WordPress
Special Features
About This Book

One of the great aspects of WordPress is its incredible adaptability. Using WordPress to the fullest means finding out about all the special features you can implement on your website. So without further adieu, here is an assortment of useful techniques. Find out more about WordPress’s responsive possibilities, such as how to build a responsive theme from the ground up and implementing your theme. Also learn how to speed up your site, integrate social-networking capabilities and manage events and author listings.

TABLE OF CONTENTS

Create A Responsive, Mobile-First WordPress Theme............................. 3
Do-It-Yourself Caching Methods With WordPress.................................... 14
Front-End Author Listing And User Search For WordPress...................... 22
How To Integrate Facebook, Twitter And Google+ In WordPress............. 36
Manage Events Like A Pro With WordPress............................................... 51
Adopting A Responsive Theme Is More Than Install-And-Go............... 65
Responsive Images With WordPress’ Featured Images.......................... 73
About The Authors .................................................................................... 80
Create A Responsive, Mobile-First WordPress Theme

BY ELLEN BAUER

Let’s assess the situation. WordPress is an extremely popular, flexible, easy to use and open-source blogging and CMS system. More and more mobile devices are flooding the market every day, changing the way people use the Internet. And the need is growing for more beautifully designed and coded WordPress themes that work well across all of these devices. So, what are we waiting for? Let’s get to work!

At first, the idea of designing and developing a fully responsive, mobile-ready WordPress theme might be overwhelming. You might be thinking, “How do I handle a responsive design with all of this flexible content that a WordPress theme has? What should I consider when designing for touch devices? And do I really have to get rid of drop-down menus and other hover elements on mobile devices?”

But after doing some research and looking more closely at some of the responsive WordPress themes and theme frameworks out there, you will probably wrap your head around the idea pretty quickly, and the evolving world of WordPress theme design will sound like a huge opportunity that you can’t wait to get started on.
It’s All About Preparation

Having a detailed design concept is even more important for a responsive WordPress theme than for a static-width theme. At this stage, you haven’t decided anything, so nothing will get in your way of creating a clever and practical layout that adapts smoothly to different screens.

First, consider what you want to achieve with your WordPress theme, which user group you are targeting, and what their needs are. With these considerations, you can create a list of useful elements for your layout.

CREATING THE THEME’S CONCEPT

Using this list, you can plan your theme by sketching the layout at various screen sizes.

When sketching, be aware that the layout widths you choose are only rough reference points to represent the common screen sizes of today’s smartphones, tablets and desktop computers. Your goal should always be to create a responsive design that adapts smoothly to a wide diversity of screen sizes.

Ethan Marcotte, author of *Responsive Web Design*¹, described his approach to responsive Web design in a recent interview², explaining:

---

I’m a big, big believer of matching breakpoints to the design, not to individual devices. If we’re after more future-proof responsive designs, we should stop thinking in terms of “320px,” “480px,” “768px,” or whatever — the Web’s so much more flexible than that, and those pixels are a snapshot of the Web as we know it today. Instead, we should focus on breakpoints tailored to the design we’re working on.

While working on your concept sketches, also think about which layout options to offer in the theme (such as header and sidebar options or multiple widget areas) and how they will adapt to different screen sizes as well.

![An optional sidebar element in a responsive layout.](image)

**TOOLS FOR CONCEPT SKETCHING**

Which tool you use to develop the theme’s concept is not important. Just choose one that allows you to work quickly and that doesn't interrupt your workflow.

If you feel most comfortable sketching on a piece of paper or in a notebook, go for it. You could also try sketching on an iPad using a popular app such as Paper³ by FiftyThree or Bamboo Paper⁴, together with a digital pen like Wacom’s Bamboo Stylus⁵. Working directly on a

---

tablet will make sharing your ideas later with the developer a lot easier. One of my all-time favorite articles is Mike Rohde’s “Sketching: The Visual Thinking Power Tool,” which promotes sketching as a simple visual tool for thinking.

A GOOD CONCEPT SAVES TIME
If you develop the concept precisely at the beginning of the project, you will save a lot of time and effort later in the design process. The layout will adapt to different screen sizes more intelligently if you have thought a lot about the design’s behavior before even opening Photoshop (or your software of choice).

Theme-Specific Challenges to Consider
Because designing a WordPress theme with very flexible content is quite a different challenge than designing a static website, at this early stage of the process you should find solutions to the following theme-specific problems:

1. WORDPRESS’ NAVIGATION MENU
Until responsive Web design found its way into WordPress theme designs, most themes seemed to rely on good old-fashioned drop-down

menus to give users multi-level navigation. But because drop-down menus rely on mouse hovering, they don’t work well on touch devices. We already have some smart solutions for developing responsive, touch device-ready navigation. Brad Frost has a very helpful resource comparing common solutions for responsive menus in his post “Responsive Navigation Patterns.”

2. RESPONSIVE LAYOUT OPTIONS

Most themes offer users at least some layout options, such as left or right sidebar, header widget and footer elements. To offer this kind of flexibility in a responsive theme, you will have to consider how all of the layout elements will behave on different screen sizes. For instance, if you want to offer a left sidebar option, consider that the content of this sidebar would appear above the main content area on mobile devices. In most cases, this wouldn’t be the best solution because mobile users want to read the most important content first (such as the latest blog post) without having to scroll down a sidebar.

3. FLEXIBLE WIDGET AREAS

Widget areas are another challenge for responsive designers. After all, designing one is not easy if you don’t know what kind of content the user will put in it. So, you need to make sure that the design works no matter which and how many widgets are used in the widget areas.

Enough Headaches. Let’s Get To The Fun.

Because you are creating a responsive website, designing the entire website pixel by pixel in Photoshop and then just handing it over to the developer would result in too static a design and too time-consuming a process.

WORKING WITH REFERENCE POINTS

Instead, the design process should be used to figure out the general look and feel of the theme. At this stage, you should also work more intensively on the challenges mentioned, such as responsive navigation, layout variations and flexible widget areas. How you prepare the design for further development will depend partly on the nature of the project and how closely you will work with

the developer. In general, showing your design in the three layout versions is a good starting point: smartphone, tablet and desktop. These “screenshots” can then be used as reference points for development.

DESIGNING IN THE BROWSER

Design details such as font sizes, white space and button styles can be defined later directly in the browser. Because browsers often treat these elements differently, designing and testing them directly in their final environments is way more efficient.

DESIGNING FOR TOUCH DEVICES

Because your design will also be used on touch devices, you have to consider the special requirements of these devices. Using a finger to navigate a website is entirely different than using a precise mouse cursor.

This is why buttons and form input fields need to be at the right size. Font sizes and white space should also be applied more generously, so that users can navigate easily and read content comfortably.

EXERCISE YOUR COMMUNICATION SKILLS

Staying in constant communication with the developer during the entire process is very important (i.e. if you are not the developer yourself). Especially in a responsive design process, incorporating the developer’s knowledge into your decisions will keep you from having to change things later on.
Development

After wrapping up the design process, the first decision to make is whether to code the theme from scratch or to use a blank or starter theme (such as Automattic’s Toolbox\(^8\) or the newer _s\(^9\) theme).

If you want to work with one of the popular responsive frameworks such as Twitter’s Bootstrap\(^10\) or ZURB’s Foundation\(^11\), then you could use a starter theme that already includes the framework, such as BootstrapWP\(^12\) or WordPress Foundation\(^13\). Another popular starter theme is Bones\(^14\), which uses 320 and Up\(^15\) as a mobile-first boilerplate.

Of course, the way you start a theme will always depend on the project and your personal preferences. But if you’re still learning, then a blank theme would serve as a solid foundation for development.

**GO MOBILE FIRST**

A smart approach is to design and develop for the smallest layout first (i.e. smartphones) and then work your way up to tablet and desktop screen sizes. To get further insight into the mobile-first approach to Web design, read the book *Mobile First*\(^16\) by Luke Wroblewski.

---

15. http://stuffandnonsense.co.uk/projects/320andup/
SUPPORTING MEDIA QUERIES IN OLD BROWSERS

With the smartphone layout as your default, you will need to rely on a JavaScript solution such as Respond.js\(^{17}\) to support media queries in old browsers (such as Internet Explorer 7 and 8).

Alternatively, you could add CSS classes for old IE browsers through conditional comments, and then add CSS styles to set a maximum width for old IE browsers outside of your media queries. You can find a detailed explanation of this method in the article “Leaving Old Internet Explorer Behind”\(^{18}\)."

IMAGES IN A RESPONSIVE THEME

With the release of high-pixel-density devices such as the new iPad and new MacBook Pro, you will also need to reconsider the images in your theme.

Alternatives to images would be to use a CSS solution or use icon fonts\(^ {19}\). Fewer images will also result in a much more lightweight theme, which will speed up performance on slow mobile Internet connections. Trent Walton shares his reflections on the Retina-optimization of Web design in his article “In Flux”\(^ {20}\)."

Test, Test, Test

Particularly when developing a responsive theme, testing your work live as soon and as often as possible is critical. This way, you can quickly correct styles during development as necessary. Also, test whether fonts are easy to read and whether images, gallery sliders and embedded elements such as video work correctly on different devices.

HOW TO TEST ON MOBILE DEVICES

Of course, checking your theme on one of the many screen-resolution-testing tools, such as Screenfly\(^ {21}\), during development is very helpful, too.

\(^{17}\) https://github.com/scottjehl/Respond
\(^{18}\) http://jonikorpi.com/leaving-old-IE-behind/
\(^{19}\) http://css-tricks.com/flat-icons-icon-fonts/
\(^{20}\) http://trentwalton.com/2012/05/08/in-flux/
\(^{21}\) http://quirktools.com/screenfly/
But because of the different behavior of mobile browsers, touchscreens and high-density screens, constantly testing your theme on actual devices is important.

Unless you work for a big company, finding ways to test your theme during the development process can be quite a challenge. Of course, you won’t be able to test on all of the devices out there, but besides the devices that you own, you could ask friends, family, other freelancers and coworkers to help you test. You can also visit your local electronics store to test on the devices there.

A helpful post with a lot of testing advice is part 5 of the recent “Build a Responsive Site in a Week” tutorial series on .NET magazine.

Responsive Theme Vs. Mobile Plugin

A mobile theme plugin such as the popular WPtouch plugin can be a great temporary solution to give mobile users a better experience on an existing website. In most cases, offering visitors an optimized mobile experience with the help of a plugin is probably better than not optimizing at all.

