my website: Dr. Carol JVF Burns
(see MathJax in action)

Beautiful math in all browsers

TeX Commands available in MathJax

Korean version of this document

Jump to the alphabetical list of commands

THIS IS A BIG PAGE.
It takes a long time to process (probably about 2–3 minutes).
You can watch the progress in the lower left corner—it loads most reliably if you ***resist the temptation to click on something before it's done***.
I think it's worth the wait (but of course I'm biased).
You can read about why it's so big below.

This document was created in Spring of 2011.
It's processed using the current version of MathJax, via the MathJax Content Distribution Network (CDN).

I (Dr. Carol JVF Burns) have prepared this page to thoroughly familiarize myself with the TeX commands that are available in MathJax,
and to provide a resource that may be useful to other MathJax users.
Davide Cervone, the lead developer of MathJax, has most generously provided extensive edits,
and this page is greatly improved due to his efforts; I owe him countless thanks.

All mistakes on this page are my own (and I welcome suggestions and corrections): **fishcaro@verizon.net**

MathJax allows a syntax modeled on both TeX and LaTeX.
Therefore, web authors can use familiar and concise commands when creating mathematics with MathJax.

Click to show/hide: WHY IS THIS SUCH A BIG PAGE?

Click to show/hide: Getting Started Links

Alphabetical List of TeX Commands available in MathJax

Click to show/hide: Characteristics of the Alphabetical Command Tables

symbols
A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z
environments

Know the *shape* of a character that you want, but not its name? Draw it here!

Depending on your configuration, to get AMSsymbols or AMSmath, you may need to load some extensions in `MathJax.Hub.Config`.
For example:

```
extensions: ["tex2jax.js", "TeX/noErrors.js", "TeX/AMSsymbols.js", "TeX/AMSmath.js"],
```

symbols

#		indicates numbered arguments in definitions Example: `\def\specialFrac#1#2{\frac{x + #1}{y + #2}}` `\specialFrac{7}{z+3}` yields $\frac{x+7}{y+z+3}$

%		used for a single-line comment;
		shows only in the source code;
		does not show in the rendered expression
		Example (showing the math block delimiters):
		`$$`
		`% Note: (x+1)^2 is NOT x^2 + 1`
		`(x+1)^2      % original expression`
		`= (x+1)(x+1) % definition of exponent`
		`= x^2 + 2x + 1 % FOIL, combine like terms`
		`$$`
		yields $(x+1)^2 = (x+1)(x+1) = x^2 + 2x + 1$
		Internet Explorer caution: show/hide more info

&		used as separators in alignment environments; used in HTML entity references within math mode;

		for a literal ampersand, use \&
		Examples:
		`\begin{matrix}`
		`a & b\cr`
		`c & d`
		`\end{matrix}` yields $\begin{matrix} a & b \\ c & d \end{matrix}$
		`a &lt; b` yields $a < b$
		`\text{Carol }\&\text{ Julia}` yields $\text{Carol }\&\text{ Julia}$

^		used to indicate exponents;
		used to indicate superscripts;
		used for limits on large operators and in some ‘vertical’ constructions (see examples)
		`<optional #1> ^ #2`
		argument #1 is optional;
		use braces, as needed, to clarify what is the exponent
		Examples:
		`^i` yields ^{i}
		`x^i_2` yields x^i_2
		`{x^i}_2` yields ${x^i}_2$
		`x^{i_2}` yields x^{i_2}
		`x^{i^2}` yields x^{i^2}
		`{x^i}^2` yields ${x^i}^2$ Note: `x^i^2` yields an error.
		`^ax^b` yields ${}^ax^b$
		`\sum_{n=1}^\infty` yields $\sum_{n=1}^\infty$ (inline mode)
		`\overbrace{x+\cdots+x}`
		`^{n\text{ times}}` yields $\overbrace{x+\cdots+x}^{n\text{ times}}$

_		used to indicate subscripts;
		used for limits on large operators and in some ‘vertical’ constructions (see examples)
		`<optional #1> _ #2`
		argument #1 is optional;
		use braces, as needed, to clarify what is the subscript
		Examples:
		`_2` yields ${}_2$
		`x_i^2` yields x_i^2
		`{x_i}^2` yields ${x_i}^2$
		`x_{i^2}` yields x_{i^2}
		`x_{i_2}` yields x_{i_2}
		`{x_i}_2` yields ${x_i}_2$ Note: `x_i_2` yields an error.
		`^a_bx^c_d` yields ${}^a_bx^c_d$
		`\sum_{n=1}^\infty` yields $\sum_{n=1}^\infty$ (inline mode)
		`\underbrace{x+\cdots+x}`
		`_{n\text{ times}}` yields $\underbrace{x+\cdots+x}_{n\text{ times}}$

{ }		braces, used for grouping; for literal braces, use \{ and \} There are two basic grouping constructs that use braces; I will refer to them as ‘arguments’ versus ‘braced groups’. If you're not aware which construct is in force, then you can get unexpected results. The examples below should clarify. ARGUMENTS: In this documentation, arguments are indicated by #1, #2, etc. An argument is either a single ‘token’ (like ‘a’ or ‘\alpha’), or is a group enclosed in braces. For example, the `\boldsymbol` command takes an argument, notated by: `\boldsymbol #1`

Thus:

`\boldsymbol aa`	yields	$\boldsymbol{a}a$	the first token, ‘a’, becomes bold
`\boldsymbol \alpha\alpha`	yields	$\boldsymbol{\alpha}\alpha$	the first token, ‘\alpha’, becomes bold
`\boldsymbol{a\alpha}a\alpha`	yields	$\boldsymbol{a\alpha}a\alpha$	braces have been used to make the argument the group ‘a\alpha’, so both become bold

BRACED GROUPS:
A ‘braced group’ is a group, enclosed by braces, inside which some behavior is in force.
The `\bf` (boldface) command operates inside a braced group, notated by:

```
{\bf ... }
```

Here, `\bf` is a switch, which ‘turns on’ boldface inside the braced group;
boldface ends when the braced group ends.

Sometimes, you may not see the opening ‘{’ that signals the start of a braced group.
In this situation, when does a command (like `\bf`) end?
It ends at whichever occurs first:

- it is replaced by a competing command (e.g., `\bf` is replaced by `\rm`)
- the end of math mode (math delimiters form an implicit local group)

Examples: (explicit braced groups are indicated in red, for your convenience)

`\bf ab`	yields	$\mathbf{ab}$	turn on boldface; stays on to end of math mode
`{\bf ab}cd`	yields	$\mathbf{ab}cd$	an explicit braced group is entered; the ‘cd’ falls outside this group
`\bf{ab}cd`	yields	$\mathbf{abcd}$	turn on boldface; stays on to end of math mode; the braces here are extraneous
`{\bf{ab}c}d`	yields	$\mathbf{abc}d$	boldface operates inside a braced group; the ‘d’ falls outside this group
`{efg\bf{ab}c}d`	yields	$efg\mathbf{abc}d$	the ‘efg’ occur before boldface is turned on
`ab \bf cd \rm ef`	yields	$ab\mathbf{cd}\mathrm{ef}$	the competing `\rm` replaces boldface
`ab \bf cd {\rm ef} gh`	yields	$ab\mathbf{cd}\mathrm{ef}\mathbf{gh}$	the ‘gh’ is still in boldface

Make sure you see the difference in the behaviors below:

`\boldsymbol{ab}cd`	yields	$\boldsymbol{ab}cd$	`\boldsymbol` takes an argument
`\bf{ab}cd`	yields	$\mathbf{abcd}$	`\bf` does not take an argument; instead, `\bf` ‘turns on’ boldface behavior

`\!`

negative thin space; i.e., it ‘back ups’ a thin space amount

Examples:

`\rm IR`	yields	$\rm IR$
`\rm I\! R`	yields	$\rm I\!R$

see also: \negthinspace

`\,`
`\:`
`\>`
`\;`

`\,`	thin space (normally $\frac{1}{6} = \frac{3}{18}$ of a quad)
`\:`	medium space (normally $\frac{2}{9} = \frac{4}{18}$ of a quad)
`\>`	alternate medium space
`\;`	thick space (normally $\frac{5}{18}$ of a quad)

Examples:

normal spacing between letters:	$abababab$
using `\,` between letters:	$a\,b\,a\,b\,a\,b\,a\,b$
using `\:` between letters:	$a\:b\:a\:b\:a\:b\:a\:b$
using `\>` between letters:	$a\>b\>a\>b\>a\>b\>a\>b$
using `\;` between letters:	$a\;b\;a\;b\;a\;b\;a\;b$

see also: \thinspace

`\ (backslash space)`

control space; class ORD
$\TeX$ often ignores spaces, or collapses multiple spaces to a single space.
A control space is used to force $\TeX$ to typeset a space.

Examples:

		`\rm This is a sentence.` yields Thisisasentence. `\rm This\ is\ a\ sentence.` yields This is a sentence. `\rm This~is~a~sentence.` yields This is a sentence. `\text{This is a sentence.}` yields This is a sentence. in MathJax, this is the same as: \nobreakspace, \space, ~ (tilde character) see also: \text
`~ (tilde character)`		In $\TeX$ this is a non-breaking space—i.e., a blank space where $\TeX$ is not allowed to break between lines. MathJax (unlike $\TeX$) doesn't do any automatic breaking of lines, so MathJax will not break at *any* space. The tilde is useful to force a space where MathJax would otherwise collapse or ignore spaces, as illustrated in the examples below. class ORD \\ Click here to see examples of what happens with very long math in MathJax. Examples: `\rm Dr. Carol J.V. Fisher` yields Dr. CarolJ. V. Fisher `\rm Dr.~Carol~J.V.~Fisher` yields Dr. Carol J. V. Fisher `\text{Dr. Carol J.V. Fisher}` yields Dr. Carol J.V. Fisher `a b      c d` yields $abcd$ `a~b~~~~~~c~d` yields $a~b~~~~~~c~d$ in MathJax, this is the same as: \nobreakspace, \space, \ (backslash space)
`\#`	$\#$	literal number sign; literal pound sign; &#x0023; class ORD needed since `#` is used to indicate arguments in definitions
`\$`	$\$$	literal dollar sign; &#x0024; class ORD needed since `$` may (optionally) be used to delimit math mode Dollar sign outside of math mode: show/hide more info
`\%`	$\%$	literal percent sign; &#x0025; class ORD needed since `%` is used to begin a single-line comment
`\&`	$\&$	literal ampersand; &#x0026; class ORD needed since ampersands are used as separators in alignment environments and for HTML entity references inside math mode see also: `\And`
`\\`		line separator in alignment modes and environments Example: `\begin{gather}a\\a+b\\a+b+c\end{gather}` yields $\begin{gathered}a\\a+b\\a+b+c\end{gathered}$ For a literal backslash, see \backslash. in MathJax, these are essentially the same: \cr, \newline
`\_`	$\_$	literal underscore; &#x005F; class ORD needed since underscores are used for subscripts Examples: `a_2` yields a_2 `a\_2` yields a_2
`\{ \}`	$\{\ \}$	literal braces; \\{ is class OPEN needed since braces are used for grouping in math mode; \\} is class CLOSE non-stretchy when used alone; stretchy when used with `\left` or `\right` Examples: `{1,2,3}` yields $1,2,3$ `\{1,2,3\}` yields $\{1,2,3\}$ `\left\{\frac ab,c\right\}` yields $\left\{\frac ab,c\right\}$ see also: \brace, \lbrace, \rbrace
`|`	$\vert$	pipe character; vertical bar; absolute value; class ORD

<table>
<tr><th></th><th></th><th></th></tr>
<tr><td></td><td></td><td>non-stretchy when used alone; stretchy when used with <code>\left</code> or <code>\right</code>
Examples:
<code>|x|</code> yields $|x|$
<code>|\frac ab|</code> yields $|\frac{a}{b}|$
<code>\left|\frac ab\right|</code> yields $\left|\frac{a}{b}\right|$
<code>\{x | x\in\Bbb Z\}</code> yields $\{x|x\in\mathbb{Z}\}$
<code>\{x\,|\,x\in\Bbb Z\}</code> yields $\{x\,|\,x\in\mathbb{Z}\}$
see also: \lvert, \rvert, \vert</td></tr>
<tr><td><code>\|</code></td><td>$\|$</td><td>double pipe character; double vertical bar; norm; &#x2225; class ORD
non-stretchy when used alone; stretchy when used with <code>\left</code> or <code>\right</code>
Examples:
<code>\|x\|</code> yields $\|x\|$
<code>\|\frac ab\|</code> yields $\|\frac{a}{b}\|$
<code>\left\|\frac ab\right\|</code> yields $\left\|\frac{a}{b}\right\|$
see also: \lVert, \rVert, \Vert</td></tr>
<tr><td><code>( )</code></td><td>$()$</td><td>parentheses; <code>(</code> is class OPEN;
non-stretchy when used alone; stretchy when used with <code>\left</code> or <code>\right</code> <code>)</code> is class CLOSE
Examples:
<code>(\frac ab,c)</code> yields $(\frac{a}{b},c)$
<code>\left(\frac ab,c\right)</code> yields $\left(\frac{a}{b},c\right)$</td></tr>
<tr><td><code>.</code></td><td>$.$</td><td>period; decimal point class PUNCT
In some math environments (but not all):
With numbers on either side, there is no surrounding space: <code>3.14</code> yields 3.14
With non-numeric characters, there is a slight amount of space on right: <code>a.b</code> yields $a.b$
To suppress this space, enclose the ‘.’ in braces: <code>a{.}b</code> yields $a{.}b$</td></tr>
<tr><td><code>/</code></td><td>$/$</td><td>forward slash; class ORD
can be used to denote division
Example:
<code>a/b</code> yields a/b</td></tr>
<tr><td><code>+</code></td><td>$+$</td><td>plus symbol; class BIN
e.g., used for addition
Example:
<code>a+b</code> yields $a+b$</td></tr>
<tr><td><code>-</code></td><td>$-$</td><td>minus symbol; class BIN
e.g., used for subtraction
Example:
<code>a-b</code> yields $a-b$
<code>-b</code> yields $-b$ in most cases, proper spacing is achieved to denote an opposite
<code>\text{first: } -a\star b</code> yields $\text{first: } -a\star b$ an unusual situation; spacing is not optimal
<code>\text{first: } {-}a\star b</code> yields $\text{first: } {-}a\star b$ in such cases, you can put the minus sign (or, the group <code>-a</code>) inside braces to suppress extra space</td></tr>
<tr><td><code>[ ]</code></td><td>$[\,]$</td><td>(square) brackets; <code>[</code> is class OPEN;
non-stretchy when used alone; stretchy when used with <code>\left</code> or <code>\right</code> <code>]</code> is class CLOSE
Examples:
<code>[\frac ab,c]</code> yields $[\frac{a}{b},c]$
<code>\left[\frac ab,c\right]</code> yields $\left[\frac{a}{b},c\right]$</td></tr>
</table>

		see also: \brack, \lbrack, \rbrack
`=`	$=$	equal; equals class REL see also: \ne, \neq

`'`	$\prime$	prime symbol class ORD
		Example:
		`f(x) = x^2,\`
		`f'(x) = 2x,\`
		`f''(x) = 2` yields $f(x) = x^2,\ f'(x) = 2x,\ f''(x) = 2$
		see also: \prime

A

`\above`		general command for making fractions; gives control over thickness of horizontal fraction bar `{ <subformula1> \above <dimen> <subformula2> }` Creates a fraction: numerator: `subformula1` denominator: `subformula2` fraction bar has thickness: `dimen` There are separate local groups for `subformula1` and `subformula2` ; if these local groups are not explicit, then unexpected results may occur, as illustrated in the choose discussion. Examples: `a+1 \above 1pt b` yields ${a+1 \above 1pt b}$ `a \above 1pt b+2` yields ${a \above 1pt b+2}$ `{a+1 \above 1.5pt b+2}+c` yields ${a+1 \above 1.5pt b+2}+c$ see also: \abovewithdelims, \atop, \atopwithdelims, \cfrac, \dfrac, \frac, \genfrac, \over, \overwithdelims
`\abovewithdelims`		general command for making fractions; gives control over thickness of horizontal fraction bar; specifies left and right enclosing delimiters `{ <subformula1> \abovewithdelims <delim1> <delim2> <dimen> <subformula2> }` Creates a fraction: numerator: `subformula1` denominator: `subformula2` fraction bar has thickness: `dimen` `delim1` is put before the fraction `delim2` is put after the fraction For an empty delimiter, use ‘.’ in place of the delimiter. There are separate local groups for `subformula1` and `subformula2` ; if these local groups are not explicit, then unexpected results may occur, as illustrated in the choose discussion. Examples: `a+1 \abovewithdelims [ ] 1pt b` yields $\left[{a+1 \over b}\right]$ `{a \abovewithdelims . | 1.5pt b+2}_{a=3}` yields $\left.{a \over b+2}\right\vert_{a=3}$ `{a+1 \abovewithdelims \{ \} 1pt b+2}+c` yields $\left\{{a+1 \over b+2}\right\}+c$ see also: \above, \atop, \atopwithdelims, \cfrac, \dfrac, \frac, \genfrac, \over, \overwithdelims

`\acute`	ˊ	`&#x02CA;`
		acute accent
		`\acute #1`
		Usually, `#1` is a single letter; otherwise, accent is centered over argument.
		Examples:
		`\acute e` yields $\acute e$
		`\acute E` yields $\acute E$
		`\acute eu` yields $\acute eu$

		\acute{eu} yields $\acute{eu}$
\aleph	$\aleph$	Hebrew letter aleph; commonly used for the cardinality of the real numbers &#x2135; class ORD
\alpha	α	lowercase Greek letter alpha &#x03B1; class ORD
\amalg	$\amalg$	this symbol is often used for co-products &#x2A3F; class BIN
\And	$\&$	ampersand &#x0026; class ORD see also: \&
\angle	$\angle$	&#x2220; class ORD
\approx	$\approx$	&#x2248; class REL
\approxeq AMSsymbols	$\approxeq$	&#x224A; class REL
\arccos	$\arccos$	does not change size; default limit placement is the same in both inline and display modes; can change limit placement using \limits; see the Big Operators Table for examples class OP If alternate notation is desired, define: \def\arccosAlt{\cos^{-1}} so that $\arccosAlt(x)$ yields $\cos^{-1}(x)$
\arcsin	$\arcsin$	does not change size; default limit placement is the same in both inline and display modes; can change limit placement using \limits; see the Big Operators Table for examples class OP If alternate notation is desired, define: \def\arcsinAlt{\sin^{-1}} so that $\arcsinAlt(x)$ yields $\sin^{-1}(x)$
\arctan	$\arctan$	does not change size; default limit placement is the same in both inline and display modes; can change limit placement using \limits; see the Big Operators Table for examples class OP If alternate notation is desired, define: \def\arctanAlt{\tan^{-1}} so that $\arctanAlt(x)$ yields $\tan^{-1}(x)$
\arg	$\arg$	the complex argument function; does not change size; default limit placement is the same in both inline and display modes; can change limit placement using \limits; see the Big Operators Table for examples class OP
\array		a synonym for \matrix \array{ <math> & <math> ... \cr <repeat as needed> } alignment occurs at the ampersands; a double-backslash can be used in place of the \cr ; the final \\ or \cr is optional Example: \array{ a & b+1 \cr c+1 & d } yields $\begin{matrix} a & b+1 \\ c+1 & d \end{matrix}$ see also: \matrix
\arrowvert	$\arrowvert$	not intended for direct use; used internally to create stretchy delimiters &#x23D0; class ORD see also: |, \vert, \lvert, \rvert,
\Arrowvert	$\Arrowvert$	not intended for direct use; used internally to create stretchy delimiters &#x2016; class PUNCT see also: \\\|, \Vert, \lVert, \rVert
\ast	$\ast$	asterisk &#x2217; class BIN
\asymp	$\asymp$	asymptotic &#x224D; class REL
\atop		general command for making a fraction-like structure, but without the horizontal fraction bar { <subformula1> \atop <subformula2> } Creates a fraction-like structure: ‘numerator’ subformula1 ’denominator’ subformula2

		There are separate local groups for `subformula1` and `subformula2` ; if these local groups are not explicit, then unexpected results may occur, as illustrated in the choose discussion. Examples: `a \atop b` yields $a \atop b$ `a+1 \atop b+2` yields ${a+1 \atop b+2}$ `{a+1 \atop b+2}+c` yields ${a+1 \atop b+2}+c$ see also: \above, \abovewithdelims, \atopwithdelims, \cfrac, \dfrac, \frac, \genfrac, \over, \overwithdelims
`\atopwithdelims`		general command for making a fraction-like structure, but without the horizontal fraction bar; specifies left and right enclosing delimiters `{ <subformula1> \atopwithdelims <delim1> <delim2> <subformula2> }` Creates a fraction-like structure: 'numerator' `subformula1` 'denominator' `subformula2` `delim1` is put before the structure `delim2` is put after the structure For an empty delimiter, use '.' in place of the delimiter. There are separate local groups for `subformula1` and `subformula2` ; if these local groups are not explicit, then unexpected results may occur, as illustrated in the choose discussion. Examples: `a \atopwithdelims [ ] b` yields $\left[{a \atop b}\right]$ `a+1 \atopwithdelims . | b+2` yields $\left.{a+1 \atop b+2}\right\vert$ `{a+1 \atopwithdelims \{ \} b+2}+c` yields $\left\{{a+1 \atop b+2}\right\}+c$ see also: \above, \abovewithdelims, \atop, \cfrac, \dfrac, \frac, \genfrac, \over, \overwithdelims

B

`\backepsilon` AMSsymbols	∍	`&#x220D;` class REL
`\backprime` AMSsymbols	‵	see also: \prime `&#x2035;` class ORD
`\backsim` AMSsymbols	∽	`&#x223D;` class REL
`\backsimeq` AMSsymbols	⋍	`&#x22CD;` class REL
`\backslash`	\	see also: \setminus `&#x2216;`
`\bar`	ˉ	bar accent (non-stretchy) `&#x02C9;` `\bar #1` Usually, `#1` is a single letter; otherwise, bar is centered over argument. Examples: `\bar x` yields $\bar x$ `\bar X` yields $\bar X$ `\bar xy` yields $\bar xy$ `\bar{xy}` yields $\bar{xy}$
`\barwedge` AMSsymbols	⊼	`&#x22BC;` class BIN
`\Bbb`		blackboard-bold for uppercase letters and lowercase 'k'; class ORD if lowercase blackboard-bold letters are not available, then they are typeset in a roman font `\Bbb #1` Whether lower-case letters are displayed in blackboard-bold, or not, depends on the fonts being used. The MathJax web-based fonts don't have lowercase blackboard-bold, but the STIX fonts do; so users with the STIX fonts installed will be able to display lowercase blackboard-bold letters. Examples: `\Bbb R` yields $\mathbb{R}$ `\Bbb ZR` yields $\mathbb{Z}R$ `\Bbb{AaBbKk}Cc` yields $\mathbb{AaBbKk}Cc$ `\Bbb{ABCDEFGHIJKLMNOPQRSTUVWXYZ}` yields $\mathbb{ABCDEFGHIJKLMNOPQRSTUVWXYZ}$ see also: \mathbb
	𝕜	

