GitSite's Guide

How to use GitSite.

Crypto Michael

https://gitsite.org/books/gitsite-guide/

2025-05-25

Index

GitSite Introduction

GitSite is a document platform based on modern Web technology, any individual and team can organize and write documents through Git, and automatically generate HTML5-compliant web sites.

Git is currently the most popular version control system, and hosting platforms such as GitHub provide powerful and reliable hosting of Git repositories. Many individuals and teams manage Markdown documents through Git on platforms like GitHub. GitSite does not require the use of specialized front-end build tools, nor does it require cumbersome configuration to quickly generate a web site based on Markdown documents from Git repositories, and all contents on the site is static HTML pages.

Design Goals

GitSite is designed to support Git repositories with no more than 10,000 Markdown documents, usually product documentation, knowledge bases, personal blogs, or a book. If the number of documents is very large, you may want to consider using database management, as GitSite's static page-based approach can slow down front-end searches significantly.

Install and Deploy

Install GitSite command line tool

Use `npm` to install GitSite command line tool:

```
$ npm install -g gitsite-cli
```

Use the following command to update the latest version:

```
$ npm update -g gitsite-cli
```

Setup a new site

First create an empty directory which will be used as your site's root directory and also a git repo. Let's name it as `awesome` :

```
$ mkdir awesome
```

Run `gitsite-cli init` under the `awesome` directory:

```
$ cd awesome
$ gitsite-cli init
```

The GitSite command line tool does the following job to initialize your new site:

1. Check if current directory is an empty directory;
2. Download sample site from GitHub;
3. Unzip the sample site.

Now you can find the following files and directories under your `awesome` directory:

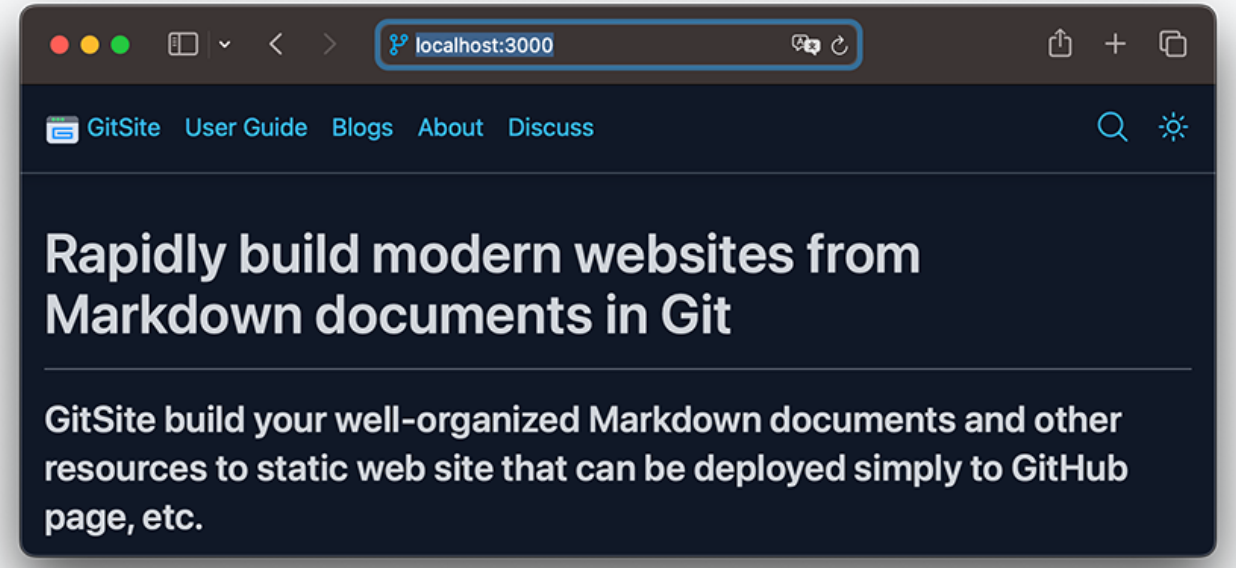
```
awesome/
├── themes/       ←── all themes
│   └── default/  ←── a theme named 'default'
│
└── source/       ←── contains all markdown docs
    │
    ├── books/            ←── all books
    │   ├── user-guide/   ←── a book
    │   │   ├── book.yml  ←── book name, author and description
    │   │   ├── 10-introduction/  ←── chapter order and short name
    │   │   │   ├── README.md     ←── chapter content
    │   │   │   └── test.png      ←── static resources used in the chapter
    │   │   ├── 20-installation/  ←── chapter
    │   │   │   ├── README.md
    │   │   │   ├── 10-create-repo/  ←── sub chapter
    │   │   │   │   └── README.md
    │   │   │   ├── 20-workflow/     ←── sub chapter
    │   │   │   │   └── README.md
    │   │   │   └── 30-deploy/       ←── sub chapter
    │   │   │       └── README.md
    │   │   └── ...  ←── more chapters
    │   └── ...  ←── more books
    │
    ├── blogs/     ←── all blogs
    │   ├── tech/  ←── tag
    │   │   ├── 2024-01-01-hello/  ←── blog date and short name
    │   │   │   ├── README.md      ←── blog content
    │   │   │   └── hello.jpg      ←── static resources used in the blog
    │   │   └── ...                ←── more blogs
    │   └── ... ←── more tags
    │
    ├── pages/             ←── all pages
    │   ├── license/       ←── about page
    │   │   └── README.md  ←── page content
    │   └── ...            ←── more pages
    │
    ├── 404.md       ←── display as 404 page if not found
    ├── README.md    ←── display as home page
    ├── favicon.ico  ←── favorite icon
    ├── site.yml     ←── site config
    └── static/      ←── static resources
        ├── custom.css
        ├── logo.png
        └── ...
```

Preview site on local

Run `gitesite-cli serve` to start a local HTTP server to serve the site:

```
$ gitsite-cli serve
```

Then you can visit your site on http://localhost:3000:



Update site settings

The site settings are stored in `source/site.yml`. You should update the settings:

- Set your site's name, description, etc;
- Set your site's navigation menus;
- Set your Disqus, Google analytics ID, or just remove it.

Deploy to GitHub page

To deploy site to GitHub page, first create a repo on GitHub and push your local files to the remote.

To enable GitHub page, go to repo - Settings - Pages - Build and deployment: select `GitHub Actions`.

Make a new push to trigger the Action for deployment.

The workflow script file is `.github/workflows/gitsite.yml`. Check the sample gitsite.yml.

You must set the `root-path: /<rootPath>` under which your site is served for GitHub pages deployment without custom domain, it is often `/<projectName>` :

```
site:
  # set when your site is served for GitHub pages without custom domain:
  root-path: /gitsite
```

Check the sample site.yml.

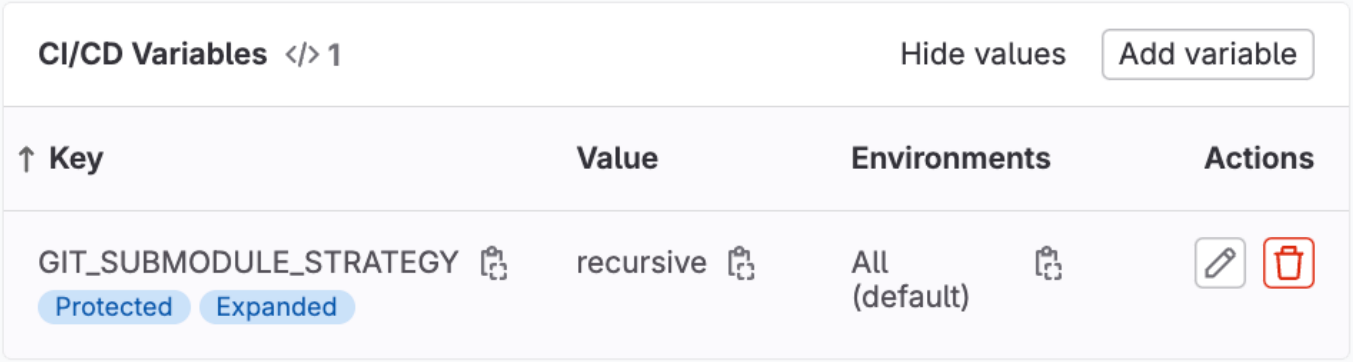
Deploy to GitLab page

It is similar to deploy site to GitLab, and GitLab requires a `.gitlab-ci.yml` script.

