Digging into the WordPress Customizer

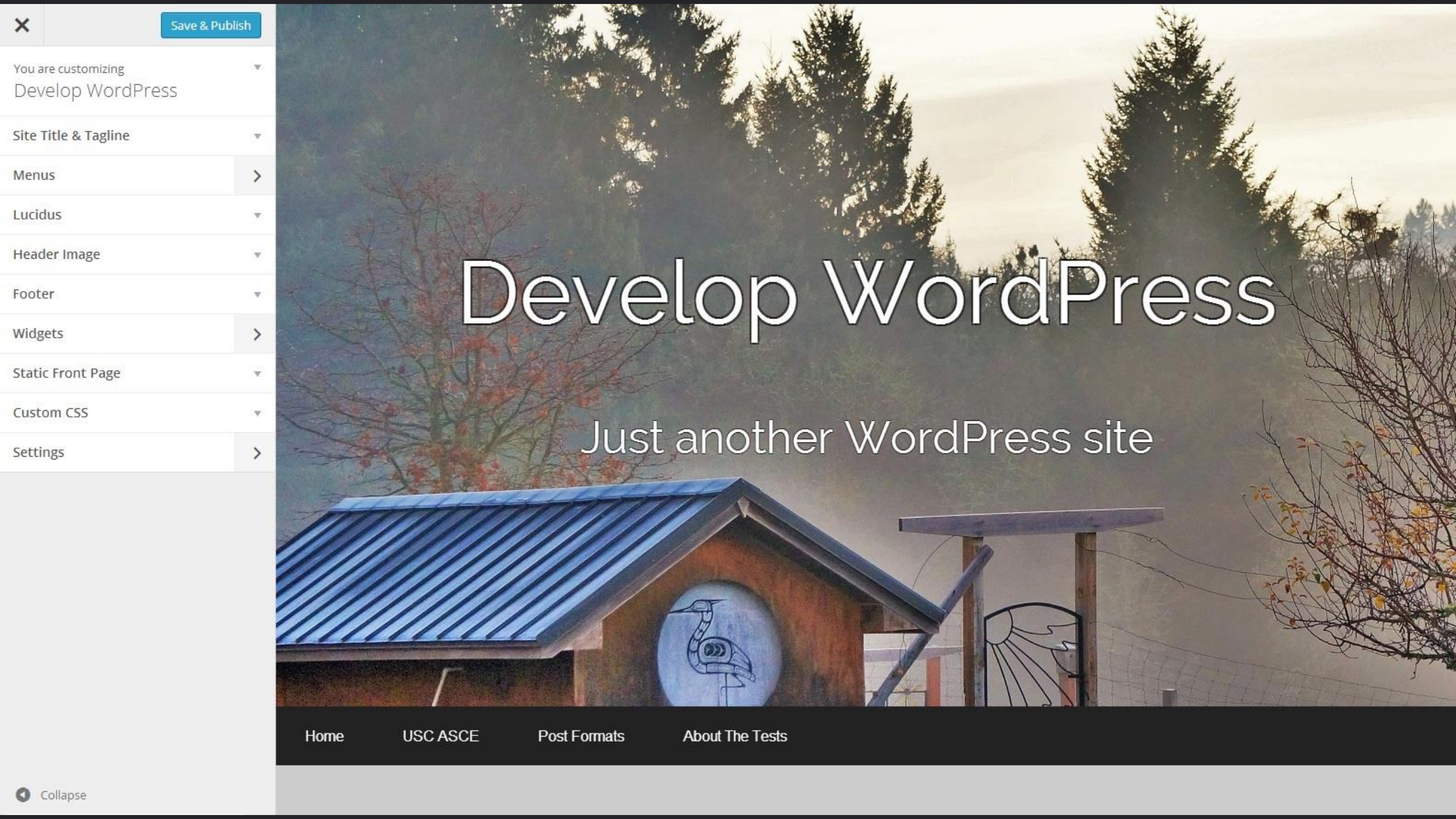
An advanced look at the Customizer API,

improvements in WordPress 4.0, and the future

“A framework for live-previewing any change to WordPress”

- Weston Ruter

Customizer


Save & Publish
You are customizing
Develop WordPress
Site Title & Tagline
Menus
Lucidus
Header Image
Footer
Widgets
Static Front Page
Custom CSS
Settings
Collapse
Develop WordPress
Just another WordPress site
Home
USC ASCE
Post Formats
About The Tests

