# [MS-CSS21E]:
# Internet Explorer Extensions to Cascading Style Sheets (CSS) 2.1 and DOM Level 2 Style Specifications

### Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation for protocols, file formats, languages, standards as well as overviews of the interaction among each of these technologies.
- **Copyrights.** This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you may make copies of it in order to develop implementations of the technologies described in the Open Specifications and may distribute portions of it in your implementations using these technologies or your documentation as necessary to properly document the implementation. You may also distribute in your implementation, with or without modification, any schema, IDL's, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications.
- **No Trade Secrets.** Microsoft does not claim any trade secret rights in this documentation.
- **Patents.** Microsoft has patents that may cover your implementations of the technologies described in the Open Specifications. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. However, a given Open Specification may be covered by Microsoft Open Specification Promise or the Community Promise. If you would prefer a written license, or if the technologies described in the Open Specifications are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.
- **Trademarks.** The names of companies and products contained in this documentation may be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights.
- **Fictitious Names.** The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

**Reservation of Rights.** All other rights are reserved, and this notice does not grant any rights other than specifically described above, whether by implication, estoppel, or otherwise.

**Tools.** The Open Specifications do not require the use of Microsoft programming tools or programming environments in order for you to develop an implementation. If you have access to Microsoft programming tools and environments you are free to take advantage of them. Certain Open Specifications are intended for use in conjunction with publicly available standard specifications and network programming art, and assumes that the reader either is familiar with the aforementioned material or has immediate access to it.

## Revision Summary

| Date | Revision History | Revision Class | Comments |
|---|---|---|---|
| 03/26/2010 | 1.0 | New | Released new document. |
| 05/26/2010 | 1.2 | None | Introduced no new technical or language changes. |
| 09/08/2010 | 1.3 | Major | Significantly changed the technical content. |
| 10/13/2010 | 1.4 | Minor | Clarified the meaning of the technical content. |
| 02/10/2011 | 2.0 | No change | Introduced no new technical or language changes. |
| 02/22/2012 | 3.0 | Major | Significantly changed the technical content. |
| 07/25/2012 | 3.1 | Minor | Clarified the meaning of the technical content. |

# Table of Contents

# 1 Introduction

This document describes extensions provided by Windows® Internet Explorer® for the *Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification* [CSS-Level2-2009] W3C Candidate Recommendation 08 September 2009, and *Document Object Model (DOM) Level 2 Style Specification* [DOM Level 2 - Style] W3C Recommendation 13 November, 2000.

Sections 1.7 and 2 of this specification are normative and can contain the terms MAY, SHOULD, MUST, MUST NOT, and SHOULD NOT as defined in RFC 2119. All other sections and examples in this specification are informative.

## 1.1 Glossary

**MAY, SHOULD, MUST, SHOULD NOT, MUST NOT:** These terms (in all caps) are used as described in [RFC2119]. All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

## 1.2 References

References to Microsoft Open Specifications documentation do not include a publishing year because links are to the latest version of the technical documents, which are updated frequently. References to other documents include a publishing year when one is available.

### 1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information. Please check the archive site, http://msdn2.microsoft.com/en-us/library/E4BD6494-06AD-4aed-9823-445E921C9624, as an additional source.

[CSS-Level2-2009] Bos, B., Celik, T., Hickson, I., and Wium Lie, H., Eds., "Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification", W3C Candidate Recommendation 08 September 2009, http://www.w3.org/TR/2009/CR-CSS2-20090908/

[DOM Level 2 - Style] W3C, "Document Object Model (DOM) Level 2 Style Specification Version 1.0", W3C Recommendation, November 2000, http://www.w3.org/TR/2000/REC-DOM-Level-2-Style-20001113/

[MS-CSS21] Microsoft Corporation, "Internet Explorer Cascading Stylesheets (CSS) 2.1 Standards Support Document".

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, http://www.rfc-editor.org/rfc/rfc2119.txt

### 1.2.2 Informative References

[DOM Level 2 - Core] W3C, "Document Object Model (DOM) Level 2 Core Specification Version 1.0", W3C Recommendation 13 November, 2000, http://www.w3.org/TR/DOM-Level-2-Core/

[DOM Level 2 - HTML] W3C, "Document Object Model (DOM) Level 2 HTML Specification Version 1.0", W3C Recommendation, January 2003, http://www.w3.org/TR/2003/REC-DOM-Level-2-HTML-20030109/

[ECMA-262/5] ECMA International, "Standard ECMA-262 ECMAScript Language Specification", 5th Edition (December 2009), http://www.ecma-international.org/publications/files/ECMA-ST-ARCH/ECMA-262%205th%20edition%20December%202009.pdf

[ECMA-262] ECMA International, "ECMAScript Language Specification" ECMA-262, December 1999, http://www.ecma-international.org/publications/standards/Ecma-262.htm

[HTML] World Wide Web Consortium, "HTML 4.01 Specification", December 1999, http://www.w3.org/TR/html4/

[MSDN-DefaultBehaviors] Microsoft Corporation, "Default Behaviors Reference", DHTML Behaviors, http://msdn.microsoft.com/en-us/library/ms531081(VS.85).aspx

[MSDN-VisualFilters] Microsoft Corporation, "Visual Filters and Transitions Reference", http://msdn.microsoft.com/en-us/library/ms532853(VS.85).aspx

[MS-DOM2CE] Microsoft Corporation, "Internet Explorer Extensions to the Document Object Model (DOM) Level 2 Core Specification".

[MS-DOM2CEX] Microsoft Corporation, "Microsoft XML Extensions to the Document Object Model (DOM) Level 2 Core Specification".

[MS-DOM2E] Microsoft Corporation, "Internet Explorer Document Object Model (DOM) Level 2 Events Standards Support Document".

[MS-DOM2EE] Microsoft Corporation, "Internet Explorer Extensions to the DOM Level 2 Events Specification".

[MS-ES3EX] Microsoft Corporation, "Microsoft JScript Extensions to the ECMAScript Language Specification 3rd Edition".

[MS-ES5EX] Microsoft Corporation, "Microsoft Internet Explorer Extensions to the ECMAScript Language Specification Fifth Edition".

[MS-HTML401E] Microsoft Corporation, "Internet Explorer Extensions to the HTML 4.01 Specification".

## 1.3 Extension Overview (Synopsis)

The extensions described in this document were selected for their applicability to [CSS-Level2-2009] and [DOM Level 2 - Style].

The additional style attributes of **CSSStyleDeclaration** are organized based on sections of [CSS-Level2-2009] as follows:

**Section 11, Visual Effects**

- overflow-x (-ms-overflow-x)**
- overflow-y (-ms-overflow-y)

**Section 14, Colors and Backgrounds**

- background-position-x (-ms-background-position-x)
- background-position-y (-ms-background-position-y)

**Section 16, Text**

- layout-flow (-ms-layout-flow)
- layout-grid (-ms-layout-grid)
- layout-grid-char (-ms-layout-grid-char)
- layout-grid-line (-ms-layout-grid-line)
- layout-grid-mode (-ms-layout-grid-mode)
- layout-grid-type (-ms-layout-grid-type)
- text-underline-position (-ms-text-underline-position)

**Section 18, User Interface**

- accelerator (-ms-accelerator)
- -ms-interpolation-mode
- scrollbar-3dlight-color (-ms-scrollbar-3dlight-color)
- scrollbar-arrow-color (-ms-scrollbar-arrow-color)
- scrollbar-base-color (-ms-scrollbar-base-color)
- scrollbar-darkshadow-color (-ms-scrollbar-darkshadow-color)
- scrollbar-face-color (-ms-scrollbar-face-color)
- scrollbar-highlight-color (-ms-scrollbar-highlight-color)
- scrollbar-shadow-color (-ms-scrollbar-shadow-color)
- scrollbar-track-color (-ms-scrollbar-track-color)
- zoom (-ms-zoom)

** Names in parentheses are synonyms available in IE8 Mode and IE9 Mode.

The additional DOM attributes and methods can be organized similarly, based on function.

