CSS Basic User Interface Module Level 3 (CSS3 UI)

W3C Recommendation, 21 June 2018

This version:
https://www.w3.org/TR/2018/REC-css-ui-3-20180621/

Latest published version:
https://www.w3.org/TR/css-ui-3/

Editor's Draft:
https://drafts.csswg.org/css-ui/

Previous Versions:
https://www.w3.org/TR/2017/PR-css-ui-3-20171214/

Test Suite:
http://test.csswg.org/suites/css-ui-3_dev/nightly-unstable/

Editors:
Tantek Çelik (Mozilla) tantek@cs.stanford.edu
Florian Rivoal (On behalf of Bloomberg)

Issue Tracking:
GitHub Issues

Please check the **errata** for any errors or issues reported since publication.

Abstract

This specification describes user interface related properties and values that are proposed for CSS level 3 to style HTML and XML (including XHTML). It includes and extends user interface related features from the properties and values of CSS level 2 revision 1. It uses various properties and values to style basic user interface elements in a document.

CSS is a language for describing the rendering of structured documents (such as HTML and XML) on screen, on paper, in speech, etc.

Status of this document

This section describes the status of this document at the time of its publication. Other documents may supersede this document. A list of current W3C publications and the latest revision of this technical report can be found in the W3C technical reports index at https://www.w3.org/TR/.

This document has been reviewed by W3C Members, by software developers, and by other W3C groups and interested parties, and is endorsed by the Director as a W3C Recommendation. It is a stable document and may be used as reference material or cited 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 was produced by the CSS Working Group.

A W3C Recommendation is a document that has been widely reviewed and is ready for implementation. W3C encourages everybody to implement this specification and return comments to GitHub issues.

This document was produced by a group operating under the W3C Patent Policy. W3C maintains a public list of any patent disclosures made in connection with the deliverables of the group; that page also includes instructions for disclosing a patent. An individual who has actual knowledge of a patent which the individual believes contains Essential Claim(s) must disclose the information in accordance with section 6 of the W3C Patent Policy.

This document is governed by the 1 February 2018 W3C Process Document.

A separate implementation report shows that each required test in the test suite was passed by at least two independent implementations. Please also see the detailed implementation report for this specification.

A complete list of changes to this document is available.

Table of Contents

1 Introduction

1.1 Purpose

2 Module Interactions

3 Box Model addition

3.1 Changing the Box Model: the ‘box-sizing’ property

4 Outline properties
4.1 Outlines Shorthand: the ‘outline’ property
4.2 Outline Thickness: the ‘outline-width’ property
4.3 Outline Patterns: the ‘outline-style’ property
4.4 Outline Colors: the ‘outline-color’ property
4.5 Offsetting the Outline: the ‘outline-offset’ property

5 Resizing & Overflow
5.1 Resizing Boxes: the ‘resize’ property
5.2 Overflow Ellipsis: the ‘text-overflow’ property

6 Pointing Devices and Keyboards
6.1 Pointer interaction
6.1.1 Styling the Cursor: the ‘cursor’ property
6.1.1.1 Cursor of the canvas
6.2 Insertion caret
6.2.1 Coloring the Insertion Caret: the ‘caret-color’ property
6.3 Keyboard control
6.3.1 Obsolete: the ime-mode property

Appendix A. Acknowledgments

Appendix B. Changes

Appendix C. Considerations for Security and Privacy

Appendix D. Default style sheet additions for HTML

Conformance
Document conventions
Conformance classes
Requirements for Responsible Implementation of CSS
Partial Implementations
Implementations of Unstable and Proprietary Features
Implementations of CR-level Features

Index
Terms defined by this specification
Terms defined by reference

References
Normative References
Informative References

Property Index

§ 1. Introduction

This module describes CSS properties which enable authors to style user interface related properties and values.

Section 2.1 of CSS1 [CSS1] and Chapter 18 of CSS2 [CSS2] introduced several user interface related properties and values. User Interface for CSS3 (16 February 2000) introduced several new user interface related features.

This Working Draft incorporates, extends, and supersedes them.

§ 1.1. Purpose

The purpose of this specification is to achieve the following objectives:

- Extend the user interface features in CSS2.1.
- Provide additional CSS mechanisms to augment or replace other dynamic presentation related features in HTML.

§ 2. Module Interactions

This document defines new features not present in earlier specifications. In addition, it replaces and supersedes the following:

- Section 18.1, section 18.4, and Information on the stacking of outlines defined in Appendix E of Cascading Style Sheets, level 2, revision 1 [CSS2]
- User Interface for CSS3 (16 February 2000)

Note: The semantics of property definition tables were first introduced in Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification §property-defs. More up-to-date definitions can be found in [css-transitions-1], [css-values-3], and [css-cascade-4]. For the reader's convenience, this specification links directly to these terms where relevant.

§ 3. Box Model addition

§ 3.1. Changing the Box Model: the 'box-sizing' property

Name:	***'box-sizing'***
Value:	content-box | border-box
Initial:	content-box
Applies to:	all elements that accept width or height
Inherited:	no
Percentages:	N/A
Media:	visual
Computed value:	specified value
Canonical order:	per grammar
Animation type:	discrete

'content-box'

This is the behavior of width and height as specified by CSS2.1. The specified width and height (and respective min/max properties) apply to the width and height respectively of the content box of the element. The padding and border of the element are laid out and drawn outside the specified width and height.

'border-box'

Length and percentages values for width and height (and respective min/max properties) on this element determine the border box of the element. That is, any padding or border specified on the element is laid out and drawn inside this specified width and height. The content width and height are calculated by subtracting the border and padding widths of the respective sides from the specified ‘width’ and ‘height’ properties. As the content width and height cannot be negative ([CSS2], section 10.2), this computation is floored at 0. Used values, as exposed for instance through getComputedStyle(), also refer to the border box.

> Note: This is the behavior of width and height as commonly implemented by legacy HTML user agents for replaced elements and input elements.

Note: In contrast to the length and percentage values, the ‘auto’ value of the ‘width’ and ‘height’ properties (as well as other keyword values introduced by later specifications, unless otherwise specified) is not influenced by the ‘box-sizing’ property, and always sets the size of the content box.

