



Document Object Model (DOM) Level 2 Style Specification

Version 1.0

W3C Recommendation 13 November, 2000

This version:

<http://www.w3.org/TR/2000/REC-DOM-Level-2-Style-20001113>
(PostScript file , PDF file , plain text , ZIP file)

Latest version:

<http://www.w3.org/TR/DOM-Level-2-Style>

Previous version:

<http://www.w3.org/TR/2000/PR-DOM-Level-2-Style-20000927>

Editors:

Chris Wilson, *Microsoft Corp.*

Philippe Le Hégaré, *W3C, team contact (from November 1999)*

Vidur Apparao, *Netscape Communications Corp.*

Copyright © 2000 W3C® (MIT, INRIA, Keio), All Rights Reserved. W3C liability, trademark, document use and software licensing rules apply.

Abstract

This specification defines the Document Object Model Level 2 Style Sheets and Cascading Style Sheets (CSS), a platform- and language-neutral interface that allows programs and scripts to dynamically access and update the content and of style sheets documents. The Document Object Model Level 2 Style builds on the Document Object Model Level 2 Core [DOM Level 2 Core] and on the Document Object Model Level 2 Views [DOM Level 2 Views].

Status of this document

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

This document has been reviewed by W3C Members and other interested parties and has been endorsed by the Director as a W3C Recommendation. It is a stable document and may be used as reference material or cited as a normative reference from another document. W3C's role in making the Recommendation is to draw attention to the specification and to promote its widespread deployment. This enhances the functionality and interoperability of the Web.

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

Please send general comments about this document to the public mailing list www-dom@w3.org. An archive is available at <http://lists.w3.org/Archives/Public/www-dom/>.

The English version of this specification is the only normative version. Information about translations of this document is available at <http://www.w3.org/2000/11/DOM-Level-2-translations>.

The list of known errors in this document is available at <http://www.w3.org/2000/11/DOM-Level-2-errata>

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

Table of contents

Expanded Table of Contents	.3
Copyright Notice	.5
Chapter 1: Document Object Model Style Sheets	.9
Chapter 2: Document Object Model CSS	15
Appendix A: IDL Definitions	79
Appendix B: Java Language Binding	93
Appendix C: ECMAScript Language Binding	111
Appendix D: Acknowledgements	125
References	127
Index	129

Expanded Table of Contents

Expanded Table of Contents	.3
Copyright Notice	.5
W3C Document Copyright Notice and License	.5
W3C Software Copyright Notice and License	.6
 Chapter 1: Document Object Model Style Sheets	.9
1.1. Introduction	.9
1.2. Style Sheet Interfaces	.9
1.3. Document Extensions	13
1.4. Association between a style sheet and a document.	13
 Chapter 2: Document Object Model CSS	15
2.1. Overview of the DOM Level 2 CSS Interfaces	15
2.2. CSS Fundamental Interfaces	15
2.2.1. Override and computed style sheet	37
2.2.2. Style sheet creation	38
2.2.3. Element with CSS inline style	39
2.3. CSS2 Extended Interface	40
 Appendix A: IDL Definitions	79
A.1. Document Object Model Style Sheets	79
A.2. Document Object Model CSS	80
 Appendix B: Java Language Binding	93
B.1. Document Object Model Style Sheets	93
B.2. Document Object Model CSS	94
 Appendix C: ECMAScript Language Binding	111
C.1. Document Object Model StyleSheets	111
C.2. Document Object Model CSS	112
 Appendix D: Acknowledgements	125
D.1. Production Systems	125
 References	127
1. Normative references	127
2. Informative references	127
 Index	129

Copyright Notice

Copyright © 2000 World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University). All Rights Reserved.

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

W3C Document Copyright Notice and License

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

Copyright © 1994-2000 World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University). All Rights Reserved.

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

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

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

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

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

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

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

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

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

W3C Software Copyright Notice and License

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

Copyright © 1994-2000 World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University). All Rights Reserved.

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

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

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

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

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

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

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

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

1. Document Object Model Style Sheets

Editors

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

1.1. 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.

The interfaces found within this section are not mandatory. A DOM application may use the `hasFeature(feature, version)` method of the `DOMImplementation` interface with parameter values "StyleSheets" and "2.0" (respectively) to determine whether or not this module is supported by the implementation. In order to fully support this module, an implementation must also support the "Core" feature defined in the DOM 2 Core specification [DOM Level 2 Core]. Please refer to additional information about *conformance* in the DOM Level 2 Core specification [DOM Level 2 Core].

1.2. Style Sheet Interfaces

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

Interface *StyleSheet* (introduced in **DOM Level 2**)

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

```
// Introduced in DOM Level 2:
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

disabled of type `boolean`

`false` if the style sheet is applied to the document. `true` if it is not. Modifying this attribute may cause a new resolution of style for the document. A stylesheet only applies if both an appropriate medium definition is present and the disabled attribute is false. So, if the media doesn't apply to the current user agent, the `disabled` attribute is ignored.

href of type `DOMString`, readonly

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*.

media of type `MediaList` [p.11], readonly

The intended destination media for style information. The media is often specified in the `ownerNode`. If no media has been specified, the `MediaList` [p.11] will be empty. 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*. Modifying the media list may cause a change to the attribute `disabled`.

ownerNode of type `Node`, readonly

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, the value of this attribute is `null`.

parentStyleSheet of type `StyleSheet` [p.9], readonly

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 this attribute is `null`.

title of type `DOMString`, readonly

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*.

type of type `DOMString`, readonly

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`. 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*.

Interface `StyleSheetList` (introduced in **DOM Level 2**)

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

The items in the `StyleSheetList` are accessible via an integral index, starting from 0.

IDL Definition

```
// Introduced in DOM Level 2:
interface StyleSheetList {
    readonly attribute unsigned long    length;
    StyleSheet          item(in unsigned long index);
};
```

Attributes

`length` of type `unsigned long`, readonly

The number of `StyleSheets` [p.9] 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. If index is greater than or equal to the number of style sheets in the list, this returns `null`.

Parameters

`index` of type `unsigned long`

Index into the collection

Return Value

<code>StyleSheet</code> [p.9]	The style sheet at the <code>index</code> position in the <code>StyleSheetList</code> , or <code>null</code> if that is not a valid index.
----------------------------------	--

No Exceptions**Interface *MediaList*** (introduced in **DOM Level 2**)

The `MediaList` interface provides the abstraction of an ordered collection of *media*, without defining or constraining how this collection is implemented. An empty list is the same as a list that contains the medium "all".

The items in the `MediaList` are accessible via an integral index, starting from 0.

IDL Definition

```
// Introduced in DOM Level 2:
interface MediaList {
    attribute DOMString          mediaText;
                                // raises(DOMException) on setting

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

Attributes

`length` of type `unsigned long`, readonly

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

`mediaText` of type `DOMString`

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 string value has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this media list is readonly.

Methods

appendMedium

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

Parameters

`newMedium` of type `DOMString`
The new medium to add.

Exceptions

DOMException **INVALID_CHARACTER_ERR**: If the medium contains characters that are invalid in the underlying style language.

NO_MODIFICATION_ALLOWED_ERR: Raised if this list is readonly.

No Return Value

deleteMedium

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

Parameters

`oldMedium` of type `DOMString`
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.

No Return Value

item

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

Parameters

`index` of type `unsigned long`
Index into the collection.

Return Value

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

No Exceptions

1.3. Document Extensions

Interface *LinkStyle* (introduced in **DOM Level 2**)

The `LinkStyle` interface provides a mechanism by which a style sheet can be retrieved from the node responsible for linking it into a document. An instance of the `LinkStyle` interface can be obtained using binding-specific casting methods on an instance of a linking node (`HTMLLinkElement`, `HTMLStyleElement` or `ProcessingInstruction` in DOM Level 2).

IDL Definition

```
// Introduced in DOM Level 2:
interface LinkStyle {
    readonly attribute StyleSheet    sheet;
};
```

Attributes

`sheet` of type `StyleSheet` [p.9], readonly
The style sheet.

Interface *DocumentStyle* (introduced in **DOM Level 2**)

The `DocumentStyle` interface provides a mechanism by which the style sheets embedded in 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 `Document` interface.

IDL Definition

```
// Introduced in DOM Level 2:
interface DocumentStyle {
    readonly attribute StyleSheetList    styleSheets;
};
```

Attributes

`styleSheets` of type `StyleSheetList` [p.10], readonly
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 (see [XML-StyleSheet]).

1.4. Association between a style sheet and a document.

HTML and Style Sheet Creation

A style sheet can be associated with an `HTMLDocument` in one of two ways:

- By creating a new `LINK` HTML element (see the `HTMLLinkElement` interface in the [DOM Level 2 HTML] and [HTML4.0]). The underlying style sheet will be created after the element is

inserted into the document and both the href and the type attribute have been set in a way indicating that the linked object is a style sheet.

- By creating a new STYLE HTML element (see the `HTMLStyleElement` interface in the [DOM Level 2 HTML] and [HTML4.0]). The underlying style sheet will be created after the element is inserted into the document and the type attribute is set in a way indicating that the element corresponds to a style sheet language interpreted by the user agent.

HTML and Style Sheet Removal

Removing a LINK HTML element or a STYLE HTML element removes the underlying style sheet from the style sheet collection associated with a document. Specifically, the removed style sheet is no longer applied to the presentation of the document.

XML and Style Sheet Creation

A new style sheet can be created and associated with an XML document by creating a processing instruction with the target 'xml-stylesheet' [XML-StyleSheet] and inserting it into the document.

XML and Style Sheet Removal

Removing a processing instruction with a target of 'xml-stylesheet' [XML-StyleSheet] removes the underlying style sheet from the style sheet collection associated with a document. Specifically, the removed style sheet is no longer applied to the presentation of the document.

2. Document Object Model CSS

Editors

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

2.1. 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 2 [CSS2]). 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 Document Object Model Style Sheets [p.9]. 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` [p.16] 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` [p.18] interface.

The most common type of rule is a style declaration. The `CSSStyleRule` [p.20] 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` [p.24] interface.

Finally, an optional `CSS2Properties` [p.40] interface is described; this interface (if implemented) provides shortcuts to the string values of all the properties in CSS level 2.

All CSS objects in the DOM are "live", that is, a change in the style sheet is reflected in the computed and actual style.

2.2. CSS Fundamental Interfaces

The interfaces within this section are considered fundamental CSS interfaces, and must be supported by all conforming implementations of the CSS module. These interfaces represent CSS style sheets specifically.

A DOM application may use the `hasFeature(feature, version)` method of the `DOMImplementation` interface with parameter values "CSS" and "2.0" (respectively) to determine whether or not this module is supported by the implementation. In order to fully support this module, an implementation must also support the "Core" feature defined in the DOM Level 2 Core specification [DOM Level 2 Core] and the "Views" feature defined in the DOM Level 2 Views specification [DOM Level 2 Views]. Please refer to additional information about *conformance* in the DOM Level 2 Core specification [DOM Level 2 Core].

Interface *CSSStyleSheet* (introduced in **DOM Level 2**)

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

```
// Introduced in DOM Level 2:
interface CSSStyleSheet : stylesheets::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

`cssRules` of type `CSSRuleList` [p.17], readonly

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

`ownerRule` of type `CSSRule` [p.18], readonly

If this style sheet comes from an `@import` rule, the `ownerRule` attribute will contain the `CSSImportRule` [p.23]. In that case, the `ownerNode` attribute in the `StyleSheet` [p.9] 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`.

Methods

`deleteRule`

Used to delete a rule from the style sheet.

Parameters

`index` of type `unsigned long`

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.

No Return Value**insertRule**

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

Parameters

rule of type `DOMString`

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 of type `unsigned long`

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

<code>unsigned long</code>	The index within the style sheet's rule collection of the newly inserted rule.
----------------------------	--

Exceptions

<code>DOMException</code>	HIERARCHY_REQUEST_ERR: Raised if the rule cannot be inserted at the specified index e.g. if an <code>@import</code> 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. NO_MODIFICATION_ALLOWED_ERR: Raised if this style sheet is readonly. SYNTAX_ERR: Raised if the specified rule has a syntax error and is unparsable.
---------------------------	--

Interface *CSSRuleList* (introduced in **DOM Level 2**)

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

The items in the `CSSRuleList` are accessible via an integral index, starting from 0.

IDL Definition

```
// Introduced in DOM Level 2:
interface CSSRuleList {
    readonly attribute unsigned long    length;
    CSSRule        item(in unsigned long index);
};
```

Attributes

length of type unsigned long, readonly

The number of CSSRules [p.18] 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. If index is greater than or equal to the number of rules in the list, this returns null.

Parameters

index of type unsigned long

Index into the collection

Return Value

CSSRule [p.18]	The style rule at the index position in the CSSRuleList, or null if that is not a valid index.
-------------------	--

No Exceptions**Interface CSSRule** (introduced in DOM Level 2)

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 the rule is not recognized by the parser. Unrecognized rules are represented using the CSSUnknownRule [p.24] interface.

IDL Definition

```
// Introduced in DOM Level 2:
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

CHARSET_RULE

The rule is a `CSSCharsetRule` [p.23] .

FONT_FACE_RULE

The rule is a `CSSFontFaceRule` [p.22] .

IMPORT_RULE

The rule is a `CSSImportRule` [p.23] .

MEDIA_RULE

The rule is a `CSSMediaRule` [p.20] .

PAGE_RULE

The rule is a `CSSPageRule` [p.22] .

STYLE_RULE

The rule is a `CSSStyleRule` [p.20] .

UNKNOWN_RULE

The rule is a `CSSUnknownRule` [p.24] .

Attributes

cssText of type `DOMString`

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.

INVALID_MODIFICATION_ERR: Raised if the specified CSS string value represents a different type of rule than the current one.

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

NO_MODIFICATION_ALLOWED_ERR: Raised if the rule is readonly.

parentRule of type `CSSRule` [p.18] , readonly

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`.

parentStyleSheet of type `CSSStyleSheet` [p.16] , readonly

The style sheet that contains this rule.

type of type `unsigned short`, readonly

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`.

Interface *CSSStyleRule* (introduced in **DOM Level 2**)

The *CSSStyleRule* interface represents a single *rule set* in a CSS style sheet.

IDL Definition

```
// Introduced in DOM Level 2:
interface CSSStyleRule : CSSRule {
    attribute DOMString      selectorText;
                                // raises(DOMException) on setting

    readonly attribute CSSStyleDeclaration style;
};
```

Attributes

selectorText of type *DOMString*

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 rule is readonly.

style of type *CSSStyleDeclaration* [p.24], readonly

The *declaration-block* of this rule set.

