Using Compass

Using Compass

Compass is an open-source Sass framework. It offers:

- A simplified management of style libraries (such as Bootstrap and others).
- A collection of mixins and functions (browser handling, etc.)
- Tools to automatically generate sprites
- ..and much more!

Installation

Compass (and Sass) require the installation of the Ruby language on your machine.

Here are some installers:

- Windows: Download http://rubyinstaller.org/ and launch it.
- Mac OS X: Install Homebrew (http://brew.sh/), then type "`brew install ruby`".
- Linux systems: In the command line, type "`sudo apt-get install ruby1.9.1`".

Once Ruby is installed on your machine, install Compass – which will in turn install Sass. Go to your Ruby command line and type "`gem install compass`".

Starting a project project

You are now ready to create your first project:

1. Go to the `/themes` folder of your local installation of PrestaShop.
2. Open a Ruby command line window for this folder, and type "`compass create`".
   Compass will automatically detect or create the `config.rb` configuration file that is necessary for the compilation of your project.

To make sure that `.scss` files are automatically compiled, you can type this Ruby command: "`compass watch`".

To use Compass, you must import it within your theme's main CSS file:

```
@import "compass";
```

This will also import the following libraries:

- CSS3: to manage the various browsers.
- Typography: mixins for text display.
- Utilities: mixin tools to manage colors, sprites, tables, etc.
- Layout (optional): to manage grids, dimensions, etc.
- Reset: to get a "Reset" CSS starter file.

Not a command line fan?

If you would rather not spend your time in the command line, Scout is the cross-platform application for you: it is a self-contained Ruby environment that gives you easy access to Compass and Sass.

Download it here: http://mhs.github.io/scout-app/

After you have installed it, you must:

1. Create your project at the root of your theme's folder.
2. Indicate the folder of your `.scss` files (Input folder).
3. Indicate the folder for your CSS files (Output folder).

Compass helpers

Colors / Color Stops

These color functions are useful for creating generic libraries that have to accept a range of inputs.

- adjust-lightness($color, $amount) : Adds $amount to $color's lightness value. $amount can be negative.
- adjust-saturation($color, $amount) : Adds $amount to $color's saturation value. $amount can be negative.
- scale-lightness($color, $amount) : Scales $color's lightness value by $amount. $amount can be negative.
- scale-saturation($color, $amount) : Scales $color's saturation value by $amount. $amount can be negative.
- shade($color, $percentage) : Darkens the $color by mixing it with black as specified by $percentage.
- tint($color, $percentage) : Lightens the $color by mixing it with white as specified by $percentage.

The color-stops helper provides a way to pass an arbitrary number of colors stops to the gradient mixins.

Any number of comma-delimited color stops can be passed, each color stop should take the form of a color followed by an optional stopping point (separated by a space).

Where stop values are not provided they will be inferred by assuming an equal distribution of colors between any specified locations.

Examples:

- color-stops(#FFF,#F00,#00C) -> #FFF 0%, #F00 50%, #00C 100%
- color-stops(#FFF, #F00 25%, #0C0, #00C) -> #FFF 0%, #F00 25%, #0C0 62.5%, #00C 100%
- color-stops(#FFF, #F00 5px, #0C0, #00C 25px) -> #FFF 0px, #F00 5px, #0C0 15px, #00C 25px

Constants

These helpers manipulate CSS Constants.

- opposite-position(left) => right
- opposite-position(top) => bottom
- opposite-position(center) => center
- opposite-position(top left) => bottom right
- opposite-position(center right) => center left

Cross Browser

Function: prefixed($prefix, $arg, ...)

Returns true if any of the arguments require the given prefix.

Function: prefix($prefix, $arg, ...)

Transforms the argument(s) into a representation for the rendering engine indicated by $prefix. Usually this means just adding a prefix, but in some cases, this may result in entirely different representations for the given rendering engine (E.g. linear-gradient).

Values that do not have a specific representation are passed through without being transformed.