Please make sure the submodule function is enabled by: Project - Settings - CI/CD - Variables - Add variable:

Key: `GIT_SUBMODULE_STRATEGY`

Value: `recursive`



Check the sample .gitlab-ci.yml.

Deploy to CloudFlare page

To deploy site to CloudFlare page, create application from GitHub repo, then open application settings - `Builds & deployments` - `Build configurations` - `Edit configurations` :

- Framework preset: None
- Build command: `npm install gitsite-cli -g && gitsite-cli build -o _site -v`

- Build output directory: `/_site`
- Root directory: leave empty.

Deploy to Vercel

To deploy site to Vercel, create a new project by import GitHub repo, then configure project:

- Framework Preset: Other
- Root Direction: `./`

Build and Output Settings:

- Build Command: `npm install -g gitsite-cli && gitsite-cli build -o dist -v`
- Output Directory: `dist`
- Install Command: leave empty.

Deploy to Self-hosted Nginx

GitSite generates pure HTML files by command `gitsite-cli build`. You can specify the output directory (default to `dist`) by `--output` or `-o`:

```
$ gitsite-cli build -o dist -v
```

You can run Nginx by Docker quickly:

```
$ docker run --rm -p 8000:80 -v /path/to/dist:/usr/share/nginx/html
nginx:latest
```

Site is served and can be previewed at `http://localhost:8000`.

Organizing Your Contents

GitSite helps you organize your contents. There are 3 types of contents that GitSite supports. All files are put in `source` directory:

```
source/
├── books/
├── blogs/
├── pages/
└── site.yml
```

Books

A book contains a set of markdown documents organized with a tree structure:

```
books/
└── guide/
    ├── book.yml    ⟵ book information
    ├── a/          ⟵ URL: /books/guide/a/index.html
    │   ├── m/      ⟵ URL: /books/guide/a/m/index.html
    │   ├── n/      ⟵ URL: /books/guide/a/n/index.html
    │   │   ├── p/  ⟵ URL: /books/guide/a/n/p/index.html
    │   │   └── q/  ⟵ URL: /books/guide/a/n/q/index.html
    │   └── l/
    ├── b/
    │   ├── x/
    │   ├── y/
    │   └── z/
    │       ├── v1/
    │       └── v2/
    └── c/
        ├── w1/
        └── w2/
```

Each directory must have a markdown file named `README.md`. GitSite scans the directory and generates book index as tree, sorted by directory name.

To specify the order, you can add sequence number in the directory name. For example:

```
books/
└── guide/
    ├── 10-b/       ⟵ URL: /books/guide/b/index.html
    ├── 20-c/       ⟵ URL: /books/guide/c/index.html
    └── 30-a/       ⟵ URL: /books/guide/a/index.html
        ├── 10-n/   ⟵ URL: /books/guide/a/n/index.html
        └── 20-m/   ⟵ URL: /books/guide/a/m/index.html
```

This re-order the index with `b`, `c` and `a`. The sequence number is used for order and will be dropped in URL, so re-order a directory does not change the URL.

Blogs

Blogs are simple markdown files organized with a list of categories, and each directory name starts with ISO date format `yyyy-MM-dd`:

```
blogs/
└── tech/                    ⟵ category: /blogs/tech/index.html
    ├── index.json           ⟵ URL: /blogs/tech/index.json
    ├── 2024-01-01-hello/    ⟵ URL: /blogs/tech/2024-01-01-hello/index.html
    ├── 2024-01-05/          ⟵ URL: /blogs/tech/2024-01-05/index.html
    └── 2024-02-10-style/    ⟵ URL: /blogs/tech/2024-02-10-style/index.html
```

Pages

Pages are used to display single pages. Each page has a simple directory name:

```
pages/
├── about/     ⟵ URL: /pages/about/index.html
├── license/   ⟵ URL: /pages/license/index.html
└── privacy/   ⟵ URL: /pages/privacy/index.html
```

Pages are usually serves pages like `License`, `Term of Service`, etc.

Index file

All markdown documents are named `README.md` for this can be viewed on GitHub directly, and build to `index.html`. The first line of `README.md` must be heading 1, which is treated as title, and the rest of lines are content:

```
# This is title

This is content ...

More content ...
```

Write Content

Markdown is an easy-to-read, easy-to-write language for formatting plain text. You can use Markdown syntax, along with some additional HTML tags, to format your writing on GitHub or any other git repository.

If you're new to Markdown, you might want to start with Basic syntax or Markdown guide.

Basic Syntax

Headings

To create a heading, add 1 ~ 6 `#` symbols before your heading text.

A first-level heading has 1 `#` symbol, a second-level heading has 2 `#` symbols, and so on:

```
# Heading 1

## Heading 2

### Heading 3

#### Heading 4

##### Heading 5

###### Heading 6
```

Heading 1

Heading 2

Heading 3

Heading 4

Heading 5

Heading 6

Styling text

You can indicate emphasis with bold, italic, strikethrough, subscript, or superscript text in comment fields and .md files.

Style	Syntax	Example	Output
Bold	`** **`	`This is a **bold** text.`	This is a **bold** text.
Italic	`* *`	`This is a *italic* text.`	This is a ***italic*** text.
Strikethrough	`~~ ~~`	`This is ~~not~~ a good idea.`	This is ~~not~~ a good idea.
Subscript	`<sub> </sub>`	`This is a <sub>subscript</sub> text.`	This is a _{subscript} text.
Superscript	`<sup> </sup>`	`This is a <sup>superscript</sup> text.`	This is a ^{superscript} text.

Quoting text

You can quote text with a `>` .

```
> To be, or not to be, that is the question
>
> -- Hamlet
```

> *To be, or not to be, that is the question.*
>
> *-- Hamlet*

Links

You can create an inline link by wrapping link text in brackets `[ ]` , and then wrapping the URL in parentheses `( )` .

```
[License](/pages/license/index.html)

[GitHub](https://github.com)
```

License

GitHub

Images

You can display an image by adding `!` and wrapping the alt text in `[ ]`. Alt text is a short text equivalent of the information in the image. Then, wrap the link for the image in parentheses `( )`.

```
![Cat Logo](cat.png)
```

Lists

You can make an unordered list by preceding one or more lines of text with `-`, `*` or `+`.

```
- JavaScript
- Python
- Rust
```

- JavaScript
- Python
- Rust

To order your list, precede each line with a number.

```
1. JavaScript
2. Python
3. Rust
```

1. JavaScript

2. Python
3. Rust

Nested lists

You can create a nested list by indenting one or more list items below another item.

```
- Programming Languages
  - JavaScript
  - Python
  - Rust
- Operating Systems
  1. Windows
  2. Linux
  3. macOS
```

- Programming Languages
  - JavaScript
  - Python
  - Rust
- Operating Systems
  1. Windows
  2. Linux
  3. macOS

Paragraphs

You can create a new paragraph by leaving a blank line between lines of text.

Footnotes

You can add footnotes to your content by using this bracket syntax:

```
Earth is the third planet from the Sun [^sun].

Jupiter is the fifth planet from the Sun and the largest in the Solar
```

```
System. [^solar_system].

[^sun]: The [Sun](https://en.wikipedia.org/wiki/Sun) is the star at the
center of the Solar System.

[^solar_system]: The Solar System is the gravitationally bound system of
the Sun and the objects that orbit it.
```

Earth is the third planet from the Sun [1].

Jupiter is the fifth planet from the Sun and the largest in the Solar System. [2].

The position of a footnote in your Markdown does not influence where the footnote will be rendered. You can write a footnote right after your reference to the footnote, and the footnote will still render at the bottom of the Markdown.

---

1. The Sun is the star at the center of the Solar System. ↩

2. The Solar System is the gravitationally bound system of the Sun and the objects that orbit it. ↩

Table

You can build tables to organize information.

Creating a table

You can create tables with pipes `|` and hyphens `-`. Hyphens are used to create each column's header, while pipes separate each column. You must include a blank line before your table in order for it to correctly render.

```
| First Header  | Second Header |
| ------------- | ------------- |
| Content Cell  | Content Cell  |
| Content Cell  | Content Cell  |
```

First Header	Second Header
Content Cell	Content Cell
Content Cell	Content Cell

The pipes on either end of the table are optional.

