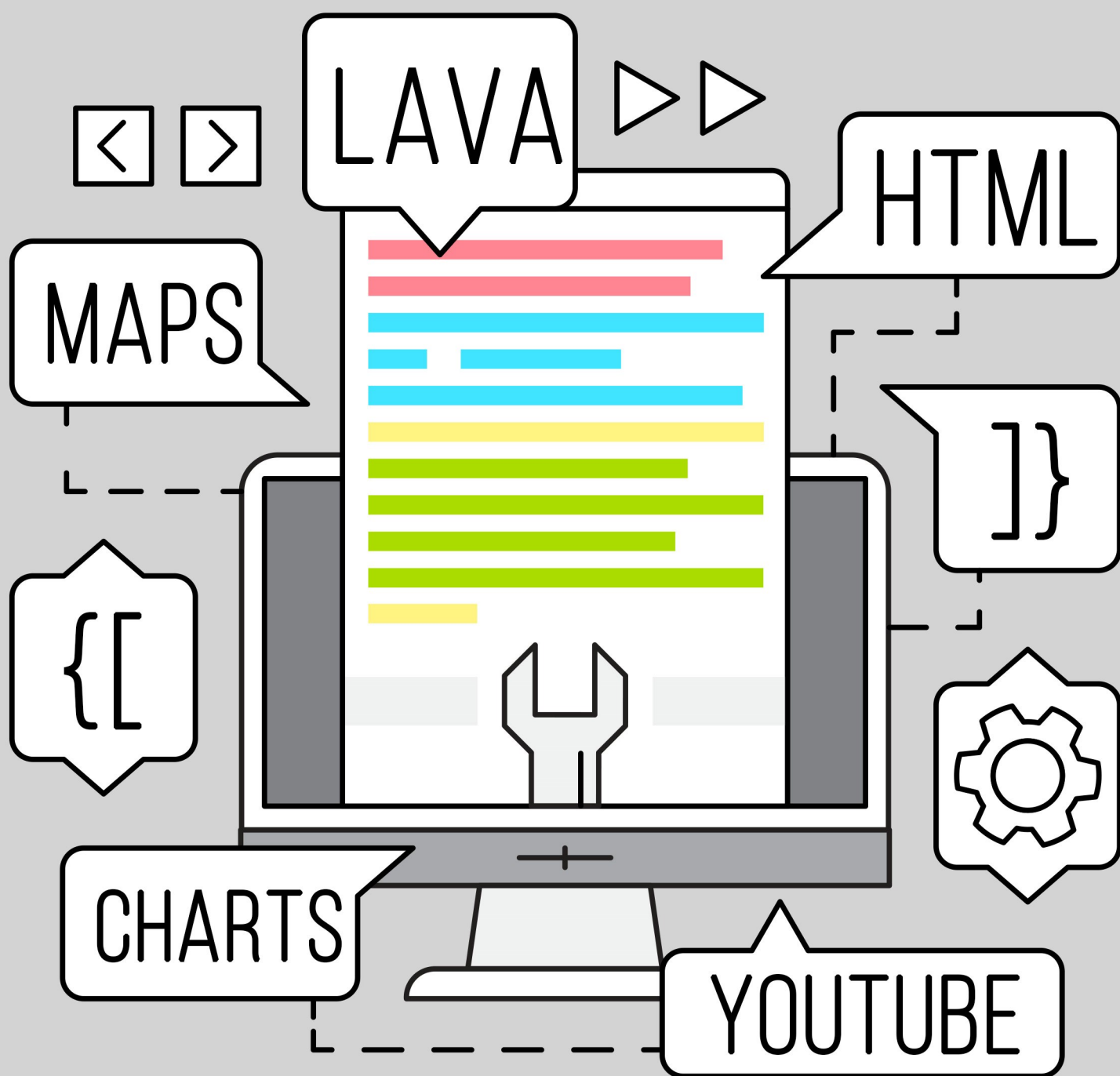


THE LONG & SHORT ON **SHORTCODES**



Getting Started

Shortcodes are a way to make Lava simpler and easier to read. They allow you to replace a simple Lava tag with a complex template written by a Lava specialist. This allows you to do some really powerful things without having to know how everything works.

While shortcodes are super powerful, they're also easy to develop. Everything you'll need to get up and running is right here in this short manual.

Types of Shortcodes

There are only two types of shortcodes: inline and block. Selecting the correct type is important as changing them in the future will break other people's Lava templates. Both short code types use the syntax: `{[shortcode]}`

Inline Shortcodes

The first type is an 'inline' shortcode. This is a simple shortcode that does not require an end tag. Just reference the shortcode and provide a list of parameters and you're done!

