W3C Recommendation

CSS Fonts Module Level 3

W3C Recommendation 20 September 2018

This version:
https://www.w3.org/TR/2018/REC-css-fonts-3-20180920/
Latest version:
https://www.w3.org/TR/css-fonts-3/
Latest editor's draft:
https://drafts.csswg.org/css-fonts/
Previous versions:
https://www.w3.org/TR/2018/PR-css-fonts-3-20180814/
https://www.w3.org/TR/2018/CR-css-fonts-3-20180626/
https://www.w3.org/TR/2018/CR-css-fonts-3-20180315/
https://www.w3.org/TR/2013/CR-css-fonts-3-20131003/
Issues List:
css-fonts-3 issues on GitHub
Discussion:
on GitHub (preferred), or www-style@w3.org with subject line "`[css-fonts] … message topic …`" (archives)
Test Suite:
https://test.csswg.org/harness/results/css-fonts-3_dev/grouped/
Editors:
John Daggett (Invited Expert)
Myles C. Maxfield (Apple Inc.)
Chris Lilley (W3C)

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

Abstract

This CSS3 module describes how font properties are specified and how font resources are loaded dynamically. The contents of this specification are a consolidation of content previously divided into CSS3 Fonts and CSS3 Web Fonts modules. The description of font load events was moved into the CSS Font Loading module.

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 as a W3C Recommendation..

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 test suite and implementation report are available.

Table of contents

1. Introduction

A font provides a resource containing the visual representation of characters [CHARMOD] [UNICODE]. At the simplest level it contains information that maps character codes to shapes (called glyphs) that represent these characters. Fonts sharing a common design style are commonly grouped into font families classified by a set of standard font properties. Within a family, the shape displayed for a given character can vary by stroke weight, slant or relative width, among others. An individual font face is described by a unique combination of these properties. For a given range of text, CSS font properties are used to select a font family and a specific font face within that family to be used when rendering that text. As a simple example, to use the bold form of Helvetica one could use:

```
body {
    font-family: Helvetica;
    font-weight: bold;
}
```

Font resources may be installed locally on the system on which a user agent is running or downloadable. For local font resources descriptive information can be obtained directly from the

font resource. For downloadable font resources (sometimes referred to as web fonts), the descriptive information is included with the reference to the font resource.

Families of fonts typically don't contain a single face for each possible variation of font properties. The CSS font selection mechanism describes how to match a given set of CSS font properties to a single font face.

2. Typography Background

This section is non-normative.

Typographic traditions vary across the globe, so there is no unique way to classify all fonts across languages and cultures. For even common Latin letters, wide variations are possible:

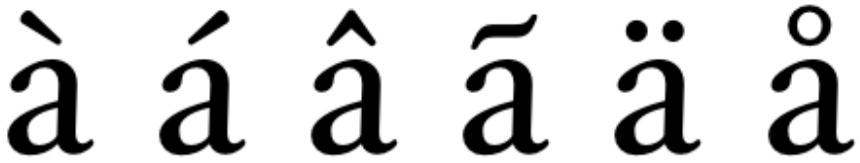
One character, many glyph variations

Differences in the anatomy of letterforms is one way to distinguish fonts. For Latin fonts, flourishes at the ends of a character's main strokes, or serifs, can distinguish a font from those without. Similar comparisons exist in non-Latin fonts between fonts with tapered strokes and those using primarily uniform strokes:

Letterforms with and without serifs

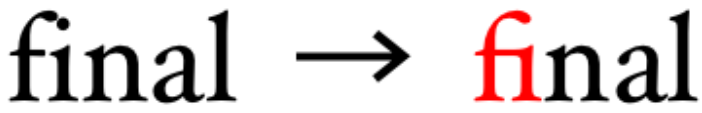
Similar groupings for Japanese typefaces

Fonts contain letterforms and the data needed to map characters to these letterforms. Often this may be a simple one-to-one mapping, but more complex mappings are also possible. The use of combining diacritic marks creates many variations for an underlying letterform:



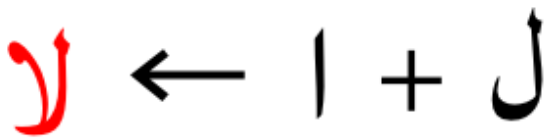
Variations with diacritic marks

A sequence of characters can be represented by a single glyph known as a ligature:



Ligature example

Visual transformations based on textual context are often stylistic option in European languages. They are required to correctly render languages like [ARABIC-TYPO], the lam and alef characters below *must* be combined when they exist in sequence:



Required Arabic ligature

The relative complexity of these shaping transformations requires additional data within the font.

Sets of font faces with various stylistic variations are often grouped together into font families. In the simplest case a regular face is supplemented with bold and italic faces, but much more extensive groupings are possible. Variations in the thickness of letterform strokes, the ***weight***, and the overall proportions of the letterform, the ***width***, are most common. In the example below, each letter uses a different font face within the Univers font family. The width used increases from top to bottom and the weight increases from left to right:

Weight and width variations within a single font family

Creating fonts that support multiple scripts is a difficult task; designers need to understand the cultural traditions surrounding the use of type in different scripts and come up with letterforms that somehow share a common theme. Many languages often share a common script and each of these languages may have noticeable stylistic differences. For example, the Arabic script, when used for Persian and Urdu, exhibits significant and systematic differences in letterforms, as does Cyrillic when used with languages such as Serbian and Russian.

The *character map* of a font defines the mapping of characters to glyphs for that font. If a document contains characters not supported by the *character maps* of the fonts contained in a font family list, a user agent may use a *system font fallback* procedure to locate an appropriate font that does. If no appropriate font can be found, some form of "missing glyph" character will be rendered by the user agent. System fallback can occur when the specified list of font families does not include a font that supports a given character.

Although the *character map* of a font maps a given character to a glyph for that character, modern font technologies such as OpenType [OPENTYPE] and AAT (Apple Advanced Typography) [AAT-FEATURES] provide ways of mapping a character to different glyphs based upon feature settings. Fonts in these formats allow these features to be embedded in the font itself and controlled by applications. Common typographic features which can be specified this way include ligatures, swashes, contextual alternates, proportional and tabular figures, and automatic fractions, to list just a few. For a visual overview of OpenType features, see the [OPENTYPE-FONT-GUIDE].

3. Basic Font Properties

The particular font face used to render a character is determined by the font family and other font properties that apply to a given element. This structure allows settings to be varied independent of each other.

3.1. Font family: the font-family property

Name:	***font-family***
Value:	[*<family-name>* | *<generic-family>*] #
Initial:	depends on user agent
Applies to:	all elements
Inherited:	yes
Percentages:	N/A
Media:	visual
Computed value:	as specified
Animatable:	no

This property specifies a prioritized list of font family names or generic family names. A font family defines a set of faces that vary in weight, width or slope. CSS uses the combination of a family name with other style attributes to select an individual face. Using this selection mechanism, rather than selecting a face via the style name as is often done in design applications, allows some degree of regularity in textual display when fallback occurs.

> Designers should note that the CSS definition of font attributes used for selection are explicitly not intended to define a font taxonomy. A type designer's idea of a family may often extend to a set of faces that vary along axes other than just the standard axes of weight, width and slope. A family may extend to include both a set of serif faces and a set of sans-serif faces or vary along axes that are unique to that family. The CSS font selection mechanism merely provides a way to determine the “closest” substitute when substitution is necessary.

Unlike other CSS properties, component values are a comma-separated list indicating alternatives. A user agent iterates through the list of family names until it matches an available font that contains a glyph for the character to be rendered. This allows for differences in available fonts across platforms and for differences in the range of characters supported by individual fonts.

A font family name only specifies a name given to a set of font faces, it does not specify an individual face. For example, given the availability of the fonts below, Futura would match but Futura Medium would not:

Family and individual face names

Consider the example below:

```
body {
    font-family: Helvetica, Verdana, sans-serif;
}
```

If Helvetica is available it will be used when rendering. If neither Helvetica nor Verdana is present, then the user-agent-defined sans serif font will be used.

There are two types of font family names:

<family-name>
: The name of a font family of choice such as Helvetica or Verdana in the previous example.

<generic-family>
: The following generic family keywords are defined: '`serif`', '`sans-serif`', '`cursive`', '`fantasy`', and '`monospace`'. These keywords can be used as a general fallback mechanism when an author's desired font choices are not available. As keywords, they must not be quoted. Authors are encouraged to append a generic font family as a last alternative for improved robustness.

Font family names other than generic families must either be given quoted as strings, or unquoted as a sequence of one or more identifiers. This means most punctuation characters and digits at the start of each token must be escaped in unquoted font family names.

To illustrate this, the following declarations are invalid:

```
font-family: Red/Black, sans-serif;
font-family: "Lucida" Grande, sans-serif;
font-family: Ahem!, sans-serif;
font-family: test@foo, sans-serif;
font-family: #POUND, sans-serif;
font-family: Hawaii 5-0, sans-serif;
```

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

To avoid mistakes in escaping, it is recommended to quote font family names that contain white space, digits, or punctuation characters other than hyphens:

```
body { font-family: "New Century Schoolbook", serif }

<BODY STYLE="font-family: '21st Century', fantasy">
```

Font family *names* that happen to be the same as keyword value (‘`inherit`’, ‘`serif`’, etc.) must be quoted to prevent confusion with the keywords with the same names. UAs must not consider these keywords as matching the *<family-name>* type. This applies to any keyword across all of CSS.

The precise way a set of fonts are grouped into font families varies depending upon the platform font management API's. The Windows GDI API only allows four faces to be grouped into a family while the DirectWrite API and API's on OSX and other platforms support font families with a variety of weights, widths and slopes (see Appendix A for more details).

Some font formats allow fonts to carry multiple localizations of the family name. User agents must recognize and correctly match all of these names independent of the underlying platform localization, system API used or document encoding:

Localized family names

The details of localized font family name matching and the corresponding issues of case sensitivity are described below in the font matching section.

3.1.1. Generic font families

All five generic font families must always result in at least one matched font face, for all CSS implementations. However, the generics may be composite faces (with different typefaces based on such things as the Unicode range of the character, the language of the containing element, user preferences and system settings, among others). They are also not guaranteed to always be different from each other.

User agents should provide reasonable default choices for the generic font families, which express the characteristics of each family as well as possible, within the limits allowed by the underlying technology. User agents are encouraged to allow users to select alternative choices for the generic fonts.

serif

Serif fonts represent the formal text style for a script. This often means but is not limited to glyphs that have finishing strokes, flared or tapering ends, or have actual serifed endings (including slab serifs). Serif fonts are typically proportionately-spaced. They often display a greater variation between thick and thin strokes than fonts from the '`sans-serif`' generic font family. CSS uses the term '`serif`' to apply to a font for any script, although other names may be more familiar for particular scripts, such as Mincho (Japanese), Sung or Song (Chinese), Batang (Korean). For Arabic, the Naskh style would correspond to '`serif`' more due to its typographic role rather than its actual design style. Any font that is so described may be used to represent the generic '`serif`' family.

Sample serif fonts

sans-serif

Glyphs in sans-serif fonts, as the term is used in CSS, are generally low contrast (vertical and horizontal stems have the close to the same thickness) and have stroke endings that are plain — without any flaring, cross stroke, or other ornamentation. Sans-serif fonts are typically proportionately-spaced. They often have little variation between thick and thin strokes, compared to fonts from the '`serif`' family. CSS uses the term '`sans-serif`' to apply to a font for any script, although other names may be more familiar for particular scripts, such as Gothic (Japanese), Hei (Chinese), or Gulim (Korean). Any font that is so described may be used to represent the generic '`sans-serif`' family.

Sample sans-serif fonts

cursive

Glyphs in cursive fonts generally use a more informal script style, and the result looks more like handwritten pen or brush writing than printed letterwork. CSS uses the term '`cursive`' to apply to a font for any script, although other names such as Chancery, Brush, Swing and Script are also used in font names.

Sample cursive fonts

fantasy

Fantasy fonts are primarily decorative or expressive fonts that contain decorative or expressive representations of characters. These do not include Pi or Picture fonts which do not represent

actual characters.

Sample fantasy fonts

monospace

The sole criterion of a monospace font is that all glyphs have the same fixed width. This is often used to render samples of computer code.

Sample monospace fonts

3.2. Font weight: the font-weight property

Name:	***font-weight***
Value:	normal | bold | bolder | lighter | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900
Initial:	normal
Applies to:	all elements
Inherited:	yes
Percentages:	N/A
Media:	visual
Computed value:	numeric weight value (see description)
Animatable:	as font weight

The '`font-weight`' property specifies the weight of glyphs in the font, their degree of blackness or stroke thickness.

Values have the following meanings:

100 to 900
: These values form an ordered sequence, where each number indicates a weight that is at least as dark as its predecessor. These roughly correspond to the commonly used weight names below:

- 100 - Thin
- 200 - Extra Light (Ultra Light)
- 300 - Light
- 400 - Normal
- 500 - Medium
- 600 - Semi Bold (Demi Bold)
- 700 - Bold
- 800 - Extra Bold (Ultra Bold)
- 900 - Black (Heavy)

normal
Same as ‘`400`’.

bold
Same as ‘`700`’.

bolder
Specifies a bolder weight than the inherited value.

lighter
Specifies a lighter weight than the inherited value.

Font formats that use a scale other than a nine-step scale should map their scale onto the CSS scale so that 400 roughly corresponds with a face that would be labeled as Regular, Book, Roman and 700 roughly matches a face that would be labeled as Bold. Or weights may be inferred from the style names, ones that correspond roughly with the scale above. The scale is relative, so a face with a larger weight value must never appear lighter. If style names are used to infer weights, care should be taken to handle variations in style names across locales.

Quite often there are only a few weights available for a particular font family. When a weight is specified for which no face exists, a face with a nearby weight is used. In general, bold weights map to faces with heavier weights and light weights map to faces with lighter weights (see the font matching section below for a precise definition). The examples here illustrate which face is used for different weights, grey indicates a face for that weight does not exist so a face with a nearby weight is used:

Weight mappings for a font family with 400, 700 and 900 weight faces

Weight mappings for a font family with 300 and 600 weight faces

Although the practice is not well-loved by typographers, bold faces are often synthesized by user agents for faces that lack actual bold faces. For the purposes of style matching, these faces must be treated as if they exist within the family. Authors can explicitly avoid this behavior by using the '`font-synthesis`' property.

Specified values of '`bolder`' and '`lighter`' indicate weights relative to the weight of the parent element. The computed weight is calculated based on the inherited '`font-weight`' value using the chart below.

Inherited value	bolder	lighter
100	400	100
200	400	100
300	400	100
400	700	100
500	700	100
600	900	400
700	900	400
800	900	700
900	900	700

The table above is equivalent to selecting the next relative bolder or lighter face, given a font family containing normal and bold faces along with a thin and a heavy face. Authors who desire finer control over the exact weight values used for a given element may use numerical values instead of relative weights.

3.3. Font width: the font-stretch property

Name:	***font-stretch***
Value:	normal | ultra-condensed | extra-condensed | condensed | semi-condensed | semi-expanded | expanded | extra-expanded | ultra-expanded
Initial:	normal
Applies to:	all elements
Inherited:	yes
Percentages:	N/A
Media:	visual
Computed value:	as specified
Animatable:	as font stretch

The '`font-stretch`' property selects a normal, condensed, or expanded face from a font family. Absolute keyword values have the following ordering, from narrowest to widest:

- ***ultra-condensed***
- ***extra-condensed***
- ***condensed***
- ***semi-condensed***
- ***normal***
- ***semi-expanded***
- ***expanded***
- ***extra-expanded***
- ***ultra-expanded***

When a face does not exist for a given width, normal or condensed values map to a narrower face, otherwise a wider face. Conversely, expanded values map to a wider face, otherwise a narrower face. The figure below shows how the nine font-stretch property settings affect font selection for font family containing a variety of widths, grey indicates a width for which no face exists and a different width is substituted:

Width mappings for a font family with condensed, normal and expanded width faces

Animation of font stretch: Font stretch is interpolated in discrete steps. The interpolation happens as though the ordered values are equally spaced real numbers. The interpolation result is rounded to the nearest value, with values exactly halfway between two values rounded towards the later value in the list above.

3.4. Font style: the font-style property

Name:	***font-style***
Value:	normal | italic | oblique
Initial:	normal
Applies to:	all elements
Inherited:	yes
Percentages:	N/A
Media:	visual
Computed value:	as specified
Animatable:	no

The ‘`font-style`’ property allows italic or oblique faces to be selected. Italic forms are generally cursive in nature while oblique faces are typically sloped versions of the regular face. Oblique faces can be simulated by artificially sloping the glyphs of the regular face. Compare the artificially sloped renderings of Palatino ‘a’ and Baskerville ‘N’ in grey with the actual italic versions:

Artificial sloping versus real italics

Values have the following meanings:

normal
: selects a face that is classified as a normal face, one that is neither italic or obliqued

italic
: selects a font that is labeled as an italic face, or an oblique face if one is not

oblique
: selects a font that is labeled as an oblique face, or an italic face if one is not

If no italic or oblique face is available, oblique faces can be synthesized by rendering non-obliqued faces with an artificial obliquing operation. The use of these artificially obliqued faces can be disabled using the '`font-synthesis`' property. The details of the obliquing operation are not explicitly defined.

> Authors should also be aware that synthesized approaches may not be suitable for scripts like Cyrillic, where italic forms are very different in shape. It is always better to use an actual italic font rather than rely on a synthetic version.

Many scripts lack the tradition of mixing a cursive form within text rendered with a normal face. Chinese, Japanese and Korean fonts almost always lack italic or oblique faces. Fonts that support a mixture of scripts will sometimes omit specific scripts such as Arabic from the set of glyphs supported in the italic face. User agents should be careful about making *character map* assumptions across faces when implementing support for *system font fallback*.

3.5. Font size: the font-size property

Name:	***font-size***
Value:	*<absolute-size>* | *<relative-size>* | *<length-percentage>*
Initial:	medium
Applies to:	all elements
Inherited:	yes
Percentages:	refer to parent element's font size
Media:	visual
Computed value:	absolute length
Animatable:	as length

This property indicates the desired height of glyphs from the font. For scalable fonts, the font-size is a scale factor applied to the EM unit of the font. (Note that certain glyphs may bleed outside their EM box.) For non-scalable fonts, the font-size is converted into absolute units and matched against the declared '`font-size`' of the font, using the same absolute coordinate space for both of the matched values. Values have the following meanings:

<absolute-size>

An *<absolute-size>* keyword refers to an entry in a table of font sizes computed and kept by the user agent. Possible values are:

```
[ xx-small | x-small | small | medium | large | x-large | xx-large ]
```

<relative-size>

A *<relative-size>* keyword is interpreted relative to the table of font sizes and the computed '`font-size`' of the parent element. Possible values are:

```
[ larger | smaller ]
```

For example, if the parent element has a font size of '`medium`', a value of '`larger`' will make the font size of the current element be '`large`'. If the parent element's size is not close to a table entry, the user agent is free to interpolate between table entries or round off to the closest one. The user agent may have to extrapolate table values if the numerical value goes beyond the keywords.

<length-percentage>

A length value [CSS-VALUES] specifies an absolute font size (independent of the user

agent's font table). Negative lengths are invalid.

A percentage value specifies an absolute font size relative to the parent element's font size. Use of percentage values, or values in ems, leads to more robust and cascadable style sheets. Negative percentages are invalid.

The following table provides user agent guidelines for the absolute-size scaling factor and their mapping to HTML heading and absolute font-sizes. The '`medium`' value is used as the reference middle value. The user agent may fine-tune these values for different fonts or different types of display devices.

CSS absolute-size values	`xx-small`	`x-small`	`small`	`medium`	`large`	`x-large`	`xx-large`	
scaling factor	3/5	3/4	8/9	1	6/5	3/2	2/1	3/1
HTML headings	h6		h5	h4	h3	h2	h1	
HTML font sizes	1		2	3	4	5	6	7

> ***Note 1.*** *To preserve readability, an UA applying these guidelines should nevertheless avoid creating font-size resulting in less than 9 device pixels per EM unit on a computer display.*

> ***Note 2.*** *In CSS1, the suggested scaling factor between adjacent indexes was 1.5 which user experience proved to be too large. In CSS2, the suggested scaling factor for computer screen between adjacent indexes was 1.2 which still created issues for the small sizes. The new scaling factor varies between each index to provide a better readability.*

The actual value of this property may differ from the computed value due a numerical value on '`font-size-adjust`' and the unavailability of certain font sizes.

Child elements inherit the computed '`font-size`' value (otherwise, the effect of '`font-size-adjust`' would compound).

```
p { font-size: 12pt; }
blockquote { font-size: larger }
em { font-size: 150% }
em { font-size: 1.5em }
```

3.6. Relative sizing: the font-size-adjust property

Name:	***font-size-adjust***
Value:	none | *<number>*
Initial:	none
Applies to:	all elements
Inherited:	yes
Percentages:	N/A
Media:	visual
Computed value:	as specified
Animatable:	as number

For any given font size, the apparent size and legibility of text varies across fonts. For scripts such as Latin or Cyrillic that distinguish between upper and lowercase letters, the relative height of lowercase letters compared to their uppercase counterparts is a determining factor of legibility. This is commonly referred to as the ***aspect value***. Precisely defined, it is equal to the `x-height` of a font divided by the font size.

In situations where font fallback occurs, fallback fonts may not share the same aspect value as the desired font family and will thus appear less readable. The '`font-size-adjust`' property is a way to preserve the readability of text when font fallback occurs. It does this by adjusting the font-size so that the x-height is the same regardless of the font used.

> The style defined below defines Verdana as the desired font family, but if Verdana is not available Futura or Times will be used.
>
> ```
> p {
>     font-family: Verdana, Futura, Times;
> }
>
> Lorem ipsum dolor sit amet, ...
> ```
>
> Verdana has a relatively high aspect value, lowercase letters are relatively tall compared to uppercase letters, so at small sizes text appears legible. Times has a lower aspect value and so if fallback occurs, the text will be less legible at small sizes than Verdana.

How text rendered in each of these fonts compares is shown below, the columns show text rendered in Verdana, Futura and Times. The same font-size value is used across cells within each row and red lines are included to show the differences in x-height. In the upper half each row is rendered in the same font-size value. The same is true for the lower half but in this half the '`font-size-adjust`' property is also set so that the actual font size is adjusted to preserve the x-height across each row. Note how small text remains relatively legible across each row in the lower half.

Text with and without the use of '`font-size-adjust`'

This property allows authors to specify an aspect value for an element that will effectively preserve the x-height of the first choice font, whether it is substituted or not. Values have the following meanings:

none
: Do not preserve the font's x-height.

<number>
: Specifies the aspect value used in the calculation below to calculate the adjusted font size:

```
c = ( a / a' ) s
```

where:

```
s  =  font-size value
a  =  aspect value as specified by the 'font-size-adjust' property
a' =  aspect value of actual font
c  =  adjusted font-size to use
```

Negative values are invalid.

This value applies to any font that is selected but in typical usage it should be based on the aspect value of the first font in the font-family list. If this is specified accurately, the `(a/a')` term in the formula above is effectively 1 for the first font and no adjustment occurs. If the value is specified inaccurately, text rendered using the first font in the family list will display differently in older user agents that don't support '`font-size-adjust`'.

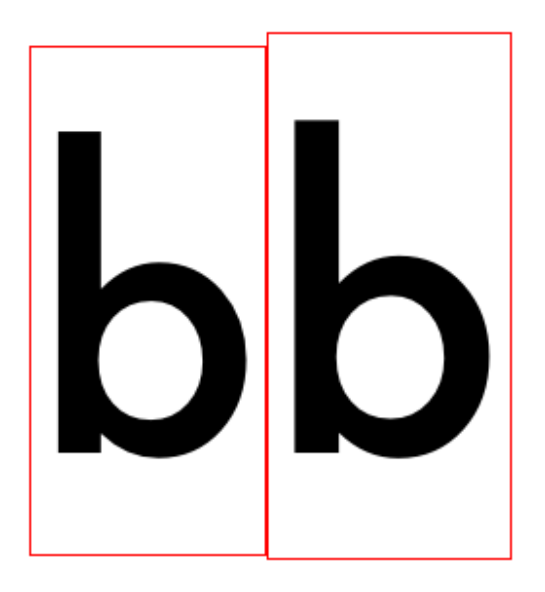
The value of '`font-size-adjust`' affects the used value of '`font-size`' but does not affect the computed value. It affects the size of relative units that are based on font metrics of the first available font such as `ex` and `ch` but does not affect the size of `em` units. Since numeric values of '`line-height`' refer to the computed size of '`font-size`', '`font-size-adjust`' does not affect the used value of '`line-height`'.

> In CSS, authors often specify '`line-height`' as a multiple of the '`font-size`'. Since the '`font-size-adjust`' property affects the used value of '`font-size`', authors should take care setting the line height when '`font-size-adjust`' is used. Setting the line height too tightly can result in overlapping lines of text in this situation.

Authors can calculate the aspect value for a given font by comparing spans with the same content but different '`font-size-adjust`' properties. If the same font-size is used, the spans will match when the '`font-size-adjust`' value is accurate for the given font.

Two spans with borders are used to determine the aspect value of a font. The '`font-size`' is the same for both spans but the '`font-size-adjust`' property is specified only for the right span. Starting with a value of 0.5, the aspect value can be adjusted until the borders around the two letters line up.

```
p {
    font-family: Futura;
    font-size: 500px;
}

span {
    border: solid 1px red;
}

.adjust {
    font-size-adjust: 0.5;
}

<p><span>b</span><span class="adjust">b</span></p>
```



Futura with an aspect value of 0.5

The box on the right is a bit bigger than the one on the left, so the aspect value of this font is something less than 0.5. Adjust the value until the boxes align.

3.7. Shorthand font property: the font property

Name:	***font***
Value:	[[<'font-style'> || <font-variant-css21> || <'font-weight'> || <'font-stretch'>]? <'font-size'> [/ <'line-height'>]? <'font-family'>] | caption | icon | menu | message-box | small-caption | status-bar
Initial:	see individual properties
Applies to:	all elements
Inherited:	yes
Percentages:	see individual properties
Media:	visual
Computed value:	see individual properties
Animatable:	see individual properties

The 'font' property is, except as described below, a shorthand property for setting 'font-style', 'font-variant', 'font-weight', 'font-stretch', 'font-size', 'line-height', 'font-family' at the same place in the stylesheet. Values for the 'font-variant' property may also be included but only those supported in CSS 2.1, none of the 'font-variant' values added in this specification can be used in the 'font' shorthand:

```
<font-variant-css21> = [normal | small-caps]
```

The syntax of this property is based on a traditional typographical shorthand notation to set multiple properties related to fonts.

All subproperties of the 'font' property are first reset to their initial values, including those listed above plus 'font-size-adjust', 'font-kerning', all subproperties of 'font-variant', and 'font-feature-settings', but *not* 'font-synthesis'. Then, those properties that are given explicit values in the 'font' shorthand are set to those values. For a definition of allowed and initial values, see the previously defined properties. For reasons of backwards compatibility, it is not possible to set 'font-size-adjust' to anything other than its initial value using the 'font' shorthand property; instead, use the individual property.

```
p { font: 12pt/14pt sans-serif }
p { font: 80% sans-serif }
p { font: x-large/110% "new century schoolbook", serif }
p { font: bold italic large Palatino, serif }
p { font: normal small-caps 120%/120% fantasy }
p { font: condensed oblique 12pt "Helvetica Neue", serif; }
```

In the second rule, the font size percentage value ('80%') refers to the computed 'font-size' of the parent element. In the third rule, the line height percentage ('110%') refers to the font size of the element itself.

The first three rules do not specify the 'font-variant' and 'font-weight' explicitly, so these properties receive their initial values ('normal'). Notice that the font family name "new century schoolbook", which contains spaces, is enclosed in quotes. The fourth rule sets the 'font-weight' to 'bold', the 'font-style' to 'italic', and implicitly sets 'font-variant' to 'normal'.

The fifth rule sets the 'font-variant' ('small-caps'), the 'font-size' (120% of the parent's font size), the 'line-height' (120% of the font size) and the 'font-family' ('fantasy'). It follows that the keyword 'normal' applies to the two remaining properties: 'font-style' and 'font-weight'.

The sixth rule sets the 'font-style', 'font-stretch', 'font-size', and 'font-family', the other font properties being set to their initial values.

Since the 'font-stretch' property was not defined in CSS 2.1, when using 'font-stretch' values within 'font' rules, authors should include a extra version compatible with older user agents:

```
p {
  font: 80% sans-serif;   /* for older user agents */
  font: condensed 80% sans-serif;
}
```

The following values refer to system fonts:

caption
: The font used for captioned controls (e.g., buttons, drop-downs, etc.).

icon
: The font used to label icons.

menu

The font used in menus (e.g., dropdown menus and menu lists).

message-box
The font used in dialog boxes.

small-caption
The font used for labeling small controls.

status-bar
The font used in window status bars.

System fonts may only be set as a whole; that is, the font family, size, weight, style, etc. are all set at the same time. These values may then be altered individually if desired. If no font with the indicated characteristics exists on a given platform, the user agent should either intelligently substitute (e.g., a smaller version of the '`caption`' font might be used for the '`small-caption`' font), or substitute a user agent default font. As for regular fonts, if, for a system font, any of the individual properties are not part of the operating system's available user preferences, those properties should be set to their initial values.

That is why this property is "almost" a shorthand property: system fonts can only be specified with this property, not with '`font-family`' itself, so '`font`' allows authors to do more than the sum of its subproperties. However, the individual properties such as '`font-weight`' are still given values taken from the system font, which can be independently varied.

Note that the keywords used for the system fonts listed above are only treated as keywords when they occur in the initial position, in other positions the same string is treated as part of the font family name:

```
font: menu;        /* use the font settings for system menus */
font: large menu;  /* use a font family named "menu" */
```

```
button { font: 300 italic 1.3em/1.7em "FB Armada", sans-serif }
button p { font: menu }
button p em { font-weight: bolder }
```

If the font used for dropdown menus on a particular system happened to be, for example, 9-point Charcoal, with a weight of 600, then P elements that were descendants of BUTTON would be displayed as if this rule were in effect:

```
button p { font: 600 9pt Charcoal }
```

Because the ‘font’ shorthand resets to its initial value any property not explicitly given a value, this has the same effect as this declaration:

```
button p {
  font-style: normal;
  font-variant: normal;
  font-weight: 600;
  font-size: 9pt;
  line-height: normal;
  font-family: Charcoal
}
```

3.8. Controlling synthetic faces: the font-synthesis property

Name:	***font-synthesis***
Value:	none | [weight || style]
Initial:	weight style
Applies to:	all elements
Inherited:	yes
Percentages:	N/A
Media:	visual
Computed value:	as specified
Animatable:	no

This property controls whether user agents are allowed to synthesize bold or oblique font faces when a font family lacks bold or italic faces. If '`weight`' is not specified, user agents must not synthesize bold faces and if '`style`' is not specified user agents must not synthesize italic faces. A value of '`none`' disallows all synthetic faces.

The style rule below disables the use of synthetically obliqued Arabic:

```
*:lang(ar) { font-synthesis: none; }
```

4. Font Resources

4.1. The `@font-face` rule

The `@font-face` rule allows for linking to fonts that are automatically fetched and activated when needed. This allows authors to select a font that closely matches the design goals for a given page rather than limiting the font choice to a set of fonts available on a given platform. A set of font descriptors define the location of a font resource, either locally or externally, along with the style characteristics of an individual face. Multiple `@font-face` rules can be used to construct font families with a variety of faces. Using CSS font matching rules, a user agent can selectively download only those faces that are needed for a given piece of text.

