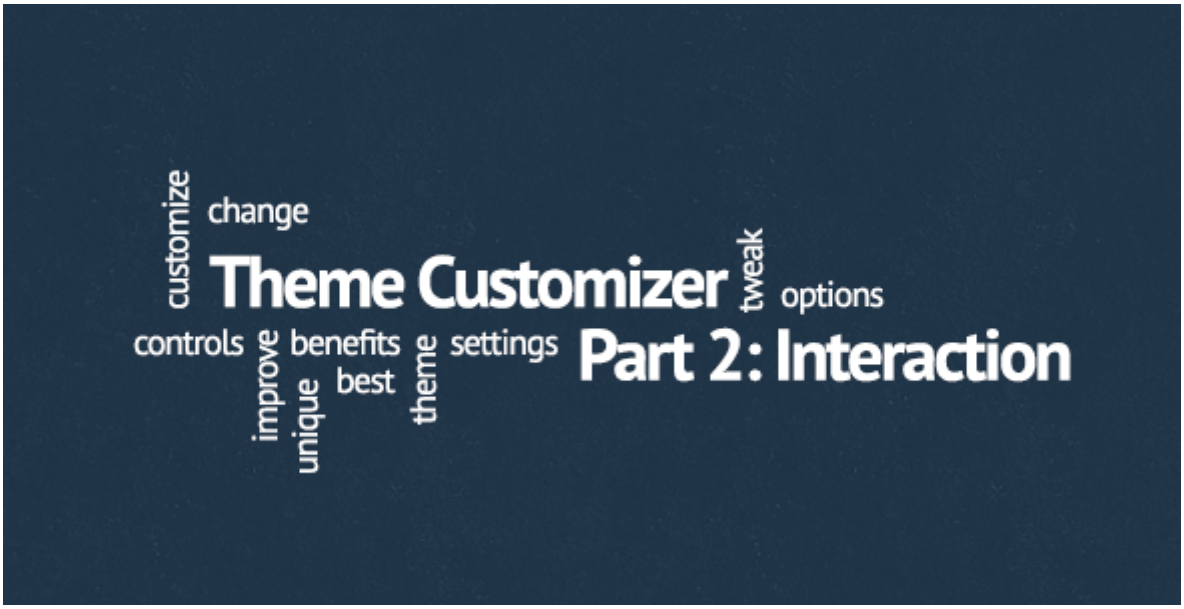
TUTORIALS

Interacting With WordPress Theme Customizer

LAST UPDATED ON: DECEMBER 27, 2017



1. Introduction To The WordPress Theme Customizer
2. **Currently Reading:** ~~*Interacting With WordPress Theme Customizer*~~
3. WordPress Theme Customizer Boilerplate
4. Extending The WordPress Theme Customizer Boilerplate
5. Theme Customizer Boilerplate – Conditional Options, Child Themes and Plugins

---

In part 1 of WordPress Theme Customizer series I mentioned that in order to interact with Theme Customizer you need to load **$wp_customize** object, which is an instance of **WP_Customize_Manager** class. To do that, you must use **customize_register** action hook:

```
1 add_action( 'customize_register', 'my_theme_customize_register' );
2 function my_theme_customize_register( $wp_customize ) {
3
```

```
    // Interacting with $wp_customize object

}
```

You can place this code in your theme's functions.php or a file that's included from it.

Adding or removing Theme Customizer elements (sections, settings and controls)

Once you have loaded **$wp_customize** object you can use any of its methods to add, get or remove settings, controls and sections in it (add_setting, get_setting, remove_setting, add_control… you get the point).

So, if you want to **get** or **remove** a section, control or setting, all you need is its ID. This line will remove Colors section (place it inside my_theme_customize_register function from first code snippet):

```
$wp_customize->remove_section( 'colors' );
```

Adding a section, control or setting is slightly different because it requires some more parameters. I will not go through all of them here for two reasons:

1. That's not really what purpose of this series is, we'll be creating a Theme Customizer boilerplate that you can just drop into your theme instead
2. Alex Mansfield already covered it in his 6000 word monster of a Theme Customizer tutorial that every WordPress theme developer must read and then tweet about (seriously, if you haven't already, go read it now).