But in the long term, a fully responsive theme has many advantages to a plugin:

• The website can maintain its unique branding across all devices.
• Users will get the same experience on all devices and thus have less trouble navigating the website.
• The website will be easier to maintain (the administrator won’t need to install and update the plugin).

---

Conclusion

Responsive Web design is often still described as a trend. And some might quietly hope that the trend will pass sooner or later. But responsive Web design is so much more than a trend: it’s a new mindset, as has been said [25]:

It’s such a shame that Responsive design is often degraded to being a ‘Web design trend’. It isn’t. It’s a new mindset.

In a multiple-device world, where the Internet seems to be available everywhere, responsive Web design feels so much more like a natural process that is just starting to show its potential.

So, what should our job as theme designers and developers be? Because responsive WordPress themes are still so new and in constant development, we must not be afraid to start from scratch, search for improvements and continue learning. And let’s share our knowledge and experience with each other along the way.

Do-It-Yourself Caching Methods With WordPress

BY MILAN PETROVIĆ

There are different ways to make your website faster: specialized plugins to cache entire rendered HTML pages, plugins to cache all SQL queries and data objects, plugins to minimize JavaScript and CSS files and even some server-side solutions.

But even if you use such plugins, using internal caching methods for objects and database results is a good development practice, so that your plugin doesn’t depend on which cache plugins the end user has. Your plugin needs to be fast on its own, not depending on other plugins to do the dirty work. And if you think you need to write your own cache handling code, you are wrong. WordPress comes with everything you need to quickly implement varying degrees of data caching. Just identify the parts of your code to benefit from optimization, and choose a type of caching.

WordPress implements two different caching methods:

1. **Non-persistent**
   The data remains in the cache during the loading of the page. (WordPress uses this to cache most database query results.)
2. **Persistent**

This depends on the database to work, and cached data can auto-expire after some time. (WordPress uses this to cache RSS feeds, update checks, etc.)

**Non-Persistent Cache**

When you use functions such as `get_posts()` or `get_post_meta()`, WordPress first checks to see whether the data you require is cached. If it is, then you will get data from the cache; if not, then a database query is run to get the data. Once the data is retrieved, it is also cached. A non-persistent cache is recommended for database results that might be reused during the creation of a page.

The code for WordPress’ internal non-persistent cache is located in the `cache.php` file in the `wp-includes` directory, and it is handled by the `WP_Object_Cache` class. We need to use two basic functions: `wp_cache_set()` and `wp_cache_get()`, along with the additional functions `wp_cache_add()`, `wp_cache_replace()`, `wp_cache_flush()` and `wp_cache_delete()`. Cached storage is organized into groups, and each entry needs its own unique key. To avoid mixing with WordPress’ default data, using your own unique group names is best.

**EXAMPLE**

For this example, we will create a function named `d4p_get_all_post_meta()`, which will retrieve all meta data associated with a post. This first version doesn’t involve caching.

```php
function d4p_get_all_post_meta($post_id) {
    global $wpdb;

    $data = array();
    $raw = $wpdb->get_results("SELECT meta_key, meta_value FROM $wpdb->postmeta WHERE post_id = $post_id", ARRAY_A);

    foreach ($raw as $row) {
        $data[$row['meta_key']][] = $row['meta_value'];
    }

    return $data;
}
```
Every time you call this function for the same post ID, an SQL query will be executed. Here is the modified function that uses WordPress' non-persistent cache:

```php
function d4p_get_all_post_meta($post_id) {
    global $wpdb;

    if (! $data = wp_cache_get($post_id, 'd4p_post_meta')) {
        $data = array();
        $raw = $wpdb->get_results("SELECT meta_key, meta_value FROM $wpdb->postmeta WHERE post_id = $post_id", ARRAY_A);

        foreach ($raw as $row) {
            $data[$row['meta_key']][] = $row['meta_value'];
        }

        wp_cache_add($post_id, $data, 'd4p_post_meta');
    }

    return $data;
}
```

Here, we are using a cache group named `d4p_post_meta`, and `post_id` is the key. With this function, we first check to see whether we need any data from the cache (line 4). If not, we run the normal code to get the data and then add it to the cache in line 13. So, if you call this function more than once, only the first one will run SQL queries; all other calls will get data from the cache. We are using the `wp_cache_add` function here, so if the key-group combination already exists in the store, it will not be replaced. Compare this with `wp_cache_set`, which will always overwrite an existing value without checking.

As you can see, we've made just a small change to the existing code but potentially saved a lot of repeated database calls during the page's loading.

**IMPORTANT NOTES**

1. Non-persistent cache is available only during the loading of the current page; once the next page loads, it will be blank once again.
2. The storage size is limited by the total available memory for PHP on the server. Do not store large data sets, or you might end up with an “Out of memory” message.

3. Using this type of cache makes sense only for operations repeated more than once in the creation of a page.

4. It works with WordPress since version 2.0.

**Database-Driven Temporarily Persistent Cache**

This type of cache relies on a feature built into WordPress called the Transient API. Transients are stored in the database (similar to most WordPress settings, in the `wp_options` table). Transients need two records in the database: one to store the expiration time and one to store the data. When cached data is requested, WordPress checks the timestamp and does one of two things. If the expiration time has passed, WordPress removes the data and returns `false` as a result. If the data has not expired, another query is run to retrieve it. The good thing about this method is that the cache persists even after the page has loaded, and it can be used for other pages for as long as the transient’s expiration time has not passed.

If your database queries are complex and/or produce results that might not change often, then storing them in the transient cache is a good idea. This is an excellent solution for most widgets, menus and other page elements.

**EXAMPLE**

Let’s say we wanted an SQL query to retrieve 20 posts from the previous month, along with some basic author data such as name, email address and URL. But we want posts from only the top 10 authors (sorted by their total number of posts in that month). The results will be displayed in a widget.

When tested on my local machine, this SQL query took 0.1710 seconds to run. If we had 1000 page views per day, this one query would take 171 seconds every 24 hours, or 5130 seconds per month. Relatively speaking, that is not much time, but we could do much better by using the transient cache to store these results with an expiration time of 30 days. Because the results of this query will not change during the month, the transient cache is a great way to optimize resources.

Returning to my local machine, the improved SQL query to get data from the transient cache is now only 0.0006 seconds, or 18 seconds per month. The advantage of this method is obvious in this case: we’ve saved 85 minutes each month with this one widget. Not bad at all.
There are cases in which you could save much, much more (such as with very complex menus). More complex SQL queries or operations would further optimize resources.

Let’s look at the actual code, both before and after implementing the transient cache. Below is the normal function to get the data. In this example, the SQL query is empty (because it is long and would take too much space here), but the entire widget is linked to at the end of this chapter.

```php
function d4p_get_query_results() {
    global $wpdb;

    $data = $wpdb->get_results('// SQL query // ');

    return $data;
}
```

And here is the function using the transient cache, with a few extra lines to check whether the data is cached.

```php
function d4p_get_query_results() {
    global $wpdb;

    $data = get_transient('my_transient_key');

    if ($data === false) {
        $data = $wpdb->get_results('// SQL query // ');
        set_transient('my_transient_key', $data, 3600 * 24);
    }

    return $data;
}
```

The function `get_transient` (or `get_site_transient` for a network) needs a name for the transient record key. If the key is not found or the record has expired, then the function will return `false`. To add a new transient cache record, you will need the record key, the object with the data and the expiration time (in seconds), and you will need to use the `set_transient` function (or `set_site_transient` for a network).

If your data changes, you can remove it from the cache. You will need the record key and the `delete_transient` function (or `delete_site_transient` for a network). In this example, if the post in
the cache is deleted or changed in some way, you could delete the cache record with this:

```php
delete_transient('my_transient_key');
```

**IMPORTANT NOTES**

1. The theoretical maximum size of data you can store in this way is 4 GB. But usually you would keep much smaller amounts of data in transient (up to couple of MB).

2. Use this method only for data (or operations) that do not change often, and set the expiration time to match the cycle of data changes.

3. In effect, you are using it to render results that are generated through a series of database queries and storing the resulting HTML in the cache.

4. The name of the transient record may not be longer than 45 characters, or 40 characters for “site transients” (used with multi-sites to store data at the network level).

5. It works with WordPress since version 3.0.

**Widget Example: Using Both Types Of Cache**

Based on our SQL query, we can create a widget that relies on both caching methods. These are two approaches to the problem, and the two widgets will produce essentially the same output, but using different methods for data retrieval and results caching. As the administrator, you can set a title for the widget and the number of days to keep the results in the cache.

Both versions are simple and can be improved further (such as by selecting the post’s type or by formatting the output), but for this chapter they are enough.

**RAW WIDGET**

The “raw” widget version stores an object with the SQL query results in the transient cache. In this case, the SQL query would return all columns from the `wp_posts` table and some columns from the `wp_users` table, along with information about the authors. Every time the widget loads, each post from our results set would get stored in the non-persistent cache object in the standard `posts` group, which is the same one used to store posts for normal WordPress operations. Be-
cause of this, functions such as `get_permalink()` can use the cached object to generate a URL to post. Information about the authors from the `wp_users` table is used to generate the URL for the archive of authors' posts.

This widget is located in the `method_raw.php` file in the `d4p_sa_method_raw` class. The function `get_data()` is the most important part of the widget. It attempts to get data from the transient cache (on line 52). If that fails, `get_data_real()` is called to run the SQL query and return the data. This data is now saved into the transient cache (line 56). After we have the data, we store each post from the set into the non-persistent cache. The `render` function is simple; it displays the results as an unordered list.

### Rendered Widget

The previous method works well, but it could have one problem. What if your permalink depends on categories (or other taxonomies) or you are running a query for a post type in a hierarchy? If that is the case, then generating a permalink for each post would require additional SQL queries. For example, to display 20 posts, you might need another 20 or more SQL queries. To fix the problem, we’ll change how we get the data and what is stored in the transient cache.

The second widget is located in the `method_rendered.php` file in the `d4p_sa_method_rendered` class. Within, the names of class methods are the same, so you can easily see now the difference between the two widgets. In this case, the transient cache is used in the `render()` method. We’re checking for cached data, and if that fails we use `get_data()` to get the data set and generate a rendered list of results.

Now, we are caching the rendered HTML output! No matter how many extra SQL queries are needed to generate the HTML (for permalinks or whatever else you might need in the widget), they are run only once, and the complete HTML is cached. Until the cache expires, we are always displaying HTML rendered without the need for any additional SQL queries or processing.

### Download the Widget

You can download this D4P Smashing Authors plugin, which contains both widgets.