Agenda

- Basic Usage: Sections, Settings, & Controls
- When should you use the Customizer?
- Intermediate Usage
  - Contextual Controls
  - postMessage Setting Transport
  - Custom Controls, in PHP and JS
  - Panels – Customizer Contexts
- Advanced Usage: Custom Settings
- The Future

```
add_action('customize_register','my_customize_register');
function my_customize_register( $wp_customize ) {
	$wp_customize->add_section();
	$wp_customize->get_section();

	$wp_customize->add_setting();
	$wp_customize->get_setting();

	$wp_customize->add_control();
	$wp_customize->get_control();
}
```

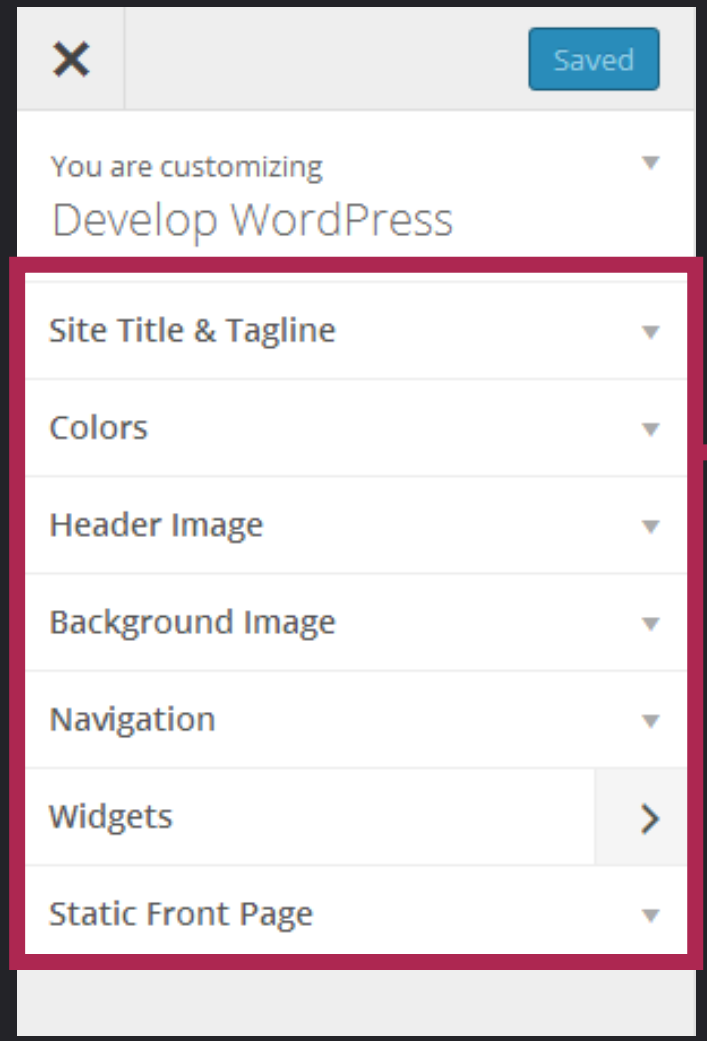
Customizer Sections

```
$wp_customize->add_section( 'custom_css', array(
	'title'           => __( 'Custom CSS' ),
	'description'     => __( 'Add custom CSS here' ),
	'panel'           => '', // Not typically needed.
	'priority'        => 160,
	'capability'      => 'edit_theme_options',
	'theme_supports' => '', // Rarely needed.
) );
```

Customize Meta Capability

```
function allow_users_who_can_edit_posts_to_customize( $caps, $cap, $user_id ) {
	$required_cap = 'edit_posts';
	if ( 'customize' === $cap && user_can( $user_id, $required_cap ) ) {
		$caps = array( $required_cap );
	}
	return $caps;
}
add_filter( 'map_meta_cap', 'allow_users_who_can_edit_posts_to_customize', 10, 3 );
```

Core Section Priorities



Section	id	Priority
Site Title & Tagline	`title_tagline`	**20**
Colors	`colors`	**40**
Header Image	`header_image`	**60**
Background Image	`background_image`	**80**
Navigation*	`nav`	**100**
Widgets (Panel)	`widgets`	**110**
Static Front Page	`static_front_page`	**120**

* Navigation section will move to Menus panel in the Menu Customizer project, priority 30.

