James Farmer – January 8, 2014

91 Comments

How To Make Twenty Fourteen (Or Any Other WP Theme!) Super

In a recent review of WordPress' latest default theme, WPMU DEV's Chris Knowles called Twenty Fourteen a "flawed beauty."

In that article, Chris recommended a number of potential improvements, and together we've put together this ultimate guide to addressing those flaws.

And what's more, these tips and techniques can be used in **practically any WordPress theme**.

So, break out the cape, fire up your favorite editor, and give Twenty Fourteen a superhero makeover.

In this article, we are going to build a child theme that implements the improvements that Chris recommended in his review of Twenty Fourteen, namely:

1. Centering the design, instead of the default left-alignment
2. Adding a slide-out sidebar for mobile devices, rather than simply moving the default sidebar below the content
3. Improving the header bar so that logos and other elements can be included
4. Enhancing the featured content functions in order to:
    - allow a different layout on a mobile device
    - specify how many posts should be used for each layout
    - include custom post types in the layouts
    - automatically scroll the slider

In working your way through these four main areas, you'll not only have a truly functional, great looking Twenty Fourteen, but you'll also have picked up plenty of tips and tricks that you can apply to virtually any other WordPress theme.

And to make your life even easier we've given you two paths to glory:

1. Skip to the end of this article and download the new zipped theme file
2. Create it yourself using the following awesome tutorial, and learn as you go

Still here? Great! So let's begin…

Before We Begin : Create a Child Theme

As we know, if we want to customize a theme, then it's essential to use a child theme. We'll be creating quite a few files in this tutorial, so go and create a child theme now, including:

- The mandatory *style.css* file
- */js* and *css* folders
- An empty *functions.php* file in the theme folder

If you don't know how to create a child theme then be sure to read Rae's excellent tutorial on how to create a child theme. The child theme's style.css needs to point to the original Twenty Fourteen theme:

```
/*
Theme Name: Awesome Twenty Fourteen
Template: twentyfourteen
*/
@import url("../twentyfourteen/style.css");
/* Theme customization starts here */
```

Once you have created your child theme, activate it just to make sure it works. Obviously, at this stage it should look and behave exactly like its parent.

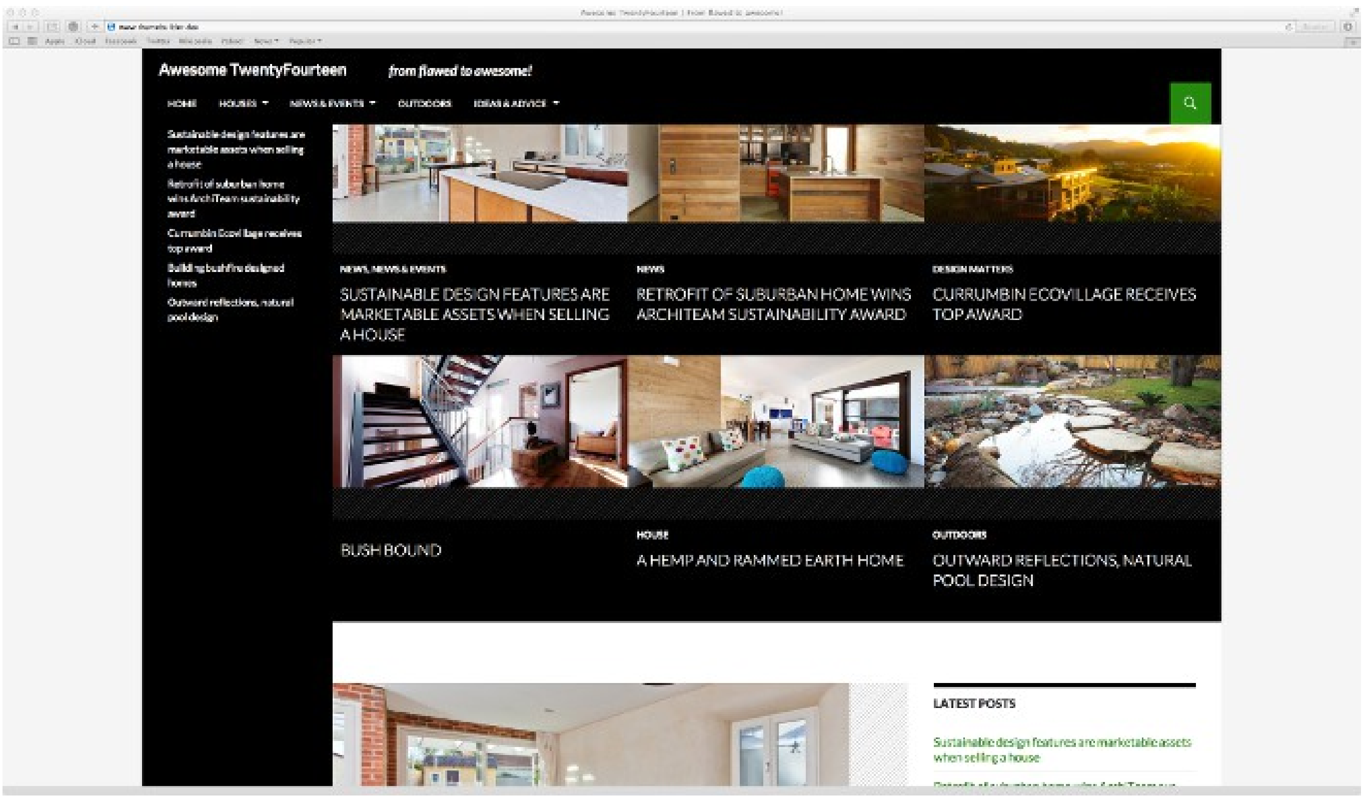
1. How To Center Twenty Fourteen

TRY FREE FOR 7 DAYS →

Unusually for a fixed-width design, Twenty Fourteen is left-aligned which can lead to large expanse of white space to the right in a large browser window and a distinctly lop-sided feel.

How To Fix It?

A simple update to the stylesheet to center the design.


