SVG Logo and Text Effects 1.1.0 reference manual

true

Abstract

This is the reference manual for the **SVG_Logo_and_Text_Effects** plugin, version **1.1.0** for WordPress. **Insert text with visually stunning SVG effects into your WordPress site.**

Introduction

Insert text with visually stunning SVG effects into your WordPress site.

Thanks!

Thank you for downloading the *SVG Logo and Text Effects* plugin, version *1.1.0* for *WordPress*.

SVG Logo and Text Effects is designed to be easy to use. If you want to jump right in, read the *Quickstart guide* (`slate-quickstart.pdf`) instead.

This document is the complete reference manual. The information presented here is **not** essential if you're just getting started.

What is it?

SVG Logo and Text Effects (or SLATE for short) is a WordPress plugin by dashed-slug that helps you produce visually stunning text effects using *SVG shortcodes*.

You can pick and choose **fonts**, **colors**, **fill patterns**, and **filters** to create unique text effects that draw attention to your content. Reinforce a brand identity, enhance calls-to-action visually, or simply be more artistic with SVG text that you can edit without the need to resort to image-editing software.

Licensing

SVG Logo and Text Effects is distributed freely under the terms of GPL version 2 or later.

SVG text on the web

The SVG standard

Scalable Vector Graphics is a technology that lets you create graphics in terms of vectors, rather than pixels. Vector graphics have a number of advantages over traditional raster formats (JPEG, PNG, GIF, etc). The most cited one is the ability to zoom into graphics without any ugly pixelation effects.

SVG offers a number of useful features. For instance, it lets you transform and filter your graphics or text, and you can even define animations.

Existing since 1999, SVG is an robust and proven technology. Traditionally it has not been used in the web due to poor browser support. In recent years, browser support for SVG has drastically improved. Most browsers including *Mozilla Firefox*, *Google Chrome*, *Microsoft Internet Explorer*, *Opera*, *Safari* and *mobile device browsers* now have built-in support for SVG. A web developer can now insert eye-catching vector graphics directly into HTML markup.

Use cases

SVG text is useful for creating fancy, eye-catching, or decorative text. Draw attention to:

- calls to action (“buy now”, “subscribe”, etc)
- artistic text
- headers and titles
- logos
- any text where you would use a “display font” rather than a “text font”

Advantages of SVG text over raster-rendered text

- **Editable**: SVG text can be edited with a text editor, while rasterized text needs to be re-rendered every time.
- **Selectable**: Unlike pre-rendered raster images, SVG text is selectable with the mouse just like any other text in the browser. You can copy the text and then paste it.
- **Zoomable**: SVG text can be zoomed without pixelation effects. This has an added benefit for web developers: you don’t need to worry about the display device’s *pixel density*. Rendering is done by the browser, where information about the device’s display is available. In contrast, raster images need to be pre-rendered at different pixel densities for optimal results on retina displays.

- **Accessible**: Screen readers can read the text correctly to visually impaired users, especially if the `<desc>` tag is used.
- **Conversion-safe**: If your site's content is ever converted to plaintext or other formats, the text content can be preserved even if the formatting cannot.
- **SEO friendly**: Search engines can read the text.
- **Can be integrated seamlessly**: SVG text can be easily rendered inline with any normal text in the page.

Common issues hindering usage of SVG on the web

SVG text is awesome. Writing SVG markup directly is not.

- **Coding is hard**: Writing SVG code requires a decent amount of knowledge of the XML-based standard. Not a task to be left to the site's content editor (end user). Even if you're an experienced developer, it is always less convenient and more time-consuming than editing normal text.
- **Graphics are also hard**: The option of producing SVG markup using a vector graphics editor, such as *Adobe Illustrator* does exist. However this does again require specialised knowledge of the editing program used and cannot be done easily by the content editor. While the text *content* is easily editable, if the graphic *effects* ever need to be modified, one must go back to the SVG editor and repeat whatever workflow was used to re-generate the SVG code and to re-insert it into the site's content.
- **Generated code is messy**: When an SVG editor is used, the auto-generated code is almost always cluttered and longer than it needs to be. Irrelevant metadata, useless tags and numbers with ridiculous amounts of precision make the code look silly.
- **Must be inserted into the page**: Inserting SVG code into a webpage so that it works seemlessly in all browsers is a non-trivial task. For older browsers and devices that do not support the SVG standard, an appropriate fallback mechanism must exist so that the user experience can degrade gracefully on old mobile devices and old versions of Internet Explorer.