Inline Example:

```
{[ youtube id:'8kpHK4YIwY4' showinfo:'false' controls:'false' ]}
```

Block Shortcodes

The second type of shortcode is the 'block' type. Like other Lava commands it has both a start and an end tag. The content you provide inside the tags will be passed to the shortcode for use in its template.

Block Example:

```
{[ parallax image:'http://cdn.wonderfulengineering.com/wp-content/uploads/2014/09/star-wars-wallpaper-4.jpg' speed:'0.2' height:'400px' position:'top' contentpadding:'20px' ]}  
<h1>Hello World</h1>  
{[ endparallax ]}
```

Authoring Shortcodes

Writing shortcodes is simple. They're just Lava templates. Below are a few points to keep in mind as you write your own shortcodes.

Select the Proper Shortcode Type

As we already mentioned it's very important that you select the proper type of shortcode before you start using it. Changing it in the future means having to update all of your Lava templates that use it. The biggest factor in your decision of which type to use will be how much data your shortcode will pass and if any of the data has repeating sections. Let's consider a couple examples.

The parallax shortcode needs to access a large amount of HTML to place inside the parallax section. While this could be passed through a standard parameter `{[parallax content:'...}` this would be limiting in a number of ways. For example, the content could not have any line breaks and it would not be allowed to have any single quotes.

Consider next the googlemap place holder. It allows you to enter a variable number of map points. Again, you could use a single inline parameter, but this would be very difficult for the person who wanted to use your shortcode.

Finally, think about the youtube shortcode. In this example only a small amount of configuration is required, so an inline shortcode works great.

It comes down to how much data is being passed by the shortcode. For larger amounts of data, use the block shortcode type. For lesser amounts, use inline.

Configuring Parameters

You can configure parameters when setting up your shortcodes. While you technically don't need to configure parameters, any parameter that another person adds to the shortcode will be passed to your Lava template. Configuring the parameters when you write your shortcode provides two advantages.

1. Configuring the parameter guarantees that it will exist when it's passed to your Lava. This means you don't have to check for its existence. Instead, if you need to check if the parameter has a value, you can do a simple check to see if it's empty.
2. Configuring a parameter allows you to provide a default value. Again, this simplifies your Lava and reduces the number of lines of code you'll need to write.

The Lava you develop for your shortcode is passed in all of the parameters found on

the shortcode tag as well as any parameters you defined as the author. Rock also provides an additional parameter 'uniqueid' which is a unique identifier for you to use for things like CSS id's and variables in your Javascript. It takes the form of 'id-4535a4a9-1b82-4d5b-9120-8c9c41eefcd6'. You don't need to use this identifier, it's just provided as a convenience to you.

Heads Up!

Your parameter keys need to be all lowercase for them to work properly; this is typical practice in Lava but is enforced here. Keys with uppercase characters will always return the default value that you provide in the shortcode definition and will not be able to be set from the HTML block where you are calling the shortcode from.