The @font-face rule consists of the @font-face at-keyword followed by a block of descriptor declarations. In terms of the grammar, this specification defines the following productions:

```
font_face_rule
  : FONT_FACE_SYM S* '{' S* descriptor_declaration? [ ';' S* descriptor_declaration? ]
  ;

descriptor_declaration
  : property ':' S* expr
  ;
```

The following new definitions are introduced:

```
-    -|\\0{0,4}2d(\r\n|[ \t\r\n\f])?
F    f|\\0{0,4}(46|66)(\r\n|[ \t\r\n\f])?
```

The following new token is introduced:

```
@{F}{O}{N}{T}{-}{F}{A}{C}{E} {return FONT_FACE_SYM;}
```

Each @font-face rule specifies a value for every font descriptor, either implicitly or explicitly. Those not given explicit values in the rule take the initial value listed with each descriptor in this specification. These descriptors apply solely within the context of the @font-face rule in which they are defined, and do not apply to document language elements. There is no notion of which elements the descriptors apply to or whether the values are inherited by child elements. When a given descriptor occurs multiple times in a given @font-face rule, only the last descriptor declaration is used and all prior declarations for that descriptor are ignored.

> To use a downloadable font called Gentium:
>
> ```
> @font-face {
>   font-family: Gentium;
>   src: url(http://example.com/fonts/Gentium.woff);
> }
>
> p { font-family: Gentium, serif; }
> ```
>
> The user agent will download Gentium and use it when rendering text within paragraph elements. If for some reason the site serving the font is unavailable, the default serif font will be used.

A given set of @font-face rules define a set of fonts available for use within the documents that

contain these rules. When font matching is done, fonts defined using these rules are considered before other available fonts on a system.

Downloaded fonts are only available to documents that reference them. The process of activating these fonts must not make them available to other applications or to documents that don't directly link to the same font. User agent implementers might consider it convenient to use downloaded fonts when rendering characters in other documents for which no other available font exists as part of the *system font fallback* procedure. However, this would cause a security leak since the contents of one page would be able to affect other pages, something an attacker could use as an attack vector. These restrictions do not affect caching behavior, fonts are cached the same way other web resources are cached.

This at-rule follows the forward-compatible parsing rules of CSS. Like properties in a declaration block, declarations of any descriptors that are not supported by the user agent must be ignored. `@font-face` rules require a font-family and src descriptor; if either of these are missing, the `@font-face` rule must not be considered when performing the font matching algorithm.

In cases where user agents have limited platform resources or implement the ability to disable downloadable font resources, `@font-face` rules must simply be ignored; the behavior of individual descriptors as defined in this specification should not be altered.

4.2. Font family: the font-family descriptor

Name:	***font-family***
Value:	*<family-name>*
Initial:	N/A

This descriptor defines the font family name that will be used in all CSS font family name matching. It is required for the `@font-face` rule to be valid. It overrides the font family names contained in the underlying font data. If the font family name is the same as a font family available in a given user's environment, it effectively hides the underlying font for documents that use the stylesheet. This permits a web author to freely choose font-family names without worrying about conflicts with font family names present in a given user's environment. Likewise, platform substitutions for a given font family name must not be used.

4.3. Font reference: the src descriptor

Name:	***src***
Value:	[<url> [format(<string> #)]? | *<font-face-name>*] #
Initial:	N/A

This descriptor specifies the resource containing font data. It is required for the `@font-face` rule to be valid. Its value is a prioritized, comma-separated list of external references or locally-installed font face names. When a font is needed the user agent iterates over the set of references listed, using the first one it can successfully activate. Fonts containing invalid data or local font faces that are not found are ignored and the user agent loads the next font in the list.

As with other URLs in CSS, the URL may be relative, in which case it is resolved relative to the location of the style sheet containing the `@font-face` rule. In the case of SVG fonts, the URL points to an element within a document containing SVG font definitions. If the element reference is omitted, a reference to the first defined font is implied. Similarly, font container formats that can contain more than one font must load one and only one of the fonts for a given `@font-face` rule. Fragment identifiers are used to indicate which font to load; these use the PostScript name of the font as defined in [RFC8081]. Conformant user agents must skip downloading a font resource if the fragment identifier is unknown or unsupported. For example, older user agents which do not support OpenType collections will skip to the next url in the list.

```
src: url(fonts/simple.woff);       /* load simple.woff relative to stylesheet location
src: url(/fonts/simple.woff);      /* load simple.woff from absolute location */
src: url(fonts/coll.otc#foo);      /* load font foo from collection coll.otc
src: url(fonts/coll.woff2#foo);    /* load font foo from woff2 collection coll.woff2
src: url(fonts.svg#simple);        /* load SVG font with id 'simple' */
```

External references consist of a URL, followed by an optional hint describing the format of the font resource referenced by that URL. The format hint contains a comma-separated list of format strings that denote well-known font formats. Conformant user agents must skip downloading a font resource if the format hints indicate only unsupported or unknown font formats. If no format hints are supplied, the user agent should download the font resource.

```
/* load WOFF2 font if possible, otherwise WOFF, else use OpenType font */
@font-face {
  font-family: bodytext;
  src: url(ideal-sans-serif.woff2) format("woff2"),
       url(good-sans-serif.woff) format("woff"),
       url(basic-sans-serif.ttf) format("opentype");
}
```

Format strings defined by this specification:

String	Font Format	Common extensions
`"woff"`	WOFF 1.0 (Web Open Font Format)	.woff
`"woff2"`	WOFF 2.0 (Web Open Font Format)	.woff2
`"truetype"`	TrueType	.ttf
`"opentype"`	OpenType	.ttf, .otf
`"embedded-opentype"`	Embedded OpenType	.eot
`"svg"`	SVG Font	.svg, .svgz

Given the overlap in common usage between TrueType and OpenType [OPENTYPE], the format hints `"truetype"` and `"opentype"` must be considered as synonymous; a format hint of `"opentype"` does not imply that the font contains Postscript CFF style glyph data or that it contains OpenType layout information (see Appendix A for more background on this).

When authors would prefer to use a locally available copy of a given font and download it if it's not, `local()` can be used. The locally-installed **<font-face-name>** argument to `local()` is a format-specific string that uniquely identifies a single font face within a larger family. The syntax for a *<font-face-name>* is a unique font face name enclosed by `"local("` and `")"`. The name can optionally be enclosed in quotes. If unquoted, the unquoted font family name processing conventions apply; the name must be a sequence of identifiers separated by whitespace which is converted to a string by joining the identifiers together separated by a single space.

```
/* regular face of Gentium */
@font-face {
  font-family: MyGentium;
  src: local(Gentium),    /* use locally available Gentium */
       url(Gentium.woff); /* otherwise, download it */
}
```

For OpenType and TrueType fonts, this string is used to match only the Postscript name or the full font name in the name table of locally available fonts. Which type of name is used varies by platform and font, so authors should include both of these names to assure proper matching

across platforms. Platform substitutions for a given font name must not be used.

```
/* bold face of Gentium */
@font-face {
  font-family: MyGentium;
  src: local(Gentium Bold),    /* full font name */
       local(Gentium-Bold),    /* Postscript name */
       url(GentiumBold.woff);  /* otherwise, download it */
  font-weight: bold;
}
```

Just as a `@font-face` rule specifies the characteristics of a single font within a family, the unique name used with `local()` specifies a single font, not an entire font family. Defined in terms of OpenType font data, the Postscript name is found in the font's name table, in the name record with `nameID = 6` (see [OPENTYPE] for more details). The Postscript name is the commonly used key for all fonts on OSX and for Postscript CFF fonts under Windows. The full font name (`nameID = 4`) is used as a unique key for fonts with TrueType glyphs on Windows.

For OpenType fonts with multiple localizations of the full font name, the US English version is used (`language ID = 0x409` for Windows and `language ID = 0` for Macintosh) or the first localization when a US English full font name is not available (the OpenType specification recommends that all fonts minimally include US English names). User agents that also match other full font names, e.g. matching the Dutch name when the current system locale is set to Dutch, are considered non-conformant. This is done not to prefer English but to avoid matching inconsistencies across font versions and OS localizations, since font style names (e.g. "Bold") are frequently localized into many languages and the set of localizations available varies widely across platform and font version. User agents that match a concatenation of family name (`nameID = 1`) with style name (`nameID = 2`) are considered non-conformant.

This also allows for referencing faces that belong to larger families that cannot otherwise be referenced.

Use a local font or reference an SVG font in another document:

```
@font-face {
  font-family: Headline;
  src: local(Futura-Medium),
       url(fonts.svg#MyGeometricModern) format("svg");
}
```

Create an alias for local Japanese fonts on different platforms:

```
@font-face {
  font-family: jpgothic;
  src: local(HiraKakuPro-W3), local(Meiryo), local(IPAPGothic);
}
```

Reference a font face that cannot be matched within a larger family:

```
@font-face {
  font-family: Hoefler Text Ornaments;
  /* has the same font properties as Hoefler Text Regular */
  src: local(HoeflerText-Ornaments);
}
```

Since localized fullnames never match, a document with the header style rules below would always render using the default serif font, regardless whether a particular system locale parameter is set to Finnish or not:

```
@font-face {
  font-family: SectionHeader;
  src: local("Arial Lihavoitu");  /* Finnish fullname for Arial Bold, should fail
  font-weight: bold;
}

h2 { font-family: SectionHeader, serif; }
```

A conformant user agent would never load the font '`gentium.eot`' in the example below, since it is included in the first definition of the '`src`' descriptor which is overridden by the second definition in the same `@font-face` rule:

```
@font-face {
  font-family: MainText;
  src: url(gentium.eot);                    /* for use with older user agents */
  src: local("Gentium"), url(gentium.woff);  /* Overrides src definition */
}
```

4.4. Font property descriptors: the font-style, font-weight, font-stretch descriptors

Name:	***font-style***
Value:	normal | italic | oblique
Initial:	normal

Name:	***font-weight***
Value:	normal | bold | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900
Initial:	normal

Name:	***font-stretch***
Value:	normal | ultra-condensed | extra-condensed | condensed | semi-condensed | semi-expanded | expanded | extra-expanded | ultra-expanded
Initial:	normal

These descriptors define the characteristics of a font face and are used in the process of matching styles to specific faces. For a font family defined with several `@font-face` rules, user agents can either download all faces in the family or use these descriptors to selectively download font faces that match actual styles used in document. The values for these descriptors are the same as those for the corresponding font properties except that relative keywords are not allowed, '`bolder`' and '`lighter`'. If these descriptors are omitted, initial values are assumed.

The value for these font face style attributes is used in place of the style implied by the underlying font data. This allows authors to combine faces in flexible combinations, even in situations where the original font data was arranged differently. User agents that implement synthetic bolding and obliquing must only apply synthetic styling in cases where the font descriptors imply this is needed, rather than based on the style attributes implied by the font data.

The font descriptors defined in this section are used for selecting a font from within the set of fonts defined by `@font-face` rules for a given family.

Consider a family containing a single, regular face:

```
@font-face {
  font-family: BaskervilleSimple;
  src: url(baskerville-regular.woff);
}
```

Unstyled text would display using the regular face defined in the `@font-face` rule:

However, italic text would display in most user agents using synthetically obliqued glyphs from the regular face, since a separate italic face is not defined:

Now consider a family for which an actual italic face is defined:

```
@font-face {
  font-family: BaskervilleFull;
  src: url(baskerville-regular.woff);
}

@font-face {
  font-family: BaskervilleFull;
  src: url(baskerville-italic.woff);
  font-style: italic;
}
```

The second `@font-face` rule defines the font resource `baskerville-italic.woff` to have

style attributes of normal weight, normal stretch and italic style. When displaying italic text, the user agent will use this font, since it's the closest match for italic text. Thus, the text will display using glyphs designed by a type designer rather than using synthetically obliqued glyphs from the regular face:

See the section on font matching for more complete details of the process used to select a particular face within a font family.

4.5. Character range: the unicode-range descriptor

Name:	***unicode-range***
Value:	*<urange>* #
Initial:	U+0-10FFFF

This descriptor defines the set of Unicode [UNICODE] codepoints that may be supported by the font face for which it is declared. The descriptor value is a comma-delimited list of Unicode range (*<urange>*) values. The union of these ranges defines the set of codepoints that serves as a hint for user agents when deciding whether or not to download a font resource for a given text run.

Each **<urange>** value is a `UNICODE-RANGE` token made up of a "U+" or "u+" prefix followed by a codepoint range in one of the three forms listed below. Ranges that do not fit one of the these forms are invalid and cause the declaration to be ignored.

single codepoint (e.g. U+416)
: a Unicode codepoint, represented as one to six hexadecimal digits

interval range (e.g. U+400-4ff)
: represented as two hyphen-separated Unicode codepoints indicating the inclusive start and end codepoints of a range

wildcard range (e.g. U+4??)
: defined by the set of codepoints implied when trailing '?' characters signify any hexadeximal digit

Individual codepoints are written using hexadecimal values that correspond to Unicode character codepoints [UNICODE]. Unicode codepoint values must be between 0 and 10FFFF inclusive. Digit values of codepoints are ASCII case-insensitive. For interval ranges, the start and end codepoints must be within the range noted above and the end codepoint must be greater than or equal to the start codepoint.

Wildcard ranges specified with '?' that lack an initial digit (e.g. "U+???") are valid and equivalent to a wildcard range with an initial zero digit (e.g. "U+0???" = "U+0000-0FFF"). Wildcard ranges that extend beyond the range of Unicode codepoints are invalid. Because of this, the maximum number of trailing '?' wildcard characters is five, even though the `UNICODE-RANGE` token accepts six.

Within the comma-delimited list of Unicode ranges in a '`unicode-range`' descriptor declaration, ranges may overlap. The union of these ranges defines the set of codepoints for which the corresponding font may be used. User agents must not download or use the font for codepoints outside this set. User agents may normalize the list of ranges into a list that is different but represents the same set of codepoints.

The associated font might not contain glyphs for the entire set of codepoints defined by the '`unicode-range`' descriptor. When the font is used, the ***effective character map*** is the intersection of the codepoints defined by '`unicode-range`' with the font's *character map*. This allows authors to define supported ranges in terms of broad ranges without worrying about the precise codepoint ranges supported by the underlying font.

4.6. Using character ranges to define composite fonts

Multiple `@font-face` rules with different unicode ranges for the same family and style descriptor values can be used to create composite fonts that mix the glyphs from different fonts for different scripts. This can be used to combine fonts that only contain glyphs for a single script (e.g. Latin, Greek, Cyrillic) or it can be used by authors as a way of segmenting a font into fonts for commonly used characters and less frequently used characters. Since the user agent will only pull down the fonts it needs this helps reduce page bandwidth.

If the unicode ranges overlap for a set of `@font-face` rules with the same family and style descriptor values, the rules are ordered in the reverse order they were defined; the last rule defined is the first to be checked for a given character.

Example ranges for specific languages or characters:

unicode-range: U+A5;
: a single code point, the yen/yuan symbol

unicode-range: U+0-7F;
: code range for basic ASCII characters

unicode-range: U+590-5ff;
: code range for Hebrew characters

unicode-range: U+A5, U+4E00-9FFF, U+30??, U+FF00-FF9F;
: code range for Japanese kanji, hiragana and katakana characters plus yen/yuan symbol

> The BBC provides news services in a wide variety of languages, many that are not well supported across all platforms. Using an `@font-face` rule, the BBC could provide a font for any of these languages, as it already does via a manual font download.
>
> ```
> @font-face {
>   font-family: BBCBengali;
>   src: url(fonts/BBCBengali.woff) format("woff");
>   unicode-range: U+00-FF, U+980-9FF;
> }
> ```

> Technical documents often require a wide range of symbols. The STIX Fonts project is one project aimed at providing fonts to support a wide range of technical typesetting in a standardized way. The example below shows the use of a font that provides glyphs for many of the mathematical and technical symbol ranges within Unicode:
>
> ```
> @font-face {
>   font-family: STIXGeneral;
>   src: local(STIXGeneral), url(/stixfonts/STIXGeneral.otf);
>   unicode-range: U+000-49F, U+2000-27FF, U+2900-2BFF, U+1D400-1D7FF;
> }
> ```

This example shows how an author can override the glyphs used for Latin characters in a Japanese font with glyphs from a different font. The first rule specifies no range so it defaults to the entire range. The range specified in the second rule overlaps but takes precedence because it is defined later.

```
@font-face {
  font-family: JapaneseWithGentium;
  src: local(MSMincho);
  /* no range specified, defaults to entire range */
}

@font-face {
  font-family: JapaneseWithGentium;
  src: url(../fonts/Gentium.woff);
  unicode-range: U+0-2FF;
}
```

Consider a family constructed to optimize bandwidth by separating out Latin, Japanese and other characters into different font files:

```
/* fallback font - size: 4.5MB */
@font-face {
  font-family: DroidSans;
  src: url(DroidSansFallback.woff);
  /* no range specified, defaults to entire range */
}

/* Japanese glyphs - size: 1.2MB */
@font-face {
  font-family: DroidSans;
  src: url(DroidSansJapanese.woff);
  unicode-range: U+3000-9FFF, U+ff??;
}

/* Latin, Greek, Cyrillic along with some
   punctuation and symbols - size: 190KB */
@font-face {
  font-family: DroidSans;
  src: url(DroidSans.woff);
  unicode-range: U+000-5FF, U+1e00-1fff, U+2000-2300;
}
```

For simple Latin text, only the font for Latin characters is downloaded:

```
body { font-family: DroidSans; }

<p>This is that</p>
```

In this case the user agent first checks the unicode-range for the font containing Latin characters (DroidSans.woff). Since all the characters above are in the range U+0-5FF, the user agent downloads the font and renders the text with that font.

Next, consider text that makes use of an arrow character (⇨):

```
<p>This &#x21e8; that<p>
```

The user agent again first checks the unicode-range of the font containing Latin characters. Since U+2000-2300 includes the arrow code point (U+21E8), the user agent downloads the font. For this character however the Latin font does not have a matching glyph, so the effective unicode-range used for font matching excludes this code point.

Next, the user agent evaluates the Japanese font. The unicode-range for the Japanese font, U+3000-9FFF and U+ff??, does not include U+21E8, so the user agent does not download the Japanese font. Next the fallback font is considered. The `@font-face` rule for the fallback font does not define unicode-range so its value defaults to the range of all Unicode code points. The fallback font is downloaded and used to render the arrow character.

4.7. Font features: the font-feature-settings descriptor

Name:	***font-feature-settings***
Value:	normal | *<feature-tag-value>* #
Initial:	normal

This descriptor defines initial settings that apply when the font defined by an `@font-face` rule is rendered. It does not affect font selection. Values are identical to those defined for the corresponding '`font-feature-settings`' property defined below except that the value '`inherit`' is omitted. When multiple font feature descriptors or properties are used, the cumulative effect on text rendering is detailed in the section Font Feature Resolution below.

4.8. Font loading guidelines

The `@font-face` rule is designed to allow lazy loading of font resources that are only downloaded when used within a document. A stylesheet can include `@font-face` rules for a library of fonts of which only a select set are used; user agents must only download those fonts that are referred to within the style rules applicable to a given page. User agents that download all fonts defined in `@font-face` rules without considering whether those fonts are in fact used within a page are considered non-conformant. In cases where a font might be downloaded in character fallback cases, user agents may download a font if it's contained within the computed value of '`font-family`' for a given text run.

```
@font-face {
  font-family: GeometricModern;
  src: url(font.woff);
}

p {
  /* font will be downloaded for pages with p elements */
  font-family: GeometricModern, sans-serif;
}

h2 {
  /* font may be downloaded for pages with h2 elements, even if Futura is available lo
  font-family: Futura, GeometricModern, sans-serif;
}
```

In cases where textual content is loaded before downloadable fonts are available, user agents may render text as it would be rendered if downloadable font resources are not available or they may render text transparently with fallback fonts to avoid a flash of text using a fallback font. In cases where the font download fails user agents must display text, simply leaving transparent text is considered non-conformant behavior. Authors are advised to use fallback fonts in their font lists that closely match the metrics of the downloadable fonts to avoid large page reflows where possible.

4.9. Font fetching requirements

For font loads, user agents must use the potentially CORS-enabled fetch method defined by the [FETCH] specification for URL's defined within @font-face rules. When fetching, user agents must use "Anonymous" mode, set the referrer source to the stylesheet's URL and set the origin to the URL of the containing document.

> The implications of this for authors are that fonts will typically not be loaded cross-origin unless authors specifically takes steps to permit cross-origin loads. Sites can explicitly allow cross-site loading of font data using the `Access-Control-Allow-Origin` HTTP header. For other schemes, no explicit mechanism to allow cross-origin loading, beyond what is permitted by the potentially CORS-enabled fetch method, is defined or required.

For the examples given below, assume that a document is located at `https://example.com/page.html` and all URL's link to valid font resources supported by the user agent. Fonts defined with the '`src`' descriptor values below will be loaded:

```
/* same origin (i.e. domain, scheme, port match document) */
src: url(fonts/simple.woff);

/* data url's with no redirects are treated as same origin */
src: url("data:application/font-woff;base64,...");

/* cross origin, different domain */
/* Access-Control-Allow-Origin response header set to '*' */
src: url(http://another.example.com/fonts/simple.woff);
```

Fonts defined with the '`src`' descriptor values below will fail to load:

```
/* cross origin, different scheme */
/* no Access-Control-xxx headers in response */
src: url(https://example.com/fonts/simple.woff);

/* cross origin, different domain */
/* no Access-Control-xxx headers in response */
src: url(http://another.example.com/fonts/simple.woff);
```

5. Font Matching Algorithm

The algorithm below describes how fonts are associated with individual runs of text. For each character in the run a font family is chosen and a particular font face is selected containing a glyph for that character.

5.1. Case sensitivity of font family names

As part of the font matching algorithm outlined below, user agents must match font family names used in style rules with actual font family names contained in fonts available in a given environment or with font family names defined in `@font-face` rules. User agents must match these names case insensitively, using the "Default Caseless Matching" algorithm outlined in the Unicode specification [UNICODE]. This algorithm is detailed in section 3.13 entitled "Default Case Algorithms". Specifically, the algorithm must be applied without normalizing the strings involved and without applying any language-specific tailorings. The case folding method

specified by this algorithm uses the case mappings with status field 'C' or 'F' in the CaseFolding.txt file of the Unicode Character Database [UNICODE].

> For authors this means that font family names are matched case insensitively, whether those names exist in a platform font or in the `@font-face` rules contained in a stylesheet. Authors should take care to ensure that names use a character sequence consistent with the actual font family name, particularly when using combining characters such as diacritical marks. For example, a family name that contains a lowercase a (U+0061) followed by a combining ring (U+030A) will **not** match a name that looks identical but which uses the precomposed lowercase a-ring character (U+00E5) instead of the combining sequence.

> Implementors should take care to verify that a given caseless string comparison implementation uses this precise algorithm and not assume that a given platform string matching routine follows it, as many of these have locale-specific behavior or use some level of string normalization [UAX15].

5.2. Matching font styles

The procedure for choosing a font for a given character in a run of text consists of iterating over the font families named by the 'font-family' property, selecting a font face with the appropriate style based on other font properties and then determining whether a glyph exists for the given character. This is done using the ***character map*** of the font, data which maps characters to the default glyph for that character. A font is considered to ***support*** a given character if (1) the character is contained in the font's *character map* and (2) if required by the containing script, shaping information is available for that character.

Some legacy fonts may include a given character in the *character map* but lack the shaping information (e.g. OpenType layout tables or Graphite tables) necessary for correctly rendering text runs containing that character.

Codepoint sequences consisting of a base character followed by a sequence of combining characters are treated slightly differently, see the section on cluster matching below.

For this procedure, the ***default face*** for a given font family is defined to be the face that would be selected if all font style properties were set to their initial value.

1. Using the computed font property values for a given element, the user agent starts with the first family name specified by the 'font-family' property.
2. If the family name is a generic family keyword, the user agent looks up the appropriate font

family name to be used. User agents may choose the generic font family to use based on the language of the containing element or the Unicode range of the character.

3. For other family names, the user agent attempts to find the family name among fonts defined via `@font-face` rules and then among available system fonts, matching names with a case-insensitive comparison as outlined in the section above. On systems containing fonts with multiple localized font family names, user agents must match any of these names independent of the underlying system locale or platform API used. If the font resources defined for a given face in an `@font-face` rule are either not available or contain invalid font data, then the face should be treated as not present in the family. If no faces are present for a family defined via `@font-face` rules, the family should be treated as missing; matching a platform font with the same name must not occur in this case.

4. If a font family match occurs, the user agent assembles the set of font faces in that family and then narrows the set to a single face using other font properties in the order given below. A group of faces defined via `@font-face` rules with identical font descriptor values but differing '`unicode-range`' values are considered to be a single ***composite face*** for this step:

    a. '`font-stretch`' is tried first. If the matching set contains faces with width values matching the '`font-stretch`' value, faces with other width values are removed from the matching set. If there is no face that exactly matches the width value the nearest width is used instead. If the value of '`font-stretch`' is '`normal`' or one of the condensed values, narrower width values are checked first, then wider values. If the value of '`font-stretch`' is one of the expanded values, wider values are checked first, followed by narrower values. Once the closest matching width has been determined by this process, faces with other widths are removed from the matching set.

    b. '`font-style`' is tried next. If the value of '`font-style`' is '`italic`', italic faces are checked first, then oblique, then normal faces. If the value is '`oblique`', oblique faces are checked first, then italic faces and then normal faces. If the value is '`normal`', normal faces are checked first, then oblique faces, then italic faces. Faces with other style values are excluded from the matching set. User agents are permitted to distinguish between italic and oblique faces within platform font families but this is not required, so all italic or oblique faces may be treated as italic faces. However, within font families defined via `@font-face` rules, italic and oblique faces must be distinguished using the value of the '`font-style`' descriptor. For families that lack any italic or oblique faces, user agents may create artificial oblique faces, if this is permitted by the value of the '`font-synthesis`' property.

    c. '`font-weight`' is matched next, so it will always reduce the matching set to a single font face. If bolder/lighter relative weights are used, the effective weight is calculated based

on the inherited weight value, as described in the definition of the ‘font-weight’ property. Given the desired weight and the weights of faces in the matching set after the steps above, if the desired weight is available that face matches. Otherwise, a weight is chosen using the rules below:

- If the desired weight is less than 400, weights below the desired weight are checked in descending order followed by weights above the desired weight in ascending order until a match is found.
- If the desired weight is greater than 500, weights above the desired weight are checked in ascending order followed by weights below the desired weight in descending order until a match is found.
- If the desired weight is 400, 500 is checked first and then the rule for desired weights less than 400 is used.
- If the desired weight is 500, 400 is checked first and then the rule for desired weights less than 400 is used.

d. ‘font-size’ must be matched within a UA-dependent margin of tolerance. (Typically, sizes for scalable fonts are rounded to the nearest whole pixel, while the tolerance for bitmapped fonts could be as large as 20%.) Further computations, e.g., by ‘em’ values in other properties, are based on the ‘font-size’ value that is used, not the one that is specified.

5. If the matched face is defined via @font-face rules, user agents must use the procedure below to select a single font:

   a. If the font resource has not been loaded and the range of characters defined by the ‘unicode-range’ descriptor value includes the character in question, load the font.

   b. After downloading, if the *effective character map* supports the character in question, select that font.

   When the matched face is a *composite face*, user agents must use the procedure above on each of the faces in the *composite face* in reverse order of @font-face rule definition.

   While the download occurs, user agents may either wait until the font is downloaded or render once with substituted font metrics and render again once the font is downloaded.

6. If no matching face exists or the matched face does not contain a glyph for the character to be rendered, the next family name is selected and the previous three steps repeated. Glyphs from other faces in the family are not considered. The only exception is that user agents may optionally substitute a synthetically obliqued version of the *default face* if that face supports a given glyph and synthesis of these faces is permitted by the value of the

'font-synthesis' property. For example, a synthetic italic version of the regular face may be used if the italic face doesn't support glyphs for Arabic.

7. If there are no more font families to be evaluated and no matching face has been found, then the user agent performs a ***system font fallback*** procedure to find the best match for the character to be rendered. The result of this procedure may vary across user agents.

8. If a particular character cannot be displayed using any font, the user agent should indicate by some means that a character is not being displayed, displaying either a symbolic representation of the missing glyph (e.g. using a Last Resort Font) or using the missing character glyph from a default font.

Optimizations of this process are allowed provided that an implementation behaves as if the algorithm had been followed exactly. Matching occurs in a well-defined order to ensure that the results are as consistent as possible across user agents, given an identical set of available fonts and rendering technology.

The ***first available font***, used for example in the definition of *font-relative lengths* such as 'ex' and 'ch' or in the definition of the 'line-height' property, is defined to be the first available font that would match the U+0020 (space) character given font families in the 'font-family' list (or a user agent's default font if none are available).

5.3. Cluster matching

When text contains characters such as combining marks, ideally the base character should be rendered using the same font as the mark, this assures proper placement of the mark. For this reason, the font matching algorithm for clusters is more specialized than the general case of matching a single character by itself. For sequences containing variation selectors, which indicate the precise glyph to be used for a given character, user agents always attempt *system font fallback* to find the appropriate glyph before using the default glyph of the base character.

A sequence of codepoints containing combining mark or other modifiers is termed a grapheme cluster (see [CSS-TEXT-3] and [UAX29] for a more complete description). For a given cluster containing a base character, *b* and a sequence of combining characters *c1, c2…*, the entire cluster is matched using these steps:

1. For each family in the font list, a face is chosen using the style selection rules defined in the previous section.
    a. If all characters in the sequence *b + c1 + c2 …* are completely supported by the font, select this font for the sequence.
    b. If a sequence of multiple codepoints is canonically equivalent to a single character and

the font *supports* that character, select this font for the sequence and use the glyph associated with the canonically equiavlent character for the entire cluster.

2. If no font was found in the font list in step 1:
    a. If *c1* is a variation selector, system fallback must be used to find a font that *supports* the full sequence of *b* + *c1*. If no font on the system *supports* the full sequence, match the single character *b* using the normal procedure for matching single characters and ignore the variation selector. Note: a sequence with more than one variation selector must be treated as an encoding error and the trailing selectors must be ignored. [UNICODE]
    b. Otherwise, the user agent may optionally use system font fallback to match a font that *supports* the entire cluster.
3. If no font is found in step 2, use the matching sequence from step 1 to determine the longest sequence that is completely *supported* by a font in the font list and attempt to match the remaining combining characters separately using the rules for single characters.

5.4. Character handling issues

CSS font matching is always performed on text runs containing Unicode characters [UNICODE], so documents using legacy encodings are assumed to have been transcoded before matching fonts. For fonts containing *character maps* for both legacy encodings and Unicode, the contents of the legacy encoding *character map* must have no effect on the results of the font matching process.

The font matching process does not assume that text runs are in either normalized or denormalized form (see [CHARMOD-NORM] for more details). Fonts may only support precomposed forms and not the decomposed sequence of base character plus combining marks. Authors should always tailor their choice of fonts to their content, including whether that content contains normalized or denormalized character streams.

If a given character is a Private-Use Area Unicode codepoint, user agents must only match font families named in the '`font-family`' list that are not generic families. If none of the families named in the '`font-family`' list contain a glyph for that codepoint, user agents must display some form of missing glyph symbol for that character rather than attempting *system font fallback* for that codepoint. When matching the replacement character U+FFFD, user agents may skip the font matching process and immediately display some form of missing glyph symbol, they are not required to display the glyph from the font that would be selected by the font matching process.

In general, the fonts for a given family will all have the same or similar *character maps*. The

process outlined here is designed to handle even font families containing faces with widely variant *character maps*. However, authors are cautioned that the use of such families can lead to unexpected results.

5.5. Font matching changes since CSS 2.1

The algorithm above is different from CSS 2.1 in a number of key places. These changes were made to better reflect actual font matching behavior across user agent implementations.

