

DocGen - a documentation tool

Owner: Mark Macdonald	Version: 2.0.0	Released: XX/XX/XXXX
Author: Mark Macdonald	Contributors:	
Module:	ID:	Link: Github

Summary

DocGen is a documentation tool for software products. It takes plain text or CommonMark (Markdown) as input, and generates both a static website and a PDF copy.

© 2015 [Mark Macdonald](#)

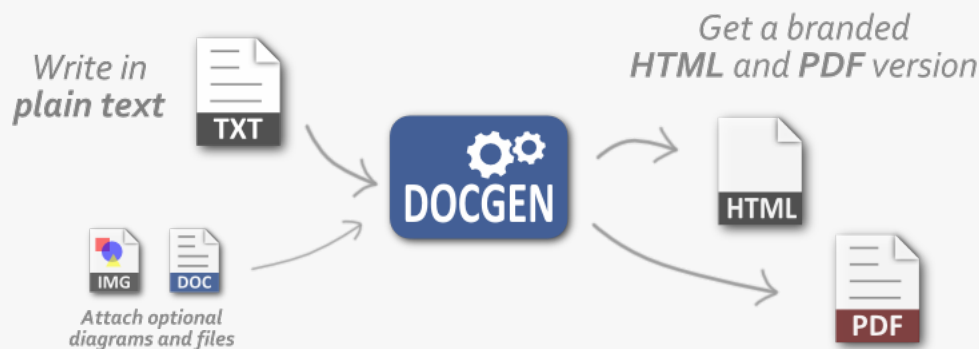
MIT License. Created by DocGen 2.0.0 on 30/03/2015 at 14:32:44.

Table of Contents

Table of Contents	2
Overview	4
Features	4
How it works	4
Browser support	5
Quick start	5
Installation	6
Supported platform	6
Dependencies	6
Optional dependencies (only for PDF)	6
Quick install with NPM	6
Install by direct download	6
Upgrading	7
DocGen 1.0.* to DocGen 2.0.0	7
Running DocGen	8
Overview	8
Command-line usage	8
Scaffold command	8
Run command	8
Writing content	9
Overview	9
Metadata	9
Plain text	9
CommonMark (Markdown)	10
HTML	10
Embedding images	10
Attaching files	10
Mathematical Expressions	10
Mathematics using KaTeX	11
Mathematics using MathJax	11
CommonMark reference	12
Advanced content	13
Advanced styles	13
Basic tables	13
HTML tables	13
Creating internal links to page sections	13
Creating internal links to other pages	13
Control of page breaks in the PDF	14
Troubleshooting	15
Displaying detailed errors	15
PDF missing content	15
PDF display errors	15
Attached files not in PDF	15

Corrupted text characters	15
Missing logo	15
Other issues	15
Using with version control	16
Recommended practice	16
Release notes	17
DocGen 2.0.0 released XX/XX/XXXX	17
DocGen 1.0.1 released 18/01/2012	17
DocGen 1.0.0 released 04/11/2011	17

Better documentation for software products.



[Download](#)

[Report Issues](#)

DocGen is a **static website generator** that's ideal for making technical user guides for software products.

Self-contained website

Creates a static website that works on any server, or as local files (CD, shared drive etc).

Table of contents

Automatically creates tables of contents, with links and PDF page numbers.

Optional PDF

Also publishes the website content as a single PDF, using [wkhtmltopdf](#).

Code syntax highlighting

Automatically highlights code blocks, using [Highlight.js](#), with language detection.

Human-friendly input

Write in plain text, or the human-friendly [CommonMark](#) format (Markdown).

Mathematical expressions

Displays mathematical expressions without plugins, using either [KaTeX](#) or [MathJax](#).