Awesome TwentyFourteen
from flawed to awesome!
HOME
HOUSES
NEWS & EVENTS
OUTDOORS
IDEAS & ADVICE
SUSTAINABLE DESIGN FEATURES ARE MARKETABLE ASSETS WHEN SELLING A HOUSE
RETROFIT OF SUBURBAN HOME WINS ARCHITEAM SUSTAINABILITY AWARD
CURRUMBIN ECOVILLAGE RECEIVES TOP AWARD
BUSH BOUND
A HEMP AND RAMMED EARTH HOME
OUTWARD REFLECTIONS, NATURAL POOL DESIGN
LATEST POSTS

Centering fixed-width designs do seem to make them feel more balanced

The Solution…

Centering With Style

The outermost container of the theme's design has the class *site,* and so we can center it by using the standard method of setting the container's *margin* to *0 auto*.

Add the following to the *style.css* you just created in your child theme:

```css
.site {
margin: 0 auto;
}
```

If you want to play with the background color, then you can set this in the theme options available

If you would like to customize the colors further, take a look at the Fourteen Colors plugin. It allows the modification of the theme's color scheme, automatically adjusting contrast to maintain accessibility and legibility.

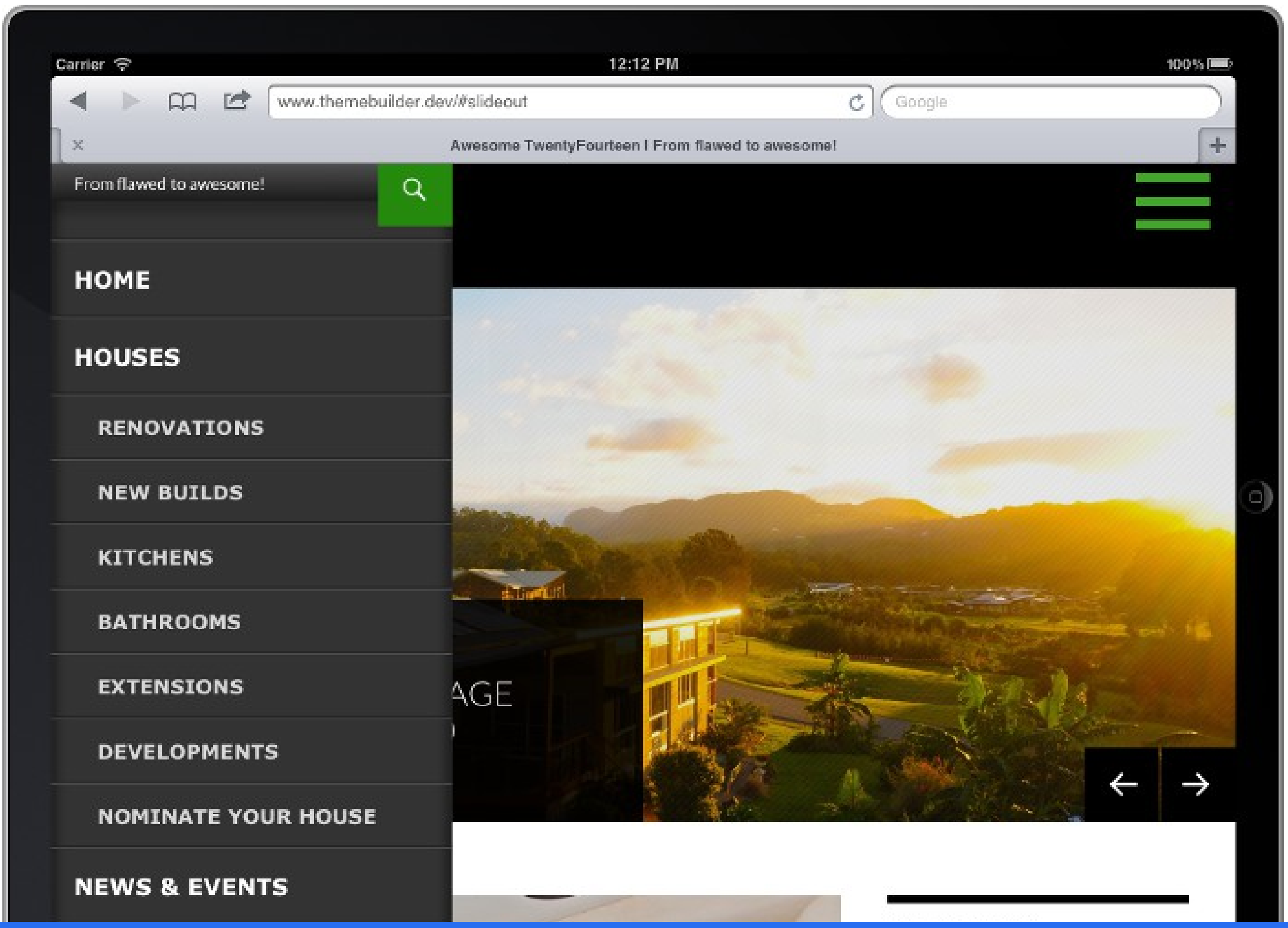
2. How To Add a Slideout Sidebar

What's The Problem?

One of the big issues with the original Twenty Fourteen theme is that on mobile devices (including tablets) the sidebar becomes an initially invisible footer. Not a great response if you have essential content in the sidebar such as navigation.

How To Fix It?

Add a mobile-specific sidebar to the theme that slides out. We'll also change the mobile menu button to activate the slide-out rather than show the menu and add a new mobile-specific menu location for easy inclusion of mobile-only navigation in the new sidebar.



TRY FREE FOR 7 DAYS →

A slide-out menu is far more useful on mobile devices

Using a mobile-specific sidebar also allows greater flexibility when targeting content to a platform.

The Solution…

We are going to implement the slide-out sidebar using Alberto Valero's Sidr plugin for jQuery. If you are interested in adding a slide-out sidebar to any theme then check out Chris's earlier tutorial.

Before we get to the plugin itself, though, we have to set up the components for the new sidebar.

Create a New Mobile Widget Area

Add the following code to your *functions.php* file to add a separate widget area for the new slide-out sidebar:

```
//Add mobile sidebar
function awesome_2014_mobile_widget_area() {
register_sidebar( array(
'name' => __( 'Mobile Sidebar', 'awesome_2014' ),
'id' => 'sidebar-mobile',
'description' => __( 'Slideout sidebar for mobile devices.',
'awesome_2014' ),
'before_widget' => '
<aside id="%1$s" class="widget %2$s mobile-widget">',
'after_widget' => '</aside>

',
'before_title' => '
<h1 class="widget-title">',
'after_title' => '</h1>

',
) );
}
add_action( 'widgets_init', 'awesome_2014_mobile_widget_area' );
```

Notice that the class *mobile-widget* is added to the widget's container to allow these widgets to be styled separately from the other widgets in the theme.

