Markdown Syntax Guide

Overview

Markdown is intended as a way of formatting text in an easy-to-read and easy-to-write way.

Readability is emphasised above all else. A Markdown-formatted document should be publishable as-is, as plain text, without looking like it's been marked up with tags or formatting instructions.

Block elements

Paragraphs and line breaks

A paragraph is simply one or more consecutive lines of text, separated by one or more blank lines. (A blank line is any line that looks like a blank line – a line containing nothing but spaces or tabs is considered blank.) Normal paragraphs should not be indented with spaces or tabs.

When you want to insert a `<br />` break tag using Markdown, you end a line with two or more spaces, then type return.

Headers

Markdown supports two styles of headers.

Setext-style headers are "underlined" using equal signs (for first-level headers) and dashes (for second-level headers). For example:

```
This is an H1
=============

This is an H2
-------------
```

Any number of underlining `=`'s or `-`'s will work.

Atx-style headers use 1-6 hash characters at the start of the line, corresponding to header levels 1-6. For example:

```
# This is an H1

## This is an H2

###### This is an H6
```

Optionally, you may "close" atx-style headers. This is purely cosmetic – you can use this if you think it looks better. The closing hashes don't even need to match the number of hashes used to open the header. (The number of opening hashes determines the header level.) :

```
# This is an H1 #

## This is an H2 ##

### This is an H3 ######
```

Lists

Markdown supports ordered (numbered) and unordered (bulleted) lists.

Unordered lists use asterisks, pluses, and hyphens – interchangably – as list markers:

```
*   Red
*   Green
*   Blue
```

is equivalent to:

```
+   Red
+   Green
+   Blue
```

and:

```
-   Red
-   Green
-   Blue
```

Ordered lists use numbers followed by periods:

```
1.  Bird
2.  McHale
3.  Parish
```

It's important to note that the actual numbers you use to mark the list have no effect on the HTML output Markdown produces. The HTML Markdown produces from the above list is:

```
<ol>
<li>Bird</li>
<li>McHale</li>
<li>Parish</li>
</ol>
```

If you instead wrote the list in Markdown like this:

```
1.  Bird
1.  McHale
1.  Parish
```

or even:

```
3. Bird
1. McHale
8. Parish
```

you'd get the exact same HTML output. The point is, if you want to, you can use ordinal numbers in your ordered Markdown lists, so that the numbers in your source match the numbers in your published HTML. But if you want to be lazy, you don't have to. List markers typically start at the left margin, but may be indented by up to three spaces. List markers must be followed by one or more spaces or a tab.

To make lists look nice, you can wrap items with hanging indents:

```
*   Lorem ipsum dolor sit amet, consectetuer adipiscing elit.
    Aliquam hendrerit mi posuere lectus. Vestibulum enim wisi,
    viverra nec, fringilla in, laoreet vitae, risus.
*   Donec sit amet nisl. Aliquam semper ipsum sit amet velit.
    Suspendisse id sem consectetuer libero luctus adipiscing.
```

But you don't have to.

If list items are separated by blank lines, Markdown will wrap the items in `<p>` tags in the HTML output. For example, this input:

```
*   Bird
*   Magic
```

will turn into:

```
<ul>
<li>Bird</li>
<li>Magic</li>
</ul>
```

But this:

```
*   Bird

*   Magic
```

will turn into:

```
<ul>
<li><p>Bird</p></li>
<li><p>Magic</p></li>
</ul>
```

List items may consist of multiple paragraphs. Each subsequent paragraph in a list item must be indented by either 4 spaces or one tab:

```
1.  This is a list item with two paragraphs. Lorem ipsum dolor
    sit amet, consectetuer adipiscing elit. Aliquam hendrerit
    mi posuere lectus.

    Vestibulum enim wisi, viverra nec, fringilla in, laoreet
    vitae, risus. Donec sit amet nisl. Aliquam semper ipsum
    sit amet velit.

2.  Suspendisse id sem consectetuer libero luctus adipiscing.
```

To put a code block within a list item, the code block needs to be indented *twice* – 8 spaces or two tabs:

```
*   A list item with a code block:

        <code goes here>
```

It's worth noting that it's possible to trigger an ordered list by accident, by writing something like this:

```
1986. What a great season.
```

In other words, a *number-period-space* sequence at the beginning of a line. To avoid this, you can backslash-escape the period:

```
1986\. What a great season.
```

Code blocks

Pre-formatted code blocks are used for writing about programming or markup source code. Rather than forming normal paragraphs, the lines of a code block are interpreted literally. Markdown wraps a code block in both `<pre>` and `<code>` tags.