Example: Footer Section for a Theme's Options

```
// Add a footer/copyright information section.
$wp_customize->add_section( 'footer' , array(
    'title'     => __( 'Footer', 'themename' ),
    'priority' => 90, // Before Navigation.
) );
```

No need for a description, so don't add one!

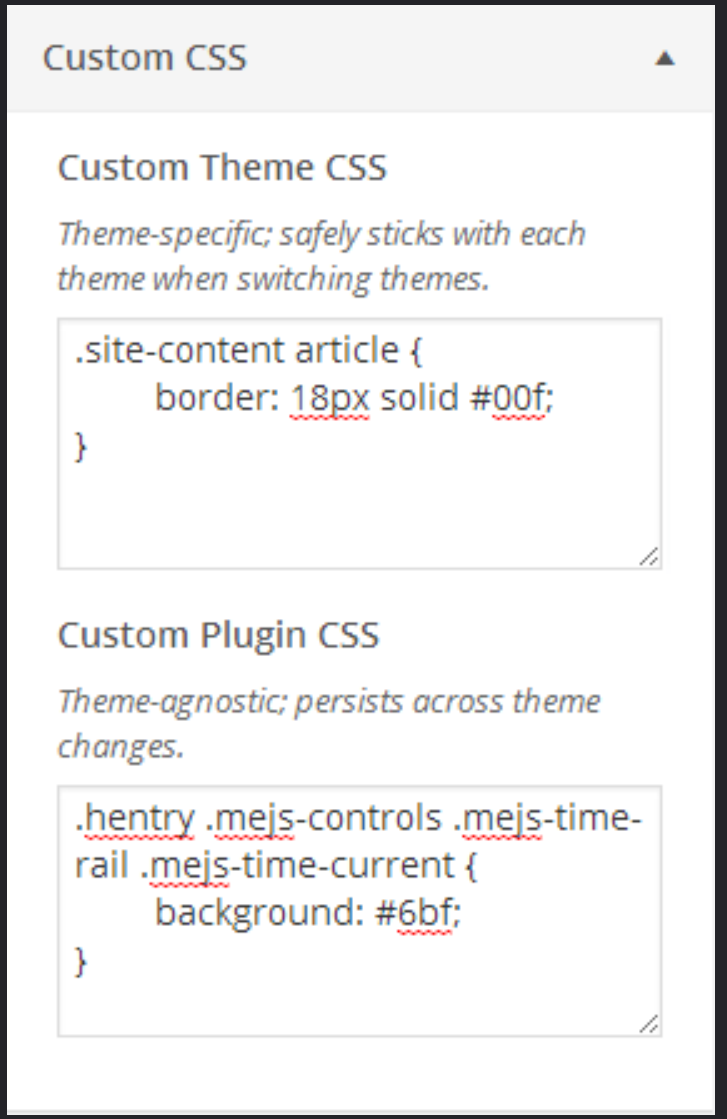
Customizer Settings

```
$wp_customize->add_setting( 'setting_id', array(
      'type'                 => 'theme_mod', // or 'option'
      'capability'           => 'edit_theme_options',
      'theme_supports'       => '', // Rarely needed.
      'default'              => '',
      'transport'            => 'refresh', // or postMessage
      'sanitize_callback'    => '',
      'sanitize_js_callback' => '', // Basically to json.
) );
```

Theme_mods vs. Options

```
$wp_customize->add_setting(
      'custom_theme_css' , array(
            'type'       => 'theme_mod',
) );

$wp_customize->add_setting(
      'custom_plugin_css' , array(
            'type'       => 'option',
) );
```



* This is not pseudo-code, this is it. Copied and pasted from the Modular Custom CSS Plugin.

Typical Theme Usage

```
$wp_customize->add_setting( 'accent_color', array(
    'default'           => '#f72525',
    'sanitize_callback' => 'sanitize_hex_color',
) );
```

Typical Plugin Usage

```
$wp_customize->add_setting( 'quickshare_options[color]', array(
      'type'                 => 'option',
      'capability'           => 'manage_options',
      'default'              => '#ff2525',
      'sanitize_callback' => 'sanitize_hex_color',
) );
```

Output

```
function cxnh_custom_css_output() {
	echo '<style type="text/css" id="custom-theme-css">' .
		get_theme_mod( 'custom_theme_css', '' ) . '</style>';
	echo '<style type="text/css" id="custom-plugin-css">' .
		get_option( 'custom_plugin_css', '' ) . '</style>';
}
add_action( 'wp_head', 'cxnh_custom_css_output');
```

Customizer Controls

```
$wp_customize->add_control( 'setting_id', array(
	'type'        => 'range', *
	'priority'    => 10, // Within the section.
	'section'     => 'title_tagline', // Required, core or custom.
	'label'       => __( 'Range' ),
	'description' => __( 'This is the range control description.' ), *
	'input_attrs' => array( *
		'min'   => 0,
		'max'   => 10,
		'step'  => 2,
		'class' => 'my-custom-class',
		'style' => 'color: #0a0',
	),
	'active_callback' => 'is_front_page', *
) );
```

* New & Improved in WordPress 4.0.

