Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Let's try some shortcodes: First, the Theorem shortcode:

Theorem 1 (Chinese Remainder Theorem) *The groups* $\mathbb{Z}/(ab)$ *and* $\mathbb{Z}/(a) \times \mathbb{Z}/(b)$ *are isomorphic if and only if* $\gcd(a, b) = 1$.

Then, sidenotes[1]: Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

[1] Like this one

Theorem 2 (Chinese Remainder Theorem) *The groups* $\mathbb{Z}/(ab)$ *and* $\mathbb{Z}/(a) \times \mathbb{Z}/(b)$ *are isomorphic if and only if* $\gcd(a, b) = 1$.

This information is important.

But this is a terrible mistake.

1 Hugo shortcodes in LaTeX's Lua Markdown

This package implements an extension to LaTeX's Lua Markdown parser (caveat: by someone who has never programmed in Lua) to process Hugo shortcodes. Although the package implements defines a `shortcode.lua` markdown extension, you still need to provide the corresponding LaTex definition for each shortcode. But some provisions are made to easily define shortcodes.

Let me state it again: You need to `\usepackage{shortcode}`, and then **you need to define your own macros for each shortcode**.

See test.tex for examples on how to define shortcodes or read on for an explanation.

1.1 How it works

The idea behind this package is a Lua markdown extension so that shortcodes

```
{{< exampleshortcode key1=value1 key2=value2 ... >}}
```

are translated into LaTeX input source as

```
\shortcode{exampleshortcode}[key1=value1,key2=value2,...]
```

When LaTeX is run on this source, LaTeX will call the macro

```
\shortcodeexampleshortcode[key1=value1,key2=value2,...]
```

You are suppose to write macro code for this last step. Note that this macro will silently do nothing if it's not defined.

There are some commands below to facilitate writing LaTeX macros.

Likewise, a closing shortcode

```
{{< /exampleshortcode>}}
```

gets translated to

```
\closeshortcode{exampleshortcode}
```

which in turn calls

```
\closeshortcodexampleshortcode
```

which, again, you have to define.

There is a very important limitation to such use of shortcodes: in Hugo, shortcodes arguments can be used in positional or named form. Technically you can do the same in LaTeX, but LaTeX isn't very well suited for this task. Below, I show how to deal with most use cases.

1.2 How to define your own LaTeX shortcodes

1.2.1 Simplest case

The simplest case is a shortcode with no options, like

```
{{< hr >}}
```

As explained above, this is translated by the markdown extension to

```
\shortcode{hr}
```

which in turn, calls the following macro that you have to define:

```
\shortcodehr
```

which you could, for instance, set like

```
\newcommand\shortcodehr{\par\medskip\hrule\par\medskip}
```

Alternatively, after loading `\usepackage{shortcode}`, you could simply declare the shortcode like this:

```
\DeclareShortcode{hr}{\par\medskip\hrule\par\medskip}
```

1.2.2 Not so simple: one argument.

Suppose now that you want to use a shortcode like this:

```
{{ icon github }}
```

to insert an icon for GitHub. This shortcode gets translated by the markdown extension to

```
\shortocode{icon}[github]
```

which in turn calls

```
\shortcodeicon[github]
```

This would be easy to define:

```
\RequirePackage{fontawesome5}
\newcommand\shortcodeicon[1][]{\faIcon{#1}}
```

1.2.3 Next step: named arguments, defaults, quoted strings

If you want a more nuanced approach, you could use a key-value approach, with a variable name of arguments. A shortcode like this, for instance

```
{{< faicon name=building >}} or {{<faicon syle=regular name=building >}}
```

