ADOBE® DREAMWEAVER® CS5 with PHP

TRAINING from the SOURCE

David Powers

Adobe

Adobe® Dreamweaver® CS5 with PHP: Training from the Source

David Powers

Adobe Press books are published by:
Peachpit
1249 Eighth Street
Berkeley, CA 94710
510/524-2178
800/283-9444

For the latest on Adobe Press books, go to www.adobepress.com
To report errors, please send a note to errata@peachpit.com
Peachpit is a division of Pearson Education.

Acquisitions Editor: Victor Gavenda
Project Editor: Rebecca Freed
Development Editor and Copyeditor: Anne Marie Walker
Production Editor: Becky Winter
Technical Editor: Tom Muck
Compositor: Danielle Foster
Indexer: Rebecca Plunkett
Cover Design: Charlene Charles-Will

Notice of Rights

Notice of Liability

Trademarks

ISBN-13: 978-0-321-71984-3
ISBN-10: 0-321-71984-0
9 8 7 6 5 4 3 2 1
Printed and bound in the United States of America

Contents

Introduction

My first encounter with PHP came about 10 years ago. By that time, I already had plenty of experience developing websites. I had started out writing HTML in a text editor before settling on Dreamweaver as my favorite authoring tool. A new project involved publishing more than 30 articles a day. It was a subscription service, so the site needed to be password-protected and searchable. An ordinary website wouldn't do. That's when PHP came to the rescue.

PHP makes communication with a database a breeze, so content can be stored in the database, making it searchable. Instead of creating a new page for every article, pages are populated dynamically with the requested items. You can also password-protect the administrative or members-only area of a site. PHP does a lot more: It can send email, upload files, and attach files to emails—all of which you'll learn how to do in this book. PHP is also the driving force behind the three most popular content management systems: Drupal, Joomla!, and WordPress.

So, where does Dreamweaver come into the picture? Dreamweaver has supported PHP to some degree since 2002, mainly through server behaviors, which automatically generate PHP code for some basic tasks. But the level of support has taken a quantum leap forward in Dreamweaver CS5. The server behaviors are still there (see Lesson 6), but they take a back seat.

The big changes lie in code hinting, embedded PHP documentation (including examples), autocompletion of variables, automatic discovery of dynamically related files, and—perhaps best of all—the ability to view and navigate through PHP pages without leaving the Document window. As a result, it's now possible to style WordPress, Joomla!, and Drupal in Dreamweaver CS5 without the need to generate static pages. These changes are described in detail in Lesson 1, but in a nutshell they should appeal to designers and developers alike.

PHP's popularity springs from being easy to learn. You can achieve practical results very quickly. Of course, like any skill, becoming an expert takes time and practice. The new PHP features in Dreamweaver CS5 not only help the learning process, but you'll find them even more useful as you gain experience. Dreamweaver is my preferred choice for designing the look of a website and organizing files, but I was beginning to use dedicated PHP authoring tools for the dynamic aspects of development. Dreamweaver CS5 has changed all that. I now have the best of both worlds in the same program.

Who This Book Is for

This is a "beyond the basics" book, so you should already have a solid understanding of how a website is built. You should also have a good understanding of HTML, because PHP code needs to be embedded in the underlying structure of a page to display the dynamic output. It's not necessary to know every tag and attribute, but if you don't know the difference between a `<ul>` and an `<li>` tag, you'll be lost. All the example files and exercises are styled with CSS, but design is not the focus of this book. You don't need to understand CSS to work through the lessons, but your web development skills would certainly be the better for it. You'll also find it makes it easier to follow Lesson 4, where you create a new WordPress theme.

You don't need prior knowledge of PHP. This book doesn't teach PHP in a formal manner, but Lesson 3 provides a crash course in how to write PHP, and Lesson 5 teaches the basics of database design using MySQL, the most popular open source database.