Enabled Lava Commands

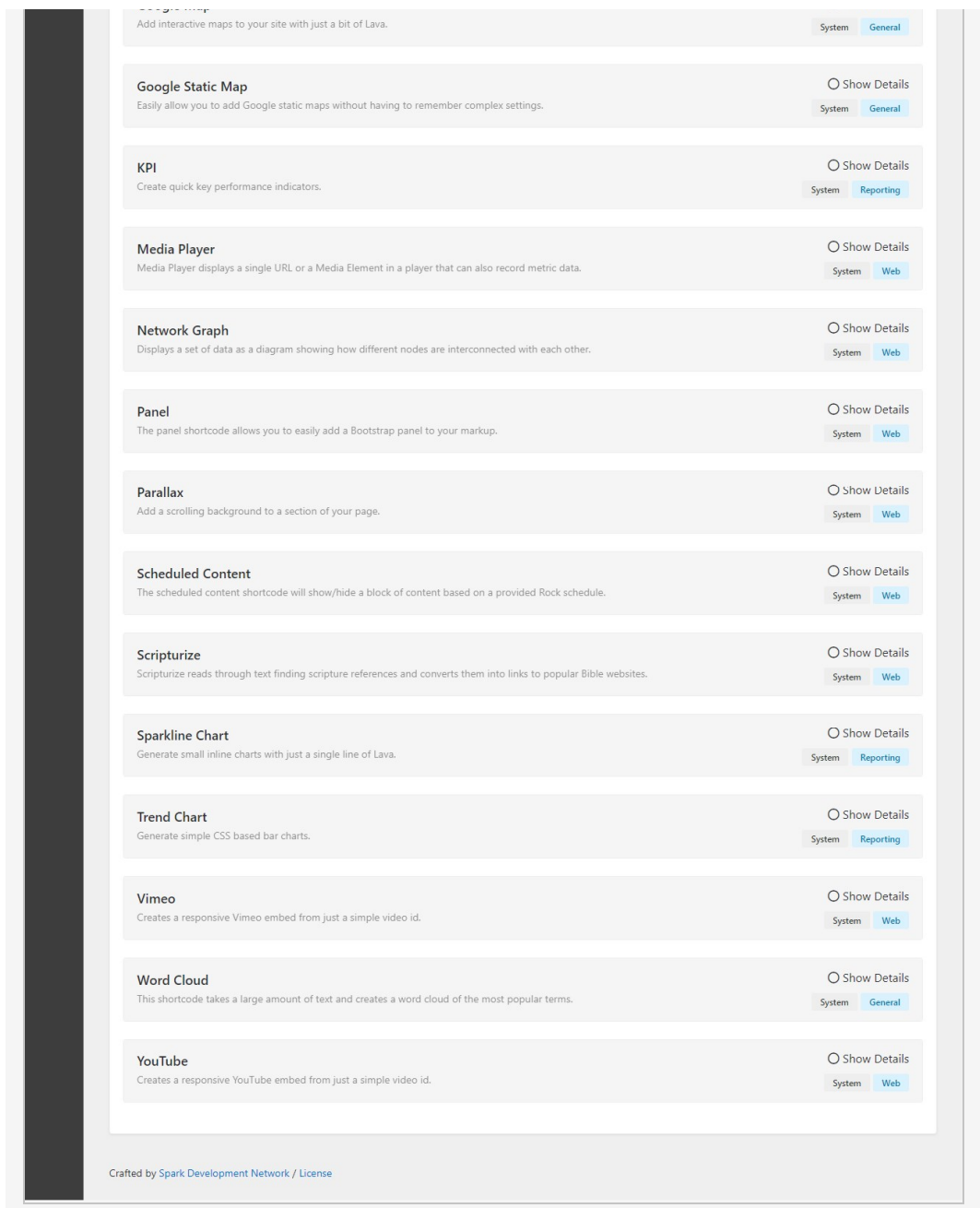
You can enable certain Lava commands to be able to run within your shortcode. These commands do not need to be configured in the source block to run. For instance, if your shortcode is being used on a web page and requires the RockEntity command, the HTML Block does not need to have the command enabled if you've configured access on your shortcode.

Adding Lava Shortcodes in Rock

Lava Shortcodes can be found at `Admin Tools > CMS Configuration > Lava Shortcodes`.

The screenshot shows the 'Lava Shortcodes' configuration page in the Rock CMS Admin interface. The page has a dark sidebar on the left with navigation icons. The main content area is titled 'Lava Shortcodes' and includes a breadcrumb trail: 'Home > Settings > CMS > Lava Shortcodes'. Below the title, there is a 'Show Inactive' toggle and a dropdown menu set to 'All Shortcodes'. The page lists several shortcodes, each with a description and a 'Show Details' button. The shortcodes listed are:

- Accordion**: Allows you to easily create a Bootstrap accordion control. (System, General)
- AI Completion (Experimental)**: Processes a completion using an AI provider. The APIs that are called by this shortcode can be slow. It's recommended that they not be used on public facing websites. They're better used in back-end or batch processes. (System, AI)
- Chart**: Adding dynamic charts to a page can be difficult, even for an experienced Javascript developer. The chart shortcode allows anyone to create charts with just a few lines of Lava. (System, Reporting)
- Easy Pie Chart**: Lightweight plugin to render simple, animated and retina optimized pie charts. (System, Reporting)
- Follow Icon**: Add an icon with the ability to follow any entity with a click. (System)
- Google Heatmap**: Add an interactive visualization to depict the intensity of data at geographical points. (System, Reporting)
- Google Map**: (System)




