Theme Documentation - Extended Shortcodes

Dillon included in documentation

03.03.2020 3245 words 16 minutes

CONTENTS

LoveIt theme provides multiple shortcodes on top of built-in ones in Hugo.

1 style

CHANGED 0.2.0

> **Note**
>
> Hugo **extended** version is necessary for `style` shortcode.

`style` is a shortcode to insert custom style in your post.

The `style` shortcode has two positional parameters.

The **first** one is the custom style content, which supports nesting syntax in SASS and `&` referring to this parent HTML element.

And the **second** one is the tag name of the HTML element wrapping the content you want to change style, and whose default value is `div` .

Example `style` input:

Markdown

```
{{< style "text-align:right; strong{color:#00b1ff;}" >}}
This is a **right-aligned** paragraph.
{{< /style >}}
```

The rendered output looks like this:

<style "text-align:right; strong{color:#00b1ff;}" >
This is a **right-aligned** paragraph.
</style >

2 link

NEW 0.2.0

`link` shortcode is an alternative to Markdown link syntax. `link` shortcode can provide some other features and can be used in code blocks.

NEW 0.2.10 The complete usage of local resource references is supported.

The `link` shortcode has the following named parameters:

- **href** *[required]* (**first** positional parameter)

  Destination of the link.

- **content** *[optional]* (**second** positional parameter)

  Content of the link, default value is the value of **href** parameter.

  Markdown or HTML format is supported.

- **title** *[optional]* (**third** positional parameter)

  `title` attribute of the HTML `a` tag, which will be shown when hovering on the link.

- **class** *[optional]*

  `class` attribute of the HTML `a` tag.

- **rel** *[optional]*

  Additional `rel` attributes of the HTML `a` tag.

Example `link` input:

Markdown

The rendered output looks like this:

- https://assemble.io
- mailto:contact@revolunet.com
- Assemble

Example `link` input with a title:

Markdown

```
{{< link "https://github.com/upstage/" Upstage "Visit Upstage!" >}}
Or
{{< link href="https://github.com/upstage/" content=Upstage title="Visit Upstage!" >}}
```

The rendered output looks like this (hover over the link, there should be a tooltip):

Upstage

3 image

CHANGED 0.2.0

`image` shortcode is an alternative to `figure` shortcode. `image` shortcode can take full advantage of the dependent libraries of lazysizes and lightgallery.js.

NEW 0.2.10 The complete usage of local resource references is supported.

The `image` shortcode has the following named parameters:

- **src** *[required]* (**first** positional parameter)

  URL of the image to be displayed.

- **alt** *[optional]* (**second** positional parameter)

  Alternate text for the image if the image cannot be displayed, default value is the value of **src** parameter.

Markdown or HTML format is supported.

- **caption** *[optional]* (**third** positional parameter)

  Image caption.

  Markdown or HTML format is supported.

- **title** *[optional]*

  Image title that will be shown when hovering on the image.

- **class** *[optional]*

  `class` attribute of the HTML `figure` tag.

- **src_s** *[optional]*

  URL of the image thumbnail, used for lightgallery, default value is the value of **src** parameter.

- **src_l** *[optional]*

  URL of the HD image, used for lightgallery, default value is the value of **src** parameter.

- **height** *[optional]*

  `height` attribute of the image.

- **width** *[optional]*

  `width` attribute of the image.

- **linked** *[optional]*

  Whether the image needs to be hyperlinked, default value is `true`.

- **rel** *[optional]*

  Additional `rel` attributes of the HTML `a` tag, if **linked** parameter is set to `true`.

Example `image` input:

Markdown

```
{{< image src="/images/lighthouse.jpg" caption="Lighthouse (`image`)" src_s="/images/lighth
```

The rendered output looks like this:

Lighthouse (`image`)

4 admonition

The `admonition` shortcode supports **12** types of banners to help you put notice in your page.

Markdown or HTML format in the content is supported.

Note

A **note** banner

Abstract

An **abstract** banner

Info

A **info** banner

Tip

A **tip** banner

Success

A **success** banner

Question

A **question** banner

Warning

A **warning** banner

Failure

A **failure** banner

Danger

A **danger** banner

Bug

A **bug** banner

Example

An **example** banner

Quote

A **quote** banner

The `admonition` shortcode has the following named parameters:

- **type** *[optional]* (**first** positional parameter)

  Type of the `admonition` banner, default value is `note`.

- **title** *[optional]* (**second** positional parameter)

  Title of the `admonition` banner, default value is the value of **type** parameter.

- **open** *[optional]* (**third** positional parameter) CHANGED 0.2.0

  Whether the content will be expandable by default, default value is `true`.

