MathML Core

W3C Candidate Recommendation Snapshot 24 June 2025

▼ **More details about this document**

This version:
https://www.w3.org/TR/2025/CR-mathml-core-20250624/

Latest published version:
https://www.w3.org/TR/mathml-core/

Latest editor's draft:
https://w3c.github.io/mathml-core/

History:
https://www.w3.org/standards/history/mathml-core/
Commit history

Test suite:
https://github.com/web-platform-tests/wpt/tree/master/mathml/

Implementation report:
https://wpt.fyi/results/?label=master&label=experimental&aligned&q=math%20%20not%28path%3A%2Fjs%29

Editors:
David Carlisle (NAG)
Frédéric Wang (Igalia)

Former editors:
Patrick Ion (Mathematical Reviews, American Mathematical Society)
Robert Miner (deceased) (Design Science, Inc.)

Feedback:
GitHub w3c/mathml-core (pull requests, new issue, open issues)

Abstract

This specification defines a core subset of Mathematical Markup Language, or MathML, that is suitable for browser implementation. MathML is a markup language for describing mathematical notation and capturing both its structure and content. The goal of MathML is to enable mathematics to

be served, received, and processed on the World Wide Web, just as HTML has enabled this functionality for text.

Status of This Document

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

This document was published by the Math Working Group as a Candidate Recommendation Snapshot using the Recommendation track.

Publication as a Candidate Recommendation does not imply endorsement by W3C and its Members. A Candidate Recommendation Snapshot has received wide review, is intended to gather implementation experience, and has commitments from Working Group members to royalty-free licensing for implementations.

This Candidate Recommendation is not expected to advance to Proposed Recommendation any earlier than 30 September 2025.

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 03 November 2023 W3C Process Document.

Table of Contents

§ 1. Introduction

This section is non-normative.

The [*MATHML3*] specification has several shortcomings that make it hard to implement consistently across web rendering engines or to extend with user-defined constructions, e.g.:

- It is a huge and standalone specification.

- It does not contain any detailed rendering rules.
- It is not driven by browser-implementation.
- It lacks automated testing.

This MathML Core specification intends to address these issues by being as accurate as possible on the visual rendering of mathematical formulas using additional rules from the TeXBook's Appendix G [*TEXBOOK*] and from the Open Font Format [*OPEN-FONT-FORMAT*], [*OPEN-TYPE-MATH-ILLUMINATED*]. It also relies on modern browser implementations and web technologies [*HTML*] [*SVG*] [*CSS2*] [*DOM*], clarifying interactions with them when needed or introducing new low-level primitives to improve the web platform layering.

Parts of MathML3 that do not fit well in this framework or are less fundamental have been omitted. Instead, they are described in a separate and larger [*MATHML4*] specification. The details of which math feature will be included in future versions of MathML Core or implemented as polyfills is still open. This question and other potential improvements are tracked on GitHub.

By increasing the level of implementation details, focusing on a workable subset, following a browser-driven design and relying on automated web platform tests, this specification is expected to greatly improve MathML interoperability. Moreover, effort on MathML layering will enable users to implement the rest of the MathML 4 specification, or more generally to extend MathML Core, using modern web technologies such as shadow trees, custom elements or APIs from [*HOUDINI*].

2. MathML Fundamentals

2.1 Elements and attributes

The term ***MathML element*** refers to any element in the MathML namespace. The MathML elements defined in this specification are called the ***MathML Core elements*** and are listed below. Any MathML element that is not listed below is called an ***Unknown MathML element***.

1. `annotation`
2. `annotation-xml`
3. `maction`
4. `math`

5. merror
6. mfrac
7. mi
8. mmultiscripts
9. mn
10. mo
11. mover
12. mpadded
13. mphantom
14. mprescripts
15. mroot
16. mrow
17. ms
18. mspace
19. msqrt
20. mstyle
21. msub
22. msubsup
23. msup
24. mtable
25. mtd
26. mtext
27. mtr
28. munder
29. munderover
30. semantics

The ***grouping elements*** are maction, math, merror, mphantom, mprescripts, mrow, mstyle, semantics and unknown MathML elements.

The ***scripted elements*** are mmultiscripts, mover, msub, msubsup, msup, munder and

munderover.

The ***radical elements*** are mroot and msqrt.

The attributes defined in this specification have no namespace and are called ***MathML attributes***:

- Global attributes
- Legacy maction attributes
- mo attributes
- mpadded attributes
- mspace attributes
- munderover attributes
- mtd attributes
- encoding
- display
- linethickness

2.1.1 The Top-Level <math> Element

MathML specifies a single top-level or root ***math*** element, which encapsulates each instance of MathML markup within a document. All other MathML content must be contained in a <math> element.

The <math> element accepts the attributes described in 2.1.3 Global Attributes as well as the following attributes:

- display
- alttext

The ***display*** attribute, if present, must be an ASCII case-insensitive match to block or inline. The user agent stylesheet described in A. User Agent Stylesheet contains rules for this attribute that affect the default values for the display (block math or inline math) and math-style (normal or compact) properties. If the display attribute is absent or has an invalid value, the User Agent stylesheet treats it the same as inline.

This specification does not define any observable behavior that is specific to the ***alttext*** attribute.

> NOTE
> The `alttext` attribute may be used as alternative text by some legacy systems that do not implement math layout.

If the `<math>` element does not have its computed `display` property equal to `block math` or `inline math` then it is laid out according to the CSS specification where the corresponding value is described. Otherwise the layout algorithm of the `mrow` element is used to produce a math content box. That math content box is used as the content for the layout of the element, as described by CSS for `display: block` (if the computed value is `block math`) or `display: inline` (if the computed value is `inline math`). Additionally, if the computed `display` property is equal to `block math` then that math content box is rendered horizontally centered within the content box.

> NOTE
> TEX's display mode `$$...$$` and inline mode `$...$` correspond to `display="block"` and `display="inline"` respectively.

In the following example, a `math` formula is rendered in display mode on a new line and taking full width, with the math content centered within the container:

```
<div style="width: 15em;">
  This mathematical formula with a big summation and the number pi
  <math display="block" style="border: 1px dotted black;">
    <mrow>
      <munderover>
        <mo>∑</mo>
        <mrow><mi>n</mi><mo>=</mo><mn>1</mn></mrow>
        <mrow><mo>+</mo><mn>∞</mn></mrow>
      </munderover>
      <mfrac>
        <mn>1</mn>
        <msup><mi>n</mi><mn>2</mn></msup>
      </mfrac>
    </mrow>
    <mo>=</mo>
    <mfrac>
      <msup><mi>π</mi><mn>2</mn></msup>
      <mn>6</mn>
    </mfrac>
  </math>
  is easy to prove.
</div>
```

This mathematical formula with a big summation and the number pi

$$\sum_{n=1}^{+\infty} \frac{1}{n^2} = \frac{\pi^2}{6}$$

is easy to prove.

As a comparison, the same formula would look as follows in inline mode. The formula is embedded in the paragraph of text without forced line breaking. The baselines specified by the layout algorithm of the `mrow` are used for vertical alignment. Note that the middle of sum and equal symbols or fractions are all aligned, but not with the alphabetical baseline of the surrounding text.

> This mathematical formula with a big summation and the number pi $\sum_{n=1}^{+\infty} \frac{1}{n^2} = \frac{\pi^2}{6}$ is easy to prove.

Because good mathematical rendering requires use of mathematical fonts, the user agent stylesheet should set the font-family to the math value on the `<math>` element instead of inheriting it. Additionally, several CSS properties that can be set on a parent container such as `font-style`, `font-weight`, `direction` or `text-indent` etc are not expected to apply to the math formula and so the user agent stylesheet has rules to reset them by default.

```
math {
  direction: ltr;
  text-indent: 0;
  letter-spacing: normal;
  line-height: normal;
  word-spacing: normal;
  font-family: math;
  font-size: inherit;
  font-style: normal;
  font-weight: normal;
  display: inline math;
  math-shift: normal;
  math-style: compact;
  math-depth: 0;
}
math[display="block" i] {
  display: block math;
  math-style: normal;
}
math[display="inline" i] {
  display: inline math;
  math-style: compact;
}
```

§ 2.1.2 Types for MathML Attribute Values

In addition to CSS data types, some MathML attributes rely on the following MathML-specific types:

unsigned-integer
: An <integer> value as defined in [*CSS-VALUES-4*], whose first character is neither U+002D HYPHEN-MINUS character (-) nor U+002B PLUS SIGN (+).

boolean
: A string that is an ASCII case-insensitive match to `true` or `false`.

2.1.3 Global Attributes

The following attributes are common to and may be specified on all MathML elements:

- `autofocus`
- `class`
- `data-*`
- `dir`
- `displaystyle`
- `id`
- `mathbackground`
- `mathcolor`
- `mathsize`
- `nonce`
- `scriptlevel`
- `style`
- `tabindex`
- `on*` event handler attributes
- additional attributes, as described in 2.1.7 Attributes Reserved as Valid

2.1.4 Attributes common to HTML and MathML elements

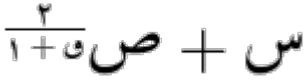
The ***id***, ***class***, ***style***, ***data-****, ***autofocus*** and ***nonce*** and ***tabindex*** attributes have the same syntax and semantics as defined for `id`, `class`, `style`, data-*, `autofocus`, `nonce` and `tabindex` attributes on HTML elements.

The ***dir*** attribute, if present, must be an ASCII case-insensitive match to `ltr` or `rtl`. In that case, the user agent is expected to treat the attribute as a presentational hint setting the element's direction property to the corresponding value. More precisely, an ASCII case-insensitive match to `rtl` is mapped to `rtl` while an ASCII case-insensitive match to `ltr` is mapped to `ltr`.

NOTE
The dir attribute is used to set the directionality of math formulas, which is often `rtl` in Arabic speaking world. However, languages written from right to left often embed math written from left to right and so the user agent stylesheet resets the direction property accordingly on the `math` elements.

In the following example, the dir attribute is used to render "س plus ص raised to the power of (٢ over, ١ plus ف)" from right-to-left.

```
<math dir="rtl">
  <mrow>
    <mi>س</mi>
    <mo>+</mo>
    <msup>
      <mi>ص</mi>
      <mfrac>
        <mn>٢</mn>
        <mrow>
          <mi>ف</mi>
          <mo>+</mo>
          <mn>١</mn>
        </mrow>
      </mfrac>
    </msup>
  </mrow>
</math>
```



All MathML elements support event handler content attributes, as described in event handler content

attributes in HTML.

All event handler content attributes noted by HTML as being supported by all HTMLElements are supported by all MathML elements as well, as defined in the MathMLElement IDL.

2.1.5 Legacy MathML Style Attributes

The ***mathcolor*** and ***mathbackground*** attributes, if present, must have a value that is a <color>. In that case, the user agent is expected to treat these attributes as a presentational hint setting the element's color and background-color properties to the corresponding values. The `mathcolor` attribute describes the foreground fill color of MathML text, bars etc while the `mathbackground` attribute describes the background color of an element.

The ***mathsize*** attribute, if present, must have a value that is a valid <length-percentage>. In that case, the user agent is expected to treat the attribute as a presentational hint setting the element's font-size property to the corresponding value. The `mathsize` property indicates the desired height of glyphs in math formulas but also scales other parts (spacing, shifts, line thickness of bars etc) accordingly.

> NOTE
> The above attributes are implemented for compatibility with full MathML. Authors whose only target is MathML Core are encouraged to use CSS for styling.

2.1.6 The `displaystyle` and `scriptlevel` attributes

The ***displaystyle*** attribute, if present, must have a value that is a boolean. In that case, the user agent is expected to treat the attribute as a presentational hint setting the element's `math-style` property to the corresponding value. More precisely, an ASCII case-insensitive match to `true` is mapped to `normal` while an ASCII case-insensitive match to `false` is mapped to `compact`. This attribute indicates whether formulas should try to minimize the logical height (value is `false`) or not (value is `true`) e.g. by changing the size of content or the layout of scripts.

The ***scriptlevel*** attribute, if present, must have value `+<U>`, `-<U>` or `<U>` where `<U>` is an unsigned-integer. In that case the user agent is expected to treat the `scriptlevel` attribute as a presentational hint setting the element's `math-depth` property to the corresponding value. More precisely, `+<U>`, `-<U>` and `<U>` are respectively mapped to `add(<U>)` `add(-<U>)` and `<U>`.

`displaystyle` and `scriptlevel` values are automatically adjusted within MathML elements. To fully implement these attributes, additional CSS properties must be specified in the user agent stylesheet as described in A. User Agent Stylesheet. In particular, for all MathML elements a default `font-size: math` is specified to ensure that `scriptlevel` changes are taken into account.

In this example, an `munder` element is used to attach a script "A" to a base "∑". By default, the summation symbol is rendered with the font-size inherited from its parent and the A as a scaled down subscript. If displaystyle is true, the summation symbol is drawn bigger and the "A" becomes an underscript. If scriptlevel is reset to 0 on the "A", then it will use the same font-size as the top-level `math` root.

```
<math>
  <munder>
    <mo>∑</mo>
    <mi>A</mi>
  </munder>
  <munder displaystyle="true">
    <mo>∑</mo>
    <mi>A</mi>
  </munder>
  <munder>
    <mo>∑</mo>
    <mi scriptlevel="0">A</mi>
  </munder>
</math>
```

$$\sum_{A} \sum_{A} \sum_{A}$$

NOTE

TEX's `\displaystyle`, `\textstyle`, `\scriptstyle`, and `\scriptscriptstyle` correspond to `displaystyle` and `scriptlevel` as `true` and `0`, `false` and `0`, `false` and `1`, and `false` and 2, respectively.

§ 2.1.7 Attributes Reserved as Valid

The attributes ***intent*** and ***arg*** are reserved as valid attributes.

This specification does not define any observable behavior that is specific to the `intent` and `arg` attributes.

> NOTE
> These attributes are described in [*MATHML4*] and future versions of this specification may or may not define them. Authors should be aware that they are currently in development and subject to change.

2.2 Integration in the Web Platform

2.2.1 HTML and SVG

MathML can be mixed with HTML and SVG as described in the relevant specifications [*HTML*] [*SVG*].

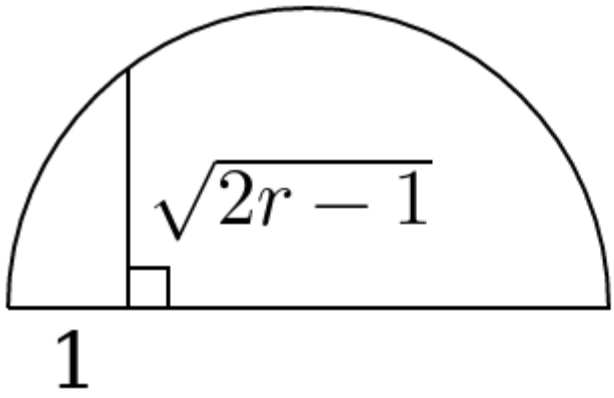
When evaluating the SVG `requiredExtensions` attribute, user agents must claim support for the language extension identified by the MathML namespace.

In this example, inline MathML and SVG elements are used inside an HTML document. SVG elements `<switch>` and `<foreignObject>` (with proper `<requiredExtensions>`) are used to embed a MathML formula with a text fallback, inside a diagram. HTML `input` element is used within the `mtext` to include an interactive input field inside a mathematical formula. See also 3.7 Semantics and Presentation for an example of SVG and HTML inside an `annotation-xml` element.

```
<svg style="font-size: 20px" width="400px" height="220px" viewBox="0 0 20
  <g transform="translate(10,80)">
    <path d="M 0 0 L 150 0 A 75 75 0 0 0 0 0
             M 30 0 L 30 -60 M 30 -10 L 40 -10 L 40 0"
          fill="none" stroke="black"></path>
    <text transform="translate(10,20)">1</text>
    <switch transform="translate(35,-40)">
      <foreignObject width="200" height="50"
                     requiredExtensions="http://www.w3.org/1998/Math/MatI
        <math>
          <msqrt>
            <mn>2</mn>
            <mi>r</mi>
            <mo>-</mo>
            <mn>1</mn>
          </msqrt>
        </math>
      </foreignObject>
      <text>\sqrt{2r - 1}</text>
    </switch>
  </g>
</svg>

<p>
  Fill the blank:
  <math>
    <msqrt>
      <mn>2</mn>
      <mtext><input onchange="..." size="2" type="text"></mtext>
      <mo>-</mo>
      <mn>1</mn>
    </msqrt>
    <mo>=</mo>
    <mn>3</mn>
```

```
  </math>
</p>
```



Fill the blank: $\sqrt{2\square - 1} = 3$

2.2.2 CSS styling

User agents must support various CSS features mentioned in this specification, including new ones described in 4. CSS Extensions for Math Layout. They must follow the computation rule for display: contents.

In this example, the MathML formula inherits the CSS color of its parent and uses the `font-family` specified via the style attribute.

```
<div style="width: 15em; color: blue">
  This mathematical formula with a big summation and the number pi
  <math display="block" style="font-family: STIX Two Math">
    <mrow>
      <munderover>
        <mo>∑</mo>
        <mrow><mi>n</mi><mo>=</mo><mn>1</mn></mrow>
        <mrow><mo>+</mo><mn>∞</mn></mrow>
      </munderover>
      <mfrac>
        <mn>1</mn>
        <msup><mi>n</mi><mn>2</mn></msup>
      </mfrac>
    </mrow>
    <mo>=</mo>
    <mfrac>
      <msup><mi>π</mi><mn>2</mn></msup>
      <mn>6</mn>
    </mfrac>
  </math>
  is easy to prove.
</div>
```

This mathematical formula with a big summation and the number pi

$$\sum_{n=1}^{+\infty} \frac{1}{n^2} = \frac{\pi^2}{6}$$

is easy to prove.

All documents containing MathML Core elements must include CSS rules described in A. User Agent Stylesheet as part of user-agent level style sheet defaults. In particular, this adds `!important` rules to force writing mode to `horizontal-lr` on all MathML elements.

The float property does not create floating of elements whose parent's computed `display` value is

`block math` or `inline math`, and does not take them out-of-flow.

The ::first-line and ::first-letter pseudo-elements do not apply to elements whose computed `display` value is `block math` or `inline math`, and such elements do not contribute a first formatted line or first letter to their ancestors.

The following CSS features are not supported and must be ignored:

- Line breaking inside math formulas: `white-space` is treated as `nowrap` on all MathML elements.
- Alignment properties: `align-content`, `justify-content`, `align-self`, `justify-self` have no effects on MathML elements.

> NOTE
> These features might be handled in future versions of this document. For now, authors are discouraged from setting a different value for these properties as that might lead to backward incompatibility issues.

§ 2.2.3 DOM and JavaScript

User agents supporting Web application APIs must ensure that they keep the visual rendering of MathML synchronized with the [*DOM*] tree, in particular perform necessary updates when MathML attributes are modified dynamically.

All the nodes representing MathML elements in the DOM must implement, and expose to scripts, the following ***`MathMLElement`*** interface.

WebIDL

```
[Exposed=Window]
interface MathMLElement : Element { };
MathMLElement includes GlobalEventHandlers;
MathMLElement includes HTMLOrForeignElement;
```

The `GlobalEventHandlers` and ***`HTMLOrForeignElement`*** interfaces are defined in [*HTML*].

In the following example, a MathML formula is used to render the fraction "α over 2". When clicking the red α, it is changed into a blue β.

```
<script>
  function ModifyMath(mi) {
      mi.style.color = 'blue';
      mi.textContent = 'β';
  }
</script>
<math>
  <mrow>
    <mfrac>
      <mi style="color: red" onclick="ModifyMath(this)">α</mi>
      <mn>2</mn>
    </mfrac>
  </mrow>
</math>
```

ISSUE
Rename HTMLOrSVGElement and define MathMLElement in [*HTML*].

2.2.4 Text layout

Because math fonts generally contain very tall glyphs such as big integrals, using typographic metrics is important to avoid excessive line spacing of text. As a consequence, user agents must take into account the USE_TYPO_METRICS flag from the OS/2 table [*OPEN-FONT-FORMAT*] when performing text layout.

2.2.5 Focus

MathML provides the ability for authors to allow for interactivity in supporting interactive user agents

using the same concepts, approach and guidance to `Focus` as described in HTML, with modifications or clarifications regarding application for MathML as described in this section.

When an element is focused, all applicable CSS focus-related pseudo-classes as defined in Selectors Level 3 apply, as defined in that specification.

The contents of embedded `math` elements (including HTML elements inside token elements) contribute to the sequential focus order of the containing owner HTML document (combined sequential focus order).

3. Presentation Markup

3.1 Visual formatting model

3.1.1 Box Model

The default `display` property is described in A. User Agent Stylesheet:

- For the `<math>` root, it is equal to `inline math` or `block math` according to the value of the `display` attribute.
- For Tabular MathML elements `mtable`, `mtr`, `mtd` it is respectively equal to `inline-table`, `table-row` and `table-cell`.
- For all but the first children of the `maction` and `semantics` elements, it is equal to `none`.
- For all the other MathML elements it is equal to `block math`.

In order to specify math layout in different writing modes, this specification uses concepts from [*CSS-WRITING-MODES-4*]:

- abstract dimensions: block dimension, inline dimension, block axis, inline axis, block size, inline size.
- flow-relative directions: inline-start, inline-end, block-start.
- line-relative directions: line-left, line-over, line-under.

> NOTE
> Unless specified otherwise, the figures in this specification use a writing mode of `horizontal-lr` and `ltr`. See Figure 4, Figure 5 and Figure 6 for examples of other writing modes that are sometimes used for math layout.

Boxes used for MathML elements rely on several parameters in order to perform layout in a way that is compatible with CSS but also to take into account very accurate positions and spacing within math formulas:

1. Inline metrics. min-content inline size, max-content inline size and inline size from CSS. See Figure 1.

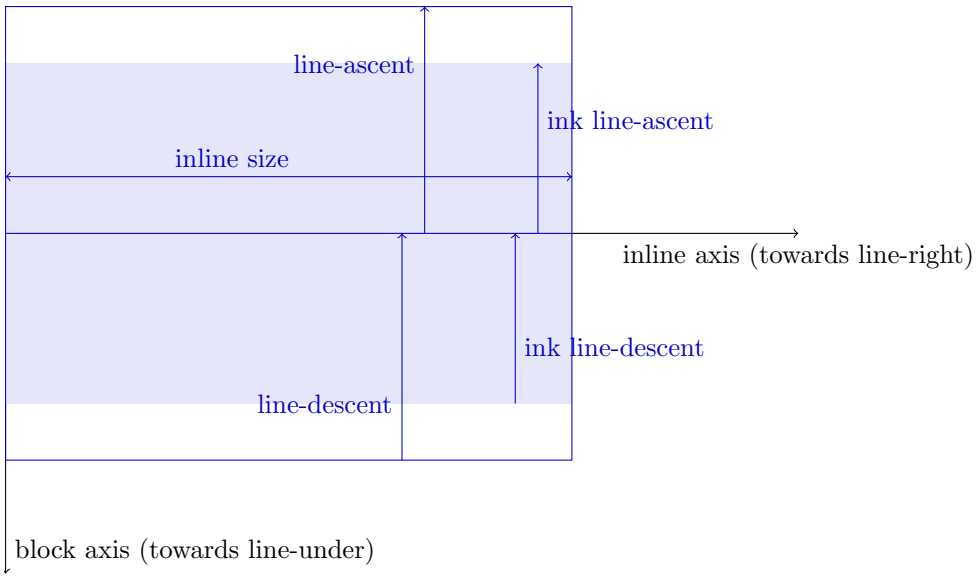


Figure 1 Generic Box Model for MathML elements

2. Block metrics. The block size, first baseline set and last baseline set. The following baselines are defined for MathML boxes:

    1. The ***alphabetic baseline*** which typically aligns with the bottom of uppercase Latin glyphs. The algebraic distance from the alphabetic baseline to the line-over edge of the box is called the ***line-ascent***. The algebraic distance from the line-under edge to the alphabetic baseline of the box is called the ***line-descent***.
    2. The ***mathematical baseline***, also called ***math axis***, which typically aligns with the fraction bar, middle of fences and binary operators. It is shifted away from the alphabetic baseline by AxisHeight towards the line-over.
    3. The ***ink-over baseline***, indicating the line-over theorical limit of the math content drawn,

excluding any extra space. If not specified, it is aligned with the line-over edge. The algebraic distance from the alphabetic baseline to the ink-over baseline is called the ***ink line-ascent***.

4. The ***ink-under baseline***, indicating the line-under theorical limit of the math content drawn, excluding any extra space. If not specified, it is aligned with the line-under edge. The algebraic distance from the ink-under baseline to the alphabetic baseline is called the ***ink line-descent***.

> NOTE
> For math layout, it is very important to rely on the ink extent when positioning text. This is not the case for more complex notations (e.g. square root). Although ink-ascent and ink-descent are defined for all MathML elements they are really only used for the token elements. In other cases, they just match normal ascent and descent.

Unless specified otherwise, the last baseline set is equal to the first baseline set for MathML boxes.

3. An optional ***italic correction*** which provides a measure of how much the text of a box is slanted along the inline axis. See Figure 2.

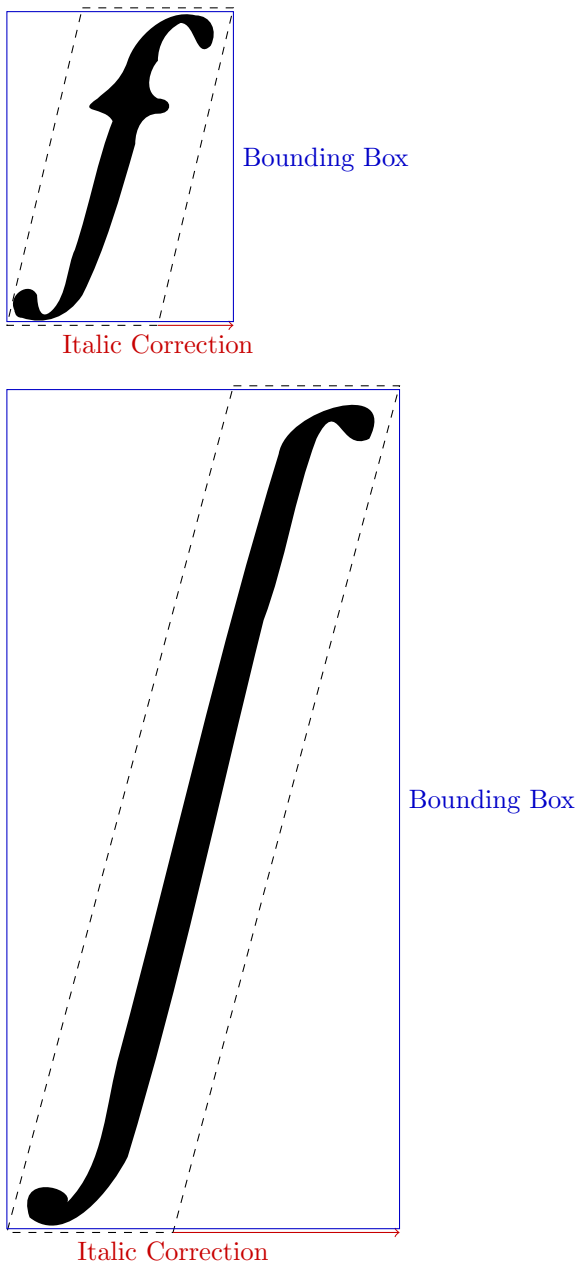


Figure 2 Examples of italic correction for italic f and large integral

If it is requested during calculation of min-content inline size and max-content inline size or during layout then 0 is used as a fallback value.

4. An optional ***top accent attachment*** which provides a reference offset on the inline axis of a box that should be used when positioning that box as an accent. See Figure 3.

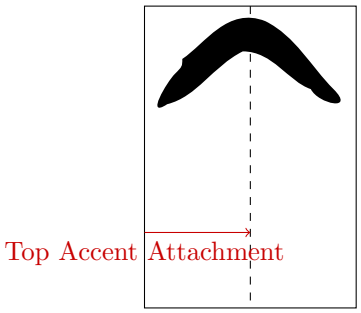


Figure 3 Example of top accent attachment for a circumflex accent

If it is requested during calculation of min-content inline size (respectively max-content inline size) then half the min-content inline size (respectively max-content inline size) is used as a fallback value. If it is requested during layout then half the inline size of the box is used as a fallback value.

Given a MathML box, the following offsets are defined:

- The ***inline offset*** of a child box is the offset between the inline-start edge of the parent box and the inline-start edge of the child box.
- The ***block offset*** of a child box is the offset between the block-start edge of the parent box and the block-start edge of the child box.
- The ***line-left offset*** of a child box is the offset between the line-left edge of the parent box and the line-left edge of the child box.

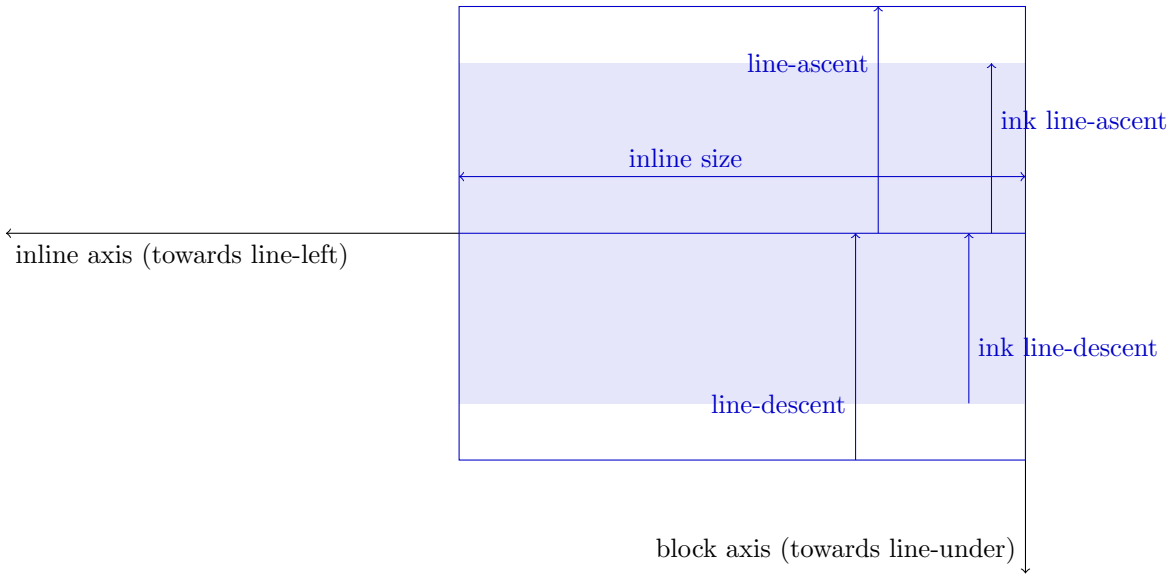


Figure 4 Box model for writing mode `horizontal-tb` and `rtl` that may be used in e.g. Arabic math.

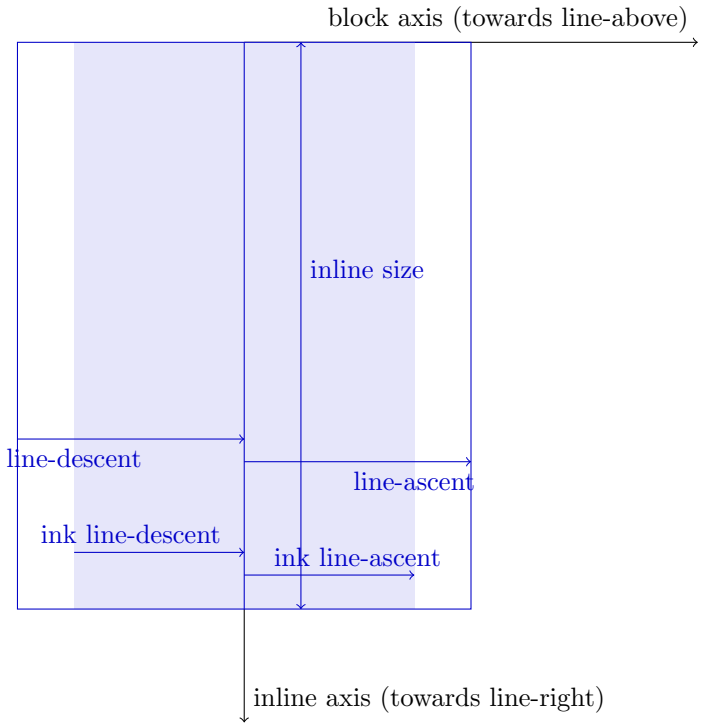


Figure 5 Box model for writing mode `vertical-lr` and `ltr` that may be used in e.g. Mongolian math.

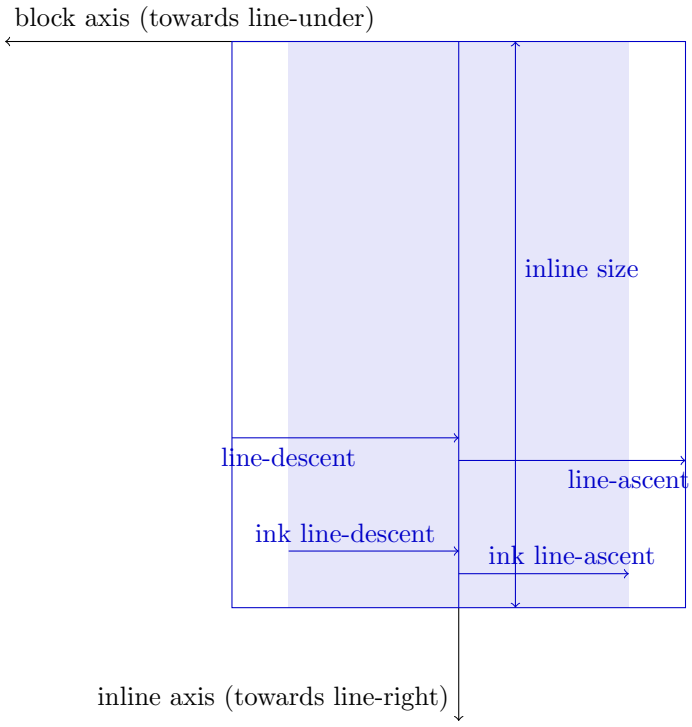


Figure 6 Box model for writing mode `vertical-rl` and `ltr` that may be used in e.g. Japanese math.

NOTE
The position of child boxes and graphical items inside a MathML box are expressed using the inline offset and block offset. For convenience, the layout algorithms may describe offsets using flow-relative directions, line-relative directions or the alphabetic baseline. It is always possible to pass from one description to the other because position of child boxes is always performed after the metrics of the box and of its child boxes are calculated.

Here are examples of offsets obtained from line-relative metrics:

- The inline offset of a child box is the line-left offset if the CSS property direction is `ltr` and is the inline size of the box − (line-left offset + inline size of the child box) otherwise.
- The block offset of the alphabetic baseline of a box is the line-ascent if the CSS property writing-mode is `horizontal-lr`, `vertical-rl` or `sideways-rl` and is the line-descent otherwise.

ISSUE 78: Ink ascent/descent **opentype/tex** **needs-tests**
Improve definition of ink ascent/descent?

3.1.2 Layout Algorithms

Each MathML element has an associated ***math content box***, which is calculated as described in this chapter's layout algorithms using the following structure:

1. Calculation of min-content inline size and max-content inline size of the math content.
2. Box layout:
    1. Layout of in-flow child boxes.
    2. Calculation of inline size, ink line-ascent, ink line-descent, line-ascent and line-ascent of the math content.
    3. Calculation of offsets of child boxes within the math content box as well as sizes and offsets of extra graphical items (bars, radical symbol, etc).
    4. Layout and positioning of absolutely-positioned and fixed-positioned boxes, as described in [*CSS-POSITION-3*].

The following extra steps must be performed:

- After calculating the line-ascent and line-descent of a math content box, its block size is set to their sum.
- By default, the box metrics, offsets, baselines, italic correction and top accent attachment of the content box are set to the one calculated for the math content box. However if a different size is specified for the content box (via width, height or related properties), that size is used instead and the inline-start and block-start edges of the math content box are aligned with the ones of the content box, unless otherwise specified in a specific layout algorithm.
- The box metrics and offsets of the padding box are obtained from the content box by taking into account the corresponding padding properties as described in CSS.

  The baselines of the padding box are the same as the one of the content box.

  If the content box has a top accent attachment then the padding box has the same property, increased by the inline-start padding. If the content box has an italic correction then the padding box has the same property, increased by the inline-end padding.

- The box metrics and offsets of the border box are obtained from the padding box by taking into account the corresponding border-width property as described in CSS.

  In general, the baselines of the border box are the same as the one of the padding box. However, if the line-over border is positive then the ink-over baseline is set to the line-over edge of the border box and if the line-under border is positive then the ink-under baseline is set to the line-under edge of the border box.

  If the padding box has a top accent attachment then the border box has the same property, increased by the border-width of its inline-start egde. If the padding box has an italic correction then the border box has the same property, increased by the border-width of its inline-end egde.

- The box metrics and offsets of the margin box are obtained from the border box by taking into account the corresponding margin properties as described in CSS.

  The baselines of the margin box are the same as the one of the border box.

  If the padding box has a top accent attachment then the margin box has the same property, increased by the inline-start margin. If the padding box has an italic correction then the margin box has the same property, increased by the inline-end margin.

> NOTE
> Per the description above, margin-collapsing does not apply to MathML elements.

During box layout, optional ***inline stretch size constraint*** and ***block stretch size constraint*** parameters may be used on embellished operators. The former indicates a target size that a core operator stretched along the inline axis should cover. The latter indicates an ink line-ascent and ink line-descent that a core operator stretched along the block axis should cover. Unless specified otherwise, these parameters are ignored during box layout and child boxes are laid out without any stretch size constraint.

> ISSUE 76: Define what inline percentages resolve against. **css/html5** **need specification update** **needs-tests**
> Define what inline percentages resolve against

> ISSUE 77: Define what block percentages resolve against. **css/html5** **need specification update** **needs-tests**
> Define what block percentages resolve against

3.1.3 Anonymous <mrow> boxes

An ***anonymous box*** is a box without any associated element in the DOM tree and which is generated for layout purpose only. The properties of anonymous boxes are inherited from the enclosing non-anonymous box while non-inherited properties have their initial value. An ***anonymous <mrow> box*** is an anonymous box with `display` equal to `block math` and which is laid out as described in section 3.3.1.2 Layout of `<mrow>`.

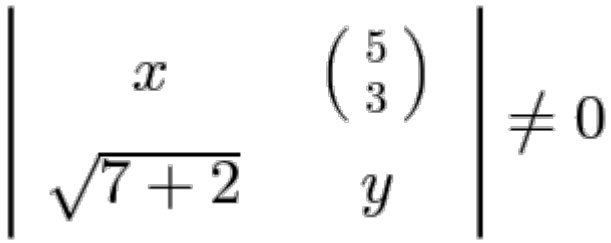
If a MathML element ***generates an anonymous <mrow> box*** then it wraps its children in an anonymous <mrow> box. I.e., its subtree in the visual formatting model is made of an anonymous <mrow> box which itself contains the boxes associated to the children of this MathML element.