**Document Object Model**

- CSSStyleDeclaration.getAttribute()
- CSSStyleDeclaration.removeAttribute()
- CSSStyleDeclaration.setAttribute()

**Dynamic Styles**

- CSSStyleRule.readOnly
- CSSStyleSheet.addImport()
- CSSStyleSheet.addPageRule()
- CSSStyleSheet.addRule()

- CSSStyleSheet.id
- CSSStyleSheet.imports
- CSSStyleSheet.isAlternate
- CSSStyleSheet.isPrefAlternate
- CSSStyleSheet.owningElement
- CSSStyleSheet.pages
- CSSStyleSheet.readOnly
- CSSStyleSheet.removeImport()
- CSSStyleSheet.removeRule()
- CSSStyleSheet.rules
- StyleSheetPage.pseudoClass
- StyleSheetPage.selector
- StyleSheetPageList.item()
- StyleSheetPageList.length

**Visual Formatting**

- CSSStyleDeclaration.pixelBottom
- CSSStyleDeclaration.pixelHeight
- CSSStyleDeclaration.pixelLeft
- CSSStyleDeclaration.pixelRight
- CSSStyleDeclaration.pixelTop
- CSSStyleDeclaration.pixelWidth
- CSSStyleDeclaration.posBottom
- CSSStyleDeclaration.posHeight
- CSSStyleDeclaration.posLeft
- CSSStyleDeclaration.posRight
- CSSStyleDeclaration.posTop
- CSSStyleDeclaration.posWidth
- CSSStyleDeclaration.styleFloat

**Text**

- CSSStyleDeclaration.textDecorationBlink

- CSSStyleDeclaration.textDecorationLineThrough
- CSSStyleDeclaration.textDecorationNone
- CSSStyleDeclaration.textDecorationOverline
- CSSStyleDeclaration.textDecorationUnderline

**Special Functions**

- CSSStyleDeclaration.getExpression()
- CSSStyleDeclaration.removeExpression()
- CSSStyleDeclaration.setExpression()
- CSSStyleDeclaration.toString()

### 1.3.1 Organization of This Documentation

This document is organized as follows:

- **Interfaces**: The extensions are listed according to interface at the highest level.
- **Attributes, Methods, Collections**: The interface members are described at the next levels.

## 1.4 Relationship to Standards and Other Extensions

The following documents provide additional extensions.

- [MS-HTML401E]: Extensions to [HTML] and the [DOM Level 2 - HTML] specifications.
- [MS-DOM2CE] and [MS-DOM2CEX]: Extensions to the [DOM Level 2 - Core] specification for Windows® Internet Explorer® and Microsoft XML Core Services.
- [MS-DOM2EE]: Extensions to the [MS-DOM2E] specification.
- [MS-ES3EX]: Extensions to the ECMAScript [ECMA-262] specification.
- [MS-ES5EX]: Extensions to the ECMAScript [ECMA-262/5] specification.

## 1.5 Applicability Statement

This document specifies a set of extensions to the [CSS-Level2-2009] and [DOM Level 2 - Style] specifications. The extensions in this document provide access to some features that are unique to Windows® Internet Explorer® 7, Windows® Internet Explorer® 8, Windows® Internet Explorer® 9, and Windows® Internet Explorer® 10.

# 2 Extensions

This section specifies additional attributes and methods to elements from [CSS-Level2-2009] and [DOM Level 2 - Style] that are available in Windows® Internet Explorer® 7, Windows® Internet Explorer® 8, Windows® Internet Explorer® 9, and Windows® Internet Explorer® 10.

The extensions are as follows:

- Additional CSS2 Properties
- Extensions to the CSSStyleDeclaration Interface
- Extensions to the CSSStyleRule Interface
- Extensions to the CSSStyleSheet Interface
- StyleSheetPage Interface (a Microsoft extension)
- StyleSheetPageList Interface (a Microsoft extension)

## 2.1 Additional CSS2 Properties

This section lists CSS properties that are implemented by Windows® Internet Explorer® in addition to those described in [CSS-Level2-2009].

These properties also contribute attributes to the **CSS2Properties** interface defined in [DOM Level 2 - Style]. Although this interface was not specifically implemented in Internet Explorer, it is closely related to the **CSSStyleDeclaration** prototype object defined by Windows® Internet Explorer® 8.

### 2.1.1 Attributes

The **CSSStyleDeclaration** interface has been extended with the following attributes:

- accelerator
- background-position-x
- background-position-y
- behavior
- filter
- ime-mode
- layout-flow
- layout-grid
- layout-grid-char
- layout-grid-line
- layout-grid-mode
- layout-grid-type
- -ms-interpolation-mode

- overflow-x
- overflow-y
- scrollbar-3dlight-color
- scrollbar-arrow-color
- scrollbar-base-color
- scrollbar-darkshadow-color
- scrollbar-face-color
- scrollbar-highlight-color
- scrollbar-shadow-color
- scrollbar-track-color
- text-underline-position
- zoom

#### 2.1.1.1 accelerator

**accelerator** of type `DOMString`, **read/write**

Sets or retrieves a string that indicates whether the object represents a keyboard shortcut. The object returns `true` if it is a keyboard shortcut; `false` otherwise.

When the option to "Hide keyboard navigation indicators until I use the Alt key" is enabled in the user's Display Properties, accelerators are not underlined until the user presses the ALT key.

| **'accelerator'** | |
|---|---|
| Value: | true \| false |
| Initial: | false |
| Applies to: | block-level elements and inline blocks |
| Inherited: | no |
| Percentages: | N/A |
| Media: | visual |
| Computed value: | As specified |

#### 2.1.1.2 background-position-x

**backgroundPositionX** of type `DOMString`, **read/write**

Sets or retrieves the x-coordinate of the **backgroundPosition** property. The string value can be in the form of an absolute units designator (cm, mm, in, pt, or pc) or a relative units designator (em, ex, or px). It can also be in the form of a percentage or a horizontal alignment value.

| **background-position-x** | |
|---|---|
| Value: | <length> \| <percentage> \| [ left \|\| center \|\| right ] |
| Initial: | 0% |
| Applies to: | All elements |
| Inherited: | no |
| Percentages: | Width or height of the element |
| Media: | visual |
| Computed value: | As specified |

### 2.1.1.3 background-position-y

**backgroundPositionY** of type `DOMString`, **read/write**

Sets or retrieves the y-coordinate of the **backgroundPosition** property. The string value can be in the form of an absolute units designator (cm, mm, in, pt, or pc) or a relative units designator (em, ex, or px). It can also be in the form of a percentage or a vertical alignment value.

| **background-position-y** | |
|---|---|
| Value: | <length> \| <percentage> \| [ top \|\| center \|\| bottom ] |
| Initial: | 0% |
| Applies to: | All elements |
| Inherited: | no |
| Percentages: | Width or height of the element |
| Media: | visual |
| Computed value: | As specified |

### 2.1.1.4 behavior

**behavior** of type `DOMString`, **read/write**

Sets or retrieves the location of the Dynamic HTML (DHTML) behavior. In a script implementation, the location can be an absolute or relative URL. In a binary implementation, the location is the **id** attribute specified for an **object** element. Default behaviors are identified by the string `#default#` plus behavior name.

| **behavior** | |
|---|---|
| Value: | <uri> |
| Initial: | none |
| Applies to: | block-level elements, table elements, and inline blocks |

| behavior | |
|---|---|
| Inherited: | no |
| Percentages: | N/A |
| Media: | visual |
| Computed value: | As specified |

Values have the following meanings:

**url(location)**

Script implementation of DHTML behavior, where `location` is absolute or relative URL.

