

# Mathematical expressions and equations in HTML with MathJax and AsciiMath

## ■ Introduction

- Mechanics, configuring MathJax
- MathJax and the AsciiMath syntax
- Securing and industrializing the loading of the MathJax library
- The underscore character `_` in AsciiMath syntaxes : `text{ }`
- Problem with the backtick character as the delimiter of AsciiMath syntaxes, conflicts with MySQL and Shell syntaxes
- CSS styles and MathJax / AsciiMath
- Centering AsciiMath equations on the sign `=`
- Combining AsciiMath and Tex/Latex syntaxes: how, when and why ?
- MathJax Menu

 September 14<sup>th</sup>, 2016 ( #272-en , last updated : )



## Introduction

Writing mathematical expressions and equations in HTML pages, even if the use may not be common, and in some instances exceptional, has always been problematic. But sometimes we need to display formulas with a good ergonomic rendering.

Either the expression is in a raw format, not very ergonomic and difficult to read, especially when the formula contains fractions, for example :

$$f(x) = (x + 2) / (2x + 1) \qquad x \in \mathbb{R}, x \neq -1/2$$

or an image is created from tools such as Formula Math in LibreOffice, image to be recreated if the formula needs to be updated and further more the user is then not able to copy some text :

$$f(x) = \frac{x+2}{2x+1} \quad x \in \mathbb{R}, x \neq -\frac{1}{2}$$

Very early, HTML 5 was scheduled to integrate MathML standards for writing mathematical expressions in HTML, but browsers publishers have decided otherwise. MathML support in the major browsers (Chrome, FireFox, MS Internet Explorer, Safari) was disparate and there were too many disagreements.

Google, the publisher of Chrome, is probably right by deciding not to bring MathML in its browser. Google argued that MathML is restricted to an audience of scientists (mathematicians, physicists, statisticians) and powerful Javascript libraries already exist to cover the needs.

Indeed, [MathJax](http://www.mathjax.org/) (<http://www.mathjax.org/>) is a Javascript library, compatible with all browsers, simple and very advanced for rendering mathematical expressions in HTML. The rendering is impressive, the coding simple and the content of the expressions thus becomes easily updatable. The above example rendered with MathJax (option AsciiMath) :

```
<div class="cmath"> `f(x) = (x+2)/(2x+1)`           `x in RR,\ x  
!=-1/2`</div>
```

MathJax manages several input syntaxes (Tex / Latex, AsciiMath, MathML MML), AsciiMath syntax being by far the easiest for simple needs.

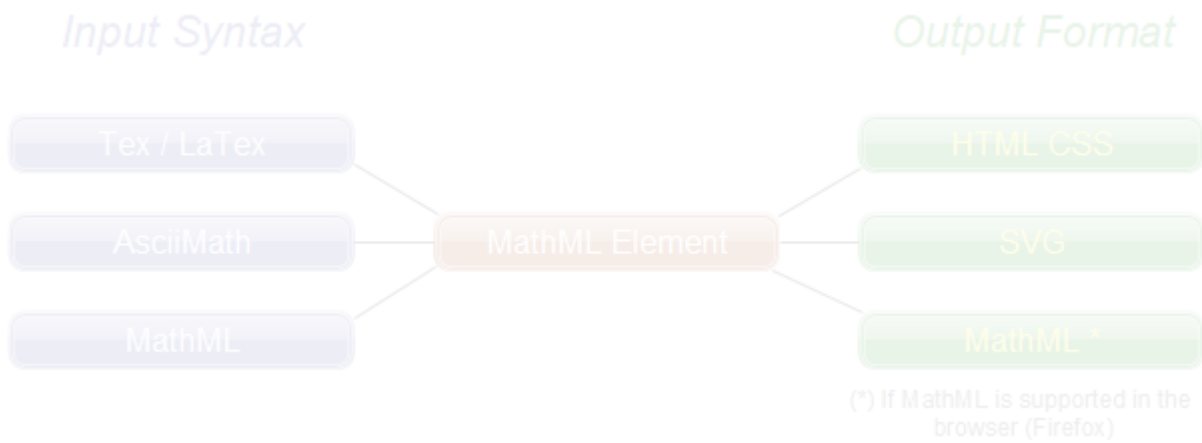
This article is not a tutorial about how to write maths/physics equations (using MathJax).