The SVG Logo and Text Effects plugin for WordPress

Concept

SVG Logo and Text Effects lets you write text that renders as SVG into your *WordPress* site. You set and tweak arguments on a WordPress shortcode to control your text's appearance. The plugin takes care of SVG rendering and provides fallback mechanisms for any browsers that do not yet support SVG.

Think logos, headers, titles, calls to action, etc… Any text that needs to be artistic or eye-catching for any purpose.

With *SVG Logo and Text Effects*, you synthesize text using a number of built-in and pluggable *resources*. These are:

- **fonts** — Any Google Fonts can be used (currently over 700 fonts). In addition, with very little coding you can include any font from file.
- **shape templates** — Control the overall capabilities of your text using the provided *shape templates*. Advanced users can create new shape templates to combine text with custom graphics and other SVG features.
- **fill templates** — Choose and tweak graphical patterns to use for the inside of your text. The letters of your text can be filled with a selection of patterns and gradients.
- **filter templates** — Specify SVG filters to use on your text, such as a *glow* effect or a *shadow* effect. Multiple filters can be combined.
- **presets** — Presets are ready-made examples of SVG text that you can use as a starting point. Choose a preset that looks close to what you are aiming for, then tweak its arguments.

SVG Logo and Text Effects is designed with modularity and extensibility in mind. All resources (templates, fonts, and presets), are provided by *plugins*. The built-in *core plugin* is always loaded and provides the google fonts and a selection of presets to get you started.

You can produce a wide array of text styles by combining the built-in resources.

Features overview

Create SVG text using shortcodes

Use shortcodes to create SVG text in your posts, pages, or in any custom post types where shortcodes are permitted. Tweak text arguments such as font size, stroke color, and stroke width.

Character-level typography

Use HTML tags or the WordPress's built-in *TinyMCE* editor to make parts of your SVG text bold, italic, underline, strikethrough, superscript, subscript, or create links.

Fill patterns

Choose a fill pattern for the inside of the letters.

Visually this looks best with heavy fonts or bold text or text with large sizes.

Google fonts

Use any Google font out-of-the-box with no additional configuration.

Glow and shadow filters

Combine filters to produce glow and shadow effects.

Caching

Utilises WordPress's caching mechanism for a ligtning-fast end-user experience.

Plugins

You can choose to install a number of plugins to extend functionality or available resources. Visit the dashed-slug to see what's available.

Developers can produce plugins that extend *SVG Logo and Text Effects* with additional *resources*.

HTML fallback

Old versions of Internet Explorer, as well as built-in browsers in some older mobile devices, do not have support for SVG.

Render the text as simple HTML for legacy browsers that cannot display SVG. The HTML preserves as many text characteristics as possible, such as size, color, typography, etc.

Installation

Requirements

SVG Logo and Text Effects has been tested from **PHP 5.3** to **PHP 5.6** with no issues.

It is recommended that you have **at least WordPress version 4.1** installed on your site.

You should be able to get away with using the plugin on **WordPress 3.9** with only minor issues.

Install via the WordPress admin interface

This is the easy method.

The installation instructions for this plugin are the same as for any *WordPress* plugin. What follows is step-by-step instructions:

For more information on installing plugins, you can also consult the relevant WordPress documentation.

1. **Make sure that you have at least WordPress version 3.9 installed, and that you are running on at least PHP 5.3.**
2. **Log in to your WordPress site** as Administrator. Your account will need to have the necessary permissions to install plugins.
3. **Navigate to the Plugins page**, found on the menu on the left.
4. **Click on the *Add Plugin* button.** If you cannot find the *Add Plugin* button, you might have a *Multisite* (aka *Network*) WordPress installation. If that is the case, navigate to *My Sites* → *Network Admin* → *Plugins*. The *My Sites* menu is located at the top left of the screen.
5. Once on the *Add Plugins* page, **click on the *Upload Plugin* button**.
6. **Click *Browse.***
7. **Locate the following file** on your computer:

   `SVG_Logo_and_Text_Effects-1.1.0-wordpress-plugin.zip`
8. **Click *Install Now.*** You should get the message *Plugin installed successfully*.
9. **Click on *Activate*** (or *Network Activate* if you have a multisite installation).