Function shortcut

- -webkit($arg, ...) : This is a shortcut for calling prefix(-webkit, $arg, ...).
- -moz($arg, ...) :This is a shortcut for calling prefix(-moz, $arg, ...).
- -o($arg, ...) : This is a shortcut for calling prefix(-o, $arg, ...).
- -ms($arg, ...) : This is a shortcut for calling prefix(-ms, $arg, ...).
- -svg($arg, ...) : This is a shortcut for calling prefix(-svg, $arg, ...). Instead of adding a prefix, it returns a representation of the arguments using SVG to render them where it can.
- -pie($arg, ...) : It it used to get CSS3 PIE support where necessary.
- -css2($arg, ...) : This is a shortcut for calling prefix(-css2, $arg, ...). It is a kind of hack to sanitize the output of experimental code into a form that can be parsed by a css2.1 compliant parser. Usually this results in causing some functions to be omitted.
- css2-fallback($value, $css2-value) : This function returns a value that is normally $value, but is $css2-value when passed through the -css2() helper function. Many of the compass css3 mixins will create a css2 fallback value if the arguments have a css2 representation (gradients have a null css2 representation).

Font Files

The font-files function takes a list of arguments containing the path to each font files relative to your project's font directory.

```
@import "compass/css3";
@include font-face("Blooming Grove", font-files("examples/bgrove.ttf", "examples/bgrove.otf"));
.example {
  font-family: "Blooming Grove";
  font-size: 1.5em;
}
```

Image Dimension

- image-width($image): Returns the width of the image found at the path supplied by $image relative to your project's images directory.
- image-height($image): Returns the height of the image found at the path supplied by $image relative to your project's images directory.

Inline Data

- inline-image($image, $mime-type): Embeds the contents of an image directly inside your stylesheet, eliminating the need for another HTTP request. For small images, this can be a performance benefit at the cost of a larger generated CSS file.
  Like the image-url() helper, the path specified should be relative to your project's images directory.
- inline-font-files([$font, $format]*): Like the font-files() helper, but the font file is embedded within the generated CSS file.

Math

Available functions:

- pi()
- sin($number)
- cos($number)
- tan($number)
- e()
- logarithm($number, $base)
- sqrt($number)
- pow($number, $exponent)

Sprite

- sprite-map($glob, ...): Generates a css sprite map from the files matching the glob pattern. Uses the keyword-style arguments passed in to control the placement.
  Only PNG files can be made into css sprites at this time.
  The $glob should be glob pattern relative to the images directory that specifies what files will be in the css sprite.

```
$icons: sprite-map("icons/*.png");
background: $icons;
```

- sprite($map, $sprite, $offset-x, $offset-y): Returns the image and background position for use in a single shorthand property.
- sprite-map-name($map): Returns the name of a css sprite map The name is derived from the folder than contains the css sprites.
- sprite-file($map, $sprite): Returns the relative path (from the images directory) to the original file used when construction the sprite. This is suitable for passing to the image-width and image-height helpers.
- sprite-url($map): Returns a url to the sprite image.
- sprite-position($map, $sprite, $offset-x, $offset-y): Returns the position for the original image in the sprite. This is suitable for use as a value to background-position:

URL

- stylesheet-url($path, $only-path): Generates a path to an asset found relative to the project's css directory.
  Passing a true value as the second argument will cause pronly the path to be returned instead of a url() function
- font-url($path, $only-path): Generates a path to an asset found relative to the project's font directory.
  Passing a true value as the second argument will cause only the path to be returned instead of a url() function
- image-url($path, $only-path, $cache-buster): Generates a path to an asset found relative to the project's images directory.
  Passing a true value as the second argument will cause only the path to be returned instead of a url() function
  The third argument is used to control the cache buster on a per-use basis. When set to false no cache buster will be used. When a string, that value will be used as the cache buster.
- generated-image-url($path, $cache-buster: false): Generates a path to an image generated during compilation using the generated_image related configuration properties.
  Most users will never call this helper directly, it is primarily provided for use by plugins that need to generate paths to images they create. E.g. The sprite-url() helper calls this helper.
  The second argument is used to control the cache buster on a per-use basis. Defaults to false, meaning that no cache buster will be used. When a string, that value will be used as the cache buster, when true Compass will generate a cache-buster based on the image's modification time.

