                                          throws DOMException;
  public float              getAngleValue(short unitType)
                                          throws DOMException;
  public void               setIdentifier(String identifier,
                                          boolean behind)
                                          throws DOMException;
}
```

org/w3c/dom/css/CSS2BackgroundPosition.java:

```
package org.w3c.dom.css;

import org.w3c.dom.*;
import org.w3c.dom.stylesheets.*;

public interface CSS2BackgroundPosition extends CSSValue {
  public short              getHorizontalType();
  public short              getVerticalType();
  public String             getHorizontalIdentifier();
  public String             getVerticalIdentifier();
  public float              getHorizontalPosition(float horizontalType)
                                                throws DOMException;
  public float              getVerticalPosition(float verticalType)
                                              throws DOMException;
  public void               setHorizontalPosition(short horizontalType,
                                                float value)
                                                throws DOMException;
  public void               setVerticalPosition(short verticalType,
                                              float value)
                                              throws DOMException;
  public void               setPositionIdentifier(String horizontalIdentifier,
                                                String verticalIdentifier)
                                                throws DOMException;
}
```

org/w3c/dom/css/CSS2BorderSpacing.java:

```
package org.w3c.dom.css;

import org.w3c.dom.*;
import org.w3c.dom.stylesheets.*;

public interface CSS2BorderSpacing extends CSSValue {
  public short              getHorizontalType();
  public short              getVerticalType();
  public float              getHorizontalSpacing(float horizontalType)
                                               throws DOMException;
  public float              getVerticalSpacing(float verticalType)
                                             throws DOMException;
  public void               setHorizontalSpacing(short horizontalType,
                                               float value)
                                               throws DOMException;
  public void               setVerticalSpacing(short verticalType,
                                             float value)
                                             throws DOMException;
  public void               setInherit()();
}
```

org/w3c/dom/css/CSS2CounterReset.java:

```
package org.w3c.dom.css;

import org.w3c.dom.*;
import org.w3c.dom.stylesheets.*;

public interface CSS2CounterReset {
  public String             getIdentifier();
  public void               setIdentifier(String identifier)
                                        throws DOMException;
  public short              getReset();
  public void               setReset(short reset)
                                         throws DOMException;
}
```

org/w3c/dom/css/CSS2CounterIncrement.java:

```
package org.w3c.dom.css;

import org.w3c.dom.*;
import org.w3c.dom.stylesheets.*;

public interface CSS2CounterIncrement {
  public String             getIdentifier();
  public void               setIdentifier(String identifier)
                                         throws DOMException;
  public short              getIncrement();
  public void               setIncrement(short increment)
                                         throws DOMException;
}
```

org/w3c/dom/css/CSS2Cursor.java:

```
package org.w3c.dom.css;

import org.w3c.dom.*;
import org.w3c.dom.stylesheets.*;

public interface CSS2Cursor extends CSSValue {
  public short              getCursorType();
  public void               setCursorType(short cursorType);
  public CSSValueList       getUris();
  public String             getPredefinedCursor();
  public void               setPredefinedCursor(String predefinedCursor)
                                         throws DOMException;
}
```

org/w3c/dom/css/CSS2PlayDuring.java:

```
package org.w3c.dom.css;

import org.w3c.dom.*;
import org.w3c.dom.stylesheets.*;

public interface CSS2PlayDuring extends CSSValue {
  public short              getPlayDuringType();
  public String             getPlayDuringIdentifier();
  public void               setPlayDuringIdentifier(String playDuringIdentifier)
                                         throws DOMException;
  public String             getUri();
  public void               setUri(String uri)
                                         throws DOMException;
  public boolean            getMix();
  public void               setMix(boolean mix)
                                         throws DOMException;
  public boolean            getRepeat();
  public void               setRepeat(boolean repeat)
                                         throws DOMException;
}
```

org/w3c/dom/css/CSS2TextShadow.java:

```
package org.w3c.dom.css;

import org.w3c.dom.*;
import org.w3c.dom.stylesheets.*;

public interface CSS2TextShadow {
  public CSSValue           getColor();
  public CSSValue           getHorizontal();
  public CSSValue           getVertical();
  public CSSValue           getBlur();
}
```

org/w3c/dom/css/CSS2FontFaceSrc.java:

```
package org.w3c.dom.css;

import org.w3c.dom.*;
import org.w3c.dom.stylesheets.*;

public interface CSS2FontFaceSrc {
  public String             getUri();
  public void               setUri(String uri)
                                         throws DOMException;
  public CSSValueList       getFormat();
  public String             getFontFaceName();
  public void               setFontFaceName(String fontFaceName)
                                         throws DOMException;
}
```

org/w3c/dom/css/CSS2FontFaceWidths.java:

```
package org.w3c.dom.css;

import org.w3c.dom.*;
import org.w3c.dom.stylesheets.*;

public interface CSS2FontFaceWidths {
  public String             getUrange();
  public void               setUrange(String urange)
                                         throws DOMException;
  public CSSValueList       getNumbers();
}
```

org/w3c/dom/css/CSS2PageSize.java:

```
package org.w3c.dom.css;

import org.w3c.dom.*;
import org.w3c.dom.stylesheets.*;

public interface CSS2PageSize extends CSSValue {
  public short              getWidthType();
  public short              getHeightType();
  public String             getIdentifier();
  public float              getWidth(float widthType)
                                     throws DOMException;
  public float              getHeightSize(float heightType)
                                          throws DOMException;
  public void               setWidthSize(short widthType,
                                         float value)
                                         throws DOMException;
  public void               setHeightSize(short heightType,
                                          float value)
                                          throws DOMException;
  public void               setIdentifier(String identifier)
                                          throws DOMException;
}
```

org/w3c/dom/css/CSS2Properties.java:

```
package org.w3c.dom.css;

import org.w3c.dom.*;
import org.w3c.dom.stylesheets.*;

