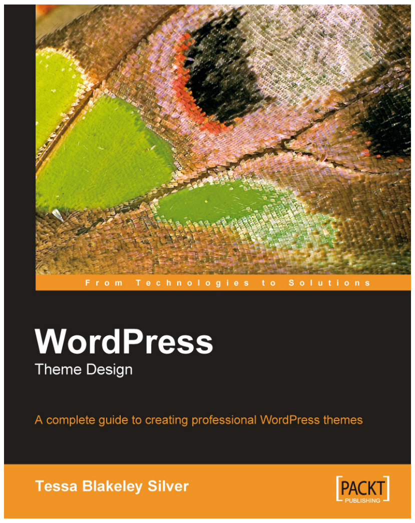
WordPress Theme Design

Tessa Blakeley Silver


From Technologies to Solutions
WordPress
Theme Design
A complete guide to creating professional WordPress themes
Tessa Blakeley Silver
PACKT PUBLISHING

Chapter No.8

"AJAX / Dynamic Content and Interactive Forms"

In this package, you will find:

A Biography of the author of the book

A preview chapter from the book, Chapter NO.8 "AJAX / Dynamic Content and Interactive Forms"

A synopsis of the book's content

Information on where to buy this book

About the Author

Tessa Blakeley Silver's background is in print design and traditional illustration. She evolved over the years into web and multi-media development, where she focuses on usability and interface design. Prior to starting her consulting and development company, hyper3media (pronounced hyper-cube media) (http://hyper3media.com), Tessa was the VP of Interactive Technologies at eHigherEducation, an online learning and technology company developing compelling multimedia simulations, interactions, and games that met online educational requirements like 508, AICC, and SCORM. She has also worked as a consultant and freelancer for J. Walter Thompson and The Diamond Trading Company (formerly known as DeBeers), and was a Design Specialist and Senior Associate for PricewaterhouseCoopers' East Region Marketing department. Tessa authors several design and web technology blogs. WordPress Theme Design is her second book for Packt Publishing.

WordPress Theme Design

The goal of this title is to explain the basic steps of creating a WordPress theme. This book focuses on the development, creation, and enhancement of WordPress themes, and therefore does not cover general 'how to' information about WordPress and all its many features and capabilities. This title assumes you have some level of understanding and experience with the basics of the WordPress publishing platform. The WordPress publishing platform has excellent online documentation, which can be found at http://codex.wordpress.org. This title does not try to replace or duplicate that documentation, but is intended as a companion to it.

My hope is to save you some time finding relevant information on how to create and modify themes in the extensive WordPress codex, help you understand how WordPress themes work, and show you how to design and build rich, in-depth WordPress themes yourself. Throughout the book, wherever applicable, I'll point you to the relevant WordPress codex documentation along with many other useful online articles and sites.

I've attempted to create a realistic WordPress theme example that anyone can take the basic concepts from and apply to a standard blog, while at the same time, show how flexible WordPress and its theme capabilities are. I hope this book's theme example shows that WordPress can be used to create unique websites that one wouldn't think of as 'just another blog'.

What This Book Covers

Chapter 1 Getting Started as a WordPress Theme Designer introduces you to the WordPress blog system and lets you know what you'll need to be aware of regarding the WordPress theme project you're ready to embark on. The chapter also covers the development tools that are recommended and web skills that you'll need to begin developing a WordPress theme.

Chapter 2 Template Design and Approach takes a look at the essential elements you need to consider when planning your WordPress theme design. It discusses the best tools and processes for making your theme design a reality. I explain my own 'Rapid Design Comping' technique and give you some tips and tricks for developing color schemes and graphic styles for your WordPress theme. By the end of the chapter, you'll have a working XHTML and CSS based 'comp' or mockup of your theme design, ready to be coded up and assembled into a fully functional WordPress theme.

Chapter 3 **Coding It Up uses the fi**nal XHTML and CSS mockup from Chapter 2 and shows you how to add WordPress PHP template tag code to it and break it down into the template pages a theme requires. Along the way, this chapter covers the essentials of what makes a WordPress theme work. At the end of the chapter, you'll have a **basic, working WordPress theme.**

Chapter 4 Debugging and Validation discusses the basic techniques of debugging and validation that you should be employing throughout your theme's development. It covers the W3C's XHTML and CSS validation services and how to use the FireFox browser and some of its extensions as a development tool, not just another browser. This chapter also covers troubleshooting some of the most common reasons 'good code goes bad', especially in IE, and best practices for fixing those problems, giving you a great-looking theme across all browsers and platforms.

Chapter 5 Your Theme in Action discuss how to properly set up your WordPress theme's CSS style sheet so that it loads into WordPress installations correctly. It also discuss compressing your theme files into the ZIP file format and running some test installations of your theme package in WordPress's administration panel so you can share your WordPress theme with the world.

Chapter 6 WordPress Reference covers key information under easy-to-look-up headers that will help you with your WordPress theme development, from the two CSS class styles that WordPress itself outputs, to WordPress's PHP template tag code, to a breakdown of "The Loop" along with WordPress functions and features you can take advantage of in your theme development. Information in this chapter is listed along with key links to bookmark to make your theme development as easy as possible.

Chapter 7 Dynamic Menus and Interactive Elements dives into taking your working, debugged, validated, and properly packaged WordPress theme from the earlier chapters, and start enhancing it with dynamic menus using the SuckerFish CSS-based method and Adobe Flash media

Chapter 8 AJAX/Dynamic Content and Interactive Forms continues showing you how to enhance your WordPress theme by taking a look at the most popular methods for leveraging AJAX techniques in WordPress using plugins and widgets. I'll also give you a complete background on AJAX and when it's best to use those techniques or skip them. The chapter also reviews some cool JavaScript toolkits, libraries, and scripts you can use to simply make your WordPress theme appear 'Ajaxy'.

Chapter 9 Design Tips for Working with WordPress reviews the main tips from the previous chapters and covere some key tips for easily implementing today's coolest CSS tricks into your theme as well as a few final SEO tips that you'll probably run into once you really start putting content into your WordPress site.