To produce a code block in Markdown, simply indent every line of the block by at least 4 spaces or 1 tab. For example, given this input:

```
This is a normal paragraph:

    This is a code block.
```

Markdown will generate:

```
<p>This is a normal paragraph:</p>

<pre><code>This is a code block.
</code></pre>
```

One level of indentation – 4 spaces or 1 tab – is removed from each line of the code block. For example, this:

```
Here is an example of AppleScript:

    tell application "Foo"
        beep
    end tell
```

will turn into:

```
<p>Here is an example of AppleScript:</p>

<pre><code>tell application "Foo"
    beep
end tell
</code></pre>
```

A code block continues until it reaches a line that is not indented (or the end of the article).

Within a code block, ampersands (`&`) and angle brackets (`<` and `>`) are automatically converted into HTML entities. This makes it very easy to include example HTML source code using Markdown – just paste it and indent it, and Markdown will handle the hassle of encoding the ampersands and angle brackets. For example, this:

```
    <div class="footer">
        &copy; 2004 Foo Corporation
    </div>
```

will turn into:

```
<pre><code>&lt;div class="footer"&gt;
    &amp;copy; 2004 Foo Corporation
&lt;/div&gt;
</code></pre>
```

Regular Markdown syntax is not processed within code blocks. E.g., asterisks are just literal asterisks within a code block. This means it's also easy to use Markdown to write about Markdown's own syntax.

Highlighted code blocks

We support syntax highlighting for the following languages (https://github.com/jneen/rouge/wiki/list-of-supported-languages-and-lexers). To enable syntax highlighting on a code block use the following format:

```
~~~ language name
# code goes here
~~~
```

For example:

```
~~~ ruby
class Dog
  def initialize(breed, name)
    @breed = breed
    @name = name
  end

  def bark
    puts 'Ruff! Ruff!'
  end
end
~~~
```

becomes:

```ruby
class Dog
  def initialize(breed, name)
    @breed = breed
    @name = name
  end

  def bark
    puts 'Ruff! Ruff!'
  end
end
```

Horizontal rules

You can produce a horizontal rule tag (`<hr />`) by placing three or more hyphens, asterisks, or underscores on a line by themselves. If you wish, you may use spaces between the hyphens or asterisks. Each of the following lines will produce a horizontal rule:

```
* * *

***

*****

- - -

---------------------------------------
```

Tables

In general we recommend no more than two-column tables because we want to ensure that they display well on mobiles in portrait mode.

The following snippet will create a two-column table with four rows (including header row).

```
| Col1 Header  | Col2 Header |
| -------------| ------------|
| data1        | data3       |
| data2        | data4       |
| data3        | data5       |
```

Blockquotes

Blockquotes are used to display quotes and excerpts in learning material.

```
> Blockquotes are displayed by preceding text with a greater than symbol. Italics can also be us
ed within blockquotes.
>
> This is a new paragraph within the blockquote.
```

Span elements

Links

Markdown supports two style of links: *inline* and *reference*.

In both styles, the link text is delimited by [square brackets].

To create an inline link, use a set of regular parentheses immediately after the link text's closing square bracket. Inside the parentheses, put the URL where you want the link to point, along with an *optional* title for the link, surrounded in quotes. For example:

```
This is [an example](http://example.com/ "Title") inline link.

[This link](http://example.net/) has no title attribute.
```

Will produce:

```
<p>This is <a href="http://example.com/" title="Title">
an example</a> inline link.</p>

<p><a href="http://example.net/">This link</a> has no
title attribute.</p>
```

Emphasis

Markdown treats asterisks (`*`) and underscores (`_`) as indicators of emphasis. Text wrapped with one `*` or `_` will be wrapped with an HTML `<em>` tag; double `*`'s or `_`'s will be wrapped with an HTML `<strong>` tag. E.g., this input:

```
*single asterisks*

_single underscores_

**double asterisks**

__double underscores__
```

will produce:

```
<em>single asterisks</em>

<em>single underscores</em>

<strong>double asterisks</strong>

<strong>double underscores</strong>
```

You can use whichever style you prefer; the lone restriction is that the same character must be used to open and close an emphasis span.

Emphasis can be used in the middle of a word:

```
un*believ*able
```

But if you surround an `*` or `_` with spaces, it'll be treated as a literal asterisk or underscore.