In the following example, the `math` and `mrow` elements are laid out as described in section 3.3.1.2 Layout of <mrow>. In particular, the `<math>` element adds proper spacing around its `<mo>≠</mo>` child and the `<mrow>` element stretches its `<mo>|</mo>` children vertically.

The `mtd` element has `display: table-cell` and the `msqrt` element displays a radical symbol around its children. However, they also place their children in a way that is similar to what is described in section 3.3.1.2 Layout of <mrow>: the `<msqrt>` element adds proper spacing around its `<mo>+</mo>` child while the `<mtd>` element stretches its `<mo>` children vertically. In order to make this possible, each of these two elements generates an anonymous <mrow> box.

```
<math>
  <mrow>
    <mo>|</mo>
    <mtable>
      <mtr>
        <mtd>
          <mi>x</mi>
        </mtd>
        <mtd>
          <mo>(</mo>
          <mfrac linethickness="0">
            <mn>5</mn>
            <mn>3</mn>
          </mfrac>
          <mo>)</mo>
        </mtd>
      </mtr>
      <mtr>
        <mtd>
          <msqrt>
            <mn>7</mn>
            <mo>+</mo>
            <mn>2</mn>
          </msqrt>
        </mtd>
        <mtd>
          <mi>y</mi>
        </mtd>
      </mtr>
    </mtable>
    <mo>|</mo>
```

```
  </mrow>
  <mo>≠</mo>
  <mn>0</mn>
</math>
```



3.1.4 Stacking contexts

MathML elements can overlap due to various spacing rules. They can as well contain extra graphical items (bars, radical symbol, etc). A MathML element with computed style `display: block math` or `display: inline math` generates a new stacking context. The painting order of in-flow children of such a MathML element is exactly the same as block elements. The extra graphical items are painted after text and background (right after step 7.2.4 for `display: inline math` and right after step 7.2 for `display: block math`).

3.2 Token Elements

Token elements in presentation markup are broadly intended to represent the smallest units of mathematical notation which carry meaning. Tokens are roughly analogous to words in text. However, because of the precise, symbolic nature of mathematical notation, the various categories and properties of token elements figure prominently in MathML markup. By contrast, in textual data, individual words rarely need to be marked up or styled specially.

NOTE
In practice, most MathML token elements just contain simple text for variables, numbers, operators etc and don't need sophisticated layout. However, it can contain text with line breaks or arbitrary HTML5 phrasing elements.

3.2.1 Text <mtext>

The ***mtext*** element is used to represent arbitrary text that should be rendered as itself. In general, the <mtext> element is intended to denote commentary text.

The <mtext> element accepts the attributes described in 2.1.3 Global Attributes.

In the following example, mtext is used to put conditional words in a definition:

```
<math>
  <mi>y</mi>
  <mo>=</mo>
  <mrow>
    <msup>
      <mi>x</mi>
      <mn>2</mn>
    </msup>
    <mtext> if </mtext>
    <mrow>
      <mi>x</mi>
      <mo>≥</mo>
      <mn>1</mn>
    </mrow>
    <mtext> and </mtext>
    <mn>2</mn>
    <mtext> otherwise.</mtext>
  </mrow>
</math>
```

$$y = x^2 \text{ if } x \geq 1 \text{ and } 2 \text{ otherwise.}$$

3.2.1.1 Layout of <mtext>

If the element does not have its computed display property equal to `block math` or `inline math` then it is laid out according to the CSS specification where the corresponding value is described. Otherwise, the layout below is performed.

If the <mtext> element contains only text content without forced line break or soft wrap opportunity then, the anonymous child node generated for that text is laid out as defined in the relevant CSS specification and:

- The min-content inline size, max-content inline size and inline size of the math content box are equal to the one of the anonymous child.
- The ink line-ascent and ink line-descent of the math content box are set so that its ink-over baseline, ink-under baseline and alphabetic baseline match the one of the ink bounding box of the text.
- The line-ascent (respectively line-descent) of the math content box is set to the ink line-ascent (respectively ink line-descent).
- If the text content is made of a single glyph and this glyph has an entry in the MathItalicsCorrectionInfo table then the specified value is used as the italic correction.
- If the text content is made of a single glyph and this glyph has an entry in the MathTopAccentAttachment table then the specified value is used as the top accent attachment of the <mtext> element.

Otherwise, the mtext element is laid out as a block box and corresponding min-content inline size, max-content inline size, inline size, block size, first baseline set and last baseline set are used for the math content box.

§ 3.2.2 Identifier <mi>

The ***mi*** element represents a symbolic name or arbitrary text that should be rendered as an identifier. Identifiers can include variables, function names, and symbolic constants.

The <mi> element accepts the attributes described in 2.1.3 Global Attributes as well as the following attribute:

- mathvariant

The layout algorithm is the same as the mtext element. The user agent stylesheet must contain the following property in order to implement automatic italic via the text-transform value introduced in 4.2 The math-auto transform:

```
mi {
  text-transform: math-auto;
```

```
}
```

The ***mathvariant*** attribute, if present, must be an ASCII case-insensitive match of `normal`. In that case, the user agent is expected to treat the attribute as a presentational hint setting the element's text-transform property to `none`. Otherwise it has no effects.

NOTE

In [*MathML3*], the `mathvariant` attribute was used to define logical classes of token elements, each class providing a collection of typographically-related symbolic tokens with specific meaning within a given mathematical expression.

In MathML Core, this attribute is only used to cancel automatic italic of the `mi` element. For other use cases, the proper Mathematical Alphanumeric Symbols [*UNICODE*] should be used instead. See also section C. Mathematical Alphanumeric Symbols.

In the following example, `mi` is used to render variables and function names. Note that per 4.2 The `math-auto` transform the default style `text-transform: math-auto` has no effect on the first `<mi>` ("cos" is made of three characters), makes the second `<mi>` render as math italic ("c" is made of a single character U+0063 Latin Small Letter C which is mapped to U+1D450 Mathematical Italic Small C per the italic table), has no effect on the third `<mi>` (overridden by `mathvariant="normal"`, setting `text-transform` to none) or on the fourth `<mi>` (no mapping defined for U+221E Infinity in the italic table).

```
<math>
  <mi>cos</mi>
  <mo>,</mo>
  <mi>c</mi>
  <mo>,</mo>
  <mi mathvariant="normal">c</mi>
  <mo>,</mo>
  <mi>∞</mi>
</math>
```

$\mathrm{cos}, c, \mathrm{c}, \infty$

3.2.3 Number <mn>

The ***mn*** element represents a "numeric literal" or other data that should be rendered as a numeric literal. Generally speaking, a numeric literal is a sequence of digits, perhaps including a decimal point, representing an unsigned integer or real number.

The <mn> element accepts the attributes described in 2.1.3 Global Attributes. Its layout algorithm is the same as the mtext element.

> In the following example, mn is used to write a decimal number.
>
> ```
> <math>
>   <mn>3.141592653589793</mn>
> </math>
> ```
>
> 3.141592653589793

3.2.4 Operator, Fence, Separator or Accent <mo>

The ***mo*** element represents an operator or anything that should be rendered as an operator. In general, the notational conventions for mathematical operators are quite complicated, and therefore MathML provides a relatively sophisticated mechanism for specifying the rendering behavior of an <mo> element.

As a consequence, in MathML the list of things that should "render as an operator" includes a number of notations that are not mathematical operators in the ordinary sense. Besides ordinary operators with infix, prefix, or postfix forms, these include fence characters such as braces, parentheses, and "absolute value" bars; separators such as comma and semicolon; and mathematical accents such as a bar or tilde over a symbol. This chapter uses the term "operator" to refer to operators in this broad sense.

The <mo> element accepts the attributes described in 2.1.3 Global Attributes as well as the following attributes:

- form
- fence
- separator

- `lspace`
- `rspace`
- `stretchy`
- `symmetric`
- `maxsize`
- `minsize`
- `largeop`
- `movablelimits`

This specification does not define any observable behavior that is specific to the ***fence*** and ***separator*** attributes.

> NOTE
> Authors may use the `fence` and `separator` to describe specific semantics of operators. The default values may be determined from the `Operators_fence` and `Operators_separator` tables, or equivalently the human-readable version of the operator dictionary.

In the following example, the mo element is used for the binary operator +. Default spacing is symmetric around that operator. A tighter spacing is used if you rely on the `form` attribute to force it to be treated as a prefix operator. Spacing can also be specified explicitly using the `lspace` and `rspace` attributes.

```
<math>
  <mn>1</mn>
  <mo>+</mo>
  <mn>2</mn>
  <mo form="prefix">+</mo>
  <mn>3</mn>
  <mo lspace="2em">+</mo>
  <mn>4</mn>
  <mo rspace="3em">+</mo>
  <mn>5</mn>
</math>
```

Another use case is for big operators such as summation. When displaystyle is true, such an operator is drawn larger but one can change that with the `largeop` attribute. When displaystyle is false, underscripts are actually rendered as subscripts but one can change that with the `movablelimits` attribute.

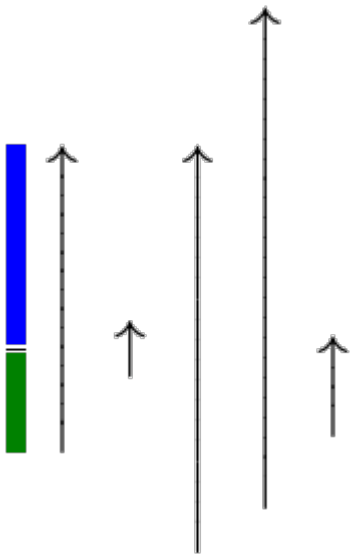
```
<math>
  <mrow displaystyle="true">
  <munder>
    <mo>∑</mo>
    <mn>5</mn>
  </munder>
  <munder>
    <mo largeop="false">∑</mo>
    <mn>6</mn>
  </munder>
  </mrow>
  <mrow>
    <munder>
      <mo>∑</mo>
      <mn>5</mn>
    </munder>
```

```
    <munder>
      <mo movablelimits="false">∑</mo>
      <mn>7</mn>
    </munder>
  </mrow>
</math>
```

$$\sum_{5} \sum_{6} \sum_{5} \sum_{7}$$

Operators are also used for stretchy symbols such as fences, accents, arrows etc. In the following example, the vertical arrow stretches to the height of the `mspace` element. One can override default stretch behavior with the `stretchy` attribute e.g. to force an unstretched arrow. The `symmetric` attribute allows to indicate whether the operator should stretch symmetrically above and below the math axis (fraction bar). Finally the `minsize` and `maxsize` attributes add additional constraints over the stretch size.

```
<math>
  <mfrac>
    <mspace height="50px" depth="50px" width="10px" style="background: b
    <mspace height="25px" depth="25px" width="10px" style="background: g
  </mfrac>
  <mo>↑</mo>
  <mo stretchy="false">↑</mo>
  <mo symmetric="true">↑</mo>
  <mo minsize="250px">↑</mo>
  <mo maxsize="50px">↑</mo>
</math>
```



Note that the default properties of operators are dictionary-based, as explained in 3.2.4.2 Dictionary-based attributes. For example a binary operator typically has default symmetric spacing around it while a fence is generally stretchy by default.

§ *3.2.4.1 Embellished operators*

A MathML Core element is an ***embellished operator*** if it is:

1. An `mo` element;
2. a scripted element or an `mfrac`, whose first in-flow child exists and is an embellished operator;
3. a grouping element or `mpadded`, whose in-flow children consist (in any order) of one embellished operator and zero or more space-like elements.

The ***core operator*** of an embellished operator is the `<mo>` element defined recursively as follows:

1. The core operator of an `mo` element; is the element itself.
2. The core operator of an embellished scripted element or `mfrac` element is the core operator of its first in-flow child.
3. The core operator of an embellished grouping element or `mpadded` is the core operator of its unique embellished operator in-flow child.

The ***stretch axis*** of an embellished operator is *inline* if its core operator contains only text content made of a single character `c`, and that character has inline intrinsic stretch axis. Otherwise, the stretch

axis of the embellished operator is *block*.

The same definitions apply for boxes in the visual formatting model where an anonymous <mrow> box is treated as a grouping element.

§ *3.2.4.2 Dictionary-based attributes*

The `form` property of an embellished operator is either `infix`, `prefix` or `postfix`. The corresponding ***form*** attribute on the `mo` element, if present, must be an ASCII case-insensitive match to one of these values.

The ***algorithm for determining the `form` of an embellished operator*** is as follows:

1. If the `form` attribute is present and valid on the core operator, then its ASCII lowercased value is used.
2. If the embellished operator is the first in-flow child of a grouping element, `mpadded` or `msqrt` with more than one in-flow child (ignoring all space-like children) then it has form `prefix`.
3. Or, if the embellished operator is the last in-flow child of a grouping element, `mpadded` or `msqrt` with more than one in-flow child (ignoring all space-like children) then it has form `postfix`.
4. Or, if the embellished operator is an in-flow child of a scripted element, other than the first in-flow child, then it has form `postfix`.
5. Otherwise, the embellished operator has form `infix`.

The ***`stretchy`***, ***`symmetric`***, ***`largeop`***, ***`movablelimits`*** properties of an embellished operator are either `false` or `true`. In the latter case, it is said that the embellished operator *has* the property. The corresponding ***stretchy***, ***symmetric***, ***largeop***, ***movablelimits*** attributes on the `mo` element, if present, must be a boolean.

The ***`lspace`***, ***`rspace`***, ***`minsize`*** properties of an embellished operator are <length-percentage>. The ***`maxsize`*** property of an embellished operator is either a <length-percentage> or ∞. The ***lspace***, ***rspace***, ***minsize*** and ***maxsize*** attributes on the `mo` element, if present, must be a <length-percentage>.

The ***algorithm for determining the properties of an embellished operator*** is as follows:

1. If the corresponding `stretchy`, `symmetric`, `largeop`, `movablelimits`, `lspace`, `rspace`, `maxsize` or `minsize` attribute is present and valid on the core operator, then the ASCII lowercased value of this property is used.

2. Otherwise, run the algorithm for determining the `form` of an embellished operator.
3. If the core operator contains only text content `Content`, then set `Category` to the result of the algorithm to determine the category of an operator `(Content, Form)` where `Form` is the `form` calculated at the previous step.
4. If `Category` is `Default` and the `form` of embellished operator was not explicitly specified as an attribute on its core operator:

    1. Set `Category` to the result of the algorithm to determine the category of an operator `(Content, Form)` where `Form` is `infix`.
    2. If `Category` is `Default`, then run the algorithm again with `Form` set to `postfix`.
    3. If `Category` is `Default`, then run the algorithm again with `Form` set to `prefix`.

5. Run the algorithm to set the properties of an operator from its category `Category`.

When used during layout, the values of `stretchy`, `symmetric`, `largeop`, `movablelimits`, `lspace`, `rspace`, `minsize` are obtained by the algorithm for determining the properties of an embellished operator with the following extra resolutions:

- Percentage values for `lspace`, `rspace` are interpreted relative to the value read from the dictionary or to the fallback value above.
- Interpretation of percentage values for `minsize` and `maxsize` are described in 3.2.4.3 Layout of operators.
- Font-relative lengths for `lspace`, `rspace`, `minsize` and `maxsize` rely on the font style of the core operator, not the one of the embellished operator.

§ *3.2.4.3 Layout of operators*

If the <mo> element does not have its computed `display` property equal to `block math` or `inline math` then it is laid out according to the CSS specification where the corresponding value is described. Otherwise, the layout below is performed.

The text of the operator must only be painted if the visibility of the <mo> element is `visible`. In that case, it must be painted with the color of the <mo> element.

Operators are laid out as follows:

1. If the content of the <mo> element is not made of a single character `c` then fall back to the layout

algorithm of 3.2.1.1 Layout of <mtext>.

2. If the operator has the stretchy property:
   - If the stretch axis of the operator is inline:
     1. If it is not possible to shape a stretchy glyph corresponding to c in the inline direction with the first available font then fall back to the layout algorithm of 3.2.1.1 Layout of <mtext>.
     2. The min-content inline size and max-content inline size of the math content are set to the one obtained by the layout algorithm of 3.2.1.1 Layout of <mtext>.
     3. If there is not any inline stretch size constraint T_{inline} then fall back to the layout algorithm of 3.2.1.1 Layout of <mtext>.
     4. The inline size and (ink) block metrics of the math content are given by algorithm to shape a stretchy glyph to inline dimension T_{inline}.
     5. The painting of the operator is performed by the algorithm to shape a stretchy glyph stretched to inline dimension T_{inline} and at position determined by the previous box metrics.
   - Otherwise, the stretch axis of the operator is block. The following steps are performed:
     1. If it is not possible to shape a stretchy glyph corresponding to c in the block direction with the first available font then fall back to the layout algorithm of 3.2.1.1 Layout of <mtext>.
     2. The min-content inline size and max-content inline size of the math content are set to the preferred inline size of a glyph stretched along the block axis.
     3. If there is not any block stretch size constraint $(U_{ascent}, U_{descent})$ then fall back to the layout algorithm of 3.2.1.1 Layout of <mtext>.
     4. If the operator has the symmetric property then set the target sizes T_{ascent} and $T_{descent}$ to S_{ascent} and $S_{descent}$ respectively:
        - $S_{ascent} = \max(U_{ascent} - \text{AxisHeight}, U_{descent} + \text{AxisHeight}) + \text{AxisHeight}$
        - $S_{descent} = \max(U_{ascent} - \text{AxisHeight}, U_{descent} + \text{AxisHeight}) - \text{AxisHeight}$

        Otherwise set them to U_{ascent} and $U_{descent}$ respectively.

> NOTE
> The property T_{ascent} − AxisHeight = $T_{descent}$ + AxisHeight means that an operator stretching exactly T_{ascent} above the baseline and $T_{descent}$ below the baseline would actually stretch symmetrically above and below the math axis. S_{ascent} and $S_{descent}$ are the minimal values, that are respectively not less than U_{ascent} and $U_{descent}$, which satisfy this property.

5. Let `minsize` and `maxsize` be the `minsize` and `maxsize` properties on the operator. Percentage values are interpreted relative to the height of the glyph for `c`. Let $T = T_{ascent} + T_{descent}$ be the target size. If `minsize` < 0 then set `minsize` to 0. If `maxsize` < `minsize` then set `maxsize` to `minsize`. With 0 ≤ `minsize` ≤ `maxsize`:
   - If $T \leq 0$ then set T_{ascent} to `minsize` / 2 + AxisHeight and then set $T_{descent}$ to `minsize` − T_{ascent}.
   - Otherwise, if $0 < T <$ `minsize` then set T_{ascent} to max(0, (T_{ascent} − AxisHeight) × `minsize` / T + AxisHeight) and $T_{descent}$ to `minsize` − T_{ascent}.
   - Otherwise, if `maxsize` < T then set T_{ascent} to max(0, (T_{ascent} − AxisHeight) × `maxsize` / T + AxisHeight) and $T_{descent}$ to `maxsize` − T_{ascent}.

> NOTE
> The default `maxsize` is value ∞ is interpreted above as being larger than any other size, i.e. `minsize` ≤ `maxsize` is always true while `maxsize` < `minsize` and `maxsize` < T are always false.

> NOTE
> This step ensures that the condition `minsize` ≤ T ≤ `maxsize` holds. Additionnally, if the target values correspond to symmetric stretching with respect to the math axis then property T_{ascent} − AxisHeight = $T_{descent}$ + AxisHeight is preserved.

6. The inline size, ink line-ascent, ink line-descent, line-ascent and line-descent of the math content are obtained by the algorithm to shape a stretchy glyph to block dimension $T_{ascent} + T_{descent}$. The inline size of the math content is the width of the stretchy glyph. The stretchy glyph is shifted towards the line-under by a value Δ so that its center aligns with the center of the target: the ink ascent of the math content is the ascent of the stretchy glyph − Δ and the ink descent of the math content is the descent of the stretchy glyph + Δ. These centers have coordinates "½(ascent − descent)" so Δ = [(ascent of stretchy glyph − descent of stretchy glyph) − (T_{ascent} −

$T_{descent}$)] / 2.

7. The painting of the operator is performed by the algorithm to shape a stretchy glyph stretched to block dimension T_{ascent} + $T_{descent}$ and at position determined by the previous box metrics shifted by Δ towards the line-over.

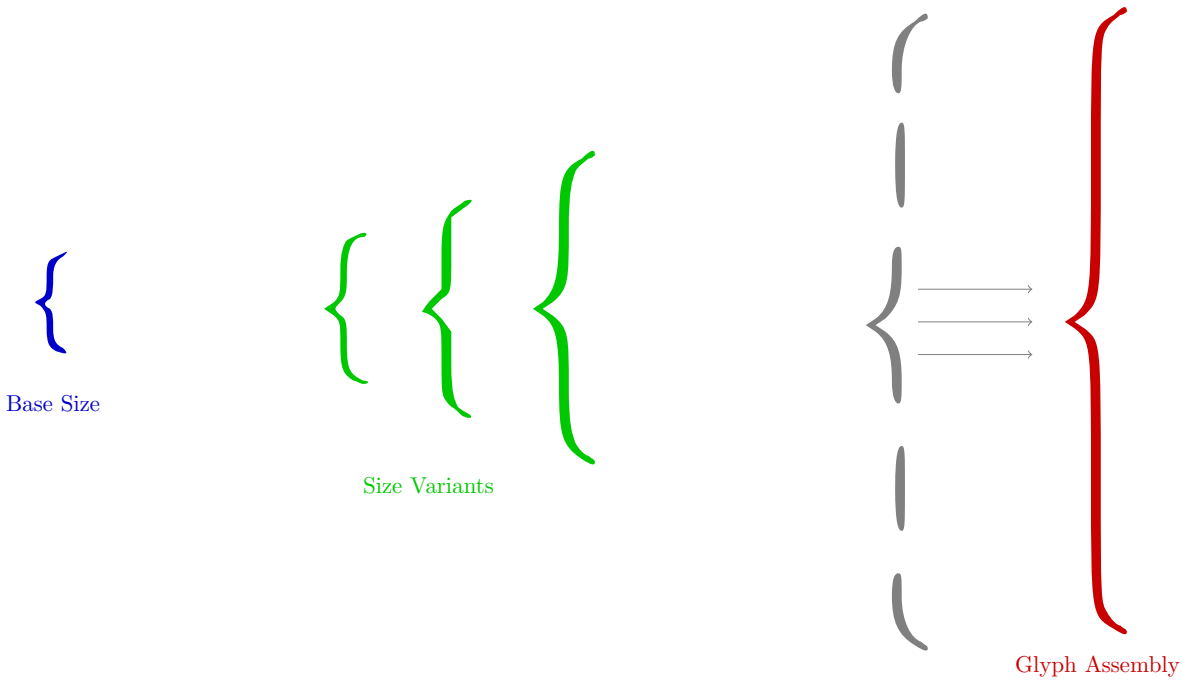


Figure 7 Base size, size variants and glyph assembly for the left brace

3. If the operator has the `largeop` property and if `math-style` on the `<mo>` element is `normal`, then:

    1. Use the `MathVariants` table to try and find a glyph of height at least DisplayOperatorMinHeight. If none is found, fall back to the largest non-base glyph. If none is found, fall back to the layout algorithm of 3.2.1.1 Layout of `<mtext>`.
    2. The min-content inline size, max-content inline size, inline size and block metrics of the math content are given by the glyph found.
    3. Paint the glyph.

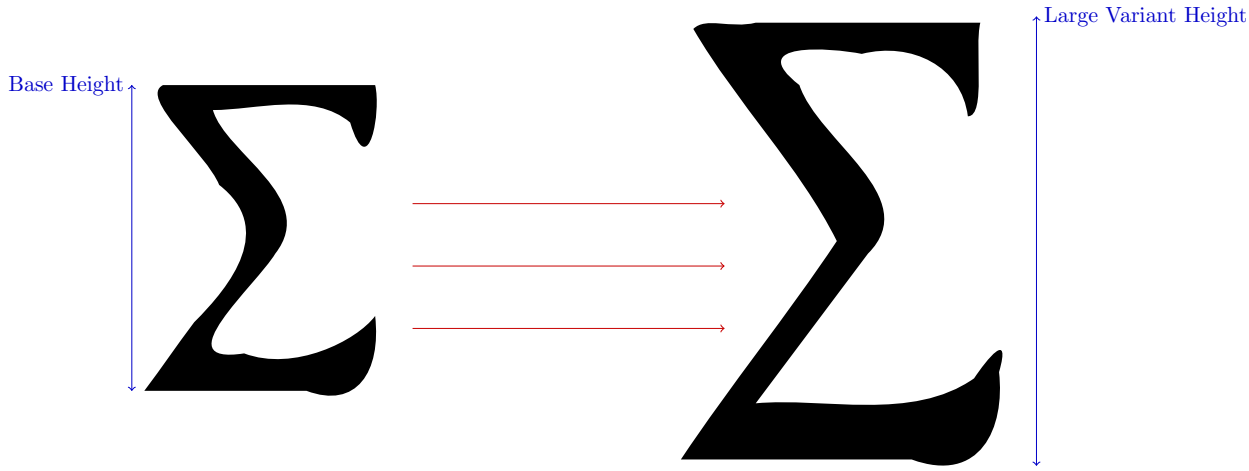


Figure 8 Base and displaystyle sizes of the summation symbol

4. Otherwise fall back to the layout algorithm of 3.2.1.1 Layout of `<mtext>`.

If the algorithm to shape a stretchy glyph has been used for one of the step above, then the italic correction of the math content is set to the value returned by that algorithm.

3.2.5 Space `<mspace>`

The ***mspace*** empty element represents a blank space of any desired size, as set by its attributes.

The `<mspace>` element accepts the attributes described in 2.1.3 Global Attributes as well as the following attributes:

- `width`
- `height`
- `depth`

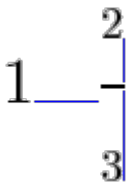
The ***width***, ***height***, ***depth***, if present, must have a value that is a valid <length-percentage>.

- If the `width` attribute is present, valid and not a percentage then that attribute is used as a presentational hint setting the element's width property to the corresponding value.
- If the `height` attribute is absent, invalid or a percentage then the requested line-ascent is `0`. Otherwise the requested line-ascent is the resolved value of the `height` attribute, clamping negative values to `0`.

- If both the `height` and `depth` attributes are present, valid and not a percentage then they are used as a presentational hint setting the element's height property to the concatenation of the strings "`calc(`", the `height` attribute value, "` + `", the `depth` attribute value, and "`)`". If only one of these attributes is present, valid and not a percentage then it is treated as a presentational hint setting the element's height property to the corresponding value.

In the following example, `mspace` is used to force spacing within the formula (a 1px blue border is added to easily visualize the space):

```
<math>
  <mn>1</mn>
  <mspace width="1em"
          style="border-top: 1px solid blue"/>
  <mfrac>
    <mrow>
      <mn>2</mn>
      <mspace depth="1em"
              style="border-left: 1px solid blue"/>
    </mrow>
    <mrow>
      <mn>3</mn>
      <mspace height="2em"
              style="border-left: 1px solid blue"/>
    </mrow>
  </mfrac>
</math>
```



If the <mspace> element does not have its computed `display` property equal to `block math` or `inline math` then it is laid out according to the CSS specification where the corresponding value is described. Otherwise, the <mspace> element is laid out as shown on Figure 9. The min-content inline size, max-content inline size and inline size of the math content are equal to the resolved value of the width property. The block size of the math content is equal to the resolved value of the height property. The line-ascent of the math content is equal to the requested line-ascent determined above.

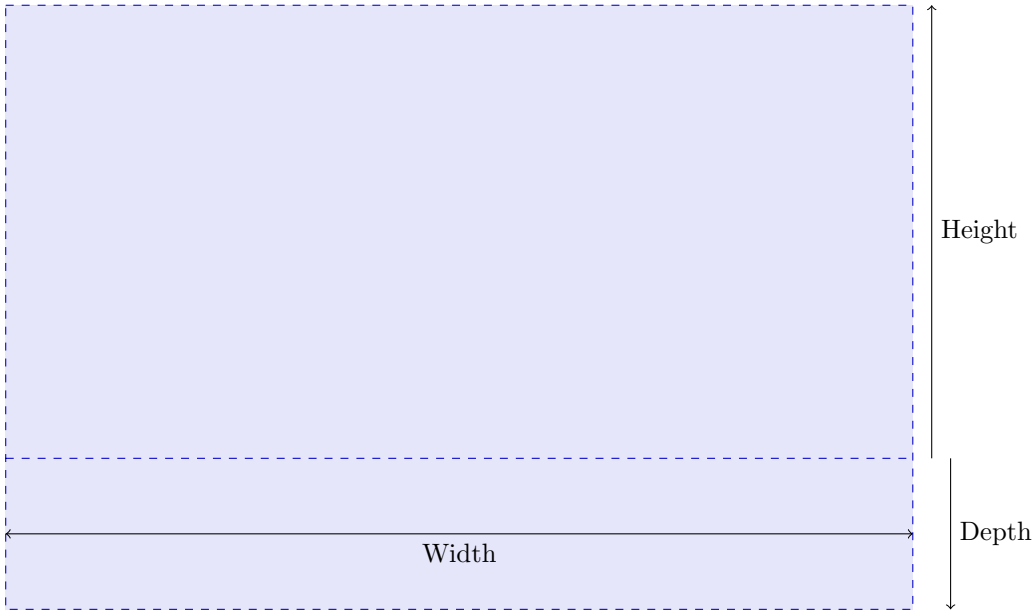


Figure 9 Box model for the <mspace> element

> NOTE
> The terminology height/depth comes from [*MATHML3*], itself inspired from [*TEXBOOK*].

§ *3.2.5.1 Definition of space-like elements*

A number of MathML presentation elements are "space-like" in the sense that they typically render as whitespace, and do not affect the mathematical meaning of the expressions in which they appear. As a consequence, these elements often function in somewhat exceptional ways in other MathML expressions.

A MathML Core element is a ***space-like element*** if it is:

1. an `mtext` or `mspace`;
2. or a grouping element or `mpadded` all of whose in-flow children are space-like.

The same definitions apply for boxes in the visual formatting model where an anonymous <mrow> box is treated as a grouping element.

NOTE

Note that an `mphantom` is not automatically defined to be space-like, unless its content is space-like. This is because operator spacing is affected by whether adjacent elements are space-like. Since the `<mphantom>` element is primarily intended as an aid in aligning expressions, operators adjacent to an `<mphantom>` should behave as if they were adjacent to the contents of the `<mphantom>`, rather than to an equivalently sized area of whitespace.

3.2.6 String Literal `<ms>`

ms element is used to represent "string literals" in expressions meant to be interpreted by computer algebra systems or other systems containing "programming languages".

The `<ms>` element accepts the attributes described in 2.1.3 Global Attributes. Its layout algorithm is the same as the `mtext` element.

In the following example, `ms` is used to write a literal string of characters:

```
<math>
  <mi>s</mi>
  <mo>=</mo>
  <ms>"hello world"</ms>
</math>
```

$s = \text{"hello world"}$

NOTE

In MathML3, it was possible to use the `lquote` and `rquote` attributes to respectively specify the strings to use as opening and closing quotes. These are no longer supported and the quotes must instead be specified as part of the text of the `<ms>` element. One can add CSS rules to legacy documents in order to preserve visual rendering. For example, in left-to-right direction:

```
ms:before, ms:after {
  content: "\0022";
}
ms[lquote]:before {
  content: attr(lquote);
}
ms[rquote]:after {
  content: attr(rquote);
}
```

3.3 General Layout Schemata

Besides tokens there are several families of MathML presentation elements. One family of elements deals with various "scripting" notations, such as subscript and superscript. Another family is concerned with matrices and tables. The remainder of the elements, discussed in this section, describe other basic notations such as fractions and radicals, or deal with general functions such as setting style properties and error handling.

3.3.1 Group Sub-Expressions `<mrow>`

The ***mrow*** element is used to group together any number of sub-expressions, usually consisting of one or more `<mo>` elements acting as "operators" on one or more other expressions that are their "operands".

In the following example, mrow is used to group a sum "1 + 2/3" as a fraction numerator (first child of mfrac) and to construct a fenced expression (first child of msup) that is raised to the power of 5. Note that mrow alone does not add visual fences around its grouped content, one has to explicitly specify them using the mo element.

Within the mrow elements, one can see that vertical alignment of children (according to the alphabetic baseline or the mathematical baseline) is properly performed, fences are vertically stretched and spacing around the binary + operator automatically calculated.

```
<math>
  <msup>
    <mrow>
      <mo>(</mo>
      <mfrac>
        <mrow>
          <mn>1</mn>
          <mo>+</mo>
          <mfrac>
            <mn>2</mn>
            <mn>3</mn>
          </mfrac>
        </mrow>
        <mn>4</mn>
      </mfrac>
      <mo>)</mo>
    </mrow>
    <mn>5</mn>
  </msup>
</math>
```

$$\left(\frac{1+\frac{2}{3}}{4}\right)^{5}$$

The <mrow> element accepts the attributes described in 2.1.3 Global Attributes. An <mrow> element with in-flow children child_1, child_2, …, child_N is laid out as shown on Figure 10. The child boxes are put in a row one after the other with all their alphabetic baselines aligned.

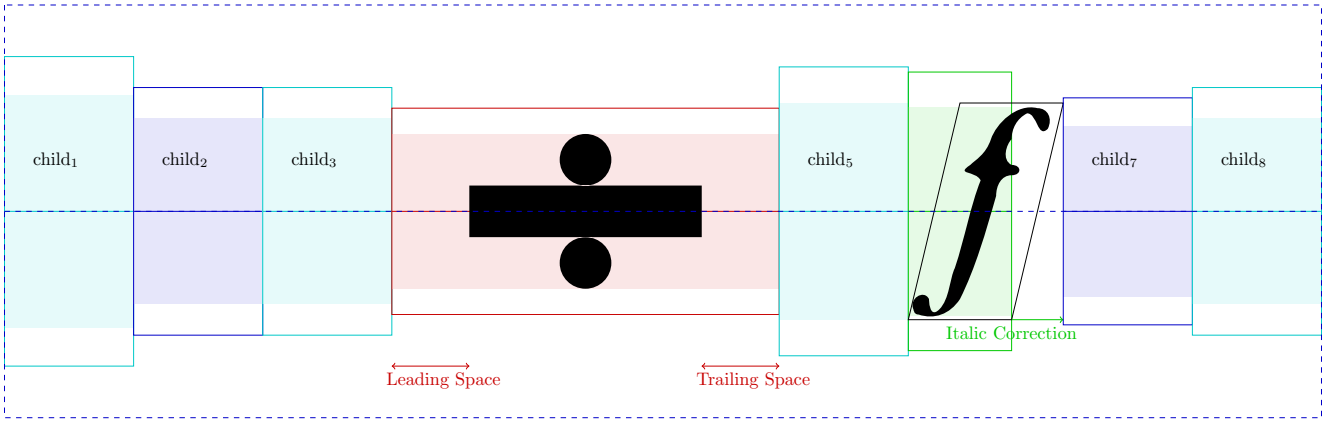


Figure 10 *Box model for the <mrow> element*

> NOTE
> Because the box model ensures alignment of alphabetic baselines, fraction bars or symmetric stretchy operators will also be aligned along the math axis in the typical case when AxisHeight is the same for all in-flow children.

§ *3.3.1.1 Algorithm for stretching operators along the block axis*

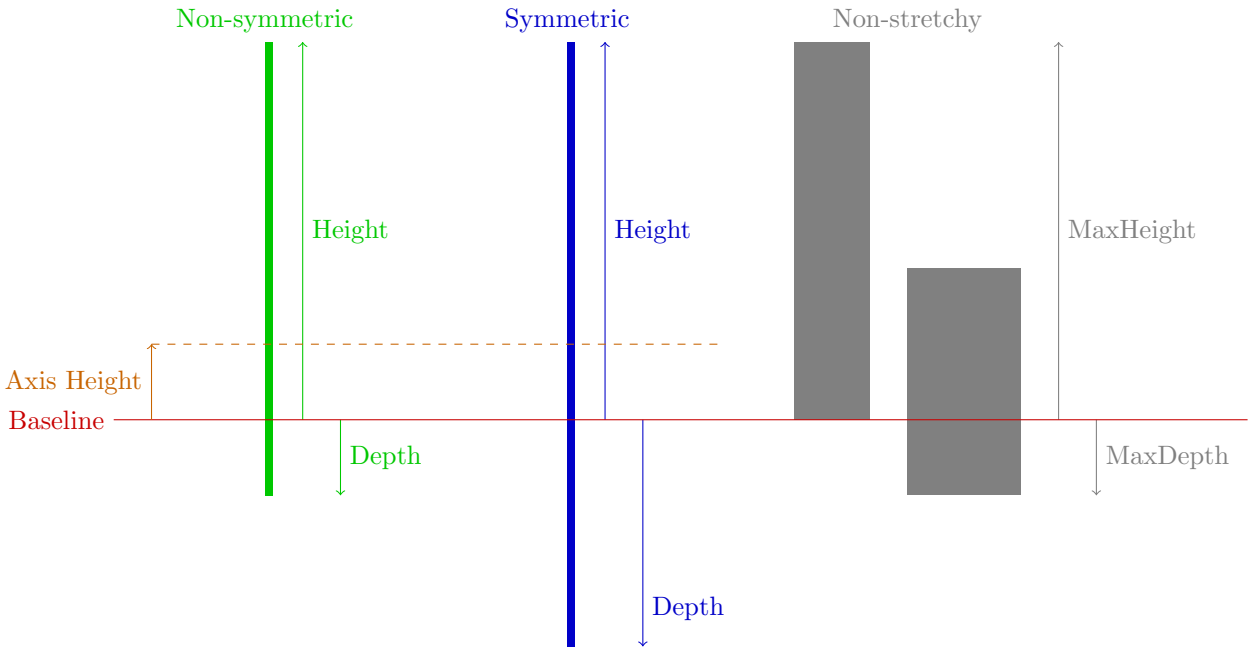


Figure 11 *Symmetric and non-symmetric stretching of operators along the block axis*

The ***algorithm for stretching operators along the block axis*** consists in the following steps:

1. If there is a block stretch size constraint or an inline stretch size constraint then the element being laid out is an embellished operator. Lay out the one in-flow child that is an embellished operator with the same stretch size constraint and all the other in-flow children without any stretch size constraint and stop.
2. Otherwise, split the list of in-flow children into a first list $L_{ToStretch}$ containing embellished operators with a `stretchy` property and block stretch axis; and a second list $L_{NotToStretch}$.
3. Perform layout without any stretch size constraint on all the items of $L_{NotToStretch}$. If $L_{ToStretch}$ is empty then stop. If $L_{NotToStretch}$ is empty, perform layout with block stretch size constraint `(0, 0)` for all the items of $L_{ToStretch}$.
4. Calculate the unconstrained target sizes U_{ascent} and $U_{descent}$ as respectively the maximum ink ascent and maximum ink descent of the margin boxes of in-flow children that have been laid out in the previous step.
5. Lay out or relayout all the elements of $L_{ToStretch}$ with block stretch size constraint $(U_{ascent}, U_{descent})$.

