Extended Syntax

Advanced features that build on the basic Markdown syntax.

Overview

The [basic syntax](#) outlined in John Gruber's original design document added many of the elements needed on a day-to-day basis, but it wasn't enough for some people. That's where extended syntax comes in.

Several individuals and organizations took it upon themselves to extend the basic syntax by adding additional elements like tables, code blocks, syntax highlighting, URL auto-linking, and footnotes. These elements can be enabled by using a lightweight markup language that builds upon the basic Markdown syntax, or by adding an extension to a compatible Markdown processor.

Availability

Extended syntax isn't available in all Markdown applications. You'll need to check whether or not the lightweight markup language your application is using supports extended syntax. If it doesn't, it may still be possible to enable extensions in your Markdown processor.

Lightweight Markup Languages

There are several lightweight markup languages that are *supersets* of Markdown. They include Gruber's basic syntax and build upon it by adding additional elements like tables, code blocks, syntax highlighting, URL auto-linking, and footnotes. Many of the most popular Markdown applications use one of the following lightweight markup languages:

- [CommonMark](#)
- [GitHub Flavored Markdown (GFM)](#)
- [Markdown Extra](#)
- [MultiMarkdown](#)

Markdown Processors

There are [dozens of Markdown processors](#) available. Many of them allow you to add extensions that enable extended syntax. Check your processor's documentation for more information.

Tables

To add a table, use three or more hyphens (`---`) to create each column's header, and use pipes (`|`) to separate each column. You can optionally add pipes on either end of the table.

```
| Syntax      | Description |
| ----------- | ----------- |
| Header      | Title       |
| Paragraph   | Text        |
```

The rendered output looks like this:

Syntax	Description
Header	Title
Paragraph	Text

Cell widths can vary, as shown below. The rendered output will look the same.

```
| Syntax | Description |
| --- | ----------- |
| Header | Title |
| Paragraph | Text |
```

> 💡 **Tip:** Creating tables with hyphens and pipes can be tedious. To speed up the process, try using the [Markdown Tables Generator](). Build a table using the graphical interface, and then copy the generated Markdown-formatted text into your file.

Alignment

You can align text in the columns to the left, right, or center by adding a colon (`:`) to the left, right, or on both side of the hyphens within the header row.

```
| Syntax     | Description | Test Text     |
| :---       |     :----:  |          ---: |
| Header     | Title       | Here's this   |
| Paragraph  | Text        | And more      |
```

The rendered output looks like this:

Syntax	Description	Test Text
Header	Title	Here's this
Paragraph	Text	And more

Formatting Text in Tables

You can format the text within tables. For example, you can add [links](), [code]() (words or phrases in tick marks (`) only, not [code blocks]()), and [emphasis]().

You can't add headings, blockquotes, lists, horizontal rules, images, or HTML tags.

Escaping Pipe Characters in Tables

You can display a pipe (`|`) character in a table by using its HTML character code (`|`).

Fenced Code Blocks

The basic Markdown syntax allows you to create [code blocks]() by indenting lines by four spaces or one tab. If you find that inconvenient, try using fenced code blocks. Depending on your Markdown processor or editor, you'll use three tick marks (` ``` `) or three tildes (`~~~`) on the lines before and after the code block. The best part? You don't have to indent any lines!

```
```
{
  "firstName": "John",
  "lastName": "Smith",
  "age": 25
}
```
```

The rendered output looks like this:

```
{
    "firstName": "John",
    "lastName": "Smith",
    "age": 25
}
```

> 💡 **Tip:** Need to display tick marks inside a code block? See [this section](#) to learn how to escape them.

Syntax Highlighting

Many Markdown processors support syntax highlighting for fenced code blocks. This feature allows you to add color highlighting for whatever language your code was written in. To add syntax highlighting, specify a language next to the tick marks before the fenced code block.

```
```json
{
    "firstName": "John",
    "lastName": "Smith",
    "age": 25
}
```
```

The rendered output looks like this:

```
{
    "firstName": "John",
    "lastName": "Smith",
    "age": 25
}
```

Footnotes

Footnotes allow you to add notes and references without cluttering the body of the document. When you create a footnote, a superscript number with a link appears where you added the footnote reference. Readers can click the link to jump to the content of the footnote at the bottom of the page.