Easy to version control

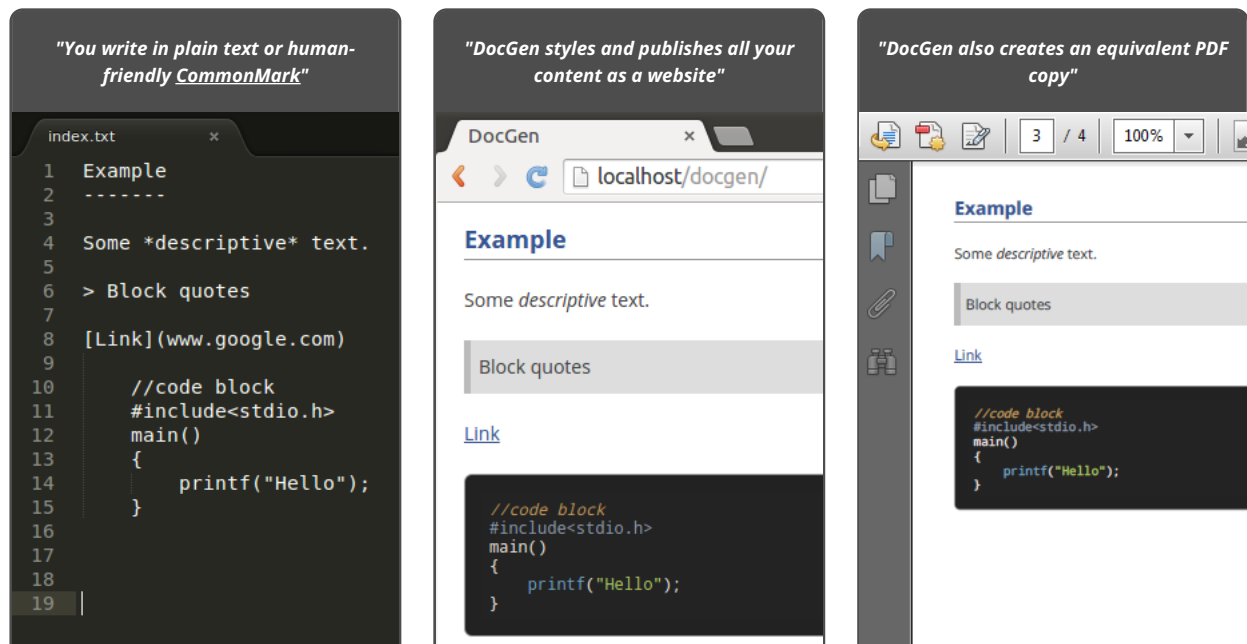
Easily version control software documentation in sync with its product.

Branding and metadata

Easily brand with a logo, attribute ownership, and attach release notes.

DocGen makes it easy for anyone to create high-quality technical documentation.

How it works



Flexible Input Formats

- Plain text
- CommonMark (Markdown)
- HTML
- LaTeX mathematical expressions
- Image diagrams
- Attach other documents

Configurable Metadata

- Branding (logo, title, organization)
- License, copyright, and legal markings
- Ownership and attribution
- Version information
- Release notes (change log)

DocGen is probably not the right tool for precision PDF layout. Or for product configurators which need to output variants based on a common template. DocGen is intended more for free-form, human generated content which is regularly updated and refactored, and standardised for each product release.

Browser support

Output created by DocGen works on modern browsers (desktop and mobile devices).

- tested in [Internet Explorer 9+](#), [Chrome](#), [Firefox](#), [Safari](#)

Quick start

In just three commands:

- Install DocGen
- Scaffold an empty template
- Generate a static website

Enter these commands in the terminal:

```
npm install -g docgen-tool
docgen scaffold
docgen run -o $HOME/docgen-example
```

See the [installation guide](#) for more detailed instructions.

Installation

[Supported platform](#)
[Dependencies](#)
[Optional dependencies \(only for PDF\)](#)
[Quick install with NPM](#)
[Install by direct download](#)

This section explains how to install DocGen.

Supported platform

'Supported platform' means the software required to **run** the DocGen tool. The static website **produced** by DocGen will work on all modern browsers on all all major operating systems (see [browser support](#)).

The supported platform for this version of DocGen is: **Ubuntu 14.04** with **Node.js 0.12.0** and **wkhtmltopdf 0.12.2.1** (with patched qt).