### Conclusion

As you can see, implementing one or both caching methods is easy and could significantly improve the performance of your plugin. If a user of
your plugin decides to use a specialized caching plugin, all the better, but make sure that your code is optimized.
Front-End Author Listing And User Search For WordPress

BY CRISTIAN ANTOHE

This chapter will guide you through the process of creating a front-end page in WordPress that lists your authors. We'll discuss why you would want to do this, we'll introduce the `WP_User_Query` class, and then we'll put it all together.

User Engagement And WordPress

At its core, WordPress is a rock-solid publishing platform. With a beautiful and easy to use interface, and support for custom post types and post formats, publishers have the flexibility to do what they do best: write content.

However, WordPress is lacking in social interaction between content authors and readers. BuddyPress is trying to solve this, but I believe it's going in the wrong direction by trying to be a full-fledged social network.

A big phrase in the publishing world is “user engagement.” This is about getting readers to spend more time on the website, actively searching for content and even generating their own. While one could write a few books on the subject, here are a few things a WordPress publisher can do:

- Create a daily or weekly newsletter, with top stories from selected categories;
- Provide an editorial-driven open forum in which editors propose themes, stories and questions and readers comment on them;
- Continue the discussion of articles on social platforms;
- Encourage users to submit articles and images for contests;
- Highlight your authors.
Listing Authors, and Why It’s a Good Thing

If you’re a publisher, your authors are your biggest asset. They are the content creators. Their writing gets consumed by millions of people all over the world.

Showcasing them exposes them for what they really are: authorities. Your authors will thank you for acknowledging them, and readers will get to see the human face behind the technology.

Coding The Perfect Author Listing

Here are the things we want to achieve with our page:

- Build it as a WordPress plugin so that we can reuse it more easily;
- Display the name, biography, number of posts and latest published post of all authors;
- Paginate the listing if we have many authors;
- Make the listing searchable.

Introducing WP_User_Query and get_users

The WP_User_Query26 class allows us to query the user database.

Besides returning an array of users, WP_User_Query returns general information about the query and, most importantly, the total number of users (for pagination).

One can use `WP_User_Query` by passing a series of arguments and listing the output.

```php
$my_authors = new WP_User_Query(
    array( 
        'blog_id' => $GLOBALS['blog_id'],
        'role' => '',
        'meta_key' => '',
        'meta_value' => '',
        'meta_compare' => '',
        'include' => array(),
        'exclude' => array(),
        'search' => '',
        'orderby' => 'login',
        'order' => 'ASC',
        'offset' => '',
        'number' => '',
        'count_total' => true,
        'fields' => 'all',
        'who' => ''
    )
);
```

We'll focus on only a few arguments, rather than go through all of them:

- **role**
  This is the user's role. In our example, we'll query for `author`.

- **offset**
  The first $n$ users to be skipped in the returned array.

- **number**
  Limit the total number of users returned.

We also have the `get_users` class, which (like `WP_User_Query`) returns a number of users based on the parameters set.

The important difference between the two is that `get_users` only returns an array of users and their meta data, whereas `WP_User_Query` returns extra information such as the total number of users (which is useful when it comes time to paginate).

---

27. http://codex.wordpress.org/Function_Reference/get_users
SIMPLE USER LISTING USING GET_USERS()

Before moving on with the full user listing, including pagination and search, let's see get_users in action.

If all you need is a simple list of authors, then you could just use wp_list_authors, like so:

```
wp_list_authors('show_fullname=1&optioncount=1&orderby=post_count&order=DESC&number=3');
```

Creating A Plugin And Shortcode With A Bit More Functionality

A simple and straightforward way to build our user listing would be to create a shortcode that we could include on any page we like. Housing this type of functionality in a plugin is ideal, so that we don’t have to worry about migrating it when we change the theme.

Let’s keep it simple. Our entire plugin will consist of just one file: simple-user-listing.php.

```php
<?php
/*
Plugin Name: Simple User Listing
Plugin URI: http://cozmoslabs.com
Description: Create a simple shortcode to list our WordPress users.
Author: Cristian Antohe
Version: 0.1
Author URI: http://cozmoslabs.com
*/

function sul_user_listing($atts, $content = null) {
    global $post;

    extract(shortcode_atts(array(
        "role" => '',
        "number" => '10'
    ), $atts));
```

28. http://codex.wordpress.org/Function_Reference/wp_list_authors
```php
$role = sanitize_text_field($role);
$number = sanitize_text_field($number);

// We're outputting a lot of HTML, and the
// easiest way to do it is with output
// buffering from PHP.
ob_start();

// Get the Search Term
$search = ( isset($_GET['as']) ) ?
sanitize_text_field($_GET['as']) : false;

// Get Query Var for pagination. This already
// exists in WordPress
$page = (get_query_var('paged')) ?
get_query_var('paged') : 1;

// Calculate the offset (i.e. how many users we
// should skip)
$offset = ($page - 1) * $number;

if ($search){
// Generate the query based on search field
    $my_users = new WP_User_Query(
        array(
            'role' => $role,
            'search' => '*'.
            $search . '**
        ));
}
else {
// Generate the query
    $my_users = new WP_User_Query(
        array(
            'role' => 'author',
            'offset' => $offset,
            'number' => $number
        ));
}

// Get the total number of authors. Based on
```
// this, offset and number per page we'll
// generate our pagination.
$total_authors = $my_users->total_users;

// Calculate the total number of pages for the
// pagination
$total_pages = intval($total_authors / $number) + 1;

// The authors object.
$authors = $my_users->get_results();

<?php

<div class="author-search">
<h2>Search authors by name</h2>
<form method="get" id="sul-searchform" action="<?php the_permalink() ?>">
  <label for="as" class="assistive-text">Search</label>
  <input type="text" class="field" name="as" id="sul-s" placeholder="Search Authors" />
  <input type="submit" class="submit" name="submit" id="sul-searchsubmit" value="Search Authors" />
</form>

<?php

if($search){ ?
  <h2>Search Results for: <em><?php echo $search; ?></em></h2>
  <a href="<?php the_permalink(); ?>">Back To Author Listing</a>
<?php } ?

</div></div-- .author-search -->

<?php if (!empty($authors)) { ?
  <ul class="author-list">
<?php

// loop through each author
foreach($authors as $author){
    $author_info = get_userdata($author->ID);
    ?>
    <li>
        <!-- php echo get_avatar($author->ID, 90 ); ? -->
    <h2><a href="<?php echo get_author_posts_url($author->ID); ?>"><?php echo $author_info->display_name; ?></a> - <?php echo count_user_posts($author->ID); ?> posts</h2>
    <p><?php echo $author_info->description; ?></p>
    <?php $latest_post = new WP_Query("author=$author->ID&post_count=1"); if (!empty($latest_post->post)){
        <p><strong>Latest Article:</strong><a href="<?php echo get_permalink($latest_post->post->ID) ?>"><?php echo get_the_title($latest_post->post->ID); ?></a></p>
        <!-- php } endif ? -->
    </p><a href="<?php echo get_author_posts_url($author->ID); ?>" >Read <?php echo $author_info->display_name; ?> posts</a></p>    </li>
    <?php }
</ul> <!-- .author-list -->
<?php } else { ?>
    <h2>No authors found</h2>
    <? } //endif ?

    <nav id="nav-single" style="clear:both; float:none; margin-top:20px;">
        <h3 class="assistive-text">Post navigation</h3>
Breaking Down The Code

The top of our plugin's main PHP file must contain the standard header of information. This header tells WordPress that our plugin exists, and it adds it to the plugin management screen so that it can be activated, loaded and run.
CREATING A SHORTCODE

Adding a new shortcode in WordPress is rather easy. We find the function that returns the desired output (in our case, `sul_user_listing`), and then we add it using the `add_shortcode` WordPress function.

```php
function sul_user_listing($atts, $content = null) {
    // return our output
}

add_shortcode('userlisting', 'sul_user_listing');
```

EXTRACTING OUR SHORTCODE ARGUMENTS

We want to be able to list users based on their roles and to control how many users are displayed on the page. We do this through shortcode arguments. We'll add the shortcode to our theme in this way: `[userlisting role="author" number="15"]`. This will allow us to reuse the plugin to list our subscribers as well.

To do this, we need to use shortcode arguments:

```php
extract(shortcode_atts(array(
    "role" => '',
    "number" => '10'
), $atts));
```

The `extract` function imports variables into our function from an array. The WordPress function `shortcode_atts` basically returns an array with our arguments; and we’ll set up some defaults in case none are found.

Note that the `role` default is an empty string, which would list all users regardless of their role.
SHORTCODES SHOULD NEVER ECHO STUFF OUT

The return value of a shortcode handler function gets inserted into the post content's output in place of the shortcode. You should use `return` and not `echo`; anything that is echoed will be outputted to the browser but will probably appear above everything else. You would also probably get “headers already sent” type of errors.

For simplicity, we’re buffering the output through `ob_start()`, so we put everything into an object and return it once we’re done.

SETTING UP OUR VARIABLES

Now we can start building our listing of authors. First, we need to set up a few variables:

- **$search**
  This takes the GET parameter as if it exists.

- **$page**
  The `get_query_var` for the pagination. This already exists in WordPress.

- **$offset**
  Calculate the offset (i.e. how many users to skip when paginating).

- **$total_authors**
  Get the total number of authors.

- **$total_pages**
  Calculate the total number of pages for the pagination.

THE QUERY

We actually have two queries: the default listing and the search results.

```php
if ($search) {
    // Generate the query based on search field
    $my_users = new WP_User_Query(
        array(
            'role' => $role,
            'search' => '*' . $search . '*'
        )
    );
} else {
```
// Generate the query

$mmy_users = new WP_User_Query(

    array(
        'role' => 'author',
        'offset' => $offset ,
        'number' => $number
    ));

});

WP_USER_QUERY->TOTAL_USERS & WP_USER_QUERY->GET_RESULTS

WP_User_Query provides us with two useful functions, among others:

- total_users
  Returns the total number of authors. This, the offset and the number of users per page will generate our pagination.

- get_results
  Returns an object with the authors alone. This is similar to what get_users() returns.