**url(#id)**

Binary implementation of DHMTL behavior, where `id` is the specified **id** attribute of the **OBJECT** element.

**url(#default#behaviorName)**

Windows® Internet Explorer® built-in default behavior, identified by `behaviorName`. For more information, see [MSDN-DefaultBehaviors].

### 2.1.1.5 filter

*Quirks Mode, IE7 Mode, IE8 Mode, and IE9 Mode (All Versions)*

**filter** of type `DOMString`**, read/write**

Sets or retrieves the filter or collection of filters applied to the object. Delimit multiple values with commas (,). In Windows® Internet Explorer® 8, enclose the values in single quotation marks (') or double quotation marks (") when using **-ms-filter**.

| filter | |
|---|---|
| Value: | progid:<filterName>([param1, param2, ...]) |
| Initial: | none |
| Applies to: | block-level elements, table elements, and inline blocks |
| Inherited: | no |
| Percentages: | N/A |
| Media: | visual |
| Computed value: | As specified |

Values have the following meanings:

**filterName**

Any filter listed in [MSDN-VisualFilters].

**param**

Parameter values as defined by the named filter.

An object must have layout for the filter to render. The **hasLayout** feature of quirks mode and IE7 mode is described in [MS-CSS21], Appendix C: hasLayout.

### 2.1.1.6 ime-mode

**imeMode** of type `DOMString`, **read/write**

Sets or retrieves the state of an Input Method Editor (IME). The default value for this attribute is `auto` where IME is not affected. The other three states are `active`, `inactive`, and `disabled`. In `disabled` state, the user cannot activate the IME.

| **ime-mode** | |
|---|---|
| Value: | auto \| [ active \|\| inactive \|\| disabled ] |
| Initial: | auto |
| Applies to: | textarea, input type="text" |
| Inherited: | no |
| Percentages: | N/A |
| Media: | visual |
| Computed value: | As specified |

Values have the following meanings:

**auto**

Default. IME is not affected.

**active**

All characters are entered through the IME. Users can still deactivate the IME.

**inactive**

All characters are entered without the IME.

**disabled**

IME is completely disabled. Users cannot activate the IME if the control has focus.

### 2.1.1.7 layout-flow

**layoutFlow** of type `DOMString`, **read/write**

Sets or retrieves the direction and flow of the content in the object.

The default value for this attribute is `horizontal`. In this mode, content in the object flows from left to right, and the next horizontal line is positioned underneath the previous line. This layout is used in most Roman-based documents. The other possible value for this attribute is `vertical-`

ideographic. In this mode, content in the object flows from top to bottom, and the next vertical line appears to the left of the previous one. This layout is used in East Asian typography.

| **layout-flow** | |
|---|---|
| Value: | horizontal \| vertical-ideographic |
| Initial: | horizontal |
| Applies to: | All elements |
| Inherited: | yes |
| Percentages: | N/A |
| Media: | visual |
| Computed value: | As specified |

Values have the following meanings:

**horizontal**

Content in the object flows from left to right, top to bottom.

**vertical-ideographic**

Content flows from top to bottom, and next vertical line appears to the left of the previous one.

### 2.1.1.8 layout-grid

**layoutGrid** of type DOMString, **read/write**

Sets or retrieves the composite document grid properties that specify the layout of text characters.

The property has a default value of both loose none none.

| **layout-grid** | |
|---|---|
| Value: | [<'layout-mode'> [<'layout-type'> [<'layout-line'> [<'layout-char'>]]]] |
| Initial: | both loose none none |
| Applies to: | block-level elements, and table elements |
| Inherited: | yes |
| Percentages: | N/A |
| Media: | visual |
| Computed value: | As specified |

### 2.1.1.9 layout-grid-char

**layoutGridChar** of type DOMString, **read/write**

Sets or retrieves the size of the character grid used for rendering the text content of an element. This property has a default value of none. The other possible values are length, percentage, and auto.

| layout-grid-char | |
|---|---|
| Value: | none \| auto \| <length> \| <percentage> |
| Initial: | none |
| Applies to: | block-level elements, and table elements |
| Inherited: | yes |
| Percentages: | Refer to parent object |
| Media: | visual |
| Computed value: | As specified |

Values have the following meaning:

**none**

Default. No character grid is set.

**auto**

Largest character in the font is used to set the character grid.

**<length>**

Floating-point number, followed by units designator.

**<percentage>**

Integer, followed by percent sign (%).

#### 2.1.1.10 layout-grid-line

**layoutGridLine** of type DOMString, **read/write**

Sets or retrieves the line height value used for rendering the text content of an element. This property has a default value of none. The other possible values are length, percentage, and auto.

| layout-grid-line | |
|---|---|
| Value: | none \| auto \| <length> \| <percentage> |
| Initial: | none |
| Applies to: | block-level elements, and table elements |
| Inherited: | yes |
| Percentages: | Refer to parent object |
| Media: | visual |

| **layout-grid-line** | |
|---|---|
| Computed value: | As specified |

Values have the following meanings:

**none**

Default. No line grid is set.

**auto**

Largest character in the font is used to set the line height.

**<length>**

Floating point number, followed by units designator.

**<percentage>**

Integer, followed by percent sign (%).

#### 2.1.1.11 layout-grid-mode

**layoutGridMode** of type `DOMString`, **read/write**

Sets or retrieves whether the text layout grid uses two dimensions. This property has a default value of `both` where both the `line` and `char` grids are enabled. The other possible values are `none`, `line`, and `char`.

| **layout-grid-mode** | |
|---|---|
| Value: | both \| none \| line \| char |
| Initial: | both |
| Applies to: | All elements |
| Inherited: | yes |
| Percentages: | N/A |
| Media: | visual |
| Computed value: | As specified |

Values have the following meaning:

**both**

Default. Both char and line grid modes are enabled.

**none**

No grid is used.

**line**

Only a line grid is used. This is recommended for inline elements, such as **span**.

**char**

Only a character grid is used. This is recommended for block-level elements, such as **blockquote**.

### 2.1.1.12 layout-grid-type

**layoutGridType** of type DOMString, **read/write**

Sets or retrieves the gridline value used for rendering the text content of an element. This property has a default value of `loose`. The other possible values are `strict` and `fixed`.

| **layout-grid-type** | |
|---|---|
| Value: | loose \| strict \| fixed |
| Initial: | loose |
| Applies to: | block-level elements, and table elements |
| Inherited: | yes |
| Percentages: | N/A |
| Media: | visual |
| Computed value: | As specified |

Values have the following meanings:

**loose**

Grid used for Japanese and Korean characters. In this mode, a constant width increment is applied to characters as follows:

- Wide characters and narrow kana characters are incremented to obtain an exact grid fit, as specified by the **layout-grid-char** property.
- Narrow characters, except connected and cursive characters, are incremented by half of the increment amount applied to wide characters.
- Other characters, including connected and cursive characters, are not incremented, and behave as if no character grid is set.