Cells can vary in width and do not need to be perfectly aligned within columns. There must be at least three hyphens in each column of the header row.

```
| Command | Description |
| --- | --- |
| git status | List all new or modified files |
| git diff | Show file differences that haven't been staged |
```

Command	Description
git status	List all new or modified files
git diff	Show file differences that haven't been staged

Formatting content within your table

You can use formatting such as links, inline code blocks, and text styling within your table:

```
| Command | Description |
| --- | --- |
| `git status` | List all *new or modified* files |
| `git diff` | Show file differences that **haven't been** staged |
```

Command	Description
`git status`	List all ***new or modified*** files
`git diff`	Show file differences that **haven't been** staged

You can align text to the left, right, or center of a column by including colons `:` to the left, right, or on both sides of the hyphens within the header row.

```
| Left-aligned | Center-aligned | Right-aligned |
| :---         |     :---:      |          ---: |
| git status   | git status     | git status    |
| git diff     | git diff       | git diff      |
```

Left-aligned	Center-aligned	Right-aligned
git status	git status	git status
git diff	git diff	git diff

To include a pipe `|` as content within your cell, use a `\` before the pipe:

```
| Name     | Character |
| ---      | ---       |
| Backtick | `         |
| Pipe     | \|        |
```

Name	Character
Backtick	`
Pipe	|

Code Blocks

You can create fenced code blocks by placing triple backticks ``` before and after the code block. We recommend placing a blank line before and after code blocks to make the raw formatting easier to read.

````
```
function test() {
    console.log('Hello, world.');
}
```
````

```
function test() {
    console.log('Hello, world.');
}
```

Syntax highlighting

You can add an optional language identifier to enable syntax highlighting in your fenced code block.

Syntax highlighting changes the color and style of source code to make it easier to read.

For example, to syntax highlight Python code:

````
```python
#!/usr/bin/env python3

def main():
    name = 'Bob'
    print(f'Hello, {name}')

if __name__ == '__main__':
    main()
```
````

This will display the code block with syntax highlighting:

```
#!/usr/bin/env python3

def main():
    name = 'Bob'
    print(f'Hello, {name}')

if __name__ == '__main__':
    main()
```

Alerts

Alerts are a Markdown extension based on the blockquote syntax that you can use to emphasize critical information. On GitHub, they are displayed with distinctive colors and icons to indicate the significance of the content.

Use alerts only when they are crucial for user success and limit them to one or two per article to prevent overloading the reader. Additionally, you should avoid placing alerts consecutively.

To add an alert, add text inside a fenced code block with the `alert type=note` identifier. This is **different** from GitHub which uses a special blockquote line to specify the alert type. Five types of alerts are available:

````
```alert type=note
Useful information that users should know, even when skimming content.
```

```alert type=tip
Helpful advice for doing things better or more easily.
```

```alert type=important
Key information users need to know to achieve their goal.
```

```alert type=warning
Urgent info that needs *immediate* user attention to avoid problems.
```

```alert type=caution
Advises about **risks** or negative outcomes of certain actions.
```
````

> Note
>
> Useful information that users should know, even when skimming content.