§ *3.3.1.2 Layout of* `<mrow>`

If the box is not an anonymous <mrow> box and the associated element does not have its computed `display` property equal to `block math` or `inline math` then it is laid out according to the CSS specification where the corresponding value is described. Otherwise, the layout below is performed.

A child box is ***slanted*** if it is not an embellished operator and has nonzero italic correction.

> NOTE
> Large operators may have nonzero italic correction but that one is used when attaching scripts. More generally, all embellished operators are treated as non-slanted since the spacing around them is calculated as specified by `lspace` and `rspace`.

The min-content inline size (respectively max-content inline size) are calculated using the following algorithm:

1. Set `add-space` to true if the box corresponds to a `math` element or is not an embellished operator; and to false otherwise.
2. Set `inline-offset` to 0.

3. Set `previous-italic-correction` to 0.
4. For each in-flow child:
    1. If the child is not slanted, then increment `inline-offset` by `previous-italic-correction`.
    2. If the child is an embellished operator and `add-space` is true then increment `inline-offset` by its `lspace` property.
    3. Increment `inline-offset` by the min-content inline size (respectively max-content inline size) of the child's margin box.
    4. If the child is slanted then set `previous-italic-correction` to its italic correction. Otherwise set it to 0.
    5. If the child is an embellished operator and `add-space` is true then increment `inline-offset` by its `rspace` property.
5. Increment `inline-offset` by `previous-italic-correction`.
6. Return `inline-offset`.

The in-flow children are laid out using the algorithm for stretching operators along the block axis.

The inline size of the math content is calculated like the min-content inline size and max-content inline size of the math content, using the inline size of the in-flow children's margin boxes instead.

The ink line-ascent (respectively line-ascent) of the math content is the maximum of the ink line-ascents (respectively line-ascents) of all the in-flow children's margin boxes. Similarly, the ink line-descent (respectively line-descent) of the math content is the maximum of the ink line-descents (respectively ink line-ascents) of all the in-flow children's margin boxes.

The in-flow children are positioned using the following algorithm:

1. Set `add-space` to true if the box corresponds to a `math` element or is not an embellished operator; and to false otherwise.
2. Set `inline-offset` to 0.
3. Set `previous-italic-correction` to 0.
4. For each in-flow child:
    1. If the child is not slanted, then increment `inline-offset` by `previous-italic-correction`.
    2. If the child is an embellished operator and `add-space` is true then increment `inline-offset` by its `lspace` property.

3. Set the inline offset of the child to `inline-offset` and its block offset such that the alphabetic baseline of the child is aligned with the alphabetic baseline.
4. Increment `inline-offset` by the inline size of the child's margin box.
5. If the child is slanted then set `previous-italic-correction` to its italic correction. Otherwise set it to 0.
6. If the child is an embellished operator and `add-space` is true then increment `inline-offset` by its `rspace` property.

The italic correction of the math content is set to the italic correction of the last in-flow child, which is the final value of `previous-italic-correction`.

3.3.2 Fractions `<mfrac>`

The ***mfrac*** element is used for fractions. It can also be used to mark up fraction-like objects such as binomial coefficients and Legendre symbols.

If the `<mfrac>` element does not have its computed `display` property equal to `block math` or `inline math` then it is laid out according to the CSS specification where the corresponding value is described. Otherwise, the layout below is performed.

The `<mfrac>` element accepts the attributes described in 2.1.3 Global Attributes as well as the following attribute:

- `linethickness`

The ***linethickness*** attribute indicates the ***fraction line thickness*** to use for the fraction bar. If present, it must have a value that is a valid <length-percentage>. If the attribute is absent or has an invalid value, FractionRuleThickness is used as the default value. A percentage is interpreted relative to that default value. A negative value is interpreted as 0.

The following example contains four fractions with different `linethickness` values. The bars are always aligned with the middle of plus and minus signs. The numerator and denominator are horizontally centered. The fractions that are not in displaystyle use smaller gaps and font-size.

```
<math>
  <mn>0</mn>
  <mo>+</mo>
  <mfrac displaystyle="true">
    <mn>1</mn>
    <mn>2</mn>
  </mfrac>
  <mo>−</mo>
  <mfrac>
    <mn>1</mn>
    <mn>2</mn>
  </mfrac>
  <mo>+</mo>
  <mfrac linethickness="200%">
    <mn>1</mn>
    <mn>234</mn>
  </mfrac>
  <mo>−</mo>
  <mrow>
    <mo>(</mo>
    <mfrac linethickness="0">
      <mn>123</mn>
      <mn>4</mn>
    </mfrac>
    <mo>)</mo>
  </mrow>
</math>
```

$$0 + \frac{1}{2} - \frac{1}{2} + \frac{1}{234} - \binom{123}{4}$$

The <mfrac> element sets `displaystyle` to `false`, or if it was already `false` increments `scriptlevel` by 1, within its children. It sets math-shift to `compact` within its second child. To avoid visual confusion between the fraction bar and another adjacent items (e.g. minus sign or another fraction's bar), a default 1-pixel space is added around the element. The user agent stylesheet must

contain the following rules:

```
mfrac {
  padding-inline: 1px;
}
mfrac > * {
  math-depth: auto-add;
  math-style: compact;
}
mfrac > :nth-child(2) {
  math-shift: compact;
}
```

If the <mfrac> element has less or more than two in-flow children, its layout algorithm is the same as the mrow element. Otherwise, the first in-flow child is called ***numerator***, the second in-flow child is called ***denominator*** and the layout algorithm is explained below.

> NOTE
> In practice, an <mfrac> element has two children that are in-flow. Hence the CSS rules basically perform scriptlevel, displaystyle and math-shift changes for the numerator and denominator.

§ *3.3.2.1 Fraction with nonzero line thickness*

If the fraction line thickness is nonzero, the <mfrac> element is laid out as shown on Figure 12. The fraction bar must only be painted if the visibility of the <mfrac> element is visible. In that case, the fraction bar must be painted with the color of the <mfrac> element.

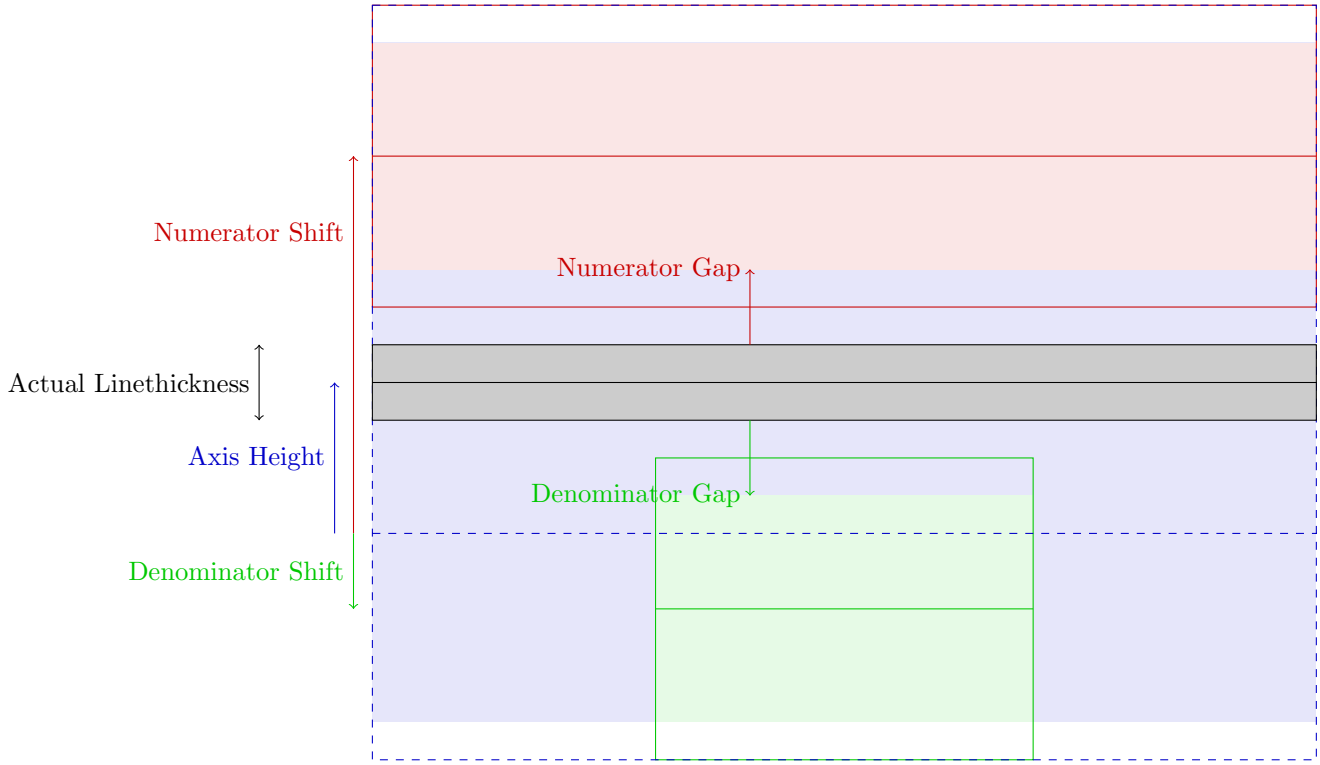


Figure 12 Box model for the `<mfrac>` element

The min-content inline size (respectively max-content inline size) of content is the maximum between the min-content inline size (respectively max-content inline size) of the numerator's margin box and the min-content inline size (respectively max-content inline size) of the denominator's margin box.

If there is an inline stretch size constraint or a block stretch size constraint then the numerator is also laid out with the same stretch size constraint, otherwise it is laid out without any stretch size constraint. The denominator is always laid out without any stretch size constraint.

The inline size of the math content is the maximum between the inline size of the numerator's margin box and the inline size of the denominator's margin box.

`NumeratorShift` is the maximum between:

- FractionNumeratorShiftUp (respectively FractionNumeratorDisplayStyleShiftUp) if the math-style is `compact` (respectively `normal`).
- The AxisHeight + half the fraction line thickness + FractionNumeratorGapMin (respectively FractionNumDisplayStyleGapMin) if the math-style is `compact` (respectively `normal`) + the ink line-descent of the numerator's margin box.

`DenominatorShift` is the maximum between:

- FractionDenominatorShiftDown (respectively FractionDenominatorDisplayStyleShiftDown) if the math-style is `compact` (respectively `normal`).
- Half the fraction line thickness + FractionDenominatorGapMin (respectively FractionDenomDisplayStyleGapMin) if the math-style is `compact` (respectively `normal`) + the ink line-ascent of the denominator's margin box − the AxisHeight.

The line-ascent of the math content is the maximum between:

- `Numerator Shift` + the line-ascent of the numerator's margin box.
- −`Denominator Shift` + the line-ascent of the denominator's margin box
- AxisHeight plus half the fraction line thickness.

The line-descent of the math content is the maximum between:

- −`Numerator Shift` + the line-descent of the numerator's margin box.
- `Denominator Shift` + the line-descent of the denominator's margin box.
- −AxisHeight plus half the fraction line thickness.
- 0

The inline offset of the numerator (respectively denominator) is half the inline size of the math content − half the inline size of the numerator's margin box (respectively denominator's margin box).

The alphabetic baseline of the numerator (respectively denominator) is shifted away from the alphabetic baseline by a distance of `NumeratorShift` (respectively `DenominatorShift`) towards the line-over (respectively line-under).

The math content box is placed within the content box so that their block-start edges are aligned and the middles of these edges are at the same position.

The inline size of the fraction bar is the inline size of the content box and its inline-start edge is the aligned with the one the content box. The center of the fraction bar is shifted away from the alphabetic baseline of the math content box by a distance of AxisHeight towards the line-over. Its block size is the fraction line thickness.

§ *3.3.2.2 Fraction with zero line thickness*

If the fraction line thickness is zero, the `<mfrac>` element is instead laid out as shown on Figure 13.

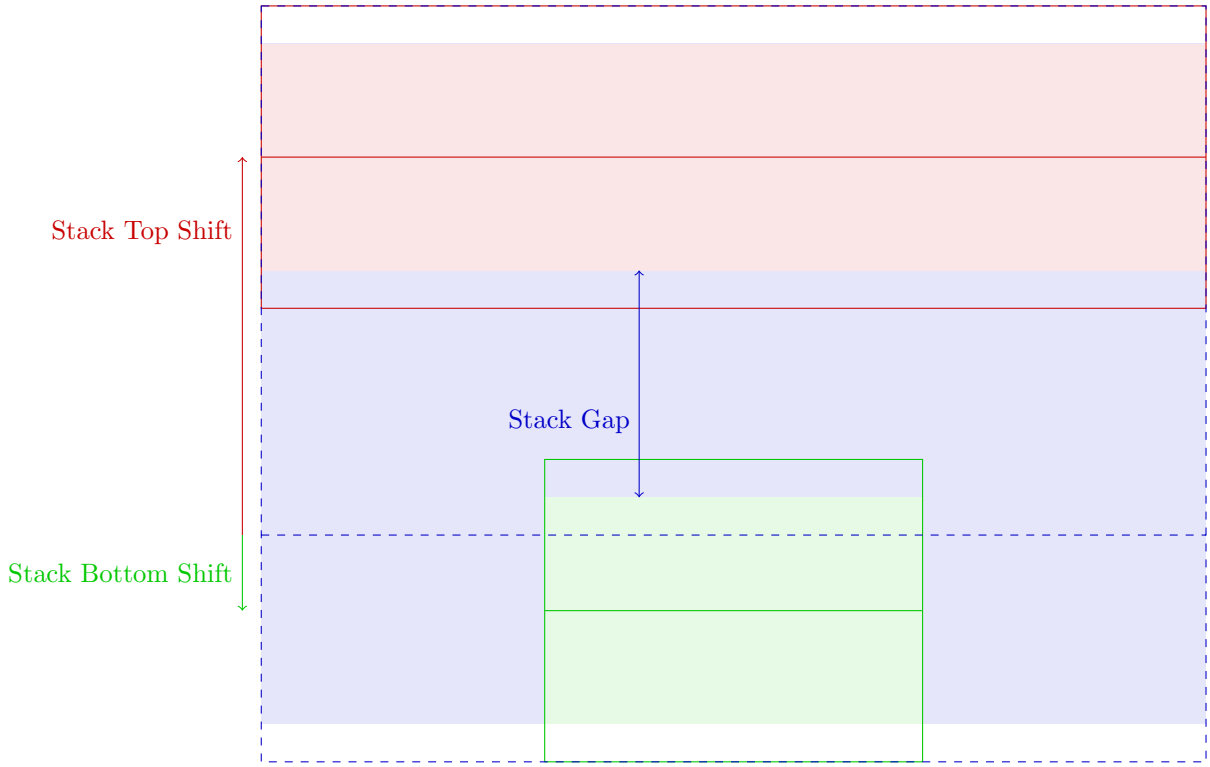


Figure 13 Box model for the `<mfrac>` element without bar

The min-content inline size, max-content inline size and inline size of the math content are calculated the same as in 3.3.2.1 Fraction with nonzero line thickness.

If there is an inline stretch size constraint or a block stretch size constraint then the numerator is also laid out with the same stretch size constraint and otherwise it is laid out without any stretch size constraint. The denominator is always laid out without any stretch size constraint.

If the math-style is `compact` then `TopShift` and `BottomShift` are respectively set to StackTopShiftUp and StackBottomShiftDown. Otherwise math-style is `normal` and they are respectively set to StackTopDisplayStyleShiftUp and StackBottomDisplayStyleShiftDown.

The `Gap` is defined to be (`BottomShift` – the ink line-ascent of the denominator's margin box) + (`TopShift` – the ink line-descent of the numerator's margin box). If math-style is `compact` then `GapMin` is StackGapMin, otherwise math-style is `normal` and it is StackDisplayStyleGapMin. If Δ = `GapMin` – `Gap` is positive then `TopShift` and `BottomShift` are respectively increased by Δ/2 and Δ – Δ/2.

The line-ascent of the math content is the maximum between:

- `TopShift` + the line-ascent of the numerator's margin box.
- −`BottomShift` + the line-ascent of the denominator's margin box.

The line-descent of the math content is the maximum between:

- −`TopShift` + the line-descent of the numerator's margin box.
- `BottomShift` + the line-descent of the denominator's margin box.
- 0

The inline offsets of the numerator and denominator are calculated the same as in 3.3.2.1 Fraction with nonzero line thickness.

The alphabetic baseline of the numerator (respectively denominator) is shifted away from the alphabetic baseline by a distance of `TopShift` (respectively – `BottomShift`) towards the line-over (respectively line-under).

The math content box is placed within the content box so that their block-start edges are aligned and the middles of these edges are at the same position.

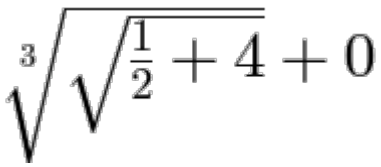
3.3.3 Radicals `<msqrt>`, `<mroot>`

The radical elements construct an expression with a root symbol √ with a line over the content. The ***msqrt*** element is used for square roots, while the ***mroot*** element is used to draw radicals with indices, e.g. a cube root.

The `<msqrt>` and `<mroot>` elements accept the attributes described in 2.1.3 Global Attributes.

The following example contains a square root written with `msqrt` and a cube root written with `mroot`. Note that `msqrt` has several children and the square root applies to all of them. `mroot` has exactly two children: it is a root of index the second child (the number 3), applied to the first child (the square root). Also note these elements only change the font-size within the `mroot` index, but it is scaled down more than within the numerator and denumerator of the fraction.

```
<math>
  <mroot>
    <msqrt>
      <mfrac>
        <mn>1</mn>
        <mn>2</mn>
      </mfrac>
      <mo>+</mo>
      <mn>4</mn>
    </msqrt>
    <mn>3</mn>
  </mroot>
  <mo>+</mo>
  <mn>0</mn>
</math>
```



The `<msqrt>` and `<mroot>` elements sets math-shift to `compact`. The `<mroot>` element increments `scriptlevel` by 2, and sets `displaystyle` to "false" in all but its first child. The user agent stylesheet must contain the following rule in order to implement that behavior:

```
mroot > :not(:first-child) {
  math-depth: add(2);
  math-style: compact;
}
mroot, msqrt {
  math-shift: compact;
}
```

If the `<msqrt>` or `<mroot>` element do not have their computed `display` property equal to `block`

`math` or `inline math` then they are laid out according to the CSS specification where the corresponding value is described. Otherwise, the layout below is performed.

If the `<mroot>` has less or more than two in-flow children, its layout algorithm is the same as the `mrow` element. Otherwise, the first in-flow child is called ***mroot base*** and the second in-flow child is called ***mroot index*** and its layout algorithm is explained below.

> NOTE
> In practice, an `<mroot>` element has two children that are in-flow. Hence the CSS rules basically perform `scriptlevel` and `displaystyle` changes for the index.

The `<msqrt>` element generates an anonymous <mrow> box called the ***msqrt base***.

§ *3.3.3.1 Radical symbol*

The radical symbol must only be painted if the visibility of the `<msqrt>` or `<mroot>` element is `visible`. In that case, the radical symbol must be painted with the color of that element.

The ***radical glyph*** is the glyph obtained for the character U+221A SQUARE ROOT.

The ***radical gap*** is given by RadicalVerticalGap if the math-style is `compact` and RadicalDisplayStyleVerticalGap if the math-style is `normal`.

The radical target size for the stretchy radical glyph is the sum of RadicalRuleThickness, radical gap and the ink height of the base.

The ***box metrics of the radical glyph*** and ***painting of the surd*** are given by the algorithm to shape a stretchy glyph to block dimension the target size for the radical glyph.

§ *3.3.3.2 Square root*

The `<msqrt>` element is laid out as shown on Figure 14.

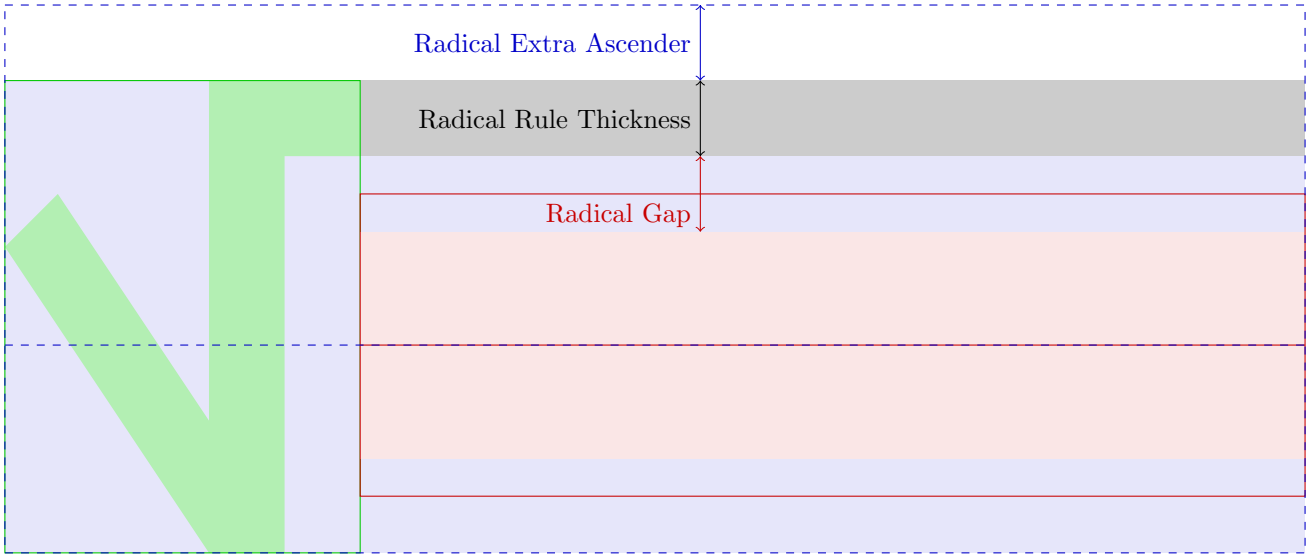


Figure 14 Box model for the `<msqrt>` element

The min-content inline size (respectively max-content inline size) of the math content is the sum of the preferred inline size of a glyph stretched along the block axis for the radical glyph and of the min-content inline size (respectively max-content inline size) of the msqrt base's margin box.

The inline size of the math content is the sum of the advance width of the box metrics of the radical glyph and of the inline size of the msqrt base's margin's box.

The line-ascent of the math content is the maximum between:

- The line-ascent of the msqrt base's margin box.
- The sum of the ink line-ascent of the msqrt base, the radical gap, the RadicalRuleThickness and RadicalExtraAscender.

The line-descent of the math content is the maximum between:

- The line-descent of the msqrt base's margin box.
- The sum of the line-ascent and line-descent for the box metrics of the radical glyph and of the RadicalExtraAscender, minus the line-ascent of the math content.

The inline size of the overbar is the inline size of the msqrt base's margin's box. The inline offsets of the msqrt base and overbar are also the same and equal to the width of the box metrics of the radical glyph.

The alphabetic baseline of the msqrt base is aligned with the alphabetic baseline. The block size of the overbar is RadicalRuleThickness. Its vertical center is shifted away from the alphabetic baseline by a distance towards the line-over equal to the line-ascent of the math content, minus the

RadicalExtraAscender, minus half the RadicalRuleThickness.

Finally, the painting of the surd is performed:

- At inline offset 0.
- At block offset shifted away from the line-over edge of the overbar towards the line-under by a distance given by the ascent of the box metrics of the radical glyph.

§ *3.3.3.3 Root with index*

The `<mroot>` element is laid out as shown on Figure 15. The mroot index is first ignored and the mroot base and radical glyph are laid out as shown on figure Figure 14 using the same algorithm as in 3.3.3.2 Square root in order to produce a margin box B (represented in green).

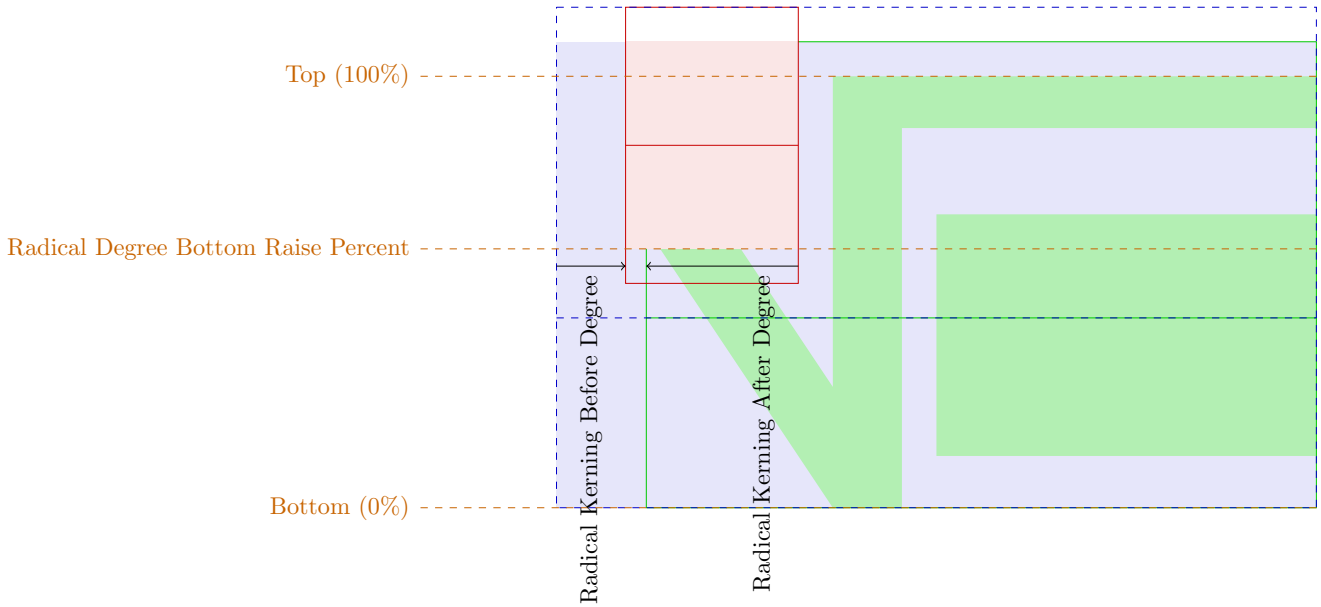


Figure 15 Box model for the `<mroot>` element

The min-content inline size (respectively max-content inline size) of the math content is the sum of max(0, RadicalKernBeforeDegree), the mroot index's min-content inline size (respectively max-content inline size) of the mroot index's margin box, max(−min-content inline size, RadicalKernAfterDegree) (respectively max(−max-content inline size of the mroot index's margin box, RadicalKernAfterDegree)) and of the min-content inline size (respectively max-content inline size) of B.

Using the same clamping, ***AdjustedRadicalKernBeforeDegree*** and ***AdjustedRadicalKernAfterDegree*** are respectively defined as max(0, RadicalKernBeforeDegree) and is max(−inline size of the index's margin box, RadicalKernAfterDegree).

The inline size of the math content is the sum of AdjustedRadicalKernBeforeDegree, the inline size of the index's margin box, AdjustedRadicalKernAfterDegree and of the inline size of B.

The line-ascent of the math content is the maximum between:

- −the line-descent of B + RadicalDegreeBottomRaisePercent × the block size of B + the line-ascent of the index's margin box
- The line-ascent of B

The line-descent of the math content is the maximum between:

- the line-descent of B − RadicalDegreeBottomRaisePercent × the block size of B + the line-ascent of the index's margin box
- The line-descent of B

The inline offset of the index is AdjustedRadicalKernBeforeDegree. The inline-offset of the mroot base is the same + the inline size of the index's margin box.

The alphabetic baseline of B is aligned with the alphabetic baseline. The alphabetic baseline of the index is shifted away from the line-under edge by a distance of RadicalDegreeBottomRaisePercent × the block size of B + the line-descent of the index's margin box.

> NOTE
> In general, the kerning before the root index is positive while the kerning after it is negative, which means that the root element will have some inline-start space and that the root index will overlap the surd.

3.3.4 Style Change `<mstyle>`

Historically, the ***mstyle*** element was introduced to make style changes that affect the rendering of its contents.

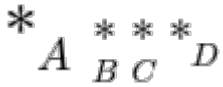
The `<mstyle>` element accepts the attributes described in 2.1.3 Global Attributes. Its layout algorithm is the same as the `mrow` element.

NOTE

`<mstyle>` is implemented for compatibility with full MathML. Authors whose only target is MathML Core are encouraged to use CSS for styling.

In the following example, `mstyle` is used to set the scriptlevel and displaystyle. Observe this is respectively affecting the font-size and placement of subscripts of their descendants. In MathML Core, one could just have used `mrow` elements instead.

```
<math>
  <munder>
    <mo movablelimits="true">*</mo>
    <mi>A</mi>
  </munder>
  <mstyle scriptlevel="1">
    <mstyle displaystyle="true">
      <munder>
        <mo movablelimits="true">*</mo>
        <mi>B</mi>
      </munder>
      <munder>
        <mo movablelimits="true">*</mo>
        <mi>C</mi>
      </munder>
    </mstyle>
    <munder>
      <mo movablelimits="true">*</mo>
      <mi>D</mi>
    </munder>
  </mstyle>
</math>
```



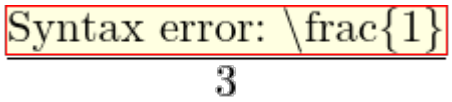
3.3.5 Error Message `<merror>`

The ***merror*** element displays its contents as an ”error message”. The intent of this element is to

provide a standard way for programs that generate MathML from other input to report syntax errors in their input.

In the following example, merror is used to indicate a parsing error for some LaTeX-like input:

```
<math>
  <mfrac>
    <merror>
      <mtext>Syntax error: \frac{1}</mtext>
    </merror>
    <mn>3</mn>
  </mfrac>
</math>
```



The <merror> element accepts the attributes described in 2.1.3 Global Attributes. Its layout algorithm is the same as the mrow element. The user agent stylesheet must contain the following rule in order to visually highlight the error message:

```
merror {
  border: 1px solid red;
  background-color: lightYellow;
}
```

§ 3.3.6 Adjust Space Around Content <mpadded>

The ***mpadded*** element renders the same as its in-flow child content, but with the size and relative positioning point of its content modified according to <mpadded>'s attributes.

The <mpadded> element accepts the attributes described in 2.1.3 Global Attributes as well as the following attributes:

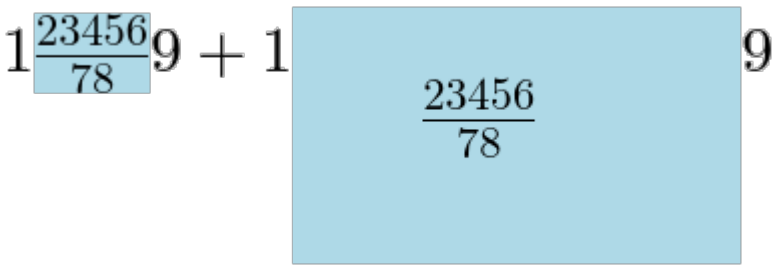
- width
- height
- depth

- `lspace`
- `voffset`

The ***width***, ***height***, ***depth***, ***lspace*** and ***voffset*** if present, must have a value that is a valid <length-percentage>.

In the following example, mpadded is used to tweak spacing around a fraction (a blue background is used to visualize it). Without attributes, it behaves like an mrow but the attributes allow to specify the size of the box (width, height, depth) and position of the fraction within that box (lspace and voffset).

```
<math>
  <mrow>
    <mn>1</mn>
    <mpadded style="background: lightblue;">
      <mfrac>
        <mn>23456</mn>
        <mn>78</mn>
      </mfrac>
    </mpadded>
    <mn>9</mn>
  </mrow>
  <mo>+</mo>
  <mrow>
    <mn>1</mn>
    <mpadded lspace="2em" voffset="-1em" height="1em" depth="3em" width='
             style="background: lightblue;">
      <mfrac>
        <mn>23456</mn>
        <mn>78</mn>
      </mfrac>
    </mpadded>
    <mn>9</mn>
  </mrow>
</math>
```



§ *3.3.6.1 Inner box and requested parameters*

The `mpadded` element generates an anonymous <mrow> box called the ***mpadded inner box*** with parameters called inner inline size, inner line-ascent and inner line-descent.

The requested `<mpadded>` parameters are determined as follows:

- The requested width is the resolved value of the width property. If the `width` attribute is present, valid and not a percentage then that attribute is used as a presentational hint setting the element's width property to the corresponding value.
- If the `height` attribute is absent, invalid or a percentage then the requested height is the inner line-ascent. Otherwise the requested height is the resolved value of the `height` attribute, clamping negative values to `0`.
- If the `depth` attribute is absent, invalid or a percentage then the requested depth is the inner line-ascent. Otherwise the requested depth is the resolved value of the `depth` attribute, clamping negative values to `0`.
- If the `lspace` attribute is absent, invalid or a percentage then the requested lspace is 0. Otherwise the requested lspace is the resolved value of the `lspace` attribute, clamping negative values to `0`.
- If the `voffset` attribute is absent, invalid or a percentage then the requested voffset is 0. Otherwise the requested voffset is the resolved value of the `voffset` attribute.

> NOTE
> Negative `voffset` values are not clamped to `0`.

3.3.6.2 Layout of `<mpadded>`

If the `<mpadded>` element does not have its computed `display` property equal to `block math` or `inline math` then it is laid out according to the CSS specification where the corresponding value is described. Otherwise, it is laid out as shown on Figure 16.

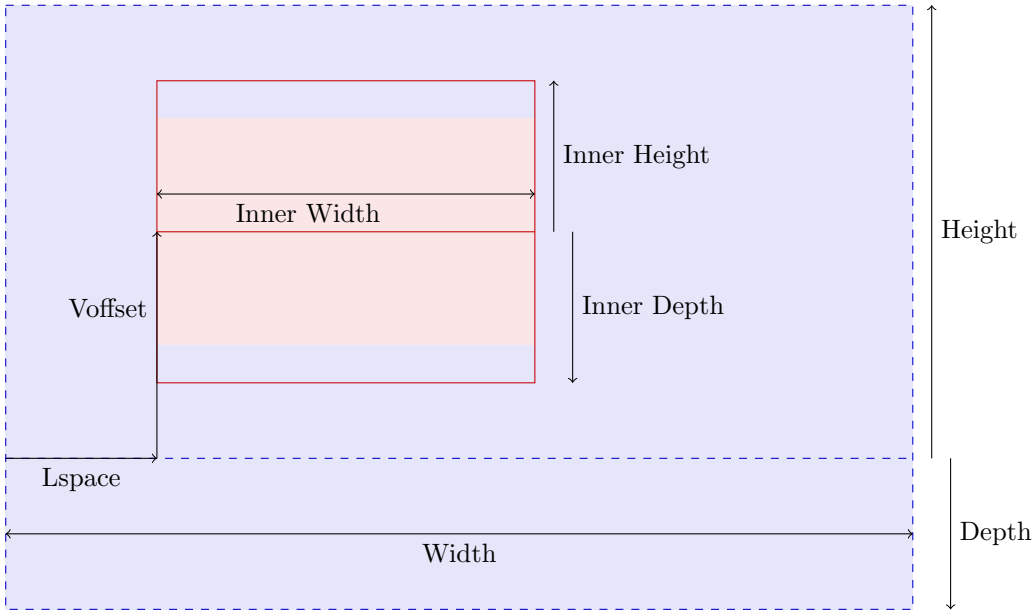


Figure 16 Box model for the <mpadded> element

The min-content inline size (respectively max-content inline size) of the math content is the requested width calculated in 3.3.6.1 Inner box and requested parameters but using the min-content inline size (respectively max-content inline size) of the mpadded inner box instead of the "inner inline size".

The inline size of the math content is the requested width calculated in 3.3.6.1 Inner box and requested parameters.

The line-ascent of the math content is the requested height. The line-descent of the math content is the requested depth.

The mpadded inner box is placed so that its alphabetic baseline is shifted away from the alphabetic baseline by the requested voffset towards the line-over.

§ 3.3.7 Making Sub-Expressions Invisible <mphantom>

Historically, the ***mphantom*** element was introduced to render its content invisibly, but with the same metrics size and other dimensions, including alphabetic baseline position that its contents would have if they were rendered normally.

In the following example, mphantom is used to ensure alignment of corresponding parts of the numerator and denominator of a fraction:

```
<math>
  <mfrac>
    <mrow>
      <mi>x</mi>
      <mo>+</mo>
      <mi>y</mi>
      <mo>+</mo>
      <mi>z</mi>
    </mrow>
    <mrow>
      <mi>x</mi>
      <mphantom>
        <mo form="infix">+</mo>
        <mi>y</mi>
      </mphantom>
      <mo>+</mo>
      <mi>z</mi>
    </mrow>
  </mfrac>
</math>
```

$$\frac{x+y+z}{x+z}$$

The <mphantom> element accepts the attributes described in 2.1.3 Global Attributes. Its layout algorithm is the same as the mrow element. The user agent stylesheet must contain the following rule in order to hide the content:

```
mphantom {
  visibility: hidden;
}
```

NOTE

<mphantom> is implemented for compatibility with full MathML. Authors whose only target is MathML Core are encouraged to use CSS for styling.

3.4 Script and Limit Schemata

The elements described in this section position one or more scripts around a base. Attaching various kinds of scripts and embellishments to symbols is a very common notational device in mathematics. For purely visual layout, a single general-purpose element could suffice for positioning scripts and embellishments in any of the traditional script locations around a given base. However, in order to capture the abstract structure of common notation better, MathML provides several more specialized scripting elements.

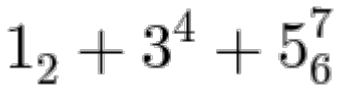
In addition to sub-/superscript elements, MathML has overscript and underscript elements that place scripts above and below the base. These elements can be used to place limits on large operators, or for placing accents and lines above or below the base.

3.4.1 Subscripts and Superscripts `<msub>`, `<msup>`, `<msubsup>`

The ***msub***, ***msup*** and ***msubsup*** elements are used to attach subscript and superscript to a MathML expression. They accept the attributes described in 2.1.3 Global Attributes.

The following example shows basic use of subscripts and superscripts. The font-size is automatically scaled down within the scripts.

```
<math>
  <msub>
    <mn>1</mn>
    <mn>2</mn>
  </msub>
  <mo>+</mo>
  <msup>
    <mn>3</mn>
    <mn>4</mn>
  </msup>
  <mo>+</mo>
  <msubsup>
    <mn>5</mn>
    <mn>6</mn>
    <mn>7</mn>
  </msubsup>
</math>
```



If the <msub>, <msup> or <msubsup> elements do not have their computed display property equal to block math or inline math then they are laid out according to the CSS specification where the corresponding value is described. Otherwise, the layout below is performed.

§ *3.4.1.1 Children of <msub>, <msup>, <msubsup>*

If the <msub> element has less or more than two in-flow children, its layout algorithm is the same as the mrow element. Otherwise, the first in-flow child is called the ***msub base***, the second in-flow child is called the ***msub subscript*** and the layout algorithm is explained in 3.4.1.2 Base with subscript.