The following terms, whose definitions vary based on the computed value of ‘box-sizing’ are introduced:

	‘box-sizing: content-box’	**‘box-sizing: border-box’**
min inner width	‘min-width’	max(0, ‘min-width’ − ‘padding-left’ − ‘padding-right’ − ‘border-left-width’ − ‘border-right-width’)
max inner width	‘max-width’	max(0, ‘max-width’ − ‘padding-left’ − ‘padding-right’ − ‘border-left-width’ − ‘border-right-width’)
min inner height	‘min-height’	max(0, ‘min-height’ − ‘padding-top’ − ‘padding-bottom’ − ‘border-top-width’ − ‘border-bottom-width’)
max inner height	‘max-height’	max(0, ‘max-height’ − ‘padding-top’ − ‘padding-bottom’ − ‘border-top-width’ − ‘border-bottom-width’)

The Visual formatting model details of [CSS2] are written assuming ‘box-sizing: content-box’. The following disambiguations are made to clarify the behavior for all values of ‘box-sizing’:

1. In 10.3.3, the second “width” in the following phrase is to be interpreted as content width: “If ‘width’ is not ‘auto’ and ‘border-left-width’ + ‘padding-left’ + ‘width’ + [...]”
2. In 10.3.7, “width” is to be interpreted as content width in the following equation: “‘left’ + ‘margin-left’ + ‘border-left-width’ + ‘padding-left’ + ‘width’ + [...]”
3. In 10.4, “width”, “height”, “min-width”, “max-width”, “min-height” and “max-height” are respectively to be interpreted as content width, content height, min inner width, max inner width, min inner height and max inner height in the following phrases:
    1. “The tentative used width is calculated [...]”
    2. “If the tentative used width is greater than ‘max-width’, the rules above are applied again, but this time using the computed value of ‘max-width’ as the computed value for ‘width’.”
    3. “If the resulting width is smaller than ‘min-width’, the rules above are applied again, but this time using the value of ‘min-width’ as the computed value for ‘width’.”
    4. “Select from the table the resolved height and width values for the appropriate constraint violation. Take the max-width and max-height as max(min, max) so that min ≤ max holds true. In this table w and h stand for the results of the width and height computations [...]”
    5. All instances of these words in the table
    6. “Then apply the rules under "Calculating widths and margins" above, as if ‘width’ were computed as this value.”
4. In 10.6.4, “height” is to be interpreted as content height in the following equation: “‘top’ + ‘margin-top’ + ‘border-top-width’ + ‘padding-top’ + ‘height’ + [...]”
5. In 10.7, “width”, “height”, “min-height” and “max-height” are respectively to be interpreted as content width, content height, min inner height and max inner height in the following phrases:
    1. “The tentative used height is calculated [...]”
    2. “If this tentative height is greater than ‘max-height’, the rules above are applied again, but this time using the value of ‘max-height’ as the computed value for ‘height’.”
    3. “If the resulting height is smaller than ‘min-height’, the rules above are applied again, but this time using the value of ‘min-height’ as the computed value for ‘height’.”
    4. “[...] use the algorithm under Minimum and maximum widths above to find the used width and height. Then apply the rules under "Computing heights and margins" above, using the resulting width and height as if they were the computed values.”

EXAMPLE 1

Using box-sizing to evenly share space

This example uses box-sizing to evenly horizontally split two divs with fixed size borders inside a div container, which would otherwise require additional markup.

sample CSS:

```
div.container {
  width:38em;
  border:1em solid black;
}

div.split {
  box-sizing:border-box;
  width:50%;
  border:1em silver ridge;
  float:left;
}
```

sample HTML fragment:

```
<div class="container">
<div class="split">This div occupies the left half.</div>
<div class="split">This div occupies the right half.</div>
</div>
```

demonstration of sample CSS and HTML:

This div should occupy the left half.

This div should occupy the right half.

The two divs above should appear side by side, each (including borders) 50% of the content width of their container. If instead they are stacked one on top of the other then your browser does not support ‘box-sizing’.

§ 4. Outline properties

At times, style sheet authors may want to create outlines around visual objects such as buttons, active form fields, image maps, etc., to make them stand out. Outlines differ from borders in the following ways:

1. Outlines do not take up space.
2. Outlines may be non-rectangular.
3. UAs often render outlines on elements in the :focus state.

The outline properties control the style of these dynamic outlines.

The stacking of the rendering of these outlines is explicitly left up to implementations to provide a better user experience per platform. This supersedes the stacking of outlines as defined in Appendix E of CSS 2.1 [CSS2].

> **Keyboard users, in particular people with disabilities who may not be able to interact with the page in any other fashion, depend on the outline being visible on elements in the :focus state, thus authors must not make the outline invisible on such elements without making sure an alternative highlighting mechanism is provided.**

The rendering of applying transforms to outlines is left explicitly undefined in CSS3-UI.

§ 4.1. Outlines Shorthand: the ‘outline’ property

Name:	***'outline'***
Value:	[<outline-color> || <outline-style> || <outline-width>]
Initial:	see individual properties
Applies to:	all elements
Inherited:	no
Percentages:	N/A
Media:	visual
Computed value:	see individual properties
Canonical order:	per grammar
Animation type:	see individual properties

§ 4.2. Outline Thickness: the 'outline-width' property

Name:	***'outline-width'***
Value:	<line-width>
Initial:	medium
Applies to:	all elements
Inherited:	no
Percentages:	N/A
Media:	visual
Computed value:	absolute length; '0' if the outline style is 'none'.
Canonical order:	per grammar
Animation type:	length

§ 4.3. Outline Patterns: the 'outline-style' property

Name:	***'outline-style'***
Value:	auto | <border-style>
Initial:	none
Applies to:	all elements
Inherited:	no
Percentages:	N/A
Media:	visual
Computed value:	as specified
Canonical order:	per grammar
Animation type:	discrete

§ 4.4. Outline Colors: the 'outline-color' property

Name:	***'outline-color'***
Value:	<color> | invert
Initial:	invert
Applies to:	all elements
Inherited:	no
Percentages:	N/A
Media:	visual
Computed value:	The computed value for 'invert' is 'invert'; the computed value of 'currentColor' is 'currentColor' (See CSS Color Module Level 3 §#currentColor); see the 'color' property for other <color> values.
Canonical order:	per grammar
Animation type:	color

The outline created with the outline properties is drawn "over" a box, i.e., the outline is always on top, and doesn't influence the position or size of the box, or of any other boxes. Therefore, displaying or suppressing outlines does not cause reflow.

Outlines may be non-rectangular. For example, if the element is broken across several lines, the outline should be an outline or minimum set of outlines that encloses all the element's boxes.

Each part of the outline should be fully connected rather than open on some sides (as borders on inline elements are when lines are broken).

The parts of the outline are not required to be rectangular. To the extent that the outline follows the border edge, it should follow the 'border-radius' curve.

The position of the outline may be affected by descendant boxes.

User agents should use an algorithm for determining the outline that encloses a region appropriate for conveying the concept of focus to the user.

> Note: This specification does not define the exact position or shape of the outline, but it is typically drawn immediately outside the border box.

The ‘outline-width’ property accepts the same values as ‘border-width’ (CSS Backgrounds 3 §4.3 Line Thickness: the border-width properties).

The ‘outline-style’ property accepts the same values as ‘border-style’ (CSS Backgrounds 3 §4.2 Line Patterns: the border-style properties), except that ‘hidden’ is not a legal outline style. In addition, in CSS3, ‘outline-style’ accepts the value ‘auto’. The ‘auto’ value permits the user agent to render a custom outline style, typically a style which is either a user interface default for the platform, or perhaps a style that is richer than can be described in detail in CSS, e.g. a rounded edge outline with semi-translucent outer pixels that appears to glow. As such, this specification does not define how the ‘outline-color’ is incorporated or used (if at all) when rendering ‘auto’ style outlines. User agents may treat ‘auto’ as ‘solid’.