Core Control Types

- checkbox
- textarea
- radio *
- select *
- dropdown-pages

* Pass an array of options to add_control(), $args['choices']

Additional Types

Think:

```
<input type="$type">
```

- text
- hidden
- number
- range
- url
- tel
- email
- search
- time
- date
- datetime
- week

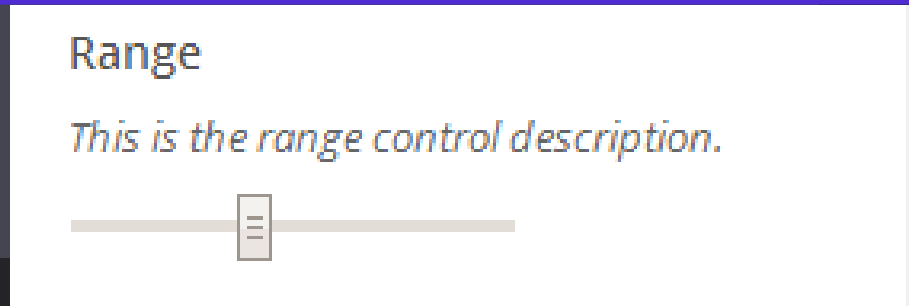
Example: Custom CSS textarea

```
$wp_customize->add_control( 'custom_theme_css', array(
      'label'   => __( 'Custom Theme CSS' ),
      'type'    => 'textarea',
      'section' => 'custom_css',
) );
```

Example: Range (Slider) Control

```
$wp_customize->add_control( 'setting_id', array(
      'type'        => 'range',
      'section'     => 'title_tagline',
      'label'       => __( 'Range' ),
      'description' => __( 'This is the range control description.' ),
      'input_attrs' => array(
            'min'   => 0,
            'max'   => 10,
            'step'  => 2,
      ),
) );
```

```
<input type="range" min="0" max="10" step="2" >
```



When should you use the Customizer?

Developing Theme Options

A. Create a theme options page.
B. Put all of the options in both their own screen and the Customizer.
C. Add the basic settings to the Customizer, but put the complicated stuff in a theme options page.
D. Add all of your options to the Customizer, and only the Customizer.

The Customizer API is easier to use, and ensures that your options blend seamlessly into WordPress core UI. Users can live-preview all of their changes, and have a better understanding of settings' context.

Developing Plugin Options

A. Create a plugin options page under the “Settings” menu.
B. Add the basic & visual settings to the Customizer, but put the complicated / under-the-hood stuff in a custom page (or into several pages).
C. Add all of your options to the Customizer, and only the customizer, creating panel if needed (but then you probably have too many options).