Differences compared to the font matching algorithm in CSS 2.1:

- The algorithm includes font-stretch matching.
- All possible font-style matching scenarios are delineated.
- Small-caps fonts are not matched as part of the font matching process, they are now handled via font features.
- Unicode variation selector matching is required.
- Cluster sequences are matched as a unit.

5.6. Font matching examples

> It's useful to note that the CSS selector syntax may be used to create language-sensitive typography. For example, some Chinese and Japanese characters are unified to have the same Unicode code point, although the abstract glyphs are not the same in the two languages.
>
> ```
> *:lang(ja) { font: 900 14pt/16pt "Heisei Mincho W9", serif; }
> *:lang(zh-Hant-TW) { font: 800 14pt/16.5pt "Li Sung", serif; }
> ```
>
> This selects any element that has the given language — Japanese or Traditional Chinese as used in Taiwan — and uses the appropriate font.

6. Font Feature Properties

Modern font technologies support a variety of advanced typographic and language-specific font features. Using these features, a single font can provide glyphs for a wide range of ligatures,

contextual and stylistic alternates, tabular and old-style figures, small capitals, automatic fractions, swashes, and alternates specific to a given language. To allow authors control over these font capabilities, the '`font-variant`' property has been expanded for CSS3. It now functions as a shorthand for a set of properties that provide control over stylistic font features.

6.1. Glyph selection and positioning

This section is non-normative

Simple fonts used for displaying Latin text use a very basic processing model. Fonts contain a *character map* which maps each character to a glyph for that character. Glyphs for subsequent characters are simply placed one after the other along a run of text. Modern font formats such as OpenType and AAT (Apple Advanced Typography) use a richer processing model. The glyph for a given character can be chosen and positioned not just based on the codepoint of the character itself, but also on adjacent characters as well as the language, script, and features enabled for the text. Font features may be required for specific scripts, or recommended as enabled by default or they might be stylistic features meant to be used under author control. The point at which font selection and positioning happens in the overall order of text processing operations (such as text transformation, text orientation and text alignment) is described in [CSS-TEXT-3], § Text Processing Order of Operations.

For a good visual overview of these features, see the [OPENTYPE-FONT-GUIDE]. For a detailed description of glyph processing for OpenType fonts, see [WINDOWS-GLYPH-PROC].

Stylistic font features can be classified into two broad categories: ones that affect the harmonization of glyph shapes with the surrounding context, such as kerning and ligature features, and ones such as the small-caps, subscript/superscript and alternate features that affect shape selection.

The subproperties of '`font-variant`' listed below are used to control these stylistic font features. They do not control features that are required for displaying certain scripts, such as the OpenType features used when displaying Arabic or Indic language text. They affect glyph selection and positioning, but do not affect font selection as described in the font matching section (except in cases required for compatibility with CSS 2.1).

To assure consistent behavior across user agents, the equivalent OpenType property settings are listed for individual properties and are normative. When using other font formats these should be used as a guideline to map CSS font feature property values to specific font features.

6.2. Language-specific display

OpenType also supports language-specific glyph selection and positioning, so that text can be displayed correctly in cases where the language dictates a specific display behavior. Many languages share a common script, but the shape of certain letters can vary across those languages. For example, certain Cyrillic letters have different shapes in Russian text than in Bulgarian. In Latin text, it's common to render "fi" with an explicit fi-ligature that lacks a dot on the "i". However, in languages such as Turkish which uses both a dotted-i and a dotless-i, it's important to not use this ligature or use a specialized version that contains a dot over the "i". The example below shows language-specific variations based on stylistic traditions found in Spanish, Italian and French orthography:

If the content language of the element is known according to the rules of the document language, user agents are required to infer the OpenType language system from the content language and use that when selecting and positioning glyphs using an OpenType font.

6.3. Kerning: the font-kerning property

Name:	***font-kerning***
Value:	auto | normal | none
Initial:	auto
Applies to:	all elements
Inherited:	yes
Percentages:	N/A
Media:	visual
Computed value:	as specified
Animatable:	no

Kerning is the contextual adjustment of inter-glyph spacing. This property controls metric kerning, kerning that utilizes adjustment data contained in the font.

auto
: Specifies that kerning is applied at the discretion of the user agent

normal
: Specifies that kerning is applied

none
: Specifies that kerning is not applied

For fonts that do not include kerning data this property will have no visible effect. When rendering with OpenType fonts, the [OPENTYPE] specification suggests that kerning be enabled by default. When kerning is enabled, the OpenType `kern` feature is enabled (for vertical text runs the `vkrn` feature is enabled instead). User agents must also support fonts that only support kerning via data contained in a `kern` font table, as detailed in the OpenType specification. If the '`letter-spacing`' property is defined, kerning adjustments are considered part of the default spacing and letter spacing adjustments are made after kerning has been applied.

When set to '`auto`', user agents can determine whether to apply kerning or not based on a number of factors: text size, script, or other factors that influence text processing speed. Authors who want proper kerning should use '`normal`' to explicitly enable kerning. Likewise, some authors may prefer to disable kerning in situations where performance is more important than precise appearance. However, in well-designed modern implementations the use of kerning

generally does not have a large impact on text rendering speed.

6.4. Ligatures: the font-variant-ligatures property

Name:	***font-variant-ligatures***
Value:	normal | none | [*<common-lig-values>* || *<discretionary-lig-values>* || *<historical-lig-values>* || *<contextual-alt-values>*]
Initial:	normal
Applies to:	all elements
Inherited:	yes
Percentages:	N/A
Media:	visual
Computed value:	as specified
Animatable:	no

Ligatures and contextual forms are ways of combining glyphs to produce more harmonized forms.

<common-lig-values> = [common-ligatures | no-common-ligatures]

<discretionary-lig-values> = [discretionary-ligatures | no-discretionary-ligatures]

<historical-lig-values> = [historical-ligatures | no-historical-ligatures]

<contextual-alt-values> = [contextual | no-contextual]

Individual values have the following meanings:

normal
: A value of 'normal' specifies that common default features are enabled, as described in detail in the next section. For OpenType fonts, common ligatures and contextual forms are on by default, discretionary and historical ligatures are not.

none

Specifies that all types of ligatures and contextual forms covered by this property are explicitly disabled. In situations where ligatures are not considered necessary, this may improve the speed of text rendering.

common-ligatures
Enables display of common ligatures (OpenType features: `liga, clig`). For OpenType fonts, common ligatures are enabled by default.

no-common-ligatures
Disables display of common ligatures (OpenType features: `liga, clig`).

discretionary-ligatures
Enables display of discretionary ligatures (OpenType feature: `dlig`). Which ligatures are discretionary or optional is decided by the type designer, so authors will need to refer to the documentation of a given font to understand which ligatures are considered discretionary.

no-discretionary-ligatures
Disables display of discretionary ligatures (OpenType feature: `dlig`).

historical-ligatures
Enables display of historical ligatures (OpenType feature: `hlig`).

no-historical-ligatures
Disables display of historical ligatures (OpenType feature: `hlig`).

contextual
Enables display of contextual alternates (OpenType feature: `calt`). Although not strictly a ligature feature, like ligatures this feature is commonly used to harmonize the shapes of glyphs with the surrounding context. For OpenType fonts, this feature is on by default.

no-contextual

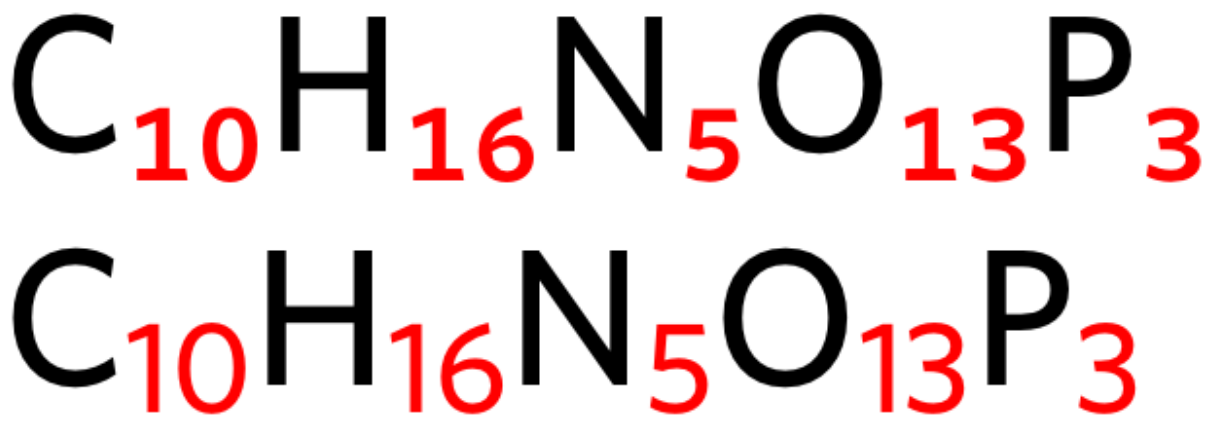
Disables display of contextual alternates (OpenType feature: `calt`).

Required ligatures, needed for correctly rendering complex scripts, are not affected by the settings above, including 'none' (OpenType feature: `rlig`).

6.5. Subscript and superscript forms: the font-variant-position property

Name:	***font-variant-position***
Value:	normal | sub | super
Initial:	normal
Applies to:	all elements
Inherited:	yes
Percentages:	N/A
Media:	visual
Computed value:	as specified
Animatable:	no

This property is used to enable typographic subscript and superscript glyphs. These are alternate glyphs designed within the same em-box as default glyphs and are intended to be laid out on the same baseline as the default glyphs, with no resizing or repositioning of the baseline. They are explicitly designed to match the surrounding text and to be more readable without affecting the line height.



Subscript glyphs (top) vs. typical synthesized subscripts (bottom)

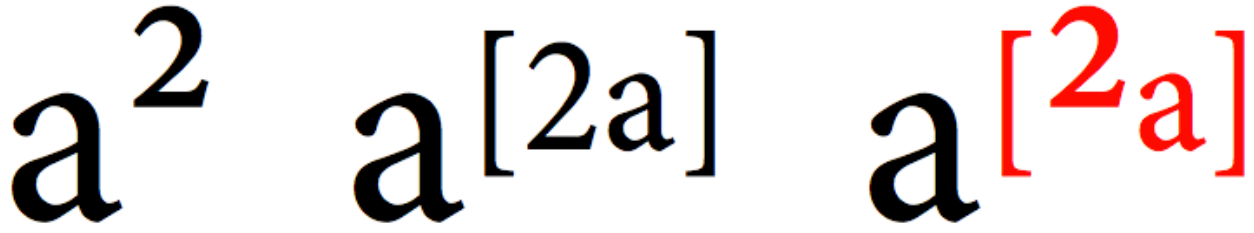
Individual values have the following meanings:

normal
: None of the features listed below are enabled.

sub
: Enables display of subscript variants (OpenType feature: subs).

super
: Enables display of superscript variants (OpenType feature: `sups`).

Because of the semantic nature of subscripts and superscripts, when the value is either '`sub`' or '`super`' for a given contiguous run of text, if a variant glyph is not available for all the characters in the run, simulated glyphs should be synthesized for all characters using reduced forms of the glyphs that would be used without this feature applied. This is done per run to avoid a mixture of variant glyphs and synthesized ones that would not align correctly. In the case of OpenType fonts that lack subscript or superscript glyphs for a given character, user agents *must* synthesize appropriate subscript and superscript glyphs.



Superscript alternate glyph (left), synthesized superscript glyphs (middle), and incorrect mixture of the two (right)

In situations where text decorations are only applied to runs of text containing superscript or subscript glyphs, the synthesized glyphs may be used, to avoid problems with the placement of decorations.

In the past, user agents have used font-size and vertical-align to simulate subscripts and superscripts for the `sub` and `sup` elements. To allow a backwards compatible way of defining subscripts and superscripts, it is recommended that authors use conditional rules [CSS3-CONDITIONAL] so that older user agents will still render subscripts and superscripts via the older mechanism.

Because `font-size: smaller` is often used for these elements, the effective scaling factor applied to subscript and superscript text varies depending upon the size. For larger text, the font size is often reduced by a third but for smaller text sizes, the reduction can be much less. This allows subscripts and superscripts to remain readable even within elements using small text sizes. User agents should consider this when deciding how to synthesize subscript and superscript glyphs.

> The OpenType font format defines subscript and superscript metrics in the OS/2 table [OPENTYPE] but these are not always accurate in practice and so cannot be relied upon when synthesizing subscript and superscript glyphs.

Authors should note that fonts typically only provide subscript and superscript glyphs for a subset of all characters supported by the font. For example, while subscript and superscript glyphs are often available for Latin numbers, glyphs for punctuation and letter characters are less frequently provided. The synthetic fallback rules defined for this property assure that subscripts and superscripts will always appear but the appearance may not match author expectations if the font used does not provide the appropriate alternate glyph for all characters contained in a subscript or superscript.

This property is not cumulative. Applying it to elements within a subscript or superscript won't nest the placement of a subscript or superscript glyph. Images contained within text runs where the value of this property is '`sub`' or '`super`' will be drawn just as they would if the value was '`normal`'.

Because of these limitations, '`font-variant-position`' is not recommended for use in user agent stylesheets. Authors should use it in cases where subscripts or superscripts will only contain the narrow range of characters supported by the fonts specified.

The variant glyphs use the same baseline as the default glyphs would use. There is no shift in the placement along the baseline, so the use of variant glyphs doesn't affect the height of the inline box or alter the height of the linebox. This makes superscript and subscript variants ideal for situations where it's important that leading remain constant, such as in multi-column layout.

A typical user agent default style for the sub element:

```
sub {
  vertical-align: sub;
  font-size: smaller;
  line-height: normal;
}
```

Using '`font-variant-position`' to specify typographic subscripts in a way that will still show subscripts in older user agents:

```
@supports ( font-variant-position: sub ) {

  sub {
    vertical-align: baseline;
    font-size: 100%;
    line-height: inherit;
    font-variant-position: sub;
  }

}
```

User agents that support the '`font-variant-position`' property will select a subscript variant glyph and render this without adjusting the baseline or font-size. Older user agents will ignore the '`font-variant-position`' property definition and use the standard defaults for subscripts.

6.6. Capitalization: the font-variant-caps property

Name:	***font-variant-caps***
Value:	normal | small-caps | all-small-caps | petite-caps | all-petite-caps | unicase | titling-caps
Initial:	normal
Applies to:	all elements
Inherited:	yes
Percentages:	N/A
Media:	visual
Computed value:	as specified
Animatable:	no

This property allows the selection of alternate glyphs used for small or petite capitals or for titling. These glyphs are specifically designed to blend well with the surrounding normal glyphs, to maintain the weight and readability which suffers when text is simply resized to fit this purpose.

Individual values have the following meanings:

normal
: None of the features listed below are enabled.

small-caps
: Enables display of small capitals (OpenType feature: `smcp`). Small-caps glyphs typically use the form of uppercase letters but are reduced to the size of lowercase letters.

all-small-caps
: Enables display of small capitals for both upper and lowercase letters (OpenType features: `c2sc, smcp`).

petite-caps
: Enables display of petite capitals (OpenType feature: `pcap`).

all-petite-caps

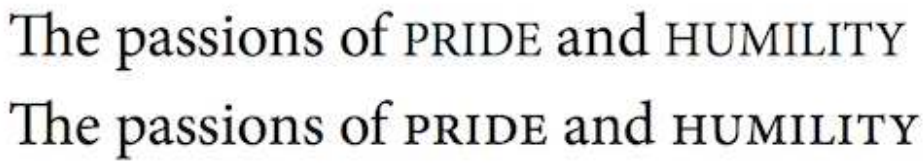
Enables display of petite capitals for both upper and lowercase letters (OpenType features: `c2pc, pcap`).

unicase
Enables display of mixture of small capitals for uppercase letters with normal lowercase letters (OpenType feature: `unic`).

titling-caps
Enables display of titling capitals (OpenType feature: `titl`). Uppercase letter glyphs are often designed for use with lowercase letters. When used in all uppercase titling sequences they can appear too strong. Titling capitals are designed specifically for this situation.

The availability of these glyphs is based on whether a given feature is defined or not in the feature list of the font. User agents can optionally decide this on a per-script basis but should explicitly not decide this on a per-character basis.

Some fonts may only support a subset or none of the features described for this property. For backwards compatibility with CSS 2.1, if '`small-caps`' or '`all-small-caps`' is specified but small-caps glyphs are not available for a given font, user agents should simulate a small-caps font, for example by taking a normal font and replacing the glyphs for lowercase letters with scaled versions of the glyphs for uppercase characters (replacing the glyphs for both upper and lowercase letters in the case of '`all-small-caps`').



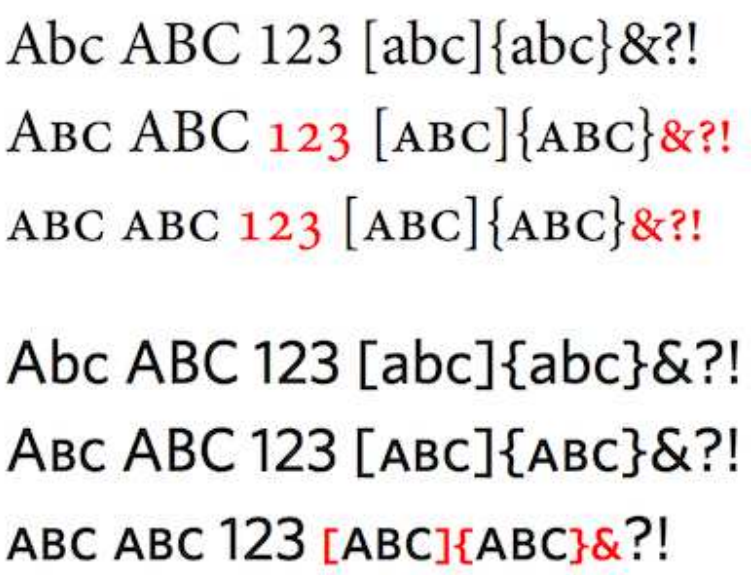
Synthetic vs. real small-caps

The '`font-feature-settings`' property does not affect the decision of whether or not to use a simulated small-caps font.

> ```
> #example1 { font-variant-caps: small-caps; }
> #example2 { font-variant-caps: small-caps; font-feature-settings: 'smcp' 0; }
> ```
>
> For fonts which don't support small caps, both #example1 and #example2 should be rendered with synthesized small caps. However, for fonts which do support small caps, #example1 should be rendered with native small caps, while #example2 should be rendered without any small-caps (native or synthesized).

To match the surrounding text, a font may provide alternate glyphs for caseless characters

when these features are enabled but when a user agent simulates small capitals, it must not attempt to simulate alternates for codepoints which are considered caseless.

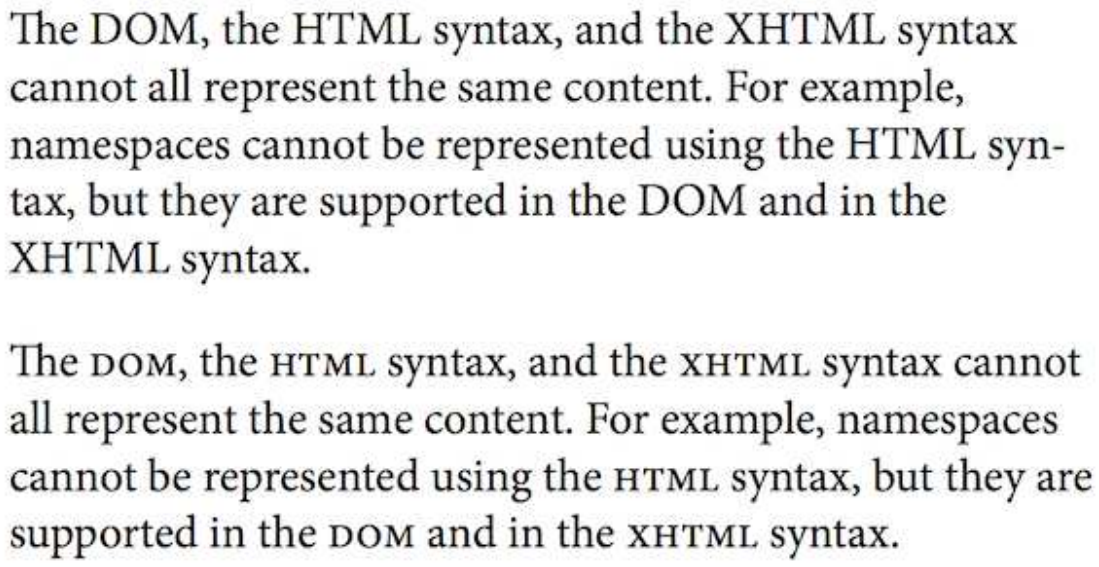


Caseless characters with small-caps, all-small-caps enabled

If either '`petite-caps`' or '`all-petite-caps`' is specified for a font that doesn't support these features, the property behaves as if '`small-caps`' or '`all-small-caps`', respectively, had been specified. If '`unicase`' is specified for a font that doesn't support that feature, the property behaves as if '`small-caps`' was applied only to lowercased uppercase letters. If '`titling-caps`' is specified with a font that does not support this feature, this property has no visible effect. When simulated small capital glyphs are used, for scripts that lack uppercase and lowercase letters, '`small-caps`', '`all-small-caps`', '`petite-caps`', '`all-petite-caps`' and '`unicase`' have no visible effect.

When casing transforms are used to simulate small capitals, the casing transformations must match those used for the '`text-transform`' property.

As a last resort, unscaled uppercase letter glyphs in a normal font may replace glyphs in a small-caps font so that the text appears in all uppercase letters.



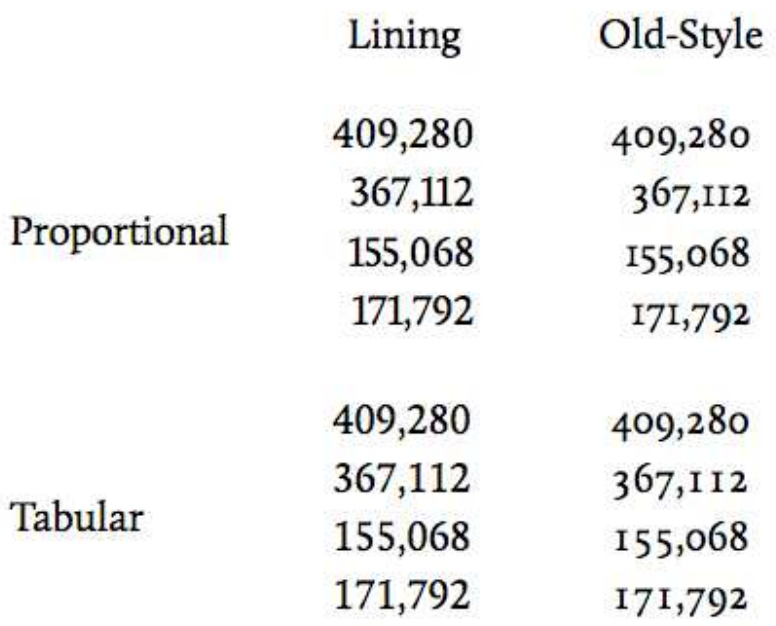
Using small capitals to improve readability in acronym-laden text

> Quotes rendered italicised, with small-caps on the first line:
>
> ```
> blockquote            { font-style: italic; }
> blockquote:first-line { font-variant: small-caps; }
>
> <blockquote>I'll be honor-bound to slap them like a haddock.</blockquote>
> ```

6.7. Numerical formatting: the font-variant-numeric property

Name:	***font-variant-numeric***
Value:	normal | [*<numeric-figure-values>* || *<numeric-spacing-values>* || *<numeric-fraction-values>* || ordinal || slashed-zero]
Initial:	normal
Applies to:	all elements
Inherited:	yes
Percentages:	N/A
Media:	visual
Computed value:	as specified
Animatable:	no

Specifies control over numerical forms. The example below shows how some of these values can be combined to influence the rendering of tabular data with fonts that support these features. Within normal paragraph text, proportional numbers are used while tabular numbers are used so that columns of numbers line up properly:



Using number styles

Possible combinations:

```
<numeric-figure-values>   = [ lining-nums | oldstyle-nums ]
<numeric-spacing-values>  = [ proportional-nums | tabular-nums ]
<numeric-fraction-values> = [ diagonal-fractions | stacked-fractions ]
```

Individual values have the following meanings:

normal
None of the features listed below are enabled.

lining-nums
Enables display of lining numerals (OpenType feature: `lnum`).

oldstyle-nums
Enables display of old-style numerals (OpenType feature: `onum`).

proportional-nums
Enables display of proportional numerals (OpenType feature: `pnum`).

tabular-nums
Enables display of tabular numerals (OpenType feature: `tnum`).

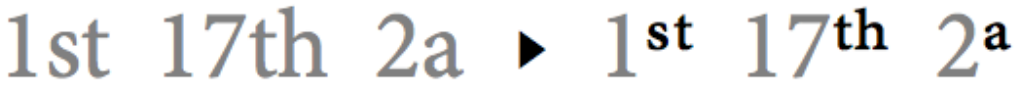
diagonal-fractions
Enables display of lining diagonal fractions (OpenType feature: `frac`).

stacked-fractions

Enables display of lining stacked fractions (OpenType feature: `afrc`).

ordinal

Enables display of letter forms used with ordinal numbers (OpenType feature: `ordn`).



slashed-zero

Enables display of slashed zeros (OpenType feature: `zero`).

> In the case of '`ordinal`', although ordinal forms are often the same as superscript forms, they are marked up differently.
>
> For superscripts, the variant property is only applied to the sub-element containing the superscript:
>
> ```
> sup { font-variant-position: super; }
> x<sup>2</sup>
> ```
>
> For ordinals, the variant property is applied to the entire ordinal number rather than just to the suffix (or to the containing paragraph):
>
> ```
> .ordinal { font-variant-numeric: ordinal; }
> 17th
> ```
>
> In this case only the "th" will appear in ordinal form, the digits will remain unchanged. Depending upon the typographic traditions used in a given language, ordinal forms may differ from superscript forms. In Italian, for example, ordinal forms sometimes include an underline in the ordinal design.

A simple flank steak marinade recipe, rendered with automatic fractions and old-style numerals:

```
.amount { font-variant-numeric: oldstyle-nums diagonal-fractions; }

<h4>Steak marinade:</h4>
<ul>
  <li><span class="amount">2</span> tbsp olive oil</li>
  <li><span class="amount">1</span> tbsp lemon juice</li>
  <li><span class="amount">1</span> tbsp soy sauce</li>
  <li><span class="amount">1 1/2</span> tbsp dry minced onion</li>
  <li><span class="amount">2 1/2</span> tsp italian seasoning</li>
  <li>Salt &amp; pepper</li>
</ul>

<p>Mix the meat with the marinade and let it sit covered in the refrigerator
for a few hours or overnight.</p>
```

Note that the fraction feature is only applied to values not the entire paragraph. Fonts often implement this feature using contextual rules based on the use of the slash (‘/’) character. As such, it's not suitable for use as a paragraph-level style.

6.8. East Asian text rendering: the font-variant-east-asian property

Name:	***font-variant-east-asian***
Value:	normal | [*<east-asian-variant-values>* || *<east-asian-width-values>* || ruby]
Initial:	normal
Applies to:	all elements
Inherited:	yes
Percentages:	N/A
Media:	visual
Computed value:	as specified
Animatable:	no

Allows control of glyph substitution and sizing in East Asian text.

<east-asian-variant-values> = [jis78 | jis83 | jis90 | jis04 | simplified | traditional]

<east-asian-width-values> = [full-width | proportional-width]

Individual values have the following meanings:

normal
None of the features listed below are enabled.

jis78
Enables rendering of JIS78 forms (OpenType feature: `jp78`).

jis83
Enables rendering of JIS83 forms (OpenType feature: `jp83`).

jis90
Enables rendering of JIS90 forms (OpenType feature: `jp90`).

jis04

Enables rendering of JIS2004 forms (OpenType feature: `jp04`).

The various JIS variants reflect the glyph forms defined in different Japanese national standards. Fonts generally include glyphs defined by the most recent national standard but it's sometimes necessary to use older variants, to match signage for example.

simplified
Enables rendering of simplified forms (OpenType feature: `smpl`).

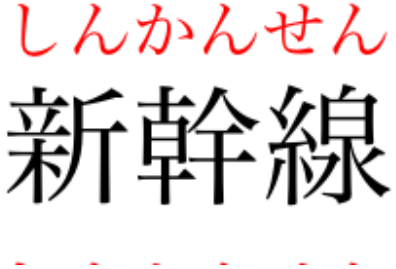
traditional
Enables rendering of traditional forms (OpenType feature: `trad`).

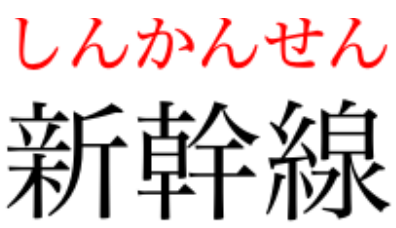
The '`simplified`' and '`traditional`' values allow control over the glyph forms for characters which have been simplified over time but for which the older, traditional form is still used in some contexts. The exact set of characters and glyph forms will vary to some degree by context for which a given font was designed.

full-width
Enables rendering of full-width variants (OpenType feature: `fwid`).

proportional-width
Enables rendering of proportionally-spaced variants (OpenType feature: `pwid`).

ruby
Enables display of ruby variant glyphs (OpenType feature: ruby). Since ruby text is generally smaller than the associated body text, font designers can design special glyphs for use with ruby that are more readable than scaled down versions of the default glyphs. Only glyph selection is affected, there is no associated font scaling or other change that affects line layout. The red ruby text below is shown with default glyphs (top) and with ruby variant glyphs (bottom). Note the slight difference in stroke thickness.





6.9. Overall shorthand for font rendering: the font-variant property

Name:	***font-variant***
Value:	normal | none | [*<common-lig-values>* || *<discretionary-lig-values>* || *<historical-lig-values>* || *<contextual-alt-values>* || [small-caps | all-small-caps | petite-caps | all-petite-caps | unicase | titling-caps] || *<numeric-figure-values>* || *<numeric-spacing-values>* || *<numeric-fraction-values>* || ordinal || slashed-zero || *<east-asian-variant-values>* || *<east-asian-width-values>* || ruby || [sub | super]]
Initial:	normal
Applies to:	all elements
Inherited:	yes
Percentages:	see individual properties
Media:	visual
Computed value:	see individual properties
Animatable:	see individual properties

The ‘`font-variant`’ property is a shorthand for all font-variant subproperties. The value ***‘normal’*** resets all subproperties of ‘`font-variant`’ to their inital value. The ***‘none’*** value sets ‘`font-variant-ligatures`’ to ‘none’ and resets all other font feature properties to their initial value. Like other shorthands, using ‘`font-variant`’ resets unspecified ‘`font-variant`’ subproperties to their initial values. It does not reset the values of ‘`font-feature-settings`’.

6.10. Low-level font feature settings control: the font-feature-settings property

Name:	***font-feature-settings***
Value:	normal | *<feature-tag-value>* #
Initial:	normal
Applies to:	all elements
Inherited:	yes
Percentages:	N/A
Media:	visual
Computed value:	as specified
Animatable:	no

This property provides low-level control over OpenType font features. It is intended as a way of providing access to font features that are not widely used but are needed for a particular use case.

> Authors should generally use ‘`font-variant`’ and its related subproperties whenever possible and only use this property for special cases where its use is the only way of accessing a particular infrequently used font feature.

```
/* enable small caps and use second swash alternate */
font-feature-settings: "smcp", "swsh" 2;
```

A value of ***‘normal’*** means that no change in glyph selection or positioning occurs due to this property.