The ‘outline-color’ property accepts all colors, as well as the keyword ***‘invert’***. ‘Invert’ is expected to perform a color inversion on the pixels on the screen. This is a common trick to ensure the focus border is visible, regardless of color background.

Conformant UAs may ignore the ‘invert’ value on platforms that do not support color inversion of the pixels on the screen.

If the UA does not support the ‘invert’ value then it must reject that value at parse-time, and the initial value of the ‘outline-color’ property is the ‘currentColor’ keyword.

The ‘outline’ property is a shorthand property, and sets all three of ‘outline-style’, ‘outline-width’, and ‘outline-color’.

> Note: The outline is the same on all sides. In contrast to borders, there are no ‘outline-top’ or ‘outline-left’ etc. properties.

This specification does not define how multiple overlapping outlines are drawn, or how outlines are drawn for boxes that are partially obscured behind other elements.

EXAMPLE 2

Here's an example of drawing a thick outline around a BUTTON element:

```
button { outline: thick solid }
```

Graphical user interfaces may use outlines around elements to tell the user which element on the page has the focus. These outlines are in addition to any borders, and switching outlines on and off should not cause the document to reflow. The focus is the subject of user interaction in a document (e.g. for entering text or selecting a button).

EXAMPLE 3

For example, to draw a thick black line around an element when it has the focus, and a thick red line when it is active, the following rules can be used:

```
:focus  { outline: thick solid black }
:active { outline: thick solid red }
```

Note: Since the outline does not affect formatting (i.e., no space is left for it in the box model), it may well overlap other elements on the page.

§ 4.5. Offsetting the Outline: the 'outline-offset' property

By default, the outline is drawn starting just outside the border edge. However, it is possible to offset the outline and draw it beyond the border edge.

Name:	***'outline-offset'***
Value:	<length>
Initial:	0
Applies to:	all elements
Inherited:	no
Percentages:	N/A
Media:	visual
Computed value:	<length> value in absolute units (px or physical).
Canonical order:	per grammar
Animation type:	length

If the computed value of 'outline-offset' is anything other than 0, then the outline is outset from the border edge by that amount.

EXAMPLE 4

For example, to leave 2 pixels of space between a focus outline and the element that has the focus or is active, the following rule can be used:

```
:focus,:active  { outline-offset: 2px }
```

Negative values must cause the outline to shrink into the border box. Both the height and the width of outside of the shape drawn by the outline should not become smaller than twice the computed value of the 'outline-width' property, to make sure that an outline can be rendered even with large negative values. User Agents should apply this constraint independently in each dimension. If the outline is drawn as multiple disconnected shapes, this constraint applies to each shape separately.

§ 5. Resizing & Overflow

CSS2.1 provides a mechanism for controlling the appearance of a scrolling mechanism (e.g. scrollbars) on block container elements. This specification adds to that a mechanism for controlling user resizability of elements as well as the ability to specify text overflow behavior.

§ 5.1. Resizing Boxes: the 'resize' property

The 'resize' property allows the author to specify whether or not an element is resizable by the user, and if so, along which axis/axes.

Name:	***'resize'***
Value:	none ¦ both ¦ horizontal ¦ vertical
Initial:	none
Applies to:	elements with 'overflow' other than visible, and optionally replaced elements such as images, videos, and iframes
Inherited:	no
Percentages:	N/A
Media:	visual
Computed value:	as specified
Canonical order:	per grammar
Animation type:	discrete

none

The UA does not present a resizing mechanism on the element, and the user is given no direct manipulation mechanism to resize the element.

both

The UA presents a bidirectional resizing mechanism to allow the user to adjust both the height and the width of the element.

horizontal
: The UA presents a unidirectional horizontal resizing mechanism to allow the user to adjust only the width of the element.

vertical
: The UA presents a unidirectional vertical resizing mechanism to allow the user to adjust only the height of the element.

Currently it is possible to control the appearance of the scrolling mechanism (if any) on an element using the 'overflow' property (e.g. `overflow: scroll` vs. `overflow: hidden` etc.). The purpose of the 'resize' property is to allow control over the appearance and function of the resizing mechanism (e.g. a resize box or widget) on the element.

> Note: The resizing mechanism is NOT the same as the scrolling mechanism, nor is it related to any UA mechanism for zooming. The scrolling mechanism allows the user to determine which portion of the contents of an element is shown. The resizing mechanism allows the user to determine the size of the element.

The 'resize' property applies to elements whose computed 'overflow' value is something other than 'visible'. UAs may also apply it, regardless of the value of the 'overflow' property, to:

- Replaced elements representing images or videos, such as `<img>`, `<video>`, `<picture>`, `<svg>`, `<object>`, or `<canvas>`.
- The `<iframe>` element.

The effect of the 'resize' property on generated content is undefined. Implementations should not apply the 'resize' property to generated content.

> Note: the 'resize' property may apply to generated content in the future if there is implementation of Interface CSSPseudoElement (See [css-pseudo-4]).

When an element is resized by the user, the user agent sets the 'width' and 'height' properties to px unit length values of the size indicated by the user, in the element's style attribute DOM, replacing existing property declaration(s), if any, without '!important', if any.

If an element is resized in only one dimension, only the corresponding property is set, not both.

The precise direction of resizing (i.e. altering the top left of the element or altering the bottom right) may depend on a number of CSS layout factors including whether the element is absolutely

positioned, whether it is positioned using the 'right' and 'bottom' properties, whether the language of the element is right-to-left etc. The UA should consider the direction of resizing (as determined by CSS layout), as well as platform conventions and constraints when deciding how to convey the resizing mechanism to the user.

The user agent must allow the user to resize the element with no other constraints than what is imposed by 'min-width', 'max-width', 'min-height', and 'max-height'.

> Note: There may be situations where user attempts to resize an element appear to be overriden or ignored, e.g. because of '!important' cascading declarations that supersede that element's style attribute 'width' and 'height' properties in the DOM.

Changes to the computed value of an element's 'resize' property do not reset changes to the style attribute made due to user resizing of that element.

> EXAMPLE 5
>
> For example, to make iframes scrollable *and* resizable, the following rule can be used:

```
iframe,object[type^="text/"],
object[type$="+xml"],object[type="application/xml"]
{
  overflow:auto;
  resize:both;
}
```

§ 5.2. Overflow Ellipsis: the 'text-overflow' property

Name:	***'text-overflow'***
Value:	clip | ellipsis
Initial:	clip
Applies to:	block containers
Inherited:	no
Percentages:	N/A
Media:	visual
Computed value:	as specified
Canonical order:	per grammar
Animation type:	discrete

This property specifies rendering when inline content overflows its end line box edge in the inline progression direction of its block container element ("the block") that has 'overflow' other than 'visible'.

Text can overflow for example when it is prevented from wrapping (e.g. due to `white-space: nowrap` or a single word is too long to fit). Values have the following meanings:

'clip'
: Clip inline content that overflows its block container element. Characters may be only partially rendered.

'ellipsis'
: Render an ellipsis character (U+2026) to represent clipped inline content. Implementations may substitute a more language, script, or writing-mode appropriate ellipsis character, or three dots "..." if the ellipsis character is unavailable.