While other operating systems and dependency versions may work, they are not tested or officially supported.

Dependencies

DocGen requires [Node.js](#). To install on Ubuntu Linux, enter these terminal commands:

```
curl -sL https://deb.nodesource.com/setup_0.12 | sudo bash -  
sudo apt-get install -y nodejs
```

This method uses the [NodeSource Linux Repositories](#) (recommended).

For other platforms (not supported), see the [download page](#) or vendor instructions.

Optional dependencies (only for PDF)

For optional PDF support, DocGen requires [wkhtmltopdf](#). See the wkhtmltopdf website for installation instructions.

Quick install with NPM

The quickest way to install DocGen is with [npm](#) (the JavaScript package manager). Enter these terminal commands:

```
npm install -g docgen-tool
```

Install by direct download

DocGen can also be installed by [direct download](#).

Upgrading

[DocGen 1.0.* to DocGen 2.0.0](#)

This section explains how to upgrade from old versions of DocGen.

DocGen 1.0.* to DocGen 2.0.0

DocGen 2 is not backwards compatible with DocGen 1, but the upgrade is easy with these instructions:

- Install DocGen 2 by following the [installation guide](#)
- Create a replacement 'source' directory, e.g.:

```
docgen scaffold -o $HOME/source
```
- Migrate the source files
 - Copy the source (.txt) files from the old 'Src' directory to the new 'source' directory
 - Remove the top-level page headings in each source file (DocGen now inserts these automatically, based on contents.json)
 - Rename '*change-log.txt*' to '*release-notes.txt*'
- Migrate the attached content
 - Place a logo with filename 'logo.png' in 'files/images' directory (the logo is now part of the source)
 - Copy the images from '*Images*' to '*files/images*'
 - Copy attached files from '*Files*' to '*files*'
- Migrate the metadata
 - Edit '*parameters.json*' with the relevant values from '*doc-parameters.yml*'
 - Edit '*contents.json*' with the relevant values from '*table-of-contents.yml*'
- Run DocGen:

```
docgen -i $HOME/source -o $HOME/output
```
- Check the styling in the latest version still works well with the old content
- Note that the PDF is no longer generated by default (pass the '-p' option)

DocGen 1 tool-behaviours.yml has been replaced by command-line options See

```
docgen run -h
```

 for help.

Running DocGen

[Overview](#)
[Command-line usage](#)
[Scaffold command](#)
[Run command](#)

Overview

DocGen is a command-line tool which takes plain text input files and outputs a static website.

Command-line usage

The DocGen command-line interface includes usage help for both the tool and its subcommands:

```
docgen --help
docgen run --help
```

Scaffold command

The **scaffold** command creates an *example* input directory. It outputs the minimum files required by DocGen, which can then be used as a template for making any new website.

Create a scaffold template in the working directory (./):

```
docgen scaffold
```

Create a scaffold template in a specified directory:

```
docgen scaffold -o $HOME/docgen-example
```

Run command

The **run** command transforms an input directory (plain text source) into an output directory (HTML+PDF).

Basic usage:

```
docgen run -i $HOME/docgen-example -o $HOME/docgen-output
```

Optionally create a PDF:

```
docgen run -i $HOME/docgen-example -o $HOME/docgen-output -p
```

Optionally create a redirect page:

```
docgen run -i $HOME/docgen-example -o $HOME/docgen-output -r
```

The optional redirect page is an 'index.html' file that is placed in the output's parent directory. The redirect page redirects the user to the homepage in the output directory. This is mostly useful for hosting the website without having to place all the files in the root directory.

Writing content

[Overview](#)
[Metadata](#)
[Plain text](#)
[CommonMark \(Markdown\)](#)
[HTML](#)
[Embedding images](#)
[Attaching files](#)
[Mathematical Expressions](#)
[Mathematics using KaTeX](#)
[Mathematics using MathJax](#)

Content for a DocGen website is authored in plain text, using any text editor. Image files can be embedded, and other types of files can be attached as links. Additionally, website metadata is configured via [JSON](#) files.

Overview

DocGen transforms source files from an input directory into output files in an output directory.