public interface CSS2Properties {
  public String             getAzimuth();
  public void               setAzimuth(String azimuth);
  public String             getBackground();
  public void               setBackground(String background);
  public String             getBackgroundAttachment();
  public void               setBackgroundAttachment(String backgroundAttachment);
  public String             getBackgroundColor();
  public void               setBackgroundColor(String backgroundColor);
  public String             getBackgroundImage();
  public void               setBackgroundImage(String backgroundImage);
  public String             getBackgroundPosition();
  public void               setBackgroundPosition(String backgroundPosition);
  public String             getBackgroundRepeat();
  public void               setBackgroundRepeat(String backgroundRepeat);
  public String             getBorder();
  public void               setBorder(String border);
  public String             getBorderCollapse();
  public void               setBorderCollapse(String borderCollapse);
  public String             getBorderColor();
  public void               setBorderColor(String borderColor);
  public String             getBorderSpacing();
  public void               setBorderSpacing(String borderSpacing);
  public String             getBorderStyle();
  public void                setBorderStyle(String borderStyle);
  public String              getBorderTop();
  public void                setBorderTop(String borderTop);
  public String              getBorderRight();
  public void                setBorderRight(String borderRight);
  public String              getBorderBottom();
  public void                setBorderBottom(String borderBottom);
  public String              getBorderLeft();
  public void                setBorderLeft(String borderLeft);
  public String              getBorderTopColor();
  public void                setBorderTopColor(String borderTopColor);
  public String              getBorderRightColor();
  public void                setBorderRightColor(String borderRightColor);
  public String              getBorderBottomColor();
  public void                setBorderBottomColor(String borderBottomColor);
  public String              getBorderLeftColor();
  public void                setBorderLeftColor(String borderLeftColor);
  public String              getBorderTopStyle();
  public void                setBorderTopStyle(String borderTopStyle);
  public String              getBorderRightStyle();
  public void                setBorderRightStyle(String borderRightStyle);
  public String              getBorderBottomStyle();
  public void                setBorderBottomStyle(String borderBottomStyle);
  public String              getBorderLeftStyle();
  public void                setBorderLeftStyle(String borderLeftStyle);
  public String              getBorderTopWidth();
  public void                setBorderTopWidth(String borderTopWidth);
  public String              getBorderRightWidth();
  public void                setBorderRightWidth(String borderRightWidth);
  public String              getBorderBottomWidth();
  public void                setBorderBottomWidth(String borderBottomWidth);
  public String              getBorderLeftWidth();
  public void                setBorderLeftWidth(String borderLeftWidth);
  public String              getBorderWidth();
  public void                setBorderWidth(String borderWidth);
  public String              getBottom();
  public void                setBottom(String bottom);
  public String              getCaptionSide();
  public void                setCaptionSide(String captionSide);
  public String              getClear();
  public void                setClear(String clear);
  public String              getClip();
  public void                setClip(String clip);
  public String              getColor();
  public void                setColor(String color);
  public String              getContent();
  public void                setContent(String content);
  public String              getCounterIncrement();
  public void                setCounterIncrement(String counterIncrement);
  public String              getCounterReset();
  public void                setCounterReset(String counterReset);
  public String              getCue();
  public void                setCue(String cue);
  public String              getCueAfter();
  public void                setCueAfter(String cueAfter);
  public String              getCueBefore();
  public void                setCueBefore(String cueBefore);
  public String             getCursor();
  public void               setCursor(String cursor);
  public String             getDirection();
  public void               setDirection(String direction);
  public String             getDisplay();
  public void               setDisplay(String display);
  public String             getElevation();
  public void               setElevation(String elevation);
  public String             getEmptyCells();
  public void               setEmptyCells(String emptyCells);
  public String             getCssFloat();
  public void               setCssFloat(String cssFloat);
  public String             getFont();
  public void               setFont(String font);
  public String             getFontFamily();
  public void               setFontFamily(String fontFamily);
  public String             getFontSize();
  public void               setFontSize(String fontSize);
  public String             getFontSizeAdjust();
  public void               setFontSizeAdjust(String fontSizeAdjust);
  public String             getFontStretch();
  public void               setFontStretch(String fontStretch);
  public String             getFontStyle();
  public void               setFontStyle(String fontStyle);
  public String             getFontVariant();
  public void               setFontVariant(String fontVariant);
  public String             getFontWeight();
  public void               setFontWeight(String fontWeight);
  public String             getHeight();
  public void               setHeight(String height);
  public String             getLeft();
  public void               setLeft(String left);
  public String             getLetterSpacing();
  public void               setLetterSpacing(String letterSpacing);
  public String             getLineHeight();
  public void               setLineHeight(String lineHeight);
  public String             getListStyle();
  public void               setListStyle(String listStyle);
  public String             getListStyleImage();
  public void               setListStyleImage(String listStyleImage);
  public String             getListStylePosition();
  public void               setListStylePosition(String listStylePosition);
  public String             getListStyleType();
  public void               setListStyleType(String listStyleType);
  public String             getMargin();
  public void               setMargin(String margin);
  public String             getMarginTop();
  public void               setMarginTop(String marginTop);
  public String             getMarginRight();
  public void               setMarginRight(String marginRight);
  public String             getMarginBottom();
  public void               setMarginBottom(String marginBottom);
  public String             getMarginLeft();
  public void               setMarginLeft(String marginLeft);
  public String             getMarkerOffset();
  public void               setMarkerOffset(String markerOffset);
  public String             getMarks();
  public void               setMarks(String marks);
  public String             getMaxHeight();
  public void               setMaxHeight(String maxHeight);
  public String             getMaxWidth();
  public void               setMaxWidth(String maxWidth);
  public String             getMinHeight();
  public void               setMinHeight(String minHeight);
  public String             getMinWidth();
  public void               setMinWidth(String minWidth);
  public String             getOrphans();
  public void               setOrphans(String orphans);
  public String             getOutline();
  public void               setOutline(String outline);
  public String             getOutlineColor();
  public void               setOutlineColor(String outlineColor);
  public String             getOutlineStyle();
  public void               setOutlineStyle(String outlineStyle);
  public String             getOutlineWidth();
  public void               setOutlineWidth(String outlineWidth);
  public String             getOverflow();
  public void               setOverflow(String overflow);
  public String             getPadding();
  public void               setPadding(String padding);
  public String             getPaddingTop();
  public void               setPaddingTop(String paddingTop);
  public String             getPaddingRight();
  public void               setPaddingRight(String paddingRight);
  public String             getPaddingBottom();
  public void               setPaddingBottom(String paddingBottom);
  public String             getPaddingLeft();
  public void               setPaddingLeft(String paddingLeft);
  public String             getPage();
  public void               setPage(String page);
  public String             getPageBreakAfter();
  public void               setPageBreakAfter(String pageBreakAfter);
  public String             getPageBreakBefore();
  public void               setPageBreakBefore(String pageBreakBefore);
  public String             getPageBreakInside();
  public void               setPageBreakInside(String pageBreakInside);
  public String             getPause();
  public void               setPause(String pause);
  public String             getPauseAfter();
  public void               setPauseAfter(String pauseAfter);
  public String             getPauseBefore();
  public void               setPauseBefore(String pauseBefore);
  public String             getPitch();
  public void               setPitch(String pitch);
  public String             getPitchRange();
  public void               setPitchRange(String pitchRange);
  public String             getPlayDuring();
  public void               setPlayDuring(String playDuring);
  public String             getPosition();
  public void               setPosition(String position);
  public String             getQuotes();
  public void               setQuotes(String quotes);
  public String             getRichness();
  public void               setRichness(String richness);
```

```
    public String             getRight();
    public void               setRight(String right);
    public String             getSize();
    public void               setSize(String size);
    public String             getSpeak();
    public void               setSpeak(String speak);
    public String             getSpeakHeader();
    public void               setSpeakHeader(String speakHeader);
    public String             getSpeakNumeral();
    public void               setSpeakNumeral(String speakNumeral);
    public String             getSpeakPunctuation();
    public void               setSpeakPunctuation(String speakPunctuation);
    public String             getSpeechRate();
    public void               setSpeechRate(String speechRate);
    public String             getStress();
    public void               setStress(String stress);
    public String             getTableLayout();
    public void               setTableLayout(String tableLayout);
    public String             getTextAlign();
    public void               setTextAlign(String textAlign);
    public String             getTextDecoration();
    public void               setTextDecoration(String textDecoration);
    public String             getTextIndent();
    public void               setTextIndent(String textIndent);
    public String             getTextShadow();
    public void               setTextShadow(String textShadow);
    public String             getTextTransform();
    public void               setTextTransform(String textTransform);
    public String             getTop();
    public void               setTop(String top);
    public String             getUnicodeBidi();
    public void               setUnicodeBidi(String unicodeBidi);
    public String             getVerticalAlign();
    public void               setVerticalAlign(String verticalAlign);
    public String             getVisibility();
    public void               setVisibility(String visibility);
    public String             getVoiceFamily();
    public void               setVoiceFamily(String voiceFamily);
    public String             getVolume();
    public void               setVolume(String volume);
    public String             getWhiteSpace();
    public void               setWhiteSpace(String whiteSpace);
    public String             getWidows();
    public void               setWidows(String widows);
    public String             getWidth();
    public void               setWidth(String width);
    public String             getWordSpacing();
    public void               setWordSpacing(String wordSpacing);
    public String             getZIndex();
    public void               setZIndex(String zIndex);
}
```

D.5: Document Object Model Level 2 Events

org/w3c/dom/events/EventTarget.java:

```
package org.w3c.dom.events;

import org.w3c.dom.*;

public interface EventTarget {
  public void               addEventListener(String type,
                                             EventListener listener,
                                             boolean useCapture);
  public void               removeEventListener(String type,
                                                EventListener listener,
                                                boolean useCapture);
}
```

org/w3c/dom/events/EventListener.java:

```
package org.w3c.dom.events;

import org.w3c.dom.*;

public interface EventListener {
  public void               handleEvent(Event event);
}
```

org/w3c/dom/events/Event.java:

```
package org.w3c.dom.events;

import org.w3c.dom.*;

public interface Event {
  // PhaseType
  public static final short           BUBBLING_PHASE       = 1;
  public static final short           CAPTURING_PHASE      = 2;
  public static final short           AT_TARGET            = 3;

  public String             getType();
  public void               setType(String type);
  public Node               getTarget();
  public void               setTarget(Node target);
  public Node               getCurrentNode();
  public void               setCurrentNode(Node currentNode);
  public short              getEventPhase();
  public void               setEventPhase(short eventPhase);
  public void               preventBubble();
  public void               preventCapture();
  public void               preventDefault();
}
```

org/w3c/dom/events/UIEvent.java:

```
package org.w3c.dom.events;

import org.w3c.dom.*;

public interface UIEvent extends Event {
  public static final int             CHAR_UNDEFINED       = 1;
  public static final int             KEY_FIRST            = 1;
  public static final int             KEY_LAST             = 1;
  public static final int             VK_0                 = 1;
  public static final int             VK_1                 = 1;
  public static final int             VK_2                 = 1;
  public static final int             VK_3                 = 1;
  public static final int             VK_4                 = 1;
  public static final int             VK_5                 = 1;
  public static final int             VK_6                 = 1;
  public static final int             VK_7                 = 1;
  public static final int             VK_8                 = 1;
  public static final int             VK_9                 = 1;
  public static final int             VK_A                 = 1;
  public static final int             VK_ACCEPT            = 1;
  public static final int             VK_ADD               = 1;
  public static final int             VK_AGAIN             = 1;
  public static final int             VK_ALL_CANDIDATES    = 1;
  public static final int             VK_ALPHANUMERIC      = 1;
  public static final int             VK_ALT               = 1;
  public static final int             VK_ALT_GRAPH         = 1;
  public static final int             VK_AMPERSAND         = 1;
  public static final int             VK_ASTERISK          = 1;
  public static final int             VK_AT                = 1;
  public static final int             VK_B                 = 1;
  public static final int             VK_BACK_QUOTE        = 1;
  public static final int             VK_BACK_SLASH        = 1;
  public static final int             VK_BACK_SPACE        = 1;
  public static final int             VK_BRACELEFT         = 1;
  public static final int             VK_BRACERIGHT        = 1;
  public static final int             VK_C                 = 1;
  public static final int             VK_CANCEL            = 1;
  public static final int             VK_CAPS_LOCK         = 1;
  public static final int             VK_CIRCUMFLEX        = 1;
  public static final int             VK_CLEAR             = 1;
  public static final int             VK_CLOSE_BRACKET     = 1;
  public static final int             VK_CODE_INPUT        = 1;
  public static final int             VK_COLON             = 1;
  public static final int             VK_COMMA             = 1;
  public static final int             VK_COMPOSE           = 1;
  public static final int             VK_CONTROL           = 1;
  public static final int             VK_CONVERT           = 1;
  public static final int             VK_COPY              = 1;
  public static final int             VK_CUT               = 1;
  public static final int             VK_D                 = 1;
  public static final int             VK_DEAD_ABOVEDOT     = 1;
  public static final int             VK_DEAD_ABOVERING    = 1;
  public static final int             VK_DEAD_ACUTE        = 1;
  public static final int             VK_DEAD_BREVE        = 1;
  public static final int             VK_DEAD_CARON        = 1;
  public static final int             VK_DEAD_CEDILLA      = 1;
  public static final int             VK_DEAD_CIRCUMFLEX   = 1;
  public static final int             VK_DEAD_DIAERESIS    = 1;
  public static final int             VK_DEAD_DOUBLEACUTE  = 1;
  public static final int             VK_DEAD_GRAVE        = 1;
  public static final int             VK_DEAD_IOTA         = 1;
  public static final int             VK_DEAD_MACRON       = 1;
  public static final int             VK_DEAD_OGONEK       = 1;
  public static final int             VK_DEAD_SEMIVOICED_SOUND = 1;
  public static final int             VK_DEAD_TILDE        = 1;
  public static final int             VK_DEAD_VOICED_SOUND = 1;
  public static final int             VK_DECIMAL           = 1;
  public static final int             VK_DELETE            = 1;
  public static final int             VK_DIVIDE            = 1;
  public static final int             VK_DOLLAR            = 1;
  public static final int             VK_DOWN              = 1;
  public static final int             VK_E                 = 1;
  public static final int             VK_END               = 1;
  public static final int             VK_ENTER             = 1;
  public static final int             VK_EQUALS            = 1;
  public static final int             VK_ESCAPE            = 1;
  public static final int             VK_EURO_SIGN         = 1;
  public static final int             VK_EXCLAMATION_MARK  = 1;
  public static final int             VK_F                 = 1;
  public static final int             VK_F1                = 1;
  public static final int             VK_F10               = 1;
  public static final int             VK_F11               = 1;
  public static final int             VK_F12               = 1;
  public static final int             VK_F13               = 1;
  public static final int             VK_F14               = 1;
  public static final int             VK_F15               = 1;
  public static final int             VK_F16               = 1;
  public static final int             VK_F17               = 1;
  public static final int             VK_F18               = 1;
  public static final int             VK_F19               = 1;
  public static final int             VK_F2                = 1;
  public static final int             VK_F20               = 1;
  public static final int             VK_F21               = 1;
  public static final int             VK_F22               = 1;
  public static final int             VK_F23               = 1;
  public static final int             VK_F24               = 1;
  public static final int             VK_F3                = 1;
  public static final int             VK_F4                = 1;
  public static final int             VK_F5                = 1;
  public static final int             VK_F6                = 1;
  public static final int             VK_F7                = 1;
  public static final int             VK_F8                = 1;
  public static final int             VK_F9                = 1;
  public static final int             VK_FINAL             = 1;
  public static final int             VK_FIND              = 1;
  public static final int             VK_FULL_WIDTH        = 1;
  public static final int             VK_G                 = 1;
  public static final int             VK_GREATER           = 1;
  public static final int             VK_H                 = 1;
  public static final int             VK_HALF_WIDTH        = 1;
  public static final int             VK_HELP              = 1;
  public static final int             VK_HIRAGANA          = 1;
  public static final int             VK_HOME              = 1;
  public static final int             VK_I                 = 1;
  public static final int             VK_INSERT            = 1;
  public static final int             VK_INVERTED_EXCLAMATION_MARK = 1;
  public static final int             VK_J                 = 1;
  public static final int             VK_JAPANESE_HIRAGANA = 1;
  public static final int             VK_JAPANESE_KATAKANA = 1;
  public static final int             VK_JAPANESE_ROMAN    = 1;
  public static final int             VK_K                 = 1;
  public static final int             VK_KANA              = 1;
  public static final int             VK_KANJI             = 1;
  public static final int             VK_KATAKANA          = 1;
  public static final int             VK_KP_DOWN           = 1;
  public static final int             VK_KP_LEFT           = 1;
  public static final int             VK_KP_RIGHT          = 1;
  public static final int             VK_KP_UP             = 1;
  public static final int             VK_L                 = 1;
  public static final int             VK_LEFT              = 1;
  public static final int             VK_LEFT_PARENTHESIS  = 1;
  public static final int             VK_LESS              = 1;
  public static final int             VK_M                 = 1;
  public static final int             VK_META              = 1;
  public static final int             VK_MINUS             = 1;
  public static final int             VK_MODECHANGE        = 1;
  public static final int             VK_MULTIPLY          = 1;
  public static final int             VK_N                 = 1;
  public static final int             VK_NONCONVERT        = 1;
  public static final int             VK_NUM_LOCK          = 1;
  public static final int             VK_NUMBER_SIGN       = 1;
  public static final int             VK_NUMPAD0           = 1;
  public static final int             VK_NUMPAD1           = 1;
  public static final int             VK_NUMPAD2           = 1;
  public static final int             VK_NUMPAD3           = 1;
  public static final int             VK_NUMPAD4           = 1;
  public static final int             VK_NUMPAD5           = 1;
  public static final int             VK_NUMPAD6           = 1;
  public static final int             VK_NUMPAD7           = 1;
  public static final int             VK_NUMPAD8           = 1;
  public static final int             VK_NUMPAD9           = 1;
  public static final int             VK_O                 = 1;
  public static final int             VK_OPEN_BRACKET      = 1;
  public static final int             VK_P                 = 1;
  public static final int             VK_PAGE_DOWN         = 1;
  public static final int             VK_PAGE_UP           = 1;
  public static final int             VK_PASTE             = 1;
  public static final int             VK_PAUSE             = 1;
  public static final int             VK_PERIOD            = 1;
  public static final int             VK_PLUS              = 1;
  public static final int             VK_PREVIOUS_CANDIDATE = 1;
  public static final int             VK_PRINTSCREEN       = 1;
  public static final int             VK_PROPS             = 1;
  public static final int             VK_Q                 = 1;
  public static final int             VK_QUOTE             = 1;
  public static final int             VK_QUOTEDBL          = 1;
  public static final int             VK_R                 = 1;
  public static final int             VK_RIGHT             = 1;
```