Feature tag values have the following syntax:

```
<feature-tag-value> = <string> [ <integer> | on | off ]?
```

The <string> is a case-sensitive OpenType feature tag. As specified in the OpenType specification [OPENTYPE], feature tags contain four ASCII characters. Tag strings longer or shorter than four characters, or containing characters outside the U+20–7E codepoint range are invalid. Feature tags need only match a feature tag defined in the font, so they are not limited to explicitly registered OpenType features. Fonts defining custom feature tags should follow the tag name rules defined in the OpenType specification [OPENTYPE-FEATURES].

Feature tags not present in the font are ignored; a user agent must not attempt to synthesize fallback behavior based on these feature tags. The one exception is that user agents may synthetically support the `kern` feature with fonts that contain kerning data in the form of a 'kern' table but lack `kern` feature support in the 'GPOS' table.

> In general, authors should use the '`font-kerning`' property to explicitly enable or disable kerning since this property always affects fonts with either type of kerning data.

If present, a value indicates an index used for glyph selection. An <integer> value must be 0 or greater. A value of 0 indicates that the feature is disabled. For boolean features, a value of 1 enables the feature. For non-boolean features, a value of 1 or greater enables the feature and indicates the feature selection index. A value of 'on' is synonymous with 1 and 'off' is synonymous with 0. If the value is omitted, a value of 1 is assumed.

The computed value of font-feature-settings is a map, so any duplicates in the specified value must not be preserved. If the same axis name appears more than once, the value associated with the last appearance supersedes any previous value for that axis.

```
font-feature-settings: "dlig" 1;       /* dlig=1 enable discretionary ligatures */
font-feature-settings: "smcp" on;      /* smcp=1 enable small caps */
font-feature-settings: 'c2sc';         /* c2sc=1 enable caps to small caps */
font-feature-settings: "liga" off;     /* liga=0 no common ligatures */
font-feature-settings: "tnum", 'hist'; /* tnum=1, hist=1 enable tabular numbers and hi
font-feature-settings: "tnum" "hist";  /* invalid, need a comma-delimited list */
font-feature-settings: "silly" off;    /* invalid, tag too long */
font-feature-settings: "PKRN";         /* PKRN=1 enable custom feature */
font-feature-settings: dlig;           /* invalid, tag must be a string */
```

When values greater than the range supported by the font are specified, the behavior is explicitly undefined. For boolean features, in general these will enable the feature. For non-boolean features, out of range values will in general be equivalent to a 0 value. However, in both cases the exact behavior will depend upon the way the font is designed (specifically, which type of lookup is used to define the feature).

Although specifically defined for OpenType feature tags, feature tags for other modern font formats that support font features may be added in the future. Where possible, features defined for other font formats should attempt to follow the pattern of registered OpenType tags.

> The Japanese text below will be rendered with half-width kana characters:
>
> ```
> body { font-feature-settings: "hwid"; /* Half-width OpenType feature */ }
>
> 毎日カレー食べてるのに、飽きない
> ```

7. Font Feature Resolution

As described in the previous section, font features can be enabled in a variety of ways, either via the use of 'font-variant' or 'font-feature-settings' in a style rule or within an @font-face rule. The resolution order for the union of these settings is defined below. Features defined via CSS properties are applied on top of layout engine default features.

7.1. Default features

For OpenType fonts, user agents must enable the default features defined in the OpenType documentation for a given script and writing mode. Required ligatures, common ligatures and contextual forms must be enabled by default (OpenType features: `rlig, liga, clig, calt`), along with localized forms (OpenType feature: `locl`), and features required for proper display of composed characters and marks (OpenType features: `ccmp, mark, mkmk`). These features must always be enabled, even when the value of the 'font-variant' and 'font-feature-settings' properties is 'normal'. Individual features are only disabled when explicitly overridden by the author, as when 'font-variant-ligatures' is set to 'no-common-ligatures'. For handling complex scripts such as Arabic [ARABIC-TYPO], Khmer or Devanagari additional features are required. For upright text within vertical text runs, vertical alternates (OpenType feature: `vert`) must be enabled.

7.2. Feature precedence

General and *font specific* font feature property settings are resolved in the order below, in ascending order of precedence. This ordering is used to construct a combined list of font features that affect a given text run.

1. Font features enabled by default, including features required for a given script.
2. If the font is defined via an @font-face rule, the font features implied by the font-feature-

settings descriptor in the `@font-face` rule.

3. Font features implied by the value of the 'font-variant' property, the related 'font-variant' subproperties and any other CSS property that uses OpenType features (e.g. the 'font-kerning' property).
4. Feature settings determined by properties other than 'font-variant' or 'font-feature-settings'. For example, setting a non-default value for the 'letter-spacing' property disables common ligatures.
5. Font features implied by the value of 'font-feature-settings' property.

This ordering allows authors to set up a general set of defaults for fonts within their `@font-face` rules, then override them with property settings for specific elements. General property settings override the settings in `@font-face` rules and low-level font feature settings override 'font-variant' property settings.

For situations where the combined list of font feature settings contains more than one value for the same feature, the last value is used. When a font lacks support for a given underlying font feature, text is simply rendered as if that font feature was not enabled; font fallback does not occur and no attempt is made to synthesize the feature except where explicitly defined for specific properties.

7.3. Feature precedence examples

> With the styles below, numbers are rendered proportionally when used within a paragraph but are shown in tabular form within tables of prices:
>
> ```
> body {
>   font-variant-numeric: proportional-nums;
> }
>
> table.prices td {
>   font-variant-numeric: tabular-nums;
> }
> ```

8. Object Model

The contents of `@font-face` rules can be accessed via the following extension to the CSS

Object Model.

8.1. The `CSSFontFaceRule` interface

The ***CSSFontFaceRule*** interface represents a `@font-face` rule.

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

Appendix A: Mapping platform font properties to CSS properties

This appendix is included as background for some of the problems and situations that are described in other sections. It should be viewed as informative only.

Font properties in CSS are designed to be independent of the underlying font formats used; they can be used to specify bitmap fonts, Type1 fonts, SVG fonts in addition to the common TrueType and OpenType fonts. But there are facets of the TrueType and OpenType formats that often cause confusion for authors and present challenges to implementers on different platforms.

Originally developed at Apple, TrueType [[TRUETYPE]] was designed as an outline font format for both screen and print. Microsoft joined Apple in developing the TrueType format and both platforms have supported TrueType fonts since then. Font data in the TrueType format consists of a set of tables distinguished with common four-letter tag names, each containing a specific type of data. For example, naming information, including copyright and license information, is stored in the '`name`' table. The *character map* ('`cmap`') table contains a mapping of character encodings to glyphs. Apple later added additional tables for supporting enhanced typographic functionality; these are now called Apple Advanced Typography, or AAT, fonts. Microsoft and Adobe developed a separate set of tables for advanced typography and called their format OpenType [OPENTYPE]. The OpenType specification is standardized at ISO as the Open Font Format [OPEN-FONT-FORMAT].

In many cases the font data used under Microsoft Windows or Linux is slightly different from the data used under Apple's Mac OS X because the TrueType format allowed for explicit variation across platforms. This includes font metrics, names and *character map* data.

Specifically, font family name data is handled differently across platforms. For TrueType and OpenType fonts these names are contained in the '`name`' table, in name records with name ID 1. Multiple names can be stored for different locales, but Microsoft recommends fonts always

include at least a US English version of the name. On Windows, Microsoft made the decision for backwards compatibility to limit this family name to a maximum of four faces; for larger groupings the "preferred family" (name ID 16) or "WWS family" (name ID 21) can be used. Other platforms such as OSX don't have this limitation, so the family name is used to define all possible groupings.

Other name table data provides names used to uniquely identify a specific face within a family. The full font name (name ID 4) and the Postscript name (name ID 6) describe a single face uniquely. For example, the bold face of the Gill Sans family has a fullname of "Gill Sans Bold" and a Postscript name of "GillSans-Bold". There can be multiple localized versions of the fullname for a given face, but the Postscript name is always a unique name made from a limited set of ASCII characters.

On various platforms, different names are used to search for a font. For example, with the Windows GDI CreateIndirectFont API, either a family or fullname can be used to lookup a face, while on Mac OS X the CTFontCreateWithName API call is used to lookup a given face using the fullname and Postscript name. Under Linux, the fontconfig API allows fonts to be searched using any of these names. In situations where platform API's automatically substitute other font choices, it may be necessary to verify a returned font matches a given name.

The weight of a given face can be determined via the usWeightClass field of the OS/2 table or inferred from the style name (name ID 2). Likewise, the width can be determined via the usWidthClass of the OS/2 table or inferred from the style name. For historical reasons related to synthetic bolding at weights 200 or lower with the Windows GDI API, font designers have sometimes skewed values in the OS/2 table to avoid these weights.

Rendering complex scripts that use contextual shaping such as Thai, Arabic and Devanagari requires features present only in OpenType or AAT fonts. Currently, complex script rendering is supported on Windows and Linux using OpenType font features while both OpenType and AAT font features are used under Mac OS X.

Changes

Changes from the 14 August 2018 CSS Fonts 3 Proposed Recommendation

- Features mentioned in the changelog as having been moved to CSS Fonts 4, now link to the corresponding section in that specification
- Date and boilerplate updates for W3C Recommendation
- Unicode reference updated to latest version

- Updated this changes section

Changes from the March 15 2018 CSS Fonts 3 Candidate Recommendation

- '`font-variant`' descriptor moved to CSS Fonts 4 due to lack of implementations
- '`font-feature-values`' at-rule moved to CSS Fonts 4 due to lack of implementations
- clarified handling of unknown fragment identifiers
- linked to CSS Values & Units for definition of length-percentage

Changes from the October 2013 CSS3 Fonts Candidate Recommendation

- '`font-language-override`' property moved to CSS Fonts 4
- CSSFontFeatureValuesRule interface moved to CSS Fonts 4
- CSSFontFaceRule interface reverted to the widely implemented one from DOM Level 2 style
- clarified that generic font families may be composite faces
- clarified that "first available font" is one that would match the U+0020 (space) character
- clarified how small-caps synthesis interacts with '`font-feature-settings`'
- all CSS keywords marked as invalid font family names
- clarified that '`font-synthesis`' is not reset by the '`font`' shorthand.
- use the phrase "installed fonts" rather than "system fonts"
- clarified that malformed @font-face rules which lack font-family: or src: still show up in the DOM, but don't affect font selection
- clarified conventional ratio range for the relative sizes when they're not modifying an absolute keyword size
- clarified that for both font-variation-settings and font-feature-settings, the computed value is a map (and thus specified dupes are removed)
- added omitted '`font-variant-position`' values to '`font-variant`' shorthand
- made negative values for font-size-adjust invalid, along with negative percentage font-size values
- removed the requirement that user agents use OS/2 table subscript/superscript metrics
- added informative link to CSS Text order of operations

- added normative link to RFC 8081, the font top-level type
- minor editorial cleanups

Acknowledgments

We'd like to thank Tal Leming, Jonathan Kew, Ken Lunde and Christopher Slye for all their help and feedback. John Hudson was kind enough to take the time to explain the subtleties of OpenType language tags and provided the example of character variant usage for displaying text on Byzantine seals. Ken Lunde and Eric Muller provided valuable feedback on CJK OpenType features and Unicode variation selectors. The idea for supporting font features by using 'font-variant' subproperties originated with Håkon Wium Lie, Adam Twardoch and Tal Leming. Elika Etemad supplied some of the initial design ideas for the `@font-feature-values` rule. Thanks also to House Industries for allowing the use of Ed Interlock in the discretionary ligatures example.

A special thanks to Robert Bringhurst for the sublime mind expansion that is *The Elements of Typographic Style*.

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:

> 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 Classes

Conformance to CSS Fonts Level 3 Module 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 CSS Fonts Level 3 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 CSS Fonts Level 3 Module if, in addition to interpreting the style sheet as defined by the appropriate specifications, it supports all the features defined by CSS Fonts Level 3 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 CSS Fonts Level 3 Module 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.

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 component 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.

Experimental Implementations

To avoid clashes with future CSS features, the CSS2.1 specification reserves a prefixed syntax for proprietary and experimental extensions to CSS.

Prior to a specification reaching the Candidate Recommendation stage in the W3C process, all implementations of a CSS feature are considered experimental. The CSS Working Group recommends that implementations use a vendor-prefixed syntax for such features, including those in W3C Working Drafts. This avoids incompatibilities with future changes in the draft.

Non-Experimental Implementations

Once a specification reaches the Candidate Recommendation stage, non-experimental implementations are possible, and implementors should release an unprefixed implementation of any CR-level feature they can demonstrate to be correctly implemented according to spec.

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.

References

Normative References

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

[FETCH]
Fetch. WhatWG Living Standard. URL: https://fetch.spec.whatwg.org/

[OPENTYPE]
OpenType specification. Microsoft. URL: http://www.microsoft.com/typography/otspec/default.htm

[OPENTYPE-FEATURES]

OpenType feature registry. Microsoft. URL: http://www.microsoft.com/typography/otspec/featurelist.htm

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

[RFC8081]
C. Lilley. The "font" Top-Level Media Type. February 2017. Proposed Standard. URL: https://tools.ietf.org/html/rfc8081

[UAX15]
Mark Davis; Ken Whistler. *Unicode Normalization Forms.* 31 August 2012. Unicode Standard Annex #15. URL: http://www.unicode.org/reports/tr15/

[UNICODE]
The Unicode Standard URL: http://www.unicode.org/versions/latest

Other References

[AAT-FEATURES]
Apple Advanced Typography font feature registry. Apple. URL: https://developer.apple.com/fonts/TrueType-Reference-Manual/RM09/AppendixF.html

[ARABIC-TYPO]
Huda Smitshuijzen AbiFares. *Arabic Typography: A Comprehensive Sourcebook.* Saqi Books. 2001. ISBN 0-86356-347-3.

[CHARMOD]
Martin J. Dürst; et al. *Character Model for the World Wide Web 1.0: Fundamentals.* 15 February 2005. W3C Recommendation. URL: http://www.w3.org/TR/2005/REC-charmod-20050215/

[CHARMOD-NORM]
Addison Phillips. *Character Model for the World Wide Web: String Matching.* 20 April 2018. W3C Working Draft. (Work in progress.) URL: https://www.w3.org/TR/2018/WD-charmod-norm-20180420/

[CSS-TEXT-3]
Elika J. Etemad / fantasai; Koji Ishii. *CSS Text Module Level 3.* 22 August 2017. W3C Working Draft. (Work in progress.) URL: https://www.w3.org/TR/2017/WD-css-text-3-20170822/

[CSS3-CONDITIONAL]
L. David Baron. *CSS Conditional Rules Module Level 3.* 4 April 2013. W3C Candidate Recommendation. (Work in progress.) URL: http://www.w3.org/TR/2013/CR-css3-conditional-20130404/

[OPEN-FONT-FORMAT]

Information technology — Coding of audio-visual objects — Part 22: Open Font Format. International Organization for Standardization. ISO/IEC 14496-22:2009. URL: http://standards.iso.org/ittf/PubliclyAvailableStandards/c052136_ISO_IEC_14496-22_2009%28E%29.zip

[OPENTYPE-FONT-GUIDE]

OpenType User Guide. FontShop International. URL: http://www.fontblog.de/wp-content/uploads/2015/11/FF_OTF_user_guide.pdf

[TRUETYPE]

TrueType™ Reference Manual. Apple. URL: https://developer.apple.com/fonts/TrueType-Reference-Manual/

[UAX29]

Mark Davis. *Unicode Text Segmentation.* 12 September 2012. Unicode Standard Annex #29. URL: http://www.unicode.org/reports/tr29/

[WINDOWS-GLYPH-PROC]

John Hudson. *Windows Glyph Processing.* Microsoft Typogrraphy. URL: http://www.microsoft.com/typography/developers/opentype/default.htm

Index

- weight, 2.
- width, 2.

Property index

Property	Values	Initial	Applies to	Inh.	Percentages	Media
font	[[<'font-style'> || <font-variant-css21> || <'font-weight'> || <'font-stretch'>]? <'font-size'> [/ <'line-height'>]? <'font-family'>] | caption | icon | menu | message-box | small-caption | status-bar	see individual properties	all elements	yes	see individual properties	visual
font-family	[<family-name> | <generic-family>] #	depends on user agent	all elements	yes	N/A	visual
font-feature-settings	normal | <feature-tag-value> #	normal	all elements	yes	N/A	visual
font-kerning	auto | normal | none	auto	all elements	yes	N/A	visual
font-size	<absolute-size> | <relative-size> | <length-percentage>	medium	all elements	yes	refer to parent element's font size	visual
font-size-adjust	none | <number>	none	all elements	yes	N/A	visual
font-stretch	normal | ultra-condensed | extra-condensed | condensed | semi-condensed | semi-expanded | expanded | extra-expanded | ultra-expanded	normal	all elements	yes	N/A	visual
font-style	normal | italic | oblique	normal	all elements	yes	N/A	visual
font-synthesis	none | [weight || style]	weight style	all elements	yes	N/A	visual
font-variant	normal | none | [<common-lig-values> || <discretionary-lig-	normal	all elements	yes	see individual properties	visual

Property	Values	Initial	Applies to	Inh.	Percentages	Media
	values> || <historical-lig-values> || <contextual-alt-values> || [small-caps | all-small-caps | petite-caps | all-petite-caps | unicase | titling-caps] || <numeric-figure-values> || <numeric-spacing-values> || <numeric-fraction-values> || ordinal || slashed-zero || <east-asian-variant-values> || <east-asian-width-values> || ruby || [sub | super]]					
font-variant-caps	normal | small-caps | all-small-caps | petite-caps | all-petite-caps | unicase | titling-caps	normal	all elements	yes	N/A	visual
font-variant-east-asian	normal | [<east-asian-variant-values> || <east-asian-width-values> || ruby]	normal	all elements	yes	N/A	visual
font-variant-ligatures	normal | none | [<common-lig-values> || <discretionary-lig-values> || <historical-lig-values> || <contextual-alt-values>]	normal	all elements	yes	N/A	visual
font-variant-numeric	normal | [<numeric-figure-values> || <numeric-spacing-values> || <numeric-fraction-values> || ordinal || slashed-zero]	normal	all elements	yes	N/A	visual
font-variant-position	normal | sub | super	normal	all elements	yes	N/A	visual
font-weight	normal | bold | bolder | lighter | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900	normal	all elements	yes	N/A	visual

Descriptor	Values	Initial
font-family	<family-name>	N/A
font-feature-settings	normal | <feature-tag-value> #	normal

Descriptor	Values	Initial
font-stretch	normal | ultra-condensed | extra-condensed | condensed | semi-condensed | semi-expanded | expanded | extra-expanded | ultra-expanded	normal
font-style	normal | italic | oblique	normal
font-weight	normal | bold | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900	normal
src	[<url> [format(<string> #)]? | <font-face-name>] #	N/A
unicode-range	<urange> #	U+0-10FFFF

W3C Recommendation

CSS Fonts Module Level 3

勧告 – 2018 年 9 月 20 日

この日本語訳は非公式な文書です…（翻訳更新：2018-09-21 ）

このページは、W3Cにより 勧告として公開された CSS Font Module Level 3 を日本語に翻訳したものです。 （公開：2013-11-25 ）

- **この翻訳の正確性は保証されません。**
- 【 と 】で括られた部分は【訳者による注釈】です。
- 各ページに共通の機能も参照されたし（左下隅の表示切替ボタンなど）。
- 誤訳その他ご指摘／ご意見は連絡先まで。

仕様メタデータ

このバージョン（原文 URL ）

https://www.w3.org/TR/css-fonts-3/

このバージョン

https://www.w3.org/TR/2018/REC-css-fonts-3-20180920/

最新発行バージョン

https://www.w3.org/TR/css-fonts-3/

編集者草案

https://drafts.csswg.org/css-fonts/

以前のバージョン

https://www.w3.org/TR/2018/PR-css-fonts-3-20180814/

https://www.w3.org/TR/2018/CR-css-fonts-3-20180626/

https://www.w3.org/TR/2018/CR-css-fonts-3-20180315/

https://www.w3.org/TR/2013/CR-css-fonts-3-20131003/

課題一覧

css-fonts-3 issues on github

Discussion:

on GitHub (preferred), or www-style@w3.org with subject line “[css-fonts] … *message topic* …” (archives)

テスト一式

https://test.csswg.org/harness/results/css-fonts-3_dev/grouped/

編集

John Daggett (Invited Expert)

Myles C. Maxfield (Apple Inc.)

Chris Lilley (W3C)

正誤表

https://www.w3.org/Style/2018/REC-css-fonts-3-20180920-errata.html

索引など

要約

この CSS3 モジュールは、フォントプロパティを指定する方法，および フォント資源を動的に読み込む方法について述べる。 この仕様の内容は、以は， 2 つのモジュール CSS3 Fonts および CSS3 Web Fonts に分割されていた内容を，統合したものである。 フォントの `load` イベントに関する記 CSS Font Loading Module に移動された。

CSS とは…

↑

W3C Recommendation

この文書の位置付け

この節では、発行時点における… 【以下，この節の他の内容は CSS 日本語訳 共通ページに委譲。】

目次

↑

1. 序論

フォントは，文字 [CHARMOD] [UNICODE] の視覚的な表現を包含している資源を供する。 最も単純なレベルでは、それは，文字コードから［ それらの
現する（グリフと呼ばれる）形状 ］へ対応付ける情報を包含する。 同じデザインスタイルを共有するフォントは、［［ 標準フォントプロパティの
基づいて分類される，フォントファミリ ］としてよくグループ化される。 同じファミリ内における［ 所与の文字用に表示される形状 ］は、種々の
ィの中でも特に［ 描線（ stroke ）のウェイト，傾き，相対字幅 ］により，変わり得る。 個々のフォント書体は、これらのプロパティからなる一
せにより，記述される。 各種 CSS フォントプロパティが、所与の範囲のテキストを描画するために利用される［ フォントファミリとそのファミリ
フォント書体 ］を選定するときに利用される。 簡単な例として、 *Helvetica* の bold 形を利用するときは，次を利用できる：

```
body {
    font-family: Helvetica;
    font-weight: bold;
}
```

フォント資源は、［ UA を走らせているシステムにて，ローカルにインストールされているもの ］にも，［ ダウンロード可能なもの ］にもなり得
カルフォント資源用の記述的情報は、フォント資源から直に得られる。 ダウンロード可能フォント資源（web フォントとも呼ばれる）用の記述的情
フォント資源への参照に伴って含められる。

フォントのファミリは、概して，［ 一連のフォントプロパティがとり得る，あるバリエーション ］に対応する書体 一つだけを包含することはない。
ォント選定の仕組みは、［ 所与の CSS フォントプロパティの集合 ］を［ 単独のフォント書体 ］に合致させる方法を記述する。

【 この訳に現れる “書体” は、 “typeface（活字書体）” ではなく， （より抽象的な） “face” （ “italic 体”，等々の “体” ）の

2. タイポグラフィの背景

この節は規範的ではない。

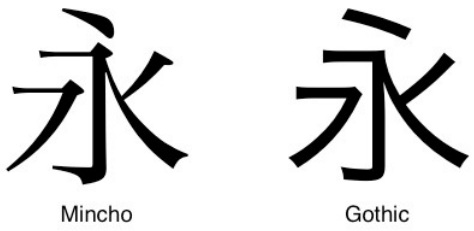
タイポグラフィック伝統様式は，世界に渡り様々なので、言語や文化に渡るすべてのフォントを分類するための，一意な仕方は存在しない。 よくあ
通字ですら、幅広いバリエーションがあり得る：

【 言語 – この仕様の中の “言語” は、ごく一部を除き，（人が話す）自然言語を意味する。 】【 “ラテン” は、 “ラテン用字系” の略
ばよいであろう。 “～語” （トルコ語など）や，“～文字”（キリル文字など），等についても、同様。 】【 普通字（ letter ） – 概念的
や記号類を除く， “普通の” 字（アルファベット，かな，漢字，等々）を意味すると見られる。 】

1 個の文字，多数のグリフバリエーション

字形（ letterform ）の細部における相違は、フォントを判別する方法の一つである。 ラテンフォントにおいては、文字の［ 主描線や serif ］の
flourish 【つる状，跳ね，等の装飾】の有無により，フォントを判別し得る。 同様の比較は，非ラテンフォントにおいても［ 描線が途中で細まる
taperd ） ］と［ 主に一様な描線を利用するもの ］との間に存在する：

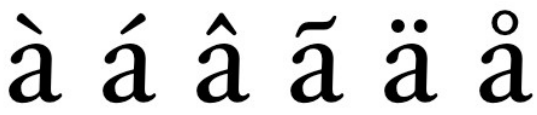
serif を伴う字形と伴わない字形



日本語活字書体に対する同様のグループ分け

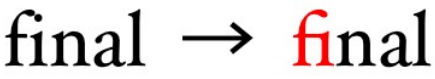
↑

フォントは、一連の字形と，［ それぞれの文字をこれらの字形に対応付けるために必要なデータ ］を包含する。 これは，単純な一対一の対応付けは
も多いが、より複階的な対応付けもあり得る。 結合発音区別符の利用は、下層の字形用に多数のバリエーションを創出する：



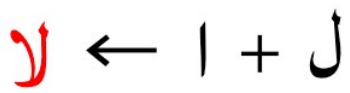
発音区別符を伴うバリエーション

合字（リガチャ）として知られる単独のグリフが、複数の文字からなる連列を表現することもある：



合字の例

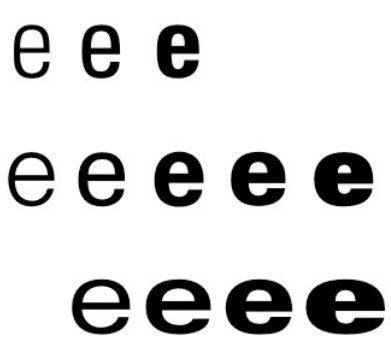
テキストによる文脈に基づく視覚的な変形は、欧州言語においては，大抵はスタイル上のオプションであるが、 [ARABIC-TYPO] のような言語を正し
ためには，必須とされている － 下の 2 つの文字［ lam と alef ］は、それらが並んで位置するときは，結合される*ものとする*：



必須とされるアラビア語合字

これらの［ 形状付け変形 ］は相対的に複階的であり，フォント内に追加のデータを要する。

種々のスタイル上のバリエーションが伴われた［ フォント書体の集合 ］は、よく一緒にされてフォントファミリにグループ化される。 最も単純な
regular 体に伴なって［ bold 体や italic 体 ］も増補されるが、ずっと広範なグループ分けも可能である。 ***ウェイト*** （ weight ） － 字形の描線
や，***字幅*** （ width ） － 字形の全体的な均衡 － におけるバリエーションは、最もよくある。 下の例では、各普通字が［ *Univers* フォントファミリ
るフォント書体 ］を利用している － それらの字幅は 上段から下段にかけて増大され，ウェイトは 左から右にかけて増大されている：



単独のフォントファミリ内におけるウェイトと字幅のバリエーション

複数の用字系をサポートするようなフォントの創出は、困難な仕事である。 デザイナ達は、異なる用字系に渡る活字の利用を，それを取り巻く文化
式を理解した上で、何らかの形で同じテーマを共有するような，字形の集合を作り上げる必要がある。 多くの言語は，しばしば同じ用字系を共有し、
の言語が，目を引くスタイル上の相違を持ち得る。 例えば，アラビア語用字系では、ペルシャ語とウルドゥー語に利用される際に，重大かつ体系的
違を呈する。 セルビア語やロシア語などの言語で利用されるキリル文字でも同様である。

フォントの文字マップは、そのフォント用に［ 文字からグリフへの対応付け ］を定義する。 文書に［［［ 所与の［ フォントファミリのリスト ］
れるフォント ］の文字マップ ］からはサポートされない文字 ］が包含されている場合、 UA は，それをサポートする適切なフォントを見つけるた
テムフォントフォールバック手続きを利用してもよい。 適切なフォントが見出されなかった場合、何らかの形の “欠落を表すグリフ（ missing gly
文字が，UA により描画されることになる。 システムフォールバックは、指定された［ フォントファミリのリスト ］が，［ 所与の文字をサポート
ト ］を含んでいないときに生じ得る。

フォントの文字マップは，所与の文字を［ その文字用のグリフ ］へ対応付けるが、 OpenType [OPENTYPE] や AAT （ Apple Advanced Typography ）
FEATURES] などの現代のフォント技術は，各種特能（ feature，特色機能）の設定群に基づいて，種々の仕方で［ 文字から 異なるグリフへの対応付

↑

する。 これらの形式によるフォントでは、それらの特徴を アプリから制御できるように，フォント自身に埋め込むことが許容されている。 この方法
得る，よくあるタイポグラフィック特徴のごく一部には［ 合字，swash 【参考】，文脈に応じた代替，［ 均衡（可変）幅／一定幅 ］の数字，自動
″1/3″ → ⅓ 等々】 ］などが挙げられる。 OpenType 特徴についての視覚的な概観は［OPENTYPE-FONT-GUIDE］にて見れる。

3. 基本的なフォントプロパティ

文字を描画するために利用される，ある特定のフォント書体は、［ その文字を内容に含む要素 ］に適用される フォントファミリや他の各種フォン
ィにより，決定される。 この構造により、これらの設定群の中の個々の設定は，互いに独立に変わり得るようになる。

3.1. フォントファミリ： font-family プロパティ

名前	*font-family*
値	[<family-name> | <generic-family>]#
初期値	UA に依存する
適用対象	すべての要素
継承	される
百分率	受容しない
算出値	指定値
アニメーション型	不可

このプロパティは、いくつかの［ フォントファミリ名や汎用ファミリ名 ］からなる，優先順位付けられたリストを指定する。 フォントファミリは
ウェイト，字幅，斜傾度 ］を備える書体の集合を定義する。 CSS は、個々の書体を選定するために，ファミリ名と他の一連のスタイル属性の組み
する。 デザインアプリにおいて よく行われる様な［ スタイル名を介する書体の選定法 ］ではなく，この選定の仕組みにより、フォールバックが
も，ある程度 規則正しいテキストによる表示を得られるようになる。

> 注記： デザイナ達は、 CSS による［ 選定に利用されるフォント属性 ］の定義が、［ 明示的に，フォントの分類法を定義することは意図してい
> とを心に留めておくべきである。 活字デザイナによるファミリのアイデアは、標準の［ ウェイト，字幅，斜傾度 ］の軸のみならず，他の軸にも
> わり得るような，書体の集合まで拡張することが多い。 ファミリは、 serif 体の集合と sans-serif 体の集合の両者を含むように，あるいは そ
> リに固有の軸に沿って変わり得るように，拡張し得る。 CSS におけるフォント選定の仕組みは、単に，代用が必要とされるときに “最も近い”
> 定するための，一つの方法を供するものに過ぎない。

他の CSS プロパティと異なり，リストの成分値はカンマで区切られる。 それらは、一連の代替候補を指示する。 UA は、［［ 描画される文字用の
を包含するような，可用なフォント ］に合致するまで、ファミリ名のリストを走査する。 これにより、［ プラットフォームに渡る，可用なフォン
］，および［ 個々のフォントからサポートされる文字範囲の相違 ］が，許容されるようになる。

フォントファミリ名は、［ フォント書体の集合に付与されている名前 ］のみを指定するものであり，個々の書体は指定しない。 例えば，次に挙げ
が可用である下では、 *Futura* は合致することになるが，*Futura Medium* は合致しない：

ファミリと個々の書体名

次の例を考える：

例

```
body {
    font-family: Helvetica, Verdana, sans-serif;
}
```

Helvetica が可用ならば、それが描画時に利用されることになる。 *Helvetica* も *Verdana* も無い場合、 UA により定義される sans-serif フォン
されることになる。