Interface *CSSMediaRule* (introduced in **DOM Level 2**)

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

```
// Introduced in DOM Level 2:
interface CSSMediaRule : CSSRule {
    readonly attribute StyleSheetList 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

cssRules of type *CSSRuleList* [p.17], readonly

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

`media` of type `styleSheets::MediaList`, readonly
 A list of *media types* for this rule.

Methods

`deleteRule`

Used to delete a rule from the media block.

Parameters

`index` of type `unsigned long`

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.

No Return Value

`insertRule`

Used to insert a new rule into the media block.

Parameters

`rule` of type `DOMString`

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` of type `unsigned long`

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

`unsigned long` 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.

NO_MODIFICATION_ALLOWED_ERR: Raised if this media rule is readonly.

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

Interface **CSSFontFaceRule** (introduced in **DOM Level 2**)

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

```
// Introduced in DOM Level 2:
interface CSSFontFaceRule : CSSRule {
    readonly attribute CSSStyleDeclaration style;
};
```

Attributes

style of type **CSSStyleDeclaration** [p.24], readonly
The *declaration-block* of this rule.

Interface **CSSPageRule** (introduced in **DOM Level 2**)

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

```
// Introduced in DOM Level 2:
interface CSSPageRule : CSSRule {
    attribute DOMString selectorText;
    // raises(DOMException) on setting

    readonly attribute CSSStyleDeclaration style;
};
```

Attributes

selectorText of type **DOMString**
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 unparseable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this rule is readonly.

style of type `CSSStyleDeclaration` [p.24] , readonly

The *declaration-block* of this rule.

Interface `CSSImportRule` (introduced in **DOM Level 2**)

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

```
// Introduced in DOM Level 2:
interface CSSImportRule : CSSRule {
    readonly attribute DOMString      href;
    readonly attribute stylesheets::MediaList media;
    readonly attribute CSSStyleSheet  styleSheet;
};
```

Attributes

href of type `DOMString`, readonly

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

media of type `stylesheets::MediaList`, readonly

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

styleSheet of type `CSSStyleSheet` [p.16] , readonly

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` (introduced in **DOM Level 2**)

The `CSSCharsetRule` interface represents a *@charset rule* in a CSS style sheet. The value of the **encoding** attribute does not affect the encoding of text data in the DOM objects; this encoding is always UTF-16. After a stylesheet is loaded, the value of the **encoding** attribute is the value found in the *@charset* rule. If there was no *@charset* in the original document, then no `CSSCharsetRule` is created. The value of the **encoding** attribute may also be used as a hint for the encoding used on serialization of the style sheet.

The value of the *@charset rule* (and therefore of the `CSSCharsetRule`) may not correspond to the encoding the document actually came in; character encoding information e.g. in an HTTP header, has priority (see *CSS document representation*) but this is not reflected in the `CSSCharsetRule`.

IDL Definition

```
// Introduced in DOM Level 2:
interface CSSCharsetRule : CSSRule {
    attribute DOMString          encoding;
                                // raises(DOMException) on setting
};
```

Attributes

encoding of type `DOMString`

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 unparseable.

`NO_MODIFICATION_ALLOWED_ERR`: Raised if this encoding rule is readonly.

Interface *CSSUnknownRule* (introduced in **DOM Level 2**)

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

IDL Definition

```
// Introduced in DOM Level 2:
interface CSSUnknownRule : CSSRule {
};
```

Interface *CSSStyleDeclaration* (introduced in **DOM Level 2**)

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 in the style sheet 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.40] interface.

This interface is also used to provide a **read-only** access to the *computed values* of an element. See also the `ViewCSS` [p.37] interface.

Note: The CSS Object Model doesn't provide an access to the *specified* or *actual* values of the CSS cascade.

IDL Definition

```
// Introduced in DOM Level 2:
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 of type **DOMString**

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

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this declaration is readonly or a property is readonly.

length of type **unsigned long**, **readonly**

The number of properties that have been explicitly set in this declaration block. The range of valid indices is 0 to length-1 inclusive.

parentRule of type **CSSRule** [p.18] , **readonly**

The CSS rule that contains this declaration block or **null** if this **CSSStyleDeclaration** is not attached to a **CSSRule** [p.18] .

Methods

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 of type **DOMString**

The name of the CSS property. See the *CSS property index*.

Return Value

CSSValue [p.27] Returns the value of the property if it has been explicitly set for this declaration block. Returns `null` if the property has not been set.

No Exceptions

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` of type `DOMString`

The name of the CSS property. See the *CSS property index*.

Return Value

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

No Exceptions

getPropertyValue

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

Parameters

`propertyName` of type `DOMString`

The name of the CSS property. See the *CSS property index*.

Return Value

`DOMString` 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.

No Exceptions

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` of type `unsigned long`

Index of the property name to retrieve.

Return Value

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

No Exceptions

removeProperty

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

Parameters

propertyName of type **DOMString**

The name of the CSS property. See the *CSS property index*.

Return Value

DOMString 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 known CSS property.

Exceptions

DOMException **NO_MODIFICATION_ALLOWED_ERR**: Raised if this declaration is readonly or the property is readonly.

setProperty

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

Parameters

propertyName of type **DOMString**

The name of the CSS property. See the *CSS property index*.

value of type **DOMString**

The new value of the property.

priority of type **DOMString**

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 or the property is readonly.

No Return Value**Interface CSSValue** (introduced in **DOM Level 2**)

The CSSValue interface represents a simple or a complex value. A CSSValue object only occurs in a context of a CSS property.

IDL Definition

```
// Introduced in DOM Level 2:
interface CSSValue {

    // UnitTypes
```

```

const unsigned short    CSS_INHERIT                = 0;
const unsigned short    CSS_PRIMITIVE_VALUE        = 1;
const unsigned short    CSS_VALUE_LIST              = 2;
const unsigned short    CSS_CUSTOM                 = 3;

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

    readonly attribute unsigned short    cssValueType;
};

```

Definition group *UnitTypes*

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

Defined Constants

CSS_CUSTOM

The value is a custom value.

CSS_INHERIT

The value is inherited and the `cssText` contains "inherit".

CSS_PRIMITIVE_VALUE

The value is a primitive value and an instance of the `CSSPrimitiveValue` [p.28] interface can be obtained by using binding-specific casting methods on this instance of the `CSSValue` interface.

CSS_VALUE_LIST

The value is a `CSSValue` list and an instance of the `CSSValueList` [p.34] interface can be obtained by using binding-specific casting methods on this instance of the `CSSValue` interface.

Attributes

`cssText` of type `DOMString`

A string representation of the current value.

Exceptions on setting

DOMException

SYNTAX_ERR: Raised if the specified CSS string value has a syntax error (according to the attached property) or is unparsable.

INVALID_MODIFICATION_ERR: Raised if the specified CSS string value represents a different type of values than the values allowed by the CSS property.

NO_MODIFICATION_ALLOWED_ERR: Raised if this value is readonly.

`cssValueType` of type `unsigned short`, readonly

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

Interface *CSSPrimitiveValue* (introduced in **DOM Level 2**)

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 property explicitly within the block. An instance of this interface might be obtained from the `getPropertyCSSValue` method of the `CSSStyleDeclaration` [p.24] interface. A `CSSPrimitiveValue` object only occurs in a context of a CSS property.

Conversions are allowed between absolute values (from millimeters to centimeters, from degrees to radians, and so on) but not between relative values. (For example, a pixel value cannot be converted to a centimeter value.) Percentage values can't be converted since they are relative to the parent value (or another property value). There is one exception for color percentage values: since a color percentage value is relative to the range 0-255, a color percentage value can be converted to a number; (see also the `RGBColor` [p.35] interface).

IDL Definition

```
// Introduced in DOM Level 2:
interface CSSPrimitiveValue : CSSValue {

    // UnitTypes
    const unsigned short    CSS_UNKNOWN           = 0;
    const unsigned short    CSS_NUMBER           = 1;
    const unsigned short    CSS_PERCENTAGE       = 2;
    const unsigned short    CSS_EMS              = 3;
    const unsigned short    CSS_EXS             = 4;
    const unsigned short    CSS_PX              = 5;
    const unsigned short    CSS_CM              = 6;
    const unsigned short    CSS_MM              = 7;
    const unsigned short    CSS_IN              = 8;
    const unsigned short    CSS_PT              = 9;
    const unsigned short    CSS_PC              = 10;
    const unsigned short    CSS_DEG            = 11;
    const unsigned short    CSS_RAD            = 12;
    const unsigned short    CSS_GRAD           = 13;
    const unsigned short    CSS_MS             = 14;
    const unsigned short    CSS_S              = 15;
    const unsigned short    CSS_HZ            = 16;
    const unsigned short    CSS_KHZ           = 17;
    const unsigned short    CSS_DIMENSION     = 18;
    const unsigned short    CSS_STRING        = 19;
    const unsigned short    CSS_URI           = 20;
    const unsigned short    CSS_IDENT         = 21;
    const unsigned short    CSS_ATTR          = 22;
    const unsigned short    CSS_COUNTER       = 23;
    const unsigned short    CSS_RECT          = 24;
    const unsigned short    CSS_RGBCOLOR      = 25;

    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)
```

```

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_ATTR

The value is a *attribute function*. The value can be obtained by using the `getStringValue` method.

CSS_CM

The value is a *length (cm)*. The value can be obtained by using the `getFloatValue` method.

CSS_COUNTER

The value is a *counter or counters function*. The value can be obtained by using the `getCounterValue` method.

CSS_DEG

The value is an *angle (deg)*. 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_EMS

The value is a *length (ems)*. The value can be obtained by using the `getFloatValue` method.

CSS_EXS

The value is a *length (exs)*. 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_HZ

The value is a *frequency (Hz)*. The value can be obtained by using the `getFloatValue` method.

CSS_IDENT

The value is an *identifier*. The value can be obtained by using the `getStringValue` method.

CSS_IN

The value is a *length (in)*. 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_MM

The value is a *length (mm)*. 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_NUMBER

The value is a simple *number*. 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_PERCENTAGE

The value is a *percentage*. The value can be obtained by using the `getFloatValue` method.

CSS_PT

The value is a *length (pt)*. The value can be obtained by using the `getFloatValue` method.

CSS_PX

The value is a *length (px)*. 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_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.

CSS_S

The value is a *time (s)*. 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_UNKNOWN

The value is not a recognized CSS2 value. The value can only be obtained by using the `cssText` attribute.

CSS_URI

The value is a *URI*. The value can be obtained by using the `getStringValue` method.

Attributes

`primitiveType` of type `unsigned short`, readonly

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

Methods

`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.36] interface.

Return Value

`Counter` [p.36] The Counter value.

Exceptions

`DOMException` `INVALID_ACCESS_ERR`: Raised if the CSS value doesn't contain a Counter value (e.g. this is not `CSS_COUNTER`).

No Parameters

`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` of type `unsigned short`

A unit code to get the float value. The unit code can only be a float unit type (i.e. `CSS_NUMBER`, `CSS_PERCENTAGE`, `CSS_EMS`, `CSS_EXS`, `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

`float` The float value in the specified unit.

Exceptions

`DOMException` `INVALID_ACCESS_ERR`: Raised if the CSS value doesn't contain a float value or if the float value can't be converted into the specified unit.

`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.35] interface.

Return Value

`RGBColor` [p.35] the RGB color value.

Exceptions

`DOMException` `INVALID_ACCESS_ERR`: Raised if the attached property can't return a RGB color value (e.g. this is not `CSS_RGBCOLOR`).

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.36] interface.

Return Value

`Rect` [p.36] The Rect value.

Exceptions

`DOMException` `INVALID_ACCESS_ERR`: Raised if the CSS value doesn't contain a Rect value. (e.g. this is not `CSS_RECT`).

No Parameters

`getStringValue`

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

Note: Some properties (like 'font-family' or 'voice-family') convert a whitespace separated list of idents to a string.

Return Value

`DOMString` The string value in the current unit. The current `primitiveType` can only be a string unit type (i.e. `CSS_STRING`, `CSS_URI`, `CSS_IDENT` and `CSS_ATTR`).

Exceptions

`DOMException` `INVALID_ACCESS_ERR`: Raised if the CSS value doesn't contain a string value.

No Parameters

`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 of type unsigned short

A unit code as defined above. The unit code can only be a float unit type (i.e. CSS_NUMBER, CSS_PERCENTAGE, CSS_EMS, CSS_EXS, 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 of type float

The new float value.

Exceptions

DOMException INVALID_ACCESS_ERR: Raised if the attached property doesn't support the float value or the unit type.

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

No Return Value

setStringValue

A method to set the string value with the 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 of type unsigned short

A string code as defined above. The string code can only be a string unit type (i.e. CSS_STRING, CSS_URI, CSS_IDENT, and CSS_ATTR).

stringValue of type DOMString

The new string value.

Exceptions

DOMException INVALID_ACCESS_ERR: Raised 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.

No Return Value

Interface CSSValueList (introduced in **DOM Level 2**)

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

Some properties allow an empty list into their syntax. In that case, these properties take the none identifier. So, an empty list means that the property has the value none.

The items in the `CSSValueList` are accessible via an integral index, starting from 0.

IDL Definition

```
// Introduced in DOM Level 2:
interface CSSValueList : CSSValue {
    readonly attribute unsigned long    length;
    CSSValue        item(in unsigned long index);
};
```

Attributes

`length` of type `unsigned long`, `readonly`

The number of `CSSValues` [p.27] in the list. The range of valid values of the indices is 0 to `length-1` inclusive.

Methods

`item`

Used to retrieve a `CSSValue` [p.27] by ordinal index. The order in this collection represents the order of the values in the CSS style property. If `index` is greater than or equal to the number of values in the list, this returns `null`.

Parameters

`index` of type `unsigned long`

Index into the collection.

Return Value

<code>CSSValue</code> [p.27]	The <code>CSSValue</code> at the <code>index</code> position in the <code>CSSValueList</code> , or <code>null</code> if that is not a valid index.
---------------------------------	--

No Exceptions

Interface *RGBColor* (introduced in **DOM Level 2**)

The `RGBColor` interface is used to represent any *RGB color* value. This interface reflects the values in the underlying style property. Hence, modifications made to the `CSSPrimitiveValue` [p.28] objects modify the style property.

A specified RGB color is not clipped (even if the number is outside the range 0-255 or 0%-100%). A computed RGB color is clipped depending on the device.

Even if a style sheet can only contain an integer for a color value, the internal storage of this integer is a float, and this can be used as a float in the specified or the computed style.

A color percentage value can always be converted to a number and vice versa.

IDL Definition