This is a quick reference on how to include MathJax automatically in articles if mathematical expressions exist (AsciiMath and/or Tex-Latex syntaxes). Compared to the Tex-Latex syntax, AsciiMath syntax is indeed the easiest to use for basic needs, but there is a lack of documentation about specific needs when using AsciiMath. This paper focuses on these topics using AsciiMath syntax :

- Applying style to a portion of an equation
- Highlighting an equation
- Numbering equations
- Centering equations on the sign =

## ■ Mechanics, configuring MathJax

To schematize the MathJax library mechanics :



One or more input syntaxes are possible (AsciiMath, Tex / Latex, MathML) and the mathematical expression is displayed graphically in an output format (HTML-CSS, SVG, MathML if the browser supports the MathML language like FireFox).

The MathJax JavaScript library is typically called using the syntax below :

`<installation path>/MathJax.js?config=<input syntaxe(s)>_<output format(s)>`

The official documentation describes the possible configurations combinations : [MathJax - Combined configurations \(https://docs.mathjax.org/en/v2.7-latest/config-files.html#common-configurations\)](https://docs.mathjax.org/en/v2.7-latest/config-files.html#common-configurations) .

Some common configurations :

Config	Description
.../MathJax.js?config=AM_CHTML	<ul style="list-style-type: none"> <li>▪ Input syntax : AsciiMath (AM)</li> <li>▪ Output format : "Common HTML" (CHTML)</li> </ul>
.../MathJax.js?config=TeX-MML-AM_CHTML	<ul style="list-style-type: none"> <li>▪ Input syntax : Tex/Latex (TeX), or MathML (MML), or AsciiMath (AM)</li> <li>▪ Output format : "Common HTML" (CHTML)</li> </ul>
.../MathJax.js?config=TeX-AMS_CHTML	<ul style="list-style-type: none"> <li>▪ Input syntax : Tex/Latex (TeX) including AMS symbols</li> <li>▪ Output format : "Common HTML" (CHTML)</li> </ul>

The MathJax library can be installed locally. It is downloadable in zip format from the MathJax GitHub site : [MathJax GitHub Downloads](https://github.com/mathjax/MathJax/releases) (<https://github.com/mathjax/MathJax/releases>) . The MathJax library size is around 70 Mb uncompressed (version 2.7.8 - August 2019).

When not choosing a local installation, MathJax 2 is available online with CDN providers (Content Delivery Network).

For a given configuration, AM\_CHTML in the below examples (AsciiMath syntax in input, Common HTML in output format), in the page's header using an online CDN context :

```
<head>...
  <script async="true"
src="https://cdn.jsdelivr.net/npm/mathjax@2/MathJax.js?config=AM_CHTML">
</script>
...</head>
```

The option mathjax@2 in the URL of the script MathJax.js guarantees the use of the most recent version 2. If for some reason, a specific version must be used (bug...) in a page or a few, for example the version 2.7.6 :

```
<head>...
  <script async="true"
src="https://cdn.jsdelivr.net/npm/mathjax@2.7.6/MathJax.js?"
```

```
config=AM_CHTML "> </script>
...</head>
```

The examples above are classic and basic calls, a next paragraph describes how to secure and industrialize the call to the MathJax library.

## ■ MathJax and the AsciiMath syntax

AsciiMath (<http://asciimath.org/>) is the easiest input syntax (compared to Tex / Latex, MathML) for a basic usage to render mathematical expressions. When advanced features are needed, no choice, switch to the Tex / Latex language.