Example `admonition` input:

Markdown

```
{{< admonition type=tip title="This is a tip" open=false >}}
A **tip** banner
{{< /admonition >}}
Or
{{< admonition tip "This is a tip" false >}}
A **tip** banner
{{< /admonition >}}
```

The rendered output looks like this:

> **This is a tip**

5 mermaid

mermaid is a library helping you to generate diagram and flowcharts from text, in a similar manner as Markdown.

Just insert your mermaid code in the `mermaid` shortcode and that's it.

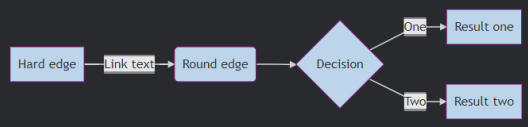
| 5.1 Flowchart

Example **flowchart** `mermaid` input:

Markdown

```
{{< mermaid >}}
graph LR;
    A[Hard edge] -->|Link text| B(Round edge)
    B --> C{Decision}
    C -->|One| D[Result one]
    C -->|Two| E[Result two]
{{< /mermaid >}}
```

The rendered output looks like this:

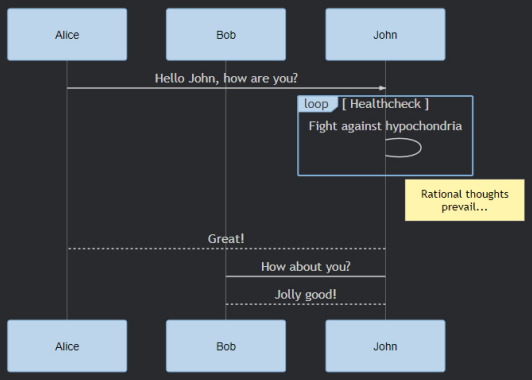


| 5.2 Sequence Diagram

Example **sequence diagram** `mermaid` input:

Markdown

The rendered output looks like this:

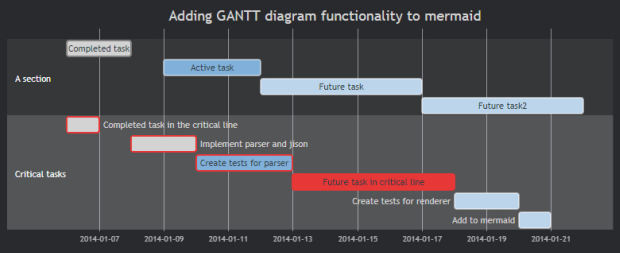


| 5.3 GANTT

Example **GANTT** `mermaid` input:

Markdown

The rendered output looks like this:



| 5.4 Class Diagram

Example **class diagram** `mermaid` input:

Markdown

The rendered output looks like this:

5.5 State Diagram

Example **state diagram** `mermaid` input:

Markdown

```
{{< mermaid >}}
stateDiagram
    [*] --> Still
    Still --> [*]
    Still --> Moving
    Moving --> Still
    Moving --> Crash
    Crash --> [*]
{{< /mermaid >}}
```

The rendered output looks like this:

5.6 Git Graph

Example **git graph** `mermaid` input:

Markdown

The rendered output looks like this:

5.7 Pie

Example **pie** `mermaid` input:

Markdown

```
{{< mermaid >}}
pie
    "Dogs" : 386
    "Cats" : 85
    "Rats" : 15
{{< /mermaid >}}
```

The rendered output looks like this:

6 echarts

ECharts is a library helping you to generate interactive data visualization.

The basic chart types ECharts supports include line series, bar series, scatter series, pie charts, candle-stick series, boxplot series for statistics, map series, heatmap series, lines series for directional information, graph series for relationships, treemap series, sunburst series, parallel series for multi-dimensional data, funnel series, gauge series. And it's extremely easy to create a combinition of them with ECharts.

Just insert your ECharts option in `JSON` / `YAML` / `TOML` format in the `echarts` shortcode and that's it.

Example `echarts` input in `JSON` format:

JSON

The same in `YAML` format:

YAML

The same in `TOML` format:

TOML

The rendered output looks like this:

The `echarts` shortcode has also the following named parameters:

- **width** *[optional]* (**first** positional parameter)

  NEW 0.2.0 Width of the data visualization, default value is `100%`.

- **height** *[optional]* (**second** positional parameter)

  NEW 0.2.0 Height of the data visualization, default value is `30rem`.

7 mapbox

NEW 0.2.0

Mapbox GL JS is a JavaScript library that uses WebGL to render interactive maps from vector tiles and Mapbox styles.

The `mapbox` shortcode has the following named parameters to use Mapbox GL JS:

- **lng** *[required]* (**first** positional parameter)

  Longitude of the inital centerpoint of the map, measured in degrees.

- **lat** *[required]* (**second** positional parameter)

  Latitude of the inital centerpoint of the map, measured in degrees.

