

About

AsciiMath is an easy-to-write markup language for mathematics.
Try it out in the interactive renderer:

Input:

```
sum_(i=1)^n i^3=((n(n+1))/2)^2
```

Rendering:

$$\sum_{i=1}^n i^3 = \left(\frac{n(n+1)}{2}\right)^2$$

Getting Started

In order to get started you have two options:

- Use [MathJax](#) to render your formulas. MathJax is a full fledged open source JavaScript display engine for mathematics and **works in all browsers**.
This is the recommended approach!

Get started by loading the default AsciiMath configuration:

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/mathjax/2.7.4/latex-t.js?config=AM_CHTML"></script>
```

Visit the MathJax [Getting Started](#) page to find other CDN options , or to learn how to install MathJax locally on your server.

Text in your HTML enclosed in ``` (backticks) will now get rendered as a math formula. The math delimiters can also be customized. Check out the [MathJax](#) website for more information!

- Load the AsciiMath javascript file ([get it on GitHub](#)) in either the `head` or the `body` tag of your website like this:

```
<script src="ASCIIMathML.js"></script>
```

This file contains JavaScript to convert AsciiMath notation and (some) LaTeX to Presentation MathML. The conversion is done while the HTML page loads.

Attention: Currently this only works in Firefox and Safari.

While HTML5 now includes MathML as an official recommendation, the remaining browsers do not appear to be implementing it. For widest browser compatibility, the use of MathJax is recommended.

Syntax

Most AsciiMath symbols attempt to mimic in text what they look like rendered, like `oo` for ∞ . Many symbols can also be displayed using a TeX alternative, but a preceeding backslash is not required.

Operation symbols			Miscellaneous symbols			Relation symbols		
Type	TeX alt	See	Type	TeX alt	See	Type	TeX alt	See
+		+	2/3	$\frac{2}{3}$	$\frac{2}{3}$	=		=
-		−	2^3		2³	!=	ne	≠
*	cdot	⋅	sqrt x		$\sqrt{x}$	<	lt	<
**	ast	*	root(3)(x)		$\sqrt[3]{x}$	>	gt	>
***	star	★	int		$\int$	<=	le	≤
//		/	oint		$\oint$	>=	ge	≥
\	backslash setminus	\	del	partial	∂	mlt	ll	$m <$
xx	times	×	grad	nabla	∇	mgt	gg	$m >$
÷	div	÷	+-	pm	±	-<	prec	⋈
><	ltimes	⋈	O/	emptyset	∅	-<=	preceq	⋈
><	rtimes	⋈	oo	infty	∞	>	succ	⋈
><	bowtie	⋈	aleph		ℵ	>=	succeq	⋈
@	circ	∘	∴	therefore	∴	in		∈
o+	oplus	⊕	∵	because	∵	!in	notin	∉
ox	otimes	⊗	...	ldots	...	sub	subset	⊂
o.	odot	⊙	cdots		⋯	sup	supset	⊃
sum		$\sum$	vdots		⋮	sube	subseteq	⊆
prod		$\prod$	ddots		⋱	supe	supseteq	⊇
^^	wedge	∧				=	equiv	≡
^^^	bigwedge	∧				~=	cong	≅
v	vee	∨				~~	approx	≈
vv	bigvee	∨				prop	propto	∝
nn	cap	∩						
nnn	bigcap	∩						
uu	cup	∪						
uuu	bigcup	∪						

Logical symbols			Grouping brackets		
Type	TeX alt	See	Type	TeX alt	See
and		and	(		(
or		or	)		)
not	neg	¬	[		[
=>	implies	⇒	]		]
if		if	{		{
<=>	iff	⇔	}		}
AA	forall	∀	(:	langle	(:
EE	exists	∃	:)	rangle	:)
⊥	bot	⊥	<<		<<
TT	top	⊤	>>		>>
--	vdash	⊢	{: x)		{: x)
==	models	⊨	(x :)		(x :)
			abs(x)		abs(x)
			floor(x)		floor(x)
			ceil(x)		ceil(x)
			norm(vecx)		norm(vecx)

Arrows

Type	TeX alt	See
uarr	uparrow	uarr
darr	downarrow	darr
rarr	rightarrow	rarr
->	to	->
>->	rightarrowtail	>->