If the <msup> element has less or more than two in-flow children, its layout algorithm is the same as the mrow element. Otherwise, the first in-flow child is called the ***msup base***, the second in-flow child is called the ***msup superscript*** and the layout algorithm is explained in 3.4.1.3 Base with superscript.

If the <msubsup> element has less or more than three in-flow children, its layout algorithm is the same as the mrow element. Otherwise, the first in-flow child is called the ***msubsup base***, the second in-flow child is called the ***msubsup subscript***, its third in-flow child is called the ***msubsup superscript*** and the layout algorithm is explained in 3.4.1.4 Base with subscript and superscript.

§ *3.4.1.2 Base with subscript*

The <msub> element is laid out as shown on Figure 17. LargeOpItalicCorrection is the italic correction of the msub base if it is an embellished operator with the largeop property and 0 otherwise.

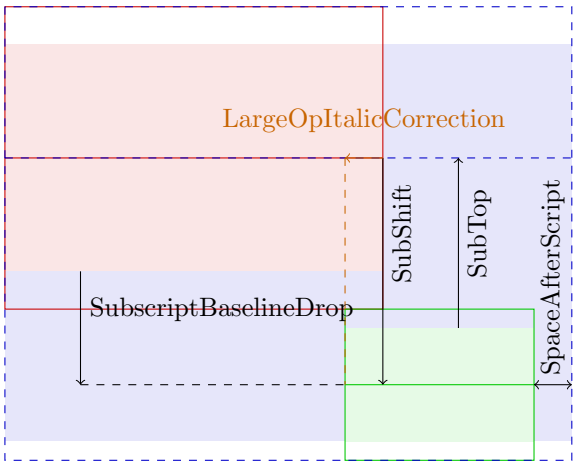


Figure 17 Box model for the <msub> element

The min-content inline size (respectively max-content inline size) of the math content is the min-content inline size (respectively max-content inline size) of the msub base's margin box − LargeOpItalicCorrection + min-content inline size (respectively max-content inline size) of the msub subscript's margin box + SpaceAfterScript.

If there is an inline stretch size constraint or a block stretch size constraint then the msub base is also laid out with the same stretch size constraint and otherwise it is laid out without any stretch size constraint. The scripts are always laid out without any stretch size constraint.

The inline size of the math content is the inline size of the msub base's margin box − LargeOpItalicCorrection + the inline size of the msub subscript's margin box + SpaceAfterScript.

`SubShift` is the maximum between:

- SubscriptShiftDown
- the ink line-ascent of the subscript's margin box − SubscriptTopMax
- SubscriptBaselineDropMin + the ink line-descent of the msub base's margin box

The line-ascent of the math content is the maximum between:

- The line-ascent of the msub base's margin box.
- The line-ascent of the subscript's margin box − `SubShift`.

The line-descent of the math content is the maximum between:

- The line-descent of the msub base's margin box.
- The line-descent of the subscript's margin box + `SubShift`.

The inline offset of the msub base is 0 and the inline offset of the msub subscript is the inline size of the msub base's margin box − `LargeOpItalicCorrection`.

The msub base is placed so that its alphabetic baseline matches the alphabetic baseline. The msub subscript is placed so that its alphabetic baseline is shifted away from the alphabetic baseline by `SubShift` towards the line-under.

3.4.1.3 Base with superscript

The `<msup>` element is laid out as shown on Figure 18. `ItalicCorrection` is the italic correction of the msup base if it is not an embellished operator with the `largeop` property and 0 otherwise.

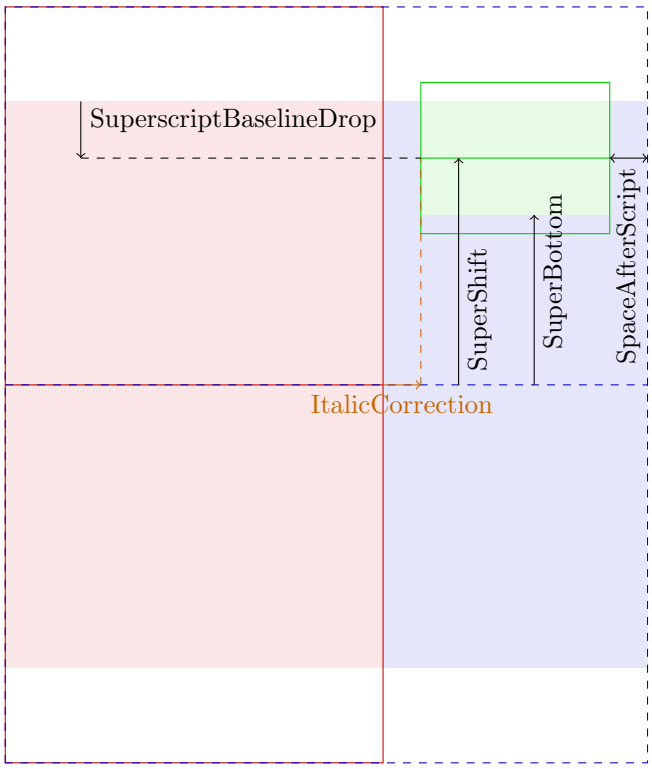


Figure 18 Box model for the `<msup>` element

The min-content inline size (respectively max-content inline size) of the math content is the min-content inline size (respectively max-content inline size) of the msup base's margin box + `ItalicCorrection` + the min-content inline size (respectively max-content inline size) of the msup superscript's margin box + SpaceAfterScript.

If there is an inline stretch size constraint or a block stretch size constraint then the msup base is also laid out with the same stretch size constraint and otherwise it is laid out without any stretch size constraint. The scripts are always laid out without any stretch size constraint.

The inline size of the math content is the inline size of the msup base's margin box + `ItalicCorrection` + the inline size of the msup superscript's margin box + SpaceAfterScript.

`SuperShift` is the maximum between:

- The value SuperscriptShiftUpCramped if the element has a computed math-shift property equal to `compact`, or SuperscriptShiftUp otherwise.
- SuperscriptBottomMin + the ink line-descent of the superscript's margin box
- The ink line-ascent of the msup base's margin box − SuperscriptBaselineDropMax

The line-ascent of the math content is the maximum between:

- The line-ascent of the msup base's margin box.
- The line-ascent of the superscript's margin box + `SuperShift`.

The line-descent of the math content is the maximum between:

- The line-descent of the msup base's margin box.
- The line-descent of the superscript's margin box − `SuperShift`.

The inline offset of the msup base is 0 and the inline offset of msup superscript is the inline size of the msup base's margin box + `ItalicCorrection`.

The msup base is placed so that its alphabetic baseline matches the alphabetic baseline. The msup superscript is placed so that its alphabetic baseline is shifted away from the alphabetic baseline by `SuperShift` towards the line-over.

§ *3.4.1.4 Base with subscript and superscript*

The `<msubsup>` element is laid out as shown on Figure 18. `LargeOpItalicCorrection` and `SubShift` are set as in 3.4.1.2 Base with subscript. `ItalicCorrection` and `SuperShift` are set as in 3.4.1.3 Base with superscript.

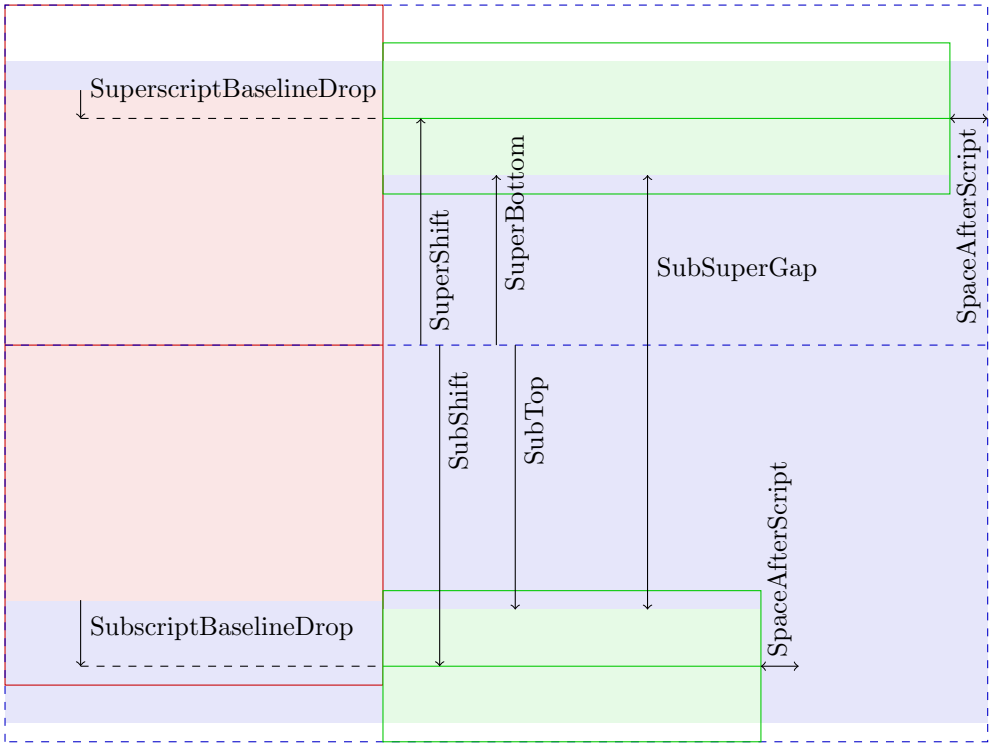


Figure 19 Box model for the <msubsup> element

The min-content inline size (respectively max-content inline size and inline size) of the math content is the maximum between the min-content inline size (respectively max-content inline size and inline size) of the math content calculated in 3.4.1.2 Base with subscript and 3.4.1.3 Base with superscript.

If there is an inline stretch size constraint or a block stretch size constraint then the msubsup base is also laid out with the same stretch size constraint and otherwise it is laid out without any stretch size constraint. The scripts are always laid out without any stretch size constraint.

If there is an inline stretch size constraint or a block stretch size constraint then the msubsup base is also laid out with the same stretch size constraint and otherwise it is laid out without any stretch size constraint. The scripts are always laid out without any stretch size constraint.

`SubSuperGap` is the gap between the two scripts along the block axis and is defined by (`SubShift` – the ink line-ascent of the msubsup subscript's margin box) + (`SuperShift` – the ink line-descent of the msubsup superscript's margin box). If `SubSuperGap` is not at least SubSuperscriptGapMin then the following steps are performed to ensure that the condition holds:

1. Let Δ be SuperscriptBottomMaxWithSubscript – (`SuperShift` – the ink line-descent of the msubsup superscript's margin box). If Δ > 0 then set Δ to the minimum between Δ set SubSuperscriptGapMin – `SubSuperGap` and increase `SuperShift` (and so `SubSuperGap` too) by Δ.

2. Let Δ be SubSuperscriptGapMin – `SubSuperGap`. If $\Delta > 0$ then increase `SubscriptShift` (and so `SubSuperGap` too) by Δ.

The ink line-ascent (respectively line-ascent, ink line-descent, line-descent) of the math content is set to the maximum of the ink line-ascent (respectively line-ascent, ink line-descent, line-descent) of the math content calculated in 3.4.1.2 Base with subscript and 3.4.1.3 Base with superscript but using the adjusted values `SubShift` and `SuperShift` above.

The inline offset and block offset of the msubsup base and scripts are performed the same as described in 3.4.1.2 Base with subscript and 3.4.1.3 Base with superscript.

> NOTE
>
> Even when the msubsup subscript (respectively msubsup superscript) is an empty box, `<msubsup>` does not generally render the same as 3.4.1.3 Base with superscript (respectively 3.4.1.2 Base with subscript) because of the additional constraint on `SubSuperGap`. Moreover, positioning the empty msubsup subscript (respectively msubsup superscript) may also change the total size.
>
> In order to keep the algorithm simple, no attempt is made to handle empty scripts in a special way.

3.4.2 Underscripts and Overscripts `<munder>`, `<mover>`, `<munderover>`

The ***munder***, ***mover*** and ***munderover*** elements are used to attach accents or limits placed under or over a MathML expression.

The `<munderover>` element accepts the attribute described in 2.1.3 Global Attributes as well as the following attributes:

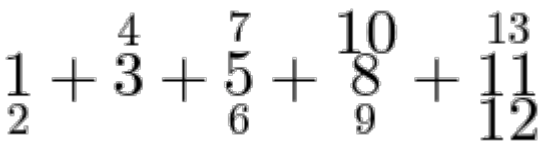
- `accent`
- `accentunder`

Similarly, the `<mover>` element (respectively `<munder>` element) accepts the attribute described in 2.1.3 Global Attributes as well as the `accent` attribute (respectively the `accentunder` attribute).

accent, ***accentunder*** attributes, if present, must have values that are booleans. If these attributes are absent or invalid, they are treated as equal to `false`. User agents must implement them as described in 3.4.4 Displaystyle, scriptlevel and math-shift in scripts.

The following example shows basic use of under- and overscripts. The font-size is automatically scaled down within the scripts, unless they are meant to be accents.

```
<math>
  <munder>
    <mn>1</mn>
    <mn>2</mn>
  </munder>
  <mo>+</mo>
  <mover>
    <mn>3</mn>
    <mn>4</mn>
  </mover>
  <mo>+</mo>
  <munderover>
    <mn>5</mn>
    <mn>6</mn>
    <mn>7</mn>
  </munderover>
  <mo>+</mo>
  <munderover accent="true">
    <mn>8</mn>
    <mn>9</mn>
    <mn>10</mn>
  </munderover>
  <mo>+</mo>
  <munderover accentunder="true">
    <mn>11</mn>
    <mn>12</mn>
    <mn>13</mn>
  </munderover>
</math>
```



If the `<munder>`, `<mover>` or `<munderover>` elements do not have their computed `display` property equal to `block math` or `inline math` then they are laid out according to the CSS specification where the corresponding value is described. Otherwise, the layout below is performed.

3.4.2.1 Children of <munder>, <mover>, <munderover>

If the <munder> element has less or more than two in-flow children, its layout algorithm is the same as the mrow element. Otherwise, the first in-flow child is called the ***munder base*** and the second in-flow child is called the ***munder underscript***.

If the <mover> element has less or more than two in-flow children, its layout algorithm is the same as the mrow element. Otherwise, the first in-flow child is called the ***mover base*** and the second in-flow child is called the ***mover overscript***.

If the <munderover> element has less or more than three in-flow children, its layout algorithm is the same as the mrow element. Otherwise, the first in-flow child is called the ***munderover base***, the second in-flow child is called the ***munderover underscript*** and its third in-flow child is called the ***munderover overscript***.

If the <munder>, <mover> or <munderover> elements have a computed math-style property equal to compact and their base is an embellished operator with the movablelimits property, then their layout algorithms are respectively the same as the ones described for <msub>, <msup> and <msubsup> in 3.4.1.2 Base with subscript, 3.4.1.3 Base with superscript and 3.4.1.4 Base with subscript and superscript.

Otherwise, the <munder>, <mover> and <munderover> layout algorithms are respectively described in 3.4.2.3 Base with underscript, 3.4.2.4 Base with overscript and 3.4.2.5 Base with underscript and overscript.

3.4.2.2 Algorithm for stretching operators along the inline axis

The ***algorithm for stretching operators along the inline axis*** is as follows.

1. If there is an inline stretch size constraint or block stretch size constraint then the element being laid out is an embellished operator. Lay out the base with the same stretch size constraint.
2. Split the list of in-flow children that have not been laid out yet into a first list $L_{\texttt{ToStretch}}$ containing embellished operators with a stretchy property and inline stretch axis; and a second list $L_{\texttt{NotToStretch}}$.
3. Perform layout without any stretch size constraint on all the items of $L_{\texttt{NotToStretch}}$. If

$L_{ToStretch}$ is empty then stop. If $L_{NotToStretch}$ is empty, perform layout with inline stretch size constraint 0 for all the items of $L_{ToStretch}$.

4. Calculate the target size T to the maximum inline size of the margin boxes of child boxes that have been laid out in the previous step.
5. Lay out or relayout all the elements of $L_{ToStretch}$ with inline stretch size constraint T.

3.4.2.3 Base with underscript

The <munder> element is laid out as shown on Figure 20. LargeOpItalicCorrection is the italic correction of the munder base if it is an embellished operator with the largeop property and 0 otherwise.

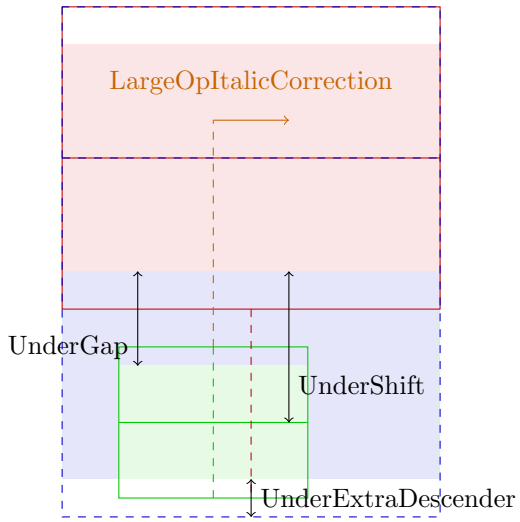


Figure 20 Box model for the <munder> element

The min-content inline size (respectively max-content inline size) of the math content are calculated like the inline size of the math content below but replacing the inline sizes of the munder base's margin box and munder underscript's margin box with the min-content inline size (respectively max-content inline size) of the munder base's margin box and munder underscript's margin box.

The in-flow children are laid out using the algorithm for stretching operators along the inline axis.

The inline size of the math content is calculated by determining the absolute difference between:

- The maximum of:

    - Half the inline size of the munder base's margin box.
    - Half the inline size of the munder underscript's margin box – half `LargeOpItalicCorrection`.
- The minimum of:
    - –half the inline size of the munder base's margin box.
    - –half the inline size of the munder underscript's margin box – half `LargeOpItalicCorrection`.

If m is the minimum calculated in the second item above then the inline offset of the munder base is –m – half the inline size of the base's margin box. The inline offset of the munder underscript is –m – half the inline size of the munder underscript's margin box – half `LargeOpItalicCorrection`.

Parameters `UnderShift` and `UnderExtraDescender` are determined by considering three cases in the following order:

1. The munder base is an embellished operator with the `largeop` property. `UnderShift` is the maximum of
    - LowerLimitBaselineDropMin
    - LowerLimitGapMin + the ascent of the munder underscript.

    `UnderExtraDescender` is 0.
2. The munder base is an embellished operator with the `stretchy` property and stretch axis inline. `UnderShift` is the maximum of:
    - StretchStackBottomShiftDown
    - StretchStackGapAboveMin + the ascent of the munder underscript.

    `UnderExtraDescender` is 0.
3. Otherwise, `UnderShift` is equal to UnderbarVerticalGap if the `accentunder` attribute is not an ASCII case-insensitive match to `true` and to zero otherwise. `UnderExtraAscender` is UnderbarExtraDescender.

The line-ascent of the math content is the maximum between:

- The line-ascent of the munder base's margin box.
- The line-ascent of the munder underscript's margin box – `UnderShift`.

The line-descent of the math content is the maximum between:

- The line-descent of the munder base's margin box.
- The line-descent of the munder underscript's margin box + `UnderShift` + `UnderExtraAscender`.

The alphabetic baseline of the munder base is aligned with the alphabetic baseline. The alphabetic baseline of the munder underscript is shifted away from the alphabetic baseline and towards the line-under by a distance equal to the ink line-descent of the munder base's margin box + `UnderShift`.

The math content box is placed within the content box so that their block-start edges are aligned and the middles of these edges are at the same position.

§ *3.4.2.4 Base with overscript*

The `<mover>` element is laid out as shown on Figure 21. `LargeOpItalicCorrection` is the italic correction of the mover base if it is an embellished operator with the `largeop` property and 0 otherwise.

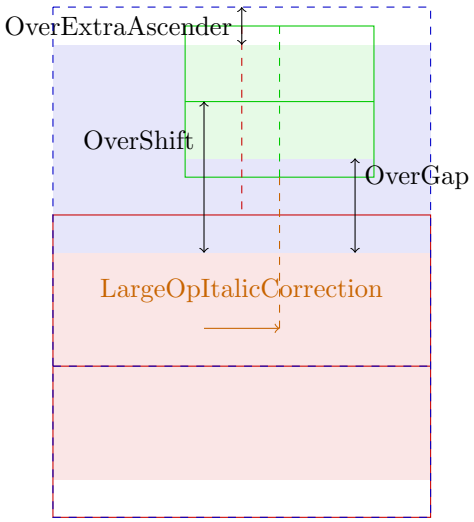


Figure 21 Box model for the `<mover>` element

The min-content inline size (respectively max-content inline size) of the math content are calculated like the inline size of the math content below but replacing the inline sizes of the mover base's margin box and mover overscript's margin box with the min-content inline size (respectively max-content inline size) of the mover base's margin box and mover overscript's margin box.

The in-flow children are laid out using the algorithm for stretching operators along the inline axis.

The `TopAccentAttachment` is the top accent attachment of the mover overscript or half the inline size of the mover overscript's margin box if it is undefined.

The inline size of the math content is calculated by applying the algorithm for stretching operators along the inline axis for layout and determining the absolute difference between:

- The maximum of:
  - Half the inline size of the mover base's margin box.
  - The inline size of the mover overscript's margin box – `TopAccentAttachment` + half `LargeOpItalicCorrection`.
- The minimum of:
  - –half the inline size of the mover base's margin box.
  - –`TopAccentAttachment` + half `LargeOpItalicCorrection`.

If m is the minimum calculated in the second item above then the inline offset of the mover base is –m – half the inline size of the base's margin. The inline offset of the mover overscript is –m – half the inline size of the mover overscript's margin box + half `LargeOpItalicCorrection`.

Parameters `OverShift` and `OverExtraDescender` are determined by considering three cases in the following order:

1. The mover base is an embellished operator with the `largeop` property. `OverShift` is the maximum of
   - UpperLimitBaselineRiseMin
   - UpperLimitGapMin + the descent of the mover overscript.

   `OverExtraAscender` is 0.
2. The mover base is an embellished operator with the `stretchy` property and stretch axis inline. `OverShift` is the maximum of:
   - StretchStackTopShiftUp
   - StretchStackGapBelowMin + the descent of the mover overscript.

   `OverExtraDescender` is 0.
3. Otherwise, `OverShift` is equal to

1. OverbarVerticalGap if the `accent` attribute is not an ASCII case-insensitive match to `true`.
2. Or AccentBaseHeight minus the line-ascent of the mover base's margin box if this difference is nonnegative.
3. Or 0 otherwise.

`OverExtraAscender` is OverbarExtraAscender.

> NOTE
> For accent overscripts and bases with line-ascents that are at most AccentBaseHeight, the rule from [*OPEN-FONT-FORMAT*] [*TEXBOOK*] is actually to align the alphabetic baselines of the overscripts and of the bases. This assumes that accent glyphs are designed in such a way that their ink bottoms are more or less AccentBaseHeight above their alphabetic baselines. Hence, the previous rule will guarantee that all the overscript bottoms are aligned while still avoiding collision with the bases. However, MathML can have arbitrary accent overscripts, so a more general and simpler rule is provided above: Ensure that the bottom of overscript is at least AccentBaseHeight above the alphabetic baseline of the base.

The line-ascent of the math content is the maximum between:

- The line-ascent of the mover base's margin box.
- The line-ascent of the mover overscript's margin box + `OverShift` + `OverExtraAscender`.

The line-descent of the math content is the maximum between:

- The line-descent of the mover base's margin box.
- The line-descent of the mover overscript's margin box – `OverShift`.

The alphabetic baseline of the mover base is aligned with the alphabetic baseline. The alphabetic baseline of the mover overscript is shifted away from the alphabetic baseline and towards the line-over by a distance equal to the ink line-ascent of the base + `OverShift`.

The math content box is placed within the content box so that their block-start edges are aligned and the middles of these edges are at the same position.

§ *3.4.2.5 Base with underscript and overscript*

The general layout of `<munderover>` is shown on Figure 22. The `LargeOpItalicCorrection`,

`UnderShift`, `UnderExtraDescender`, `OverShift`, `OverExtraDescender` parameters are calculated the same as in 3.4.2.3 Base with underscript and 3.4.2.4 Base with overscript.

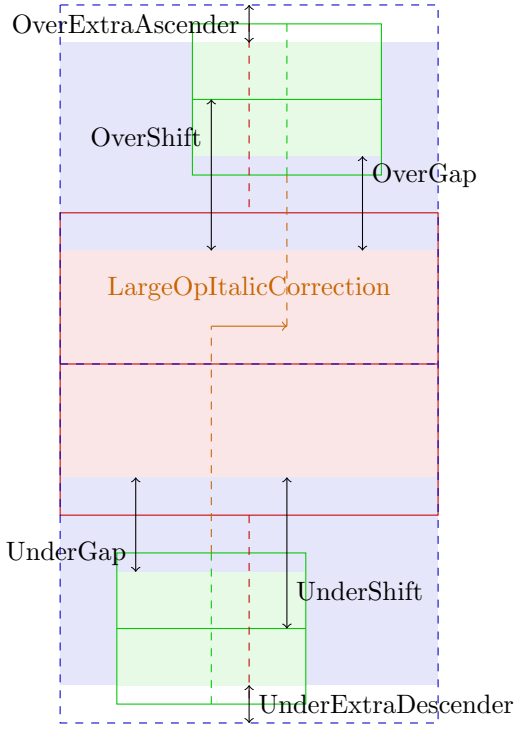


Figure 22 Box model for the `<munderover>` element

The min-content inline size, max-content inline size and inline size of the math content are calculated as an absolute difference between a maximum inline offset and minimum inline offset. These extrema are calculated by taking the extremum value of the corresponding extrema calculated in 3.4.2.3 Base with underscript and 3.4.2.4 Base with overscript. The inline offsets of the munderover base, munderover underscript and munderover overscript are calculated as in these sections but using the new minimum m (minimum of the corresponding minima).

Like in these sections, the in-flow children are laid out using the algorithm for stretching operators along the inline axis.

The line-ascent and line-descent of the math content are also calculated by taking the extremum value of the extrema calculated in 3.4.2.3 Base with underscript and 3.4.2.4 Base with overscript.

Finally, the alphabetic baselines of the munderover base, munderover underscript and munderover overscript are calculated as in sections 3.4.2.3 Base with underscript and 3.4.2.4 Base with overscript.

The math content box is placed within the content box so that their block-start edges are aligned and the middles of these edges are at the same position.

NOTE

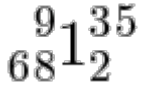
When the underscript (respectively overscript) is an empty box, the base and overscript (respectively underscript) are laid out similarly to 3.4.2.4 Base with overscript (respectively 3.4.2.3 Base with underscript) but the position of the empty underscript (respectively overscript) may add extra space. In order to keep the algorithm simple, no attempt is made to handle empty scripts in a special way.

3.4.3 Prescripts and Tensor Indices <mmultiscripts>

Presubscripts and tensor notations are represented by the ***mmultiscripts*** element. The ***mprescripts*** element is used as a separator between the postscripts and prescripts. These two elements accept the attributes described in 2.1.3 Global Attributes.

The following example shows basic use of prescripts and postscripts, involving a `mprescripts`. Empty `mrow` elements are used at positions where no scripts are rendered. The font-size is automatically scaled down within the scripts.

```
<math>
  <mmultiscripts>
    <mn>1</mn>
    <mn>2</mn>
    <mn>3</mn>
    <mrow></mrow>
    <mn>5</mn>
    <mprescripts/>
    <mn>6</mn>
    <mrow></mrow>
    <mn>8</mn>
    <mn>9</mn>
  </mmultiscripts>
</math>
```



If the <mmultiscripts> or <mprescripts> elements do not have their computed `display`

property equal to `block math` or `inline math` then they are laid out according to the CSS specification where the corresponding value is described. Otherwise, the layout below is performed.

The `<mprescripts>` element is laid out as an `mrow` element.

A valid `<mmultiscripts>` element contains the following in-flow children:

- A first in-flow child, called the ***mmultiscripts base***, that is not an `mprescripts` element.
- Followed by an even number of in-flow children called ***mmultiscripts postscripts***, none of them being a `mprescripts` element. These scripts form a (possibly empty) list subscript, superscript, subscript, superscript, subscript, superscript, etc. Each consecutive couple of children subscript, superscript is called a ***subscript/superscript pair***.
- Optionally followed by an `mprescripts` element and an even number of in-flow children called ***mmultiscripts prescripts***, none of them being a `mprescripts` element. These scripts form a (possibly empty) list of subscript/superscript pair.

If an `<mmultiscripts>` element is not valid then it is laid out the same as the `mrow` element. Otherwise the layout algorithm is performed as in 3.4.3.1 Base with prescripts and postscripts.

§ *3.4.3.1 Base with prescripts and postscripts*

The `<mmultiscripts>` element is laid out as shown on Figure 23. For each subscript/superscript pair of mmultiscripts postscripts, the `ItalicCorrection` `LargeOpItalicCorrection` are defined as in 3.4.1.2 Base with subscript and 3.4.1.3 Base with superscript.

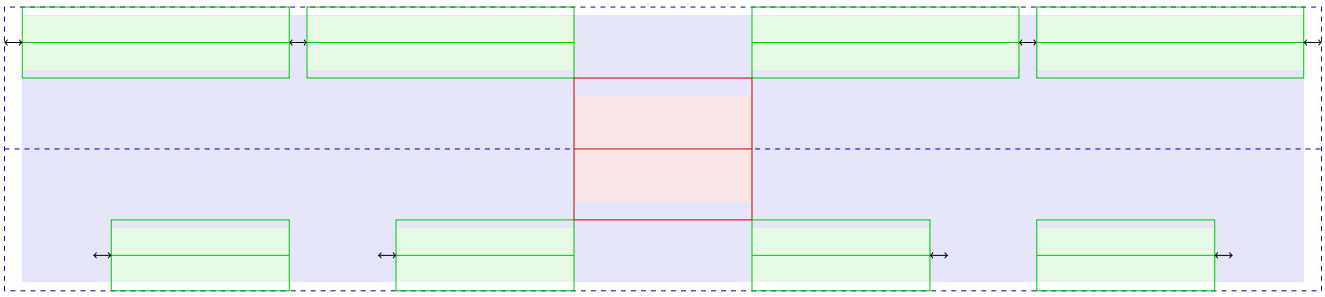


Figure 23 Box model for the `<mmultiscripts>` element

The min-content inline size (respectively max-content inline size) of the math content is calculated the same as the inline size of the math content below, but replacing "inline size" with "min-content inline

size" (respectively "max-content inline size") for the mmultiscripts base's margin box and scripts' margin boxes.

If there is an inline stretch size constraint or a block stretch size constraint the mmultiscripts base is also laid out with the same stretch size constraint. Otherwise it is laid out without any stretch size constraint. The other elements are always laid out without any stretch size constraint.

The inline size of the math content is calculated with the following algorithm:

1. Set `inline-offset` to 0.
2. For each subscript/superscript pair of mmultiscripts prescripts, increment `inline-offset` by SpaceAfterScript + the maximum of
   - The inline size of the subscript's margin box.
   - The inline size of the superscript's margin box.
3. Increment `inline-offset` by the inline size of the mmultiscripts base's margin box and set `inline-size` to `inline-offset`.
4. For each subscript/superscript pair of mmultiscripts postscripts, modify `inline-size` to be at least:
   - x + the inline size of the subscript's margin box + `LargeOpItalicCorrection`.
   - x + the inline size of the superscript's margin box − `ItalicCorrection`.

   Increment `inline-offset` to the maximum of:
   - the inline size of the subscript's margin box.
   - the inline size of the superscript's margin box.

   Increment `inline-offset` by SpaceAfterScript.
5. Return `inline-size`.

`SubShift` (respectively `SuperShift`) is calculated by taking the maximum of all subshifts (respectively supershifts) of each subscript/superscript pair as described in 3.4.1.4 Base with subscript and superscript.

The line-ascent of the math content is calculated by taking the maximum of all the line-ascent of each subscript/superscript pair as described in 3.4.1.4 Base with subscript and superscript but using the `SubShift` and `SuperShift` values calculated above.

The line-descent of the math content is calculated by taking the maximum of all the line-descent of each subscript/superscript pair as described in 3.4.1.4 Base with subscript and superscript but using the `SubShift` and `SuperShift` values calculated above.

Finally, the placement of the in-flow children is performed using the following algorithm:

1. Set `inline-offset` to 0.
2. For each subscript/superscript pair of mmultiscripts prescripts:

    1. Increment `inline-offset` by SpaceAfterScript.
    2. Set `pair-inline-size` to the maximum of
        - the inline size of the subscript's margin box.
        - the inline size of the superscript's margin box.
    3. Place the subscript at inline-start position `inline-offset` + `pair-inline-size` – the inline size of the subscript's margin box.
    4. Place the superscript at inline-start position `inline-offset` + `pair-inline-size` – the inline size of the superscript's margin box.
    5. Place the subscript (respectively superscript) so its alphabetic baseline is shifted away from the alphabetic baseline by `SubShift` (respectively `SuperShift`) towards the line-under (respectively line-over).
    6. Increment `inline-offset` by `pair-inline-size`.
3. Place the mmultiscripts base and `<mprescripts>` boxes at inline offsets `inline-offset` and with their alphabetic baselines aligned with the alphabetic baseline.
4. For each subscript/superscript pair of mmultiscripts postscripts:

    1. Set `pair-inline-size` to the maximum of
        - the inline size of the subscript's margin box.
        - the inline size of the superscript's margin box.
    2. Place the subscript at inline-start position `inline-offset` – `LargeOpItalicCorrection`.
    3. Place the superscript at inline-start position `inline-offset` + `ItalicCorrection`.
    4. Place the subscript (superscript) so its alphabetic baseline is shifted away from the alphabetic baseline by `SubShift` (respectively `SuperShift`) towards the line-under (respectively line-over).
    5. Increment `inline-offset` by `pair-inline-size`.

6. Increment `inline-offset` by SpaceAfterScript.

> NOTE
>
> An <mmultiscripts> with only one subscript/superscript pair of mmultiscripts postscripts is laid out the same as a <msubsup> with the same in-flow children. However, as noticed for <msubsup>, if additionally the subscript (respectively superscript) is an empty box then it is not necessarily laid out the same as an <msub> (respectively <msup>) element. In order to keep the algorithm simple, no attempt is made to handle empty scripts in a special way.

3.4.4 Displaystyle, scriptlevel and math-shift in scripts

For all scripted elements, the rule of thumb is to set `displaystyle` to `false` and to increment `scriptlevel` in all child elements but the first one. However, an `mover` (respectively `munderover`) element with an `accent` attribute that is an ASCII case-insensitive match to `true` does not increment scriptlevel within its second child (respectively third child). Similarly, `mover` and `munderover` elements with an `accentunder` attribute that is an ASCII case-insensitive match to `true` do not increment scriptlevel within their second child.

<mmultiscripts> sets `math-shift` to `compact` on its children at even position if they are before an `mprescripts`, and on those at odd position if they are after an `mprescripts`. The <msub> and <msubsup> elements set `math-shift` to `compact` on their second child. `mover` and `munderover` elements with an `accent` attribute that is an ASCII case-insensitive match to `true` also set `math-shift` to `compact` within their first child.

The A. User Agent Stylesheet must contain the following style in order to implement this behavior:

```
msub > :not(:first-child),
msup > :not(:first-child),
msubsup > :not(:first-child),
mmultiscripts > :not(:first-child),
munder > :not(:first-child),
mover > :not(:first-child),
munderover > :not(:first-child) {
  math-depth: add(1);
  math-style: compact;
}
munder[accentunder="true" i] > :nth-child(2),
```

```
mover[accent="true" i] > :nth-child(2),
munderover[accentunder="true" i] > :nth-child(2),
munderover[accent="true" i] > :nth-child(3) {
  font-size: inherit;
}
msub > :nth-child(2),
msubsup > :nth-child(2),
mmultiscripts > :nth-child(even),
mmultiscripts > mprescripts ~ :nth-child(odd),
mover[accent="true" i] > :first-child,
munderover[accent="true" i] > :first-child {
  math-shift: compact;
}
mmultiscripts > mprescripts ~ :nth-child(even) {
  math-shift: inherit;
}
```

NOTE

In practice, all the children of the MathML elements described in this section are in-flow and the `<mprescripts>` is empty. Hence the CSS rules essentially perform automatic `displaystyle` and `scriptlevel` changes for the scripts; and `math-shift` changes for subscripts and sometimes the base.

§ 3.5 Tabular Math

Matrices, arrays and other table-like mathematical notation are marked up using `mtable` `mtr` `mtd` elements. These elements are similar to the `table`, `tr` and `td` elements of [*HTML*].

The following example shows how tabular layout allows to write a matrix. Note that it is vertically centered with the fraction bar and the middle of the equal sign.

```
<math>
  <mfrac>
    <mi>A</mi>
    <mn>2</mn>
  </mfrac>
  <mo>=</mo>
  <mrow>
    <mo>(</mo>
    <mtable>
      <mtr>
        <mtd><mn>1</mn></mtd>
        <mtd><mn>2</mn></mtd>
        <mtd><mn>3</mn></mtd>
      </mtr>
      <mtr>
        <mtd><mn>4</mn></mtd>
        <mtd><mn>5</mn></mtd>
        <mtd><mn>6</mn></mtd>
      </mtr>
      <mtr>
        <mtd><mn>7</mn></mtd>
        <mtd><mn>8</mn></mtd>
        <mtd><mn>9</mn></mtd>
      </mtr>
    </mtable>
    <mo>)</mo>
  </mrow>
</math>
```

$$\frac{A}{2} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

3.5.1 Table or Matrix <mtable>

The ***mtable*** is laid out as an `inline-table` and sets `displaystyle` to `false`. The user agent stylesheet must contain the following rules in order to implement these properties:

```
mtable {
  display: inline-table;
  math-style: compact;
}
```

The `mtable` element is as a CSS table and the min-content inline size, max-content inline size, inline size, block size, first baseline set and last baseline set sets are determined accordingly. The center of the table is aligned with the math axis.

The <mtable> accepts the attributes described in 2.1.3 Global Attributes.

3.5.2 Row in Table or Matrix <mtr>

The ***mtr*** is laid out as `table-row`. The user agent stylesheet must contain the following rules in order to implement that behavior:

```
mtr {
  display: table-row;
}
```

The <mtr> accepts the attributes described in 2.1.3 Global Attributes.

3.5.3 Entry in Table or Matrix <mtd>

The ***mtd*** is laid out as a `table-cell` with content centered in the cell and a default padding. The user agent stylesheet must contain the following rules:

```
mtd {
  display: table-cell;
```

```
  /* Centering inside table cells should rely on box alignment properties.
     See https://github.com/w3c/mathml-core/issues/156 */
  text-align: center;
  padding: 0.5ex 0.4em;
}
```

The <mtd> accepts the attributes described in 2.1.3 Global Attributes as well as the following attributes:

- columnspan
- rowspan

The ***columnspan*** (respectively ***rowspan***) attribute has the same syntax and semantics as the colspan (respectively rowspan) attribute on the <td> element from [*HTML*]. In particular, the parsing of these attributes is handled as described in the algorithm for processing rows, always reading "colspan" as "columnspan".

> NOTE
> The name of the column spanning attribute in [*MathML3*] and earlier versions is columnspan and is preserved for backward compatibility reasons.