```
  public static final int             VK_RIGHT_PARENTHESIS = 1;
  public static final int             VK_ROMAN_CHARACTERS  = 1;
  public static final int             VK_S                 = 1;
  public static final int             VK_SCROLL_LOCK       = 1;
  public static final int             VK_SEMICOLON         = 1;
  public static final int             VK_SEPARATER         = 1;
  public static final int             VK_SHIFT             = 1;
  public static final int             VK_SLASH             = 1;
  public static final int             VK_SPACE             = 1;
  public static final int             VK_STOP              = 1;
  public static final int             VK_SUBTRACT          = 1;
  public static final int             VK_T                 = 1;
  public static final int             VK_TAB               = 1;
  public static final int             VK_U                 = 1;
  public static final int             VK_UNDEFINED         = 1;
  public static final int             VK_UNDERSCORE        = 1;
  public static final int             VK_UNDO              = 1;
  public static final int             VK_UP                = 1;
  public static final int             VK_V                 = 1;
  public static final int             VK_W                 = 1;
  public static final int             VK_X                 = 1;
  public static final int             VK_Y                 = 1;
  public static final int             VK_Z                 = 1;
  public int                 getScreenX();
  public void                setScreenX(int screenX);
  public int                 getScreenY();
  public void                setScreenY(int screenY);
  public int                 getClientX();
  public void                setClientX(int clientX);
  public int                 getClientY();
  public void                setClientY(int clientY);
  public boolean             getCtrlKey();
  public void                setCtrlKey(boolean ctrlKey);
  public boolean             getShiftKey();
  public void                setShiftKey(boolean shiftKey);
  public boolean             getAltKey();
  public void                setAltKey(boolean altKey);
  public boolean             getMetaKey();
  public void                setMetaKey(boolean metaKey);
  public int                 getKeyCode();
  public void                setKeyCode(int keyCode);
  public int                 getCharCode();
  public void                setCharCode(int charCode);
  public short               getButton();
  public void                setButton(short button);
  public short               getClickCount();
  public void                setClickCount(short clickCount);
}
```

org/w3c/dom/events/MutationEvent.java:

```
package org.w3c.dom.events;

import org.w3c.dom.*;

public interface MutationEvent extends Event {
  public Node                getRelatedNode();
  public void                setRelatedNode(Node relatedNode);
  public String              getPrevValue();
  public void                setPrevValue(String prevValue);
  public String              getNewValue();
  public void                setNewValue(String newValue);
  public String              getAttrName();
  public void                setAttrName(String attrName);
}
```

D.6: Document Object Model Level 2 Filters and Iterators

org/w3c/dom/fi/NodeIterator.java:

```
package org.w3c.dom.fi;

import org.w3c.dom.*;

public interface NodeIterator {
  public int                 getWhatToShow();
  // Constants for whatToShow
  public static final int             SHOW_ALL              = 0xFFFF;
  public static final int             SHOW_ELEMENT          = 0x00000001;
  public static final int             SHOW_ATTRIBUTE        = 0x00000002;
  public static final int             SHOW_TEXT             = 0x00000004;
  public static final int             SHOW_CDATA_SECTION    = 0x00000008;
  public static final int             SHOW_ENTITY_REFERENCE = 0x00000010;
  public static final int             SHOW_ENTITY           = 0x00000020;
  public static final int             SHOW_PROCESSING_INSTRUCTION = 0x00000040;
  public static final int             SHOW_COMMENT          = 0x00000080;
  public static final int             SHOW_DOCUMENT         = 0x00000100;
  public static final int             SHOW_DOCUMENT_TYPE    = 0x00000200;
  public static final int             SHOW_DOCUMENT_FRAGMENT = 0x00000400;
  public static final int             SHOW_NOTATION         = 0x00000800;

  public NodeFilter          getFilter();
  public Node                nextNode();
  public Node                previousNode();
}
```

org/w3c/dom/fi/NodeFilter.java:

```
package org.w3c.dom.fi;

import org.w3c.dom.*;

public interface NodeFilter {
  // Constants returned by acceptNode
  public static final short           FILTER_ACCEPT         = 1;
  public static final short           FILTER_REJECT         = 2;
  public static final short           FILTER_SKIP           = 3;

  public short               acceptNode(Node n);
}
```

org/w3c/dom/fi/TreeWalker.java:

```
package org.w3c.dom.fi;

import org.w3c.dom.*;

public interface TreeWalker {
  public int                getWhatToShow();
  // Constants for whatToShow
  public static final int             SHOW_ALL              = 0xFFFF;
  public static final int             SHOW_ELEMENT          = 0x00000001;
  public static final int             SHOW_ATTRIBUTE        = 0x00000002;
  public static final int             SHOW_TEXT             = 0x00000004;
  public static final int             SHOW_CDATA_SECTION    = 0x00000008;
  public static final int             SHOW_ENTITY_REFERENCE = 0x00000010;
  public static final int             SHOW_ENTITY           = 0x00000020;
  public static final int             SHOW_PROCESSING_INSTRUCTION = 0x00000040;
  public static final int             SHOW_COMMENT          = 0x00000080;
  public static final int             SHOW_DOCUMENT         = 0x00000100;
  public static final int             SHOW_DOCUMENT_TYPE    = 0x00000200;
  public static final int             SHOW_DOCUMENT_FRAGMENT = 0x00000400;
  public static final int             SHOW_NOTATION         = 0x00000800;

  public NodeFilter         getFilter();
  public Node               current();
  public Node               parentNode();
  public Node               firstChild();
  public Node               lastChild();
  public Node               previousSibling();
  public Node               nextSibling();
}
```

org/w3c/dom/fi/DocumentIF.java:

```
package org.w3c.dom.fi;

import org.w3c.dom.*;

public interface DocumentIF {
  public short              createNodeIterator(Node root,
                                                   short whatToShow,
                                                   NodeFilter filter);
}
```

D.7: Document Object Model Level 2 Range

org/w3c/dom/range/RangeException.java:

```
package org.w3c.dom.range;

import org.w3c.dom.*;

public abstract class RangeException extends RuntimeException {
  public RangeException(short code, String message) {
     super(message);
     this.code = code;
  }
  public short   code;
  // RangeExceptionCode
  public static final short           BAD_ENDPOINTS_ERR    = 201;
  public static final short           INVALID_NODE_TYPE_ERR = 202;
  public static final short           NULL_NODE_ERR        = 203;

}
```

org/w3c/dom/range/Range.java:

```
package org.w3c.dom.range;

import org.w3c.dom.*;

public interface Range {
  public Node               getStartContainer();
  public int                getStartOffset();
  public Node               getEndContainer();
  public int                getEndOffset();
  public boolean            getIsCollapsed();
  public Node               getCommonAncestorContainer();
  public void               setStart(Node node,
                                     int offset)
                                     throws RangeException;
  public void               setEnd(Node node,
                                   int offset)
                                   throws RangeException;
  public void               setStartBefore(Node node)
                                           throws RangeException;
  public void               setStartAfter(Node node)
                                          throws RangeException;
  public void               setEndBefore(Node node)
                                         throws RangeException;
  public void               setEndAfter(Node node)
                                        throws RangeException;
  public void               collapse(boolean toStart);
  public void               selectNode(Node node)
                                       throws RangeException;
  public void               selectNodeContents(Node node)
                                               throws RangeException;

  public static final int StartToStart = 1;
  public static final int StartToEnd   = 2;
  public static final int EndToEnd     = 3;
  public static final int EndToStart   = 4;


  public short              compareEndPoints(int how,
                                             Range sourceRange)
                                             throws DOMException;
  public void               deleteContents()
                                           throws DOMException;
  public DocumentFragment   extractContents()
```