THE SEARCH FORM

For the search, we're using a simple form. There's nothing complex here.

```php
<?php

<div class="author-search">
<h2>Search authors by name</h2>
<form method="get" id="sul-searchform"
action="<?php the_permalink() ?>">
    <label for="as"
class="assistive-text">Search</label>
    <input type="text" class="field"
name="as" id="s" placeholder="Search Authors" />
    <input type="submit" class="submit"
name="submit" id="searchsubmit" value="Search Authors" />
</form>

<?php
```
if($search){ ?>
   <h2>Search Results for: <em><? echo $search; ?></em></h2>
?}</em></h2>
   <a href="<? echo the_permalink(); ?>">Back To Author Listing</a>
<?php } ?>

</div><!-- .author-search -->

USER DATA AND LISTING THE AUTHORS

Looping through our results is fairly simple. However, getting information about users is a bit confusing in WordPress. You see, there are a lot of ways to get user data. We could get it directly from the returned query; we could use general functions such as get_userdata, get_user_meta, the_author_meta and get_the_author_meta; or we could even use dedicated functions such as the_author_link and the_author_posts.

We'll just use get_userdata plus two other functions: get_author_posts_url and get_avatar.

<?php if (!empty($authors)) { ?>
   <ul class="author-list">
<?php
   // loop through each author
   foreach($authors as $author){
      $author_info = get_userdata($author->ID);
      ?>
      <li>
      <?php echo get_avatar($author->ID, 90 ); ?>
      <h2><a href="<? echo get_author_posts_url($author->ID); ?>"></a> - <<?php echo $author_info->display_name; ?>></a> - <?php echo count_user_posts($author->ID ); ?> posts</h2>
      <p><?php echo $author_info->description; ?></p>
      <?php $latest_post = new WP_Query( "author=$author->ID&post_count=1" );
      if (!empty($latest_post->post)){
      ?>
      <!-- more code here -->
      <?php }
   }?>
   </li>
<?php }?>
</ul>

</div><!-- .author-search -->

33
We need pagination because each listing will generate two extra queries. So, if we were listing 100 people, we would end up with 200 extra queries per page. That’s a bit much, so pagination is really needed. Otherwise, for websites with many authors, the load could get so heavy that it brings down the website.

PAGINATION

Front-End Author Listing and User Search for WordPress

34
Final Thoughts

We've discussed the code for an authors listing, but it has so many more uses:

- List your company's employees;
- Showcase users who have won a competition (by listing users with the role of “winners”);
- Present your company's departments, each with its respective team (based on user roles).

If you allow users to register on your website, you could use more or less the same code to generate any listing of users based on your needs.

If you require users to log in in order to comment (an effective way to stop automated spam), then listing users and their number of comments could increase engagement.

```php
<?php if ($page < $total_pages ) { ?>
  <span class="nav-next"><a rel="next" href="<?php the_permalink() ?>page/<?php echo $page + 1; ?>">Next</a></span>
<?php } ?>
</nav>
```
How To Integrate Facebook, Twitter And Google+ In WordPress

BY KEVIN MULDOON

Integrating social media services in your website design is vital if you want to make it easy for readers to share your content. While some users are happy with the social media buttons that come built into their design template, the majority of WordPress users install a plugin to automatically embed sharing links on their pages. Many of you will find that a plugin does exactly what you need; others not so much. Some are poorly coded, and most include services that you just don’t need. And while some great social media plugins are out there, they don’t integrate with every WordPress design.

If you aren’t comfortable editing your WordPress templates, a plugin is probably the best solution. If you are comfortable making a few edits to your theme, then consider manually integrating social media so that you have more control over what services appear on your website.

Today, we’ll show you how to manually integrate the three most popular social media services on your website: Twitter, Facebook and Google+. First, you’ll learn how to integrate Facebook comments on your WordPress website, to make it easier for readers to discuss your posts. Then, we’ll show you the most common ways to display your latest tweets in the sidebar, which should encourage more people to follow you on Twitter. Finally, we’ll show you how to add sharing buttons for all three social media services to your home page, posts and pages.
Please make sure to back up all of your template files before making any changes, so that you can revert back if something goes wrong. Testing your changes in a non-production area first would also be prudent.

**Integrate Facebook Comments On Your Website**

Because most people are signed into Facebook when they browse the Web, enabling Facebook comments on your website is a great way to encourage people to leave comments. It also curbs spam. While many solutions purport to reduce spam comments on WordPress, most are either ineffective or frustrate visitors by blocking legitimate comments.

Feature-rich commenting solutions such as IntenseDebate and Disqus have benefits, of course, because they allow users to comment using Facebook and a number of other services; but before visitors can comment, they have to grant access to the application, an additional step that discourages some from commenting. By comparison, integrating Facebook comments directly enables visitors to comment with no fuss. Also, this commenting system allows users to comment by signing into Facebook, Yahoo, AOL or Hotmail.

Before integrating Facebook on WordPress Mods at the end of September, I looked at a few solutions. I followed a great tutorial by Joseph Badow and tried a few plugins, such as Facebook Comments For WordPress. The reality, though, is that the official Facebook comment plugin is the quickest and easiest way to add Facebook comments to your website.

Simply follow the steps below to get up and running.

### 1. CREATE A FACEBOOK APPLICATION

To use Facebook comments on your website, create a new comment application for your website on the Facebook Application page. This step is required, whether you add Facebook comments manually using a third-party plugin or with the official Facebook plugin.

Simply click on the “+ Create New App” button on the Facebook Application page, and enter a unique name for your application in the “App Display Name” field. The “App Namespace” field doesn’t have to

---

33. http://wordpress.org/extend/plugins/facebook-comments-for-wordpress/
35. https://developers.facebook.com/apps
be filled in for Facebook comments (it’s used with the Facebook Open Graph Protocol\textsuperscript{36}).

![New App](image)

You will then be provided with an “App ID/API key” and an “App secret key.” You don’t need to remember these numbers because the official Facebook comments plugin\textsuperscript{37} automatically inserts them into the code that you need to add to your website.

![App Settings](image)

\textbf{2. ADD THE CODE TO YOUR WEBSITE}

Next, go back to the Facebook Comments plugin\textsuperscript{38} page and get the code for your website. The box allows you to change the URL on which comments will be placed, the number of comments to be shown, the width of the box and the color scheme (light or dark).

You don’t have to worry about what you enter in the box because all of the attributes can be modified manually. And it doesn’t matter what URL you enter because we will be replacing it later with the WordPress permalink:

- **href**
  The URL for this Comments plugin. News feed stories on Facebook will link to this URL.

- **width**
  The width of the plugin in pixels. The minimum recommended width is 400 pixels.

- **colorscheme**
  The color scheme for the plugin (either light or dark).

- **num_posts**
  The number of comments to show by default. The default is 10, and the minimum is 1.

- **mobile (beta)**
  Whether to show the mobile version. The default is `false`.

When you click on the “Get Code” button, a box will appear with your plugin code (choose the HTML5 option, because FBML is being deprecated). Make sure to select the application that you set up earlier for your comments so that the correct application ID is added to the code.
Insert the first piece of code directly after the `<body>` tag in your `header.php` template:

```
<div id="fb-root"></div>
<script>(function(d, s, id) {
var js, fjs = d.getElementsByTagName(s)[0];
if (d.getElementById(id)) return;
js = d.createElement(s); js.id = id;
js.src = "//connect.facebook.net/en_GB/all.js#xfbml=1&appId=YOURAPPLICATIONID";
fjs.parentNode.insertBefore(js, fjs);
})(document, 'script', 'facebook-jssdk');</script>
```

Put the second line of code where you want to show the comments. Make sure the static URL is replaced with the WordPress permalink (`<?php the_permalink(); ?>`) so that comments show correctly on every page of your website.

```
<div class="fb-comments" data-href="<?php the_permalink(); ?>" data-num-posts="15" data-width="500"></div>
```

To put Facebook comments above WordPress comments, add the above code just below the line that reads `<!-- You can start editing here. -->` in the `comments.php` template. To put Facebook comments below WordPress comments, add the above code below the `<form>` tag (again in the `comments.php` template).
If you plan to completely replace your WordPress comments with Facebook comments, simply replace the call to your `comments.php` template with the call to your Facebook comments. For example, to replace comments in posts, simply add the code to the `single.php` template. Similarly, edit the `page.php` template to show Facebook comments on pages.

Your should now see the Facebook comments box displayed on your website. To get an update whenever someone leaves a comment using Facebook, add yourself as a moderator to your application on the Comment Moderation tool page.

**Show Your Latest Tweets In The Sidebar**

Displaying your latest tweets is a good way to encourage people to follow you on Twitter. The most common place to display tweets is in the sidebar, although you can add them to any area of the website.

**DISPLAY YOUR LATEST TWEETS MANUALLY**

I have tried a few manual solutions for showing tweets on my websites, and my favorite comes from Chris Coyier of CSS-Tricks. His RSS fetching snippet is a quick and effective way to show the latest tweets

from your account. The RSS address of your Twitter account is
http://api.twitter.com/1/statuses/user_timeline.rss?screen_name=xxxxx (where xxxxx is your Twitter
user name). For the tweets that you favorite, use
http://twitter.com/favorites/xxxxx.rss. For example, the RSS for
the latest tweets from Smashing Magazine is
http://api.twitter.com/1/statuses/user_timeline.rss?screen_name=smashingmag; and to display only
the favorites, https://twitter.com/favorites/smashingmag.rss.
Once you've got your Twitter RSS address, simply add it to Chris' PHP
snippet.

<?php
include_once(ABSPATH . WPINC . '/feed.php');
$rss = fetch_feed('https://api.twitter.com/1/statuses/user_timeline.rss?screen_name=smashingmag');
$maxitems = $rss->get_item_quantity(3);
$rss_items = $rss->get_items(0, $maxitems);
?>

<ul>
<?php if ($maxitems == 0) echo '<li>No items.</li>'; else

// Loop through each feed item and display each item as
// a hyperlink.
foreach ($rss_items as $item) :
<li>
<a href='<?php echo $item->get_permalink(); ?>'>
<?php echo $item->get_title(); ?></a>
</li>
</foreach; ?>
</ul>

For a more stylish way to display tweets manually, check out Martin
Angelov’s tutorial “Display Your Favorite Tweets Using PHP and
jQuery,” or Sea of Cloud’s “Javascript Plugin Solution.”

DISPLAY YOUR LATEST TWEETS USING THE OFFICIAL TWITTER WIDGET

The official Twitter profile widget looks great and is easy to customize. You can define the number of tweets to display and whether the box should expand to show all tweets or provide a scroll bar.

The dimensions can be adjusted manually, or you can use an auto-width option. The color scheme can easily be changed in the settings area, too. Once the widget is the way you want it, simply grab the code and add it to the appropriate WordPress template.

