Table of Contents

DocFX Overview

DocFX is an API documentation generator for .NET, which currently supports C#, VB and F#. It generates API reference documentation from triple-slash comments in your source code. It also allows you to use Markdown files to create additional topics such as tutorials and how-tos, and to customize the generated reference documentation. DocFX builds a static HTML website from your source code and Markdown files, which can be easily hosted on any web server (for example, github.io). Also, DocFX provides you the flexibility to customize the layout and style of your website through templates.

DocFX also has the following cool features:

- Integration with your source code. You can click "View Source" on an API to navigate to the source code in GitHub (your source code must be pushed to GitHub).
- Cross-platform support. We have an exe version that runs natively on Windows and with Mono it can also run on Linux and macOS.
- Integration with Visual Studio. You can seamlessly use DocFX within Visual Studio.
- Markdown extensions. We introduced DocFX Flavored Markdown(DFM) to help you write API documentation. DFM is 100% compatible with GitHub Flavored Markdown(GFM) with some useful extensions, like file inclusion, code snippet, cross reference, and yaml header. For a detailed description about DFM, please refer to DFM.

> **⚠ WARNING**
>
> **Prerequisites** Visual Studio 2019 is needed for DocFX metadata msbuild projects. It's not required when generating metadata directly from source code (.cs, .vb) or assemblies (.dll)

Setting up DocFX

DocFX is an API documentation generator for .NET, which currently supports C#, VB and F#. It generates API reference documentation from triple-slash comments in your source code. It also allows you to use Markdown files to create additional topics such as tutorials and how-tos, and to customize the generated reference documentation. DocFX builds a static HTML website from your source code and Markdown files, which can be easily hosted on any web server (for example, github.io). Also, DocFX provides you the flexibility to customize the layout and style of your website through templates.

DocFX also has the following cool features:

- Integration with your source code. You can click "View Source" on an API to navigate to the source code in GitHub (your source code must be pushed to GitHub).
- Cross-platform support. We have an exe version that runs natively on Windows and with Mono it can also run on Linux and macOS.
- Integration with Visual Studio. You can seamlessly use DocFX within Visual Studio.
- Markdown extensions. We introduced DocFX Flavored Markdown(DFM) to help you write API documentation. DFM is 100% compatible with GitHub Flavored Markdown(GFM) with some useful extensions, like file inclusion, code snippet, cross reference, and yaml header. For a detailed description about DFM, please refer to DFM.

> ⚠ **WARNING**
>
> **Prerequisites** Visual Studio 2019 is needed for DocFX metadata msbuild projects. It's not required when generating metadata directly from source code (.cs, .vb) or assemblies (.dll)

Use DocFX as a command-line tool

Downloading DocFX

1. Download DocFX from GitHub
2. Extract **docfx.zip** to a local folder
3. Add it to **PATH** so you can run it anywhere

Create a project

1. Navigate to a folder of your choice
2. Right-click and select **Open with Code**
3. Within **Visual Studio Code**, Create a new Terminal Window (`CTRL + SHIFT + '`)
4. Within the Terminal, type `docfx init -q`
5. The command generates a default project named `docfx_project`

Build the website

1. Within the Terminal, type `docfx docfx_project\docfx.json --serve`
2. Now you can view the generated website on http://localhost:8080

Updating the project

1. Open **index.md**
2. Make a small change and save the document
3. Within **Visual Studio Code**, Create a new Terminal Window (`CTRL + SHIFT + '`)
4. Within the Terminal, type `docfx docfx_project\docfx.json`
5. Goto your browser and refresh the website and refresh the page

> ⚠ **WARNING**
>
> Keep the originol Terminal open as this is the local lite webserver When making any changes, use the command `docfx docfx_project\docfx.json` and refresh the website within your browser

Adding a Template to your website

The following will add **DiscordFX** template to your website. This will make the website look and feel like Discord documentation portal

1. Download the source or the zipped file from the releases page
2. In your DocFX project folder, create a new directory named `templates`, if it doesn't already exist
3. Copy the discordfx folder from this repository into your templates folder
4. In your `docfx.json` configuration file, add `templates/discordfx` path into `build.template` property

```
{
  "build": {
    "template": ["default", "templates/discordfx"]
  }
}
```

1. Within the Terminal, type `docfx docfx_project\docfx.json`
2. Goto your browser and refresh the website and refresh the page