->>	twoheadrightarrow	->>
>->>	twoheadrightarrowtail	>->>
->	mapsto	->
larr	leftarrow	larr
harr	leftrightarrow	harr
rArr	Rightarrow	rArr
lArr	Leftarrow	lArr
hArr	Lefttrightarrow	hArr

Accents

Type	TeX alt	See
hat x		hat x
bar x	overline x	bar x
ul x	underline x	ul x
vec x		vec x
tilde x		tilde x
dot x		dot x
ddot x		ddot x
overset(x)(=)	overset(x)(=)	overset(x)(=)
underset(x)(=)		underset(x)(=)
ubrace(1+2)	underbrace(1+2)	ubrace(1+2)
obrace(1+2)	overbrace(1+2)	obrace(1+2)
overarc(AB)	overparen(AB)	overarc(AB)
color(red)(x)		color(red)(x)
cancel(x)		cancel(x)

Greek Letters

Type	See	Type	See
alpha	alpha		
beta	beta		
gamma	gamma	Gamma	Gamma
delta	delta	Delta	Delta
epsilon	epsilon		
varepsilon	varepsilon		
zeta	zeta		
eta	eta		
theta	theta	Theta	Theta
vartheta	vartheta		
iota	iota		
kappa	kappa		
lambda	lambda	Lambda	Lambda
mu	mu		
nu	nu		
xi	xi	Xi	Xi
pi	pi	Pi	Pi
rho	rho		
sigma	sigma	Sigma	Sigma
tau	tau		
upsilon	upsilon		
phi	phi	Phi	Phi
varphi	varphi		
chi	chi		
psi	psi	Psi	Psi
omega	omega	Omega	Omega

Font commands

Type	TeX alt	See
bb "AaBbCc"	mathbf "AaBbCc"	bb "AaBbCc"
bbb "AaBbCc"	mathbb "AaBbCc"	bbb "AaBbCc"
cc "AaBbCc"	mathcal "AaBbCc"	cc "AaBbCc"
tt "AaBbCc"	mathtt "AaBbCc"	tt "AaBbCc"
fr "AaBbCc"	mathfrak "AaBbCc"	fr "AaBbCc"
sf "AaBbCc"	mathsf "AaBbCc"	sf "AaBbCc"

Standard Functions

sin, cos, tan, sec, csc, cot, arcsin, arccos, arctan, sinh, cosh, tanh, sech, csch, coth, exp, log, ln, det, dim, mod, gcd, lcm, lub, glb, min, max, f, g.

Special Cases

Matrices: `[[a,b],[c,d]]` yields to `[[a,b],[c,d]]`

Column vectors: `((a),(b))` yields to `((a),(b))`

Augmented matrices: `[[a,b|,c],[d,e|,f]]` yields to `[[a,b|,c],[d,e|,f]]`

Matrices can be used for layout: `{(2x,+,17y,=,23),(x,-,y,=,5):}` yields `{(2x,+,17y,=,23),(x,-,y,=,5):}`

Complex subscripts: `lim_(N->oo) sum_(i=0)^N` yields to `lim_(N->oo) sum_(i=0)^N`

Subscripts must come before superscripts: `int_0^1 f(x)dx` yields to `int_0^1 f(x)dx`

Derivatives: `f'(x) = dy/dx` yields `f'(x) = dy/dx`

For variables other than x,y,z, or t you will need grouping symbols: `(dq)/(dp)` for `(dq)/(dp)`

Overbraces and underbraces: `ubrace(1+2+3+4)_( "4 terms" )` yields `ubrace(1+2+3+4)_( "4 terms" )`.

`obrace(1+2+3+4)^( "4 terms" )` yields `obrace(1+2+3+4)^( "4 terms" )`.

Attention: Always try to surround the `>` and `<` characters with spaces so that the html parser does not confuse it with an opening or closing tag!

The Grammar

Here is a definition of the grammar used to parse AsciiMath expressions. In the Backus-Naur form given below, the letter on the left of the `::=` represents a category of symbols that could be one of the possible sequences of symbols listed on the right. The vertical bar `|` separates the alternatives.

`v ::= [A-Za-z] | greek letters | numbers | other constant symbols`
`u ::= sqrt | text | bb | other unary symbols for font commands`
`b ::= frac | root | stackrel | other binary symbols`

`l ::= ( | [ | { | ( : | { : | other left brackets`
`r ::= ) | ] | } | : ) | : } | other right brackets`

`s ::= v | lEr | uS | bSS` Simple expression

`E ::= S_S | S^S | S` Intermediate expression

`E ::= lE | lI` Expression