
# xDev Magazine Markup Guide

## How to use Markdown to format articles

July 12, 2013—v1.2

# Contents

**Introduction** . . . . . **3**

The Goals of Our Formatting System . . . . . 3

**Introducing Markdown** . . . . . **3**

The Markdown Method . . . . . 4

Article Structure . . . . . 5

Markdown Basics . . . . . 6

*Paragraphs*. . . . . *7*

*Headlines*. . . . . *7*

*Quoted Text* . . . . . *7*

*Bulleted Lists*. . . . . *7*

*Code*. . . . . *8*

*Graphics* . . . . . *9*

*Sidebars* . . . . . *10*

*Special Characters* . . . . . *10*

*Tables*. . . . . *11*

*RSD Tags* . . . . . *11*

# Introduction

Welcome to the *xDev Magazine* Markup Guide. This is a tutorial to explain the system *xDev* uses for submitted articles for layout in the magazine. Having all authors format articles in the same manner makes things easier for everyone, and helps the magazine maintain a consistent look. Properly formatting your articles saves us time and improves accuracy as we aren't guessing as to how you wanted the article formatted.

Formatting simply means you indicating to us which parts of your article are which: for instance, you can indicate that a line of text is a headline, a caption, or some source code.

## The Goals of Our Formatting System

We had several objectives in developing our system. We wanted a format that was:

- plain text, so there's no dependency upon a particular word processing file format or operating system.
- powerful enough to express all of the complex formatting required for technical articles.
- flexible, so we can repurpose your writing for the web or other media.
- simple, so authors can focus on writing instead of formatting.
- visually clean, for easier proofreading and editing.

Originally our format for doing this was an XML format. XML met most of the above goals, except for the last two: XML is not as simple as we'd like (and minor formatting errors introduced by authors really slow down the production process), and with its visible tags XML is rather ugly and hard to read.

# Introducing Markdown

Thus in 2012 we have introduced a new system based on the popular `Markdown` (http://daringfireball.net/projects/markdown/) format created by John Gruber. Markdown is a plain text writing format designed for web writing. A large part of its purpose is to look clean and be readable without ugly HTML-style tags. Since Markdown easily translates into HTML or XML, this works great for *xDev* as we can simply convert Markdown articles to XML and continue to use our standard XML-based workflow.

This means as an author, you are free to use either Markdown or XML for your articles. However, since Markdown is so much easier, I suspect most authors will use it instead of messing with XML. We'd also encourage it, as it makes proofreading easier, and since Markdown article all convert consistently to our XML format, we have fewer XML errors to slow down production.

Another advantage of Markdown is that since it is designed with the web in mind, it supports the use of HTML/XML tags within regular Markdown text. This means you are free to incorporate traditional *xDev* XML tags within your Markdown articles. This necessary because Markdown has limited formatting capabilities and doesn't support every XML tag *xDev* supports. Since most

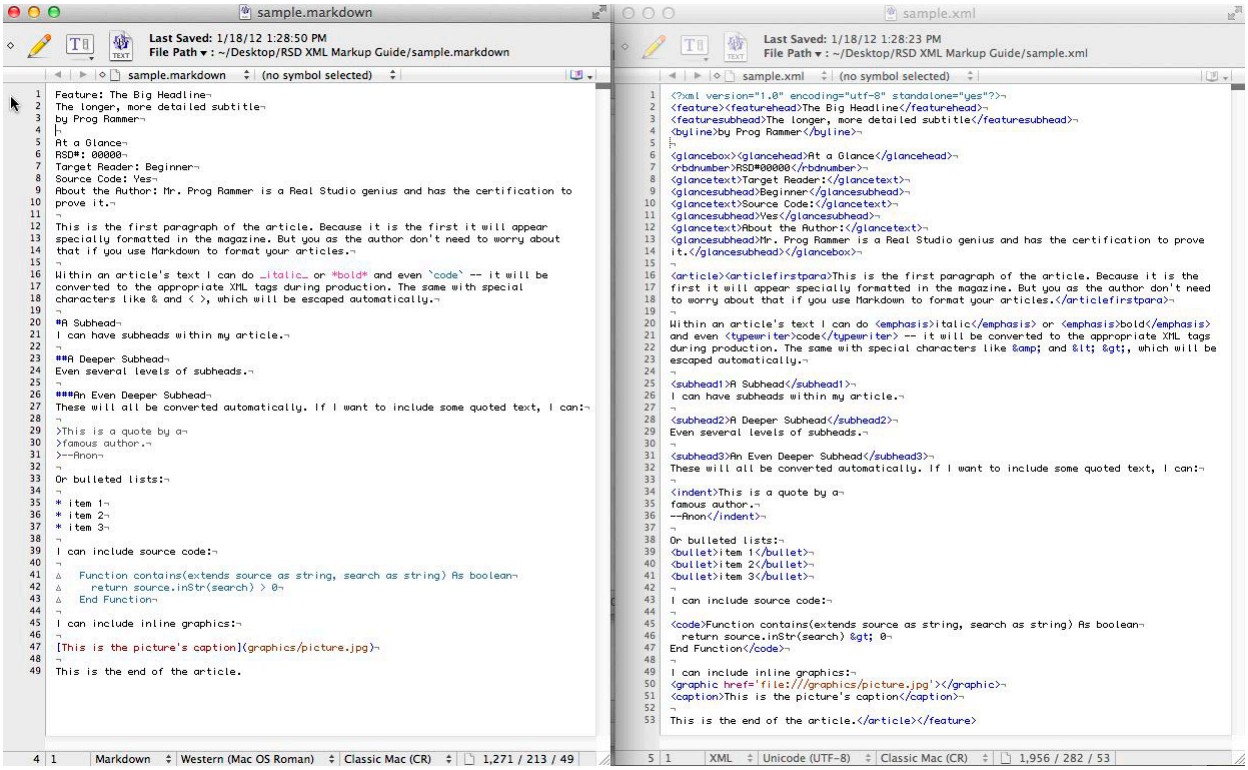
of those formatting features are rarely used, you will only need to use XML occasionally within a Markdown article. Thus is it is still important your learn the regular *xDev* XML format, just in case you need to use some of those specialized tags.