\Bbbk AMSsymbols		blackboard-bold lowercase k &#x006B; class ORD
\because AMSsymbols	∵	&#x2235; class REL
\begin		used in \begin{xxx} ... \end{xxx} environments
\beta	β	lowercase Greek letter beta &#x03B2; class ORD
\beth AMSsymbols	ℶ	Hebrew letter beth &#x2136; class ORD
\between AMSsymbols	≬	&#x226C; class REL
\bf		turns on boldface; affects uppercase and lowercase letters, and digits class ORD {\bf ... } Examples: \bf AaBb\alpha\beta123 yields $\mathbf{AaBb}\boldsymbol{\alpha\beta}\mathbf{123}$ {\bf A B} A B yields $\mathbf{AB}AB$ \bf AB \rm CD yields $\mathbf{AB}\mathrm{CD}$ \bf{AB}CD yields $\mathbf{ABCD}$ see also: \mathbf, \boldsymbol
\Bigg \bigg \Big \big		used to obtain various-sized delimiters; may be followed by any of these Variable-Sized Delimiters Examples: $\Bigg[$ \Bigg[2.470 em $\bigg[$ \bigg[2.047 em $\Big[$ \Big[1.623 em $\big[$ \big[1.2 em [[
\Biggl \Biggm \Biggr \biggl \biggm \biggr \Bigl \Bigm \Bigl \bigl \bigm \bigr		Used to obtain various-sized delimiters, with a left/right/middle context; may be followed by any of these Variable-Sized Delimiters. The ‘l’ (left), ’m’ (middle), and ‘r’ (right) specifications may make reading the source code more meaningful, especially when there are delimiters inside delimiters. Whereas (say) \Bigg produces results of class ORD, we have: - \Biggl produces results of class OPEN - \Biggr produces results of class CLOSE - \Biggm produces results of class REL The spacing for these differ (but may not always be apparent, as it depends on the class of what is next to it). For example, $x\big\| y$ ($x\big\vert y$) has less space than $x\bigm\| y$ ($x\bigm\vert y$). Therefore, these commands affect typeset results in a fundamental way; it is best to use the form appropriate for the position of the desired delimiter.
\bigcap	⋂	changes size; &#x22C2; class OP can change limit placement using \limits and \nolimits; see the Big Operators Table for examples
\bigcirc	◯	&#x25EF; class BIN
\bigcup	⋃	changes size; &#x22C3; class OP can change limit placement using \limits and \nolimits; see the Big Operators Table for examples
\bigodot \bigoplus \bigotimes	⨀ ⨁ ⨂	all change size; &#x2A00; class OP can change limit placement using \limits and \nolimits; &#x2A01; class OP see the Big Operators Table for examples &#x2A02; class OP
\bigsqcup	⨆	changes size; &#x2A06; class OP can change limit placement using \limits and \nolimits; see the Big Operators Table for examples
\bigstar AMSsymbols	★	&#x2605; class ORD
\bigtriangledown	▽	&#x25BD; class BIN
\bigtriangleup	△	&#x25B3; class REL
\biguplus	⨄	changes size; &#x2A04; class OP can change limit placement using \limits and \nolimits; see the Big Operators Table for examples
\bigvee	⋁	changes size; &#x22C1; class OP can change limit placement using \limits and \nolimits;

		see the Big Operators Table for examples
`\bigwedge`	$\bigwedge$	changes size; can change limit placement using \limits and \nolimits; see the Big Operators Table for examples `&#x22C0;` class OP
`\binom` AMSmath		notation commonly used for binomial coefficients `\binom #1 #2` Examples: `\binom n k` yields (inline mode) $\binom{n}{k}$ `\binom n k` yields (display mode) $\displaystyle\binom{n}{k}$ `\binom{n-1}k-1` yields $\binom{n-1}{k}-1$ `\binom{n-1}{k-1}` yields $\binom{n-1}{k-1}$ see also: \binom, \choose, \dbinom, \tbinom
`\blacklozenge` AMSsymbols	$\blacklozenge$	`&#x29EB;` class ORD
`\blacksquare` AMSsymbols	$\blacksquare$	`&#x25A0;` class ORD

`\blacktriangle`
`\blacktriangledown`
both AMSsymbols
	$\blacktriangle$
	$\blacktriangledown$
		`&#x25B2;` class ORD
		`&#x25BC;` class ORD

`\blacktriangleleft`
`\blacktriangleright`
both AMSsymbols
	$\blacktriangleleft$
	$\blacktriangleright$
		`&#x25C0;` class BIN
		`&#x25B6;` class BIN

`\bmod`	mod	properly spaced as a binary operator class BIN
`\boldsymbol`		as opposed to \bf and \mathbf , class ORD `\boldsymbol` applies to nearly *all* symbols, not just letters and numbers `\boldsymbol #1` Examples: `\boldsymbol aa` yields $\boldsymbol{a}a$ `\boldsymbol \alpha\alpha` yields $\boldsymbol{\alpha}\alpha$ `\boldsymbol{a\alpha}a\alpha` yields $\boldsymbol{a\alpha}a\alpha$ `\boldsymbol{a+2+\alpha+\frac{x+3}{\beta+4}}` yields $\boldsymbol{a+2+\alpha+\frac{x+3}{\beta+4}}$ `\mathbf{a+2+\alpha+\frac{x+3}{\beta+4}}` yields $\mathbf{a+2+\alpha+\frac{x+3}{\beta+4}}$ see also: \bf, \mathbf
`\bot`	$\bot$	`&#x22A5;` class ORD
`\bowtie`	$\bowtie$	`&#x22C8;` class REL
`\Box` AMSsymbols	$\Box$	`&#x25A1;` class ORD
`\boxdot` AMSsymbols	$\boxdot$	`&#x22A1;` class BIN
`\boxed` AMSmath		puts a box around argument; argument is in math mode `\boxed #1` Examples: `\boxed ab` yields $\boxed{a}b$ `\boxed{ab}` yields $\boxed{ab}$ `\boxed{ab\strut}` yields $\boxed{ab\strut}$ `\boxed{\text{boxed text}}` yields $\boxed{\text{boxed text}}$ see also: \fbox
`\boxminus` AMSsymbols `\boxplus` AMSsymbols `\boxtimes` AMSsymbols	$\boxminus$ $\boxplus$ $\boxtimes$	`&#x229F;` class BIN `&#x229E;` class BIN `&#x22A0;` class BIN
`\brace`		creates a braced structure `{ <subformula1> \brace <subformula2> }`

		Examples: `\brace` yields $\left\{ \right\}$ `a\brace b` yields $\left\{ a \atop b \right\}$ `a+b+c\brace d+e+f` yields $\left\{ a+b+c \atop d+e+f \right\}$ `a+{b+c\brace d+e}+f` yields $a+\left\{ b+c \atop d+e \right\}+f$
`\bracevert`		not intended for direct use; used internally to create stretchy delimiters `&#x23AA;` class ORD
`\brack`		creates a bracketed structure `{ <subformula1> \brack <subformula2> }` Examples: `\brack` yields $\left[\right]$ `a\brack b` yields $\left[a \atop b \right]$ `a+b+c\brack d+e+f` yields $\left[a+b+c \atop d+e+f \right]$ `a+{b+c\brack d+e}+f` yields $a+\left[b+c \atop d+e \right]+f$
`\breve`	˘	breve accent `&#x02D8;` `\breve #1` Usually, `#1` is a single letter; otherwise, accent is centered over argument. Examples: `\breve e` yields $\breve e$ `\breve E` yields $\breve E$ `\breve eu` yields $\breve e u$ `\breve{eu}` yields $\breve{eu}$
`\buildrel ... \over ...`		`\buildrel <subformula1> \over #1` The result is of class `REL` (binary relation), so it has the spacing of a relation. Examples: `\buildrel \alpha\beta \over \longrightarrow` yields $\overset{\alpha\beta}{\longrightarrow}$ `\buildrel \rm def \over {:=}` yields $\overset{\rm def}{:=}$
`\bullet`	$\bullet$	`&#x2219;` class BIN
`\Bumpeq` AMSsymbols `\bumpeq` AMSsymbols	$\Bumpeq$ $\bumpeq$	`&#x224E;` class REL `&#x224F;` class REL

C

`\cal`		class ORD turns on calligraphic mode; only affects uppercase letters and digits `{\cal ... }` Examples: `\cal ABCDEFGHIJKLMNOPQRSTUVWXYZ` yields $\mathcal{ABCDEFGHIJKLMNOPQRSTUVWXYZ}$ `\cal 0123456789` yields $\mathcal{0123456789}$ `\cal abcdefghijklmnopqrstuvwxyz` yields $abcdefghijklmnopqrstuvwxyz$ `abcdefghijklmnopqrstuvwxyz` yields $abcdefghijklmnopqrstuvwxyz$ `{\cal AB}AB` yields $\mathcal{AB}AB$ `\cal AB \rm AB` yields $\mathcal{AB}\rm AB$ `\cal{AB}CD` yields $\mathcal{ABCD}$ see also: \oldstyle, \mathcal

`\cancel`		Used to ‘cancel’ (strikeout).
		`\cancel #1`
		`\bcancel #1`
		Examples:

		\frac{(x+1)\cancel{(x+2)}}{3\cancel{(x+2)}} yields $\frac{(x+1)\cancel{(x+2)}}{3\cancel{(x+2)}}$ \frac{\bcancel{\frac13}}{\bcancel{\frac13}} = 1 yields $\frac{\bcancel{\frac13}}{\bcancel{\frac13}} = 1$
\Cap AMSsymbols	⋒	&#x22D2; class BIN see also: \bigcap, \cap, \Cup, \cup, \doublecap, \doublecup
\cap	∩	&#x2229; class BIN see also: \bigcap, \Cap, \Cup, \cup, \doublecap, \doublecup

\cases		class OPEN
		for piecewise-defined functions
		\cases{ <math> & <math> \cr <repeat as needed> }
		a double-backslash can be used in place of \cr ;
		the final \\ or \cr is optional
		In TeX , the second column is automatically in text-mode, while in MathJax it is in math-mode.
		This behavior will be changed to be consistent with TeX in a future release of MathJax.
		Example:
		|x| =
		\cases{
		x & \text{if } x\ge 0\cr
		-x & \text{if } x\lt 0
		}
		yields $\lvert x\rvert = \begin{cases} x & \text{if } x \ge 0 \\ -x & \text{if } x < 0 \end{cases}$

\cdot	⋅	&#x22C5; class BIN centered dot Examples: a\cdot b yields $a \cdot b$ a\cdotp b yields $a\cdotp b$ a\centerdot b yields $a \centerdot b$ see also: \cdotp, \cdots, \centerdot
\cdotp	·	&#x22C5; class PUNCT centered dot, punctuation symbol Examples: \rm s \cdot h yields $\rm s \cdot h$ \rm s \cdotp h yields $\rm s \cdotp h$ see also: \cdot, \centerdot
\cdots	⋯	&#x22EF; class INNER centered dots; dot dot dot Example: x_1 + \cdots + x_n yields $x_1 + \cdots + x_n$ see also: \dots, \ldots
\centerdot AMSsymbols	·	&#x22C5; class BIN centered dot Examples: a\cdot b yields $a \cdot b$ a\cdotp b yields $a\cdotp b$ a\centerdot b yields $a \centerdot b$ see also: \cdot, \cdotp
\cfrac AMSmath		use for continued fractions \cfrac #1 #2 Examples: \frac{2}{1+\frac{2}{1+\frac{2}{1}}} yields $\frac{2}{1+\frac{2}{1+\frac{2}{1}}}$ \cfrac{2}{1+\cfrac{2}{1+\cfrac{2}{1}}} yields

		$$\frac{2}{1+\frac{2}{1+\frac{2}{1}}}$$ see also: \above, \abovewithdelims, \atop, \atopwithdelims, \dfrac, \frac, \genfrac, \over, \overwithdelims

`\check`	ˇ	`&#x02C7;`
		check accent
		`\check #1`
		Usually, `#1` is a single letter; otherwise, accent is centered over argument.
		Examples:
		`\check o` yields $\check o$
		`\check O` yields $\check O$
		`\check oe` yields $\check oe$
		`\check{oe}` yields $\check{oe}$

`\checkmark` AMSsymbols	✓	`#x2713;` class ORD
`\chi`	χ	`&#x03C7;` class ORD lowercase Greek letter chi

`\choose`		notation commonly used for binomial coefficients;
		different versions for inline and display modes
		`{ <subformula1> \choose <subformula2> }`
		There are separate local groups for `subformula1` and `subformula2`;
		if these local groups are not explicit, then unexpected results may occur, as illustrated next.
		Examples (showing the math delimiters):
		`$\displaystyle`
		`n+1`
		`\choose`
		`k+2`
		`$` yields ${\displaystyle n+1}\choose{k+2}$
		Without an explicit braced group, the local group for `subformula1` extends back to the opening math delimiter. That is, this code is interpreted as (color added for emphasis): `${\displaystyle n+1}\choose{k+2}$`
		Now it is clear that only the `n+1` is affected by the `\displaystyle` switch.
		`$\displaystyle`
		`{n+1`
		`\choose`
		`k+2}`
		`$` yields $\displaystyle{n+1 \choose k+2}$
		Here, an explicit braced group is used for the `\choose` command, making both subformulas clear—and the expected result is obtained.
		Note that it may appear that `\displaystyle` is taking an argument, but this is not the case: instead, `\displaystyle` acts as a switch which turns on display mode, and the entire `choose` command is affected.
		Examples (showing math delimiters):
		`$n+1 \choose k+2$` yields ${n+1 \choose k+2}$
		`$$n+1 \choose k+2$$` yields $\displaystyle{n+1 \choose k+1}$
		`$1+{n \choose 2}+k$` yields $1+{n \choose 2}+k$
		see also: \binom, \dbinom, \tbinom

`\circ`	$\circ$	`&#x2218;` class BIN Examples: `(f\circ g)(x) = f(g(x))` yields $(f\circ g)(x) = f(g(x))$ `45^\circ` yields 45°
`\circeq` AMSsymbols	≗	`&#x2257;` class REL
`\circlearrowleft` AMSsymbols `\circlearrowright` AMSsymbols	↺ ↻	`&#x21BA;` counterclockwise class REL `&#x21BB;` clockwise class REL
`\circledast` AMSsymbols `\circledcirc` AMSsymbols `\circleddash` AMSsymbols	⊛ ⊚ ⊝	`&#x229B;` circled asterisk class BIN `&#x229A;` circled circle class BIN `&#x229D;` circled dash class BIN

\circledR	AMSsymbols	®	&#x00AE; circled R class ORD
\circledS	AMSsymbols	Ⓢ	&#x24C8; circled S class ORD

\class	[HTML]		non-standard; extension is loaded automatically when used;
			used to specify a CSS class for styling mathematics
			\class #1 #2
			where:
			- #1 is a CSS class name (without quotes)
			- #2 is the mathematics to be styled
			
			Example:
			Suppose this CSS style information is provided outside of math mode:
			`<style type="text/css">`
			`.smHighlightRed {`
			`font-size:small;`
			`background-color:yellow;`
			`color:red;`
			`}`
			`</style>`
			Then,
			`ab\class{smHighlightRed}{cdef}gh` yields $abcdefgh$

\clubsuit		♣	&#x2663; class ORD see also: \diamondsuit, \heartsuit, \spadesuit
\colon		:	&#x003A; class PUNCT a colon, treated as a punctuation mark (instead of a relation) Examples: `f:A\to B` yields $f : A \to B$ `f\colon A\to B` yields $f\colon A \to B$
\color			used to specify a color in mathematics \color #1 #2 where: #1 is the desired color #2 is the mathematics to be colored This works differently from standard LaTeX (where \color is a switch). In a future version of MathJax, it will be possible to load an extension to make the command behave like the LaTeX version. Examples: `\color{red}{ \frac{1+\sqrt{5}}{2} }` yields $\frac{1+\sqrt{5}}{2}$ `\color{#0000FF}AB` yields AB
\complement	AMSsymbols	∁	&#x2201; class ORD
\cong		≅	&#x2245; class REL congruent see also: \ncong
\coprod		∐	&#x2210; class OP coproduct
\cos		cos	class OP cosine; does not change size; default limit placement is the same in both inline and display modes; can change limit placement using \limits; see the Big Operators Table for more examples Examples: `\cos x` yields $\cos x$ `\cos(2x-1)` yields $\cos(2x-1)$ see also: \sin
\cosh		cosh	class OP hyperbolic cosine; does not change size; default limit placement is the same in both inline and display modes; can change limit placement using \limits;

		see the Big Operators Table for more examples hyperbolic cosine Examples: `\cosh x` yields $\cosh x$ `\cosh(2x-1)` yields $\cosh(2x-1)$ see also: \sinh
`\cot`	$\cot$	class OP cotangent; does not change size; default limit placement is the same in both inline and display modes; can change limit placement using \limits; see the Big Operators Table for more examples Examples: `\cot x` yields $\cot x$ `\cot(2x-1)` yields $\cot(2x-1)$ see also: \tan
`\coth`	$\coth$	class OP hyperbolic cotangent; does not change size; default limit placement is the same in both inline and display modes; can change limit placement using \limits; see the Big Operators Table for more examples Examples: `\coth x` yields $\coth x$ `\coth(2x-1)` yields $\coth(2x-1)$
`\cr`		carriage return; line separator in alignment modes and environments in MathJax, these are essentially the same: \\, \newline
`\csc`	$\csc$	class OP cosecant does not change size; default limit placement is the same in both inline and display modes; can change limit placement using \limits; see the Big Operators Table for more examples Examples: `\csc x` yields $\csc x$ `\csc(2x-1)` yields $\csc(2x-1)$ see also: \sec

`\cssId` `[HTML]`		non-standard; class ORD; extension is loaded automatically when used;
		used to set a MathML element's ID attribute, so it can be accessed dynamically
		(e.g., to add an event handler, add CSS styling, or set display status)
		`\cssId #1 #2`
		where:
		- `#1` is an ID attribute (without quotes)
		- `#2` is the mathematics to be identified by the ID
		
		Example:
		
		Suppose this HTML and Javascript is provided outside of math mode:
		`<button type="button" onclick="turnRed();">`
		`Click button to turn something red`
		`</button>`
		
		`<script type="text/javascript">`
		`function turnRed() {`
		`document.getElementById('testID').style.color = "red";`
		`}`
		`</script>`
		
		Suppose further that the following MathJax code is provided:
		`$$`
		`abc`
		`\cssId{testID}{def\text{ Something will turn red! }ghi}`
		`jkl`
		`$$`
		
		Then, this HTML/Javascript/MathJax produces:

			Click button to turn something red $abcdef$ Something will turn red! $ghijkl$ A more meaningful example (with well-commented source code) is provided by Design Science, Inc., and shows how you can display the steps in a proof one line at a time.
\Cup	AMSsymbols	⋓	&#x22D3; class BIN see also: \bigcup, \Cap, \cap, \cup, \doublecap, \doublecup
\cup		∪	&#x222A; class BIN see also: \bigcup, \Cap, \cap, \Cup, \doublecap, \doublecup
\curlyeqprec \curlyeqsucc	AMSsymbols AMSsymbols	⋞ ⋟	&#x22DE; class REL &#x22DF; class REL
\curlyvee \curlywedge	AMSsymbols AMSsymbols	⋎ ⋏	&#x22CE; class BIN &#x22CF; class BIN
\curvearrowleft \curvearrowright	AMSsymbols AMSsymbols	↶ ↷	&#x21B6; counterclockwise class REL &#x21B7; clockwise class REL

D

\dagger \ddagger		† ‡	&#x2020; dagger class BIN &#x2021; double dagger class BIN
\daleth	AMSsymbols	ℸ	&#x2138; class ORD Hebrew letter daleth
\dashleftarrow \dashrightarrow	AMSsymbols AMSsymbols	⇠ ⇢	&#x21E0; dashed left arrow; non-stretchy class REL &#x21E2; dashed right arrow; non-stretchy class REL
\dashv		⊣	&#x22A3; class REL
\dbinom	AMSmath		notation commonly used for binomial coefficients; display version (in both inline and display modes) `\dbinom #1 #2` Examples: `\dbinom n k` yields (inline mode) $\dbinom{n}{k}$ `\dbinom n k` yields (display mode) $\dbinom{n}{k}$ `\dbinom{n-1}k-1` yields $\dbinom{n-1}{k}-1$ `\dbinom{n-1}{k-1}` yields $\dbinom{n-1}{k-1}$ see also: \binom, \choose, \tbinom

\dot
\ddot
\dddot
\ddddot
	
	AMSmath
	AMSmath
		˙
		¨
		···
		····
			&#x02D9; dot accent
			&#x00A8; double dot accent
			triple dot accent
			quadruple dot accent
			`\dot #1`
			`\ddot #1`
			`\dddot #1`
			`\ddddot #1`
			Usually, `#1` is a single letter; otherwise, accent is centered over argument.
			Examples:
			`\dot x` yields $\dot{x}$
			`\ddot x` yields $\ddot{x}$
			`\dddot x` yields $\dddot{x}$

		`\ddddot x` yields $\ddddot{x}$ `\ddot x(t)` yields $\ddot{x}(t)$ `\ddddot{y(x)}` yields $\ddddot{y(x)}$
`\ddots`	$\ddots$	`&#x22F1;` class INNER three diagonal dots
`\DeclareMathOperator AMSmath`		Multi-letter operator names (like $\log$, $\sin$, and $\lim$) are traditionally typeset in a roman font. `\DeclareMathOperator` allows you to define your own operator names; they are subsequently typeset using the proper font and spacing; you can control the way that limits appear (see examples below) `\DeclareMathOperator #1 #2` where: • `#1` is the operator name, including the preceding backslash; only letters a–z and A–Z are allowed; in particular, no numbers are allowed in operator names • `#2` is the replacement text for the operator name A named operator is available in any mathematics that appears *after* it is defined on the page. Examples: `myOp(x)` yields $myOp(x)$ poor style; the function name should appear in a roman font `\text{myOp}(x)` yields $\text{myOp}(x)$ better; a nuisance to type if used frequently `\DeclareMathOperator {\myOp}{myOp} \myOp(x)` yields $\operatorname{myOp}(x)$ best; once an operator is declared, it can be used in any subsequent mathematics `\myOp_a^b(x)` yields (inline mode) $\operatorname{myOp}_a^b(x)$ standard subscript and superscript position for inline mode `\myOp_a^b(x)` yields (display mode) $\operatorname*{myOp}_a^b(x)$ standard subscript and superscript position for display mode `\DeclareMathOperator* {\myOP}{myOP} \myOP_a^b(x)` yields (inline mode) $\operatorname*{myOP}_a^b(x)$ operator names are case-sensitive, so `\myOp` is different from `\myOP` ; if displaystyle limits are desired in *both* inline and display modes, then use `DeclareMathOperator*` instead of `DeclareMathOperator`

`\def`		for defining your own commands (control sequences, macros, definitions);
		must appear (within math delimiters) before it is used;
		alternatively, you can define macros using the MathJax configuration options in the `<head>`
		`\def\myCommandName{ <replacement text> }`
		Example:
		`\def\myHearts{\color{purple}{\heartsuit}\kern-2.5pt\color{green}{\heartsuit}}`
		`\myHearts\myHearts`
		yields: ♡♡♡♡
		A definition may take one or more arguments:
		Example:
		`\def\myHearts#1#2{\color{#1}{\heartsuit}\kern-2.5pt\color{#2}{\heartsuit}}`
		`\myHearts{red}{blue}`
		yields: ♡♡
		see also: \newcommand

`\deg`	$\deg$	class OP degree; does not change size; default limit placement is the same in both inline and display modes; can change limit placement using \limits; see the Big Operators Table for examples
`\Delta` `\delta`	Δ δ	`&#x0394;` uppercase Greek letter delta class ORD `&#x03B4;` lowercase Greek letter delta class ORD see also: \varDelta

`\det`	$\det$	class OP
		determinant;
		does not change size;

		default limit placement can be changed using `\limits` and `\nolimits`; does not change size; see the Big Operators Table for more examples Examples: `\det_{\rm sub}` yields (inline mode) $\det_{\rm sub}$ `\det_{\rm sub}` yields (display mode) $\det\limits_{\rm sub}$ `\det\limits_{\rm sub}` yields (inline mode) $\det\limits_{\rm sub}$ `\det\nolimits_{\rm sub}` yields (display mode) $\det\nolimits_{\rm sub}$
`\dfrac` AMSmath		fractions; display version (in both inline and display modes) `\dfrac #1 #2` Examples: `\dfrac a b` yields (inline mode) $\dfrac a b$ `\dfrac a b` yields (display mode) $\dfrac a b$ `\frac a b` yields (inline mode) $\frac a b$ `\dfrac{a-1}b-1` yields $\dfrac{a-1}b-1$ `\dfrac{a-1}{b-1}` yields $\dfrac{a-1}{b-1}$ see also: \above, \abovewithdelims, \atop, \atopwithdelims, \cfrac, \frac, \genfrac, \over, \overwithdelims
`\diagdown` AMSsymbols `\diagup` AMSsymbols	╲ ╱	`&#x2572;` diagonal down (from left to right) class ORD `&#x2571;` diagonal up (from left to right) class ORD
`\Diamond` AMSsymbols `\diamond`	◊ ⋄	`&#x25CA;` large diamond class ORD `&#x22C4;` small diamond class BIN
`\diamondsuit`	♢	`&#x2662;` class ORD see also: \clubsuit, \heartsuit, \spadesuit
`\digamma` AMSsymbols	ϝ	`&#x03DD;` class ORD
`\dim`	dim	class OP dimension; does not change size; default limit placement is the same in both inline and display modes; can change limit placement using \limits; see the Big Operators Table for examples