The <mtd> element generates an anonymous <mrow> box.

3.6 Enlivening Expressions

Historically, the ***maction*** element provides a mechanism for binding actions to expressions.

The <maction> element accepts the attributes described in 2.1.3 Global Attributes as well as the following attributes:

- actiontype
- selection

This specification does not define any observable behavior that is specific to the ***actiontype*** and ***selection*** attributes.

The following example shows the "toggle" action type from [*MathML3*] where the renderer alternately displays the selected subexpression, starting from "one third" and cycling through them when there is a click on the selected subexpression ("one quarter", "one half", "one third", etc). This is not part of MathML Core but can be implemented using JavaScript and CSS polyfills. The default behavior is just to render the first child.

```
<math>
  <maction actiontype="toggle" selection="2">
    <mfrac>
      <mn>1</mn>
      <mn>2</mn>
    </mfrac>
    <mfrac>
      <mn>1</mn>
      <mn>3</mn>
    </mfrac>
    <mfrac>
      <mn>1</mn>
      <mn>4</mn>
    </mfrac>
  </maction>
</math>
```

$$\frac{1}{2}$$

The layout algorithm of the `<maction>` element is the same as the `<mrow>` element. The user agent stylesheet must contain the following rules in order to hide all but its first child element, which is the default behavior for the legacy actiontype values:

```
maction > :not(:first-child) {
  display: none;
}
```

NOTE

`<maction>` is implemented for compatibility with full MathML. Authors whose only target is MathML Core are encouraged to use other HTML, CSS and JavaScript mechanisms to implement custom actions. They may rely on maction attributes defined in [*MathML3*].

§ 3.7 Semantics and Presentation

The ***semantics*** element is the container element that associates annotations with a MathML expression. Typically, the `<semantics>` element has as its first child element a MathML expression to be annotated while subsequent child elements represent text annotations within an ***annotation*** element, or more complex markup annotations within an ***annotation-xml*** element.

The following example shows how the fraction "one half" can be annotated with a textual annotation (LaTeX) or an XML annotation (content MathML), which are not intended to be rendered by the user agent. This fraction is also annotated with equivalent SVG and HTML markup.

```
<math>
  <semantics>
    <mfrac>
      <mn>1</mn>
      <mn>2</mn>
    </mfrac>
    <annotation encoding="application/x-tex">\frac{1}{2}</annotation>
    <annotation-xml encoding="application/mathml-content+xml">
      <apply>
        <divide/>
         <cn>1</cn>
         <cn>2</cn>
      </apply>
    </annotation-xml>
    <annotation-xml>
      <svg width="25" height="75" xmlns="http://www.w3.org/2000/svg">
        <path stroke-width="5.8743"
              d="m5.9157 27.415h6.601v-22.783l-7.1813 1.4402v-3.6805l7.14
                 -1.4402h4.0406v26.464h6.601v3.4005h-17.203z"/>
        <path stroke="#000000" stroke-width="2.3409"
              d="m0.83496 39.228h23.327"/>
        <path stroke-width="5.8743"
              d="m8.696 70.638h14.102v3.4005h-18.963v-3.4005q2.3004-2.38(
                 6.2608-6.3813 3.9806-4.0206 5.0007-5.1808 1.9403-2.1803
                 2.7004-3.6805 0.78011-1.5202 0.78011-2.9804 0-2.3804
                 -1.6802-3.8806-1.6603-1.5002-4.3406-1.5002-1.9003 0-4.0(
                 0.6601-2.1003 0.6601-4.5007 2.0003v-4.0806q2.4404-0.980(
                 4.5607-1.4802 2.1203-0.50007 3.8806-0.50007 4.6407 0 7.4
                 2.3203 2.7604 2.3203 2.7604 6.2009 0 1.8403-0.7001 3.50(
                 -0.68013 1.6402-2.5004 3.8806-0.50007 0.58009-3.1805 3.3
                 -2.6804 2.7604-7.5614 7.7412z"/>
      </svg>
    </annotation-xml>
    <annotation-xml encoding="application/xhtml+xml">
      <div style="display: inline-flex;
                  flex-direction: column; align-items: center;">
```

```
        <div>1</div>
        <div>—</div>
        <div>2</div>
      </div>
    </annotation-xml>
  </semantics>
</math>
```

$$\frac{1}{2}$$

The `<semantics>` element accepts the attributes described in 2.1.3 Global Attributes. Its layout algorithm is the same as the `mrow` element. The user agent stylesheet must contain the following rule in order to only render the annotated MathML expression:

```
semantics > :not(:first-child) {
  display: none;
}
```

The `<annotation-xml>` and `<annotation>` element accepts the attributes described in 2.1.3 Global Attributes as well as the following attribute:

- `encoding`

This specification does not define any observable behavior that is specific to the ***encoding*** attribute.

The layout algorithm of the `<annotation-xml>` and `<annotation>` element is the same as the `mtext` element.

NOTE

Authors can use the `encoding` attribute to distinguish annotations for HTML integration point, clipboard copy, alternative rendering, etc. In particular, CSS can be used to render alternative annotations, e.g.

```
/* Hide the annotated child. */
semantics > :first-child { display: none; }
 /* Show all text annotations. */
semantics > annotation { display: inline; }
/* Show all HTML annotations. */
semantics > annotation-xml[encoding="text/html" i],
semantics > annotation-xml[encoding="application/xhtml+xml" i] {
  display: inline-block;
}
```

4. CSS Extensions for Math Layout

4.1 The `display: block math` and `display: inline math` value

The `display` property from *CSS Display Module Level 3* is extended with a new inner display type:

Name:	display
New values:	<display-outside> || [<display-inside> | math]

For elements that are not MathML elements, if the specified value of `display` is `block math` or `inline math` then the computed value is `block flow` and `inline flow` respectively. For the `mtable` element the computed value is `block table` and `inline table` respectively. For the `mtr` element, the computed value is `table-row`. For the `mtd` element, the computed value is `table-cell`.

MathML elements with a computed `display` value equal to `block math` or `inline math` control box generation and layout according to their tag name, as described in the relevant sections. Unknown

MathML elements behave the same as the `mrow` element.

NOTE

The `display: block math` and `display: inline math` values provide a default layout for MathML elements while at the same time allowing to override it with either native display values or custom values. This allows authors or polyfills to define their own custom notations to tweak or extend MathML Core.

In the following example, the default layout of the MathML `mrow` element is overridden to render its content as a grid.

```
<math>
  <msup>
    <mrow>
      <mo symmetric="false">[</mo>
      <mrow style="display: block; width: 4.5em;">
        <mrow style="display: grid;
                     grid-template-columns: 1.5em 1.5em 1.5em;
                     grid-template-rows: 1.5em 1.5em;
                     justify-items: center;
                     align-items: center;">
          <mn>12</mn>
          <mn>34</mn>
          <mn>56</mn>
          <mn>7</mn>
          <mn>8</mn>
          <mn>9</mn>
        </mrow>
      </mrow>
      <mo symmetric="false">]</mo>
    </mrow>
    <mi>α</mi>
  </msup>
</math>
```

$$\begin{bmatrix} 12 & 34 & 56 \\ 7 & 8 & 9 \end{bmatrix}^{\alpha}$$

4.2 The `math-auto` transform

The text-transform property from *CSS Text Module Level 4* has a new value `math-auto`. On text nodes containing a single character, if the computed value is `math-auto` and the character is present in the "Original" column of C.1 `italic` mappings then it is converted to the corresponding character from the "italic" column.

> A common style convention is to render identifiers with multiple letters (e.g. the function name "exp") with normal style and identifiers with a single letter (e.g. the variable "n") with italic style. The `math-auto` property is intended to implement this default behavior, which can be overridden by authors if necessary. Note that mathematical fonts are designed with a special kind of italic glyphs located at the Unicode positions of C.1 `italic` mappings, which differ from the shaping obtained via italic font style. Compare this mathematical formula rendered with the Latin Modern Math font using `font-style: italic` (left) and `text-transform: math-auto` (right):
>
> $$x^n + y^n = z^n \text{ VS } x^n + y^n = z^n$$

4.3 The `math-style` property

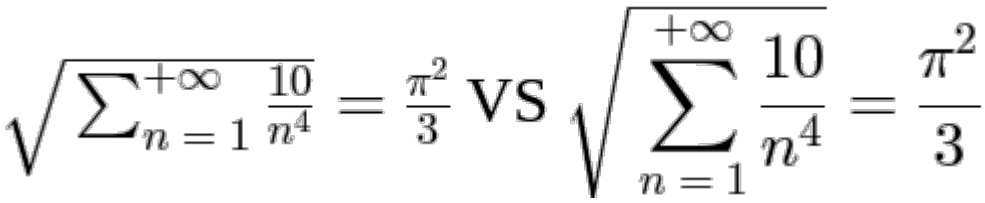
Name:	***math-style***
Value:	normal | compact
Initial:	normal
Applies to:	All elements
Inherited:	yes
Percentages:	n/a
Computed value:	specified keyword

Canonical order:	n/a
Animation type:	not animatable
Media:	visual

When `math-style` is `compact`, the math layout on descendants tries to minimize the logical height by applying the following rules:

- The font-size is scaled down when its specified value is `math` and the computed value of `math-depth` is `auto-add` (default for `mfrac`) as described in 4.5 The `math-depth` property.
- Operators with the `largeop` property do not follow rules from 3.2.4.3 Layout of operators to make them bigger.
- Under-/overscripts attached to an operator with the `movablelimits` property are actually drawn as sub-/superscripts as described in 3.4.2.1 Children of `<munder>`, `<mover>`, `<munderover>`.
- Smaller vertical gaps and shifts from the OpenType MATH table are used for fractions and radicals, as described in 3.3.2.1 Fraction with nonzero line thickness, 3.3.2.2 Fraction with zero line thickness and 3.3.3.1 Radical symbol.

The following example shows a mathematical formula rendered with its `math` root styled with `math-style: compact` (left) and `math-style: normal` (right). In the former case, the font-size is automatically scaled down within the fractions and the summation limits are rendered as subscript and superscript of the ∑. In the latter case, the ∑ is drawn bigger than normal text and vertical gaps within fractions (even relative to current font-size) are larger.



These two `math-style` values typically correspond to mathematical expressions in inline and display mode respectively [*TeXBook*]. A mathematical formula in display mode may automatically switch to inline mode within some subformulas (e.g. scripts, matrix elements, numerators and denominators, etc) and it is sometimes desirable to override this default behavior. The math-style property allows to easily implement these features for MathML in the user agent stylesheet and with the displaystyle attribute; and also exposes them to polyfills.

§ 4.4 The `math-shift` property

Name:	***math-shift***
Value:	normal | compact
Initial:	normal
Applies to:	All elements
Inherited:	yes
Percentages:	n/a
Computed value:	specified keyword

Canonical order:	n/a
Animation type:	not animatable
Media:	visual

If the value of `math-shift` is `compact`, the math layout on descendants will use the superscriptShiftUpCramped parameter to place superscript. If the value of `math-shift` is `normal`, the math will use the superscriptShiftUp parameter instead.

This property is used for positioning superscript during the layout of MathML scripted elements. See § 3.4.1 Subscripts and Superscripts `<msub>`, `<msup>`, `<msubsup>`, 3.4.3 Prescripts and Tensor Indices `<mmultiscripts>` and 3.4.2 Underscripts and Overscripts `<munder>`, `<mover>`, `<munderover>`.

In the following example, the two "x squared" are rendered with compact math-style and the same `font-size`. However, the one within the square root is rendered with compact `math-shift` while the other one is rendered with normal `math-shift`, leading to subtle different shift of the superscript "2".

$$\sqrt{x^2} \neq x^2$$

Per [*TeXBook*], a mathematical formula uses normal style by default but may switch to compact style ("cramped" in TeX's terminology) within some subformulas (e.g. radicals, fraction denominators, etc). The math-shift property allows to easily implement these rules for MathML in the user agent stylesheet. Page authors or developers of polyfills may also benefit from having access to this property to tweak or refine the default implementation.

§ 4.5 The `math-depth` property

A new math-depth property is introduced to describe a notion of "depth" for each element of a mathematical formula, with respect to the top-level container of that formula. Concretely, this is used

to determine the computed value of the font-size property when its specified value is `math`.

Name:	***math-depth***
Value:	auto-add | add(<integer>) | <integer>
Initial:	0
Applies to:	All elements
Inherited:	yes
Percentages:	n/a
Computed value:	an integer, see below
Canonical order:	n/a
Animation type:	not animatable
Media:	visual

The computed value of the math-depth value is determined as follows:

- If the specified value of math-depth is `auto-add` and the inherited value of math-style is `compact` then the computed value of math-depth of the element is its inherited value plus one.
- If the specified value of math-depth is of the form `add(<integer>)` then the computed value of math-depth of the element is its inherited value plus the specified integer.
- If the specified value of math-depth is of the form `<integer>` then the computed value of math-depth of the element is the specified integer.
- Otherwise, the computed value of math-depth of the element is the inherited one.

If the specified value of font-size is `math` then the computed value of font-size is obtained by multiplying the inherited value of font-size by a nonzero scale factor calculated by the following procedure:

1. Let A be the inherited math-depth value, B the computed math-depth value, C be 0.71 and S be

1.0

2. ◦ If A = B then return S.
   ◦ If B < A, swap A and B and set `InvertScaleFactor` to true.
   ◦ Otherwise B > A and set `InvertScaleFactor` to false.
3. Let E be B - A > 0.
4. If the inherited first available font has an OpenType MATH table:
   ◦ If A ≤ 0 and B ≥ 2 then multiply S by scriptScriptPercentScaleDown and decrement E by 2.
   ◦ Otherwise if A = 1 then multiply S by scriptScriptPercentScaleDown / scriptPercentScaleDown and decrement E by 1.
   ◦ Otherwise if B = 1 then multiply S by scriptPercentScaleDown and decrement E by 1.
5. Multiply S by C^E.
6. Return S if `InvertScaleFactor` is false and 1/S otherwise.

The following example shows a mathematical formula with normal math-style rendered with the Latin Modern Math font. When entering subexpressions like scripts or fractions, the font-size is automatically scaled down according to the values of MATH table contained in that font. Note that font-size is scaled down when entering the superscripts but even faster when entering a root's prescript. Also it is scaled down when entering the inner fraction but not when entering the outer one, due to automatic change of math-style in fractions.

$$A^{A^A} + \sqrt[A]{A} + \frac{A + \frac{A}{A}}{A}$$

These rules from [*TeXBook*] are subtle and it's worth having a separate `math-depth` mechanism to express and handle them. They can be implemented in MathML using the user agent stylesheet. Page authors or developers of polyfills may also benefit from having access to this property to tweak or refine the default implementation. In particular, the scriptlevel attribute from MathML provides a way to perform `math-depth` changes.

§ 5. OpenType `MATH` table

This chapter describes features provided by `MATH` table of an OpenType font [*OPEN-FONT-FORMAT*]. Throughout this chapter, a C-like notation `Table.Subtable1[index].Subtable2.Parameter` is used to denote OpenType parameters. Such parameters may not be available (e.g. if the font lacks one of the subtable, has an invalid offset, etc) and so fallback options are provided.

> NOTE
> It is strongly encouraged to render MathML with a math font with the proper OpenType features. There is no guarantee that the fallback options provided will provide good enough rendering.

OpenType values expressed in design units (perhaps indirectly via a `MathValueRecord` entry) are scaled to appropriate values for layout purpose, taking into account `head.unitsPerEm`, CSS font-size or zoom level.

5.1 Layout constants (`MathConstants`)

These are global layout constants for the first available font:

Default fallback constant
: 0

Default rule thickness
: `post.underlineThickness` or Default fallback constant if the constant is not available.

scriptPercentScaleDown
: `MATH.MathConstants.scriptPercentScaleDown / 100` or 0.71 if `MATH.MathConstants.scriptPercentScaleDown` is null or not available.

scriptScriptPercentScaleDown
: `MATH.MathConstants.scriptScriptPercentScaleDown / 100` or 0.5041 if `MATH.MathConstants.scriptScriptPercentScaleDown` is null or not available.

displayOperatorMinHeight
: `MATH.MathConstants.displayOperatorMinHeight` or Default fallback constant if the constant is not available.

axisHeight
: `MATH.MathConstants.axisHeight` or half `OS/2.sxHeight` if the constant is not available.

accentBaseHeight
: `MATH.MathConstants.accentBaseHeight` or `OS/2.sxHeight` if the constant is not available.

subscriptShiftDown
: `MATH.MathConstants.subscriptShiftDown` or `OS/2.ySubscriptYOffset` if the constant is not available.

subscriptTopMax
: `MATH.MathConstants.subscriptTopMax` or ⅘ × `OS/2.sxHeight` if the constant is not available.

subscriptBaselineDropMin
: `MATH.MathConstants.subscriptBaselineDropMin` or Default fallback constant if the constant is not available.

superscriptShiftUp
: `MATH.MathConstants.superscriptShiftUp` or `OS/2.ySuperscriptYOffset` if the constant is not available.

superscriptShiftUpCramped
: `MATH.MathConstants.superscriptShiftUpCramped` or Default fallback constant if the constant is not available.

superscriptBottomMin
: `MATH.MathConstants.superscriptBottomMin` or ¼ × `OS/2.sxHeight` if the constant is not available.

superscriptBaselineDropMax
: `MATH.MathConstants.superscriptBaselineDropMax` or Default fallback constant if the constant is not available.

subSuperscriptGapMin
: `MATH.MathConstants.subSuperscriptGapMin` or 4 × default rule thickness if the constant is not available.

superscriptBottomMaxWithSubscript
: `MATH.MathConstants.superscriptBottomMaxWithSubscript` or ⅘ × `OS/2.sxHeight` if the constant is not available.

spaceAfterScript
: `MATH.MathConstants.spaceAfterScript` or 1/24em if the constant is not available.

upperLimitGapMin
: `MATH.MathConstants.upperLimitGapMin` or Default fallback constant if the constant is not available.

upperLimitBaselineRiseMin
: `MATH.MathConstants.upperLimitBaselineRiseMin` or Default fallback constant if the constant is not available.

lowerLimitGapMin
: `MATH.MathConstants.lowerLimitGapMin` or Default fallback constant if the constant is not

available.

lowerLimitBaselineDropMin
: `MATH.MathConstants.lowerLimitBaselineDropMin` or Default fallback constant if the constant is not available.

stackTopShiftUp
: `MATH.MathConstants.stackTopShiftUp` or Default fallback constant if the constant is not available.

stackTopDisplayStyleShiftUp
: `MATH.MathConstants.stackTopDisplayStyleShiftUp` or Default fallback constant if the constant is not available.

stackBottomShiftDown
: `MATH.MathConstants.stackBottomShiftDown` or Default fallback constant if the constant is not available.

stackBottomDisplayStyleShiftDown
: `MATH.MathConstants.stackBottomDisplayStyleShiftDown` or Default fallback constant if the constant is not available.

stackGapMin
: `MATH.MathConstants.stackGapMin` or 3 × default rule thickness if the constant is not available.

stackDisplayStyleGapMin
: `MATH.MathConstants.stackDisplayStyleGapMin` or 7 × default rule thickness if the constant is not available.

stretchStackTopShiftUp
: `MATH.MathConstants.stretchStackTopShiftUp` or Default fallback constant if the constant is not available.

stretchStackBottomShiftDown
: `MATH.MathConstants.stretchStackBottomShiftDown` or Default fallback constant if the constant is not available.

stretchStackGapAboveMin
: `MATH.MathConstants.stretchStackGapAboveMin` or Default fallback constant if the constant is not available.

stretchStackGapBelowMin
: `MATH.MathConstants.stretchStackGapBelowMin` or Default fallback constant if the constant is not available.

fractionNumeratorShiftUp
: `MATH.MathConstants.fractionNumeratorShiftUp` or Default fallback constant if the constant is not available.

fractionNumeratorDisplayStyleShiftUp
: `MATH.MathConstants.fractionNumeratorDisplayStyleShiftUp` or Default fallback constant if the constant is not available.

fractionDenominatorShiftDown
: `MATH.MathConstants.fractionDenominatorShiftDown` or Default fallback constant if the constant is not available.

fractionDenominatorDisplayStyleShiftDown
: `MATH.MathConstants.fractionDenominatorDisplayStyleShiftDown` or Default fallback constant if the constant is not available.

fractionNumeratorGapMin
: `MATH.MathConstants.fractionNumeratorGapMin` or default rule thickness if the constant is not available.

fractionNumDisplayStyleGapMin
: `MATH.MathConstants.fractionNumDisplayStyleGapMin` or 3 × default rule thickness if the constant is not available.

fractionRuleThickness
: `MATH.MathConstants.fractionRuleThickness` or default rule thickness if the constant is not available.

fractionDenominatorGapMin
: `MATH.MathConstants.fractionDenominatorGapMin` or default rule thickness if the constant is not available.

fractionDenomDisplayStyleGapMin
: `MATH.MathConstants.fractionDenomDisplayStyleGapMin` or 3 × default rule thickness if the constant is not available.

overbarVerticalGap
: `MATH.MathConstants.overbarVerticalGap` or 3 × default rule thickness if the constant is not available.

overbarExtraAscender
: `MATH.MathConstants.overbarExtraAscender` or default rule thickness if the constant is not available.

underbarVerticalGap
: `MATH.MathConstants.underbarVerticalGap` or 3 × default rule thickness if the constant is not available.

underbarExtraDescender
: `MATH.MathConstants.underbarExtraDescender` or default rule thickness if the constant is not available.

radicalVerticalGap

`MATH.MathConstants.radicalVerticalGap` or 1¼ × default rule thickness if the constant is not available.

radicalDisplayStyleVerticalGap
`MATH.MathConstants.radicalDisplayStyleVerticalGap` or default rule thickness + ¼ `OS/2.sxHeight` if the constant is not available.

radicalRuleThickness
`MATH.MathConstants.radicalRuleThickness` or default rule thickness if the constant is not available.

radicalExtraAscender
`MATH.MathConstants.radicalExtraAscender` or default rule thickness if the constant is not available.

radicalKernBeforeDegree
`MATH.MathConstants.radicalKernBeforeDegree` or 5/18em if the constant is not available.

radicalKernAfterDegree
`MATH.MathConstants.radicalKernAfterDegree` or −10/18em if the constant is not available.

radicalDegreeBottomRaisePercent
`MATH.MathConstants.radicalDegreeBottomRaisePercent / 100.0` or 0.6 if the constant is not available.

5.2 Glyph information (`MathGlyphInfo`)

> NOTE
> MathTopAccentAttachment is at risk.

These are per-glyph tables for the first available font:

MathItalicsCorrectionInfo
The subtable `MATH.MathGlyphInfo.MathItalicsCorrectionInfo` of italics correction values. Use the corresponding value in `MATH.MathGlyphInfo.MathItalicsCorrectionInfo.italicsCorrection` if there is one for the requested glyph or `0` otherwise.

MathTopAccentAttachment
The subtable `MATH.MathGlyphInfo.MathTopAccentAttachment` of positioning top math accents along the inline axis. Use the corresponding value in

`MATH.MathGlyphInfo.MathTopAccentAttachment.topAccentAttachment` if there is one for the requested glyph or half the advance width of the glyph otherwise.

5.3 Size variants for operators (`MathVariants`)

This section describes how to handle stretchy glyphs of arbitrary size using the `MATH.MathVariants` table.

5.3.1 The `GlyphAssembly` table

This section is based on [*OPEN-TYPE-MATH-IN-HARFBUZZ*]. For convenience, the following definitions are used:

- o_{min} is `MATH.MathVariant.minConnectorOverlap`.
- A `GlyphPartRecord` is an *extender* if and only if `GlyphPartRecord.partFlags` has the `fExtender` flag set.
- A `GlyphAssembly` is *horizontal* if it is obtained from `MathVariant.horizGlyphConstructionOffsets`. Otherwise it is *vertical* (and obtained from `MathVariant.vertGlyphConstructionOffsets`).
- For a given `GlyphAssembly` table, N_{Ext} (respectively N_{NonExt}) is the number of extenders (respectively non-extenders) in `GlyphAssembly.partRecords`.
- For a given `GlyphAssembly` table, S_{Ext} (respectively S_{NonExt}) is the sum of `GlyphPartRecord.fullAdvance` for all extenders (respectively non-extenders) in `GlyphAssembly.partRecords`.
- $S_{Ext,NonOverlapping} = S_{Ext} - o_{min}\ N_{Ext}$ is the sum of maximum non overlapping parts of extenders.

User agents must treat the `GlyphAssembly` as invalid if the following conditions are not satisfied:

- $N_{Ext} > 0$. Otherwise, the assembly cannot be grown by repeating extenders.
- $S_{Ext,NonOverlapping} > 0$. Otherwise, the assembly does not grow when joining extenders.
- For each `GlyphPartRecord` in `GlyphAssembly.partRecords`, the values of

`GlyphPartRecord.startConnectorLength` and `GlyphPartRecord.endConnectorLength` must be at least o_{min}. Otherwise, it is not possible to satisfy the condition of `MathVariant.minConnectorOverlap`.

In this specification, a glyph assembly is built by repeating each extender r times and using the same overlap value o between each glyph. The number of glyphs in such an assembly is ***AssemblyGlyphCount***(r) = N_{NonExt} + r N_{Ext} while the stretch size is ***AssembySize***(o, r) = S_{NonExt} + r S_{Ext} − o (AssemblyGlyphCount(r) − 1).

r_{min} is the minimal number of repetitions needed to obtain an assembly of size at least T, i.e. the minimal r such that AssembySize(o_{min}, r) ≥ T. It is defined as the maximum between 0 and the ceiling of ((T − S_{NonExt} + o_{min} (N_{NonExt} − 1)) / $S_{Ext,NonOverlapping}$).

$o_{max,theorical}$ = (AssembySize(0, r_{min}) − T) / (AssemblyGlyphCount(r_{min}) − 1) is the theorical overlap obtained by splitting evenly the extra size of an assembly built with null overlap.

o_{max} is the maximum overlap possible to build an assembly of size at least T by repeating each extender r_{min} times. If AssemblyGlyphCount(r_{min}) ≤ 1, then the actual overlap value is irrelevant. Otherwise, o_{max} is defined to be the minimum of:

- $o_{max,theorical}$.
- `GlyphPartRecord.startConnectorLength` for all the entries in `GlyphAssembly.partRecords`, excluding the last one if it is not an extender.
- `GlyphPartRecord.endConnectorLength` for all the entries in `GlyphAssembly.partRecords`, excluding the first one if it is not an extender.

The ***glyph assembly stretch size*** for a target size T is AssembySize(o_{max}, r_{min}).

The ***glyph assembly width***, ***glyph assembly ascent*** and ***glyph assembly descent*** are defined as follows:

- If `GlyphAssembly` is vertical, the width is the maximum advance width of the glyphs of ID `GlyphPartRecord.glyphID` for all the `GlyphPartRecord` in `GlyphAssembly.partRecords`, the ascent is the glyph assembly stretch size for a given target size `T` and the descent is 0.
- Otherwise, the `GlyphAssembly` is horizontal, the width is glyph assembly stretch size for a given target size `T` while the ascent (respectively descent) is the maximum ascent (respectively descent) of the glyphs of ID `GlyphPartRecord.glyphID` for all the `GlyphPartRecord` in `GlyphAssembly.partRecords`.

The ***glyph assembly height*** is the sum of the glyph assembly ascent and glyph assembly descent.

> NOTE
> The horizontal (respectively vertical) metrics for a vertical (respectively horizontal) glyph assembly do not depend on the target size `T`.

The ***shaping of the glyph assembly*** is performed with the following algorithm:

1. Calculate r_{min} and o_{max}.
2. Set `(x, y)` to `(0, 0)`, `RepetitionCounter` to 0 and `PartIndex` to -1.
3. Repeat the following steps:
    1. If `RepetitionCounter` is 0:
        1. Increment `PartIndex`.
        2. If `PartIndex` is `GlyphAssembly.partCount` then stop.
        3. Otherwise, set `Part` to `GlyphAssembly.partRecords[PartIndex]`. Set `RepetitionCounter` to r_{min} if `Part` is an extender and to 1 otherwise.
    2. - If the glyph assembly is horizontal then draw the glyph of ID `Part.glyphID` so that its (left, baseline) coordinates are at position `(x, y)`. Set `x` to `x + Part.fullAdvance -` o_{max}.
       - Otherwise (if the glyph assembly is vertical), then draw the glyph of id `Part.glyphID` so that its (left, bottom) coordinates are at position `(x, y)`. Set `y` to `y - Part.fullAdvance +` o_{max}.
    3. Decrement `RepetitionCounter`.

5.3.2 Algorithms for glyph stretching

The ***preferred inline size of a glyph stretched along the block axis*** is calculated using the following algorithm:

1. Set `S` to the glyph's advance width.
2. If there is a `MathGlyphConstruction` table in the `MathVariants.vertGlyphConstructionOffsets` table for the given glyph:
    1. For each `MathGlyphVariantRecord` in `MathGlyphConstruction.mathGlyphVariantRecord`, ensure that `S` is at least the

advance width of the glyph of id `MathGlyphVariantRecord.variantGlyph`.

2. If there is valid `GlyphAssembly` subtable, then ensure that `S` is at least the glyph assembly width.

3. Return `S`.

NOTE
The preferred inline size of a glyph stretched along the block axis will return the maximum width of all possible vertical constructions for that glyph. In practice, math fonts are designed so that vertical constructions are almost constant width, so possible over-estimation of the actual width is small.

The algorithm to ***shape a stretchy glyph*** to inline (respectively block) dimension `T` is the following:

1. If there is not any `MathGlyphConstruction` table in the `MathVariants.horizGlyphConstructionOffsets` table (respectively `MathVariants.vertGlyphConstructionOffsets` table) for the given glyph then exit with failure.
2. If the glyph's advance width (respectively height) is at least `T` then use normal shaping and bounding box for that glyph, the MathItalicsCorrectionInfo for that glyph as italic correction and exit with success.
3. Browse the list of `MathGlyphVariantRecord` in `MathGlyphConstruction.mathGlyphVariantRecord`. If one `MathGlyphVariantRecord.advanceMeasurement` is at least `T` then use normal shaping and bounding box for `MathGlyphVariantRecord.variantGlyph`, the MathItalicsCorrectionInfo for that glyph as italic correction and exit with success.
4. If there is valid `GlyphAssembly` subtable then use the bounding box given by glyph assembly width, glyph assembly height, glyph assembly ascent, glyph assembly descent, the value `GlyphAssembly.italicsCorrection` as italic correction, perform shaping of the glyph assembly and exit with success.
5. If none of the stretch options above allowed to cover the target size `T`, then choose last one that was tried and exit with success.

NOTE
If a font does not provide tables for stretchy constructions, User Agents may use their own internal constructions as a fallback such as the one suggested in B.4 Unicode-based Glyph Assemblies.

A. *User Agent Stylesheet*

```
@namespace url(http://www.w3.org/1998/Math/MathML);

/* Universal rules */
* {
  font-size: math;
  display: block math;
  writing-mode: horizontal-tb !important;
}

/* The <math> element */
math {
  direction: ltr;
  text-indent: 0;
  letter-spacing: normal;
  line-height: normal;
  word-spacing: normal;
  font-family: math;
  font-size: inherit;
  font-style: normal;
  font-weight: normal;
  display: inline math;
  math-shift: normal;
  math-style: compact;
  math-depth: 0;
}
math[display="block" i] {
  display: block math;
  math-style: normal;
}
math[display="inline" i] {
  display: inline math;
  math-style: compact;
}

/* <mrow>-like elements */
semantics > :not(:first-child) {
  display: none;
}
maction > :not(:first-child) {
```

```
  display: none;
}
merror {
  border: 1px solid red;
  background-color: lightYellow;
}
mphantom {
  visibility: hidden;
}

/* Token elements */
mi {
  text-transform: math-auto;
}

/* Tables */
mtable {
  display: inline-table;
  math-style: compact;
}
mtr {
  display: table-row;
}
mtd {
  display: table-cell;
  /* Centering inside table cells should rely on box alignment properties.
     See https://github.com/w3c/mathml-core/issues/156 */
  text-align: center;
  padding: 0.5ex 0.4em;
}

/* Fractions */
mfrac {
  padding-inline: 1px;
}
mfrac > * {
  math-depth: auto-add;
  math-style: compact;
}
mfrac > :nth-child(2) {
  math-shift: compact;
}
```

```
/* Other rules for scriptlevel, displaystyle and math-shift */
mroot > :not(:first-child) {
  math-depth: add(2);
  math-style: compact;
}
mroot, msqrt {
  math-shift: compact;
}
msub > :not(:first-child),
msup > :not(:first-child),
msubsup > :not(:first-child),
mmultiscripts > :not(:first-child),
munder > :not(:first-child),
mover > :not(:first-child),
munderover > :not(:first-child) {
  math-depth: add(1);
  math-style: compact;
}
munder[accentunder="true" i] > :nth-child(2),
mover[accent="true" i] > :nth-child(2),
munderover[accentunder="true" i] > :nth-child(2),
munderover[accent="true" i] > :nth-child(3) {
  font-size: inherit;
}
msub > :nth-child(2),
msubsup > :nth-child(2),
mmultiscripts > :nth-child(even),
mmultiscripts > mprescripts ~ :nth-child(odd),
mover[accent="true" i] > :first-child,
munderover[accent="true" i] > :first-child {
  math-shift: compact;
}
mmultiscripts > mprescripts ~ :nth-child(even) {
  math-shift: inherit;
}
```

B. Operator Tables

B.1 Operator Dictionary

§

NOTE
This section describes how to determine values of 3.2.4.2 Dictionary-based attributes and stretch axis of operators. Compact tables below are suitable for computer manipulation, see B.2 Operator Dictionary (human-readable) for an alternative presentation.

The ***algorithm to set the properties of an operator from its category*** is as follows:

- Set `minsize` to `100%`.
- Set `maxsize` to ∞.
- Find the row corresponding to the specified category on Figure 26.
- Set `lspace` and `rspace` to the value specified in the corresponding column.
- For each property `stretchy`, `symmetric`, `largeop`, `movablelimits`, set that property to `true` if it is listed in the last column or to `false` otherwise.

The ***algorithm to determine the category of an operator*** (`Content`, `Form`) is as folllows:

1. If `Content` as an UTF-16 string does not have length or 1 or 2 then exit with category `Default`.
2. If `Content` is a single character in the range U+0320–U+03FF then exit with category `Default`. Otherwise, if it has two characters:
    - If `Content` is the surrogate pairs corresponding to U+1EEF0 ARABIC MATHEMATICAL OPERATOR MEEM WITH HAH WITH TATWEEL or U+1EEF1 ARABIC MATHEMATICAL OPERATOR HAH WITH DAL and `Form` is `postfix`, exit with category `I`.
    - If the second character is U+0338 COMBINING LONG SOLIDUS OVERLAY or U+20D2 COMBINING LONG VERTICAL LINE OVERLAY then replace `Content` with the first character and move to step 3.
    - Otherwise, if `Content` is listed in `Operators_2_ascii_chars` then replace `Content` with the Unicode character "U+0320 plus the index of `Content` in `Operators_2_ascii_chars`" and move to step 3.
    - Otherwise exit with category `Default`.
3. If `Form` is infix and `Content` corresponds to one of U+007C VERTICAL LINE or U+223C TILDE OPERATOR then exit with category `ForceDefault`. If the category of (`Content`, `Form`) provided by table Figure 25 has N/A encoding in table Figure 26 (namely if it has category

`L` or `M`), then exit with that category. Otherwise:

- Set `Key` to `Content` if it is in range U+0000–U+03FF; or to `Content` – 0x1C00 if it is in range U+2000–U+2BFF. Otherwise, exit with category `Default`.
- Add 0x0000, 0x1000, 0x2000 to `Key` according to whether `Form` is `infix`, `prefix`, `postfix` respectively.
- Assert: `Key` is at most 0x2FFF.
- Search an `Entry` in table Figure 27 such that `Entry` % 0x4000 is equal to `Key`. If one is found then return the category corresponding to encoding `Entry` / 0x1000 in Figure 26. Otherwise, return category `Default`.

Special Table	Entries
`Operators_2_ascii_chars`	18 entries (2-characters ASCII strings): `'!!', '!=', '&&', '**', '*=', '++', '+=', '--', '-=', '->', '//', '/=', ':=', '<=', '<>', '==', '>=', '||',`
`Operators_fence`	61 entries (16 Unicode ranges): `[U+0028–U+0029], {U+005B}, {U+005D}, [U+007B–U+007D], {U+0331}, {U+2016}, [U+2018–U+2019], [U+201C–U+201D], [U+2308–U+230B], [U+2329–U+232A], [U+2772–U+2773], [U+27E6–U+27EF], {U+2980}, [U+2983–U+2999], [U+29D8–U+29DB], [U+29FC–U+29FD],`
`Operators_separator`	3 entries: `U+002C, U+003B, U+2063,`

Figure 24 Special tables for the operator dictionary.
Total size: 82 entries, 90 bytes
(assuming characters are UTF-16 and 1-byte range lengths).