フォントファミリ名は，次の 2 種類に分けられる：

<family-name>

上の例の *Helvetica* や *Verdana* など，候補とされるフォントファミリの名前

<generic-family>

次に定義される，汎用ファミリキーワード ：

serif,

sans-serif, cursive,

fantasy, monospace

これらのキーワードは、作者が欲するフォント候補が可用でないときの，一般フォールバックの仕組みとして利用できるものである。 キーワー
で、引用符で括られてはならない。 作者には、頑健性を向上するため，最後の代替候補として汎用フォントファミリを付加しておくことが奨励

<family-name> によるフォントファミリ名は、 CSS 文字列として，引用符で括った上で与えるか、または［ 1 個以上の CSS 識別子が成す連列 ］
引用符で括らずに与えなければならない。 従って，引用符で括らない場合、各トークンの先頭に位置する［ 約物のほとんど／数字 ］は，エスケー
ればならない。

例

例えば，次に示す宣言は、どれも無効である:

```
font-family: Red/Black, sans-serif;
font-family: "Lucida" Grande, sans-serif;
font-family: Ahem!, sans-serif;
font-family: test@foo, sans-serif;
font-family: #POUND, sans-serif;
font-family: Hawaii 5-0, sans-serif;
```

フォントファミリ名として識別子連列が与えられた場合、その算出値は，連列の中の各識別子を 1 個のスペースで区切って順に連結して 文字列に変
果の名前になる。

エスケープの誤記を避けるため、［ 空白／数字／［ ハイフン以外の約物 ］］を包含するようなフォントファミリ名は，引用符で括ることが推奨さ

```
body { font-family: "New Century Schoolbook", serif }

<BODY STYLE="font-family: '21st Century', fantasy">
```

キーワード値（ inherit, serif, 等々）とたまたま同じになるフォントファミリ *名* は、キーワードと混同されないように，引用符で括られなければい。 UA は、これらのキーワードを［ *<family-name>* 型に合致している ］と見なさないものとする。 このことは、すべての CSS にわたる どのにも適用される。

フォントの集合をいくつかのフォントファミリにグループ化する精確な仕方は、プラットフォームのフォント管理 API に依存して様々である。 Win API においては，グループに許容される書体数は一つのファミリにつき 4 個までである一方、 DirectWrite API や OSX 他のプラットフォーム上のは，種々の［ ウェイト，字幅，斜傾度 ］を伴うフォントファミリがサポートされる（詳細は 付録 A にて見れる）。

一部のフォント形式では、フォントによる複数の［ ファミリ名の地域化版 ］の保持が許容されている。 UA は、それを認識した上で，［ 下層のプームによる地域化名，利用するシステム API，文書の符号化法 ］から独立に，これらの名前すべてを正しく合致させるものとする：

ファミリ名とその地域化版の例

ファミリ名	地域化版
GulimChe	굴림체
Hiragino Kaku Gothic Pro	ヒラギノ角ゴ Pro
Meiryo	メイリオ
MingLiu	細明體
MS Mincho	MS 明朝
Raanana	רענה

［ 地域化されたフォントファミリ名の照合 ］と，それに呼応する［ 文字大小区別についての課題 ］についての詳細は、後述の フォント照合 節に
れる。

3.1.1. 汎用フォントファミリ

5 種の汎用フォントファミリ（ *<generic-family>* ）のそれぞれは、常に，すべての CSS 実装において， 1 つ以上のフォント書体に合致するもの
しかしながら，これらの汎用フォントファミリは、組成書体（［ 文字の Unicode 範囲，包含している要素の言語，利用者選好，システム設定群 ］づいて，複数の活字書体からなるもの）にもなり得る。 それらはまた、常に互いに異なることは保証されない。

UA は、汎用フォントファミリ用の既定の候補として，［ 下層の技術的制約から許される範囲で可能な限り，それぞれのファミリの特徴を表出するな，適度なものをいくつか供するべきである。 UA には、利用者が［ 汎用フォント用の代替候補 ］を選定できるようにすることが奨励される。

serif

serif フォントは、用字系における公式的用途のテキストスタイルを表現する 【 serif − 描線先端の小さな飾り】 。 これは，大抵［［ 終
finishing stroke ）や，末広がり／先細りの末端 ］，あるいは［ 実際に serif 化された末端仕上げ（ slab serif も含む） ］］を備える
フを意味するが，それらに限られない。 serif フォントは、概して，均衡がとられた可変の字幅にされている。 それらは大抵、 sans-serif
トファミリのフォントよりも，描線の太さの変動幅が大きく表示される。 CSS における語 serif は、どの用字系のフォントにも適用されるも
て，利用される − ［ *Mincho* （日本語），*Sung*, *Song* （中国語），*Batang* （韓国語） ］など、ある特定の用字系においては，他の名前の方
染まれているが。 アラビア語用の Naskh スタイルは − その実際のデザインスタイルでなく，そのタイポグラフィック上の役割に因り − より
対応することになるであろう。 このようなフォントが，汎用 serif ファミリを表現するために利用され得る／できる。

serif フォントの見本

sans-serif

CSS 用語としての，sans-serif 【 “serif の無い” 】フォントのグリフは、一般に，縦横の画線の太さが概ね一様で，質素な − すなわち
flaring，画線を横切る細い線【 cross stroke − “t” の横線など】，その他の装飾様式 ］を伴わない − 描線の末端仕上げを備える。 sans
ォントは、概して，均衡がとられた可変の字幅にされている。 それらは大抵、 serif ファミリのフォントより，描線の太さの変動幅が小さい
おける語 sans-serif は、どの用字系のフォントにも適用されるものとして，利用される。 ある特定の用字系においては，他の名前 − 例えば
（日本語），*Hei* （中国語），*Gulim* （韓国語） ］など − の方がより馴染まれているが。 このようなフォントが，汎用 sans-serif ファミリ
るために利用され得る／できる。

sans-serif フォントの見本

cursive

cursive（筆記的）フォントのグリフは、一般に，より非公式な用字系スタイルを利用し，その結果は、印刷物よりも，ペンや絵筆で手書きさ
える。 CSS における語 cursive は、どの用字系のフォントにも適用されるものとして，利用される − フォント名には、［ *Chancery*, *Brush*,
Script ］などの他の名前も利用されるが。

筆記的フォントの見本

fantasy

fantasy フォントは、装飾や表情の豊かさを主とするフォントであり，文字の装飾的な，あるいは表情豊かな表現を包含する。 ［ *Pi*, *Pictu*
の，実際の文字を表現しないフォントは、これらには含まれない。

fantasy フォントの見本

monospace

monospace （等幅）フォントとされるための，唯一の判定基準は、すべてのグリフが同じ固定幅を持つことである。 これは、コンピュータコ
を描画する際に，よく利用される。

↑

等幅フォントの見本

3.2. フォントウェイト： font-weight プロパティ

名前	***font-weight***
値	normal | bold | bolder | lighter | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900
初期値	normal
適用対象	すべての要素
継承	される
百分率	受容しない
算出値	数的字幅（記述を見よ）
アニメーション型	font-weight 値として

font-weight プロパティは、フォント内の一連のグリフのウェイト，すなわち それらの 黒っぽさ／描線の太さ の程度を指定する。

各種値の意味は，次で与えられる：

100, 200, 300, 400, 500, 600, 700, 800, 900

これらの値の各数が指示するウェイトは、大きいもの程，少なくともより小さいもの以上の濃さを表す。 これらは概ね、次に挙げる，よく利用
ェイト名に対応する：

- ***100*** − Thin 【最も細い／薄い／軽い】
- ***200*** − Extra Light (Ultra Light)
- ***300*** − Light (細字)
- ***400*** − Normal (通常)
- ***500*** − Medium
- ***600*** − Semi Bold (Demi Bold)
- ***700*** − Bold (太字)
- ***800*** − Extra Bold (Ultra Bold)
- ***900*** − Black (Heavy) 【最も太い／濃い／重い】

normal

400 と同じ。

bold

700 と同じ。

bolder

継承値より bold なウェイトを指定する。

lighter

継承値より light なウェイトを指定する。

9 段階でない *等級* を利用するフォント形式は、その等級を［ 400 が概ね［ *Regular* ／ *Book* ／ *Roman* ］とされている書体に対応し，700 が概ねとされている書体に合致する ］ように，CSS の等級へ対応付けるべきである。 あるいは、スタイル名から，前述の等級に概ね対応するウェイトが指もよい。 等級は相対的であり、より大きなウェイト値を伴う書体は，より light に現れないものとする。 スタイル名を利用してウェイトを推定するロケールに渡る，スタイル名のバリエーション ］の取り扱いに注意を払うべきである。

ある特定のフォントファミリに対し，限られた少数のウェイトしか可用でないことは、ごく普通にある。 指定されたウェイト用の書体が存在しないれに近いウェイトによる書体が利用される。 一般に、 bold ウェイトのものは より重いウェイトの書体に，light ウェイトのものは より軽いウェに対応付けられる（精確な定義は，下の フォント照合アルゴリズム 節を見よ）。 次の例に、それぞれのウェイトにどの書体が利用されるかを図解色のものは、ウェイトに対応する書体が存在せず，それに近いウェイトによる書体が利用されることを指示する：

↑

ウェイト 400，700，900 の書体が伴われたフォントファミリ用の，ウェイトの対応付け

ウェイト 300，600 の書体が伴われたフォントファミリ用の，ウェイトの対応付け

タイポグラファー達からは あまり好まれていない慣行だが、 bold 体は，実際の bold 体を欠く書体用に，UA により合成されることが多い。 スタ目的においては、これらの書体は，それらがファミリ内に存在しているかのように扱うものとする。 作者は、 font-synthesis プロパティを利用してるまいを明示的に避けることもできる。

［ bolder ／ lighter ］による指定値は、親要素のウェイトに相対的なウェイトを指示する。 ウェイトの算出値は、下の表を利用して， font-weigh に基づいて計算される：

継承値	bolder	lighter
100	400	100
200	400	100
300	400	100
400	700	100
500	700	100
600	900	400
700	900	400
800	900	700
900	900	700

上の一覧は、 normal ， bold に加えて thin ， heavy の書体も包含しているフォントファミリが与えられている下での，相対的に次に bold ／ligh選定法に、等価である。 作者は、要素に利用される正確なウェイト値についての，より精緻な制御を欲するなら、相対ウェイトの代わりに，数を表してもよい。

3.3. フォントの字幅： font-stretch プロパティ

名前	*font-stretch*
値	normal | ultra-condensed | extra-condensed | condensed | semi-condensed | semi-expanded | expanded | extra-expanded | ultra-expanded
初期値	normal
適用対象	すべての要素
継承	される
百分率	受容しない
算出値	指定値
アニメーション型	（注釈文を見よ）

font-stretch プロパティは、フォントファミリから［ normal，…-condensed，…-expanded ］の書体を選定する。 最も幅狭なものから最も幅広なもの に、次の絶対的キーワード値をとり得る：

- *ultra-condensed*
- *extra-condensed*
- *condensed*

- *semi-condensed*
- *normal*
- *semi-expanded*
- *expanded*
- *extra-expanded*
- *ultra-expanded*

所与の字幅用の書体が存在しない下では、［ normal／*-condensed ］値は より幅狭な書体に，他の値は より幅広な書体に対応付けられる。 逆に言
*-expanded 値は より幅広な書体に，他の値は より幅狭な書体に対応付けられる。 下の図に、種々の字幅を包含しているフォントファミリに対し，
font-stretch プロパティ設定のそれぞれがフォントの選定に与える影響を示す － 灰色のものは、字幅に対応する書体が存在せず，異なる字幅で代用
を指示する：

字幅［ condensed，normal，expanded ］の書体を伴うフォントファミリ用の，字幅の対応付け

***font-stretch* に対するアニメーション** は，離散的な階段により補間される。 補間は、順序付けられた一連の値が 等間隔に並ぶ実数であるかのよ
る。 補間の結果は、最も近い値に丸められる － 隣り合う 2 つの値のちょうど中間の値は，上のリストの中で より後に示されている値に向けて丸め

3.4. フォントスタイル： font-style プロパティ

名前	*font-style*
値	normal | italic | oblique
初期値	normal
適用対象	すべての要素
継承	される
百分率	受容しない
算出値	指定値
アニメーション型	不可

font-style プロパティは、 italic 体や oblique 体の選定を可能にする。 一般に、 italic 形は その資質から筆記的である一方，oblique 体は
regular 体の斜傾バージョンである。 oblique 体は、 regular 体のグリフを人工的に傾けて模造されることもある。

人工的に斜傾にされた［ *Palatino* a と *Baskerville* N ］（灰色で示される）と，実際の italic バージョンとの，描画の比較：

人工的な斜傾と本物の italic の比較対照

各種値の意味は，次で与えられる：

normal

italic でも oblique でもない，normal 体（“通常体”）に分類される書体を選定する。

italic

italic 体とされているフォントを，それが無いときは oblique 体を，選定する。

oblique

oblique 体（斜字体／斜体）とされているフォントを，それが無いときは italic 体を，選定する。

italic 体，oblique 体ともに可用でない場合、 oblique 体は，［ 非 oblique 体を利用した，人工的な oblique 化演算を伴う描画 ］により合成さ
これらの人工的な oblique 化された書体の利用は、 font-synthesis プロパティにより不能化できる。 oblique 化演算についての詳細は、明示的に
い。

↑

注記： 作者は、キリル文字のような用字系においては、合成による近似が形状として italic 形と著しく異なり，相応しくない場合もあることを
べきである。 合成によるバージョンに依拠するよりも，実際の italic フォントを利用する方が、常により良いものになる。

多くの用字系では、 normal 体で描画されるテキスト内に筆記的形を混在させる慣習がない。 中国語／日本語／韓国語 のフォントは、ほとんど常に
体や oblique 体を欠いている。 また、用字系の混在をサポートするフォントでは、 italic 体の中でサポートされるグリフの集合から，アラビア語
定の用字系が省略されることがある。 UA は、システムフォントフォールバックを実装する際には，書体に渡る文字マップについての前提の置き方に
意深くあるべきである。

3.5. フォントサイズ： font-size プロパティ

名前	*font-size*
値	*<absolute-size>* | *<relative-size>* | *<length-percentage>*
初期値	medium
適用対象	すべての要素
継承	される
百分率	親要素のフォントサイズに相対的
算出値	絶対長さ
アニメーション型	算出された値型による

このプロパティは、フォントのグリフの，欲される高さを指示する。 拡縮可能なフォントに対しては、 font-size は，フォントの EM 単位に適用さ
率になる（ある種のグリフは，その EM ボックスから外側にはみ出し得ることに注意）。 拡縮可能でないフォントに対しては、 font-size は，絶対
換された上で［ フォントに宣言された font-size ］に対する照合が試みられる － 両値に同じ絶対的な座標系が利用される下で。 各種値の意味は，
られる：

<absolute-size>

各種 *<absolute-size>* キーワードは、［ UA により算出され，保たれる，一連のフォントサイズからなる一覧 ］ － 以下では ***サイズ表*** と言
の，あるエントリを指す。 次の値をとり得る：

[xx-small | x-small | small | medium | large | x-large | xx-large]

<relative-size>

<relative-size> によるキーワードは、［ サイズ表と，親要素の font-size の算出値 ］に相対的に解釈される。 次の値をとり得る：

[*larger* | *smaller*]

例えば，親要素のフォントサイズが medium の場合、値 larger は、現在の要素のフォントサイズを large に設定することになる。 親要素のサ
サイズ表のエントリに近い値でない場合、 UA は，サイズ表の各エントリの合間を補間するか，最も近いものへ丸めるか，どちらを選んでもよ
は、数を表す値がキーワードを超えるときには，サイズ表に値を外挿する必要が生じ得る。

<length-percentage>

長さ値 [CSS-VALUES] は、（サイズ表に依存しない）絶対的フォントサイズを指定する。 負の長さは無効。

百分率値は、親要素のフォントサイズに相対的な，絶対的フォントサイズを指定する。 ［ 百分率や em ］による値の利用は、スタイルシート
健でカスケード可能なものにする。 負の百分率は無効。

次の一覧に、絶対的サイズの拡縮率を［ HTML 見出し ／ 絶対的 font-size ］に対応付けるための，UA への指針を供する。 値 medium は，基準に
として利用される。 UA は、異なるフォントや異なる種類の表示機器のそれぞれに対し，これらの値をより精緻に調整してもよい。

各種 *<absolute-size>* 値（上端行）との対応付け

	xx-small	*x-small*	*small*	*medium*	*large*	*x-large*	*xx-large*	
拡縮率	3/5	3/4	8/9	1	6/5	3/2	2/1	3/1
HTML 見出し	<h6>		<h5>	<h4>	<h3>	<h2>	<h1>	
HTML font-size	1		2	3	4	5	6	7

注記： 可読性を保つため、 UA は、この指針を適用する下でも，コンピュータ表示上で［ EM 単位あたり 9 機器画素 ］を下回るような font-si
出しないようにするべきである。

注記： CSS1 においては、隣のサイズに相対的な拡縮率には 1.5 が示唆されていたが，利用者体験から，大き過ぎるものと判明した。 CSS2 にお
コンピュータスクリーン用に示唆される，隣のサイズに相対的な拡縮率は 1.2 であったが、依然として，小さいサイズにおいて問題があった。 縮
率は、可読性をより良くするため，各サイズの合間ごとに変わり得る。

↑

このプロパティの実際の値は、［ font-size-adjust に対する数を表す値と，ある種のフォントサイズの不可用性 ］に因り，算出値と異なり得る。

子要素は font-size の算出値を継承する（さもなければ、 font-size-adjust による効果は複合的になるであろう）。

例

```
p { font-size: 12pt; }
blockquote { font-size: larger }
em { font-size: 150% }
em { font-size: 1.5em }
```

3.6. 相対的サイズ法： font-size-adjust プロパティ

名前	***font-size-adjust***
値	none | *<number>*
初期値	none
適用対象	すべての要素
継承	される
百分率	受容しない
算出値	指定値
アニメーション型	算出された値型による

所与のどのフォントサイズに対しても、［ 見かけ上のサイズ ］と［ テキストの判読し易さ ］は，フォント間に渡り様々である。 ラテンやキリルの，大文字と小文字を区別する用字系においては、小文字の，対応する大文字に相対的な高さが、判読し易さを決定する要因の一つになる。 この比に ***アスペクト値*** と呼ばれている。 精確には、フォントの x-height をフォントサイズで割り算した結果として定義される。

フォントフォールバックが生じる状況下では、フォールバックフォントが，欲されたフォントファミリと同じアスペクト値を共有しないかもしれず、読み難くなることもある。 font-size-adjust プロパティは、フォントフォールバックが生じたときでも テキストの可読性を保たせる方法を与える。 font-size を［ その x-height が，利用されるフォントに関わらず同じになる ］ように調整することにより行われる。

例
下に定義されるスタイルは，欲されるフォントファミリとして *Verdana* を定義するが、それが可用でない場合は，*Futura* または *Times* が利用さになる：

```
p {
    font-family: Verdana, Futura, Times;
}

<p>Lorem ipsum dolor sit amet, ...</p>
```

Verdana のアスペクト値は比較的高く，小文字は大文字に比して背が高いので、小サイズのテキストは より判読し易く現れる。 *Times* のアスペク り低いので、フォールバックが生じた場合，小サイズのテキストは *Verdana* より判読し難くなる。

これらのフォントによるテキスト描画を比較した様子を下に示す。 各縦列のテキストは，左から順に *Verdana*, *Futura*, *Times* により描画されてい font-size 値は，各段ごとに一定であるが、下半分においては，各縦列に渡る x-height を保全するために［ 実際のフォントサイズを調整する font-size-adjust プロパティ ］も設定されている。 最上段と最下段には x-height の相違を示す，赤線が引かれている。 下半分では、どの列の小トも，判読し易さが比較的 保たれている点に注目。

↑

font-size-adjust の利用を伴わないテキスト（上半分）と伴うテキスト（下半分）

このプロパティは、作者が，要素におけるアスペクト値を指定できるようにする。 それは、代用されたかどうかに関係なく，第一候補のフォントの を実質的に保全する。 各種値の意味は，次で与えられる：

none

フォントの x-height は保全されない

<number>

アスペクト値を指定する – それは、調整済みフォントサイズの計算公式に利用される：

$c = (a \div A) \times s$

ここで ：

s = font-size の値，a = font-size-adjust プロパティに指定されたアスペクト値，
A = フォントの実際のアスペクト値，c
= font-size
の調整後の使用値

負の値は無効。

この値は，選定されたどのフォントにも適用されるが、典型的な用法においては，［ フォントファミリ名リストの中の最初のフォントのアスペ に基づくべきである。 これが正確に指定された下では、上の公式の中の項（ $a \div A$ ）は，最初のフォントに対しては 1 になり，調整は実質 い。 値が正確に指定されていない場合、［ font-size-adjust をサポートしない 古い UA ］においては，［ ファミリリストの最初のフォント 描画されるテキスト ］の表示が異なってくる。

font-size-adjust の値は， font-size の使用値に影響するが、算出値には影響しない。 それは、［ ex や ch ］などの［ 可用な最初のフォントのフ 量に基づく，相対単位 ］のサイズに影響するが，em 単位のサイズには影響しない。 数的 line-height 値は， font-size の算出値を参照するので、 font-size-adjust は，line-height の使用値には影響しない。

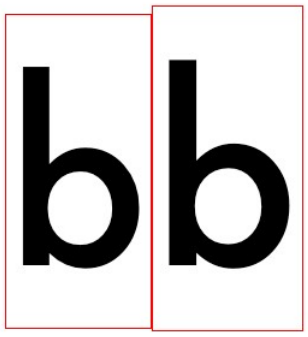
> 注記： CSS においては、作者は line-height を font-size の倍数として指定することが多い。 font-size-adjust プロパティは， font-size の使用 響するので、作者は， font-size-adjust を利用するときは 行高の設定にも注意を払うべきである。 この状況下では、行高の設定を狭小にし過ぎ キスト行が互いに重なり合うおそれがある。

作者は、［ 同じ内容，かつ font-size-adjust プロパティが異なる ］ような 2 つの区間を比較することにより，所与のフォント用のアスペクト値を る。 同じ font-size が利用された下での， 2 つの区間は、そのフォント用の font-size-adjust 値が正確であるときに，合致することになる。

> 例

フォントのアスペクト値を決定するために，境界を伴う 2 つの区間を利用する例。 両区間（ ）とも font-size は同じであるが、右側の
は font-size-adjust プロパティも指定されている。 値 0.5 から始めて、 2 つの普通字を囲む境界が揃うまで，アスペクト値を調整する：

```
p {
    font-family: Futura;
    font-size: 500px;
}

span {
    border: solid 1px red;
}

.adjust {
    font-size-adjust: 0.5;
}

<p><span>b</span><span class="adjust">b</span></p>
```



Futura, アスペクト値 0.5

図右のボックスは，図左のものより少しばかり大きいので、このフォントのアスペクト値は， 0.5 より幾分小さい。 2 つのボックスが揃うまで値
れば正確な値が得られる。

3.7. フォント略式： font プロパティ

名前	*font*
値	[[<‘*font-style*’> || *<font-variant-css21>* || <‘*font-weight*’> || <‘*font-stretch*’>]? <‘*font-size*’> [/ <‘*line-height*’>]? <‘*font-family*’>] | caption | icon | menu | message-box | small-caption | status-bar
初期値	個々のプロパティを見よ
適用対象	個々のプロパティを見よ
継承	個々のプロパティを見よ
百分率	個々のプロパティを見よ
算出値	個々のプロパティを見よ
アニメーション型	個々のプロパティを見よ

↑

font プロパティは、下に述べるものを除く，スタイルシートの中の同じ場所における［ font-style，font-variant，font-weight，font-stretch，font-line-height，font-family ］をまとめて設定するための，略式プロパティである。 ただし， font-variant プロパティ用の値は、 CSS 2.1 にてサポートするものに限られる － この仕様にて font-variant 用に追加されたどの値も， font 略式プロパティには利用できない：

```
<font-variant-css21>
     = [ normal | small-caps ]
```

このプロパティの構文は、フォントに関係する複数のプロパティを設定するための，伝統的タイポグラフィック上の略式記法に基づく。

font プロパティの下位プロパティ － 上に挙げたものに加えて， font-size-adjust，font-kerning，および font-variant のすべての下位プロパティ，font-feature-settings，も含むが， font-synthesis は含まない － すべては、先ず，それぞれの初期値に再設定される。 次に，各 下位プロパティに略式プロパティに明示的に値が与えられていれば，その値が設定される － 許容される値／初期値の定義は、個々の下位プロパティの定義を見よ。 後の理由から、 font 略式プロパティでは， font-size-adjust を初期値以外の値に設定できない － 代わりに、個々のプロパティを利用すること。

例

```
p { font: 12pt/14pt sans-serif }
p { font: 80% sans-serif }
p { font: x-large/110% "new century schoolbook", serif }
p { font: bold italic large Palatino, serif }
p { font: normal small-caps 120%/120% fantasy }
p { font: condensed oblique 12pt "Helvetica Neue", serif; }
```

2 個目の規則におけるフォントサイズ百分率値（ 80% ）は、親要素の font-size の算出値を参照する。 3 個目の規則における行高百分率（ 110% ）は、要素自身のフォントサイズを参照する。

最初の 3 個の規則は［ font-variant と font-weight ］を明示的に指定しないので、これらのプロパティは，それらの初期値（ normal ）を受け取る。フォントファミリ名 "new century schoolbook" はスペースを包含しているので、引用符で括られていることに注意。 4 個目の規則は，［ font-weight は bold，font-style を italic ］に設定し， font-variant を暗黙的に normal に設定する

5 個目の規則は［ font-variant （ small-caps ），font-size （親のフォントサイズの 120% ），line-height （フォントサイズの 120% ），font-family （ fantasy ） ］を設定する。 従って，キーワード normal は残りの 2 つのプロパティ［ font-style，font-weight ］に適用される。

6 個目の規則は［ font-style，font-stretch，font-size，font-family ］を設定し，他のフォントプロパティは それぞれの初期値に設定させておく

font-stretch プロパティは CSS 2.1 では定義されていないので、作者は， font 規則内で font-stretch 値を利用する際には，古い UA とも互換な手ージョンを含めるべきである：

```
p {
  font: 80% sans-serif;   /*  古い UA 用  */
  font: condensed 80% sans-serif;
}
```

次の値はシステムフォントを参照する：

caption

キャプション付きのコントロール（例えば，ボタン，ドロップダウン，等々）用に利用されるフォント

【利用中のブラウザでは、このように呈示される。】

icon

アイコンにラベルを貼るために利用されるフォント

【利用中のブラウザでは、このように呈示される。】

menu

メニュー（例えばドロップダウンメニューやメニューリスト）に利用されるフォント

【利用中のブラウザでは、このように呈示される。】

message-box

ダイアログボックスに利用されるフォント

【利用中のブラウザでは、このように呈示される。】

small-caption

小さなコントロールにラベルを貼るために利用されるフォント

【利用中のブラウザでは、このように呈示される。】

status-bar

ウィンドウのステータスバーに利用されるフォント

【利用中のブラウザでは、このように呈示される。】

↑

システムフォントは一体としてのみ設定し得る – すなわち，［ フォントファミリ，サイズ，ウェイト，スタイル，等々 ］は、すべて同時に設定さかる後，欲されるなら、これらの値は個々に改めれる。 指示された特徴を備えるフォントが，所与のプラットフォーム上に存在しない場合、 UA は用するか（例えば small-caption フォント用には， caption フォントの より小さいバージョンを利用するなど），または UA による既定のフォントるべきである。 定例のフォントと同じく，システムフォントに対しては、個々のプロパティのうち［ OS にて可用な利用者選好の一部を成さない ］いては，それぞれの初期値に設定されるべきである。

これが、このプロパティが “ほぼ” 略式プロパティとされている理由である： システムフォントを指定できるのは – font-family ではなく – このィのみなので、作者は font により［ その一連の下位プロパティの総和 ］より多くのことを行える。 一方で、 font-weight などの個々のプロパテ然としてシステムフォントから採られる値が与えられ、独立に変えられる。

上に挙げられたシステムフォント用のキーワードは、最初に現れるときに限り，キーワードとして扱われる – 他所に位置している場合、フォントフ一部として扱われることに注意：

```
font: menu;        /*  システムメニュー用のフォント設定群を利用  */
font: large menu;  /*  "menu" という名前のフォントファミリを利用  */
```

例

```
button { font: 300 italic 1.3em/1.7em "FB Armada", sans-serif }
button p { font: menu }
button p em { font-weight: bolder }
```

例えば、ある特定のシステム上に［ ドロップダウンメニュー用に利用されるフォント ］が居合わせていて，それが［ 9-point *Charcoal*, ウェイを伴うものであったとするなら、 <button> 要素の子孫の <p> 要素は、次の規則が有効であったかのように表示されることになる：

```
button p { font: 600 9pt Charcoal }
```

font 略式プロパティは，明示的に与えられていない どのプロパティの値も，その初期値に再設定するので、これは，次の宣言と同じ効果になる：

```
button p {
  font-style: normal;
  font-variant: normal;
  font-weight: 600;
  font-size: 9pt;
  line-height: normal;
  font-family: Charcoal
}
```

3.8. 書体の合成を制御する： font-synthesis プロパティ

名前	*font-synthesis*
値	none ｜ [weight ‖ style]
初期値	weight style
適用対象	すべての要素
継承	される
百分率	受容しない
算出値	指定値
アニメーション型	不可

このプロパティは、フォントファミリが［ bold 体／italic 体 ］を欠いているときに，UA による合成による［ bold 体／oblique 体 ］を許容すを制御する：

weight

この値が指定されていない場合、 UA は，bold 体を合成しないものとする。

style

この値が指定されていない場合、 UA は，italic 体を合成しないものとする。

none

この値は、合成による書体は許容されないことを指示する。

注記： 欠落している書体に対しては、合成されない場合には，フォントフォールバックが生じる。

例
次のスタイル規則は、合成により oblique 化されたアラビア語書体の利用を不能化する：

```
*:lang(ar) { font-synthesis: none; }
```

4. フォント資源

4.1. @font-face 規則

@font-face 規則は、［ 必要時に自動的に fetch された上で作動化される ］ようなフォントに向けて，リンクを張れるようにする。 これにより，フォントの候補を［ 所与のプラットフォーム上で可用なフォントの集合 ］に限ることなく，［ ページのデザイン目標に近く合致するフォント ］をるようになる。 ［ この規則が与える，一連のフォント記述子からなる集合 ］は、個々の書体のスタイル特徴に沿うような，ローカルの または外部ト資源の所在を定義する。 複数の @font-face 規則を利用して，［ 種々の書体を備えるフォントファミリ ］を構築することもできる。 UA は、ント照合規則 ］の利用を通して，［ 所与のテキスト断片に必要な書体 ］のみを，選択的にダウンロードできるようになる。

@font-face 規則は［ @font-face at-キーワード，記述子宣言のブロック ］並びからなる。 文法においては、この仕様は，次の生成規則を定義す

```
font_face_rule
  : FONT_FACE_SYM S* '{' S* descriptor_declaration? [ ';' S* descriptor_declaration? ]* '}' S*
  ;

descriptor_declaration
  : property ':' S* expr
  ;
```

S および *property*, *expr* は、 [CSS21] の 字句スキャナ および 文法 にて定義される。

次の新たな定義が【字句スキャナに】導入される：

```
-    -|\\0{0,4}2d(\r\n|[ \t\r\n\f])?
F    f|\\0{0,4}(46|66)(\r\n|[ \t\r\n\f])?
```