```
// Introduced in DOM Level 2:
interface RGBColor {
    readonly attribute CSSPrimitiveValue  red;
    readonly attribute CSSPrimitiveValue  green;
    readonly attribute CSSPrimitiveValue  blue;
};
```

Attributes

blue of type `CSSPrimitiveValue` [p.28], readonly

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

green of type `CSSPrimitiveValue` [p.28], readonly

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

red of type `CSSPrimitiveValue` [p.28], readonly

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

Interface *Rect* (introduced in **DOM Level 2**)

The `Rect` interface is used to represent any *rect* value. This interface reflects the values in the underlying style property. Hence, modifications made to the `CSSPrimitiveValue` [p.28] objects modify the style property.

IDL Definition

```
// Introduced in DOM Level 2:
interface Rect {
    readonly attribute CSSPrimitiveValue top;
    readonly attribute CSSPrimitiveValue right;
    readonly attribute CSSPrimitiveValue bottom;
    readonly attribute CSSPrimitiveValue left;
};
```

Attributes

bottom of type `CSSPrimitiveValue` [p.28], readonly

This attribute is used for the bottom of the rect.

left of type `CSSPrimitiveValue` [p.28], readonly

This attribute is used for the left of the rect.

right of type `CSSPrimitiveValue` [p.28], readonly

This attribute is used for the right of the rect.

top of type `CSSPrimitiveValue` [p.28], readonly

This attribute is used for the top of the rect.

Interface *Counter* (introduced in **DOM Level 2**)

The `Counter` interface is used to represent any *counter* or *counters function* value. This interface reflects the values in the underlying style property.

IDL Definition

```
// Introduced in DOM Level 2:
interface Counter {
    readonly attribute DOMString identifier;
    readonly attribute DOMString listStyle;
    readonly attribute DOMString separator;
};
```

Attributes

identifier of type `DOMString`, readonly

This attribute is used for the identifier of the counter.

`listStyle` of type `DOMString`, readonly
 This attribute is used for the style of the list.

`separator` of type `DOMString`, readonly
 This attribute is used for the separator of the nested counters.

2.2.1. Override and computed style sheet

Interface *ViewCSS* (introduced in **DOM Level 2**)

This interface represents a CSS view. The `getComputedStyle` method provides a **read only access** to the *computed values* of an element.

The expectation is that an instance of the *ViewCSS* interface can be obtained by using binding-specific casting methods on an instance of the *AbstractView* interface.

Since a computed style is related to an *Element* node, if this element is removed from the document, the associated *CSSStyleDeclaration* [p.24] and *CSSValue* [p.27] related to this declaration are no longer valid.

IDL Definition

```
// Introduced in DOM Level 2:
interface ViewCSS : views::AbstractView {
    CSSStyleDeclaration getComputedStyle(in Element elt,
                                         in DOMString pseudoElt);
};
```

Methods

`getComputedStyle`

This method is used to get the computed style as it is defined in [CSS2].

Parameters

`elt` of type *Element*

The element whose style is to be computed. This parameter cannot be null.

`pseudoElt` of type *DOMString*

The pseudo-element or `null` if none.

Return Value

<i>CSSStyleDeclaration</i> [p.24]	The computed style. The <i>CSSStyleDeclaration</i> is read-only and contains only absolute values.
--------------------------------------	--

No Exceptions

Interface *DocumentCSS* (introduced in **DOM Level 2**)

This interface represents a document with a CSS view.

The `getOverrideStyle` method provides a mechanism through which a DOM author could effect immediate change to the style of an element without modifying the explicitly linked style sheets of a document or the inline style of elements in the style sheets. This style sheet comes after the author style sheet in the cascade algorithm and is called *override style sheet*. The override style sheet takes precedence over author style sheets. An `!important` declaration still takes precedence over a normal declaration. Override, author, and user style sheets all may contain `!important` declarations. User `!important` rules take precedence over both override and author `!important` rules, and override `!important` rules take precedence over author `!important` rules.

The expectation is that an instance of the `DocumentCSS` interface can be obtained by using binding-specific casting methods on an instance of the `Document` interface.

IDL Definition

```
// Introduced in DOM Level 2:
interface DocumentCSS : stylesheets::DocumentStyle {
    CSSStyleDeclaration getOverrideStyle(in Element elt,
                                         in DOMString pseudoElt);
};
```

Methods

`getOverrideStyle`

This method is used to retrieve the override style declaration for a specified element and a specified pseudo-element.

Parameters

`elt` of type `Element`

The element whose style is to be modified. This parameter cannot be null.

`pseudoElt` of type `DOMString`

The pseudo-element or null if none.

Return Value

`CSSStyleDeclaration` [p.24] The override style declaration.

No Exceptions

2.2.2. Style sheet creation

Interface *DOMImplementationCSS* (introduced in DOM Level 2)

This interface allows the DOM user to create a `CSSStyleSheet` [p.16] outside the context of a document. There is no way to associate the new `CSSStyleSheet` with a document in DOM Level 2.

IDL Definition

```
// Introduced in DOM Level 2:
interface DOMImplementationCSS : DOMImplementation {
    CSSStyleSheet      createCSSStyleSheet(in DOMString title,
                                           in DOMString media)
                                           raises(DOMException);
};
```

Methods

createCSSStyleSheet

Creates a new `CSSStyleSheet` [p.16] .

Parameters

title of type `DOMString`

The advisory title. See also the Style Sheet Interfaces [p.10] section.

media of type `DOMString`

The comma-separated list of media associated with the new style sheet. See also the Style Sheet Interfaces [p.10] section.

Return Value

`CSSStyleSheet` [p.16] A new CSS style sheet.

Exceptions

`DOMException` `SYNTAX_ERR`: Raised if the specified media string value has a syntax error and is unparseable.

2.2.3. Element with CSS inline style

Interface *ElementCSSInlineStyle* (introduced in **DOM Level 2**)

Inline style information attached to elements is exposed through the `style` attribute. This represents the contents of the *STYLE* attribute for HTML elements (or elements in other schemas or DTDs which use the *STYLE* attribute in the same way). The expectation is that an instance of the `ElementCSSInlineStyle` interface can be obtained by using binding-specific casting methods on an instance of the `Element` interface when the element supports inline CSS style informations.

IDL Definition

```
// Introduced in DOM Level 2:
interface ElementCSSInlineStyle {
    readonly attribute CSSStyleDeclaration style;
};
```

Attributes

style of type `CSSStyleDeclaration` [p.24] , readonly
The style attribute.

2.3. CSS2 Extended Interface

The interface found within this section are not mandatory. A DOM application may use the `hasFeature(feature, version)` method of the `DOMImplementation` interface with parameter values "CSS2" and "2.0" (respectively) to determine whether or not this module is supported by the implementation. In order to fully support this module, an implementation must also support the "CSS" feature defined in CSS Fundamental Interfaces [p.15]. Please refer to additional information about *conformance* in the DOM Level 2 Core specification [DOM Level 2 Core].

Interface *CSS2Properties* (introduced in **DOM Level 2**)

The `CSS2Properties` interface represents a convenience mechanism for retrieving and setting properties within a `CSSStyleDeclaration` [p.24]. 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` interface. Setting an attribute of this interface is equivalent to calling the `setProperty` method of the `CSSStyleDeclaration` interface.

A conformant implementation of the CSS module 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.24] 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

```
// Introduced in DOM Level 2:
interface CSS2Properties {
    attribute DOMString      azimuth;
                                // raises(DOMException) on setting

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

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

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

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

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

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

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

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

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

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

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

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

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

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

    attribute DOMString      borderLeft;
                                // raises(DOMException) on setting
}
```

attribute DOMString	borderTopColor; // raises(DOMException) on setting
attribute DOMString	borderRightColor; // raises(DOMException) on setting
attribute DOMString	borderBottomColor; // raises(DOMException) on setting
attribute DOMString	borderLeftColor; // raises(DOMException) on setting
attribute DOMString	borderTopStyle; // raises(DOMException) on setting
attribute DOMString	borderRightStyle; // raises(DOMException) on setting
attribute DOMString	borderBottomStyle; // raises(DOMException) on setting
attribute DOMString	borderLeftStyle; // raises(DOMException) on setting
attribute DOMString	borderTopWidth; // raises(DOMException) on setting
attribute DOMString	borderRightWidth; // raises(DOMException) on setting
attribute DOMString	borderBottomWidth; // raises(DOMException) on setting
attribute DOMString	borderLeftWidth; // raises(DOMException) on setting
attribute DOMString	borderWidth; // raises(DOMException) on setting
attribute DOMString	bottom; // raises(DOMException) on setting
attribute DOMString	captionSide; // raises(DOMException) on setting
attribute DOMString	clear; // raises(DOMException) on setting
attribute DOMString	clip; // raises(DOMException) on setting
attribute DOMString	color; // raises(DOMException) on setting
attribute DOMString	content; // raises(DOMException) on setting

attribute DOMString	counterIncrement; // raises(DOMException) on setting
attribute DOMString	counterReset; // raises(DOMException) on setting
attribute DOMString	cue; // raises(DOMException) on setting
attribute DOMString	cueAfter; // raises(DOMException) on setting
attribute DOMString	cueBefore; // raises(DOMException) on setting
attribute DOMString	cursor; // raises(DOMException) on setting
attribute DOMString	direction; // raises(DOMException) on setting
attribute DOMString	display; // raises(DOMException) on setting
attribute DOMString	elevation; // raises(DOMException) on setting
attribute DOMString	emptyCells; // raises(DOMException) on setting
attribute DOMString	cssFloat; // raises(DOMException) on setting
attribute DOMString	font; // raises(DOMException) on setting
attribute DOMString	fontFamily; // raises(DOMException) on setting
attribute DOMString	fontSize; // raises(DOMException) on setting
attribute DOMString	fontSizeAdjust; // raises(DOMException) on setting
attribute DOMString	fontStretch; // raises(DOMException) on setting
attribute DOMString	fontStyle; // raises(DOMException) on setting
attribute DOMString	fontVariant; // raises(DOMException) on setting
attribute DOMString	fontWeight; // raises(DOMException) on setting

attribute DOMString	height; // raises(DOMException) on setting
attribute DOMString	left; // raises(DOMException) on setting
attribute DOMString	letterSpacing; // raises(DOMException) on setting
attribute DOMString	lineHeight; // raises(DOMException) on setting
attribute DOMString	listStyle; // raises(DOMException) on setting
attribute DOMString	listStyleImage; // raises(DOMException) on setting
attribute DOMString	listStylePosition; // raises(DOMException) on setting
attribute DOMString	listStyleType; // raises(DOMException) on setting
attribute DOMString	margin; // raises(DOMException) on setting
attribute DOMString	marginTop; // raises(DOMException) on setting
attribute DOMString	marginRight; // raises(DOMException) on setting
attribute DOMString	marginBottom; // raises(DOMException) on setting
attribute DOMString	marginLeft; // raises(DOMException) on setting
attribute DOMString	markerOffset; // raises(DOMException) on setting
attribute DOMString	marks; // raises(DOMException) on setting
attribute DOMString	maxHeight; // raises(DOMException) on setting
attribute DOMString	maxWidth; // raises(DOMException) on setting
attribute DOMString	minHeight; // raises(DOMException) on setting
attribute DOMString	minWidth; // raises(DOMException) on setting

attribute DOMString	orphans; // raises(DOMException) on setting
attribute DOMString	outline; // raises(DOMException) on setting
attribute DOMString	outlineColor; // raises(DOMException) on setting
attribute DOMString	outlineStyle; // raises(DOMException) on setting
attribute DOMString	outlineWidth; // raises(DOMException) on setting
attribute DOMString	overflow; // raises(DOMException) on setting
attribute DOMString	padding; // raises(DOMException) on setting
attribute DOMString	paddingTop; // raises(DOMException) on setting
attribute DOMString	paddingRight; // raises(DOMException) on setting
attribute DOMString	paddingBottom; // raises(DOMException) on setting
attribute DOMString	paddingLeft; // raises(DOMException) on setting
attribute DOMString	page; // raises(DOMException) on setting
attribute DOMString	pageBreakAfter; // raises(DOMException) on setting
attribute DOMString	pageBreakBefore; // raises(DOMException) on setting
attribute DOMString	pageBreakInside; // raises(DOMException) on setting
attribute DOMString	pause; // raises(DOMException) on setting
attribute DOMString	pauseAfter; // raises(DOMException) on setting
attribute DOMString	pauseBefore; // raises(DOMException) on setting
attribute DOMString	pitch; // raises(DOMException) on setting

attribute DOMString	pitchRange; // raises(DOMException) on setting
attribute DOMString	playDuring; // raises(DOMException) on setting
attribute DOMString	position; // raises(DOMException) on setting
attribute DOMString	quotes; // raises(DOMException) on setting
attribute DOMString	richness; // raises(DOMException) on setting
attribute DOMString	right; // raises(DOMException) on setting
attribute DOMString	size; // raises(DOMException) on setting
attribute DOMString	speak; // raises(DOMException) on setting
attribute DOMString	speakHeader; // raises(DOMException) on setting
attribute DOMString	speakNumeral; // raises(DOMException) on setting
attribute DOMString	speakPunctuation; // raises(DOMException) on setting
attribute DOMString	speechRate; // raises(DOMException) on setting
attribute DOMString	stress; // raises(DOMException) on setting
attribute DOMString	tableLayout; // raises(DOMException) on setting
attribute DOMString	textAlign; // raises(DOMException) on setting
attribute DOMString	textDecoration; // raises(DOMException) on setting
attribute DOMString	textIndent; // raises(DOMException) on setting
attribute DOMString	textShadow; // raises(DOMException) on setting
attribute DOMString	textTransform; // raises(DOMException) on setting