DISPLAY YOUR LATEST TWEETS USING A WORDPRESS PLUGIN

If you don’t want to code things manually or use the official Twitter profile widget, you could try one of the many plugins available:

- Cardoza Twitter Box
- Floating Tweets
- Latest Twitter Sidebar Widget
- My Twitter Ticker
- Tweet Blender
- Twitter Plugin for WordPress

---

44. http://tweet.seaofclouds.com/
45. https://twitter.com/about/resources/widgets/widget_profile
47. http://wordpress.org/extend/plugins/floating-tweets/
Add Social-Media Sharing Buttons To Your WordPress Website

Adding social-media sharing and voting buttons is very straightforward and enables readers to share your content on the Web. Simply get the code directly from the following pages:

- [Facebook](http://developers.facebook.com/docs/reference/plugins/like/)
- [Google+](http://www.google.com/webmasters/+1/button/)
- [Twitter](https://twitter.com/about/resources/buttons)

The buttons you get from the above links work well when added directly to posts (single.php) and pages (page.php). But they don’t work correctly on the home page (index.php) or the archive (archive.php) by default, because we want to show the number of likes, pluses and retweets for each individual article, rather than the page that lists the article. That is, if you simply add the default code to index.php, every button will show the number of shares for your home page, not for each article.

To resolve this, simply make sure that each button uses the article permalink, rather than the URL of the page it is on. To add sharing buttons only to posts, simply choose the button you want from the links above and copy the code to single.php; to add the buttons only to pages, just add the code to page.php.

To show the number of likes, pluses and retweets that an article has on the home page and in the archives, follow the steps noted below for Facebook, Google+ and Twitter below (the code for showing a sharing button on the index page will work for posts and pages, too). You can see an example of sharing buttons integrated in post excerpts on my own website [WordPress Mods](http://www.wpmods.com) and on popular blogs such as [Mashable](http://www.mashable.com).

---

54. http://www.google.com/webmasters/+1/button/
55. https://twitter.com/about/resources/buttons
Facebook’s Like button comes with a lot of options. Choose from three layouts: standard, button count and box count. An email button (labelled “Send”) can be added, and you can set the width of the box, too. You can also show profile pictures below the button, choose between the labels “Like” and “Recommend,” choose between a light and dark color scheme, and set the font.

---

You need to add two pieces of code to your website. First, add the JavaScript SDK code directly after the <body> tag (in the header.php template). This code has to be added only once (i.e. if you've already added the code to show Facebook comments on your website, you don't need to add it again).

Put the second piece of code where you want to show the Like button. To ensure that the correct page is referenced, add $href="<?php echo get_permalink($post->ID); ?>" to the second piece of code. It should look something like this:

```html
<div class="fb-like" data-href="http://www.facebook.com/smashmag" $href="<?php echo get_permalink($post->ID); ?>"
  data-send="false" data-layout="box_count"
  data-width="450" data-show-faces="true"
  data-font="arial"></div>
```
More information on how to customize the Like button can be found on the Facebook Like Button page\(^59\).

**GOOGLE+**

Google+\(^60\) offers four sizes of sharing buttons: small, medium, standard and tall. The number of votes that a page has received can be shown inline, shown in a bubble or removed altogether.

![Customize your +1 button and +Snippet](image)

Linking to your article's permalink is very easy. Just append `hre-f="<?php the_permalink(); ?>"` to the `g:plusone` tag. For example, to show a tall inline Google+ button, you would use the following code:

```
<!-- Place this tag where you want the +1 button to render -->
<g:plusone size="tall" annotation="inline" href="<?php the_permalink(); ?>"></g:plusone>

<!-- Place this render call where appropriate -->
<script type="text/javascript">
(function() {
    var po = document.createElement('script'); po.type = 'text/javascript'; po.async = true;
    po.src = 'https://apis.google.com/js/plusone.js';
    var s = document.getElementsByTagName('script')[0];
```

\(^59\) http://developers.facebook.com/docs/reference/plugins/like/

\(^60\) http://www.google.com/webmasters/+1/button/
For more tips on customizing the Google+ button, please view the official Google+ button documentation page\(^61\).

**TWITTER**

Twitter\(^62\) offers four types of buttons: one for sharing links, one for inviting people to follow you, a hash tag button for tweeting stories, and another for mentions (used for contacting others via Twitter). The button you need to show the number of shares that an article has gotten is called “Share a link.”

On the button customization page, you can choose whether to show the number of retweets and can append “Via,” “Recommend” and “Hashtag” mentions to the shared link.

To make sure Twitter uses the title of your article and the correct URL, simply add `data-text="<?php the_title(); ?>"` and `data-`.

---

\(^{61}\) https://developers.google.com+/plugins/+1button/

\(^{62}\) https://twitter.com/about/resources/buttons
url="<?php the_permalink(); ?>" to your link. For example, if you were using the small button, you would use:

```html
<a href="https://twitter.com/share"
class="twitter-share-button" data-via="smashingmag"
<script>function(d,s,id){var js,fjs=d.getElementsByTagName(s)[0];if(!d.getElementById(id)){js=d.createElement(s);js.id=id;js.src="//platform.twitter.com/widgets.js";fjs.parentNode.insertBefore(js,fjs);}(document,"script", "twitter-wjs");</script>
```

To show the larger button instead, simply append `data-size="large"` to the link. To show the popular vertical button (shown below) instead of the default horizontal button, append `data-count="vertical"` to the link.

For more tips on customizing the Twitter button, please view the official Twitter button documentation page.

**Summary**

Many WordPress users continue to use plugins to integrate social-media sharing buttons and activity on their websites. As we've seen, though, integrating social-media services manually is straightforward and, for many users, a better solution than simply installing a plugin and making do with whatever features it offers.

Integrating Facebook comments on your website takes only a few minutes and is much less complicated than any of the available plugins. While good tutorials are available that show you how to manually add Twitter to your website, the official widget from Twitter is the best all-around solution for most websites.

---

63. https://dev.twitter.com/docs/tweet-button
64. https://twitter.com/about/resources/widgets/widget_profile
Some fantastic plugins exist for WordPress to automatically insert social-media voting buttons in your design. Installing and setting them up takes only a few minutes, although manually adding the buttons enables you to give them maximum visibility.

Remember, play it safe and make any changes in a test area first before applying the changes to the live website. I also recommend backing up all of your template files before changing anything (and your database if required). A few minutes of preparation could save you hours of troubleshooting, so try not to skip this step. »
Manage Events Like A Pro With WordPress

BY DANIEL PATAKI

If you’ve ever tried working with, coding for or just thinking about anything to do with events, you know they are a total nightmare in every possible way. Repeating events, schedules, multiple days, multiple tracks, multiple prices, multiple speakers, multiple organizations, multiple payment options — the list goes on on for quite some time.

Today we’ll show you how to make event management an easy — nay, enjoyable — task by making WordPress do the grunt work for you. We’ll be looking at out-of-the-box WordPress features, plugins and themes and a DIY approach to managing events.

In A Nutshell

I know some people don’t like to read lengthy reviews, so here are my recommendations in the shortest possible form. We’ll look at all of these recommendations in depth, so read on if you want to know more about them.

If money is not an issue or you just want the best possible combination of products, I recommend using Event Espresso to manage the events and Eventure from ThemeForest to display them. This will set you back at least $125 (more if you need add-ons for Event Espresso),
but it will give you one of the most powerful event-management setups you can get without touching any code.

If you don’t need a payment gateway, multiple-day event-specific options or other advanced features or you’re on a budget, you could use Events Manager Free Version 66, Event Espresso Lite 67 or Events Made Easy 68. These are all free and easy to use, providing roughly the same functionality. You might also want to purchase a theme to display your events nicely, which will set you back at least $35, but this is in no way required.

**Event Management Features**

Before we get to specific tools, let’s look at some of the features we get from an event-management system. You might not need all of these, but looking at them is useful when planning your system.

**EVENTS**

Obviously, our event-management plugin should at least support events. The ability to create events that are separate from your regular posts is a powerful feature, allowing you to add events to your website's existing content.

**EVENT TAXONOMIES**

Regular posts can be ordered into taxonomies — categories and tags. Having separate taxonomies for events (i.e. event categories and event tags) is useful for separating them from your regular content. If you organize Web development conferences, you might want to differentiate between design- and coding-related ones, or you might want to single out JavaScript- and Ruby-related ones. Your regular content might have nothing to do with Web development, so having separate taxonomies would come in handy.

**REGISTRATIONS**

Allowing people to register for events right there on your website can greatly boost attendance. The path a user has to take from discovering

---

67. http://wordpress.org/extend/plugins/event-espresso-free/
68. http://wordpress.org/extend/plugins/events-made-easy/
your event to participating becomes that much shorter, which translates into a better user experience and more registrants.

You will also need to be able to manage registrations through the back end. Registrants should be listed somewhere, with easy access to their details.

**PAYMENT GATEWAYS**

The ability to accept payments online breaks down another barrier between your events and potential attendees. A feature that allows you to accept the widest variety of payment methods would be ideal.

**SPEAKER MANAGEMENT**

As a software programmer, I don’t like when I enter data somewhere and it’s not stored in an easily reusable way. The ability to manage speakers across your events is a big plus because it opens up access to powerful features later on. This feature should include the ability to add biographies and photos of speakers to the website.

**VENUE MANAGEMENT**

As with speaker management, having all your locations stored properly will make them easy to reuse in future. If you need to schedule another event at the same venue, there’s no need to reenter the details; just select it from a menu, and off you go.

**PARTICIPATING ORGANIZATIONS**

Another nice feature is being able to attach companies to events. Companies will often host events, and giving them some recognition for it is a nice thing to do in return.

**SPONSORS**

Almost all major events have sponsors that contribute in some way (usually with money). They often require you to add their logo in various places. Being able to add the names, descriptions and logos of sponsors for an event would be handy.

**NOTIFICATION MANAGEMENT**

There are two kinds of notifications we might want to control. On-site notifications are shown to users once they perform specific actions. When a user successfully pays for a ticket or encounters an error while
registering, an on-site notification should pop up to let them know what's going on. Being able to tailor the language of these to your style would be a nice feature.

The second type of notification are email messages to participants. Confirmations, reminders and so on would all be customized to your style.

FORM MANAGEMENT
Controlling the information to gather from registrants is key to finding sponsors and making the lives of users easier. Being able to control this on an event-by-event basis would be best. Some events require less information from users, others more.

COUPON MANAGEMENT
Many events offer coupons for promotional purposes. If you want to engage users beyond your website, then giving coupons for third parties to distribute is a great tactic. Creating multiple coupons for various events would enable you to manage a full-blown coupon campaign.