**strict**

Grid used for Chinese, as well as Japanese (Genko) and Korean characters. Only the ideographs, kanas, and wide characters are snapped to the grid. Other characters are rendered as usual, as though the **layout-grid-mode** attribute is set to `none` or `line` for text spans containing these characters. This mode also disables special text justification and character width adjustments normally applied to the element. Finally, if there is no line-break opportunity in a text span that exceeds the line boundary, the text is pushed to the next line and the last part of the previous line is left blank.

**fixed**

Grid used for monospaced layout. The layout rules are as follows:

- All non-cursive characters are treated as equal; every character is centered within a single grid space by default.
- Runs of cursive characters are treated as strips; the same as in a strict grid.
- Justification or any other character-width changing behaviors are disabled.

### 2.1.1.13 -ms-interpolation-mode

**msInterpolationMode** of type `DOMString`, **read/write**

Sets or retrieves the interpolation (resampling) method used to stretch images. This property has no default value.

| **-ms-interpolation-mode** | |
|---|---|
| Value: | [ nearest-neighbor \|\| bicubic ] |
| Initial: | depends on user-agent |
| Applies to: | stretched and zoomed images |
| Inherited: | no |
| Percentages: | N/A |
| Media: | visual |
| Computed value: | As specified |

Values have the following meaning:

**nearest-neighbor**

Use nearest-neighbor (low-quality) interpolation.

**bicubic**

Use bicubic (high-quality) interpolation.

**Note** In Windows® Internet Explorer® 7 at 100% zoom level, the default interpolation is `nearest-neighbor`; otherwise, `bicubic` mode is used. In Windows® Internet Explorer® 8, `bicubic` is always used.

### 2.1.1.14 overflow-x

**overflowX** of type `DOMString`, **read/write**

Sets or retrieves how to manage the content of the object when the content exceeds the width of the object. This property has a default value of `visible`. The other possible values are `scroll`, `hidden`, and `auto`.

| **overflow-x** | |
|---|---|
| Value: | [ visible \|\| scroll \|\| hidden \|\| auto ] |
| Initial: | visible (except textarea, which is `hidden`) |

| **overflow-x** | |
|---|---|
| Applies to: | All elements |
| Inherited: | no |
| Percentages: | N/A |
| Media: | visual |
| Computed value: | As specified |

See description of **overflow** in [DOM Level 2 - Style].

### 2.1.1.15 overflow-y

**overflowY** of type `DOMString`, **read/write**

Sets or retrieves how to manage the content of the object when the content exceeds the height of the object. This property has a default value of `visible`. The other possible values are `scroll`, `hidden`, and `auto`.

| **overflow-y** | |
|---|---|
| Value: | [ visible \|\| scroll \|\| hidden \|\| auto ] |
| Initial: | visible (except textarea, which is `auto`) |
| Applies to: | All elements |
| Inherited: | no |
| Percentages: | N/A |
| Media: | visual |
| Computed value: | As specified |

See description of **overflow** in [DOM Level 2 - Style].

### 2.1.1.16 scrollbar-3dlight-color

**scrollbar3dLightColor** of type `DOMString`, **read/write**

Sets or retrieves the color of the top and left edges of the scroll box and scroll arrows of a scroll bar. This property has no default value. Possible values for this attribute are any color name or RGB values.

| **scrollbar-3dlight-color** | |
|---|---|
| Value: | <color> |
| Initial: | depends on user-agent |
| Applies to: | scroll bars of element or window |
| Inherited: | yes |

| scrollbar-3dlight-color | |
|---|---|
| Percentages: | N/A |
| Media: | visual |
| Computed value: | As specified |

### 2.1.1.17 scrollbar-arrow-color

**scrollbarArrowColor** of type `DOMString`, **read/write**

Sets or retrieves the color of the arrow elements of a scroll arrow. This property has no default value. Possible values for this attribute are any color name or RGB values.

| scrollbar-arrow-color | |
|---|---|
| Value: | <color> |
| Initial: | depends on user-agent |
| Applies to: | scroll bars of element or window |
| Inherited: | yes |
| Percentages: | N/A |
| Media: | visual |
| Computed value: | As specified |

### 2.1.1.18 scrollbar-base-color

**scrollbarBaseColor** of type `DOMString`, **read/write**

Sets or retrieves the color of the main elements of a scroll bar, which include the scroll box, track, and scroll arrows. This property has no default value. Possible values for this attribute are any color name or RGB values.

| scrollbar-base-color | |
|---|---|
| Value: | <color> |
| Initial: | depends on user-agent |
| Applies to: | scroll bars of element or window |
| Inherited: | yes |
| Percentages: | N/A |
| Media: | visual |
| Computed value: | As specified |

### 2.1.1.19 scrollbar-darkshadow-color

**scrollbarDarkShadowColor** of type `DOMString`, **read/write**

Sets or retrieves the color of the gutter of a scroll bar. This property has no default value. Possible values for this attribute are any color name or RGB values.

| **scrollbar-darkshadow-color** | |
|---|---|
| Value: | <color> |
| Initial: | depends on user-agent |
| Applies to: | scroll bars of element or window |
| Inherited: | yes |
| Percentages: | N/A |
| Media: | visual |
| Computed value: | As specified |

### 2.1.1.20 scrollbar-face-color

**scrollbarFaceColor** of type `DOMString`, **read/write**

Sets or retrieves the color of the scroll box and scroll arrows of a scroll bar. This property has no default value. Possible values for this attribute are any color name or RGB values.

| **scrollbar-face-color** | |
|---|---|
| Value: | <color> |
| Initial: | depends on user-agent |
| Applies to: | scroll bars of element or window |
| Inherited: | yes |
| Percentages: | N/A |
| Media: | visual |
| Computed value: | As specified |

### 2.1.1.21 scrollbar-highlight-color

**scrollbarHighlightColor** of type `DOMString`, **read/write**

Sets or retrieves the color of the top and left edges of the scroll box and scroll arrows of a scroll bar. This property has no default value. Possible values for this attribute are any color name or RGB values.

| **scrollbar-highlight-color** | |
|---|---|
| Value: | <color> |

| scrollbar-highlight-color | |
|---|---|
| Initial: | depends on user-agent |
| Applies to: | scroll bars of element or window |
| Inherited: | yes |
| Percentages: | N/A |
| Media: | visual |
| Computed value: | As specified |

#### 2.1.1.22 scrollbar-shadow-color

**scrollbarShadowColor** of type `DOMString`, **read/write**

Sets or retrieves the color of the bottom and right edges of the scroll box and scroll arrows of a scroll bar. This property has no default value. Possible values for this attribute are any color name or RGB values.

| scrollbar-shadow-color | |
|---|---|
| Value: | <color> |
| Initial: | depends on user-agent |
| Applies to: | scroll bars of element or window |
| Inherited: | yes |
| Percentages: | N/A |
| Media: | visual |
| Computed value: | As specified |

#### 2.1.1.23 scrollbar-track-color

**scrollbarTrackColor** of type `DOMString`, **read/write**

Sets or retrieves the color of the track element of a scroll bar. This property has no default value. Possible values for this attribute are any color name or RGB values.