```
                                            throws DOMException;
  public DocumentFragment   cloneContents()
                                          throws DOMException;
  public void               insertNode(Node node)
                                       throws DOMException, RangeException;
  public void               surroundContents(Node node)
                                             throws DOMException, RangeException;
  public Range              cloneRange();
  public String             toString();
}
```

org/w3c/dom/range/Range.java:

Appendix E: ECMA Script Language Binding

This appendix contains the complete ECMA Script binding for the Level 2 Document Object Model definitions. The definitions are divided into Core [p.205] , Namespaces [p.206] , Stylesheets [p.207] , CSS [p.208] , Events [p.221] , Filters and Iterators [p.222] , and Range [p.223] .

E.1: Document Object Model Level 2 Core

Object **DocumentType2**

DocumentType2 has the all the properties and methods of **DocumentType** as well as the properties and methods defined below.

The **DocumentType2** object has the following properties:

publicID

This property is of type **String**.

systemID

This property is of type **String**.

Object **DOMImplementation2**

DOMImplementation2 has the all the properties and methods of **DOMImplementation** as well as the properties and methods defined below.

The **DOMImplementation2** object has the following methods:

createDocumentType(name, publicID, systemID)

This method returns a **DocumentType**. The **name** parameter is of type **DOMString**. The **publicID** parameter is of type **DOMString**. The **systemID** parameter is of type **DOMString**.

createDocument(name, doctype)

This method returns a **Document**. The **name** parameter is of type **DOMString**. The **doctype** parameter is of type **DocumentType**.

Object **Document2**

Document2 has the all the properties and methods of **Document** as well as the properties and methods defined below.

The **Document2** object has the following methods:

importNode(importedNode, deep)

This method returns a **Node**. The **importedNode** parameter is of type **Node**. The **deep** parameter is of type **boolean**.

Object **Node2**

Node2 has the all the properties and methods of **Node** as well as the properties and methods defined below.

The **Node2** object has the following methods:

supports(feature, version)

This method returns a **boolean**. The **feature** parameter is of type **DOMString**. The **version** parameter is of type **DOMString**.

Object **Attr2**

Attr2 has the all the properties and methods of **Attr** as well as the properties and methods defined below.

The **Attr2** object has the following properties:

ownerElement

This property is of type **Element**.

Object **HTMLDOMImplementation**

HTMLDOMImplementation has the all the properties and methods of **DOMImplementation** as well as the properties and methods defined below.

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

createHTMLDocument(title)

This method returns a **HTMLDocument**. The **title** parameter is of type **DOMString**.

E.2: Document Object Model Level 2 Namespaces

Object **NodeNS**

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

namespaceName

This property is of type **String**.

prefix

This property is of type **String**.

localName

This property is of type **String**.

Object **DocumentNS**

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

createElementNS(namespaceName, qualifiedName)

This method returns a **Element**. The **namespaceName** parameter is of type **DOMString**. The **qualifiedName** parameter is of type **DOMString**.

createAttributeNS(namespaceName, qualifiedName)

This method returns a **Attr**. The **namespaceName** parameter is of type **DOMString**. The **qualifiedName** parameter is of type **DOMString**.

getElementsByTagNameNS(namespaceName, localName)

This method returns a **NodeList**. The **namespaceName** parameter is of type **DOMString**. The **localName** parameter is of type **DOMString**.

Object **ElementNS**

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

getAttributeNS(namespaceName, localName)

This method returns a **DOMString**. The **namespaceName** parameter is of type **DOMString**. The **localName** parameter is of type **DOMString**.

setAttributeNS(namespaceName, localName, value)

This method returns a **void**. The **namespaceName** parameter is of type **DOMString**. The **localName** parameter is of type **DOMString**. The **value** parameter is of type **DOMString**.

removeAttributeNS(namespaceName, localName)

This method returns a **void**. The **namespaceName** parameter is of type **DOMString**. The **localName** parameter is of type **DOMString**.

getAttributeNodeNS(namespaceName, localName)

This method returns a **Attr**. The **namespaceName** parameter is of type **DOMString**. The **localName** parameter is of type **DOMString**.

setAttributeNodeNS(newAttr)
This method returns a **Attr**. The **newAttr** parameter is of type **Attr**.
getElementsByTagNameNS(namespaceName, localName)
This method returns a **NodeList**. The **namespaceName** parameter is of type **DOMString**. The **localName** parameter is of type **DOMString**.

Object **NodeNS**
The **NodeNS** object has the following properties:
universalName
This property is of type **String**.
namespaceName
This property is of type **String**.
prefix
This property is of type **String**.
localName
This property is of type **String**.

Object **Document changes**
The **Document changes** object has the following methods:
createElement(universalName)
This method returns a **Element**. The **universalName** parameter is of type **DOMString**.
createAttribute(universalName)
This method returns a **Attr**. The **universalName** parameter is of type **DOMString**.
getElementsByTagName(universalName)
This method returns a **NodeList**. The **universalName** parameter is of type **DOMString**.

Object **Element changes**
The **Element changes** object has the following methods:
getAttribute(universalName)
This method returns a **DOMString**. The **universalName** parameter is of type **DOMString**.
setAttribute(universalName, value)
This method returns a **void**. The **universalName** parameter is of type **DOMString**. The **value** parameter is of type **DOMString**.
removeAttribute(universalName)
This method returns a **void**. The **universalName** parameter is of type **DOMString**.
getAttributeNode(universalName)
This method returns a **Attr**. The **universalName** parameter is of type **DOMString**.
setAttributeNode(newAttr)
This method returns a **Attr**. The **newAttr** parameter is of type **Attr**.
getElementsByTagName(universalName)
This method returns a **NodeList**. The **universalName** parameter is of type **DOMString**.

E.3: Document Object Model Level 2 Stylesheets

Object **StyleSheet**
The **StyleSheet** object has the following properties:
type
This property is of type **String**.

disabled
This property is of type **boolean**.
ownerNode
This property is of type **Node**.
parentStyleSheet
This property is of type **StyleSheet**.
href
This property is of type **String**.
title
This property is of type **String**.
media
This property is of type **MediaList**.

Object **StyleSheetList**
The **StyleSheetList** object has the following properties:
length
This property is of type **int**.
The **StyleSheetList** object has the following methods:
item(index)
This method returns a **StyleSheet**. The **index** parameter is of type **unsigned long**.

Object **MediaList**
The **MediaList** object has the following properties:
cssText
This property is of type **String**.
length
This property is of type **int**.
The **MediaList** object has the following methods:
item(index)
This method returns a **DOMString**. The **index** parameter is of type **unsigned long**.
delete(oldMedium)
This method returns a **void**. The **oldMedium** parameter is of type **DOMString**.
append(newMedium)
This method returns a **void**. The **newMedium** parameter is of type **DOMString**.

Object **DocumentStyle**
The **DocumentStyle** object has the following properties:
styleSheets
This property is of type **StyleSheetList**.

E.4: Document Object Model Level 2 CSS

Object **CSSStyleSheet**
CSSStyleSheet has the all the properties and methods of **StyleSheet** as well as the properties and methods defined below.
The **CSSStyleSheet** object has the following properties:
ownerRule
This property is of type **CSSRule**.

cssRules
This property is of type **CSSRuleList**.
The **CSSStyleSheet** object has the following methods:
insertRule(rule, index)
This method returns a **unsigned long**. The **rule** parameter is of type **DOMString**. The **index** parameter is of type **unsigned long**.
deleteRule(index)
This method returns a **void**. The **index** parameter is of type **unsigned long**.
Object **CSSRuleList**
The **CSSRuleList** object has the following properties:
length
This property is of type **int**.
The **CSSRuleList** object has the following methods:
item(index)
This method returns a **CSSRule**. The **index** parameter is of type **unsigned long**.
Object **CSSRule**
The **CSSRule** object has the following properties:
type
This property is of type **short**.
cssText
This property is of type **String**.
parentStyleSheet
This property is of type **CSSStyleSheet**.
parentRule
This property is of type **CSSRule**.
Object **CSSStyleRule**
CSSStyleRule has the all the properties and methods of **CSSRule** as well as the properties and methods defined below.
The **CSSStyleRule** object has the following properties:
selectorText
This property is of type **String**.
style
This property is of type **CSSStyleDeclaration**.
Object **CSSMediaRule**
CSSMediaRule has the all the properties and methods of **CSSRule** as well as the properties and methods defined below.
The **CSSMediaRule** object has the following properties:
media
This property is of type **MediaList**.
cssRules
This property is of type **CSSRuleList**.
The **CSSMediaRule** object has the following methods:
insertRule(rule, index)
This method returns a **unsigned long**. The **rule** parameter is of type **DOMString**. The **index** parameter is of type **unsigned long**.

deleteRule(index)
This method returns a **void**. The **index** parameter is of type **unsigned long**.

Object **CSSFontFaceRule**
CSSFontFaceRule has the all the properties and methods of **CSSRule** as well as the properties and methods defined below.
The **CSSFontFaceRule** object has the following properties:
style
This property is of type **CSSStyleDeclaration**.

Object **CSSPageRule**
CSSPageRule has the all the properties and methods of **CSSRule** as well as the properties and methods defined below.
The **CSSPageRule** object has the following properties:
selectorText
This property is of type **String**.
style
This property is of type **CSSStyleDeclaration**.

Object **CSSImportRule**
CSSImportRule has the all the properties and methods of **CSSRule** as well as the properties and methods defined below.
The **CSSImportRule** object has the following properties:
href
This property is of type **String**.
media
This property is of type **MediaList**.
styleSheet
This property is of type **CSSStyleSheet**.

Object **CSSCharsetRule**