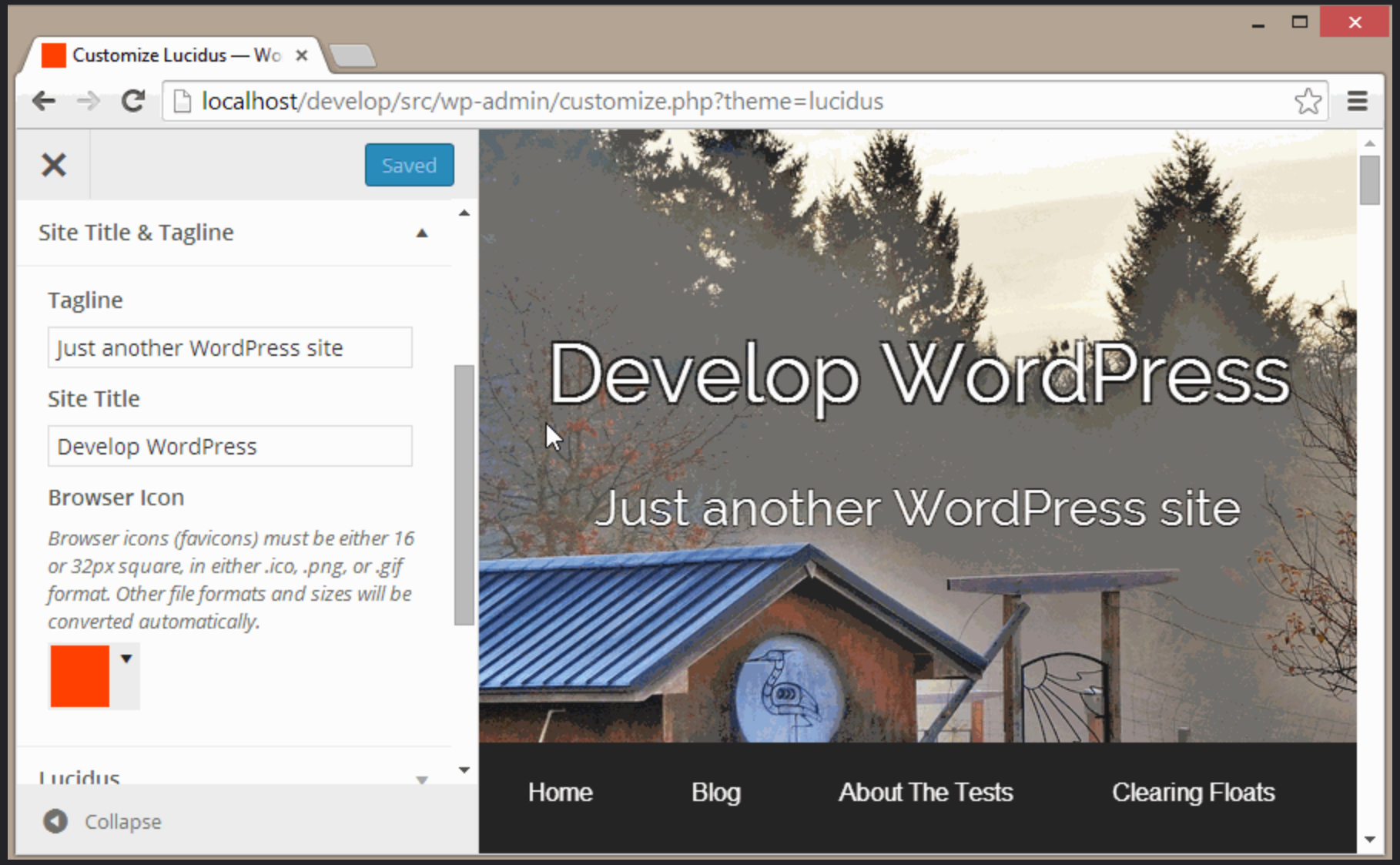
For new plugins, strongly consider using only the Customizer. If your plugin really needs to have options (most don’t, really), and the options affect the front-end in any way, use the Customizer.

Contextual Controls

Only show controls that are relevant to the current preview context.

Let’s say your theme only displays the header image and the site tagline on the front page.

```
// Hide core controls when they aren't used on the current page.
$wp_customize->get_control( 'header_image' )->active_callback =
                                                    'is_front_page';
$wp_customize->get_control( 'blogdescription' )->active_callback =
                                                    'is_front_page';
```


Customize Lucidus — Wo
localhost/develop/src/wp-admin/customize.php?theme=lucidus
Saved
Site Title & Tagline
Tagline
Just another WordPress site
Site Title
Develop WordPress
Browser Icon
Browser icons (favicons) must be either 16 or 32px square, in either .ico, .png, or .gif format. Other file formats and sizes will be converted automatically.
Lucidus
Collapse
Develop WordPress
Just another WordPress site
Home
Blog
About The Tests
Clearing Floats