The term "character" is used in this property definition for better readability and means "grapheme cluster" [UAX29] for implementation purposes.

For the ellipsis value implementations must hide characters and atomic inline-level elements at the end edge of the line as necessary to fit the ellipsis, and place the ellipsis immediately adjacent to the end edge of the remaining inline content. The first character or atomic inline-level element on a line must be clipped rather than ellipsed.

> EXAMPLE 6
>
> **Bidi ellipsis examples**
>
> These examples demonstrate which characters get hidden to make room for the ellipsis in a bidi situation: those visually at the end edge of the line.
>
> Sample CSS:
>
> ```
> div {
>   font-family: monospace;
>   white-space: pre;
>   overflow: hidden;
>   width: 9ch;
>   text-overflow: ellipsis;
> }
> ```
>
> Sample HTML fragments, renderings, and your browser:
>
> | HTML | Reference rendering | Your Browser |
> | --- | --- | --- |
> | `<div>123456 שלום</div>` | 123456 ם… | 123456 ם… |
> | `<div dir=rtl>123456 שלום</div>` | …456 שלום | ..3456 שלום |

§ ellipsing details

- Ellipsing only affects rendering and must not affect layout nor dispatching of pointer events: The UA should dispatch any pointer event on the ellipsis to the elided element, as if 'text-overflow' had been 'none'.
- The ellipsis is styled and baseline-aligned according to the block.

- Ellipsing occurs after relative positioning and other graphical transformations.
- If there is insufficient space for the ellipsis, then clip the rendering of the ellipsis itself (on the same side that neutral characters on the line would have otherwise been clipped with the ‘text-overflow:clip’ value).

§ user interaction with ellipsis

- When the user is interacting with content (e.g. editing, selecting, scrolling), the user agent may treat ‘text-overflow: ellipsis’ as ‘text-overflow: clip’.
- Selecting the ellipsis should select the ellipsed text. If all of the ellipsed text is selected, UAs should show selection of the ellipsis. Behavior of partially-selected ellipsed text is up to the UA.

EXAMPLE 7

text-overflow examples

These examples demonstrate setting the text-overflow of a block container element that has text which overflows its dimensions:

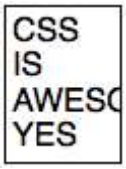
sample CSS for a div:

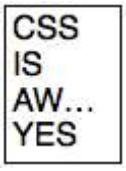
```
div {  font-family:Helvetica,sans-serif; line-height:1.1;
  width:3.1em; padding:.2em; border:solid .1em black; margin:1em 0;
}
```

sample HTML fragments, renderings, and your browser:

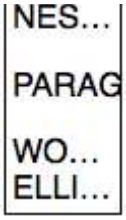
HTML	sample rendering	your browser

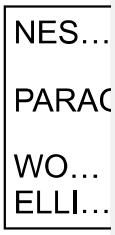
`<div>`
`CSS IS AWESOME, YES`
`</div>`
		

`<div style="text-overflow:clip; overflow:hidden">`
`CSS IS AWESOME, YES`
`</div>`
		

`<div style="text-overflow:ellipsis; overflow:hidden">`
`CSS IS AWESOME, YES`
`</div>`
		

```
<div style="text-overflow:ellipsis; overflow:hidden">
NESTED
 <p>PARAGRAPH</p>
WON'T ELLIPSE.
</div>
```





Note: the side of the line that the ellipsis is placed depends on the 'direction' of the block. E.g. an overflow hidden right-to-left (`direction: rtl`) block clips inline content on the left side, thus would place a text-overflow ellipsis on the left to represent that clipped content.

§ ellipsis interaction with scrolling interfaces

This section applies to elements with text-overflow other than 'text-overflow:clip' (non-clip text-overflow) and overflow:scroll.

When an element with non-clip text-overflow has overflow of scroll in the inline progression dimension of the text, and the browser provides a mechanism for scrolling (e.g. a scrollbar on the element, or a touch interface to swipe-scroll, etc.), there are additional implementation details that provide a better user experience:

When an element is scrolled (e.g. by the user, DOM manipulation), more of the element's content is shown. The value of text-overflow should not affect whether more of the element's content is shown or not. If a non-clip text-overflow is set, then as more content is scrolled into view, implementations should show whatever additional content fits, only truncating content which would otherwise be clipped (or is necessary to make room for the ellipsis/string), until the element is scrolled far enough to display the edge of the content at which point that content should be displayed rather than an ellipsis/string.

EXAMPLE 8

This example uses text-overflow on an element with overflow scroll to demonstrate the above described behavior.

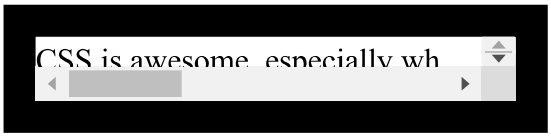
sample CSS:

```
div.crawlbar {
  text-overflow: ellipsis;
  height: 2em;
  overflow: scroll;
  white-space: nowrap;
  width: 15em;
  border:1em solid black;
}
```

sample HTML fragment:

```
<div class="crawlbar">
CSS is awesome, especially when you can scroll
to see extra text instead of just
having it overlap other text by default.
</div>
```

demonstration of sample CSS and HTML:



While the content is being scrolled, implementations may adjust their rendering of ellipses (e.g. align to the box edge rather than line edge).

§ 6. Pointing Devices and Keyboards

§ 6.1. Pointer interaction

§ 6.1.1. Styling the Cursor: the 'cursor' property

Name:	***'cursor'***
Value:	[[<url> [<x> <y>]?,]* [auto ¦ default ¦ none ¦ context-menu ¦ help ¦ pointer ¦ progress ¦ wait ¦ cell ¦ crosshair ¦ text ¦ vertical-text ¦ alias ¦ copy ¦ move ¦ no-drop ¦ not-allowed ¦ grab ¦ grabbing ¦ e-resize ¦ n-resize ¦ ne-resize ¦ nw-resize ¦ s-resize ¦ se-resize ¦ sw-resize ¦ w-resize ¦ ew-resize ¦ ns-resize ¦ nesw-resize ¦ nwse-resize ¦ col-resize ¦ row-resize ¦ all-scroll ¦ zoom-in ¦ zoom-out]]
Initial:	auto
Applies to:	all elements
Inherited:	yes
Percentages:	N/A
Media:	visual, interactive
Computed value:	as specified, except with any relative URLs converted to absolute
Canonical order:	per grammar
Animation type:	discrete

This property specifies the type of cursor to be displayed for the pointing device when the cursor's hotspot is within the element's border edge.

Note: As per CSS Backgrounds 3 §5.1 Curve Radii: the border-radius properties, the border edge is affected by 'border-radius'.

In the case of overlapping elements, which element determines the type of cursor is based on hit testing: the element determining the cursor is the one that would receive a click initiated from this

position.

> Note: The specifics of hit testing are out of scope of this specification. Hit testing will hopefully be defined in a future revision of CSS or HTML.

User agents may ignore the cursor property over native user-agent controls such as scrollbars, resizers, or other native UI widgets e.g. those that may be used inside some user agent specific implementations of form elements. User agents may also ignore the cursor property and display a cursor of their choice to indicate various states of the UA's user interface, such as a busy cursor when the page is not responding, or a text cursor when the user is performing text selection.