8
AJAX / Dynamic Content and Interactive Forms

AJAX—it's the buzzword that hit the Web with a bullet in 2005, thanks to Jesse James Garrett, a user-experience expert who founded AdaptivePath.com. If you're totally new to AJAX, I'll just point out that; at its core, AJAX is nothing that scary or horrendous. AJAX isn't even a new technology or language!

Essentially, AJAX is an acronym for Asynchronous JavaScript and XML, and it is the technique of using JavaScript and XML to send and receive data between a web browser and a web server. The biggest advantage this technique has is that you can dynamically update a *piece* of content on your web page or web form with data from the server (preferably formatted in XML), without forcing the entire page to reload. The implementation of this technique has made it obvious to many web developers that they can start making advanced web applications (sometimes called RIAs—Rich Interface Applications) that work and *feel* more like software applications, instead of like web pages.

Keep in mind that the word AJAX is starting to have its own meaning (as you'll also note its occasional use here as well as all over the web as a proper noun, rather than an all-cap acronym). For example, a Microsoft web developer may use VBScript instead of JavaScript to serve up Access Database data that is transformed into JSON (not XML) using a .NET server-side script. Today, that guy's site would still be considered an AJAX site, rather than an AVAJ site (yep, AJAX just sounds cooler).

In fact, it's getting to the point where just about *anything* on a website (that isn't in Flash) that slides, moves, fades, or pops up without rendering a new browser window is considered an 'Ajaxy' site. In truth, a large portion of these sites don't truly qualify as using AJAX, they're just using straight-up JavaScripting. Generally, if you use cool JavaScripts in your WordPress site, it will probably be considered 'Ajaxy', despite not being asynchronous or using any XML.

We're going to take a look at the most popular methods to get you going with AJAX in WordPress using plug-ins and widgets to help you include dynamic self-updating content and create interactive forms in your WordPress site. While we're at it, we'll also look at some cool JavaScript toolkits, libraries, and scripts you can use to appear 'Ajaxy'.

Want more info on this AJAX business? The w3schools site has an excellent introduction to AJAX, explaining it in straight-forward, simple terms. They even have a couple of great tutorials that are fun and easy to accomplish, even if you only have a little HTML, JavaScript, and server-side script (PHP or ASP) experience (no XML experience required) (`http://w3schools.com/ajax/`).

Preparing for Dynamic Content and Interactive Forms

Gone are the days of clicking, submitting, and waiting for the next page to load, or manually compiling your own content from all your various online identities to post into your site.

A web page using AJAX techniques (if applied properly) will give the user a smoother and leaner experience. Click on a drop-down option and the checkbox menus underneath are updated immediately with the relevant choices—no submitting, no waiting. Complicated forms that, in the past, took two or three screens to process can be reduced into one convenient screen by implementing the form with AJAX.

As wonderful as this all sounds, I must again offer a quick disclaimer. I understand that, as with drop-down menus and Flash, you may want or your clients are demanding that AJAX be in their sites. Just keep in mind, AJAX techniques are best used in situations where they truly benefit the user's experience of the page, for example, being able to add relevant content via a widget painlessly or cutting a lengthy web process from three pages down to one. In a nutshell, using an AJAX technique simply to say your site is an AJAX site is probably not a good idea.

You should be aware that, if not implemented properly, some uses of AJAX can compromise the security of your site. You may inadvertently end up disabling key web browser features (like back buttons or the history manager). Then there are all the basic usability and accessibility issues that JavaScript, in general, can bring to a site.

Some screen readers may not be able to read a 'new' screen area that's been generated by JavaScript. If you cater to users who rely on tabbing through content, navigation may be compromised once new content is updated. There are also interface design problems that AJAX brings to the table (and Flash developers can commiserate). Many times, in trying to limit screen real estate and simplify a process, developers actually end up creating a form or interface that is complex and confusing, especially when your user is expecting the web page to *act* like a *normal* web page!

You Still Want AJAX on Your Site?

OK! You're here and reading this chapter because you want AJAX in your WordPress site. I only ask you take the just discussed into consideration and do one or more of the following to prepare.

Help your client assess their site's target users first. If everyone is web 2.0 aware, using newer browsers, and are fully mouse-able, then you'll have no problems, AJAX away. But if any of your users are inexperienced with RIA (Rich Interface Application) sites or have accessibility requirements, take some extra care. Again, it's not that you can't or shouldn't use AJAX techniques, just be sure to make allowances for these users. You can easily adjust your site's user expectations upfront, by explaining how to expect the interface to act. Again, you can also offer alternative solutions and themes for people with disabilities or browsers that can't accommodate the AJAX techniques.

Remember to check in with *Don't Make Me Think*, that Steve Krug book I recommended in Chapter 7 for help with any interface usability questions you may run into. Also, if you're really interested in taking on some AJAX programming yourself, I highly recommend *AJAX and PHP* by Cristian Darie, Bogdan Brinzarea, Filip Chereches-Tosa, and Mihai Bucica. In it, you'll learn the ins and outs of AJAX development, including handling *security issues*. You'll also do some very cool stuff like make your own Google-style auto-suggest form and a drag-and-drop sortable list (and that's just two of the *many* fun things to learn in the book).

So, that said, you're now all equally warned and armed with the knowledgeable resources I can think to throw at you. Let's get to it; how exactly do you go about getting something 'Ajaxy' into your WordPress site?

Plug-ins and Widgets

In these next few sections we're going to cover plug-ins and widgets. Plug-ins and widgets *are not* a part of your theme. They are additional files with WordPress compatible PHP code that are installed separately into their own directories in your WordPress installation (again, not in your theme directory). Once installed, they are available to be used with *any* theme that is also installed in your WordPress installation.

Even though plug-ins and widgets are not the part of your theme, you might have to prepare your theme to be compatible with them.

Let's review a bit about plug-ins and widgets first.