Adding a Contextual Core Control

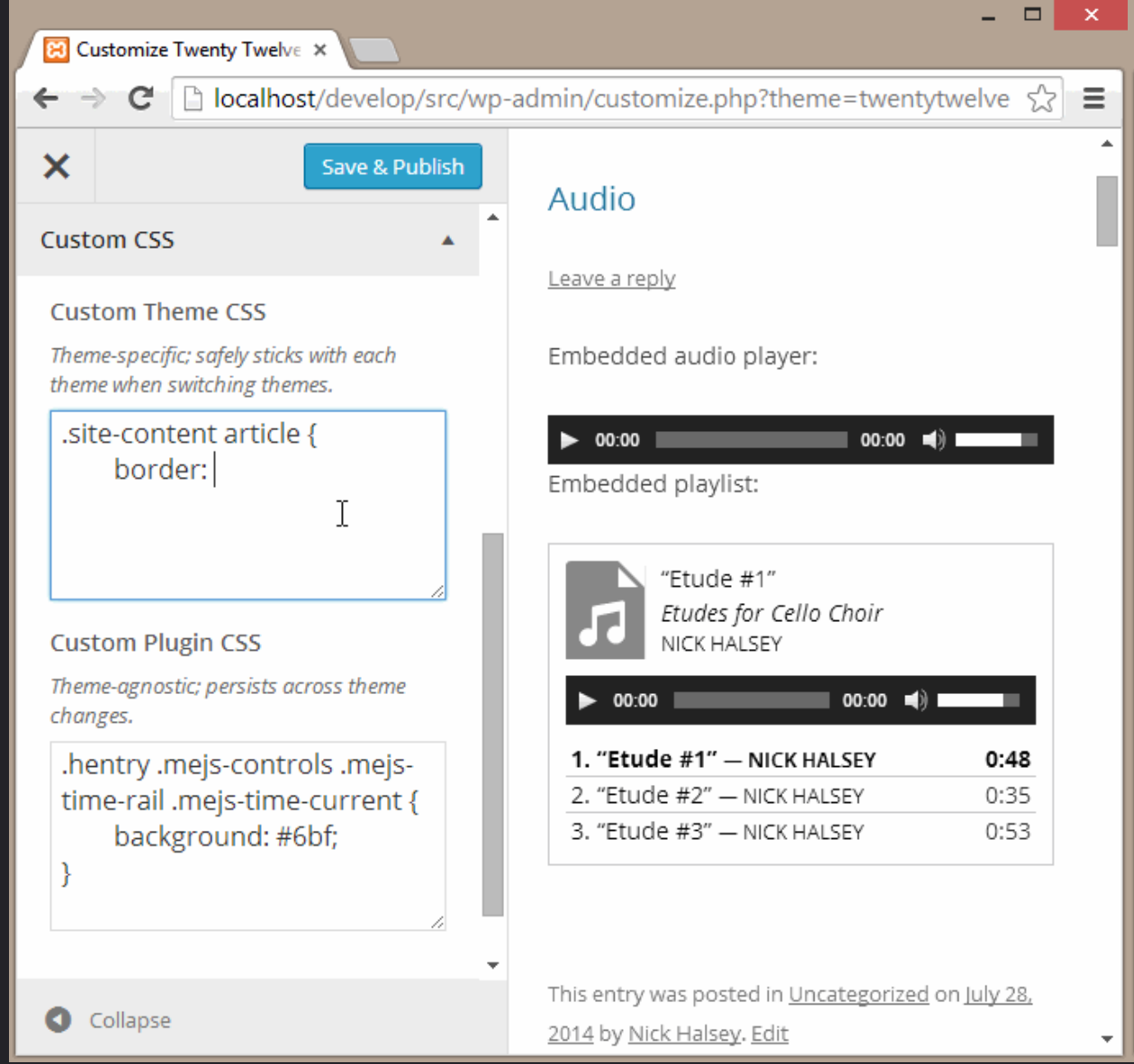
```
$wp_customize->add_control( 'front_page_greeting', array(
      'label'           => __( 'Home Page Greeting' ),
      'type'            => 'textarea',
      'section'         => 'title_tagline',
      'active_callback' => 'is_front_page',
) );
```

