CSS Speech Module

W3C Candidate Recommendation 10 March 2020

This version:
https://www.w3.org/TR/2020/CR-css-speech-1-20200310/

Latest version:
https://www.w3.org/TR/css-speech-1/

Previous versions:
https://www.w3.org/TR/2018/NOTE-css3-speech-20180605/
https://www.w3.org/TR/2012/CR-css3-speech-20120320/
https://www.w3.org/TR/2011/WD-css3-speech-20110818/

Feedback:
www-style@w3.org with subject line "`[css-speech] … message topic …`" (archives)

Editor:
Daniel Weck (DAISY Consortium)

Former editors:
Dave Raggett (W3C/Canon)
Daniel Glazman (Disruptive Innovations)
Claudio Santambrogio (Opera Software)

Abstract

CSS (Cascading Style Sheets) is a language that describes the rendering of markup documents (e.g. HTML, XML) on various supports, such as screen, paper, speech, etc. The Speech module defines aural CSS properties that enable authors to declaratively control the rendering of documents via speech synthesis, and using optional audio cues. Note that this standard was developed in cooperation with the Voice Browser Activity.

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 was produced by the CSS Working Group as a Candidate Recommendation. This document is intended to become a W3C Recommendation. This document will remain a Candidate Recommendation at least until 8 April 2020 in order to ensure the opportunity for wide review.

GitHub Issues are preferred for discussion of this specification. When filing an issue, please put the text "css-speech" in the title, preferably like this: "[css-speech] *...summary of comment...*". All issues and comments are archived, and there is also a historical archive.

Publication as a Candidate Recommendation does not imply endorsement by the W3C Membership. This is a draft document and may be updated, replaced or obsoleted by other documents at any time. It is inappropriate to cite this document as other than work in progress.

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 March 2019 W3C Process Document.

This document is based on the Candidate Recommendation (20 March 2012) and includes changes that reflect the outcome of the disposition of comments.

Before the specification can progress to Proposed Recommendation, the CR exit criteria must be met. The specification will not become Proposed Recommendation before 8 April 2020. A test suite and an implementation report will be made during the Candidate Recommendation period.

The following features are at-risk and may be dropped at the end of the Candidate Recommendation period if there has not been enough interest from implementers: '`voice-balance`', '`voice-duration`', '`voice-pitch`', '`voice-range`', and '`voice-stress`'.

Table of contents

1. Introduction, design goals

Note that this section is informative.

The aural presentation of information is commonly used by people who are blind, visually-impaired or otherwise print-disabled. For instance, "screen readers" allow users to interact with visual interfaces that would otherwise be inaccessible to them. There are also circumstances in which *listening* to content (as opposed to *reading*) is preferred, or sometimes even required, irrespective of a person's physical ability to access information. For instance: playing an e-book whilst driving a vehicle, learning how to manipulate industrial and medical devices, interacting with home entertainment systems, teaching young children how to read.

The CSS properties defined in the Speech module enable authors to declaratively control the presentation of a document in the aural dimension. The aural rendering of a document combines speech synthesis (also known as "TTS", the acronym for "Text to Speech") and auditory icons (which are referred-to as "audio cues" in this specification). The CSS Speech properties provide the ability to control speech pitch and rate, sound levels, TTS voices, etc. These stylesheet properties can be used together with visual properties (mixed media), or as a complete aural alternative to a visual presentation.

2. Background information, CSS 2.1

Note that this section is informative.

The CSS Speech module is a re-work of the informative CSS2.1 Aural appendix, within which the "aural" media type was described, but also deprecated (in favor of the "speech" media type). Although the [CSS21] specification reserves the "speech" media type, it doesn't actually define the corresponding properties. The Speech module describes the CSS properties that apply to the "speech" media type, and defines a new "box" model specifically for the aural dimension.

Content creators can conditionally include CSS properties dedicated to user agents with text to speech synthesis capabilities, by specifying the "speech" media type via the `media` attribute of the `link` element, or with the `@media` at-rule, or within an `@import` statement. When styles are authored within the scope of such conditional statements, they are ignored by user agents that do not support the Speech module.

3. Relationship with SSML

Note that this section is informative.

Some of the features in this specification are conceptually similar to functionality described in the Speech Synthesis Markup Language (SSML) Version 1.1 [SSML]. However, the specificities of the CSS model mean that compatibility with SSML in terms of syntax and/or semantics is only partially achievable. The definition of each property in the Speech module includes informative statements, wherever necessary, to clarify their relationship with similar functionality from SSML.

4. CSS values

This specification follows the CSS property definition conventions from [CSS2]. Value types not defined in this specification are defined in CSS Values & Units [CSS-VALUES-3]. Other CSS modules may expand the definitions of these value types.

In addition to the property-specific values listed in their definitions, all properties defined in this specification also accept the CSS-wide keywords keywords as their property value. For readability they have not been repeated explicitly.

5. Example

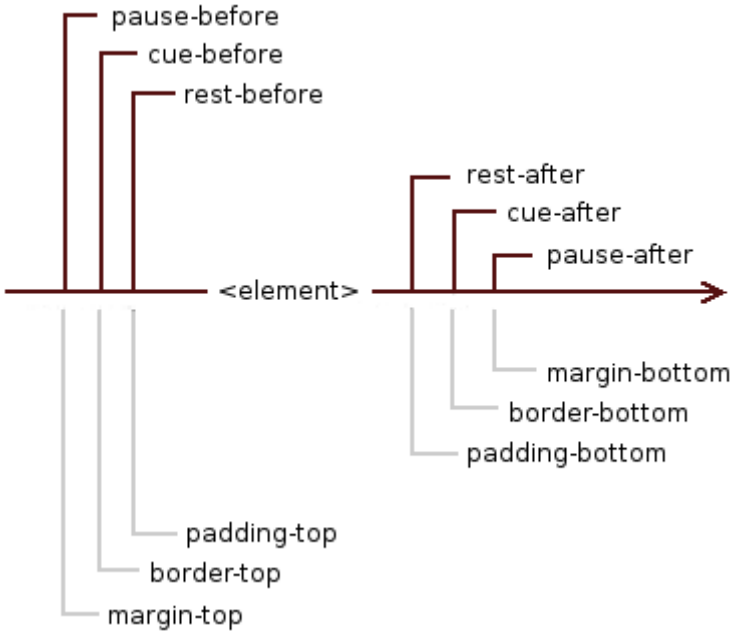
This example shows how authors can tell the speech synthesizer to speak HTML headings with a voice called "paul", using "moderate" emphasis (which is more than normal) and how to insert an audio cue (pre-recorded audio clip located at the given URL) before the start of TTS rendering for each heading. In a stereo-capable sound system, paragraphs marked with the CSS class "heidi" are rendered on the left audio channel (and with a female voice, etc.), whilst the class "peter" corresponds to the right channel (and to a male voice, etc.). The volume level of text spans marked with the class "special" is lower than normal, and a prosodic boundary is created by introducing a strong pause after it is spoken (note how the `span` inherits the voice-family from its parent paragraph).

```
h1, h2, h3, h4, h5, h6
{
  voice-family: paul;
  voice-stress: moderate;
  cue-before: url(../audio/ping.wav);
  voice-volume: medium 6dB;
}
p.heidi
{
  voice-family: female;
  voice-balance: left;
  voice-pitch: high;
  voice-volume: -6dB;
}
p.peter
{
  voice-family: male;
  voice-balance: right;
  voice-rate: fast;
}
span.special
{
  voice-volume: soft;
  pause-after: strong;
}

...

<h1>I am Paul, and I speak headings.</h1>
<p class="heidi">Hello, I am Heidi.</p>
<p class="peter">
  <span class="special">Can you hear me ?</span>
  I am Peter.
</p>
```

6. The aural formatting model

The CSS formatting model for aural media is based on a sequence of sounds and silences that occur within a nested context similar to the visual box model, which we name the ***aural "box" model***. The aural "canvas" consists of a two-channel (stereo) space and of a temporal dimension, within which synthetic speech and audio cues coexist. The selected element is surrounded by '`rest`', '`cue`' and '`pause`' properties (from the innermost to the outermost position). These can be seen as aural equivalents to '`padding`', '`border`' and '`margin`', respectively. When used, the '`:before`' and '`:after`' pseudo-elements [CSS21] get inserted between the element's contents and the '`rest`'.