Plug-ins

WordPress has been built to be a lean, no frills publishing platform. Its simplicity means that with a little coding and PHP know-how, you can easily expand WordPress's capabilities to tailor to your site's specific needs. Plug-ins were developed so that even *without* a little coding and PHP know-how, users could add extra features and functionality to their WordPress site painlessly, via the Administration Panel. These extra features can be just about anything—from enhancing the experience of your content and forms with AJAX, to adding self-updating 'listening/watching now' lists, Flickr feeds, Google Map info and Events Calendars; you name it, someone has probably written a WordPress plug-in for it.

Take a look at the WordPress Plug-in page to see what's available:

`http://wordpress.org/extend/plugins/`

Widgets

Widgets are basically just another plug-in! The widget plug-in was developed by **AUTOMATTIC** (`http://automattic.com/code/widgets/`), and it allows you to add many more kinds of self-updating content bits and other useful 'do-dads' to your WordPress site. Widgets are intended to be smaller and a little more contained than a full, stand-alone plug-in, and they usually display *within* the side bar of your theme (or wherever you want; don't panic if you're designing a theme without a sidebar).

If you're using WordPress version **2.2 and up**, the widget plug-in has become a part of WordPress itself, so you no longer need to install it before installing widgets. Just look through the widget library on WordPress's widget blog and see what you'd like! (`http://widgets.wordpress.com/`)

Trying to download Widgets but the links keep taking you to Plug-in download pages? You'll find that many WordPress Widgets 'piggyback' on WordPress Plug-ins, meaning you'll need the full plug-in installed in order for the widget to work or the widget is an additional feature of the plug-in. So don't be confused when searching for widgets and all of a sudden you're directed to a plug-in page.

WordPress Widgets are intended to perform much the same way Mac OS's Dashboard Widgets and Windows Vista Gadgets work. They're there to offer you a quick overview of content or data and maybe let you access a small piece of often used functionality from within a full application or website, without having to take the time to launch the application or navigate to the website directly. In a nutshell, widgets can be very powerful, while at the same time, just don't expect too much.

Getting Your Theme Ready for Plug-ins and Widgets

In this chapter, we'll take a look at what needs to be done to prepare your theme for plugins and widgets.

Plug-in Preparations

Most WordPress Plug-ins can be installed and will work just fine with your theme, with no extra effort on your part. You'll generally upload the plug-in into your `wp_content/plugins` directory and activate it in your Administration Panel. Here are a few quick tips for getting a plug-in displaying well in your theme:

1. When getting ready to work with a plug-in, read *all* the documentation provided with the plug-in before installing it and follow the developer's instructions for installing it (don't assume just because you've installed one plug-in, they all get installed the same way).
2. Occasionally, a developer may mention the plug-in was made to work best with a specific theme, and/or the plug-in may generate content with XHTML markup containing a specific CSS `id` or `class` rule. In order to have maximum control over the plug-in's display, you might want to make sure your theme's stylesheet accommodates any `id` or `class` rules the plug-in outputs.
3. If the developer mentions the plug-in works with say, the Kubrick theme, then, when you install the plug-in, view it using the Kubrick theme (or any other theme they say it works with), so you can see how the plug-in author *intended* the plug-in to display and work within the theme. You'll then be able to duplicate the appropriate appearance in your theme.

Installing the AJAX Comments Plug-ins

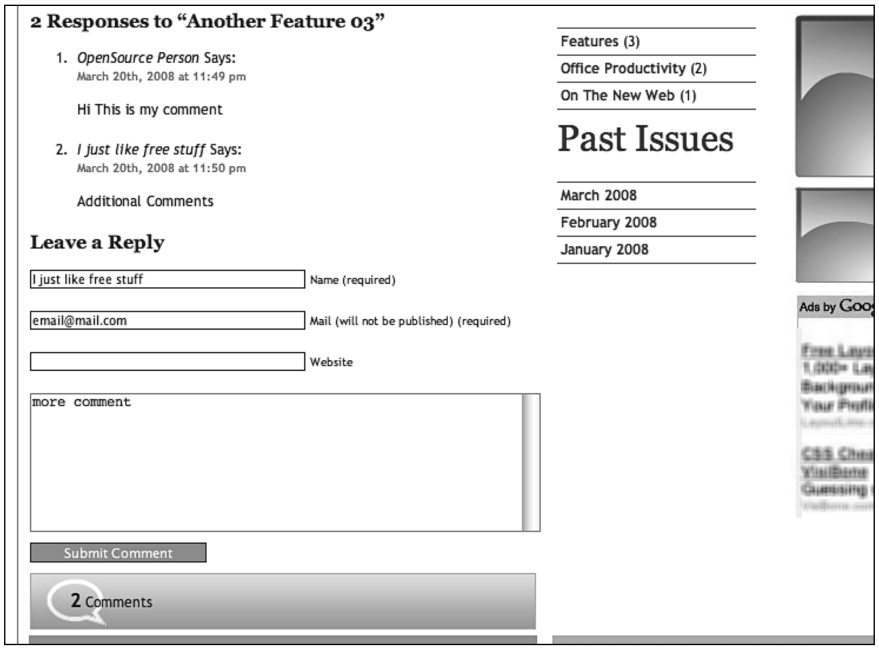
As I mentioned earlier, AJAX can really enhance the user's experience when it comes to forms. The most used form on a blog would be the comment form. Let's look at a plug-in that can really speed and tidy up the comment process. I'll be installing Mike Smullin's AJAX Comments Plug-in. You can get it from `http://wordpress.smullindesign.com/plugins/ajax-comments`.

If you can't spare the dollar that ol' Mike is asking for, you can also use Regua's **AJAX Comment Posting** plug-in (`http://wordpress.org/extend/plugins/ajax-comment-posting/`).