Install using wp-cli

This method is more suited to advanced users and system administrators.

If you have SSH access to your host server and have the WordPress CLI installed, you should be able to install the plugin from the command line, if you so prefer. First upload

`SVG_Logo_and_Text_Effects-1.1.0-wordpress-plugin.zip`

to your server with FTP, SFTP, SCP or any method that you use to upload files. Then, install and activate with:

```
wp --path=WORDPRESS_PATH plugin install \
    SVG_Logo_and_Text_Effects-1.1.0-wordpress-plugin.zip
wp --path=WORDPRESS_PATH plugin activate slate
```

Replace `WORDPRESS_PATH` with your actual WordPress path. On linux systems this is probably something like `/var/www/wordpress`. If unsure, consult your system administrator.

If you have a multi-site installation then append the parameter `--network` to both commands:

```
wp --path=WORDPRESS_PATH plugin install --network \
    SVG_Logo_and_Text_Effects-1.1.0-wordpress-plugin.zip
wp --path=WORDPRESS_PATH plugin activate slate
```

You can now verify that the plugin with the `slate` slug is installed and activated:

```
wp --path=WORDPRESS_PATH plugin list
```

The `slate` plugin should be in the list, marked active.

You should now be ready to use *SVG Logo and Text Effects*.

Configure settings (optional)

To visit the plugin's settings page, navigate to *Settings → SVG Logo and Text Effects*.

SVG caching

Enable if you wish SVG and PNG fallback output to be cached by WordPress (recommended). You might wish to disable this while developing new PHP templates.

Other settings

If you choose to install any premium plugins, this page might contain additional settings. Consult the documentation of those plugins for details.

Usage

Writing SVG shortcodes

If you prefer to insert SVG text via a friendly UI, install the Visual Composer integration plugin for SLATE.

Example syntax

To better understand the shortcode syntax, let's look at an example.

```
[slate_plain
    _font_family="Ubuntu"
    _font_size="40pt"
    _stroke="#dd9933"
    _stroke_width="2px"
    fill="gradient_linear"
    fill_gradient_linear_color1="#8224e3"
    fill_gradient_linear_color2="#ff2323"
    filter_glow_radius="5"
    filter_glow_color="#eeee22"
    filter_shadow_radius="10"
    filter_shadow_xoff="10"
 filter_shadow_yoff="20"]Hello <i>SVG</i> World![/slate_plain]
```

Syntax reference

Shape template name

Shortcodes follow the general form

```
[slate_SHAPETEMPLATE ...]your text here[/slate_SHAPETEMPLATE]
```

where `SHAPETEMPLATE` is the name of a shape template. The default shape template provided by SLATE is `plain`. Installing the *More Shapes extension* will give you access to more shapes.

Here we use the `plain` shape template, which lives in `slate/resources/shapes/plain.php`. This is a simple SVG template offered by the *core plugin*. All *resources* in *SVG Logo and Text Effects* are provided by plugins. See the *Plugins* section for more details.

Text content

Enter your text between the square brackets. All of the following is allowed:

- You may use **character-level typography** such bold, italic, underline, strikethrough, superscript and subscript. Normally typography tags such as `<b>`, `<strong>`, etc are

not valid in SVG. However you may still use these tags, and the plugin will transparently transform them to styled `<tspan>` tags, which work in SVG. This means you can also use the corresponding TinyMCE editor buttons without issues.
- You may also insert **links** with the 🔗button.
- You may paste any **Unicode special symbols** into your text (as long as you are using a font that supports them).
- Your text can potentially include **shortcodes from other plugins**.

Font Family

Set the `_font_family` attribute to specify a font family for your text.

- The Google fonts are loaded by the built-in `core` plugin and are always available. Use any Google font family such as `Droid Sans` or `Lobster`.
- If you have installed a plugin that provides additional font families, you can reference those too with this attribute.
- If this attribute is not specified, the SVG text inherits the font family of the surrounding text.

Font Size

Specify a font size with the `_font_size` attribute.

- You may use any valid HTML units and font sizes, such as `12px`, `14pt`, `120%`, etc. Refer to the relevant W3C documentation for details.

  Note: If you use font sizes that are valid in HTML but not SVG, such as `larger`, the raster fallback plugin will not be able to render your text correctly for browsers that do not support SVG. This is a limitation of IMagick, the backend renderer that the raster fallback plugin relies on.