The following diagram illustrates the equivalence between properties of the visual and aural box models, applied to the selected <element>:



7. Mixing properties

7.1. The '`voice-volume`' property

Name:	***voice-volume***
Value:	silent | [[x-soft | soft | medium | loud | x-loud] || <decibel>]
Initial:	medium
Applies to:	all elements
Inherited:	yes
Percentages:	N/A
Computed value:	'`silent`', or a keyword value and optionally also a decibel offset (if not zero)

The '`voice-volume`' property allows authors to control the amplitude of the audio waveform generated by the speech synthesiser, and is also used to adjust the relative volume level of audio cues within the aural box model of the selected element.

Note that although the functionality provided by this property is similar to the `volume` attribute of the `prosody` element from the SSML markup language [SSML], there are notable discrepancies. For example, CSS Speech volume keywords and decibels units are not mutually-exclusive, due to how values are inherited and combined for selected elements.

silent
Specifies that no sound is generated (the text is read "silently").

Note that this has the same effect as using negative infinity decibels. Also note that there is a difference between an element whose '`voice-volume`' property has a value of '`silent`', and an element whose '`speak`' property has the value '`none`'. With the former, the selected element takes up the same time as if it was spoken, including any pause before and after the element, but no sound is generated (descendants within the aural box model of the selected element can override the '`voice-volume`' value, and may therefore generate audio output). With the latter, the selected element is not rendered in the aural dimension and no time is allocated for playback (descendants within the aural box model of the selected element can override the '`speak`' value, and may therefore generate audio output).

x-soft, soft, medium, loud, x-loud
This sequence of keywords corresponds to monotonically non-decreasing volume levels, mapped to implementation-dependent values that meet the listener's requirements with regards

to perceived loudness. These audio levels are typically provided via a preference mechanism that allow users to calibrate sound options according to their auditory environment. The keyword '`x-soft`' maps to the user's *minimum audible* volume level, '`x-loud`' maps to the user's *maximum tolerable* volume level, '`medium`' maps to the user's *preferred* volume level, '`soft`' and '`loud`' map to intermediary values.

<decibel>

A number immediately followed by "dB" (decibel unit). This represents a change (positive or negative) relative to the given keyword value (see enumeration above), or to the default value for the root element, or otherwise to the inherited volume level (which may itself be a combination of a keyword value and of a decibel offset, in which case the decibel values are combined additively). When the inherited volume level is '`silent`', this '`voice-volume`' resolves to '`silent`' too, regardless of the specified <decibel> value. Decibels represent the ratio of the squares of the new signal amplitude (a1) and the current amplitude (a0), as per the following logarithmic equation: volume(dB) = 20 log10 (a1 / a0)

> Note that -6.0dB is approximately half the amplitude of the audio signal, and +6.0dB is approximately twice the amplitude.

> Note that perceived loudness depends on various factors, such as the listening environment, user preferences or physical abilities. The effective volume variation between '`x-soft`' and '`x-loud`' represents the dynamic range (in terms of loudness) of the audio output. Typically, this range would be compressed in a noisy context, i.e. the perceived loudness corresponding to '`x-soft`' would effectively be closer to '`x-loud`' than it would be in a quiet environment. There may also be situations where both '`x-soft`' and '`x-loud`' would map to low volume levels, such as in listening environments requiring discretion (e.g. library, night-reading).

7.2. The '`voice-balance`' property

Name:	***voice-balance***
Value:	<number> | left | center | right | leftwards | rightwards
Initial:	center
Applies to:	all elements
Inherited:	yes
Percentages:	N/A
Computed value:	the specified value resolved to a <number> between '`-100`' and '`100`' (inclusive)

The '`voice-balance`' property controls the spatial distribution of audio output across a lateral sound stage: one extremity is on the left, the other extremity is on the right hand side, relative to the listener's position. Authors can specify intermediary steps between left and right extremities, to represent the audio separation along the resulting left-right axis.

Note that the functionality provided by this property has no match in the SSML markup language [SSML].

<number>
: A number between '`-100`' and '`100`' (inclusive). Values smaller than '`-100`' are clamped to '`-100`'. Values greater than '`100`' are clamped to '`100`'. The value '`-100`' represents the left side, and the value '`100`' represents the right side. The value '`0`' represents the center point whereby there is no discernible audio separation between left and right sides (in a stereo sound system, this corresponds to equal distribution of audio signals between left and right speakers).

left
: Same as '`-100`'.

center
: Same as '`0`'.

right
: Same as '`100`'.

leftwards

Moves the sound to the left, by subtracting 20 from the inherited '`voice-balance`' value, and by clamping the resulting number to '`-100`'.

rightwards
Moves the sound to the right, by adding 20 to the inherited '`voice-balance`' value, and by clamping the resulting number to '`100`'.

user agents may be connected to different kinds of sound systems, featuring varying audio mixing capabilities. The expected behavior for mono, stereo, and surround sound systems is defined as follows:

- When user agents produce audio via a mono-aural sound system (i.e. single-speaker setup), the '`voice-balance`' property has no effect.
- When user agents produce audio through a stereo sound system (e.g. two speakers, a pair of headphones), the left-right distribution of audio signals can precisely match the authored values for the '`voice-balance`' property.
- When user agents are capable of mixing audio signals through more than 2 channels (e.g. 5-speakers surround sound system, including a dedicated center channel), the physical distribution of audio signals resulting from the application of the '`voice-balance`' property should be performed so that the listener perceives sound as if it was coming from a basic stereo layout. For example, the center channel as well as the left/right speakers may be used altogether in order to emulate the behavior of the '`center`' value.

Future revisions of the CSS Speech module may include support for three-dimensional audio, which would effectively enable authors to specify "azimuth" and "elevation" values. In the future, content authored using the current specification may therefore be consumed by user agents which are compliant with the version of CSS Speech that supports three-dimensional audio. In order to prepare for this possibility, the values enabled by the current '`voice-balance`' property are designed to remain compatible with "azimuth" angles. More precisely, the mapping between the current left-right audio axis (lateral sound stage) and the envisioned 360 degrees plane around the listener's position is defined as follows:

- The value '`0`' maps to zero degrees ('`center`'). This is in "front" of the listener, not from "behind".
- The value '`-100`' maps to -40 degrees ('`left`'). Negative angles are in the counter-clockwise direction (the audio stage is seen from the top).
- The value '`100`' maps to 40 degrees ('`right`'). Positive angles are in the clockwise direction (the audio stage is seen from the top).

- Intermediary values on the scale from '`-100`' to '`100`' map to the angles between -40 and 40 degrees in a numerically linearly-proportional manner. For example, '`-50`' maps to -20 degrees.

Note that sound systems may be configured by users in such a way that it would interfere with the left-right audio distribution specified by document authors. Typically, the various "surround" modes available in modern sound systems (including systems based on basic stereo speakers) tend to greatly alter the perceived spatial arrangement of audio signals. The illusion of a three-dimensional sound stage is often achieved using a combination of phase shifting, digital delay, volume control (channel mixing), and other techniques. Some users may even configure their system to "downgrade" any rendered sound to a single mono channel, in which case the effect of the '`voice-balance`' property would obviously not be perceivable at all. The rendering fidelity of authored content is therefore dependent on such user customizations, and the '`voice-balance`' property merely specifies the desired end-result.

Note that many speech synthesizers only generate mono sound, and therefore do not intrinsically support the '`voice-balance`' property. The sound distribution along the left-right axis consequently occurs at post-synthesis stage (when the speech-enabled user agent mixes the various audio sources authored within the document)

8. Speaking properties

8.1. The '`speak`' property

Name:	***speak***
Value:	auto | never | always
Initial:	auto
Applies to:	all elements
Inherited:	yes
Percentages:	N/A
Computed value:	specified value

The '`speak`' property determines whether or not to render text aurally.

> Note that the functionality provided by this property has no match in the SSML markup language [SSML].

auto
Resolves to a computed value of ‘`never`’ when ‘`display`’ is ‘`none`’, otherwise resolves to a computed value of ‘`auto`’. The used value is then equivalent to ‘`always`’ if ‘`visibility`’ is ‘`visible`’ and to ‘`never`’ otherwise.

> Note that the ‘`none`’ value of the ‘`display`’ property cannot be overridden by descendants of the selected element, but the ‘`auto`’ value of ‘`speak`’ can however be overridden using either of ‘`never`’ or ‘`always`’.

never
This value causes an element (including pauses, cues, rests and actual content) to not be rendered (i.e., the element has no effect in the aural dimension).