`\displaylines`		to display any number of centered formulas (without any alignment)
		`\displaylines{ <math> \cr <repeat as needed> }`
		a double-backslash can be used in place of the `\cr`;
		the final `\\` or `\cr` is optional
		Example:
		`\displaylines{`
		`a = a\\`
		`\text{if } a=b \text{ then } b=a\\`
		`\text{if } a=b \text{ and } b=c \text{ then } a=c`
		`}`
		yields
		$a = a$
		$\text{if } a=b \text{ then } b=a$
		$\text{if } a=b \text{ and } b=c \text{ then } a=c$
		see also: gather

`\displaystyle`		class ORD
		used to over-ride automatic style rules and force display style;
		stays in force until the end of math mode or the braced group, or until another style is selected
		`{ \displaystyle ... }`
		Example:
		In inline mode:
		`\frac ab+\displaystyle\frac ab+\textstyle\frac ab`
		`+\scriptstyle\frac ab+\scriptscriptstyle\frac ab`
		yields:
		$\frac ab+\displaystyle\frac ab+\textstyle\frac ab+\scriptstyle\frac ab+\scriptscriptstyle\frac ab$
		Example:

		In inline mode: \frac ab + {\displaystyle \frac cd + \frac ef} + \frac gh yields $\tfrac{a}{b}+\dfrac{c}{d}+\dfrac{e}{f}+\tfrac{g}{h}$ Example: In inline mode: \frac ab + \displaystyle{\frac cd + \frac ef} + \frac gh yields $\tfrac{a}{b}+\dfrac{c}{d}+\dfrac{e}{f}+\dfrac{g}{h}$ see also: \textstyle, \scriptstyle, \scriptscriptstyle
\div	÷	&#x00F7; class BIN division symbol
\divideontimes AMSsymbols	⋇	&#x22C7; class BIN
\Doteq AMSsymbols \doteq	≑ ≐	&#x2251; class REL &#x2250; class REL
\dotplus AMSsymbols	∔	&#x2214; class BIN
\dots	…	&#x2026; class INNER lower dots; ellipsis; ellipses; dot dot dot In LaTeX , \dots chooses either \cdots or \ldots depending on the context; MathJax, however, always gives lower dots. Examples: x_1, \dots, x_n yields $x_1, \ldots, x_n$ x_1 + \dots + x_n yields $x_1 + \cdots + x_n$ x_1 + \dotsb + x_n yields $x_1 + \cdots + x_n$ x_1 + \cdots + x_n yields $x_1 + \cdots + x_n$ see also: \cdots, \ldots, \dotsb, \dotsc, \dotsi, \dotsm, \dotso
\dotsb \dotsc \dotsi \dotsm \dotso		&#x22EF; \dotsb class INNER dots with binary operations and relations $x_1 + x_2 + \cdots + x_n$ &#x2026; \dotsc class INNER dots with commas $x_1, x_2, \ldots, x_n$ &#x22EF; \dotsi class INNER dots with integrals $\int_{A_1}\int_{A_2}\cdots\int_{A_n}$ &#x22EF; \dotsm class INNER dots with multiplication $x_1 x_2 \cdots x_n$ &#x2026; \dotso class INNER other dots $A_1 \ldots A_n$ see also: \cdots, \dots, \ldots
\doublebarwedge AMSsymbols	⩞	&#x2A5E; BIN
\doublecap AMSsymbols \doublecup AMSsymbols	⋒ ⋓	&#x22D2; class BIN &#x22D3; class BIN see also: \Cap, \Cup, \cap, \cup
\downarrow \Downarrow	↓ ⇓	&#x2193; down arrow; non-stretchy class REL &#x21D3; double down arrow; non-stretchy class REL
\downdownarrows AMSsymbols	⇊	&#x21CA; class REL down down arrows; non-stretchy
\downharpoonleft AMSsymbols \downharpoonright AMSsymbols	⇃ ⇂	&#x21C3; down harpoon left; non-stretchy class REL &#x21C2; down harpoon right; non-stretchy class REL see also: \leftharpoondown, \leftharpoonup

E

\ell	ℓ	&#x2113; class ORD
\emptyset	∅	&#x2205; class ORD empty set see also: \varnothing
\end		used in \begin{xxx} ... \end{xxx} environments
\enspace		\enspace is a 0.5em space

<table>
<tr><td></td><td></td><td>Example:
`|\enspace|\enspace|` yields | | |</td></tr>
<tr><td>`\epsilon`</td><td>ϵ</td><td>`&#x03F5;` class ORD
lowercase Greek letter epsilon

see also: \varepsilon</td></tr>
<tr><td>`\eqalign`</td><td></td><td>equation alignment;
for aligning multi-line displays at a single place
`\eqalign{ <math> & <math> \cr <repeat as needed> }`
the ampersand is placed where alignment is desired;
a double-backslash can be used in place of the `\cr` ;
the final `\\` or `\cr` is optional;
supports only a single `\tag`, which is vertically centered

Example:

`\eqalign{`
`3x - 4y &= 5\cr`
`x  +  7 &= -2y`
`}`

yields:
$$\begin{aligned} 3x - 4y &= 5 \\ x + 7 &= -2y \end{aligned}$$
Example:
A `<math>` component may be empty:

`\eqalign{`
`(a+b)^2 &= (a+b)(a+b) \\`
`        &= a^2 + ab + ba + b^2 \\`
`        &= a^2 + 2ab + b^2`
`}`

yields:
$$\begin{aligned} (a+b)^2 &= (a+b)(a+b) \\ &= a^2 + ab + ba + b^2 \\ &= a^2 + 2ab + b^2 \end{aligned}$$
Example:
The result of `\eqalign` is a vertically-centered block;
you can use more than one in the same display:

`\left\{`
`\eqalign{`
`a &= 1\\`
`b &= 2\\`
`c &= 3`
`}\right\}`
`\qquad`
`\eqalign{`
`ax + by &= c \\`
` x + 2y &= 3`
` }`

yields:
$$\left\{\begin{aligned} a &= 1 \\ b &= 2 \\ c &= 3 \end{aligned}\right\} \qquad \begin{aligned} ax + by &= c \\ x + 2y &= 3 \end{aligned}$$
see also: \eqalignno, the align environment, \tag</td></tr>
<tr><td>`\eqalignno`</td><td></td><td>equation alignment with optionally numbered (tagged) lines
`\eqalignno{ <math> & <math> & <equation tag> \cr <repeat as needed> }`
the first ampersand is placed where alignment is desired;
the second ampersand is used just before a tag;
if there is no tag, then the final `& <equation tag>` is omitted;
a double-backslash can be used in place of the `\cr` ;
the final `\\` or `\cr` is optional

Example:

`\eqalignno{`
`3x - 4y &= 5   &(\dagger) \cr`
`x  +  7 &= -2y &(\ddagger)\cr`
`      z &= 2`
`}`

yields:</td></tr>
</table>

		$3x - 4y = 5 \qquad (\dagger)$ $x + 7 = -2y \qquad (\ddagger)$ $z = 2$ see also: \eqalign, \leqalignno, the align environment
`\eqcirc` AMSsymbols	≖	`&#x2256;` class REL
`\eqsim` AMSsymbols	≂	`&#x2242;` class REL
`\eqslantgtr` AMSsymbols `\eqslantless` AMSsymbols	⪖ ⪕	`&##x2A96;` class REL `&##x2A95;` class REL
`\equiv`	≡	`&#x2261;` class REL

`Error Messages;`
`page processing log`
		When you're working with a MathJax page, you may want to see the log of messages generated during page processing (particularly if something has gone wrong).
		To do this, type
		`javascript:alert(MathJax.Message.Log())`
		in the browser's location URL box, and then refresh the page.
		If the alert box is too big to see the close button, just press ‘enter’ to close the alert box.

`\eta`	η	`&#x03B7;` class ORD lowercase Greek letter eta
`\eth` AMSsymbols	ð	`&#x00F0;` class ORD
`\exists`	∃	`&#x2203;` class ORD there exists see also: \nexists
`\exp`	exp	class OP exponential function; does not change size; default limit placement is the same in both inline and display modes; can change limit placement using \limits; see the Big Operators Table for examples

F

`\fallingdotseq` AMSsymbols	≒	`&#x2252;` class REL falling dot sequence; see also: \risingdotseq
`\fbox`		puts a box around argument; argument is in text mode equivalent to: `\boxed{\text{#1}}` `\fbox #1` where #1 is rendered as text Examples: `\boxed{Hi there!}` yields $\boxed{Hi there!}$ `\fbox{Hi there!}` yields $\boxed{\text{Hi there!}}$ see also: \boxed
`\Finv` AMSsymbols	Ⅎ	`&#x2132;` class ORD
`\flat`	♭	`&#x266D;` class ORD musical flat symbol see also: \natural, \sharp
`\forall`	∀	`&#x2200;` class ORD universal quantifier; for all; for every; for each
`\frac` AMSmath		fractions; displays differently in inline and display modes `\frac #1 #2` Examples: `\frac a b` yields (inline mode) $\frac a b$ `\frac a b` yields (display mode) $\dfrac a b$ `\frac{a-1}b-1` yields $\frac{a-1}b-1$ `\frac{a-1}{b-1}` yields $\frac{a-1}{b-1}$

		see also: \above, \abovewithdelims, \atop, \atopwithdelims, \cfrac, \dfrac, \genfrac, \over, \overwithdelims
`\frak`		class ORD turns on fraktur; affects uppercase and lowercase letters, and digits `{\frak ... }` Examples: `\frak ABCDEFGHIJKLMNOPQRSTUVWXYZ` yields $\mathfrak{ABCDEFGHIJKLMNOPQRSTUVWXYZ}$ `\frak 0123456789` yields $\mathfrak{0123456789}$ `\frak abcdefghijklmnopqrstuvwxyz` yields $\mathfrak{abcdefghijklmnopqrstuvwxyz}$ `{\frak AB}AB` yields $\mathfrak{AB}AB$ `\frak AB \rm AB` yields $\mathfrak{AB}\mathrm{AB}$ `{\frak AB \cal AB} AB` yields $\mathfrak{AB}\mathcal{AB}AB$ see also: \mathfrak
`\frown`	$\frown$	`&#x2322;` class REL see also: \smallfrown, \smallsmile, \smile

G

`\Game` `AMSsymbols`	$\Game$	`&#x2141;` class ORD
`\Gamma`	Γ	`&#x0393;` class ORD uppercase Greek letter gamma see also: \varGamma
`\gamma`	γ	`&#x03B3;` class ORD lowercase Greek letter gamma
`\gcd`	$\gcd$	class OP greatest common divisor; does not change size; can change limit placement using \limits and \nolimits; see the Big Operators Table for examples Examples: `\gcd_{\rm sub}^{\rm sup}` yields (inline mode) $\gcd_{\rm sub}^{\rm sup}$ `\gcd_{\rm sub}^{\rm sup}` yields (display mode) $\gcd\limits_{\rm sub}^{\rm sup}$

`\ge`
`\geq`
`\geqq` `AMSsymbols`
`\geqslant` `AMSsymbols`
	$\ge$
	$\geq$
	$\geqq$
	$\geqslant$
		`&#x2265;` `\ge`
		`&#x2265;` `\geq`
		`&#x2267;` `\geqq`
		`&#x2A7E;` `\geqslant`
		all class REL
		greater than or equal to
		see also: \ngeq, \ngeqq, \ngeqslant

`\genfrac` `AMSmath`		the most general command for defining fractions with optional delimiters, line thickness, and specified style `\genfrac #1 #2 #3 #4 #5 #6` where: • `#1` is the left delimiter (empty, for no left delimiter) • `#2` is the right delimiter (empty, for no right delimiter) • `#3` is the fraction bar thickness (set to 0pt to make it disappear) • `#4` is either 0, 1, 2, or 3, where: ○ 0 denotes `\displaystyle` ○ 1 denotes `\textstyle` ○ 2 denotes `\scriptstyle` ○ 3 denotes `\scriptscriptstyle` • `#5` is the numerator • `#6` is the denominator Example: `\genfrac(]{0pt}{2}{a+b}{c+d}` yields $\genfrac(]{0pt}{2}{a+b}{c+d}$ see also: \above, \abovewithdelims, \atop, \atopwithdelims,

		\cfrac, \dfrac, \frac, \over, \overwithdelims
`\gets`	$\gets$	`&#x2190;` class REL left arrow; non-stretchy
`\gg`	$\gg$	`&#x226B;` class REL
`\ggg` `AMSsymbols`	$\ggg$	`&#x22D9;` class REL
`\gggtr` `AMSsymbols`	$\gggtr$	`&#x22D9;` class REL
`\gimel` `AMSsymbols`	$\gimel$	`&#x2137;` class ORD Hebrew letter gimel
`\gtrapprox` `AMSsymbols`	$\gtrapprox$	`&#x2A86;` class REL
`\gnapprox` `AMSsymbols`	$\gnapprox$	`&#x2A8A;` class REL
`\gneq` `AMSsymbols`	$\gneq$	`&#x2A88;` class REL
`\gneqq` `AMSsymbols`	$\gneqq$	`&#x2269;` class REL
`\gvertneqq` `AMSsymbols`	$\gvertneqq$	`&#x2269;` class REL
`\gtrsim` `AMSsymbols`	$\gtrsim$	`&#x2273;` class REL
`\gnsim` `AMSsymbols`	$\gnsim$	`&#x22E7;` class REL

`\grave`	$\grave{}$	`&#x02CB;`
		grave accent
		`\grave #1`
		Usually, `#1` is a single letter; otherwise, accent is centered over argument.
		
		Examples:
		`\grave e` yields $\grave e$
		`\grave E` yields $\grave E$
		`\grave eu` yields $\grave e u$
		`\grave{eu}` yields $\grave{eu}$

`\gt`	$>$	`&#x003E;` class REL greater than see also: \ngtr
`\gtrdot` `AMSsymbols`	$\gtrdot$	`&#x22D7;` class REL
`\gtreqless` `AMSsymbols`	$\gtreqless$	`&#x22DB;` class REL
`\gtreqqless` `AMSsymbols`	$\gtreqqless$	`&#x2A8C;` class REL
`\gtrless` `AMSsymbols`	$\gtrless$	`&#x2277;` class REL

H

`\hat`	$\hat{}$	`&#x02CA;`
		non-stretchy hat accent
		`\hat #1`
		Usually, `#1` is a single letter; otherwise, accent is centered over argument.
		
		Examples:
		`\hat\imath` yields $\hat\imath$
		`\hat\jmath` yields $\hat\jmath$
		`\hat ab` yields $\hat a b$
		`\hat{ab}` yields $\hat{ab}$
		
		see also: \widehat

`\hbar`	$\hbar$	`&#x210F;` class ORD Planck's constant
`\hbox`		class ORD horizontal box; contents are treated as text, but you can switch to math mode inside; text appears in `\rm` `\hbox #1` Examples: `\hbox{\alpha a }\alpha a` yields \alpha a αa

		`\hbox{This is a sentence.}` yields This is a sentence. `\hbox{for all $x > 0$}` yields for all $x > 0$ in MathJax, these are essentially the same: \text, \mbox see also: \rm

`\hdashline`
`\hline`
		works in many of the environments to create a horizontal line (`\hline`), or a horizontal dashed line (`\hdashline`)
		
		Putting `\hdashline` or `\hline` first or last encases the entire structure
		(which is different from standard LaTeX behavior):
		
		`\begin{matrix}`
		`\hdashline`
		`x_{11} & x_{12} \\`
		`x_{21} & x_{22} \\`
		`x_{31} & x_{32}`
		`\end{matrix}`
		yields $\begin{array}{cc}\hline x_{11} & x_{12}\\ x_{21} & x_{22}\\ x_{31} & x_{32}\end{array}$
		
		`\begin{matrix}`
		`x_{11} & x_{12} \\`
		`x_{21} & x_{22} \\`
		`x_{31} & x_{32} \\`
		`\hline`
		`\end{matrix}`
		yields $\begin{array}{cc}x_{11} & x_{12}\\ x_{21} & x_{22}\\ x_{31} & x_{32}\\ \hline\end{array}$
		
		Putting `\hdashline` or `\hline` at the beginning of any subsequent row puts a line **over** that row:
		
		`\begin{matrix}`
		`x_{11} & x_{12} \\`
		`x_{21} & x_{22} \\`
		`\hline`
		`x_{31} & x_{32}`
		`\end{matrix}`
		yields $\begin{matrix}x_{11} & x_{12}\\ x_{21} & x_{22}\\ x_{31} & x_{32}\end{matrix}$
		
		You can combine effects, and put in struts (as desired) for additional vertical spacing:
		
		`\begin{matrix}`
		`\hline`
		`x_{11} & x_{12} \\`
		`x_{21} & x_{22} \strut \\`
		`\hdashline`
		`x_{31} & x_{32} \strut`
		`\end{matrix}`
		yields $\begin{array}{cc}\hline x_{11} & x_{12}\\ x_{21} & x_{22}\strut\\ \hdashline x_{31} & x_{32}\strut\end{array}$

`\heartsuit`	$\heartsuit$	`&#x2661;` class ORD see also: \clubsuit, \diamondsuit, \spadesuit
`\hom`	hom	class OP homomorphism; does not change size; default limit placement is the same in both inline and display modes; can change limit placement using \limits; see the Big Operators Table for examples
`\hookleftarrow` `\hookrightarrow`	$\hookleftarrow$ $\hookrightarrow$	`&#x21A9;` non-stretchy `&#x21AA;` non-stretchy both class REL

`\hphantom`		class ORD
		horizontal phantom
		
		Sometimes you want to *pretend* that something is there, for spacing reasons,
		but you don't want it to appear—you want it to be invisible—you want it to be a phantom.
		
		The box created by `\hphantom` has the width of its argument,
		but its height and depth are zero (so it doesn't contribute to any vertical spacing issues).
		In other words, `\hphantom` creates horizontal space equal to that produced by its argument,
		but doesn't create any vertical space.
		
		`\hphantom #1`
		
		Example:
		
		`\begin{array}{l}`
		`\text{Side Angle Side}\\`
		`\text{S}\hphantom{\text{ide }}\text{A}\hphantom{\text{ngle }}\text{S}`
		`\end{array}`
		
		yields
		
		Side Angle Side
		S A S

		see also: \phantom, \vphantom
`\href`		used to make a math object into a link `\href{ <url> } #1` where the argument (`#1`) is the clickable area Example: `\href{http://www.onemathematicalcat.org}{M^{A^{T^H}}}` yields $M^{A^{T^H}}$

`\hskip`		horizontal glue; horizontal space; horizontal skipping;
		`\hskip <dimen>`
		Example:
		`w\hskip1em i\hskip2em d\hskip3em e\hskip4em r`
		yields
		$w\hskip1em i\hskip2em d\hskip3em e\hskip4em r$
		in MathJax, these all behave the same: \hspace, \kern, \mkern, \mskip, \mspace

`\hslash` `AMSsymbols`	$\hslash$	`&#x210F;` class ORD perhaps an alternative form of Planck's constant

`\hspace`		horizontal glue; horizontal space; horizontal skipping
		`\hspace <dimen>`
		Example:
		`s\hspace7ex k\hspace6ex i\hspace5ex n\hspace4ex n\hspace3ex i\hspace2ex e\hspace1ex r`
		yields
		$s\hspace7ex k\hspace6ex i\hspace5ex n\hspace4ex n\hspace3ex i\hspace2ex e\hspace1ex r$
		in MathJax, these all behave the same: \hskip, \kern, \mkern, \mskip, \mspace

`\Huge`
`\huge`
		both class ORD
		turns on huge mode and an even bigger Huge mode
		`{\Huge ... }`
		`{\huge ... }`
		Examples:
		`\huge AaBb\alpha\beta123\frac ab\sqrt x` yields $\huge AaBb\alpha\beta123\frac ab\sqrt x$
		`{\huge A B} A B` yields ${\huge A B} A B$
		`A\alpha\huge A\alpha \Huge A\alpha` yields $A\alpha\huge A\alpha \Huge A\alpha$
		see also: \LARGE, \Large, \large

I

`\iddots`	$\iddots$
	Not in MathJax Library
		inner diagonal dots;
		
		This macro must be supplied by the user, if desired.
		Davide Cervone provided the code (given here) in the MathJax User Group.
		
		To use this macro, put the following definition in either inline or display mathematics:
		`$`
		`\def\iddots{`
		`  {\kern3mu\raise1mu{.}\kern3mu\raise6mu{.}\kern3mu\raise12mu{.}}}`
		`$`
		Then, in any subsequent mathematics:
		
		\iddots yields $\iddots$
		
		Instead of providing the definition inside math delimiters in the body,
		you can add the definition to your configuration using the `Macros` property of the `TeX` block:
		
		`<script type="text/x-mathjax-config">`
		`MathJax.Hub.Config({`
		`TeX: {`
		`Macros: {`
		`  iddots: "{\\kern3mu\\raise1mu{.}\\kern3mu\\raise6mu{.}\\kern3mu\\raise12mu{.}}"`
		`}}});`
		`</script>`

\idotsint AMSmath	$\idotsint$	class OP
		changes size;
		can change limit placement using \limits;
		see the Big Operators Table for examples

\iff	$\iff$	&#x27FA; with a thick space on both sides if and only if; is equivalent to; non-stretchy Example: A\iff B yields $A \iff B$

\iiiint AMSmath
\iiint
\iint
\int
	$\iiiint$
	$\iiint$
	$\iint$
	$\int$
		four occurrences of &#x222B;
		&#x222D;
		&#x222C;
		&#x222B;
		all class OP;
		see the Big Operators Table for examples
		
		Compare the different limit placements (both in display mode):
		\int_a^b yields $\displaystyle\int_a^b$
		\intop_a^b yields $\displaystyle\intop_a^b$
		
		see also: \intop

\intop	$\intop$	&#x222B; (with movable limits) class OP See the Big Operators Table for examples. see also: \iiiint, \iiint, \iint, \int
\Im	$\Im$	&#x2111; class ORD
\imath	$\imath$	&#x0131; class ORD a dotless ‘i’; better to use when accented Examples: \hat i yields $\hat i$ \hat\imath yields $\hat\imath$ see also: \jmath
\impliedby AMSsymbols	$\impliedby$	&#x27F8; with a thick space on both sides non-stretchy Example: P\impliedby Q yields $P \impliedby Q$
\implies AMSsymbols	$\implies$	&#x27F9; with a thick space on both sides non-stretchy Example: P\implies Q yields $P \implies Q$
\in	$\in$	&#x2208; class REL is in; is an element of; indicates membership in a set; see also: \ni, \notin, \owns
\inf	$\inf$	class OP infimum; least upper bound; does not change size; can change limit placement using \limits and \nolimits; see the Big Operators Table for examples Examples: \inf_{\rm limit} yields (inline mode) $\inf_{\rm limit}$ \inf_{\rm limit} yields (display mode) $\displaystyle\inf_{\rm limit}$ see also: \sup

`\infty`	∞	`&#x221E;` class ORD infinity
`\injlim` AMSmath	$\operatorname{inj\,lim}$	class OP injective limit; does not change size; can change limit placement using \limits and \nolimits; see the Big Operators Table for examples see also: \varinjlim
`\intercal` AMSsymbols	$\intercal$	`&#x22BA;` class BIN
`\iota`	ι	`&#x03B9;` class ORD lowercase Greek letter iota
`\it`		class ORD turns on math italic mode; to return to math italic mode if it had been turned off `{\it ... }` Examples: `{\bf ab \it ab} ab` yields $\mathbf{ab}\mathit{ab}ab$ `\rm for\ all\ {\it x}\ in\ \Bbb R` yields $\rm for\ all\ {\it x}\ in\ \Bbb R$ `\Delta\Gamma\Lambda{\it \Delta\Gamma\Lambda}` yields $\Delta\Gamma\Lambda{\it \Delta\Gamma\Lambda}$ see also: \mathit, \mit

J

`\jmath`	$\jmath$	`&#x0237;` class ORD a dotless 'j'; better to use when accented Examples: `\hat j` yields $\hat j$ `\hat\jmath` yields $\hat\jmath$ see also: \imath
`\Join` AMSsymbols	$\Join$	`&#x22C8;` class REL

K

`\kappa`	κ	`&#x03BA;` class ORD lowercase Greek letter kappa see also: \varkappa
`\ker`	$\ker$	class OP kernel; does not change size; default limit placement is the same in both inline and display modes; can change limit placement using \limits; see the Big Operators Table for examples
`\kern`		to get a specified amount of horizontal space; a negative argument forces 'backing up', so items can overlap `\kern <dimen>` Examples: `|\kern 2ex|\kern 2em|\kern 2pt|` yields $\vert\kern 2ex\vert\kern 2em\vert\kern 2pt\vert$ `\rm I\kern-2.5pt R` yields $\rm I\kern-2.5pt R$ in MathJax, these all behave the same: \hskip, \hspace, \mkern, \mskip, \mspace