- **zoom** *[optional]* (**third** positional parameter)

  The initial zoom level of the map, default value is `10`.

- **marked** *[optional]* (**fourth** positional parameter)

  Whether to add a marker at the inital centerpoint of the map, default value is `true`.

- **light-style** *[optional]* (**fifth** positional parameter)

  Style for the light theme, default value is the value set in the front matter or the site configuration.

- **dark-style** *[optional]* (**sixth** positional parameter)

  Style for the dark theme, default value is the value set in the front matter or the site configuration.

- **navigation** *[optional]*

  Whether to add NavigationControl, default value is the value set in the front matter or the site configuration.

- **geolocate** *[optional]*

  Whether to add GeolocateControl, default value is the value set in the front matter or the site configuration.

- **scale** *[optional]*

  Whether to add ScaleControl, default value is the value set in the front matter or the site configuration.

- **fullscreen** *[optional]*

  Whether to add FullscreenControl, default value is the value set in the front matter or the site configuration.

- **width** *[optional]*

  Width of the map, default value is `100%`.

- **height** *[optional]*

  Height of the map, default value is `20rem`.

Example simple `mapbox` input:

Markdown

```
{{< mapbox 121.485 31.233 12 >}}
Or
{{< mapbox lng=121.485 lat=31.233 zoom=12 >}}
```

The rendered output looks like this:

Example `mapbox` input with the custom style:

Markdown

```
{{< mapbox -122.252 37.453 10 false "mapbox://styles/mapbox/navigation-preview-day-v4" "map
Or
{{< mapbox lng=-122.252 lat=37.453 zoom=10 marked=false light-style="mapbox://styles/mapbox
```

The rendered output looks like this:

8 music

The `music` shortcode embeds a responsive music player based on APlayer and MetingJS.

There are three ways to use it the `music` shortcode.

| 8.1 Custom Music URL

NEW 0.2.10 The complete usage of local resource references is supported.

The `music` shortcode has the following named parameters by custom music URL:

- **server** *[required]*

  URL of the custom music.

- **name** *[optional]*

  Name of the custom music.

- **artist** *[optional]*

  Artist of the custom music.

- **cover** *[required]*

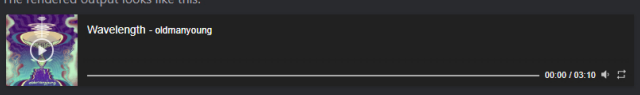
  URL of the custom music cover.

Example `music` input by custom music URL:

Markdown

```
{{< music url="/music/Wavelength.mp3" name=Wavelength artist=oldmanyoung cover="/images/Wav
```

The rendered output looks like this:



| 8.2 Music Platform URL Automatic Identification

The `music` shortcode has one named parameter by music platform URL automatic identification:

- **auto** *[required]* (**first** positional parameter)

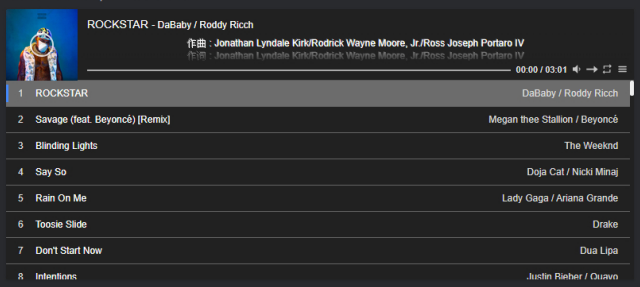
  URL of the music platform URL for automatic identification, which supports `netease`, `tencent` and `xiami` music platform.

Example `music` input by music platform URL automatic identification:

Markdown

```
{{< music auto="https://music.163.com/#/playlist?id=60198" >}}
Or
{{< music "https://music.163.com/#/playlist?id=60198" >}}
```

The rendered output looks like this:



8.3 Custom Server, Type and ID

The `music` shortcode has the following named parameters by custom music platform:

- **server** *[required]* (**first** positional parameter)

  [`netease` , `tencent` , `kugou` , `xiami` , `baidu`]

  Music platform.

- **type** *[required]* (**second** positional parameter)

  [`song` , `playlist` , `album` , `search` , `artist`]

  Type of the music.

- **id** *[required]* (**third** positional parameter)

  Song ID, or playlist ID, or album ID, or search keyword, or artist ID.

Example `music` input by custom music platform:

Markdown

```
{{< music server="netease" type="song" id="1868553" >}}
Or
{{< music netease song 1868553 >}}
```

The rendered output looks like this:

8.4 Other Parameters

The `music` shortcode has other named parameters applying to the above three ways:

- **theme** *[optional]*

  CHANGED 0.2.0 Main color of the music player, default value is `#448aff` .