Adding a Contextual Custom Control

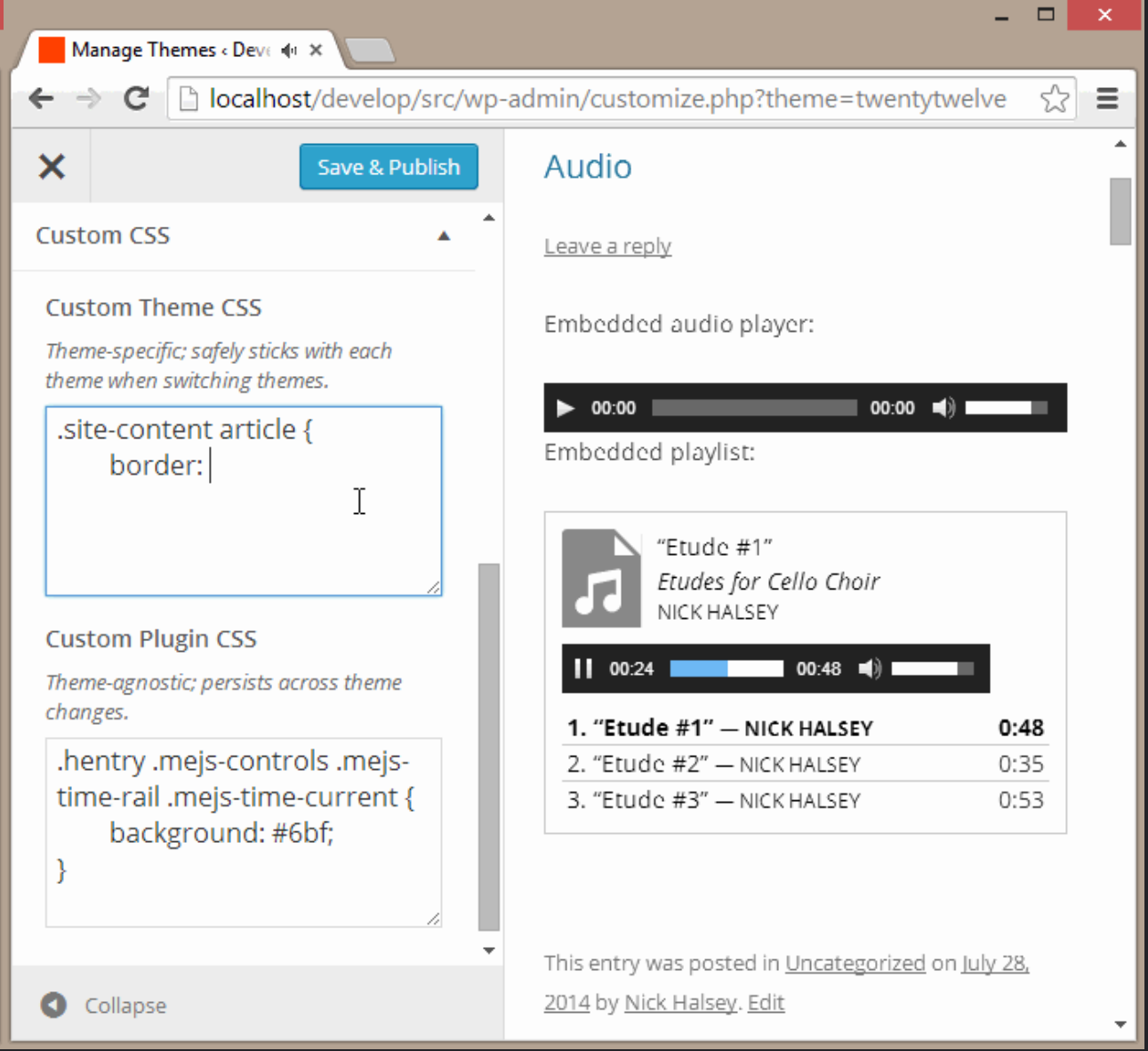
```
class Customize_Greeting_Control extends WP_Customize_Control {
	// ...

	function active_callback() {
		return is_front_page();
	}
}
```

Using postMessage for truly live-previewing settings



'transport' => 'refresh'



'transport' => 'postMessage'

```
function my_preview_js() {
      wp_enqueue_script( 'custom_css_preview',
                         'path/to/file.js',
                         array( 'customize-preview' ),
      );
}
add_action( 'customize_preview_init', 'my_preview_js' );
```

```
( function( $ ) {
	wp.customize( 'custom_theme_css', function( value ) {
		value.bind( function( to ) {
			$( '#custom-theme-css' ).html( to );
		} );
	} );
	wp.customize( 'custom_plugin_css', function( value ) {
		value.bind( function( to ) {
			$( '#custom-plugin-css' ).html( to );
		} );
	} );
} )( jQuery );
```

Custom Controls

In 4.0, odds are you don’t need most that you’ve created before.

```
class WP_New_Menu_Customize_Control extends WP_Customize_Control {
	public $type = 'new_menu';

	/**
	 * Render the control's content.
	 */
	public function render_content() {
		?>
		<span class="button button-primary" id="create-new-menu-submit" tabindex="0"><?php _e( 'Create Menu' ); ?></span>
		<?php
	}
}
```