(In any order). This gets translated by the markdown extension to

```
\shortcode{faicon}[name=building] or \shortcode{faicon}[name=building,style=regula
```

and then, when these are expanded in LaTeX, to

```
\shortcodefaicon[name=building] or \shortcodefaicon[name=building,style=regular]
```

but we still need to provide the LaTeX definitions. Here you can leverage a key-value package; I'm partial to `pgfkeys`. So you can define something like

```
\RequirePackage{fontawesome5}
\RequirePackage{pgfkeys}
\pgfkeys{/faicon/.is family, /faicon/.cd,
    name/.default={},
    name/.store in=\shtcIconName,
    style/.store in=\shtcIconStyle,
    style/.default=solid,
    name, style }
\newcommand\shortcodefaicon[1][]{\pgfkeys{/faicon/.cd,#1}\faIcon[\shtcIconStyle]{\
```

1.2.4 Managing quoted strings

In the example above, the shortcode is far more likely to come in quoted form:

```
{{< faicon name="building" >}} or {{<faicon syle="regular" name="building" >}}
```

which translates to

```
\shortcode{faicon}[name="building"] or \shortcode{faicon}[name="building",style="r
```

Often you will want to process the arguments minus the quotes, so there is a provision for removing quotes from arguments if they are present, the `.unquote and store in=` key handler.

```
\pgfkeys{/faicon/.is family, /faicon/.cd,
    name/.unquote and store in=\shtcIconName,
    style/.unquote and store in=\shtcIconStyle,
    style/.default=solid,
    style }
\newcommand\shortcodefaicon[1][]{\pgfkeys{/faicon/.cd,#1}\faIcon[\shtcIconStyle]{\
```

1.2.5 Simple environments

Some shortcodes have a closing shortcode:

```
{{< abstract >}}
lorem ipsum...
{{< /abstract >}}
```

This gets translated to

```
\shortcode{abstract}
lorem ipsum...
\closeshortcode{abstract}
```

which gets expanded to

```
\shortcodeabstract
lorem ipsum...
\closeshortcodeabstract
```

So a sensible definition would be

```
\newcommand{\shortcodeabstract}{\begin{abstract}}
\newcommand{\closeshortcodeabstract}{\end{abstract}}
```

1.2.6 From environments to commands

Occasionally, a pair of opening/closing shortodes have to be translated to a single LaTeX command:

```
{{< alert warning >}}
Lorem ipsum ...
{{< /alert >}}
```

You need some TeX treachery to transform the above to `\alertwarning{Lorem ipsum...}`. For starters, the above gets translated to

```
\shortcode{alert}[warning]
Lorem ipsum...
\closeshortode{alert}
```

which, when expanded, translates to

```
\shortcodealert[warning]
Lorem ipsum...
\closeshortodealert
```

So a good LaTeX definition would be:

```
\RequirePackage{alertmessage}
\def\shortcodealert[#1]#2\closeshortcode#3{\expandafter\csname alert#1\endcsname{#
```

For simpler shortcodes, without optional arguments, a similar trick works: say you want to use a `sidenote` shortcode:

```
In principle {{< sidenote >}}This is of course, the content of a sidenote{{< /side
```

Here, the content of the sidenote shortcode should be given as argument to the `\sidenote`command, To do that, you can use the following definition:

```
\def\shortcodesidenote#1\closeshortcode#2{\sidenote{#1}}
```

1.2.7 Managing unnamed quoted strings

Say you want to use the `theorem` shortcode above,

```
{{< theorem name="Chinese Remainder Theorem" >}} The groups $\mathbb{Z}/(ab)$ and
{{< /theorem >}}
```

To make thing worse, suppose you also would like to use the following form, both in the same document.

```
{{< theorem "Chinese Remainder Theorem" >}} The groups $\mathbb{Z}/(ab)$ and $\mat
{{< /theorem >}}
```

In this case, managing the quoted strings and the optional `name=` part becomes cumbersome. But here is a solution that uses `pgfkeys`.

```
\pgfkeys{
    /theorem/.is family, /theorem/.cd,
    name/.unquote and store in=\optionalStatementName,
    /handlers/first char syntax=true,
    /handlers/first char syntax/the character "/.initial=\unquoteandstore\optional
}
\newcommand\shortcodetheorem[1][]{\pgfkeys{/theorem,#1}\begin{theorem}[\optionalSt
\newcommand\closeshortcodetheorem{\end{theorem}}
```