Regua's plug-in is good, but I just really like Mike Smullin's plug-in it's very light and works quickly. Well worth the dollar I spent on it. Here's the best part installing it:

Time For Action:

1. Unzip and upload the `ajax-comments` directory into the `wp-content/plugins` directory.
2. Go to **Administrator | Plug-ins** panel and **Activate** it.
3. Use it. That's it! The user sees their comment updated immediately with a note that the comment is awaiting approval. It's nice for the moment and they feel 'heard', but you might not ever actually approve the comment depending on its content.


2 Responses to "Another Feature 03"

1. *OpenSource Person* Says:
   March 20th, 2008 at 11:49 pm

   Hi This is my comment

2. *I just like free stuff* Says:
   March 20th, 2008 at 11:50 pm

   Additional Comments

Leave a Reply

I just like free stuff	Name (required)
email@mail.com	Mail (will not be published) (required)
	Website

```
more comment
```

Widget Preparations

Some plug-ins, like the widget plug-in (again you don't have to install this if you're using version WordPress 2.2 and up), do require your theme to go through some more formal preparation. You'll need to do the following to make your theme compatible with widgets (a.k.a 'Widgetized').

Time For Action:

1. Your side bar should ideally be set up using an unordered list format. If it is, you can add this code within your side bar: (If your sidebar is not set up using an unordered list format, ignore this step, but pay attention in step 3.)

```
<ul id="sidebar">
<?php if ( !function_exists('dynamic_sidebar')
        || !dynamic_sidebar() ) : ?>
 <li id="about">
  <h2>About</h2>
  <p>This is my blog.</p>
 </li>
</ul>
```

2. Because we deconstructed the default WordPress theme, based on the famous Kubrick theme, there is a `funcitons.php` file in our theme that already has the widgets registered for the sidebar. If by some chance you started completely from scratch or lost that file, you simply need to create a `functions.php` file in your `themes` folder and add this code to it:

```
<?php
if ( function_exists('register_sidebar') )
    register_sidebar(array(
        'before_widget' => '<li id="%1$s"
                  class="widget %2$s">',
        'after_widget' => '</li>',
        'before_title' => '<h2 class="widgettitle">',
        'after_title' => '</h2>',
    ));
?>
```

3. My problem is that my sidebar format is much customized and it's not in a simple unordered list. Plus, I have two sidebars. I'd want the second sidebar that holds my GoogleAdSense to contain a widget or two, but not my 'Table of Contents' sidebar. Not a problem! The code we entered above in the `functions.php` file helps us with our more traditional div-header-list structure. Add this code to your non-unordered list sidebar:

   ```
   <div id="sidebar">
   <?php if ( !function_exists('dynamic_sidebar')
                    || !dynamic_sidebar() ) : ?>
    <div class="title">About</div>
    <p>This is my blog.</p>
    <div class="title">Links</div>
    <ul>
     <li><a href="http://example.com">Example</a></li>
    </ul>
   <?php endif; ?>
   </div>
   ```

4. You've got two sidebars and you want them both to be dynamic? Instead of `register_sidebar()`, use `register_sidebars(n)`, where `n` is the number of sidebars. Place them before the array bit of code if you're using a non-unordered list sidebar, like so:

   ```
   <?php
   if ( function_exists('register_sidebar') )
       register_sidebar(n, array(
           'before_widget' => '<li id="%1$s"
                      class="widget %2$s">',
           'after_widget' => '</li>',
           'before_title' => '<h2 class="widgettitle">',
           'after_title' => '</h2>',
       ));
   ?>
   ```

Then place the appropriate number in the `dynamic_sidebar()` function, starting with `1`. For example:

```
<div id="sidebar1">
<?php if ( !function_exists('dynamic_sidebar')
                || !dynamic_sidebar(1) ) : ?>
<div class="title">About</div>
```

Your theme is now 'Widgetized'. For those of you who are looking forward to creating commercial themes be sure to tell everyone your theme is widget-friendly.

Like Widgets? Learn all about how to control their display in your theme and even develop your own. Check out AUTOMATTIC's Widget API Documentation at `http://automattic.com/code/widgets/api/`.

Additional Considerations: There are no concrete standards for widgets as of yet (though, the W3C is working on it (`http://www.w3.org/TR/widgets/`). Many WordPress widgets, like Google Reader, are flexible and can handle just about any size column. Some widgets may require a minimum column size! You may need to adjust your theme if the widget has an inflexible size. Some widgets (especially the ones that display monetized ads for your site) have display requirements and restrictions. Be sure to thoroughly investigate and research any widget you're interested in installing on your site.

Installing the Google Reader Widget

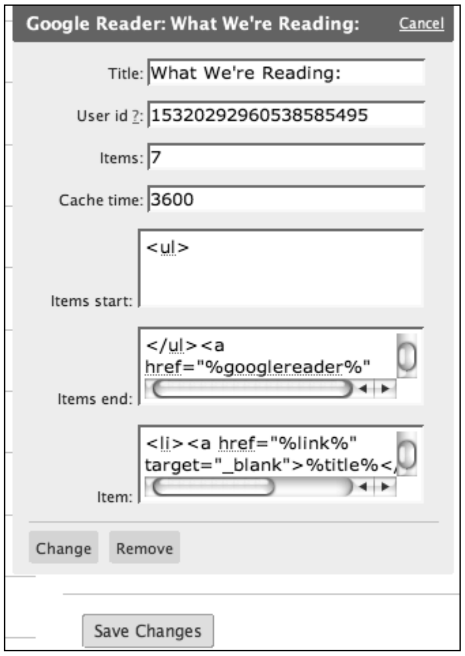
I do a lot of online reading, thank goodness for RSS feeds. I used to load-in all sorts of RSS feeds to my site to show people what I was reading, but that's not very accurate. It only shows what sites I usually go to, and what I *might* have read on that site. With all the new sites and blogs coming and going, I'd have old feeds left on my site, it got to be ugly, and I eventually stripped them all out.

Google Reader has a *shared* feed that lets people know exactly what I really have been reading and interested in. Thanks to this handy widget by James Wilson, I can share what I'm really reading, in real-time, quickly and easily. Once your theme is widget-compatible, it's pretty much just as simple to get a widget up and running as a plug-in. Get the Google Reader Widget from `http://wordpress.org/extend/plugins/google-reader-widget/`.

