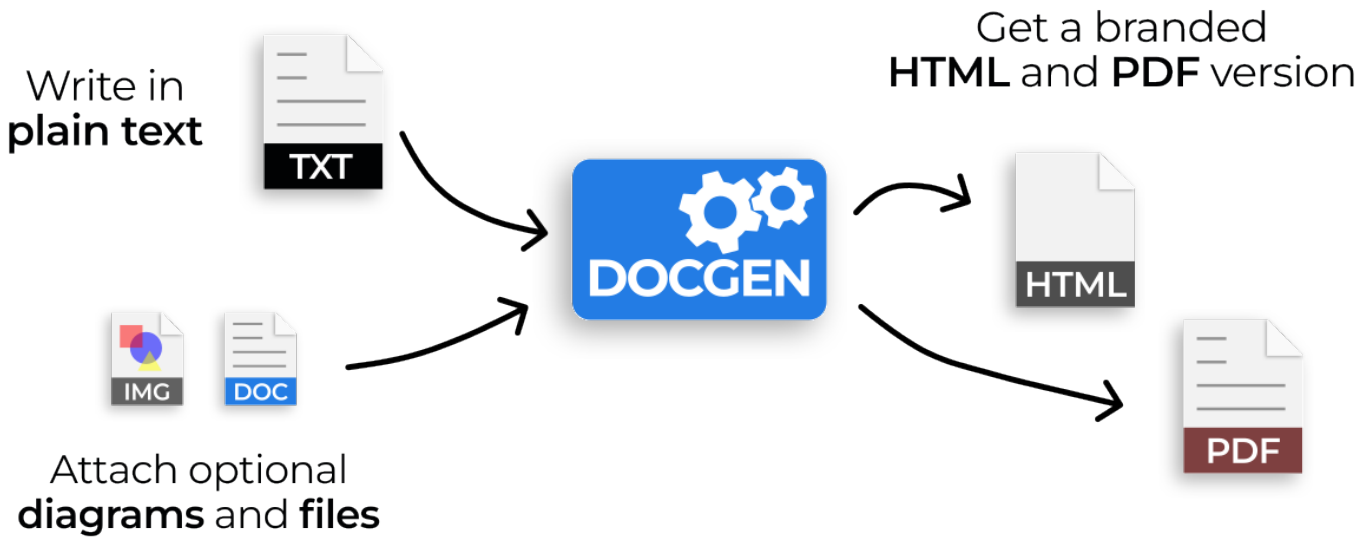
The Leading Open Source Documentation Tool

DocGen generates HTML websites and PDF documents from plain text for free.



Download Report Issues

DocGen is a Static Website Generator

DocGen is an open-source website and PDF generator that makes it easy to create high-quality documentation.

Features

Self-contained website

Creates a static website that works on any server, or as local files.

Optional PDF

Also publishes the website content as a single PDF, using React-pdf.

Human-friendly input

Write in plain text, or the human-friendly Markdown format.

Easy to version control

Plain text input formats work well with all version control systems.

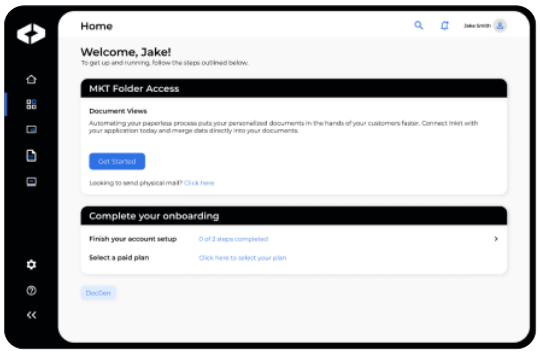
Table of contents

Automatically creates tables of contents

Branding and metadata

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

DocGen is sponsored by Inkit

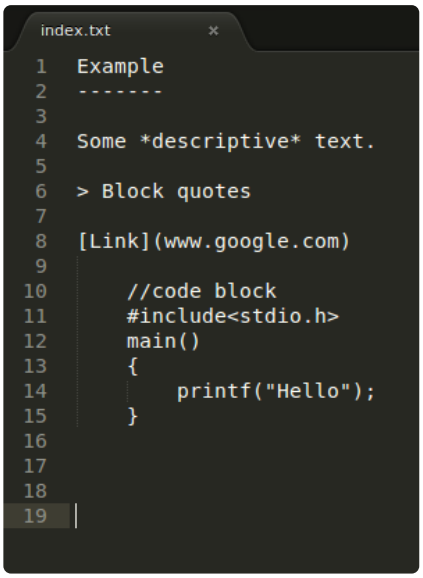


DocGen is open-source software sponsored by Inkit, the leading Zero Trust Document Generation Platform.