(Content, Form) keys	Category
313 entries (35 Unicode ranges) in **infix** form: [U+2190–U+2195], [U+219A–U+21AE], [U+21B0–U+21B5], {U+21B9}, [U+21BC–U+21D5], [U+21DA–U+21F0], [U+21F3–U+21FF], {U+2794}, {U+2799}, [U+279B–U+27A1], [U+27A5–U+27A6], [U+27A8–U+27AF], {U+27B1}, {U+27B3}, {U+27B5}, {U+27B8}, [U+27BA–U+27BE], [U+27F0–U+27F1], [U+27F4–U+27FF], [U+2900–U+2920], [U+2934–U+2937], [U+2942–U+2975], [U+297C–U+297F], [U+2B04–U+2B07], [U+2B0C–U+2B11], [U+2B30–U+2B3E], [U+2B40–U+2B4C], [U+2B60–U+2B65], [U+2B6A–U+2B6D], [U+2B70–U+2B73], [U+2B7A–U+2B7D], [U+2B80–U+2B87], {U+2B95}, [U+2BA0–U+2BAF], {U+2BB8},	A
108 entries (31 Unicode ranges) in **infix** form: {U+002B}, {U+002D}, {U+00B1}, {U+00F7}, {U+0322}, {U+2044}, [U+2212–U+2216], [U+2227–U+222A], {U+2236}, {U+2238}, [U+228C–U+228E], [U+2293–U+2296], {U+2298}, [U+229D–U+229F], [U+22BB–U+22BD], [U+22CE–U+22CF], [U+22D2–U+22D3], [U+2795–U+2797], {U+29B8}, {U+29BC}, [U+29C4–U+29C5], [U+29F5–U+29FB], [U+2A1F–U+2A2E], [U+2A38–U+2A3A], {U+2A3E}, [U+2A40–U+2A4F], [U+2A51–U+2A63], {U+2ADB}, {U+2AF6}, {U+2AFB}, {U+2AFD},	B
64 entries (33 Unicode ranges) in **infix** form: {U+0025}, {U+002A}, {U+002E}, [U+003F–U+0040], {U+005E}, {U+00B7}, {U+00D7}, {U+0323}, {U+032E}, {U+2022}, {U+2043}, [U+2217–U+2219], {U+2240}, {U+2297}, [U+2299–U+229B], [U+22A0–U+22A1], {U+22BA}, [U+22C4–U+22C7], [U+22C9–U+22CC], [U+2305–U+2306], {U+27CB}, {U+27CD}, [U+29C6–U+29C8], [U+29D4–U+29D7], {U+29E2}, [U+2A1D–U+2A1E], [U+2A2F–U+2A37], [U+2A3B–U+2A3D], {U+2A3F}, {U+2A50}, [U+2A64–U+2A65], [U+2ADC–U+2ADD], {U+2AFE},	C
52 entries (22 Unicode ranges) in **prefix** form: {U+0021}, {U+002B}, {U+002D}, {U+00AC}, {U+00B1}, {U+0331}, {U+2018}, {U+201C}, [U+2200–U+2201], [U+2203–U+2204], {U+2207}, [U+2212–U+2213], [U+221F–U+2222], [U+2234–U+2235], {U+223C}, [U+22BE–U+22BF], {U+2310}, {U+2319}, [U+2795–U+2796], {U+27C0}, [U+299B–U+29AF], [U+2AEC–U+2AED],	D
40 entries (21 Unicode ranges) in **postfix** form: [U+0021–U+0022], [U+0025–U+0027], {U+0060}, {U+00A8}, {U+00B0}, [U+00B2–U+00B4], [U+00B8–U+00B9], [U+02CA–U+02CB], [U+02D8–U+02DA], {U+02DD}, {U+0311}, {U+0320}, {U+0325}, {U+0327}, {U+0331}, [U+2019–U+201B], [U+201D–U+201F], [U+2032–U+2037], {U+2057}, [U+20DB–U+20DC], {U+23CD},	E
30 entries in **prefix** form: U+0028, U+005B, U+007B, U+007C, U+2016, U+2308, U+230A, U+2329, U+2772, U+27E6, U+27E8, U+27EA, U+27EC, U+27EE, U+2980, U+2983, U+2985, U+2987, U+2989, U+298B, U+298D, U+298F,	F

U+2991, U+2993, U+2995, U+2997, U+2999, U+29D8, U+29DA, U+29FC,	
30 entries in **postfix** form: U+0029, U+005D, U+007C, U+007D, U+2016, U+2309, U+230B, U+232A, U+2773, U+27E7, U+27E9, U+27EB, U+27ED, U+27EF, U+2980, U+2984, U+2986, U+2988, U+298A, U+298C, U+298E, U+2990, U+2992, U+2994, U+2996, U+2998, U+2999, U+29D9, U+29DB, U+29FD,	G
27 entries (2 Unicode ranges) in **prefix** form: [U+222B–U+2233], [U+2A0B–U+2A1C],	H
22 entries (13 Unicode ranges) in **postfix** form: [U+005E–U+005F], {U+007E}, {U+00AF}, [U+02C6–U+02C7], {U+02C9}, {U+02CD}, {U+02DC}, {U+02F7}, {U+0302}, {U+203E}, [U+2322–U+2323], [U+23B4–U+23B5], [U+23DC–U+23E1],	I
22 entries (6 Unicode ranges) in **prefix** form: [U+220F–U+2211], [U+22C0–U+22C3], [U+2A00–U+2A0A], [U+2A1D–U+2A1E], {U+2AFC}, {U+2AFF},	J
8 entries (5 Unicode ranges) in **infix** form: {U+002F}, {U+005C}, {U+005F}, [U+2061–U+2064], {U+2206},	K
6 entries (3 Unicode ranges) in **prefix** form: [U+2145–U+2146], {U+2202}, [U+221A–U+221C],	L
3 entries in **infix** form: U+002C, U+003A, U+003B,	M

Figure 25 Mapping from operator (Content, Form) to a category.
Total size: 725 entries, 639 bytes
(assuming characters are UTF-16 and 1-byte range lengths).

Category	Form	Encoding	lspace	rspace	properties
Default	N/A	N/A	`0.2777777777777778em`	`0.2777777777777778em`	N/A
ForceDefault	N/A	N/A	`0.2777777777777778em`	`0.2777777777777778em`	N/A
A	infix	0x0	`0.2777777777777778em`	`0.2777777777777778em`	stretchy
B	infix	0x4	`0.2222222222222222em`	`0.2222222222222222em`	N/A
C	infix	0x8	`0.16666666666666666em`	`0.16666666666666666em`	N/A
D	prefix	0x1	`0`	`0`	N/A
E	postfix	0x2	`0`	`0`	N/A
F	prefix	0x5	`0`	`0`	stretchy symmetric
G	postfix	0x6	`0`	`0`	stretchy symmetric
H	prefix	0x9	`0.16666666666666666em`	`0.16666666666666666em`	symmetric largeop
I	postfix	0xA	`0`	`0`	stretchy
J	prefix	0xD	`0.16666666666666666em`	`0.16666666666666666em`	symmetric largeop movablelimits
K	infix	0xC	`0`	`0`	N/A
L	prefix	N/A	`0.16666666666666666em`	`0`	N/A
M	infix	N/A	`0`	`0.16666666666666666em`	N/A

Figure 26 Operators values for each category.
The third column provides a 4-bit encoding of the categories
where the 2 least significant bits encode the form infix (0), prefix (1) and postfix (2).

716 entries (236 ranges of length at most 16): {0x8025}, {0x802A}, {0x402B}, {0x402D}, {0x802E}, {0xC02F}, [0x803F–0x8040], {0xC05C}, {0x805E}, {0xC05F}, {0x40B1}, {0x80B7}, {0x80D7}, {0x40F7}, {0x4322}, {0x8323}, {0x832E}, {0x8422}, {0x8443}, {0x4444}, [0xC461–0xC464], [0x0590–0x0595], [0x059A–0x05A9], [0x05AA–0x05AE], [0x05B0–0x05B5], {0x05B9}, [0x05BC–0x05CB], [0x05CC–0x05D5], [0x05DA–0x05E9], [0x05EA–0x05F0], [0x05F3–0x05FF], {0xC606}, [0x4612–0x4616], [0x8617–0x8619], [0x4627–0x462A], {0x4636}, {0x4638}, {0x8640}, [0x468C–0x468E], [0x4693–0x4696], {0x8697}, {0x4698}, [0x8699–0x869B], [0x469D–0x469F], [0x86A0–0x86A1], {0x86BA}, [0x46BB–0x46BD], [0x86C4–0x86C7], [0x86C9–0x86CC], [0x46CE–0x46CF], [0x46D2–0x46D3], [0x8705–0x8706], {0x0B94}, [0x4B95–0x4B97], {0x0B99}, [0x0B9B–0x0BA1], [0x0BA5–0x0BA6], [0x0BA8–0x0BAF], {0x0BB1}, {0x0BB3}, {0x0BB5}, {0x0BB8}, [0x0BBA–0x0BBE], {0x8BCB}, {0x8BCD}, [0x0BF0–0x0BF1], [0x0BF4–0x0BFF], [0x0D00–0x0D0F], [0x0D10–0x0D1F], {0x0D20}, [0x0D34–0x0D37], [0x0D42–0x0D51], [0x0D52–0x0D61], [0x0D62–0x0D71], [0x0D72–0x0D75], [0x0D7C–0x0D7F], {0x4DB8}, {0x4DBC}, [0x4DC4–0x4DC5], [0x8DC6–0x8DC8], [0x8DD4–0x8DD7], {0x8DE2}, [0x4DF5–0x4DFB], [0x8E1D–0x8E1E], [0x4E1F–0x4E2E], [0x8E2F–0x8E37], [0x4E38–0x4E3A], [0x8E3B–0x8E3D], {0x4E3E}, {0x8E3F}, [0x4E40–0x4E4F], {0x8E50}, [0x4E51–0x4E60], [0x4E61–0x4E63], [0x8E64–0x8E65], {0x4EDB}, [0x8EDC–0x8EDD], {0x4EF6}, {0x4EFB}, {0x4EFD}, {0x8EFE}, [0x0F04–0x0F07], [0x0F0C–0x0F11], [0x0F30–0x0F3E], [0x0F40–0x0F4C], [0x0F60–0x0F65], [0x0F6A–0x0F6D], [0x0F70–0x0F73], [0x0F7A–0x0F7D], [0x0F80–0x0F87], {0x0F95}, [0x0FA0–0x0FAF], {0x0FB8}, {0x1021}, {0x5028}, {0x102B}, {0x102D}, {0x505B}, [0x507B–0x507C], {0x10AC}, {0x10B1}, {0x1331}, {0x5416}, {0x1418}, {0x141C}, [0x1600–0x1601], [0x1603–0x1604], {0x1607}, [0xD60F–0xD611], [0x1612–0x1613], [0x161F–0x1622], [0x962B–0x9633], [0x1634–0x1635], {0x163C}, [0x16BE–0x16BF], [0xD6C0–0xD6C3], {0x5708}, {0x570A}, {0x1710}, {0x1719}, {0x5729}, {0x5B72}, [0x1B95–0x1B96], {0x1BC0}, {0x5BE6}, {0x5BE8}, {0x5BEA}, {0x5BEC}, {0x5BEE}, {0x5D80}, {0x5D83}, {0x5D85}, {0x5D87}, {0x5D89}, {0x5D8B}, {0x5D8D}, {0x5D8F}, {0x5D91}, {0x5D93}, {0x5D95}, {0x5D97}, {0x5D99}, [0x1D9B–0x1DAA], [0x1DAB–0x1DAF], {0x5DD8}, {0x5DDA}, {0x5DFC}, [0xDE00–0xDE0A], [0x9E0B–0x9E1A], [0x9E1B–0x9E1C], [0xDE1D–0xDE1E], [0x1EEC–0x1EED], {0xDEFC}, {0xDEFF}, [0x2021–0x2022], [0x2025–0x2027], {0x6029}, {0x605D}, [0xA05E–0xA05F], {0x2060}, [0x607C–0x607D], {0xA07E}, {0x20A8}, {0xA0AF}, {0x20B0}, [0x20B2–0x20B4], [0x20B8–0x20B9], [0xA2C6–0xA2C7], {0xA2C9}, [0x22CA–0x22CB], {0xA2CD}, [0x22D8–0x22DA], {0xA2DC}, {0x22DD}, {0xA2F7}, {0xA302}, {0x2311}, {0x2320}, {0x2325}, {0x2327}, {0x2331}, {0x6416}, [0x2419–0x241B], [0x241D–0x241F], [0x2432–0x2437], {0xA43E}, {0x2457}, [0x24DB–0x24DC], {0x6709}, {0x670B}, [0xA722–

```
0xA723], {0x672A}, [0xA7B4–0xA7B5], {0x27CD}, [0xA7DC–0xA7E1], {0x6B73},
  {0x6BE7}, {0x6BE9}, {0x6BEB}, {0x6BED}, {0x6BEF}, {0x6D80}, {0x6D84},
  {0x6D86}, {0x6D88}, {0x6D8A}, {0x6D8C}, {0x6D8E}, {0x6D90}, {0x6D92},
    {0x6D94}, {0x6D96}, [0x6D98–0x6D99], {0x6DD9}, {0x6DDB}, {0x6DFD},
```

Figure 27 List of entries for the largest categories, sorted by key.
`Key` is `Entry` % 0x4000, category encoding is `Entry` / 0x1000.
Total size: 716 entries, 590 bytes
(assuming 4 bits for range lengths).

NOTE

- Tables of Figure 25 and Figure 27 are encoded as ranges to take profit of the presence of many contiguous Unicode blocks.
- To quickly find an entry in these tables, one can still perform a binary search over the range starts, followed by an extra check on the range length.
- Since log is concave, it is more efficient to perform one binary search on the whole table of Figure 27 rather than on each large subtable of Figure 25.

The ***intrinsic stretch axis*** a Unicode character `c` is *inline* if it belongs to the list below. Otherwise, the intrinsic stretch axis of `c` is *block*.

```
U+003D, U+005E, U+005F, U+007E, U+00AF, U+02C6, U+02C7, U+02C9, U+02CD,
U+02DC, U+02F7, U+0302, U+0332, U+203E, U+20D0, U+20D1, U+20D6, U+20D7,
U+20E1, U+2190, U+2192, U+2194, U+2198, U+2199, U+219A, U+219B, U+219C,
U+219D, U+219E, U+21A0, U+21A2, U+21A3, U+21A4, U+21A6, U+21A9, U+21AA,
U+21AB, U+21AC, U+21AD, U+21AE, U+21B4, U+21B9, U+21BC, U+21BD, U+21C0,
U+21C1, U+21C4, U+21C6, U+21C7, U+21C9, U+21CB, U+21CC, U+21CD, U+21CE,
U+21CF, U+21D0, U+21D2, U+21D4, U+21DA, U+21DB, U+21DC, U+21DD, U+21E0,
U+21E2, U+21E4, U+21E5, U+21E6, U+21E8, U+21F0, U+21F4, U+21F6, U+21F7,
U+21F8, U+21F9, U+21FA, U+21FB, U+21FC, U+21FD, U+21FE, U+21FF, U+2322,
U+2323, U+23B4, U+23B5, U+23DC, U+23DD, U+23DE, U+23DF, U+23E0, U+23E1,
U+2500, U+2794, U+2799, U+279B, U+279C, U+279D, U+279E, U+279F, U+27A0,
U+27A1, U+27A5, U+27A6, U+27A8, U+27A9, U+27AA, U+27AB, U+27AC, U+27AD,
U+27AE, U+27AF, U+27B1, U+27B3, U+27B5, U+27B8, U+27BA, U+27BB, U+27BC,
U+27BD, U+27BE, U+27F4, U+27F5, U+27F6, U+27F7, U+27F8, U+27F9, U+27FA,
U+27FB, U+27FC, U+27FD, U+27FE, U+27FF, U+2900, U+2901, U+2902, U+2903,
U+2904, U+2905, U+2906, U+2907, U+290C, U+290D, U+290E, U+290F, U+2910,
U+2911, U+2914, U+2915, U+2916, U+2917, U+2918, U+2919, U+291A, U+291B,
U+291C, U+291D, U+291E, U+291F, U+2920, U+2942, U+2943, U+2944, U+2945,
U+2946, U+2947, U+2948, U+294A, U+294B, U+294E, U+2950, U+2952, U+2953,
U+2956, U+2957, U+295A, U+295B, U+295E, U+295F, U+2962, U+2964, U+2966,
U+2967, U+2968, U+2969, U+296A, U+296B, U+296C, U+296D, U+2970, U+2971,
U+2972, U+2973, U+2974, U+2975, U+297C, U+297D, U+2B04, U+2B05, U+2B0C,
U+2B30, U+2B31, U+2B32, U+2B33, U+2B34, U+2B35, U+2B36, U+2B37, U+2B38,
U+2B39, U+2B3A, U+2B3B, U+2B3C, U+2B3D, U+2B3E, U+2B40, U+2B41, U+2B42,
U+2B43, U+2B44, U+2B45, U+2B46, U+2B47, U+2B48, U+2B49, U+2B4A, U+2B4B,
U+2B4C, U+2B60, U+2B62, U+2B64, U+2B6A, U+2B6C, U+2B70, U+2B72, U+2B7A,
U+2B7C, U+2B80, U+2B82, U+2B84, U+2B86, U+2B95, U+FE35, U+FE36, U+FE37,
                        U+FE38, U+1EEF0, U+1EEF1,
```

Figure 28 Sorted list of Unicode code points corresponding to operators with inline stretch axis.
Total size: 246 entries, 492 bytes (assuming 16 bits for all but the non-BMP entries).

NOTE

The intrinsic stretch axis could be included as a boolean property of the operator dictionary. But since it does not depend on the form and since very few operators can stretch along the inline axis, it is better implemented as a separate sorted array. Each entry can be encoded with 16 bytes if U+1EEF0 ARABIC MATHEMATICAL OPERATOR MEEM WITH HAH WITH TATWEEL and U+1EEF1 ARABIC MATHEMATICAL OPERATOR HAH WITH DAL are tested separately.

B.2 Operator Dictionary (human-readable)

This section is non-normative.

The following dictionary provides a human-readable version of B.1 Operator Dictionary. Please refer to 3.2.4.2 Dictionary-based attributes for explanation about how to use this dictionary and how to determine the values `Content` and `Form` indexing together the dictionary.

The values for `rspace` and `lspace` are indicated in the corresponding columns. The values of `stretchy`, `symmetric`, `largeop`, `movablelimits` are `true` if they are listed in the "properties" column.

Content	Stretch Axis	form	lspace	rspace	properties
< U+003C	block	infix	0.277777777777778em	0.277777777777778em	N/A
= U+003D	inline	infix	0.277777777777778em	0.277777777777778em	N/A
> U+003E	block	infix	0.277777777777778em	0.277777777777778em	N/A
| U+007C	block	infix	0.277777777777778em	0.277777777777778em	fence
↖ U+2196	block	infix	0.277777777777778em	0.277777777777778em	N/A
↗ U+2197	block	infix	0.277777777777778em	0.277777777777778em	N/A
↘ U+2198	inline	infix	0.277777777777778em	0.277777777777778em	N/A
↙ U+2199	inline	infix	0.277777777777778em	0.277777777777778em	N/A
↯ U+21AF	block	infix	0.277777777777778em	0.277777777777778em	N/A
↶ U+21B6	block	infix	0.277777777777778em	0.277777777777778em	N/A
↷ U+21B7	block	infix	0.277777777777778em	0.277777777777778em	N/A
↸ U+21B8	block	infix	0.277777777777778em	0.277777777777778em	N/A
↺ U+21BA	block	infix	0.277777777777778em	0.277777777777778em	N/A
↻ U+21BB	block	infix	0.277777777777778em	0.277777777777778em	N/A
⇖ U+21D6	block	infix	0.277777777777778em	0.277777777777778em	N/A
⇗ U+21D7	block	infix	0.277777777777778em	0.277777777777778em	N/A
⇘ U+21D8	block	infix	0.277777777777778em	0.277777777777778em	N/A

Content	Stretch Axis	form	lspace	rspace	properties
⇙ U+21D9	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⇱ U+21F1	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⇲ U+21F2	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
∈ U+2208	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
∉ U+2209	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
∊ U+220A	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
∋ U+220B	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
∌ U+220C	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
∍ U+220D	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
∝ U+221D	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
| U+2223	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
∤ U+2224	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
∥ U+2225	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
∦ U+2226	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
∷ U+2237	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
∹ U+2239	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
∺ U+223A	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
∻ U+223B	block	infix	0.2777777777777778em	0.2777777777777778em	N/A

Content	Stretch Axis	form	lspace	rspace	properties
∼ U+223C	block	infix	0.277777777777778em	0.277777777777778em	N/A
∽ U+223D	block	infix	0.277777777777778em	0.277777777777778em	N/A
∾ U+223E	block	infix	0.277777777777778em	0.277777777777778em	N/A
≁ U+2241	block	infix	0.277777777777778em	0.277777777777778em	N/A
≃ U+2242	block	infix	0.277777777777778em	0.277777777777778em	N/A
≄ U+2243	block	infix	0.277777777777778em	0.277777777777778em	N/A
≅ U+2244	block	infix	0.277777777777778em	0.277777777777778em	N/A
≆ U+2245	block	infix	0.277777777777778em	0.277777777777778em	N/A
≇ U+2246	block	infix	0.277777777777778em	0.277777777777778em	N/A
≈ U+2247	block	infix	0.277777777777778em	0.277777777777778em	N/A
≉ U+2248	block	infix	0.277777777777778em	0.277777777777778em	N/A
≊ U+2249	block	infix	0.277777777777778em	0.277777777777778em	N/A
≋ U+224A	block	infix	0.277777777777778em	0.277777777777778em	N/A
≌ U+224B	block	infix	0.277777777777778em	0.277777777777778em	N/A
≍ U+224C	block	infix	0.277777777777778em	0.277777777777778em	N/A
≎ U+224D	block	infix	0.277777777777778em	0.277777777777778em	N/A
≏ U+224E	block	infix	0.277777777777778em	0.277777777777778em	N/A

Content	Stretch Axis	form	lspace	rspace	properties
≒ U+224F	block	infix	0.277777777777778em	0.277777777777778em	N/A
≐ U+2250	block	infix	0.277777777777778em	0.277777777777778em	N/A
≑ U+2251	block	infix	0.277777777777778em	0.277777777777778em	N/A
≒ U+2252	block	infix	0.277777777777778em	0.277777777777778em	N/A
≓ U+2253	block	infix	0.277777777777778em	0.277777777777778em	N/A
≔ U+2254	block	infix	0.277777777777778em	0.277777777777778em	N/A
≕ U+2255	block	infix	0.277777777777778em	0.277777777777778em	N/A
≖ U+2256	block	infix	0.277777777777778em	0.277777777777778em	N/A
≗ U+2257	block	infix	0.277777777777778em	0.277777777777778em	N/A
≘ U+2258	block	infix	0.277777777777778em	0.277777777777778em	N/A
≙ U+2259	block	infix	0.277777777777778em	0.277777777777778em	N/A
≚ U+225A	block	infix	0.277777777777778em	0.277777777777778em	N/A
≛ U+225B	block	infix	0.277777777777778em	0.277777777777778em	N/A
≜ U+225C	block	infix	0.277777777777778em	0.277777777777778em	N/A
≝ U+225D	block	infix	0.277777777777778em	0.277777777777778em	N/A
≞ U+225E	block	infix	0.277777777777778em	0.277777777777778em	N/A
≟ U+225F	block	infix	0.277777777777778em	0.277777777777778em	N/A

Content	Stretch Axis	form	lspace	rspace	properties
≠ U+2260	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
≡ U+2261	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
≢ U+2262	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
≣ U+2263	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
≤ U+2264	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
≥ U+2265	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
≦ U+2266	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
≧ U+2267	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
≨ U+2268	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
≩ U+2269	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
≪ U+226A	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
≫ U+226B	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
≪̸ U+226C	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
≺ U+226D	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
≻ U+226E	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
≼ U+226F	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
≽ U+2270	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
≾ U+2271	block	infix	0.2777777777777778em	0.2777777777777778em	N/A

Content	Stretch Axis	form	lspace	rspace	properties
≲ U+2272	block	infix	0.277777777777778em	0.277777777777778em	N/A
≳ U+2273	block	infix	0.277777777777778em	0.277777777777778em	N/A
≴ U+2274	block	infix	0.277777777777778em	0.277777777777778em	N/A
≵ U+2275	block	infix	0.277777777777778em	0.277777777777778em	N/A
≶ U+2276	block	infix	0.277777777777778em	0.277777777777778em	N/A
≷ U+2277	block	infix	0.277777777777778em	0.277777777777778em	N/A
≸ U+2278	block	infix	0.277777777777778em	0.277777777777778em	N/A
≹ U+2279	block	infix	0.277777777777778em	0.277777777777778em	N/A
≺ U+227A	block	infix	0.277777777777778em	0.277777777777778em	N/A
≻ U+227B	block	infix	0.277777777777778em	0.277777777777778em	N/A
≼ U+227C	block	infix	0.277777777777778em	0.277777777777778em	N/A
≽ U+227D	block	infix	0.277777777777778em	0.277777777777778em	N/A
≾ U+227E	block	infix	0.277777777777778em	0.277777777777778em	N/A
≿ U+227F	block	infix	0.277777777777778em	0.277777777777778em	N/A
⊀ U+2280	block	infix	0.277777777777778em	0.277777777777778em	N/A
⊁ U+2281	block	infix	0.277777777777778em	0.277777777777778em	N/A
⊂ U+2282	block	infix	0.277777777777778em	0.277777777777778em	N/A

Content	Stretch Axis	form	lspace	rspace	properties
⊃ U+2283	block	infix	0.277777777777778em	0.277777777777778em	N/A
⊄ U+2284	block	infix	0.277777777777778em	0.277777777777778em	N/A
⊅ U+2285	block	infix	0.277777777777778em	0.277777777777778em	N/A
⊆ U+2286	block	infix	0.277777777777778em	0.277777777777778em	N/A
⊇ U+2287	block	infix	0.277777777777778em	0.277777777777778em	N/A
⊈ U+2288	block	infix	0.277777777777778em	0.277777777777778em	N/A
⊉ U+2289	block	infix	0.277777777777778em	0.277777777777778em	N/A
⊊ U+228A	block	infix	0.277777777777778em	0.277777777777778em	N/A
⊋ U+228B	block	infix	0.277777777777778em	0.277777777777778em	N/A
⊏ U+228F	block	infix	0.277777777777778em	0.277777777777778em	N/A
⊐ U+2290	block	infix	0.277777777777778em	0.277777777777778em	N/A
⊑ U+2291	block	infix	0.277777777777778em	0.277777777777778em	N/A
⊒ U+2292	block	infix	0.277777777777778em	0.277777777777778em	N/A
⊖ U+229C	block	infix	0.277777777777778em	0.277777777777778em	N/A
⊢ U+22A2	block	infix	0.277777777777778em	0.277777777777778em	N/A
⊣ U+22A3	block	infix	0.277777777777778em	0.277777777777778em	N/A
⊦ U+22A6	block	infix	0.277777777777778em	0.277777777777778em	N/A

Content	Stretch Axis	form	lspace	rspace	properties
⊧ U+22A7	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⊨ U+22A8	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⊩ U+22A9	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⊪ U+22AA	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⊫ U+22AB	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⊬ U+22AC	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⊭ U+22AD	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⊮ U+22AE	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⊯ U+22AF	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⊰ U+22B0	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⊱ U+22B1	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⊲ U+22B2	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⊳ U+22B3	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⊴ U+22B4	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⊵ U+22B5	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⊶ U+22B6	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⊷ U+22B7	block	infix	0.2777777777777778em	0.2777777777777778em	N/A

Content	Stretch Axis	form	lspace	rspace	properties
⊸ U+22B8	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⋈ U+22C8	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⋍ U+22CD	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⋐ U+22D0	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⋑ U+22D1	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⋔ U+22D4	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⋕ U+22D5	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⋖ U+22D6	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⋗ U+22D7	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⋘ U+22D8	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⋙ U+22D9	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⋚ U+22DA	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⋛ U+22DB	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⋜ U+22DC	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⋝ U+22DD	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⋞ U+22DE	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⋟ U+22DF	block	infix	0.2777777777777778em	0.2777777777777778em	N/A

Content	Stretch Axis	form	lspace	rspace	properties
⫝̸ U+22E0	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⫝ U+22E1	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⫋ U+22E2	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⫌ U+22E3	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⫍ U+22E4	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⫎ U+22E5	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⋖̸ U+22E6	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⋗̸ U+22E7	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⋦̸ U+22E8	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⋧̸ U+22E9	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⋪ U+22EA	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⋫ U+22EB	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⋬ U+22EC	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⋭ U+22ED	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⋲ U+22F2	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⋳ U+22F3	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⋴ U+22F4	block	infix	0.2777777777777778em	0.2777777777777778em	N/A

Content	Stretch Axis	form	lspace	rspace	properties
⋰ U+22F0	block	infix	0.277777777777778em	0.277777777777778em	N/A
⋱ U+22F1	block	infix	0.277777777777778em	0.277777777777778em	N/A
⋲ U+22F2	block	infix	0.277777777777778em	0.277777777777778em	N/A
⋳ U+22F3	block	infix	0.277777777777778em	0.277777777777778em	N/A
⋴ U+22F4	block	infix	0.277777777777778em	0.277777777777778em	N/A
⋵ U+22F5	block	infix	0.277777777777778em	0.277777777777778em	N/A
⋶ U+22F6	block	infix	0.277777777777778em	0.277777777777778em	N/A
⋷ U+22F7	block	infix	0.277777777777778em	0.277777777777778em	N/A
⋸ U+22F8	block	infix	0.277777777777778em	0.277777777777778em	N/A
⋹ U+22F9	block	infix	0.277777777777778em	0.277777777777778em	N/A
⋺ U+22FA	block	infix	0.277777777777778em	0.277777777777778em	N/A
⋻ U+22FB	block	infix	0.277777777777778em	0.277777777777778em	N/A
⋼ U+22FC	block	infix	0.277777777777778em	0.277777777777778em	N/A
⋽ U+22FD	block	infix	0.277777777777778em	0.277777777777778em	N/A
⋾ U+22FE	block	infix	0.277777777777778em	0.277777777777778em	N/A
⋿ U+22FF	block	infix	0.277777777777778em	0.277777777777778em	N/A
⌁ U+2301	block	infix	0.277777777777778em	0.277777777777778em	N/A
⍼ U+237C	block	infix	0.277777777777778em	0.277777777777778em	N/A
⎋ U+238B	block	infix	0.277777777777778em	0.277777777777778em	N/A
⤡ U+2798	block	infix	0.277777777777778em	0.277777777777778em	N/A
⤢ U+279A	block	infix	0.277777777777778em	0.277777777777778em	N/A
⤧ U+27A7	block	infix	0.277777777777778em	0.277777777777778em	N/A
⤲ U+27B2	block	infix	0.277777777777778em	0.277777777777778em	N/A

Content	Stretch Axis	form	lspace	rspace	properties
⤴ U+27B4	block	infix	0.277777777777778em	0.277777777777778em	N/A
➶ U+27B6	block	infix	0.277777777777778em	0.277777777777778em	N/A
➷ U+27B7	block	infix	0.277777777777778em	0.277777777777778em	N/A
➹ U+27B9	block	infix	0.277777777777778em	0.277777777777778em	N/A
⊥ U+27C2	block	infix	0.277777777777778em	0.277777777777778em	N/A
⟲ U+27F2	block	infix	0.277777777777778em	0.277777777777778em	N/A
⟳ U+27F3	block	infix	0.277777777777778em	0.277777777777778em	N/A
⤡ U+2921	block	infix	0.277777777777778em	0.277777777777778em	N/A
⤢ U+2922	block	infix	0.277777777777778em	0.277777777777778em	N/A
⤣ U+2923	block	infix	0.277777777777778em	0.277777777777778em	N/A
⤤ U+2924	block	infix	0.277777777777778em	0.277777777777778em	N/A
⤥ U+2925	block	infix	0.277777777777778em	0.277777777777778em	N/A
⤦ U+2926	block	infix	0.277777777777778em	0.277777777777778em	N/A
⤧ U+2927	block	infix	0.277777777777778em	0.277777777777778em	N/A
⤨ U+2928	block	infix	0.277777777777778em	0.277777777777778em	N/A
⤩ U+2929	block	infix	0.277777777777778em	0.277777777777778em	N/A
⤪ U+292A	block	infix	0.277777777777778em	0.277777777777778em	N/A
⤫ U+292B	block	infix	0.277777777777778em	0.277777777777778em	N/A
⤬ U+292C	block	infix	0.277777777777778em	0.277777777777778em	N/A
⤭ U+292D	block	infix	0.277777777777778em	0.277777777777778em	N/A
⤮ U+292E	block	infix	0.277777777777778em	0.277777777777778em	N/A

Content	Stretch Axis	form	lspace	rspace	properties
⤳ U+292F	block	infix	0.277777777777778em	0.277777777777778em	N/A
⤴ U+2930	block	infix	0.277777777777778em	0.277777777777778em	N/A
⤵ U+2931	block	infix	0.277777777777778em	0.277777777777778em	N/A
⤶ U+2932	block	infix	0.277777777777778em	0.277777777777778em	N/A
⤷ U+2933	block	infix	0.277777777777778em	0.277777777777778em	N/A
⤸ U+2938	block	infix	0.277777777777778em	0.277777777777778em	N/A
⤹ U+2939	block	infix	0.277777777777778em	0.277777777777778em	N/A
⤺ U+293A	block	infix	0.277777777777778em	0.277777777777778em	N/A
⤻ U+293B	block	infix	0.277777777777778em	0.277777777777778em	N/A
⤼ U+293C	block	infix	0.277777777777778em	0.277777777777778em	N/A
⤽ U+293D	block	infix	0.277777777777778em	0.277777777777778em	N/A
⤾ U+293E	block	infix	0.277777777777778em	0.277777777777778em	N/A
⤿ U+293F	block	infix	0.277777777777778em	0.277777777777778em	N/A
⥀ U+2940	block	infix	0.277777777777778em	0.277777777777778em	N/A
⥁ U+2941	block	infix	0.277777777777778em	0.277777777777778em	N/A
⥶ U+2976	block	infix	0.277777777777778em	0.277777777777778em	N/A
⥷ U+2977	block	infix	0.277777777777778em	0.277777777777778em	N/A
⥸ U+2978	block	infix	0.277777777777778em	0.277777777777778em	N/A
⥹ U+2979	block	infix	0.277777777777778em	0.277777777777778em	N/A
⥺ U+297A	block	infix	0.277777777777778em	0.277777777777778em	N/A
⥻ U+297B	block	infix	0.277777777777778em	0.277777777777778em	N/A

Content	Stretch Axis	form	lspace	rspace	properties
● U+2981	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⦘ U+2982	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⦶ U+29B6	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⦷ U+29B7	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⦹ U+29B9	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⧀ U+29C0	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⧁ U+29C1	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⧎ U+29CE	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⧏ U+29CF	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⧐ U+29D0	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⧑ U+29D1	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⧒ U+29D2	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⧓ U+29D3	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⧟ U+29DF	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⧡ U+29E1	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⧣ U+29E3	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⧤ U+29E4	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⧥ U+29E5	block	infix	0.2777777777777778em	0.2777777777777778em	N/A

Content	Stretch Axis	form	lspace	rspace	properties
⩪ U+29E6	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⩲ U+29F4	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⩶ U+2A66	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⩷ U+2A67	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⩸ U+2A68	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⩹ U+2A69	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⩺ U+2A6A	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⩻ U+2A6B	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⩼ U+2A6C	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⩽ U+2A6D	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⩾ U+2A6E	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⩿ U+2A6F	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⪀ U+2A70	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⪁ U+2A71	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⪂ U+2A72	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⪃ U+2A73	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⪄ U+2A74	block	infix	0.2777777777777778em	0.2777777777777778em	N/A

Content	Stretch Axis	form	lspace	rspace	properties
== U+2A75	block	infix	0.277777777777778em	0.277777777777778em	N/A
=== U+2A76	block	infix	0.277777777777778em	0.277777777777778em	N/A
⩷ U+2A77	block	infix	0.277777777777778em	0.277777777777778em	N/A
⩸ U+2A78	block	infix	0.277777777777778em	0.277777777777778em	N/A
⪹ U+2A79	block	infix	0.277777777777778em	0.277777777777778em	N/A
⪺ U+2A7A	block	infix	0.277777777777778em	0.277777777777778em	N/A
⪻ U+2A7B	block	infix	0.277777777777778em	0.277777777777778em	N/A
⪼ U+2A7C	block	infix	0.277777777777778em	0.277777777777778em	N/A
⩽ U+2A7D	block	infix	0.277777777777778em	0.277777777777778em	N/A
⩾ U+2A7E	block	infix	0.277777777777778em	0.277777777777778em	N/A
⪿ U+2A7F	block	infix	0.277777777777778em	0.277777777777778em	N/A
⪀ U+2A80	block	infix	0.277777777777778em	0.277777777777778em	N/A
⪁ U+2A81	block	infix	0.277777777777778em	0.277777777777778em	N/A
⪂ U+2A82	block	infix	0.277777777777778em	0.277777777777778em	N/A
⪃ U+2A83	block	infix	0.277777777777778em	0.277777777777778em	N/A
⪄ U+2A84	block	infix	0.277777777777778em	0.277777777777778em	N/A
⪅ U+2A85	block	infix	0.277777777777778em	0.277777777777778em	N/A

Content	Stretch Axis	form	lspace	rspace	properties
⪆ U+2A86	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⪇ U+2A87	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⪈ U+2A88	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⪉ U+2A89	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⪊ U+2A8A	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⪋ U+2A8B	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⪌ U+2A8C	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⪍ U+2A8D	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⪎ U+2A8E	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⪏ U+2A8F	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⪐ U+2A90	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⪑ U+2A91	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⪒ U+2A92	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⪓ U+2A93	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⪔ U+2A94	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⪕ U+2A95	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⪖ U+2A96	block	infix	0.2777777777777778em	0.2777777777777778em	N/A

Content	Stretch Axis	form	lspace	rspace	properties
⪗ U+2A97	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⪘ U+2A98	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⪙ U+2A99	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⪚ U+2A9A	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⪛ U+2A9B	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⪜ U+2A9C	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⪝ U+2A9D	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⪞ U+2A9E	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⪟ U+2A9F	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⪠ U+2AA0	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⪡ U+2AA1	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⪢ U+2AA2	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⪣ U+2AA3	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⪤ U+2AA4	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⪥ U+2AA5	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⪦ U+2AA6	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⪧ U+2AA7	block	infix	0.2777777777777778em	0.2777777777777778em	N/A

Content	Stretch Axis	form	lspace	rspace	properties
⪸ U+2AA8	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⪹ U+2AA9	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⪺ U+2AAA	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⪻ U+2AAB	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⪼ U+2AAC	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⪽ U+2AAD	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⪾ U+2AAE	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⪿ U+2AAF	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⫀ U+2AB0	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⫁ U+2AB1	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⫂ U+2AB2	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⫃ U+2AB3	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⫄ U+2AB4	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⫅ U+2AB5	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⫆ U+2AB6	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⫇ U+2AB7	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⫈ U+2AB8	block	infix	0.2777777777777778em	0.2777777777777778em	N/A

Content	Stretch Axis	form	lspace	rspace	properties
⪹ U+2AB9	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⪺ U+2ABA	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⪻ U+2ABB	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⪼ U+2ABC	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⪽ U+2ABD	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⪾ U+2ABE	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⪿ U+2ABF	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⫀ U+2AC0	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⫁ U+2AC1	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⫂ U+2AC2	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⫃ U+2AC3	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⫄ U+2AC4	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⫅ U+2AC5	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⫆ U+2AC6	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⫇ U+2AC7	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⫈ U+2AC8	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⫉ U+2AC9	block	infix	0.2777777777777778em	0.2777777777777778em	N/A

Content	Stretch Axis	form	lspace	rspace	properties
⪊ U+2ACA	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⪋ U+2ACB	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⫌ U+2ACC	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⫍ U+2ACD	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⫎ U+2ACE	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⫏ U+2ACF	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⫐ U+2AD0	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⫑ U+2AD1	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⫒ U+2AD2	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⫓ U+2AD3	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⫔ U+2AD4	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⫕ U+2AD5	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⫖ U+2AD6	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⫗ U+2AD7	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⫘ U+2AD8	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⫙ U+2AD9	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⫚ U+2ADA	block	infix	0.2777777777777778em	0.2777777777777778em	N/A

Content	Stretch Axis	form	lspace	rspace	properties
⫤ U+2ADE	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⫥ U+2ADF	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⫦ U+2AE0	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⫧ U+2AE1	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⫨ U+2AE2	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⫩ U+2AE3	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⫪ U+2AE4	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⫫ U+2AE5	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⫬ U+2AE6	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⫭ U+2AE7	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⫮ U+2AE8	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⫯ U+2AE9	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⫰ U+2AEA	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⫱ U+2AEB	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⫮ U+2AEE	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⫲ U+2AF2	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⫳ U+2AF3	block	infix	0.2777777777777778em	0.2777777777777778em	N/A