CSSCharsetRule has the all the properties and methods of **CSSRule** as well as the properties and methods defined below.
The **CSSCharsetRule** object has the following properties:
encoding
This property is of type **String**.

Object **CSSUnknownRule**
CSSUnknownRule has the all the properties and methods of **CSSRule** as well as the properties and methods defined below.

Object **CSSStyleDeclaration**
The **CSSStyleDeclaration** object has the following properties:
cssText
This property is of type **String**.
length
This property is of type **int**.
parentRule
This property is of type **CSSRule**.
The **CSSStyleDeclaration** object has the following methods:
getPropertyValue(propertyName)
This method returns a **DOMString**. The **propertyName** parameter is of type **DOMString**.

getPropertyCSSValue(propertyName)
This method returns a **CSSValue**. The **propertyName** parameter is of type **DOMString**.
removeProperty(propertyName)
This method returns a **DOMString**. The **propertyName** parameter is of type **DOMString**.
getPropertyPriority(propertyName)
This method returns a **DOMString**. The **propertyName** parameter is of type **DOMString**.
setProperty(propertyName, value, priority)
This method returns a **void**. The **propertyName** parameter is of type **DOMString**. The **value** parameter is of type **DOMString**. The **priority** parameter is of type **DOMString**.
item(index)
This method returns a **DOMString**. The **index** parameter is of type **unsigned long**.

Object **CSSValue**
The **CSSValue** object has the following properties:
cssText
This property is of type **String**.
valueType
This property is of type **short**.

Object **CSSPrimitiveValue**
CSSPrimitiveValue has the all the properties and methods of **CSSValue** as well as the properties and methods defined below.
The **CSSPrimitiveValue** object has the following properties:
primitiveType
This property is of type **short**.
The **CSSPrimitiveValue** object has the following methods:
setFloatValue(unitType, floatValue)
This method returns a **void**. The **unitType** parameter is of type **unsigned short**. The **floatValue** parameter is of type **float**.
getFloatValue(unitType)
This method returns a **float**. The **unitType** parameter is of type **unsigned short**.
setStringValue(stringType, stringValue)
This method returns a **void**. The **stringType** parameter is of type **unsigned short**. The **stringValue** parameter is of type **DOMString**.
getStringValue()
This method returns a **DOMString**.
getCounterValue()
This method returns a **Counter**.
getRectValue()
This method returns a **Rect**.
getRGBColorValue()
This method returns a **RGBColor**.

Object **CSSValueList**
CSSValueList has the all the properties and methods of **CSSValue** as well as the properties and methods defined below.
The **CSSValueList** object has the following properties:
length
This property is of type **int**.

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

item(index)

This method returns a **CSSValue**. The **index** parameter is of type **unsigned long**.

Object **RGBColor**

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

red

This property is of type **CSSValue**.

green

This property is of type **CSSValue**.

blue

This property is of type **CSSValue**.

Object **Rect**

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

top

This property is of type **CSSValue**.

right

This property is of type **CSSValue**.

bottom

This property is of type **CSSValue**.

left

This property is of type **CSSValue**.

Object **Counter**

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

identifier

This property is of type **String**.

listStyle

This property is of type **String**.

separator

This property is of type **String**.

Object **CSS2Azimuth**

CSS2Azimuth has the all the properties and methods of **CSSValue** as well as the properties and methods defined below.

The **CSS2Azimuth** object has the following properties:

azimuthType

This property is of type **short**.

identifier

This property is of type **String**.

behind

This property is of type **boolean**.

The **CSS2Azimuth** object has the following methods:

setAngleValue(unitType, floatValue)

This method returns a **void**. The **unitType** parameter is of type **unsigned short**. The **floatValue** parameter is of type **float**.

getAngleValue(unitType)

This method returns a **float**. The **unitType** parameter is of type **unsigned short**.

setIdentifier(identifier, behind)
This method returns a **void**. The **identifier** parameter is of type **DOMString**. The **behind** parameter is of type **boolean**.

Object **CSS2BackgroundPosition**
CSS2BackgroundPosition has the all the properties and methods of **CSSValue** as well as the properties and methods defined below.
The **CSS2BackgroundPosition** object has the following properties:
horizontalType
This property is of type **short**.
verticalType
This property is of type **short**.
horizontalIdentifier
This property is of type **String**.
verticalIdentifier
This property is of type **String**.
The **CSS2BackgroundPosition** object has the following methods:
getHorizontalPosition(horizontalType)
This method returns a **float**. The **horizontalType** parameter is of type **float**.
getVerticalPosition(verticalType)
This method returns a **float**. The **verticalType** parameter is of type **float**.
setHorizontalPosition(horizontalType, value)
This method returns a **void**. The **horizontalType** parameter is of type **unsigned short**. The **value** parameter is of type **float**.
setVerticalPosition(verticalType, value)
This method returns a **void**. The **verticalType** parameter is of type **unsigned short**. The **value** parameter is of type **float**.
setPositionIdentifier(horizontalIdentifier, verticalIdentifier)
This method returns a **void**. The **horizontalIdentifier** parameter is of type **DOMString**. The **verticalIdentifier** parameter is of type **DOMString**.

Object **CSS2BorderSpacing**
CSS2BorderSpacing has the all the properties and methods of **CSSValue** as well as the properties and methods defined below.
The **CSS2BorderSpacing** object has the following properties:
horizontalType
This property is of type **short**.
verticalType
This property is of type **short**.
The **CSS2BorderSpacing** object has the following methods:
getHorizontalSpacing(horizontalType)
This method returns a **float**. The **horizontalType** parameter is of type **float**.
getVerticalSpacing(verticalType)
This method returns a **float**. The **verticalType** parameter is of type **float**.
setHorizontalSpacing(horizontalType, value)
This method returns a **void**. The **horizontalType** parameter is of type **unsigned short**. The **value** parameter is of type **float**.

setVerticalSpacing(verticalType, value)
This method returns a **void**. The **verticalType** parameter is of type **unsigned short**. The **value** parameter is of type **float**.
setInherit()()
This method returns a **void**.

Object **CSS2CounterReset**
The **CSS2CounterReset** object has the following properties:
identifier
This property is of type **String**.
reset
This property is of type **short**.

Object **CSS2CounterIncrement**
The **CSS2CounterIncrement** object has the following properties:
identifier
This property is of type **String**.
increment
This property is of type **short**.

Object **CSS2Cursor**
CSS2Cursor has the all the properties and methods of **CSSValue** as well as the properties and methods defined below.
The **CSS2Cursor** object has the following properties:
cursorType
This property is of type **short**.
uris
This property is of type **CSSValueList**.
predefinedCursor
This property is of type **String**.

Object **CSS2PlayDuring**
CSS2PlayDuring has the all the properties and methods of **CSSValue** as well as the properties and methods defined below.
The **CSS2PlayDuring** object has the following properties:
playDuringType
This property is of type **short**.
playDuringIdentifier
This property is of type **String**.
uri
This property is of type **String**.
mix
This property is of type **boolean**.
repeat
This property is of type **boolean**.

Object **CSS2TextShadow**
The **CSS2TextShadow** object has the following properties:
color
This property is of type **CSSValue**.

horizontal
This property is of type **CSSValue**.
vertical
This property is of type **CSSValue**.
blur
This property is of type **CSSValue**.

Object **CSS2FontFaceSrc**
The **CSS2FontFaceSrc** object has the following properties:
uri
This property is of type **String**.
format
This property is of type **CSSValueList**.
fontFaceName
This property is of type **String**.

Object **CSS2FontFaceWidths**
The **CSS2FontFaceWidths** object has the following properties:
urange
This property is of type **String**.
numbers
This property is of type **CSSValueList**.

Object **CSS2PageSize**
CSS2PageSize has the all the properties and methods of **CSSValue** as well as the properties and methods defined below.
The **CSS2PageSize** object has the following properties:
widthType
This property is of type **short**.
heightType
This property is of type **short**.
identifier
This property is of type **String**.
The **CSS2PageSize** object has the following methods:
getWidth(widthType)
This method returns a **float**. The **widthType** parameter is of type **float**.
getHeightSize(heightType)
This method returns a **float**. The **heightType** parameter is of type **float**.
setWidthSize(widthType, value)
This method returns a **void**. The **widthType** parameter is of type **unsigned short**. The **value** parameter is of type **float**.
setHeightSize(heightType, value)
This method returns a **void**. The **heightType** parameter is of type **unsigned short**. The **value** parameter is of type **float**.
setIdentifier(identifier)
This method returns a **void**. The **identifier** parameter is of type **DOMString**.

Object **CSS2Properties**
The **CSS2Properties** object has the following properties:

azimuth
This property is of type **String**.
background
This property is of type **String**.
backgroundAttachment
This property is of type **String**.
backgroundColor
This property is of type **String**.
backgroundImage
This property is of type **String**.
backgroundPosition
This property is of type **String**.
backgroundRepeat
This property is of type **String**.
border
This property is of type **String**.
borderCollapse
This property is of type **String**.
borderColor
This property is of type **String**.
borderSpacing
This property is of type **String**.
borderStyle
This property is of type **String**.
borderTop
This property is of type **String**.
borderRight
This property is of type **String**.
borderBottom
This property is of type **String**.
borderLeft
This property is of type **String**.
borderTopColor
This property is of type **String**.
borderRightColor
This property is of type **String**.
borderBottomColor
This property is of type **String**.
borderLeftColor
This property is of type **String**.
borderTopStyle
This property is of type **String**.
borderRightStyle
This property is of type **String**.
borderBottomStyle
This property is of type **String**.

borderLeftStyle
This property is of type **String**.
borderTopWidth
This property is of type **String**.
borderRightWidth
This property is of type **String**.
borderBottomWidth