> **TIP**
>
> For more templates, see Templates

Working Template

```
{
  "metadata": [
    {
      "src": [
        {
          "files": [
            "src/**.csproj"
          ]
        }
      ],
      "dest": "api",
      "disableGitFeatures": true,
      "disableDefaultFilter": false
    }
  ],
  "build": {
    "content": [
      {
        "files": [
          "api/**.yml",
          "api/index.md"
        ]
      },
      {
        "files": [
          "articles/**.md",
          "articles/**/toc.yml",
          "toc.yml",
          "*.md"
        ]
      }
    ],
    "resource": [
      {
```

```
            "files": [
              "images/**"
            ]
          }
        ],
        "overwrite": [
          {
            "files": [
              "apidoc/**.md"
            ],
            "exclude": [
              "obj/**",
              "_site/**"
            ]
          }
        ],
        "dest": "_site",
        "disableGitFeatures": true,
        "disableDefaultFilter": false,
        "globalMetadataFiles": [],
        "fileMetadataFiles": [],
        "template": [
          "default"
        ],
        "postProcessors": [],
        "markdownEngineName": "markdig",
        "noLangKeyword": false,
        "keepFileLink": false,
        "cleanupCacheHistory": false,
        "globalMetadata": {
          "_appTitle": "DocFX website",
          "_appFooter": "&#0169; 2021 Jason Rose",
          "_enableSearch": "true",
          "_disableContribution": "true",
          "_disableAffix": "false"
        }
      }
    }
```

Markdown Overview

DocFX Flavored Markdown

DocFX supports `DocFX Flavored Markdown`, aka DFM. It supports all GitHub Flavored Markdown syntax and compatible with CommonMark. Also, DFM adds new syntax to support additional functionalities, including cross reference and file inclusion.

> **NOTE**
>
> The default markdown engine generated by `docfx init` has been switched to markdig engine, which is built on the top of markdig. Previous markdown engine `dfm` and `dfm-latest` will be kept for compatibiilty.

Yaml Header

Yaml header in DFM is considered as the metadata for the Markdown file. It will transform to `yamlheader` tag when processed. Yaml header MUST be the first thing in the file and MUST take the form of valid YAML set between triple-dashed lines. Here is a basic example:

```
---
uid: A.md
title: A
---
```

Cross Reference

Cross reference allows you to link to another topic by using its unique identifier (called UID) instead of using its file path.

For conceptual Markdown files UID can be defined by adding a `uid` metadata in YAML header:

```
---
uid: uid_of_the_file
---

This is a conceptual topic with `uid` specified.
```

For reference topics, UIDs are auto generated from source code and can be found in generated YAML files.

You can use one of the following syntax to cross reference a topic with UID defined:

1. Markdown link: `[link_text](xref:uid_of_the_topic)`
2. Auto link: `<xref:uid_of_the_topic>`
3. Shorthand form: `@"uid_of_the_topic"`

All will render to:

```
<a href="url_of_the_topic">link_text</a>
```

If `link_text` is not specified, DocFX will extract the title from the target topic and use it as the link text.

Do not use the `@uid` link in brackets (like this: `(@uid)`). DocFX cannot parse this link. The `@uid` link should be separated with white spaces. If you need to add a link in brackets, use `[](xref:uid)`.

> **NOTE**
>
> Hashtag in `xref` is always treated as separator between file name and anchor name. That means if you have `#` in UID, it has to be encoded to `%23`.
>
> Actually `xref` format follows URI standard so all reserved characters should be encoded.

File Inclusion

DFM adds syntax to include other file parts into current file, the included file will also be considered as in DFM syntax.

There are two types of file inclusion: Inline and block, as similar to inline code span and block code.

> **NOTE**
>
> YAML header is **NOT** supported when the file is an inclusion.

Inline

Inline file inclusion is in the following syntax, in which `<title>` stands for the title of the included file, and `<filepath>` stands for the file path of the included file. The file path can be either absolute or relative.`<filepath>` can be wrapped by `'` or `"`.