```

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

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

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

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

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

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

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

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

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

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

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

};

```

Attributes

azimuth of type DOMString

See the *azimuth property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

background of type DOMString

See the *background property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

backgroundAttachment of type **DOMString**

See the *background-attachment property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

backgroundColor of type **DOMString**

See the *background-color property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

backgroundImage of type **DOMString**

See the *background-image property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

backgroundPosition of type **DOMString**

See the *background-position property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

backgroundRepeat of type **DOMString**

See the *background-repeat property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

border of type **DOMString**

See the *border property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

borderBottom of type **DOMString**

See the *border-bottom property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

borderBottomColor of type **DOMString**

See the *border-bottom-color property definition* in CSS2.

Exceptions on setting

DOMException **SYNTAX_ERR**: Raised if the new value has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

borderBottomStyle of type **DOMString**

See the *border-bottom-style property definition* in CSS2.

Exceptions on setting

DOMException **SYNTAX_ERR**: Raised if the new value has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

borderBottomWidth of type **DOMString**

See the *border-bottom-width property definition* in CSS2.

Exceptions on setting

DOMException **SYNTAX_ERR**: Raised if the new value has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

borderCollapse of type **DOMString**

See the *border-collapse property definition* in CSS2.

Exceptions on setting

DOMException **SYNTAX_ERR**: Raised if the new value has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

borderColor of type **DOMString**

See the *border-color property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

borderLeft of type **DOMString**

See the *border-left property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

borderLeftColor of type **DOMString**

See the *border-left-color property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

borderLeftStyle of type **DOMString**

See the *border-left-style property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

borderLeftWidth of type **DOMString**

See the *border-left-width property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

borderRight of type **DOMString**

See the *border-right property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

borderRightColor of type **DOMString**

See the *border-right-color property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

borderRightStyle of type **DOMString**

See the *border-right-style property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

borderRightWidth of type **DOMString**

See the *border-right-width property definition* in CSS2.

Exceptions on setting

DOMException **SYNTAX_ERR**: Raised if the new value has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

borderSpacing of type **DOMString**

See the *border-spacing property definition* in CSS2.

Exceptions on setting

DOMException **SYNTAX_ERR**: Raised if the new value has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

borderStyle of type **DOMString**

See the *border-style property definition* in CSS2.

Exceptions on setting

DOMException **SYNTAX_ERR**: Raised if the new value has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

borderTop of type **DOMString**

See the *border-top property definition* in CSS2.

Exceptions on setting

DOMException **SYNTAX_ERR**: Raised if the new value has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

borderTopColor of type **DOMString**

See the *border-top-color property definition* in CSS2.

Exceptions on setting

DOMException **SYNTAX_ERR**: Raised if the new value has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

borderTopStyle of type **DOMString**

See the *border-top-style property definition* in CSS2.

Exceptions on setting

DOMException **SYNTAX_ERR**: Raised if the new value has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

borderTopWidth of type **DOMString**

See the *border-top-width property definition* in CSS2.

Exceptions on setting

DOMException **SYNTAX_ERR**: Raised if the new value has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

borderWidth of type **DOMString**

See the *border-width property definition* in CSS2.

Exceptions on setting

DOMException **SYNTAX_ERR**: Raised if the new value has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

bottom of type **DOMString**

See the *bottom property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

captionSide of type DOMString

See the *caption-side property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

clear of type DOMString

See the *clear property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

clip of type DOMString

See the *clip property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

color of type DOMString

See the *color property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

content of type DOMString

See the *content property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

counterIncrement of type DOMString

See the *counter-increment property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

counterReset of type DOMString

See the *counter-reset property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

cssFloat of type DOMString

See the *float property definition* in CSS2.

Exceptions on setting

DOMException **SYNTAX_ERR**: Raised if the new value has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

cue of type **DOMString**

See the *cue property definition* in CSS2.

Exceptions on setting

DOMException **SYNTAX_ERR**: Raised if the new value has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

cueAfter of type **DOMString**

See the *cue-after property definition* in CSS2.

Exceptions on setting

DOMException **SYNTAX_ERR**: Raised if the new value has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

cueBefore of type **DOMString**

See the *cue-before property definition* in CSS2.

Exceptions on setting

DOMException **SYNTAX_ERR**: Raised if the new value has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

cursor of type **DOMString**

See the *cursor property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

direction of type DOMString

See the *direction property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

display of type DOMString

See the *display property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

elevation of type DOMString

See the *elevation property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

emptyCells of type DOMString

See the *empty-cells property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

font of type DOMString

See the *font property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

fontFamily of type DOMString

See the *font-family property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

fontSize of type DOMString

See the *font-size property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

fontSizeAdjust of type DOMString

See the *font-size-adjust property definition* in CSS2.

Exceptions on setting

DOMException **SYNTAX_ERR**: Raised if the new value has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

fontStretch of type **DOMString**

See the *font-stretch property definition* in CSS2.

Exceptions on setting

DOMException **SYNTAX_ERR**: Raised if the new value has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

fontStyle of type **DOMString**

See the *font-style property definition* in CSS2.

Exceptions on setting

DOMException **SYNTAX_ERR**: Raised if the new value has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

fontVariant of type **DOMString**

See the *font-variant property definition* in CSS2.

Exceptions on setting

DOMException **SYNTAX_ERR**: Raised if the new value has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

fontWeight of type **DOMString**

See the *font-weight property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

height of type DOMString

See the *height property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

left of type DOMString

See the *left property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

letterSpacing of type DOMString

See the *letter-spacing property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

lineHeight of type DOMString

See the *line-height property definition* in CSS2.

Exceptions on setting

DOMException **SYNTAX_ERR**: Raised if the new value has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

listStyle of type **DOMString**

See the *list-style property definition* in CSS2.

Exceptions on setting

DOMException **SYNTAX_ERR**: Raised if the new value has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

listStyleImage of type **DOMString**

See the *list-style-image property definition* in CSS2.

Exceptions on setting

DOMException **SYNTAX_ERR**: Raised if the new value has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

listStylePosition of type **DOMString**

See the *list-style-position property definition* in CSS2.

Exceptions on setting

DOMException **SYNTAX_ERR**: Raised if the new value has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

listStyleType of type **DOMString**

See the *list-style-type property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

margin of type DOMString

See the *margin property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

marginBottom of type DOMString

See the *margin-bottom property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

marginLeft of type DOMString

See the *margin-left property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

marginRight of type DOMString

See the *margin-right property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

marginTop of type DOMString

See the *margin-top property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

markerOffset of type DOMString

See the *marker-offset property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

marks of type DOMString

See the *marks property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

maxHeight of type DOMString

See the *max-height property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

maxWidth of type DOMString

See the *max-width property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

minHeight of type DOMString

See the *min-height property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

minWidth of type DOMString

See the *min-width property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

orphans of type DOMString

See the *orphans property definition* in CSS2.

Exceptions on setting

DOMException **SYNTAX_ERR**: Raised if the new value has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

outline of type **DOMString**

See the *outline property definition* in CSS2.

Exceptions on setting

DOMException **SYNTAX_ERR**: Raised if the new value has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

outlineColor of type **DOMString**

See the *outline-color property definition* in CSS2.

Exceptions on setting

DOMException **SYNTAX_ERR**: Raised if the new value has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

outlineStyle of type **DOMString**

See the *outline-style property definition* in CSS2.

Exceptions on setting

DOMException **SYNTAX_ERR**: Raised if the new value has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

outlineWidth of type **DOMString**

See the *outline-width property definition* in CSS2.

Exceptions on setting

DOMException **SYNTAX_ERR**: Raised if the new value has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

overflow of type **DOMString**

See the *overflow property definition* in CSS2.

Exceptions on setting

DOMException **SYNTAX_ERR**: Raised if the new value has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

padding of type **DOMString**

See the *padding property definition* in CSS2.

Exceptions on setting

DOMException **SYNTAX_ERR**: Raised if the new value has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

paddingBottom of type **DOMString**

See the *padding-bottom property definition* in CSS2.

Exceptions on setting

DOMException **SYNTAX_ERR**: Raised if the new value has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

paddingLeft of type **DOMString**

See the *padding-left property definition* in CSS2.

Exceptions on setting

DOMException **SYNTAX_ERR**: Raised if the new value has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

paddingRight of type **DOMString**

See the *padding-right property definition* in CSS2.

Exceptions on setting

DOMException **SYNTAX_ERR**: Raised if the new value has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

paddingTop of type **DOMString**

See the *padding-top property definition* in CSS2.

Exceptions on setting

DOMException **SYNTAX_ERR**: Raised if the new value has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

page of type **DOMString**

See the *page property definition* in CSS2.

Exceptions on setting

DOMException **SYNTAX_ERR**: Raised if the new value has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

pageBreakAfter of type **DOMString**

See the *page-break-after property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

pageBreakBefore of type **DOMString**

See the *page-break-before property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

pageBreakInside of type **DOMString**

See the *page-break-inside property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

pause of type **DOMString**

See the *pause property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

pauseAfter of type **DOMString**

See the *pause-after property definition* in CSS2.

Exceptions on setting

DOMException **SYNTAX_ERR**: Raised if the new value has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

pauseBefore of type **DOMString**

See the *pause-before property definition* in CSS2.

Exceptions on setting

DOMException **SYNTAX_ERR**: Raised if the new value has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

pitch of type **DOMString**

See the *pitch property definition* in CSS2.

Exceptions on setting

DOMException **SYNTAX_ERR**: Raised if the new value has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

pitchRange of type **DOMString**

See the *pitch-range property definition* in CSS2.

Exceptions on setting

DOMException **SYNTAX_ERR**: Raised if the new value has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

playDuring of type **DOMString**

See the *play-during property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

position of type DOMString

See the *position property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

quotes of type DOMString

See the *quotes property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

richness of type DOMString

See the *richness property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

right of type DOMString

See the *right property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

size of type DOMString

See the *size property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

speak of type DOMString

See the *speak property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

speakHeader of type DOMString

See the *speak-header property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

speakNumeral of type DOMString

See the *speak-numeral property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

speakPunctuation of type DOMString

See the *speak-punctuation property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

speechRate of type DOMString

See the *speech-rate property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

stress of type DOMString

See the *stress property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

tableLayout of type DOMString

See the *table-layout property definition* in CSS2.

Exceptions on setting

DOMException **SYNTAX_ERR**: Raised if the new value has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

textAlign of type **DOMString**

See the *text-align property definition* in CSS2.

Exceptions on setting

DOMException **SYNTAX_ERR**: Raised if the new value has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

textDecoration of type **DOMString**

See the *text-decoration property definition* in CSS2.

Exceptions on setting

DOMException **SYNTAX_ERR**: Raised if the new value has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

textIndent of type **DOMString**

See the *text-indent property definition* in CSS2.

Exceptions on setting

DOMException **SYNTAX_ERR**: Raised if the new value has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

textShadow of type **DOMString**

See the *text-shadow property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

textTransform of type **DOMString**

See the *text-transform property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

top of type **DOMString**

See the *top property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

unicodeBidi of type **DOMString**

See the *unicode-bidi property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

verticalAlign of type **DOMString**

See the *vertical-align property definition* in CSS2.

Exceptions on setting

DOMException **SYNTAX_ERR**: Raised if the new value has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

visibility of type **DOMString**

See the *visibility property definition* in CSS2.

Exceptions on setting

DOMException **SYNTAX_ERR**: Raised if the new value has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

voiceFamily of type **DOMString**

See the *voice-family property definition* in CSS2.

Exceptions on setting

DOMException **SYNTAX_ERR**: Raised if the new value has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

volume of type **DOMString**

See the *volume property definition* in CSS2.

Exceptions on setting

DOMException **SYNTAX_ERR**: Raised if the new value has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

whiteSpace of type **DOMString**

See the *white-space property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

widows of type DOMString

See the *widows property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

width of type DOMString

See the *width property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

wordSpacing of type DOMString

See the *word-spacing property definition* in CSS2.

Exceptions on setting

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

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

zIndex of type DOMString

See the *z-index property definition* in CSS2.

Exceptions on setting

DOMException **SYNTAX_ERR**: Raised if the new value has a syntax error and is unparsable.

NO_MODIFICATION_ALLOWED_ERR: Raised if this property is readonly.

Appendix A: IDL Definitions

This appendix contains the complete OMG IDL [OMGIDL] for the Level 2 Document Object Model Style definitions. The definitions are divided into Stylesheets [p.79] and CSS [p.80] .

The IDL files are also available as:

<http://www.w3.org/TR/2000/REC-DOM-Level-2-Style-20001113/idl.zip>

A.1: Document Object Model Style Sheets

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;

    // Introduced in DOM Level 2:
    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;
    };

    // Introduced in DOM Level 2:
    interface StyleSheetList {
        readonly attribute unsigned long   length;
        StyleSheet      item(in unsigned long index);
    };

    // Introduced in DOM Level 2:
    interface MediaList {
        attribute DOMString      mediaText;
        // raises(dom::DOMException) on setting

        readonly attribute unsigned long   length;
        DOMString      item(in unsigned long index);
        void            deleteMedium(in DOMString oldMedium)
    };
};
```

```

        raises(dom::DOMException);
void          appendMedium(in DOMString newMedium)
        raises(dom::DOMException);
};

// Introduced in DOM Level 2:
interface LinkStyle {
    readonly attribute StyleSheet      sheet;
};

// Introduced in DOM Level 2:
interface DocumentStyle {
    readonly attribute StyleSheetList  styleSheets;
};
};

#endif // _STYLESHEETS_IDL_

```

A.2: Document Object Model CSS

css.idl:

```

// File: css.idl

#ifndef _CSS_IDL_
#define _CSS_IDL_

#include "dom.idl"
#include "stylesheets.idl"
#include "views.idl"

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

    typedef dom::DOMString DOMString;
    typedef dom::Element Element;
    typedef dom::DOMImplementation DOMImplementation;

    interface CSSRule;
    interface CSSStyleSheet;
    interface CSSStyleDeclaration;
    interface CSSValue;
    interface Counter;
    interface Rect;
    interface RGBColor;

    // Introduced in DOM Level 2:
    interface CSSRuleList {
        readonly attribute unsigned long    length;
        CSSRule          item(in unsigned long index);
    };

    // Introduced in DOM Level 2:
    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(dom::DOMException) on setting

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

// Introduced in DOM Level 2:
interface CSSStyleRule : CSSRule {
        attribute DOMString          selectorText;
                                   // raises(dom::DOMException) on setting

        readonly attribute CSSStyleDeclaration    style;
};

// Introduced in DOM Level 2:
interface CSSMediaRule : CSSRule {
        readonly attribute StyleSheetList    media;
        readonly attribute CSSRuleList      cssRules;
        unsigned long    insertRule(in DOMString rule,
                                   in unsigned long index)
                                   raises(dom::DOMException);
        void              deleteRule(in unsigned long index)
                                   raises(dom::DOMException);
};

// Introduced in DOM Level 2:
interface CSSFontFaceRule : CSSRule {
        readonly attribute CSSStyleDeclaration    style;
};

// Introduced in DOM Level 2:
interface CSSPageRule : CSSRule {
        attribute DOMString          selectorText;
                                   // raises(dom::DOMException) on setting

        readonly attribute CSSStyleDeclaration    style;
};

// Introduced in DOM Level 2:
interface CSSImportRule : CSSRule {
        readonly attribute DOMString          href;
        readonly attribute StyleSheetList    media;
        readonly attribute CSSStyleSheet      styleSheet;
};