PRICE MANAGEMENT
Another way to persuade visitors to register is to offer different price options, such as early-bird pricing, student discounts, last-minute offers and so on.

MULTIPLE DAY EVENTS
Many events have so much going on that splitting them into multiple days is the only way to go. Being able to control this from the administration section would be a great plus, especially when coupled with price-management options (such as registration for one day only).

REPEATING EVENTS
If you are organizing a repeating event, you wouldn’t want to have to create it from scratch a hundred times a year. Scheduling and repeating tools would help minimize your effort.

POWERFUL GLOBAL AND MISCELLANEOUS SETTINGS
A great event-management system has to have great global and miscellaneous settings. Settings for creating an events listing page, changing
currencies, setting time zones and so on are all part of a complete sys-
tem.

**Complete Solutions**

All of the WordPress plugins in this section are paid plugins, but if
you’re running a serious operation, then the first two listed here are
well worth the money.

The three best plugins around are Events Planner\(^69\), Events Manag-
er\(^70\) and Event Espresso\(^71\). Event Espresso is by far the best of the lot,
but all three are versatile and under constant development.

**EVENT ESPRESSO**

Event Espresso\(^72\) is the cream of the crop. It has built-in support for al-
most all of the features mentioned above (except perhaps sponsor man-
agement) — and much more! It enables you to set up multiple forms of
payment, multiple event dates and times, multiple prices, discounts,
promotions (coupons), locations (even virtual ones) and emails. It also
creates posts for events automatically and does so much more!

Event Espresso also has a free “Lite” version, which gives you a taste
of the solution. The lite version is actually pretty robust and can be
used for simple situations. It includes event and attendee management,
automated emails, customizable registration and PayPal Standard Pay-
ment.

You can easily tailor the design of event listings to your current
theme. If you are willing to dish out the money for this plugin, I recom-
mand getting a premium website theme as well and modifying that as
needed.

Event Espresso is not cheap, but its feature set is top notch, so the
price is justified. The basic version costs $89.95, which contains all of
the features that 95% of people will need. From there, you can download

---

\(^69\) http://www.wpeventsplanner.com/
\(^70\) http://wp-events-plugin.com/
\(^71\) http://eventespresso.com/
\(^72\) http://eventespresso.com/
free and paid add-ons to the basic system. Some free add-ons are for payment gateways, social media and calendars.

MailChimp integration, recurring events management, developer customization options, WordPress integration, Groupon integration, multiple event registration and shopping cart integration (coming soon) is available at between $25 and $35 a pop. Most of these are well worth their money, although getting the WordPress members integration for free would have been nice, because that’s not a huge programming leap.
Events Planner\textsuperscript{73} is another well-rounded system. It doesn’t have all of the features of Event Espresso, but it does give you a lot to work with. Event categories, tags, instructors, locations, companies, notifications, payments, registrations and more can be managed with ease.

The main difference between Events Planner and Event Espresso is that the former’s UI is less polished, and some features found in both are not as well implemented in it. Despite this, Events Planner remains extremely flexible and robust. If you don’t want to part with almost a hundred bucks, you’ll be able to grab Events Planner for $39, plus another $24 if you need plugins that supports advanced date- and time-specific functions.

Events Planner does not have a lite version, but you can create a custom installation yourself and test drive\textsuperscript{74} the pro version. This is a little unusual for plugins, but it does mean you can fully test it before purchasing.

\textsuperscript{73} http://www.wpeventsplanner.com/
\textsuperscript{74} http://www.wpeventsplanner.com/test-drive-events-planner-pro/
EVENTS MANAGER

Events Manager is very similar to Events Planner in many ways. Some features have a better UI in Events Planner, while others are better in Events Manager. Were the price not so different, it would be a matter of preference, but because Events Manager costs a lot more than Events Planner, I would not recommend this solution.

Events Manager will set you back $75, and the price buys you only one year’s worth of upgrades. There are no plugins or add-ons here (which could be a good thing), but the higher price and losing access to updates after a year seems a bit cheeky at this price point.

Events Manager has a free version that gives you a lot of functionality. It supports event and booking management, recurring events, locations and more.

76. http://wordpress.org/extend/plugins/events-manager/
Of the three, Event Espresso is the clear winner. It supports every feature the other two do and a lot more. It also has handy (albeit slightly expensive) plugins, with more to come. Even at $89, if you run a suc-
cessful business (or plan to), it isn’t a high price to pay for the features you get.

If you can’t spend that much on a plugin, then Events Planner is a very capable alternative that will not leave you wanting. When all is said and done, it does cost less than half of Event Espresso and still has 80% of its features. I would still heartily recommend it.

If you don’t need payment options, however, and you need a free solution, the free version of this plugin might be your best option. Have a look at the partial solutions below.

**Partial Solutions**

Quite a few solutions do not offer advanced features such as payment gateways and coupon management but do allow some flexibility and customizations for events.

The best options for a simpler approach are All-in-One Event Calendar, Event Organiser and Events Made Easy, as well as the free versions of Event Espresso and Events Manager. In a showdown, it would be a close call between Event Espresso and Events Manager.

All-in-One Event Calendar[^77] creates a new post type for your events, allowing you to keep blog posts and events side by side. It supports event categories, tags and a few other options. Because it allows you to create a calendar page, it’s a great solution if you need something simple and workable in minutes.

Event Organiser[^78] has all of the same functions plus a lot more! It has permission settings, permalink settings, importing and exporting options and even venue support. In addition, it has an admin calendar view that gives you a useful overview of your events.

Events Made Easy[^79] has all of the features of All-In-One Event Calendar (except event tags), and it supports registrations and locations. If you absolutely need to support on-site registration, this would be the easiest to use. The UI is the least polished, though, so it won’t look as pretty in the administration section, but the features are solid.

**FINAL VERDICT**

Despite the great features offered by these plugins, I would stick with Event Espresso Lite or the free version of Events Manager. Apart from

[^77]: http://wordpress.org/extend/plugins/all-in-one-event-calendar/
[^78]: http://wordpress.org/extend/plugins/event-organiser/
[^79]: http://wordpress.org/extend/plugins/events-made-easy/
offering more functionality, they will also ease your transition if you need the full-blown system later on.

**Using WordPress Out Of The Box**

If you don’t need to manage data for each event, WordPress’s core functionality will do just fine. Here are a few ideas to get you started:

- Use posts to store events.
- If you want to be able to have regular posts as well, distinguish them using categories.
- Create top-level categories for distinguishing organizations, sponsors and venues.
- Use your website’s registration functionality to manage past attendees, or use it as a master attendee list if separate registration is not required for your events.
- Create pages for important information such as payment options.
- Use PayPal buttons in event posts to let people register and pay directly through PayPal.

Many of these features are far from optimal for event-intensive websites, but if you just need something simple that you can set up in 10 minutes, give it a go.

If you do choose this option, pay close attention to consistency. If your goal is expansion, you are guaranteed to want a better system later on, and consistency will ensure that you can make the switch without a hiccup.

**Comparing All Of The Options**

There is a lot to learn and a lot of options if you want to get started with event management. To make your life easier, here is a table with all of the features discussed, along with the solutions that support them. Click on the image to go to the large version (it’s a bit small to look at here).
Event-Friendly Themes

While the plugins do a nice job of helping you manage events, they are not designed to make your website pretty, which is equally important. No matter which route you take, you will need to do some work to make things fit perfectly, but some premium themes out there will shorten this process.

Eventure

$35 | Large screenshot | Live preview
Diarise

$70 | Large screenshot | Live preview

Events (from Elegant Themes)

Large screenshot | Live preview

86. http://demo.woothemes.com/?name=diarise
Conclusion

Whichever solution you choose, you will have to put in a few hours of work to make your website work well and look good. I usually advise using free software whenever possible, but this happens to be one of those areas where I would go with a complete solution. Getting it right from the get-go will save you a lot of headaches in the long run.

If you can afford to spend over $100 on managing events, go for Event Espresso, coupled with one of the premium themes mentioned above.

If you want to spend as little as possible, then try Events Manager Free Version, Event Espresso Lite or Events Made Easy. If you don’t plan on expanding a lot or you need multiple price points, go with Events Made Easy because it is completely free, with no paid version, so supporting the developer by using his product would be a nice gesture.

If you do plan on expanding, go with Event Espresso Lite because the pro version will have everything you need when you’re ready to buy it and you won’t have any migration or data problems.

---

95. http://wordpress.org/extend/plugins/event-espresso-lite/
Adopting A Responsive WordPress Theme Is More Than Install-And-Go

BY BEN GREMILLION

As iOS, Android, and Windows 8 take the Web to smaller screens, designers are adopting techniques to make their websites usable on handheld devices.

Responsive Web designs present different formatting and layout to suit the device on which their pages are displayed. Browsers choose the appropriate styles on page load, freeing website owners from having to maintain different sets of pages for different display scenarios.

The most common responsive method is to use CSS media queries to serve different style sheets (or parts of style sheets) based on the number of pixels available. Most often, this is applied to handheld devices such as smartphones, but it could be applied to 13-inch laptops, 30-inch TVs or Kindle-sized readers. Responsive designs respond to their environment.

No Shortage Of Quick Fixes

The term “responsive design” is only two years old, but website owners can choose today from many mobile and widescreen themes for popular content management systems. Third-party developers have created paid and free themes that adapt based on browser width for WordPress,
Drupal, Joomla and ExpressionEngine. At the time of this writing there are:

- 62 responsive themes\textsuperscript{97} on Drupal.org,
- 40 free or upgradable themes\textsuperscript{98} on WordPress.org,
- 587 responsive themes for all platforms\textsuperscript{99} at ThemeForest,
- 300 responsive WordPress themes\textsuperscript{100} at ThemeForest.

Designers handy with CSS can also find a few\textsuperscript{101} do-it-yourself\textsuperscript{102} frameworks\textsuperscript{103}. But responsive themes are as varied as the problems they are meant to solve. Not all are created with the same technique, features or attention to detail. Aesthetics aside, how should someone choose a theme?

\section*{How Do Responsive Themes Perform?}

Large themes cause delays for both the Web server and the end user. Servers require more time to fetch each additional file, and the milliseconds add up. For users, though, sluggishness comes from the number of kilobytes total, not just the number of files.