L

`\Lambda` `\lambda`	Λ λ	uppercase Greek letter lambda `&#x039B;` class ORD lowercase Greek letter lambda `&#x03BB;` class ORD see also: \varLambda
`\land`	$\land$	logical AND `&#x2227;` class BIN see also: \lor, \wedge
`\langle`	$\langle$	left angle bracket; `&#x27E8;` class OPEN

		non-stretchy when used alone;
		stretchy when used with `\left` or `\right` (see below)
		Example:
		`\left\langle`
		`\matrix{a & b\cr c & d}` yields $\left\langle\begin{matrix}a & b\\ c & d\end{matrix}\right\rangle$
		`\right\rangle`
		see also: \rangle

`\LARGE`
`\Large`
`\large`
		turns on large typestyles; affects all math — all class ORD
		`{\LARGE ... }`
		`{\Large ... }`
		`{\large ... }`
		Examples:
		`\Large AaBb\alpha\beta123\frac ab` yields $\Large AaBb\alpha\beta123\frac ab$
		`{\Large A B} A B` yields ${\Large A B} A B$
		`AB \large AB \Large AB \LARGE AB` yields $AB \large AB \Large AB \LARGE AB$
		`\Large{AB}CD` yields $\Large{AB}CD$
		see also: \huge, \Huge

`\LaTeX`	$\LaTeX$	the LaTeX logo — class ORD Example: `\rm\LaTeX` yields $\rm\LaTeX$ see also: \TeX
`\lbrace`	$\{$	left brace: — class OPEN non-stretchy when used alone; stretchy when used with `\left` or `\right` (see below) Examples: `\lbrace \frac ab, c \rbrace` yields $\lbrace \frac ab, c \rbrace$ `\left\lbrace \frac ab, c \right\rbrace` yields $\left\lbrace \frac ab, c \right\rbrace$ see also: \rbrace, \{ \}
`\lbrack`	$[$	left bracket: — class OPEN non-stretchy when used alone; stretchy when used with `\left` or `\right` (see below); Examples: `\lbrack \frac ab, c \rbrack` yields $\lbrack \frac ab, c \rbrack$ `\left\lbrack \frac ab, c \right\rbrack` yields $\left\lbrack \frac ab, c \right\rbrack$ see also: \rbrack, [ ]

`\lceil`	$\lceil$	left ceiling; — `&#x2308;` class OPEN
		non-stretchy when used alone;
		stretchy when used with `\left` or `\right` (see below)
		Example:
		`\left\lceil`
		`\matrix{a & b\cr c & d}` yields $\left\lceil\begin{matrix}a & b\\ c & d\end{matrix}\right\rceil$
		`\right\rceil`
		see also: \rceil, \lfloor, \rfloor

`\ldotp`	$.$	lower dot, punctuation symbol — `&#x002E;` class PUNCT Examples: `\rm s \ldotp h` yields $\rm s \ldotp h$ `\rm s.h` yields $\rm s.h$ see also: \cdotp
`\ldots`	$\ldots$	lower dots; ellipsis; ellipses; dot dot dot — `&#x2026;` class INNER Example:

		`x_1,\ldots,x_n` yields $x_1,\ldots,x_n$ see also: \cdots, \dots

`\le`
`\leq`
`\leqq` AMSsymbols
`\leqslant` AMSsymbols
	≤
	≤
	≦
	⩽
		less than or equal to `&#x2264;` class REL
		less than or equal to `&#x2264;` class REL
		less than or equal to `&#x2266;` class REL
		less than or equal to `&#x2A7D;` class REL
		see also: \nleq, \nleqq, \nleqslant

`\leadsto` AMSsymbols	⇝	`&#x21DD;` class REL
`\left`		used for stretchy delimiters; see the Variable-Sized Delimiters Table for details Examples: `\left( \frac12 \right)` yields $\left(\frac12 \right)$ `\left\updownarrow \phantom{\frac12} \right\Updownarrow` yields $\left\updownarrow \right\Updownarrow$ see also: \right
`\leftarrow` `\Leftarrow`	← ⇐	left arrow; non-stretchy `&#x2190;` class REL left arrow; non-stretchy `&#x21D0;` class REL see also: \nleftarrow, \nLeftarrow
`\leftarrowtail` AMSsymbols	↢	left arrow tail; non-stretchy `&#x21A2;` class REL see also: \rightarrowtail
`\leftharpoondown` `\leftharpoonup`	↽ ↼	left harpoon arrow; non-stretchy `&#x21BD;` class REL left harpoon arrow; non-stretchy `&#x21BC;` class REL
`\leftleftarrows` AMSsymbols	⇇	left left arrows; non-stretchy `&#x21C7;` class REL
`\leftrightarrow` `\Leftrightarrow`	↔ ⇔	left right arrow; non-stretchy `&#x2194;` class REL left right arrow; non-stretchy `&#x21D4;` class REL see also: \nleftrightarrow, \nLeftrightarrow
`\leftrightarrows` AMSsymbols	⇆	left right arrows; non-stretchy `&#x21C6;` class REL
`\leftrightharpoons` AMSsymbols	⇋	left right harpoons; non-stretchy `&#x21CB;` class REL
`\leftrightsquigarrow` AMSsymbols	↭	left right squiqgle arrow; non-stretchy `&#x21AD;` class REL

`\leftroot`		used to fine-tune the placement of the index inside `\sqrt` or `\root` (see examples)
		`\sqrt[... \leftroot #1 ...]{...}`
		`\root ... \leftroot #1 ... \of {...}`
		where the argument is a small integer:
		a positive integer moves the index to the left;
		a negative integer moves the index to the right
		Examples:
		`\sqrt[3]{x}` yields $\sqrt[3]{x}$
		`\sqrt[3\leftroot1]{x}` yields $\sqrt[3\leftroot1]{x}$
		`\root 3 \of x` yields $\root 3 \of x$
		`\root 3\leftroot{-1} \of x` yields $\root 3\leftroot{-1} \of x$
		`\root 3\leftroot{-1}\uproot2 \of x` yields $\root 3\leftroot{-1}\uproot2 \of x$
		see also: \uproot, \root

`\leftthreetimes` AMSsymbols	⋋	`&#x22CB;` class BIN
`\leqalignno`		equation alignment with optionally numbered (tagged) lines; in TEX, `\leqalignno` puts the tags on the left, but MathJax doesn't implement this behavior; currently, tags appear in a column on the right separated from the equations by a fixed amount of space (so they don't work like tags in the AMS math environments); this may be fixed in a future version of MathJax `\leqalignno{ <math> & <math> & <equation tag> \cr <repeat as needed> }`

			the first ampersand is placed where alignment is desired;
			the second ampersand is used just before a tag;
			if there is no tag, then the final `& <equation tag>` is omitted;
			a double-backslash can be used in place of the `\cr`;
			the final `\\` or `\cr` is optional;
			output is the same in both inline and display modes
			(except for the amount of vertical space before and after);
			
			Example:
			`\leqalignno{`
			`3x - 4y &= 5   &(\dagger) \cr`
			`x  +  7 &= -2y &(\ddagger)\cr`
			`      z &= 2`
			`}`
			yields:
			$\begin{aligned} 3x-4y &= 5 && (\dagger) \\ x+7 &= -2y && (\ddagger) \\ z &= 2 \end{aligned}$
			see also: \eqalignno; the align environment	

`\lessapprox`	`AMSsymbols`	⪅	see also: \lnapprox	`&#x2A85;` class REL
`\lessdot`	`AMSsymbols`	⋖		`&#x22D6;` class REL
`\lesseqgtr`	`AMSsymbols`	⋚		`&#x22DA;` class REL
`\lesseqqgtr`	`AMSsymbols`	⪋		`&#x2A8B;` class REL
`\lessgtr`	`AMSsymbols`	≶		`&#x2276;` class REL
`\lesssim`	`AMSsymbols`	≲	see also: \lnsim	`&#x2272;` class REL
`\lfloor`		⌊	left floor; non-stretchy when used alone; stretchy when used with `\left` or `\right` see also: \rfloor, \lceil, \rceil	`&#x230A;` class OPEN
`\lg`		lg	does not change size; default limit placement is the same in both inline and display modes; can change limit placement using \limits; see the Big Operators Table for examples	class OP

`\lgroup`		⟮	left group;
			non-stretchy when used alone;
			stretchy when used with `\left` or `\right`
			
			Example:
			`\left\lgroup`
			`\matrix{a & b\cr c & d}`
			`\right\rgroup` yields $\left\lgroup\begin{matrix} a & b \\ c & d \end{matrix}\right\rgroup$
			
			see also: \rgroup
				`&#x27EE;` class OPEN

`\lhd`	`AMSsymbols`	⊲	left-hand diamond see also: \rhd	`&#x22B2;` class REL
`\lim`		lim	limit; does not change size; can change limit placement using `\limits` and `\nolimits`; see the Big Operators Table for examples Examples: `\lim_{n\rightarrow\infty} f(x) = \ell` (inline mode) yields $\lim_{n\rightarrow\infty} f(x) = \ell$ `\lim_{n\rightarrow\infty} f(x) = \ell` (display mode) yields $\displaystyle\lim_{n\rightarrow\infty} f(x) = \ell$	class OP
`\liminf`		lim inf	limit inferior; does not change size; can change limit placement using `\limits` and `\nolimits`; see the Big Operators Table for examples	class OP

		Examples: `\liminf_{n\rightarrow\infty} x_n = \ell` (inline mode) yields $\liminf_{n\rightarrow\infty} x_n = \ell$ `\liminf_{n\rightarrow\infty}\ x_n = \ell` (display mode) yields $\displaystyle\liminf_{n\rightarrow\infty}\ x_n = \ell$ see also: \varliminf
`\limits`		used to set limits above/below any token of class `OP`; see the Big Operators table for more information and examples Examples: `\int_a^b f(x)\,dx` (inline mode) yields $\int_a^b f(x)\,dx$ `\int\limits_a^b f(x)\,dx` (inline mode) yields $\int\limits_a^b f(x)\,dx$ `\int_a^b f(x)\,dx` (display mode) yields $\displaystyle\int_a^b f(x)\,dx$ `\int\limits_a^b f(x)\,dx` (display mode) yields $\displaystyle\int\limits_a^b f(x)\,dx$ `\mathop{x}\limits_0^1` yields $\mathop{x}\limits_0^1$ see also: \nolimits
`\limsup`	lim sup	limit superior; class OP does not change size; can change limit placement using `\limits` and `\nolimits`; see the Big Operators Table for examples Examples: `\limsup_{n\rightarrow\infty} x_n` (inline mode) yields $\limsup_{n\rightarrow\infty} x_n$ `\limsup_{n\rightarrow\infty}\ x_n` (display mode) yields $\displaystyle\limsup_{n\rightarrow\infty}\ x_n$ see also: \varlimsup
`\ll`	$\ll$	`&x226A;` class REL

`\llap`		left overlap class ORD
		`\llap #1`
		creates a box of width zero;
		the argument is then placed just to the left of this zero-width box
		(and hence will overlap whatever lies to the left);
		proper use of `\llap` and `\rlap` in math expressions is somewhat delicate
		Examples:
		`a\mathrel{{=}\llap{/}}b` yields $a\mathrel{{=}\llap{/}}b$ `{=}` forces the equal to not have `REL` spacing (since it is not adjacent to `ORD`'s) and \mathrel{ } forces the compound symbol (equal with overlapping slash) to be treated as a single `REL`
		`a\mathrel{{=}\llap{/\,}}b` yields $a\mathrel{{=}\llap{/\,}}b$ the thinspace '`\,`' improves the spacing
		`a=\mathrel{\llap{/\,}}b` yields $a=\mathrel{\llap{/\,}}b$ this works because the spacing between adjacent `REL`'s is zero
		see also: \rlap

`\llcorner` AMSsymbols `\lrcorner` AMSsymbols	⌞ ⌟	lower left corner `&#x2514;` class REL lower right corner `&#x2518;` class REL These are technically delimiters, but MathJax doesn't stretch them like it should. see also: \ulcorner, \urcorner
`\Lleftarrow` AMSsymbols	⇚	non-stretchy `&#x21DA;` class REL

\lll	AMSsymbols	⋘		&#x22D8;	class REL
\llless	AMSsymbols	⋘		&#x22D8;	class REL
\lmoustache		⎰	left moustache; non-stretchy when used alone; stretchy when used with `\left` or `\right` (see below) Example: `\left\lmoustache \phantom{\matrix{a & b\cr c & d}} \right\rmoustache` yields $\left\lmoustache\phantom{\matrix{a & b\cr c & d}}\right\rmoustache$ see also: \rmoustache	&#x23B0;	class OPEN
\ln		ln	natural logarithm; does not change size; default limit placement is the same in both inline and display modes; can change limit placement using \limits; see the Big Operators Table for examples		class OP
\lnapprox	AMSsymbols	⪉	see also: \lessapprox	&#x2A89;	class REL
\lneq	AMSsymbols	⪇	see also: \leq	&#x2A87;	class REL
\lneqq	AMSsymbols	≨	see also: \leqq	&#x2268;	class REL
\lnot		¬	logical not see also: \neg	&#x00AC;	class ORD
\lnsim	AMSsymbols	⋦	see also: \lesssim	&#x22E6;	class REL
\log		log	logarithm; does not change size; default limit placement is the same in both inline and display modes; can change limit placement using \limits; see the Big Operators Table for examples		class OP
\longleftarrow		⟵	non-stretchy	&#x27F5;	class REL
\Longleftarrow		⟸	non-stretchy	&#x27F8;	class REL
\longrightarrow		⟶	non-stretchy	&#x27F6;	class REL
\Longrightarrow		⟹	non-stretchy	&#x27F9;	class REL
\longleftrightarrow		⟷	non-stretchy	&#x27F7;	class REL
\Longleftrightarrow		⟺	non-stretchy	&#x27FA;	class REL
\longmapsto		⟼	long maps to see also: \mapsto	&#x27FC;	class REL
\looparrowleft	AMSsymbols	↫	non-stretchy	&#x21AB;	class REL
\looparrowright	AMSsymbols	↬	non-stretchy	&#x21AC;	class REL
\lor		∨	logical OR see also: \land, \vee	&#x2228;	class BIN
\lower			`\lower <dimen> #1` lowers the argument by the amount specified in <dimen>; in actual TeX, the argument to `\lower` (and `\raise`) must be an `\hbox`, but in MathJax it can be any expression (using an `\hbox` is allowed, but not required) Example: `l\lower 2pt {owe} r` yields $l\lower 2pt {owe} r$ see also: \raise		
\lozenge	AMSsymbols	◊		&#x25CA;	class ORD
\Lsh	AMSsymbols	↰	left shift; non-stretchy see also: \Rsh	&#x21B0;	class REL
\lt		<	less than	&#x003C;	class REL

		see also: \nless
\ltimes AMSsymbols	$\ltimes$	see also: \rtimes &#x22C9; class BIN
\lvert AMSmath \lVert AMSmath	$\lvert$ $\lVert$	both non-stretchy when used alone; &#x2223; class OPEN stretchy when used with \left or \right &#x2225; class OPEN Example: `\left\lvert\frac{\frac ab}{\frac cd}\right\rvert` yields $\left\lvert\frac{\frac ab}{\frac cd}\right\rvert$ see also: \rvert, \rVert, |, \\\|
\lvertneqq AMSsymbols	$\lvertneqq$	&#x2268; class REL

M

\maltese AMSsymbols	✠	&#x2720; class ORD
\mapsto	$\mapsto$	maps to; non-stretchy math operator &#x21A6; class REL see also: \longmapsto
\mathbb		blackboard-bold for uppercase letters and lowercase ‘k’; class ORD if lowercase blackboard-bold letters are not available, then they are typeset in a roman font \mathbb #1 Whether lower-case letters are displayed in blackboard-bold, or not, depends on the fonts being used. The MathJax web-based fonts don't have lowercase blackboard-bold, but the STIX fonts do; so users with the STIX fonts installed will be able to display lowercase blackboard-bold letters. Examples: `\mathbb R` yields $\mathbb R$ `\mathbb ZR` yields $\mathbb ZR$ `\mathbb{AaBbKk}Cc` yields $\mathbb{AaBbKk}Cc$ `\mathbb{ABCDEFGHIJKLMNOPQRSTUVWXYZ}` yields $\mathbb{ABCDEFGHIJKLMNOPQRSTUVWXYZ}$ see also: \Bbb
\mathbf		boldface for uppercase and lowercase letters and digits class ORD \mathbf #1 Examples: `\mathbf{AaBb\alpha\beta123}` yields $\mathbf{AaBb\alpha\beta123}$ `\mathbf ZR` yields $\mathbf ZR$ `\mathbf{uvw}xyz` yields $\mathbf{uvw}xyz$ see also: \bf, \boldsymbol
\mathbin		gives the correct spacing to make an object into a binary operator; class BIN binary operators have some extra space around them; creates an element of class BIN \mathbin #1 Examples: `a\text{op} b` yields $a\text{op} b$ `a\mathbin{\text{op}} b` yields $a\mathbin{\text{op}} b$ `a\Diamond b` yields $a\Diamond b$ `a\mathbin{\Diamond}b` yields $a\mathbin{\Diamond}b$
\mathcal		calligraphic font for uppercase letters and digits class ORD \mathcal #1 Examples: `\mathcal{ABCDEFGHIJKLMNOPQRSTUVWXYZ}` yields $\mathcal{ABCDEFGHIJKLMNOPQRSTUVWXYZ}$ `\mathcal{0123456789}` yields $\mathcal{0123456789}$ `\mathcal{abcdefghijklmnopqrstuvwxyz}` yields $\mathcal{abcdefghijklmnopqrstuvwxyz}$ `abcdefghijklmnopqrstuvwxyz` yields $abcdefghijklmnopqrstuvwxyz$ `\mathcal{AB}AB` yields $\mathcal{AB}AB$

		see also: \cal, \oldstyle

`\mathchoice`		provides content that is dependent on the current style (display, text, script, or scriptscript);
		can be used in defining a macro for general use
		`\mathchoice #1 #2 #3 #4`
		where:
		- #1 is rendered when the `\mathchoice` appears in display style
		- #2 is rendered when the `\mathchoice` appears in text style
		- #3 is rendered when the `\mathchoice` appears in script style
		- #4 is rendered when the `\mathchoice` appears in scriptscript style
		Examples:
		`\mathchoice{D}{T}{S}{SS}   (in display style)` yields D
		`\mathchoice{D}{T}{S}{SS}   (in text style)` yields T
		`\mathchoice{D}{T}{S}{SS}   (in script style)` yields $\scriptstyle S$
		`\mathchoice{D}{T}{S}{SS}   (in scriptscript style)` yields $\scriptscriptstyle SS$
		Here's a nice example from the $\TeX$Book:
		Define:
		`\def\puzzle{\mathchoice{D}{T}{S}{SS}}`
		Then:
		`\puzzle{\puzzle\over\puzzle^{\puzzle^\puzzle}}` yields (in display mode) $\displaystyle D\frac{T}{T^{S^{SS}}}$
		`\puzzle{\puzzle\over\puzzle^{\puzzle^\puzzle}}` yields (in inline mode) $T\frac{S}{S^{SS^{SS}}}$

`\mathclose`		forces the argument to be treated in the ‘closing’ class; for example, like ‘$)$’ and ‘$]$’; class CLOSE creates an element of class CLOSE `\mathclose #1` Examples: `a + \lt b\gt + c` yields $a+ < b > +c$ `a + \mathopen\lt b\mathclose\gt + c` yields $a + \mathopen{<}b\mathclose{>} + c$ see also: \mathopen
`\mathfrak`		fraktur font for uppercase and lowercase letters and digits (and a few other characters) class ORD `\mathfrak #1` Examples: `\mathfrak{ABCDEFGHIJKLMNOPQRSTUVWXYZ}` yields $\mathfrak{ABCDEFGHIJKLMNOPQRSTUVWXYZ}$ `\mathfrak{0123456789}` yields $\mathfrak{0123456789}$ `\mathfrak{abcdefghijklmnopqrstuvwxyz}` yields $\mathfrak{abcdefghijklmnopqrstuvwxyz}$ `\mathfrak{AB}AB` yields $\mathfrak{AB}AB$ see also: \frak
`\mathinner`		some constructions are meant to appear ‘inside’ other formulas, class INNER and should be surrounded by additional space in certain circumstances; this classification is forced on the argument by using `\mathinner` `\mathinner #1` Examples: `ab\text{inside}cd` yields $ab\text{inside}cd$ `ab\mathinner{\text{inside}}cd` yields $ab\mathinner{\text{inside}}cd$
`\mathit`		math italic mode class ORD `\mathit #1` Examples: `\rm abc \mathit{def} ghi` yields $\rm abc \mathit{def} ghi$ in MathJax, this is the same as: \mit and \it
`\mathop`		forces the argument to be treated in the ‘large operator’ class; for example, like ‘$\sum$’; class OP

		creates an element of class OP \mathop #1 Examples: `atbtc` yields $atbtc$ `a\mathop{t}b\mathop{t}c` yields $a\mathop{t}b\mathop{t}c$ `\star_a^b` yields (in display mode) $\star_a^b$ `\mathop{\star}_a^b` yields (in display mode) $\displaystyle\mathop{\star}_a^b$
\mathopen		forces the argument to be treated in the ‘opening’ class; for example, like ‘(’ and ‘[’; class OPEN creates an element of class OPEN \mathopen #1 Examples: `a + \lt b\gt + c` yields $a+<b>+c$ `a + \mathopen\lt b\mathclose\gt + c` yields $a + \mathopen\lt b\mathclose\gt + c$ see also: \mathclose
\mathord		forces the argument to be treated in the ‘ordinary’ class; for example, like ‘/’; class ORD spacing is determined by pairs of tokens; there is no extra spacing between adjacent ORD's (as in the second example below); there is extra spacing between an ORD and a BIN (as in the first example below); creates an element of class ORD \mathord #1 Examples: `a+b+c` yields $a+b+c$ `a\mathord{+}b\mathord{+}c` yields $a\mathord{+}b\mathord{+}c$ `1,234,567` yields $1,234,567$ `1\mathord{,}234{,}567` yields $1\mathord{,}234{,}567$
\mathpunct		forces the argument to be treated in the ‘punctuation’ class; for example, like ‘,’; class PUNCT punctuation tends to have some extra space after the symbol; returns an element of class PUNCT \mathpunct #1 Examples: `1.234` yields 1.234 `1\mathpunct{.}234` yields $1\mathpunct{.}234$
\mathrel		forces the argument to be treated in the ‘relation’ class; for example, like ‘$=$’ and ‘$>$’; class REL relations have a bit more space on both sides than binary operators; returns an element of class REL \mathrel #1 Examples: `a \# b` yields $a\#b$ `a \mathrel{\#} b` yields $a \mathrel{\#} b$
\mathring AMSmath	˚	&#x2DA; \mathring #1 Examples: `\mathring A` yields $\mathring A$ `\mathring{AB}C` yields $\mathring{AB}C$
\mathrm		roman typestyle for uppercase and lowercase letters class ORD \mathrm #1 Examples: `\mathrm{AaBb\alpha\beta123}` yields $\mathrm{AaBb\alpha\beta123}$ `\mathrm ZR` yields $\mathrm ZR$