If you already know some PHP, all the better. This book moves at a fairly rapid pace. Lessons 7–12 make extensive use of the Zend Framework, a powerful library of PHP components that take a lot of hard work out of creating dynamic sites. Lesson 12 also uses the jQuery JavaScript framework. Again, you don't need prior knowledge of jQuery or JavaScript, but it will certainly help.

How to Use This Book

Time is precious, so you probably want to jump straight to the solution for your current problem. If you have considerable PHP experience, that approach might work. However, the majority of readers should start with Lesson 1 and work through each one in sequence because each lesson builds on the previous one. If you skip ahead, you're likely to miss a vital explanation and will need to backtrack anyway.

The "Approximate Time" at the beginning of each lesson is simply an estimate of the time it will take to work through the exercises. Don't regard it as a challenge, and don't feel downcast if you take much longer. Each lesson is packed with information. Take time to absorb it, and break the lesson into smaller chunks to match your own pace.

Most lessons contain reference sections followed by hands-on exercises. Each step explains not only what to do, but also why you're doing it. The idea is to help you think about how you could apply the same techniques to your own projects. This isn't a point-and-click book, but instead is one that aims to stimulate your problem-solving abilities. The more you think, the more you're likely to get out of it.

Accompanying files

The accompanying CD contains all the files necessary to complete the exercises in this book. The only exceptions are the PHP/MySQL development environments described in Lesson 2 and the LightBox Gallery Widget in Lesson 12. PHP and MySQL are updated frequently, so it makes more sense to get the most recent versions from the source. In the case of the LightBox Gallery Widget, one object of the exercise is to show you how to install the Adobe Widget Browser and download widgets from the Adobe Exchange.

Lesson 2 describes how to set up the Dreamweaver site to work through the exercises in this book. The files for each lesson are in folders named lesson01, lesson02, and so on. There are no files for Lesson 13. For each lesson that contains exercises, there are normally three subfolders: completed, start, and workfiles. The workfiles folder is deliberately left empty; it's where you should create and save the files for the lesson's exercises. If you follow this structure, the exercise files will use the common style sheets that are stored in the styles folder.

To save time, many exercises have partially completed pages, which you should copy from the start folder to the workfiles folder for that lesson. The completed folder contains copies of the exercise files shown at various stages of completion.

In Lessons 10 and 11, you should create a folder called cms in the site root. The cms_complete folder contains a full working copy of the completed project.

NOTE: The files were created on a Windows computer but are fully compatible with Mac OS X. However, the path in library.php needs to be adjusted to match the location of the Zend Framework files. See Lesson 7 for details.

Windows/Mac differences

The few Dreamweaver CS5 and PHP differences between Windows and Mac OS X have been pointed out at relevant places in the book.

Keyboard shortcuts are given in the order Windows/Mac, but in the rare cases where there is no Mac equivalent, this has been pointed out. On some Mac keyboards, the Opt(ion) key is labeled Alt. On a UK Mac keyboard, use Alt+3 to type the hash symbol (#).

Using a multi-button mouse with a Mac is now so common that the instructions refer only to right-click. If you prefer a single-button mouse, use Ctrl-click.

Code portability

One of the pleasures of working with PHP is that it's platform-neutral. All the PHP code in this book works equally well on Windows, Mac OS X, and Linux. However, it's important to realize that different versions of PHP and MySQL have different functionality. Also, server administrators have the ability to turn off certain features. To use this book, your web server *must* be running PHP 5.2 and MySQL 4.1 or later. The code will not work with earlier versions.

Getting help

When you encounter a problem, the first person to look to for help is you. Did you skip a step or mistype the name of a variable or function? One of the quickest ways of finding an error is to use Dreamweaver's File Compare feature (choose Help > Using Dreamweaver CS5 > Creating and Managing Files > Comparing files for differences) to compare your file with the version in the completed folder.