Content	Stretch Axis	form	lspace	rspace	properties
⫴ U+2AF4	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⫵ U+2AF5	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⫷ U+2AF7	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⫸ U+2AF8	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⫹ U+2AF9	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⫺ U+2AFA	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⇗ U+2B00	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⇖ U+2B01	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⇘ U+2B02	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⇙ U+2B03	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⬈ U+2B08	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⬉ U+2B09	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⬊ U+2B0A	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⬋ U+2B0B	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
⬿ U+2B3F	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
U+2B4D	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
U+2B4E	block	infix	0.2777777777777778em	0.2777777777777778em	N/A

Content	Stretch Axis	form	lspace	rspace	properties
 U+2B4F	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
 U+2B5A	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
 U+2B5B	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
 U+2B5C	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
 U+2B5D	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
 U+2B5E	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
 U+2B5F	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
 U+2B66	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
 U+2B67	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
 U+2B68	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
 U+2B69	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
U+2B6E	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
U+2B6F	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
 U+2B76	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
 U+2B77	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
 U+2B78	block	infix	0.2777777777777778em	0.2777777777777778em	N/A
 U+2B79	block	infix	0.2777777777777778em	0.2777777777777778em	N/A

Content	Stretch Axis	form	lspace	rspace	properties
 U+2B88	block	infix	0.277777777777778em	0.277777777777778em	N/A
 U+2B89	block	infix	0.277777777777778em	0.277777777777778em	N/A
 U+2B8A	block	infix	0.277777777777778em	0.277777777777778em	N/A
 U+2B8B	block	infix	0.277777777777778em	0.277777777777778em	N/A
 U+2B8C	block	infix	0.277777777777778em	0.277777777777778em	N/A
 U+2B8D	block	infix	0.277777777777778em	0.277777777777778em	N/A
 U+2B8E	block	infix	0.277777777777778em	0.277777777777778em	N/A
 U+2B8F	block	infix	0.277777777777778em	0.277777777777778em	N/A
 U+2B94	block	infix	0.277777777777778em	0.277777777777778em	N/A
U+2BB0	block	infix	0.277777777777778em	0.277777777777778em	N/A
U+2BB1	block	infix	0.277777777777778em	0.277777777777778em	N/A
U+2BB2	block	infix	0.277777777777778em	0.277777777777778em	N/A
 U+2BB3	block	infix	0.277777777777778em	0.277777777777778em	N/A
 U+2BB4	block	infix	0.277777777777778em	0.277777777777778em	N/A
 U+2BB5	block	infix	0.277777777777778em	0.277777777777778em	N/A
 U+2BB6	block	infix	0.277777777777778em	0.277777777777778em	N/A
 U+2BB7	block	infix	0.277777777777778em	0.277777777777778em	N/A

Content	Stretch Axis	form	lspace	rspace	properties
U+2BD1	block	infix	0.277777777777778em	0.277777777777778em	N/A
String != U+0021 U+003D	block	infix	0.277777777777778em	0.277777777777778em	N/A
String *= U+002A U+003D	block	infix	0.277777777777778em	0.277777777777778em	N/A
String += U+002B U+003D	block	infix	0.277777777777778em	0.277777777777778em	N/A
String -= U+002D U+003D	block	infix	0.277777777777778em	0.277777777777778em	N/A
String -> U+002D U+003E	block	infix	0.277777777777778em	0.277777777777778em	N/A
String // U+002F U+002F	block	infix	0.277777777777778em	0.277777777777778em	N/A
String /= U+002F U+003D	block	infix	0.277777777777778em	0.277777777777778em	N/A
String := U+003A U+003D	block	infix	0.277777777777778em	0.277777777777778em	N/A
String <= U+003C U+003D	block	infix	0.277777777777778em	0.277777777777778em	N/A
String == U+003D U+003D	block	infix	0.277777777777778em	0.277777777777778em	N/A
String >= U+003E U+003D	block	infix	0.277777777777778em	0.277777777777778em	N/A

Content	Stretch Axis	form	lspace	rspace	properties
String ‖ U+007C U+007C	block	infix	0.277777777777778em	0.277777777777778em	fence
← U+2190	inline	infix	0.277777777777778em	0.277777777777778em	stretchy
↑ U+2191	block	infix	0.277777777777778em	0.277777777777778em	stretchy
→ U+2192	inline	infix	0.277777777777778em	0.277777777777778em	stretchy
↓ U+2193	block	infix	0.277777777777778em	0.277777777777778em	stretchy
↔ U+2194	inline	infix	0.277777777777778em	0.277777777777778em	stretchy
↕ U+2195	block	infix	0.277777777777778em	0.277777777777778em	stretchy
↚ U+219A	inline	infix	0.277777777777778em	0.277777777777778em	stretchy
↛ U+219B	inline	infix	0.277777777777778em	0.277777777777778em	stretchy
↜ U+219C	inline	infix	0.277777777777778em	0.277777777777778em	stretchy
↝ U+219D	inline	infix	0.277777777777778em	0.277777777777778em	stretchy
↞ U+219E	inline	infix	0.277777777777778em	0.277777777777778em	stretchy
↟ U+219F	block	infix	0.277777777777778em	0.277777777777778em	stretchy
↠ U+21A0	inline	infix	0.277777777777778em	0.277777777777778em	stretchy
↡ U+21A1	block	infix	0.277777777777778em	0.277777777777778em	stretchy
↢ U+21A2	inline	infix	0.277777777777778em	0.277777777777778em	stretchy
↣ U+21A3	inline	infix	0.277777777777778em	0.277777777777778em	stretchy
↤ U+21A4	inline	infix	0.277777777777778em	0.277777777777778em	stretchy

Content	Stretch Axis	form	lspace	rspace	properties
↕ U+21A5	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
↦ U+21A6	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
↧ U+21A7	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
↨ U+21A8	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
↩ U+21A9	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
↪ U+21AA	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
↫ U+21AB	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
↬ U+21AC	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
↭ U+21AD	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
↮ U+21AE	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
↰ U+21B0	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
↱ U+21B1	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
↲ U+21B2	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
↳ U+21B3	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
↴ U+21B4	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
↵ U+21B5	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
↹ U+21B9	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy

Content	Stretch Axis	form	lspace	rspace	properties
← U+21BC	inline	infix	0.277777777777778em	0.277777777777778em	stretchy
← U+21BD	inline	infix	0.277777777777778em	0.277777777777778em	stretchy
↾ U+21BE	block	infix	0.277777777777778em	0.277777777777778em	stretchy
↿ U+21BF	block	infix	0.277777777777778em	0.277777777777778em	stretchy
→ U+21C0	inline	infix	0.277777777777778em	0.277777777777778em	stretchy
→ U+21C1	inline	infix	0.277777777777778em	0.277777777777778em	stretchy
⇂ U+21C2	block	infix	0.277777777777778em	0.277777777777778em	stretchy
⇃ U+21C3	block	infix	0.277777777777778em	0.277777777777778em	stretchy
⇄ U+21C4	inline	infix	0.277777777777778em	0.277777777777778em	stretchy
⇅ U+21C5	block	infix	0.277777777777778em	0.277777777777778em	stretchy
⇆ U+21C6	inline	infix	0.277777777777778em	0.277777777777778em	stretchy
⇇ U+21C7	inline	infix	0.277777777777778em	0.277777777777778em	stretchy
⇈ U+21C8	block	infix	0.277777777777778em	0.277777777777778em	stretchy
⇉ U+21C9	inline	infix	0.277777777777778em	0.277777777777778em	stretchy
⇊ U+21CA	block	infix	0.277777777777778em	0.277777777777778em	stretchy
⇋ U+21CB	inline	infix	0.277777777777778em	0.277777777777778em	stretchy
⇌ U+21CC	inline	infix	0.277777777777778em	0.277777777777778em	stretchy
⇍ U+21CD	inline	infix	0.277777777777778em	0.277777777777778em	stretchy

Content	Stretch Axis	form	lspace	rspace	properties
⇎ U+21CE	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⇏ U+21CF	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⇐ U+21D0	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⇑ U+21D1	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⇒ U+21D2	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⇓ U+21D3	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⇔ U+21D4	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⇕ U+21D5	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⇚ U+21DA	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⇛ U+21DB	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⇜ U+21DC	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⇝ U+21DD	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⇞ U+21DE	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⇟ U+21DF	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⇠ U+21E0	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⇡ U+21E1	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⇢ U+21E2	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy

Content	Stretch Axis	form	lspace	rspace	properties
⇣ U+21E3	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⇤ U+21E4	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⇥ U+21E5	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⇦ U+21E6	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⇧ U+21E7	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⇨ U+21E8	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⇩ U+21E9	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⇪ U+21EA	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⇫ U+21EB	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⇬ U+21EC	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⇭ U+21ED	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⇮ U+21EE	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⇯ U+21EF	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⇰ U+21F0	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⇳ U+21F3	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⇴ U+21F4	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⇵ U+21F5	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy

Content	Stretch Axis	form	lspace	rspace	properties
⥱ U+21F6	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⥰ U+21F7	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⥵ U+21F8	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⥷ U+21F9	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⥸ U+21FA	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⥹ U+21FB	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⥺ U+21FC	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⥻ U+21FD	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⥼ U+21FE	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⥽ U+21FF	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
➔ U+2794	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
➙ U+2799	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
➛ U+279B	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
➜ U+279C	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
➝ U+279D	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
➞ U+279E	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
➟ U+279F	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy

Content	Stretch Axis	form	lspace	rspace	properties
➠ U+27A0	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
➡ U+27A1	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
➥ U+27A5	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
➦ U+27A6	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
➨ U+27A8	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
➩ U+27A9	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
➪ U+27AA	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
➫ U+27AB	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
➬ U+27AC	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
➭ U+27AD	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
➮ U+27AE	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
➯ U+27AF	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
➱ U+27B1	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
➳ U+27B3	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
➵ U+27B5	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
➸ U+27B8	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
➺ U+27BA	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy

Content	Stretch Axis	form	lspace	rspace	properties
➻ U+27BB	inline	infix	0.277777777777778em	0.277777777777778em	stretchy
➼ U+27BC	inline	infix	0.277777777777778em	0.277777777777778em	stretchy
➽ U+27BD	inline	infix	0.277777777777778em	0.277777777777778em	stretchy
⟾ U+27BE	inline	infix	0.277777777777778em	0.277777777777778em	stretchy
⤰ U+27F0	block	infix	0.277777777777778em	0.277777777777778em	stretchy
⤱ U+27F1	block	infix	0.277777777777778em	0.277777777777778em	stretchy
⟴ U+27F4	inline	infix	0.277777777777778em	0.277777777777778em	stretchy
⟵ U+27F5	inline	infix	0.277777777777778em	0.277777777777778em	stretchy
⟶ U+27F6	inline	infix	0.277777777777778em	0.277777777777778em	stretchy
⟷ U+27F7	inline	infix	0.277777777777778em	0.277777777777778em	stretchy
⟸ U+27F8	inline	infix	0.277777777777778em	0.277777777777778em	stretchy
⟹ U+27F9	inline	infix	0.277777777777778em	0.277777777777778em	stretchy
⟺ U+27FA	inline	infix	0.277777777777778em	0.277777777777778em	stretchy
⟻ U+27FB	inline	infix	0.277777777777778em	0.277777777777778em	stretchy
⟼ U+27FC	inline	infix	0.277777777777778em	0.277777777777778em	stretchy
⟽ U+27FD	inline	infix	0.277777777777778em	0.277777777777778em	stretchy
⟾ U+27FE	inline	infix	0.277777777777778em	0.277777777777778em	stretchy

Content	Stretch Axis	form	lspace	rspace	properties
⟿ U+27FF	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⤠ U+2900	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⤡ U+2901	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⤢ U+2902	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⤣ U+2903	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⤤ U+2904	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⤥ U+2905	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⤦ U+2906	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⤧ U+2907	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⤨ U+2908	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⤩ U+2909	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⤪ U+290A	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⤫ U+290B	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⤬ U+290C	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⤭ U+290D	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⤮ U+290E	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⤯ U+290F	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⤰ U+2910	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy

Content	Stretch Axis	form	lspace	rspace	properties
⟿ U+2911	inline	infix	0.277777777777778em	0.277777777777778em	stretchy
⤒ U+2912	block	infix	0.277777777777778em	0.277777777777778em	stretchy
⤓ U+2913	block	infix	0.277777777777778em	0.277777777777778em	stretchy
⤔ U+2914	inline	infix	0.277777777777778em	0.277777777777778em	stretchy
⤕ U+2915	inline	infix	0.277777777777778em	0.277777777777778em	stretchy
⤖ U+2916	inline	infix	0.277777777777778em	0.277777777777778em	stretchy
⤗ U+2917	inline	infix	0.277777777777778em	0.277777777777778em	stretchy
⤘ U+2918	inline	infix	0.277777777777778em	0.277777777777778em	stretchy
⤙ U+2919	inline	infix	0.277777777777778em	0.277777777777778em	stretchy
⤚ U+291A	inline	infix	0.277777777777778em	0.277777777777778em	stretchy
⤛ U+291B	inline	infix	0.277777777777778em	0.277777777777778em	stretchy
⤜ U+291C	inline	infix	0.277777777777778em	0.277777777777778em	stretchy
⤝ U+291D	inline	infix	0.277777777777778em	0.277777777777778em	stretchy
⤞ U+291E	inline	infix	0.277777777777778em	0.277777777777778em	stretchy
⤟ U+291F	inline	infix	0.277777777777778em	0.277777777777778em	stretchy
⤠ U+2920	inline	infix	0.277777777777778em	0.277777777777778em	stretchy
⤴ U+2934	block	infix	0.277777777777778em	0.277777777777778em	stretchy
⤵ U+2935	block	infix	0.277777777777778em	0.277777777777778em	stretchy
⤶ U+2936	block	infix	0.277777777777778em	0.277777777777778em	stretchy

Content	Stretch Axis	form	lspace	rspace	properties
⤷ U+2937	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⇄ U+2942	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⇅ U+2943	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⇆ U+2944	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⇈ U+2945	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⇉ U+2946	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⇶ U+2947	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⇔ U+2948	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⇕ U+2949	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⇽ U+294A	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⇁ U+294B	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⥥ U+294C	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⥋ U+294D	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⇀ U+294E	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⥯ U+294F	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⇾ U+2950	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⥏ U+2951	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⇂ U+2952	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⇃ U+2953	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy

Content	Stretch Axis	form	lspace	rspace	properties
↾ U+2954	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⇂ U+2955	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
↿ U+2956	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⇃ U+2957	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
↾ U+2958	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⇂ U+2959	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
↿ U+295A	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⇃ U+295B	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
↿ U+295C	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⇃ U+295D	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
↿ U+295E	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⇃ U+295F	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
↿ U+2960	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⇃ U+2961	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⇐ U+2962	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⇑ U+2963	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⇒ U+2964	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⇓ U+2965	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⇐ U+2966	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⇒ U+2967	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⇐ U+2968	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy

Content	Stretch Axis	form	lspace	rspace	properties
⥯ U+2969	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⥪ U+296A	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⥫ U+296B	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⥬ U+296C	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⥭ U+296D	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⥮ U+296E	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⥯ U+296F	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⥰ U+2970	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⥱ U+2971	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⥲ U+2972	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⥳ U+2973	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⥴ U+2974	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⥵ U+2975	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⥼ U+297C	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⥽ U+297D	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⥾ U+297E	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⥿ U+297F	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy

Content	Stretch Axis	form	lspace	rspace	properties
⇔ U+2B04	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⬅ U+2B05	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⬆ U+2B06	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⬇ U+2B07	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⬌ U+2B0C	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⬍ U+2B0D	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⬎ U+2B0E	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⬏ U+2B0F	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⬐ U+2B10	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⬑ U+2B11	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⬰ U+2B30	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⬱ U+2B31	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⬲ U+2B32	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⬳ U+2B33	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⬴ U+2B34	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⬵ U+2B35	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⬶ U+2B36	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy

Content	Stretch Axis	form	lspace	rspace	properties
⬷ U+2B37	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⬸ U+2B38	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⬹ U+2B39	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⬺ U+2B3A	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⬻ U+2B3B	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⬼ U+2B3C	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⬽ U+2B3D	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⬾ U+2B3E	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⬿ U+2B40	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⭀ U+2B41	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⭁ U+2B42	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⭂ U+2B43	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⭃ U+2B44	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⭄ U+2B45	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⭅ U+2B46	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⭆ U+2B47	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⭇ U+2B48	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy

Content	Stretch Axis	form	lspace	rspace	properties
⇜ U+2B49	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⇜≈ U+2B4A	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⇜≈ U+2B4B	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⇝ U+2B4C	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
← U+2B60	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
↑ U+2B61	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
→ U+2B62	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
↓ U+2B63	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
↔ U+2B64	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
↕ U+2B65	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
U+2B6A	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
U+2B6B	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
U+2B6C	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
U+2B6D	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
U+2B70	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
U+2B71	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
U+2B72	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy

Content	Stretch Axis	form	lspace	rspace	properties
⥳ U+2B73	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⥺ U+2B7A	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⥻ U+2B7B	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⥼ U+2B7C	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⥽ U+2B7D	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⥾ U+2B80	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⥿ U+2B81	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⇄ U+2B82	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⇵ U+2B83	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⦀ U+2B84	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⦁ U+2B85	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⦂ U+2B86	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⦃ U+2B87	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
➡ U+2B95	inline	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⦐ U+2BA0	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⦑ U+2BA1	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⦒ U+2BA2	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy

Content	Stretch Axis	form	lspace	rspace	properties
⨓ U+2BA3	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⨔ U+2BA4	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⨕ U+2BA5	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⨖ U+2BA6	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⨗ U+2BA7	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⨘ U+2BA8	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⨙ U+2BA9	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⨚ U+2BAA	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⨛ U+2BAB	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⨜ U+2BAC	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⨝ U+2BAD	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⨞ U+2BAE	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⨟ U+2BAF	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
⫸ U+2BB8	block	infix	0.2777777777777778em	0.2777777777777778em	stretchy
+ U+002B	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
- U+002D	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
± U+00B1	block	infix	0.2222222222222222em	0.2222222222222222em	N/A

Content	Stretch Axis	form	lspace	rspace	properties
÷ U+00F7	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⁄ U+2044	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
− U+2212	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
∓ U+2213	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
∔ U+2214	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
∕ U+2215	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
∖ U+2216	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
∧ U+2227	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
∨ U+2228	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
∩ U+2229	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
∪ U+222A	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
∶ U+2236	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
∸ U+2238	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⊌ U+228C	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⊍ U+228D	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⊎ U+228E	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⊓ U+2293	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⊔ U+2294	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⊕ U+2295	block	infix	0.2222222222222222em	0.2222222222222222em	N/A

Content	Stretch Axis	form	lspace	rspace	properties
⊖ U+2296	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⊘ U+2298	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⊝ U+229D	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⊞ U+229E	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⊟ U+229F	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⊻ U+22BB	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⊼ U+22BC	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⊽ U+22BD	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⋎ U+22CE	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⋏ U+22CF	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⋒ U+22D2	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⋓ U+22D3	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
➕ U+2795	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
➖ U+2796	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
➗ U+2797	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⦸ U+29B8	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⦼ U+29BC	block	infix	0.2222222222222222em	0.2222222222222222em	N/A

Content	Stretch Axis	form	lspace	rspace	properties
⧄ U+29C4	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⧅ U+29C5	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⧵ U+29F5	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⧶ U+29F6	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⧷ U+29F7	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⧸ U+29F8	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⧹ U+29F9	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⧺ U+29FA	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⧻ U+29FB	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⨟ U+2A1F	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⨠ U+2A20	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⨡ U+2A21	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⨢ U+2A22	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⨣ U+2A23	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⨤ U+2A24	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⨥ U+2A25	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⨦ U+2A26	block	infix	0.2222222222222222em	0.2222222222222222em	N/A

Content	Stretch Axis	form	lspace	rspace	properties
⨼ U+2A27	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⨽ U+2A28	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⨾ U+2A29	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⨿ U+2A2A	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⩀ U+2A2B	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⩁ U+2A2C	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⩂ U+2A2D	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⩃ U+2A2E	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⩄ U+2A38	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⩅ U+2A39	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⩆ U+2A3A	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⩇ U+2A3E	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⩈ U+2A40	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⩉ U+2A41	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⩊ U+2A42	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⩋ U+2A43	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⩌ U+2A44	block	infix	0.2222222222222222em	0.2222222222222222em	N/A

Content	Stretch Axis	form	lspace	rspace	properties
⩅ U+2A45	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⩆ U+2A46	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⩇ U+2A47	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⩈ U+2A48	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⩉ U+2A49	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⩊ U+2A4A	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⩋ U+2A4B	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⩌ U+2A4C	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⩍ U+2A4D	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⩎ U+2A4E	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⩏ U+2A4F	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⩑ U+2A51	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⩒ U+2A52	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⩓ U+2A53	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⩔ U+2A54	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⩕ U+2A55	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⩖ U+2A56	block	infix	0.2222222222222222em	0.2222222222222222em	N/A

Content	Stretch Axis	form	lspace	rspace	properties
⩗ U+2A57	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⩘ U+2A58	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⩙ U+2A59	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⩚ U+2A5A	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⩛ U+2A5B	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⩜ U+2A5C	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⩝ U+2A5D	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⩞ U+2A5E	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⩟ U+2A5F	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⩠ U+2A60	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⩡ U+2A61	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⩢ U+2A62	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⩣ U+2A63	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⫛ U+2ADB	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⫶ U+2AF6	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⫻ U+2AFB	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
⫽ U+2AFD	block	infix	0.2222222222222222em	0.2222222222222222em	N/A

Content	Stretch Axis	form	lspace	rspace	properties
String && U+0026 U+0026	block	infix	0.2222222222222222em	0.2222222222222222em	N/A
% U+0025	block	infix	0.16666666666666666em	0.16666666666666666em	N/A
* U+002A	block	infix	0.16666666666666666em	0.16666666666666666em	N/A
. U+002E	block	infix	0.16666666666666666em	0.16666666666666666em	N/A
? U+003F	block	infix	0.16666666666666666em	0.16666666666666666em	N/A
@ U+0040	block	infix	0.16666666666666666em	0.16666666666666666em	N/A
^ U+005E	inline	infix	0.16666666666666666em	0.16666666666666666em	N/A
· U+00B7	block	infix	0.16666666666666666em	0.16666666666666666em	N/A
× U+00D7	block	infix	0.16666666666666666em	0.16666666666666666em	N/A
• U+2022	block	infix	0.16666666666666666em	0.16666666666666666em	N/A
‐ U+2043	block	infix	0.16666666666666666em	0.16666666666666666em	N/A
∗ U+2217	block	infix	0.16666666666666666em	0.16666666666666666em	N/A
∘ U+2218	block	infix	0.16666666666666666em	0.16666666666666666em	N/A
∙ U+2219	block	infix	0.16666666666666666em	0.16666666666666666em	N/A
≀ U+2240	block	infix	0.16666666666666666em	0.16666666666666666em	N/A
⊗ U+2297	block	infix	0.16666666666666666em	0.16666666666666666em	N/A
⊙ U+2299	block	infix	0.16666666666666666em	0.16666666666666666em	N/A
⊙ U+229A	block	infix	0.16666666666666666em	0.16666666666666666em	N/A
⊛ U+229B	block	infix	0.16666666666666666em	0.16666666666666666em	N/A
⊠ U+22A0	block	infix	0.16666666666666666em	0.16666666666666666em	N/A

Content	Stretch Axis	form	lspace	rspace	properties
⊡ U+22A1	block	infix	0.1666666666666666em	0.1666666666666666em	N/A
⊺ U+22BA	block	infix	0.1666666666666666em	0.1666666666666666em	N/A
⋄ U+22C4	block	infix	0.1666666666666666em	0.1666666666666666em	N/A
⋅ U+22C5	block	infix	0.1666666666666666em	0.1666666666666666em	N/A
⋆ U+22C6	block	infix	0.1666666666666666em	0.1666666666666666em	N/A
⋇ U+22C7	block	infix	0.1666666666666666em	0.1666666666666666em	N/A
⋉ U+22C9	block	infix	0.1666666666666666em	0.1666666666666666em	N/A
⋊ U+22CA	block	infix	0.1666666666666666em	0.1666666666666666em	N/A
⋋ U+22CB	block	infix	0.1666666666666666em	0.1666666666666666em	N/A
⋌ U+22CC	block	infix	0.1666666666666666em	0.1666666666666666em	N/A
⌅ U+2305	block	infix	0.1666666666666666em	0.1666666666666666em	N/A
⍱ U+2306	block	infix	0.1666666666666666em	0.1666666666666666em	N/A
⟋ U+27CB	block	infix	0.1666666666666666em	0.1666666666666666em	N/A
⟍ U+27CD	block	infix	0.1666666666666666em	0.1666666666666666em	N/A
⧆ U+29C6	block	infix	0.1666666666666666em	0.1666666666666666em	N/A
⧇ U+29C7	block	infix	0.1666666666666666em	0.1666666666666666em	N/A
⧈ U+29C8	block	infix	0.1666666666666666em	0.1666666666666666em	N/A
⧔ U+29D4	block	infix	0.1666666666666666em	0.1666666666666666em	N/A

Content	Stretch Axis	form	lspace	rspace	properties
⤝ U+29D5	block	infix	0.16666666666666666em	0.16666666666666666em	N/A
⧖ U+29D6	block	infix	0.16666666666666666em	0.16666666666666666em	N/A
⧗ U+29D7	block	infix	0.16666666666666666em	0.16666666666666666em	N/A
⩢ U+29E2	block	infix	0.16666666666666666em	0.16666666666666666em	N/A
⋈ U+2A1D	block	infix	0.16666666666666666em	0.16666666666666666em	N/A
⟑ U+2A1E	block	infix	0.16666666666666666em	0.16666666666666666em	N/A
⨯ U+2A2F	block	infix	0.16666666666666666em	0.16666666666666666em	N/A
⨰ U+2A30	block	infix	0.16666666666666666em	0.16666666666666666em	N/A
⨱ U+2A31	block	infix	0.16666666666666666em	0.16666666666666666em	N/A
⨲ U+2A32	block	infix	0.16666666666666666em	0.16666666666666666em	N/A
⨳ U+2A33	block	infix	0.16666666666666666em	0.16666666666666666em	N/A
⨴ U+2A34	block	infix	0.16666666666666666em	0.16666666666666666em	N/A
⨵ U+2A35	block	infix	0.16666666666666666em	0.16666666666666666em	N/A
⨶ U+2A36	block	infix	0.16666666666666666em	0.16666666666666666em	N/A
⨷ U+2A37	block	infix	0.16666666666666666em	0.16666666666666666em	N/A
⨻ U+2A3B	block	infix	0.16666666666666666em	0.16666666666666666em	N/A
⨼ U+2A3C	block	infix	0.16666666666666666em	0.16666666666666666em	N/A

Content	Stretch Axis	form	lspace	rspace	properties
⌐ U+2A3D	block	infix	0.1666666666666666em	0.1666666666666666em	N/A
⩏ U+2A3F	block	infix	0.1666666666666666em	0.1666666666666666em	N/A
⩐ U+2A50	block	infix	0.1666666666666666em	0.1666666666666666em	N/A
⪤ U+2A64	block	infix	0.1666666666666666em	0.1666666666666666em	N/A
⪥ U+2A65	block	infix	0.1666666666666666em	0.1666666666666666em	N/A
⪼ U+2ADC	block	infix	0.1666666666666666em	0.1666666666666666em	N/A
⫝ U+2ADD	block	infix	0.1666666666666666em	0.1666666666666666em	N/A
⫾ U+2AFE	block	infix	0.1666666666666666em	0.1666666666666666em	N/A
String ** U+002A U+002A	block	infix	0.1666666666666666em	0.1666666666666666em	N/A
String <> U+003C U+003E	block	infix	0.1666666666666666em	0.1666666666666666em	N/A
! U+0021	block	prefix	0	0	N/A
+ U+002B	block	prefix	0	0	N/A
- U+002D	block	prefix	0	0	N/A
¬ U+00AC	block	prefix	0	0	N/A
± U+00B1	block	prefix	0	0	N/A
‘ U+2018	block	prefix	0	0	fence
“ U+201C	block	prefix	0	0	fence
∀ U+2200	block	prefix	0	0	N/A

Content	Stretch Axis	form	lspace	rspace	properties
∁ U+2201	block	prefix	0	0	N/A
∃ U+2203	block	prefix	0	0	N/A
∄ U+2204	block	prefix	0	0	N/A
∇ U+2207	block	prefix	0	0	N/A
− U+2212	block	prefix	0	0	N/A
∓ U+2213	block	prefix	0	0	N/A
∟ U+221F	block	prefix	0	0	N/A
∠ U+2220	block	prefix	0	0	N/A
∡ U+2221	block	prefix	0	0	N/A
∢ U+2222	block	prefix	0	0	N/A
∴ U+2234	block	prefix	0	0	N/A
∵ U+2235	block	prefix	0	0	N/A
∼ U+223C	block	prefix	0	0	N/A
⊾ U+22BE	block	prefix	0	0	N/A
⊿ U+22BF	block	prefix	0	0	N/A
⌐ U+2310	block	prefix	0	0	N/A
⌟ U+2319	block	prefix	0	0	N/A
✚ U+2795	block	prefix	0	0	N/A
➖ U+2796	block	prefix	0	0	N/A

Content	Stretch Axis	form	lspace	rspace	properties
⟀ U+27C0	block	prefix	0	0	N/A
⦛ U+299B	block	prefix	0	0	N/A
⦜ U+299C	block	prefix	0	0	N/A
⦝ U+299D	block	prefix	0	0	N/A
⦞ U+299E	block	prefix	0	0	N/A
⦟ U+299F	block	prefix	0	0	N/A
⦠ U+29A0	block	prefix	0	0	N/A
⦡ U+29A1	block	prefix	0	0	N/A
⦢ U+29A2	block	prefix	0	0	N/A
⦣ U+29A3	block	prefix	0	0	N/A
⦤ U+29A4	block	prefix	0	0	N/A
⦥ U+29A5	block	prefix	0	0	N/A
⦦ U+29A6	block	prefix	0	0	N/A
⦧ U+29A7	block	prefix	0	0	N/A
⦨ U+29A8	block	prefix	0	0	N/A
⦩ U+29A9	block	prefix	0	0	N/A
⦪ U+29AA	block	prefix	0	0	N/A

Content	Stretch Axis	form	lspace	rspace	properties
⪫ U+29AB	block	prefix	0	0	N/A
⪬ U+29AC	block	prefix	0	0	N/A
⪭ U+29AD	block	prefix	0	0	N/A
⪮ U+29AE	block	prefix	0	0	N/A
⪯ U+29AF	block	prefix	0	0	N/A
⫬ U+2AEC	block	prefix	0	0	N/A
⫭ U+2AED	block	prefix	0	0	N/A
String ‖ U+007C U+007C	block	prefix	0	0	fence
! U+0021	block	postfix	0	0	N/A
" U+0022	block	postfix	0	0	N/A
% U+0025	block	postfix	0	0	N/A
& U+0026	block	postfix	0	0	N/A
' U+0027	block	postfix	0	0	N/A
` U+0060	block	postfix	0	0	N/A
¨ U+00A8	block	postfix	0	0	N/A
° U+00B0	block	postfix	0	0	N/A
² U+00B2	block	postfix	0	0	N/A
³ U+00B3	block	postfix	0	0	N/A
´ U+00B4	block	postfix	0	0	N/A
¸ U+00B8	block	postfix	0	0	N/A
¹ U+00B9	block	postfix	0	0	N/A

Content	Stretch Axis	form	lspace	rspace	properties
´ U+02CA	block	postfix	0	0	N/A
` U+02CB	block	postfix	0	0	N/A
˘ U+02D8	block	postfix	0	0	N/A
˙ U+02D9	block	postfix	0	0	N/A
˚ U+02DA	block	postfix	0	0	N/A
˝ U+02DD	block	postfix	0	0	N/A
̑ U+0311	block	postfix	0	0	N/A
’ U+2019	block	postfix	0	0	fence
‚ U+201A	block	postfix	0	0	N/A
‛ U+201B	block	postfix	0	0	N/A
” U+201D	block	postfix	0	0	fence
„ U+201E	block	postfix	0	0	N/A
‟ U+201F	block	postfix	0	0	N/A
′ U+2032	block	postfix	0	0	N/A
″ U+2033	block	postfix	0	0	N/A
‴ U+2034	block	postfix	0	0	N/A
‵ U+2035	block	postfix	0	0	N/A
‶ U+2036	block	postfix	0	0	N/A
‷ U+2037	block	postfix	0	0	N/A
⁗ U+2057	block	postfix	0	0	N/A
⃜ U+20DB	block	postfix	0	0	N/A
⃜ U+20DC	block	postfix	0	0	N/A
⏍ U+23CD	block	postfix	0	0	N/A

Content	Stretch Axis	form	lspace	rspace	properties
String !! U+0021 U+0021	block	postfix	0	0	N/A
String ++ U+002B U+002B	block	postfix	0	0	N/A
String -- U+002D U+002D	block	postfix	0	0	N/A
String ‖ U+007C U+007C	block	postfix	0	0	fence
(U+0028	block	prefix	0	0	stretchy symmetric fence
[U+005B	block	prefix	0	0	stretchy symmetric fence
{ U+007B	block	prefix	0	0	stretchy symmetric fence
‖ U+007C	block	prefix	0	0	stretchy symmetric fence
‖ U+2016	block	prefix	0	0	stretchy symmetric fence
⌈ U+2308	block	prefix	0	0	stretchy symmetric fence
⌊ U+230A	block	prefix	0	0	stretchy symmetric fence
⟨ U+2329	block	prefix	0	0	stretchy symmetric

Content	Stretch Axis	form	lspace	rspace	properties
					fence
⦇ U+2772	block	prefix	0	0	stretchy symmetric fence
⟦ U+27E6	block	prefix	0	0	stretchy symmetric fence
⟨ U+27E8	block	prefix	0	0	stretchy symmetric fence
⟪ U+27EA	block	prefix	0	0	stretchy symmetric fence
⟬ U+27EC	block	prefix	0	0	stretchy symmetric fence
⟮ U+27EE	block	prefix	0	0	stretchy symmetric fence
⦘ U+2980	block	prefix	0	0	stretchy symmetric fence
⦃ U+2983	block	prefix	0	0	stretchy symmetric fence
⦅ U+2985	block	prefix	0	0	stretchy symmetric fence
⦇ U+2987	block	prefix	0	0	stretchy symmetric fence
⦉ U+2989	block	prefix	0	0	stretchy symmetric fence

Content	Stretch Axis	form	lspace	rspace	properties
⦋ U+298B	block	prefix	0	0	stretchy symmetric fence
⦍ U+298D	block	prefix	0	0	stretchy symmetric fence
⦏ U+298F	block	prefix	0	0	stretchy symmetric fence
⦑ U+2991	block	prefix	0	0	stretchy symmetric fence
⦓ U+2993	block	prefix	0	0	stretchy symmetric fence
⦕ U+2995	block	prefix	0	0	stretchy symmetric fence
⦗ U+2997	block	prefix	0	0	stretchy symmetric fence
⦙ U+2999	block	prefix	0	0	stretchy symmetric fence
⧘ U+29D8	block	prefix	0	0	stretchy symmetric fence
⧚ U+29DA	block	prefix	0	0	stretchy symmetric fence
⧼ U+29FC	block	prefix	0	0	stretchy symmetric fence
) U+0029	block	postfix	0	0	stretchy symmetric

Content	Stretch Axis	form	lspace	rspace	properties
					fence
] U+005D	block	postfix	0	0	stretchy symmetric fence
| U+007C	block	postfix	0	0	stretchy symmetric fence
} U+007D	block	postfix	0	0	stretchy symmetric fence
‖ U+2016	block	postfix	0	0	stretchy symmetric fence
⌉ U+2309	block	postfix	0	0	stretchy symmetric fence
⌋ U+230B	block	postfix	0	0	stretchy symmetric fence
〉 U+232A	block	postfix	0	0	stretchy symmetric fence
❳ U+2773	block	postfix	0	0	stretchy symmetric fence
⟧ U+27E7	block	postfix	0	0	stretchy symmetric fence
⟩ U+27E9	block	postfix	0	0	stretchy symmetric fence
⟫ U+27EB	block	postfix	0	0	stretchy symmetric fence

Content	Stretch Axis	form	lspace	rspace	properties
⟫ U+27ED	block	postfix	0	0	stretchy symmetric fence
⟯ U+27EF	block	postfix	0	0	stretchy symmetric fence
⦄ U+2980	block	postfix	0	0	stretchy symmetric fence
⦄ U+2984	block	postfix	0	0	stretchy symmetric fence
⦆ U+2986	block	postfix	0	0	stretchy symmetric fence
⦈ U+2988	block	postfix	0	0	stretchy symmetric fence
⦊ U+298A	block	postfix	0	0	stretchy symmetric fence
⦌ U+298C	block	postfix	0	0	stretchy symmetric fence
⦎ U+298E	block	postfix	0	0	stretchy symmetric fence
⦐ U+2990	block	postfix	0	0	stretchy symmetric fence
⦒ U+2992	block	postfix	0	0	stretchy symmetric fence
⦔ U+2994	block	postfix	0	0	stretchy symmetric

Content	Stretch Axis	form	lspace	rspace	properties
					fence
⧛ U+2996	block	postfix	0	0	stretchy symmetric fence
⦆ U+2998	block	postfix	0	0	stretchy symmetric fence
⦙ U+2999	block	postfix	0	0	stretchy symmetric fence
⧙ U+29D9	block	postfix	0	0	stretchy symmetric fence
⧛ U+29DB	block	postfix	0	0	stretchy symmetric fence
⟩ U+29FD	block	postfix	0	0	stretchy symmetric fence
∫ U+222B	block	prefix	0.1666666666666666em	0.1666666666666666em	symmetric largeop
∬ U+222C	block	prefix	0.1666666666666666em	0.1666666666666666em	symmetric largeop
∭ U+222D	block	prefix	0.1666666666666666em	0.1666666666666666em	symmetric largeop
∮ U+222E	block	prefix	0.1666666666666666em	0.1666666666666666em	symmetric largeop
∯ U+222F	block	prefix	0.1666666666666666em	0.1666666666666666em	symmetric largeop
∰ U+2230	block	prefix	0.1666666666666666em	0.1666666666666666em	symmetric largeop
∱ U+2231	block	prefix	0.1666666666666666em	0.1666666666666666em	symmetric largeop