> Note that any of the descendants of the affected element are allowed to override this value, so descendants can actually take part in the aural rendering despite using ‘`display: none`’ at this level. However, the pauses, cues, and rests of the ancestor element remain "deactivated" in the aural dimension, and therefore do not contribute to the collapsing of pauses or additive behavior of adjoining rests.

always
The element is rendered aurally (regardless of its ‘`display`’ value, or the ‘`display`’ or ‘`speak`’ values of its ancestors).

> Note that using this value can result in the element being rendered in the aural dimension even though it would not be rendered on the visual canvas.

8.2. The ‘`speak-as`’ property

Name:	***speak-as***
Value:	normal | spell-out || digits || [literal-punctuation | no-punctuation]
Initial:	normal
Applies to:	all elements
Inherited:	yes
Percentages:	N/A
Computed value:	specified value

The '`speak-as`' property determines in what manner text gets rendered aurally, based upon a predefined list of possibilities.

Note that the functionality provided by this property is conceptually similar to the `say-as` element from the SSML markup language [SSML] (whose possible values are described in the [SSML-SAYAS] W3C Note). Although the design goals are similar, the CSS model is limited to a basic set of pronunciation rules.

normal
: Uses language-dependent pronunciation rules for rendering the element's content. For example, punctuation is not spoken as-is, but instead rendered naturally as appropriate pauses.

spell-out
: Spells the text one letter at a time (useful for acronyms and abbreviations). In languages where accented characters are rare, it is permitted to drop accents in favor of alternative unaccented spellings. As as example, in English, the word "rôle" can also be written as "role". A conforming implementation would thus be able to spell-out "rôle" as "R O L E".

digits
: Speak numbers one digit at a time, for instance, "twelve" would be spoken as "one two", and "31" as "three one".

Speech synthesizers are knowledgeable about what a *number* is. The '`speak-as`' property enables some level of control on how user agents render numbers, and may be implemented as a preprocessing step before passing the text to the actual speech synthesizer.

literal-punctuation
Punctuation such as semicolons, braces, and so on is named aloud (i.e. spoken literally) rather than rendered naturally as appropriate pauses.

no-punctuation
Punctuation is not rendered: neither spoken nor rendered as pauses.

9. Pause properties

9.1. The '`pause-before`' and '`pause-after`' properties

Name:	***pause-before***
Value:	<time> | none | x-weak | weak | medium | strong | x-strong
Initial:	none
Applies to:	all elements
Inherited:	no
Percentages:	N/A
Computed value:	specified value

Name:	***pause-after***
Value:	<time> | none | x-weak | weak | medium | strong | x-strong
Initial:	none
Applies to:	all elements
Inherited:	no
Percentages:	N/A
Computed value:	specified value

The ‘pause-before’ and ‘pause-after’ properties specify a prosodic boundary (silence with a specific duration) that occurs before (or after) the speech synthesis rendition of the selected element, or if any ‘cue-before’ (or ‘cue-after’) is specified, before (or after) the cue within the aural box model.

Note that although the functionality provided by this property is similar to the break element from the SSML markup language [SSML], the application of ‘pause’ prosodic boundaries within the aural box model of CSS Speech requires special considerations (e.g. "collapsed" pauses).

<time>
: Expresses the pause in absolute time units (seconds and milliseconds, e.g. "+3s", "250ms"). Only non-negative values are allowed.

none
: Equivalent to 0ms (no prosodic break is produced by the speech processor).

x-weak, weak, medium, strong, and x-strong
: Expresses the pause by the strength of the prosodic break in speech output. The exact time is implementation-dependent. The values indicate monotonically non-decreasing (conceptually increasing) break strength between elements.

Note that stronger content boundaries are typically accompanied by pauses. For example, the breaks between paragraphs are typically much more substantial than the breaks between words within a sentence.

This example illustrates how the default strengths of prosodic breaks for specific elements (which are defined by the user agent stylesheet) can be overridden by authored styles.

```
p { pause: none } /* pause-before: none; pause-after: none */
```

9.2. The ‘pause’ shorthand property

Name:	***pause***
Value:	<‘pause-before’> <‘pause-after’>?
Initial:	N/A (see individual properties)
Applies to:	all elements
Inherited:	no
Percentages:	N/A
Computed value:	N/A (see individual properties)

The ‘pause’ property is a shorthand property for ‘pause-before’ and ‘pause-after’. If two values are given, the first value is ‘pause-before’ and the second is ‘pause-after’. If only one value is given, it applies to both properties.

Examples of property values:

```
h1 { pause: 20ms; } /* pause-before: 20ms; pause-after: 20ms */
h2 { pause: 30ms 40ms; } /* pause-before: 30ms; pause-after: 40ms */
h3 { pause-after: 10ms; } /* pause-before: unspecified; pause-after: 10ms */
```

9.3. Collapsing pauses

The pause defines the minimum distance of the aural "box" to the aural "boxes" before and after it. Adjoining pauses are merged by selecting the strongest named break and the longest absolute time interval. For example, "strong" is selected when comparing "strong" and "weak", "1s" is selected when comparing "1s" and "250ms", and "strong" and "250ms" take effect additively when comparing "strong" and "250ms".

The following pauses are adjoining:

1. The ‘pause-after’ of an aural "box" and the ‘pause-after’ of its last child, provided the former has no ‘rest-after’ and no ‘cue-after’.
2. The ‘pause-before’ of an aural "box" and the ‘pause-before’ of its first child, provided the former has no ‘rest-before’ and no ‘cue-before’.

3. The ‘pause-after’ of an aural "box" and the ‘pause-before’ of its next sibling.
4. The ‘pause-before’ and ‘pause-after’ of an aural "box", if the the "box" has a ‘voice-duration’ of "0ms" and no ‘rest-before’ or ‘rest-after’ and no ‘cue-before’ or ‘cue-after’, or if the the "box" has no rendered content at all (see ‘speak’).

A collapsed pause is considered adjoining to another pause if any of its component pauses is adjoining to that pause.

Note that ‘pause’ has been moved from between the element's contents and any ‘cue’ to outside the ‘cue’. This is not backwards compatible with the informative CSS2.1 Aural appendix [CSS21].

10. Rest properties

10.1. The ‘rest-before’ and ‘rest-after’ properties

Name:	***rest-before***
Value:	<time> | none | x-weak | weak | medium | strong | x-strong
Initial:	none
Applies to:	all elements
Inherited:	no
Percentages:	N/A
Computed value:	specified value

Name:	***rest-after***
Value:	<time> | none | x-weak | weak | medium | strong | x-strong
Initial:	none
Applies to:	all elements
Inherited:	no
Percentages:	N/A
Computed value:	specified value

The '`rest-before`' and '`rest-after`' properties specify a prosodic boundary (silence with a specific duration) that occurs before (or after) the speech synthesis rendition of an element within the aural box model.

Note that although the functionality provided by this property is similar to the `break` element from the SSML markup language [SSML], the application of '`rest`' prosodic boundaries within the aural box model of CSS Speech requires special considerations (e.g. interspersed audio cues, additive adjacent rests).

<time>
: Expresses the rest in absolute time units (seconds and milliseconds, e.g. "+3s", "250ms"). Only non-negative values are allowed.

none
: Equivalent to 0ms (no prosodic break is produced by the speech processor).

x-weak, weak, medium, strong, and x-strong
: Expresses the rest by the strength of the prosodic break in speech output. The exact time is implementation-dependent. The values indicate monotonically non-decreasing (conceptually increasing) break strength between elements.

As opposed to pause properties, the rest is inserted between the element's content and any '`cue-before`' or '`cue-after`' content. Adjoining rests are treated additively, and do not collapse.

10.2. The '`rest`' shorthand property

Name:	***rest***
Value:	<‘rest-before’> <‘rest-after’>?
Initial:	N/A (see individual properties)
Applies to:	all elements
Inherited:	no
Percentages:	N/A
Computed value:	N/A (see individual properties)

The ‘rest’ property is a shorthand for ‘rest-before’ and ‘rest-after’. If two values are given, the first value is ‘rest-before’ and the second is ‘rest-after’. If only one value is given, it applies to both properties.