```

```

// Introduced in DOM Level 2:
interface CSSCharsetRule : CSSRule {
    attribute DOMString          encoding;
                                // raises(dom::DOMException) on setting
};

// Introduced in DOM Level 2:
interface CSSUnknownRule : CSSRule {
};

// Introduced in DOM Level 2:
interface CSSStyleDeclaration {
    attribute DOMString          cssText;
                                // raises(dom::DOMException) on setting

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

// Introduced in DOM Level 2:
interface CSSValue {

    // UnitTypes
    const unsigned short    CSS_INHERIT          = 0;
    const unsigned short    CSS_PRIMITIVE_VALUE = 1;
    const unsigned short    CSS_VALUE_LIST       = 2;
    const unsigned short    CSS_CUSTOM          = 3;

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

    readonly attribute unsigned short    cssValueType;
};

// Introduced in DOM Level 2:
interface CSSPrimitiveValue : CSSValue {

    // UnitTypes
    const unsigned short    CSS_UNKNOWN          = 0;
    const unsigned short    CSS_NUMBER          = 1;
    const unsigned short    CSS_PERCENTAGE       = 2;
    const unsigned short    CSS_EMS             = 3;
    const unsigned short    CSS_EXS             = 4;
    const unsigned short    CSS_PX              = 5;
    const unsigned short    CSS_CM              = 6;
    const unsigned short    CSS_MM              = 7;

```

```

const unsigned short    CSS_IN                = 8;
const unsigned short    CSS_PT                = 9;
const unsigned short    CSS_PC                = 10;
const unsigned short    CSS_DEG               = 11;
const unsigned short    CSS_RAD               = 12;
const unsigned short    CSS_GRAD              = 13;
const unsigned short    CSS_MS                = 14;
const unsigned short    CSS_S                 = 15;
const unsigned short    CSS_HZ                = 16;
const unsigned short    CSS_KHZ               = 17;
const unsigned short    CSS_DIMENSION         = 18;
const unsigned short    CSS_STRING            = 19;
const unsigned short    CSS_URI               = 20;
const unsigned short    CSS_IDENT             = 21;
const unsigned short    CSS_ATTR              = 22;
const unsigned short    CSS_COUNTER           = 23;
const unsigned short    CSS_RECT              = 24;
const unsigned short    CSS_RGBCOLOR          = 25;

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

// Introduced in DOM Level 2:
interface CSSValueList : CSSValue {
    readonly attribute unsigned long    length;
    CSSValue                            item(in unsigned long index);
};

// Introduced in DOM Level 2:
interface RGBColor {
    readonly attribute CSSPrimitiveValue    red;
    readonly attribute CSSPrimitiveValue    green;
    readonly attribute CSSPrimitiveValue    blue;
};

// Introduced in DOM Level 2:
interface Rect {
    readonly attribute CSSPrimitiveValue    top;
    readonly attribute CSSPrimitiveValue    right;
    readonly attribute CSSPrimitiveValue    bottom;
    readonly attribute CSSPrimitiveValue    left;
};

```

```

};

// Introduced in DOM Level 2:
interface Counter {
    readonly attribute DOMString      identifier;
    readonly attribute DOMString      listStyle;
    readonly attribute DOMString      separator;
};

// Introduced in DOM Level 2:
interface ElementCSSInlineStyle {
    readonly attribute CSSStyleDeclaration style;
};

// Introduced in DOM Level 2:
interface CSS2Properties {
    attribute DOMString      azimuth;
                                // raises(dom::DOMException) on setting

    attribute DOMString      background;
                                // raises(dom::DOMException) on setting

    attribute DOMString      backgroundAttachment;
                                // raises(dom::DOMException) on setting

    attribute DOMString      backgroundColor;
                                // raises(dom::DOMException) on setting

    attribute DOMString      backgroundImage;
                                // raises(dom::DOMException) on setting

    attribute DOMString      backgroundPosition;
                                // raises(dom::DOMException) on setting

    attribute DOMString      backgroundRepeat;
                                // raises(dom::DOMException) on setting

    attribute DOMString      border;
                                // raises(dom::DOMException) on setting

    attribute DOMString      borderCollapse;
                                // raises(dom::DOMException) on setting

    attribute DOMString      borderColor;
                                // raises(dom::DOMException) on setting

    attribute DOMString      borderSpacing;
                                // raises(dom::DOMException) on setting

    attribute DOMString      borderStyle;
                                // raises(dom::DOMException) on setting

    attribute DOMString      borderTop;
                                // raises(dom::DOMException) on setting

    attribute DOMString      borderRight;
                                // raises(dom::DOMException) on setting

```

css.idl:

attribute DOMString	borderBottom; // raises(dom::DOMException) on setting
attribute DOMString	borderLeft; // raises(dom::DOMException) on setting
attribute DOMString	borderTopColor; // raises(dom::DOMException) on setting
attribute DOMString	borderRightColor; // raises(dom::DOMException) on setting
attribute DOMString	borderBottomColor; // raises(dom::DOMException) on setting
attribute DOMString	borderLeftColor; // raises(dom::DOMException) on setting
attribute DOMString	borderTopStyle; // raises(dom::DOMException) on setting
attribute DOMString	borderRightStyle; // raises(dom::DOMException) on setting
attribute DOMString	borderBottomStyle; // raises(dom::DOMException) on setting
attribute DOMString	borderLeftStyle; // raises(dom::DOMException) on setting
attribute DOMString	borderTopWidth; // raises(dom::DOMException) on setting
attribute DOMString	borderRightWidth; // raises(dom::DOMException) on setting
attribute DOMString	borderBottomWidth; // raises(dom::DOMException) on setting
attribute DOMString	borderLeftWidth; // raises(dom::DOMException) on setting
attribute DOMString	borderWidth; // raises(dom::DOMException) on setting
attribute DOMString	bottom; // raises(dom::DOMException) on setting
attribute DOMString	captionSide; // raises(dom::DOMException) on setting
attribute DOMString	clear; // raises(dom::DOMException) on setting
attribute DOMString	clip; // raises(dom::DOMException) on setting

css.idl:

attribute DOMString	color; // raises(dom::DOMException) on setting
attribute DOMString	content; // raises(dom::DOMException) on setting
attribute DOMString	counterIncrement; // raises(dom::DOMException) on setting
attribute DOMString	counterReset; // raises(dom::DOMException) on setting
attribute DOMString	cue; // raises(dom::DOMException) on setting
attribute DOMString	cueAfter; // raises(dom::DOMException) on setting
attribute DOMString	cueBefore; // raises(dom::DOMException) on setting
attribute DOMString	cursor; // raises(dom::DOMException) on setting
attribute DOMString	direction; // raises(dom::DOMException) on setting
attribute DOMString	display; // raises(dom::DOMException) on setting
attribute DOMString	elevation; // raises(dom::DOMException) on setting
attribute DOMString	emptyCells; // raises(dom::DOMException) on setting
attribute DOMString	cssFloat; // raises(dom::DOMException) on setting
attribute DOMString	font; // raises(dom::DOMException) on setting
attribute DOMString	fontFamily; // raises(dom::DOMException) on setting
attribute DOMString	fontSize; // raises(dom::DOMException) on setting
attribute DOMString	fontSizeAdjust; // raises(dom::DOMException) on setting
attribute DOMString	fontStretch; // raises(dom::DOMException) on setting
attribute DOMString	fontStyle; // raises(dom::DOMException) on setting

css.idl:

```
attribute DOMString    fontVariant;
                        // raises(dom::DOMException) on setting

attribute DOMString    fontWeight;
                        // raises(dom::DOMException) on setting

attribute DOMString    height;
                        // raises(dom::DOMException) on setting

attribute DOMString    left;
                        // raises(dom::DOMException) on setting

attribute DOMString    letterSpacing;
                        // raises(dom::DOMException) on setting

attribute DOMString    lineHeight;
                        // raises(dom::DOMException) on setting

attribute DOMString    listStyle;
                        // raises(dom::DOMException) on setting

attribute DOMString    listStyleImage;
                        // raises(dom::DOMException) on setting

attribute DOMString    listStylePosition;
                        // raises(dom::DOMException) on setting

attribute DOMString    listStyleType;
                        // raises(dom::DOMException) on setting

attribute DOMString    margin;
                        // raises(dom::DOMException) on setting

attribute DOMString    marginTop;
                        // raises(dom::DOMException) on setting

attribute DOMString    marginRight;
                        // raises(dom::DOMException) on setting

attribute DOMString    marginBottom;
                        // raises(dom::DOMException) on setting

attribute DOMString    marginLeft;
                        // raises(dom::DOMException) on setting

attribute DOMString    markerOffset;
                        // raises(dom::DOMException) on setting

attribute DOMString    marks;
                        // raises(dom::DOMException) on setting

attribute DOMString    maxHeight;
                        // raises(dom::DOMException) on setting

attribute DOMString    maxWidth;
```

attribute DOMString	minHeight; // raises(dom::DOMException) on setting
attribute DOMString	minWidth; // raises(dom::DOMException) on setting
attribute DOMString	orphans; // raises(dom::DOMException) on setting
attribute DOMString	outline; // raises(dom::DOMException) on setting
attribute DOMString	outlineColor; // raises(dom::DOMException) on setting
attribute DOMString	outlineStyle; // raises(dom::DOMException) on setting
attribute DOMString	outlineWidth; // raises(dom::DOMException) on setting
attribute DOMString	overflow; // raises(dom::DOMException) on setting
attribute DOMString	padding; // raises(dom::DOMException) on setting
attribute DOMString	paddingTop; // raises(dom::DOMException) on setting
attribute DOMString	paddingRight; // raises(dom::DOMException) on setting
attribute DOMString	paddingBottom; // raises(dom::DOMException) on setting
attribute DOMString	paddingLeft; // raises(dom::DOMException) on setting
attribute DOMString	page; // raises(dom::DOMException) on setting
attribute DOMString	pageBreakAfter; // raises(dom::DOMException) on setting
attribute DOMString	pageBreakBefore; // raises(dom::DOMException) on setting
attribute DOMString	pageBreakInside; // raises(dom::DOMException) on setting
attribute DOMString	pause; // raises(dom::DOMException) on setting
attribute DOMString	pauseAfter; // raises(dom::DOMException) on setting

```

attribute DOMString    pauseBefore;
                        // raises(dom::DOMException) on setting

attribute DOMString    pitch;
                        // raises(dom::DOMException) on setting

attribute DOMString    pitchRange;
                        // raises(dom::DOMException) on setting

attribute DOMString    playDuring;
                        // raises(dom::DOMException) on setting

attribute DOMString    position;
                        // raises(dom::DOMException) on setting

attribute DOMString    quotes;
                        // raises(dom::DOMException) on setting

attribute DOMString    richness;
                        // raises(dom::DOMException) on setting

attribute DOMString    right;
                        // raises(dom::DOMException) on setting

attribute DOMString    size;
                        // raises(dom::DOMException) on setting

attribute DOMString    speak;
                        // raises(dom::DOMException) on setting

attribute DOMString    speakHeader;
                        // raises(dom::DOMException) on setting

attribute DOMString    speakNumeral;
                        // raises(dom::DOMException) on setting

attribute DOMString    speakPunctuation;
                        // raises(dom::DOMException) on setting

attribute DOMString    speechRate;
                        // raises(dom::DOMException) on setting

attribute DOMString    stress;
                        // raises(dom::DOMException) on setting

attribute DOMString    tableLayout;
                        // raises(dom::DOMException) on setting

attribute DOMString    textAlign;
                        // raises(dom::DOMException) on setting

attribute DOMString    textDecoration;
                        // raises(dom::DOMException) on setting

attribute DOMString    textIndent;
                        // raises(dom::DOMException) on setting

```

css.idl:

```
    attribute DOMString      textShadow;
                                // raises(dom::DOMException) on setting

    attribute DOMString      textTransform;
                                // raises(dom::DOMException) on setting

    attribute DOMString      top;
                                // raises(dom::DOMException) on setting

    attribute DOMString      unicodeBidi;
                                // raises(dom::DOMException) on setting

    attribute DOMString      verticalAlign;
                                // raises(dom::DOMException) on setting

    attribute DOMString      visibility;
                                // raises(dom::DOMException) on setting

    attribute DOMString      voiceFamily;
                                // raises(dom::DOMException) on setting

    attribute DOMString      volume;
                                // raises(dom::DOMException) on setting

    attribute DOMString      whiteSpace;
                                // raises(dom::DOMException) on setting

    attribute DOMString      widows;
                                // raises(dom::DOMException) on setting

    attribute DOMString      width;
                                // raises(dom::DOMException) on setting

    attribute DOMString      wordSpacing;
                                // raises(dom::DOMException) on setting

    attribute DOMString      zIndex;
                                // raises(dom::DOMException) on setting

};

// Introduced in DOM Level 2:
interface CSSStyleSheet : stylesheets::StyleSheet {
    readonly attribute CSSRule      ownerRule;
    readonly attribute CSSRuleList  cssRules;
    unsigned long    insertRule(in DOMString rule,
                                in unsigned long index)
                                raises(dom::DOMException);
    void              deleteRule(in unsigned long index)
                                raises(dom::DOMException);
};

// Introduced in DOM Level 2:
interface ViewCSS : views::AbstractView {
    CSSStyleDeclaration getComputedStyle(in Element elt,
                                in DOMString pseudoElt);
};
```

```

};

// Introduced in DOM Level 2:
interface DocumentCSS : stylesheets::DocumentStyle {
    CSSStyleDeclaration getOverrideStyle(in Element elt,
                                         in DOMString pseudoElt);
};

// Introduced in DOM Level 2:
interface DOMImplementationCSS : DOMImplementation {
    CSSStyleSheet createCSSStyleSheet(in DOMString title,
                                     in DOMString media)
        raises(DOMException);
};
};

#endif // _CSS_IDL_

```

css.idl:

Appendix B: Java Language Binding

This appendix contains the complete Java Language [Java] binding for the Level 2 Document Object Model Style. The definitions are divided into StyleSheets [p.93] and CSS [p.94] .