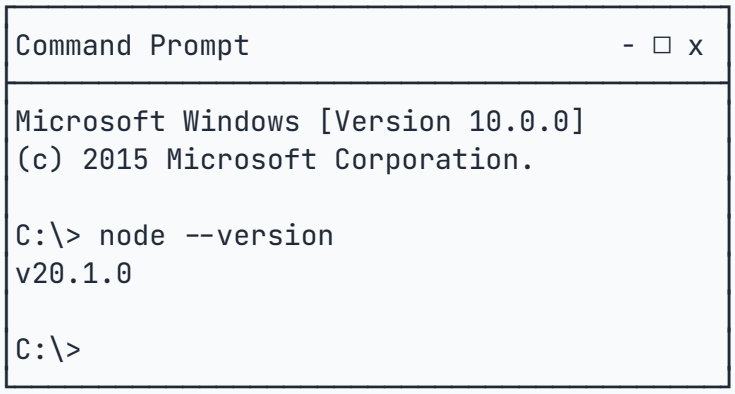
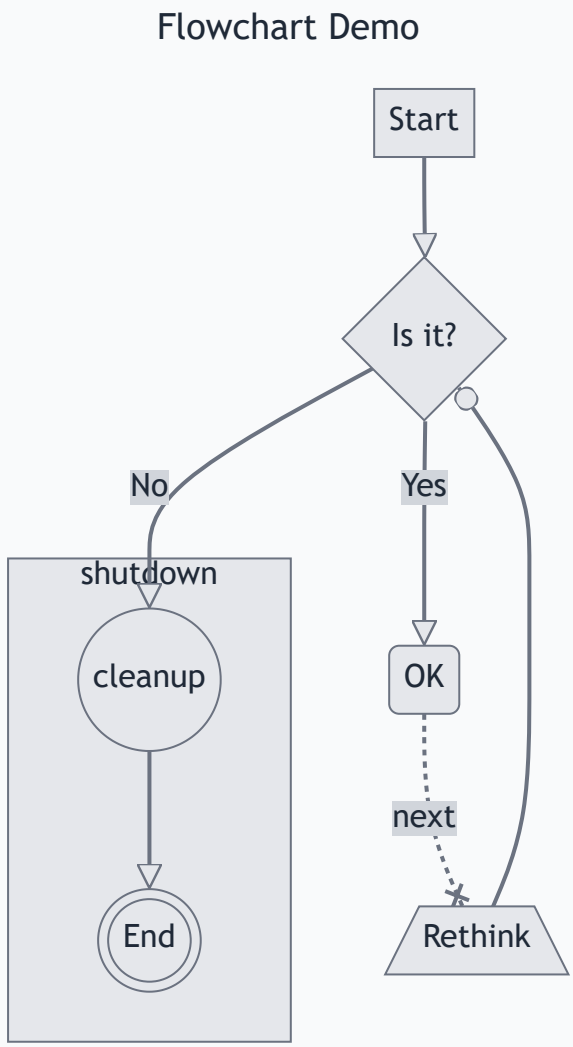
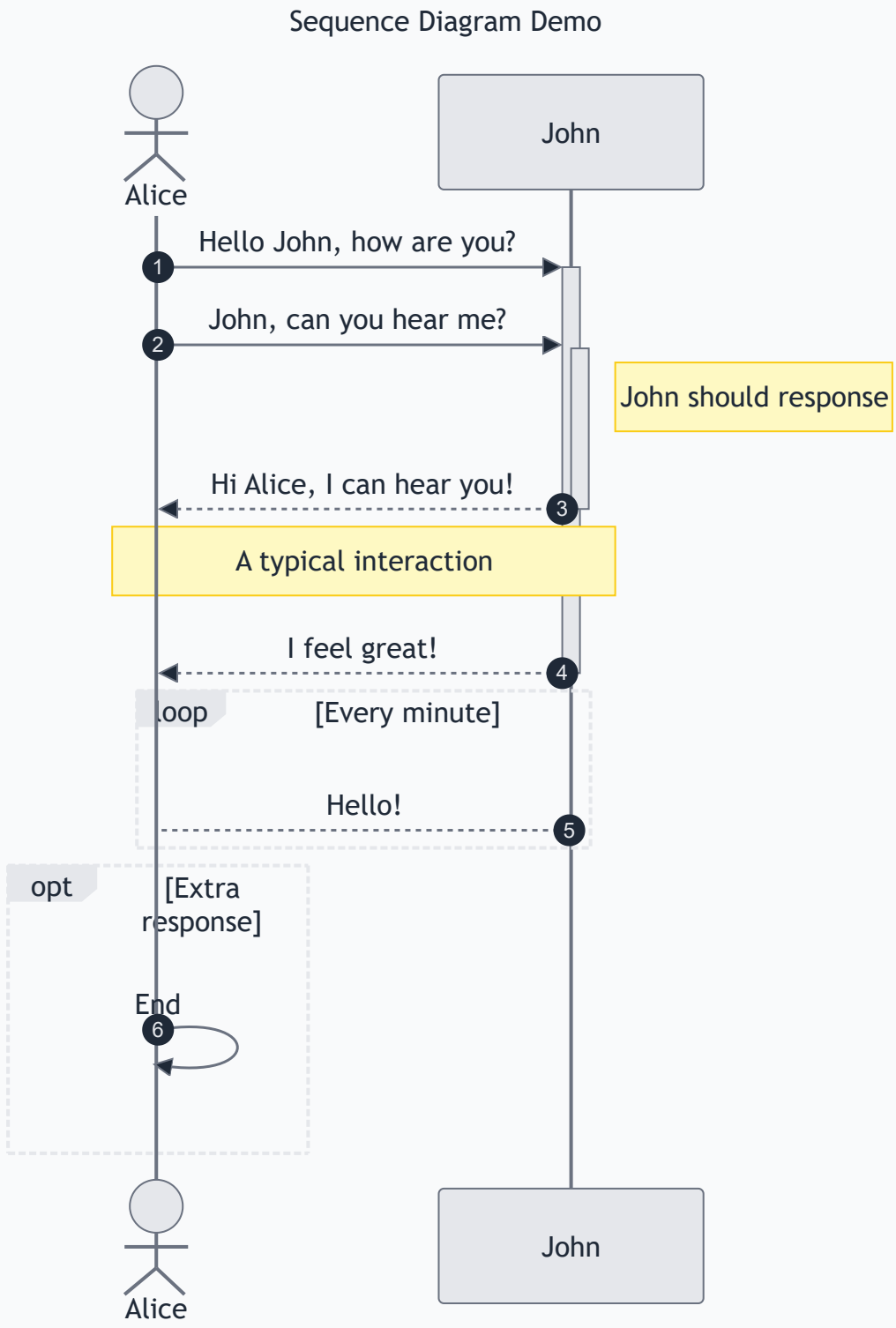
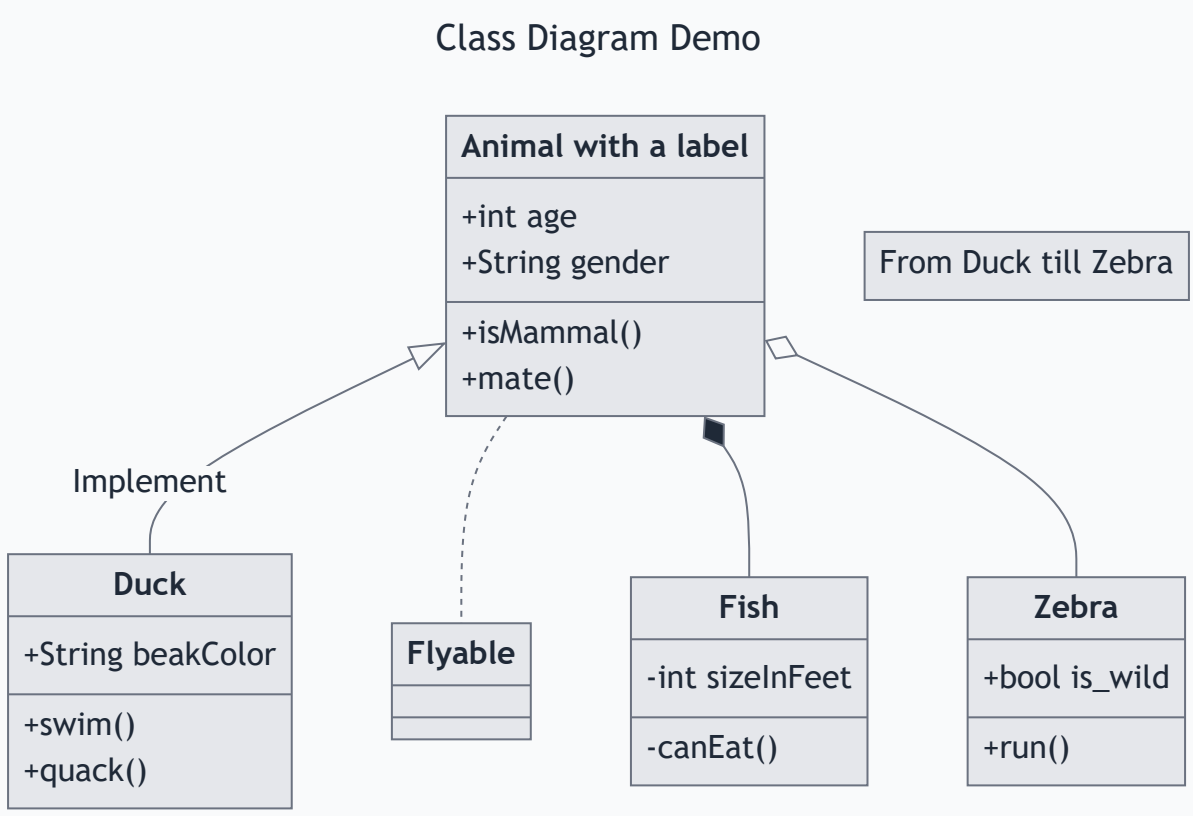
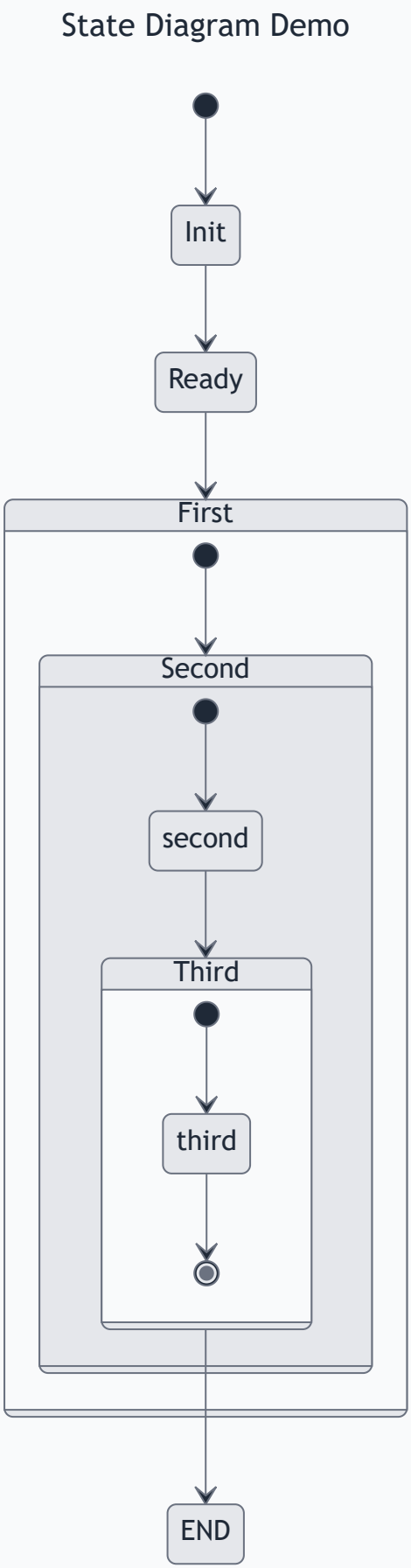
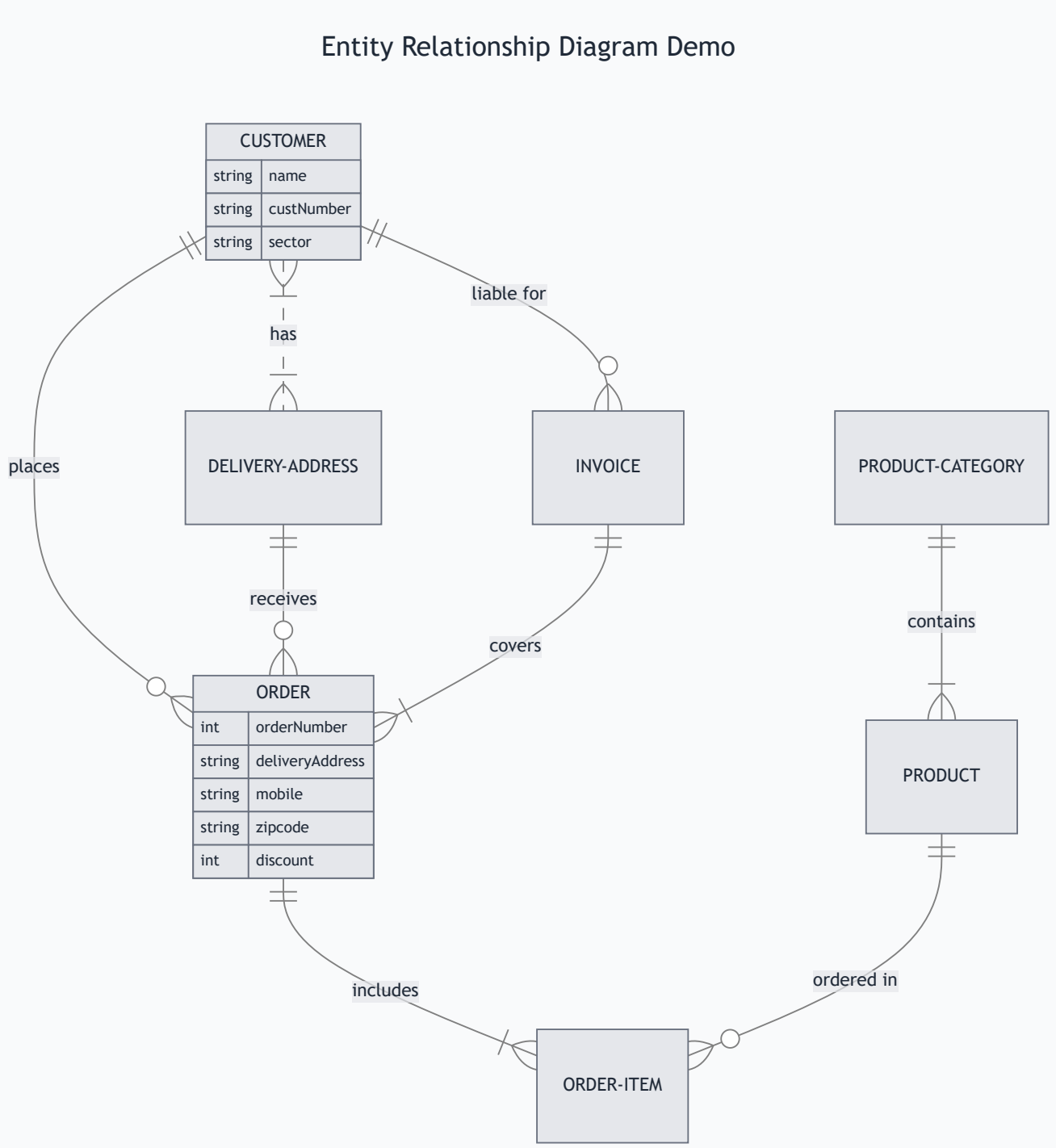
