Documentation

Project documentation

Author

Table of Contents

1. Welcome

Welcome to the project documentation.

> **Version 2.0** — Added versioning and PDF download for multiple languages.

1.1 Getting started

Add sections and pages as needed. Navigation is configured in `mkdocs.yml` under the `nav` section.

1.2 Structure

- **Home** — this page
- **Download PDF** — get the docs as PDF in English or Portuguese
- **Examples** — Hello World and more code samples in multiple languages
- Add new pages under `docs/` and reference them in `mkdocs.yml`

1.3 Export to PDF

PDFs are generated on every build. The default-language PDF is at `site/pdf/documentation.pdf`. To build PDFs for both English and Portuguese, run:

```
./scripts/build-pdf-all.sh
```

WeasyPrint is required; on macOS install system dependencies first:

```
brew install cairo pango gdk-pixbuf
```

2. Download PDF

Download the documentation as a single PDF. Choose your language:

Download PDF — English **Download PDF — Português**

Two PDFs require the build script

A normal `mkdocs build` or `mkdocs serve` runs two language builds; the PDF is written to the same path each time, so you only get **one** file (Portuguese) and the PT link can 404. To get **both** PDFs (EN and PT) and working download links, run the script first, then serve the built site:

```
./scripts/build-pdf-all.sh
./scripts/serve-with-pdfs.sh
```

```
Then open the URL shown (e.g. http://127.0.0.1:8000/repo/ or http://127.0.0.1:8000/). Both download buttons will work.
```

One-off build (both PDFs, no server):

```
./scripts/build-pdf-all.sh
```

Single PDF only (e.g. for CI): run `mkdocs build` or `mkdocs serve`; the file is at `site/pdf/documentation.pdf` (content will be from the last build, usually PT).

3. Versioning

Documentation is versioned with mike. Use the version selector in the header to switch between **1.0**, **2.0**, and **2.1**. The version selector in the header lets users switch between deployed versions.

3.1 Prerequisites

- Set `site_url` in `mkdocs.yml` to your deployment URL (e.g. `https://username.github.io/repo/`).
- Deploy the site to a branch (e.g. `gh-pages`) that mike will use.

3.2 Deploying a new version

Deploy the current docs as a new version and set the `latest` alias:

```
mike deploy --push --update-aliases 1.0 latest
```

- `1.0` — version identifier (can be `2.0`, `dev`, etc.).
- `latest` — alias; visitors to the root URL are redirected to this version.

Deploy another version without changing the default:

```
mike deploy --push 0.9
```

3.3 Setting the default version

Redirect the site root to the version pointed to by an alias:

```
mike set-default --push latest
```

After this, opening `site_url` will redirect to the `latest` version.

3.4 Aliases

Use aliases for stable, dev, or specific releases:

```
mike deploy --push --update-aliases 2.0 latest
mike deploy --push 2.0-beta dev
mike deploy --push 1.0 stable
```

Then set the default, for example:

```
mike set-default --push latest
```

3.5 Local development

Running `mkdocs serve` does not use mike; the version selector is populated only after deploying with mike. To test versioning, build and deploy at least one version to your branch.

3.6 Testing with local tags

This repo has example tags **1.0**, **2.0**, and **2.1**. To deploy them with mike (e.g. to `gh-pages`):

```
mike deploy 1.0
mike deploy 2.0
mike deploy 2.1 latest
mike set-default --push latest
```

Then open the deployed site and use the version selector to switch between 1.0, 2.0, and 2.1.

4. Examples

4.1 Hello World

Classic "Hello, World!" examples in different programming languages. Switch between them using the tabs below.

Python JavaScript Go Java C++ Ruby

```python
print("Hello, World!")
```

```javascript
console.log("Hello, World!");
```

```go
package main

import "fmt"

func main() {
    fmt.Println("Hello, World!")
}
```

```java
public class Main {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

```cpp
#include <iostream>

int main() {
    std::cout << "Hello, World!" << std::endl;
    return 0;
}
```

```ruby
puts "Hello, World!"
```

4.1.1 Annotated example

The block below uses **line numbers** (`linenums="1"`), **highlighted lines** (`hl_lines="2 3"`), a **title**, and **annotations** — hover or tap the marker to see the note.

```python
1  def greet():
2      print("Hello, World!")  # (1)!
3      return None             # (2)!
```

1. Outputs the greeting to stdout. Use `print()` for simple console output.
2. Explicit `return None` is optional in Python; the function returns `None` by default.

4.2 More examples

Additional "Hello, World!" samples in other languages. Use the tabs to switch.

Rust PHP Swift Kotlin C Bash

```rust
fn main() {
    println!("Hello, World!");
}
```

```php
<?php
echo "Hello, World!";
```

```swift
print("Hello, World!")
```

```kotlin
fun main() {
    println("Hello, World!")
}
```

```c
#include <stdio.h>

int main(void) {
    printf("Hello, World!\n");
    return 0;
}
```

```bash
echo "Hello, World!"
```