But still, let's take a look at how you can add your own setting with a control in a new Theme Customizer section, as well as some of the arguments. Since it's much easier to work with real examples, here's what we're going after:

- A new section, labeled "Layout"
- A new setting that stores your theme's layout
- A new radio control with two options – sidebar on the left and sidebar on the right

First thing to add to Theme Customizer is “Layout” section:

```
$wp_customize->add_section(
    // ID
    'layout_section',
    // Arguments array
    array(
        'title' => __( 'Layout', 'my_theme' ),
        'capability' => 'edit_theme_options',
        'description' => __( 'Allows you to edit your theme\'s layout.',
    )
);
```

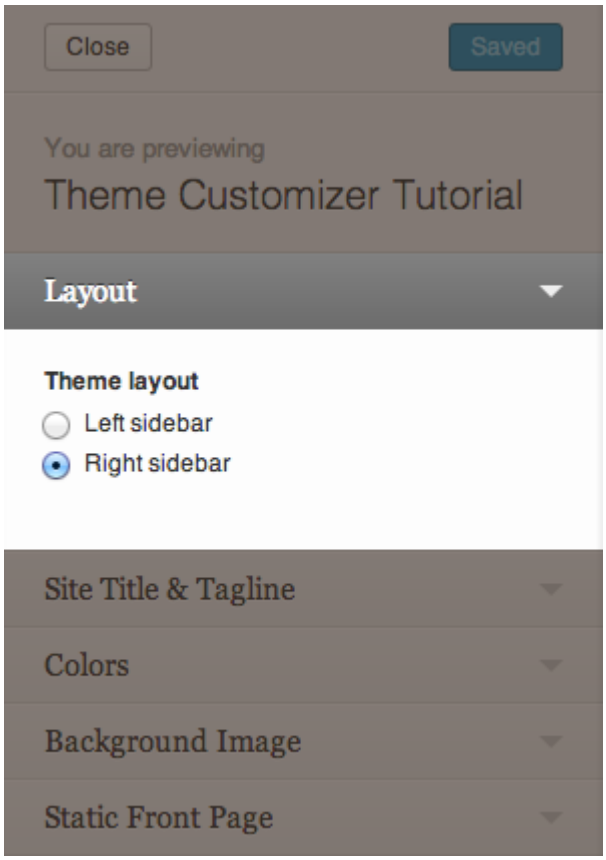
Don’t try to see it in customizer yet, a section will not be shown unless it has a setting and a control added to it. So let’s add both:

```
$wp_customize->add_setting(
    // ID
    'my_theme_settings[layout_setting]',
    // Arguments array
    array(
        'default' => 'right-sidebar',
        'type' => 'option'
    )
);
$wp_customize->add_control(
    // ID
    'layout_control',
    // Arguments array
    array(
        'type' => 'radio',
        'label' => __( 'Theme layout', 'my_theme' ),
        'section' => 'layout_section',
        'choices' => array(
            'left-sidebar' => __( 'Left sidebar', 'my_theme' ),
            'right-sidebar' => __( 'Right sidebar', 'my_theme' )
        ),
        // This last one must match setting ID from above
        'settings' => 'my_theme_settings[layout_setting]'
    )
);
```

Presuming you read Alex’s tutorial and/or Codex pages there’s only one parameter in add_setting arguments array — ‘type’ — that I’d like to focus on. You have two possibilities here, ‘option’ and ‘theme_mod’ and you can retrieve

them by using get_option and get_theme_mod, respectively. I always use 'option' simply because it allows you to serialize your theme settings values by giving them IDs like **my_theme_settings[setting_1]**, **my_theme_settings[setting_2]** etc. That way all values will be stored as one database entry in your wp_options table.

And finally, after you added those two code snippets to function you hooked into **customize_register** action hook (first code snippet in this post), Theme Customizer has been customized:



New section added to Theme Customizer

Using Theme Customizer settings values in your theme

After you have given your users ability to store this setting, you can grab its value, hook into **body_class** filter hook and add it to array of existing body classes:

```
1  add_filter( 'body_class', 'my_theme_body_classes' );
2  function my_theme_body_classes( $classes ) {
3
4      /*
```

```
     * Since we used 'option' in add_setting arguments array
     * we retrieve the value by using get_option function
     */
    $my_theme_settings = get_option( 'my_theme_settings' );

    $classes[] = $my_theme_settings['layout_setting'];

    return $classes;

}
```

This will add either .left-sidebar or .right-sidebar to array of body classes in your theme. By using these two classes in your theme's style.css file you'll be able to create two different layouts. For example:

```
/* Sidebar on the right is default layout */
#content {
    float: left;
    width: 60%;
}
#sidebar {
    float: right;
    width: 30%;
}

/* Using .left-sidebar class to override default layout */
.left-sidebar #content {
    float: right;
}
.left-sidebar #sidebar {
    float: left;
}
```

Best of all, thanks to WordPress Theme Customizer, users can preview both layouts before saving anything. Take that, theme settings pages!

Summary and Further Reading

TL;DR version of this post would go something like this: **You can get $wp_customize object and then either add something (section, setting or control) to or remove from it.** Everything else boils down to settings parameters.

Part Three is where this series gets interesting as we'll start automating the entire process and working on Theme Customizer Boilerplate that you can drop into your theme and start using right away. Stay tuned!

Article by Slobodan Manic
WPExplorer.com author