Content	Stretch Axis	form	lspace	rspace	properties
∲ U+2232	block	prefix	0.1666666666666666em	0.1666666666666666em	symmetric largeop
∳ U+2233	block	prefix	0.1666666666666666em	0.1666666666666666em	symmetric largeop
⨋ U+2A0B	block	prefix	0.1666666666666666em	0.1666666666666666em	symmetric largeop
⨌ U+2A0C	block	prefix	0.1666666666666666em	0.1666666666666666em	symmetric largeop
⨍ U+2A0D	block	prefix	0.1666666666666666em	0.1666666666666666em	symmetric largeop
⨎ U+2A0E	block	prefix	0.1666666666666666em	0.1666666666666666em	symmetric largeop
⨏ U+2A0F	block	prefix	0.1666666666666666em	0.1666666666666666em	symmetric largeop
⨐ U+2A10	block	prefix	0.1666666666666666em	0.1666666666666666em	symmetric largeop
⨑ U+2A11	block	prefix	0.1666666666666666em	0.1666666666666666em	symmetric largeop
⨒ U+2A12	block	prefix	0.1666666666666666em	0.1666666666666666em	symmetric largeop
⨓ U+2A13	block	prefix	0.1666666666666666em	0.1666666666666666em	symmetric largeop
⨔ U+2A14	block	prefix	0.1666666666666666em	0.1666666666666666em	symmetric largeop
⨕ U+2A15	block	prefix	0.1666666666666666em	0.1666666666666666em	symmetric largeop
⨖ U+2A16	block	prefix	0.1666666666666666em	0.1666666666666666em	symmetric largeop
⨗ U+2A17	block	prefix	0.1666666666666666em	0.1666666666666666em	symmetric largeop
⨘ U+2A18	block	prefix	0.1666666666666666em	0.1666666666666666em	symmetric largeop
⨙ U+2A19	block	prefix	0.1666666666666666em	0.1666666666666666em	symmetric largeop

Content	Stretch Axis	form	lspace	rspace	properties
⨑ U+2A1A	block	prefix	0.1666666666666666em	0.1666666666666666em	symmetric largeop
⨒ U+2A1B	block	prefix	0.1666666666666666em	0.1666666666666666em	symmetric largeop
⨓ U+2A1C	block	prefix	0.1666666666666666em	0.1666666666666666em	symmetric largeop
^ U+005E	inline	postfix	0	0	stretchy
_ U+005F	inline	postfix	0	0	stretchy
~ U+007E	inline	postfix	0	0	stretchy
¯ U+00AF	inline	postfix	0	0	stretchy
ˆ U+02C6	inline	postfix	0	0	stretchy
ˇ U+02C7	inline	postfix	0	0	stretchy
ˉ U+02C9	inline	postfix	0	0	stretchy
ˍ U+02CD	inline	postfix	0	0	stretchy
˜ U+02DC	inline	postfix	0	0	stretchy
˷ U+02F7	inline	postfix	0	0	stretchy
̂ U+0302	inline	postfix	0	0	stretchy
‾ U+203E	inline	postfix	0	0	stretchy
⏜ U+2322	inline	postfix	0	0	stretchy
⏝ U+2323	inline	postfix	0	0	stretchy
⏞ U+23B4	inline	postfix	0	0	stretchy
⏟ U+23B5	inline	postfix	0	0	stretchy
⏠ U+23DC	inline	postfix	0	0	stretchy

Content	Stretch Axis	form	lspace	rspace	properties
⏟ U+23DD	inline	postfix	0	0	stretchy
⏠ U+23DE	inline	postfix	0	0	stretchy
⏡ U+23DF	inline	postfix	0	0	stretchy
⏜ U+23E0	inline	postfix	0	0	stretchy
⏝ U+23E1	inline	postfix	0	0	stretchy
﴾ U+1EEF0	inline	postfix	0	0	stretchy
﴿ U+1EEF1	inline	postfix	0	0	stretchy
∏ U+220F	block	prefix	0.1666666666666666em	0.1666666666666666em	symmetric largeop movablelimits
∐ U+2210	block	prefix	0.1666666666666666em	0.1666666666666666em	symmetric largeop movablelimits
∑ U+2211	block	prefix	0.1666666666666666em	0.1666666666666666em	symmetric largeop movablelimits
⋀ U+22C0	block	prefix	0.1666666666666666em	0.1666666666666666em	symmetric largeop movablelimits
⋁ U+22C1	block	prefix	0.1666666666666666em	0.1666666666666666em	symmetric largeop movablelimits
⋂ U+22C2	block	prefix	0.1666666666666666em	0.1666666666666666em	symmetric largeop movablelimits
⋃ U+22C3	block	prefix	0.1666666666666666em	0.1666666666666666em	symmetric largeop

Content	Stretch Axis	form	lspace	rspace	properties
					movablelimits
⨀ U+2A00	block	prefix	0.1666666666666666em	0.1666666666666666em	symmetric largeop movablelimits
⨁ U+2A01	block	prefix	0.1666666666666666em	0.1666666666666666em	symmetric largeop movablelimits
⨂ U+2A02	block	prefix	0.1666666666666666em	0.1666666666666666em	symmetric largeop movablelimits
⨃ U+2A03	block	prefix	0.1666666666666666em	0.1666666666666666em	symmetric largeop movablelimits
⨄ U+2A04	block	prefix	0.1666666666666666em	0.1666666666666666em	symmetric largeop movablelimits
⨅ U+2A05	block	prefix	0.1666666666666666em	0.1666666666666666em	symmetric largeop movablelimits
⨆ U+2A06	block	prefix	0.1666666666666666em	0.1666666666666666em	symmetric largeop movablelimits
⨇ U+2A07	block	prefix	0.1666666666666666em	0.1666666666666666em	symmetric largeop movablelimits
⨈ U+2A08	block	prefix	0.1666666666666666em	0.1666666666666666em	symmetric largeop movablelimits
⨉ U+2A09	block	prefix	0.1666666666666666em	0.1666666666666666em	symmetric largeop movablelimits
⨊ U+2A0A	block	prefix	0.1666666666666666em	0.1666666666666666em	symmetric largeop movablelimits

Content	Stretch Axis	form	lspace	rspace	properties
⋈ U+2A1D	block	prefix	0.1666666666666666em	0.1666666666666666em	symmetric largeop movablelimits
⩞ U+2A1E	block	prefix	0.1666666666666666em	0.1666666666666666em	symmetric largeop movablelimits
⫼ U+2AFC	block	prefix	0.1666666666666666em	0.1666666666666666em	symmetric largeop movablelimits
⫿ U+2AFF	block	prefix	0.1666666666666666em	0.1666666666666666em	symmetric largeop movablelimits
/ U+002F	block	infix	0	0	N/A
\ U+005C	block	infix	0	0	N/A
_ U+005F	inline	infix	0	0	N/A
U+2061	block	infix	0	0	N/A
U+2062	block	infix	0	0	N/A
U+2063	block	infix	0	0	separator
U+2064	block	infix	0	0	N/A
Δ U+2206	block	infix	0	0	N/A
ⅅ U+2145	block	prefix	0.1666666666666666em	0	N/A
ⅆ U+2146	block	prefix	0.1666666666666666em	0	N/A
∂ U+2202	block	prefix	0.1666666666666666em	0	N/A
√ U+221A	block	prefix	0.1666666666666666em	0	N/A
∛ U+221B	block	prefix	0.1666666666666666em	0	N/A
∜ U+221C	block	prefix	0.1666666666666666em	0	N/A
, U+002C	block	infix	0	0.1666666666666666em	separator
: U+003A	block	infix	0	0.1666666666666666em	N/A
; U+003B	block	infix	0	0.1666666666666666em	separator

Figure 29 Mapping from operator (Content, Form) to properties.
Total size: 1177 entries, ≥ 3679 bytes
(assuming 'Content' uses at least one UTF-16 character, 'Stretch Axis' 1 bit, 'Form' 2 bits, the different combinations of 'rspace' and 'space' at least 3 bits, and the different combinations of properties 3 bits).

B.3 Combining Character Equivalences

This section is non-normative.

The following table gives mappings between spacing and non spacing characters when used in MathML accent constructs.

Combining

Non Combining		Style	Combining
U+002B	plus sign	below	U+031F combining plus sign below
U+002D	hyphen-minus	above	U+0305 combining overline
U+002D	hyphen-minus	below	U+0320 combining minus sign below
U+002D	hyphen-minus	below	U+0332 combining low line
U+002E	full stop	above	U+0307 combining dot above
U+002E	full stop	below	U+0323 combining dot below
U+005E	circumflex accent	above	U+0302 combining circumflex accent
U+005E	circumflex accent	below	U+032D combining circumflex accent below
U+005F	low line	below	U+0332 combining low line
U+0060	grave accent	above	U+0300 combining grave accent
U+0060	grave accent	below	U+0316 combining grave accent below
U+007E	tilde	above	U+0303 combining tilde
U+007E	tilde	below	U+0330 combining tilde below
U+00A8	diaeresis	above	U+0308 combining diaeresis
U+00A8	diaeresis	below	U+0324 combining diaeresis below
U+00AF	macron	above	U+0304 combining macron
U+00AF	macron	above	U+0305 combining overline

Non Combining	Style	Combining
U+00B4 acute accent	above	U+0301 combining acute accent
U+00B4 acute accent	below	U+0317 combining acute accent below
U+00B8 cedilla	below	U+0327 combining cedilla
U+02C6 modifier letter circumflex accent	above	U+0302 combining circumflex accent
U+02C7 caron	above	U+030C combining caron
U+02C7 caron	below	U+032C combining caron below
U+02D8 breve	above	U+0306 combining breve
U+02D8 breve	below	U+032E combining breve below
U+02D9 dot above	above	U+0307 combining dot above
U+02D9 dot above	below	U+0323 combining dot below
U+02DB ogonek	below	U+0328 combining ogonek
U+02DC small tilde	above	U+0303 combining tilde
U+02DC small tilde	below	U+0330 combining tilde below
U+02DD double acute accent	above	U+030B combining double acute accent
U+203E overline	above	U+0305 combining overline
U+2190 leftwards arrow	above	U+20D6
U+2192 rightwards arrow	above	U+20D7 combining right arrow above
U+2192 rightwards arrow	above	U+20EF combining right arrow below
U+2212 minus sign	above	U+0305 combining overline
U+2212 minus sign	below	U+0332 combining low line
U+27F6 long rightwards arrow	above	U+20D7 combining right arrow above
U+27F6 long rightwards arrow	above	U+20EF combining right arrow below

§ Non Combining

Combining	Style	Non Combining
U+0300 combining grave accent	above	U+0060 grave accent
U+0301 combining acute accent	above	U+00B4 acute accent
U+0302 combining circumflex accent	above	U+005E circumflex accent
U+0302 combining circumflex accent	above	U+02C6 modifier letter circumflex accent
U+0303 combining tilde	above	U+007E tilde
U+0303 combining tilde	above	U+02DC small tilde

Combining		Style	Non Combining	
U+0304	combining macron	above	U+00AF	macron
U+0305	combining overline	above	U+002D	hyphen-minus
U+0305	combining overline	above	U+00AF	macron
U+0305	combining overline	above	U+203E	overline
U+0305	combining overline	above	U+2212	minus sign
U+0306	combining breve	above	U+02D8	breve
U+0307	combining dot above	above	U+02E	
U+0307	combining dot above	above	U+002E	full stop
U+0307	combining dot above	above	U+02D9	dot above
U+0308	combining diaeresis	above	U+00A8	diaeresis
U+030B	combining double acute accent	above	U+02DD	double acute accent
U+030C	combining caron	above	U+02C7	caron
U+0312	combining turned comma above	above	U+0B8	
U+0316	combining grave accent below	below	U+0060	grave accent
U+0317	combining acute accent below	below	U+00B4	acute accent
U+031F	combining plus sign below	below	U+002B	plus sign
U+0320	combining minus sign below	below	U+002D	hyphen-minus
U+0323	combining dot below	below	U+002E	full stop
U+0323	combining dot below	below	U+02D9	dot above
U+0324	combining diaeresis below	below	U+00A8	diaeresis
U+0327	combining cedilla	below	U+00B8	cedilla
U+0328	combining ogonek	below	U+02DB	ogonek
U+032C	combining caron below	below	U+02C7	caron
U+032D	combining circumflex accent below	below	U+005E	circumflex accent
U+032E	combining breve below	below	U+02D8	breve
U+0330	combining tilde below	below	U+007E	tilde
U+0330	combining tilde below	below	U+02DC	small tilde
U+0332	combining low line	below	U+002D	hyphen-minus
U+0332	combining low line	below	U+005F	low line
U+0332	combining low line	below	U+2212	minus sign
U+0338	combining long solidus overlay	over	U+02F	
U+20D7	combining right arrow above	above	U+2192	rightwards arrow

Combining	Style	Non Combining
U+20D7 combining right arrow above	above	U+27F6 long rightwards arrow
U+20EF combining right arrow below	above	U+2192 rightwards arrow
U+20EF combining right arrow below	above	U+27F6 long rightwards arrow

B.4 Unicode-based Glyph Assemblies

This section is non-normative.

The following table provides fallback that user agents may use for stretching a given *base character* when the font does not provide a `MATH.MathVariants` table. The algorithms of 5.3 Size variants for operators (`MathVariants`) work the same except with some adjustments:

- Entries are indexed by the base character.
- All the glyph IDs and metrics have to be deduced from Unicode code points.
- If the *glyph construction* is horizontal then the entry corresponds to a `MathVariants.horizGlyphConstructionOffsets[]` item; if it is vertical it corresponds to a `MathVariants.vertGlyphConstructionOffsets[]` item.
- The `MathGlyphConstruction.mathGlyphVariantRecord` is always empty.
- The `MathVariants.minConnectorOverlap`, `GlyphPartRecord.startConnectorLength` and `GlyphPartRecord.endConnectorLength` are treated as 0.
- The array of `MathGlyphConstruction.GlyphAssembly.partRecords` is built from each table row as follows:
  1. A (non-extender) *bottom/left* character
  2. Followed by an *extender* character.
  3. Optionally followed by this:
     - Optionally, a (non-extender) *middle* character and the same extender character previously mentioned.
     - A (non-extender) *top/right* character.

Base Character	Glyph Construction	Extender Character	Bottom/Left Character	Middle Character	Top/Right Character
U+0028 (	Vertical	U+239C ⎜	U+239D ⎝	N/A	U+239B ⎛

Base Character	Glyph Construction	Extender Character	Bottom/Left Character	Middle Character	Top/Right Character
U+0029)	Vertical	U+239F ⎟	U+23A0 ⎠	N/A	U+239E ⎞
U+003D =	Horizontal	U+003D =	U+003D =	N/A	N/A
U+005B [	Vertical	U+23A2 ⎢	U+23A3 ⎣	N/A	U+23A1 ⎡
U+005D]	Vertical	U+23A5 ⎥	U+23A6 ⎦	N/A	U+23A4 ⎤
U+005F _	Horizontal	U+005F _	U+005F _	N/A	N/A
U+007B {	Vertical	U+23AA ⎪	U+23A9 ⎩	U+23A8 ⎨	U+23A7 ⎧
U+007C ∣	Vertical	U+007C ∣	U+007C ∣	N/A	N/A
U+007D }	Vertical	U+23AA ⎪	U+23AD ⎭	U+23AC ⎬	U+23AB ⎫
U+00AF ¯	Horizontal	U+00AF ¯	U+00AF ¯	N/A	N/A
U+2016 ‖	Vertical	U+2016 ‖	U+2016 ‖	N/A	N/A
U+203E ‾	Horizontal	U+203E ‾	U+203E ‾	N/A	N/A
U+2190 ←	Horizontal	U+23AF ⎯	U+2190 ←	N/A	U+23AF ⎯
U+2191 ↑	Vertical	U+23D0 ⏐	U+23D0 ⏐	N/A	U+2191 ↑
U+2192 →	Horizontal	U+23AF ⎯	U+23AF ⎯	N/A	U+2192 →
U+2193 ↓	Vertical	U+23D0 ⏐	U+2193 ↓	N/A	U+23D0 ⏐
U+2194 ↔	Horizontal	U+23AF ⎯	U+2190 ←	N/A	U+2192 →
U+2195 ↕	Vertical	U+23D0 ⏐	U+2193 ↓	N/A	U+2191 ↑
U+21A4 ↤	Horizontal	U+23AF ⎯	U+2190 ←	N/A	U+22A3 ⊣
U+21A6 ↦	Horizontal	U+23AF ⎯	U+22A2 ⊢	N/A	U+2192 →
U+21BC ↼	Horizontal	U+23AF ⎯	U+21BC ↼	N/A	U+23AF ⎯
U+21BD ↽	Horizontal	U+23AF ⎯	U+21BD ↽	N/A	U+23AF ⎯
U+21C0 ⇀	Horizontal	U+23AF ⎯	U+23AF ⎯	N/A	U+21C0 ⇀
U+21C1 ⇁	Horizontal	U+23AF ⎯	U+23AF ⎯	N/A	U+21C1 ⇁
U+2223 ∣	Vertical	U+2223 ∣	U+2223 ∣	N/A	N/A
U+2225 ∥	Vertical	U+2225 ∥	U+2225 ∥	N/A	N/A
U+2308 ⌈	Vertical	U+23A2 ⎢	U+23A2 ⎢	N/A	U+23A1 ⎡
U+2309 ⌉	Vertical	U+23A5 ⎥	U+23A5 ⎥	N/A	U+23A4 ⎤
U+230A ⌊	Vertical	U+23A2 ⎢	U+23A3 ⎣	N/A	N/A
U+230B ⌋	Vertical	U+23A5 ⎥	U+23A6 ⎦	N/A	N/A
U+23B0 ⎰	Vertical	U+23AA ⎪	U+23AD ⎭	N/A	U+23A7 ⎧
U+23B1 ⎱	Vertical	U+23AA ⎪	U+23A9 ⎩	N/A	U+23AB ⎫

Base Character	Glyph Construction	Extender Character	Bottom/Left Character	Middle Character	Top/Right Character
U+27F5 ⟵	Horizontal	U+23AF ⎯	U+2190 ←	N/A	U+23AF ⎯
U+27F6 ⟶	Horizontal	U+23AF ⎯	U+23AF ⎯	N/A	U+2192 →
U+27F7 ⟷	Horizontal	U+23AF ⎯	U+2190 ←	N/A	U+2192 →
U+294E ⥎	Horizontal	U+23AF ⎯	U+21BC ↼	N/A	U+21C0 ⇀
U+2950 ⥐	Horizontal	U+23AF ⎯	U+21BD ↽	N/A	U+21C1 ⇁
U+295A ⥚	Horizontal	U+23AF ⎯	U+21BC ↼	N/A	U+22A3 ⊣
U+295B ⥛	Horizontal	U+23AF ⎯	U+22A2 ⊢	N/A	U+21C0 ⇀
U+295E ⥞	Horizontal	U+23AF ⎯	U+21BD ↽	N/A	U+22A3 ⊣
U+295F ⥟	Horizontal	U+23AF ⎯	U+22A2 ⊢	N/A	U+21C1 ⇁

C. Mathematical Alphanumeric Symbols

This section is non-normative.

As detailed in [*xml-entity-names*] mathematical alphanumeric symbols with form bold, italic, fraktur, monospace, double-struck etc are available in Unicode.

These alphanumeric symbols should be accessed using their Unicode code points. It is sometimes needed to distinguish between Chancery and Roundhand style for MATHEMATICAL SCRIPT characters. These are notably used in LaTeX for the `\mathcal` and `\mathscr` commands. One way to do that is to rely on Chapter 23.4 Variation Selectors of Unicode which describes a way to specify selection of particular glyph variants [*UNICODE*]. Indeed, the `StandardizedVariants.txt` file from the Unicode Character Database indicates that variant selectors U+FE00 and U+FE01 can be used on capital script to specify Chancery and Roundhand respectively.

Alternatively, some mathematical fonts rely on `salt` or `ssXY` properties from [*OPEN-FONT-FORMAT*] to provide both styles. Page authors may use the font-variant-alternates property with corresponding OpenType font features to access these glyphs.

In addition, the `italic` math alphanumeric characters may be accessed as described above using the CSS `text-transform: math-auto` transform which is applied by default to single character `<mi>` elements. As a convenience the mapping to math italic is shown below.

§ C.1 `italic` mappings

Original	italic	$\Delta_{\text{code point}}$
A U+0041	*A* U+1D434	1D3F3
B U+0042	*B* U+1D435	1D3F3
C U+0043	*C* U+1D436	1D3F3
D U+0044	*D* U+1D437	1D3F3
E U+0045	*E* U+1D438	1D3F3
F U+0046	*F* U+1D439	1D3F3
G U+0047	*G* U+1D43A	1D3F3
H U+0048	*H* U+1D43B	1D3F3
I U+0049	*I* U+1D43C	1D3F3
J U+004A	*J* U+1D43D	1D3F3
K U+004B	*K* U+1D43E	1D3F3
L U+004C	*L* U+1D43F	1D3F3
M U+004D	*M* U+1D440	1D3F3
N U+004E	*N* U+1D441	1D3F3
O U+004F	*O* U+1D442	1D3F3
P U+0050	*P* U+1D443	1D3F3
Q U+0051	*Q* U+1D444	1D3F3
R U+0052	*R* U+1D445	1D3F3
S U+0053	*S* U+1D446	1D3F3
T U+0054	*T* U+1D447	1D3F3
U U+0055	*U* U+1D448	1D3F3
V U+0056	*V* U+1D449	1D3F3
W U+0057	*W* U+1D44A	1D3F3
X U+0058	*X* U+1D44B	1D3F3
Y U+0059	*Y* U+1D44C	1D3F3
Z U+005A	*Z* U+1D44D	1D3F3
a U+0061	*a* U+1D44E	1D3ED
b U+0062	*b* U+1D44F	1D3ED
c U+0063	*c* U+1D450	1D3ED
d U+0064	*d* U+1D451	1D3ED
e U+0065	*e* U+1D452	1D3ED

f U+0066	*f* U+1D453	1D3ED
g U+0067	*g* U+1D454	1D3ED
h U+0068	*ℎ* U+0210E	20A6
i U+0069	*i* U+1D456	1D3ED
j U+006A	*j* U+1D457	1D3ED
k U+006B	*k* U+1D458	1D3ED
l U+006C	*l* U+1D459	1D3ED
m U+006D	*m* U+1D45A	1D3ED
n U+006E	*n* U+1D45B	1D3ED
o U+006F	*o* U+1D45C	1D3ED
p U+0070	*p* U+1D45D	1D3ED
q U+0071	*q* U+1D45E	1D3ED
r U+0072	*r* U+1D45F	1D3ED
s U+0073	*s* U+1D460	1D3ED
t U+0074	*t* U+1D461	1D3ED
u U+0075	*u* U+1D462	1D3ED
v U+0076	*v* U+1D463	1D3ED
w U+0077	*w* U+1D464	1D3ED
x U+0078	*x* U+1D465	1D3ED
y U+0079	*y* U+1D466	1D3ED
z U+007A	*z* U+1D467	1D3ED
ı U+0131	*ı* U+1D6A4	1D573
ȷ U+0237	*ȷ* U+1D6A5	1D46E
Α U+0391	*Α* U+1D6E2	1D351
Β U+0392	*Β* U+1D6E3	1D351
Γ U+0393	*Γ* U+1D6E4	1D351
Δ U+0394	*Δ* U+1D6E5	1D351
Ε U+0395	*Ε* U+1D6E6	1D351
Ζ U+0396	*Ζ* U+1D6E7	1D351
Η U+0397	*Η* U+1D6E8	1D351
Θ U+0398	*Θ* U+1D6E9	1D351
Ι U+0399	*Ι* U+1D6EA	1D351
Κ U+039A	*Κ* U+1D6EB	1D351

Λ	U+039B	Λ U+1D6EC	1D351
M	U+039C	M U+1D6ED	1D351
N	U+039D	N U+1D6EE	1D351
Ξ	U+039E	Ξ U+1D6EF	1D351
O	U+039F	O U+1D6F0	1D351
Π	U+03A0	Π U+1D6F1	1D351
P	U+03A1	P U+1D6F2	1D351
ϴ	U+03F4	$\varTheta$ U+1D6F3	1D2FF
Σ	U+03A3	Σ U+1D6F4	1D351
T	U+03A4	T U+1D6F5	1D351
Υ	U+03A5	Υ U+1D6F6	1D351
Φ	U+03A6	Φ U+1D6F7	1D351
X	U+03A7	X U+1D6F8	1D351
Ψ	U+03A8	Ψ U+1D6F9	1D351
Ω	U+03A9	Ω U+1D6FA	1D351
∇	U+2207	∇ U+1D6FB	1B4F4
α	U+03B1	α U+1D6FC	1D34B
β	U+03B2	β U+1D6FD	1D34B
γ	U+03B3	γ U+1D6FE	1D34B
δ	U+03B4	δ U+1D6FF	1D34B
ε	U+03B5	ε U+1D700	1D34B
ζ	U+03B6	ζ U+1D701	1D34B
η	U+03B7	η U+1D702	1D34B
θ	U+03B8	θ U+1D703	1D34B
ι	U+03B9	ι U+1D704	1D34B
ϰ	U+03BA	κ U+1D705	1D34B
λ	U+03BB	λ U+1D706	1D34B
μ	U+03BC	μ U+1D707	1D34B
ν	U+03BD	ν U+1D708	1D34B
ξ	U+03BE	ξ U+1D709	1D34B
o	U+03BF	o U+1D70A	1D34B
π	U+03C0	π U+1D70B	1D34B
ϱ	U+03C1	ρ U+1D70C	1D34B

ς U+03C2	𝜍 U+1D70D	1D34B
σ U+03C3	𝜎 U+1D70E	1D34B
τ U+03C4	𝜏 U+1D70F	1D34B
υ U+03C5	𝜐 U+1D710	1D34B
ϕ U+03C6	𝜑 U+1D711	1D34B
χ U+03C7	𝜒 U+1D712	1D34B
ψ U+03C8	𝜓 U+1D713	1D34B
ω U+03C9	𝜔 U+1D714	1D34B
∂ U+2202	𝜕 U+1D715	1B513
ϵ U+03F5	𝜖 U+1D716	1D321
ϑ U+03D1	𝜗 U+1D717	1D346
ϰ U+03F0	𝜘 U+1D718	1D328
ϕ U+03D5	𝜙 U+1D719	1D344
ϱ U+03F1	𝜚 U+1D71A	1D329
ϖ U+03D6	𝜛 U+1D71B	1D345

D. Acknowledgments

This section is non-normative.

MathML Core is based on MathML3. See the appendix E of [*MathML3*] for the people that contributed to that specification.

MathML Core was initially developed by the MathML Community Group, and then by the Math Working Group. Working Group or Community Group members who regularly participated in MathML Core meetings during the development of this specification: Brian Kardell, Bruce Miller, Daniel Marques, David Carlisle, David Farmer, Deyan Ginev, Frédéric Wang, Louis Mahler, Moritz Schubotz, Murray Sargent, Neil Soiffer, Patrick Ion, Rob Buis, Steve Noble and Sam Dooley.

In addition, we would like to extend special thanks to Brian Kardell, Neil Soiffer and Rob Buis for help with the editing.

Many thanks also to the following people for their help with the test suite: Brian Kardell, Frédéric Wang, Neil Soiffer and Rob Buis. Several tests are also based on MathML tests from browser repositories and we are grateful to the Mozilla and WebKit contributors.

We would like to thank the people who, through their input and feedback on public communication channels, have helped us with the creation of this specification: André Greiner-Petter, Anne van Kesteren, Boris Zbarsky, Brian Smith, Elika Etemad, Emilio Cobos Álvarez, ExE Boss, Ian Kilpatrick, Koji Ishii, L. David Baron, Michael Kohlhase, Michael Smith, Ryosuke Niwa, Sergey Malkin, Tab Atkins Jr., Viktor Yaffle and frankvel.

E. Security Considerations

This section is non-normative.

This specification adds script execution mechanisms via the MathML event handler attributes described in 2.1.3 Global Attributes. UAs may decide to prevent execution of scripts specified in these attributes, following the same security restrictions as those applying to HTML or SVG elements.

NOTE

In [*MathML3*], it was possible to make any element linkable via `href` or `xlink:href` attributes, with an URL pointing to an untrusted resource or even `javascript:` execution. These attributes are not available in MathML Core. However, as described in 2.2.1 HTML and SVG it is possible to embed HTML or SVG content inside MathML, including HTML or SVG links.

NOTE

In [*MathML3*], it was possible to use the `maction` element with the `actiontype` value set to `"statusline"` in order to override the text of the browser statusline. In particular, an attacker could use this to hide the URL text of an untrusted link e.g.

```
<math>
  <maction actiontype="statusline">
    <mtext><a href="javascript:alert('JS execution')">Click me!</a></mte
    <mtext>./this-is-a-safe-link.html</mtext>
  </maction>
</math>
```

This feature is not available in MathML Core, where the `maction` element essentially behaves like an `mrow` container with extra style.

An attacker can try to hang the UA by inserting very large stretchy operators, effectively making the algorithm shaping of the glyph assembly deal with a huge amount of glyphs. UAs may work around this issue by limiting r_{min} and `GlyphAssembly.partCount` to maximum values.

As described in CSS Fonts Module, an attacker can try to rely on malformed or malicious fonts to exploit potential security faults in browser implementations. Because the OpenType MATH table is used extensively in this specification, UAs should ensure their font sanitization mechanisms are able to deal with that table.

Finally, in order to reduce attack surface, some UAs expose runtime options to disable part of the web platform. Disabling MathML layout can essentially be achieved by forcing elements in the DOM tree to be put in the HTML namespace and disabling 4. CSS Extensions for Math Layout.

F. Privacy Considerations

This section is non-normative.

As explained in 2.2.1 HTML and SVG, MathML can be embedded into an SVG image via the `<foreignObject>` element which can thus be used in a `canvas` element. UA may decide to implement any measure to prevent potential information leakage such as tainting the canvas and returning a "*`SecurityError`*" when one tries to access the canvas' content via JavaScript APIs.

In the following example, the canvas image is set to the image of some MathML content with an HTML link to `https://example.org/`. It should not be possible for an attacker to determine whether that link was visited by reading pixels via `context.getImageData()`. For more about links in MathML, see E. Security Considerations.

```
let svg = `
  <svg xmlns="http://www.w3.org/2000/svg" width="100px" height="100px">
    <foreignObject width="100" height="100"
                   requiredExtensions="http://www.w3.org/1998/Math/MathMl
      <math xmlns="http://www.w3.org/1998/Math/MathML">
        <msqrt style="font-size: 25px">
          <mtext>&#x25a0;</mtext>
          <mtext><a href="https://example.org/">&#x25a0;</a></mtext>
        </msqrt>
      </math>
    </foreignObject>
  </svg>`;
let image = new Image();
image.width = 100;
image.height = 100;
image.onload = () => {
  let canvas = document.createElement('canvas');
  canvas.width = 100;
  canvas.height = 100;
  canvas.style = "border: 1px solid black";
  document.body.appendChild(canvas);
  let context = canvas.getContext("2d");
  context.drawImage(image, 0, 0);
};
image.src = `${window.btoa(svg)}`;
```

This specification describes layout of DOM elements which may involve system fonts. Like for HTML/CSS layout, it is thus possible to use JavaScript APIs (e.g. `context.getImageData()` on content embedded in a canvas context, or even just `getBoundingClientRect()`) to measure box sizes and positions and infer data from system fonts. By combining miscellaneous tests on such fonts and comparing measurements against results of well-known fonts, an attacker can try and determine the default fonts of the user.

The following HTML+CSS+JavaScript document relies on a Web font with exotic metrics to try and determine whether `A Well Known System Font` is available by default.

```
<style>
  @font-face {
    font-family: MyWebFontWithVeryWideGlyphs;
    src: url("/fonts/my-web-fonts-with-very-wide-glyphs.woff");
  }
  #container {
    font-family: AWellKnownSystemFont, MyWebFontWithVeryWideGlyphs;
  }
</style>
<div id="container">SOMETEXT</div>
<div id="reference">SOMETEXT</div>
<script>
document.fonts.ready.then(() => {
  let containerWidth =
    document.getElementById("container").getBoundingClientRect().width;
  let referenceWidth =
    document.getElementById("reference").getBoundingClientRect().width;
  let isWellKnownSystemFontAvailable =
    Math.abs(containerWidth - referenceWidth) < 1;
});
</script>
```

The following HTML+CSS+JavaScript document tries to determine whether the UI serif font provides Asian glyphs:

```
<style>
  @font-face {
    font-family: MyWebFontWithVeryWideAsianGlyphs;
    src: url("/fonts/my-web-fonts-with-very-wide-asian-glyphs.woff");
  }
  #container {
    font-family: ui-serif, MyWebFontWithVeryWideAsianGlyphs
  }
  #reference {
    font-family: MyWebFontWithVeryWideAsianGlyphs;
  }
</style>
<div id="container">王</div>
<div id="reference">王</div>
<script>
document.fonts.ready.then(() => {
  let containerWidth =
    document.getElementById("container").getBoundingClientRect().width;
  let referenceWidth =
    document.getElementById("reference").getBoundingClientRect().width;
  let uiSerifFontDoesNotContainAsianGlyph =
    Math.abs(containerWidth - referenceWidth) < 1;
});
</script>
```

The following HTML+CSS document contains the same text rendered with text-decoration-thickness set to `from-font` and `1em` (here 100 pixels) respectively. By comparing the heights of the two underlines, one can calculate a good approximation of the `underlineThickness` value from the PostScript Table [*OPEN-FONT-FORMAT*].

```
<style>
  #test {
    font-size: 100px;
  }
  #container {
    text-decoration-line: underline;
    text-decoration-thickness: from-font;
  }
  #reference {
    text-decoration-line: underline;
    text-decoration-thickness: 1em;
  }
</style>
<div id="test">
  <div id="container">SOMETEXT</div>
  <div id="reference">SOMETEXT</div>
</div>
```

This specification relies on information from 5. OpenType `MATH` table to render MathML content. One can get good approximation of most layout parameters from `MathConstants` and `MathGlyphInfo` using measurement techniques similar to what is described above for HTML+CSS+JavaScript document. The use of the `MathVariants` table for MathML rendering can also be observed by putting stretchy operators of different sizes inside a `canvas` context.

Although none of these parameters taken individually are personal, implementing this specification increases the set of exposed font information that can be used by an attacker to implement fingerprinting techniques. Typically, they could help determine available and preferred math fonts for a user.

G. Conformance

Conformance requirements are expressed with a combination of descriptive assertions and RFC 2119

terminology. The key words "*MUST*", "*MUST NOT*", "*REQUIRED*", "*SHALL*", "*SHALL NOT*", "*SHOULD*", "*SHOULD NOT*", "*RECOMMENDED*", "*MAY*", and "*OPTIONAL*" in the normative parts of this document are to be interpreted as described in RFC 2119. However, for readability, these words do not appear in all uppercase letters in this specification.

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

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

> This is an example of an informative example.

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

> NOTE
>
> Note, this is an informative note.

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

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

H. References

H.1 Normative references

[CSS-CASCADE-4]
CSS Cascading and Inheritance Level 4. Elika Etemad; Tab Atkins Jr. W3C. 13 January 2022. W3C Candidate Recommendation. URL: https://www.w3.org/TR/css-cascade-4/

[CSS-DISPLAY-3]
CSS Display Module Level 3. Elika Etemad; Tab Atkins Jr. W3C. 30 March 2023. W3C

Candidate Recommendation. URL: https://www.w3.org/TR/css-display-3/

[CSS-POSITION-3]

CSS Positioned Layout Module Level 3. Elika Etemad; Tab Atkins Jr. W3C. 11 March 2025. W3C Working Draft. URL: https://www.w3.org/TR/css-position-3/

[CSS-TEXT-4]

CSS Text Module Level 4. Elika Etemad; Koji Ishii; Alan Stearns; Florian Rivoal. W3C. 29 May 2024. W3C Working Draft. URL: https://www.w3.org/TR/css-text-4/

[CSS-VALUES-4]

CSS Values and Units Module Level 4. Tab Atkins Jr.; Elika Etemad. W3C. 12 March 2024. W3C Working Draft. URL: https://www.w3.org/TR/css-values-4/

[CSS-WRITING-MODES-4]

CSS Writing Modes Level 4. Elika Etemad; Koji Ishii. W3C. 30 July 2019. W3C Candidate Recommendation. URL: https://www.w3.org/TR/css-writing-modes-4/

[CSS2]

Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification. Bert Bos; Tantek Çelik; Ian Hickson; Håkon Wium Lie. W3C. 7 June 2011. W3C Recommendation. URL: https://www.w3.org/TR/CSS2/

[DOM]

DOM Standard. Anne van Kesteren. WHATWG. Living Standard. URL: https://dom.spec.whatwg.org/

[HTML]

HTML Standard. Anne van Kesteren; Domenic Denicola; Dominic Farolino; Ian Hickson; Philip Jägenstedt; Simon Pieters. WHATWG. Living Standard. URL: https://html.spec.whatwg.org/multipage/

[OPEN-FONT-FORMAT]

Information technology — Coding of audio-visual objects — Part 22: Open Font Format. ISO/IEC. January 2019. Published. URL: https://www.iso.org/standard/74461.html

[RFC2119]

Key words for use in RFCs to Indicate Requirement Levels. S. Bradner. IETF. March 1997. Best Current Practice. URL: https://www.rfc-editor.org/rfc/rfc2119

[SELECT]

Selectors Level 3. Tantek Çelik; Elika Etemad; Daniel Glazman; Ian Hickson; Peter Linss; John Williams. W3C. 6 November 2018. W3C Recommendation. URL: https://www.w3.org/TR/selectors-3/

[SVG]

Scalable Vector Graphics (SVG) 1.0 Specification. Jon Ferraiolo. W3C. 4 September 2001. W3C Recommendation. URL: https://www.w3.org/TR/SVG/

H.2 Informative references

[CSS-FONTS-4]
CSS Fonts Module Level 4. Chris Lilley. W3C. 1 February 2024. W3C Working Draft. URL: https://www.w3.org/TR/css-fonts-4/

[CSS-LAYOUT-API-1]
CSS Layout API Level 1. Greg Whitworth; Ian Kilpatrick; Tab Atkins Jr.; Shane Stephens; Robert O'Callahan; Rossen Atanassov. W3C. 12 April 2018. FPWD. URL: https://www.w3.org/TR/css-layout-api-1/

[HOUDINI]
CSS-TAG Houdini Editor Drafts. URL: https://drafts.css-houdini.org/

[MATHML3]
Mathematical Markup Language (MathML) Version 3.0 2nd Edition. David Carlisle; Patrick D F Ion; Robert R Miner. W3C. 10 April 2014. W3C Recommendation. URL: https://www.w3.org/TR/MathML3/

[MATHML4]
Mathematical Markup Language (MathML) Version 4.0. David Carlisle et al.W3C Editor's Draft. URL: https://w3c.github.io/mathml/

[OPEN-TYPE-MATH-ILLUMINATED]
OpenType Math Illuminated. Ulrik Vieth. 2009. URL: https://www.tug.org/TUGboat/tb30-1/tb94vieth.pdf

[OPEN-TYPE-MATH-IN-HARFBUZZ]
OpenType MATH in HarfBuzz. Frédéric Wang. URL: https://frederic-wang.fr/2016/04/16/opentype-math-in-harfbuzz/

[TEXBOOK]
The TeXBook. Knuth, Donald E. Addison-Wesley Professional. 1984.

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

[webidl]
Web IDL Standard. Edgar Chen; Timothy Gu. WHATWG. Living Standard. URL: https://webidl.spec.whatwg.org/

[xml-entity-names]
XML Entity Definitions for Characters (3rd Edition). Patrick D F Ion; David Carlisle. W3C. 7 March 2023. W3C Recommendation. URL: https://www.w3.org/TR/xml-entity-names/

↥