> Note: [HTML] defines special handling of image maps for the 'cursor' property.

Values have the following meanings:

image cursors

<url>
: The user agent retrieves the cursor from the resource designated by the URI. If the user agent cannot handle the first cursor of a list of cursors, it must attempt to handle the second, etc. If the user agent cannot handle any user-defined cursor, it must use the cursor keyword at the end of the list. Conforming User Agents may, instead of <url>, support <image> which is a superset.

  The UA must support the following image file formats:

  - PNG, as defined in [PNG]
  - SVG, as defined in [SVG11], in secure static mode [SVG2], if it has an intrinsic size.
  - any other non-animated image file format that they support for <image> in other properties, such as the the 'background-image' property

  In addition, the UA should support the following image file formats:

  - SVG, as defined in [SVG11], in secure animated mode [SVG2], if it has an intrinsic size.
  - any other animated image file format that they support for <image> in other properties, such as the the 'background-image' property

  The UA may also support additional file formats, including SVG, as defined in [SVG11], in secure static mode or secure animated mode [SVG2], even if it does not have an intrinsic size.

> Note: The CSS Working group initially intended support for all SVG, intrinsically sized or not. Support for non intrinsically sized SVG was downgraded from mandatory to optional due to lack of implementations.

> Note: At the time of writing this specification (spring 2015), the only file formats supported for cursors in common desktop browsers are the .ico and .cur file formats, as designed by Microsoft. For compatibility with legacy content, UAs are encouraged to support these, even though the lack of an open specification makes it impossible to have a normative requirement about these formats. Some information on these formats can be found on Wikipedia.

The default object size for cursor images is a UA-defined size that should be based on the size of a typical cursor on the UA's operating system.

The concrete object size is determined using the default sizing algorithm. If an operating system is **incapable** of rendering a cursor above a given size, cursors larger than that size must be shrunk to within the OS-supported size bounds, while maintaining the cursor image's intrinsic ratio, if any.

The optional <x> and <y> coordinates identify the exact position within the image which is the pointer position (i.e., the hotspot).

<x>
<y>

Each is a <number>. The x-coordinate and y-coordinate of the position in the cursor's coordinate system (left/top relative) which represents the precise position that is being pointed to.

> Note: This specification does not define how the coordinate systems of the various types of <image> are established, and defers these definitions to [CSS4-IMAGES].

If the values are unspecified, then the intrinsic hotspot defined inside the image resource itself is used. If both the values are unspecific and the referenced cursor has no defined hotspot, the effect is as if a value of "0 0" were specified.

If the coordinates of the hotspot, as specified either inside the image resource or by <x> and <y> values, fall outside of the cursor image, they must be clamped (independently) to fit.

general purpose cursors

'auto'

The UA determines the cursor to display based on the current context, specifically: auto behaves as ‘text’ over selectable text or editable elements, and ‘default’ otherwise.

‘default’

The platform-dependent default cursor. Often rendered as an arrow.

‘none’

No cursor is rendered for the element.

links and status cursors

‘context-menu’

A context menu is available for the object under the cursor. Often rendered as an arrow with a small menu-like graphic next to it.

‘help’

Help is available for the object under the cursor. Often rendered as a question mark or a balloon.

‘pointer’

The cursor is a pointer that indicates a link.

‘progress’

A progress indicator. The program is performing some processing, but is different from ‘wait’ in that the user may still interact with the program. Often rendered as a spinning beach ball, or an arrow with a watch or hourglass.

‘wait’

Indicates that the program is busy and the user should wait. Often rendered as a watch or hourglass.

selection cursors

‘cell’

Indicates that a cell or set of cells may be selected. Often rendered as a thick plus-sign with a dot in the middle.

‘crosshair’

A simple crosshair (e.g., short line segments resembling a "+" sign). Often used to indicate a two dimensional bitmap selection mode.

‘text’

Indicates text that may be selected. Often rendered as a vertical I-beam. User agents may automatically display a horizontal I-beam/cursor (e.g. same as the ‘vertical-text’ keyword) for vertical text, or for that matter, any angle of I-beam/cursor for text that is rendered at any particular angle.

‘vertical-text’

Indicates vertical-text that may be selected. Often rendered as a horizontal I-beam.

drag and drop cursors

'alias'

Indicates an alias of/shortcut to something is to be created. Often rendered as an arrow with a small curved arrow next to it.

'copy'

Indicates something is to be copied. Often rendered as an arrow with a small plus sign next to it.

'move'

Indicates something is to be moved.

'no-drop'

Indicates that the dragged item cannot be dropped at the current cursor location. Often rendered as a hand or pointer with a small circle with a line through it.

'not-allowed'

Indicates that the requested action will not be carried out. Often rendered as a circle with a line through it.

'grab'

Indicates that something can be grabbed (dragged to be moved). Often rendered as the backside of an open hand.

'grabbing'

Indicates that something is being grabbed (dragged to be moved). Often rendered as the backside of a hand with fingers closed mostly out of view.

resizing and scrolling cursors

'e-resize', ***'n-resize'***, ***'ne-resize'***, ***'nw-resize'***, ***'s-resize'***, ***'se-resize'***, ***'sw-resize'***, ***'w-resize'***

Indicates that some edge is to be moved. For example, the 'se-resize' cursor is used when the movement starts from the south-east corner of the box.

'ew-resize', ***'ns-resize'***, ***'nesw-resize'***, ***'nwse-resize'***

Indicates a bidirectional resize cursor.

'col-resize'

Indicates that the item/column can be resized horizontally. Often rendered as arrows pointing left and right with a vertical bar separating them.

'row-resize'

Indicates that the item/row can be resized vertically. Often rendered as arrows pointing up and down with a horizontal bar separating them.

'all-scroll'

Indicates that the something can be scrolled in any direction. Often rendered as arrows pointing up, down, left, and right with a dot in the middle.

¶ **zooming cursors**

'zoom-in'*, *'zoom-out'

Indicates that something can be zoomed (magnified) in or out, and often rendered as a magnifying glass with a "+" or "-" in the center of the glass, for 'zoom-in' and 'zoom-out' respectively.

EXAMPLE 9

Example: cursor fallback

Here is an example of using several cursor values.

```
:link,:visited {
    cursor: url(example.svg#linkcursor),
            url(hyper.cur),
            url(hyper.png) 2 3,
            pointer
}
```

This example sets the cursor on all hyperlinks (whether visited or not) to an external SVG cursor ([SVG11], section 16.8.3). User agents that don't support SVG cursors would simply skip to the next value and attempt to use the "hyper.cur" cursor. If that cursor format was also not supported, the UA could attempt to use the "hyper.png" cursor with the explicit hotspot. Finally if the UA does not support any of those image cursor formats, the UA would skip to the last value and render the 'pointer' cursor.

§ *6.1.1.1. Cursor of the canvas*

The document canvas is the infinite surface over which the document is rendered [CSS2]. Since no element corresponds to the canvas, in order to allow styling of the cursor when not over any element, the canvas cursor re-uses the root element's cursor. However, if no boxes are generated for the root element (for example, if the root element has 'display: none'), then the canvas cursor is the platform-dependent default cursor.