11. Cue properties

11.1. The ‘cue-before’ and ‘cue-after’ properties

Name:	***cue-before***
Value:	<uri> <decibel>? | none
Initial:	none
Applies to:	all elements
Inherited:	no
Percentages:	N/A
Computed value:	specified value

Name:	***cue-after***
Value:	<uri> <decibel>? | none
Initial:	none
Applies to:	all elements
Inherited:	no
Percentages:	N/A
Computed value:	specified value

The '`cue-before`' and '`cue-after`' properties specify auditory icons (i.e. pre-recorded / pre-generated sound clips) to be played before (or after) the selected element within the aural box model.

Note that although the functionality provided by this property may appear related to the `audio` element from the SSML markup language [SSML], there are in fact major discrepancies. For example, the aural box model means that audio cues are associated to the selected element's volume level, and CSS Speech's auditory icons provide limited functionality compared to SSML's `audio` element.

<uri>
The URI designates an auditory icon resource. When a user agent is not able to render the specified auditory icon (e.g. missing file resource, or unsupported audio codec), it is recommended to produce an alternative cue, such as a bell sound.

none
Specifies that no auditory icon is used.

<decibel>
A number immediately followed by "dB" (decibel unit). This represents a change (positive or negative) relative to the computed value of the '`voice-volume`' property within the aural box model of the selected element (as a result, the volume level of an audio cue changes when the '`voice-volume`' property changes). When omitted, the implied value computes to 0dB.

When the computed value of the '`voice-volume`' property is '`silent`', the audio cue is also set to '`silent`' (regardless of this specified <decibel> value). Otherwise (when not '`silent`'), '`voice-volume`' values are always specified relatively to the volume level keywords (see the

definition of ‘voice-volume’), which map to a user-calibrated scale of "preferred" loudness settings. If the inherited ‘voice-volume’ value already contains a decibel offset, the dB offset specific to the audio cue is combined additively.

Decibels express the ratio of the squares of the new signal amplitude (a1) and the current amplitude (a0), as per the following logarithmic equation: volume(dB) = 20 log10 (a1 / a0)

Note that -6.0dB is approximately half the amplitude of the audio signal, and +6.0dB is approximately twice the amplitude.

Note that there is a difference between an audio cue whose volume is set to ‘silent’ and one whose value is ‘none’. In the former case, the audio cue takes up the same time as if it had been played, but no sound is generated. In the latter case, the there is no manifestation of the audio cue at all (i.e. no time is allocated for the cue in the aural dimension).

Examples of property values:

```
a
{
  cue-before: url(/audio/bell.aiff) -3dB;
  cue-after: url(dong.wav);
}

h1
{
  cue-before: url(../clips-1/pop.au) +6dB;
  cue-after: url(../clips-2/pop.au) 6dB;
}

div.caution { cue-before: url(./audio/caution.wav) +8dB; }
```

11.2. Relation between audio cues and speech synthesis volume levels

Note that this section is informative.

The volume levels of audio cues and of speech synthesis within the aural box model of a selected element are related. For example, the desired effect of an audio cue whose volume level is set at +0dB (as specified by the <decibel> value) is that its perceived loudness during playback is close to that of the speech synthesis rendition of the selected element, as dictated by the computed value

of the ‘voice-volume’ property. Note that a ‘silent’ computed value for the ‘voice-volume’ property results in audio cues being "forcefully" silenced as well (i.e. regardless of the specified audio cue ‘decibel’ value)

The volume keywords of the ‘voice-volume’ property are user-calibrated to match requirements not known at authoring time (e.g. auditory environment, personal preferences). Therefore, in order to achieve this approximate loudness alignment of audio cues and speech synthesis, authors should ensure that the volume level of audio cues (on average, as there may be discrete variations of perceived loudness due to changes in the audio stream, such as intonation, stress, etc.) matches the output of a speech synthesis rendition based on the ‘voice-family’ intended for use, given "typical" listening conditions (i.e. default system volume levels, centered equalization across the frequency spectrum). As speech processors are capable of directly controlling the waveform amplitude of generated text-to-speech audio, and because user agents are able to adjust the volume output of audio cues (i.e. amplify or attenuate audio signals based on the intrinsic waveform amplitude of digitized sound clips), this sets a baseline that enables implementations to manage the loudness of both TTS and cue audio streams within the aural box model, relative to user-calibrated volume levels (see the keywords defined in the ‘voice-volume’ property).

Due to the complex relationship between perceived audio characteristics (e.g. loudness) and the processing applied to the digitized audio signal (e.g. signal compression), we refer to a simple scenario whereby the attenuation is indicated in decibels, typically ranging from 0dB (i.e. maximum audio input, near clipping threshold) to -60dB (i.e. total silence). Given this context, a "standard" audio clip would oscillate between these values, the loudest peak levels would be close to -3dB (to avoid distortion), and the relevant audible passages would have average (RMS) volume levels as high as possible (i.e. not too quiet, to avoid background noise during amplification). This would roughly provide an audio experience that could be seamlessly combined with text-to-speech output (i.e. there would be no discernible difference in volume levels when switching from pre-recorded audio to speech synthesis). Although there exists no industry-wide standard to support such convention, different TTS engines tend to generate comparably-loud audio signals when no gain or attenuation is specified. For voice and soft music, -15dB RMS seems to be pretty standard.

11.3. The ‘cue’ shorthand property

Name:	***cue***
Value:	<‘cue-before’> <‘cue-after’>?
Initial:	N/A (see individual properties)
Applies to:	all elements
Inherited:	no
Percentages:	N/A
Computed value:	N/A (see individual properties)

The ‘cue’ property is a shorthand for ‘cue-before’ and ‘cue-after’. If two values are given the first value is ‘cue-before’ and the second is ‘cue-after’. If only one value is given, it applies to both properties.

Example of shorthand notation:

```
h1
{
  cue-before: url(pop.au);
  cue-after: url(pop.au);
}
/* ...is equivalent to: */
h1
{
  cue: url(pop.au);
}
```

12. Voice characteristic properties

12.1. The ‘voice-family’ property

Name:	***voice-family***
Value:	[[<name> | <generic-voice>],]* [<name> | <generic-voice>] | preserve
Initial:	implementation-dependent
Applies to:	all elements
Inherited:	yes
Percentages:	N/A
Computed value:	specified value

The '`voice-family`' property specifies a prioritized list of component values that are separated by commas to indicate that they are alternatives (this is analogous to '`font-family`' in visual style sheets). Each component value potentially designates a speech synthesis voice instance, by specifying match criteria (see the voice selection section on this topic).

<generic-voice> = [<age>? <gender> <integer>?]

Note that although the functionality provided by this property is similar to the `voice` element from the SSML markup language [SSML], CSS Speech does not provide an equivalent to SSML's sophisticated voice language selection. This technical limitation may be alleviated in a future revision of the Speech module.

<name>

Values are specific voice instances (e.g., Mike, comedian, mary, carlos2, "valley girl"). Voice names must either be given quoted as strings, or unquoted as a sequence of one or more identifiers.

Note that as a result, most punctuation characters, or digits at the start of each token, must be escaped in unquoted voice names.

If a sequence of identifiers is given as a voice name, the computed value is the name converted to a string by joining all the identifiers in the sequence by single spaces.

Voice names that happen to be the same as the gender keywords ('`male`', '`female`' and '`neutral`') or that happen to match the keywords '`inherit`' or '`preserve`' must be quoted to

disambiguate with these keywords. The keywords '`initial`' and '`default`' are reserved for future use and must also be quoted when used as voice names.

> Note that in [SSML], voice names are space-separated and cannot contain whitespace characters.

It is recommended to quote voice names that contain white space, digits, or punctuation characters other than hyphens - even if these voice names are valid in unquoted form - in order to improve code clarity. For example: `voice-family: "john doe", "Henry the-8th";`

<age>
Possible values are '`child`', '`young`' and '`old`', indicating the preferred age category to match during voice selection.

> Note that a recommended mapping with [SSML] ages is: '`child`' = 6 y/o, '`young`' = 24 y/o, '`old`' = 75 y/o. More flexible age ranges may be used by the processor-dependent voice-matching algorithm.

<gender>
One of the keywords '`male`', '`female`', or '`neutral`', specifying a male, female, or neutral voice, respectively.

> Note that the interpretation of the relationship between a person's age or gender, and a recognizable type of voice, cannot realistically be defined in a universal manner as it effectively depends on numerous criteria (cultural, linguistic, biological, etc.). The functionality provided by this specification therefore represent a simplified model that can be reasonably applied to a broad variety of speech contexts, albeit at the cost of a certain degree of approximation. Future versions of this specification may refine the level of precision of the voice-matching algorithm, as speech processor implementations become more standardized.

