Markdown compiler

Margo starts with ordinary Markdown. The root package exposes a small pipeline: compile a source document, render it for a target, and choose the final HTML shape at the host boundary.

The Go path

```
compiler := margo.New()

document, err := compiler.Compile(ctx, margo.Source{
    Name:    "guide.md",
    Content: markdown,
})
if err != nil {
    return err
}

rendered, err := compiler.Render(ctx, document)
if err != nil {
    return err
}

page, err := margo.RenderStandalone(rendered)
if err != nil {
    return err
}
```

The compiler owns Markdown parsing and document semantics. The application still owns where a page lives, how navigation is composed, and which assets are served.

The authoring surface

```
---
title: A small guide
language: en
---

# A small guide

Write headings, links, tables, code, images, and Mermaid diagrams as
Markdown.
```

Ordinary local images, tables, Mermaid, and code do not require a policy file. Privileged raw HTML and iframe embeds are a separate, explicitly authorized surface; see Policy.

Why this is useful

- The same source can feed HTML, a site, a PDF, or a deck.
- Metadata can travel with the document and still be overridden by an explicit API or CLI option.
- `margo.Check` can run before rendering when a workflow needs compatibility findings without producing an artifact.

The result is a content pipeline that is easy to embed and easy to inspect.