Aside from using media queries, many themes use variations of techniques to respond to browsers. I’ve tested 40 responsive themes on WordPress.com\textsuperscript{104}, comparing them to the stock Twenty Eleven\textsuperscript{105} and Twenty Ten\textsuperscript{106} themes.

\begin{itemize}
\item \textsuperscript{97} http://smashed.by/drupal
\item \textsuperscript{98} http://wordpress.org/extend/themes/search.php?q=responsive
\item \textsuperscript{99} http://ThemeForest.net/search?utf8=✓&term=responsive
\item \textsuperscript{100} http://ThemeForest.net/search?term=responsive&category=wordpress
\item \textsuperscript{101} http://responsive.gs/
\item \textsuperscript{102} http://goldengridsystem.com/
\item \textsuperscript{103} http://www.columnal.com/
\item \textsuperscript{104} http://www.designinfluences.com/fluid960gs/
\item \textsuperscript{105} http://coding.smashingmagazine.com/2012/03/19/gridpak-the-responsive-grid-generator/
\item \textsuperscript{106} http://wordpress.org/extend/themes/
\item \textsuperscript{107} http://wordpress.org/extend/themes/twentyeleven
\item \textsuperscript{108} http://wordpress.org/extend/themes/twentyten
\end{itemize}
The chart above shows that:

• The number of files that a theme loads and the theme’s weight in kilobytes have no direct relationship;

• With few exceptions, most themes make 25 requests or fewer;

• WordPress’ stock themes perform very well, but a few other themes provide responsive capability and better performance.

Bear in mind that these are empty themes, measured before any content or modifications have increased their load. Because data costs money for people who are accessing the Web through cellular networks, themes that require fewer downloads per page load are more likely to earn repeat visits. Of the themes sampled:

• Only one theme did not use CSS media queries. This theme’s unusual method was to detect page width with jQuery and then change the body class, which in turn would change the layout with animated transitions. The extra time taken to load and implement JavaScript compromised the goal of responsiveness.

• More than half had three break points: mobile (480 pixels or less), medium (481 to 1024 pixels) and wide (1025 pixels or more). The medium-sized layouts were most often measured with percentages, ems or min- and max-width, rather than strictly by number of pixels.

• Left-to-right layouts on wide screens always became top-to-bottom layouts on mobile. That is, the left-most column in a widescreen layout...
would always appear at the top of the page in a mobile layout, regardless of its width or the type of content. Likewise, right-hand columns would become footers in mobile layouts. This means that the content in your left-most column should not discourage users from scrolling down when it’s formatted for mobile devices.

- All mobile designs had 10 to 20 pixels of horizontal margin. None deliberately allowed horizontal scrolling or used app-simulating frameworks such as jQuery Mobile.

- None provided in-page navigation.

- Two themes used **select** lists for navigation in their mobile layouts. None used multi-level navigation.

- Loading a page with three paragraphs of placeholder text, the themes averaged 306.57 KB per page load and 25.4 resources retrieved (including images, CSS files, JavaScript files and the like).

- The lightest theme weighed only 57.11 KB before the content itself (text and images) loaded. The largest weighed 1382.4 KB before the content loaded.

Remember that screen width does not necessarily equal browser width. Most themes are not built on the assumption that users will have their browser windows open as big as possible; rather, their layouts are designed for screen widths well under common sizes.
As seen in the graph above, most themes will use `max-width` media queries to resize layouts when browsers reach 1280, 800, 767 and 480 pixels wide. But most screens surveyed by Lifehacker\(^\text{109}\), StatCounter\(^\text{110}\) and W3 Counter\(^\text{111}\) start at 1280 pixels wide.

**Picking A Theme That Reflects Your Priorities**

Making a website responsive is more than about varying the number of columns on a page. The same critical questions emerge for all mobile-friendly websites, regardless of CMS.

- **Does the project merit a mobile layout?**
  Increasingly, the answer is yes. But the march towards mobile doesn’t mean that every website should follow suit. Pages that contain complicated tables, multi-month calendars, detailed images, complex navigation and other content unsuitable for small screens could negate any

\(^{109}\) http://lifehacker.com/5428806/whats-your-screen-resolution

\(^{110}\) http://gs.statcounter.com/#resolution-ww-monthly-201105-201205

\(^{111}\) http://www.w3counter.com/globalstats.php
benefit offered by responsive designs. “Can I?” and “should I?” are two different questions.

- **Would the website benefit from mobile-first thinking?**
  Designing a website to be mobile forces the content editors to answer hard questions. A screen measuring 320 pixels wide has no room for excess. This brings the design into focus, forcing you to eliminate distractions from whatever the website is meant to convey.

- **How many steps do we need?**
  RESPONSIVELY designed websites often rely on the width of the device on which they’re being viewed. But there’s more to it than asking “Mobile or not?” Responsive designs must address not only how a website handles on narrow screens, but also when wide becomes too wide. But a better option would be to consider using a device-agnostic approach to Web design focusing on content rather than device properties.

- **How do the layout and formatting change?**
  Deciding which elements on a given page users should see first, second and third will affect how the widescreen layout functions. For example, the font in headings in the widescreen layouts could be three, four or five times larger than the body copy, whereas giant headings are cumbersome on tiny screens.

- **What should a mobile page not show?**
  Multiple columns encourage a hierarchy of information: primary content that is unique to each page, and secondary content (often relegated to sidebars) that appears on more than one page. But mobile design makes multiple columns difficult to pull off. If the secondary content is unnecessary, how should it disappear? If it’s important, how does one design it without letting pages run longer than users are willing to scroll? (A good rule of thumb is that if an element does not support the page’s title, then it is not primary content.)

**Think Beyond The Theme**

Having a responsive theme does not guarantee a good mobile user experience.

Designing for mobile is not just about cutting material, but also about planning for limited attention. By nature of the medium, mobile users absorb information in limited chunks. Long pages can work if they’re divided into phone screen-sized sections. Unlike widescreen
users, mobile users are more inclined to scroll down “below the fold” (i.e. below what they first see when the page loads).

Consider higher-contrast colors for mobile — particularly, the contrast between the body text and the background — for improved legibility outdoors.

Extensive navigation bars with sublevels and sub-sublevels are impractical on mobile devices. Offering search functionality, creating pages dedicated to navigation, and flattening the website’s structure are common solutions; anything that reduces the number of taps between pages helps.

**When To Consider A Separate Mobile Website**

Responsive designs do more than make a website work well on a variety of screen sizes; they also force the owner to make their website easier, more focused and faster. But they’re a tool, not a requirement. Adaptive layouts and media queries aren’t always the best answer for mobile design problems. When big content simply doesn’t fit on small screens, maintaining a supplemental website would outweigh the benefit of having one website that serves many audiences. The key is to create a companion website that carries essential information organized for mobile use — and then find out what mobile users are missing.

Your website could warrant a separate mobile version if:

- You find yourself creating duplicate pages for mobile users on the same website;
- Short pages that look great on mobile phones don’t take advantage of large screens;
- You plan to phase out the widescreen layout in favor of a more streamlined user experience.

**Other Resources**

You may be interested in the following articles and related resources:

- *Mobile First*[^113], Luke Wroblewski
  This book by the former Yahoo designer discusses the reasons why thinking small from the start makes more sense than cramming later.

[^113]: http://www.abookapart.com/products/mobile-first
• “Supporting Multiple Screens”\textsuperscript{114}, Android Developers
  Google’s overview of screen sizes, pixel density, and how (and why) to
  achieve density dependence.

• “Platform Characteristics”\textsuperscript{115}, iOS Developer Library
  Apple’s guidelines on graphics and presentation for apps and Web con-
  tent for iOS.

• “Responsive Web Design”\textsuperscript{116}, Ethan Marcotte
  This article examines the benefits of, and the mental shift required for,
  Web designs that respond to the user’s browser.

• “Chrome Developer Tools: Network Panel”\textsuperscript{117}, Google Developers
  Shows how to activate and use the inspector in Chrome. It also works
  for most Webkit-based browsers.

• “Screen Resolution ≠ Browser Window”\textsuperscript{118}, Chris Coyier
  An explanation of how screen width is a deceptive metric, and what you
  can do about it.

• jQuery Mobile\textsuperscript{119}
  A framework that simulates app behaviors with JavaScript, CSS and
  HTML5.

• Less Framework\textsuperscript{120}
  A straightforward desktop-to-mobile CSS grid, with four specific steps.

• Whiteboard Framework for WordPress\textsuperscript{121}
  A barebones theme that serves as a starting point for design, rather
  than a finished product to be tweaked.

• Omega\textsuperscript{122}, Drupal theme
  “The Omega Drupal 7 base theme is a highly configurable HTML5/960
  grid base theme that is 100% configurable.”

\textsuperscript{114} http://developer.android.com/guide/practices/screens_support.html
\textsuperscript{115} http://developer.apple.com/library/ios/#documentation/userexperience/conceptual/mo-
  bilehig/Characteristics/Characteristics.html#//apple_ref/doc/uid/TP40006556-CH7-SW1
\textsuperscript{116} http://www.alistapart.com/articles/responsive-web-design/
\textsuperscript{117} https://developers.google.com/chrome-developer-tools/docs/network
\textsuperscript{118} http://css-tricks.com/screen-resolution-notequalto-browser-window/
\textsuperscript{119} http://jquerymobile.com/
\textsuperscript{120} http://lessframework.com/
\textsuperscript{121} http://whiteboardframework.com/
\textsuperscript{122} http://drupal.org/project/omega
Responsive Images With WordPress’ Featured Images

BY RACHEL MCCOLLIN

It’s been a couple of years now since the concept of responsive design took the Web design world by storm, and more and more websites are going responsive. But there are still some barriers and potential problems, not the least of these being the challenge of reducing the size of files that you’re sending to mobile devices.

In this chapter, we’ll look at how to use WordPress’ built-in featured images capability to deliver different-sized image files to different devices. “Featured images,” sometimes referred to as thumbnails, is a feature of WordPress that has been vastly improved since version 3. It enables you to specify a featured image for each post and display it with the post’s content. The fact that the image is stored in the post’s metadata, rather than embedded in the post’s content, means we can use it more flexibly, as we shall see.

Why Worry About Image Size?

OK, so a lot of people do use their mobile devices to surf the Web while sitting on the sofa, hooked up to their Wi-Fi, one eye on the phone and one on the TV. And many browse for information sparked by a conversation with the people around them. This type of website visitor is becoming more and more common. You might even be reading this chapter in this way right now.

But there are and will always be people who use the Web from a mobile device while out and about, possibly using 3G in an area with a dodgy signal or on an expensive data plan. We Web designers and developers tend to invest in data plans with all the bells and whistles; but, believe it or not, plenty of people out there don’t use the Internet as much as we do and so choose a limited plan.