File Compare requires a third-party file comparison utility. If you don't have one installed, WinMerge (http://winmerge.org) for Windows and TextWrangler (www.barebones.com/products/textwrangler/) for Mac OS X are both free.

If you can't solve the problem on your own and a quick search on the Internet doesn't produce the answer, post a question in the Adobe forums. The best one for PHP questions is the Dreamweaver Application Development forum at http://forums.adobe.com/community/dreamweaver/dreamweaver_development. I'm frequently there providing help, so you might even get an answer from me.

I also post updates and tutorials on my website at http://foundationphp.com/, and you can follow me on Twitter @foundationphp.

Every care has been taken to eliminate errors, but if you think you have found one, please email errata@peachpit.com with the details.

Layout conventions

The following text conventions are used throughout this book:

- **Boldface text.** Words in **bold text** indicate input that you should type in a field or the name of a file you should create.

- **Boldface code.** Code that is added or changes is displayed in boldface.

```
if ($_POST) {
  if (empty($_POST['username']) || empty($_POST['password'])) {
    $failed = TRUE;
  } else {
    require_once('library.php');
```

- **Long code.** Sometimes, code won't fit on a single line on the printed page. Where this happens, an arrow indicates the continuation of a broken line like this:

```
$result = $recaptcha->verify($_POST['recaptcha_challenge_field'],
➥ $_POST['recaptcha_response_field']);
```

- **Italics.** Text in *italics* is for emphasis or to introduce important concepts.

Let the Journey Begin

Above all, enjoy the experience that lies ahead. Even if you find working with code uncomfortable to begin with, PHP is not hard. Welcome to the ever-growing PHP community.

This page intentionally left blank

LESSON 4

What You Will Learn

In this lesson, you will:

- Examine the basic structure of Drupal, Joomla!, and WordPress
- Install WordPress 3.0 in your local testing environment
- Create a child theme based on the default WordPress Twenty Ten theme
- Use Live View, the CSS Styles panel, and Code Navigator to style WordPress
- Enable site-specific code hints for WordPress
- Edit a WordPress template

Approximate Time

This lesson takes approximately 2 hours 30 minutes to complete.

Lesson Files

Media Files:

lesson04/start/images/birds_bg_gradient.jpg
lesson04/start/images/cormorants.jpg
lesson04/start/images/cormorants-thumbnail.jpg
lesson04/start/images/screenshot.png
lesson04/start/images/seagulls.jpg
lesson04/start/images/seagulls-thumbnail.jpg

Starting Files:

lesson04/start/auth_keys.txt
lesson04/start/wordpress-3.0.zip

Completed Files:

lesson04/completed/functions.php
lesson04/completed/header.php
lesson04/completed/style.css