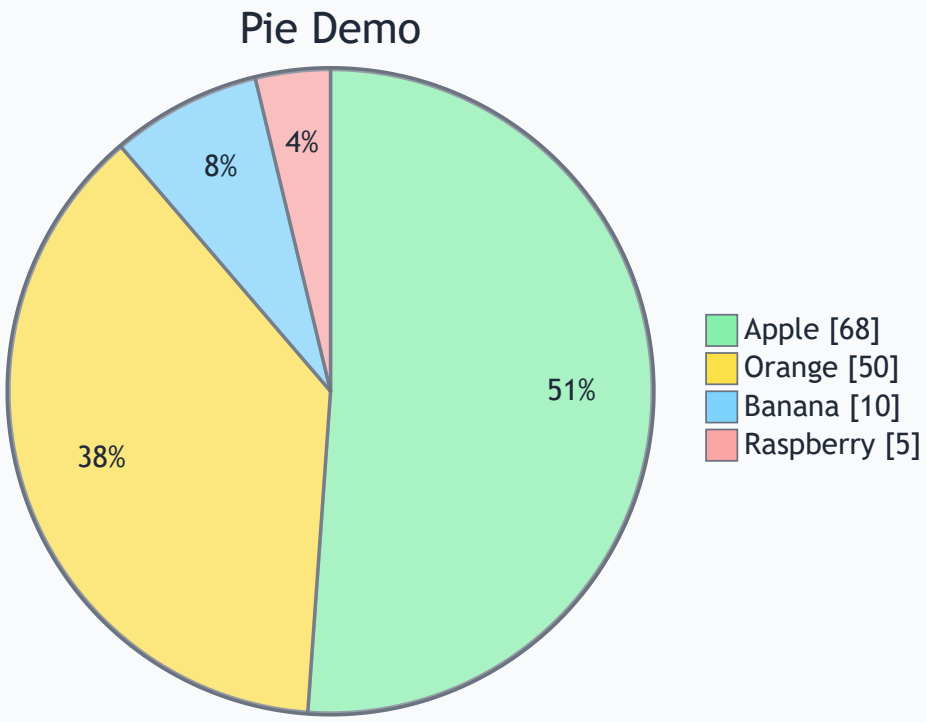
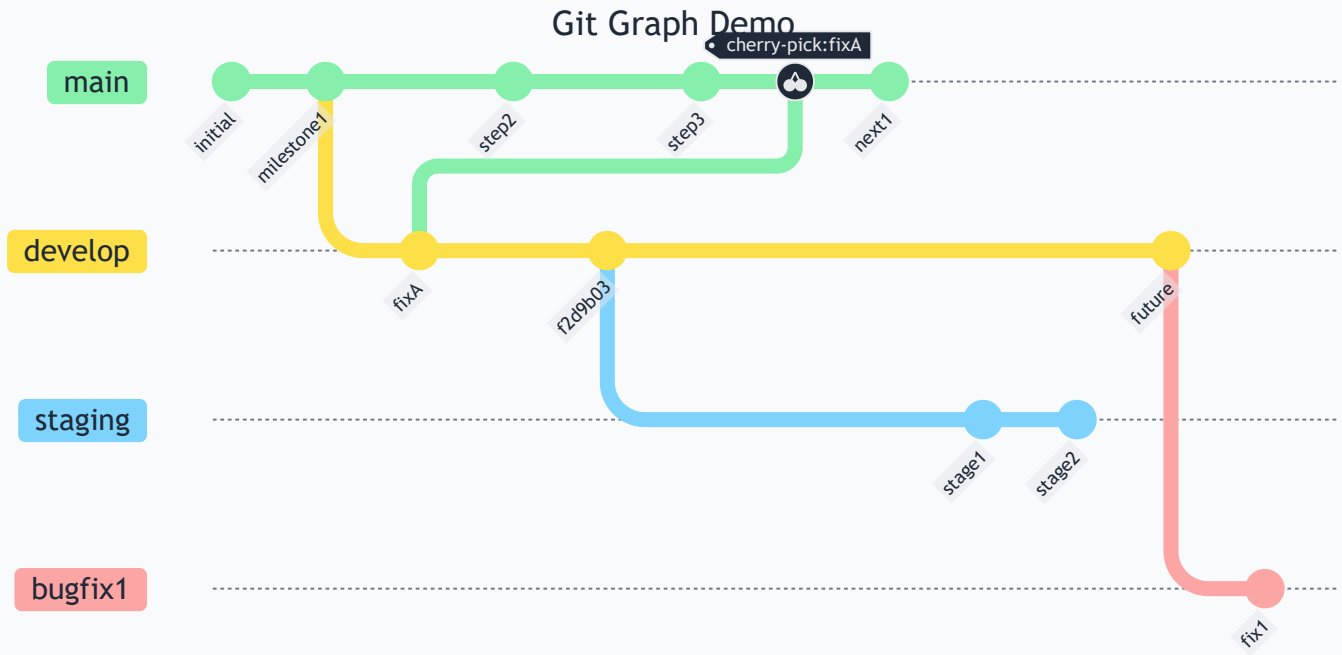
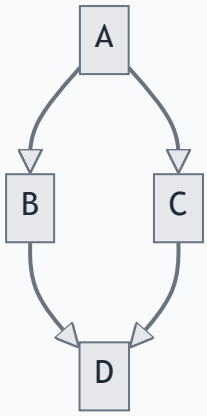
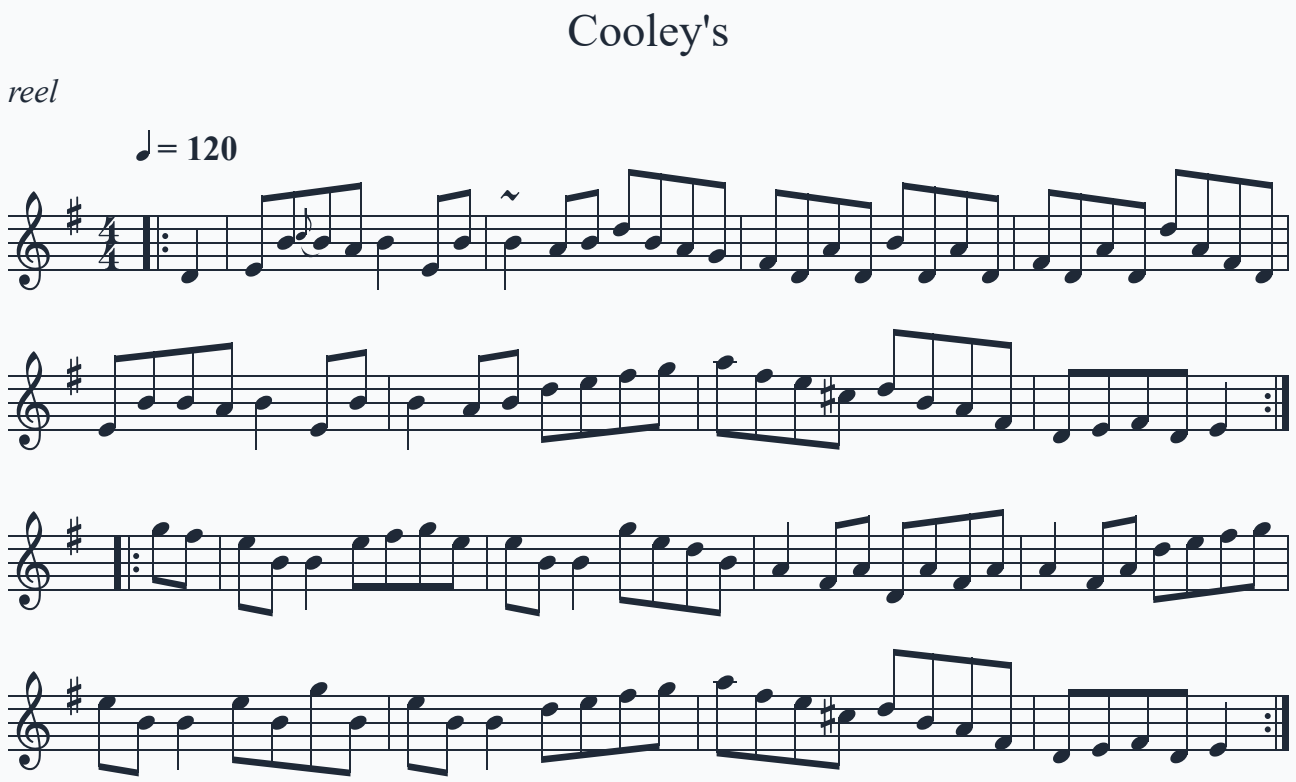
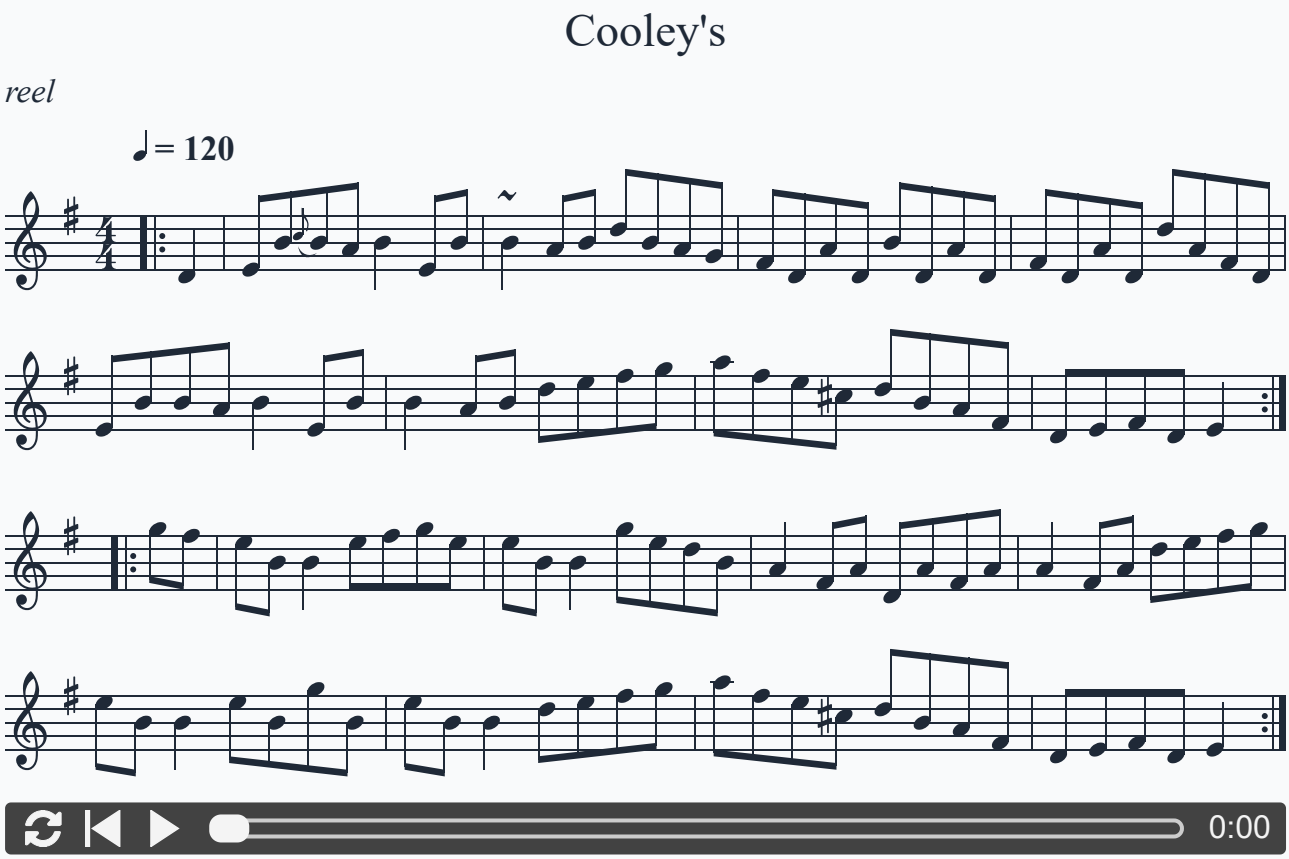
> Tip
>
> Helpful advice for doing things better or more easily.