		`\mathrm{uvw}xyz` yields $\mathrm{uvw}xyz$ see also: \rm
`\mathscr`		script typestyle for uppercase letters; if lowercase script letters are not available, then they are typeset in a roman typestyle — class ORD `\mathscr #1` Whether lower-case letters are displayed in script, or not, depends on the fonts being used. The MathJax web-based fonts don't have lowercase script, but the STIX fonts do; so users with the STIX fonts installed will be able to display lowercase script letters. Examples: `\mathscr{ABCDEFGHIJKLMNOPQRSTUVWXYZ}` yields $\mathscr{ABCDEFGHIJKLMNOPQRSTUVWXYZ}$ `\mathscr{0123456789}` yields $\mathrm{0123456789}$ `\mathscr{abcdefghijklmnopqrstuvwxyz}` yields $\mathrm{abcdefghijklmnopqrstuvwxyz}$ `abcdefghijklmnopqrstuvwxyz` yields $abcdefghijklmnopqrstuvwxyz$ `\mathscr{AB}AB` yields $\mathscr{AB}AB$ see also: \scr
`\mathsf`		sans serif typestyle for uppercase and lowercase letters and digits; also affects uppercase greek (as do the other font switches, like `\rm`, `\it`, `\bf`, `\mathrm`, `\mathit`, `\mathbf`, etc). — class ORD `\mathsf #1` Examples: `\mathsf{ABCDEFGHIJKLMNOPQRSTUVWXYZ}` yields $\mathsf{ABCDEFGHIJKLMNOPQRSTUVWXYZ}$ `\mathsf{0123456789}` yields $\mathsf{0123456789}$ `\mathsf{abcdefghijklmnopqrstuvwxyz}` yields $\mathsf{abcdefghijklmnopqrstuvwxyz}$ `\Delta\Gamma\Lambda\mathsf{\Delta\Gamma\Lambda}` yields $\Delta\Gamma\Lambda\mathsf{\Delta\Gamma\Lambda}$ `abcdefghijklmnopqrstuvwxyz` yields $abcdefghijklmnopqrstuvwxyz$ `\mathsf{AB}AB` yields $\mathsf{AB}AB$ see also: \sf
`\mathstrut`		an invisible box whose width is zero; its height and depth are the same as a parenthesis '('; can be used to achieve more uniform appearance in adjacent formulas — class ORD Examples: `\sqrt3 + \sqrt\alpha` yields $\sqrt3 + \sqrt\alpha$ `\sqrt{\mathstrut 3} + \sqrt{\mathstrut\alpha}` yields $\sqrt{\mathstrut 3} + \sqrt{\mathstrut\alpha}$
`\mathtt`		typewriter typestyle for uppercase and lowercase letters and digits; also affects uppercase Greek — class ORD `\mathtt #1` Examples: `\mathtt{ABCDEFGHIJKLMNOPQRSTUVWXYZ}` yields $\mathtt{ABCDEFGHIJKLMNOPQRSTUVWXYZ}$ `\mathtt{0123456789}` yields $\mathtt{0123456789}$ `\mathtt{abcdefghijklmnopqrstuvwxyz}` yields $\mathtt{abcdefghijklmnopqrstuvwxyz}$ `abcdefghijklmnopqrstuvwxyz` yields $abcdefghijklmnopqrstuvwxyz$ `\Delta\Gamma\Lambda\mathtt{\Delta\Gamma\Lambda}` yields $\Delta\Gamma\Lambda\mathtt{\Delta\Gamma\Lambda}$ `\mathtt{AB}AB` yields $\mathtt{AB}AB$ see also: \sf
`\matrix`		matrix (without any delimiters) `\matrix{ <math> & <math> ... \cr <repeat as needed> }` alignment occurs at the ampersands; a double-backslash can be used in place of the `\cr`; the final `\\` or `\cr` is optional Example: `\matrix{ a & b \cr c & d }` yields $\begin{matrix} a & b \\ c & d \end{matrix}$ see also: \array
`\max`	max	

		maximum; class OP does not change size; can change limit placement using `\limits` and `\nolimits`; see the Big Operators Table for examples Examples: `\max_{\rm sub}` yields (inline mode) $\max_{\rm sub}$ `\max_{\rm sub}` yields (display mode) $\displaystyle\max_{\rm sub}$ see also: \min
`\mbox`		creates a box just wide enough to hold the text in its argument; class ORD no linebreaks are allowed in the text; text appears in `\rm` `\mbox <text argument>` Examples: `a + b \mbox{ (are you paying attention?) } = c` yields $a + b \mbox{ (are you paying attention?) } = c$ `a + b \text{ (are you paying attention?) } = c` yields $a + b \text{ (are you paying attention?) } = c$ in MathJax, these are essentially the same: \text, \hbox see also: \rm
`\measuredangle` AMSsymbols	∡	&#x2221; class ORD
`\mho` AMSsymbols	℧	&#x2127; class ORD
`\mid`	$\mid$	the spacing is perfect for use in set-builder notation &#x2223; class REL Examples: `\{x | x\gt 1\}` yields $\{x\vert x\gt 1\}$ `\{x \mid x\gt 1\}` yields $\{x \mid x\gt 1\}$ see also: \nmid, \shortmid, \nshortmid
`\min`	min	minimum; class OP does not change size; can change limit placement using `\limits` and `\nolimits`; see the Big Operators Table for examples Examples: `\min_{\rm sub}` yields (inline mode) $\min_{\rm sub}$ `\min_{\rm sub}` yields (display mode) $\displaystyle\min_{\rm sub}$ see also: \max
`\mit`		math italic typestyle class ORD `\mit #1` Examples: `\mit{\Gamma\Delta\Theta\Omega}` yields $\mit{\Gamma\Delta\Theta\Omega}$ `\mathit{\Gamma\Delta\Theta\Omega}` yields $\mathit{\Gamma\Delta\Theta\Omega}$ `\Gamma\Delta\Theta\Omega` yields $\Gamma\Delta\Theta\Omega$ in MathJax, this is the same as: \mathit and \it
`\mkern`		`\mkern <dimen>` gives horizontal space Examples: `ab` yields ab `a\mkern18mu b` yields $a\mkern18mu b$ `a\mkern18pt b` yields $a\mkern18pt b$ in MathJax, these all behave the same: \hskip, \hspace, \kern, \mskip, \mspace
`\mod`	mod	modulus operator; modulo; the leading space depends on the style: displaystyle has 18 mu, others 12 mu; 2 thinspaces of following space; for things like equations modulo a number `\mod #1` Example: `3\equiv 5 \mod 2` yields $3\equiv 5 \mod 2$

		see also: \pmod, \bmod
`\models`	⊨	`&#x22A8;` class REL

`\moveleft`
`\moveright`
		shifts boxes to the left or right
		`\moveleft <dimen> <box>`
		`\moveright <dimen> <box>`
		In actual TEX, these require an `\hbox` (or some box) as an argument, and can only appear in vertical mode;
		MathJax is less picky: you don't need an actual box, and MathJax doesn't have a vertical mode;
		these are not really designed as user-level macros, but instead allow existing macros to work;
		the box takes up its original space (unlike something like `\llap` or `\rlap`), but its contents are shifted (without affecting its bounding box)
		
		Examples:
		`\rm tight` yields tight
		`\rm t\moveleft3pt ight` yields ti ght
		`\rm t\moveleft3pt i\moveleft3pt g\moveleft3pt h\moveleft3pt t` yields tight
		`\rm t\moveleft3pt i\moveleft6pt g\moveleft9pt h\moveleft12pt t` yields tight
		`\square\square\moveleft 2em {\diamond\diamond}` yields ◇◇□□
		`\square\square\moveright 2em {\diamond\diamond}` yields □□ ◇◇
		see also: \raise, \lower

`\mp`	∓	minus plus `&#x2213;` class BIN see also: \pm
`\mskip`		`\mskip <dimen>` gives horizontal space Examples: `ab` yields ab `a\mskip18mu b` yields $a\mskip18mu b$ `a\mskip18pt b` yields $a\mskip18pt b$ in MathJax, these all behave the same: \hskip, \hspace, \kern, \mkern, \mspace
`\mspace`		`\mspace <dimen>` gives horizontal space Examples: `ab` yields ab `a\mspace18mu b` yields $a\mskip18mu b$ `a\mspace18pt b` yields $a\mskip18pt b$ in MathJax, these all behave the same: \hskip, \hspace, \kern, \mkern, \mskip
`\mu`	μ	lowercase Greek letter mu `&#x03BC;` class ORD
`\multimap` AMSsymbols	⊸	`&#x22B8;` class REL

N

`\nabla`	∇	`&#x2207;` class ORD
`\natural`	♮	see also: \flat, \sharp `&#x266E;` class ORD
`\ncong` AMSsymbols	≇	not congruent `&#x2246;` class REL see also: \cong
`\ne`	≠	not equal `&#x2260;` class REL see also: equals, \neq
`\nearrow`	↗	northeast arrow; non-stretchy `&#x2197;` class REL see also: \nwarrow, \searrow, \swarrow
`\neg`	¬	negate; negation `&#x00AC;` class ORD see also: \lnot
`\negthinspace` AMSmath `\negmedspace` AMSmath `\negthickspace` AMSmath		negative thin space negative medium space negative thick space

		Examples: `ab` yields ab `a\negthinspace b` yields $a\negthinspace b$ `a\negmedspace b` yields $a\negmedspace b$ `a\negthickspace b` yields $a\negthickspace b$ see also: \thinspace
`\neq`	$\neq$	see also: equals, \ne `&#x2260;` class REL

`\newcommand`		for defining your own commands (control sequences, macros, definitions);
		`\newcommand` must appear (within math delimiters) before it is used;
		if desired, you can use the `TeX.Macros` property of the configuration to define macros in the head
		`\newcommand\myCommandName`
		`  [ <optional # of arguments, from 1 to 9> ]`
		`  { <replacement text> }`
		The bracketed # of arguments is omitted when there are no arguments.
		Example (no arguments):
		`\newcommand\myHearts`
		`  {\color{purple}{\heartsuit}\kern-2.5pt\color{green}{\heartsuit}}`
		`\myHearts\myHearts`
		yields: ♡♡♡♡
		A definition may take one or more arguments:
		Example (two arguments):
		`\newcommand\myHearts[2]`
		`  {\color{#1}{\heartsuit}\kern-2.5pt\color{#2}{\heartsuit}}`
		`\myHearts{red}{blue}`
		yields: ♡♡
		see also: \def, \newenvironment

`\newenvironment`		for defining your own environments;
		`\newenvironment` must appear (within math delimiters) before it is used
		`\newenvironment{myEnvironmentName}`
		`  [ <optional # of arguments, from 1 to 9> ]`
		`  { <replacement text for each occurrence of \begin{myEnvironmentName}> }`
		`  { <replacement text for each occurrence of \end{myEnvironmentName}> }`
		The bracketed # of arguments is omitted when there are no arguments.
		There must not be a command having the same name as the environment:
		for example, to use `\begin{myHeart}...\end{myHeart}` there may not be a command `\myHeart`.
		Example (no arguments):
		`\newenvironment{myHeartEnv}`
		`  {\color{purple}{\heartsuit}\kern-2.5pt\color{green}{\heartsuit}}`
		`  {\text{ forever}}`
		`\begin{myHeartEnv}`
		`\end{myHeartEnv}`
		yields: ♡♡ forever
		An environment may take one or more arguments:
		Example (two arguments):
		`\newenvironment{myHeartEnv}[2]`
		`  {\color{#1}{\heartsuit}\kern-2.5pt\color{#2}{\heartsuit}}`
		`  {\text{ forever}}`
		`\begin{myHeartEnv}{red}{blue}`
		`\end{myHeartEnv}`
		yields: ♡♡ forever
		see also: \def, \newcommand

`\newline`		line separator in alignment modes and environments in MathJax, these are essentially the same: \cr, \\

\nexists	AMSsymbols	$\nexists$	see also: \exists	&#x2204;	class ORD
\ngeq \ngeqq	AMSsymbols AMSsymbols	$\ngeq$ $\ngeqq$	not greater than or equal to not greater than or equal to see also: \geq, \geqq	&#x2271; &#x2271;	class REL class REL
\ngeqslant	AMSsymbols	$\ngeqslant$	slanted not greater than or equal to see also: \geqslant	&#x2A88;	class REL
\ngtr	AMSsymbols	$\ngtr$	not greater than see also: \gt	&#x226F;	class REL
\ni		$\ni$	backwards ‘in’; contains see also: \in	&#x220B;	class REL

\nleftarrow
\nLeftarrow
	AMSsymbols
	AMSsymbols
		$\nleftarrow$
		$\nLeftarrow$
			see also: \leftarrow, \Leftarrow	&#x219A;
				&#x21CD;
					class REL
					class REL

\nleftrightarrow
\nLeftrightarrow
	AMSsymbols
	AMSsymbols
		$\nleftrightarrow$
		$\nLeftrightarrow$
			see also: \leftrightarrow, \Leftrightarrow	&#x21AE;
				&#x21CE;
					class REL
					class REL

\nleq \nleqq	AMSsymbols AMSsymbols	$\nleq$ $\nleqq$	not less than or equal to not less than or equal to see also: \leq, \leqq	&#x2270; &#x2270;	class REL class REL
\nleqslant	AMSsymbols	$\nleqslant$	slanted not less than or equal to see also: \leqslant	&#x2A87;	class REL
\nless	AMSsymbols	$\nless$	see also: \lt	&#x226E;	class REL
\nmid	AMSsymbols	$\nmid$	see also: \mid	&#x2224;	class REL
\nobreakspace	AMSmath		Example: a\nobreakspace b yields $a~b$ in MathJax, this is the same as: \ (backslash space)	&#xA0;	class ORD
\nolimits			used to change the default placement of limits; only allowed on items of class OP Examples: \sum_{k=1}^n a_k yields (in display mode) $\displaystyle\sum_{k=1}^n a_k$ \sum\nolimits_{k=1}^n a_k yields (in display mode) $\displaystyle\sum\nolimits_{k=1}^n a_k$ see also: \limits		
\normalsize			turns on normal size {\normalsize ... } Example: \rm \scriptsize script \normalsize normal \large large yields $\rm \scriptsize script \normalsize normal \large large$ see also: \scriptsize		class ORD
\not		/	used to negate relations Examples: \not\gt yields $\not\gt$ \ngtr yields $\ngtr$	&#x002F;	class REL
\notag	AMSmath		used in AMS math environments that do automatic equation numbering, to suppress the equation number; since MathJax doesn't implement auto-numbering (as of version 1.1a), it is basically a no-op, although it *will* cancel an explicit \tag ; when auto-numbering is added, then this will work as expected; \notag is included now for compatibility with existing TeX code (to prevent throwing an error, even though it has no effect)		class ORD
\notin		$\notin$	see also: \in	&#x2209;	class REL

\nparallel	AMSsymbols	∦	not parallel see also: \parallel	&#x2226;	class REL
\nprec	AMSsymbols	⊀	see also: \prec	&#x2280;	class REL
\npreceq	AMSsymbols	⋠	see also: \preceq	&#x22E0;	class REL
\nrightarrow	AMSsymbols	↛		&#x219B;	class REL
\nRightarrow	AMSsymbols	⇏		&#x21CF;	class REL
			see also: \rightarrow, \Rightarrow		
\nshortmid	AMSsymbols	∤	see also: \mid, \shortmid	&#x2224;	class REL
\nshortparallel	AMSsymbols	∦	see also: \parallel, \shortparallel	&#x2226;	class REL
\nsim	AMSsymbols	≁	see also: \sim	&#x2241;	class REL
\nsubseteq	AMSsymbols	⊈		&#x2288;	class REL
\nsubseteqq	AMSsymbols	⫅̸		&#x2288;	class REL
			see also: \subseteq, \subseteqq		
\nsucc	AMSsymbols	⊁		&#x2281;	class REL
\nsucceq	AMSsymbols	⋡		&#x22E1;	class REL
			see also: \succ, \succeq		
\nsupseteq	AMSsymbols	⊉		&#x2289;	class REL
\nsupseteqq	AMSsymbols	⫆̸		&#x2289;	class REL
			see also: \supseteq, \supseteqq		
\ntriangleleft	AMSsymbols	⋪		&#x22EA;	class REL
\ntrianglelefteq	AMSsymbols	⋬		&#x22EC;	class REL
			see also: \triangleleft, \trianglelefteq		
\ntriangleright	AMSsymbols	⋫		&#x22EB;	class REL
\ntrianglerighteq	AMSsymbols	⋭		&#x22ED;	class REL
			see also: \triangleright, \trianglerighteq		
\nu		ν	lowercase Greek letter nu	&#x03BD;	class ORD
\nVDash	AMSsymbols	⊯		&#x22AF;	class REL
\nVdash	AMSsymbols	⊮		&#x22AE;	class REL
\nvDash	AMSsymbols	⊭		&#x22AD;	class REL
\nvdash	AMSsymbols	⊬		&#x22AC;	class REL
			see also: \Vdash, \vDash, \vdash		
\nwarrow		↖	northwest arrow; non-stretchy see also: \nearrow, \searrow, \swarrow	&#x2196;	class REL

O

\odot	⊙		&#x2299;	class BIN
\ominus	⊖		&#x2296;	class BIN
\oplus	⊕		&#x2295;	class BIN
\oslash	⊘		&#x2298;	class BIN
\otimes	⊗		&#x2297;	class BIN
\oint	∮	changes size; can change limit placement using \limits; see the Big Operators Table for examples	&#x222E;	class OP
\oldstyle		this is intended for oldstyle numbers; it is a switch that turns on oldstyle mode; the way it works in TeX is to select the caligraphic font (which is where the oldstyle numbers are stored), so it has the side effect of selecting caligraphic upper-case letters; MathJax does the same for compatibility `{\oldstyle ... }` Examples: `\oldstyle 0123456789` yields 0123456789 `\oldstyle ABCDEFGHIJKLMNOPQRSTUVWXYZ` yields ABCDEFGHIJKLMNOPQRSTUVWXYZ		class ORD

		\oldstyle abcdefghijklmnopqrstuvwxyz yields abcdefghijklmnopqrstuvwxyz abcdefghijklmnopqrstuvwxyz yields $abcdefghijklmnopqrstuvwxyz$ {\oldstyle AB}AB yields $\mathcal{AB}AB$ \oldstyle AB \rm AB yields $\mathcal{AB}\mathrm{AB}$ \oldstyle{AB}CD yields $\mathcal{ABCD}$ see also: \cal, \mathcal
\omega \Omega	ω Ω	lowercase Greek letter omega &#x03C9; class ORD uppercase Greek letter omega &#x03A9; class ORD see also: \varOmega
\omicron	o	lowercase Greek letter omicron &#x03BF; class ORD
\operatorname AMSmath		This is similar to \DeclareMathOperator, but rather than defining a macro, it produces an instance of an operator like \lim . class OP For example, \operatorname{myOp} is equivalent to the use of \myOp , after having defined \DeclareMathOperator{\myOp}{myOp} If displaystyle limits are desired in both inline and display modes, then use operatorname* instead of operatorname Examples: \operatorname{myFct}(x) yields $\operatorname{myFct}(x)$ \operatorname*{myFct}_a^b(x) yields (in inline mode) $\operatorname*{myFct}_a^b(x)$ See \DeclareMathOperator for further explanation and examples.
\over		general command for making fractions { <subformula1> \over <subformula2> } Creates a fraction: numerator: subformula1 denominator: subformula2 Examples: a \over b yields ${a \over b}$ a+1 \over b+2 yields ${a+1 \over b+2}$ {a+1 \over b+2}+c yields ${a+1 \over b+2}+c$ see also: \above, \abovewithdelims, \atop, \atopwithdelims, \cfrac, \dfrac, \frac, \genfrac, \overwithdelims
\overbrace		puts a (stretchy) over-brace over the argument; can use ‘^’ to place an optional superscript over the overbrace; can use ‘_’ to place an optional subscript below the argument \overbrace #1 Example: \overbrace{x + \cdots + x}^{n\rm\ times}_{\text{(note here)}} yields $\overbrace{x + \cdots + x}^{n\rm\ times}_{\text{(note here)}}$ see also: \underbrace
\overleftarrow \overrightarrow \overleftrightarrow	$\leftarrow$ $\rightarrow$ $\leftrightarrow$	&#x2190; stretchy over left arrow &#x2192; stretchy over right arrow &#x2194; stretchy over left right arrow \overleftarrow #1 \overrightarrow #1 \overleftrightarrow #1 Examples: \overleftarrow{\text{the argument}} yields $\overleftarrow{\text{the argument}}$ \overrightarrow{AB} yields $\overrightarrow{AB}$

		`\overrightarrow{AB\strut}` yields $\overrightarrow{AB\strut}$ `\overleftrightarrow{\hspace1in}` yields $\overleftrightarrow{\hspace1in}$
`\overline`	‾	stretchy overline &#x203E; **`\overline #1`** Examples: `\overline{AB}` yields $\overline{AB}$ `\overline a` yields $\overline a$ `\overline{\text{a long argument}}` yields $\overline{\text{a long argument}}$
`\overset`		**`\overset #1 #2`** oversets argument #1 (in scriptstyle) over argument #2 Examples: `\overset{\rm top}{\rm bottom}` yields $\overset{\rm top}{\rm bottom}$ `\overset ab` yields $\overset ab$ `a\,\overset{?}{=}\,b` yields $a\,\overset{?}{=}\,b$ see also: \atop, \underset
`\overwithdelims`		general command for making fractions; uses default thickness for fraction bar for current size { <subformula1> \overwithdelims <delim1> <delim2> <subformula2> } Creates a fraction: numerator `subformula1` denominator `subformula2` `delim1` is put before the fraction `delim2` is put after the fraction For an empty delimiter, use ‘.’ in place of the delimiter. Examples: `a \overwithdelims [ ] b` yields $a \overwithdelims [ ] b$ `a+1 \overwithdelims . | b+2` yields $a+1 \overwithdelims . \| b+2$ `{a+1 \overwithdelims \{ \} b+2}+c` yields ${a+1 \overwithdelims \{ \} b+2}+c$ see also: \above, \abovewithdelims, \atop, \atopwithdelims, \cfrac, \dfrac, \frac, \genfrac, \over
`\owns`	∋	see also: \ni, \in &#x220B; class REL

P

`\parallel`	∥	see also: \nparallel &#x2225; class REL
`\partial`	∂	Example: &#x2202; class ORD `\frac{\partial f}{\partial x}` yields $\frac{\partial f}{\partial x}$
`\perp`	⊥	perpendicular to &#x22A5; class REL
`\phantom`		phantom (both horizontal and vertical) class ORD Sometimes you want to *pretend* that something is there, for spacing reasons, but you don't want it to appear—you want it to be invisible—you want it to be a phantom. The box created by `\phantom` has width, height and depth equal to its argument. In other words, `\phantom` creates horizontal and vertical space equal to that of its argument, even though the argument isn't visible. **`\phantom #1`** Examples: `\sqrt{\frac ab}` `\sqrt{\phantom{\frac ab}}` yields $\sqrt{\frac ab}\sqrt{}$

		`\frac{2x+3y-\phantom{5}z}{\phantom{2}x+\phantom{3}y+5z}` yields $\frac{2x+3y-z}{x+y+5z}$ `\Gamma^{\phantom{i}j}_{i\phantom{j}k}` yields Γ^{j}_{ik} `\matrix{1&-1\cr 2&\phantom{-}3}` yields $\matrix{1&-1\cr 2&3}$ see also: \hphantom, \vphantom
`\phi` `\Phi`	ϕ Φ	lowercase Greek letter phi &##x03D5; class ORD uppercase Greek letter phi &#x03A6; class ORD see also: \varphi, \varPhi
`\pi` `\Pi`	π Π	lowercase Greek letter pi &#x03C0; class ORD uppercase Greek letter Pi &#x03A0; class ORD see also: \varpi, \varPi
`\pitchfork` AMSsymbols	$\pitchfork$	&#x22D4; class REL
`\pm`	$\pm$	plus or minus &x00B1; class BIN see also: \mp
`\pmatrix`		matrix enclosed in parentheses class OPEN `\pmatrix{ <math> & <math> ... \cr <repeat as needed> }` alignment occurs at the ampersands; a double-backslash can be used in place of the `\cr`; the final `\\` or `\cr` is optional Example: `A = \pmatrix{ a_{11} & a_{12} & \ldots & a_{1n} \cr a_{21} & a_{22} & \ldots & a_{2n} \cr \vdots & \vdots & \ddots & \vdots \cr a_{m1} & a_{m2} & \ldots & a_{mn} \cr }` yields $A = \pmatrix{ a_{11} & a_{12} & \ldots & a_{1n} \cr a_{21} & a_{22} & \ldots & a_{2n} \cr \vdots & \vdots & \ddots & \vdots \cr a_{m1} & a_{m2} & \ldots & a_{mn} \cr }$ see also: \matrix
`\pmb`		poor man's bold; class ORD it works by duplicating its argument slightly offset, giving a bold effect (at least in the horizontal direction); doesn't work well for horizontal lines, like $-$ or $+$ `\pmb #1` Examples: `a \pmb a \boldsymbol a` yields $a \pmb a \boldsymbol a$ `\pmb{a+b-c}\ \ a+b-c` yields $\pmb{a+b-c}\ \ a+b-c$
`\pmod`	(mod)	parenthesized modulus operator; parenthesized modulo; 18 mu of leading space before the opening parenthesis in display style; 8 mu of leading space before the opening parenthesis in other styles; 6 mu of space after the word `mod` `\pmod #1` Examples: `5\equiv 8 \pmod 3` yields $5\equiv 8 \pmod 3$ `\pmod{n+m}` yields $\pmod{n+m}$ see also: \mod, \bmod
`\pod`	()	parenthesized argument with leading space; 18 mu of leading space before the opening parenthesis in display style; 8 mu of leading space before the opening parenthesis in other styles `\pod #1` Examples: `x=y\pod{\text{inline mode}}` yields $x=y\pod{\text{inline mode}}$ `x=y\pod{\text{display mode}}` yields $x=y\quad(\text{display mode})$
`\Pr`	Pr	does not change size; class OP default limit placement can be changed using `\limits` and `\nolimits`; does not change size;

		see the Big Operators Table for more examples Examples: `\Pr_{\rm sub}` yields (inline mode) $\Pr_{\rm sub}$ `\Pr_{\rm sub}` yields (display mode) $\displaystyle\Pr_{\rm sub}$
`\prec`	$\prec$	see also: \nprec &#x227A; class REL
`\precapprox` AMSsymbols `\precnapprox` AMSsymbols	$\precapprox$ $\precnapprox$	&#x2AB7; class REL &#x2AB9; class REL
`\preccurlyeq` AMSsymbols	$\preccurlyeq$	&#x227C; class REL
`\preceq` `\precneqq` AMSsymbols	$\preceq$ $\precneqq$	&#x2AAF; class REL &#x2AB5; class REL see also: \npreceq
`\precsim` AMSsymbols `\precnsim` AMSsymbols	$\precsim$ $\precnsim$	&#x227E; class REL &#x22E8; class REL
`\prime`	$\prime$	prime character &#x2032; class ORD Examples: `f'` yields f' `f\prime` yields $f\prime$ `f^\prime` yields $f^\prime$ `f^{\prime\prime}` yields $f^{\prime\prime}$ `f''` yields f'' see also: \backprime, prime symbol
`\prod`	$\prod$	changes size; &#x220F; class OP can change limit placement using \limits and \nolimits; see the Big Operators Table for more examples Examples: `\prod_{j=1}^n` yields (in inline mode) $\prod_{j=1}^n$ `\prod_{j=1}^n` yields (in display mode) $\displaystyle\prod_{j=1}^n$
`\projlim` AMSmath	$\operatorname{proj\,lim}$	projective limit; class OP does not change size; can change limit placement using \limits and \nolimits; see the Big Operators Table for examples see also: \varprojlim
`\propto`	$\propto$	see also: \varpropto &#x221D; class REL
`\psi` `\Psi`	ψ Ψ	lowercase Greek letter psi &#x03C9; class ORD uppercase Greek letter psi &#x03A9; class ORD see also: \varPsi