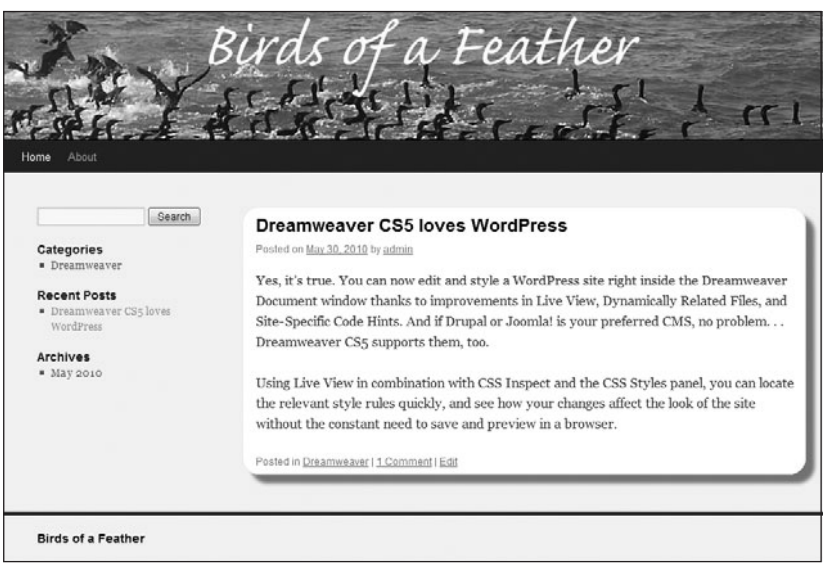
Dw

LESSON 4

Restyling a WordPress Site

Open source content management systems (CMSs), such as Drupal, Joomla!, and WordPress, take much of the hard work out of creating a dynamic website. WordPress claims—with some justification—that it takes only five minutes to install. The difficult part is trying to style a CMS to give it a unique look. That job is now considerably easier thanks to several new features in Dreamweaver CS5: navigable Live View, dynamically related files, CSS Inspect, and site-specific code hints.

Instead of constantly reloading the site in a browser to see the effect of your changes, you can now redesign your CMS entirely in the Document window. In this lesson, you'll adapt the default theme for a WordPress 3.0 site. The same basic principles apply to styling Drupal and Joomla!



The redesigned WordPress site.

Understanding the Structure of a CMS

Before embarking on restyling a WordPress site, it's worth spending a few moments examining how a CMS like WordPress, Drupal, or Joomla! is structured. A bare-bones Drupal installation consists of more than 460 files in 58 folders; WordPress has nearly 800 files in 79 folders; and Joomla! weighs in at a whopping 3,913 files in 711 folders. Unlike a website built with HTML, these files don't contain any of the site's content. In fact, the only page that most users ever see is index.php.

With the exception of images and other media files, all the content is stored in a database. The job of the army of files is to insert, update, and delete content in the database, and to serve visitors to the site with the information they want to see. If you open index.php in any of the CMSs, you see just a handful of PHP commands. There's nothing recognizable as a web page. Each part of the final web page is generated separately. Different scripts handle the page header, menus, main content, and footer.

This mass of files can be intimidating, even if you have a good understanding of PHP. As a result, many designers opt for using third-party themes (or templates, as Joomla! calls them) to improve the look of their CMS. There are plenty of good themes and templates available, and the default Twenty Ten theme in WordPress 3.0 is very attractive. But with the help of Dreamweaver CS5, it's not difficult to do your own customization—providing you have a strong grasp of CSS.

With a CMS, it's important to apply security fixes as soon as they're released, so you need to install your custom files in a place where they won't be overwritten. The location depends on the CMS you're using:

- **Drupal.** Create two subfolders called **modules** and **themes** in sites/all. The themes folder is where you install third-party themes or create your own.
- **Joomla!** Create a subfolder in the templates folder.
- **WordPress.** Create a subfolder in wp-content/themes.

Although the instructions in this lesson concentrate on creating a WordPress theme, the same principles of editing the CSS apply to Drupal and Joomla!

TIP: There's a tutorial by David Karlins on modifying Drupal themes with Dreamweaver CS5 at www.peachpit.com/articles/article.aspx?p=1590589.

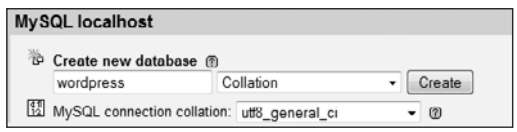
Installing WordPress

The following instructions assume you have created a PHP local testing environment as described in Lesson 2, and that your web server and MySQL are running.

Setting up a MySQL database and user account

Before you can install WordPress, you need to create a MySQL database and user account. Both subjects are covered in greater detail in Lesson 5, but the following steps guide you through the process of setting up a WordPress database.

1. Load phpMyAdmin in your browser, and log in as the root user if necessary.
2. In the "MySQL localhost" section in the center of the screen type **wordpress** in the "Create new database" text field. Leave all other settings at their default, and click Create.