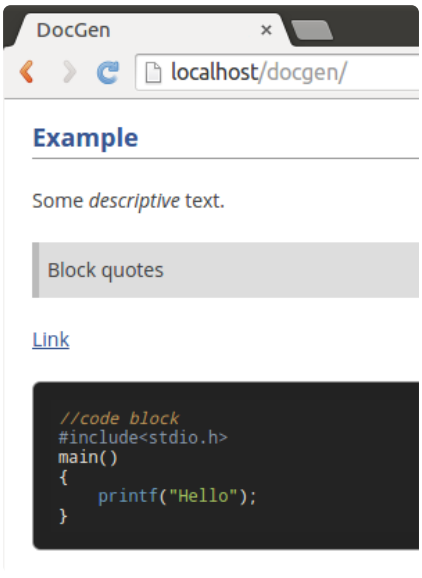
Learn More

How it works

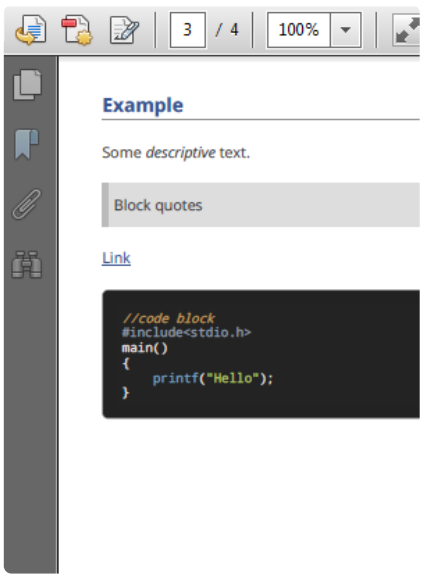
Simply install DocGen, and run the tool to generate websites and PDF documents.



001 | **Create content in plain text or human-friendly Markdown**



002 | **DocGen styles and publishes all your content as a website**



003 | **DocGen also creates an equivalent PDF copy**

Flexible Input Formats

- Plain text
- CommonMark (Markdown)
- Image diagrams

Configurable Metadata

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

> *NOTE: DocGen is intended for free-form, human-generated content which is regularly updated and improved, then automatically laid out according to a template. It is not intended as a precision PDF editing tool.*

Browser Support

Websites and documents generated by DocGen work in most browsers including Chrome, Edge, Firefox and Safari.

Quick Start

In just three steps:

1. **Install DocGen**
2. **Scaffold an empty template**
3. **Generate a static website and PDF**

Simply enter these commands in the terminal:

```
npx docgen-tool scaffold
npx docgen-tool run -o $HOME/docgen-example
```

See the installation guide for more detailed instructions.

This section explains how to install DocGen.

DocGen is a command-line (CLI) tool. It runs on most major operating systems.

> *NOTE - the websites and PDF documents generated by DocGen can be used on all major operating systems and browsers without any setup or installation. These setup instructions are needed only when running the tool to generate documents.*

Install Dependencies

Before installing the tool, you need some dependencies.

Node.js and NPM

DocGen needs Node.js (JavaScript engine) and its package manager NPM which are downloaded and installed together.

See Node.js downloads and choose the installer or package for you operating system.

- **Windows:** use the official installer or nvm-windows
- **OS X:** use the official installer or nvm
- **Ubuntu Linux:** use nvm or see other instructions

Run directly with NPX

The easiest way to run DocGen is with NPX (Node Package Runner).

```
npx docgen-tool --help
```

Install with NPM

Alternatively, you can install DocGen globally with NPM (Node Package Manager).

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

Direct Download

You can download the DocGen source code directly from Github.

Overview

DocGen is a command-line (CLI) tool which takes plain text or markdown input files and outputs a static website. It also optionally outputs a PDF copy of the website content.

DocGen works by transforming an input directory (source files) into an output directory (website + PDF).

Command-line usage

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

```
npx docgen-tool --help
```

Scaffold command

Use the scaffold command for creating a new project. It creates an *example* input directory, by generating the minimum the skeleton input files required by DocGen. After generating them, you can customise them to create your website.