<integer>
An integer indicating the preferred variant (e.g. "the second male child voice"). Only positive integers (i.e. excluding zero) are allowed. The value "1" refers to the first of all matching voices.

preserve
Indicates that the '`voice-family`' value gets inherited and used regardless of any potential language change within the content markup (see the section below about voice selection and

language handling). This value behaves as ‘`inherit`’ when applied to the root element.

> Note that descendants of the selected element automatically inherit the ‘`preserve`’ value, unless it is explicitly overridden by other ‘`voice-family`’ values (e.g. name, gender, age).

Examples of invalid declarations:

```
voice-family: john/doe; /* forward slash character should be escaped */
voice-family: john "doe"; /* identifier sequence cannot contain strings */
voice-family: john!; /* exclamation mark should be escaped */
voice-family: john@doe; /* "at" character should be escaped */
voice-family: #john; /* identifier cannot start with hash character */
voice-family: john 1st; /* identifier cannot start with digit */
```

12.1.1. Voice selection, content language

The ‘`voice-family`’ property is used to guide the selection of the speech synthesis voice instance. As part of this selection process, speech-capable user agents must also take into account the language of the selected element within the markup content. The "name", "gender", "age", and preferred "variant" (index) are voice selection hints that get carried down the content hierarchy as the ‘`voice-family`’ property value gets inherited by descendant elements. At any point within the content structure, the language takes precedence (i.e. has a higher priority) over the specified CSS voice characteristics.

The following list outlines the voice selection algorithm (note that the definition of "language" is loose here, in order to cater for dialectic variations):

1. If only a single voice instance is available for the language of the selected content, then this voice must be used, regardless of the specified CSS voice characteristics.
2. If several voice instances are available for the language of the selected content, then the chosen voice is the one that most closely matches the specified name, or gender, age, and preferred voice variant. The actual definition of "best match" is processor-dependent. For example, in a system that only has male and female adult voices available, a reasonable match for "voice-family: young male" may well be a higher-pitched female voice, as this tone of voice would be close to that of a young boy. If no voice instance matches the characteristics provided by any of the ‘`voice-family`’ component values, the first available voice instance (amongst those suitable for the language of the selected content) must be used.

3. If no voice is available for the language of the selected content, it is recommended that user agents let the user know about the lack of appropriate TTS voice.

The speech synthesizer voice must be re-evaluated (i.e. the selection process must take place once again) whenever any of the CSS voice characteristics change within the content flow. The voice must also be re-calculated whenever the content language changes, unless the '`preserve`' keyword is used (this may be useful in cases where embedded foreign language text can be spoken using a voice not designed for this language, as demonstrated by the example below).

Note that dynamically computing a voice may lead to unexpected lag, so user agents should try to resolve concrete voice instances in the document tree before the playback starts.

Examples of property values:

```
h1 { voice-family: announcer, old male; }
p.romeo  { voice-family: romeo, young male; }
p.juliet { voice-family: juliet, young female; }
p.mercutio { voice-family: young male; }
p.tybalt { voice-family: young male; }
p.nurse { voice-family: amelie; }

...

<p class="romeo" xml:lang="en-US">
  The French text below will be spoken with an English voice:
  <span style="voice-family: preserve;" xml:lang="fr-FR">Bonjour monsieur !</s

  The English text below will be spoken with a voice different
  than that corresponding to the class "romeo"
  (which is inherited from the "p" parent element):
  <span style="voice-family: female;">Hello sir!</span>
</p>
```

12.2. The '`voice-rate`' property

Name:	***voice-rate***
Value:	[normal | x-slow | slow | medium | fast | x-fast] || <percentage>
Initial:	normal
Applies to:	all elements
Inherited:	yes
Percentages:	refer to default value
Computed value:	a keyword value, and optionally also a percentage relative to the keyword (if not 100%)

The '`voice-rate`' property manipulates the rate of generated synthetic speech in terms of words per minute.

Note that although the functionality provided by this property is similar to the `rate` attribute of the `prosody` element from the SSML markup language [SSML], there are notable discrepancies. For example, CSS Speech rate keywords and percentage modifiers are not mutually-exclusive, due to how values are inherited and combined for selected elements.

normal

Represents the default rate produced by the speech synthesizer for the currently active voice. This is processor-specific and depends on the language, dialect and on the "personality" of the voice.

x-slow, slow, medium, fast and x-fast

A sequence of monotonically non-decreasing speaking rates that are implementation and voice -specific. For example, typical values for the English language are (in words per minute) x-slow = 80, slow = 120, medium = between 180 and 200, fast = 500.

<percentage>

Only non-negative percentage values are allowed. This represents a change relative to the given keyword value (see enumeration above), or to the default value for the root element, or otherwise to the inherited speaking rate (which may itself be a combination of a keyword value and of a percentage, in which case percentages are combined multiplicatively). For example, 50% means that the speaking rate gets multiplied by 0.5 (half the value). Percentages above 100% result in faster speaking rates (relative to the base keyword), whereas percentages below 100% result in slower speaking rates.

Examples of inherited values:

```
<body>
  <e1>
    <e2>
      <e3>
        ...
      </e3>
    </e2>
  </e1>
</body>
```

```
body { voice-rate: inherit; } /* the initial value is 'normal'
                                 (the actual speaking rate value
                                 depends on the active voice) */

e1 { voice-rate: +50%; } /* the computed value is
                          ['normal' and 50%], which will resolve
                          to the rate corresponding to 'normal'
                          multiplied by 0.5 (half the speaking rate) */

e2 { voice-rate: fast 120%; } /* the computed value is
                          ['fast' and 120%], which will resolve
                          to the rate corresponding to 'fast'
                          multiplied by 1.2 */

e3 { voice-rate: normal; /* "resets" the speaking rate to the intrinsic voice
                            the computed value is 'normal' (see comment below

     voice-family: "another-voice"; } /* because the voice is different,
                                         the calculated speaking rate may vary
                                         compared to "body" (even though the comp
                                         'voice-rate' value is the same) */
```

12.3. The ‘voice-pitch’ property

Name:	***voice-pitch***
Value:	<frequency> && absolute | [[x-low | low | medium | high | x-high] || [<frequency> | <semitones> | <percentage>]]
Initial:	medium
Applies to:	all elements
Inherited:	yes
Percentages:	refer to inherited value
Computed value:	one of the predefined pitch keywords if only the keyword is specified by itself, otherwise an absolute frequency calculated by converting the keyword value (if any) to a fixed frequency based on the current voice-family and by applying the specified relative offset (if any)

The '`voice-pitch`' property specifies the "baseline" pitch of the generated speech output, which depends on the used '`voice-family`' instance, and varies across speech synthesis processors (it approximately corresponds to the average pitch of the output). For example, the common pitch for a male voice is around 120Hz, whereas it is around 210Hz for a female voice.

Note that although the functionality provided by this property is similar to the `pitch` attribute of the `prosody` element from the SSML markup language [SSML], there are notable discrepancies. For example, CSS Speech pitch keywords and relative changes (frequency, semitone or percentage) are not mutually-exclusive, due to how values are inherited and combined for selected elements.

<frequency>

A value in frequency units (Hertz or kiloHertz, e.g. "100Hz", "+2kHz"). Values are restricted to positive numbers when the '`absolute`' keyword is specified. Otherwise (when the '`absolute`' keyword is not specified), a negative value represents a decrement, and a positive value represents an increment, relative to the inherited value. For example, "2kHz" is a positive offset (strictly equivalent to "+2kHz"), and "+2kHz absolute" is an absolute frequency (strictly equivalent to "2kHz absolute").

absolute

If specified, this keyword indicates that the specified frequency represents an absolute value. If a negative frequency is specified, the computed frequency will be zero.

<semitones>

Specifies a relative change (decrement or increment) to the inherited value. The syntax of allowed values is a <number> followed immediately by "st" (semitones). A semitone interval corresponds to the step between each note on an equal temperament chromatic scale. A semitone can therefore be quantified as the difference between two consecutive pitch frequencies on such scale. The ratio between two consecutive frequencies separated by exactly one semitone is the twelfth root of two (approximately 11011/10393, which equals exactly 1.0594631). As a result, the value in Hertz corresponding to a semitone offset is relative to the initial frequency the offset is applied to (in other words, a semitone doesn't correspond to a fixed numerical value in Hertz).