Q

`\quad` `\qquad`		`\quad` is a 1em space `\qquad` is a 2em space Examples: `|\quad|\quad|` yields $\|\quad\|\quad\|$ `|\qquad\hphantom{|}|` yields $\|\qquad\hphantom{\|}\|$

R

`\raise`		

		`\raise <dimen> #1` raises the argument by the amount specified in <dimen>; in actual TEX, the argument to `\raise` (and `\lower`) must be an `\hbox` , but in MathJax it can be any expression (using an `\hbox` is allowed, but not required) Example: `h\raise 2pt {ighe} r` yields $h\raise 2pt {ighe} r$ see also: \lower

`\rangle`	⟩	right angle bracket; &#x27E9; class CLOSE
		non-stretchy when used alone;
		stretchy when used with `\left` or `\right` (see below)
		
		Example:
		`\left\langle`
		`\matrix{a & b\cr c & d}` yields $\left\langle\matrix{a & b\cr c & d}\right\rangle$
		`\right\rangle`
		
		see also: \langle

`\rbrace`	}	right brace; class CLOSE
		non-stretchy when used alone;
		stretchy when used with `\left` or `\right` (see below)
		
		Example:
		`\left\lbrace`
		`\matrix{a & b\cr c & d}` yields $\left\lbrace\matrix{a & b\cr c & d}\right\rbrace$
		`\right\rbrace`
		
		see also: \lbrace

`\rbrack`	]	right bracket; class CLOSE non-stretchy when used alone; stretchy when used with `\left` or `\right` (see below) Examples: `\lbrack \frac ab, c \rbrack` yields $\lbrack \frac ab, c \rbrack$ `\left\lbrack \frac ab, c \right\rbrack` yields $\left\lbrack \frac ab, c \right\rbrack$ see also: \lbrack, [ ]

`\rceil`	⌉	right ceiling; &#x2309; class CLOSE
		non-stretchy when used alone;
		stretchy when used with `\left` or `\right` (see below)
		
		Example:
		`\left\lceil`
		`\matrix{a & b\cr c & d}` yields $\left\lceil\matrix{a & b\cr c & d}\right\rceil$
		`\right\rceil`
		
		see also: \lceil, \lfloor, \rfloor

`\Re`	ℜ	&#x211C; class ORD
`\renewcommand`		equivalent to \newcommand; for clarity of code, you may choose to use `\renewcommand` when re-defining a macro; this is different from actual TEX , where `\renewcommand` only allows redefining of an existing command see also: \def, \newcommand, \newenvironment
`\require (non-standard)`		This is a MathJax-specific macro that can be used to load MathJax TEX extensions (like the AMSmath extension) from within math mode, rather than having to include it in the configuration. For example, `$\require{AMSsymbols}$` would cause MathJax to load the `extensions/TeX/AMSsymbols.js` file at that point. Since many people use MathJax in blogs and wikis that may not have all the extensions loaded, this makes it possible to load a lesser-used extension on a particular page, without having to include it in *every* page.
`\restriction AMSsymbols`	↾	&#x21BE; class REL
`\rfloor`	⌋	right floor; &#x230B; class CLOSE non-stretchy when used alone; stretchy when used with `\left` or `\right`

			see also: \lfloor, \lceil, \rceil
`\rgroup`		$\rgroup$	right group; `&#x27EE;` class CLOSE non-stretchy when used alone; stretchy when used with `\left` or `\right` Example: `\left\lgroup` `\matrix{a & b\cr c & d}` `\right\rgroup` yields $\left\lgroup\begin{matrix}a & b\\ c & d\end{matrix}\right\rgroup$ see also: \lgroup
`\rhd`	`AMSsymbols`	$\rhd$	right-hand diamond `&#x22B3;` class REL see also: \lhd
`\rho`		ρ	lowercase Greek letter rho `&#x0000;` class ORD see also: \varrho
`\right`			used for stretchy delimiters; see the Variable-Sized Delimiters Table for details Can be followed by: delimiter: sample code: yields: `(` `)` — `\left( \frac12 \right)` — $\left(\frac12 \right)$ `\updownarrow` `\Updownarrow` — `\left\updownarrow \phantom{\frac12} \right\Updownarrow` — $\left\updownarrow \right\Updownarrow$ see also: \left
`\rightarrow` `\Rightarrow`		$\rightarrow$ $\Rightarrow$	non-stretchy `&#x2192;` class REL non-stretchy `&#x21D2;` class REL see also: \nrightarrow, \nRightarrow, \to
`\rightarrowtail`	`AMSsymbols`	$\rightarrowtail$	right arrow tail; non-stretchy `&#x21A3;` class REL see also: \leftarrowtail
`\rightharpoondown` `\rightharpoonup`		$\rightharpoondown$ $\rightharpoonup$	non-stretchy `&#x21C1;` class REL non-stretchy `&#x21C0;` class REL see also: \leftharpoondown, \rightharpoondown
`\rightleftarrows`	`AMSsymbols`	$\rightleftarrows$	right left arrows; non-stretchy `&#x21C4;` class REL
`\rightleftharpoons`	`AMSsymbols`	$\rightleftharpoons$	right left harpoons; non-stretchy `&#x21CC;` class REL
`\rightrightarrows`	`AMSsymbols`	$\rightrightarrows$	right right arrows; non-stretchy `&#x21C9;` class REL
`\rightsquigarrow`	`AMSsymbols`	$\rightsquigarrow$	right squiggle arrow; non-stretchy `&#x21DD;` class REL
`\rightthreetimes`	`AMSsymbols`	$\rightthreetimes$	right three times `&#x22CC;` class BIN
`\risingdotseq`	`AMSsymbols`	$\risingdotseq$	rising dot sequence `&#x2253;` class REL see also: \fallingdotseq
`\rlap`			right overlap class ORD `\rlap #1` creates a box of width zero; the argument is then placed just to the right of this zero-width box (and hence will overlap whatever lies to the right) Example: `a\mathrel{\rlap{\;/}{=}}b` yields $a\mathrel{\rlap{\;/}{=}}b$ In this example, `{=}` forces the equal to not have `REL` spacing (since it is not adjacent to `ORD`'s); `\mathrel{}` forces the compound symbol (equal with overlapping slash) to be treated as a single `REL`; the `\;` improves the spacing for the slash. see also: \llap
`\rm`			turns on roman; affects uppercase and lowercase letters, and digits; class ORD also affects uppercase Greek `{\rm ... }`

		Examples: `\rm AaBb\alpha\beta123` yields $\rm AaBb\alpha\beta123$ `{\rm A B} A B` yields ${\rm A B} A B$ `\Delta\Gamma\Lambda{\rm\Delta\Gamma\Lambda}` yields $\Delta\Gamma\Lambda{\rm\Delta\Gamma\Lambda}$ `\rm AB \bf CD` yields $\rm AB \bf CD$ `\rm{AB}CD` yields $\rm{AB}CD$ see also: \text, \hbox, \mathrm
`\rmoustache`	⎱	right moustache; &#x23B1; class CLOSE non-stretchy when used alone; stretchy when used with `\left` or `\right` (see below) Example: `\left\lmoustache \phantom{\matrix{a & b\cr c & d}} \right\rmoustache` yields $\left\lmoustache \phantom{\matrix{a & b\cr c & d}} \right\rmoustache$ see also: \lmoustache
`\root ... \of`		`\root <index> \of #1` Examples: `\root 3 \of x` yields $\root 3 \of x$ `\root 13 \of {\frac 12}` yields $\root 13 \of {\frac 12}$ `\root n+1 \of x + 2` yields $\root n+1 \of x + 2$ see also: \sqrt, \leftroot, \uproot
`\Rrightarrow` AMSsymbols	⇛	non-stretchy &#x21DB; class REL
`\Rsh` AMSsymbols	↱	right shift; non-stretchy &#x21B1; class REL see also: \Lsh
`\rtimes` AMSsymbols	⋊	see also: \ltimes &#x22CA; class BIN
`\Rule (non-standard)`		a MathJax-specific macro giving a rule with a specified width, height, and depth `\Rule <dimenWidth> <dimenHeight> <dimenDepth>` where each argument is a `dimension` Examples: `x\Rule{3px}{1ex}{2ex}x` yields $x\Rule{3px}{1ex}{2ex}x$ `x\Rule{3px}{2ex}{1ex}x` yields $x\Rule{3px}{2ex}{1ex}x$
`\rvert` AMSmath `\rVert` AMSmath	| ‖	&#x2223; class CLOSE &#x2225; class CLOSE both non-stretchy when used alone; stretchy when used with `\left` or `\right` Example: `\left\lvert\frac{\frac ab}{\frac cd}\right\rvert` yields $\left\lvert\frac{\frac ab}{\frac cd}\right\rvert$ see also: \lvert, \lVert, |, \\\|

S

`\S`	§	section symbol &#xA700; class ORD

`\scr`		turns on script typestyle for uppercase letters; class ORD
		lowercase letters are in a roman typestyle
		`{ \scr ... }`
		Examples:
		`\scr ABCDEFGHIJKLMNOPQRSTUVWXYZ` yields $\scr ABCDEFGHIJKLMNOPQRSTUVWXYZ$
		`\scr 0123456789abcdefghijklmnopqrstuvwxyz` yields $\scr 0123456789abcdefghijklmnopqrstuvwxyz$
		`0123456789abcdefghijklmnopqrstuvwxyz` yields $0123456789abcdefghijklmnopqrstuvwxyz$
		`{\scr AB}AB`

		yields $\mathscr{A}\mathscr{B}AB$ `\scr AB \rm AB` yields $\mathscr{A}\mathscr{B}\mathrm{AB}$ `\scr{AB}CD` yields $\mathscr{A}\mathscr{B}\mathscr{C}\mathscr{D}$ see also: \mathscr

`\scriptscriptstyle`		used to over-ride automatic style rules and force scriptscript style; class ORD
		stays in force until the end of math mode or the braced group, or until another style is selected
		`{ \scriptscriptstyle ... }`
		Example:
		In inline mode:
		`\frac ab+\displaystyle\frac ab+\textstyle\frac ab+\scriptstyle\frac ab+\scriptscriptstyle\frac ab`
		yields:
		$\frac ab+\displaystyle\frac ab+\textstyle\frac ab+\scriptstyle\frac ab+\scriptscriptstyle\frac ab$
		Example:
		In inline mode:
		`\frac ab + {\scriptscriptstyle \frac cd + \frac ef} + \frac gh`
		yields
		$\frac ab + {\scriptscriptstyle \frac cd + \frac ef} + \frac gh$
		Example:
		In inline mode:
		`\frac ab + \scriptscriptstyle{\frac cd + \frac ef} + \frac gh`
		yields
		$\frac ab + \scriptscriptstyle{\frac cd + \frac ef} + \frac gh$
		see also: \displaystyle, \scriptstyle, \textstyle

`\scriptsize`		turns on script size class ORD `{ \scriptsize ... }` Example: `\rm \scriptsize script \normalsize normal \large large` yields $\rm \scriptsize script \normalsize normal \large large$ see also: \normalsize

`\scriptstyle`		used to over-ride automatic style rules and force script style; class ORD
		stays in force until the end of math mode or the braced group, or until another style is selected
		`{ \scriptstyle ... }`
		Example:
		In inline mode:
		`\frac ab+\displaystyle\frac ab+\textstyle\frac ab+\scriptstyle\frac ab+\scriptscriptstyle\frac ab`
		yields:
		$\frac ab+\displaystyle\frac ab+\textstyle\frac ab+\scriptstyle\frac ab+\scriptscriptstyle\frac ab$
		Example:
		In inline mode:
		`\frac ab + {\scriptstyle \frac cd + \frac ef} + \frac gh`
		yields
		$\frac ab + {\scriptstyle \frac cd + \frac ef} + \frac gh$
		Example:
		In inline mode:
		`\frac ab + \scriptstyle{\frac cd + \frac ef} + \frac gh`
		yields
		$\frac ab + \scriptstyle{\frac cd + \frac ef} + \frac gh$
		see also: \displaystyle, \scriptscriptstyle, \textstyle

`\searrow`	$\searrow$	southeast arrow; non-stretchy `&#x2198;` class ORD see also: \nearrow, \nwarrow, \swarrow
`\sec`	$\sec$	secant; class OP does not change size; default limit placement is the same in both inline and display modes; can change limit placement using \limits; see the Big Operators Table for more examples Examples: `\sec x` yields $\sec x$ `\sec(2x-1)` yields $\sec(2x-1)$ see also: \csc

`\setminus`	$\setminus$	set minus &#x2216; class BIN Examples: `A\setminus B` yields $A \setminus B$ `A\backslash B` yields $A\backslash B$ see also: \backslash
`\sf`		turns on sans serif mode for uppercase and lowercase letters and digits, and for uppercase Greek class ORD `{ \sf ... }` Examples: `\sf ABCDEFGHIJKLMNOPQRSTUVWXYZ` yields $\mathsf{ABCDEFGHIJKLMNOPQRSTUVWXYZ}$ `\sf 0123456789` yields $\mathsf{0123456789}$ `\sf abcdefghijklmnopqrstuvwxyz` yields $\mathsf{abcdefghijklmnopqrstuvwxyz}$ `ABCDE 01234 abcde` yields $ABCDE 01234 abcde$ `{\sf AB\Delta\Gamma\Lambda}\ AB\Delta\Gamma\Lambda` yields $\mathsf{AB\Delta\Gamma\Lambda}\ AB\Delta\Gamma\Lambda$ `\sf AB \rm AB` yields $\mathsf{AB}\mathrm{AB}$ `\sf{AB}CD` yields ABCD see also: \mathsf
`\sharp`	$\sharp$	musical sharp symbol &#x266F; class ORD see also: \flat, \natural
`\shortmid` AMSsymbols	$\shortmid$	see also: \nshortmid, \mid &#x2223; class REL
`\shortparallel` AMSsymbols	$\shortparallel$	see also: \nshortparallel &#x2225; class REL

`\shoveleft` AMSmath
`\shoveright` AMSmath
		forces flush left or flush right typesetting in a \multline or \multline* environment (see examples)
		
		Example:
		`\begin{multline}`
		`(a+b+c+d)^2 \\`
		`+ (e+f)^2 + (g+h)^2 + (i+j)^2 + (k+l)^2 \\`
		`+ (m+n)^2 + (o+p)^2 + (q+r)^2 + (s+t)^2 + (u+v)^2  \\`
		`+ (w+x+y+z)^2`
		`\end{multline}`
		
		yields
		$(a+b+c+d)^2$
		$+(e+f)^2+(g+h)^2+(i+j)^2+(k+l)^2$
		$+(m+n)^2+(o+p)^2+(q+r)^2+(s+t)^2+(u+v)^2$
		$+(w+x+y+z)^2$
		
		Example:
		`\begin{multline}`
		`(a+b+c+d)^2 \\`
		`\shoveleft{+ (e+f)^2 + (g+h)^2 + (i+j)^2 + (k+l)^2} \\`
		`\shoveright{+ (m+n)^2 + (o+p)^2 + (q+r)^2 + (s+t)^2 + (u+v)^2}  \\`
		`+ (w+x+y+z)^2`
		`\end{multline}`
		
		yields
		$(a+b+c+d)^2$
		$+(e+f)^2+(g+h)^2+(i+j)^2+(k+l)^2$
		$+(m+n)^2+(o+p)^2+(q+r)^2+(s+t)^2+(u+v)^2$
		$+(w+x+y+z)^2$

`\sideset` AMSmath		used for putting symbols at the four ‘corners’ of a large operator (like $\sum$ or $\prod$) `\sideset{_#1^#2}{_#3^#4} <large operator>` where: • #1 = lower left • #2 = upper left • #3 = lower right • #4 = upper right Examples:

		\sideset{_1^2}{_3^4}\sum yields $\sideset{_1^2}{_3^4}\sum$
\sigma \Sigma	σ Σ	lowercase Greek letter sigma &#x03C3; class ORD uppercase Greek letter sigma &#x03A3; class ORD see also: \sum, \varsigma, \varSigma
\sim \simeq	$\sim$ $\simeq$	&#x223C; class REL &#x2243; class REL see also: \nsim
\sin	sin	sine; class OP does not change size; default limit placement is the same in both inline and display modes; can change limit placement using \limits; see the Big Operators Table for more examples Examples: \sin x yields $\sin x$ \sin(2x-1) yields $\sin(2x-1)$ see also: \cos
\sinh	sinh	hyperbolic sine; class OP does not change size; default limit placement is the same in both inline and display modes; can change limit placement using \limits; see the Big Operators Table for more examples Examples: \sinh x yields $\sinh x$ \sinh(2x-1) yields $\sinh(2x-1)$ see also: \cosh
\skew		used to finely adjust the positioning on accents; particularly useful for adjusting superaccents (accents on accents); usually requires trial-and-error adjustment for proper positioning \skew #1 <accent> where #1 is a positive integer (the skew amount) Examples: \hat A yields $\hat A$ \skew7\hat A yields $\skew7\hat A$ \tilde M yields $\tilde M$ \skew{8}\tilde M yields $\skew{8}\tilde M$ \hat{\hat A} yields $\hat{\hat A}$ \skew4\hat{\hat A} yields $\skew4\hat{\hat A}$
\small		turns on small size; affects all math class ORD {\small ... } Example: \rm\tiny tiny \Tiny Tiny \small small \normalsize normal \large lg \Large Lg \LARGE LG \huge hg \Huge Hg yields tinyTinysmallnormallgLgLGhgHg \def\myExp{\alpha\frac xy} \tiny\myExp \Tiny\myExp \small\myExp \normalsize\myExp \large\myExp \Large\myExp \LARGE\myExp \huge\myExp \Huge\myExp yields $\alpha\frac{x}{y}\ \alpha\frac{x}{y}\ \alpha\frac{x}{y}\ \alpha\frac{x}{y}\ \alpha\frac{x}{y}\ \alpha\frac{x}{y}\ \alpha\frac{x}{y}\ \alpha\frac{x}{y}\ \alpha\frac{x}{y}$ ab{\small cd} cd yields $abcdcd$ ab\small{cd} cd yields $abcdcd$

		see also: \tiny, \Tiny, \normalsize, \large, \Large, \LARGE, \huge, \Huge
\smallfrown AMSsymbols	$\smallfrown$	small frown &#x2322; class REL see also: \frown, \smile, \smallsmile
\smallint	$\smallint$	small integral &#x222B; class OP see also: \int
\smallsetminus AMSsymbols	$\smallsetminus$	small set minus &#x2216; class BIN see also: \setminus
\smallsmile AMSsymbols	$\smallsmile$	small smile &#x2323; class REL see also: \smile, \frown, \smallfrown
\smash		By using \smash, \phantom, \hphantom, \vphantom, \rlap, \llap, class ORD you can typeset any mathematics, yet give it the width and/or height and/or depth of any other mathematics. \smash #1 Typesets the argument in a box with the same width as the argument, but with height and depth equal to zero. In other words: the argument of \smash is visible, and has its natural width, but does not contribute any height or depth to the surrounding mathematics (hence leaving the surrounding mathematics to dictate height and depth). Here are some scenarios: • to vertically \smash the box containing this and make it instead behave vertically like that : \smash{this}\vphantom{that} Examples: \sqrt{\frac ab} \sqrt{\smash{7}\vphantom{\frac ab}} yields $\sqrt{\frac ab}\sqrt{\smash{7}\vphantom{\frac ab}}$ \sqrt{\frac{\frac ab}{\frac cd}} \sqrt{\smash{\frac ef}\vphantom{\frac{\frac ab}{\frac cd}}} yields $\sqrt{\frac{\frac ab}{\frac cd}}\sqrt{\smash{\frac ef}\vphantom{\frac{\frac ab}{\frac cd}}}$ • to horizontally compress the box containing this and make it instead behave horizontally like that : \rlap{this}\hphantom{that} or \hphantom{that}\llap{this} Examples: \sqrt{\rm very\ wide} \sqrt{\rlap{\rm thin}\hphantom{\rm very\ wide}} yields $\sqrt{\rm very\ wide}\sqrt{\rlap{\rm thin}\hphantom{\rm very\ wide}}$ \sqrt{\rm very\ wide} \sqrt{\hphantom{\rm very\ wide}\llap{\rm thin}} yields $\sqrt{\rm very\ wide}\sqrt{\hphantom{\rm very\ wide}\llap{\rm thin}}$ • to both vertically smash and horizontally compress the box containing this and make it instead behave both vertically and horizontally like that : \rlap{\smash{this}} or \llap{\smash{this}} Examples: \sqrt{\matrix{a & b\cr c & d}} \sqrt{ \rlap{\smash{\rm Hi!}} \phantom{\matrix{a & b\cr c & d}} } yields $\sqrt{\begin{matrix}a & b\\ c & d\end{matrix}}\sqrt{\rlap{\smash{\rm Hi!}}\phantom{\begin{matrix}a & b\\ c & d\end{matrix}}}$ see also: \hphantom, \vphantom, \phantom, \llap, \rlap
\smile	$\smile$	smile &#x2323; class REL see also: \smallsmile, \frown, \smallfrown
\space		Example: &#xA0; class ORD a\space b yields $a\ b$ in MathJax, this is the same as: \ (backslash space), \nobreakspace
\Space (non-standard)		a MathJax-specific macro giving space with a specified width, height, and depth

		`\Space <dimenWidth> <dimenHeight> <dimenDepth>` where each argument is a `dimension` Compare: `a\Rule{5px}{4ex}{2ex}^b_c d` yields $a\rule{5px}{4ex}^b_c d$ `a\Space{5px}{4ex}{2ex}^b_c d` yields $a\hspace{5px}^b_c d$ see also: \Rule	
`\spadesuit`	♠	see also: \clubsuit, \diamondsuit, \heartsuit	&#x2660; class ORD
`\sphericalangle AMSsymbols`	∢		&#x2222; class ORD
`\sqcap` `\sqcup`	⊓ ⊔	square cap square cup	&#x2293; class BIN &#x2294; class BIN

`\sqrt`	√	square root (and other roots)
		`\sqrt #1`
		`\sqrt[n]{op}` is equivalent to `\root n \of {op}`
		Examples:
		`\sqrt x` yields $\sqrt x$
		`\sqrt xy` yields $\sqrt xy$
		`\sqrt{xy}` yields $\sqrt{xy}$
		`\sqrt[3]{x+1}` yields $\sqrt[3]{x+1}$
		see also: \root
			class ORD