Stroke Color

Specify a stroke color with the `_stroke` attribute.

- This will be the color of the outline around the text's letters.
- You may specify a color in hex notation (e.g. `#FF0000`), or in RGB or RGBA notation (e.g. `rgba(255, 0, 0, 0.5)`).

Stroke Width

Enter a stroke witdth in pixels for the text's outline using the `_stroke_width` attribute.

- This value can be a decimal number, e.g. `0.5` .
- To disable text outline, set this to zero.

X-axis skew angle

Set `_skew_x_deg` to specify an angle in degrees.

- The text will be rotated along the X axis by this angle, effectively skewing the text towards a trapezium-like shape.

  Note: Because this feature relies on CSS 3D transforms, the raster fallback plugin will not be able to render your text correctly for browsers that do not support SVG. This is a limitation of IMagick, the backend renderer that the raster fallback plugin relies on.

Other shape template arguments

Set values for arguments that are specific to the shape template you are using.

- The `plain` shape template offers one argument, `bleed_percent`. Use this to define a "margin" around your text where filters will apply. This lets shadows, glows and other effects extend out of the bounding box of the text.
- Other templates can define their own custom arguments to control various aspects of text presentation. You can specify any custom arguments defined by other shape templates. Refer to the documentation of the plugin that provides your shape template for details. If documentation is not available from the shape vendor, you can check the PHPdoc headers of the respective PHP templates.

Text fill

Optionally set a *fill pattern* using the `fill` attribute. This will be used to fill the inside of the text's letters.

- In the example, we are using the fill `gradient_linear`, which lives in `resources/fill/gradient_linear`. The PHPdoc header of this file defines the arguments that we can specify:

```
/**
 * Apply a linear gradient fill into your text.
 *
 * @var color $color1 (default:#000000) The first color
* @var string $x1 (default:50%) The horizontal position of the first color
* @var string $y1 (default:0%) The vertical position of the first color
 * @var color $color2 (default:#ffffff) The second color
* @var string $x2 (default:50%) The horizontal position of the second color
* @var string $y2 (default:100%) The vertical position of the second color
 */
```

- In the example above, we specify the arguments `color1` and `color2`.
- The attribute name follows the pattern `fill_FILL_PARAM`, where `FILL` is the fill template name and `PARAM` is the name of the argument.
- Any arguments we ommit in the shortcode, such as `x1` and `y1`, will take their default values.

- To see the built-in fill patterns and their arguments, explore the files under `resources/fill`.

Text filters

Two filters are currently provided by the built-in `core` plugin: *glow* and *shadow*. Built-in filters live in `resources/filter`. Here's their PHPdoc:

```
/**
 * Apply a glow filter around your text
 *
 * @var integer $radius (default:0) The glow's radius in pixels
 * @var integer $radiusmax (default:0) The glow's maximum radius in pixels (for animation)
 * @var float $duration (default:0) Duration in seconds (for animation)
 * @var color $color (default:#ff0000) The glow's color in hex rgb
 */

/**
 * Apply a shadow filter on your text.
 *
 * @var string $radius (default:0) The glow's radius in pixels
 * @var string $xoff (default:0) The glow's offset on the x axis
 * @var string $yoff (default:0) The glow's offset on the y axis
 */
```

To pass an attribute to one of the filters, set an attribute of the form `filter_FILTER_ATT="VALUE"`, where `FILTER` is the filter's name, `ATT` is the template attribute you want to set, and `VALUE` is the value to set.

The following example uses the `shadow` filter to create a drop shadow effect of 5 pixels radius, and positioned 8 pixels to the right and 8 pixels down of the text:

```
filter_shadow_radius="5"
filter_shadow_xoff="8"
filter_shadow_yoff="8"
```

Here is an example of a yellow glow effect with a radius of 5 pixels:

```
filter_glow_radius="5"
filter_glow_color="#FFFF00"
```

And here is the same glow effect, with the radius animated between 5 and 20 pixels, where the animation duration is 2 seconds:

```
filter_glow_radius="5"
filter_glow_radiusmax="20"
filter_glow_duration="2"
filter_glow_color="#FFFF00"
```

Extending *SVG Logo and Text Effects* via plugins

Using SLATE extension plugins

Additional **resources** in *SVG Logo and Text Effects* are provided by **SLATE extension plugins**.

This means that SLATE plugins can add to the pool of available **fonts**, **shape templates**, **filter templates**, **fill templates**, and **presets** that you can use.