## The Markdown Method

If you're accustomed to the *xDev* XML method of formatting articles, the Markdown method will seem strangely bereft of formatting. Markdown "tags" are designed to be ordinary text used in a way that makes plain text look formatted even when it's not converted to HTML. For instance, *italics* are done by putting underscores on either side of a word or phrase, you make a bulleted list simply by putting an asterix or number in front of each line, headlines are preceded by one or more # signs (to indicate the head level), and you tell Markdown something is code simply by indenting it.

In a sense, Markdown has no tags!

This is a graphic showing a comparison of the same short article formatted as Markdown and XML:




**Which would** *you rather read?*

If you're using a Markdown-aware editor, Markdown elements are color-coded which is helpful (though not essential). Any text editor or word processor can be used to edit Markdown. I use Bare Bones Software's BBEdit (http://www.barebones.com/products/bbedit/index.html) on the Mac, but many text editors support Markdown, even several editors for the iPad.

Some editors, such as BBEdit, allow customization, which can make working with Markdown even easier. For instance, I have added keyboard shortcuts in BBEdit to automatically put underscores or astrixes around the selected text (for italics and bold). BBEdit (or other text editors that can run Perl scripts on your text) can even convert your text from Markdown to HTML, which can be a useful way to preview your article.

We've also got *xDev Author*, our own free editor for Mac and Windows which is optimized for producing articles for the magazine. It includes templates to make it easier for you to set up an basic article structure, keyboard shortcuts for quickly "tagging" text in Markdown format and doing things like indenting or unindenting text, and a nice preview function so you can get an idea of how your article will appear when published.

Typically Markdown files have a .md extension, but this is mere convention: since Markdown is plain text, you can use .txt if that works better for you.

## Article Structure

A Markdown article for *xDev* follows a specific structure. It's not very complicated, but it must be followed precisely. Here is the outline of a typical article:

Header
At a Glance Box
Article
Graphics
Sidebars
Code Listings

All articles have the first three elements—the post-article elements are *optional*. Each of these elements must be separated by a blank line. The end of the article is defined by at least three blank lines (so don't use three blank lines in your article). Graphics, Sidebars, and Code Listings follow the article (in any order you'd like). They each have specific formatting, so read the sections about them to see how they are formatted.

The "Header" is the first three lines of the article and consists of the article's title, subtitle, and byline (author's name). The first line *always* begins with the type of article and a colon, either "Feature:" or "Column:" followed by the full title. So an article header looks like this:

Feature: Short Title Here
Longer more detailed and descriptive subtitle here
by Arthur Nayme

All articles are required to have these three lines and the byline needs to be as shown (with the lowercase word "by" preceeding the author's name). If desired, an author may also include an email address after the name, in parenthesis as in:

by Arthur Nayme (arthur_nayme48398@aol.com)

After the header, there's a blank line and the "At a Glance Box." The exact contents of "At a Glance" varies according to the article, but it generally looks something like this:

At a Glance
XD#: 00000
Target Reader: Intermediate
Source Code: Yes
RS Version Required: 2013r1+
Platform(s) Supported: OSX, Windows
Platform(s) Untested: Linux
About the Author: Arthur is a world-famous snuffendorf export and he knows everything about using Xojo to track their migration patterns. He frequently writes for *xDev Magazine* as well as *MacTech*, *Dr. Dobbs Journal*, and many other publications. He lives in the U.S. Northwest.