`\sqsubset AMSsymbols`
`\sqsupset AMSsymbols`
	⊏
	⊐
		square subset
		square superset
			&#x228F; class REL
			&#x2290; class REL

`\sqsubseteq` `\sqsupseteq`	⊑ ⊒		&#x2291; class REL &#x2292; class REL
`\square AMSsymbols`	□		&#x25A1; class ORD
`\stackrel`		stack relations; you can stack anything (not just relations) but it creates an item of class `REL` (and usually the bottom is a `REL` to start with, but doesn't have to be) `\stackrel #1 #2` where #1 (in superscript style) is stacked on top of #2 Examples: `\stackrel{\rm def}{=}` yields $\stackrel{\rm def}{=}$ `\stackrel{\rm top}{\rm bottom}` yields $\stackrel{\rm top}{\rm bottom}$	
`\star`	⋆		&#x22C6; class BIN

`\strut`		an invisible box with no width, height 8.6pt and depth 3pt;
		note that `\mathstrut` changes with the current size, but `\strut` does not
		Examples:
		`\sqrt{(\ )}`
		`\sqrt{\mathstrut\rm mathstrut}`
		`\sqrt{\strut\rm strut}` yields $\sqrt{(\ )}\sqrt{\mathstrut\rm mathstrut}\sqrt{\strut\rm strut}$
		`\Tiny`
		`\sqrt{(\ )}`
		`\sqrt{\mathstrut\rm mathstrut}`
		`\sqrt{\strut\rm strut}` yields $\tiny\sqrt{(\ )}\sqrt{\mathstrut\rm mathstrut}\sqrt{\strut\rm strut}$
		`\Large`
		`\sqrt{(\ )}`
		`\sqrt{\mathstrut\rm mathstrut}`
		`\sqrt{\strut\rm strut}` yields $\Large\sqrt{(\ )}\sqrt{\mathstrut\rm mathstrut}\sqrt{\strut\rm strut}$	

			see also: \mathstrut

`\style`			[HTML] non-standard;
			used to apply CSS styling to mathematics
			`\style #1 #2`
			where:
			• `#1` is a (single) CSS style declaration
			• `#2` is the mathematics to be styled
			Examples:
			`\frac{\style{color:red}{x+1}}{y+2}` yields $\frac{x+1}{y+2}$
			`\style{background-color:yellow}{\frac{x+1}{y+2}}` yields $\frac{x+1}{y+2}$
			Example:
			Consider the following HTML/Javascript/MathJax code:
			`<button type="button" onclick="makeVisible()">Click to reveal answer</button>`
			`<script type="text/javascript">`
			`function makeVisible() {`
			`document.getElementById('answer').style.visibility = "visible";`
			`}`
			`</script>`
			`$$`
			`(x+1)^2 = \cssId{answer}\style{visibility:hidden}{(x+1)(x+1)}`
			`$$`
			Then, the result of this HTML/Javascript/MathJax code is:
			Click to reveal answer
			$(x+1)^2 =$
			see also: \class, \cssId

`\subset`		$\subset$	&#x2282; class REL
`\Subset`	`AMSsymbols`	$\Subset$	&#x22D0; class REL
`\subseteq`		$\subseteq$	&#x2286; class REL
`\subsetneq`	`AMSsymbols`	$\subsetneq$	&#x228A; class REL
`\subseteqq`	`AMSsymbols`	$\subseteqq$	&#x2AC5; class REL
`\subsetneqq`	`AMSsymbols`	$\subsetneqq$	&#x2ACB; class REL
			see also: \nsubseteq, \nsubseteqq, \varsubsetneq, \varsubsetneqq

`\substack`	`AMSmath`		use for multi-line subscripts or superscripts
			Examples:
			`\sum_{`
			`\substack{`
			`1\lt i\lt 3 \\`
			`1\le j\lt 5`
			`}}`
			`a_{ij}`
			yields (display mode) $\sum_{\substack{1<i<3\\1\le j<5}} a_{ij}$
			`^{\substack{\text{a very} \\`
			`\text{contrived} \\`
			`\text{example}`
			`}}`
			`{\frac ab}_{\substack{`
			`\text{isn't} \\`
			`\text{it?}`
			`}}`
			yields (display mode) ${}^{\substack{\text{a very}\\\text{contrived}\\\text{example}}}{\frac ab}_{\substack{\text{isn't}\\\text{it?}}}$
			see also: \begin{subarray}

`\succ`		$\succ$	see also: \nsucc &#x227B; class REL
`\succapprox`	`AMSsymbols`	$\succapprox$	&#x2AB8; class REL
`\succnapprox`	`AMSsymbols`	$\succnapprox$	&#x2ABA; class REL

`\succcurlyeq` AMSsymbols	$\succcurlyeq$	`&#x227D;` class REL
`\succeq` `\succneqq` AMSsymbols	$\succeq$ $\succneqq$	`&#x2AB0;` class REL `&#x2AB6;` class REL see also: \nsucceq
`\succsim` AMSsymbols `\succnsim` AMSsymbols	$\succsim$ $\succnsim$	`&#x227F;` class REL `&#x22E9;` class REL
`\sum`	$\sum$	summation notation; changes size; can change limit placement using \limits and \nolimits; see the Big Operators Table for examples see also: \Sigma `&#x2211;` class OP
`\sup`	sup	supremum; greatest lower bound; does not change size; can change limit placement using \limits and \nolimits; see the Big Operators Table for examples Examples: `\sup_{\rm limit}` yields (inline mode) $\sup_{\rm limit}$ `\sup_{\rm limit}` yields (display mode) $\displaystyle\sup_{\rm limit}$ see also: \inf class OP
`\supset`	$\supset$	`&#x2283;` class REL
`\Supset` AMSsymbols	$\Supset$	`&#x22D1;` class REL
`\supseteq` `\supsetneq` AMSsymbols `\supseteqq` AMSsymbols `\supsetneqq` AMSsymbols	$\supseteq$ $\supsetneq$ $\supseteqq$ $\supsetneqq$	`&#x2287;` class REL `&#x228B;` class REL `&#x2AC6;` class REL `&#x2ACC;` class REL see also: \nsupseteq, \nsupseteqq, \varsupsetneq, \varsupsetneqq
`\surd`	$\surd$	`&#x221A;` class ORD
`\swarrow`	$\swarrow$	southwest arrow; non-stretchy see also: \nearrow, \nwarrow, \searrow `&#x2199;` class REL

T

`\tag` AMSmath		used primarily in AMS math environments to get tags (equation numbers, labels);
		can, however, be used on any equation;
		the argument of `\tag` is typeset in text mode, but math mode can be used within the text:
		for example, `\tag{\$\bullet\$}`
		You can use dollar signs in text-mode regardless of the settings of the `inlineMath` delimiters in the tex2jax preprocessor.
		`\tag #1`
		Example:
		`\eqalign{`
		`3x - 4y &= 5\cr`
		`x + 7 &= -2y`
		`}`
		`\tag{3.1c}`
		yields $\begin{aligned}3x-4y&=5\\x+7&=-2y\end{aligned}\qquad(3.1c)$

`\tan`	tan	tangent; does not change size; default limit placement is the same in both inline and display modes; can change limit placement using \limits; see the Big Operators Table for more examples Examples: `\tan x` yields $\tan x$ `\tan(2x-1)` yields $\tan(2x-1)$ class OP

		see also: \cot
`\tanh`	$\tanh$	hyperbolic tangent; class OP does not change size; default limit placement is the same in both inline and display modes; can change limit placement using \limits; see the Big Operators Table for more examples Examples: `\tanh x` yields $\tanh x$ `\tanh(2x-1)` yields $\tanh(2x-1)$ see also: \cosh, \sinh
`\tau`	τ	lowercase Greek letter tau `&#x03C4;` class ORD
`\tbinom` AMSmath		notation commonly used for binomial coefficients; in textstyle `\tbinom #1 #2` Examples: `\tbinom n k` yields (inline mode) $\tbinom n k$ `\tbinom n k` yields (display mode) $\tbinom n k$ `\binom n k` yields (display mode) $\binom n k$ `\tbinom{n-1}k-1` yields $\tbinom{n-1}k-1$ `\tbinom{n-1}{k-1}` yields $\tbinom{n-1}{k-1}$ see also: \binom, \choose, \dbinom
`\TeX`	$\TeX$	the TeX logo class ORD Examples: `\TeX` yields $\TeX$ `\rm\TeX` yields $\rm\TeX$ see also: \LaTeX

`\text`
`\textbf`
`\textit`
`\textrm`
		text class ORD
		boldface text
		italic text
		roman text
		
		used to produce text-mode material (in a given font) within a mathematical expression;
		MathJax does not process any macros within the text (unlike TeX itself);
		you can get math mode within the text using `\(...\)` delimiters
		`\text #1`
		`\textbf #1`
		`\textit #1`
		`\textrm #1`
		Example:
		`|x| = x \text{ for all \(x \ge 0\)}` yields $\lvert x\rvert = x \text{ for all } x \ge 0$
		`\text{\alpha in text mode }\alpha` yields \alpha in text mode α
		`\textbf{\alpha in textbf mode }\alpha` yields **\alpha in textbf mode** α
		`\textit{\alpha in textit mode }\alpha` yields *\alpha in textit mode* α
		`\textrm{\alpha in textrm mode }\alpha` yields \alpha in textrm mode α
		
		in MathJax, `\text` is the same as: \hbox, \mbox
		see also: \rm

`\textstyle`		used to over-ride automatic style rules and force text (inline) style; class ORD
		stays in force until the end of math mode or the braced group, or until another style is selected
		`{ \textstyle ... }`
		Example:
		In display mode:
		`\frac ab + {\textstyle \frac cd + \frac ef} + \frac gh`
		yields

		$\frac{a}{b} + \tfrac{c}{d} + \tfrac{e}{f} + \frac{g}{h}$ Example: In inline mode: `\frac ab+{\displaystyle\frac ab}+\frac ab+\scriptstyle\frac ab+\scriptscriptstyle\frac ab` yields: $\frac{a}{b} + {\displaystyle\frac{a}{b}} + \frac{a}{b} + {\scriptstyle\frac{a}{b} + {\scriptscriptstyle\frac{a}{b}}}$ see also: \displaystyle, \scriptstyle, \scriptscriptstyle
`\tfrac` AMSmath		textstyle fraction `\tfrac #1 #2` Examples: `\tfrac ab \frac ab` (display mode) yields $\tfrac{a}{b}\ \dfrac{a}{b}$ `\tfrac ab \frac ab` (inline mode) yields $\tfrac{a}{b}\ \tfrac{a}{b}$ see also: \frac, \dfrac
`\therefore` AMSsymbols	$\therefore$	`&#x2234` class REL
`\theta` `\Theta`	θ Θ	lowercase Greek letter theta `&#x03B8;` class ORD uppercase Greek letter theta `&#x0398;` class ORD see also: \vartheta, \varTheta
`\thickapprox` AMSsymbols	$\thickapprox$	Example: `&#x2248;` class REL `approx\ \ \thickapprox` yields $\approx\ \ \thickapprox$ see also: \approx
`\thicksim` AMSsymbols	$\thicksim$	Example: `&#x223C;` class REL `sim\ \ \thicksim` yields $\sim\ \ \thicksim$
`\thinspace`		thin space; normally $\frac{1}{6}$ of a quad Example: thinspaces between letters: $a\,b\,c\,d$ see also: symbols for spaces, \negthinspace
`\tilde`	˜	non-stretchy tilde accent `&#x02DC;` `\tilde #1` Usually, `#1` is a single letter; otherwise, accent is centered over argument. Examples: `\tilde e` yields $\tilde e$ `\tilde E` yields $\tilde E$ `\tilde eu` yields $\tilde e u$ `\tilde{eu}` yields $\tilde{eu}$
`\times`	$\times$	`&#x00D7;` class BIN
`\tiny`		turns on tiny; a bit smaller than `\Tiny` class ORD `{\tiny ... }` Examples: `\tiny AaBb\alpha\beta123` yields $\tiny AaBb\alpha\beta123$ `{\tiny A B} A B` yields ${\tiny AB}AB$ `\tiny AB \Tiny CD` yields $\tiny AB \Tiny AB$ `\tiny{AB}CD` yields $\tiny ABCD$
`\Tiny` non-standard		turns on Tiny; a bit bigger than `\tiny` class ORD `{\Tiny ... }` Examples: `\Tiny AaBb\alpha\beta123` yields $\Tiny AaBb\alpha\beta123$ `{\Tiny A B} A B` yields ${\Tiny AB}AB$ `\Tiny AB \tiny CD` yields $\Tiny AB \tiny AB$ `\Tiny{AB}CD` yields $\Tiny ABCD$
`\to`	$\to$	non-stretchy `&#x2192;` class REL see also: \rightarrow

tool tips		Tool tips are not built into MathJax, but you can click here to benefit from a posting by Davide P. Cervone (April 2011) at the MathJax Users Group.
\top	⊤	&#x22A4; class ORD
\triangle \triangledown AMSsymbols	△ ▽	&#x25B3; class ORD &#x25BD; class ORD see also: \ntriangleleft, \ntriangleright, \vartriangle, \vartriangleleft, \vartriangleright
\triangleleft \triangleright	◃ ▹	&#x25C3; class BIN &#x25B9; class BIN see also: \ntriangleleft, \ntriangleright, \vartriangle, \vartriangleleft, \vartriangleright
\trianglelefteq AMSsymbols \trianglerighteq AMSsymbols	⊴ ⊵	&#x22B4; class REL &#x22B5 class REL see also: \ntrianglelefteq, \ntrianglerighteq
\triangleq AMSsymbols	≜	&#x225C; class REL
\tt		turns on typewriter type class ORD `{\tt ... }` Examples: `\tt AaBb\alpha\beta123` yields $\mathtt{AaBb}\alpha\beta\mathtt{123}$ `{\tt A B} A B` yields $\mathtt{AB}AB$ `\tt AB \rm CD` yields $\mathtt{AB}\mathrm{AB}$ `\tt{AB}CD` yields $\mathtt{ABCD}$
\twoheadleftarrow AMSsymbols \twoheadrightarrow AMSsymbols	↞ ↠	non-stretchy &#x219E; class REL non-stretchy &#x21A0; class REL

U

\ulcorner AMSsymbols \urcorner AMSsymbols	┌ ┐	upper left corner &#x250C; class REL upper right corner &#x2510; class REL These are technically delimiters, but MathJax doesn't stretch them. They are valid after `\left`, `\right`, and the various `\big` commands. see also: \llcorner, \lrcorner
\underbrace		puts a (stretchy) under-brace under the argument; can use '^' to place an optional superscript over the argument; can use '_' to place an optional subscript below the underbrace `\underbrace #1` Example: `\underbrace{x + \cdots + x}_{n\rm\ times}^{\text{(note here)}}` yields $\underbrace{x+\cdots+x}_{n\rm\ times}^{\text{(note here)}}$ see also: \overbrace

\underleftarrow
\underrightarrow
\underleftrightarrow
	←
	→
	↔
		stretchy under left arrow &#x2190;
		stretchy under right arrow &#x2192;
		stretchy under left right arrow &#x2194;
		`\underleftarrow #1`
		`\underrightarrow #1`
		`\underleftrightarrow #1`
		Examples:
		`\underleftarrow{\text{the argument}}` yields $\underleftarrow{\text{the argument}}$
		`\underrightarrow{AB}` yields $\underrightarrow{AB}$
		`\underrightarrow{AB\strut}` yields $\underrightarrow{AB\strut}$
		`\underleftrightarrow{\hspace1in}` yields $\underleftrightarrow{\hspace{1in}}$

\underline	_	stretchy underline \underline #1 Examples: \underline{AB} yields $\underline{AB}$ \underline a yields $\underline{a}$ \underline{\text{a long argument}} yields $\underline{\text{a long argument}}$	&#x005F;
\underset		\underset #1 #2 undersets argument #1 (in scriptstyle) under argument #2; the top item is properly aligned with the surrounding text (their baselines match) Examples: \underset{\rm bottom}{\rm top} yields $\underset{\rm bottom}{\rm top}$ \underset ab yields $\underset{a}{b}$ see also: \overset	
\unicode non-standard		implements a \unicode{} extension to TEX that allows arbitrary unicode code points to be entered in mathematics; can optionally specify height and depth of character (width is determined by browser); can optionally specify the default font from which to take the character; once a size and font are provided for a given unicode point, they need not be specified again in subsequent \unicode{} calls for that character See MathJax TeX and LateX Support; Unicode Support for more details. \unicode[optHeight,optDepth][optFont]#1 Examples: \unicode{x263a} yields ☺ &#x263a; yields (in math mode) ☺ \unicode[.55,0.05]{x22D6} yields ⋖ less-than with dot, with height 0.55em and depth 0.05em \unicode[.55,0.05][Geramond]{x22D6} yields ⋖ same, taken from Geramond font \unicode[Geramond]{x22D6} yields ⋖ same, but with default (height,depth) of (0.8em,0.2em)	class ORD
\unlhd AMSsymbols	⊴	underlined left-hand (left-pointing) diamond	&#x22B4; class REL
\unrhd AMSsymbols	⊵	underlined right-hand (right-pointing) diamond	&#x22B5; class REL
\uparrow	↑	non-stretchy	&#x2191; class REL
\Uparrow	⇑	non-stretchy	&#x21D1; class REL
\updownarrow	↕	non-stretchy	&#x2195; class REL
\Updownarrow	⇕	non-stretchy	&#x21D5; class REL
\upharpoonleft AMSsymbols	↿	non-stretchy	&#x21BF; class REL
\upharpoonright AMSsymbols	↾	non-stretchy	&#x21BE; class REL
\uplus	⊎		&#x228E; class BIN
\uproot		used to fine-tune the placement of the index inside \sqrt or \root (see examples) \sqrt[... \uproot #1 ...]{...} \root ... \uproot #1 ... \of {...} where the argument is a small integer: a positive integer moves the index up; a negative integer moves the index down In actual TeX, \uproot is not allowed in \root , so this is a difference between MathJax and TEX. Examples: \sqrt[3]{x} yields $\sqrt[3]{x}$	

			`\sqrt[3\uproot2]{x}` yields $\sqrt[3]{x}$ `\root 3 \of x` yields $\sqrt[3]{x}$ `\root 3\uproot{-2} \of x` yields $\sqrt[3]{x}$ see also: \leftroot, \root
`\upsilon` `\Upsilon`		υ Υ	lowercase Greek letter upsilon &#x03C5; class ORD uppercase Greek letter upsilon &#x03A5; class ORD see also: \varupsilon, \varUpsilon
`\upuparrows`	`AMSsymbols`	$\upuparrows$	non-stretchy &#x21C8; class REL

V

`\varDelta`	`AMSsymbols`	$\varDelta$	uppercase Greek letter delta; variant &#x0394; class ORD see also: \Delta
`\varepsilon`		ε	lowercase Greek letter epsilon; variant &#x03B5; class ORD see also: \epsilon
`\varGamma`	`AMSsymbols`	$\varGamma$	uppercase Greek letter gamma; variant &#x0393; class ORD see also: \Gamma
`\varinjlim`	`AMSmath`	$\varinjlim$	injective limit; variant; does not change size; can change limit placement using \limits and \nolimits; see the Big Operators Table for examples class OP see also: \injlim
`\varkappa`	`AMSsymbols`	$\varkappa$	lowercase Greek letter kappa; variant &#x03F0; class ORD see also: \kappa
`\varLambda`	`AMSsymbols`	$\varLambda$	uppercase Greek letter lambda; variant &#x039B; class ORD see also: \Lambda
`\varlimsup` `\varliminf`	`AMSmath` `AMSmath`	$\varlimsup$ $\varliminf$	limit superior; variant class OP limit inferior; variant class OP do not change size; can change limit placement using `\limits` and `\nolimits`; see the Big Operators Table for examples see also: \limsup, \liminf
`\varnothing`	`AMSsymbols`	$\varnothing$	see also: \emptyset &#x2205; class ORD
`\varOmega`	`AMSsymbols`	$\varOmega$	uppercase Greek letter omega; variant &#x03A9; class ORD see also: \Omega
`\varphi`		φ	lowercase Greek letter phi; variant &#x03C6; class ORD see also: \phi
`\varPhi`	`AMSsymbols`	$\varPhi$	uppercase Greek letter phi; variant &#x03A6; class ORD see also: \Phi
`\varpi`		ϖ	lowercase Greek letter pi; variant &#x03D6; class ORD see also: \pi
`\varPi`	`AMSsymbols`	$\varPi$	uppercase Greek letter pi; variant &#x03A0; class ORD see also: \Pi
`\varprojlim`	`AMSmath`	$\varprojlim$	projective limit; variant; does not change size; can change limit placement using \limits and \nolimits; see the Big Operators Table for examples see also: \projlim
		$\propto$	

`\varpropto`	`AMSsymbols`		proportional to; variant see also: \propto	`&#x221D;`	class REL
`\varPsi`	`AMSsymbols`	$\varPsi$	uppercase Greek letter pi; variant see also: \Psi	`&#x03A8;`	class ORD
`\varrho`	`AMSsymbols`	ϱ	lowercase Greek letter rho; variant see also: \rho	`&#x03F1;`	class ORD
`\varsigma`	`AMSsymbols`	ς	lowercase Greek letter sigma; variant see also: \sigma	`&#x03C2;`	class ORD
`\varSigma`	`AMSsymbols`	$\varSigma$	uppercase Greek letter sigma; variant see also: \Sigma	`&#x03C2;`	class ORD

`\varsubsetneq`
`\varsubsetneqq`
	`AMSsymbols`
	`AMSsymbols`
		$\varsubsetneq$
		$\varsubsetneqq$
			see also: \subsetneq, \subsetneqq	`&#x228A;`
				`&#x2ACB;`
					class REL
					class REL

`\varsupsetneq`
`\varsupsetneqq`
	`AMSsymbols`
	`AMSsymbols`
		$\varsupsetneq$
		$\varsupsetneqq$
			see also: \supsetneq, \supsetneqq	`&#x228B;`
				`&#x2ACC;`
					class REL
					class REL

`\vartheta`
`\varTheta`
	
	`AMSsymbols`
		ϑ
		$\varTheta$
			lowercase Greek letter theta; variant
			uppercase Greek letter theta; variant
			see also: \theta, \Theta
				`&#x03D1;`
				`&#x0398;`
					class ORD
					class ORD

`\vartriangle`
`\vartriangleleft`
`\vartriangleright`
	`AMSsymbols`
	`AMSsymbols`
	`AMSsymbols`
		$\vartriangle$
		$\vartriangleleft$
		$\vartriangleright$
			see also: \triangle, \triangleleft, \triangleright	`&#x25B3;`
				`&#x22B2;`
				`&#x22B3;`
					class REL
					class REL
					class REL

`\varUpsilon`	`AMSsymbols`	$\varUpsilon$	uppercase Greek letter upsilon; variant see also: \upsilon	`&#x03A5;`	class ORD
`\varXi`	`AMSsymbols`	$\varXi$	uppercase Greek letter xi; variant see also: \Xi	`&#x039E;`	class ORD
`\vcenter`			**`\vcenter #1`** centers the argument on the ‘math axis’, which is at half the height of an ‘x’, or about the position of a minus sign; one of the reasons for `\vcenter` is to get stretchy delimiters to match the contents better Examples: `\left(\Rule{1ex}{2em}{0pt}\right)` yields $\left(\rule{1ex}{2em}\right)$ `\left(\vcenter{\Rule{1ex}{2em}{0pt}}\right)` yields $\left(\vcenter{\rule{1ex}{2em}}\right)$ `\left(\frac{a+b}{\dfrac{c}{d}}\right)` yields $\left(\frac{a+b}{\dfrac{c}{d}}\right)$ `\left(\vcenter{\frac{a+b}{\dfrac{c}{d}}}\right)` yields $\left(\vcenter{\frac{a+b}{\dfrac{c}{d}}}\right)$		
`\vdash`		$\vdash$	see also: \nvdash	`&#x22A2;`	class REL
`\Vdash`	`AMSsymbols`	$\Vdash$		`&#x22A9;`	class REL

`\vDash` AMSsymbols	⊨	`&#x22A8;` class REL see also: \nVdash, \nvDash
`\vdots`	⋮	vertical dots `&#x22EE;` class ORD
`\vec`		non-stretchy vector symbol `\vec #1` Examples: `\vec v` yields $\vec{v}$ `\vec{AB}` yields $\vec{AB}$ see also: \overrightarrow
`\vee`	∨	see also: \lor `&#x2228;` class BIN
`\veebar` AMSsymbols	⊻	`&#x22BB;` class BIN