Requesting AsciiMath in the configuration (AM) and without custom parameters defined before, you're done :

```
<head>...
  <script async="true"
src="https://cdn.jsdelivr.net/npm/mathjax@2/MathJax.js?config=AM_CHTML">
</script>
...</head>
```

Just delimit the mathematical expressions written in AsciiMath syntax using backtick characters (`).

```
<div class="cmath">
  `f(x)=x^2 + 2x + 1 = 0`           Derivative of the function :
  `f^(x)=2x + 2`

  `frac(df(x))(dx) = lim_(h->0)(f(x+h)-f(x))/h`

  `f(t) = a_(0)/2 + sum_(n=1)^oo [a_(n) cos(nt) + b_(n) sin(nt)]`

  `x^' = frac(-b -sqrt(Delta))(2a)`
</div>
```

Derivative of the function :

In the above examples, AsciiMath expressions are written in `div` blocks. The CSS class `cmath` is applied on the container `div` : in this class the property `white-space` is set to `pre`. Spaces and new lines in the container are better managed using this property.

```
div.cmath {  
  display: block;  
  margin: 20px 0px 20px 30px;  
  padding-left: 50px; padding-top: 8px; padding-bottom: 8px;  
  white-space : pre;  
}
```

Expressions can be "in line" within a paragraph using `span` tags.

```
<p>An expression : <span> `sum_(i=1)^n i^3=((n(n+1))/2)^2` </span> within a  
phrase.</p>
```

An expression : within a phrase.

## ■ Securing and industrializing the loading of the MathJax library

With few lines of JavaScript code, the industrialization and optimization of the MathJax library loading can be handled by a function.

```
<head>  
  ...  
  <script type="text/javascript" src="./js/lib.js"> </script>  
  ...  
</head>
```

 ./js/lib.js ↴

```
run_maths = function() {  
  
  if (document.querySelector('[class*="cmath"]') !== null) {  
  
    if (typeof (mjax_path)=='undefined') {  
mjax_path='https://cdn.jsdelivr.net/npm/mathjax@2'; }  
    if (typeof (mjax_config)=='undefined') { mjax_config='AM_CHTML'; }  
  
    smjax = document.createElement ('script');  
    smjax.setAttribute('src',`${mjax_path}/MathJax.js?`  
config=${mjax_config}`);  
    smjax.setAttribute('async',true);  
    document.getElementsByTagName('head')[0].appendChild(smjax);  
  }  
}
```

```

    }
};

if (document.readyState === 'loading') {
window.addEventListener('DOMContentLoaded', run_maths); }
else { run_maths(); }

```

The function `run_maths` is called when the event `DOMContentLoaded` is going to happen or if the `document.readyState` property is already `interactive` or `complete`. All the document's nodes must be loaded before inserting dynamically and firing MathJax.

In the function `run_maths`

- The MathJax library is dynamically inserted only if there is at least one element in the document containing the class `cmath`. Using this method, the MathJax library, which has a significant cost in the page's load, will only be called if the page needs it.
- If a configuration is not already explicitly specified using the variable `mjax_config`, a default configuration is applied.

```
if (typeof (mjax_config)=='undefined') { mjax_config='AM_CHTML'; }
```

- Ditto for the path to the MathJax library using the variable `mjax_path`.

```
if (typeof (mjax_path)=='undefined') {
mjax_path='https://cdn.jsdelivr.net/npm/mathjax@2'; }
```

The path to MathJax and / or the configuration can then be modified for a given page by defining the variables `mjax_path` and / or `mjax_config` to override the default values. It doesn't matter where the definition is performed in the document, MathJax is not loaded before the `DOMContentLoaded` event.

```

<script> var mjax_config='TeX-MML-AM_CHTML'; </script>
<script> var mjax_path='https://cdn.jsdelivr.net/npm/mathjax@2.7.6';
</script>

```

## ■ The underscore character `_` in AsciiMath syntaxes : `text{ }`

A quick note about the underscore character because the examples of equations in documentations and tutorials do not address this point very much. Often, variables in expressions contain underscore characters `_`, the character underscore is interpreted in AsciiMath syntax to define subscript letters or symbols. To prevent this character from being interpreted, use the option `text{variable_1}`.

```
<div class="cmath">`R=D/L=frac (text{del_lf_rows})(text{lf_rows}) >= 0,1`</div>
```

A lighter syntax is possible by escaping the variables containing the underscore character with double quotes.

```
<div class="cmath">`R=D/L=frac ("del_lf_rows")("lf_rows") >= 0,1`</div>
```

Using the syntax `text { }` or escaping with double quotes, unfortunately we lose the mathematical typography (italics, etc.), to recover it, we have to force the font on the elements inside the `mjx-mtext` class, class applied by the MathJax engine in this context.

```
.mjx-mtext > * { font-family: MJXc-TeX-math-I,MJXc-TeX-math-Ix,MJXc-TeX-math-Iw !important; }
```



This issue also occurs in MathJax v3. Using MathJax 3, the following workaround works :

```
mjx-mtext > * { font-family: MJXZERO, MJXTEX-I !important; }
```

The font family is then not forced on the `mjx-mtext` class but on the `mjx-mtext` tag. MathJax v2 and MathJax v3 fonts are different.