It takes every source file (plain text) specified in **contents.json** and converts it. Each source file becomes a separate page in the website and a separate chapter in the PDF.

DocGen adds metadata that is specified in **parameters.json**, and copies the images and files in the **files** directory to the output.

Always save input files with **UTF-8** encoding. This makes non-standard characters (ø © é etc) work.

Metadata

parameters.json

The parameters file is used to specify metadata describing the product.

- **title** - the product title
- **name** - the product name (also used to name the PDF)
- **version** - the release version
- **date** - the release date
- **organization** - the company or organization
- **author** - the lead author of the document
- **owner** - the owner of the product
- **contributors** - list of contributors
- **website** - a link to the product website
- **module** - module name (useful for larger products with submodules)
- **id** - reference number (e.g. product id in a change management tool)
- **summary** - a descriptive summary of the product
- **marking** - license or other protective markings
- **legalese** - document markings (confidentiality, disclaimers, smallprint etc)

Values can be empty strings, but the elements are required in the JSON file.

Parameters with URLs can be either website URLs, or email addresses (specify *'mailto:name@address.com'*).

contents.json

The contents file specifies the names, locations, order, and hierarchy of the source files. It is used to generate both the web and PDF table of contents.

release-notes.txt

The release notes source file is a mandatory source file (that does not need to be listed in contents.json). Use it to summarize the change history for each version of the product.

Plain text

The simplest input format is just to write in plain text. Here is an example of the source and output:

Example paragraph.

Example paragraph.

CommonMark (Markdown)

[CommonMark](#) (also known as [Markdown](#)) is a human-friendly plain text markup format. The source format is easy to read and write, and the CommonMark parser translates it into HTML. DocGen uses the [markdown-it](#) implementation of CommonMark. Here is an example of the source and output:

CommonMark Example

Paragraphs are text blocks separated by new lines.
Text can be styled: **emphasised** and ****strong****.
Here is an [\[example link\]\(http://www.google.com\)](#).

To make a code block, just indent with a tab
"Hello World" in Ruby:
5.times { puts "Hello!" }

CommonMark Example

Paragraphs are text blocks separated by new lines.
Text can be styled: *emphasised* and **strong**.
Here is an [example link](#).

```
# To make a code block, just indent with a tab
# "Hello World" in Ruby:
5.times { puts "Hello!" }
```

For more examples, see the [CommonMark reference](#).

HTML

For more complex pages not covered by CommonMark's syntax, simply use inline HTML:

```
<table>
  <tr>
    <td>Foo</td>
    <td>Bar</td>
    <td>Baz</td>
  </tr>
</table>
```

Foo

Bar

Baz

For more examples, see [writing advanced content](#).

Inline HTML is still parsed by the CommonMark parser (HTML is allowed in CommonMark documents). In DocGen, it is also possible to bypass the CommonMark parser altogether and specify a pure HTML input page, by setting `"html": true` in a page object in `contents.json`.

Embedding images

Diagrams (in image form, e.g. JPEG, PNG, GIF etc) should be put the `files/images` directory, and embedded as images.



Attaching files

Other files you want to attach should go into `files` directory.

[attachment](user_guide.pdf)

[attachment](#)

Mathematical Expressions

[LaTeX](#) is the most common markup format for mathematical expressions.

Modern web browsers do not yet consistently support a common standard for authoring mathematical expressions. For this reason, extra libraries are needed. DocGen supports two widely-used web mathematics libraries out-the-box:

- [KaTeX](#) is fast, lightweight, and [supports fewer features](#)
- [MathJax](#) is slower, larger, and [supports more features](#)

The document author decides which one (or both) to use.

Mathematics using KaTeX

KaTeX is the recommended choice. It is bundled with DocGen but must be enabled by passing the **-m** option.

```
<div class="katex-math">
f(x) = \int_{-\infty}^{\infty} \hat{f}(\xi) e^{2\pi i \xi x} d\xi
</div>
```

$$f(x) = \int_{-\infty}^{\infty} \hat{f}(\xi) e^{2\pi i \xi x} d\xi$$

Mathematics using MathJax

MathJax is the fallback choice for expressions not yet supported by KaTeX. When required, MathJax can be enabled by passing the **-n** option.

