1 Examples

Simple shortcode, no arguments, one command

Simple shortcode with one positional argument , and shortcodes with keyvalues .

Pair shortcodes: the theorem environment.

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

Shortcodes that transform its content to arguments, for intance sidenotes[1]. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

[1] Like this one

Default handling of unnamed and quoted strings:

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

More complex transformations: positional arguments and transform to command argument.

This information is important.

But this is a terrible mistake.

2 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.

2.1 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.

2.2 How to define your own LaTeX shortcodes

2.2.1 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}
```

2.2.2 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}}
```

2.2.3 Next step: named arguments, defaults, quoted strings

Where shortcodes get interesting is when you use them with named 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=regul
```

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 is where declaring shortcode named options can be useful:

```
\DeclareShortcodeOption{faicon}{name}{\faiconName}
\DeclareShortcodeOptionDefault{faicon}{style}{\faiconStyle}{solid}
\DeclareShortcode{faicon}{\faIcon[\faiconStyle]{\faiconName}}
```

The command `\DeclareShortcodeOption{shortcodename}{optionname}{\commandname}` scans the named optional arguments to the short code `shortcodename` and stores the value of the option in the command given as third argument. Note that `\DeclareShortcodeOption` does remove the outer quotation marks if they are present: both `name="value"` and `name=value` store `value` in the corresponding `\commandname`.

When an option may or may not be present but you want to provide a default value, use `\DeclareShortcodeOptionDefault`, which takes one additional argument, which is the default value to be used if the `optioname` is not present.

The code snippet above can take a lot of abuse; it will correcty process any of these:

```
{{< faicon name=apple }} -> \faicon[solid]{apple}
{{< faicon name="apple" }} -> \faicon[solid]{apple}
{{< faicon style=regular name="apple" }} -> \faicon[regular]{apple}
{{< faicon style="regular" name=apple }} -> \faicon[regular]{apple}
```

Technically speaking, `\DeclareShortcode` leverages the package `pgfkeys`. Internally, the package executes code like this:

```
\RequirePackage{fontawesome5}
\RequirePackage{pgfkeys}
\pgfkeys{/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]{\
```

Note that the removing of optional quoted strings is achieved through the `pgfkeys`handle `/.unquote and store in`, which is available for your own macros.

2.2.4 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}}
```

However, for opening and closing shortcodes pairs you can use the macro `\DeclareShortcodePair`:

```
\DeclareShortcodePair{abstract}{\begin{abstract}}{\end{abstract}}
```

And moreover, you can use the options mechanism for shortcode pairs: for a shortcode like this:

```
{{< theorem name="Chinese Remainder Theorem" >}}
Let $a$ and $b$ be two coprime integerers, then ...
{{< /theorem >}}
```

You can define

```
\DeclareShortcodePair{theorem}{\begin{theorem}[\optionalStatementName]}
                                {\end{theorem}}
\DeclareShortcodeOption{theorem}{name}{\optionalStatementName}
```

2.2.5 From environments to commands

Occasionally, you might need to take the contents of a open/close shortcode pair, and pass it to a LaTeX command, like this:

```
Testing sidenotes{{< sidenote >}}Like this one.{{< /sidenote >}} using Markdown.
```

This is achieved with the `DeclareShortcodePairCommand` construct.

```
\DeclareShortcodePairCommand{sidenote}{\sidenote}
```

Here, the second argument must end in a command that takes one parameter, which will be the content of the `sidenote` shortcodes pair.

For more complex construct, like this one, which has to be converted to an argument AND has a positional parameter, one might need to resort to more complex constructs:

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

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

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

which, when expanded, translates to

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

a good LaTeX definition would be:

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

2.2.6 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 >}}
```

But now, 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 using `pgfkeys` and the handler `first char syntax`:

```
\pgfkeys{
    /handlers/first char syntax=true,
    /handlers/first char syntax/the character "/.initial=\unquoteandstore\optional
}
\DeclareShortcodePair{theorem}{\begin{theorem}[\optionalStatementName]}
                               {\end{theorem}}
\DeclareShortcodeOption{theorem}{name}{\optionalStatementName}
```