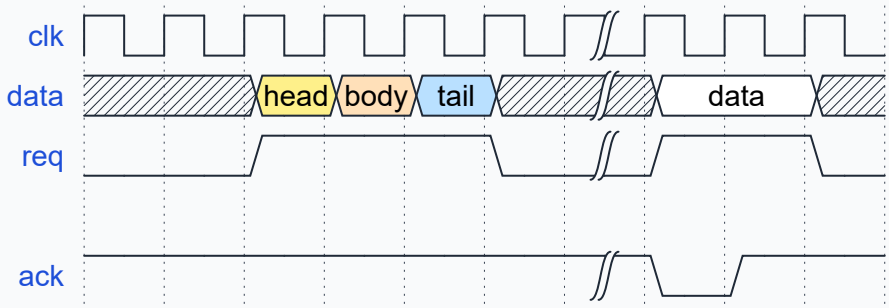
> Important
>
> Key information users need to know to achieve their goal.

> Warning
>
> Urgent info that needs ***immediate*** user attention to avoid problems.

> Caution
>
> Advises about **risks** or negative outcomes of certain actions.

To specify the alert title, use `title=Xyz` :

```
```alert type=caution title=Error
There is an ERROR!
```
```

> Error
>
> There is an ERROR!

Title that contains spaces must be quoted with `"` :

```
```alert type=tip title="Useful Tips"
Helpful advice for doing things better or more easily.
```
```

> Useful Tips
>
> Helpful advice for doing things better or more easily.