`\verb`		verbatim mode;
		useful for code snippets and for displaying special characters 'as is' (i.e., not interpreted by MathJax).
		Only works in display mode.
		Usually, verbatim content is typeset in a sans serif font.
		`\verb ◇ <non-interpreted material> ◇`
		where ◇ denotes a non-letter character that does *not* appear in the `<non-interpreted material>`.
		To use `\verb` :
		• First look through the material that is to be typeset 'as is' (verbatim).
		• Choose a non-letter character that does *not* appear in this material.
		• This chosen non-letter character will mark the beginning and end of the verbatim material, as illustrated in the examples below.
		Examples (in display mode):
		`\verb*$x^2\sqrt y$* \text{ yields } x^2\sqrt y`
		yields:
		`$x^2\sqrt y$` yields $x^2\sqrt{y}$
		`\verb!Text and $\frac ab$ in \verb mode!`
		yields:
		`Text and $\frac ab$ in \verb mode`

`\vert` `\Vert`	| ‖	class ORD `&#x2225;` class ORD both non-stretchy when used alone; stretchy when used with `\left` or `\right` see also: |, \\|, \lvert, \lVert, \rvert, \rVert
`\vphantom`		vertical phantom Sometimes you want to *pretend* that something is there, for spacing reasons, but you don't want it to appear—you want it to be invisible—you want it to be a phantom. The box created by `\vphantom` has the height and depth of its argument, but its width is zero (so it doesn't contribute to any horizontal spacing issues). In other words, `\vphantom` creates vertical space equal to that produced by its argument, but doesn't create any horizontal space. `\vphantom #1` Examples: `\binom{\frac ab}c \binom{\vphantom{\frac ab}?}c` yields $\binom{\frac ab}{c}\binom{\vphantom{\frac ab}?}{c}$ see also: \phantom, \hphantom, \smash
`\Vvdash` AMSsymbols	⊪	`&#x22AA;` class REL

W

`\wedge`	$\wedge$	see also: \land &#x2227; class BIN
`\widehat`	$\widehat{}$	stretchy hat accent &#x02C6; `\widehat #1` Examples: `\widehat a` yields $\widehat a$ `\widehat A` yields $\widehat A$ `\widehat AB` yields $\widehat A B$ `\widehat{AB}` yields $\widehat{AB}$ see also: \hat
`\widetilde`	$\widetilde{}$	stretchy tilde accent &#x02DC; `\widetilde #1` Examples: `\widetilde a` yields $\widetilde a$ `\widetilde A` yields $\widetilde A$ `\widetilde AB` yields $\widetilde A B$ `\widetilde{AB}` yields $\widetilde{AB}$
`\wp`	$\wp$	‘wriggly’ letter p &#x2118; class ORD
`\wr`	$\wr$	‘wriggle’ symbol; &#x2240; class BIN

X

`\Xi`	Ξ	uppercase Greek letter xi &#x039E; class ORD see also: \varXi
`\xi`	ξ	lowercase Greek letter xi &#x03BE; class ORD

`\xleftarrow` AMSmath
`\xrightarrow` AMSmath
		stretchy arrows with mathematical overset and optional mathematical underset class REL
		`\xleftarrow[optionalArgument] #1`
		`\xrightarrow[optionalArgument] #1`
		where the optional arguments (inside brackets, if desired) appear below the arrows (see examples).
		Examples:
		`\xrightarrow a` yields $\xrightarrow{a}$
		`\xrightarrow ab` yields $\xrightarrow{a} b$
		`\xrightarrow{ab}` yields $\xrightarrow{ab}$
		`\xleftarrow{\text{see equation (1)}}` yields $\xleftarrow{\text{see equation (1)}}$
		`\xrightarrow[f]{\text{see (1)}}` yields $\xrightarrow[f]{\text{see (1)}}$

Y

`\yen` AMSsymbols	¥	&#x00A5; class ORD

Z

`\zeta`	ζ	lowercase Greek letter zeta &#x03B6; class ORD

environments

LaTeX environments of the form `\begin{XXX} ... \end{XXX}` are provided, as listed in the table below.
The `processEnvironments` value in the `tex2jax` block of the MathJax configuration controls processing behavior:

- `processEnvironments: true` (the default) causes environments to be processed both inside *and outside* of math delimiters
- `processEnvironments: false` causes environments to be processed only when they appear inside math delimiters

See the tex2jax Preprocessor for details.

`align` AMSmath

`\begin{align} ... \end{align}`

For vertical alignment of two or more lines at one or more places:

- ampersand(s) ‘&’ are used to indicate desired alignments (see examples below)
- a double backslash ‘`\\`’ or carriage return ‘`\cr`’ separates lines
- individual lines may be tagged using the `\tag{}` command:
  - default input for `\tag{}` is text
  - you may get mathematical content inside `\tag{}` by using math delimiters; e.g., `\tag{$\alpha$}`

EXAMPLES:

Alignment at a single location:

- use a single ampersand where alignment should occur
- you may tag (or not tag) any desired subset of lines

```
\begin{align}
(a+b)^2 &= (a+b)(a+b)          \tag{3.1c}      \\
        &= a^2 + ab + ba + b^2 \tag{$\dagger$} \\
        &= a^2 + 2ab + b^2     \tag{$\ast$}
\end{align}
```

yields

$$\begin{align}
(a+b)^2 &= (a+b)(a+b) \tag{3.1c} \\
&= a^2 + ab + ba + b^2 \tag{$\dagger$} \\
&= a^2 + 2ab + b^2 \tag{$\ast$}
\end{align}$$

Alignment at more than one location is trickier.
It is best illustrated with an example: show/hide more info

see also: \eqalign, \eqalignno, \leqalignno, \begin{aligned}

`align*` AMSmath

[May 2011] same as align

`alignat` AMSmath

`\begin{alignat}{<num>} ... \end{alignat}`

For vertical alignment of two or more lines at one or more places;
produces a more horizontally-compressed display than align:

- the alignat environment is started with `\begin{alignat}{<num>}`, where `num` is a positive integer $(1,2,3,\ldots)$ that indicates the number of places where alignment is desired
- ampersand(s) ‘&’ are used to indicate desired alignments (see examples below)
- a double backslash ‘`\\`’ or carriage return ‘`\cr`’ separates lines
- individual lines may be tagged using the `\tag{}` command:
  - default input for `\tag{}` is text
  - you may get mathematical content inside `\tag{}` by using math delimiters; e.g., `\tag{$\alpha$}`

Let n denote the number of places where alignment is desired.
Then, there will be $2n-1$ ampersands used, as follows:

- STEP 1:
  The odd-numbered ampersands (1st, 3rd, 5th, etc.) are placed where alignment is desired.
  Position these ampersands first:

  ```
  a   &= bbbbbb &=   cc   &= d \\
  aaa &=  bbbb  &= cccccc &= ddd
  ```

- STEP 2:
  Now, focus attention on the content *between* the previously-positioned ampersands.
  What part of this content belongs on the left? On the right?
  In each group, use an ampersand to separate the content into two pieces (a piece may be empty).
  Think of this ampersand as a solid ‘wall’ that is pushing content to the left or right.

  Compare these three scenarios:

  Pushing all content to the left:

  ```
  \begin{alignat}{3}
  a   &= bbbbbb& &=   cc&   &= d \tag{3.1} \\
  aaa &=  bbbb&  &= cccccc& &= ddd \tag{3.2}
  \end{alignat}
  ```

  yields

  $$\begin{alignat}{3}
  a &= bbbbbb& &= cc& &= d \tag{3.1} \\
  aaa &= bbbb& &= cccccc& &= ddd \tag{3.2}
  \end{alignat}$$

  Pushing all content to the right:

  ```
  \begin{alignat}{3}
  a   &= &bbbbbb &=   &cc   &= d \\
  aaa &=  &bbbb  &= &cccccc &= ddd
  \end{alignat}
  ```

		yields
		
		$\begin{alignat}{3} a&= bbbbbb&&= && cc= d \\ aaa&= && bbbb=cccccc&&= ddd \end{alignat}$
		
		Splitting the content, with half left and half right:
		
		`\begin{alignat}{3}`
		`a   &= bbb&bbb &=   c&c   &= d \\`
		`aaa &=  bb&bb  &= ccc&ccc &= ddd`
		`\end{alignat}`
		
		yields
		
		$\begin{alignat}{3} a&= bbb&bbb&= c&c&= d \\ aaa&= bb&bb&= ccc&ccc&= ddd \end{alignat}$
		
		see also: \eqalignat, \eqalignatno, \leqalignatno, \begin{alignedat}

`alignat*` AMSmath		[May 2011] same as alignat

`aligned` AMSmath		same as \begin{align}, but allows only a single tag, which is vertically centered on the group
		
		Examples:
		
		`\begin{aligned}`
		`\tag{3.1}`
		`x_1 &= 1\cr`
		`x_2 &= 1 + 2\cr`
		`x_3 &= 1 + 2 + 3`
		`\end{aligned}`
		
		`\begin{aligned}`
		`x_1 &= 1 \tag{3.1}\cr`
		`x_2 &= 1 + 2\cr`
		`x_3 &= 1 + 2 + 3`
		`\end{aligned}`
		
		`\begin{aligned}`
		`x_1 &= 1\cr`
		`x_2 &= 1 + 2\cr`
		`\tag{3.1} x_3 &= 1 + 2 + 3`
		`\end{aligned}`
		
		all yield the same display:
		
		$\begin{aligned} x_1 &= 1 \\ x_2 &= 1+2 \\ x_3 &= 1+2+3 \end{aligned} \qquad (3.1)$

`alignedat` AMSmath		same as \begin{alignat}, but allows only a single tag, which is vertically centered on the group
		
		Examples:
		
		`\begin{alignedat}{1}`
		`\tag{3.1}`
		`x_1 &= 1\cr`
		`x_2 &= 1 + 2\cr`
		`x_3 &= 1 + 2 + 3`
		`\end{alignedat}`
		
		`\begin{alignedat}{1}`
		`x_1 &= 1 \tag{3.1}\cr`
		`x_2 &= 1 + 2\cr`
		`x_3 &= 1 + 2 + 3`
		`\end{alignedat}`
		
		`\begin{alignedat}{1}`
		`x_1 &= 1\cr`
		`x_2 &= 1 + 2\cr`
		`\tag{3.1} x_3 &= 1 + 2 + 3`
		`\end{alignedat}`
		
		all yield the same display:
		
		$\begin{alignedat}{1} x_1&= 1 \\ x_2&= 1+2 \\ x_3&= 1+2+3 \end{alignedat} \qquad (3.1)$

`array`
 	
`\begin{array}{<justification info>}`
`... \end{array}`
		Used to create an array (matrix),
		where columns can be individually left-justified, centered, or right-justified.
		• suppose that n columns are desired in the array;
		then, $n-1$ ampersands are used to separate the columns
		• the array environment is started with `\begin{array}{<justification info>}` ,
		where `<justification info>` is a series of n letters, one for each column:
		○ ‘l’ for left-justified
		○ ‘c’ for centered
		○ ‘r’ for right-justified
		○ pipe character(s) ‘|’ can be used in the justification information to specify optional separating vertical line(s) (see example below)
		• a double backslash ‘`\\`’ or carriage return ‘`\cr`’ separates rows
		
		Compare these scenarios:
		
		both columns left-justified:
		
		`\begin{array}{ll}`
		`aaa & b\cr`
		`c   & ddd`
		`\end{array}`
		
		yields

		$\begin{array}{ll} aaa & b \\ c & ddd \end{array}$
		
		both columns right-justified:
		`\begin{array}{rr}`
		`aaa & b\cr`
		`c   & ddd`
		`\end{array}`
		yields
		$\begin{array}{rr} aaa & b \\ c & ddd \end{array}$
		
		both columns centered, with separating line:
		`\begin{array}{c|c}`
		`aaa & b\cr`
		`c   & ddd`
		`\end{array}`
		yields
		$\begin{array}{cc} aaa & b \\ c & ddd \end{array}$
		
		first column left-justified; second column right-justified:
		`\begin{array}{lr}`
		`aaa & b\cr`
		`c & ddd`
		`\end{array}`
		yields
		$\begin{array}{lr} aaa & b \\ c & ddd \end{array}$
		
		Putting a pipe character ‘|’ at the beginning or end of the justification info encloses the entire structure, which is different from standard $\TeX$:
		`\begin{array}{|lr}`
		`aaa & b\cr`
		`c & ddd`
		`\end{array}`
		yields
		$\begin{array}{\|lr} aaa & b \\ c & ddd \end{array}$
		see also: \begin{matrix}, \begin{subarray}

`Bmatrix`
 	
`\begin{Bmatrix}`
`      ... \end{Bmatrix}`
		Used to create a matrix (an array) with braces $\{\ ,\ \}$ as enclosing delimiters;
		columns are centered.
		- suppose that n columns are desired in the array;
		then, $n-1$ ampersands are used to separate the columns
		- a double backslash ‘`\\`’ or carriage return ‘`\cr`’ separates rows
		
		Example:
		`\begin{Bmatrix}`
		`aaa & b\cr`
		`c   & ddd`
		`\end{Bmatrix}`
		yields $\begin{Bmatrix} aaa & b \\ c & ddd \end{Bmatrix}$
		
		see also: \begin{array}, \begin{matrix}

`bmatrix`
 	
`\begin{bmatrix}`
`      ... \end{bmatrix}`
		Used to create a matrix (an array) with brackets $[\ ,\ ]$ as enclosing delimiters;
		columns are centered.
		- suppose that n columns are desired in the array;
		then, $n-1$ ampersands are used to separate the columns
		- a double backslash ‘`\\`’ or carriage return ‘`\cr`’ separates rows
		
		Example:
		`\begin{bmatrix}`
		`aaa & b\cr`

		`c   & ddd`
		`\end{bmatrix}` yields $\begin{bmatrix} aaa & b \\ c & ddd \end{bmatrix}$
		see also: \begin{array}, \begin{matrix}

`cases`
`\begin{cases}`
`... \end{cases}`
		Used for piecewise-defined functions
		• an ampersand ‘&’ is used to separate the function cases and their definitions
		• a double backslash ‘`\\`’ or carriage return ‘`\cr`’ separates rows
		Example:
		`|x| =`
		`\begin{cases}`
		`x  & \text{ if } x\ge 0 \\`
		`-x & \text{ if } x \lt 0`
		`\end{cases}`
		yields $\lvert x\rvert = \begin{cases} x & \text{if } x \ge 0 \\ -x & \text{if } x < 0 \end{cases}$
		see also: \cases

`eqnarray`
`\begin{eqnarray}`
`... \end{eqnarray}`
		for ‘equation arrays’;
		aligns at one or more places;
		surround the character(s) to be aligned with ampersands, as shown below;
		content between alignment characters (or between alignment characters and end-of-line) is left-justified;
		a double backslash ‘`\\`’ or carriage return ‘`\cr`’ separates rows
		Examples:
		`\begin{eqnarray}`
		`y &=& (x-1)^2 \\`
		`  &=& (x-1)(x-1) \\`
		`  &=& x^2 - 2x + 1`
		`\end{eqnarray}`
		yields
		$\begin{aligned} y &= (x-1)^2 \\ &= (x-1)(x-1) \\ &= x^2 - 2x + 1 \end{aligned}$
		`\begin{eqnarray}`
		`(x-1)^2 &=& (x-1)(x-1)      &=&  x^2-2x + 1 \\`
		`(x-1)^3 &=& (x-1)(x-1)(x-1) &=&  (x-1)^2(x-1)`
		`\end{eqnarray}`
		yields
		$\begin{aligned} (x-1)^2 &= (x-1)(x-1) & &= x^2 - 2x + 1 \\ (x-1)^3 &= (x-1)(x-1)(x-1) & &= (x-1)^2(x-1) \end{aligned}$

`eqnarray*`		[May 2011] same as equarray

`equation`
`\begin{equation}`
`... \end{equation}`
		[May 2011] ignored, until MathJax implements automatic numbering

`equation*`		[May 2011] ignored

`gather` `AMSmath`		to display any number of centered formulas (without any alignment);
		a double backslash ‘`\\`’ or carriage return ‘`\cr`’ separates rows;
		individual lines may be tagged using the `\tag{}` command:
		• default input for `\tag{}` is text
		• you may get mathematical content inside `\tag{}` by using math delimiters; e.g., `\tag{$\alpha$}`
		Example:
		`\begin{gather}`
		`a = a \tag{$*$}\\`
		`\text{if } a=b \text{ then } b=a \tag{$\dagger$}\\`
		`\text{if } a=b \text{ and } b=c \text{ then } a=c\tag{3.1}`
		`\end{gather}`
		yields:
		$a = a \qquad (*)$
		$\text{if } a = b \text{ then } b = a \qquad (\dagger)$
		$\text{if } a = b \text{ and } b = c \text{ then } a = c \qquad (3.1)$
		see also: \displaylines, \begin{gathered}

`gather*` AMSmath		[May 2011] same as gather

`gathered` AMSmath		same as \begin{gather}, but allows only a single tag, which is vertically centered on the group
		Examples:
		`\begin{gathered}`
		`\tag{3.1}`
		`x = 1\cr`
		`y = 2\cr`
		`z = 3`
		`\end{gathered}`
		
		`\begin{gathered}`
		`x = 1 \tag{3.1}\cr`
		`y = 2\cr`
		`z = 3`
		`\end{gathered}`
		
		`\begin{gathered}`
		`x = 1\cr`
		`y = 2\cr`
		`\tag{3.1} z = 3`
		`\end{gathered}`
		all yield the same display:
		$$\begin{gathered} x = 1 \\ y = 2 \\ z = 3 \end{gathered} \tag{3.1}$$

`matrix`
`\begin{matrix}`
`... \end{matrix}`
		Used to create a matrix (an array) without any enclosing delimiters;
		columns are centered.
		• suppose that n columns are desired in the array;
		then, $n-1$ ampersands are used to separate the columns
		• a double backslash '`\\`' or carriage return '`\cr`' separates rows
		Example:
		`\begin{matrix}`
		`aaa & b\cr`
		`c   & ddd`
		`\end{matrix}`
		yields $\begin{matrix} aaa & b \\ c & ddd \end{matrix}$
		see also: \begin{array}

`multline` AMSmath
`\begin{multline}`
`... \end{multline}`
		a multi-line environment;
		typically used for formulas/equations that don't fit on a single line
		• the first (or only) line is displayed left-justified
		• the last line is displayed right-justified
		• any intermediate line(s) are centered
		The justification of intermediate lines can be adjusted with \shoveleft and \shoveright.
		Examples:
		`\begin{multline}`
		`\rm first\ line \\`
		`\rm second\ line \\`
		`\rm third\ line \\`
		`\rm fourth\ line`
		`\end{multline}`
		yields:
		$$\begin{multline} \rm first\ line \\ \rm second\ line \\ \rm third\ line \\ \rm fourth\ line \end{multline}$$
		`\begin{multline}`
		`\rm first\ line \\`
		`\shoveleft\rm second\ line \\`
		`\shoveright\rm third\ line \\`
		`\rm fourth\ line`
		`\end{multline}`
		yields:
		$$\begin{multline} \rm first\ line \\ \shoveleft\rm second\ line \\ \shoveright\rm third\ line \\ \rm fourth\ line \end{multline}$$
		see also: \begin{split}

`multline* [AMSmath]`		[May 2011] same as multline see also: \shoveleft, \shoveright

`pmatrix`
`\begin{pmatrix}`
`... \end{pmatrix}`
		Used to create a matrix (an array) with parentheses $($, $)$ as enclosing delimiters;
		columns are centered.
		• suppose that n columns are desired in the array;
		then, $n-1$ ampersands are used to separate the columns

		• a double backslash ‘`\\`’ or carriage return ‘`\cr`’ separates rows
		
		Example:
		`\begin{pmatrix}`
		`aaa & b\cr`
		`c   & ddd`
		`\end{pmatrix}` yields $\begin{pmatrix} aaa & b \\ c & ddd \end{pmatrix}$
		
		see also: \begin{array}, \begin{matrix}

`smallmatrix   AMSmath`
 	
`\begin{smallmatrix}`
`  ... \end{smallmatrix}`
		Used to create a small matrix (an array);
		particularly suited for use in text;
		columns are centered.
		• suppose that n columns are desired in the array;
		then, $n-1$ ampersands are used to separate the columns
		• a double backslash ‘`\\`’ or carriage return ‘`\cr`’ separates rows
		
		Examples:
		`the matrix`
		`$\begin{smallmatrix}`
		`aaa & b\cr`
		`c   & ddd`
		`\end{smallmatrix}$`
		`is...` yields the matrix $\begin{smallmatrix} aaa & b \\ c & ddd \end{smallmatrix}$ is...
		
		`\left[`
		`\begin{smallmatrix}`
		`aaa & b\cr`
		`c   & ddd`
		`\end{smallmatrix}`
		`\right]` yields (in display mode) $\left[\begin{smallmatrix} aaa & b \\ c & ddd \end{smallmatrix}\right]$
		
		`\left[`
		`\begin{smallmatrix}`
		`aaa & b\cr`
		`c   & ddd`
		`\end{smallmatrix}`
		`\right]` yields (in inline mode) $\left[\begin{smallmatrix} aaa & b \\ c & ddd \end{smallmatrix}\right]$
		
		see also: \begin{array}, \begin{matrix}

`split          AMSmath`		for single equations that are too long to fit on one line, and hence must be split into multiple lines;
		allows for (optional) alignment at one or more places, using ‘&’ to mark alignment points
		
		Examples:
		`\begin{split}`
		`\text{first line}\\`
		`&\text{first aligned place}           &\text{second aligned place}  \\`
		`&\text{and more first aligned}\qquad  &\text{and more second aligned} \\`
		`\text{no ampersands on this line} \\`
		`&                                     &\text{aligned at second place} \\`
		`\text{no amps here either}`
		`\end{split}`
		
		yields:
		$\begin{split}\text{first line}\\ &\text{first aligned place} &\text{second aligned place} \\ &\text{and more first aligned}\qquad &\text{and more second aligned} \\ \text{no ampersands on this line} \\ & &\text{aligned at second place} \\ \text{no amps here either}\end{split}$
		
		see also: \begin{multline}

`subarray`
 	
`\begin{subarray}`
`{<justification info>}`
`     ... \end{subarray}`
		a more compact version of \begin{array};
		can be used for multi-subscripts and multi-superscripts on large operators;
		columns can be individually left-justified, centered, or right-justified
		• suppose that n columns are desired in the subarray;
		then, $n-1$ ampersands are used to separate the columns
		• the subarray environment is started with `\begin{subarray}{<justification info>}`,
		where `<justification info>` is a series of n letters, one for each column:
		  ○ ‘l’ for left-justified
		  ○ ‘c’ for centered
		  ○ ‘r’ for right-justified
		• a double backslash ‘`\\`’ or carriage return ‘`\cr`’ separates rows
		
		Example:
		`\prod_{\begin{subarray}{rl}`
		`i\lt 5                 & j\gt 1 \\`

		`k\ge2,\,k\ne 5 \quad    & \ell\le 5,\,\ell\ne 2`
		`\end{subarray}}`
		`x_{ijk\ell}`
		yields
		$\prod_{\substack{i<5 \quad j>1\\ k\ge2,\,k\ne5 \quad \ell\le5,\,\ell\ne2}} x_{ijk\ell}$
		see also: \substack, \begin{array}

`Vmatrix`
`\begin{Vmatrix}`
`... \end{Vmatrix}`
		Used to create a matrix (an array) with $\Vert$, $\Vert$ as enclosing delimiters; columns are centered.
		• suppose that n columns are desired in the array; then, $n-1$ ampersands are used to separate the columns
		• a double backslash ‘`\\`’ or carriage return ‘`\cr`’ separates rows
		Example:
		`\begin{Vmatrix}`
		`aaa & b\cr`
		`c   & ddd`
		`\end{Vmatrix}`
		yields $\begin{Vmatrix} aaa & b\\ c & ddd \end{Vmatrix}$
		see also: \begin{array}, \begin{matrix}

`vmatrix`
`\begin{vmatrix}`
`... \end{vmatrix}`
		Used to create a matrix (an array) with $\vert$, $\vert$ as enclosing delimiters; columns are centered.
		• suppose that n columns are desired in the array; then, $n-1$ ampersands are used to separate the columns
		• a double backslash ‘`\\`’ or carriage return ‘`\cr`’ separates rows
		Example:
		`\begin{vmatrix}`
		`aaa & b\cr`
		`c   & ddd`
		`\end{vmatrix}`
		yields $\begin{vmatrix} aaa & b\\ c & ddd \end{vmatrix}$
		see also: \begin{array}, \begin{matrix}