SVG Logo and Text Effects extension plugins are installed into WordPress just like any other plugin, from the *Plugins* page, found on the menu on the left part of your admin screens.

If you have *Visual Composer* and the *SLATE Visual Composer Integration extension* installed, the new resources should be available in the Visual Composer UI as soon as the plugin is activated.

Even if you do not have *Visual Composer* and the *SLATE Visual Composer Integration extension* installed, you can still look at an overview of which resources are loaded. Navigate to WordPress's *Control Panel* (first option at the left hand side admin menu), and refer to the *SVG Logo and Text Effects* widget. Click on the *Resources* tab. Here you see counts of each type of resource loaded. Hover your mouse over the numbers to see the names of the loaded resources.

Developing SLATE extension plugins

Developers can create SLATE extension plugins, to provide additional **resources**.

- Creating plugins that only introduce **fonts** or **presets** does not require any special knowledge besides PHP.
- Creating plugins that introduce new **shape**, **fill** or **filter** templates requires knowledge of the SVG standard.

Hooking into SVG Logo and Text Effects

SVG Logo and Text Effects discovers resources by querying WordPress filters. You may also want to refer to the Plugin API documentation for background reading on WordPress filters. The available filter hooks are:

`slate_shape_files`

An array of files or directories containing SVG *shape templates* (PHP files).

`slate_font_files_or_urls`

An associative array where keys are **font family** names and values are arrays that define external fonts. The arrays can take the following keys:

- `url` — A URL pointing to a font file.
- `source` — Currently can take the values `google` and `custom`. May be extended in future versions.

`slate_fill_files`

An array of files or directories containing SVG **fill templates** (PHP files).

`slate_filter_files`

An array of files or directories containing SVG **filter templates** (PHP files).

`slate_presets`

An associative array of SVG **text presets** (i.e. sample texts), where keys are template names and values are texts in shortcode form.

Follow the examples

Development of SVG extension plugins is best illustrated with examples:

- *SVG Logo and Text Effects* always loads the built-in **core** extension plugin (`core.php`).
- If you downloaded the bundle `.zip` file, you will have received an example of an external extension plugin, `slate-plugin-example.zip`.

When creating your own extension plugins, consult this code first.

Templates and variable naming

When creating new templates, make sure to only use allowed characters to name your PHP template files, and their variables. Allowed characters are letters, numeric digits and underscores. Always start your names with a letter.

First create a PHP file for your plugin. This needs to be a valid relevant WordPress documentation.

Start with a specially formatted header. The header from `slate-plugin-example` can be used as a starting point:

```
<?php
/*
 * Plugin Name: Example plugin for SVG Logo and Title Effects for WordPress
 * Description: Example code showing how to create a plugin for
 * SVG Logo and Title Effects. Consult the documentation for more.
 * Version: 1.0.0
 * Author: Dashed Slug <info@dashed-slug.net>
```

```
 * Author URI: http://dashed-slug.net
 * License: GPLv2 or later
 */
```

After this header, add your code to bind to any of the five WordPress filters, as described below.

PHPdoc

Shape, **filter**, and **fill** templates are PHP files with some special syntax:

When writing PHP templates, use PHPdoc comments to give a short textual description of what the template does. Use the `@var` tag to specify input variables.

The following **fill template** is from the built-in core plugin:

```
<?php

/**
 * Apply a simple color fill into your text.
 *
 * @var color $color (default:#ffffff) The color
 */
?>

<pattern
    id="<?php echo $_id; ?>"
    x="0" y="0"
    width="1" height="1"
    patternUnits="userSpaceOnUse">

    <rect
        x="0" y="0"
        width="1" height="1"
        fill="<?php echo $color; ?>"
        stroke-width="0">
    </rect>

</pattern>
```

Note that the `@var` tag is slightly different from the PHPdoc specification:

- Valid argument types are: `color`, `string`, `html`, `integer`, `float`.
- Default values are specified after the variable name as shown. In the example given here, the default color is `#ffffff` (white).

Some notes on how variables are handled:

- *Visual Composer* shows a colorpicker for variables of type `color` and a plain text field for all other types.

- **Integers** and **floating point** values are passed through intval() and floatval() respectively.
- **HTML** is parsed so that basic typography is converted to `<tspan>` tags which are compatible with SVG. Use it if you want your variable to displayed as text with bold, italic, underline, etc.