| scrollbar-track-color | |
|---|---|
| Value: | <color> |
| Initial: | depends on user-agent |
| Applies to: | scroll bars of element or window |
| Inherited: | yes |
| Percentages: | N/A |
| Media: | visual |

| scrollbar-track-color | |
|---|---|
| Computed value: | As specified |

#### 2.1.1.24 text-underline-position

**textunderlineposition** of type DOMString, **read/write**

Sets or retrieves the position of the underline decoration that is set through the **textDecoration** property of the object. This property has a default value of auto. The other possible values are above, below, and auto-pos.

| text-underline-position | |
|---|---|
| Value: | [ auto \|\| above \|\| below \|\| auto-pos ] |
| Initial: | auto |
| Applies to: | All elements |
| Inherited: | yes |
| Percentages: | N/A |
| Media: | visual |
| Computed value: | As specified |

Values have the following meaning:

**auto**

Decoration appears above the text if the **lang** attribute is set to ja, which is the language code abbreviation for Japanese, and the **-ms-writing-mode** attribute is set to tb-rl, which causes vertical inline text progression. If not, the decoration appears below the text.

**above**

Decoration appears above the text.

**below**

Decoration appears below the text.

**auto-pos**

Same as auto.

#### 2.1.1.25 zoom

**zoom** of type Integer or DOMString, **read/write**

Sets or retrieves the magnification scale of the object. This property has a default value of normal. The other possible values are number and percentage.

| **zoom** | |
|---|---|
| Value: | normal \| <number> \| <percentage> |
| Initial: | normal |
| Applies to: | All elements |
| Inherited: | no |
| Percentages: | percent of element's normal size |
| Media: | visual |
| Computed value: | As specified |

Values have the following meaning:

**normal**

No zoom. The object renders at normal magnification.

**<number>**

Floating-point number that specifies the scale, where `1.0` is normal.

**<percentage>**

The value is a percentage of the scale, where `100%` is normal.

## 2.2 Extensions to the CSSStyleDeclaration Interface

This section lists extensions to the **CSSStyleDeclaration** interface defined in [DOM Level 2 - Style].

The **CSSStyleDeclaration** interface as implemented in Windows® Internet Explorer® defines the properties and methods inherited by objects in the **CSSStyleDeclaration** prototype chain.

### 2.2.1 Attributes

The **CSSStyleDeclaration** interface has been extended with the following attributes:

- pixelBottom
- pixelHeight
- pixelLeft
- pixelRight
- pixelTop
- pixelWidth
- posBottom
- posHeight
- posLeft

- posRight
- posTop
- posWidth
- styleFloat
- textDecorationBlink
- textDecorationLineThrough
- textDecorationNone
- textDecorationOverline
- textDecorationUnderline

### 2.2.1.1 pixelBottom

**pixelBottom** of type `integer`, **read/write**

Sets or retrieves the bottom position of the object in relation to the bottom of the next positioned object in the document hierarchy.

The value is interpreted to be in pixels and reflects the value of the `bottom` attribute.

A DHTML expression is a valid value for quirks mode and IE7 mode.

This property has no default value.

### 2.2.1.2 pixelHeight

**pixelHeight** of type `integer`, **read/write**

Sets or retrieves the height of the object.

The value is interpreted to be in pixels and reflects the value of the `height` attribute.

A DHTML expression is a valid value for quirks mode and IE7 mode.

This property has no default value.

### 2.2.1.3 pixelLeft

**pixelLeft** of type `integer`, **read/write**

Sets or retrieves the left position of the object.

The value is interpreted to be in pixels and reflects the value of the `left` attribute.

A DHTML expression is a valid value for quirks mode and IE7 mode.

This property has no default value.

### 2.2.1.4 pixelRight

**pixelRight** of type `integer`, **read/write**

Sets or retrieves the right position of the object.

The value is interpreted to be in pixels and reflects the value of the `right` attribute.

A DHTML expression is a valid value for quirks mode and IE7 mode.

This property has no default value.

#### 2.2.1.5 pixelTop

**pixelTop** of type `integer`**, read/write**

Sets or retrieves the top position of the object.

The value is interpreted to be in pixels and reflects the value of the `top` attribute.

A DHTML expression is a valid value for quirks mode and IE7 mode.

This property has no default value.

#### 2.2.1.6 pixelWidth

**pixelWidth** of type `integer`**, read/write**

Sets or retrieves the width of the object.

The value is interpreted to be in pixels and reflects the value of the `width` attribute.

A DHTML expression is a valid value for quirks mode and IE7 mode.

This property has no default value.

#### 2.2.1.7 posBottom

**posBottom** of type `float`**, read/write**

Sets or retrieves the bottom position of the object.

The value reflects the value and length units of the `bottom` attribute.

This property always returns zero for nonpositioned items because `bottom` has meaning only when the object is positioned. If the `bottom` attribute is not set, the `posBottom` property returns `0` (zero).

A DHTML expression is a valid value for quirks mode and IE7 mode.

This property has no default value.

#### 2.2.1.8 posHeight

**posHeight** of type `float`**, read/write**

Sets or retrieves the height of the object.

The value reflects the value and length units of the `height` attribute.

This property always returns zero for nonpositioned items because `height` has meaning only when the object is positioned. If the `height` attribute is not set, the `posHeight` property returns `0` (zero).

A DHTML expression is a valid value for quirks mode and IE7 mode.

This property has no default value.

#### 2.2.1.9 posLeft

**posLeft** of type `float`**, read/write**

Sets or retrieves the left position of the object.

The value reflects the value and length units of the `left` attribute.

This property always returns zero for nonpositioned items because `left` has meaning only when the object is positioned. If the `left` attribute is not set, the `posLeft` property returns `0` (zero).

A DHTML expression is a valid value for quirks mode and IE7 mode.

This property has no default value.

#### 2.2.1.10 posRight

**posRight** of type `float`**, read/write**

Sets or retrieves the right position of the object.

The value reflects the value and length units of the `right` attribute.

This property always returns zero for nonpositioned items because `right` has meaning only when the object is positioned. If the `right` attribute is not set, the `posRight` property returns `0` (zero).

A DHTML expression is a valid value for quirks mode and IE7 mode.

This property has no default value.

#### 2.2.1.11 posTop

**posTop** of type `float`**, read/write**

Sets or retrieves the top position of the object.

The value reflects the value and length units of the `top` attribute.

This property always returns zero for nonpositioned items because `top` has meaning only when the object is positioned. If the `top` attribute is not set, the `posTop` property returns `0` (zero).

A DHTML expression is a valid value for quirks mode and IE7 mode.

This property has no default value.

#### 2.2.1.12 posWidth

**posWidth** of type `float`**, read/write**

Sets or retrieves the width of the object.

The value reflects the value and length units of the `width` attribute.

This property always returns zero for nonpositioned items because width has meaning only when the object is positioned. If the width attribute is not set, the posWidth property returns 0 (zero).

A DHTML expression is a valid value for quirks mode and IE7 mode.

This property has no default value.

### 2.2.1.13 styleFloat

**styleFloat** of type DOMString, **read/write**

Sets or retrieves on which side of the object the text will flow. Compare to **cssFloat** in [DOM Level 2 - Style].

**Possible Values**

none

Default. Object displays in normal flow.

left

Object floats left, and text flows around the right of the object.

right

Object floats right, and text flows around the left of the object.