<percentage>

Positive and negative percentage values are allowed, to represent an increment or decrement (respectively) relative to the inherited value. Computed values are calculated by adding (or subtracting) the specified fraction of the inherited value, to (from) the inherited value. For example, 50% (which is equivalent to +50%) with a inherited value of 200Hz results in `200 + (200*0.5)` = 300Hz. Conversely, -50% results in `200-(200*0.5)` = 100Hz.

x-low, low, medium, high, x-high

A sequence of monotonically non-decreasing pitch levels that are implementation and voice specific. When the computed value for a given element is only a keyword (i.e. no relative offset is specified), then the corresponding absolute frequency will be re-evaluated on a voice change. Conversely, the application of a relative offset requires the calculation of the resulting frequency based on the current voice at the point at which the relative offset is specified, so the computed frequency will inherit absolutely regardless of any voice change further down the style cascade. Authors should therefore only use keyword values in cases where they wish that voice changes trigger the re-evaluation of the conversion from a keyword to a concrete, voice-dependent frequency.

Computed absolute frequencies that are negative are clamped to zero Hertz. Speech-capable user agents are likely to support a specific range of values rather than the full range of possible calculated numerical values for frequencies. The actual values in user agents may therefore be clamped to implementation-dependent minimum and maximum boundaries. For example: although the 0Hz frequency can be legitimately calculated, it may be clamped to a more meaningful value in the context of the speech synthesizer.

Examples of property values:

```
h1 { voice-pitch: 250Hz; } /* positive offset relative to the inherited absolu
h1 { voice-pitch: +250Hz; } /* identical to the line above */
h2 { voice-pitch: +30Hz absolute; } /* not an increment */
h2 { voice-pitch: absolute 30Hz; } /* identical to the line above */
h3 { voice-pitch: -20Hz; } /* negative offset (decrement) relative to the inhe
h4 { voice-pitch: -20Hz absolute; } /* illegal syntax => value ignored ("absol
h5 { voice-pitch: -3.5st; } /* semitones, negative offset */
h6 { voice-pitch: 25%; } /* this means "add a quarter of the inherited value,
h6 { voice-pitch: +25%; } /* identical to the line above */
```

12.4. The '`voice-range`' property

Name:	***voice-range***
Value:	<frequency> && absolute | [[x-low | low | medium | high | x-high] || [<frequency> | <semitones> | <percentage>]]
Initial:	medium
Applies to:	all elements
Inherited:	yes
Percentages:	refer to inherited value
Computed value:	one of the predefined pitch keywords if only the keyword is specified by itself, otherwise an absolute frequency calculated by converting the keyword value (if any) to a fixed frequency based on the current voice-family and by applying the specified relative offset (if any)

The '`voice-range`' property specifies the variability in the "baseline" pitch, i.e. how much the fundamental frequency may deviate from the average pitch of the speech output. The dynamic pitch range of the generated speech generally increases for a highly animated voice, for example when variations in inflection are used to convey meaning and emphasis in speech. Typically, a low range produces a flat, monotonic voice, whereas a high range produces an animated voice.

> Note that although the functionality provided by this property is similar to the range attribute of the prosody element from the SSML markup language [SSML], there are notable discrepancies. For example, CSS Speech pitch range keywords and relative changes (frequency, semitone or percentage) are not mutually-exclusive, due to how values are inherited and combined for selected elements.

<frequency>

A value in frequency units (Hertz or kiloHertz, e.g. "100Hz", "+2kHz"). Values are restricted to positive numbers when the '`absolute`' keyword is specified. Otherwise (when the '`absolute`' keyword is not specified), a negative value represents a decrement, and a positive value represents an increment, relative to the inherited value. For example, "2kHz" is a positive offset (strictly equivalent to "+2kHz"), and "+2kHz absolute" is an absolute frequency (strictly equivalent to "2kHz absolute").

absolute

If specified, this keyword indicates that the specified frequency represents an absolute value. If a negative frequency is specified, the computed frequency will be zero.

<semitones>

Specifies a relative change (decrement or increment) to the inherited value. The syntax of allowed values is a <number> followed immediately by "st" (semitones). A semitone interval corresponds to the step between each note on an equal temperament chromatic scale. A semitone can therefore be quantified as the difference between two consecutive pitch frequencies on such scale. The ratio between two consecutive frequencies separated by exactly one semitone is the twelfth root of two (approximately 11011/10393, which equals exactly 1.0594631). As a result, the value in Hertz corresponding to a semitone offset is relative to the initial frequency the offset is applied to (in other words, a semitone doesn't correspond to a fixed numerical value in Hertz).

<percentage>

Positive and negative percentage values are allowed, to represent an increment or decrement (respectively) relative to the inherited value. Computed values are calculated by adding (or subtracting) the specified fraction of the inherited value, to (from) the inherited value. For example, 50% (which is equivalent to +50%) with a inherited value of 200Hz results in `200 + (200*0.5)` = 300Hz. Conversely, -50% results in `200-(200*0.5)` = 100Hz.

x-low, low, medium, high, x-high

A sequence of monotonically non-decreasing pitch levels that are implementation and voice specific. When the computed value for a given element is only a keyword (i.e. no relative offset is specified), then the corresponding absolute frequency will be re-evaluated on a voice change. Conversely, the application of a relative offset requires the calculation of the resulting

frequency based on the current voice at the point at which the relative offset is specified, so the computed frequency will inherit absolutely regardless of any voice change further down the style cascade. Authors should therefore only use keyword values in cases where they wish that voice changes trigger the re-evaluation of the conversion from a keyword to a concrete, voice-dependent frequency.

Computed absolute frequencies that are negative are clamped to zero Hertz. Speech-capable user agents are likely to support a specific range of values rather than the full range of possible calculated numerical values for frequencies. The actual values in user agents may therefore be clamped to implementation-dependent minimum and maximum boundaries. For example: although the 0Hz frequency can be legitimately calculated, it may be clamped to a more meaningful value in the context of the speech synthesizer.

Examples of inherited values:

```
<body>
  <e1>
    <e2>
      <e3>
        <e4>
          <e5>
            <e6>
            ...
            </e6>
          </e5>
        </e4>
      </e3>
    </e2>
  </e1>
</body>
```

```
body { voice-range: inherit; } /* the initial value is 'medium'
                                  (the actual frequency value
                                  depends on the current voice) */

e1 { voice-range: +25%; } /* the computed value is
                           ['medium' + 25%] which resolves
                           to the frequency corresponding to 'medium'
                           plus 0.25 times the frequency
                           corresponding to 'medium' */

e2 { voice-range: +10Hz; } /* the computed value is
                           [FREQ + 10Hz] where "FREQ" is the absolute frequency
                           calculated in the "e1" rule above.
                           */

e3 { voice-range: inherit; /* this could be omitted,
                            but we explicitly specify it for clarity purposes *

     voice-family: "another-voice"; } /* this voice change would have resulted
                                 the re-evaluation of the initial 'medium' keywor
                                 inherited by the "body" element
                                 (i.e. conversion from a voice-dependent keyword
                                 to a concrete, absolute frequency),
                                 but because relative offsets were applied down t
                                 cascade, the inherited value is actually the fre
```

```
                                calculated at the "e2" rule above. */

e4 { voice-range: 200Hz absolute; } /* override with an absolute frequency
                                     which doesn't depend on the current voice

e5 { voice-range: 2st; } /* the computed value is an absolute frequency,
                          which is the result of the
                          calculation: 200Hz + two semitones
                          (reminder: the actual frequency corresponding to a se
                          depends on the base value to which it applies) */

e6 { voice-range: inherit; /* this could be omitted,
                            but we explicitly specify it for clarity purposes *

     voice-family: "yet-another-voice"; } /* despite the voice change,
                                the computed value is the same as
                                for "e5" (i.e. an absolute frequency value,
                                independent from the current voice) */
```

12.5. The ‘voice-stress’ property

Name:	***voice-stress***
Value:	normal | strong | moderate | none | reduced
Initial:	normal
Applies to:	all elements
Inherited:	yes
Percentages:	N/A
Computed value:	specified value

The ‘voice-stress’ property manipulates the strength of emphasis, which is normally applied using a combination of pitch change, timing changes, loudness and other acoustic differences. The precise meaning of the values therefore depend on the language being spoken.