> **NOTE**
>
> For inline file inclusion, the file included will be considered as containing only inline tags, for example, `### header` inside the file will not transfer since `<h3>` is a block tag, while `[a](b)` will transform to `<a href='b'>a</a>` since `<a>` is an inline tag. Also, ending white spaces will be **trimmed**, considering ending white spaces in inline inclusion in most cases are typos.

```
...Other inline contents... [!include[<title>](<filepath>)]
```

Block

Block file inclusion must be in a single line and with no prefix characters before the start `[`. Content inside the included file will transform using DFM syntax.

```
[!include[<title>](<filepath>)]
```

Section definition

User may need to define section. Mostly used for code table. Give an example below.

```
> [!div class="tabbedCodeSnippets" data-resources="OutlookServices.Calendar"]
> ```cs
> <cs code text>
> ```
> ```javascript
> <js code text>
> ```
```

The above blockquote Markdown text will transform to section html as in the following:

```
<div class="tabbedCodeSnippets" data-resources="OutlookServices.Calendar">
  <pre><code>cs code text</code></pre>
  <pre><code>js code text</code></pre>
</div>
```

Code Snippet

Allows you to insert code with code language specified. The content of specified code path will expand.

```
[!code-<language>[<name>](<codepath><queryoption><queryoptionvalue> "<title>")]
```

- `<language>` can be made up of any number of character and '-'. However, the recommended value should follow Highlight.js language names and aliases.
- `<codepath>` is the path relative to the file containing this markdown content in file system, which indicates the code snippet file that you want to expand.
- `<queryoption>` and `<queryoptionvalue>` are used together to retrieve part of the code snippet file in the line range or tag name way. We have 2 query string options to represent these two ways:

	QUERY STRING USING #	QUERY STRING USING ?
1. line range	#L{startlinenumber}-L{endlinenumber}	?start={startlinenumber}&end={endlinenumber}
2. tagname	#{tagname}	?name={tagname}
3. multiple region range	*Unsupported*	?range={rangequerystring}
4. highlight lines	*Unsupported*	?highlight={rangequerystring}
5. dedent	*Unsupported*	?dedent={dedentlength}

- In `?` query string, the whole file will be included if none of the first three option is specified.
- If `dedent` isn't specified, the maximum common indent will be trimmed automatically.
- `<title>` can be omitted as it doesn't affect the DocFX markup result, but it can beautify the result of other Markdown engine, like GitHub Preview.

Code Snippet Sample

```
[!code-csharp[Main](Program.cs)]

[!code[Main](Program.cs#L12-L16 "This is source file")]
[!code-vb[Main](../Application/Program.vb#testsnippet "This is source file")]

[!code[Main](index.xml?start=5&end=9)]
[!code-javascript[Main](../jquery.js?name=testsnippet)]
[!code[Main](index.xml?range=2,5-7,9-) "This includes the lines 2, 5, 6, 7 and lines 9 to the last line"]
[!code[Main](index.xml?highlight=2,5-7,9-) "This includes the whole file with lines 2,5-7,9- highlighted"]
```

Tag Name Representation in Code Snippet Source File

DFM currently supports the following `<language>` values to be able to retrieve by tag name:

- C family
  - Start with: `// <{name}>`
  - End with: `// </{name}>`
  - Languages: actionscript, arduino, assembly (alias: nasm), c (alias: cpp, c++, objective-c, obj-c, objc, objectivec), csharp (alias: cs), cshtml, cuda, d (alias: dlang), fsharp (alias: fs), go (alias: golang), java, javascript (alias: js, node), pascal, php, processing, rust, scala, smalltalk, swift, typescript (alias: ts)
  - File extensions: .as, .asm, .ino, .c, .cc, .cpp, .cs, .cshtml .cu, .cuh, .d, .fs, .fsi, .fsx, .go, .h, .hpp, .java, .js, .pas, .php, .pde, .rs, .scala, .st, .swift, .ts

- Basic family
  - Start with: `' <{name}>`
  - End with: `' </{name}>`
  - Languages: vb, vbhtml, vbnet, vbscript
  - File extensions: .bas, .vb, .vba, .vbhtml, .vbs

- Markup language family
  - Start with: `<!-- <{name}> -->`
  - End with: `<!-- </{name}> -->`
  - Languages: cshtml, html, vbhtml, wsdl, xml, xsl, xslt, xsd, xaml
  - File extensions: .asp, .aspx, .csdl, .cshtml, .edmx, .jsp, .vbhtml, .wsdl, .xaml, .xml, .xsd, .xsl, .xslt, .html

- Sql family
  - Start with: `-- <{name}>`
  - End with: `-- </{name}>`
  - Languages: sql
  - File extensions: .sql