DocGen uses the same MathJax configuration as the popular [Stack Exchange](#) websites.

Because MathJax is a large library, it is not bundled with DocGen, but is served from a third-party CDN (content delivery network). This means users need an active Internet connection for the MathJax feature to work.


MathJax can be slow to render. When used with the PDF feature, it may be necessary to allow more rendering time by passing the **-d [milliseconds]** option.

```
$$
f(n) = \begin{cases}
n/2, & \text{if } n \text{ is even} \\
3n+1, & \text{if } n \text{ is odd}
\end{cases}
$$
```

$$f(n) = \begin{cases} n/2, & \text{if } n \text{ is even} \\ 3n+1, & \text{if } n \text{ is odd} \end{cases}$$

CommonMark reference

This page shows examples of how to write content with CommonMark. See the [CommonMark Spec](#) for more detailed information.

Element	What you type	What it looks like in DocGen						
Page Heading	<code>Page Heading</code> =====	Page Heading						
Section Heading	<code>Section Heading</code> -----	Section Heading						
Minor Heading	<code>### Minor Heading</code>	Minor Heading						
Fake Heading (not in PDF contents list)	<code><p class="dg-fakeHeading">Fake</p></code>	Fake						
Emphasis (italic)	<code>This text has *emphasis*</code>	This text has <i>emphasis</i>						
Strong (bold)	<code>This text is **bold**</code>	This text is bold						
Block quotes	<code>> This is a block quote.</code>	This is a block quote						
Code block (indent with tab)	<code>int some_code=0</code>	<code>int some_code=0;</code>						
Unordered List	<code>* unordered list</code> <code>* (items)</code>	<ul style="list-style-type: none"> unordered list (items) 						
Ordered List	<code>1. ordered list</code> <code>2. (items)</code>	<ol style="list-style-type: none"> ordered list (items) 						
External Links	<code>[link](http://www.google.com)</code>	link						
Links to a local file	<code>[attachment](user_guide.pdf)</code>	attachment						
Diagrams (embedding images)	<code></code>							
Simple table	<code> Cell 1 Cell 2 Cell 3 </code> <code> Cell 4 Cell 5 Cell 6 </code>	<table> <tr> <td>Cell 1</td><td>Cell 2</td><td>Cell 3</td></tr> <tr> <td>Cell 4</td><td>Cell 5</td><td>Cell 6</td></tr> </table>	Cell 1	Cell 2	Cell 3	Cell 4	Cell 5	Cell 6
Cell 1	Cell 2	Cell 3						
Cell 4	Cell 5	Cell 6						

Advanced content

[Advanced styles](#)
[Basic tables](#)
[HTML tables](#)
[Creating internal links to page sections](#)
[Creating internal links to other pages](#)
[Control of page breaks in the PDF](#)

Advanced styles

DocGen is built on the [Webknife CSS framework](#). Any of the HTML elements and styles from Webknife can be used in a DocGen source file. For details, see the [Webknife reference guide](#).

Basic tables

Basic tables can be inserted with the Github-flavoured Markdown [table extension](#).

```
| Heading | Heading | Heading |
|:-----:|:-----:|:-----:|
| Something | Something | ? |
| Another | Another | ? |
| One more | One more | ? |
```

Heading	Heading	Heading
Something	Something	?
Another	Another	?
One more	One more	?