This issue has been submitted to MathJax community : [GitHub MathJax - Italic fonts not applied using AsciiMath and text { } syntax, MathJax 2 and 3](https://github.com/mathjax/MathJax/issues/2641).

(<https://github.com/mathjax/MathJax/issues/2641>) .

## ■ Problem with the backtick character as the delimiter of AsciiMath syntaxes, conflicts with MySQL and Shell syntaxes

Using the backtick ``` character as the delimiter raises a question : what happens if a page contains this character in the text or in a MySQL or Shell syntax, languages that use the backtick ?

```
insert into `matable` values ...
```

No problem, backticks are not interpreted by MathJax when they are in the below HTML tags :

- `script`, `noscript`
- `style`



- `textarea`, `pre`, `code`
- `annotation`, `annotation-xml`

For MySQL syntaxes and Shell scripts, except in specific cases, `pre` and `code` tags are generally used in order to benefit from syntax highlighting with HighlightJs or Prism Javascript libraries. In the other cases, use the `code` or `annotation` tags to encapsulate the backtick so that MathJax does not interpret the content.

```
<code>`</code> <annotation>`</annotation>
```

If the backtick as a delimiter of mathematical expressions generates too many conflicts with the content in a page, the AsciiMath delimiter can be modified with the code below before loading the MathJax library :

```
<script type="text/javascript">
  window.MathJax = {
    asciimath2jax: {
      delimiters: [['~', '~']]
    }
  };
</script>
<script type="text/javascript" src="./js/lib.js"></script>
```

The backtick character ` is replaced by the character tilde ~ as the delimiter in the above example.

## ■ CSS styles and MathJax / AsciiMath

Is it possible to apply a CSS style to all or part of an equation written in AsciiMath with MathJax ? Yes.



The features discussed in this paragraph are available starting MathJax 2 version 2.7.5 which supports AsciiMath 2. AsciiMath 2 adds some Latex notations, including the notation `class{}`

The output format must be "Common HTML" (CHTML) in the configuration.

About MathJax 3, AsciiMath 2 is supported as of version 3.1.3 (april 2020).

For the ease of read, it is sometimes needed to change the style (notes...) on all or part of an equation. Using `span` tags associated to a CSS style class could be one solution, but there is the syntax `class{<classname>}{<equation code>}` ensuring more consistency in writing the equation and thus avoiding HTML tags.

In the below example, we want the comment to appear with a spacing of 100px on the left, a specific color and a smaller font size:

The class CSS `cmjx-note` is defined for this purpose, in a stylesheet script of the page or in the HTML code with `style` tags :

```
.cmjx-note {  
  transform: translate(100px);  
  font-size: 0.8em;  
  color: #DD4A68;  
}
```

The CSS class `cmjx-note` is applied to the desired location in the equation with the syntax : ``... class{cmjx-note} { ... Delta=b^2-4ac > 0 } ...``

```
<div class="cmath">  
  
  `ax^2 + bx + c = 0`  
  
  `x = frac(-b +- sqrt(Delta))(2a) class{cmjx-note}{ text{rem : } \  
Delta=b^2-4ac > 0 }`  
  
</div>
```

Several CSS classes can be defined using the syntaxe `class` :

```
`x = frac(-b +- sqrt(Delta))(2a) class{cmjx-note cmjx-note1 cmjx-note2}{  
text{rem : } \ Delta=b^2-4ac > 0 }`
```

Knowing this trick about AsciiMath syntaxes, many practical applications :

- Equations to be highlighted with borders.
- Numbering equations.