次の新たなトークンが導入される：

```
@{F}{O}{N}{T}{-}{F}{A}{C}{E} {return FONT_FACE_SYM;}
```

@font-face 規則は、すべてのフォント記述子に対し，それぞれの値を［ 暗黙的に，または明示的に ］指定する。 規則の中で値が明示的に与えらものについては、［ この仕様の中で，それぞれの記述子の定義に挙げられている初期値 ］を値にとる。 これらの記述子は、もっぱら，［ それが定@font-face 規則の文脈 ］の下でのみ適用され，文書言語の要素に【直に】適用されるものではない。 記述子がどの要素に適用されるかや，値が子承されるかどうかの概念は存在しない。 所与の @font-face 規則の中に，同じ名前の記述子が複数回現れたときは、最後の記述子宣言のみが利用さ記述子より前に現れている宣言は すべて無視される。

例
Gentium と称されるダウンロード可能フォントを利用するためには：

```
@font-face {
  font-family: Gentium;
  src: url(http://example.com/fonts/Gentium.woff);
}

p { font-family: Gentium, serif; }
```

UA は、 *Gentium* をダウンロードした上で，［ <p> 要素内のテキスト ］の描画時に，それを利用することになる。 何らかの理由でフォントをササイトが可用にされてない場合、既定の serif フォントが利用されることになる。

所与の @font-face 規則の集合が，［ これらの規則を包含する文書において可用なフォントの集合 ］を定義する。 フォント照合を終えたときは、規則を利用して定義されるフォントが，［ システム上の他の可用なフォント ］の前に考慮される。

ダウンロードされたフォントは、それらを参照する文書においてのみ可用である。 これらのフォントを作動化させる処理は、それらを［ 同じフォンリンクしていない他のアプリや文書 ］からは可用にしないものとする。 UA の実装者は、［ ダウンロードされたフォントを［ システムフォントフク手続きの下で，可用なフォントが他に存在しない ］ような他の文書 ］内の文字の描画時に利用すること ］が簡便であると考えるかもしれないがジの内容が他のページに影響できることになれば，攻撃者から何らかの攻撃手段に利用され得ることになり、セキュリティに穴を開けかねない。 こは，キャッシュ処理のふるまいには影響しない － フォントは、他の web 資源がキャッシュされるのと同じ仕方でキャッシュされる。

↑

この at-規則は、 CSS の前方互換な構文解析規則に従う。 宣言ブロック内のプロパティと同様、 UA は，自身がサポートしない どの記述子に対し
宣言を無視するものとする。 @font-face 規則は、 [font-family，および src] 記述子を要求する － これらのいずれかが欠落している @font-fa
フォント照合アルゴリズム を遂行する際には無視されるものとする。

UA のプラットフォーム資源が限られているか，もしくは UA がダウンロード可能フォント資源を不能化する能を実装している下では、 @font-face
単純に無視されるものとする。 すなわち，この仕様にて定義される [個々の記述子のふるまい] は、改められるべきではない。

4.2. フォントファミリ： font-family 記述子

名前	***font-family***
値	*<family-name>*
初期値	なし（必須）

この記述子は、すべての [CSS フォントファミリ名の照合] に利用されることになる，フォントファミリ名を定義する。 @font-face 規則が妥当
には，これが要求される。 それは，下層のフォントデータに包含されているフォントファミリ名を上書きする。 すなわち、フォントファミリ名が，
環境にて可用なフォントファミリ] の名前と同じである場合、これは，その下層のフォント － この記述子が記されているスタイルシートを利用する
の，下層のフォント － を実質的に隠蔽する。 これにより，web 作者は、 [利用者環境に在るフォントファミリ名] との競合を気にせずに，自由に
ファミリ名を選べるようになる。 同様に、所与のフォントファミリ名に対しいぇは，プラットフォーム代用は利用しないものとする。

4.3. フォント参照： src 記述子

名前	***src***
値	[*<url>* [format(*<string>*#)]? | *<font-face-name>*]#
初期値	なし（必須）

この記述子は、フォントデータを包含している資源を指定する。 @font-face 規則が妥当になるためには，これが要求される。 その値は，カンマで
たリストであり、優先順位付けられた，一連の [外部参照または [ローカルにインストールされたフォント書体名]] からなる。 UA は，フォン
なったときに、リストされている参照の集合を走査した上で，成功裡に作動化できる最初のものを利用する。 無効なデータを包含しているフォント
ルフォント書体が見出されないものは，無視され、 UA はリスト内で次にあるフォントを読み込む。

CSS における他の URL と同様に，URL は相対的でもよい － その場合、 [@font-face 規則を包含しているスタイルシートの所在] から相対的に解
る。 SVG フォントの場合、 URL は [SVG フォント定義を包含している文書] 内の要素を指す － 要素参照が省略されている場合、 [定義されてい
トのうち，最初のもの] への参照が黙示される。 同様に，フォントコンテナ形式が複数のフォントを包含し得る場合も、所与の 1 個の @font-fa
に読み込まれるフォントは，きっかり 1 個になるものとする。 素片識別子は、どのフォントを読み込むかを指示するために利用される － これらは
[RFC8081] にて定義される，フォントの PostScript 名を利用する。 適合 UA は、素片識別子が [未知／未サポート] である場合には，そのフォ
ダウンロードすることなく飛ばすものとする。 例えば，OpenType フォント集をサポートしない旧 UA は、リスト内で次にある URL へ飛ぶことにな

```
src: url(fonts/simple.woff);   /* スタイルシートの所在から相対的な "simple.woff" を読み込む */
src: url(/fonts/simple.woff);  /* 絶対的な所在から "simple.woff" を読み込む */
src: url(fonts/coll.otc#foo);  /* フォント集 "coll.otc" からフォント "foo" を読み込む */
src: url(fonts/coll.woff2#foo);/* WOFF 2 フォント集 "coll.woff2" からフォント "foo" を読み込む */
src: url(fonts.svg#simple);    /* id に "simple" が付与された SVG フォントを読み込む */
```

外部参照は、 [1 個の URL ，および 任意選択の形式ヒント] 並びからなる。 形式ヒントは、 URL が参照するフォント資源の形式について記述す
あり、いくつかの [[周知のフォント形式] を表示する形式文字列] からなる，カンマで区切られたリスト（ format() の引数）を包含する。 適
は、その形式ヒントが指示しているフォント形式が どれも [未知／未サポート] である場合には，そのフォント資源をダウンロードすることなく
とする。 形式ヒントが給されていない場合、そのフォント資源をダウンロードするべきである。

```
/* 可能なら WOFF2 フォントを読み込んで， 他の場合は WOFF フォントを読み込んで， 他の場合は OpenType フォントを利用する */
@font-face {
  font-family: bodytext;
  src: url(ideal-sans-serif.woff2) format("woff2"),
       url(good-sans-serif.woff) format("woff"),
       url(basic-sans-serif.ttf) format("opentype");
}
```

この仕様にて定義される形式文字列：

↑

文字列	フォント形式	よくある拡張子
"woff"	WOFF 1.0 (Web Open Font Format)	.woff
"woff2"	WOFF 2.0 (Web Open Font Format)	.woff2
"truetype"	TrueType	.ttf
"opentype"	OpenType	.ttf, .otf
"embedded-opentype"	Embedded OpenType	.eot
"svg"	SVG Font	.svg, .svgz

TrueType と OpenType [OPENTYPE] の間ではよくある用法が重なり合っているので、形式ヒント ["truetype" と "opentype"] は，同義と見なされる － 形式ヒント "opentype" は、フォントが [Postscript CFF スタイルグリフデータ あるいは OpenType レイアウト情報] を包含することを意味ではない（これについての より深い背景は 付録 A に）。

作者は，所与のフォント用に [ローカルに可用な複製が在ればそれを利用し，無ければダウンロードする] ことを選好する場合には、 ***local()*** を使る。 その引数 *<font-face-name>* †は、ローカルにインストールされた，より大きなファミリ内で単独のフォント書体を一意に識別するような，文字列を与える。 *<font-face-name>* の構文†は "local(" , ")" で括られたフォントファミリ名である。 名前は、任意選択で引用符で括ることも括られていない場合の名前は、引用符で括られないファミリ名の処理規約に従って，空白で区切られた識別子たちが成す連列でなければならない － 別子たちを単独のスペースで区切って順に連結することにより，文字列に変換される。

【† *<font-face-name>* の構文が "local(…)" の括りも含むのか含まないのか，記述が紛らわしいが、 src 記述子の値定義欄に整合させるため含める必要がある。 】

```
/*  Gentium の regular 体  */
@font-face {
  font-family: MyGentium;
  src: local(Gentium),    /*  Gentium がローカルに可用なら，それを利用  */
       url(Gentium.woff); /*  他の場合はダウンロード  */
}
```

OpenType および TrueType のフォントに対しては、この文字列が合致する対象は [ローカルに可用なフォントの [名前テーブル] の中の [Postscript 名] と [全部的フォント名]] に限られる。 どちらの種類の名前が利用されるかは、プラットフォームとフォントにより様々なので、作者は，プラットフォーム渡る適正な照合を確保するためには，これらの名前の両者を含めるべきである。 所与のフォント名に対しては，プラットフォーム代用を利用しない。

```
/*  Gentium の bold 体  */
@font-face {
  font-family: MyGentium;
  src: local(Gentium Bold),    /*  全部的フォント名  */
       local(Gentium-Bold),    /*  Postscript 名  */
       url(GentiumBold.woff);  /*  上のいずれも利用できないときは，ダウンロード  */
  font-weight: bold;
}
```

@font-face 規則が [1 個のファミリ内の単独のフォント] の特徴を指定するのとちょうど同じく， local() に利用される一意な名前は、フォント全体ではなく，単独のフォントを指定する。 OpenType フォントデータの用語で言えば、 Postscript 名は， [フォントの 名前テーブル] の中の nameID = 6 の name record] の中から見出される（詳細は [OPENTYPE] に）。 Postscript 名が、 [OSX 上ではすべてのフォント，Windows 上では Postscript フォント] 用に共通して利用される key である。 全部的フォント名 (nameID = 4) は、 Windows 上では TrueType グリフを伴うフォント用の一として利用される。

複数個の [全部的フォント名の地域化版] を伴うような OpenType フォントに対しては、 [US 英語版 (Windows 上では languageID = 0x409, Mac 上では languageID = 0)] か，あるいは US 英語の全部的フォント名が可用でないときは [最初の地域化名] が利用される (OpenType 仕様ではントにも最小限 US 英語名は含める ことが推奨されている）。 他の全部的フォント名 にも 合致させる UA （ 例えば，現在のシステムロケールが に設定されている下での，オランダ語名への照合) は、不適合と見なされる。 これは、 [英語を選好しない，かつ [[フォントのバージョンや 地域化名] の相違による，照合の不一致を避ける]] ように，行われる － フォントスタイル名（例えば， "Bold") は，多くの言語にて常習的にており、可用な地域化された名前からなる集合は， [多岐に渡る，プラットフォームやフォントのバージョン] 間で様々なので。 [ファミリ名 1) とスタイル名 (nameID = 2)] の連結に合致させる UA も，不適合と見なされる。

また，これにより、他の方法では参照し得ない様な， [より大きなファミリに属する書体] へ【直に】参照することも可能になる。

例
ローカルフォントを利用する，または 別の文書の中の SVG フォントを参照する：

```
@font-face {
  font-family: Headline;
  src: local(Futura-Medium),
       url(fonts.svg#MyGeometricModern) format("svg");
}
```

↑

異なるプラットフォーム上に渡る，ローカル日本語フォント用の別名を作成する：

```
@font-face {
  font-family: jpgothic;
  src: local(HiraKakuPro-W3), local(Meiryo), local(IPAPGothic);
}
```

より大きなファミリ内では合致され得ない 1 個のフォント書体を参照する：

```
@font-face {
  font-family: Hoefler Text Ornaments;
  /* は、 Hoefler Text Regular と同じフォントプロパティを備える【ので，従来の方法では選定できない（ような例として挙げられている？）】 */
  src: local(HoeflerText-Ornaments);
}
```

地域化された全部名は，決して合致しないので、下の見出し用スタイル規則を伴う文書は、ある特定のシステムロケールパラメタがフィンランド語れているかどうかに関わらず，常に既定の serif フォントを利用して描画される：

```
@font-face {
  font-family: SectionHeader;
  src: local("Arial Lihavoitu");  /* Arial Bold 用のフィンランド語による全部名は失敗するべき */
  font-weight: bold;
}

h2 { font-family: SectionHeader, serif; }
```

適合 UA は、下の例の中のフォント gentium.eot を決して読み込まないことになる − それは src 記述子の最初の定義に含まれており，同じ @fon規則の中の 2 個目の定義により上書きされるので：

```
@font-face {
  font-family: MainText;
  src: url(gentium.eot);                    /* 古い UA 用 */
  src: local("Gentium"), url(gentium.woff);  /* src 定義を上書きする */
}
```

4.4. フォントプロパティ記述子： font-style ， font-weight ， font-stretch 記述子

名前	*font-style*
値	normal | italic | oblique
初期値	normal

名前	*font-weight*
値	normal | bold | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900
初期値	normal

名前	*font-stretch*
値	normal | ultra-condensed | extra-condensed | condensed | semi-condensed | semi-expanded | expanded | extra-expanded | ultra-expanded
初期値	normal

これらの記述子は、フォント書体の特徴を定義し，スタイルを特定の書体へ合致させる処理に利用される。 何個かの @font-face 規則により定義さントファミリ用には、 UA は、［ ファミリの中の書体をすべてダウンロードする ］か，または［ これらの記述子を利用して，［［ 文書の中で利用際のスタイル ］に合致するフォント書体 ］を選択的にダウンロードする ］ことができる。 これらの記述子がとり得る値は、相対キーワード［ bo lighter ］が許容されないことを除いて，対応するフォントプロパティがとり得る値と同じである。 省略された記述子については、初期値をとるもれる。

これらのフォント書体スタイル属性にあてがわれる値は、下層のフォントデータにより黙示されるスタイルに代わって利用される。 これにより，作のフォントデータが異なる形に編成されている状況下でも，いくつかの書体を柔軟に組み合せることが可能になる。 合成による［ bold 化／obliqu

↑

実装する UA は、［ フォントデータにより黙示されるスタイル属性 ］に基づいてではなく，フォント記述子から必要と黙示された所でのみ、合成に
イル付けを適用するものとする。

例
この節に定義するフォント記述子は、［ @font-face 規則により，所与のファミリ用に定義されたフォントの集合 ］から，フォントを選定すると
される。

単独の regular 体のみを包含しているファミリを考える：

```
@font-face {
  font-family: BaskervilleSimple;
  src: url(baskerville-regular.woff);
}
```

スタイル付けされていないテキストは， @font-face 規則の中で定義された regular 体を利用して表示されることになる：

fiddlesticks!

しかしながら、 italic テキストについては，別々に定義された italic 体はないので、大多数の UA は，regular 体からのグリフを 合成により
化して表示することになる：

fiddlesticks!

次に、実際に italic 体が定義されているファミリを考える：

```
@font-face {
  font-family: BaskervilleFull;
  src: url(baskerville-regular.woff);
}

@font-face {
  font-family: BaskervilleFull;
  src: url(baskerville-italic.woff);
  font-style: italic;
}
```

2 個目の @font-face 規則は、スタイル属性に［ italic -style, normal -weight, normal -stretch ］を持つフォント資源 baskerville-italic.
定義する。 italic テキストを表示するときは、 UA は，italic テキストに最も近く合致する このフォントを利用することになる。 従って、 r
からのグリフを合成により oblique 化するのではなく，活字デザイナによりデザインされたグリフを利用して、テキストが表示される：

fiddlesticks!

フォントファミリの中からある特定の書体を選定する処理についての，より完全な詳細は、 フォントスタイルの照合 節に。

4.5. 文字範囲： unicode-range 記述子

名前	*unicode-range*
値	*<urange>#*
初期値	U+0-10FFFF

この記述子を宣言することにより、フォント書体がサポートし得る Unicode [UNICODE] 符号位置たちの集合を定義できる。 記述子は、一連の Unic
<urange> ）からなる，カンマ区切りのリストを値にとる。 符号位置の集合は、これらの範囲の和集合として定義され，UA が［ 所与のテキスト連
ォント資源をダウンロードするかどうか ］を決める際のヒントになる。

各 *<urange>* 値は、 *UNICODE-RANGE* トークンを成す［［ 接頭辞 U+ または u+ ］，符号位置範囲 ］並びで与えられる。 符号位置範囲は、次に挙げ
ずれかの形をとる － この形に収まらないものは無効であり，当の宣言は無視されることになる：

単独の符号位置（例： U+416 ）

1 〜 6 個の 16 進数字として表現される， 1 個の Unicode 符号位置のみからなる範囲

区間範囲（例： U+400-4ff ）

ハイフンで区切られた 2 個の 16 進数字列として表現される範囲 − 2 個の数字列は、順に，範囲の［ 始端，終端 ］を指示する。

ワイルドカード範囲（例： U+4?? ）

尾部の各 文字 ? が任意の 16 進数字を表すとするときの，符号位置の集合として定義される範囲。

ここで：

- 個々の符号位置は、 Unicode 文字符号位置 に対応する 16 進値で記される。
- いずれにせよ、符号位置を与える各 数字は，文字大小無視であり、指示される符号位置は，範囲 { 0 〜 10FFFF } に入らなければならない。
- 区間範囲における終端の符号位置は、始端の符号位置以上でなければならない。
- 先頭の数字を欠いているワイルドカード範囲（例： U+??? ）も妥当であり，先頭に数字 0 を付与したワイルドカード範囲（例： U+0??? ，すな U+0000-0FFF ）と等価になる。 Unicode 符号位置の範囲に収まり切らないワイルドカード範囲は無効になる。 したがって、 *UNICODE-RANGE* トー 文字まで受容するが，尾部のワイルドカード文字 ? は 5 個までになる。

unicode-range 記述子の宣言に含まれる Unicode 範囲リストにおいては，複数の範囲が互いに重なり合ってもよい。 これらの範囲の和集合が，対応 ォントから利用され得る［ 符号位置の集合 ］を定義する。 UA は、この集合に入らない符号位置に対しては，当該のフォントを［ ダウンロードし しない ］ものとする。 UA は、範囲リストを，同じ［ 符号位置の集合 ］を表現する，異なるリストに正規化してもよい。

当該のフォントは、［ unicode-range 記述子により定義される符号位置集合 ］用のグリフたち全部は包含しないかもしれない。 フォントが利用され ***有効文字マップ*** とは、［ この符号位置集合 ］と［ フォントの文字マップ ］との共通部分（積集合）として，定義される。 これにより，作者は、 ォントがサポートする符号位置の精確な範囲について気にせずに，サポートされる範囲を おおまかな範囲で定義できるようになる。

4.6. 組成フォントを定義するための文字範囲の用法

同じ［ ファミリと［ 一連のスタイル記述子の値 ］］用に，異なる unicode-range を伴うような、複数の @font-face 規則を利用して、［ 用字系 るフォント ］からのグリフを混合する［ 組成フォント ］を創出できる。 これを利用すれば、［［ 単独の用字系（例えば， ラテン，ギリシア文字 文字 ）用のグリフ ］のみを包含するフォント ］をいくつか組み合せることができる。 あるいは，作者は、あるフォントを［ よく利用される文字 ト ］と［ 頻出しない文字用のフォント ］に区分する仕方として，これを利用できる。 UA は，必要なフォントのみを取り寄せることになるので、 幅の節約にもなる。

同じ［ ファミリと［ 一連のスタイル記述子の値 ］］を伴う 複数の @font-face 規則 において， unicode-range が重なり合っている場合、それら 検査される順序は，それらが定義された順序の逆順になる。 したがって，所与の文字に対し最初に検査される規則は、最後に定義されたそれになる

特定の［ 言語や文字 ］用の範囲の例：

unicode-range: U+A5;

単独の符号位置：［円／元］通貨記号

unicode-range: U+0-7F;

基本 ASCII文字の符号範囲

unicode-range: U+590-5ff;

ヘブライ語文字の符号範囲

unicode-range: U+A5, U+4E00-9FFF, U+30??, U+FF00-FF9F;

日本語［ 漢字，平仮名，片仮名 ］文字，および［円／元］通貨記号の符号範囲

例

BBC はニュースサービスを様々な言語で供しているが、その多くは，すべてのプラットフォームに渡りきちんとサポートされているとは限らない。 face 規則を利用して、 BBC は，これらのどの言語用のフォントも［ 手動によるフォントダウンロードにより すでに行われていた ］かのように ようになるであろう。

```
@font-face {
  font-family: BBCBengali;
  src: url(fonts/BBCBengali.woff) format("woff");
  unicode-range: U+00-FF, U+980-9FF;
}
```

例

技術文書では、多岐に渡る記号を要することが多い。 STIX フォントプロジェクトは、［ 標準化された仕方で，多岐に渡る技術用植字をサポート のフォント ］を供することを目指しているプロジェクトの一つである。 下の例に，［ Unicode 内の［ 多数の 数学用／技術用 記号 ］用のグリ するフォントの利用を示す：

↑

```
@font-face {
  font-family: STIXGeneral;
  src: local(STIXGeneral), url(/stixfonts/STIXGeneral.otf);
  unicode-range: U+000-49F, U+2000-27FF, U+2900-2BFF, U+1D400-1D7FF;
}
```

例

次の例は、作者が，［ 日本語フォントの中で利用されるラテン文字 ］のグリフを，異なるフォントからのグリフで上書きする方法を示している。の規則は，範囲を指定していないので、既定で全範囲になる。 ［ 2 個目の規則の中で指定される範囲 ］は重なり合うことになるが、より後の方れているので，より優先される。

```
@font-face {
  font-family: JapaneseWithGentium;
  src: local(MSMincho);
  /*  範囲は指定されていないので、既定で全範囲になる。  */
}

@font-face {
  font-family: JapaneseWithGentium;
  src: url(../fonts/Gentium.woff);
  unicode-range: U+0-2FF;
}
```

例

帯域幅を最適化するために、［ ラテン，日本語，その他 ］の文字を，それぞれ異なるフォントファイルに分離することにより構築される、ファミる：

```
/*  フォールバックフォント – サイズ： 4.5MB  */
@font-face {
  font-family: DroidSans;
  src: url(DroidSansFallback.woff);
  /*  範囲は指定されていないので、既定で全範囲になる。  */
}

/*  日本語グリフ – サイズ： 1.2MB  */
@font-face {
  font-family: DroidSans;
  src: url(DroidSansJapanese.woff);
  unicode-range: U+3000-9FFF, U+ff??;
}

/*  いくつかの約物と記号も伴われた ラテン，ギリシア文字，キリル文字 – サイズ： 190KB  */
@font-face {
  font-family: DroidSans;
  src: url(DroidSans.woff);
  unicode-range: U+000-5FF, U+1e00-1fff, U+2000-2300;
}
```

単純なラテンテキスト用には、ラテン文字用のフォントのみがダウンロードされる：

```
body { font-family: DroidSans; }

<p>This is that</p>
```

この場合、 UA はまず，［ ラテン文字を包含しているフォント（ DroidSans.woff ） ］用の unicode-range を検査する。 上の文字はすべて，範囲 U+0-5FF に入るので、 UA は，そのフォントをダウンロードした上で，テキストをそのフォントにより描画する。

次に，矢印文字（ ⇨ ）を利用するテキストを考える：

```
<p>This &#x21e8; that<p>
```

ここでも UA は，最初に［ ラテン文字を包含しているフォント ］の unicode-range を検査する。 矢印の符号位置（ U+21E8 ）は，範囲 U+2000-入るので、 UA は，そのフォントをダウンロードする。 しかしながら，ラテンフォントは，この文字に合致しているグリフを持たないので、このは，［ フォント照合用に利用される有効 unicode-range ］からは除外される。 次に，UA は日本語フォントを評価する。 日本語フォント用の unicode-range は［ U+3000-9FFF，および U+ff?? ］であり， U+21E8 は含まれていないので、 UA は，その日本語フォントをダウンロードしない。フォールバックフォントが考慮される。 フォールバックフォント用の @font-face 規則は， unicode-range を定義していないので、その値は，既すべての Unicode 符号位置からなる集合 ］になる。 フォールバックフォントが，矢印文字を描画するためにダウンロードされて利用される。

↑

4.7. フォント特能： font-feature-settings 記述子

名前	*font-feature-settings*
値	normal | *<feature-tag-value>#*
初期値	normal

この記述子は、［ @font-face 規則により定義されるフォント ］の描画時に適用される，初期設定群を定義する。 これはフォントの選定には影響
各種 値は、値 inherit は除外されることを除き，対応する font-feature-settings プロパティ用に定義されるものに一致する。 複数の［ フォント
述子／プロパティ ］］が利用されるときの，［ テキスト描画における累積的な効果 ］の詳細は、 フォント特能解決 節にて。

4.8. フォント読み込みの指針

@font-face 規則は、［［ フォント資源が，文書内で利用されるときにのみダウンロードされる ］ようにする，遅延読み込み ］を許容するように
いる。 スタイルシートには、［［ 選定された集合のみを利用するような［ フォントのライブラリ ］］用の， @font-face 規則 ］を含ませるこ
る。 UA は、［ スタイル規則内から参照されているフォント ］のうち，［ 所与のページに適用可能なもの ］のみをダウンロードするものとする。
らのフォントがページ内で実際に利用されるかどうか ］について考慮せずに，［ @font-face 規則にて定義されるフォント ］すべてをダウンロー
は、不適合と見なされる。 ［ 文字フォールバックが生じる場合にフォントがダウンロードされ得る ］下では、 UA は［［［ 所与のテキスト連な
font-family ］の算出値 ］に包含されているフォント ］をダウンロードしてもよい。

```
@font-face {
  font-family: GeometricModern;
  src: url(font.woff);
}

p {
  /* フォントは、 <p> 要素を伴うページ用にダウンロードされることになる */
  font-family: GeometricModern, sans-serif;
}

h2 {
  /* <h2> 要素を伴うページ用には、 Futura がローカルに可用であっても，フォントがダウンロードされ得る */
  font-family: Futura, GeometricModern, sans-serif;
}
```

ダウンロード可能フォントが可用になる前に，テキストによる内容が読み込まれた所では、 UA は，テキストを［［ ダウンロード可能フォント資源
いときに描画されることになる ］ような形 ］に描画するか，あるいは、 フォールバックフォントの利用によるテキストの明滅 【 ダウンロードが
にフォールバックフォントによるテキストが一瞬表示される現象（ “FOUT” ）】 を避けるために，フォールバックフォントによるテキストを 一時
に描画してもよい。 UA は、フォントのダウンロードに失敗した所では，テキストを表示するものとする － 単純にテキストを透明なまま放置するこ
合なふるまいと見なされる。 作者には，巨大ページの再流し込みを避けるため、可能な所では，フォールバックフォントには［ フォントリストの中
ンロード可能フォントの計量に近く合致するもの ］を利用することを勧める。

4.9. フォント fetch 処理に課される要件

UA は、 @font-face 規則内で定義される URL に対するフォントの読み込みには， CORS も可能化され得る fetch † を利用するものとする －（ u
CORS 属性状態 ）として（ スタイルシートの URL ，″style″，Anonymous ）を与える下で。 [FETCH]

【† この用語 “potentially CORS-enabled fetch” の 原文によるリンク先 （ 原文による以前のリンク先 ）では意味が通らないので、この
により補完している。 】

注記： 作者にとってこれが意味する所は、作者がそれを許可する手続きを特に踏まない限り，非同一生成元からのフォントは，概ね読み込まれな
とである。 サイトは Access-Control-Allow-Origin HTTP ヘッダの利用により，サイトをまたがるフォントデータの読み込みを明示的に許容できる
HTTP(S) 以外の 】スキーム用には、 “potentially CORS-enabled fetch method” により許可されるものを超えて［ 定義される または要求さ
うな，［ 非同一生成元からの読み込みを許容する明示的な仕組み ］は存在しない。

例
下に与える例では、文書の所在は ″https://example.com/page.html″ であり，かつ すべての URL のリンクは［ UA からサポートされる妥当なフォ
］を指しているとする。 下の［ 各種 src 記述子 の値 ］により定義されるフォントは、読み込まれることになる：

↑

```
/*  同一生成元（すなわち，［ ドメイン，スキーム，ポート番号 ］が文書のそれに合致する）  */
src: url(fonts/simple.woff);

/*  リダイレクトを伴わない data: URL は，同一生成元として扱われる  */
src: url("data:application/font-woff;base64,...");

/*  非同一生成元 – ドメインが異なる  */
/*  が，  Access-Control-Allow-Origin 応答ヘッダは '*' に設定されているとする  */
src: url(http://another.example.com/fonts/simple.woff);
```

下の［ 各種 src 記述子の値 ］により定義されるフォントの読み込みは，失敗することになる（いずれも，応答の中に Access-Control-… ヘッダはる）：

```
/*  非同一生成元 – スキームが異なる  */
src: url(https://example.com/fonts/simple.woff);

/*  非同一生成元 – ドメインが異なる  */
src: url(http://another.example.com/fonts/simple.woff);
```

5. フォント照合アルゴリズム

個々のテキスト連なりが，どのようにしてフォントに結び付けられるかは、以下のアルゴリズムに述べられる。 テキスト連なりの中の各文字ごとに，文字用のグリフを包含しているフォントファミリ ］が選ばれ，ある特定のフォント書体が選定される。

5.1. フォントファミリ名の文字大小区別

以下に要旨するフォント照合アルゴリズムの一部として、 UA は，［ スタイル規則の中で利用されている一連のフォントファミリ名 ］と［ 所与の可用なフォントに包含されている実際のフォントファミリ名 ／ @font-face 規則の中で定義されるフォントファミリ名 ］とを照合するものとするこれらの名前を， Unicode 仕様 [UNICODE] にて要旨されている［ “Default Caseless Matching” アルゴリズム ］を利用して，文字大小無視のるものとする。 このアルゴリズムの詳細は、その仕様の［ 3.13 “Default Case Algorithms” ］の節に示されている。 具体的には、このアルゴ文字列を正規化することなく，かつ 言語特有の いかなるあつらえも適用することなく ］適用されるものとする。 このアルゴリズムにより指定さ字正規化メソッド ］は、［ Unicode Character Database のファイル ″CaseFolding.txt″ [UNICODE] ］の中の，［ ″status″ フィールドに［ ″C″ ］が伴われた［ 文字大小 対応付け ］］を利用する。

> 注記： 作者にとっては、これは次を意味する： フォントファミリ名は、それらの名前が［ プラットフォームフォントの中に，または［ スタイ包含されている @font-face 規則 ］の中に存在するかどうかに応じて，文字大小無視の下で照合される。 作者は、発音区別符などの結合文字をいるときは特に，名前が確実に［ 実際のフォントファミリ名に整合する文字連列 ］になるように，注意を払うべきである。 例えば、［ 小文字a U+0061 ），COMBINING RING ABOVE （ U+030A ） ］が成す並びを包含するファミリ名は，その結合文字連列に代えて［ 合成済みの［ 小文字の字 （ U+00E5, ″å″ ） ］］を利用する名前と，見かけ上は一致するが、**合致しない**ことになる。

> 注記： 実装者は、［ 所与の［ 大小が無い文字列比較の実装 ］が，この精確なアルゴリズムを利用しているかどうか ］を検証し，［ 所与の［ フォームの文字列照合ルーチン ］が，それに従っている ］ものと見做さないように、注意を払うべきである – これらの多くが、ロケール特有のを備えていたり，あるいは 何らかのレベルの文字列正規化 [UAX15] を利用しているので。