Adding SVG *shape templates*

Shape templates control the flow of code within the `<svg>` tag. They combine your text, with filters and your chosen fill, if any. Note that the `svg` tag itself is rendered by the plugin, not the shape template.

Declaring in plugin code

The `shape` directory is appended to the array. Its PHP files will be loaded as SVG *shape templates*.

```
function slate_core_shapes( $file_names ) {
    $file_names[] = realpath( __DIR__ . '/resources/shape/' );
    return $file_names;
}

add_filter( 'slate_shape_files', 'slate_core_shapes' );
```

PHP template example

The file `slate/resources/shape/plain.php` will be found and added. Its contents control the markup within the `<svg>` tag:

```
<?php
/**
 * Plain old vector text.
 */
?>
<defs>
    <filter id="<?php echo $_filters_id; ?>" x="-100%" y="-100%"
    width="300%" height="300%" filterUnits="objectBoundingBox">
        <?php echo $_filters; ?>
    </filter>
    <?php echo $_fill; ?>
</defs>

<text y="100%"><?php echo $_text_tspan; ?></text>
```

Special shape template variables

Some special variables are always available in *shape templates*:

- `$_id` — A unique ID for this SVG text. Here we append the `-filter` suffix to obtain a unique filter ID.
- `$_filters` — The markup for all user-selected filters are rendered into this variable. To be used within the SVG `<filter>` tag.
- `$_fill` — The markup for the user-selected fill. To be used within the SVG `<defs>` tag.
- `$_filters_id` — An ID that is unique to the combined filters of this SVG text.
- `$_fill_id` — An ID that is unique to the fill pattern of this SVG text.
- `$_text` — The text content of the SVG shortcode, with any nested shortcodes already computed.
- `$_text_tspan` — Same as `$_text`, but any HTML tags that control character-level typography are replaced by `<tspan>` tags: So `<b>` or `<strong>` becomes `<tspan style="font-weight: bold">`, `<i>` or `<emph>` becomes `<tspan style="font-style: italic">`, etc. This is necessary when printing text within SVG `<text>` tags, where normal HTML tags are not valid.

Adding SVG *fill templates*

Declaring in plugin code

The `fill` directory is appended to the array. Its PHP files will be loaded as SVG *fill templates*.

```
function slate_core_fills( $file_names ) {
    $file_names[] = realpath( __DIR__ . '/resources/fill/' );
    return $file_names;
}

add_filter( 'slate_fill_files', 'slate_core_fills' );
```

PHP template example

The file `slate/resources/fill/color.php` will be found and added. Its contents will be rendered within the SVG `<defs>` tag:

```
<?php

/**
 * Apply a simple color fill into your text.
 *
 * @var color $color (default:#fffff) The color
 */
?>
```

```
<pattern
    id="<?php echo $_id; ?>"
    x="0" y="0"
    width="1" height="1"
    patternUnits="userSpaceOnUse">

    <rect
        x="0" y="0"
        width="1" height="1"
        fill="<?php echo $color; ?>"
        stroke-width="0"></rect>

</pattern>
```

Special fill template variables

The `$_id` variable is always available in *fill templates*:

- `$_id` — A unique ID for this SVG text. Here we use it to obtain a unique fill ID.

Adding SVG *filter templates*

Declaring in plugin code

The `filter` directory is appended to the array. Its PHP files will be loaded as SVG *filter templates*.

```
function slate_core_filters( $file_names ) {
    $file_names[] = realpath( __DIR__ . '/resources/filter/' );
    return $file_names;
}

add_filter( 'slate_filter_files', 'slate_core_filters' );
```

PHP template example

The file `slate/resources/filter/shadow.php` will be found and added. Its contents will be rendered under the SVG `<defs>` tag:

```
<?php
/**
 * Apply a nice shadow filter on your text.
 *
 * @var string $radius (default:0) The glow's radius in pixels
 * @var string $xoff (default:0) The glow's offset on the x axis
 * @var string $yoff (default:0) The glow's offset on the y axis
 */
```

```
$radius = isset( $radius ) ? intval( $radius ) : 0;
$xoff = isset( $xoff ) ? intval( $xoff ) : 0;
$yoff = isset( $yoff ) ? intval( $yoff ) : 0;

?>
<feGaussianBlur
    in="SourceAlpha"
    stdDeviation="<?php echo "$radius $radius"; ?>" />

<feOffset
    dx="<?php echo $xoff; ?>"
    dy="<?php echo $yoff; ?>"
    result="<?php echo $_result; ?>" />
```

Special filter template variables

The `$_id` and `$_result` variables are always available in *filter templates*:

- `$_id` — A unique ID for this SVG text.
- `$_result` — A unique result name to specify as the SVG result of the filter.