## ■ Highlighting equations

An immediate and very simple practical application: highlighting an AsciiMath equation by applying a border. Useful to normalize the borders with the site's ergonomics (colors, etc.).

```
.cmjx-highlight { border: 2px solid #DD4A68; padding: 8px; margin-right: 4px; }
```

```
<div class="cmath">
  `class{cmjx-highlight} { e^x = lim_(n->oo) (1 + x/n)^n }`
</div>
```

Much less used, but this class can be applied on "inline" expressions :

The volume of a sphere is , therefore ...

```
<p> The volume of a sphere is <span class="cmath"> `class{cmjx-highlight}
{V = 4/3\pi R^3} ` </span>, therefore ...</p>
```

## ■ Numbering equations

The solution is not perfect but by using a CSS class in which the position is absolute, 600px on the left for example, the numbering of AsciiMath equations becomes quite simple with the class {} syntax.

```
.cmjx-nb { position: absolute; left: 600px; }
```

```
<div class="cmath">
  `cos^2 \theta + sin^2 \theta = 1 class{cmjx-nb}{{(9)}}`

  `cos^2 \theta - sin^2 \theta = cos 2\theta class{cmjx-nb}{{(10)}}`
</div>
```

The absolute position may have to be adjusted through another class depending on the context (equations complexity, etc.).

If rendering on mobile devices is a concern, to summarize very briefly: when equations start to be numbered, it concerns more pages around topics specific to mathematics, physics, etc., pages that should generally be viewed on a desktop device.

## ■ Centering AsciiMath equations on the sign =

To center AsciiMath equations on the sign , matrices are used for the layout. With the Tex / Latex syntax, it is quite different.

The sign is then an element of the matrix. The opening and closing element of the matrix is hidden ( $\{ : \}$ ) :

```
<div class="cmath">
`{ :
  ( f(x)      ,=, (x+2)(x-3)      ) ,
  (           ,=, x^2 -3x +2x -6   ) ,
  (           ,=, x^2 -x  - 6      ) ,
  ( f^(x)     ,=, 2x -1           )
:} `
</div>
```

When the equations are linear on both sides in the matrix, the rendering is perfect. As soon as there are fractions, the rendering is less optimal.

```
<div class="cmath">
`{ :
  ( f(x)      ,=, frac(x+1)(x-2)      ) ,
  ( f^(x)     ,=, frac((x-2) - (x+1))((x-2)^2) ) ,
  (           ,=, - frac(3)(x^2 - 4x + 4)      )
:} `
</div>
```

To solve this issue, define a CSS class in order to adjust in particular the font size, the adjustment is performed here with the unit rem :

```
.cmjx-lg { font-size: 1.65rem; padding-bottom:8px;}
```

The unit and the value to use depend on the context of the page and they are determined empirically.

The class is then applied on the elements to adjust/enlarge, so in this use case the fractions :

```
<div class="cmath">
`{ :
  ( f(x)      ,=, class{cmjx-lg} { frac(x+1)(x-2) }      ) ,
  ( f^(x)     ,=, class{cmjx-lg} { frac((x-2) - (x+1))((x-2)^2) } )
,
  (           ,=, class{cmjx-lg} { - frac(3)(x^2 - 4x + 4) }      )
:} `
</div>
```

```
:}`  
</div>
```

## ■ Combining AsciiMath and Tex/Latex syntaxes: how, when and why ?

When mathematical expressions are basic, the simplicity of AsciiMath syntax is perfect. Whenever there is complexity, the AsciiMath syntax shows its weaknesses.

In a page, the 2 syntaxes can be combined. The Tex / Latex syntax is then used in blocks of equations involving cases that the AsciiMath syntax cannot cover.

To combine the 2 syntaxes, the configuration TeX-MML-AM\_CHTML is defined when calling MathJax. Some configuration specific to Tex / Latex is done before calling MathJax using the object window.MathJax

```
<head> ...  
  <script>  
    window.MathJax = {  
      tex2jax : { inlineMath : [ ['##', '##'], ["\\(", "\\)"] ],  
                    displayMath: [ ['$$', '$$'], ["\\[", "\\]"] ],  
                    processEscapes : true  
      },  
      displayAlign: "left"  
    };  
  </script>  
  <script async="true"  
src="https://cdn.jsdelivr.net/npm/mathjax@2/MathJax.js?config=TeX-MML-  
AM_CHTML"> <script>  
  
... </head>
```

The run\_maths function discussed previously in the paragraph "Securing and industrializing the loading of the MathJax library" is customized to predefine the properties for Tex / Latex syntaxes in the window.MathJax object.