See `wp-includes/class-wp-customize-control.php` for more info

```
(function( wp, $ ){
	var api = wp.customize;
	api.Menus.NewMenuControl = api.Control.extend({
		ready: function() {
			// Do stuff.
		},
		toggle: function ( active ) { // Used when the active_callback state changes.
			if ( active ) {
				this.container.fadeIn(); // Default is slide, fade instead.
			} else {
				this.container.fadeOut();
			}
		},
	});
	$.extend( api.controlConstructor, {
		new_menu: api.Menus.NewMenuControl
	});
})( window.wp, jQuery );
```

Custom_Customize_Control::$type

See wp-admin/js/customize-controls.js for more info

Core Custom Controls

Core comes with a few ready-to-use custom controls.

- WP_Customize_Color_Control
- WP_Customize_Upload_Control
  - *Needs to be re-done in core.*
- WP_Customize_Image_Control
  - *Needs to be re-done in core.*

```
$wp_customize->add_control(
	new WP_Customize_Color_Control(
		$wp_customize, // WP_Customize_Manager
		'accent_color', // Setting id
		array( // Args, including any custom ones.
			'label'    => __( 'Accent Color' ),
			'section'  => 'colors',
		)
	)
);
```

Customizer Panels

Creating distinct control contexts

Manage Themes ‹ Develo ×

localhost/develop/src/wp-admin/customize.php?theme=figureground

Save & Publish

You are customizing
Develop WordPress

- Site Title & Tagline
- Menus
- Colors
- Header Image
- Figure/Ground
- Footer
- Widgets
- Static Front Page
- Custom CSS
- Settings

Collapse

Develop WordPress

07 27 14

Unattached Image in Contents, No Featured Image

Panels

- Distinct contexts for the Customizer.
- Rarely if ever needed for themes – integrate with core sections and add a couple custom sections if needed.
- *Don't* use because you have too many options.

Examples:

- Widgets
- Menus
- Themes
- Settings (think an adaptation of the settings menu from wp-admin)
- Posts (think all metaboxes on post.php are sections in a panel)

```
$description = __( '<p>This screen is used for managing your custom navigation menus. You can add pages, posts, categories, tags, and custom links to your menus.</p><p>Menus can be displayed in locations defined by your theme, and also used in sidebars by adding a "Custom Menu" widget on the Widgets screen.</p>' );
$wp_customize->add_panel( 'menus', array(
	'title'       => __( 'Menus' ),
	'description' => $description,
	'priority'    => 160, // Mixed with top-level-section hierarchy.
) );
$wp_customize->add_section( $section_id , array(
	'title' => $menu->name,
	'panel' => 'menus',
) );
```

Panels and Sections are Containers

WP_Customize_Panel and WP_Customize_Section
(will) extend WP_Customize_Container

https://core.trac.wordpress.org/ticket/29197

Custom Settings

```
$wp_customize->add_setting( $nav_menu_setting_id, array(
	'type'    => 'nav_menu',
	'default' => $item_ids,
) );
```

Now the setting is not saved or previewed as an option or a theme mod.

```
function menu_customizer_update_nav_menu( $value, $setting ) {
	$menu_id = str_replace( 'nav_menu_', '', $setting->id );
	// ...
	foreach( $value as $item_id ) { // $value is ordered array of item ids.
		menu_customizer_update_menu_item_order( $menu_id, $item_id, $i );
		$i++;
	}
}
add_action( 'customize_update_nav_menu', 'menu_customizer_update_nav_menu', 10, 2 );
```

Custom Setting Types

No saving/previewing by default.

Actions (setting instance is passed to each):

`customize_update_$setting->type`

`customize_preview_$setting->id*`

`customize_preview_$setting->type*`

See `wp-includes/class-wp-customize-setting` for more info

The Future of the Customizer

Menu Customizer

https://wordpress.org/plugins/menu-customizer

Theme Customizer

http://celloexpressions.com/blog/proposed-wordpress-customizer-theme-switching-ux/

Post Customizer

Front-end-editor Integration

https://github.com/x-team/wp-customize-posts

https://wordpress.org/plugins/wp-front-end-editor/

UI Improvements & Customizing with a front-end context

Performance & Mobile

Media

Better JS API

Partial Preview Refreshes

Enabling broader usage of postMessage, without touching JS

Customizer Roadmap

Coming soon to make/core

Contributors Needed!

Weekly office hours in #wordpress-dev on IRC

Fridays at Noon PDT