Adding fonts

Google fonts

The core plugin uses an external list of Google font family names. In the `$fonts` array, keys are created for each font family name. The corresponding values are *font-specifying arrays*. The `source` is set to be `google` for Google fonts. This is currently the only valid font source, besides `custom` (default) which can be omitted when specifying fonts via `url`.

```
function slate_core_fonts( $fonts ) {
    foreach ( file(
        __DIR__ . '/resources/font/google-font-families.txt',
     FILE_IGNORE_NEW_LINES | FILE_SKIP_EMPTY_LINES ) as $font_family ) {
        $fonts[$font_family] = array( 'source' => 'google' );
    }
    return $fonts;
}

add_filter( 'slate_font_files_or_urls', 'slate_core_fonts' );
```

Fonts from file

The example plugin (`slate-plugin-example.php`) loads an external TTF file. The file is packaged with the plugin. Make sure to check distribution limitations placed on a font by its author before redistributing in a plugin.

Do not set the `source` parameter. Instead set the `url` parameter to be a full URL to the font file. Since the font is packaged with the `slate-plugin-example` plugin under the `fonts` directory, the plugin_dir_url() WordPress function can be used to get a URL to the file:

```
function slate_example_fonts( $fonts ) {
    $fonts['Final Frontier Old Style'] = array(
     'url' => plugin_dir_url( 'slate-plugin-example/fonts/allen-r-walden_final-frontier-old-
              . 'allen-r-walden_final-frontier-old-style.ttf' );
    return $fonts;
}

add_filter( 'slate_font_files_or_urls', 'slate_example_fonts' );
```

Adding presets

Presets are key-value pairs where keys are names and values are shortcodes.

Here is an example taken from the `slate-plugin-example` plugin:

```
function slate_example_presets( $presets ) {
    $presets['Space Path'] = '[slate_plain ' .
        '_font_family="Final Frontier Old Style" ' .
      '_font_size="36pt" _stroke="#000000" _stroke_width="0px" ' .
       'fill="color" fill_color_color="#226fc1"]' .
       '<i>SPACE PATH</i>[/slate_plain]';

    return $presets;
}

add_filter( 'slate_presets', 'slate_example_presets' );
```

When creating new plugins, try to create good examples that showcase your plugin's features.

Disable caching when developing extension plugins

SVG Logo and Text Effects utilizes caching when rendering SVG text. This means that subsequent renderings of the same shortcode are pulled from cache, for performance reasons. When developing **fills**, **filters**, or **shapes**, this can interfere with your development cycle. You can disable caching from *Admin → Settings → SVG Logo and Text Effects settings → SVG caching*.

Packaging your plugin

When you are developing a new PHP template for *SVG Logo and Text Effects*, you can simply place it in the appropriate directory within the *core plugin* and it will be loaded:

- Place **shape templates** in `wordpress/wp-content/plugins/slate/resources/shape`.
- Place **filter templates** in `wordpress/wp-content/plugins/slate/resources/filter`.
- Place **fill templates** in `wordpress/wp-content/plugins/slate/resources/fill`.

However *SVG Logo and Text Effects* extension plugins need to be properly packaged before distribution:

Create a zip file containing the plugin's PHP code and its resources. Use `slate-plugin-example.zip` as a starting point.

There's more

You can extend SLATE with various extension plugins.

slate-shapes: More Shapes extension

Extends the WordPress SLATE plugin to add more SVG shapes. Shapes are essentially SVG templates that allow more complex text effects.

slate-vc: Visual Composer integration extension

Integrates the SVG Logo and Text Effects plugin with WPBakery's Visual Composer.

slate-raster: Raster Fallback extension

Extends the WordPress SLATE plugin to add raster fallback for browsers that cannot render SVG.

slate-titles: Title Replacement extension

Extends the WordPress SLATE plugin to allow replacing the site title and page / post titles with SVG text.

Support

To submit any issue, bug, question, feature suggestion, etc please use the issue tracker.