5.2. フォントスタイルの照合

テキスト連なりの中の所与の文字用に，フォントを選ぶ際の手続きは、次から構成される：

- font-family プロパティにより名前が与えられている，一連のフォントファミリの走査法。
- ［ 他の各種フォントプロパティ ］に基づく適切なスタイルを伴うような，フォント書体の選定法。
- 所与の文字用のグリフが存在するかどうかの決定法。 これは、フォントの ***文字マップ*** – すなわち，文字をその文字用の既定のグリフに対応付タ – を利用して行われる。

フォントは、次が成立するとき，所与の文字を ***サポート*** するものと見なされる：

- その文字はフォントの文字マップに包含されている，かつ
- その文字を包含している用字系から要求されている場合は，その文字用の形状付け情報が【フォントにて】可用である。

旧来のフォントには、所与の文字を文字マップの中に含みつつ，［ その文字を包含しているテキスト連なり ］を正しく描画するために必要とされる付け情報（例えば， OpenType レイアウトテーブル や Graphite tables ）を欠いているものもある。

［ 基底文字，結合文字連列 ］並びを成す［ 符号位置連列 ］の扱いは，少しばかり異なる – それについては， クラスタ照合 節にて見れる。

↑

この手続きにおける［ 所与のフォントファミリ用の ***既定の書体*** ］とは、［［ すべてのフォントスタイルプロパティが，それぞれの初期値に設定さ ］としたときに，選定される書体 ］として定義される。

1. UA は，所与の要素用の［ フォントプロパティの算出値 ］を利用して，［ font-family プロパティに指定されている最初のファミリ名 ］から始する。
2. ファミリ名が汎用ファミリキーワードである場合 ：
   UA は，利用されることになる適切なフォントファミリ名を表引きする。 UA は、利用する汎用フォントファミリを［［ 包含している要素や，［ その文字の Unicode 範囲 ］］に基づいて，選んでもよい。
3. 他の場合（他のファミリ名に対しては） ：
   UA は，［ @font-face 規則を介して定義される一連のフォント ］の中から，ファミリ名を見出した上で、上の節で要旨したように［ 可ムフォント ］の中から， 文字大小無視による比較 により名前の照合を試みる：
   - システムに，複数の［ 地域化されたフォントファミリ名 ］を包含するフォントがある下では， UA は，これらのどの名前に対する照合もしている下層の［ システムロケールやプラットフォーム API ］］からは独立に行うものとする。
   - ［ @font-face 規則の中で与えられた書体 ］用に定義されているフォント資源が，［ 可用でない，または 無効なフォントデータを包含 ］場合 ：
     その書体は，［ そのファミリの中に無い ］ものと扱われるべきである。
   - ［ @font-face 規則を介して定義されるファミリ ］用の書体が無い場合 ：
     そのファミリは欠落していると扱われるべきであり，同じ名前のプラットフォームフォントと合致させないものとする。
4. フォントファミリとの合致が生じた場合 ：
   UA は、そのファミリの中のフォント書体の集合（以下， “***照合集合***” と記される）を組み上げた上で，その集合から［ 下に与える順にフォントプロパティを利用して ］単独の書体に絞り込む。 @font-face 規則を介して定義される［ フォント記述子値には一致しつつ， un 値は異なる ］ような［ 書体のグループ ］は、この段のためには，［ 単独の ***組成書体*** ］をなすものと見なされる：
   a. 最初に， font-stretch について試行される：
      - 照合集合が［ font-stretch 値に合致している字幅値 ］を伴う書体を包含している場合 ：
        ［ 他の字幅値を伴う書体 ］は、照合集合から除去する。
      - 他の場合（字幅値に正確に合致する書体が無い場合）、代わりに最も近い字幅を利用する：
        - font-stretch の値が［ normal，または いずれかの …-condensed 値 ］である場合 ：
          各 字幅値を，幅狭なものから幅広なものへ順に検査していく。
        - font-stretch の値がいずれかの …-expanded 値である場合 ：
          各 字幅値を，幅広なものから幅狭なものへ順に検査していく。

      この処理により［ 最も近く合致する字幅 ］が決定されたなら、［ 他の字幅を伴う書体 ］は，照合集合から除去する
   b. 次に， font-style について試行される：
      - font-style の値に応じて，対応する項目に示された順で 書体を検査する：

        italic
        italic → oblique → normal

        oblique
        oblique → italic → normal

        normal
        normal → oblique → italic

        ［ 他のスタイル値 ］を伴う書体は，照合集合から除外する。
      - UA には［ プラットフォームフォントファミリ ］内の［ italic 体と oblique 体 ］の判別が許可されるが，これは要求されないのての［ italic 体／oblique 体 ］を italic 体として扱ってもよい。 しかしながら，［ @font-face 規則を介して定義されるフォミリ ］内では、［ italic 体と oblique 体 ］は，［ font-style 記述子の値 ］を利用して判別されるものとする。
      - ［ italic 体／oblique 体 ］を欠くどのファミリに対しても、 UA は， font-synthesis プロパティの値により許可されるのであれな oblique 体を創出してもよい。
   c. 最後に font-weight が照合される – これにより，照合集合は常に単独のフォント書体に絞り込まれるようになる：
      1. 相対ウェイト［ bolder／lighter ］が利用されている場合 ：
         実質的なウェイトは、 font-weight プロパティの定義に述べたように，ウェイトの継承値に基づいて計算される。
      2. 照合集合の一連の書体のウェイトの中に，欲されたウェイト（以下 *w* と記す）として可用なものがある場合 ：
         その書体が合致する。
      3. 他の場合 ：
         ウェイトは、 *w* に応じて，以下の規則を利用して選ばれる：

↑

w < 400 の場合

合致が見出されるまで、まず w より下のウェイトを降順に検査して，次に w より上のウェイトを昇順に検査する。

w > 500 の場合

合致が見出されるまで、まず w より上のウェイトを昇順に検査して，次に w より下のウェイトを降順に検査する。

w = 400 の場合

最初に 500 を検査した上で， w < 400 用の規則を利用する。

w = 500 の場合

最初に 400 を検査した上で， w < 400 用の規則を利用する。

d. font-size は、 UA に依存する許容差に収まるサイズに合致するものとする（概して、拡縮可能なフォント用のサイズは，ピクセルの整数近いサイズに丸められる一方、ビットマップ化されたフォント用の許容差は 20% くらいの大きさにされる）。 更なる計算 – 例えば、他ィの中の em 値によるサイズ – は、 font-size の指定値ではなく，使用値に基づく。

5. 合致した書体が @font-face 規則を介して定義されている場合 ：
UA は、単独のフォントを選定するために，次の手続きを利用するものとする：

a. ［ フォント資源がまだ読み込まれていない ］，かつ［ unicode-range 記述子の値により定義される文字範囲に 当の文字が含まれているそのフォントを読み込む。

b. ダウンロードの後，その有効文字マップが当の文字をサポートする場合 ：
そのフォントを選定する。

合致した書体が組成書体であるときは、 UA は，組成書体の中の各 書体に対し［ @font-face 規則定義の逆順 ］で上の手続きを利用するも

ダウンロードの間、 UA は，［ フォントがダウンロードされるまで待機する ］か，または［ いったん，代用によるフォント計量の下で描画しダウンロード後にもう一度描画する ］のいずれを選んでもよい。

6. ［ 合致している書体が存在しない ］，または［ 合致した書体が［ 描画される文字用のグリフ ］を包含しない ］場合 ：
次にあるファミリ名が選定され，［ 前の 3 個の段 ］が繰り返される。 ファミリの中の他の書体からのグリフは、考慮されない – ただし書体が所与のグリフをサポートする ］，かつ［ font-synthesis プロパティの値により，これらの書体の合成が許可されている ］場合は、任意選択で［ 既定の書体を合成により oblique 化したバージョン ］で代用してもよい。 例えば，italic 体が［ アラビア語用のグリフートしない場合には、［ regular 体の合成による italic バージョン ］が利用されてもよい。

7. ［ 評価されるフォントファミリが尽きた ］，かつ［ 合致する書体が見出されなかった ］場合 ：
UA は、［［ 描画される文字 ］に最も釣り合う書体 ］を見出すために，［ ***システムフォントフォールバック*** 手続き ］を遂行する。 この結果は、 UA 間に渡り変わり得る。 【すなわち、この手続きの内容は［ UA ／実装 ］に依存する – この仕様の中では規定されていない

8. ある特定の文字が，どのフォントを利用しても表示できない場合 ：
UA は、次のいずれかを行うべきである：

・ 何らかの手段により，［ 象徴的な表現を成す欠落を表すグリフ ］を表示する（例えば， Last Resort Font を利用して）

・ 既定のフォントからの欠落を表す文字グリフを利用して，文字が表示されていないことを指示する

この処理の最適化は、実装がアルゴリズムに正確に従ったかのようにふるまう限り，許容される。 同じ［［ 可用なフォントの集合と，描画技術 ］］の下で現れる結果が，可能な限り UA 間で一致するよう、照合は well-defined な順序で生じる【照合の順序が，このアルゴリズムと矛盾しないよものとする？】。

例えば［ ex や ch などの，フォントに相対的な長さ ］や［ line-height プロパティ ］などの定義にて利用されている ***可用な最初のフォント*** とは font-family リスト内に所与のフォントファミリにて可用なフォントのうち， U+0020 (space) に合致する最初のもの（可用なものが無ければ，UA フォント）として定義される。

5.3. クラスタ照合

テキストが結合マークなどの文字を包含するとき、理想的には，［ 基底文字が，そのマークと同じフォントを利用して描画される ］べきである。り，マークの適正な配置が確保される。 この理由から、［ クラスタ用のフォント照合アルゴリズム ］は，一般の場合の［ 単独の文字自身による照も特化されている。 ［［ 所与の文字に利用する精確なグリフ ］を指示する異体字選択子 【 参考】 ］を包含している連列に対しては、 UA は常に文字の既定のグリフを利用する前に，適切なグリフを見出す ］ためシステムフォントフォールバックを試みる。

【以下、この節の内容は未訳。】

5.4. 文字の取り扱いの問題

CSS フォント照合は、常に，［ Unicode 文字 [UNICODE] を包含しているテキスト連なり ］上で遂行されるので、旧来の符号化法を利用している文ント照合の前に符号変換されているものと見做される。 Unicode 用の他に，旧来の符号化法用の文字マップも包含しているフォントに対しては、そ字マップの内容は，フォント照合処理の結果に影響しないものとする。

フォント照合処理は、テキスト連なりが正規化形であるとも，その逆の形にされているとも，見做さない（詳細は [CHARMOD-NORM] に）。 フォント文字，結合マークたち ］からなる連列に分解された形をサポートせずに，合成済みの形のみをサポートしてもよい。 作者は常に，内容の文字スト正規化形，またはその逆 ］のいずれになるかも考慮した上で、フォントの候補をあつらえるべきである。

所与の文字の符号位置が，Unicode 私用領域に属する場合、 UA は：

- ［ font-family リストの中で名前が与えられた，汎用ファミリでないフォントファミリ ］のみに合致させるものとする。
- font-family リストの中で名前が与えられたどのファミリも，その符号位置用のグリフを包含していない場合 ：
  UA は、［ その符号位置用にシステムフォントフォールバックを試みる ］ことなく，その文字用に［ 何らかの形の欠落を表すグリフ記号するものとする。

置換文字 U+FFFD に対する照合に際しては、 UA は，フォント照合処理を飛ばして［ 何らかの形の欠落を表すグリフ記号 ］を即時に表示してもよらについては、［ フォント照合処理により選定されることになるフォント ］からのグリフを表示することは要求されない。

一般には、所与のあるファミリ用の一連のフォントは，すべて同じまたは同様の 文字マップを持つことになる。 ここに要旨する処理は、フォント多様な文字マップを伴う書体を包含しているときでも，取り扱えるように設計されている。 しかしながら、その種のファミリの利用が 予期されないたらし得ることも，作者に忠告しておく。

5.5. フォント照合に関する CSS 2.1 からの変更点

上のアルゴリズムは、いくつかの重要な点で， CSS 2.1 によるものと異なる。 これらの変更は、［ UA 間の実装に渡る，実際のフォント照合のふる より良く反映するために，加えられた。

CSS 2.1 によるフォント照合アルゴリズムとの相違は：

- このアルゴリズムには、 font-stretch の照合も含まれている。
- 可能なすべての font-style 照合事例が想定されている。
- small-caps フォントは、フォント照合処理の一部として合致されることはない。 それらは今やフォント特能を介して取り扱われる。
- Unicode 異体字選択子の照合が要求される。
- クラスタは 1 個の単位として合致させられる。

5.6. フォント照合の例

例
CSS 選択子構文を，［ 言語に応じて切り替わるタイポグラフィ ］の創出に利用し得ることも、知っておくと有用であろう。 例えば，一部の中国語の文字は、両言語における抽象グリフは同じでないにもかかわらず，同じ Unicode 符号位置に統一されているが、例えば次の CSS は：

```
*:lang(ja) { font: 900 14pt/16pt "Heisei Mincho W9", serif; }
*:lang(zh-Hant-TW) { font: 800 14pt/16.5pt "Li Sung", serif; }
```

所与の言語 – ここでは（台湾で利用される）日本語および繁体字中国語 – を備えるすべての要素を選択して，それらに適切なフォントが利用さ する。

6. フォント特能プロパティ

現代のフォント技術は、種々の［ 先進的タイポグラフィックや，言語特有のフォント特能 ］をサポートする。 これらの特能を利用して，単独のフ 多岐に渡る合字，文脈に応じたスタイル上の代替，［ 一定幅／old-style ］の数字，小 capital 【“小さく” された大文字】，自動的分数，swash 言語特有の代替 ］用のグリフを供せる。 作者がこれらのフォント能力を制御できるようにするため、 CSS3 用に font-variant プロパティが拡張さ れは今や，［ フォントの各種スタイル上の特能用の制御を供する各種プロパティ ］用の略式プロパティとして機能する。

6.1. グリフの選定と位置決め

この節は規範的ではない。

ラテンテキストの表示に利用される単純なフォントは、ごく基本的な処理モデルを利用する。 フォントは、各文字をその文字用のグリフに対応付け ップを包含する。 後続の文字用のグリフは、単純に，テキスト連なりに沿って他の文字の後に置かれる。 OpenType や AAT （ Apple Advanced Typ などの，現代のフォント形式は、より多彩な処理モデルを利用する。 所与の文字用のグリフは、文字自身の符号位置のみならず，［ 隣接の文字や 字系，テキスト用に可能化された各種 特能 ］］に基づいて［ 選ばれ，位置も補正され ］得る。 フォント特能は、特定の用字系用に要求されるこ ば，既定で可能化されるよう推奨されることもあれば，作者による制御用として意図された スタイル上の特能のこともある。 全体的なテキスト処理 キスト変形，テキスト方位，テキスト整列など）において フォント選定，位置決めがどこで起こるかについては、 [CSS-TEXT-3] の 各種テキスト適 順序 にて述べられる。

これらの特能の有益な視覚的概観は［OPENTYPE-FONT-GUIDE］にて見れる。 OpenType フォント用のグリフ処理の詳細な記述については、［WINDOWS-PROC］にて見れる。

フォントのスタイル上の特能は、大きく 2 つに分類できる： 一つは［ カーニング，合字 ］特能などの，［ グリフの形状と，周囲の文脈との調和するもの、もう一つは［ small-caps ，［下付き／上付き］文字，代替 ］特能などの，形状の選定に影響するものである。

下に挙げられる［ font-variant の下位プロパティ ］が、これらの［ フォントのスタイル上の特能 ］を制御するために利用される。 それらは、ラやインド系の言語テキストを表示する際に利用される 各種 OpenType 特能など，ある種の用字系の表示に要求される特能は制御しない。 それらは、［ 選定と位置決め ］に影響するが，フォント照合節にて述べられているフォント選定には影響しない（ただし， CSS 2.1 との互換性を得るために場合は除く）。

UA 間に渡るふるまいの一貫性を確保するため、個々のプロパティ用に，等価な［ OpenType プロパティ設定群 ］が挙げられ，規範的とされる。 他形式の利用に際しては、これらは，各種［ CSS フォント特能プロパティ値 ］を特定のフォント特能へ対応付けるための指針として、利用されるべき

6.2. 言語特有の表示

OpenType は、［ 言語が規定する，特定の［ 表示のふるまい ］］に従って，テキストを正しく表示できるようにするために、言語特有の，グリフの位置決め ］もサポートする。 多くの言語が同じ用字系を共有するが、ある種の普通字の形状は，言語間に渡り変わり得る。 例えば，ある種のキリ字の形状は、ロシア語テキストとブルガリア語テキストとでは互いに異なる。 ラテンテキストにおいては、 "fi" は［ 小文字 "i" の上のドットをな fi-合字 ］でよく描画される。 しかしながら、 ドット付き "i" とドットなし "i" の両者を利用するトルコ語などの言語においては、 この合字ないか，あるいは［ "i" の上にドットを包含する，特化されたバージョンの合字 ］を利用することが重要になる。 下の例に［ スペイン語，イタリランス語 ］の正書法にて見出される，［ スタイル上の伝統様式に基づく，言語特有のバリエーション ］を示す：

要素の内容言語が 文書言語 の規則に則って既知である場合、 UA には，内容言語から OpenType 言語システムを推定した上で、 OpenType フォントるグリフの［ 選定法と位置決め ］に，その言語システムを利用することが要求される。

6.3. カーニング： font-kerning プロパティ

名前	***font-kerning***
値	auto | normal | none
初期値	auto
適用対象	すべての要素
継承	される
百分率	受容しない
算出値	指定値
アニメーション型	不可

カーニングとは、文脈に応じた，グリフ間の間隔調整【アキ組】である。 このプロパティは、計量カーニング － フォントに包含されている調整用の活用するカーニング － を制御する。

auto

カーニングの適用は，UA の裁量に任せることを指定する。

normal

カーニングは適用することを指定する。

none

カーニングは適用しないことを指定する。

カーニングデータが含まれないフォントに対しては、このプロパティによる可視効果はないことになる。 OpenType フォントによる描画の下では、仕様から，カーニングは既定で可能化されるものと示唆されている。 カーニングが可能化されている下では、 OpenType kern 特能が可能化されるテキスト連なり用には代わりに vkrn 特能が可能化される）。 UA は、［ kern フォントテーブル － 詳細は OpenType 仕様にて － に包含されている介してのみ，カーニングをサポートするようなフォント ］も，サポートするものとする。 letter-spacing プロパティが定義されている場合、カーニ既定の間隔調整の一部と見なされ、普通字の間隔調整は，カーニングの適用後に行われる。

↑

kerning-auto に設定されているときは、 UA は，カーニングを適用するかどうかを［ テキストサイズ，用字系，テキスト処理の速度に波及する他の基づいて決定できる。 適正なカーニングを求める作者は、カーニングを明示的に可能化するために， normal を利用するべきである。 反対に、精確り処理能が重要な状況では，カーニングの不能化を選好する作者もいるかもしれない。 しかしながら，現代の高度に設計された実装の下では、カー用がテキスト描画の速度に大きく響くことはない。

6.4. 合字： font-variant-ligatures プロパティ

名前	font-variant-ligatures
値	normal | none | <common-lig-values> | <discretionary-lig-values> | <historical-lig-values> | <contextual-alt-values>
初期値	normal
適用対象	すべての要素
継承	される
百分率	受容しない
算出値	指定値
アニメーション型	不可

合字と［ 文脈に応じた形 ］は、より調和した形を生産するために，グリフを結合する仕方である。

```
<common-lig-values>
      = [ common-ligatures | no-common-ligatures ]

<discretionary-lig-values>
      = [ discretionary-ligatures | no-discretionary-ligatures ]

<historical-lig-values>
      = [ historical-ligatures | no-historical-ligatures ]

<contextual-alt-values>
      = [ contextual | no-contextual ]
```

各種値の意味は，次で与えられる：

normal

よくある既定の特能を可能化させることを指定する － 詳細は フォント特能解決 節に述べられる。 OpenType フォントに対しては、共通合字と応じた形 ］］は既定で可能化され，随意合字と歴史的合字はそうでない。

none

このプロパティが受け持つすべての種類の［ 合字と［ 文脈に応じた形 ］］を、明示的に不能化させることを指定する。 合字を考慮する必要では、これによりテキスト描画の速度が向上し得る。

common-ligatures

共通合字（ OpenType 特能： liga, clig ）による表示を可能化する。 OpenType フォントに対しては、共通合字は，既定で可能化される。

no-common-ligatures

共通合字（ OpenType 特能： liga, clig ）による表示を不能化する。

discretionary-ligatures

随意合字（ OpenType 特能： dlig ）による表示を可能化する。 どの合字が［ 随意または任意選択【 discretionary or optional － 同じ意味は，活字デザイナが決めるので、作者は， どの合字が随意と見なされているかを理解するために，所与のフォントについての文献を参照する必とになる。

no-discretionary-ligatures

随意合字（ OpenType 特能： dlig ）による表示を不能化する。

historical-ligatures

歴史的合字（ OpenType 特能： `hlig` ）による表示を可能化する。

no-historical-ligatures

歴史的合字（ OpenType 特能： `hlig` ）による表示を不能化する。

contextual

文脈に応じた代替（ OpenType 特能： `calt` ）による表示を可能化する。 厳密には合字 特能ではないが、合字と同様に，この特能は グリフの
囲の文脈を調和させるために利用される点で共通する。 OpenType フォントに対しては、この特能は既定でオンである。

no-contextual

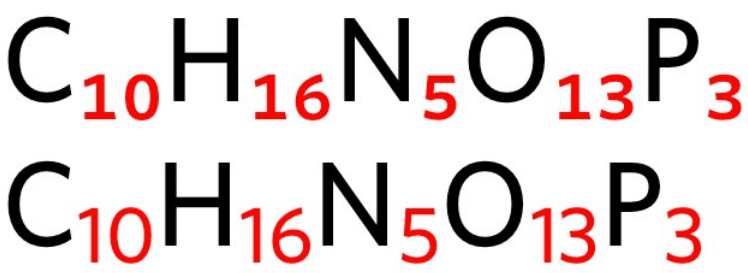
文脈に応じた代替（ OpenType 特能： `calt` ）による表示を不能化する。

複階的な用字系を正しく描画するために必要とされる，必須 合字（ OpenType 特能： `rlig` ）は、 none も含め，上の設定群からは影響されない。

6.5. ［下付き／上付き］文字形： font-variant-position プロパティ

名前	*font-variant-position*
値	normal | sub | super
初期値	normal
適用対象	すべての要素
継承	される
百分率	受容しない
算出値	指定値
アニメーション型	不可

このプロパティは、タイポグラフィック［下付き／上付き］文字グリフを可能化するために利用される。 これらは，既定のグリフと同じ em-box 内
された代替グリフであり、個別の［ サイズ法や基底線の位置決め ］は利用せずに，既定のグリフと同じ基底線上にレイアウトさせる用途に意図さ
これらは、行高への影響を伴わずに，周囲のテキストと釣り合うように，かつ より読み易くなるように、明示的にデザインされたものである。



下付き文字グリフ（上段）と，典型的な合成された下付き文字（下段）の比較対照

各種値の意味は，次で与えられる：

normal

下に挙げられるどの特能も，可能化されない。

sub

下付き文字異体（ OpenType 特能： `subs` ）による表示を可能化する。

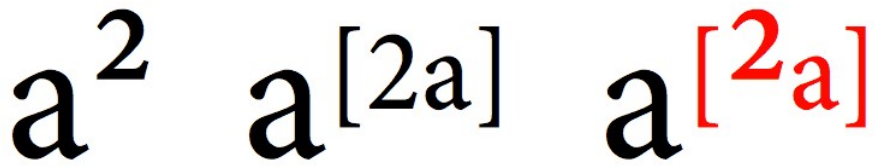
super

上付き文字異体（ OpenType 特能： `sups` ）による表示を可能化する。

［下付き／上付き］文字の意味論上の資質から、所与のテキスト連なり用の値が sub ／ super である下で，そのテキスト連なりの中に［ 異体グリ
ない文字 ］が一つでもある場合、すべての文字に対し，［ この特能が適用されなかったとするときに利用されるグリフ ］に還元された形を利用し
よるグリフが合成されるべきである。 これは、［ 異体グリフと合成されたものとの混在 – その場合，正しく揃わないことになる ］を避けるため，

↑

連なりごとに行われるとする。 所与の文字用の［下付き／上付き］文字グリフを欠いた OpenType フォントの場合、 UA は，適切な［下付き／上付
リフを合成するものとする。



上付き文字による代替グリフ（図左），合成された上付き文字グリフ（図中央），この 2 つの不正な混在（図右）

テキスト装飾が［ ［下付き／上付き］文字グリフを包含しているテキスト連なり ］のみに適用される状況【？】においては、装飾の配置に伴う問題
ために，合成されたグリフが利用されてもよい。

過去においては、 UA は， <sub> ／ <sup> 要素用の［下付き／上付き］文字を模造するために， font-size と vertical-align を利用していた。 ［
上付き］文字の定義法に後方互換性が保たれるようにするため、作者には，［ 古い UA が，依然として古い仕組みを介して［下付き／上付き］文字
ことになる ］ように，条件付き規則 [CSS3-CONDITIONAL] を利用することが推奨される。

これらの要素には font-size: smaller が利用されることが多いので、［下付き／上付き］文字に適用される実質的な拡縮率は，サイズに依存して変
素のテキストサイズが大きいときは，約 3 分の 1 削減されることが多い一方で、サイズが小さいときの削減量は，［下付き／上付き］文字を読み易
め，ずっと少ない。 UA は、［下付き／上付き］文字グリフを合成する方法を決めるときに，これを考慮するべきである。

> OpenType フォント形式は，［下付き／上付き］文字の計量を OS/2 テーブル [OPENTYPE] にて定義しているが、実施においては常に正確とは限ら
> で，［下付き／上付き］文字を合成するときには依拠できない。

作者は、フォントから供される［ ［下付き／上付き］文字用のグリフ ］は，概して［ フォントからサポートされる文字 ］の一部分に限られるこ
するべきである。 例えば，［下付き／上付き］文字グリフは、ラテン数字用には大抵は可用である一方、約物や普通字文字用のグリフが供されるこ
ど多くはない。 このプロパティ用に定義される［ 合成によるフォールバック規則 ］により，［下付き／上付き］文字が常に現れるようになること
るが、利用されるフォントが，［下付き／上付き］文字に包含されている文字のうち，どれか一つでも適切な代替グリフを供していない場合の外観に
期待に沿わなくなるかもしれない。

このプロパティは累積的でない。 このプロパティが［下付き／上付き］文字内の要素に適用されても、［下付き／上付き］文字グリフの配置が入れ
ことはない － このプロパティの値が［ sub や super ］にされていても、テキスト連なり内に包含されるイメージは，値が normal であったときと同
れることになる。

これらの制限のため、 UA スタイルシート用途には， font-variant-position は推奨されない。 作者は、［下付き／上付き］文字を，それらが［ 指
フォントからサポートされる文字 ］による狭い範囲のみを包含することになる所に限って，利用するべきである。

> 注記： 異体グリフは、既定のグリフと同じ基底線を利用する。 配置の際に，基底線によりズラされることはないので、異体グリフの利用により，
> クスの高さが影響されたり，行ボックスの高さが改められることはない。 したがって，［下付き／上付き］文字異体は、複 column レイアウトな
> が一定に揃うことが重要になる状況においては，理想的なものになる。

例

<sub> 要素用の，代表的な［ UA 既定のスタイル ］：

```
sub {
  vertical-align: sub;
  font-size: smaller;
  line-height: normal;
}
```

古い UA においても依然として下付き文字が示されるように，タイポグラフィック下付き文字を指定するための、 font-variant-position の用法：

```
@supports ( font-variant-position: sub ) {

  sub {
    vertical-align: baseline;
    font-size: 100%;
    line-height: inherit;
    font-variant-position: sub;
  }

}
```

font-variant-position プロパティをサポートする UA は、下付き文字異体グリフを選定して，基底線や font-size の調整を伴わずに それを描画
になる。 より古い UA は、 font-variant-position プロパティ定義を無視して，下付き文字用の標準の既定を利用することになる。

↑

6.6. capital 化： font-variant-caps プロパティ

名前	***font-variant-caps***
値	normal | small-caps | all-small-caps | petite-caps | all-petite-caps | unicase | titling-caps
初期値	normal
適用対象	すべての要素
継承	される
百分率	受容しない
算出値	指定値
アニメーション型	不可

このプロパティは，［［ 小 capital や極小 capital ，あるいは題字 ］用に利用される代替グリフ ］の選定を可能にする。 これらのグリフは、周
のグリフに自然に溶け込むように，特にデザインされている － ウェイト，および［［ この目的に合わせるため，テキストが単純にリサイズされた
きに損なわれる，可読性 ］を保守するために。

各種値の意味は，次で与えられる：

normal

下に挙げられるどの特能も，可能化されない。

small-caps

小 capital （ OpenType 特能： smcp ）による表示を可能化する。 small-caps グリフには，概して大文字の形が利用されるが、そのサイズは
抑えられる。

all-small-caps

大文字と小文字の両者用に，小 capital （ OpenType 特能： c2sc, smcp ）による表示を可能化する。

petite-caps

極小 capital （ OpenType 特能： pcap ）による表示を可能化する。

all-petite-caps

大文字と小文字の両者用に，極小 capital （ OpenType 特能： c2pc, pcap ）による表示を可能化する。

unicase

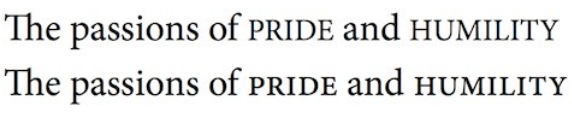
［ 大文字用の小 capital と，通常の小文字 ］の混在（ OpenType 特能： unic ）による表示を可能化する。 【大小アルファベットの一部は
残りは小文字の形に “統一” した上で（その結果，見かけ上の大小の区別が無くなる）、それらの大きさが一定に揃えられた書体】

titling-caps

題字用 capital （ OpenType 特能： titl ）による表示を可能化する。 大文字グリフは大抵，小文字と併用されるようにデザインされている
が大文字ばかりの題字【タイトルバックなど】に利用された場合、強く現れ過ぎになることがある。 題字用 capital は、この状況のために特
されている。

これらのグリフの可用性は、当該の特能が，フォントの特能リストの中に定義されているかどうかに基づく。 UA は，任意選択で用字系ごとにこれを
るが、文字ごとには，明示的に決めるべきでない。

一部のフォントには、このプロパティに述べられた特能のうち，一部分のみをサポートする，あるいは全くサポートしないものもある。 CSS 2.1 と
性を得るため、［ small-caps ／ all-small-caps ］が指定されつつ，所与のフォント用の small-caps グリフが可用でない場合、 UA は、 small-ca
トを模造するべきである － 例えば、通常のフォントから，大文字用のグリフを取り出して、その縮小バージョンにより，小文字グリフを（ all-smal
場合は，大文字のグリフも）置換するなど。



合成によるものと，本物の small-caps の比較対照

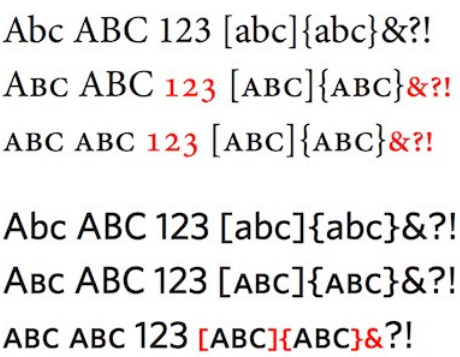
font-feature-settings プロパティは、模造された small-caps フォントを利用するかどうかには影響しない。

例

```
#example1 { font-variant-caps: small-caps; }
#example2 { font-variant-caps: small-caps; font-feature-settings: 'smcp' 0; }
```

small-caps をサポートしないフォントに対しては、 #example1, #example2 の両者とも合成された small-caps で描画されるべきである。 一方で, caps をサポートするフォントに対しては、 #example1 はネイティブ small-caps で, #example2 は（ネイティブ／合成された）small-caps なして れるべきである。