Note that the functionality provided by this property is similar to the `emphasis` element from the SSML markup language [SSML].

normal
Represents the default emphasis produced by the speech synthesizer.

none
Prevents the synthesizer from emphasizing text it would normally emphasize.

moderate and strong
These values are monotonically non-decreasing in strength. Their application results in more emphasis than what the speech synthesizer would normally produce (i.e. more than the value corresponding to '`normal`').

reduced
Effectively the opposite of emphasizing a word.

Examples of property values, with HTML sample:

```
.default-emphasis { voice-stress: normal; }
.lowered-emphasis { voice-stress: reduced; }
.removed-emphasis { voice-stress: none; }
.normal-emphasis { voice-stress: moderate; }
.huge-emphasis { voice-stress: strong; }

...

<p>This is a big car.</p>
<!-- The speech output from the line above is identical to the line below: -->
<p>This is a <em class="default-emphasis">big</em> car.</p>

<p>This car is <em class="lowered-emphasis">massive</em>!</p>
<!-- The "em" below is totally de-emphasized, whereas the emphasis in the line
<p>This car is <em class="removed-emphasis">massive</em>!</p>

<!-- The lines below demonstrate increasing levels of emphasis: -->
<p>This is a <em class="normal-emphasis">big</em> car!</p>
<p>This is a <em class="huge-emphasis">big</em> car!!!</p>
```

13. Voice duration property

13.1. The ‘voice-duration’ property

Name:	***voice-duration***
Value:	auto | <time>
Initial:	*auto*
Applies to:	all elements
Inherited:	no
Percentages:	N/A
Computed value:	specified value

The ‘voice-duration’ property specifies how long it should take to render the selected element's content (not including audio cues, pauses and rests). Unless the value ‘auto’ is specified, this property takes precedence over the ‘voice-rate’ property, and should be used to determine a suitable speaking rate for the voice. An element for which the ‘voice-duration’ property value is not ‘auto’ may have descendants for which the ‘voice-duration’ and ‘voice-rate’ properties are specified, but these must be ignored. In other words, when a ‘time’ is specified for the ‘voice-duration’ of a selected element, it applies to the entire element subtree (children cannot override the property).

Note that the functionality provided by this property is similar to the duration attribute of the prosody element from the SSML markup language [SSML].

auto
: Resolves to a used value corresponding to the duration of the speech synthesis when using the inherited ‘voice-rate’.

<time>
: Specifies a value in absolute time units (seconds and milliseconds, e.g. "+3s", "250ms"). Only non-negative values are allowed.

14. List items and counters styles

The ‘ list-style-type’ property of [CSS21] specifies three types of list item markers: glyphs, numbering systems, and alphabetic systems. The values allowed for this property are also used for the counter() function of the ‘content’ property. The CSS Speech module defines how to render these styles in the aural dimension, using speech synthesis. The ‘list-style-image’ property of [CSS21] is ignored, and instead the ‘list-style-type’ is used.

> Note that the speech rendering of new features from the CSS Lists and Counters Module Level 3 [CSS3LIST] is not covered in this level of CSS Speech, but may be defined in a future specification.

disc, circle, square
: For these list item styles, the user agent defines (possibly based on user preferences) what equivalent phrase is spoken or what audio cue is played. List items with graphical bullets are therefore announced appropriately in an implementation-dependent manner.

decimal, decimal-leading-zero, lower-roman, upper-roman, georgian, armenian
: For these list item styles, corresponding numbers are spoken as-is by the speech synthesizer, and may be complemented with additional audio cues or speech phrases in the document's language (i.e. with the same TTS voice used to speak the list item content) in order to indicate the presence of list items. For example, when using the English language, the list item counter could be prefixed with the word "Item", which would result in list items being announced with "Item one", "Item two", etc.

lower-latin, lower-alpha, upper-latin, upper-alpha, lower-greek
: These list item styles are spelled out letter-by-letter by the speech synthesizer, in the document language (i.e. with the same TTS voice used to speak the list item content). For example, ‘`lower-greek`’ in English would be read out as "alpha", "beta", "gamma", etc. Conversely, ‘`upper-latin`’ in French would be read out as /a/, /be/, /se/, etc. (phonetic notation)

> Note that it is common for user agents such as screen readers to announce the nesting depth of list items, or more generally, to indicate additional structural information pertaining to complex hierarchical content. The verbosity of these additional audio cues and/or speech output can usually be controlled by users, and contribute to increasing usability. These navigation aids are implementation-dependent, but it is recommended that user agents supporting the CSS Speech module ensure that these additional audio cues and speech output don't generate redundancies or create inconsistencies (for example: duplicated or different list item numbering scheme).

15. Inserted and replaced content

Note that this entire section is informative.

Sometimes, authors will want to specify a mapping from the source text into another string prior to the application of the regular pronunciation rules. This may be used for uncommon abbreviations or acronyms which are unlikely to be recognized by the synthesizer. The '`content`' property can be used to replace one string by another. The functionality provided by this property is similar to the `alias` attribute of the `sub` element from the SSML markup language [SSML].

In this example, the abbreviation is rendered using the content of the title attribute instead of the element's content.

```
/* This replaces the content of the selected element
by the string "World Wide Web Consortium". */
abbr { content: attr(title); }
...

<abbr title="World Wide Web Consortium">W3C</abbr>
```

In a similar way, text strings in a document can be replaced by a previously recorded version.

In this example - assuming the format is supported, the file is available and the UA is configured to do so - a recording of Sir John Gielgud's declamation of the famous monologue is played. Otherwise the UA falls back to render the text using synthesized speech.

```
.hamlet { content: url(./audio/gielgud.wav); }
...

<div class="hamlet">
To be, or not to be: that is the question:
</div>
```

Furthermore, authors (or users via a user stylesheet) may add some information to ease the understanding of structures during non-visual interaction with the document. They can do so by using the '`::before`' and '`::after`' pseudo-elements. Note that different stylesheets can be used to define the level of verbosity for additional information spoken by screen readers.

This example inserts the string "Start list: " before a list and the string "List item: " before the content of each list item. Likewise, the string "List end: " gets inserted after the list to inform the user that the list speech output is over.

```
ul::before { content: "Start list: "; }
ul::after  { content: "List end. "; }
li::before { content: "List item: "; }
```

Detailed information can be found in the CSS3 Generated and Replaced Content module [CSS-CONTENT-3].

16. Pronunciation, phonemes

Note that this entire section is informative.

CSS does not specify how to define the pronunciation (expressed using a well-defined phonetic alphabet) of a particular piece of text within the markup document. A "phonemes" property was described in earlier drafts of this specification, but objections were raised due to breaking the principle of separation between content and presentation (the "phonemes" authored within aural CSS stylesheets would have needed to be updated each time text changed within the markup document). The "phonemes" functionality is therefore considered out-of-scope in CSS (the presentation layer) and should be addressed in the markup / content layer.

The "pronunciation" `rel` value allows importing pronunciation lexicons in HTML documents using the `link` element (similar to how CSS stylesheets can be included). The W3C PLS (Pronunciation Lexicon Specification) [PRONUNCIATION-LEXICON] is one format that can be used to describe such a lexicon.

Additionally, an attribute-based mechanism can be used within the markup to author text-pronunciation associations. At the time of writing, such mechanism isn't formally defined in the W3C HTML standard(s). However, the EPUB 3.0 specification allows (x)HTML5 documents to contain attributes derived from the [SSML] specification, that describe how to pronounce text based on a particular phonetic alphabet.