Add A Mobile-only Menu Location

As we are hijacking the header's menu button to slide out our mobile specific sidebar instead of displaying the top bar navigation, we need to include a menu in the new sidebar sidebar. To provide maximum flexibility we'll add a "mobile" option to the existing menu locations which allow us to

Add the following code to *functions.php*:

```
//add mobile menu
function awesome_2014_setup() {
register_nav_menus( array(
'mobile' => __( 'Mobile menu in left sidebar', 'awesome_2014' ),
) );
}
add_action( 'after_setup_theme', 'awesome_2014_setup' );
```

This simply registers the new location and allows you to assign a menu to the location. Go to *Themes* > *Menus* in the WordPress Admin interface and add a menu to the mobile menu location.

Outputting the Mobile Sidebar

Now that we have the new widget area and the new menu, it's time to create ensure that the correct sidebar is output.

What we are actually going to do is override Twenty Fourteen's *sidebar.php* template and use the *wp_is_mobile()* function to output the slideout sidebar if appropriate.

Copy the *sidebar.php* template from the Twenty Fourteen folder to the child theme folder, open it and replace the existing code with the following:

```
<?php /** * The Sidebar containing the main widget area * */ ?>

<?php if ( wp_is_mobile() ) : // display mobile sidebar ?>
<div id="slideout" class="sidr">
<div id="slideout-top"><?php $description = get_bloginfo(
'description', 'display' ); if ( ! empty ( $description ) ) : ?>
<h2 class="site-description"><?php echo esc_html( $description ); ?>

<?php endif; ?>
<div class="search-toggle"><a class="screen-reader-text"
href="#search-container"><?php _e( 'Search', 'twentyfourteen' ); ?>

<div id="search-container" class="search-box-wrapper hide">
<div class="search-box"><?php get_search_form(); ?>

</div>
</div>
<?php if ( has_nav_menu( 'mobile' ) ) : ?>

<nav class="navigation site-navigation" role="navigation"><?php
wp_nav_menu( array( 'theme_location' => 'mobile' ) ); ?>

<?php endif; ?>

<?php if ( is_active_sidebar( 'sidebar-mobile' ) ) : ?>
```

```

<!-- #mobile-sidebar -->
<?php endif;?>

</div>
<!-- #seconday-mobile -->

<?php else: // display normal sidebar ?>
<div id="secondary"><?php if ( has_nav_menu( 'secondary' ) ) : ?>

<?php wp_nav_menu( array( 'theme_location' => 'secondary' ) ); ?>

<?php endif; ?>

<?php if ( is_active_sidebar( 'sidebar-1' ) ) : ?>
<div id="primary-sidebar" class="primary-sidebar widget-area"
role="complementary"><?php dynamic_sidebar( 'sidebar-1' ); ?>

<!-- #primary-sidebar -->
<?php endif; ?>

</div>
<!-- #secondary -->

<?php endif; ?>
```

This new sidebar uses the device detection function *wp_is_mobile()* to output a different sidebar for mobile devices. If you want to learn more about using device detection and learn how to modify this to only change on phones, but not tablets, check out our article on adaptive layout changes for WordPress themes.

The mobile sidebar outputs the mobile menu widget area and menus. It also contains the search toggle and search area HTML from the header, which we will remove from the mobile header in a later step.

The mobile sidebar also places the id of *#slideout* and the class of *sidr* on the outermost container of the sidebar.

Rearranging The Header

Much like we did with the sidebars, we are going to use *wp_is_mobile()* to output two different headers, depending on the device making the request.

Whilst the desktop header will remain unchanged, the mobile header will allow the slide-out action to be triggered by the menu icon and prevent the default mobile menu from displaying.

This time we need to override the parent header template, so create a *header.php* file in the child theme root folder and add the following:

```
<?php
/**
* The Header for our theme
*
* Displays all of the <head> section and everything up till
<div id="main">
*/
?><!DOCTYPE html>
<!--[if IE 7]>
<html class="ie ie7" <?php language_attributes(); ?>>
<![endif]-->
<!--[if IE 8]>
<html class="ie ie8" <?php language_attributes(); ?>>
<![endif]-->
<!--[if !(IE 7) | !(IE 8) ]><!-->
<html <?php language_attributes(); ?>>
<!--<![endif]-->
<head>
<meta charset="<?php bloginfo( 'charset' ); ?>">
<meta name="viewport" content="width=device-width">
<title><?php wp_title( '|', true, 'right' ); ?></title>
<link rel="profile" href="http://gmpg.org/xfn/11">
<link rel="pingback" href="<?php bloginfo( 'pingback_url' ); ?>">
<!--[if lt IE 9]>
<img src="data:image/gif;base64,R0lGODlhAQABAIAAAAAAAP///yH5BAEAAAAALAA
preserve="%3Cscript%20src%3D%22%26lt%3B%3Fphp%20echo%20get_template_dir
data-mce-resize="false" data-mce-placeholder="1" class="mce-object" wic
<![endif]-->
<?php wp_head(); ?>
</head>

<body <?php body_class(); ?>>

<div id="page" class="hfeed site">
<?php if ( get_header_image() ) : ?>

<div id="site-header">
<a href="<?php echo esc_url( home_url( '/' ) ); ?>" rel="home">
<img src="<?php header_image(); ?>" width="<?php echo get_custom_header
alt="">
</a>
</div>

<?php endif; ?>

<?php if ( wp_is_mobile() ) : ?>


<header id="masthead" class="site-header" role="banner">

<div class="header-main">

<h1 class="site-title"><a href="<?php echo esc_url( home_url( '/' ) );
```