Videos

To embed a video, add URL inside a fenced code block with the `video` identifier.

```
```video
https://www.youtube.com/watch?v=RcnksOUugcA
```
```

The default player aspect ratio is 4:3. You can set ratio by `ratio=16:9`, and set max width with 640px by `max-width=640`. Alignment can also be set to `left`, `center` or `right`:

```
```video ratio=16:9 max-width=640 align=center
https://www.youtube.com/watch?v=RcnksOUugcA
```
```

GitSite now support videos from YouTube and Bilibili.

Advanced Formatting

GitSite uses markdown plugins to extend the markdown format. You can create mathematical expressions, ASCII art, QRCode, diagrams and question forms in an easy way.

Mathematical Expressions

To enable clear communication of mathematical expressions, GitSite supports LaTeX formatted math within Markdown. For more information, see LaTeX/Mathematics in Wikibooks.

GitSite's math rendering capability uses Katex: an open source, JavaScript-based display engine.

Writing inline expressions

You can surround the expression with dollar symbols `$` .

```
This sentence uses `$` delimiters to show math inline:  $\sqrt{3x-1}+(1+x)^2$.
```

This sentence uses `$` delimiters to show math inline: $\sqrt{3x-1}+(1+x)^2$.

Writing expressions as blocks

To add a math expression as a block, add expression inside a fenced code block with the `math` identifier.

```
```math
\left( \sum_{k=1}^n a_k b_k \right)^2 \leq \left( \sum_{k=1}^n a_k^2 \right)
```
```

$$\left(\sum_{k=1}^n a_k b_k \right)^2 \leq \left(\sum_{k=1}^n a_k^2 \right) \left(\sum_{k=1}^n b_k^2 \right)$$

You can set the alignment to center or right. For example, use `align=center` to set alignment to center:

```
```math align=center
\left( \sum_{k=1}^n a_k b_k \right)^2 \leq \left( \sum_{k=1}^n a_k^2 \right)
```
```

$$\left(\sum_{k=1}^n a_k b_k \right)^2 \leq \left(\sum_{k=1}^n a_k^2 \right) \left(\sum_{k=1}^n b_k^2 \right)$$

You can still use GitHub-styled math expression block by start a new line and delimit the expression with two dollar symbols `$$`, but you cannot set alignment by this way (always display in center):

```
$$
\left( \sum_{k=1}^n a_k b_k \right)^2 \leq \left( \sum_{k=1}^n a_k^2
\right) \left( \sum_{k=1}^n b_k^2 \right)
$$
```

$$\left(\sum_{k=1}^n a_k b_k \right)^2 \leq \left(\sum_{k=1}^n a_k^2 \right) \left(\sum_{k=1}^n b_k^2 \right)$$

> Tip
>
> Use the visual editor provided by Wiris can help you write math expressions much easier.

Writing chemical equations

The Katex also supports chemical equations by the mhchem extension.

```
Chemical equation example: $\ce{CO2 + C -> 2 CO}$.
```

Chemical equation example: $\ce{CO2 + C -> 2 CO}$.

A complex chemical equation as a block:

````
```math
\ce{Hg^2+ ->[I-] HgI2 ->[I-] [Hg^{II}I4]^2-}
```
````

$$\ce{Hg^2+ ->[I-] HgI2 ->[I-] [Hg^{II}I4]^2-}$$

ASCII Art

To create an ASCII art, just add identifier `ascii` to code blocks:

```
```ascii
┌──────────────────────────────────────────┐
│Command Prompt                    - □ x   │
├──────────────────────────────────────────┤
│Microsoft Windows [Version 10.0.0]        │
│(c) 2015 Microsoft Corporation.           │
│                                          │
│C:\> node --version                       │
│v20.1.0                                   │
│                                          │
│C:\>                                      │
└──────────────────────────────────────────┘
```
```



QRCode

To create a QRCode, add text inside a fenced code block with the `qrcode` identifier.

GitSite's QRCode rendering capability uses qrcode-svg: an open source, JavaScript-based library.

````
```qrcode
https://gitsite.org
```
````

The text inside the fenced code block will be converted to a QRCode:

Image can be set by URL:

````
```qrcode image=like.png
https://gitsite.org
```
````

You can specify some extra arguments to control the QRCode:

Parameter	Description
ecl=[l|m|h|q]	The error correction level, default to `ecl=l` .
width=[size-in-px]	The width in px, default to `width=200` .
align=[left|center|right]	The alignment of the QRCode, default to `align=left` .
image=[URL]	The image URL, default to none.
image-width=[size-in-px]	The image width, default to auto.
info	Display the text content, if present.
link	Display the text as URL, if `info` present and the text is a valid link.

Example:

````
```qrcode ecl=m width=300 align=center info link
https://gitsite.org
```
````

https://gitsite.org

Diagrams

You can create diagrams to convey information through charts and graphs.

Creating Mermaid diagrams

Mermaid is a Markdown-inspired tool that renders text into diagrams. For example, Mermaid can render flow charts, sequence diagrams, pie charts and more. For more information, see the Mermaid documentation.

GitSite don't require Mermaid JavaScript on page because it pre-compile Mermaid diagrams to SVGs which are embeded into HTML.

To create a Mermaid diagram, add Mermaid syntax inside a fenced code block with the `mermaid` identifier.

> Tip
>
> It is a good practice that use Mermaid live editor to make sure your markdown text can be compiled to Mermaid diagrams successfully.

Creating flow chart

You can create a flow chart by specifying values and arrows.

```
```mermaid
---
title: Flowchart Demo
---
flowchart TD
    A[Start] --> B{Is it?}
    B -->|Yes| C(OK)
    C -. next .-x D[/Rethink\]
    D --o B
    B -->|No| E((cleanup))
    subgraph shutdown
        E --> F(((End)))
```
```

```
    end
```



Creating sequence diagram

A sequence diagram is an interaction diagram that shows how processes operate with one another and in what order.

```
```mermaid
---
title: Sequence Diagram Demo
---
sequenceDiagram
    actor Alice
    autonumber
    Alice->>+John: Hello John, how are you?
    Alice->>+John: John, can you hear me?
```
```

```
    Note right of John: John should response
    John-->>-Alice: Hi Alice, I can hear you!
    Note over Alice,John: A typical interaction
    John-->>-Alice: I feel great!
    loop Every minute
        John-->Alice: Hello!
    end
    opt Extra response
        Alice->>Alice: End
    end
```



Creating class diagram

The class diagram is the main building block of object-oriented modeling.

```
```mermaid
---
title: Class Diagram Demo
---
```
```

```
classDiagram
    class Animal["Animal with a label"]
    note "From Duck till Zebra"
    Animal <|-- Duck : Implement
    Animal .. Flyable
    Animal *-- Fish
    Animal o-- Zebra
    Animal : +int age
    Animal : +String gender
    Animal: +isMammal()
    Animal: +mate()
    class Duck{
    +String beakColor
    +swim()
    +quack()
    }
    class Fish{
    -int sizeInFeet
    -canEat()
    }
    class Zebra{
    +bool is_wild
    +run()
    }
```



Creating state diagram

A state diagram is a type of diagram used in computer science and related fields to describe the behavior of systems.

```
```mermaid
---
title: State Diagram Demo
---
stateDiagram-v2
    [*] --> Init
    Init --> Ready
    Ready --> First
    state First {
        [*] --> Second

        state Second {
            [*] --> second
            second --> Third

            state Third {
                [*] --> third
```
```

```
                third --> [*]
            }
        }
    }
    Third --> END
```



Creating entity relationship diagram

An entity relationship model (or ER model) describes interrelated things of interest in a specific domain of knowledge.

```mermaid
---
title: Entity Relationship Diagram Demo
---
erDiagram
    CUSTOMER {
        string name
        string custNumber
        string sector
    }
    ORDER {
        int orderNumber
        string deliveryAddress
        string mobile
        string zipcode
        int discount
    }
    CUSTOMER }|..|{ DELIVERY-ADDRESS : has
    CUSTOMER ||--o{ ORDER : places
    CUSTOMER ||--o{ INVOICE : "liable for"
    DELIVERY-ADDRESS ||--o{ ORDER : receives
    INVOICE ||--|{ ORDER : covers
    ORDER ||--|{ ORDER-ITEM : includes
    PRODUCT-CATEGORY ||--|{ PRODUCT : contains
    PRODUCT ||--o{ ORDER-ITEM : "ordered in"
```



Creating pie chart

A pie chart is a circular statistical graphic, which is divided into slices to illustrate numerical proportion.

```mermaid
pie showData
    title Pie Demo
    "Apple" : 68
```

```
    "Orange" : 50
    "Banana" : 10
    "Raspberry" : 5
```



Creating git graph

A Git Graph is a pictorial representation of git commits and git actions on various branches.

```mermaid
---
title: Git Graph Demo
---
gitGraph
    commit id: "initial"
    commit id: "milestone1"
    branch develop
    commit id:"fixA"
    checkout main
    commit id:"step2"
    checkout develop
    commit id:"f2d9b03"
```

```
    checkout main
    commit id:"step3"
    cherry-pick id:"fixA"
    commit id:"next1"
    checkout develop
    branch staging
    commit id:"stage1"
    commit id:"stage2"
    checkout develop
    commit id:"future"
    branch bugfix1
    commit id:"fix1"
```



Alignment

To align a Mermaid diagram, add `align=left` , `align=center` or `align=right` after the `mermaid` identifier:

```
```mermaid align=center
---
title: Flowchart in center
---
graph TD;
```
```

```
    A-->B;
    A-->C;
    B-->D;
    C-->D;
```

Flowchart in center



Questions

Creating Questions

You can create a question form by add identifier `question` and subtype with a code block.