Appendix A — Property index

Property	Values	Initial	Applies to	Inh.	Percentages	Media
cue	<‘cue-before’> <‘cue-after’>?	N/A (see individual properties)		no	N/A	speech
cue-after	<uri> <decibel>? | none	none		no	N/A	speech
cue-before	<uri> <decibel>? | none	none		no	N/A	speech
pause	<‘pause-before’> <‘pause-after’>?	N/A (see individual properties)		no	N/A	speech
pause-after	<time> | none | x-weak | weak | medium | strong | x-strong	none		no	N/A	speech
pause-before	<time> | none | x-weak | weak | medium | strong | x-strong	none		no	N/A	speech
rest	<‘rest-before’> <‘rest-after’>?	N/A (see individual properties)		no	N/A	speech
rest-after	<time> | none | x-weak | weak | medium | strong | x-strong	none		no	N/A	speech
rest-before	<time> | none | x-weak | weak | medium | strong | x-strong	none		no	N/A	speech
speak	auto | never | always	auto		yes	N/A	speech
speak-as	normal | spell-out || digits || [literal-punctuation | no-punctuation]	normal		yes	N/A	speech
voice-balance	<number> | left | center | right | leftwards | rightwards	center		yes	N/A	speech
voice-duration	auto | <time>	auto		no	N/A	speech
voice-family	[[<name> | <generic-voice>],]* [<name> | <generic-voice>] | preserve	implementation-dependent		yes	N/A	speech
voice-pitch	<frequency> && absolute | [[x-low | low | medium | high | x-high] || [<frequency> | <semitones> | <percentage>]]	medium		yes	refer to inherited value	speech

Property	Values	Initial	Applies to	Inh.	Percentages	Media
voice-range	<frequency> && absolute | [[x-low | low | medium | high | x-high] || [<frequency> | <semitones> | <percentage>]]	medium		yes	refer to inherited value	speech
voice-rate	[normal | x-slow | slow | medium | fast | x-fast] || <percentage>	normal		yes	refer to default value	speech
voice-stress	normal | strong | moderate | none | reduced	normal		yes	N/A	speech
voice-volume	silent | [[x-soft | soft | medium | loud | x-loud] || <decibel>]	medium		yes	N/A	speech

The following properties are defined in other modules or specifications:

- ***display*** [CSS21]
- ***padding*** [CSS21]
- ***border*** [CSS21]
- ***margin*** [CSS21]
- ***font-family*** [CSS21]
- ***content*** [CSS-CONTENT-3]
- ***list-style-type*** [CSS21]
- ***list-style-image*** [CSS21]

The following definitions are provided by other modules or specifications:

- ***cascade*** [CSS21]
- ***visual box model*** [CSS21]
- ***time*** [CSS-VALUES-3]
- ***frequency*** [CSS-VALUES-3]
- ***number*** [CSS-VALUES-3]
- ***integer*** [CSS-VALUES-3]
- ***non-negative-number*** [CSS-VALUES-3]
- ***percentage*** [CSS-VALUES-3]
- ***identifier*** [CSS21]

- ***strings*** [CSS21]

Appendix B — Index

- rest-before, **10.1.**
- speak, **8.1.**
- speak-as, **8.2.**
- strings, **??**
- style sheet, **??**
  - as conformance class, **??**
- time, **??**
- UA, **??**
- User Agent, **??**
- visual box model, **??**
- voice-balance, **7.2.**
- voice-duration, **13.1.**
- voice-family, **12.1.**
- voice-pitch, **12.3.**
- voice-range, **12.4.**
- voice-rate, **12.2.**
- voice-stress, **12.5.**
- voice-volume, **7.1.**

Appendix C — Definitions

Glossary

The following terms and abbreviations are used in this module.

UA
User Agent
: A program that reads and/or writes CSS style sheets on behalf of a user in either or both of these categories: programs whose purpose is to render documents (e.g., browsers) and programs whose purpose is to create style sheets (e.g., editors). A UA may fall into both categories. (There are other programs that read or write style sheets, but this module gives no rules for them.)

document

A tree-structured document with elements and attributes, such as an SGML or XML document [XML11].

style sheet
A CSS style sheet

Conformance

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:

> 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.

Conformance to the CSS3 Speech module is defined for three 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 the CSS3 Speech module if all of its declarations that use properties defined in this module have values that are valid according to the generic CSS grammar and the individual grammars of each property as given in this module.

A renderer is conformant to the CSS3 Speech module if, in addition to interpreting the style sheet as defined by the appropriate specifications, it supports all the properties defined by CSS3 Speech module 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 CSS3 Speech module if it writes syntactically correct style sheets, according to the generic CSS grammar and the individual grammars of each property in this module.

CR exit criteria

As described in the W3C process document, a Candidate Recommendation (CR) is a specification that W3C recommends for use on the Web. The next stage is "Recommendation" when the specification is sufficiently implemented.

For this specification to be proposed as a W3C Recommendation, the following conditions shall be met. There must be at least two independent, interoperable implementations of each feature. Each feature may be implemented by a different set of products, there is no requirement that all features be implemented by a single product. For the purposes of this criterion, we define the following terms:

independent
: each implementation must be developed by a different party and cannot share, reuse, or derive from code used by another qualifying implementation. Sections of code that have no bearing on the implementation of this specification are exempt from this requirement.

interoperable
: passing the respective test case(s) in the official CSS test suite, or, if the implementation is not a Web browser, an equivalent test. Every relevant test in the test suite should have an equivalent test created if such a user agent (UA) is to be used to claim interoperability. In addition if such a UA is to be used to claim interoperability, then there must one or more additional UAs which can also pass those equivalent tests in the same way for the purpose of interoperability. The equivalent tests must be made publicly available for the purposes of peer review.

implementation

a user agent which:

1. implements the specification.
2. is available to the general public. The implementation may be a shipping product or other publicly available version (i.e., beta version, preview release, or "nightly build"). Non-shipping product releases must have implemented the feature(s) for a period of at least one month in order to demonstrate stability.
3. is not experimental (i.e., a version specifically designed to pass the test suite and is not intended for normal usage going forward).

A minimum of sixth months of the CR period must have elapsed. This is to ensure that enough time is given for any remaining major errors to be caught.

Features will be dropped if two or more interoperable implementations are not found by the end of the CR period.

Features may/will also be dropped if adequate/sufficient (by judgment of CSS WG) tests have not been produced for those feature(s) by the end of the CR period.

Appendix D — Acknowledgements

The editors would like to thank the members of the W3C Voice Browser and Cascading Style Sheets working groups for their assistance in preparing this specification. Special thanks to Ellen Eide (IBM) for her detailed comments, and to Elika Etemad (Fantasai) for her thorough reviews.

Appendix E — Changes

The following changes have been made since the 2012 Candidate Recommendation:

- Renamed the ‘`none`’ and ‘`normal`’ values of ‘`speak`’ to ‘`never`’ and ‘`always`’ for clarity. See Issue 510.
- Made the ‘`auto`’ value of ‘`speak`’ respond to ‘`visibility`’. See Issue 511.

Appendix F — References

Normative references

[CSS-VALUES-3]
Tab Atkins; fantasai. *CSS Values and Units Module Level 3.* 29 September 2016. W3C Candidate Recommendation. (Work in progress.) URL: https://www.w3.org/TR/2016/CR-css-values-3-20160929/

[CSS2]
Ian Jacobs; et al. *Cascading Style Sheets, level 2 (CSS2) Specification.* 11 April 2008. W3C Recommendation. URL: http://www.w3.org/TR/2008/REC-CSS2-20080411

[CSS21]
Bert Bos; et al. *Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification.* 7 June 2011. W3C Recommendation. URL: http://www.w3.org/TR/2011/REC-CSS2-20110607

[RFC2119]
S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels.* RFC 2119. URL: http://www.ietf.org/rfc/rfc2119.txt

[SSML]
Daniel C. Burnett; 双志伟 (Zhi Wei Shuang). *Speech Synthesis Markup Language (SSML) Version 1.1.* 7 September 2010. W3C Recommendation. URL: http://www.w3.org/TR/2010/REC-speech-synthesis11-20100907/

[XML11]
Eve Maler; et al. *Extensible Markup Language (XML) 1.1 (Second Edition).* 16 August 2006. W3C Recommendation. URL: http://www.w3.org/TR/2006/REC-xml11-20060816

Other references

[CSS-CONTENT-3]
Elika J. Etemad / fantasai; Dave Cramer. *CSS Generated Content Module Level 3.* 2 June 2016. W3C Working Draft. (Work in progress.) URL: http://www.w3.org/TR/2016/WD-css-content-3-20160602/

[CSS3LIST]
Tab Atkins. *CSS Lists and Counters Module Level 3.* 20 March 2014. W3C Working Draft. (Work in progress.) URL: http://www.w3.org/TR/2014/WD-css-lists-3-20140320/

[PRONUNCIATION-LEXICON]
Paolo Baggia. *Pronunciation Lexicon Specification (PLS) Version 1.0.* 14 October 2008. W3C Recommendation. URL: http://www.w3.org/TR/2008/REC-pronunciation-lexicon-20081014/

[SSML-SAYAS]
Daniel C. Burnett; et al. *SSML 1.0 say-as attribute values.* 26 May 2005. W3C Working Group Note. URL: http://www.w3.org/TR/2005/NOTE-ssml-sayas-20050526