To create a footnote reference, add a caret and an identifier inside brackets (`[^1]`). Identifiers can be numbers or words, but they can't contain spaces or tabs. Identifiers only correlate the footnote reference with the footnote itself — in the output, footnotes are numbered sequentially.

Add the footnote using another caret and number inside brackets with a colon and text (`[^1]: My footnote.`). You don't have to put footnotes at the end of the document. You can put them anywhere except inside other elements like lists, block quotes, and tables.

```
Here's a simple footnote,[^1] and here's a longer one.[^bignote]

[^1]: This is the first footnote.

[^bignote]: Here's one with multiple paragraphs and code.

    Indent paragraphs to include them in the footnote.

    `{ my code }`

    Add as many paragraphs as you like.
```

The rendered output looks like this:

Here's a simple footnote,[1] and here's a longer one.[2]

---

1. This is the first footnote. ↩

2. Here's one with multiple paragraphs and code.

   Indent paragraphs to include them in the footnote.

   ```
   { my code }
   ```

   Add as many paragraphs as you like. ↩

Heading IDs

Many Markdown processors support custom IDs for [headings](#) — some Markdown processors automatically add them. Adding custom IDs allows you to link directly to headings and modify them with CSS. To add a custom heading ID, enclose the custom ID in curly braces on the same line as the heading.

```
### My Great Heading {#custom-id}
```

The HTML looks like this:

```html
<h3 id="custom-id">My Great Heading</h3>
```

Linking to Heading IDs

You can link to headings with custom IDs in the file by creating a [standard link](#) with a number sign (`#`) followed by the custom heading ID.

Markdown	HTML	Rendered Output
`[Heading IDs](#heading-ids)`	`<a href="#heading-ids">Heading IDs</a>`	[Heading IDs](#)

Other websites can link to the heading by adding the custom heading ID to the full URL of the webpage (e.g, `[Heading IDs](https://www.markdownguide.org/extended-syntax#heading-ids)`).

Definition Lists

Some Markdown processors allow you to create *definition lists* of terms and their corresponding definitions. To create a definition list, type the term on the first line. On the next line, type a colon followed by a space and the definition.

```
First Term
: This is the definition of the first term.

Second Term
: This is one definition of the second term.
: This is another definition of the second term.
```

The HTML looks like this:

```
<dl>
  <dt>First Term</dt>
  <dd>This is the definition of the first term.</dd>
  <dt>Second Term</dt>
  <dd>This is one definition of the second term. </dd>
  <dd>This is another definition of the second term.</dd>
</dl>
```

The rendered output looks like this:

First Term
This is the definition of the first term.

Second Term
This is one definition of the second term.

This is another definition of the second term.

Strikethrough

You can "strikethrough" words by putting a horizontal line through the center of them. The result looks ~~like this~~. This feature allows you to indicate that certain words are a mistake not meant for inclusion in the document. To strikethrough words, use two tilde symbols (~~) before and after the words.

```
~~The world is flat.~~ We now know that the world is round.
```

The rendered output looks like this:

~~The world is flat.~~ We now know that the world is round.

Task Lists

Task lists allow you to create a list of items with checkboxes. In Markdown applications that support task lists, checkboxes will be displayed next to the content. To create a task list, add dashes (-) and brackets with a space ([ ]) in front of task list items. To select a checkbox, add an x in between the brackets ([x]).

```
- [x] Write the press release
- [ ] Update the website
- [ ] Contact the media
```

The rendered output looks like this:

- [x] Write the press release
- [ ] Update the website
- [ ] Contact the media

Automatic URL Linking

Many Markdown processors automatically turn URLs into links. That means if you type http://www.example.com, your Markdown processor will automatically turn it into a link even though you haven't [used brackets](#).

```
http://www.example.com
```

The rendered output looks like this:

http://www.example.com

Disabling Automatic URL Linking

If you don't want a URL to be automatically linked, you can remove the link by [denoting the URL as code](#) with tick marks.

```
`http://www.example.com`
```

The rendered output looks like this:

http://www.example.com

Want to learn more Markdown?

Hey there! 👋 I'm Matt Cone. I'm a technical writer who's been using Markdown for years. If you liked this and want to hear about my new Markdown tutorials, enter your email address below. I don't send spam.

> Your email address

Subscribe

[About](#)   [API](#)   [Contact](#)   [GitHub](#)   [Privacy Policy](#)

© 2019. A [Matt Cone](#) project. [CC BY-SA 4.0](#).