- Script family
  - Start with: `# <{name}>`
  - End with: `# </{name}>`
  - Languages: perl, powershell (alias: posh), python, r, ruby (alias: ru), shell (alias: sh, bash)
  - File extensions: .bash, .pl, .ps1, .py, .r, .ru, .ruby, .sh

- Special language
  - batchfile
    - Start with: `rem <{name}>`
    - End with: `rem </{name}>`
    - Languages: batchfile
    - File extensions: .bat .cmd

  - csharp
    - Start with: `#region {name}`
    - End with: `#endregion`
    - Languages: csharp (alias: cs)
    - File extensions: .cs .cshtml

  - erlang
    - Start with: `% <{name}>`
    - End with: `% </{name}>`
    - Languages: erlang
    - File extensions: .erl

  - haskell
    - Start with: `-- <{name}>`
    - End with: `-- </{name}>`
    - Languages: haskell
    - File extensions: .hs

  - matlab
    - Start with: `% <{name}>`
    - End with: `% </{name}>`
    - Languages: matlab
    - File extensions: .matlab

  - lisp
    - Start with: `; <{name}>`
    - End with: `; </{name}>`
    - Languages: lisp
    - File extensions: .lisp, .lsp

- lua
  - Start with: `-- <{name}>`
  - End with: `-- </{name}>`
  - Languages: lua
  - File extensions: .lua

- vb
  - Start with: `#Region {name}`
  - End with: `#End Region`
  - Languages: vb (alias: vbnet)
  - File extensions: .vb .vbhtml

> **NOTE**
>
> If dev-lang is not specified, file extension will be used to determine the language.

Code Snippet for Jupyter Notebooks

Allows you to insert code from a code cell of a Jupyter Notebook. The `source` content in the specified code cell will expand.

Steps to use this:

1. In your Jupyter Notebook, add metadata to the code cell you will reference:

```
"metadata": {
  "name": "{tagname}"
 }
```

2. In your .md file, use `name` to identify the cell.

```
[!notebook-<language>[](<codepath>?name={tagname})]]
```

Code Snippet for Jupyter Notebooks Sample

For this Jupyter Notebook cell:

```
{
 "cells": [
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {
    "name": "import"
   },
   "outputs": [],
   "source": [
    "import azureml.core\n",
    "print(azureml.core.VERSION)"
   ]
  },
```

Use the markup:

```
[!notebook-python[](<codepath>?name={import})]]
```

to display the lines of code in the `source` part of the cell:

```
import azureml.core
print(azureml.core.VERSION)
```

Note (Warning/Tip/Important)

Using specific syntax inside block quote to indicate the following content is Note.

```
> [!NOTE]
> <note content>
> [!WARNING]
> <warning content>
```

The above content will be transformed to the following html:

```
<div class="NOTE">
  <h5>NOTE</h5>
  <p>note content</p>
</div>
<div class="WARNING">
  <h5>WARNING</h5>
  <p>WARNING content</p>
</div>
```

Here are all the supported note types with the styling of the default theme applied:

> **NOTE**
>
> This is a note which needs your attention, but it's not super important.

> **TIP**
>
> This is a note which needs your attention, but it's not super important.

> **WARNING**
>
> This is a warning containing some important message.

> **IMPORTANT**
>
> This is a warning containing some important message.

> **CAUTION**
>
> This is a warning containing some important message.

Tabbed content

Syntax

- Start a tab by a special markdown title (any level).
  - Title content should be a markdown link.
  - Link target is `#tab/{tabid}` or `#tab/{tabid}/{condition}`
- Continue by any other content.
- End by a markdown hr.

Example

```
Tab group 1:

# [Tab Text 1](#tab/tabid-1)

Tab content-1-1.

# [Tab Text 2](#tab/tabid-2)

Tab content-2-1.

***

Tab group 2:

# [Tab Text A](#tab/tabid-a)

Tab content-a-1.

# [Tab Text B](#tab/tabid-b)

Tab content-b-1.

***

Tab group 3:

# [Tab Text 1](#tab/tabid-1)

Tab content-1-1.

# [Tab Text 2](#tab/tabid-2)

Tab content-2-1.

***

Tab group 4:

# [Tab Text A](#tab/tabid-a)

Tab content-a-2.

# [Tab Text B](#tab/tabid-b)

Tab content-b-2.

***
```

The result will be:

Tab group 1:

- Tab Text 1
- Tab Text 2

Tab content-1-1.