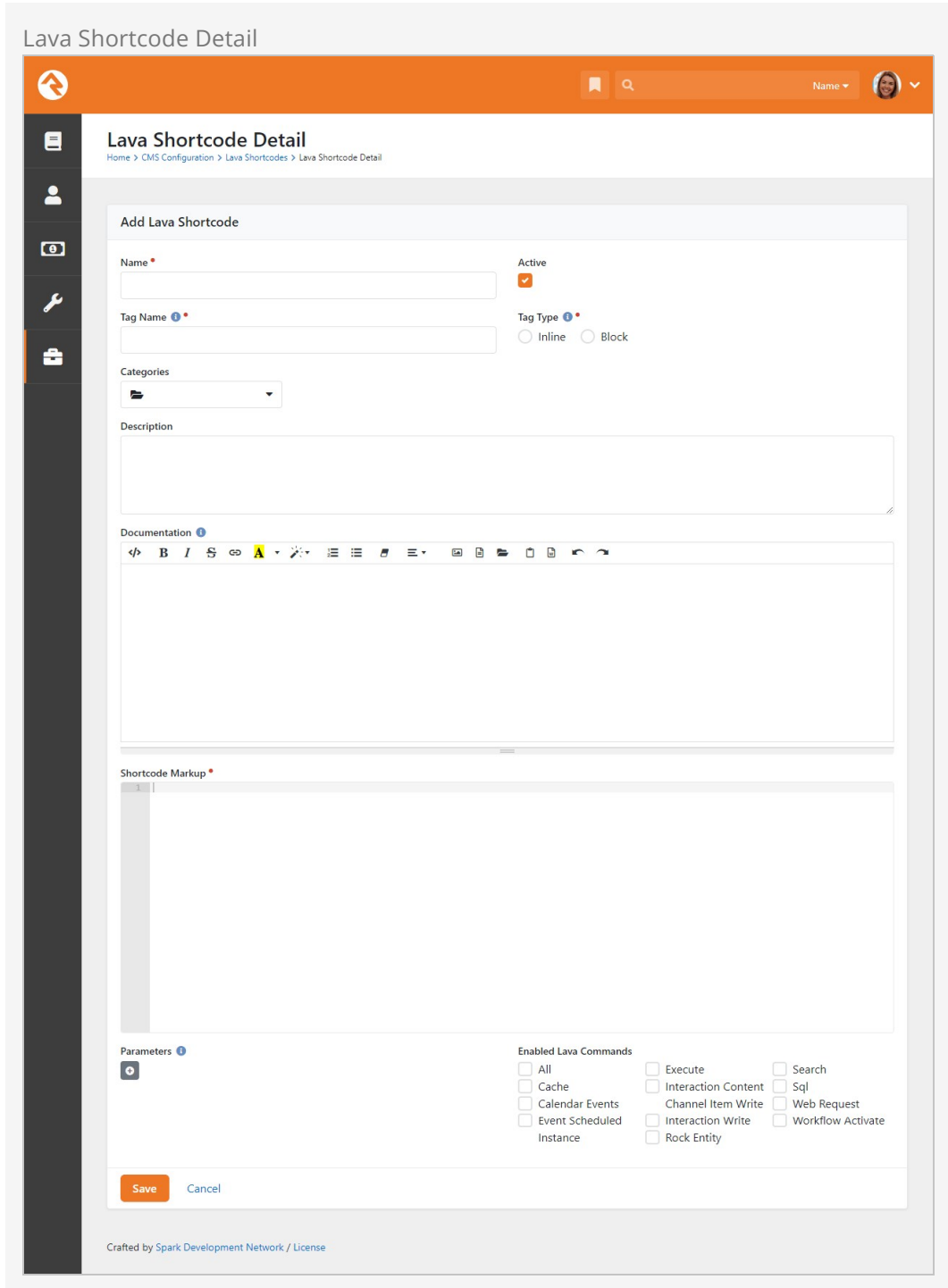
Pictured above, you can assign categories to your shortcodes. This lets you group shortcodes according to their function, and allows you to filter the list of shortcodes by category. A single shortcode can be assigned to multiple categories. The categories for a shortcode can be added from just the Lava Shortcode Detail page, pictured below.

We want to ensure the shortcode list is searchable in this manual. For this reason, we've listed below the shortcodes that are pictured above.

Name	Description
Accordion	Allows you to easily create a Bootstrap accordion control.
AI Completion (Experimental)	Processes a completion using a AI provider. The APIs that are called by this shortcode can be slow. It's recommended that they not be used on public facing websites. They're better used in back-end or