Other than the title "At a Glance" each line is a heading followed by a colon and the text for that item. Authors can add or delete lines as required. For instance, if an article doesn't have source code, there's no point in including the "Platform(s) Supported" and "Platform(s) Untested:" lines, so those can be removed.

All articles should have the first four items ("At a Glance", "xDev#", "Target Reader", "Source Code") and end with the "About the Author" line.

After another blank line, the full text of the article follows. Just don't forget that the end of the article is defined by at least three blank lines, and you can follow that with sections of Graphics, Sidebars, and Code Listings as needed.

## Markdown Basics

If you want to learn more about Markdown, your best source is to read Gruber's full documentation (http://daringfireball.net/projects/markdown/syntax). However, if you just want to learn enough about it to use for formatting your *xDev* articles, here's a quick roundup of just what you'll need.

### *Paragraphs*

One important aspect of Markdown is that you need to put blank lines between paragraphs. If you run everything together, it will not work as the lines will be joined into a single paragraph.

(One exception to this is text after a headline doesn't require a blank line before it, so you can start the paragraph on the next line after the subhead.)

### *Headlines*

There are three types of headlines used in *xDev*: the main article headline, the longer article subtitle, and various levels of subheads used throughout an article.

Because the first two headlines are special, they require no formatting at all: they are simply the first two lines of your article.

For subheads, just use one for more # characters at the beginning of subheads. One translates to *Subhead1*, two to *Subhead2*, and so on. Like this:

```
#First level subhead
##Second level subhead
###Third level subhead
```

### *Quoted Text*

To indent some quoted text, just precede each line with a > symbol. This is exactly how old-fashioned plain text email programs quote text so it's familiar and most text editors support adding/removing quoted text in that fashion.

Because you, the author, aren't sure about how the text will wrap in the final article, it is best if you put a > in front of each paragraph in the quoted material. These lines make look weird in your editor since they may wrap, depending on the size of your window and length of paragraph, but the text will wrap correctly in the final article. If, however, you have special text that needs to break a particular place (like a poem), just manually break each line and put a > in front of each line.

Note that when you use quoted text, Markdown sytax still works within that text. This isn't too useful for `xDev`, as the magazine doesn't support, say, indented subheads, but it will work for things such as *italics* or `code`.

### *Bulleted Lists*

xDev supports two types of bullet lists: regular bullets and numbered lists. To do regular bullets, just place an asterisk in front of each line like this:

```
* item 1
* item 2
* item 3
```

To do numbered lines, just put a number (any number, it doesn't matter, as it won't be honored).

```
1. item 1
1. item 2
1. item 3
```

### *Code*

There are three types of code formatting used in *xDev*. First is *inline* code. This is code used in the middle of a paragraph. It is also used for variable names or other items a user might type. It translates to the `Typewriter` XML tag. To do this, simply surround the text with backtick marks (`\`, as in \`type this text\`becomes`type this text`` (I have a keyboard shortcut in BBEdit to mark text like this for me).

When you want to display a block of code, just indent it with tab characters (note that for clarity I'm using a → to represent an invisible tab character):

```
→ Sub drawRotatedText(g as graphics, rotation as double, text as string, x as integer, y
     as integer)
→
→   dim s as stringShape = new stringShape
→   dim d as new group2D
→
→   s.text = text
→   s.textSize = g.textSize
→   s.textFont = g.textFont
→   s.bold = g.bold
→   s.italic = g.italic
→   s.x = g.stringWidth(text) / 2
→   s.y = g.stringHeight(text, 1000) / 2
→
→   d.append s
→   d.rotation = degreesToRadians(rotation)
→
→   g.drawObject d, x, y
→
→ End Sub
```

(You can indent with four or more spaces, but I find that's more cumbersome.)

The third type of code used in `xDev` is longer code that appears in a Code Listing at the end of an article. You do this in the same way—indent each line with a tab—however you don't place this

within the article but at the very end. You should prefix each code listing with a level 3 subhead that titles it, followed by a blank line. These following the format: "Code Listing X: title" where X is the listing number (you can then refer the reader to that listing by number throughout your article) and title is the name or description of the listing.

A code listing block looks like this:

```
###Code Listing 1: The DrawRotatedText Method
→  Sub drawRotatedText(g as graphics, rotation as double, text as string, x as integer, y
     as integer)
→   dim s as stringShape = new stringShape
→   dim d as new group2D
→   s.text = text
→   s.textSize = g.textSize
→   s.textFont = g.textFont
→   s.bold = g.bold
→   ...
```

Note that is *very* important that this first line is **not** indented with a tab. Only the code itself should be indented. That's because Markdown does not convert its syntax in anything it considers to be code (i.e. indented content, which it interprets literally). It assumes that code is formatted the way it should be. The singular exception is that Markdown will still convert special symbols such as `@` and `<` and `>` and `&` symbols so you don't need to do anything special to escape or encode those, even within code. Thus if you indent the Code Listing subhead, the `###` won't translate into a proper subhead.