Note: An element might be invisible, but still generate boxes. For example, if the element has ‘visibility: hidden’ but not ‘display: none’, boxes are generated for it and its cursor is used for the canvas.

§ 6.2. Insertion caret

§ 6.2.1. Coloring the Insertion Caret: the ‘caret-color’ property

Name:	***‘caret-color’***
Value:	auto | <color>
Initial:	auto
Applies to:	all elements
Inherited:	yes
Percentages:	N/A
Media:	interactive
Computed value:	The computed value for ‘auto’ is ‘auto’; the computed value of ‘currentColor’ is ‘currentColor’ (See CSS Color Module Level 3 §#currentColor); see the ‘color’ property for other <color> values.
Canonical order:	per grammar
Animation type:	color

‘auto’

User agents should use currentColor. User agents may automatically adjust the color of caret to ensure good visibility and contrast with the surrounding content, possibly based on the currentColor, background, shadows, etc.

<color>

The insertion caret is colored with the specified color.

The caret is a visible indicator of the insertion point in an element where text (and potentially other content) is inserted by the user. This property controls the color of that visible indicator.

> Note: caret shape and blinking is outside the scope of this feature and thus unspecified.

> Note: UAs might have additional things that count as “carets”. For example, some UAs can show a “navigation caret”, which acts similarly to an insertion caret but can be moved around in non-editable text, and is functionally a caret. On the other hand, the cursor image shown when hovering over text when the ‘cursor’ property is ‘auto’, or when hovering over an element where the ‘cursor’ property is ‘text’ or ‘vertical-text’, though it sometimes resembles a caret, is not a caret (it’s a cursor).

> EXAMPLE 10
>
> Example: a textarea with `caret-color:#00aacc;`

```
caret-color:#00aacc
```

§ 6.3. Keyboard control

§ 6.3.1. Obsolete: the ime-mode property

"ime-mode" is a property somewhat implemented in some browsers, that is problematic and officially obsoleted by this specification.

There is documentation of non-interoperability of these implementations.

User agents should not support the ‘ime-mode’ property.

Authors must not use the ime-mode property.

Users may use the ime-mode property only for repair use-cases where they have to work around bad sites and legacy implementations, e.g. with a user style sheet rule like:

EXAMPLE 11

Example: user preference

```
input[type=password] {    ime-mode: auto !important;
}
```

This example CSS may be placed into a user style sheet file to force password input fields to behave in a default manner.

This specification deliberately does not attempt to document the functionality of legacy ime-mode implementations nor what they specifically support because it does not make sense to pursue or recommend any such path.

Note: there are several [HTML] features which authors should use to provide information to user agents that allow them to provide a better input user experience:

- The global `lang` attribute
- The `inputmode`, `pattern`, and `type` attributes of the input element

§ Appendix A. Acknowledgments

This appendix is *informative*.

This specification was edited and written for the most part by Tantek Çelik from 1999 to the present, first while representing Microsoft, then as an Invited Expert, and most recently while representing Mozilla.

Thanks to Florian Rivoal, working on this specification on behalf of Bloomberg, for his recent work documenting issues from www-style emails, proposing resolutions & changes, and in particular for researching & writing greatly improved details for the 'box-sizing' property.

Thanks to feedback and contributions from Rossen Atanassov, Tab Atkins, L. David Baron, Bert Bos, Matthew Brealey, Rick Byers, Ada Chan, James Craig, Michael Cooper, Axel Dahmen, Michael Day,

Micah Dubinko, Elika E., Steve Falkenburg, Andrew Fedoniouk, Al Gilman, Ian Hickson, Bjoern Hoehrmann, Alan Hogan, David Hyatt, Richard Ishida, Sho Kuwamoto, Yves Lafon, Stuart Langridge, Susan Lesch, Peter Linss, Kang-Hao Lu, Masayuki Nakano, Mats Palmgren, Brad Pettit, Chris Rebert, François Remy, Andrey Rybka, Simon Sapin, Alexander Savenkov, Sebastian Schnitzenbaumer, Lea Verou, Etan Wexler, David Woolley, Frank Yan, Boris Zbarsky, and Domel.

§ Appendix B. Changes

This appendix is *informative*.

Since Proposed Recommendation

This portion of the appendix describes changes from the Proposed Recommendation (PR) of 14 December 2017.

- Updated references to latest versions
- Date and boilerplate changes for W3C Recommendation
- Link to errata document added
- Updated this changes section

Since Candidate Recommendation

This portion of the appendix describes changes from the Candidate Recommendation (CR) of 2 March 2017.

- Updated references to latest versions
- Editorial Clarification about the resize property
- Move (at risk) directional focus navigation properties from level 3 to level 4
- Add informative link to HTML about special handling of 'cursor' over image maps
- Clarify (as a SHOULD) the implications of text-overflow on pointer events to capture implementor consensus (corresponding test).
- Clarify that UAs may ignore the cursor property to reflect the UA's UI state
- Allow, but stop requiring support for SVG images without intrinsic sizes for cursors (corresponding test update).

- Align the spec with implementations, and make ‘cursor: auto’ look like ‘text’ over **selectable** text, and over editable elements (corresponding tests).

§ Appendix C. Considerations for Security and Privacy

This appendix is *informative*.

The W3C TAG is developing a Self-Review Questionnaire: Security and Privacy for editors of specifications to informatively answer.

Per the Questions to Consider

1. Does this specification deal with personally-identifiable information?
   No.

2. Does this specification deal with high-value data?
   No.

3. Does this specification introduce new state for an origin that persists across browsing sessions?
   No.

4. Does this specification expose persistent, cross-origin state to the web?
   No.

5. Does this specification expose any other data to an origin that it doesn’t currently have access to?
   No.

6. Does this specification enable new script execution/loading mechanisms?
   Yes to loading, but not to execution. The ‘cursor’ property accepts <image> values which may include URLs to be loaded. These may be SVG documents which may contain scripts, but this specification requires that scripts must not be run.

7. Does this specification allow an origin access to a user’s location?
   No.

8. Does this specification allow an origin access to sensors on a user’s device?
   No.

9. Does this specification allow an origin access to aspects of a user’s local computing environment?
   No.

10. Does this specification allow an origin access to other devices?

No.

11. Does this specification allow an origin some measure of control over a user agent's native UI?
    Yes. The 'cursor' and 'caret-color' properties enable the page to change the display of the cursor and text insertion caret of the user agent's native UI. In addition the 'outline-style' property's 'auto' value (and thus 'outline' shorthand) enable the page to potentially display a native focused element outline presentation around any element.

12. Does this specification expose temporary identifiers to the web?
    No.

13. Does this specification distinguish between behavior in first-party and third-party contexts?
    No.

14. How should this specification work in the context of a user agent's "incognito" mode?
    No differently.

15. Does this specification persist data to a user's local device?
    No.

16. Does this specification have a "Security Considerations" and "Privacy Considerations" section?
    Yes.

17. Does this specification allow downgrading default security characteristics?
    No.

§ Appendix D. Default style sheet additions for HTML

This appendix is *informative*.

