cli-press-book

Table of Contents

Introduction to *Pressdown*

Pressdown is an extension of Markdown Extra that adds special processing directives to handle the specific needs of documentation tasks. Some of the directives help reduce markup, and others allow for easily escaping Twig, Markdown, and ***Pressdown*** directives. Most of all, though, ***Pressdown*** has lots of directives to make building your documentation easy and to make it beautiful.

> ***Pressdown*** **has lots of directives to make building your documentation easy and to make it beautiful.**

Pull Quotes

The first directive we'll take a look at is the Pull Quote Directive - `pq{content}(placement)` . If you want to bring attention to an important phrase or idea on a page, you can use a pull quote to highlight it. They are often used to grab the reader and draw them in to a specific section, to prevent them from flipping past. As you may have guessed, `content` should be the word or phrase you want to use, and `placement` is either 'right' or 'left' and optionally the name of an anchor to instruct ***Pressdown*** how and where to float it. Simply use the Pull Quote Directive wherever the phrase would normally occur in the flow of the document. It will appear as normal in that location and also appear floated either to the left or the right. As an example, here's the markup used to display the pull quotes on this page.

```
pq{{{ pressDown }} has lots of directives to make building your documentation easy and to make it beautiful.}(left)

pq{{{ pressDown }} has an even better way to create figures and captions.}(right far-away)
```

In the example above, `{{ pressDown }}` is a Twig expression for the press variable `pressDown` . You'll find more information about it below.

The first example on this page appears nearly in the same place as the original text. What if you wanted to make it appear a bit further away? In the `placement` section of the directive, add the name of a Pull Quote Anchor Directive. The second pull quote on this page actually comes from a quote on the next one! It is anchored with the following: `pqa{far-away}` to the end of this paragraph.

The Classed Block Directive

> ***Pressdown*** **has an even better way to create figures and captions.**

One main feature is the Classed Block Directive - `@{@tag-class content tag@}` . This allows you to briefly create HTML blocks with a specific assigned class. The `tag` can be replaced by a tag name, or one or two letters that represent the following tags: a, b(lockquote), c(ode), f(igure), f(ig)c(aption), d(iv), l(i), o(l), p, s(pan), and u(l). The `class` can be replaced by a single class name or a series of class names separated by a `.` . You can also leave it empty and no class name will be applied to the block at all. See Figure 1 below for an example.

With ***Pressdown*** this is how you make an element with a specific class:

```
this:
@{s-z I need to be a span with "z" class s@}

becomes this:
<span class="z">I need to be a span with "z" class</span>
```

Then you can use `.z` in your CSS files to do something cool.

Figure 1: The Classed Block Directive `{@x-c .+ x@}`

You can nest classed blocks and use Markdown Extra within them! Below is how I created the above sample. You'll see a Twig expression for the press variable `pressDown` . It itself evaluates to the classed block directive: `{@s-a Pressdown s@}` .

```
{@f-
  With {{ pressDown }} this is how you make an element with a specific class:

  ```none
  this:
  @{s-z I need to be a span with "z" class s@}

  becomes this:
  <span class="z">I need to be a span with "z" class</span>
  ```

  Then you can use `.z` in your CSS files to do something cool.
f@}
{@fc- Figure 1: The Classed Block Directive `{@x-c .+ x@}` fc@}
```

After Twig, ***Pressdown***, and Markdown Extra parsing, the final html markup looks like this:

```
<figure>With <span class="a">Pressdown</span> this is how you make an element with a specific class:
<pre><code class="language-none">
  this:
  @{s-z I need to be a span with "z" class s@}

  becomes this:
  <span class="z">I need to be a span with "z" class</span></code></pre>
Then you can use <code>.z</code> in your CSS files to do something cool.</figure>
<figcaption>Figure 1: The Classed Block Directive <code>{@x-c .+ x@}</code></figcaption>
```

Figures and Captions

If you were paying attention above, you noticed that Figure 1 uses classed blocks of the `f(igure)` and `f(ig)c(aption)` variety. ***CLI Press*** automatically styles figures and captions to look like insets as the one above. But as simple as the classed block directive is, ***Pressdown*** has an even better way to create figures and captions. The Figure Directive - `{=fig-name .+ fig=}(caption)?` .

Let's break down what that does, and how ***Pressdown*** helps you with it. The `.+` is the content that will go in the `figure` tag. The `name` should be a unique name for the figure. This name allows you to create a link to the figure from any file using the Figure Link Directive. Finally the `caption` is an optional caption tha will be added to the bottom of the figure, like the caption on the example above. And here's the best part: ***Pressdown*** will automatically number your figures for you, and prefix your caption with "Figure X:". Don't worry about renumbering figures. It's handled.

To see the directive in action, here's an example of how to create *Figure 1* using the Figure Directive.

```
{=fig-CBD
  With {{ pressDown }} this is how you make an element with a specific class:

  ```none
  this:
  @{s-z I need to be a span with "z" class s@}

  becomes this:
  <span class="z">I need to be a span with "z" class</span>
  ```

  Then you can use `.z` in your CSS files to do something cool. fig=}(The Classed Block Directive `{@x-c .+ x@}`)
```

And here's that ***Pressdown*** in action:

With ***Pressdown*** this is how you make an element with a specific class:

```
this:
@{s-z I need to be a span with "z" class s@}

becomes this:
<span class="z">I need to be a span with "z" class</span>
```

Then you can use `.z` in your CSS files to do something cool.

Figure 2: The Classed Block Directive `{@x-c .+ x@}`

Now suppose you want to refer to this figure later or earlier in the document. ***Pressdown*** can create a link for you that will automatically reference the proper number of the figure, regardless of where, or what order, it was defined. Just use the name you provided in the Figure Directive and ***Pressdown*** will handle the rest. This magic is accomplished with the Figure Link Directive: . The `name` must match the one in the Figure Directive. The `link text` , if defined, will be appended to the default link text of "Figure X". For example `f{Classed Block Directive}(CBD)` , would create a link that says "Figure 1: Classed Block Directive". If it were empty, the link would instead simply say "Figure 1".