The Java files are also available as

<http://www.w3.org/TR/2000/REC-DOM-Level-2-Style-20001113/java-binding.zip>

B.1: Document Object Model Style Sheets

org/w3c/dom/stylesheets/StyleSheet.java:

```
package org.w3c.dom.stylesheets;

import org.w3c.dom.Node;

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;

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.DOMException;

public interface MediaList {
```

```

    public String getMediaText();
    public void setMediaText(String mediaText)
        throws DOMException;

    public int getLength();

    public String item(int index);

    public void deleteMedium(String oldMedium)
        throws DOMException;

    public void appendMedium(String newMedium)
        throws DOMException;
}

```

org/w3c/dom/stylesheets/LinkStyle.java:

```

package org.w3c.dom.stylesheets;

public interface LinkStyle {
    public StyleSheet getSheet();
}

```

org/w3c/dom/stylesheets/DocumentStyle.java:

```

package org.w3c.dom.stylesheets;

public interface DocumentStyle {
    public StyleSheetList getStyleSheets();
}

```

B.2: Document Object Model CSS

org/w3c/dom/css/CSSStyleSheet.java:

```

package org.w3c.dom.css;

import org.w3c.dom.DOMException;
import org.w3c.dom.stylesheets.StyleSheet;

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;

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.DOMException;

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.DOMException;

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.DOMException;  
import org.w3c.dom.stylesheets.MediaList;  
  
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;  
  
public interface CSSFontFaceRule extends CSSRule {  
    public CSSStyleDeclaration getStyle();  
}
```

org/w3c/dom/css/CSSPageRule.java:

```
package org.w3c.dom.css;  
  
import org.w3c.dom.DOMException;  
  
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.stylesheets.MediaList;

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.DOMException;

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;

public interface CSSUnknownRule extends CSSRule {
}
```

org/w3c/dom/css/CSSStyleDeclaration.java:

```
package org.w3c.dom.css;

import org.w3c.dom.DOMException;

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.DOMException;

public interface CSSValue {
    // UnitTypes
    public static final short CSS_INHERIT           = 0;
    public static final short CSS_PRIMITIVE_VALUE  = 1;
    public static final short CSS_VALUE_LIST       = 2;
    public static final short CSS_CUSTOM           = 3;

    public String getCssText();
    public void setCssText(String cssText)
                  throws DOMException;

    public short getCssValueType();
}

```

org.w3c.dom.css.CSSPrimitiveValue.java:

```

package org.w3c.dom.css;

import org.w3c.dom.DOMException;

public interface CSSPrimitiveValue extends CSSValue {
    // UnitTypes
    public static final short CSS_UNKNOWN           = 0;
    public static final short CSS_NUMBER           = 1;
    public static final short CSS_PERCENTAGE       = 2;
    public static final short CSS_EMS              = 3;
    public static final short CSS_EXS              = 4;
    public static final short CSS_PX               = 5;
    public static final short CSS_CM               = 6;
    public static final short CSS_MM               = 7;
    public static final short CSS_IN               = 8;
    public static final short CSS_PT               = 9;
    public static final short CSS_PC               = 10;
    public static final short CSS_DEG              = 11;
    public static final short CSS_RAD              = 12;
    public static final short CSS_GRAD             = 13;
}

```

```

public static final short CSS_MS           = 14;
public static final short CSS_S           = 15;
public static final short CSS_HZ           = 16;
public static final short CSS_KHZ         = 17;
public static final short CSS_DIMENSION   = 18;
public static final short CSS_STRING      = 19;
public static final short CSS_URI          = 20;
public static final short CSS_IDENT        = 21;
public static final short CSS_ATTR         = 22;
public static final short CSS_COUNTER      = 23;
public static final short CSS_RECT         = 24;
public static final short CSS_RGBCOLOR     = 25;

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;

public interface CSSValueList extends CSSValue {
    public int getLength();

    public CSSValue item(int index);
}

```

org/w3c/dom/css/RGBColor.java:

```
package org.w3c.dom.css;

public interface RGBColor {
    public CSSPrimitiveValue getRed();

    public CSSPrimitiveValue getGreen();

    public CSSPrimitiveValue getBlue();
}
```

org/w3c/dom/css/Rect.java:

```
package org.w3c.dom.css;

public interface Rect {
    public CSSPrimitiveValue getTop();

    public CSSPrimitiveValue getRight();

    public CSSPrimitiveValue getBottom();

    public CSSPrimitiveValue getLeft();
}
```

org/w3c/dom/css/Counter.java:

```
package org.w3c.dom.css;

public interface Counter {
    public String getIdentifier();

    public String getListStyle();

    public String getSeparator();
}
```

org/w3c/dom/css/ViewCSS.java:

```
package org.w3c.dom.css;

import org.w3c.dom.views.AbstractView;
import org.w3c.dom.Element;

public interface ViewCSS extends AbstractView {
    public CSSStyleDeclaration getComputedStyle(Element elt,
                                                String pseudoElt);
}
```

org/w3c/dom/css/DocumentCSS.java:

```
package org.w3c.dom.css;

import org.w3c.dom.stylesheets.DocumentStyle;
import org.w3c.dom.Element;

public interface DocumentCSS extends DocumentStyle {
    public CSSStyleDeclaration getOverrideStyle(Element elt,
                                                String pseudoElt);
}
```

org/w3c/dom/css/DOMImplementationCSS.java:

```
package org.w3c.dom.css;

import org.w3c.dom.DOMImplementation;
import org.w3c.dom.DOMException;

public interface DOMImplementationCSS extends DOMImplementation {
    public CSSStyleSheet createCSSStyleSheet(String title,
                                             String media)
        throws DOMException;
}
```

org/w3c/dom/css/ElementCSSInlineStyle.java:

```
package org.w3c.dom.css;

public interface ElementCSSInlineStyle {
    public CSSStyleDeclaration getStyle();
}
```

org/w3c/dom/css/CSS2Properties.java:

```
package org.w3c.dom.css;

import org.w3c.dom.DOMException;

public interface CSS2Properties {
    public String getAzimuth();
    public void setAzimuth(String azimuth)
        throws DOMException;

    public String getBackground();
    public void setBackground(String background)
        throws DOMException;

    public String getBackgroundAttachment();
    public void setBackgroundAttachment(String backgroundAttachment)
        throws DOMException;
}
```

```
public String getBackgroundColor();
public void setBackgroundColor(String backgroundColor)
                                throws DOMException;

public String getBackgroundImage();
public void setBackgroundImage(String backgroundImage)
                                throws DOMException;

public String getBackgroundPosition();
public void setBackgroundPosition(String backgroundPosition)
                                throws DOMException;

public String getBackgroundRepeat();
public void setBackgroundRepeat(String backgroundRepeat)
                                throws DOMException;

public String getBorder();
public void setBorder(String border)
                                throws DOMException;

public String getBorderCollapse();
public void setBorderCollapse(String borderCollapse)
                                throws DOMException;

public String getBorderColor();
public void setBorderColor(String borderColor)
                                throws DOMException;

public String getBorderSpacing();
public void setBorderSpacing(String borderSpacing)
                                throws DOMException;

public String getBorderStyle();
public void setBorderStyle(String borderStyle)
                                throws DOMException;

public String getBorderTop();
public void setBorderTop(String borderTop)
                                throws DOMException;

public String getBorderRight();
public void setBorderRight(String borderRight)
                                throws DOMException;

public String getBorderBottom();
public void setBorderBottom(String borderBottom)
                                throws DOMException;

public String getBorderLeft();
public void setBorderLeft(String borderLeft)
                                throws DOMException;

public String getBorderTopColor();
public void setBorderTopColor(String borderTopColor)
                                throws DOMException;

public String getBorderRightColor();
```

```
public void setBorderRightColor(String borderRightColor)
                                throws DOMException;

public String getBorderBottomColor();
public void setBorderBottomColor(String borderBottomColor)
                                throws DOMException;

public String getBorderLeftColor();
public void setBorderLeftColor(String borderLeftColor)
                                throws DOMException;

public String getBorderTopStyle();
public void setBorderTopStyle(String borderTopStyle)
                                throws DOMException;

public String getBorderRightStyle();
public void setBorderRightStyle(String borderRightStyle)
                                throws DOMException;

public String getBorderBottomStyle();
public void setBorderBottomStyle(String borderBottomStyle)
                                throws DOMException;

public String getBorderLeftStyle();
public void setBorderLeftStyle(String borderLeftStyle)
                                throws DOMException;

public String getBorderTopWidth();
public void setBorderTopWidth(String borderTopWidth)
                                throws DOMException;

public String getBorderRightWidth();
public void setBorderRightWidth(String borderRightWidth)
                                throws DOMException;

public String getBorderBottomWidth();
public void setBorderBottomWidth(String borderBottomWidth)
                                throws DOMException;

public String getBorderLeftWidth();
public void setBorderLeftWidth(String borderLeftWidth)
                                throws DOMException;

public String getBorderWidth();
public void setBorderWidth(String borderWidth)
                                throws DOMException;

public String getBottom();
public void setBottom(String bottom)
                                throws DOMException;

public String getCaptionSide();
public void setCaptionSide(String captionSide)
                                throws DOMException;

public String getClear();
public void setClear(String clear)
```

```
throws DOMException;

public String getClip();
public void setClip(String clip)
throws DOMException;

public String getColor();
public void setColor(String color)
throws DOMException;

public String getContent();
public void setContent(String content)
throws DOMException;

public String getCounterIncrement();
public void setCounterIncrement(String counterIncrement)
throws DOMException;

public String getCounterReset();
public void setCounterReset(String counterReset)
throws DOMException;

public String getCue();
public void setCue(String cue)
throws DOMException;

public String getCueAfter();
public void setCueAfter(String cueAfter)
throws DOMException;

public String getCueBefore();
public void setCueBefore(String cueBefore)
throws DOMException;

public String getCursor();
public void setCursor(String cursor)
throws DOMException;

public String getDirection();
public void setDirection(String direction)
throws DOMException;

public String getDisplay();
public void setDisplay(String display)
throws DOMException;

public String getElevation();
public void setElevation(String elevation)
throws DOMException;

public String getEmptyCells();
public void setEmptyCells(String emptyCells)
throws DOMException;

public String getCssFloat();
public void setCssFloat(String cssFloat)
throws DOMException;
```

```
public String getFont();
public void setFont(String font)
                                throws DOMException;

public String getFontFamily();
public void setFontFamily(String fontFamily)
                                throws DOMException;

public String getFontSize();
public void setFontSize(String fontSize)
                                throws DOMException;

public String getFontSizeAdjust();
public void setFontSizeAdjust(String fontSizeAdjust)
                                throws DOMException;

public String getFontStretch();
public void setFontStretch(String fontStretch)
                                throws DOMException;

public String getFontStyle();
public void setFontStyle(String fontStyle)
                                throws DOMException;

public String getFontVariant();
public void setFontVariant(String fontVariant)
                                throws DOMException;

public String getFontWeight();
public void setFontWeight(String fontWeight)
                                throws DOMException;

public String getHeight();
public void setHeight(String height)
                                throws DOMException;

public String getLeft();
public void setLeft(String left)
                                throws DOMException;

public String getLetterSpacing();
public void setLetterSpacing(String letterSpacing)
                                throws DOMException;

public String getLineHeight();
public void setLineHeight(String lineHeight)
                                throws DOMException;

public String getListStyle();
public void setListStyle(String listStyle)
                                throws DOMException;

public String getListStyleImage();
public void setListStyleImage(String listStyleImage)
                                throws DOMException;
```

```
public String getListStylePosition();
public void setListStylePosition(String listStylePosition)
                                throws DOMException;

public String getListStyleType();
public void setListStyleType(String listStyleType)
                                throws DOMException;

public String getMargin();
public void setMargin(String margin)
                                throws DOMException;

public String getMarginTop();
public void setMarginTop(String marginTop)
                                throws DOMException;

public String getMarginRight();
public void setMarginRight(String marginRight)
                                throws DOMException;

public String getMarginBottom();
public void setMarginBottom(String marginBottom)
                                throws DOMException;

public String getMarginLeft();
public void setMarginLeft(String marginLeft)
                                throws DOMException;

public String getMarkerOffset();
public void setMarkerOffset(String markerOffset)
                                throws DOMException;

public String getMarks();
public void setMarks(String marks)
                                throws DOMException;

public String getMaxHeight();
public void setMaxHeight(String maxHeight)
                                throws DOMException;

public String getMaxWidth();
public void setMaxWidth(String maxWidth)
                                throws DOMException;

public String getMinHeight();
public void setMinHeight(String minHeight)
                                throws DOMException;

public String getMinWidth();
public void setMinWidth(String minWidth)
                                throws DOMException;

public String getOrphans();
public void setOrphans(String orphans)
                                throws DOMException;

public String getOutline();
```

```
public void setOutline(String outline)
                                throws DOMException;

public String getOutlineColor();
public void setOutlineColor(String outlineColor)
                                throws DOMException;

public String getOutlineStyle();
public void setOutlineStyle(String outlineStyle)
                                throws DOMException;

public String getOutlineWidth();
public void setOutlineWidth(String outlineWidth)
                                throws DOMException;

public String getOverflow();
public void setOverflow(String overflow)
                                throws DOMException;

public String getPadding();
public void setPadding(String padding)
                                throws DOMException;

public String getPaddingTop();
public void setPaddingTop(String paddingTop)
                                throws DOMException;

public String getPaddingRight();
public void setPaddingRight(String paddingRight)
                                throws DOMException;

public String getPaddingBottom();
public void setPaddingBottom(String paddingBottom)
                                throws DOMException;

public String getPaddingLeft();
public void setPaddingLeft(String paddingLeft)
                                throws DOMException;

public String getPage();
public void setPage(String page)
                                throws DOMException;

public String getPageBreakAfter();
public void setPageBreakAfter(String pageBreakAfter)
                                throws DOMException;

public String getPageBreakBefore();
public void setPageBreakBefore(String pageBreakBefore)
                                throws DOMException;

public String getPageBreakInside();
public void setPageBreakInside(String pageBreakInside)
                                throws DOMException;

public String getPause();
public void setPause(String pause)
```

```
throws DOMException;

public String getPauseAfter();
public void setPauseAfter(String pauseAfter)
    throws DOMException;

public String getPauseBefore();
public void setPauseBefore(String pauseBefore)
    throws DOMException;

public String getPitch();
public void setPitch(String pitch)
    throws DOMException;

public String getPitchRange();
public void setPitchRange(String pitchRange)
    throws DOMException;

public String getPlayDuring();
public void setPlayDuring(String playDuring)
    throws DOMException;

public String getPosition();
public void setPosition(String position)
    throws DOMException;

public String getQuotes();
public void setQuotes(String quotes)
    throws DOMException;

public String getRichness();
public void setRichness(String richness)
    throws DOMException;

public String getRight();
public void setRight(String right)
    throws DOMException;

public String getSize();
public void setSize(String size)
    throws DOMException;

public String getSpeak();
public void setSpeak(String speak)
    throws DOMException;

public String getSpeakHeader();
public void setSpeakHeader(String speakHeader)
    throws DOMException;

public String getSpeakNumeral();
public void setSpeakNumeral(String speakNumeral)
    throws DOMException;

public String getSpeakPunctuation();
public void setSpeakPunctuation(String speakPunctuation)
    throws DOMException;
```

```
public String getSpeechRate();
public void setSpeechRate(String speechRate)
                                throws DOMException;

public String getStress();
public void setStress(String stress)
                                throws DOMException;

public String getTableLayout();
public void setTableLayout(String tableLayout)
                                throws DOMException;

public String getTextAlign();
public void setTextAlign(String textAlign)
                                throws DOMException;

public String getTextDecoration();
public void setTextDecoration(String textDecoration)
                                throws DOMException;

public String getTextIndent();
public void setTextIndent(String textIndent)
                                throws DOMException;

public String getTextShadow();
public void setTextShadow(String textShadow)
                                throws DOMException;

public String getTextTransform();
public void setTextTransform(String textTransform)
                                throws DOMException;

public String getTop();
public void setTop(String top)
                                throws DOMException;

public String getUnicodeBidi();
public void setUnicodeBidi(String unicodeBidi)
                                throws DOMException;

public String getVerticalAlign();
public void setVerticalAlign(String verticalAlign)
                                throws DOMException;

public String getVisibility();
public void setVisibility(String visibility)
                                throws DOMException;

public String getVoiceFamily();
public void setVoiceFamily(String voiceFamily)
                                throws DOMException;

public String getVolume();
public void setVolume(String volume)
                                throws DOMException;
```

```
public String getWhiteSpace();
public void setWhiteSpace(String whiteSpace)
                                throws DOMException;

public String getWidows();
public void setWidows(String widows)
                                throws DOMException;

public String getWidth();
public void setWidth(String width)
                                throws DOMException;

public String getWordSpacing();
public void setWordSpacing(String wordSpacing)
                                throws DOMException;

public String getZIndex();
public void setZIndex(String zIndex)
                                throws DOMException;
}
```

Appendix C: ECMAScript Language Binding

This appendix contains the complete ECMAScript [ECMAScript] binding for the Level 2 Document Object Model Style definitions. The definitions are divided into StyleSheets [p.111] and CSS [p.112] .