 ./js/lib.js ↴

```
run_maths = function() {  
  
  if (document.querySelector('[class*="cmath"]') !== null) {
```

```

if (typeof (mjax_path)=='undefined') {
mjax_path='https://cdn.jsdelivr.net/npm/mathjax@2'; }
if (typeof (mjax_config)=='undefined') { mjax_config='AM_CHTML'; }

if (typeof(window.MathJax) == 'undefined') { window.MathJax = { }; }

if (mjax_config.toLowerCase().indexOf('tex') >= 0) {
    m = window.MathJax;
    if (typeof(m.displayAlign) == 'undefined') { m.displayAlign =
'left'; }
    if (typeof(m.tex2jax) == 'undefined') { m.tex2jax = { }; }
    if (typeof(m.tex2jax.inlineMath) == 'undefined') {
m.tex2jax.inlineMath      = [ ['##', '##'], ["\\(", "\\)"] ]; }
    if (typeof(m.tex2jax.displayMath) == 'undefined') {
m.tex2jax.displayMath     = [ ['$$', '$$'], ["\\[", "\\]"] ]; }
    if (typeof(m.tex2jax.processEscapes) == 'undefined') {
m.tex2jax.processEscapes = true; }
    }

    smjax = document.createElement ('script');
    smjax.setAttribute('src', `${mjax_path}/MathJax.js?
config=${mjax_config}`);
    smjax.setAttribute('async', true);
    document.getElementsByTagName('head')[0].appendChild(smjax);
}
};

if (document.readyState === 'loading') {
window.addEventListener('DOMContentLoaded', run_maths); }
else { run_maths(); }

```

When a page needs the combination of the 2 syntaxes (AsciiMath and Tex/Latex), define the variable `mjax_config` and assign the value `TeX-MML-AM_CHTML` :

```

...
<script> var mjax_config='TeX-MML-AM_CHTML'; </script>
...
<script type="text/javascript" src="./js/lib.js"> </script>
...

```

One may ask why a such complicated function code : by checking if each object or property already exists or not, the function ensures no custom property needed in a page is overwritten, much less the object `window.MathJax`. Example :

```

...
<script>

```

```

window.MathJax = {
  tex2jax : { inlineMath: [ ['~','~'] ]; }
};
</script>
<script> var mjax_config='TeX-MML-AM_CHTML'; </script>
<script type="text/javascript" src="./js/lib.js"> </script>
...

```

In the default configuration applied by the function `run_maths`, for a page combining the 2 syntaxes :

Syntax	Mode	Code writing
AsciiMath	Block or "in line"	<code>`... AsciiMath code...`</code>
Tex/Latex	Block	<code>\$\$... Latex code...\$\$</code> <code>\[... Latex code ...\]</code>
Tex/Latex	In line	<code>##... Latex code ...##</code> <code>\(... Latex code ...\)</code>

By default, in block mode, expressions written in Latex are centered in the parent container (`div ...`). The property `displayAlign` in the `window.MathJax` object overrides this default behaviour. In this article, left alignment is forced (`displayAlign: 'left'`).

In the following sub-paragraphs, some cases in which the Latex syntax solves some issues encountered with the syntax AsciiMath.

## ■ Matrices with fractions (`dfrac`)

In a matrix containing fractions, using AsciiMath the fractions are badly sized. The CSS class `cmjx-lg` (`font-size: 1.65rem`) can be applied in order to adjust the elements to resize, but it does not solve the issue about the opening and closing character of the matrix, offsets are obvious :

```
<div class="cmath">
  `[bbrho] = ( ( 1/2 , 0 ),
               ( 0 , 1/2 )
             )`
</div>
```

```
<div class="cmath">
  `[bbrho] = ( ( class{cmjx-lg}{1/2} , 0
               ( 0 , class{cmjx-lg}{1/2} )
             )`
</div>
```

The Latex syntax is less easy but the rendering is perfect. Using AsciiMath syntax, this would have been technically impossible. The opening and closing character of the matrix embraces its elements :

```
<div class="cmath">$$
  [\pmb{\rho}] =
    \begin{pmatrix}
      \dfrac{1}{2} & 0 \\
      0 & \dfrac{1}{2}
    \end{pmatrix}
  $$</div>
```

Why `\dfrac` and not `\frac` in the Latex syntax ? Using `\dfrac` (`displaystyle`), the overall style is applied to the fraction, the fraction is not resized in relation to the matrix row.