### 2.2.1.14 textDecorationBlink

**textDecorationBlink** of type Boolean, **read/write**

Sets or retrieves the Boolean value that indicates whether the **text-decoration** rule is set to blink. If true, the **textDecoration** rule is set to blink. This property has no default value.

### 2.2.1.15 textDecorationLineThrough

**textDecorationLineThrough** of type Boolean, **read/write**

Sets or retrieves a Boolean value indicating whether the text in the object has a line drawn through it. If true, a line is drawn through the affected text. This property has no default value.

### 2.2.1.16 textDecorationNone

**textDecorationNone** of type Boolean, **read/write**

Sets or retrieves the Boolean value indicating whether the **textDecoration** property for the object has been set to none. If true, the **textDecoration** property to is set to none; otherwise, the **textDecoration** property is set to an empty string. This property has no default value.

### 2.2.1.17 textDecorationOverline

**textDecorationOverline** of type Boolean, **read/write**

Sets or retrieves a Boolean value indicating whether the text in the object has a line drawn over it. If true, a line is drawn over the affected text. This property has no default value.

#### 2.2.1.18 textDecorationUnderline

**textDecorationUnderline** of type `Boolean`, **read/write**

Sets or retrieves a Boolean value indicating whether the text in the object is underlined. If `true`, a line is drawn under the affected text. This property has no default value.

### 2.2.2 Methods

The **CSSStyleDeclaration** interface has been extended with the following methods:

- getAttribute
- getExpression
- removeAttribute
- removeExpression
- setAttribute
- setExpression
- toString

#### 2.2.2.1 getAttribute

Retrieves the value of the specified attribute.

**Syntax**

```
  vAttrValue = object.getAttribute(sAttrName [, iFlags])
```

**Parameters**

**sAttrName** of type `DOMString`

A required string that specifies the name of the attribute.

**iFlags** of type `integer`

Optional. Integer that specifies one or more of the following flags:

`0`

Default. Performs a property search that is not case-sensitive, and returns an interpolated value if the property is found.

`1`

Performs a case-sensitive property search. To find a match, the uppercase and lowercase letters in **sAttrName** must exactly match those in the attribute name.

`2`

Returns attribute value as a string. This flag does not work for event properties.

4

Returns attribute value as a fully expanded URL. Only works for URL attributes.

**Return Value**

If the attribute is not present, this method returns null.

**No Errors**

### 2.2.2.2 getExpression

*Quirks Mode and IE7 Mode (All Versions)*

Retrieves the expression for the specified property.

**Syntax**

```
vExpression = object.getExpression(sPropertyName)
```

**Parameters**

**sPropertyName** of type `DOMString`

A required string that specifies the name of the property from which to retrieve the expression.

**Return Value**

Returns the expression as a string.

### 2.2.2.3 removeAttribute

Removes an attribute from an object.

**Syntax**

```
bSuccess = object.removeAttribute(sAttrName [, iCaseSensitive])
```

**Parameters**

**sAttrName** of type `DOMString`

A required string that specifies an attribute name.

**iCaseSensitive** of type integer

Optional. Integer that specifies whether to use a case-sensitive search to locate the attribute. Can be one of the following values:

1

The case of `sAttrName` is respected.

0

Match `sAttrName` regardless of case.

### Return Value

Returns a Boolean with one of the following possible values:

`true`

The attribute was successfully removed.

`false`

The attribute was not removed.

### Remarks

If your pages are displayed in IE7 mode, be careful when spelling attribute names. If two or more attributes have the same name—differing only in capitalization—and **iCaseSensitive** is set to 0, this method removes only the last attribute created with this name. All other attributes of the same name are ignored.

### Example

The following examples demonstrate how to use the **getExpression** method to retrieve CSS properties.

This example uses the **getExpression** method to retrieve the width property of a span object.

```
<body>
<span id="trueBlueSpan"
    style="background-color:lightblue; width:100px">
    The width of this blue span is set inline at 100 pixels.
</span>
<span id="oldYellowSpan" style="background-color:lightyellow;
    width:200px">
    The width of this yelllow span is set inline at 200 pixels.
</span>
<br>
<span id="AlGreenSpan" style="background-color:lightgreen;
    width:expression(trueBlueSpan.style.pixelWidth +
    oldYellowSpan.style.pixelWidth)">
    Click the button below to see the expression used to set
    the width of this span.
</span>
<br>
<button onclick=alert(AlGreenSpan.style.getExpression("width"));>
    See Expression</button>
</body>
```

In the following example, the **setExpression** method is used to set the width property of a blue input type=text object equal to the sum of the values in two other input type=text objects. When the user clicks the input type=button element, the **getExpression** method is used to display the expression.

```
<html>
<head>
<script language="JScript">
var s;
function fnInit() {
```

```
Box3.style.setExpression("width","eval(Box1.value) + eval(Box2.value)",
"jscript");
}
function getexp() {
s=Box3.style.getExpression("width");
alert("Expression for the width of the blue box is \n\n" + s +
"\n\nThe width property has a value of " + Box3.style.width);
}
</script>
</head>
<body onload=fnInit();>
<input type=text id="Box1" value=40>
<br><input type=text id="Box2" value=40>
<br><input type=text id="Box3" style="background-color:blue">
<br><input type=button id="Button2" value="Get expression" onclick="getexp()">
</body>
</html>
```

### 2.2.2.4 removeExpression

*Quirks Mode and IE7 Mode (All Versions)*

Removes the expression from the specified property.

**Syntax**

```
bSuccess = object.removeExpression(sPropertyName)
```

**Parameters**

**sPropertyName** of type `DOMString`

A required string that specifies the name of the property from which to remove an expression.

**Return Value**

Returns `true` to indicate that the expression was successfully removed; `false` if not.

**Remarks**

After the expression is removed from the specified property, the value of the property equals the last computed value of the expression. To remove expressions set by the **setExpression** method, use **removeExpression**.

### 2.2.2.5 setAttribute

Sets the value of the specified attribute.

**Syntax**

```
object.setAttribute(sAttrName, vValue [, iCaseSensitive])
```

**Parameters**

**sAttrName** of type `DOMString`

A required string that specifies the name of the attribute.

**vValue** of type `DOMString`

The value to assign to the attribute.

**iCaseSensitive** of type `Integer`

An optional integer that specifies whether to use a case-sensitive search to locate the attribute. Can be one of the following values:

`1`

The case of `sAttrName` is respected.

`0`

Match `sAttrName` regardless of case.

**Return Value**

No return value.

**Remarks**

- If the specified attribute is not already present, the **setAttribute** method adds the attribute to the object and sets the value.
- If your pages are displayed in IE7 mode, be careful when spelling attribute names. If you set iCaseSensitive to 1 and the sAttrName parameter does not have the same uppercase and lowercase letters as the attribute, a new attribute is created for the object. If two or more attributes have the same name, differing only in case, and iCaseSensitive is set to 0, this method assigns values only to the first attribute created with this name. All other attributes of the same name are ignored.

### 2.2.2.6 setExpression

*Quirks Mode and IE7 Mode (All Versions)*

Sets an expression for the specified object.

**Syntax**

```
object.setExpression(sPropertyName, sExpression [, sLanguage])
```

**Parameters**

**sPropertyName** of type `DOMString`

A required string that specifies the name of the property to which `sExpression` is added.