TRY FREE FOR 7 DAYS →

```
<h2 class="topbar-description"><?php echo esc_html( $description ); ?><

<?php endif; ?>
</div>

<a class="screen-reader-text skip-link" href="#content"><?php _e( 'Skip
<a id="menu-toggle" class="second" title="<?php _e( 'Click To Show Side
genericon-menu"></span></a>
</header>

<!-- #masthead -->

<?php else: ?>


<header id="masthead" class="site-header" role="banner">

<div id="big-top">

<div class="header-main">

<h1 class="site-title"><a href="<?php echo esc_url( home_url( '/' ) );

</div>

<?php $description = get_bloginfo( 'description', 'display' ); if ( ! e

<h2 class="topbar-description"><?php echo esc_html( $description ); ?><

<?php endif; ?>
</div>



<nav id="primary-navigation" class="site-navigation primary-navigation"

<div class="search-toggle">
<a href="#search-container" class="screen-reader-text"><?php _e( 'Searc
</div>


<div id="search-container" class="search-box-wrapper hide">

<div class="search-box">
<?php get_search_form(); ?>
</div>

</div>


<h1 class="menu-toggle"><?php _e( 'Primary Menu', 'twentyfourteen' ); ?

<a class="screen-reader-text skip-link" href="#content"><?php _e( 'Skip
<?php wp_nav_menu( array( 'theme_location' => 'primary', 'menu_class' =

</nav>


</header>
```

TRY FREE FOR 7 DAYS →

```
<?php endif; ?>


<div id="main" class="site-main">
```

```
<div id="page" class="hfeed site">

<div id="main" class="site-main">
```

You will notice that we have replaced the responsive menu toggle in the mobile header with our own toggle, which will be hooked up to open the slide-out sidebar. We have also removed the search toggle, which now appears in the slide-out sidebar.There are a few extra bits and pieces in here that we are going to make use of in the next part of this tutorial when we address the header bar, but don’t worry about that for now.

Yet More Style Fine-Tuning

In order to get the correct layout for the mobile header, we need to fine-tune the styling. Go back to the *style.css* and add the following:

```
/* Make room for the search toggle and site description in mobile
sidebar */
#slideout-top {
height:48px;
}

#slideout-top .site-description {
display: block;
text-align: left;
font-family: lato;
width: 205px;
overflow: hidden;
}

#slideout-top .search-toggle {
margin-right: 0;
}

/* Override Sidr Style For 2014 Search */
#slideout-top .search-box .search-field {
background-color: white;
color: black;
}

/* Keep mobile toggle on the right and resize the genricon */
#menu-toggle {
float: right;
display: inline;
}
```

```
padding: 0 1em 0 0;
z-index: 999;
}

/* enhancing the topbar */
#big-top {
display: block; }

#masthead {
height: 88px;
position: fixed;
top: 0;
}

.header-main {
display: inline;
}

.site-title{
margin: 0 0 0 20px;
}

nav#primary-navigation {
width: 100%;
display: block;
top: 40px;
left: 0px;
position: absolute;
padding-left: 30px;
}

.topbar-description {
color: #fff;
display: inline;
font-weight: bold;
font-size: 100%;
text-transform: lowercase;
line-height: 50px;
vertical-align: baseline;
margin-left: 20px;
font-style: italic;
}
```

Among other things, this CSS keeps the icons the default Twenty Fourteen green color. You can change the color by adding a color property for *#menu-toggle .genericon*. It also carves out a space for the site description and search toggle at the top of the of the slide-out sidebar.

Activating The Slideout Sidebar

Now that we've created all the components for the sidebar, we can actually get around to adding the code to activate it. First, though, we need the plugin itself, so download the plugin, unzip it, rename the folder *sidr* and copy it to the *js* folder in the child theme.

```
/* hook up the Sidr functionality */

(function($) {

$(document).ready(function(){

// hook up the left side menu

$("#menu-toggle").sidr({
name: "slideout",
side: "left",
displace: false,
});

});

})(jQuery);
```

This javascript hooks up the menu toggle icon in the mobile header to the mobile slide-out sidebar. Tapping on the icon will slide the sidebar in or out depending on its current visibility.

Adding the Javascript and CSS

Almost there. Everything is now in place, we just need to let WordPress know about it which means enqueuing the Sidr javascript file, our activation javascript file and a stylesheet for the slide-out sidebar. We are going to just use the dark theme that comes with Sidr.

Add the following code to the functions.php file:

```
/**
* Add slideout Sidebar
*/
// Enqueue the css/js for slideout
function awesome_2014_add_slideout() {
wp_enqueue_script( 'sidr',
get_stylesheet_directory_uri().'/js/sidr/jquery.sidr.min.js',
array('jquery'), null, true );
wp_enqueue_script( 'slideout',
get_stylesheet_directory_uri().'/js/slideout.js', array('sidr'), null, t
);
wp_enqueue_style( 'slideout',
get_stylesheet_directory_uri().'/js/sidr/stylesheets/jquery.sidr.dark.cs
}

// add only if is mobile
if ( wp_is_mobile() ) {
add_action( 'wp_enqueue_scripts', 'awesome_2014_add_slideout', 10 );
}
```

This uses the wp_enqueue_scripts action to ensure that the necessary javavscript and CSS files are and that the action is only called if a mobile device is making the call.

If you want to change the styling for the slide-out sidebar then best practice would be to copy the dark CSS to a new location, edit it there and then change the location in the call to wp_enqueue_style.

That was quite involved. We set up a new sidebar and menu location for the Admin interface, installed the Sidr plugin and added some styling to make it all work and look presentable.

Make sure you've assigned some widgets to the mobile sidebar and/or a menu to the mobile menu location, and then view your test site on a tablet or phone (or on an emulator if you are working locally). If all is well, then clicking on the 3-bar menu icon will slide out the menu, and clicking on the same icon will close it.

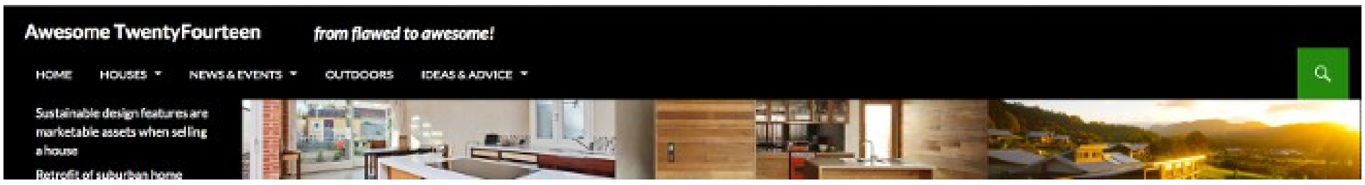
3. How To Improve the Header

What's The Problem?