周囲のテキストと釣り合うようにするため、フォントは, これらの特能が可能化されたときに［ 大小が無い文字用の代替グリフ ］を供することがま が小 capital を模造するときは、［ 大小が無いと見なされている符号位置用の代替 ］の模造を試みないものとする。

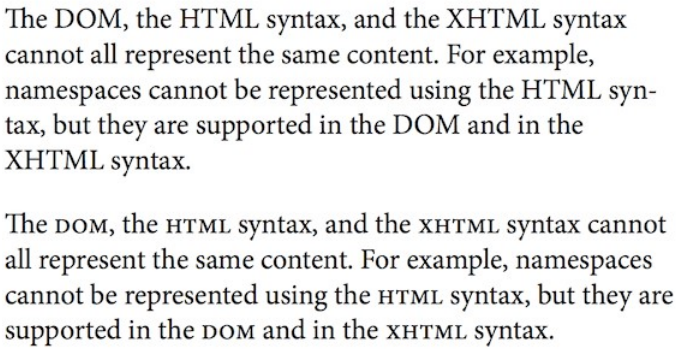


通常の表示（上半分／下半分の上段）と, ［ small-caps （同, 中段）, all-small-caps （同, 下段） ］が可能化された大小が無い文字

これらの特能をサポートしないフォントに対し［ petite-caps ／ all-petite-caps ］が指定された場合、 このプロパティは、［ small-caps ／ all-s ］（同順）が指定されていたかのようにふるまう。 その特能をサポートしないフォントに対し unicase が指定された場合、このプロパティは、 sma ［ “小文字化された”大文字 ］にのみ適用されていたかのようにふるまう 【どの大文字が “小文字化された”ものと見なされる？（次の段落に 。 この特能をサポートしないフォントに titling-caps が指定された場合、このプロパティによる可視効果はない。 模造による小 capital グリフ る下では、大文字と小文字を欠く用字系に対しては, ［ small-caps, all-small-caps, petite-caps, all-petite-caps, unicase ］による可視効果はない

小 capital を模造するために利用する大小変換は、 text-transform プロパティに利用されるものに合致するそれを利用するものとする。

最終的な結果として, normal フォントの中の縮小された大文字グリフが、テキストがすべて大文字として現れるように, small-caps フォントの中 置換し得る。



頭字語が頻出するテキストの可読性を向上させるための, 小 capital の利用

例

italic 化されて描画される引用文は、最初の行では, small-caps にもされる：

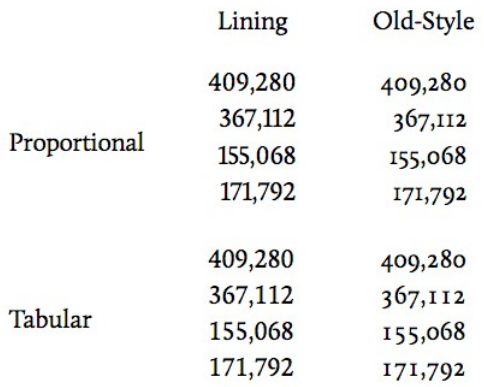
```
blockquote            { font-style: italic; }
blockquote:first-line { font-variant: small-caps; }

<blockquote>I'll be honor-bound to slap them like a haddock.</blockquote>
```

↑

6.7. 数を表す整形： font-variant-numeric プロパティ

名前	*font-variant-numeric*
値	normal | [*<numeric-figure-values>* || *<numeric-spacing-values>* || *<numeric-fraction-values>* || ordinal || slashed-zero]
初期値	normal
適用対象	すべての要素
継承	される
百分率	受容しない
算出値	指定値
アニメーション型	不可

数を表す形に対する制御を指定する。 下の例に、これらの値のいくつかを組み合せて，［ これらの特能をサポートするフォントによる，テーブルも
の描画 ］に波及させる方法を示す。 通常の段落テキスト内では，均衡（可変）幅の数字が利用される一方で、一連の［ 数字の縦列 ］を適正に整列
に，一定幅の数字が利用される：



数字スタイルの用例

可能な組み合せは：

<numeric-figure-values>
= [lining-nums | oldstyle-nums]

<numeric-spacing-values>
= [proportional-nums | tabular-nums]

<numeric-fraction-values>
= [diagonal-fractions | stacked-fractions]

各種値の意味は，次で与えられる：

normal
下に挙げられるどの特能も，可能化されない。

lining-nums
lining 数字（ OpenType 特能： lnum ）による表示を可能化する。

oldstyle-nums
old-style 数字（ OpenType 特能： onum ）による表示を可能化する。

proportional-nums
均衡（可変）幅の数字（ OpenType 特能： pnum ）による表示を可能化する。

tabular-nums
一定幅数字（ OpenType 特能： tnum ）による表示を可能化する。

↑

diagonal-fractions

斜め括線による分数（ OpenType 特能： `frac` ）による表示を可能化する。

stacked-fractions

水平括線による分数（ OpenType 特能： `afrc` ）による表示を可能化する。

ordinal

序数を伴なう普通字形（ OpenType 特能： `ordn` ）による表示を可能化する。

slashed-zero

スラッシュ付きのゼロ（ OpenType 特能： `zero` ）による表示を可能化する。

例

`ordinal` においては、序数形が上付き文字形と同じになることが多いが、それらのマークアップは異なる。 上付き文字に対しては、異体プロパテ
の上付き文字を包含している下位要素にのみ適用される。

```
sup { font-variant-position: super; }
x<sup>2</sup>
```

序数に対しては、異体プロパティは，接尾辞のみならず，序数全体（または包含している段落）に適用される：

```
.ordinal { font-variant-numeric: ordinal; }
<span class="ordinal">17th</span>
```

この事例では、 ″th″ のみが序数形で現れることになり，数字は不変のままにされる。 所与の言語に利用されているタイポグラフィック伝統様式
て、序数形は，上付き文字形から異なり得る。 例えば，イタリア語においては、序数形のデザインに下線が含められることがある。

例

自動的な分数と old-style 数字により描画される，簡単なレシピ：

```
.amount { font-variant-numeric: oldstyle-nums diagonal-fractions; }

<h4>脇腹肉のステーキマリネ：</h4>
<ul>
  <li><span class="amount">2</span> tbsp. オリーブ油</li>
  <li><span class="amount">1</span> tbsp. レモン汁</li>
  <li><span class="amount">1</span> tbsp. しょうゆ</li>
  <li><span class="amount">1 1/2</span> tbsp. タマネギのみじん切り</li>
  <li><span class="amount">2 1/2</span> tsp. イタリア系香草類</li>
  <li>塩，こしょう</li>
</ul>

<p>肉とマリネードを混ぜ，覆いを掛けて冷蔵庫に数時間か一晩程置く。</p>
```

分数の特能は，段落全体でない所にのみ適用されることに注意。 フォントは大抵，この特能をスラッシュ（ `/` ）文字の利用に基づく文脈に応じた
用して実装する。 そのようなわけで、それは，段落レベルのスタイル用途には相応しくない。

↑

6.8. 東アジア圏のテキストの描画： font-variant-east-asian プロパティ

名前	*font-variant-east-asian*
値	normal | [*<east-asian-variant-values>* || *<east-asian-width-values>* || ruby]
初期値	normal
適用対象	すべての要素
継承	される
百分率	受容しない
算出値	指定値
アニメーション型	不可

東アジア圏のテキストにおけるグリフの［ 代用とサイズ法 ］の制御を可能にする。

<east-asian-variant-values>
= [jis78 | jis83 | jis90 | jis04 | simplified | traditional]

<east-asian-width-values>
= [full-width | proportional-width]

各種値の意味は，次で与えられる：

normal

下に挙げられるどの特能も，可能化されない。

jis78

JIS78 形（ OpenType 特能： jp78 ）の描画を可能化する。

jis83

JIS83 形（ OpenType 特能： jp83 ）の描画を可能化する。

jis90

JIS90 形（ OpenType 特能： jp90 ）の描画を可能化する。

jis04

JIS2004 形（ OpenType 特能： jp04 ）の描画を可能化する。

各種 日本語の国家標準にて定義されるグリフ形が，種々の JIS 異体に反映される。 フォントは一般に，最新の国家標準により定義されるグリ
が、例えば，サイネージに合致させるために、より古い異体の利用が必要とされることもある。

simplified

簡略化形（ OpenType 特能： smpl ）の描画を可能化する。

traditional

伝統的形（ OpenType 特能： trad ）の描画を可能化する。

値 simplified と traditional は，［［［ 年月に渡り簡略化されてきた形であるが，一部の文脈においては，より古い伝統的な形も依然として利用さ
］ような文字 ］用のグリフ形 ］に対する制御を可能にする。 ［ 文字とグリフ形 ］の正確な集合は、所与のフォントがデザインされた文脈により
変わり得ることになる。

full-width

全角異体（ OpenType 特能： fwid ）の描画を可能化する。

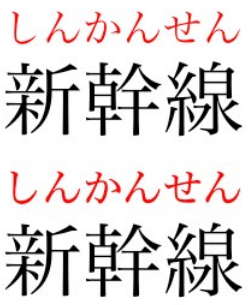
proportional-width

均衡（可変）幅の異体（ OpenType 特能： pwid ）の描画を可能化する。

↑

ruby

ルビ異体グリフ（ OpenType 特能： `ruby` ）による表示を可能化する。 ルビテキストは一般に，結び付けられている本文テキストより小さいの
ントデザイナは，ルビ用途の，［ 既定のグリフの縮小バージョン ］より読み易い，特別なグリフをデザインすることがある。 グリフ選定のア
れ、［ 結び付けられているフォントの拡縮法 ］や［ テキスト行のレイアウトに影響するような他の変化 ］は生じない。 下の赤色のルビテキ
既定のグリフによるもの（上段）と，ルビ異体グリフによるもの（下段）を示す。 描線の太さに若干の相違があることに注意。



6.9. フォント描画用の全般的な略式： font-variant プロパティ

名前	*font-variant*
値	normal | none | [*<common-lig-values>* || *<discretionary-lig-values>* || *<historical-lig-values>* || *<contextual-alt-values>* || [small-caps | all-small-caps | petite-caps | all-petite-caps | unicase | titling-caps] || *<numeric-figure-values>* || *<numeric-spacing-values>* || *<numeric-fraction-values>* || ordinal || slashed-zero || *<east-asian-variant-values>* || *<east-asian-width-values>* || ruby || [sub | super]]
初期値	normal
適用対象	個々のプロパティを見よ
継承	個々のプロパティを見よ
百分率	個々のプロパティを見よ
算出値	個々のプロパティを見よ
アニメーション型	個々のプロパティを見よ

font-variant プロパティは、そのすべての下位プロパティ用の略式プロパティである。 値 ***normal*** は、 font-variant のすべての下位プロパティを，の初期値に再設定する。 値 ***none*** は、 font-variant-ligatures を none に設定し，他のすべてのフォント特能プロパティを それぞれの初期値に再設
他の略式プロパティと同様、 font-variantの利用により，未指定の font-variant 下位プロパティは，それぞれの初期値に再設定される。 それは、
font-feature-settings の値は再設定しない。

↑

6.10. 低次のフォント特能設定群の制御： font-feature-settings プロパティ

名前	*font-feature-settings*
値	normal | <*feature-tag-value*>#
初期値	normal
適用対象	すべての要素
継承	される
百分率	受容しない
算出値	指定値
アニメーション型	不可

このプロパティは、 OpenType フォント特能に対する低次の制御を供する。 これには、［［［［［ 広範には利用されてはいないが，ある特定の利用必要になる ］ようなフォント特能 ］に対するアクセス ］を供する ］ための用途 ］が意図されている。

> 作者は、一般に，可能な所では［ font-variant とその各種 下位プロパティ ］を利用し、［［ たまにしか利用されない，ある特定のフォント特能クセスする手段が，他にない ］ような特別な場合に限って，このプロパティを利用するべきである。

```
/*  small-caps を可能化して，2 番目の swash 代替を利用する  */
font-feature-settings: "smcp", "swsh" 2;
```

値 ***normal*** は、このプロパティに因り，グリフ選定や位置決めが変わることはないことを意味する。

特能タグ値の構文は次で与えられる：

```
<feature-tag-value>
     = <string> [ <integer> | on | off ]?
```

<string> は OpenType 特能タグであり，文字大小は区別される。 OpenType 仕様 [OPENTYPE] による指定に従い，特能タグは 4 個の ASCII文字を［ 長さが 4 文字に一致しない ］，あるいは［ 符号位置の範囲 U+20-7E に入らない文字を包含している ］タグ文字列は、無効である。 特能タグォントの中で定義される特能タグ ］に合致させるのみで十分なので、それらは［ 明示的に登録されている OpenType 特能 ］に制限されない。 カスタグを定義するフォントは、 OpenType 仕様 [OPENTYPE-FEATURES] に定義される タグ名規則 に従うべきである。

フォント内に無い特能タグは，無視される － UA は、［ そのような特能タグに基づく，フォールバックのふるまい ］を合成しようと試みないものと一の例外として，UA は、［［ kern テーブルの形でカーニングデータを包含するが， GPOS テーブルの中では kern 特能のサポートを欠いている ］ント ］による kern 特能を，合成によりサポートしてもよい。

> 注記： 一般に，カーニングを明示的に可能化／不能化するためには、作者は font-kerning プロパティを利用するべきである。 このプロパティはちらの種類のカーニングデータを伴うフォントにも影響するので。

特能タグがフォント内に在るならば、 *<integer>* 値は，グリフ選定用に利用される index を指示する。 値は 0 以上でなければならない。 値 0 不能化することを指示する。 真偽値の特能に対する値 1 は、その特能を可能化する。 真偽値でない特能に対する 1 以上の値は、その特能を可能化時に，その特能選定 index を指示する。 値 on は 1 と同義であり， off は 0 と同義である。 省略された場合の値は， 1 と見做される。

font-feature-settings の算出値はマップなので、指定値内の duplicates は保全しないものとする。 同じタグ名を伴う *<feature-tag-value>* が複た場合、それらのうち最後のものが他を上書きする。

```
font-feature-settings: "dlig" 1;
    /*  dlig=1 は随意合字を可能化する  */
font-feature-settings: "smcp" on;
    /*  smcp=1 は small-caps を可能化する  */
font-feature-settings: 'c2sc';
    /*  c2sc=1 は capital →小 capital （ caps to small caps ）を可能化する  */
font-feature-settings: "liga" off;
    /*  liga=0 は共通合字を不要とする  */
font-feature-settings: "tnum", 'hist';
    /*  tnum=1, hist=1 は 一定幅数字，および歴史的形を可能化する  */
font-feature-settings: "tnum" "hist";
    /*  無効 － カンマ区切りのリストにする必要がある  */
font-feature-settings: "silly" off;
    /*  無効 － タグが長過ぎる  */
font-feature-settings: "PKRN";
    /*  PKRN=1 はカスタム特能を可能化する  */
font-feature-settings: dlig;
    /*  無効 － タグは【識別子ではなく，】文字列でなければならない  */
```

↑

［ フォントからサポートされる範囲 ］より大きい値が指定されているときのふるまいは、明示的に未定義になる。 真偽値の特能に対しては、これに，その特能を可能化することになる。 真偽値でない特能に対しては、範囲外の値は，一般に値 0 と等価になる。 しかしながら，いずれの場合もるまいは，フォントがどのようにデザインされているかに（具体的には，特能が どの種別の表引きを利用して定義されているかに）依存することに

特能タグは，特に OpenType 特能タグ用に定義されているが、将来においては， フォント特能をサポートする他の現代のフォント形式用に，特能タれ得る。 可能な所では，他のフォント形式用に定義される特能は、登録された OpenType タグのパタンに従うよう試みられるべきである。

例
下の日本語テキストは、半角カナ文字により描画されることになる：

```
body { font-feature-settings: "hwid"; /*  半角 OpenType 特能  */ }

<p>毎日カレー食べてるのに、飽きない</p>
```

7. フォント特能解決

前節にて述べた様に、フォント特能は，［ スタイル規則の中，あるいは@font-face 規則内 ］における［ font-variant や font-feature-settings ］介して，種々の仕方で可能化され得る。 ［ これらの設定の和集合 ］用の解決順序は、以下にて定義される。 CSS プロパティを介して定義される各は、レイアウトエンジンの既定の特能の上層で適用される。

7.1. 既定の特能

OpenType フォントに対しては、 UA は，［ OpenType の文献に与えられた［ 用字系，および書字モード ］用に定義されている既定の特能 ］を可能のとする。 要求される［ 合字，共通合字，文脈に応じた形（ OpenType 特能： rlig，liga，clig，calt ）］は、［ 地域化された形（ OpenType 特）］，および［［ 組にされた［ 文字やマーク（ OpenType 特能： ccmp，mark，mkmk ）］の適正な表示 ］用に要求される特能 ］に沿うように，既するものとする。 これらの特能は、［ font-variant，font-feature-settings ］プロパティの値が normal のときでも，常に可能化するものとする。能が不能化されるのは、作者から明示的に上書きされたときに限られる － font-variant-ligatures が no-common-ligatures に設定されたときなど。[ARABIC-TYPO]，Khmer，Devanagari ］などの複階的な用字系を取り扱うためには、追加の特能が要求される。 縦書きのテキスト連なり内の正立（テキスト用には、縦書きの代替（ OpenType 特能： vert ）が可能化するものとする。 【例えば括弧類は、横書き用のものと異ならせる必要がある。

7.2. 特能の優先度

［ 一般の／フォント特有の ］フォント特能プロパティ設定群は、下に与える順序 － 後のもの程，優先度は高いとする － により，解決される。 こは、所与のテキスト連なりに影響するような［ 一連のフォント特能からなる，一つに組み合わされたリスト ］を構築するために利用される。

1. フォント特能は、所与の用字系用に要求される特能も含め，既定で可能化される
2. フォントが @font-face 規則を介して定義される場合、そのフォント特能は，［ @font-face 規則の中の font-feature-settings 記述子 ］にされる。
3. フォント特能は、［［ font-variant プロパティとその各種 下位プロパティ ］，および［ OpenType特能を利用する（例えば，font-kerning プ他の CSS プロパティ ］］の値により黙示される。
4. ［ font-variant や font-feature-settings ］以外のプロパティにより決定される特能設定群。 例えば， letter-spacing プロパティ用に既定で設定することにより、共通合字は不能化される。
5. font-feature-settings プロパティの値により黙示されるフォント特能。

この順序付けにより、作者は，［ @font-face 規則の中のフォント用の，一般の既定設定の集合 ］を与えた上で、特定の要素用には，プロパティ設り それらを上書きできるようになる。 一般プロパティ設定群は， @font-face 規則の中の設定群を上書きし、低次のフォント特能設定群は， fontプロパティ設定群を上書きする。

［ 一連のフォント特能からなる，一つに組み合わされたリスト ］が，同じ特能用の値を複数個 包含するような状況下では、それらのうち最後の値る。 フォントが［ 所与の下層のフォント特能 ］用のサポートを欠いているときは， テキストは、単純に，そのフォント特能が可能化されなかったに，描画される － 特定のプロパティ用に明示的に定義されている所を除き，フォントフォールバックは生じず，特能の合成は試みられない。

7.3. 特能の優先度の例

例
下の一連のスタイルにより，一連の数字は， 段落内で利用されるときには均衡（可変）幅に描画される一方で，価格表（ table.prices ）内では示される。

↑

```
body {
  font-variant-numeric: proportional-nums;
}

table.prices td {
  font-variant-numeric: tabular-nums;
}
```

8. オブジェクトモデル

@font-face 規則の内容は、 CSS Object Model に対する次の拡張を介して，アクセスできる。

8.1. CSSFontFaceRule インタフェース

CSSFontFaceRule インタフェースは、 @font-face 規則を表現する。

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

付録 A： プラットフォームフォントプロパティから CSS プロパティへの対応付け

この付録は、他の節にて述べられた問題と状況の一部を説明するための，背景情報として収録されている。 あくまで参考として読まれるべきである。

CSS におけるフォントプロパティは、利用される下層のフォント形式に依存しないように設計されている － それらは，よくある［ TrueType ， OpenT
ォントに加えて［ ビットマップ，Type1 ， SVG ］フォントの指定にも利用し得る。 しかしながら，［ TrueType ， OpenType ］形式には、作者の混
く，異なるプラットフォームにわたる実装にも難題を突きつける様相がある。

元々は Apple により開発された TrueType [TRUETYPE] は、スクリーンと印刷 両用のアウトラインフォント形式として設計されている。 Microsoft
の TrueType 形式の開発に参加して以来、 TrueType フォントは，両プラットフォームでサポートされるようになった。 TrueType 形式の中のフォン
は、共通の［ 4 字で記されるタグ名 ］により判別され，それぞれが特定の型のデータを包含する，テーブルの集合からなる。 例えば，著作権や許
含む命名情報は、 name テーブルに格納される。 文字マップ （ cmap ）テーブルは、文字符号化法たちからグリフたちへの対応付けを包含する。 Ap
に，タイポグラフィックの機能強化をサポートするためのテーブルを追加した。 これらは今では， Apple Advanced Typography，略して AAT フォン
れている。 Microsoft と Adobe は、先進的タイポグラフィ用に別々のテーブルの集合を開発し，それらの形式を OpenType [OPENTYPE] と称した。
仕様は ISO にて Open Font Format [OPEN-FONT-FORMAT] として標準化されている。

TrueType 形式においては，プラットフォーム間に渡るバリエーションが明示的に許容されていたので、 Microsoft Windows や Linux の下で利用さ
トデータは，多くの場合， Apple の Mac OS X の下で利用されるデータと少しばかり異なっている。 これには［ フォント計量，名前，文字マップ
も含まれる。

具体的には、フォントファミリ名データの取り扱いは，プラットフォーム間に渡り異なっている。 ［ TrueType ／ OpenType ］フォントに対しては、
名前は［ name テーブルの中の，nameID = 1 の name record ］に包含される。 異なるロケール用に複数の名前を格納させることもできるが、 Micr
フォントには常に少なくとも US 英語版の名前を含ませることを推奨している。 Windows 上では、 Microsoft は － 後方互換性を得るため － このフ
に対する書体数の上限を 4 個までとすることに決めた。 より大きなグループ分け “preferred ファミリ” （ nameID = 16 ）や “WWS ファミリ”
= 21 ）も利用できる。 OSX などの他のプラットフォームには，この制限は無く、ファミリ名は，可能なすべてのグループ分けを定義するために利用

他の 名前テーブル データは、ファミリ内の特定の書体を一意に識別するための名前を供する。 全部的フォント名 （ nameID = 4 ），および Postsc
nameID = 6 ）は、単独の書体を一意に記述する。 例えば， *Gill Sans* ファミリの bold 体は、全部名 *Gill Sans Bold*，および Postscript 名 *Gil*
を持つ。 所与の書体用の［ 地域化された版による全部名 ］は複数個あり得るが、 Postscript 名は常に，一定の ASCII 文字のみからなる一意な名
る。

種々のプラットフォーム上で、同じフォントを検索する際に，異なる名前が利用されている。 例えば， Windows GDI CreateIndirectFont API におい
を表引きする際に ファミリや全部名を利用し得る一方で、 Mac OS X 上では，所与の書体を表引きする際に全部名や Postscript 名を利用する
CTFontCreateWithName API call が利用される。 Linux の下では、 fontconfig API により，これらのどの名前を利用するフォントの検索も許容されて
ラットフォーム API による，［ 他のフォント候補への自動的な代用 ］がある状況下では、返されたフォントが所与の名前に合致するかどうかの検
されることもある。

所与の書体のウェイトは、 OS/2 テーブル の usWeightClass フィールドを介して決定され得るか，あるいは スタイル名 （ nameID = 2 ）から推定さ
字幅も同様に OS/2 テーブルの usWidthClass を介して決定され得るか，あるいは スタイル名から推定され得る。 Windows GDI API の下での［ 200
ェイトに対する合成による bold 化 ］に関係する歴史的な理由から、フォントデザイナは、これらのウェイトを避けるため， OS/2 テーブルの中の
えていることがある。

［ タイ語，アラビア語，デヴァーナーガリーなどの，文脈に応じた形状付けを利用する複階的な用字系 ］の描画には、 OpenType や AAT フォント
特能を要する。 今では、複階的な用字系の描画は、 Windows および Linux では，OpenType フォント特能によりサポートされている一方で、 Mac
では，［ OpenType ， AAT ］両フォント特能が利用されている。

変更点

2018 年 8 月 14 日 CSS3 Fonts Proposed Recommendation からの変更点：

変更点における CSS Fonts 4 に移動された特色機能についての言及に， その仕様の対応する節へのリンクを付与した。

W3C 勧告用の日付と boilerplate 用の更新。

Unicode 参照文献を最新バージョンに更新した。

この変更点節を更新した。

2018 年 3 月 15 日 CSS3 Fonts 勧告候補 からの変更点：

`font-variant` 記述子は、実装の欠如により CSS Fonts 4 へ移動された。

`@font-feature-values` at-規則は、実装の欠如により CSS Fonts 4 へ移動された。

未知の素片識別子に対する取り扱いを明確化した。

`<length-percentage>` の定義を [CSS-VALUES] へリンクした

2013 年 10 月付 CSS3 Fonts 勧告候補 からの変更点：

`font-language-override` プロパティは、 CSS Fonts 4 へ移動された。

`CSSFontFeatureValuesRule` インタフェースは、 CSS Fonts 4 へ移動された。

`CSSFontFaceRule` インタフェースは、 DOM level 2 style から広く実装されているものに復帰された

汎用フォントファミリは組成書体になり得ることを明確化した。

可用な最初のフォントは、 U+0020 (space) 文字に合致するものになることを明確化した。

`small-caps` の合成が `font-feature-settings` とどう相互作用するかを明確化した。

すべての CSS キーワードは、フォントファミリ名として無効になるものとした。

`font` 略式は `font-synthesis` を再設定しないことを明確化した。

“システムフォント” ではなく “インストールされたフォント” を利用するようにした。

`font-family` や `src` を欠いている不正な形の ***`@font-face`*** 規則であっても、 DOM 内には依然として現れるが，フォント選定には影響しないこ
した。

相対サイズ†用の慣例の比率範囲††を明確化した – それが絶対的キーワードサイズ†††を改変していない††††ときの。 【† *`<relati`*
？／††？／††† *`<absolute-size>`* ？／††††？】

`font-variation-settings` †, `font-feature-settings`, 両者に対し，算出値はマップになる††ことを明確化した（したがって，指定された dupe
除去された）。 【†レベル 4 仕様／††？／††††？】

抜けていた `font-variant-position` 値を `font-variant` 略式プロパティに追加した。

`font-size-adjust` に対する負の値は `font-size` に対する負の百分率値とともに無効にした。

UA が［下付き／上付き］文字の計量に OS/2 テーブルを利用する要件は除去した。

[CSS-TEXT-3] による演算順序へのリンク（参考）を追加した。

RFC 8081 によるフォントの最上位 種別への規範的リンクを追加した。

小さな編集上の整理。

謝辞

Tal Leming, Jonathan Kew, Ken Lunde, Christopher Slye 各氏からの助力とフィードバックに感謝する。 John Hudson 氏からは、緻密で複雑な Op
語タグの説明に十分な時間を割いていただき， Byzantine seal 上でのテキスト表示用に，文字異体の用例も供していただいた。 Ken Lunde 氏と Er
氏からは、 CJK OpenType 特能と Unicode 異体字選択子についての有益なフィードバックを供していただいた。 一連の `font-variant` 下位プロパテ
よるフォント特能のサポートのアイデアは， Håkon Wium Lie, Adam Twardoch, Tal Leming 三氏による発案である。 Elika Etemad 氏は `@font-fea`
`values` 規則用の初期設計案のいくつかを供された。 随意合字の例における Ed Interlock の利用を承諾していただいた House Industries にも感謝

Robert Bringhurst 氏による，卓越した mind expansion： *The Elements of Typographic Style* に特別な謝意を。

適合性

【この節の内容はCSS 日本語訳 共通ページに委譲。】

↑

参照文献

文献（規範）

[CSS-VALUES]

Tab Atkins Jr.; Elika Etemad. CSS Values and Units Module Level 3 29 September 2016. CR.

https://www.w3.org/TR/css-values/

日本語訳 Level 4 日本語訳2

[FETCH]

Fetch. WhatWG Living Standard.

https://fetch.spec.whatwg.org/

日本語訳

[OPENTYPE]

OpenType specification. Microsoft.

http://www.microsoft.com/typography/otspec/default.htm

[OPENTYPE-FEATURES]

OpenType feature registry. Microsoft.

http://www.microsoft.com/typography/otspec/featurelist.htm

[RFC2119]

S. Bradner. Key words for use in RFCs to Indicate Requirement Levels. RFC 2119.

http://www.ietf.org/rfc/rfc2119.txt

日本語訳 日本語訳2 日本語訳3 日本語訳4

[RFC8081]

C. Lilley. The "font" Top-Level Media Type. February 2017. Proposed Standard.

https://tools.ietf.org/html/rfc8081

[UAX15]

Mark Davis; Ken Whistler. Unicode Normalization Forms. 31 August 2012. Unicode Standard Annex #15.

http://www.unicode.org/reports/tr15/

[UNICODE]

The Unicode Standard

http://www.unicode.org/versions/latest

文献（参考）

[AAT-FEATURES]

Apple Advanced Typography font feature registry. Apple.

https://developer.apple.com/fonts/TrueType-Reference-Manual/RM09/AppendixF.html

[ARABIC-TYPO]

Huda Smitshuijzen AbiFares. Arabic Typography: A Comprehensive Sourcebook. Saqi Books. 2001. ISBN 0-86356-347-3.

[CHARMOD]

Martin J. Dürst; et al. Character Model for the World Wide Web 1.0: Fundamentals. 15 February 2005. W3C Recommendation.

http://www.w3.org/TR/2005/REC-charmod-20050215/

日本語訳

[CHARMOD-NORM]

Addison Phillips. Character Model for the World Wide Web: String Matching. 20 April 2018. W3C Working Draft. (Work in progress.)

https://www.w3.org/TR/2018/WD-charmod-norm-20180420/

[CSS-TEXT-3]

Elika J. Etemad / fantasai; Koji Ishii. CSS Text Module Level 3. 22 August 2017. W3C Working Draft. (Work in progress.)

https://www.w3.org/TR/2017/WD-css-text-3-20170822/

日本語訳 日本語訳2

[CSS3-CONDITIONAL]

L. David Baron. CSS Conditional Rules Module Level 3. 4 April 2013. W3C Candidate Recommendation. (Work in progress.)

http://www.w3.org/TR/2013/CR-css3-conditional-20130404/

日本語訳

[OPEN-FONT-FORMAT]

Information technology - Coding of audio-visual objects - Part 22: Open Font Format. International Organization for Standardizat ISO/IEC 14496-22:2009.

http://standards.iso.org/ittf/PubliclyAvailableStandards/c052136_ISO_IEC_14496-22_2009%28E%29.zip

[OPENTYPE-FONT-GUIDE]

OpenType User Guide. FontShop International.

http://www.fontblog.de/wp-content/uploads/2015/11/FF_OTF_user_guide.pdf

[TRUETYPE]

TrueType™ Reference Manual. Apple.

https://developer.apple.com/fonts/TrueType-Reference-Manual/

[UAX29]

Mark Davis. Unicode Text Segmentation. 12 September 2012. Unicode Standard Annex #29.

http://www.unicode.org/reports/tr29/

[WINDOWS-GLYPH-PROC]

John Hudson. Windows Glyph Processing. Microsoft Typogrraphy.

http://www.microsoft.com/typography/developers/opentype/default.htm

↑