These people won’t thank you for using up their data by sending huge image files to their devices. They may well abandon your website before looking at it if the image downloads are slowing everything down. Moreover, mobile visitors might only check your website quickly in the middle of doing something else, and a slow website could harm your search engine rankings. Taking all this into account, surely reducing file sizes for mobile devices is a no-brainer.
Responsive Images: What’s It All About?

By now, you probably know what responsive design is: it uses a combination of a fluid layout and media queries to define breakpoints at which a website’s layout or content changes to fit a particular screen size. Most responsive websites use media queries to target phones; some target tablets such as iPads as well.

In the early days of responsive design, making your images responsive meant using CSS to ensure they stayed nicely inside their containing element, with this code:

```css
img {
  max-width: 100%;
}
```

This technique makes the images look tidy, but all it really does is shrink the same large image to fit the layout. It does nothing to alter the actual size of the file, which could lead to huge image files sneaking onto your mobile design and seriously slowing down the website. Displaying large images that have been shrunk with CSS is discouraged by the W3C, and it uses processing power and data, so it’s a particularly bad idea on mobile devices.

When we do responsive design now, we generally include some way of making the image’s actual size smaller, using one of a variety of established techniques.

More Than One Way To Skin This Cat

If you’ve taken an interest in responsive design, you’ll have noticed that quite a few methods of delivering truly responsive images have emerged. Here are some of those methods:

- Replace images in the markup with background images, using images of different sizes depending on the device. This method has serious accessibility and SEO drawbacks because the images don’t have alt tags for screen readers to read or search engines to index.

- Use a smaller image in the markup and a larger image as the background of the containing element in the desktop version of the website, and then use CSS to hide the smaller image on larger screens. This method, described in detail by Harry Roberts on CSS Wizardry[^123^], is better than the first one, but it does require that you manually create

two images and then, in the style sheet, define the background image for every single image on the website. I don’t know about you, but I certainly don’t have the time or patience for that!

• Use a JavaScript-based solution that employs URL parameters or data attributes. This avoids the repetitive work above, but it involves more processing and can slow the website down – the opposite of what you intended.

• Use a solution such as Matt Wilcox’s Adaptive Images, which does the work for you. This is the smoothest option I’ve seen, but it requires that you separate the images that you want to be resized from those that you don’t – a potential problem when handing over a CMS-based website to a client or editor who isn’t technologically savvy.

The fact that Adaptive Images uses PHP got me thinking about how this could fit WordPress, which, after all, is written in PHP. Is there a way to use WordPress’ built-in functionality to deliver responsive images in a way that the client would not be able to break?

The answer is yes… with the addition of just one free plugin.

**WordPress Responsive Images: Making It Work**

I’ll demonstrate this technique using a website that I recently developed for a client, What’s a Mayor For?[^124]. This website is responsive and gets a significant portion of visits from mobile devices. At the moment, it uses `max-width` to resize images, but it doesn’t send different image files to mobile devices.

This is what the website looks like in desktop and mobile browsers:

[^124]: http://whatsamayorfor.org.uk
As you can see, the images scale to fit the layout. But what you can’t see is that the image’s actual size stays the same. We need to change that.

The solution we’ll follow here uses the following elements:

1. A free plugin called Mobble, which detects devices and provides conditional PHP functions that you can use to deliver different content to different devices in your theme’s files;

2. The `single.php` and `page.php` files, which we’ll edit to display the post or page’s featured image, but altering the image’s size according to the type of device;

3. The featured image functionality in WordPress’ dashboard, which we’ll use to define the image used for each post and page.

Let’s get started!
DOWNLOAD THE MOBBLE PLUGIN

First, download the Mobble plugin. This will check the browser’s user-agent string to determine which device the user is on. These checks are wrapped in WordPress-style conditional functions, such as `is_mobile()`, which is the one we’ll be using. Purists will balk at this approach, but in my experience it’s very reliable, and the plugin does get a high rating in WordPress’ repository.

Download and activate the plugin.

EDIT THE SINGLE.PHP AND PAGE.PHP FILES TO CALL THE POST’S THUMBNAIL

Using a text editor or WordPress’ editor, open the `single.php` file. Find the following code or something like it:

```html
<article id="post-<?php the_ID(); ?>" <?php post_class(); ?>>
<h1 class="entry-title"><?php the_title(); ?></h1>
<section class="entry-content">
```

In our example, the image needs to be displayed immediately after the heading and before the content, so our code needs to be inserted between the `h1` and `section` tags. If your website’s theme doesn’t use HTML5, you may put a `div` there instead of `section`.

Here is the code that displays the featured image for a given post:

```php
<?php the_post_thumbnail(); ?>
```

The function has some parameters that we can use, the most relevant being image size. You can use any of the sizes defined by WordPress:

- **thumbnail**
  Thumbnail: by default, a maximum of 150 × 150 pixels

- **medium**
  Medium resolution: by default a maximum of 300 × 300 pixels

- **large**
  Large resolution: by default, a maximum of 640 × 640 pixels

- **full**
  Full resolution: the original uploaded size

This is where our conditional function plays with the image’s size. Here is the full code we’ll need:

<?php
    if (is_mobile()) {
        the_post_thumbnail('medium');
    } else {
        the_post_thumbnail('large');
    }
?

This code does the following:

1. Checks whether the website is being viewed on a mobile device: if (is_mobile());

2. If so, outputs the medium resolution of the post’s thumbnail (or featured image): {the_post_thumbnail('medium')};

3. If not (i.e. else), outputs the large resolution: {the_post_thumbnail('large')};

Having set up the single.php file, let’s do the same for page.php.
Then, we need to change any embedded images to featured images via the WordPress dashboard.

**USE WORDPRESS’ FEATURED IMAGE FUNCTIONALITY TO DISPLAY THE IMAGES CORRECTLY**

Adding featured images in WordPress has been very simple since the functionality was incorporated in the user interface in version 3.0. Just follow these three steps:

1. In the WordPress dashboard, open the editing screen for each post and page.

2. Delete the existing image (it will remain in the gallery for that post or page, which will be helpful in a moment).

3. Click “Set featured image” in the bottom right of the screen.

4. In the “Set featured image” pop-up, click the “Gallery” tab. The image you just deleted will be displayed. All you need to do now is click “Use as featured image,” and then click the little cross in the top right of the window. Don’t insert the image into the post or else the image will be displayed twice.
5. Finally, click the “Update” button to save the changes that you’ve made to the post, and test it.

**Summary**

As you can see, using WordPress' featured images functionality to make responsive websites faster on mobile devices is a fairly simple process. All it takes is three steps:

1. Install the Mobble plugin.
2. Add conditional tags to the `single.php` and `page.php` files to call different versions of the image depending on the device.
3. Replace images in the body of the content with the featured images.

Of course, this method isn’t perfect for all situations. It only works if the image should appear above (or below) the rest of the content, and it does require that anyone who adds a post or page use the featured image functionality instead of just inserting an image in the body of the content. All you need to do now is educate the editors of your website to use featured images instead of images within the content. But how you do that is for another day!

---

About The Authors

**Ben Gremillion**

Ben Gremillion started playing with pixels in the mid-1980s and building websites circa 1997. When not hiking or stargazing, he crafts HTML and CSS, spends a lot of time debugging PHP and MySQL, ponders the details of user experience, and injects bits of personality into staid websites. He blogs about his lessons learned in Web design at ben-thinkin.net\(^{127}\). You can also find him on Twitter\(^{128}\).

**Cristian Antohe**

Cristian Antohe is part of the team that built Profile Builder Pro\(^{129}\), a front end registration, login, edit profile and user listing plugin for WordPress. He’s a WordPress developer, web designer and Open Source fanatic. A couple of years ago he co-founded Reflection Media\(^{130}\), a "custom WordPress solutions" company and every Thursday curates wpMail.me\(^{131}\), the free weekly WordPress newsletter. You an follow the author on Twitter\(^{132}\).

**Daniel Pataki**

Daniel Pataki is a guitar wielding Web developer obsessed with Web technology, best practices and the awesomeness of WordPress. His website: http://danielpataki.com\(^{133}\). You can also follow the author on Twitter\(^{134}\).

**Ellen Bauer**

Ellen Bauer is a WordPress theme developer\(^{135}\) and blogger running the German Web design and WordPress blog Elmastudio\(^{136}\) together with

---

127. http://benthinkin.net/
128. https://twitter.com/benthinkin
131. http://wpmail.me/
132. https://twitter.com/sareiodata
133. http://danielpataki.com/
134. https://twitter.com/danielpataki
Manuel Esposito. Next to Ellens passion for beautiful WordPress themes and blogging she loves traveling, veggie food and is always up for new adventures. Follow her on Twitter: @allenbauer.

Kevin Muldoon  
Kevin Muldoon is a Scottish webmaster and blogger who currently lives in South America. His labour of love is WordPress Mods; a blog which focuses on WordPress Themes, Plugins, Tutorials, News, Modifications. Follow the author on Twitter.

Milan Petrovic  

Rachel McCollin  

137. https://twitter.com/ellenbauer  
140. https://twitter.com/KevinMuldoon  
144. https://twitter.com/milangd  
145. http://compass-design.co.uk/  
146. https://twitter.com/@rachelmccollin
About Smashing Magazine

Smashing Magazine is an online magazine dedicated to Web designers and developers worldwide. Its rigorous quality control and thorough editorial work has gathered a devoted community exceeding half a million subscribers, followers and fans. Each and every published article is carefully prepared, edited, reviewed and curated according to the high quality standards set in Smashing Magazine’s own publishing policy. Smashing Magazine publishes articles on a daily basis with topics ranging from business, visual design, typography, front-end as well as back-end development, all the way to usability and user experience design. The magazine is — and always has been — a professional and independent online publication neither controlled nor influenced by any third parties, delivering content in the best interest of its readers. These guidelines are continually revised and updated to assure that the quality of the published content is never compromised.

About Smashing Media GmbH

Smashing Media GmbH is one of the world’s leading online publishing companies in the field of Web design. Founded in 2009 by Sven Lennartz and Vitaly Friedman, the company’s headquarters is situated in southern Germany, in the sunny city of Freiburg im Breisgau. Smashing Media’s lead publication, Smashing Magazine, has gained worldwide attention since its emergence back in 2006, and is supported by the vast, global Smashing community and readership. Smashing Magazine had proven to be a trustworthy online source containing high quality articles on progressive design and coding techniques as well as recent developments in the Web design industry.

147. http://www.smashingmagazine.com