Managing sprite with Compass

Compass makes it really easy to create sprites from a bunch of images.

See the online tutorial here: http://compass-style.org/help/tutorials/spriting/

The CSS library

This library provides cross-browser mixins for CSS properties introduced in CSS3, for example `border-radius` and `text-shadow`.

It is automatically imported with Compass, but you can import that specific part of Compass with this line:

```
@import "compass/css3";
```

It has many available imports:

- Appearance – Specify the CSS3 appearance property.
- Background Clip – Specify the background clip for all browsers.
- Background Origin – Specify the background origin for all browsers.
- Background Size – Specify the background size for all browsers.
- Border Radius – Specify the border radius for all browsers.
- Box – This module provides mixins that pertain to the CSS3 Flexible Box.
- Box Shadow – Specify the box shadow for all browsers.
- Box Sizing – Specify the box sizing for all browsers.
- Columns – Specify a columnar layout for all browsers.

- Filter – Specify the (image) filter for all browsers.
- Font Face – Specify a downloadable font face for all browsers.
- Hyphenation – Mixin for breaking space and injecting hypens into overflowing text
- Images – Specify linear gradients and radial gradients for many browsers.
- Inline Block – Declare an element inline block for all browsers.
- Opacity – Specify the opacity for all browsers.
- CSS Regions – Specify CSS Regions for supported browsers.
- Text Shadow – Specify the text shadow for all browsers.
- Transform – Specify transformations for many browsers.
- Transition – Specify a style transition for all browsers.
- User Interface – Declare an element inline block for all browsers.

You can learn more about each on the official site: http://compass-style.org/reference/compass/css3/

The Typography library

This library provides some basic mixins for common text styling patterns.

It is automatically imported with Compass, but you can import that specific part of Compass with this line:

```
@import "compass/typography";
```

It has four available imports:

- Links – Tools for styling anchor links.
- Lists – Tools for styling lists.
- Text – Style helpers for your text.
- Vertical Rhythm – Create and maintain a vertical rhythm for your type.

You can learn more about each on the official site: http://compass-style.org/reference/compass/typography/

The Utilities library

This module provides some basic mixins for common styling patterns.

It is automatically imported with Compass, but you can import that specific part of Compass with this line:

```
@import "compass/utilities";
```

It has five available imports:

- Color – Utilities for working with colors.
- General – Generally useful utilities that don't fit elsewhere.
- Print - Control what gets printed
- Sprites – Sprite mixins.
- Tables – Style helpers for your tables.

You can learn more about each on the official site: http://compass-style.org/reference/compass/utilities/

The Layout module

This module provides tools to help you with page layout.

To use this library, you must explicitly import that part of Compass with this line:

```
@import "compass/layout";
```

It has three available imports:

- Grid Background - Generate fixed, fluid and elastic grid-images on-the-fly using css3 gradients
- Sticky Footer - Lay out your footer such that it sticks to the bottom of the page.
- Stretching - Style absolutely positioned elements such that they will stretch to fill their positioning parent.

You can learn more about each on the official site: http://compass-style.org/reference/compass/layout/

The Reset library

This module applies the global reset to your stylesheet by simply importing it.

To use this library, you must explicitly import that part of Compass with this line:

```
@import "compass/reset";
```

It has one available import: Compass Reset Utilities, which adds a reset CSS to your stylesheets.

It is based on Eric Meyer's Reset 2.0 code: http://meyerweb.com/eric/tools/css/reset/index.html

You can learn more about each on the official site: http://compass-style.org/reference/compass/reset/