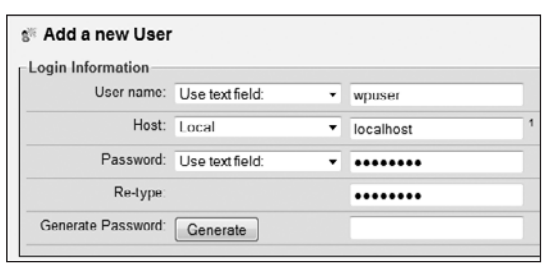


   You should see a message that the database has been created. You don't need to create any tables. WordPress does it for you.

3. Click the Home icon at the top left of the phpMyAdmin screen to return to the previous page. Then click the Privileges tab at the top of the screen.

CAUTION! Don't be tempted to click the Privileges tab on the previous screen. You must return to the phpMyAdmin welcome page to access the correct screen.

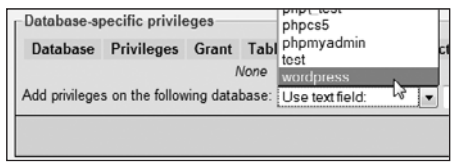
4. Click the "Add a new User" link halfway down the screen.
5. In the "Add a new User" section, type **wpuser** in the "User name" field.
6. Select Local from the Host menu to insert localhost in the Host field.
7. Type **P3@chp!T** in the Password field, and again in the Re-type field.



8 Scroll to the bottom of the page, and click Go.

phpMyAdmin reports that it has created the user and displays a page where you can edit the user's privileges. The first section, "Global privileges," gives the user the same privileges on all databases, which is insecure.

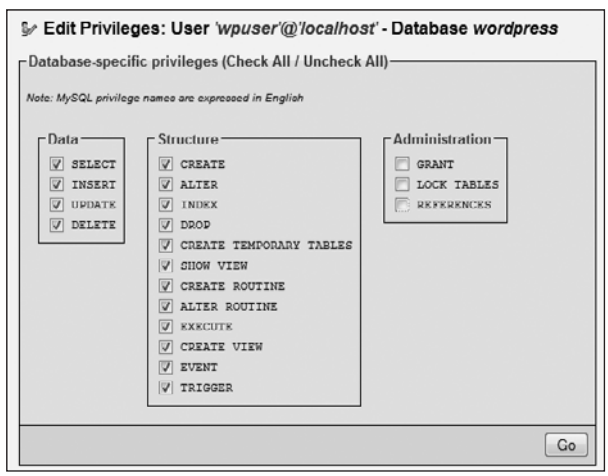
9 Scroll down to "Database-specific privileges" and select **wordpress** from the menu labeled "Add privileges on the following database."



This loads a new screen where you define the database-specific privileges.

10 You need to select all checkboxes except the three in the Administration box. The quickest way is to click "Check all," and then deselect the three Administration checkboxes.

11 Click the Go button in the "Database-specific privileges" section.



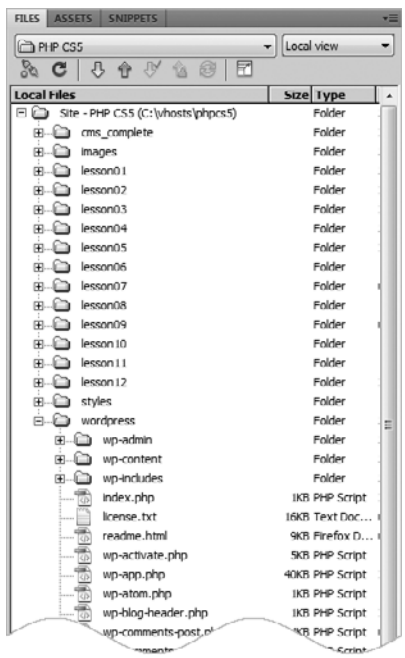
CAUTION! There are two Go buttons on this page. Make sure you click the top one.

You should see a message saying you have updated the privileges for 'wpuser'@'localhost'. You're now ready to install WordPress.