**sExpression** of type `DOMString`

A required string that specifies any valid script (JScript, JavaScript, or VBScript) statement without quotations or semicolons. This string can include references to other properties on the current page. Array references are not allowed on object properties included in this script.

**sLanguage** of type `DOMString`

An optional string that specifies one of the following values:

`JScript`

Default. Language is JScript.

`VBScript`

Language is VBScript.

`JavaScript`

Language is JavaScript.

**Return Value**

No return value.

**Remarks**

The following **expression()** syntax can be used to set an expression on a CSS attribute in HTML.

```
<ELEMENT STYLE="sAttributeName:expression(sExpression)">
```

The data type of the evaluated expression in the **sLanguage** parameter must match one of the possible values allowed for the **sExpression** parameter. If the property or attribute specified by the first parameter requires a string, the data type of the second parameter must be a string. Otherwise, the second parameter is evaluated prior to invoking **setExpression**, causing the expression to be set to the result of the evaluation.

Authors can use the **uniqueID** property of an object in an expression to refer back to the object. Using **uniqueID** is an alternative to specifying an id for expressions that use an object reference.

### 2.2.2.7 toString

*Quirks Mode, IE7 Mode, and IE8 Mode (All Versions) only*

Returns the type of an object as a string.

**Syntax**

```
sObject = object.style.toString()
```

**Parameters**

None.

**Return Value**

Returns the type of the object type as the string `[object]`.

## 2.3 Extensions to the CSSStyleRule Interface

This section lists extensions to the **CSSStyleRule** interface defined in [DOM Level 2 - Style].

### 2.3.1 Attributes

The **CSSStyleRule** interface has been extended with the readOnly attribute.

#### 2.3.1.1 readOnly

**readOnly** of type `Boolean`, **read-only**

Retrieves whether the rule or style sheet is defined on the page or is imported. If `true`, the style sheet is linked to the page or is imported through the **@import** rule. If `false`, the style sheet is defined in the page. This property has no default value.

Style sheets obtained through a link object or the **@import** rule cannot be modified if the **designMode** property is enabled.

## 2.4 Extensions to the CSSStyleSheet Interface

This section lists extensions to the **CSSStyleSheet** interface defined in [DOM Level 2 - Style].

The **CSSStyleSheet** interface is extended by Attributes, Methods, and Collections.

### 2.4.1 Attributes

The **CSSStyleSheet** interface has been extended with the following attributes:

- isAlternate
- isPrefAlternate
- owningElement
- id
- readOnly

#### 2.4.1.1 isAlternate

**isAlternate** of type `Boolean`, **read-only**

Retrieves a value that indicates whether the **IHTMLStyleSheet3** object is an alternative style sheet. If `true`, the style sheet is an alternate style sheet.

A style sheet is alternate if one or both of the following is true:

- **link** element's **rel** attribute contains both "alternate" and "stylesheet"
- **link** element's **rel** attribute contains "stylesheet" and **title** is specified and not empty

Windows® Internet Explorer® 8 allows users to select alternate style sheets, or disable styles entirely. (Use the **View** menu and point to **Style** submenu.) However, to appear as a named selection, an alternate style sheet must declare a **title** attribute.

This attribute is not available in Windows® Internet Explorer® 7.

#### 2.4.1.2 isPrefAlternate

**isPrefAlternate** of type `Boolean`, **read-only**

Retrieves a value that indicates whether the **IHTMLStyleSheet3** object is the preferred style sheet. If `true`, the object is a preferred alternative style sheet. If more than one style sheet exist then this object is the preferred one.

An author may specify the default style sheet by setting the following:

- **link** element's **rel** attribute contains "stylesheet", but not "alternate"
- **link** element's **title** attribute is specified and not empty

Windows® Internet Explorer® 8 uses the preferred style sheet when the page is loaded, and marks it as default on the **Style** submenu.

This attribute is not available in Windows® Internet Explorer® 7.

This property has no default value.

#### 2.4.1.3 owningElement

**owningElement** of type `Element`, **read-only**

Retrieves the **HTMLStyleElement** or **HTMLLinkElement** associated with the **CSSStyleSheet** object.

#### 2.4.1.4 id

**id** of type `DOMString`, **read/write**

Sets or retrieves the string identifying the object.

The **id** should be unique throughout the scope of the current document. If a document contains more than one object with the same identifier, the objects are exposed as a collection that can be referenced only in ordinal position.

#### 2.4.1.5 readOnly

**readOnly** of type `Boolean`, **read-only**

Retrieves whether the rule or style sheet is defined on the page or is imported. If `true`, the style sheet is linked to the page or is imported through the **@import** rule. If `false`, the style sheet is defined in the page. This property has no default value.

Style sheets obtained through a link object or the **@import** rule cannot be modified if the **designMode** property is enabled.

### 2.4.2 Methods

The **CSSStyleSheet** interface has been extended with the following methods:

- addImport
- addPageRule
- addRule
- removeImport

- removeRule

### 2.4.2.1 addImport

```
addImport()
```

Used to add **@import** rule to the related **CSSStyleSheet** object.

**Parameters**

**sURL** of type `DOMString`

String that represents the location of the source file for the imported style sheet.

**iIndex** of type `long`

Optional. Ordinal index that specifies the requested position of the object in the collection. If this value is not supplied, or if the value is larger than the number of items in the collection, the **@import** rule is added to the end of the collection.

**Return Value**

| | |
|---|---|
| `Long` | Returns the zero-based index position of the new imported style sheet. |

**No Errors**

### 2.4.2.2 addPageRule

```
addPageRule()
```

Used to add **@page** rule to the related **CSSStyleSheet** object. See **StyleSheetPage** in section 2.4. This method is not implemented in either Windows® Internet Explorer® 7 or Windows® Internet Explorer® 8.

**Parameters**

**sSelector** of type `DOMString`

String that specifies the selector (name) for the new **@page** object.

**sStyle** of type `DOMString`

String that specifies the CSS rule assignments for this **@page** object.

**iIndex** of type `long`

Optional. Ordinal index that specifies the requested position of the object in the collection. If this value is not supplied, or if the value is larger than the number of items in the collection, the **@page** rule is added to the end of the collection.

**Return Value**

| | |
|---|---|
| -1 | Reserved. Always returns -1. |

**No Error**

### 2.4.2.3 addRule

`addRule()`

Used to create a new rule in a style sheet. Up to 4095 rules can be added to a single style sheet with this method. If you apply rules to a disabled style sheet, they do not apply until the style sheet is enabled.

**Parameters**

**sSelector** of type `DOMString`

String that specifies the selector for the new rule. Only single selectors are valid; grouped selectors cause "Invalid Argument" error.

**sDeclaration** of type `DOMString`

String that specifies one or more semi-colon separated declarations.

**iIndex** of type `long`

Optional. Ordinal index that specifies the requested position of the object in the collection. If this value is not supplied, or if the value is larger than the number of items in the collection, or if value is -1, the rule is added to the end of the collection.

**Return Value**

| | |
|---|---|
| -1 | Reserved. Always returns -1. |

**JScript Error**

E_INVALIDARG (0x80040057)Raised for grouped selectors, or more than 4095 style rules.