Note: Exceptions handling is only supported by ECMAScript implementation conformant with the Standard ECMA-262 3rd. Edition ([ECMAScript]).

C.1: Document Object Model StyleSheets

Object **StyleSheet**

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

type

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

disabled

This property is of type **Boolean**.

ownerNode

This read-only property is a **Node** object.

parentStyleSheet

This read-only property is a **StyleSheet** object.

href

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

title

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

media

This read-only property is a **MediaList** object.

Object **StyleSheetList**

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

length

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

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

item(index)

This method returns a **StyleSheet** object.

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

Note: This object can also be dereferenced using square bracket notation (e.g. obj[1]).

Dereferencing with an integer **index** is equivalent to invoking the **item** method with that index.

Object **MediaList**

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

mediaText

This property is of type **String** and can raise a **DOMException** object on setting.

length

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

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

item(index)

This method returns a **String**.

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

Note: This object can also be dereferenced using square bracket notation (e.g. obj[1]).

Dereferencing with an integer **index** is equivalent to invoking the **item** method with that index.

deleteMedium(oldMedium)

This method has no return value.

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

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

appendMedium(newMedium)

This method has no return value.

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

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

Object **LinkStyle**

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

sheet

This read-only property is a **StyleSheet** object.

Object **DocumentStyle**

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

styleSheets

This read-only property is a **StyleSheetList** object.

C.2: Document Object Model CSS

Object **CSSStyleSheet**

CSSStyleSheet has all the properties and methods of the **StyleSheet** object as well as the properties and methods defined below.

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

ownerRule

This read-only property is a **CSSRule** object.

cssRules

This read-only property is a **CSSRuleList** object.

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

insertRule(rule, index)

This method returns a **Number**.

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

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

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

deleteRule(index)

This method has no return value.

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

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

Object **CSSRuleList**

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

length

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

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

item(index)

This method returns a **CSSRule** object.

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

Note: This object can also be dereferenced using square bracket notation (e.g. obj[1]).

Dereferencing with an integer **index** is equivalent to invoking the **item** method with that index.

Prototype Object **CSSRule**

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

CSSRule.UNKNOWN_RULE

This constant is of type **Number** and its value is **0**.

CSSRule.STYLE_RULE

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

CSSRule.CHARSET_RULE

This constant is of type **Number** and its value is **2**.

CSSRule.IMPORT_RULE

This constant is of type **Number** and its value is **3**.

CSSRule.MEDIA_RULE

This constant is of type **Number** and its value is **4**.

CSSRule.FONT_FACE_RULE

This constant is of type **Number** and its value is **5**.

CSSRule.PAGE_RULE

This constant is of type **Number** and its value is **6**.

Object **CSSRule**

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

type

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

cssText

This property is of type **String** and can raise a **DOMException** object on setting.

parentStyleSheet

This read-only property is a **CSSStyleSheet** object.

parentRule

This read-only property is a **CSSRule** object.

Object **CSSStyleRule**

CSSStyleRule has all the properties and methods of the **CSSRule** object as well as the properties and methods defined below.

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

selectorText

This property is of type **String** and can raise a **DOMException** object on setting.

style

This read-only property is a **CSSStyleDeclaration** object.

Object **CSSMediaRule**

CSSMediaRule has all the properties and methods of the **CSSRule** object as well as the properties and methods defined below.

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

media

This read-only property is a **MediaList** object.

cssRules

This read-only property is a **CSSRuleList** object.

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

insertRule(rule, index)

This method returns a **Number**.

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

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

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

deleteRule(index)

This method has no return value.

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

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

Object **CSSFontFaceRule**

CSSFontFaceRule has all the properties and methods of the **CSSRule** object as well as the properties and methods defined below.

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

style

This read-only property is a **CSSStyleDeclaration** object.

Object **CSSPageRule**

CSSPageRule has all the properties and methods of the **CSSRule** object as well as the properties and methods defined below.

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

selectorText

This property is of type **String** and can raise a **DOMException** object on setting.

style

This read-only property is a **CSSStyleDeclaration** object.

Object **CSSImportRule**

CSSImportRule has all the properties and methods of the **CSSRule** object as well as the properties and methods defined below.

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

href

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

media

This read-only property is a **MediaList** object.

styleSheet

This read-only property is a **CSSStyleSheet** object.

Object **CSSCharsetRule**

CSSCharsetRule has all the properties and methods of the **CSSRule** object as well as the properties and methods defined below.

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

encoding

This property is of type **String** and can raise a **DOMException** object on setting.

Object **CSSUnknownRule**

CSSUnknownRule has the all the properties and methods of the **CSSRule** object 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** and can raise a **DOMException** object on setting.

length

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

parentRule

This read-only property is a **CSSRule** object.

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

getPropertyValue(propertyName)

This method returns a **String**.

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

getPropertyCSSValue(propertyName)

This method returns a **CSSValue** object.

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

removeProperty(propertyName)

This method returns a **String**.

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

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

getPropertyPriority(propertyName)

This method returns a **String**.

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

setProperty(propertyName, value, priority)

This method has no return value.

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

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

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

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

item(index)

This method returns a **String**.

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

Note: This object can also be dereferenced using square bracket notation (e.g. obj[1]).

Dereferencing with an integer **index** is equivalent to invoking the **item** method with that index.

Prototype Object **CSSValue**

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

CSSValue.CSS_INHERIT

This constant is of type **Number** and its value is **0**.

CSSValue.CSS_PRIMITIVE_VALUE

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

CSSValue.CSS_VALUE_LIST

This constant is of type **Number** and its value is **2**.

CSSValue.CSS_CUSTOM

This constant is of type **Number** and its value is **3**.

Object CSSValue

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

cssText

This property is of type **String** and can raise a **DOMException** object on setting.

cssValueType

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

Prototype Object CSSPrimitiveValue

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

CSSPrimitiveValue.CSS_UNKNOWN

This constant is of type **Number** and its value is **0**.

CSSPrimitiveValue.CSS_NUMBER

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

CSSPrimitiveValue.CSS_PERCENTAGE

This constant is of type **Number** and its value is **2**.

CSSPrimitiveValue.CSS_EMS

This constant is of type **Number** and its value is **3**.

CSSPrimitiveValue.CSS_EXS

This constant is of type **Number** and its value is **4**.

CSSPrimitiveValue.CSS_PX

This constant is of type **Number** and its value is **5**.

CSSPrimitiveValue.CSS_CM

This constant is of type **Number** and its value is **6**.

CSSPrimitiveValue.CSS_MM

This constant is of type **Number** and its value is **7**.

CSSPrimitiveValue.CSS_IN

This constant is of type **Number** and its value is **8**.

CSSPrimitiveValue.CSS_PT

This constant is of type **Number** and its value is **9**.

CSSPrimitiveValue.CSS_PC

This constant is of type **Number** and its value is **10**.

CSSPrimitiveValue.CSS_DEG

This constant is of type **Number** and its value is **11**.

CSSPrimitiveValue.CSS_RAD

This constant is of type **Number** and its value is **12**.

CSSPrimitiveValue.CSS_GRAD

This constant is of type **Number** and its value is **13**.

CSSPrimitiveValue.CSS_MS

This constant is of type **Number** and its value is **14**.

CSSPrimitiveValue.CSS_S

This constant is of type **Number** and its value is **15**.

CSSPrimitiveValue.CSS_HZ

This constant is of type **Number** and its value is **16**.

CSSPrimitiveValue.CSS_KHZ

This constant is of type **Number** and its value is **17**.

CSSPrimitiveValue.CSS_DIMENSION

This constant is of type **Number** and its value is **18**.

CSSPrimitiveValue.CSS_STRING

This constant is of type **Number** and its value is **19**.

CSSPrimitiveValue.CSS_URI

This constant is of type **Number** and its value is **20**.

CSSPrimitiveValue.CSS_IDENT

This constant is of type **Number** and its value is **21**.

CSSPrimitiveValue.CSS_ATTR

This constant is of type **Number** and its value is **22**.

CSSPrimitiveValue.CSS_COUNTER

This constant is of type **Number** and its value is **23**.

CSSPrimitiveValue.CSS_RECT

This constant is of type **Number** and its value is **24**.

CSSPrimitiveValue.CSS_RGBCOLOR

This constant is of type **Number** and its value is **25**.

Object CSSPrimitiveValue

CSSPrimitiveValue has the all the properties and methods of the **CSSValue** object as well as the properties and methods defined below.

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

primitiveType

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

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

setFloatValue(unitType, floatValue)

This method has no return value.

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

The **floatValue** parameter is a **float** object.

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

getFloatValue(unitType)

This method returns a **float** object.

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

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

setStringValue(stringType, stringValue)

This method has no return value.

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

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

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

getStringValue()

This method returns a **String**.

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

getCounterValue()

This method returns a **Counter** object.

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

getRectValue()

This method returns a **Rect** object.

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

getRGBColorValue()

This method returns a **RGBColor** object.

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

Object **CSSValueList**

CSSValueList has all the properties and methods of the **CSSValue** object as well as the properties and methods defined below.

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

length

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

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

item(index)

This method returns a **CSSValue** object.

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

Note: This object can also be dereferenced using square bracket notation (e.g. obj[1]).

Dereferencing with an integer **index** is equivalent to invoking the **item** method with that index.

Object **RGBColor**

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

red

This read-only property is a **CSSPrimitiveValue** object.

green

This read-only property is a **CSSPrimitiveValue** object.

blue

This read-only property is a **CSSPrimitiveValue** object.

Object **Rect**

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

top

This read-only property is a **CSSPrimitiveValue** object.

right

This read-only property is a **CSSPrimitiveValue** object.

bottom

This read-only property is a **CSSPrimitiveValue** object.

left

This read-only property is a **CSSPrimitiveValue** object.

Object **Counter**

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

identifier

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

listStyle

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

separator

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

Object ViewCSS

ViewCSS has the all the properties and methods of the **AbstractView** object as well as the properties and methods defined below.

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

getComputedStyle(elt, pseudoElt)

This method returns a **CSSStyleDeclaration** object.

The **elt** parameter is a **Element** object.

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

Object DocumentCSS

DocumentCSS has the all the properties and methods of the **DocumentStyle** object as well as the properties and methods defined below.

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

getOverrideStyle(elt, pseudoElt)

This method returns a **CSSStyleDeclaration** object.

The **elt** parameter is a **Element** object.

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

Object DOMImplementationCSS

DOMImplementationCSS has the all the properties and methods of the **DOMImplementation** object as well as the properties and methods defined below.

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

createCSSStyleSheet(title, media)

This method returns a **CSSStyleSheet** object.

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

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

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

Object ElementCSSInlineStyle

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

style

This read-only property is a **CSSStyleDeclaration** object.

Object CSS2Properties

The **CSS2Properties** object has the following properties:

azimuth

This property is of type **String** and can raise a **DOMException** object on setting.

background

This property is of type **String** and can raise a **DOMException** object on setting.

backgroundAttachment

This property is of type **String** and can raise a **DOMException** object on setting.

backgroundColor

This property is of type **String** and can raise a **DOMException** object on setting.

backgroundImage

This property is of type **String** and can raise a **DOMException** object on setting.

backgroundPosition

This property is of type **String** and can raise a **DOMException** object on setting.

backgroundRepeat

This property is of type **String** and can raise a **DOMException** object on setting.

border

This property is of type **String** and can raise a **DOMException** object on setting.

borderCollapse

This property is of type **String** and can raise a **DOMException** object on setting.

borderColor

This property is of type **String** and can raise a **DOMException** object on setting.

borderSpacing

This property is of type **String** and can raise a **DOMException** object on setting.

borderStyle

This property is of type **String** and can raise a **DOMException** object on setting.

borderTop

This property is of type **String** and can raise a **DOMException** object on setting.

borderRight

This property is of type **String** and can raise a **DOMException** object on setting.

borderBottom

This property is of type **String** and can raise a **DOMException** object on setting.

borderLeft

This property is of type **String** and can raise a **DOMException** object on setting.

borderTopColor

This property is of type **String** and can raise a **DOMException** object on setting.

borderRightColor

This property is of type **String** and can raise a **DOMException** object on setting.

borderBottomColor

This property is of type **String** and can raise a **DOMException** object on setting.

borderLeftColor

This property is of type **String** and can raise a **DOMException** object on setting.

borderTopStyle

This property is of type **String** and can raise a **DOMException** object on setting.

borderRightStyle

This property is of type **String** and can raise a **DOMException** object on setting.

borderBottomStyle

This property is of type **String** and can raise a **DOMException** object on setting.

borderLeftStyle

This property is of type **String** and can raise a **DOMException** object on setting.

borderTopWidth

This property is of type **String** and can raise a **DOMException** object on setting.

borderRightWidth

This property is of type **String** and can raise a **DOMException** object on setting.

borderBottomWidth

This property is of type **String** and can raise a **DOMException** object on setting.

borderLeftWidth

This property is of type **String** and can raise a **DOMException** object on setting.

borderWidth

This property is of type **String** and can raise a **DOMException** object on setting.