Potential additions to the base style sheet to express HTML form controls, and a few dynamic presentation attributes:

```
:enabled:focus {
 outline: 2px inset;
}

button,
input[type=button],
input[type=reset],
input[type=submit],
```

```
input[type=checkbox],
input[type=radio],
textarea,
input,
input[type=text],
input[type=password],
input[type=image]
{
 display: inline-block;
}

input[type=button],
input[type=reset],
input[type=submit],
input[type=checkbox],
input[type=radio],
input,
input[type=text],
input[type=password],
input[type=image]
{
 white-space: nowrap;
}

button
{
/* white space handling of BUTTON tags in particular */
 white-space:normal;
}

input[type=reset]:lang(en)
{
/* default content of HTML input type=reset button, per language */
 content: "Reset";
}

input[type=submit]:lang(en)
{
/* default content of HTML input type=submit button, per language */
 content: "Submit";
}

/* UAs should use language-specific Reset/Submit rules for others. */
```

```
input[type=button],
input[type=reset][value],
input[type=submit][value]
{
/* text content/labels of HTML "input" buttons */
 content: attr(value);
}

textarea
{
/* white space handling of TEXTAREA tags in particular */
 white-space:pre-wrap;
 resize: both;
}

input[type=hidden]
{
/* appearance of the HTML hidden text field in particular */
 display: none !important;
}

input[type=image]
{
 content: attr(src,url);
 border: none;
}

select[size]
{
/* HTML4/XHTML1 <select> w/ size more than 1 - appearance of list */
 display: inline-block;
 height: attr(size,em);
}

select,select[size=1]
{
/* HTML4/XHTML1 <select> without size, or size=1 - popup-menu */
 display: inline-block;
 height: 1em;
 overflow: hidden;
}
```

```
select[size]:active
{
/* active HTML <select> w/ size more than 1 - appearance of active list */
 display: inline-block;
}

optgroup,option
{
 display: block;
 white-space: nowrap;
}

optgroup[label],option[label]
{
 content: attr(label);
}

option[selected]::before
{
 display: inline;
 content: check;
}

/* Though FRAME resizing is not directly addressed by this specification,
   the following rules may provide an approximation of reasonable behavior. */

/*

frame { resize:both }
frame[noresize] { resize:none }

*/
```

§ Conformance

§ Document conventions

Conformance requirements are expressed with a combination of descriptive assertions and RFC 2119 terminology. The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in the normative

parts of this document are to be interpreted as described in RFC 2119. However, for readability, these words do not appear in all uppercase letters in this specification.

All of the text of this specification is normative except sections explicitly marked as non-normative, examples, and notes. [RFC2119]

Examples in this specification are introduced with the words “for example” or are set apart from the normative text with `class="example"`, like this:

> EXAMPLE 12
>
> This is an example of an informative example.

Informative notes begin with the word “Note” and are set apart from the normative text with `class="note"`, like this:

> Note, this is an informative note.

Advisements are normative sections styled to evoke special attention and are set apart from other normative text with `<strong class="advisement">`, like this:

> **UAs MUST provide an accessible alternative.**

§ Conformance classes

Conformance to this specification is defined for three conformance classes:

style sheet
: A CSS style sheet.

renderer
: A UA that interprets the semantics of a style sheet and renders documents that use them.

authoring tool
: A UA that writes a style sheet.

A style sheet is conformant to this specification if all of its statements that use syntax defined in this module are valid according to the generic CSS grammar and the individual grammars of each feature defined in this module.

A renderer is conformant to this specification if, in addition to interpreting the style sheet as defined by the appropriate specifications, it supports all the features defined by this specification by parsing them correctly and rendering the document accordingly. However, the inability of a UA to correctly render a document due to limitations of the device does not make the UA non-conformant. (For example, a UA is not required to render color on a monochrome monitor.)

An authoring tool is conformant to this specification if it writes style sheets that are syntactically correct according to the generic CSS grammar and the individual grammars of each feature in this module, and meet all other conformance requirements of style sheets as described in this module.

§ Requirements for Responsible Implementation of CSS

The following sections define several conformance requirements for implementing CSS responsibly, in a way that promotes interoperability in the present and future.

§ Partial Implementations

So that authors can exploit the forward-compatible parsing rules to assign fallback values, **CSS renderers *must* treat as invalid (and ignore as appropriate) any at-rules, properties, property values, keywords, and other syntactic constructs for which they have no usable level of support**. In particular, user agents *must not* selectively ignore unsupported property values and honor supported values in a single multi-value property declaration: if any value is considered invalid (as unsupported values must be), CSS requires that the entire declaration be ignored.

§ Implementations of Unstable and Proprietary Features

To avoid clashes with future stable CSS features, the CSSWG recommends following best practices for the implementation of unstable features and proprietary extensions to CSS.

§ Implementations of CR-level Features

Once a specification reaches the Candidate Recommendation stage, implementers should release an unprefixed implementation of any CR-level feature they can demonstrate to be correctly implemented according to spec, and should avoid exposing a prefixed variant of that feature.

To establish and maintain the interoperability of CSS across implementations, the CSS Working Group requests that non-experimental CSS renderers submit an implementation report (and, if necessary, the

testcases used for that implementation report) to the W3C before releasing an unprefixed implementation of any CSS features. Testcases submitted to W3C are subject to review and correction by the CSS Working Group.

Further information on submitting testcases and implementation reports can be found from on the CSS Working Group's website at https://www.w3.org/Style/CSS/Test/. Questions should be directed to the public-css-testsuite@w3.org mailing list.

§ Index

§ Terms defined by this specification

alias, in §6.1.1

all-scroll, in §6.1.1

auto

- value for cursor, in §6.1.1
- value for caret-color, in §6.2.1

border-box, in §3.1

box-sizing, in §3.1

caret-color, in §6.2.1

cell, in §6.1.1

clip, in §5.2

col-resize, in §6.1.1

content-box, in §3.1

context-menu, in §6.1.1

copy, in §6.1.1

crosshair, in §6.1.1

cursor, in §6.1.1

default, in §6.1.1

ellipsis, in §5.2

e-resize, in §6.1.1

ew-resize, in §6.1.1

grab, in §6.1.1

grabbing, in §6.1.1

help, in §6.1.1

invert, in §4.4

max inner height, in §3.1

max inner width, in §3.1

min inner height, in §3.1

min inner width, in §3.1

move, in §6.1.1

ne-resize, in §6.1.1

nesw-resize, in §6.1.1

no-drop, in §6.1.1

none, in §6.1.1

not-allowed, in §6.1.1

n-resize, in §6.1.1

ns-resize, in §6.1.1

nw-resize, in §6.1.1

nwse-resize, in §6.1.1

outline, in §4.1

outline-color, in §4.4

outline-offset, in §4.5

outline-style, in §4.3

outline-width, in §4.2

pointer, in §6.1.1

progress, in §6.1.1

resize, in §5.1

row-resize, in §6.1.1

se-resize, in §6.1.1

s-resize, in §6.1.1

sw-resize, in §6.1.1

text, in §6.1.1

text-overflow, in §5.2

vertical-text, in §6.1.1

wait, in §6.1.1

w-resize, in §6.1.1

zoom-in, in §6.1.1

zoom-out, in §6.1.1

§ Terms defined by reference

[css-backgrounds-3] defines the following terms:

- <line-width>
- border-radius
- none

[css-color-3] defines the following terms:

- <color>
- color

[css-values-3] defines the following terms:

- *
- ,
- <length>
- <number>
- <url>
- ?
- |
- ||

[css-writing-modes-3] defines the following terms:

- direction
- end
- left

[CSS2] defines the following terms:

- auto
- background-image
- border edge
- border-bottom-width
- border-left-width
- border-right-width
- border-style
- border-top-width
- border-width
- bottom
- content height
- content width
- display
- height
- left
- margin-left
- margin-top
- max-height
- max-width
- min-height
- min-width
- overflow
- padding-bottom
- padding-left
- padding-right
- padding-top
- right
- top
- visibility
- visible
- width

[css3-images] defines the following terms:

- <image>
- concrete object size
- default object size
- default sizing algorithm

[HTML] defines the following terms:

- canvas
- iframe
- img
- object
- picture
- video

[SVG2] defines the following terms:

- svg

§ References

§ Normative References

[CSS-BACKGROUNDS-3]
Bert Bos; Elika Etemad; Brad Kemper. CSS Backgrounds and Borders Module Level 3. 17 October 2017. CR. URL: https://www.w3.org/TR/css-backgrounds-3/

[CSS-COLOR-3]
Tantek Çelik; Chris Lilley; David Baron. CSS Color Module Level 3. 15 March 2018. PR. URL: https://www.w3.org/TR/css-color-3/

[CSS-VALUES-3]
Tab Atkins Jr.; Elika Etemad. CSS Values and Units Module Level 3. 29 September 2016. CR. URL: https://www.w3.org/TR/css-values-3/

[CSS-WRITING-MODES-3]
Elika Etemad; Koji Ishii. CSS Writing Modes Level 3. 24 May 2018. CR. URL: https://www.w3.org/TR/css-writing-modes-3/

[CSS2]
Bert Bos; et al. Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification. 7 June 2011. REC. URL: https://www.w3.org/TR/CSS2/

[CSS3-IMAGES]
Elika Etemad; Tab Atkins Jr.. CSS Image Values and Replaced Content Module Level 3. 17 April 2012. CR. URL: https://www.w3.org/TR/css3-images/

[HTML]
Anne van Kesteren; et al. HTML Standard. Living Standard. URL: https://html.spec.whatwg.org/multipage/

[PNG]
Tom Lane. Portable Network Graphics (PNG) Specification (Second Edition). 10 November 2003. REC. URL: https://www.w3.org/TR/PNG/

[RFC2119]
S. Bradner. Key words for use in RFCs to Indicate Requirement Levels. March 1997. Best Current Practice. URL: https://tools.ietf.org/html/rfc2119

[SVG11]
Erik Dahlström; et al. Scalable Vector Graphics (SVG) 1.1 (Second Edition). 16 August 2011. REC. URL: https://www.w3.org/TR/SVG11/

[SVG2]
Nikos Andronikos; et al. Scalable Vector Graphics (SVG) 2. 15 September 2016. CR. URL: https://www.w3.org/TR/SVG2/

[UAX29]
Mark Davis; Laurențiu Iancu. Unicode Text Segmentation. 13 June 2017. Unicode Standard Annex #29. URL: https://www.unicode.org/reports/tr29/tr29-31.html

§ Informative References

[CSS-CASCADE-4]

Elika Etemad; Tab Atkins Jr.. CSS Cascading and Inheritance Level 4. 14 January 2016. CR. URL: https://www.w3.org/TR/css-cascade-4/

[CSS-PSEUDO-4]

Daniel Glazman; Elika Etemad; Alan Stearns. CSS Pseudo-Elements Module Level 4. 7 June 2016. WD. URL: https://www.w3.org/TR/css-pseudo-4/

[CSS-TRANSITIONS-1]

David Baron; Dean Jackson; Brian Birtles. CSS Transitions. 30 November 2017. WD. URL: https://www.w3.org/TR/css-transitions-1/

[CSS1]

Håkon Wium Lie; Bert Bos. Cascading Style Sheets (CSS1) Level 1 Specification. 11 April 2008. REC. URL: https://www.w3.org/TR/REC-CSS1/

[CSS4-IMAGES]

Tab Atkins Jr.; Elika Etemad; Lea Verou. CSS Image Values and Replaced Content Module Level 4. 13 April 2017. WD. URL: https://www.w3.org/TR/css-images-4/

§ Property Index

Name	Value	Initial	Applies to	Inh.	%ages	Media	Anim-ation type	Canonical order	Computed value
'box-sizing'	content-box | border-box	content-box	all elements that accept width or height	no	N/A	visual	discrete	per grammar	specified value

Name	Value	Initial	Applies to	Inh.	%ages	Media	Anim-ation type	Canonical order	Computed value
'caret-color'	auto | <color>	auto	all elements	yes	N/A	interactive	color	per grammar	The computed value for auto is auto; the computed value of currentColor is currentColor (See); see the color property for other <color> values.
'cursor'	[[<url> [<x> <y>]?,]* [auto | default | none | context-menu | help | pointer | progress | wait | cell | crosshair | text | vertical-text | alias | copy | move | no-drop | not-allowed | grab | grabbing | e-resize | n-resize | ne-resize | nw-resize | s-resize | se-resize | sw-resize | w-resize | ew-resize | ns-resize | nesw-resize | nwse-resize | col-resize | row-resize | all-scroll | zoom-in | zoom-out]]	auto	all elements	yes	N/A	visual, interactive	discrete	per grammar	as specified, except with any relative URLs converted to absolute

Name	Value	Initial	Applies to	Inh.	%ages	Media	Anim-ation type	Canonical order	Computed value
'outline'	[<outline-color> || <outline-style> || <outline-width>]	see individual properties	all elements	no	N/A	visual	see individual properties	per grammar	see individual properties
'outline-color'	<color> | invert	invert	all elements	no	N/A	visual	color	per grammar	The computed value for invert is invert; the computed value of currentColor is currentColor (See); see the color property for other <color> values.
'outline-offset'	<length>	0	all elements	no	N/A	visual	length	per grammar	<length> value in absolute units (px or physical).
'outline-style'	auto | <border-style>	none	all elements	no	N/A	visual	discrete	per grammar	as specified
'outline-width'	<line-width>	medium	all elements	no	N/A	visual	length	per grammar	absolute length; 0 if the outline style is none.

Name	Value	Initial	Applies to	Inh.	%ages	Media	Anim-ation type	Canonical order	Computed value
‘resize’	none | both | horizontal | vertical	none	elements with overflow other than visible, and optionally replaced elements such as images, videos, and iframes	no	N/A	visual	discrete	per grammar	as specified
‘text-overflow’	clip | ellipsis	clip	block containers	no	N/A	visual	discrete	per grammar	as specified