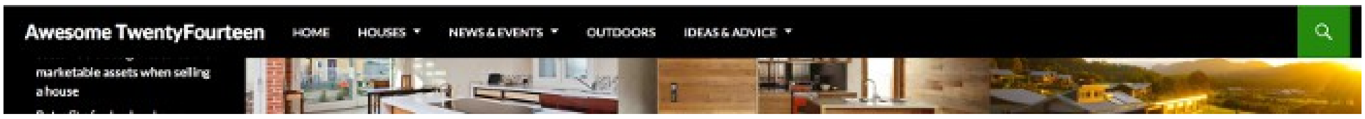
It's too short. You'll have a hard time getting a logo in there or any other elements such as links to social media presences; there's not even any room for the site's tagline.

How To Fix It?

Make it bigger when the page first loads and shrink when the user starts scrolling. When the header shrinks, elements that are not required will be hidden, and those that are remaining will be relocated, all with eye-catching animation.


Awesome TwentyFourteen
from flawed to awesome!
HOME
HOUSES
NEWS & EVENTS
OUTDOORS
IDEAS & ADVICE
Sustainable design features are marketable assets when selling a house
Retrofit of suburban home

The initial header view is now much more accommodating



TRY FREE FOR 7 DAYS →

The Solution…

Preparing The Header

We don't actually need to do anything for the header because we incorporated the changes we need into the *header.php* template when we updated it for the slide-out sidebar. So, on to the next step.

Outputting The Required CSS and Javascript

Since this will only affect non-mobile devices, we'll place the css and jQuery in separate files from the main theme Javascript and CSS and only add them when the site is not being shown on a mobile device.

This requires creating a Javascript file in the child theme's *js* directory and adding some CSS to the style.css. We'll then *enqueue* the new Javascript file in it's own function hooked to the *wp_enqueue_scripts* action that is only added if *wp_is_mobile()* does not return true. No need to load additional Javascript on mobile devices when it is not needed, especially one such as this that makes continuous adjustments as the page is scrolled.

Add the following code to the *functions.php* file:

```
//enqueue topbar js
function awesome_2014_add_topbarjs() {
wp_enqueue_script( 'topbarjs',
get_stylesheet_directory_uri().'/js/topbar.js', array('jquery'), null,
true );
}
//add if is not mobile
if ( !wp_is_mobile() ) {
add_action( 'wp_enqueue_scripts', 'awesome_2014_add_topbarjs' );
}
```

Creating the CSS file

In order to get things to display properly when the page loads, we will do as much of the initial, full-size topbar as we can with CSS.

The basic strategy is to create two rows by ensuring #top-big is a block element, while the others are set to be inline elements. We actually set these rules when rearranging the header for the slide-out menu, so all we need to concern ourselves about now is resizing of the browser window.

Even though we are using an adaptive strategy for handling mobile navigation via the slide-out sidebar, we still need to ensure that our design responds to a user shrinking their desktop browser window which we can do by adding an appropriate *@media* query.

Once again, add this code to the *style.css* file:

```
/**Adjustments for the desktop topbar when screen is small**/
@media screen and (max-width: 768px) {
.search-toggle {
position: absolute;
top: -8px;
right: 0px;
}
#masthead {
height: 80px;
}
.topbar-menu {
display: block;
background-color: black;
}
.primary-navigation.toggled-on {
display: block;
padding: 48px 0 0 0px;
background-color: black;
}
.nav-toggled {
padding-top: 48px !important;
padding-left: 0px !important;
}
h2.topbar-description {
display:none;
}
}
```

Creating The Javascript File

The script that will rearrange the header when not scrolled the whole way to the top is powered by jQuery's *scrollTop()* function.

This function returns the position of the vertical scroll for a container, which in this case is the body. This will allow us to condense the scrollbar whenever the page is scrolled away from the top of the page by testing if the scroll position of the body is greater than zero and to perform the reverse and put everything back when the scroll position gets back to zero.

This script also includes two functions for ensuring that the topbar does not overlap the main content and the WordPress admin bar does not overlap the topbar. These corrections have to be done via jQuery since the size of the *#masthead* and *#wpadmin* bar can change based on scroll position and screen width. We can easily calculate their heights using jQuery's *height()* function and then use those values to add top margins with *css()*.

Save the following code in the */js* folder as *topbar.js*:

```
jQuery(document).ready(function($) {
//set element ID/classes in vars
var social = '.topbar-description';
var nav = 'nav#primary-navigation';
var headerMain = '.header-main';
var header = '#masthead';
var main = '#main';

//margin fix for masthead function
var mastFix = function() {
$(header).css({
marginTop: $('#wpadminbar').height() + 'px'
});
};
mastFix();
//ensure that #wpadminbar doesn't move
$('#wpadminbar').css( 'position', 'fixed' );

//function for changing sizes
$(function(){
$(header).data('size','big');
});

//the main scroll function
$(window).scroll(function(){
//set container of the nav element
var $nav = $(header);
//when scrolled away from top
if ($('body').scrollTop() || $('html').scrollTop() > 0) {
if ($nav.data('size') == 'big') {
mastFix();
$( social ).css('display', 'none');
$( nav ).css({
display: 'inline',
top: '0px',
});
$( headerMain ).fadeOut("fast");
$( nav ).animate({
paddingLeft: $( 'h1.site-title' ).width() + 45 + 'px',
}), {queue:false, duration:600};
$( nav ).css('top', '0px');
$nav.data('size','small').stop().animate({
height:'48px'
}, 600);
$( headerMain ).animate({
left:200, opacity:"show"}, 600);
};
}
//when scrolled back
else {
```

TRY FREE FOR 7 DAYS →

```
$( nav ).animate({
display: 'block',
top: '40px',
}), {queue:false, duration:600}; ;
$( nav ).animate({
paddingLeft: '30px',
}), {queue:false, duration:600};
$nav.data('size','big').stop().animate({
height:'88px'
}, 600);
}
}
});
//Function to fixing margin for #main
var marginFix = function() {
$(main).css({
marginTop: $(header).height() + 'px',
});
};
//do marginFix and again on window resize along with mastFix
marginFix();
$( window ).resize(function() {
marginFix();
mastFix();
});

}); //end jQuery noConflict wrapper
```