**Example**

The following example demonstrates how to add a rule to the style sheet.

```
<style type="text/css">
p {
  color:red;
}
</style>
<script type="text/javascript">
window.onload = function() {
  var s = document.styleSheets[0];
  var idx = s.addRule('#test:hover','color:green');
}
</script>
</head>

<body>
<p id="test">This text should turn green on hover.</p>
</body>
```

### 2.4.2.4 removeImport

`removeImport()`

Used to delete an imported style sheet from the imports collection. See section 2.4.3.1.

**Parameters**

**iIndex** of type `long`

The ordinal index of the imported style sheet to remove.

**No Return Value**

**JScript Error**

E_INVALIDARG (0x80040057)The specified index was too large or less than `0`.

### 2.4.2.5 removeRule

```
removeRule()
```

Used to remove existing rules from the style sheet.

**Parameters**

**iIndex** of type `long`

The ordinal index of the rule to remove.

**No Return Value**

**JScript Error**

E_INVALIDARG (0x80040057)The specified index was too large or less than `0`.

## 2.4.3 Collections

The **CSSStyleSheet** interface has been extended with the following collections:

- imports
- pages
- rules

### 2.4.3.1 imports

The **imports** property retrieves a **StyleSheetList** (IHTMLStyleSheetsCollection) collection of imported style sheets defined for the respective **CSSStyleSheet** (IHTMLStyleSheet) object. An imported style sheet is one that is linked to the document using the cascading style sheets (CSS) **@import** rule.

The collection contains the same number of style sheets objects that would be referenced by the list of **CSSImportRule** objects in a **CSSRuleList**. The **CSSImportRule** interface is not supported by Windows® Internet Explorer® 7 or Windows® Internet Explorer® 8.

#### 2.4.3.2 pages

The **pages** property retrieves a **StyleSheetPageList** (IHTMLStyleSheetPagesCollection) collection of page objects for the respective **CSSStyleSheet** (IHTMLStyleSheet) object. A page object represents a cascading style sheets (CSS) **@page** rule.

The **StyleSheetPageList** interface is described in section 2.5.

#### 2.4.3.3 rules

The **rules** property retrieves a **CSSRuleList** (IHTMLStyleSheetRulesCollection) collection of rules defined in the respective **CSSStyleSheet** (IHTMLStyleSheet) object.

This collection is always accessible, even if the style sheet is not enabled. Rules are added to the rules collection with the addRule method on the style sheet. A rule that is added to a disabled style sheet does not apply to the document unless the style sheet is enabled. Rules are deleted with the removeRule method.

## 2.5 StyleSheetPage Interface

The **StyleSheetPage** (IHTMLStyleSheetPage) is analogous to the **CSSPageRule** in [DOM Level 2 - Style].

The interface represents a particular @page rule in a style sheet.

**IDL Definition**

```
// Introduced in Internet Explorer
interface StyleSheetPage {
  readonly attribute DOMString      pseudoClass;
  readonly attribute DOMString      selector;
}
```

### 2.5.1 Attributes

The **StyleSheetPage** interface has been extended with the following attributes:

- pseudoClass
- selector

#### 2.5.1.1 pseudoClass

**pseudoClass** of type `DOMString`**, readonly**

A textual representation of the pseudo class used in the **@page** rule.

**Example**

In the example below, `right` is the pseudo-class. The colon is not included.

```
@page :right {margin-left:15px;}
```

#### 2.5.1.2 selector

**selector** of type `DOMString`, **readonly**

A textual representation of the identifier used in a named **@page** rule.

**Example**

In the example below, `rotated` is the selector.

```
@page rotated {size: landscape}
```

## 2.6 StyleSheetPageList Interface

The **StyleSheetPageList** (IHTMLStyleSheetPagesCollection) provides a collection of **@page** rules in a **StyleSheet** object.

**IDL Definition**

```
// Introduced in Internet Explorer
interface StyleSheetPageList : StyleSheetList {
  readonly attribute unsigned long    length;
  StyleSheetPage       item(in unsigned long index);
};
```

### 2.6.1 Attributes

The **StyleSheetPageList** interface has been extended with the length attribute.

#### 2.6.1.1 length

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

The number of **StyleSheetPage** objects in the collection. The range of valid indices ranges from `0` to `length-1` inclusive.

### 2.6.2 Methods

The **StyleSheetPageList** interface has been extended with the item method.

#### 2.6.2.1 item

item

Used to retrieve a **StyleSheetPage** object by ordinal index. If `index` is greater than or equal to the number of objects in the collection, `item` returns `null`.

**Parameters**

**index** of type `unsigned long`

Index into the collection.

**Return Value**

**StyleSheetPage** The CSS **@page** rule at the `index` position in the collection, or `null`.

**No Errors**

# 3 Security Considerations

There are no additional security considerations.

# 4 Appendix A: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include released service packs:

- Windows® Internet Explorer® 7
- Windows® Internet Explorer® 8
- Windows® Internet Explorer® 9
- Windows® Internet Explorer® 10

Exceptions, if any, are noted below. If a service pack or Quick Fix Engineering (QFE) number appears with the product version, behavior changed in that service pack or QFE. The new behavior also applies to subsequent service packs of the product unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

Unless otherwise specified, any statement of optional behavior in this specification that is prescribed using the terms SHOULD or SHOULD NOT implies product behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that the product does not follow the prescription.

# 5 Change Tracking

This section identifies changes that were made to the [MS-CSS21E] protocol document between the February 2012 and July 2012 releases. Changes are classified as New, Major, Minor, Editorial, or No change.

The revision class **New** means that a new document is being released.

The revision class **Major** means that the technical content in the document was significantly revised. Major changes affect protocol interoperability or implementation. Examples of major changes are:

- A document revision that incorporates changes to interoperability requirements or functionality.
- An extensive rewrite, addition, or deletion of major portions of content.
- The removal of a document from the documentation set.
- Changes made for template compliance.

The revision class **Minor** means that the meaning of the technical content was clarified. Minor changes do not affect protocol interoperability or implementation. Examples of minor changes are updates to clarify ambiguity at the sentence, paragraph, or table level.

The revision class **Editorial** means that the language and formatting in the technical content was changed. Editorial changes apply to grammatical, formatting, and style issues.

The revision class **No change** means that no new technical or language changes were introduced. The technical content of the document is identical to the last released version, but minor editorial and formatting changes, as well as updates to the header and footer information, and to the revision summary, may have been made.

Major and minor changes can be described further using the following change types:

- New content added.
- Content updated.
- Content removed.
- New product behavior note added.
- Product behavior note updated.
- Product behavior note removed.
- New protocol syntax added.
- Protocol syntax updated.
- Protocol syntax removed.
- New content added due to protocol revision.
- Content updated due to protocol revision.
- Content removed due to protocol revision.
- New protocol syntax added due to protocol revision.

- Protocol syntax updated due to protocol revision.
- Protocol syntax removed due to protocol revision.
- New content added for template compliance.
- Content updated for template compliance.
- Content removed for template compliance.
- Obsolete document removed.

Editorial changes are always classified with the change type **Editorially updated.**

Some important terms used in the change type descriptions are defined as follows:

- **Protocol syntax** refers to data elements (such as packets, structures, enumerations, and methods) as well as interfaces.
- **Protocol revision** refers to changes made to a protocol that affect the bits that are sent over the wire.

The changes made to this document are listed in the following table. For more information, please contact protocol@microsoft.com.

| Section | Tracking number (if applicable) and description | Major change (Y or N) | Change type |
|---|---|---|---|
| 1 Introduction | Updated document to remove beta tagging. | N | Content updated. |

# 6 Index