Tab group 2:

- Tab Text A
- Tab Text B

Tab content-a-1.

Tab group 3:

- Tab Text 1
- Tab Text 2

Tab content-1-1.

Tab group 4:

- Tab Text A
- Tab Text B

Tab content-a-2.

Behavior

Tab groups with a same set of id are linkable in one page.

In example, tab group 1, 3 have same id set: `tabid-1`, `tabid-2`, tab group 2, 4 have same id set: `tabid-a`, `tabid-b`.

So tab group 1, 3 are linked, tab group 2, 4 are linked. When tab `tabid-1` in tab group 1 is clicked, tab `tabid-1` in tab group 3 will be selected in same time. But tab group 2, 4 do not have any changed.

Condition

Condition is the tab id of other table groups.

e.g.:

```
Tab Group 1:

# [**Tab Text 1**](#tab/tabid-1)

Tab content-1.

# [**Tab Text 2**](#tab/tabid-2)

Tab content-2.

***

Tab Group 2:

# [Tab Text A](#tab/tabid-a/tabid-1)

Tab content-a for 1.

# [Tab Text A](#tab/tabid-a/tabid-2)

Tab content-a for 2.

# [Tab Text B](#tab/tabid-b/tabid-1)

Tab content-b for 1.

# [Tab Text B](#tab/tabid-b/tabid-2)

Tab content-b for 2.

***
```

Result:

Tab Group 1:

- **Tab Text 1**
- **Tab Text 2**

Tab content-1.

Tab Group 2:

- Tab Text A
- Tab Text B

Tab content-a for 1.

When select `tabid-1` in tab group 1, you can get content-a or content-b for 1 in group 2.\ When select `tabid-2` in tab group 1, you can get content-a or content-b for 2 in group 2.

Video

Allows you to add videos to your topics.

Syntax:

```
> [!Video embed_link]
```

> **NOTE**
>
> You must provide the **embed** uri of the video you wish to add to your topic.

Example:

```
> [!Video https://www.youtube.com/embed/TAaG0nUUy6A]
```

Result:

Differences introduced by DFM syntax

> **WARNING**
>
> DFM introduces more syntax to support more functionalities. When GFM does not support them, preview the Markdown file inside *GFM Preview* can lead to different results.

YAML header

In GFM, YAML header must start at the very beginning of the Markdown file. In DFM, YAML header contains more powerful meanings. Refer to Yaml Header for details.

```
...some text...

---
a: b
---
```

In GFM, it would be rendered as `<hr>a: b<hr>`.
In DFM, it would be rendered as a YAML header.

If you want to get `<hr>` in html in DFM, use:

```
- - -
***
* * *
```

or change content to make it not in YAML format:

```
---
a\: b
---
```

Text after block extension

Some block extension in DFM cannot be recognized in GFM. In GFM, it would be treated as a part of paragraph. Then, following content would be treated as a part of paragraph.

For example:

```
> [!NOTE]
>     This is code.
```

In GFM, it will be rendered as a paragraph with content `[!NOTE] This is code.` in blockquote.
In DFM, it will be rendered as a code in note.

Support

Overview

Build succeeded with warning.
[21-04-30 05:51:04.012]Warning:t detected for extracting metadata.
[21-04-30 05:51:04.032]Warnible to find either toc.yml or toc.md inside ~/articles/. Make sure the file is included in config file docfx.json!
[21-04-30 05:51:04.052]Warniable to find either toc.yml or toc.md inside ~/articles/support/. Make sure the file is included in config file docfx.json!

Support

Contact

DocFX is an API documentation generator for .NET, which currently supports C#, VB and F#. It generates API reference documentation from triple-slash comments in your source code. It also allows you to use Markdown files to create additional topics such as tutorials and how-tos, and to customize the generated reference documentation. DocFX builds a static HTML website from your source code and Markdown files, which can be easily hosted on any web server (for example, github.io). Also, DocFX provides you the flexibility to customize the layout and style of your website through templates.

	TELEPHONE	FAX
Gloucester	07790 928864	01454 299978
Warrington	07790 299978	01454 928864

IMPORTANT

Essential information required for user success

TIP

Essential information required for user success

Downloads

Documentation PDF file

Download documentation PDF file

Something else