4. How To Make Featured Content More Flexible

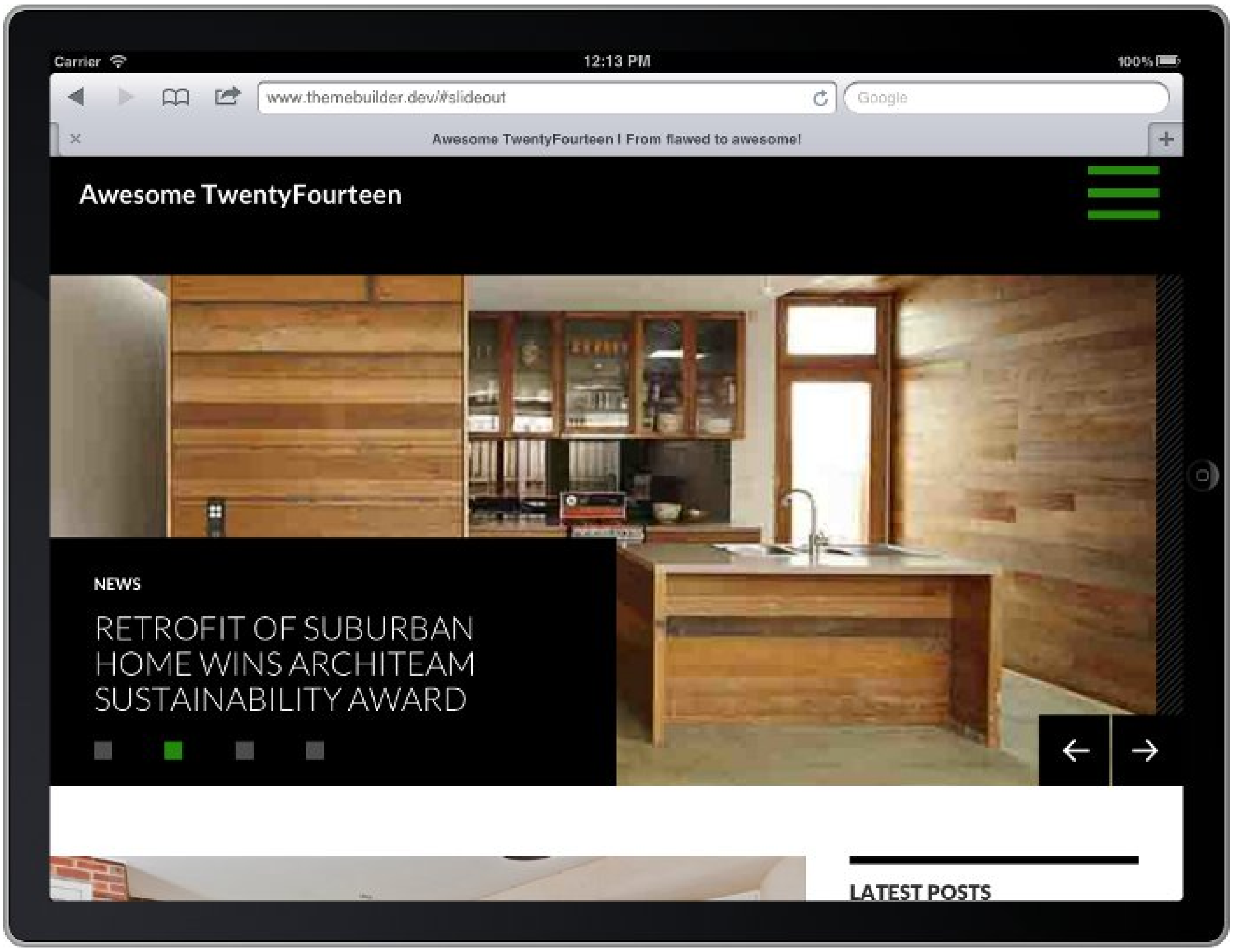
What's The Problem?

For many people, the implementation of the featured content in Twenty Fourteen is too basic. You can only specify a layout–grid or slider–for all platforms. You can't set how many posts are to be displayed, and custom post types cannot be included. The slider also only allows for manual scrolling. The javascript that powers the slider is based on WooThemes' Flexslider jQuery plugin. The Twenty Fourteen development team chose to remove all of the functionality that they didn't need from the plugin. This created a smaller file, but removes some of the cool features that Flexslider provides.

How To Fix It?

Add the capability to specify a layout for mobile devices, add the ability to specify how many posts should be displayed in each layout type, and extend the selection of featured content to include custom post types. To make it even more flexible, add these as options to the Featured Content section in the theme customization panel.

To address the automatic scrolling issue with the slider means replacing it with its original source code from WooThemes and tweaking the included functions.js file. It will also require creating a few more controls in the theme customizer.


Carrier
12:13 PM
100%
www.themebuilder.dev/#slideout
Google
Awesome TwentyFourteen | From flawed to awesome!
Awesome TwentyFourteen
NEWS
RETROFIT OF SUBURBAN HOME WINS ARCHITEAM SUSTAINABILITY AWARD
LATEST POSTS