Creating single selection

Use `question type=radio` to create a single selection form:

````
```question type=radio
Which is the first browser that support **JavaScript**?
---
    Internet Explorer
[x] Netscape Navigator
    Mozilla Firefox
    Google Chrome
```
````

The question and anwsers are seperated by `---`, and use `[x]` to mark the correct answer.

Which is the first browser that support **JavaScript**?

◯ Internet Explorer

◯ Netscape Navigator

◯ Mozilla Firefox

◯ Google Chrome

Creating multiple selections

Use `question type=checkbox` to create a multiple selection form:

````
```question type=checkbox
The planets that in the solar system:
---
````
````

```
[x] Earth
[x] Mercury
    Moon
    Halley
[x] Jupiter
```

The planets that in the solar system:

- [ ] Earth
- [ ] Mercury
- [ ] Moon
- [ ] Halley
- [ ] Jupiter

Creating text input

Use `question type=text` to create a text input form:

```
```question type=text
The largest planet in the solar system is:
----
Jupiter
```
```

The largest planet in the solar system is:

Add `ignorecase` to ignore the case of answer:

```
```question type=text ignorecase
The largest planet in the solar system is (ignore case):
----
Jupiter
```
```

Both `Jupiter` and `jupiter` are correct answer:

The largest planet in the solar system is (ignore case):

Creating date input

Use `question type=date` to create a date input form:

````
```question type=date
When were the first modern Olympic Games held? (July 6, 1896)
----
1896-04-06
```
````

The provided answer must be ISO-8601 date format `yyyy-MM-dd` .

When were the first modern Olympic Games held? (July 6, 1896)

年 /月/日

Customizing labels

You can also specify the label of submit, correct and wrong:

````
```question type=radio submit=Validate correct="Yes, it is correct!" wrong="S
Which is the first browser that support **JavaScript**?
---
    Internet Explorer
[x] Netscape Navigator
    Mozilla Firefox
    Google Chrome
```
````

Which is the first browser that support **JavaScript**?

◯ Internet Explorer

◯ Netscape Navigator

◯ Mozilla Firefox


Google Chrome

Music

To create a sheet music, add text inside a fenced code block with the `abcjs` identifier.

GitSite uses abcjs to render sheet music. Abcjs is an open source, JavaScript-based library that makes it easy to incorporate sheet music on websites.

You have to add the JavaScript because it cannot be rendered offline:

```
<script src="/static/abcjs-basic.js"></script>
<script src="/static/abcjs-init.js"></script>
```

The Abc notation inside the fenced code block will be converted to a sheet music:

````
```abcjs max-width=640
T: Cooley's
M: 4/4
Q: 1/4=120
L: 1/8
R: reel
K: Emin
|:D2|EB{c}BA B2 EB|~B2 AB dBAG|FDAD BDAD|FDAD dAFD|
EBBA B2 EB|B2 AB defg|afe^c dBAF|DEFD E2:|
|:gf|eB B2 efge|eB B2 gedB|A2 FA DAFA|A2 FA defg|
eB B2 eBgB|eB B2 defg|afe^c dBAF|DEFD E2:|
```
````



You can specify the max width by `max-width=640` and the alignment by `align=center`. The really interesting thing is that you can play the music by `controls` instruction:

````
```abcjs max-width=640 align=center controls
T: Cooley's
M: 4/4
Q: 1/4=120
L: 1/8
R: reel
K: Emin
|:D2|EB{c}BA B2 EB|~B2 AB dBAG|FDAD BDAD|FDAD dAFD|
EBBA B2 EB|B2 AB defg|afe^c dBAF|DEFD E2:|
|:gf|eB B2 efge|eB B2 gedB|A2 FA DAFA|A2 FA defg|
eB B2 eBgB|eB B2 defg|afe^c dBAF|DEFD E2:|
```
````



Tip

The offical abcjs website provides a live editor with synth, so you can edit your sheet and check if it has errors.

Digital Timing

To create a digital timing diagram (wave form), add text inside a fenced code block with the `wavedrom` identifier.

GitSite uses wavedrom to render digital timing diagrams. Wavedrom is an open source, JavaScript-based library that makes it easy to render digital timing diagrams into SVG vector graphics.

````
```wavedrom
{ signal: [
  { name: "clk",  wave: "p.....|..." },
  { name: "data", wave: "x.345x|=.x", data: ["head", "body", "tail", "data"]
  { name: "req",  wave: "0.1..0|1.0" },
  {},
  { name: "ack",  wave: "1.....|01." }
]}
```
````



You can specify the alignment by `align=center` .

> Tip
>
> Use the wavedrom online editor to edit your digital timing digram.

Create PDF

GitSite has an excellent feature that can export the whole book to a single PDF file.

To create PDF you must run the site on your local computer:

```
$ gitsite-cli serve
```

To create a PDF based on a book, for example, `gitsite-guide`, just open the url http://localhost:3000/books/gitsite-guide/pdf in browser and PDF is generated on-the-fly.

You can download the generated PDF gitsite-guide.pdf here.

Technical Details

GitSite uses Puppeteer and pdf-lib to generate PDF from HTML. In general, there are 3 HTML pages which generated by gitsite-cli:

1. The content page: `/books/gitsite-guide/pdf.html`;
2. The front cover page: `/books/gitsite-guide/pdf.front.html`;
3. The back cover page: `/books/gitsite-guide/pdf.back.html`.

The URLs listed above are only available on local development mode. You can check the pages to preview PDF content.

Header and footer on every page can also be customized. Header and footer are defined in the following templates:

- Header template: `pdf_header.html`;
- Footer template: `pdf_footer.html`.

Developer's Guide

GitSite uses Nunjucks as the template engine, rendering Markdown documents in real time for local preview, and converting markdown documents to HTML files in build mode.

All themes are stored in the `themes` directory. You can install multiple themes, and the `default` theme is used by default. To specify a particular theme, you can configure it in `site.yml` :

```
site:
  theme: default
```

Template files are required under a specific theme:

```
default
├── blog.html          ←── for blog full page
├── blog_content.html ←── for blog content page
├── book.html          ←── for book chapter page
├── book_content.html ←── for book chapter content page
├── page.html          ←── for any single document page
├── pdf.html           ←── for generate PDF content
├── pdf_front.html     ←── for generate PDF front cover
├── pdf_back.html      ←── for generate PDF back cover
├── pdf_header.html    ←── for generate PDF header
├── pdf_footer.html    ←── for generate PDF footer
├── post-build.mjs     ←── (optional) auto executed after build
├── pre-build.mjs      ←── (optional) auto executed before build
└── static             ←── (optional) store static files
```

Nunjucks template engine enables the inheritance and referencing of other templates within a single template. This feature proves advantageous for reusing components, allowing for the extraction of common parts such as header and footer.

Create a new theme

To create a new theme, the best way is copying the `default` theme and rename it, then update the theme setting in `site.yml` . This allows you change the theme progressively.

Themes are version controlled with source in the same repo.

GitSite's Guide

How to use GitSite.

Crypto Michael

Website: https://gitsite.org/books/gitsite-guide/

Version: 2025-05-25