## ■ Continuous fractions (cfrac)

Writing continued fractions with AsciiMath syntax is badly rendered, no easy technical solution (CSS classes...) :

```
<div class="cmath">
  `x = frac(1)(sqrt2 + frac(1)(sqrt2 + frac(1)(sqrt2 +
  ...)))`
</div>
```



Using the Latex syntax and `cfrac`, the rendering is correct and the code is even more readable :

```
<div class="cmath">$$
  x = \cfrac{1}{\sqrt{2}}+
      \cfrac{1}{\sqrt{2}}+
      \cfrac{1}{\sqrt{2}}+\dotsb
    }
  }
  $$</div>
```

## ■ Centering equations on the sign =

Using AsciiMath, when equations are linear on either side of the sign , no issue to center on the sign using matrices for layouts. Rendering is more complicated when there is a fraction, fractions are displayed smaller : this issue is solved by applying, on the elements to be adjusted, the CSS class `cmjx-lg` (font-size: 1.65rem).

```
<div class="cmath">
`{:
  ( f(x)      ,=, class{cmjx-lg} { frac(x+1)(x-2)          } ) ,
  ( f'(x)     ,=, class{cmjx-lg} { frac((x-2) - (x+1))((x-2)^2) } ) ,
  (          ,=, class{cmjx-lg} { - frac(3)(x^2 - 4x + 4)    } )
:}`
</div>
```

Using the Latex syntax, the adjustment of the fractions is no longer a concern. The Latex syntax is not more or less complex than the AsciiMath syntax to achieve centering.

```
<div class="cmath">$$
  \begin{align}
    f(x)   &= \frac{x+1}{x-2} \\
    f'(x)  &= \frac{(x-2) - (x+1)}{(x-2)^2} \\
           &= - \frac{3}{x^2 - 4x + 4}
  \end{align}
  $$</div>
```

## ■ Numbering equations, references

Using `\tag` in the Tex/Latex syntax, numbering equations is easy, which is not the case using AsciiMath syntax. We kill two birds with one stone : the equation number is correctly centered on the right side of its container and the equations can be centered on the sign .

```
<div class="cmath">$$
  \begin{align}
    \cos 2\theta &= \cos^2 \theta - \sin^2 \theta \\
    \tag{9} \quad \label{cos2x} &\\
    \cos^2 \theta + \sin^2 \theta &= 1 \\
    \tag{10} \quad \label{10} &\\
  \end{align}
$$</div>
```

Using `\label` in the above example, bookmarks are easily defined. It is not necessary that `tag` and `label` are the same. Very convenient to refer to an expression described elsewhere on the page using `\ref` of the Latex syntax :

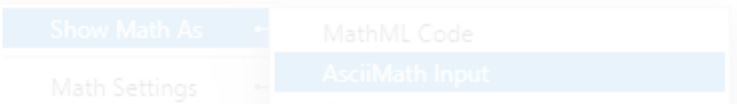
```
<div class="cmath">
We deduce from the formulas \(\ref{cos2x}\) and \(\ref{10}\) :
$$
  \begin{align}
    \cos 2\theta &= \cos^2 \theta + \sin^2 \theta - \sin^2 \theta - \\
    \sin^2 \theta &\\
    &= 1 - 2\sin^2 \theta \\
  \end{align}
$$</div>
```

We deduce from the formulas and :

## ■ MathJax Menu

A right click on a mathematical expression displays a useful MathJax menu :