Use a grid on a desktop and slider, that auto-scrolls, on your tablet

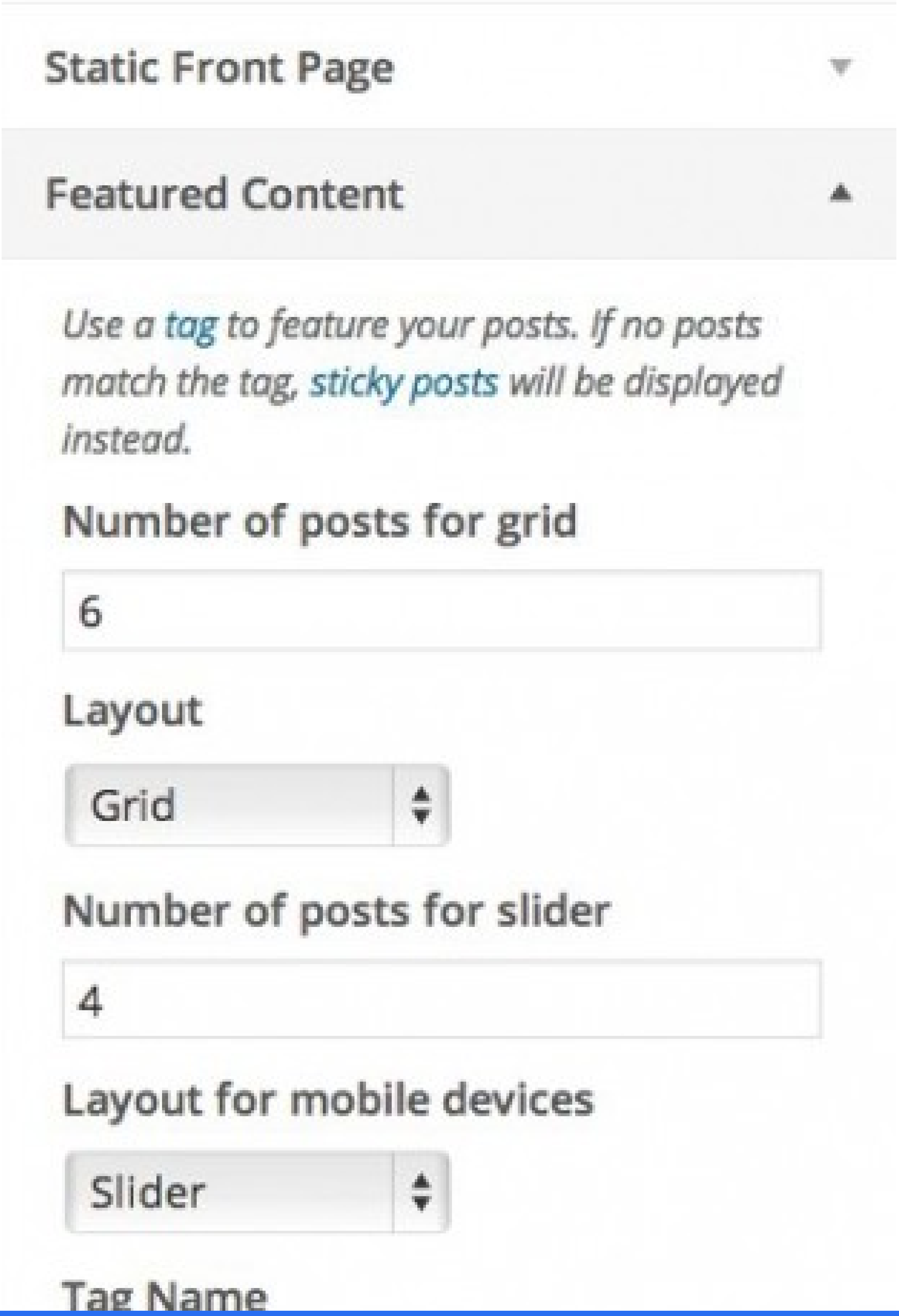
The Solution...

Adding Options To The Theme > Customize Panel

Adding these options will allow you to set the mobile layout and the number of posts for each layout in the Admin interface, removing the need for code changes if you want to change the settings.

We'll achieve this by making use of the Customize API to add more options to the Featured Content section in the theme customize screen (Theme > Customize).

```
function awesome_2014_customize_register() {

global $wp_customize;

//add extended featured content section

//add controls
$wp_customize->add_setting( 'num_posts_grid', array( 'default' => '6'
) );
$wp_customize->add_setting( 'num_posts_slider', array( 'default' =>
'6' ) );
$wp_customize->add_setting( 'layout_mobile', array( 'default' =>
'grid' ) );

$wp_customize->add_control( 'num_posts_grid', array(
'label' => __( 'Number of posts for grid', 'text-domain'),
'section' => 'featured_content',
'settings' => 'num_posts_grid',
) );

$wp_customize->add_control( 'num_posts_slider', array(
'label' => __( 'Number of posts for slider', 'text-domain'),
'section' => 'featured_content',
'settings' => 'num_posts_slider',
) );

$wp_customize->add_control( 'layout_mobile', array(
'label' => __( 'Layout for mobile devices', 'text-domain'),
'section' => 'featured_content',
'settings' => 'layout_mobile',
'type' => 'select',
'choices' => array(
'grid' => 'Grid',
'slider' => 'Slider',
),
) );
}

add_action( 'customize_register', 'awesome_2014_customize_register'
);
```


Static Front Page

Featured Content

Use a tag to feature your posts. If no posts match the tag, sticky posts will be displayed instead.

Number of posts for grid: 6

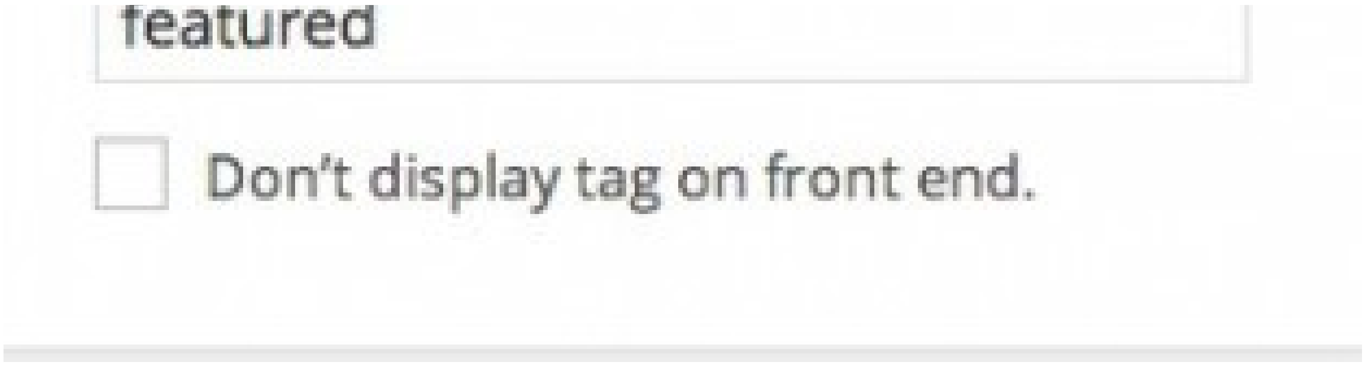
Layout: Grid

Number of posts for slider: 4

Layout for mobile devices: Slider

Tag Name

TRY FREE FOR 7 DAYS →



This code adds three new settings, with defaults, to the existing featured_content panel:

1. *num_posts_grid* – the number of posts to appear in the grid
2. *num_posts_slider* – the number of posts to appear in the slider
3. *layout_mobile* – which layout (grid or slider) to use on mobile devices

We can now get the value of these settings simply by calling *get_theme_mod* which offers far more flexibility than hardcoding the options into our *functions.php*.

Specifying A Different Layout For Mobile Devices

Now that we've added the option to set which layout mobile devices use, we need to ensure that it gets used when the home page is viewed on a smartphone or a tablet.