To produce a literal asterisk or underscore at a position where it would otherwise be used as an emphasis delimiter, you can backslash escape it:

```
\*this text is surrounded by literal asterisks\*
```

Code

To indicate a span of code, wrap it with backtick quotes (`` ` ``). Unlike a pre-formatted code block, a code span indicates code within a normal paragraph. For example:

```
Use the `printf()` function.
```

will produce:

```
<p>Use the <code>printf()</code> function.</p>
```

To include a literal backtick character within a code span, you can use multiple backticks as the opening and closing delimiters:

```
``There is a literal backtick (`) here.``
```

which will produce this:

```
<p><code>There is a literal backtick (`) here.</code></p>
```

The backtick delimiters surrounding a code span may include spaces – one after the opening, one before the closing. This allows you to place literal backtick characters at the beginning or end of a code span:

```
    A single backtick in a code span: `` ` ``

    A backtick-delimited string in a code span: `` `foo` ``
```

will produce:

```
<p>A single backtick in a code span: <code>`</code></p>

<p>A backtick-delimited string in a code span: <code>`foo`</code></p>
```

With a code span, ampersands and angle brackets are encoded as HTML entities automatically, which makes it easy to include example HTML tags. Markdown will turn this:

```
Please don't use any `<blink>` tags.
```

into:

```
<p>Please don't use any <code>&lt;blink&gt;</code> tags.</p>
```

You can write this:

```
`&#8212;` is the decimal-encoded equivalent of `&mdash;`.
```

to produce:

```
<p><code>&amp;#8212;</code> is the decimal-encoded
equivalent of <code>&amp;mdash;</code>.</p>
```

Images

Markdown uses an image syntax that is intended to resemble the syntax for links, allowing for two styles: *inline* and *reference*.

Inline image syntax looks like this:

```
![Alt text](https://example.com/path/to/img.jpg)

![Alt text](https://example.com/path/to/img.jpg "Optional title")
```

That is:

- An exclamation mark: `!`;
- followed by a set of square brackets, containing the `alt` attribute text for the image;
- followed by a set of parentheses, containing the URL or path to the image, and an optional `title` attribute enclosed in double or single quotes.
- Only use landscape images and make sure they are exactly 648px wide.
- Only use images that are hosted securely. Secure images will have a URL that begins with `https://`.

As of this writing, Markdown has no syntax for specifying the dimensions of an image; please make sure your images are sized correctly.

Subscript and superscript

You can use the HTML tags directly for superscript and subscript text. This works as follows:

```
This is text with some <sub>subscript</sub>
```

will produce

This is text with some _{subscript}

And

```
This is text with some <sup>superscript</sup>
```

will produce

This is text with some ^{superscript}

Miscellaneous

Automatic Links

Markdown supports a shortcut style for creating "automatic" links for URLs and email addresses: simply surround the URL or email address with angle brackets. What this means is that if you want to show the actual text of a URL or email address, and also have it be a clickable link, you can do this:

```
<http://example.com/>
```

Markdown will turn this into:

```
<a href="http://example.com/">http://example.com/</a>
```

Backslash Escapes

Markdown allows you to use backslash escapes to generate literal characters which would otherwise have special meaning in Markdown's formatting syntax. For example, if you wanted to surround a word with literal asterisks (instead of an HTML `<em>` tag), you can use backslashes before the asterisks, like this:

```
\*literal asterisks\*
```

Markdown provides backslash escapes for the following characters:

```
\   backslash
`   backtick
*   asterisk
_   underscore
{}  curly braces
[]  square brackets
()  parentheses
#   hash mark
    +   plus sign
    -   minus sign (hyphen)
.   dot
!   exclamation mark
```

Mathematical symbols

The Markdown used on FutureLearn supports the embedding of mathematics inline and as block elements. Maths is written using fragments of the $\LaTeX$ language surrounded by `$$` symbols. This is rendered in the page using the MathJax (http://www.mathjax.org/) library, which has excellent documentation on its homepage, including details of which features (http://docs.mathjax.org/en/latest/tex.html) of $\LaTeX$ are supported.

For example:

```
We can render $$x$$, $$y$$ and $$\frac{x}{y}$$
```

Will be rendered as:

We can render x, y and $\frac{x}{y}$

In addition block level mathematics are also supported. For example:

```
Here is an interesting equation:
$$
\frac{\partial^2\psi}{\partial x^2} = \frac{1}{c^2}\frac{\partial^2\psi}{\partial t^2}
$$
What is it?
```

Will be rendered as:

Here is an interesting equation: $$\frac{\partial^2\psi}{\partial x^2} = \frac{1}{c^2}\frac{\partial^2\psi}{\partial t^2}$$ What is it?