This property is of type **String**.
borderLeftWidth
This property is of type **String**.
borderWidth
This property is of type **String**.
bottom
This property is of type **String**.
captionSide
This property is of type **String**.
clear
This property is of type **String**.
clip
This property is of type **String**.
color
This property is of type **String**.
content
This property is of type **String**.
counterIncrement
This property is of type **String**.
counterReset
This property is of type **String**.
cue
This property is of type **String**.
cueAfter
This property is of type **String**.
cueBefore
This property is of type **String**.
cursor
This property is of type **String**.
direction
This property is of type **String**.
display
This property is of type **String**.
elevation
This property is of type **String**.
emptyCells
This property is of type **String**.
cssFloat
This property is of type **String**.

font
: This property is of type **String**.

fontFamily
: This property is of type **String**.

fontSize
: This property is of type **String**.

fontSizeAdjust
: This property is of type **String**.

fontStretch
: This property is of type **String**.

fontStyle
: This property is of type **String**.

fontVariant
: This property is of type **String**.

fontWeight
: This property is of type **String**.

height
: This property is of type **String**.

left
: This property is of type **String**.

letterSpacing
: This property is of type **String**.

lineHeight
: This property is of type **String**.

listStyle
: This property is of type **String**.

listStyleImage
: This property is of type **String**.

listStylePosition
: This property is of type **String**.

listStyleType
: This property is of type **String**.

margin
: This property is of type **String**.

marginTop
: This property is of type **String**.

marginRight
: This property is of type **String**.

marginBottom
: This property is of type **String**.

marginLeft
: This property is of type **String**.

markerOffset
: This property is of type **String**.

marks
: This property is of type **String**.

maxHeight
This property is of type **String**.
maxWidth
This property is of type **String**.
minHeight
This property is of type **String**.
minWidth
This property is of type **String**.
orphans
This property is of type **String**.
outline
This property is of type **String**.
outlineColor
This property is of type **String**.
outlineStyle
This property is of type **String**.
outlineWidth
This property is of type **String**.
overflow
This property is of type **String**.
padding
This property is of type **String**.
paddingTop
This property is of type **String**.
paddingRight
This property is of type **String**.
paddingBottom
This property is of type **String**.
paddingLeft
This property is of type **String**.
page
This property is of type **String**.
pageBreakAfter
This property is of type **String**.
pageBreakBefore
This property is of type **String**.
pageBreakInside
This property is of type **String**.
pause
This property is of type **String**.
pauseAfter
This property is of type **String**.
pauseBefore
This property is of type **String**.
pitch
This property is of type **String**.

pitchRange
: This property is of type **String**.

playDuring
: This property is of type **String**.

position
: This property is of type **String**.

quotes
: This property is of type **String**.

richness
: This property is of type **String**.

right
: This property is of type **String**.

size
: This property is of type **String**.

speak
: This property is of type **String**.

speakHeader
: This property is of type **String**.

speakNumeral
: This property is of type **String**.

speakPunctuation
: This property is of type **String**.

speechRate
: This property is of type **String**.

stress
: This property is of type **String**.

tableLayout
: This property is of type **String**.

textAlign
: This property is of type **String**.

textDecoration
: This property is of type **String**.

textIndent
: This property is of type **String**.

textShadow
: This property is of type **String**.

textTransform
: This property is of type **String**.

top
: This property is of type **String**.

unicodeBidi
: This property is of type **String**.

verticalAlign
: This property is of type **String**.

visibility
: This property is of type **String**.

voiceFamily
This property is of type **String**.
volume
This property is of type **String**.
whiteSpace
This property is of type **String**.
widows
This property is of type **String**.
width
This property is of type **String**.
wordSpacing
This property is of type **String**.
zIndex
This property is of type **String**.

E.5: Document Object Model Level 2 Events

Object **EventTarget**
The **EventTarget** object has the following methods:
addEventListener(type, listener, useCapture)
This method returns a **void**. The **type** parameter is of type **DOMString**. The **listener** parameter is of type **EventListener**. The **useCapture** parameter is of type **boolean**.
removeEventListener(type, listener, useCapture)
This method returns a **void**. The **type** parameter is of type **DOMString**. The **listener** parameter is of type **EventListener**. The **useCapture** parameter is of type **boolean**.
Object **EventListener**
The **EventListener** object has the following methods:
handleEvent(event)
This method returns a **void**. The **event** parameter is of type **Event**.
Object **Event**
The **Event** object has the following properties:
type
This property is of type **String**.
target
This property is of type **Node**.
currentNode
This property is of type **Node**.
eventPhase
This property is of type **short**.
The **Event** object has the following methods:
preventBubble()
This method returns a **void**.
preventCapture()
This method returns a **void**.

preventDefault()
This method returns a **void**.

Object **UIEvent**

UIEvent has the all the properties and methods of **Event** as well as the properties and methods defined below.

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

screenX
This property is of type **long**.
screenY
This property is of type **long**.
clientX
This property is of type **long**.
clientY
This property is of type **long**.
ctrlKey
This property is of type **boolean**.
shiftKey
This property is of type **boolean**.
altKey
This property is of type **boolean**.
metaKey
This property is of type **boolean**.
keyCode
This property is of type **int**.
charCode
This property is of type **int**.
button
This property is of type **short**.
clickCount
This property is of type **short**.

Object **MutationEvent**

MutationEvent has the all the properties and methods of **Event** as well as the properties and methods defined below.

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

relatedNode
This property is of type **Node**.
prevValue
This property is of type **String**.
newValue
This property is of type **String**.
attrName
This property is of type **String**.

E.6: Document Object Model Level 2 Filters and Iterators

Object **NodeIterator**

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

whatToShow

This property is of type **long**.

filter

This property is of type **NodeFilter**.

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

nextNode()

This method returns a **Node**.

previousNode()

This method returns a **Node**.

Object **NodeFilter**

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

acceptNode(n)

This method returns a **short**. The **n** parameter is of type **Node**.

Object **TreeWalker**

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

whatToShow

This property is of type **long**.

filter

This property is of type **NodeFilter**.

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

current()

This method returns a **Node**.

parentNode()

This method returns a **Node**.

firstChild()

This method returns a **Node**.

lastChild()

This method returns a **Node**.

previousSibling()

This method returns a **Node**.

nextSibling()

This method returns a **Node**.

Object **DocumentIF**

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

createNodeIterator(root, whatToShow, filter)

This method returns a **short**. The **root** parameter is of type **Node**. The **whatToShow** parameter is of type **short**. The **filter** parameter is of type **NodeFilter**.

E.7: Document Object Model Level 2 Range

Object **Range**

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

startContainer

This property is of type **Node**.

startOffset

This property is of type **long**.

endContainer

This property is of type **Node**.

endOffset

This property is of type **long**.

isCollapsed

This property is of type **boolean**.

commonAncestorContainer

This property is of type **Node**.

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

setStart(node, offset)

This method returns a **void**. The **node** parameter is of type **Node**. The **offset** parameter is of type **long**.

setEnd(node, offset)

This method returns a **void**. The **node** parameter is of type **Node**. The **offset** parameter is of type **long**.

setStartBefore(node)

This method returns a **void**. The **node** parameter is of type **Node**.

setStartAfter(node)

This method returns a **void**. The **node** parameter is of type **Node**.

setEndBefore(node)

This method returns a **void**. The **node** parameter is of type **Node**.

setEndAfter(node)

This method returns a **void**. The **node** parameter is of type **Node**.

collapse(toStart)

This method returns a **void**. The **toStart** parameter is of type **boolean**.

selectNode(node)

This method returns a **void**. The **node** parameter is of type **Node**.

selectNodeContents(node)

This method returns a **void**. The **node** parameter is of type **Node**.

compareEndPoints(how, sourceRange)

This method returns a **short**. The **how** parameter is of type **CompareHow**. The **sourceRange** parameter is of type **Range**.

deleteContents()

This method returns a **void**.

extractContents()

This method returns a **DocumentFragment**.

cloneContents()
  This method returns a **DocumentFragment**.
insertNode(node)
  This method returns a **void**. The **node** parameter is of type **Node**.
surroundContents(node)
  This method returns a **void**. The **node** parameter is of type **Node**.
cloneRange()
  This method returns a **Range**.
toString()
  This method returns a **DOMString**.

References

CORBA

OMG (Object Management Group) *The Common Object Request Broker: Architecture and Specification*. See http://www.omg.org/corba/corbiiop.htm.

DOM-Level-1

W3C (World Wide Web Consortium) *DOM Level 1 Specification*. See http://www.w3.org/TR/REC-DOM-Level-1.

ECMAScript

ECMA (European Computer Manufacturers Association) *ECMAScript Language Specification*. See http://www.ecma.ch/stand/ECMA-262.htm.

HTML4.0

W3C (World Wide Web Consortium) *HTML 4.0 Specification*. See http://www.w3.org/TR/REC-html40.

Java

Sun *The Java Language Specification*. See http://java.sun.com/docs/books/jls/.

Namespaces

W3C (World Wide Web Consortium) *Namespaces in XML* . See http://www.w3.org/TR/REC-xml-names.

Unicode

The Unicode Consortium. *The Unicode Standard, Version 2.0.* Reading, Mass.: Addison-Wesley Developers Press, 1996.

XML

W3C (World Wide Web Consortium) *Extensible Markup Language (XML) 1.0*. See http://www.w3.org/TR/REC-xml.

Index