- **fixed** *[optional]*

  Whether to enable fixed mode, default value is `false` .

- **mini** *[optional]*

  Whether to enable mini mode, default value is `false` .

- **autoplay** *[optional]*

  Whether to autoplay music, default value is `false` .

- **volume** *[optional]*

  Default volume when the player is first opened, which will be remembered in the browser, default value is `0.7` .

- **mutex** *[optional]*

  Whether to pause other players when this player starts playing, default value is `true` .

The `music` shortcode has the following named parameters only applying to the type of music list:

- **loop** *[optional]*

  [`all` , `one` , `none`]

  Loop mode of the music list, default value is `none` .

- **order** *[optional]*

  [`list` , `random`]

  Play order of the music list, default value is `list` .

- **list-folded** *[optional]*

  Whether the music list should be folded at first, default value is `false` .

- **list-max-height** *[optional]*

  Max height of the music list, default value is `340px` .

9 bilibili

CHANGED 0.2.0

The `bilibili` shortcode embeds a responsive video player for bilibili videos.

When the video only has one part, only the BV `id` of the video is required, e.g.:

Code

```
https://www.bilibili.com/video/BV1Sx411T7QQ
```

Example `bilibili` input:

Markdown

```
{{< bilibili BV1Sx411T7QQ >}}
Or
{{< bilibili id=BV1Sx411T7QQ >}}
```

The rendered output looks like this:

When the video has multiple parts, in addition to the BV `id` of the video, `p` is also required, whose default value is `1`, e.g.:

Code

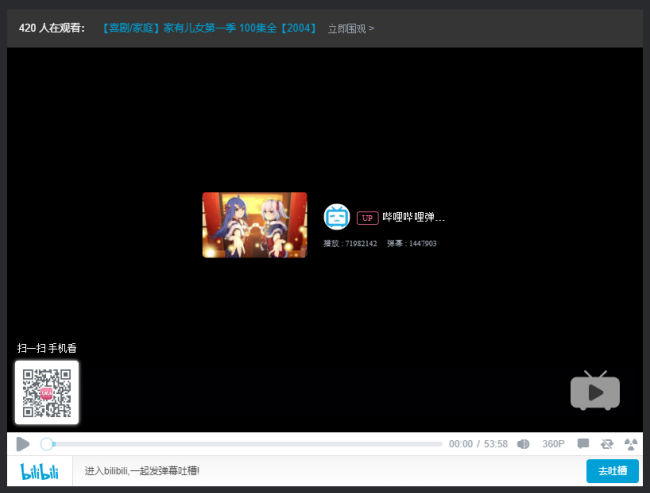
```
https://www.bilibili.com/video/BV1TJ411C7An?p=3
```

Example `bilibili` input with `p`:

Markdown

```
{{< bilibili BV1TJ411C7An 3 >}}
Or
{{< bilibili id=BV1TJ411C7An p=3 >}}
```

The rendered output looks like this:



10 typeit

The `typeit` shortcode provides typing animation based on TypeIt.

Just insert your content in the `typeit` shortcode and that's it.

| 10.1 Simple Content

Simple content is allowed in `Markdown` format and **without** rich block content such as images and more...

Example `typeit` input:

Markdown

```
{{< typeit >}}
This is a *paragraph* with **typing animation** based on [TypeIt](https://typeitjs.com/)...
{{< /typeit >}}
```

The rendered output looks like this:

Alternatively, you can use custom **HTML tags**.

Example `typeit` input with `h4` tag:

Markdown

```
{{< typeit tag=h4 >}}
This is a *paragraph* with **typing animation** based on [TypeIt](https://typeitjs.com/)...
{{< /typeit >}}
```

The rendered output looks like this:

| 10.2 Code Content

Code content is allowed and will be highlighted by named parameter `code` for the type of code language.

Example `typeit` input with `code`:

Markdown

```
{{< typeit code=java >}}
public class HelloWorld {
    public static void main(String []args) {
        System.out.println("Hello World");
    }
}
{{< /typeit >}}
```

The rendered output looks like this:

| 10.3 Group Content

All typing animations start at the same time by default. But sometimes you may want to start a set of `typeit` contents in order.

A set of `typeit` contents with the same value of named parameter `group` will start typing animation in sequence.

Example `typeit` input with `group`:

Markdown

```
{{< typeit group=paragraph >}}
**First** this paragraph begins
{{< /typeit >}}

{{< typeit group=paragraph >}}
**Then** this paragraph begins
{{< /typeit >}}
```

The rendered output looks like this:

11 script

NEW 0.2.8

`script` is a shortcode to insert custom **Javascript** in your post.

> **Note**
>
> The script content can be guaranteed to be executed in order after all third-party libraries are loaded. So you