Time For Action:

1. Unzip and drop `googlereader.php` file into the `wp-content/plugins` directory. (Depending on the widget, be sure to read the author's instructions. Some will want you to install to the `wp-content/plugins` directory and some will want you to install to the `wp-content/plugins/widgets` directory. You might have to create the widget directory.)
2. Go to **Administration | Plug-ins** and **Activate** it.

3. Go to **Administration | Presentation | Widgets** and drag the widget to your *sidebar* area.



Title: What We're Reading:

User id ?: 15320292960538585495

Items: 7

Cache time: 3600

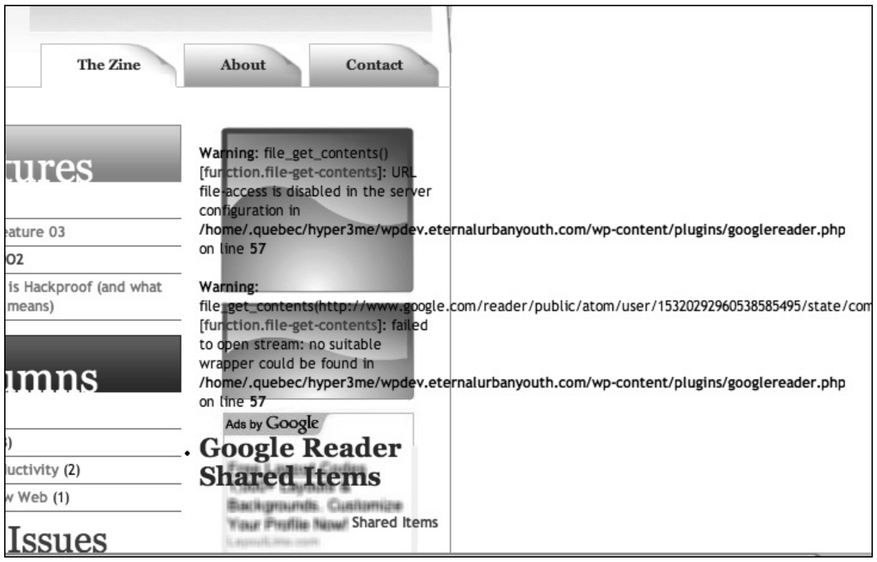
Items start: `<ul>`

Items end: `</ul><a href="%googlereader%"`

Item: `<li><a href="%link%" target="_blank">%title%<`

4. View it on your site.

I ran into a snag with the Google Reader Widget:



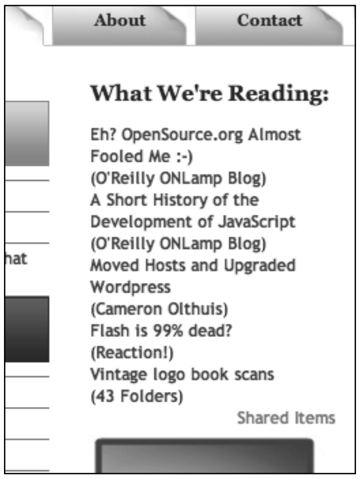
```
Warning: file_get_contents() [function.file-get-contents]: URL file-access is disabled in the server configuration in /home/.quebec/hyper3me/wpdev.eternalurbanyouth.com/wp-content/plugins/googlereader.php on line 57

Warning: file_get_contents(http://www.google.com/reader/public/atom/user/15320292960538585495/state/com [function.file-get-contents]: failed to open stream: no suitable wrapper could be found in /home/.quebec/hyper3me/wpdev.eternalurbanyouth.com/wp-content/plugins/googlereader.php on line 57
```

I had to read the FAQ for the Google Reader Widget to learn that my hosting provider doesn't approve of the `file_get_contents()` method (`http://wordpress.org/extend/plugins/google-reader-widget/faq/`). So I had to modify my `googlereader.php` file at line 57 with the following workaround the widget author recommended:

```
$ch = curl_init();
$timeout = 5; // set to zero for no timeout
curl_setopt ($ch, CURLOPT_URL, $uri);
curl_setopt ($ch, CURLOPT_RETURNTRANSFER, 1);
curl_setopt ($ch, CURLOPT_CONNECTTIMEOUT, $timeout);
$stories = curl_exec($ch);
curl_close($ch);
```

After making this tweak, the Widget worked fine:

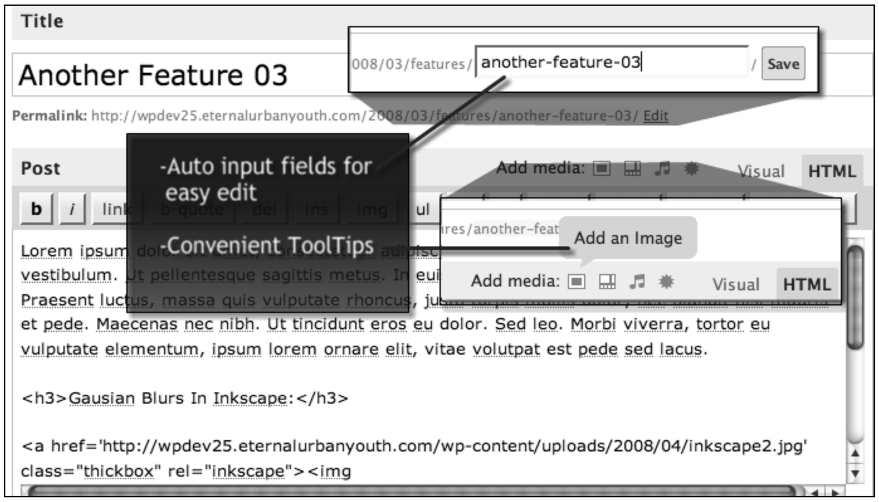


What We're Reading:

Eh? OpenSource.org Almost Fooled Me :-)
(O'Reilly ONLamp Blog)
A Short History of the Development of JavaScript
(O'Reilly ONLamp Blog)
Moved Hosts and Upgraded Wordpress
(Cameron Olthuis)
Flash is 99% dead?
(Reaction!)
Vintage logo book scans
(43 Folders)

AJAX–It's Not Just for Your Site Users

I've already mentioned how, when applied properly, AJAX can aid in interface usability. WordPress attempts to take advantage of this within its Administration Panel by enhancing it with relevant information and compressing multiple page forms into one single-screen area. The following is a quick look at how WordPress uses AJAX to enhance it's Administration Panel forms:



Title

Another Feature 03

008/03/features/ another-feature-03 / Save

Permalink: http://wpdev25.eternalurbanyouth.com/2008/03/features/another-feature-03/ Edit

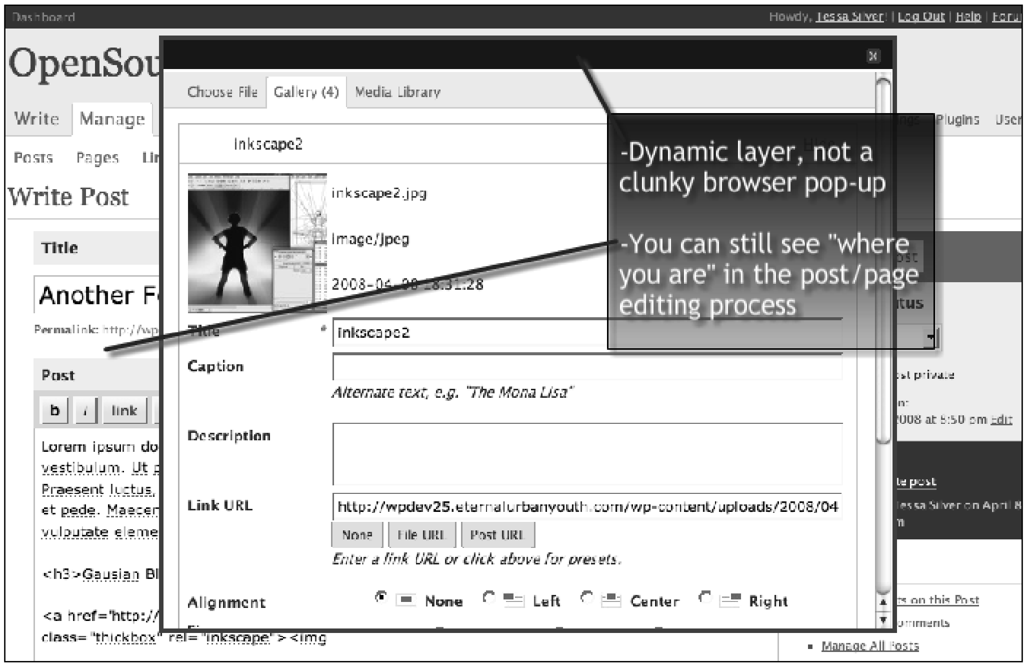
Post — Add media: — Visual — HTML

b i link b-quote del ins img ul

Lorem ipsum dolor ... vestibulum. Ut pellentesque sagittis metus. In eui... Praesent luctus, massa quis vulputate rhoncus, ju... et pede. Maecenas nec nibh. Ut tincidunt eros eu dolor. Sed leo. Morbi viverra, tortor eu vulputate elementum, ipsum lorem ornare elit, vitae volutpat est pede sed lacus.

```
<h3>Gausian Blurs In Inkscape:</h3>

<a href='http://wpdev25.eternalurbanyouth.com/wp-content/uploads/2008/04/inkscape2.jpg'
class="thickbox" rel="inkscape"><img
```


-Dynamic layer, not a clunky browser pop-up
-You can still see "where you are" in the post/page editing process
Choose File
Gallery (4)
Media Library
Dashboard
Write
Manage
Posts
Pages
Write Post

Inkscape2

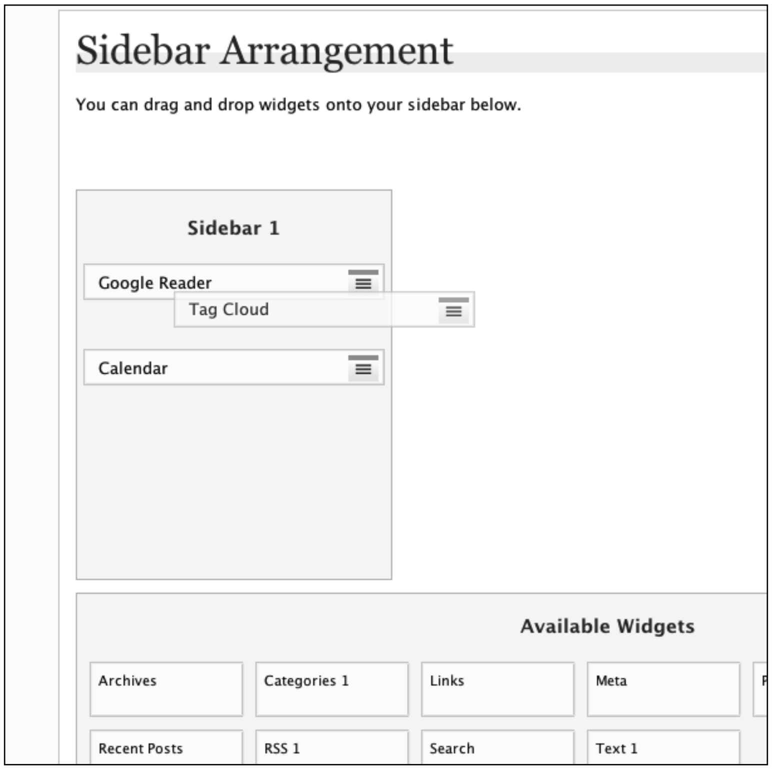
inkscape2.jpg

image/jpeg

2008-04-08 18:31:28

Title *	Inkscape2
Caption	
	Alternate text, e.g. "The Mona Lisa"
Description	
Link URL	http://wpdev25.eternalurbanyouth.com/wp-content/uploads/2008/04
	None / File URL / Post URL
	Enter a link URL or click above for presets.
Alignment	(● None) ○ Left ○ Center ○ Right