The key is to make sure that any call to *get_theme_mod* for the *featured_content_layout* setting returns the correct layout for the device. As always, there's a filter for that.

A filter can actually be set for every customization option by using the filter name *theme_mod_[option name]*, so in our case we'll be using the filter *theme_mod_featured_content_layout*.

Add this code to your *functions.php*:

```
function awesome_2014_theme_mod( $value ) {

if ( wp_is_mobile() ){
return get_theme_mod( 'layout_mobile', 'grid' );
} else {
return $value;
}
```

TRY FREE FOR 7 DAYS →

```
add_filter( 'theme_mod_featured_content_layout',
'awesome_2014_theme_mod' );
```

All the code does is check to see if the device is a mobile and, if it is, either the setting for the mobile layout is returned, or the default value of *grid*, if no *theme_mod* is set. If not, it returns the original value.

Add the above code to your *functions.php*, change the option for the mobile layout so that its different to the original layout, and view the site on a tablet or mobile.

Setting the Number of Posts For Each Layout

To bring back the number of posts as specified in the theme options we need to utilize another filter, this time *twentyfourteen_get_featured_posts,* to redo the query that generates the list of posts for the featured content.

Again, add this code to functions.php:

```
function awesome_2014_get_featured_posts( $posts ){

$fc_options = (array) get_option( 'featured-content' );

if ( $fc_options ) {
$tag_name = $fc_options['tag-name'];
} else {
$tag_name = 'featured';
}

$layout = get_theme_mod( 'featured_content_layout' );
$max_posts = get_theme_mod( 'num_posts_' . $layout, 2 );

$args = array(
'tag' => $tag_name,
'posts_per_page' => $max_posts,
'order_by' => 'post_date',
'order' => 'DESC',
'post_status' => 'publish',
);

$new_post_array = get_posts( $args );

if ( count($new_post_array) > 0 ) {
return $new_post_array;
} else {
return $posts;
}

}
```

The function starts by determining the tag name that is being used to identify featured content. For some reason, this is not stored with the other options and therefore is not accessible via the *get_theme_mod*. Instead it is contained in an array stored in a regular WordPress option.

Once the tag name is identified, the number of posts to return is determined by first finding out which layout is being displayed (our *theme_mod* filter above will kick in here and alter the layout if a mobile device) and then get the layout specific theme option.

The number of posts is then inserted into the query arguments and the query is executed. If posts are found, then these are used for the featured content. Otherwise, the original list of posts is used.

Add the above code to your *functions.php* file, change the number of posts for the grid and slider, and browse to the home page to see the number of posts change.

Including Custom Post Types In Post Selection

Now that we've added our own custom query for the featured posts, amending the code to include custom post types is as simple as adding one line of to the above code.

Add this element to the *$args* array definition:

```
'post_type' => array( 'post', '[custom post type]'),
```

This is simply a comma separated list, so add as many custom post types as you have. I haven't included this as a theme customization option as it felt more development orientated.

Making the Slider Auto-Scroll

When WooThemes' Flexslider was cut-down for use in Twenty Fourteen, one of the things that was removed was the slideshow functionality.

The "fix" is actually to use the full Flexslider code instead of the cut-down version, which is as simple as dequeuing the original slider javascript and enqueueing its replacement. But first you need to download Flexslider and move the *jquery.flexslider-min.js* into the child theme's *js* folder.

Now, add this code to *functions.php*:

```
wp_dequeue_script( 'twentyfourteen-script' );
wp_dequeue_script( 'twentyfourteen-slider' );

wp_enqueue_script( 'awesome_2014-script',
get_stylesheet_directory_uri() . '/js/functions.js', array( 'jquery'
), '' , true );
if ( is_front_page() && 'slider' == get_theme_mod(
'featured_content_layout' ) ) {
wp_enqueue_script( 'awesome_2014-slider',
get_stylesheet_directory_uri() . '/js/jquery.flexslider-min.js',
array( 'jquery', 'awesome_2014-script' ), '' , true );
wp_localize_script( 'awesome_2014-slider', 'featuredSliderDefaults',
array(
'prevText' => __( 'Previous', 'awesome_2014' ),
'nextText' => __( 'Next', 'awesome_2014' )
) );
}
}
add_action( 'wp_enqueue_scripts' ,
'awesome_2014_featured_content_scripts' , 999 );
```

This removes the default slider code and the functions script that, amongst other things, initiates the slider and replaces it with our versions. The Flexslider script will only be loaded if the slider is needed, which will avoid wasting resources to load it unnecessarily.

This code also removes both of the parent theme's functions.js file. In order to get the slider to work seamlessly with the full version of Flexslider and ensure that the existing CSS is applied, we do need to make a small tweak to the theme's functions.js file. Copy that file to the child theme's */js* folder, open it up and at the bottom replace the Initialize Featured Content Slider code with the following:

```
// Initialize Featured Content slider.
_window.load( function() {
if ( body.is( '.slider' ) ) {
$( '.featured-content' ).flexslider( {
selector: '.featured-content-inner > article',
controlsContainer: '.featured-content',
slideshow: true,
slideshowSpeed: 4500,
namespace: 'slider-',
} );
}
} );
```

This ensures that Flexslider is initiated and that slideshow (automatic scrolling) is switched on. Setting the namespace argument ensures that all the current styling for the slider in Twenty Fourteen gets applied to our new scroller. This code also sets the slideshow speed to 4.5 seconds. The default is 7 seconds, which is a bit long, but you can change this to your personal preference.

You will need to keep an eye on this file in the parent theme and check every time the theme is updated. If any new essential functionality is introduced, you will need to manually carry that across to your child theme version.

Refresh your home page and, assuming you've specified "slider" as your layout, you should find that your slider is now automatically scrolling.

You Made It!

This is a big post, and there's a lot to digest, but it's worthwhile because it makes for a pretty comprehensive makeover for the Twenty Fourteen theme and there are plenty of useful tips and techniques that are applicable to virtually any theme.

In Chris' original review he called Twenty Fourteen a "flawed beauty." With those flaws addressed, this child theme is now simply a "beauty," and in my opinion, it is awesome.

Download the Twenty Fourteen child theme: awesome_2014.zip.

Alternatively, I have been working on a Twenty Fourteen child theme that includes many of these changes, with the Star Wars inspired name of "The Falcon" which you can download from GitHub.

Photo credit: JD Hancock