Adding WordPress to the phpcs5 site

Installing WordPress involves unzipping the files into the folder where you want to locate the CMS. This can be the site root or a subfolder. For this lesson, use a subfolder of the phpcs5 site you set up in Lesson 2. After extracting the files, you need to edit a configuration file filling in the details of the MySQL database. The rest of the installation process is automated.

1. Use lesson04/start/wordpress-3.0.zip or download the most recent version of WordPress from http://wordpress.org/.
2. Extract the files to the phpcs5 site root. This should create a folder called wordpress inside the phpcs5 site. The folder contains about 25 files and three subfolders: wp-admin, wp-content, and wp-includes.
3. Click the Refresh icon in the Dreamweaver Files panel to see the newly added folders and files.



4. Double-click wp-config-sample.php in the wordpress folder to open it in the Document window, and switch to Code view.

The first part of the script (around lines 18–34) defines the MySQL settings for the CMS. Replace the placeholder text in the first three lines with the name of the database, the user name, and password that you created in the previous section like this:

```
/** The name of the database for WordPress */
define('DB_NAME', 'wordpress');

/** MySQL database username */
define('DB_USER', 'wpuser');

/** MySQL database password */
define('DB_PASSWORD', 'P3@chp!T');
```

5 Scroll down to around line 45 to the following section of code:

```
define('AUTH_KEY',         'put your unique phrase here');
define('SECURE_AUTH_KEY',  'put your unique phrase here');
define('LOGGED_IN_KEY',    'put your unique phrase here');
define('NONCE_KEY',        'put your unique phrase here');
define('AUTH_SALT',        'put your unique phrase here');
define('SECURE_AUTH_SALT', 'put your unique phrase here');
define('LOGGED_IN_SALT',   'put your unique phrase here');
define('NONCE_SALT',       'put your unique phrase here');
```

This defines a series of measures designed to make it extremely difficult, if not impossible, for anyone to reuse cookies if the security of your site is breached. When creating your own WordPress site, you can use your own imagination to create unique character sequences, or you can use the automatic key generator at https://api.wordpress.org/secret-key/1.1/salt/.

For this lesson, use the values in lesson04/start/auth_keys.txt to replace the eight lines shown here.

NOTE: In the event that the security of a live WordPress site is breached, you should replace these eight values and update the file on your remote server immediately.

6 Save wp-config-sample.php as **wp-config.php**, and close both files.

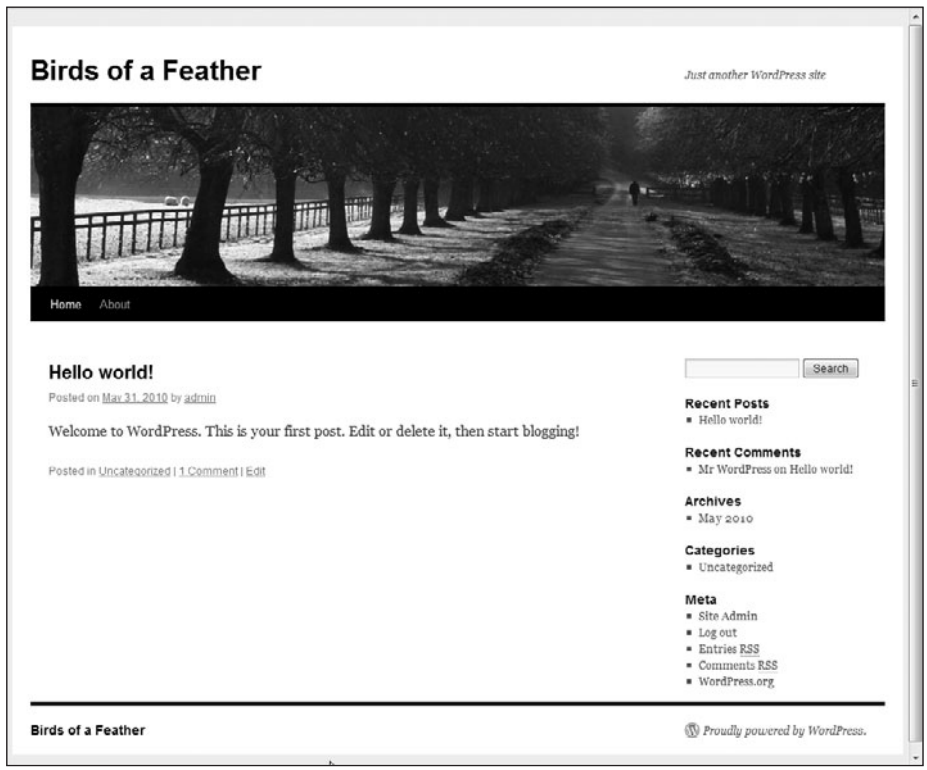
7 Launch your browser, and open wordpress/wp-admin/install.php in your phpcs5 site. The URL depends on how you set up your testing environment:

- **Virtual host.** http://phpcs5/wordpress/wp-admin/install.php
- **Localhost.** http://localhost/phpcs5/wordpress/wp-admin/install.php

NOTE: If you are using the MAMP default ports on a Mac, add :8888 after phpcs5 for a virtual host, or after localhost.

8. The install page asks for some basic information to set up the site. Type **Birds of a Feather** in the Site Title field.
9. Leave username at the default admin.
10. Type **C0rm0R@nT** in both Password fields.
11. For a live site, you should use a real email address in Your E-mail, because it's used to send alerts about posts waiting for approval. It's also used if you forget your password. A dummy address is fine for testing.
12. Deselect the checkbox that allows your site to appear in search engines like Google and Technorati. You won't be deploying this exercise on the Internet.
13. Click Install WordPress. In a few moments, you'll see a screen telling you that WordPress has been installed and inviting you to log in as admin.
14. Click Log In to open the login screen. Type **admin** in the Username field and **C0rm0R@nT** in the Password field. It's also a good idea to select the Remember Me checkbox to avoid the need to type these details every time.
15. Click Log In to enter the WordPress Dashboard, the administration center for a WordPress site.

16 Click the name of the site (Birds of a Feather) next to the WordPress logo at the top of the page to view the front page.



Why Can't I See the Multiscreen Button?

The screen shot on the next page shows a Multiscreen button in the Document toolbar, which isn't part of a default installation of Dreamweaver CS5. It comes from the HTML5 Pack that was released in May 2010, a month after Dreamweaver CS5 became available for purchase. Although the Multiscreen button isn't used in this book, the HTML5 Pack upgrades the version of the WebKit browser engine used in Live View to support CSS3 properties that are used later in this lesson.

If you don't see the Multiscreen button, check the status of the HTML5 Pack at http://labs.adobe.com/technologies/html5pack/. It's possible that during the lifetime of this book the pack's functionality will be added to Dreamweaver through the Adobe Updater. Download and install the HTML5 Pack using whichever method is available.

17 In Dreamweaver, double-click index.php in the wordpress folder to open it in the Document window. In Code view, there are just two lines of PHP code, together with a dozen or so lines of comments.

Switch to Design view, and click the Live View button. After a few moments, you should see the Birds of a Feather site in the Dreamweaver Document window. There are several files called index.php in a WordPress site. If you don't see the front page of the Birds of a Feather site in Live View, make sure you opened index.php in the top wordpress folder.

The default Twenty Ten theme in WordPress 3.0 has been designed to look good straight out of the box. But with the help of Dreamweaver CS5's new features, you'll learn how to develop your own theme to style WordPress.

Creating a WordPress Theme

Developing a WordPress theme from scratch requires considerable knowledge of CSS, HTML, PHP, and the WordPress application programming interface (API). The good news is that you can stand on the shoulders of others to adapt an existing theme as a *child theme*.