Name	Description
	batch processes.
Chart	Adding dynamic charts to a page can be difficult, even for an experienced JavaScript developer. The chart shortcode allows anyone to create charts with just a few lines of Lava.
Easy Pie Chart	Lightweight plugin to render simple, animated and retina optimized pie charts.
Follow Icon	Add an icon with the ability to follow any entity with a click.
Google Heatmap	Add an interactive visualization to depict the intensity of data at geographical points.
Google Map	Add interactive maps to your site with just a bit of Lava.
Google Static Map	Easily allow you to add Google static maps without having to remember complex settings.
KPI	Create quick key performance indicators.
Media Player	Media Player displays a single URL or a Media Element in a player that can also record metric data.
Network Graph	Displays a set of data as a diagram showing how different nodes are interconnected with each other.
Panel	The panel shortcode allows you to easily add a Bootstrap panel to your markup.
Parallax	Add a scrolling background to a section of your page.
Scheduled Content	The scheduled content shortcode will show/hide a block of content based on a provided Rock schedule.
Scripturize	Scripturize reads through text finding scripture references and converts them into links to popular Bible websites.
Sparkline Chart	Generate small inline charts with just a single line of Lava.
Trend Chart	Generate simple CSS based bar charts.
Vimeo	Creates a responsive Vimeo embed from just a simple video id.
Word Cloud	This shortcode takes a large amount of text and creates a word cloud of the most popular terms.
YouTube	Creates a responsive YouTube embed from just a simple video id.

Click *Show Details* for any of the shortcodes to view its summary, code and parameters. To add a new shortcode, press the  button. This will take you to the Lava Shortcode Detail screen.



This screen is where you designate which type of shortcode you're creating, which parameters to use, the actual Lava markup code, and any Lava commands you want enabled. Remember, it's always a good idea to supply a description for the benefit of other users. Click `Save` when you're done. Super easy!

The Power of Shortcode Blocks

Shortcode blocks are very powerful as they allow you to provide advanced configuration options. Rock does a lot of the heavy lifting for you to help parse the configuration.

Block Content

The content (everything between the start and end tags) that the user adds to your shortcode is accessible to your template using the `{{ blockContent }}` variable. For instance, the parallax shortcode would use this variable to place the content inside the parallax section of a page.

Lava will be run across the markup of the `{{ blockContent }}`. You can disable this behavior by adding a parameter to your shortcode with the key of `disablelavamerge` and a value of 'true'.

Block Configuration

The block configuration can also be placed in the block contents using the syntax:

```
[[ itemname parameter:'value' parmeter2:'value' ]] content [[ enditemname]]
```

When Rock is parsing your shortcode, these options will be parsed and removed from the `{{ blockContent }}`. They will then be provided to your template as variables. How these variables are created is super powerful and will help reduce the amount of Lava that is created. Let's dig deeper to see how this works by looking at some examples.

As you've probably seen, the syntax of the googlemap shortcode is:

```
{[ googlemap height:'400px' scrollwheel:'false' draggable:'false' ]}
  [[ marker location:'33.640705,-112.280198' title:'Spark Global Headquarters']]
  <strong>Spark Global Headquarters</strong><p>
  It's not as grand as it sounds.<br>
  <img src='https://rockrms.blob.core.windows.net/misc/spark-logo.png' width="179" height="47" />
  [[ endmarker ]]
  [[ marker location:'33.52764, -112.262571']] [[ endmarker ]]
{[ endgooglemap ]}
```

After parsing, your template would have access to the 'height, scrollwheel, draggable' variables just like any shortcode. It would also be provided an array of 'markers' using the variable name 'markers'. (Yes, we know... clever...) Each 'marker' in the 'markers' array would have a variable for 'location, title, ...' It would also have a variable called 'content' which would contain the text within the start and end configuration tags (in this case, the HTML for the Google Maps Info Window).

Single vs Repeating Configuration Items

You know which configuration items can have multiple values (such as the markers on the maps) and which should only have one (such as the map style item), but Rock does not. To deal with this, Rock will provide an array of items AND a single value for each type of configuration item it finds. For example, in the case of markers there will be a 'markers' array variable and a 'marker' variable. The 'marker' variable will have the first item in the array. Your Lava template can use either based on whether it should be a single configuration or multi-value item.

Passing In Objects

Version Note

Passing in objects requires Rock v10 or above.

Up until this point we've seen how we can pass primitive types (strings, numbers, booleans, etc) into our shortcodes. What happens though when we want to pass in a Lava variable? For instance, what if we had a collection of groups, loaded from an entity command, and we want our shortcode to format them. No problem, simply drop the quotes and Lava will pass the contents of the variable into the key you provide. Let's see an example.

```
{% group where:'GroupId == 25'%}  
  {[ grouplistformat groups:groupItems ]}  
{% endgroup %}
```

Just keep in mind that there are some limitations to passing in objects. Specifically, using the "dot" notation to provide an object (see example below) will not work for you.

Will Not Work

```
{% group where:'GroupId == 25'%}  
  {[ grouplistformat groups:ParentObject.groupItems ]}  
{% endgroup %}
```