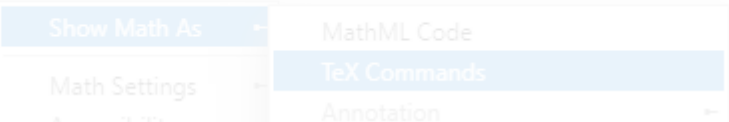
From the menu "Show Math As", the AsciiMath, Tex/Latex source code can be retrieved easily to get a copy. The input syntax (AsciiMath, Tex/Latex) is automatically detected.

$$\frac{df(x)}{dx} = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h}$$


MathJax Equation Source - Google Chrome

about:blank

$$\frac{df(x)}{dx} = \lim_{h \rightarrow 0} (f(x+h) - f(x)) / h$$

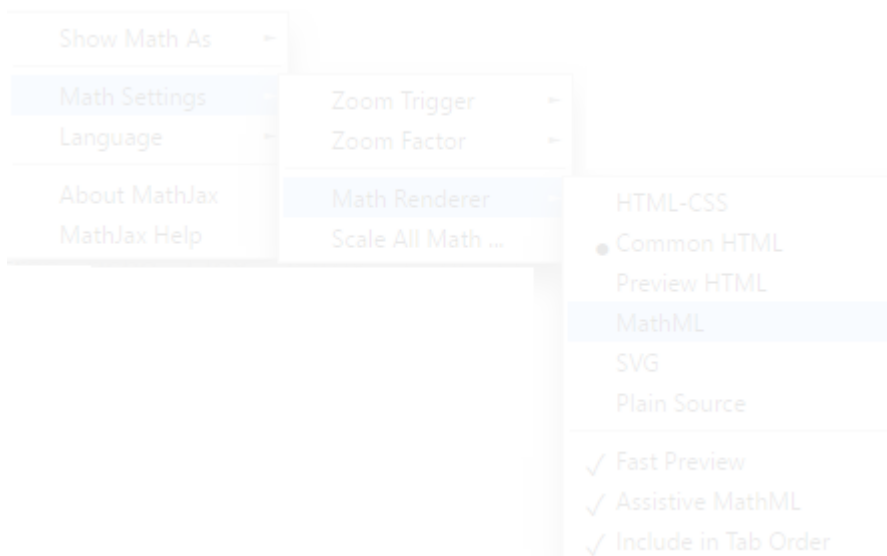
$$x' = \frac{-b - \sqrt{\Delta}}{2a}$$


MathJax Equation Source - Google Chrome

about:blank

$$x' = \frac{-b - \sqrt{\Delta}}{2a}$$

The menu "Math Settings ► Math Renderer" allows rendering in other output formats than the one defined in the configuration (Common HTML, HTML-CSS, SVG, MathML...) :



Useful for debugging purposes, using this functionality in the MathJax's menu we figure out that classes in AsciiMath or Tex/Latex syntaxes only work if the output format is set to "Common HTML" (CHTML).

As mentioned in the introduction, MathML has not become a standard with HTML 5 and is not necessarily available in the used browser, so rendering in MathML output format will not always work. Using Google Chrome : KO status.

www.sqlpac.uat indique :

Your browser doesn't seem to support MathML natively, so switching to MathML output may cause the mathematics on the page to become unreadable.

Switch the renderer anyway?

(Press OK to switch, CANCEL to continue with the current renderer)

☐ Empêcher cette page de générer des boîtes de dialogue supplémentaires

OK

Annuler

## Other resources

- Mathjax Library (<https://www.mathjax.org/>)
- AsciiMath (<http://asciimath.org/>)
- Syntax for entering math using ASCIIMathML (<https://www.intmath.com/help/send-math-email-syntax.php>)
- Mathjax - Combined configurations (<https://docs.mathjax.org/en/v2.7-latest/config-files.html#common-configurations>)
- MathJax basic tutorial and quick reference - Tex (<https://math.meta.stackexchange.com/questions/5020/mathjax-basic-tutorial-and-quick-reference>)
- GitHub MathJax - Italic fonts not applied using AsciiMath and text { } syntax, MathJax 2 and 3 (<https://github.com/mathjax/MathJax/issues/2641>)
- GitHub MathJax - MathJax 3, class{ } and asciimath. Scheduled release date, question (<https://github.com/mathjax/MathJax/issues/2640>)