bottom

This property is of type **String** and can raise a **DOMException** object on setting.

captionSide

This property is of type **String** and can raise a **DOMException** object on setting.

clear

This property is of type **String** and can raise a **DOMException** object on setting.

clip

This property is of type **String** and can raise a **DOMException** object on setting.

color

This property is of type **String** and can raise a **DOMException** object on setting.

content

This property is of type **String** and can raise a **DOMException** object on setting.

counterIncrement

This property is of type **String** and can raise a **DOMException** object on setting.

counterReset

This property is of type **String** and can raise a **DOMException** object on setting.

cue

This property is of type **String** and can raise a **DOMException** object on setting.

cueAfter

This property is of type **String** and can raise a **DOMException** object on setting.

cueBefore

This property is of type **String** and can raise a **DOMException** object on setting.

cursor

This property is of type **String** and can raise a **DOMException** object on setting.

direction

This property is of type **String** and can raise a **DOMException** object on setting.

display

This property is of type **String** and can raise a **DOMException** object on setting.

elevation

This property is of type **String** and can raise a **DOMException** object on setting.

emptyCells

This property is of type **String** and can raise a **DOMException** object on setting.

cssFloat

This property is of type **String** and can raise a **DOMException** object on setting.

font

This property is of type **String** and can raise a **DOMException** object on setting.

fontFamily

This property is of type **String** and can raise a **DOMException** object on setting.

fontSize

This property is of type **String** and can raise a **DOMException** object on setting.

fontSizeAdjust

This property is of type **String** and can raise a **DOMException** object on setting.

fontStretch

This property is of type **String** and can raise a **DOMException** object on setting.

fontStyle

This property is of type **String** and can raise a **DOMException** object on setting.

fontVariant

This property is of type **String** and can raise a **DOMException** object on setting.

fontWeight

This property is of type **String** and can raise a **DOMException** object on setting.

height

This property is of type **String** and can raise a **DOMException** object on setting.

left

This property is of type **String** and can raise a **DOMException** object on setting.

letterSpacing

This property is of type **String** and can raise a **DOMException** object on setting.

lineHeight

This property is of type **String** and can raise a **DOMException** object on setting.

listStyle

This property is of type **String** and can raise a **DOMException** object on setting.

listStyleImage

This property is of type **String** and can raise a **DOMException** object on setting.

listStylePosition

This property is of type **String** and can raise a **DOMException** object on setting.

listStyleType

This property is of type **String** and can raise a **DOMException** object on setting.

margin

This property is of type **String** and can raise a **DOMException** object on setting.

marginTop

This property is of type **String** and can raise a **DOMException** object on setting.

marginRight

This property is of type **String** and can raise a **DOMException** object on setting.

marginBottom

This property is of type **String** and can raise a **DOMException** object on setting.

marginLeft

This property is of type **String** and can raise a **DOMException** object on setting.

markerOffset

This property is of type **String** and can raise a **DOMException** object on setting.

marks

This property is of type **String** and can raise a **DOMException** object on setting.

maxHeight

This property is of type **String** and can raise a **DOMException** object on setting.

maxWidth

This property is of type **String** and can raise a **DOMException** object on setting.

minHeight

This property is of type **String** and can raise a **DOMException** object on setting.

minWidth

This property is of type **String** and can raise a **DOMException** object on setting.

orphans

This property is of type **String** and can raise a **DOMException** object on setting.

outline

This property is of type **String** and can raise a **DOMException** object on setting.

outlineColor

This property is of type **String** and can raise a **DOMException** object on setting.

outlineStyle

This property is of type **String** and can raise a **DOMException** object on setting.

outlineWidth

This property is of type **String** and can raise a **DOMException** object on setting.

overflow

This property is of type **String** and can raise a **DOMException** object on setting.

padding

This property is of type **String** and can raise a **DOMException** object on setting.

paddingTop

This property is of type **String** and can raise a **DOMException** object on setting.

paddingRight

This property is of type **String** and can raise a **DOMException** object on setting.

paddingBottom

This property is of type **String** and can raise a **DOMException** object on setting.

paddingLeft

This property is of type **String** and can raise a **DOMException** object on setting.

page

This property is of type **String** and can raise a **DOMException** object on setting.

pageBreakAfter

This property is of type **String** and can raise a **DOMException** object on setting.

pageBreakBefore

This property is of type **String** and can raise a **DOMException** object on setting.

pageBreakInside

This property is of type **String** and can raise a **DOMException** object on setting.

pause

This property is of type **String** and can raise a **DOMException** object on setting.

pauseAfter

This property is of type **String** and can raise a **DOMException** object on setting.

pauseBefore

This property is of type **String** and can raise a **DOMException** object on setting.

pitch

This property is of type **String** and can raise a **DOMException** object on setting.

pitchRange

This property is of type **String** and can raise a **DOMException** object on setting.

playDuring

This property is of type **String** and can raise a **DOMException** object on setting.

position

This property is of type **String** and can raise a **DOMException** object on setting.

quotes

This property is of type **String** and can raise a **DOMException** object on setting.

richness

This property is of type **String** and can raise a **DOMException** object on setting.

right

This property is of type **String** and can raise a **DOMException** object on setting.

size

This property is of type **String** and can raise a **DOMException** object on setting.

speak

This property is of type **String** and can raise a **DOMException** object on setting.

speakHeader

This property is of type **String** and can raise a **DOMException** object on setting.

speakNumeral

This property is of type **String** and can raise a **DOMException** object on setting.

speakPunctuation

This property is of type **String** and can raise a **DOMException** object on setting.

speechRate

This property is of type **String** and can raise a **DOMException** object on setting.

stress

This property is of type **String** and can raise a **DOMException** object on setting.

tableLayout

This property is of type **String** and can raise a **DOMException** object on setting.

textAlign

This property is of type **String** and can raise a **DOMException** object on setting.

textDecoration

This property is of type **String** and can raise a **DOMException** object on setting.

textIndent

This property is of type **String** and can raise a **DOMException** object on setting.

textShadow

This property is of type **String** and can raise a **DOMException** object on setting.

textTransform

This property is of type **String** and can raise a **DOMException** object on setting.

top

This property is of type **String** and can raise a **DOMException** object on setting.

unicodeBidi

This property is of type **String** and can raise a **DOMException** object on setting.

verticalAlign

This property is of type **String** and can raise a **DOMException** object on setting.

visibility

This property is of type **String** and can raise a **DOMException** object on setting.

voiceFamily

This property is of type **String** and can raise a **DOMException** object on setting.

volume

This property is of type **String** and can raise a **DOMException** object on setting.

whiteSpace

This property is of type **String** and can raise a **DOMException** object on setting.

widows

This property is of type **String** and can raise a **DOMException** object on setting.

width

This property is of type **String** and can raise a **DOMException** object on setting.

wordSpacing

This property is of type **String** and can raise a **DOMException** object on setting.

zIndex

This property is of type **String** and can raise a **DOMException** object on setting.

Appendix D: Acknowledgements

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

Lauren Wood (SoftQuad Software Inc., *chair*), Andrew Watson (Object Management Group), Andy Heninger (IBM), Arnaud Le Hors (W3C and IBM), Ben Chang (Oracle), Bill Smith (Sun), Bill Shea (Merrill Lynch), Bob Sutor (IBM), Chris Lovett (Microsoft), Chris Wilson (Microsoft), David Brownell (Sun), David Singer (IBM), Don Park (invited), Eric Vasilik (Microsoft), Gavin Nicol (INSO), Ian Jacobs (W3C), James Clark (invited), James Davidson (Sun), Jared Sorensen (Novell), Joe Kesselman (IBM), Joe Lapp (webMethods), Joe Marini (Macromedia), Johnny Stenback (Netscape), Jonathan Marsh (Microsoft), Jonathan Robie (Texcel Research and Software AG), Kim Adamson-Sharpe (SoftQuad Software Inc.), Laurence Cable (Sun), Mark Davis (IBM), Mark Scardina (Oracle), Martin Dürst (W3C), Mick Goulish (Software AG), Mike Champion (Arbortext and Software AG), Miles Sabin (Cromwell Media), Patti Lutsky (Arbortext), Paul Grosso (Arbortext), Peter Sharpe (SoftQuad Software Inc.), Phil Karlton (Netscape), Philippe Le Hégarret (W3C, *W3C team contact*), Ramesh Lekshmyanarayanan (Merrill Lynch), Ray Whitmer (iMall, Excite@Home and Netscape), Rich Rollman (Microsoft), Rick Gessner (Netscape), Scott Isaacs (Microsoft), Sharon Adler (INSO), Steve Byrne (JavaSoft), Tim Bray (invited), Tom Pixley (Netscape), Vidur Apparao (Netscape), Vinod Anupam (Lucent).

Thanks to all those who have helped to improve this specification by sending suggestions and corrections.

D.1: Production Systems

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

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

For DOM Level 2, we used Xerces as the basis DOM implementation and wish to thank the authors. Philippe Le Hégarret and Arnaud Le Hors wrote the Java programs which are the DOM application.

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

References

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

E.1: Normative references

DOM Level 2 Core

W3C (World Wide Web Consortium) Document Object Model Level 2 Core Specification, November 2000. Available at <http://www.w3.org/TR/2000/REC-DOM-Level-2-Core-20001113>

CSS2

W3C (World Wide Web Consortium) Cascading Style Sheets, level 2 Specification, May 1998. Available at <http://www.w3.org/TR/1998/REC-CSS2-19980512>

ECMAScript

ECMA (European Computer Manufacturers Association) ECMAScript Language Specification. Available at <http://www.ecma.ch/ecma1/STAND/ECMA-262.HTM>

HTML4.0

W3C (World Wide Web Consortium) HTML 4.0 Specification, April 1998. Available at <http://www.w3.org/TR/1998/REC-html40-19980424>

Java

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

OMGIDL

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

DOM Level 2 Views

W3C (World Wide Web Consortium) Document Object Model Level 2 Views Specification, November 2000. Available at <http://www.w3.org/TR/2000/REC-DOM-Level-2-Views-20001113>

XML-StyleSheet

W3C (World Wide Web Consortium) Associating Style Sheets with XML documents Version 1.0, June 1999. Available at <http://www.w3.org/1999/06/REC-xml-stylesheet-19990629>.

E.2: Informative references

DOM Level 2 HTML

W3C (World Wide Web Consortium) Document Object Model Level 2 HTML Specification. Available at <http://www.w3.org/TR/DOM-Level-2-HTML>

Index

appendMedium

azimuth

background

backgroundAttachment

backgroundColor

backgroundImage

backgroundPosition

backgroundRepeat

blue

border

borderBottom

borderBottomColor

borderBottomStyle

borderBottomWidth

borderCollapse

borderColor

borderLeft

borderLeftColor

borderLeftStyle

borderLeftWidth

borderRight

borderRightColor

borderRightStyle

borderRightWidth

borderSpacing

borderStyle

borderTop

borderTopColor

borderTopStyle

borderTopWidth

borderWidth

bottom 36, 54

captionSide

CHARSET_RULE

clear

clip

color

content

Counter

counterIncrement

counterReset

createCSSStyleSheet

CSS2 15, 37, 127

CSS2Properties

CSS_ATTR

CSS_CM

CSS_COUNTER

CSS_CUSTOM

CSS_DEG

CSS_DIMENSION

CSS_EMS

CSS_EXS

CSS_GRAD

CSS_HZ

CSS_IDENT

CSS_IN

CSS_INHERIT

CSS_KHZ

CSS_MM

CSS_MS

CSS_NUMBER

CSS_PC

CSS_PERCENTAGE

CSS_PRIMITIVE_VALUE

CSS_PT

CSS_PX

CSS_RAD

CSS_RECT

CSS_RGBCOLOR

CSS_S

CSS_STRING

CSS_UNKNOWN	CSS_URI	CSS_VALUE_LIST
CSSCharsetRule	cssFloat	CSSFontFaceRule
CSSImportRule	CSSMediaRule	CSSPageRule
CSSPrimitiveValue	CSSRule	CSSRuleList
cssRules 16, 20	CSSStyleDeclaration	CSSStyleRule
CSSStyleSheet	cssText 19, 25, 28	CSSUnknownRule
CSSValue	CSSValueList	cssValueType
cue	cueAfter	cueBefore
cursor		
deleteMedium	deleteRule 16, 21	direction
disabled	display	DocumentCSS
DocumentStyle	DOM Level 2 Core 9, 15, 40, 127	DOM Level 2 HTML 13, 127
DOM Level 2 Views 15, 127	DOMImplementationCSS	
ECMAScript	ElementCSSInlineStyle	elevation
emptyCells	encoding	
font	FONT_FACE_RULE	fontFamily
fontSize	fontSizeAdjust	fontStretch
fontStyle	fontVariant	fontWeight
getComputedStyle	getCounterValue	getFloatValue
getOverrideStyle	getPropertyCSSValue	getPropertyPriority
getPropertyValue	getRectValue	getRGBColorValue
getStringValue	green	

height	href 10, 23	HTML4.0 13, 127
identifier	IMPORT_RULE	insertRule 17, 21
item 11, 12, 18, 26, 35		
Java		
left 36, 61	length 11, 11, 18, 25, 35	letterSpacing
lineHeight	LinkStyle	listStyle 37, 62
listStyleImage	listStylePosition	listStyleType
margin	marginBottom	marginLeft
marginRight	marginTop	markerOffset
marks	maxHeight	maxWidth
media 10, 21, 23	MEDIA_RULE	MediaList
mediaText	minHeight	minWidth
OMGIDL	orphans	outline
outlineColor	outlineStyle	outlineWidth
overflow	ownerNode	ownerRule
padding	paddingBottom	paddingLeft
paddingRight	paddingTop	page
PAGE_RULE	pageBreakAfter	pageBreakBefore
pageBreakInside	parentRule 19, 25	parentStyleSheet 10, 19
pause	pauseAfter	pauseBefore
pitch	pitchRange	playDuring
position	primitiveType	

quotes

Rect

RGBColor

selectorText 20, 22

setProperty

size

speakNumeral

stress

StyleSheet 9, 23

tableLayout

textIndent

title

unicodeBidi

verticalAlign

voiceFamily

whiteSpace

wordSpacing

XML-StyleSheet 13, 13, 127

zIndex

red

richness

separator

setStringValue

speak

speakPunctuation

style 20, 22, 23, 39

StyleSheetList

textAlign

textShadow

top 36, 75

UNKNOWN_RULE

ViewCSS

volume

widows

removeProperty

right 36, 71

setFloatValue

sheet

speakHeader

speechRate

STYLE_RULE

styleSheets

textDecoration

textTransform

type 10, 19

UNKNOWN_RULE

visibility

width