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

```
npx docgen-tool scaffold
```

Create a scaffold template in a specified directory:

```
npx docgen-tool 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:

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

Optionally create a PDF:

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

Optionally create a redirect page:

```
npx docgen-tool 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.*

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.

Metadata

parameters.json

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

- **title** - the website title
- **name** - the website 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 document
- **contributors** - list of contributors
- **website** - a link to the parent website
- **backlink** - a link back to another site (useful for integrated documentation)
- **module** - module name (useful for larger sites with submodules)
- **id** - reference number (e.g. id in a change management tool)
- **summary** - a descriptive summary of the website/document
- **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.md

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.

Content for a DocGen website is authored in a human-friendly plain text format called Markdown.

Content files

DocGen content files are stored in Markdown files with the extension .md. These can be edited with any text editor.

Each Markdown file is converted to a page in the website and one or more pages in the PDF. You can create as many input files as you need.

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

How to write Markdown files

Markdown is a simple markup language that uses plain text formatting syntax. It is designed to be easy to read and write. DocGen converts the Markdown content into an HTML website and a PDF copy.

This section shows some examples of what to type, and how it looks in the page.

Paragraphs

What you type:

```
This is a paragraph. Paragraphs are text blocks separated by new lines.

This is another paragraph. Text can be styled: *emphasised* and **strong**.
```

How it looks:

This is a paragraph. Paragraphs are text blocks separated by new lines.

This is another paragraph. Text can be styled: *emphasised* and **strong**.

Headings

What you type:

```
# Heading level 1

## Heading level 2

### Heading level 3
```

How it looks:

Heading level 1

Heading level 2

Heading level 3

Links

What you type:

```
This is a link to [Google](http://www.google.com).
```

How it looks:

This is a link to Google.

Bullet-points

What you type:

```
- one
- two
- three

1. one
2. two
3. three
```

How it looks:

- one
- two
- three

1. one
2. two
3. three

Quotes

What you type:

```
> This is a quote.
```

How it looks:

> *This is a quote.*

Admonitions

What you type:

```
!!! Information Title
 Information details

!!! Warning Title
 Warning details

!!! Success Title
 Success details

!!! Error Title
 Error details
```

How it looks:

Title

Information details

Title

Warning details

Title

Success details

Title

Error details

Code blocks

What you type:

```
// To make a code block, just indent with a tab

const hello = () => {
 console.log("Hello, World!");
};
hello();
```

How it looks:

```
// To make a code block, just indent with a tab

const hello = () => {
 console.log("Hello, World!");
};
hello();
```

Images

Images can be added to documents - save the image files to the *files/images* directory (in .jpg or .png formats).

What you type:

```
![logo](files/images/example-image.png)
```

How it looks:

For more examples, see the CommonMark reference.

One of the benefits of using DocGen for product software documentation is that its plain text source files are easy to version control using any modern version control tool such as git.

Recommended practice

It is recommended to store the documentation **source files** (DocGen input directory) in the same version control repository as its parent project. For example, if you are using DocGen to document a software product, each release of the software app can then have a matching documentation version.

> *It is not necessary to version control the DocGen output, because this can always be regenerated.*

This page explains some technical details about how DocGen works.

> *You don't need to read these details to use DocGen (they explain how to the tool works internally for advanced users).*

CLI

DocGen is a CLI (command-line interface) tool that is written TypeScript for Node.js

Markdown

DocGen uses the markdown-it Markdown parser with CommonMark spec for parsing Markdown content files.

Website

The website is generated using React as template engine.

Styles

DocGen uses Style Dictionary for defining design tokens and generating styles.

The website styles are defined in Sass.

The PDF styles are defined in CSS-in-JS format. The React PDF engine uses Yoga Layout, which supports a print-friendly subset of modern CSS.

PDF

DocGen uses ReactPDF as its engine for generating PDF documents, which is built on PDFKit.

Additionally, DocGen uses the react-pdf-html package for converting Markdown HTML output into React PDF primitives (it internally parses HTML with node-html-parser and CSSTree).

This section 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.

Corrupted text characters

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

Missing logo

The logo must be in PNG format and saved with the path *files/images/logo.png*. It must have suitable dimensions for the header (height and width).

Other issues

For any other problems, please submit an issue ticket.