Even if you haven't upgraded to WordPress **2.5**, WordPress **2.3** makes use of AJAX on the widgets panel, allowing you to easily drag-and-drop to add and arrange your sidebar widgets. (For some reason, this has been redesigned in **2.5**; I would have preferred if it had stayed the same).



pageMash

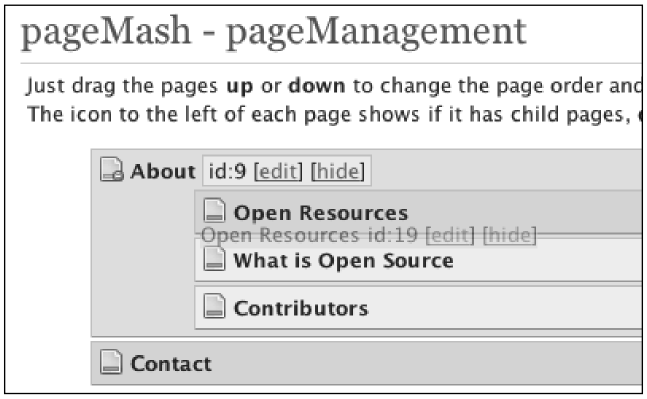
In addition to finding plug-ins and widgets that enhance your theme, you should consider looking for plug-ins that enhance your administration experience of WordPress! For example, if your WordPress site has a lot of pages and/or you display your page links as drop-down menus, as discussed in Chapter 7, then, Joel Starnes pageMash plug-in is for you.

pageMash is a great little plug-in that uses the MooTools framework and Moo.fx library. Instead of having to go into each *individual* page's editor view and then use the **Page Parent** view to manipulate your pages around into your hierarchical structure, this plug-in lets you reorder and assign pages as parents and sub-pages on-the-fly.

Time For Action:

1. Download the pageMash plug-in from: `http://wordpress.org/extend/plugins/pagemash/`.
2. Unzip the files and upload the `pagemash` directory to your `/wp-content/plugins/` directory.
3. Go to **Administration | Plug-ins** and **Activate** it. pageMash will then show up under the **Administration | Manage tag**.

I hope you can get an idea by the following screenshot about how much easier and quicker it is to arrange your WordPress pages with pageMash.


pageMash - pageManagement

Just drag the pages **up** or **down** to change the page order and
The icon to the left of each page shows if it has child pages,

About id:9 [edit] [hide]
Open Resources
Open Resources id:19 [edit] [hide]
What is Open Source
Contributors
Contact

The AJAX Factor

Aside from the many-interface enhancing, time-saving benefits of Ajax, sometimes you do just want to 'wow' your site visitors. It's easy to give your site an 'Ajaxy' feel, regardless of asynchronously updating it with server-side XML, just by sprucing up your interface with some snappy JavaScripts. The easiest way to get many of these effects is to reference a JavaScript library (sometimes called a toolkit or framework, depending on how robust the provider feels the code is). A few of the leading favorites in the AJAX community (in no particular order) are:

1. Script.aculo.us: (`http://script.aculo.us/`)
2. Prototype: (`http://www.prototypejs.org/`)
3. jQuery: (`http://jquery.com/`)

There's also:

4. MooTools: (`http://mootools.net/`)
5. Moo.fx: (`http://moofx.mad4milk.net/`)

Prototype is more of a framework and Script.aculo.us is more of an add-on toolkit or set of libraries for neat effects. In fact, Script.aculo.us references the Prototype framework, so your best bet is probably to use Script.aculo.us, but if you do work with it, *be sure* to check out Prototype's site and try to understand what that framework does.

Moo.fx is the smallest JavaScript effects library (boasting a 3k footprint), but again, it needs to be supported by the MooTools or Prototype frameworks.

jQuery is my personal favorite. It pretty much stands on its own without needing to be backed up by a more robust framework (like Prototype), but you can still do some very robust things with it, manipulating data and the DOM, plus it's packed with neat and cute visual effects, similar to Script.aculo.us.

Using JavaScript libraries like the above, you'll be able to implement their features and effects with simple calls into your WordPress posts and pages.

JavaScript Component Scripts

The fun doesn't stop there! What's that? You don't have time to go read up on how to use a JavaScript library like jQuery? Never fear! There are many other JavaScript effect components and libraries that are built using the libraries above. One of the most popular scripts out there that makes a big hit on any website is **Lightbox JS**:

`http://www.huddletogether.com/projects/lightbox2/`

Lightbox JS is a 'simple, unobtrusive script used to overlay images on the current page.' It's great, but it uses both the Prototype and Script.aculo.us libraries to achieve its effects. I also found that Lightbox was limited to only displaying images and a hair difficult to manipulate it to handle anything more than that. What if I wanted to display XHTML text, or markup containing YouTube videos, maybe even make an AJAX request to the server?

Enter **Thickbox**: `http://jquery.com/demo/thickbox/`

Thickbox is very similar to Lightbox JS. It uses only the jQuery library, and in addition to handling images similar to Lightbox JS, it can also handle in-line content, iFrame content, and AJAX content (be sure to check out the examples on the ThickBox page!). The downside—Thickbox doesn't do that *smooth* animation that Lightbox JS (version 2) does when images are different sizes. This is the trade-off I made when I decided it was more important to be able to display more than just images in my OpenSource Magazine theme.

Depending on your theme's design and what type of blog or site you're creating it for, you may opt to use Lightbox instead or something all together different! It's your theme, don't feel limited to what I specifically discuss in this book. I'll walk you through the process of installing ThickBox, but many 'Ajaxy' scripts that use these JavaScript libraries/frameworks are installed similarly. Just follow the instructions in the `ReadMe` files and you're on your way to an enhanced theme.