### *Graphics*

Linking to a graphic in an `xDev` article is a little ugly in XML and it must be done exactly right or it causes errors. However, if you use Markdown, it is very simple. The syntax is:

```
[caption text](path-to-graphic)
```

It is generally suggested you place your graphics inside a folder called "graphics" and since `xDev` graphics are usually labeled with a Figure number, a typical graphic would be something like this:

```
[Figure 1: This is picture1's caption](graphics/picture1.jpg)
```

Such graphics are placed at the *end* of your article and you refer to them by number throughout your article.

*xDev* also uses *inline* graphics—which are smaller graphics that appear in the middle of an article and don't require a caption—you just do the same thing with a exclamation mark in front and use the word "graphic" for the caption text:

```
![graphic](graphics/picture2.jpg)
```

You would place this within the text of your article, but be sure it on its own line (separated by blank lines above and below and not indented).

Use inline graphics only for smaller graphics or when it is impertive the graphic be near the associated text (when using Figure-style graphics, the graphic may or may not appear on a page near the associated text). You should use inline graphics sparingly, as they can cause layout problems.

### *Sidebars*

Sidebars are blocks of text on a related subject to your main article, but are too specific or too off-topic to include as part of the main text. They can be any length. Sidebars are *always* appended to the *end* of your article. The first line should be the title, preceded by a single `#` mark and the word Sidebar with a colon: `#Sidebar:`

```
#Sidebar: Title Goes Here
```

Sidebars are not numbered—in your article you refer to them by their title.

The entire sidebar—including the title line—needs to be preceeded by > marks just like quoted text. Within the sidebar you can include other Markdown elements, such as subheads, code, quoted text, graphics, etc. The only key is that every line (including blank lines between elements) needs to be preceded by a > mark to indicate it's part of the sidebar.

The easiest way to do this is to write your text first and just do everything the way you would if the text was part of the main article. Then as the final step, use your text editor's quoted text or prefix feature to add a > in front of each line.

### *Special Characters*

Markdown has really nice support for special characters. It is intelligent and understands how you're using various characters, allowing you to simply use them naturally. For instance, don't worry about making your quotation marks "smart"—Markdown will do that automatically.

The same goes for the < and > and `&` characters—you no longer need to worry about encoding them, either within your text or within your code.

Dashes are created automatically from double-hyphens.

If you want to use HTML entities for special characters, you can: Markdown won't modify them. (The exception for this that they won't work within code. If you use HTML entities within code,

the & will be converted to an entity itself and thus the HTML for your entity will appear instead of the entity itself.)

If Markdown is converting a symbol you don't want converted, you can ensure it doesn't get converted by preceeding it with a backslash () to escape it.

### *Tables*

Markdown does not support tables, so these must be done using XML table tags. Just put your tab-delimited table between HTML-style `<table></table>` tags. Each row is a separate line, and you put a tab between each column.

Here's an example (→ is used to show a tab character):

```
<table>
Platform→ Line Ending→  Shortcut Key→ File System
Mac OS X→ chr(13)→  Command→  HFS+
Linux→    chr(10)→  Control→  ext2, ext3 and ext4
Windows→  chr(13) + chr(10)→    Control→  FAT, NTFS, exFAT
</table>
```

You can also use space-delimited tables, as long as there's at least three spaces between each column. If you use an editor with a monospaced font, a space-delimited tables align nicely and look better than tabs:

```
<table>
Platform   Line Ending         Shortcut Key   File System
Mac OS X   chr(13)             Command        HFS+
Linux      chr(10)             Control        ext2, ext3 and ext4
Windows    chr(13) + chr(10)   Control        FAT, NTFS, exFAT
</table>
```

If you use our *xDev Author* software, it will assist you in creating tables like these. These tables will also preview correctly.

### *xDev Tags*

The magazine supports some formatting (such as tables) that Markdown doesn't. For these, just use the xDev XML tags. For instance, `x<superscript>2</superscript>` will appear as x<sup>2</sup>. *xDev* tags that Markdown doesn't support include:

```
XML Tag             Markdown Equivalent
```

```
<authorcomment>    use <!— HTML-style —>comment instead
<booktitle>        use _italics_ instead
<codecomment>      not used any more; comments in code are automatically detected
<subscript>        use XML (i.e. H<subscript>2</subscript>0)
<superscript>      use XML (i.e. item<superscript>1</superscript>)
<table>            use XML (i.e. wrap tab-delimited text with <table></table>tags)
```