HTML tables

Regular HTML can also be used for tables, allowing full custom styling (including table and column widths).

```
<table style="width:100%;">
<tr>
<th>Heading</th>
<th>Heading</th>
<th>Heading</th>
</tr>
<tr>
<td class="bold">Example</td>
<td>Example</td>
<td>Example</td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td class="w-success-text">Example</td>
<td><span class="w-icon" data-name="checkmark"></span></td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td class="w-error-text">Example</td>
<td><span class="w-icon" data-name="close"></span></td>
</tr>
</table>
```

Heading	Heading	Heading
Example	Example	Example
Example	Example	✓
Example	Example	✗

Creating internal links to page sections

To create links to other sections within one content page, put hyphens between the words in the heading and prepend with #:

```
[link to heading](#this-is-a-heading)

... other page content here ...

This is a heading
-----
```

Creating internal links to other pages

To create links to *other* content pages provide the relative url to the page:

```
[link to heading](example-page.html)
```

Control of page breaks in the PDF

DocGen does not provide precision control over PDF layout. However, some steps can be taken in case of page break issues (the most common problem).

To *force* a page break, insert the following before an element that should appear on a new page:

```
<span class="force-break"></span>
```

DocGen automatically tries to eliminate page breaks *inside* code blocks, block quotes, and table rows. To apply the same technique to other elements, revert to HTML and apply the **avoid-break** class. For example:

```
<p class="avoid-break">A long paragraph</p>
```

Troubleshooting

[Displaying detailed errors](#)
[PDF missing content](#)
[PDF display errors](#)
[Attached files not in PDF](#)
[Corrupted text characters](#)
[Missing logo](#)
[Other issues](#)

This page gives help on solving common issues with DocGen.

Displaying detailed errors

Pass the `-v` (verbose) option when running DocGen to get more detailed error messages.

PDF missing content

In complex pages, the PDF generator (wkhtmltopdf) needs to be given enough time for dynamic content to be rendered. Pass the `-d [milliseconds]` option to increase the rendering time for each page, if required.

PDF display errors

There are known text display issues in Firefox PDF viewer. Try an [alternative viewer](#).

Attached files not in PDF

Attached files are not converted to PDF, only the web content is.

Corrupted text characters

Make sure all the input text files are saved with UTF-8 encoding.

Missing logo

The logo must be PNG format and saved with the path `files/images/logo.png`.

Other issues

For any other problems, please submit a [an issue ticket](#).

Using with version control

[Recommended practice](#)

One of the benefits of using DocGen for product software documentation is that its plain text source files are easy to version control in sync with the product.

Recommended practice

It is recommended to store the documentation **source files** (DocGen input directory) in the same version control repository as the product code. This allows each release of the product to have a matching version of its documentation.

It is not necessary to version control the DocGen output, because this can always be regenerated. If the DocGen output is version controlled, file renames, additions, and deletions have to be performed manually in the version control tool.

Release notes

[DocGen 2.0.0 released XX/XX/XXXX](#)

[DocGen 1.0.1 released 18/01/2012](#)

[DocGen 1.0.0 released 04/11/2011](#)

DocGen 2.0.0 released XX/XX/XXXX

- DocGen is now open source
- Rewritten in JavaScript for Node.js
- Much easier to install (hosted on npm)
- Dependencies are now version controlled (using npm)
- Modernized visual style (uses Webknife CSS framework)
- Input metadata files are now in JSON rather than YAML format
- Top-level page headings are now inserted automatically (from contents.json)
- The web and PDF tables of contents both correspond to contents.json
- Command-line options are now used for configuration, rather than a config file
- Command-line output is now color coded (green success, red error)
- Added usage information to the command line interface
- Generating the PDF is now an optional feature
- Upgraded to the latest version of the PDF generator (wkhtmltopdf)
- Added support for mathematical expressions (with KaTeX or MathJax)
- Added support for a list of document contributors (for multiple authors)
- Added time to the generation timestamp
- Renamed 'change log' to 'release notes'
- Fixed issues with fonts and text kerning in the PDF copy
- Fixed defect where unexpected text appeared on some pages with a page table of contents
- Dropped support for Internet Explorer 7 and 8
- Dropped formal support for tool to run on multiple operating systems
- Removed support for 'Mark of the Web'

DocGen 1.0.1 released 18/01/2012

- Fixed a bug causing the table of contents headings to sometimes appear in the wrong order

DocGen 1.0.0 released 04/11/2011

- Ruby implementation (not released as open source)
- Creates a static website from Markdown input files
- Also creates a PDF copy using wkhtmltopdf