Time For Action:

1. This is an extremely easy-to-implement script. After downloading it, add the key `js` and `css` files to your WordPress theme's `home.php` and `header.php` files using the `bloginfo` template tag to target your theme:

   ```
   <script type="text/javascript" src="<?php bloginfo
   ('template_directory'); ?>/js/jquery.js"></script>
   <script type="text/javascript" src="<?php bloginfo
   ('template_directory'); ?>/js/thickbox.js"></script>
   ```

2. You'll also add in a call to the ThickBox CSS file:

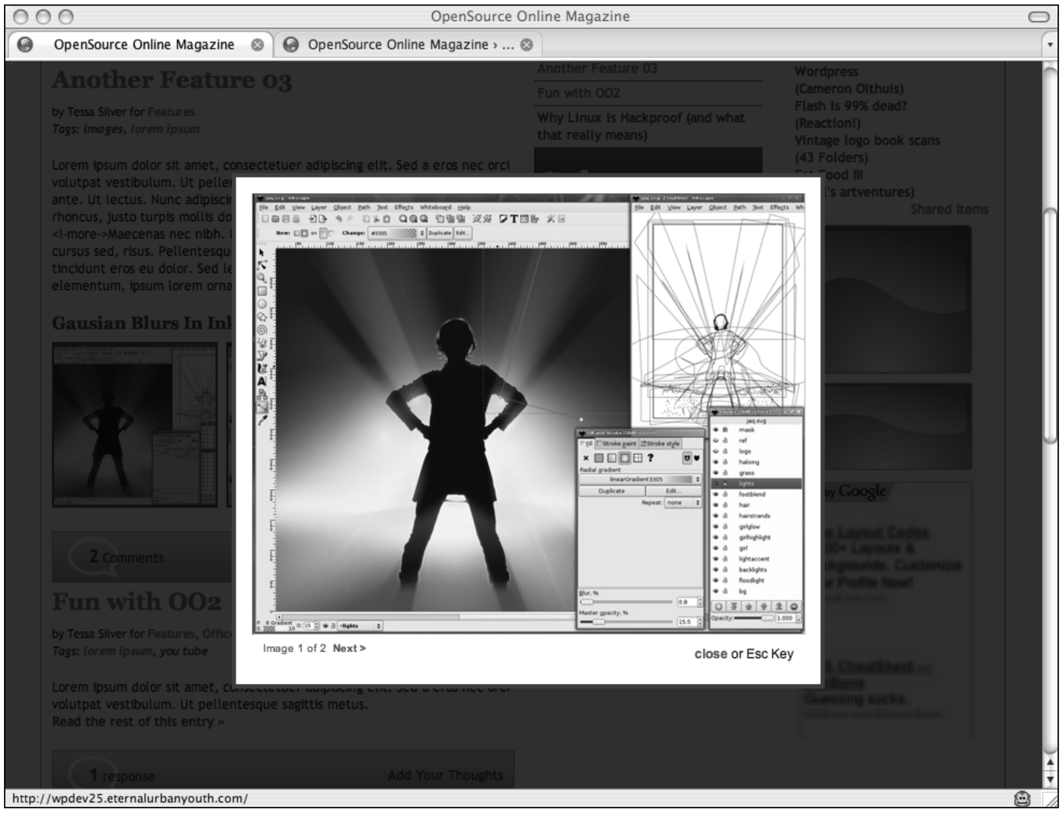
   ```
   <style type="text/css" media="all">@import "<?php bloginfo
   ('template_directory'); ?>/thickbox.css";</style>
   ```

3. Don't forget to upload the `loadingAnimation.gif` and `macFFBgHack.png` images to your theme directory and update the `thickbox.js` and `thickbox.css` files as per the `ReadMe` file instructions.
4. Then, you can create a post (or page) in your Administration Panel and using the Code View add in basic `href` links around your image tags, like so:

   ```
   <a href='/wp-content/uploads/2008/04/inkscape2.jpg'
   class="thickbox" rel="inkscape"><img src="/wp-content/
   uploads/2008/04/inkscape2-150x150.jpg" alt="" title="inkscape2"
   width="150" height="150" class="alignnone size-thumbnail
   wp-image-15" /></a>
   <a href='/wp-content/uploads/2008/04/inkscape1.jpg'
   class="thickbox" rel="inkscape"><img src="/wp-content/
   uploads/2008/04/inkscape1-150x150.jpg" alt="" title="inkscape1"
   width="150" height="150" class="alignnone size-thumbnail
   wp-image-14" /></a>
   ```

I uploaded the images via WordPress's built-in up-loader and let WordPress create the thumbnails; I just added the captions to the title attribute, the `rel` attribute and the `thickbox` class by hand.

That's it!


OpenSource Online Magazine
OpenSource Online Magazine
OpenSource Online Magazine › ...
Another Feature 03
by Tessa Silver for Features
Tags: images, lorem ipsum
Another Feature 03
Fun with 002
Why Linux is Hackproof (and what that really means)
Wordpress (Cameron Olthuis)
Flash is 99% dead? (Reaction!)
Vintage logo book scans (43 Folders)
Shared Items
Gausian Blurs In Inl
2 Comments
Fun with 002
by Tessa Silver for Features, Offic
Tags: lorem ipsum, you tube
1 response
Add Your Thoughts
Image 1 of 2 Next >
close or Esc Key
http://wpdev25.eternalurbanyouth.com/

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Sed a eros nec orci volutpat vestibulum. Ut pellentesque sagittis metus.
Read the rest of this entry »

Summary

In this chapter, we reviewed a few ways to take advantage of AJAX on your WordPress site. We 'Wigitized' our theme and downloaded and installed a couple of useful plug-ins, and looked at using jQuery and ThickBox to enhance post and page content. Up next—let's take a look at some final design tips to working with WordPress.