Child themes work on a similar principle to CSS. The child theme automatically inherits all the features of the existing theme, but you can decide which elements to override. The advantage is that the original files remain intact, so you can revert to the default if you change your mind or make a mistake. Also, if the parent theme is updated, you can replace all its files without worrying about losing your customizations because they're all stored in the child theme.

Most themes can be adapted as child themes. Before doing so, check the license. Some commercial themes have restrictions on how they can be used. The Twenty Ten theme used in the following exercises is released under the GNU General Public License (www.gnu.org/licenses/gpl.html), which means you are free to modify and redistribute it.

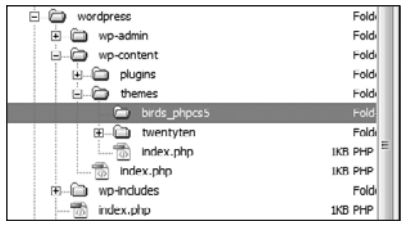
Preparing the files for a child theme

Themes consist of at least one style sheet and a number of WordPress templates. A WordPress template doesn't control a complete page. It's more like a Dreamweaver Library item in that it represents a fragment of a page. Each template is named after the part of the page it controls. For example, the Twenty Ten theme has templates called comments.php, footer.php, header.php, sidebar.php, and so on. If you open any of these files, you'll see a mixture of HTML and PHP. If you don't have any PHP experience, the code probably looks incomprehensible, but much of it is based on conditional statements. You'll gain plenty of experience with conditional statements in later lessons, so the code should be a lot easier to decipher by the time you have completed this book.

However, you don't really need to worry about the PHP code in the templates. A child theme requires only one file—a style sheet, which must be called style.css and reside in the child theme's top level folder. The child theme automatically uses the parent theme's templates and custom functions. In other words, at its simplest level, creating a child theme is just the WordPress way of attaching your own style sheet to an existing theme. But if you're feeling more ambitious, you can create your own templates and functions. When the active theme is a child theme, WordPress always looks first in the child theme's folder. If it finds the appropriate template or function there, it uses it. Otherwise, it uses the version in the parent theme's folder. For example, if you create your own version of header.php, WordPress uses it. But if you don't have your own version of footer.php, WordPress uses the one from the parent theme. This gives you the opportunity to experiment. You can copy a template file from the parent theme, and make some changes. If you like the result, you're on the way to developing your own theme. If it doesn't work, just delete the template file from your child theme's folder, and revert to the parent template.

Developing WordPress themes is a vast subject, so the exercises in this lesson only scratch the surface, but they demonstrate how quickly you can begin to style a WordPress site in Dreamweaver CS5.

1 In the Dreamweaver Files panel, expand the wordpress and wp-content folders, select the themes folder, right-click, and choose New Folder. Name the new folder **birds_phpcs5**. The new folder should be inside the themes folder at the same level as twentyten.



2 Expand the twentyten folder, and double-click style.css to open it in the Document window.

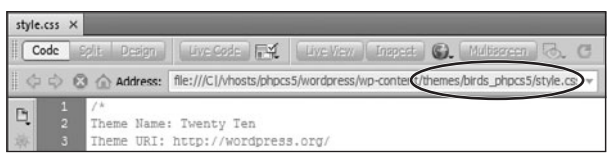
3 The first eight lines of style.css look like this:

```
/*
Theme Name: Twenty Ten
Theme URI: http://wordpress.org/
Description: The 2010 default theme for WordPress.
Author: the WordPress team
Version: 1.0
Tags: black, blue, white, two-columns, fixed-width, custom-header,
➥ custom-background, theme-options, threaded-comments, sticky-post,
➥ translation-ready, microformats, rtl-language-support, editor-style
*/
```

This tells WordPress what the theme is called, plus some basic information about the theme.

4 Choose File > Save As or press Ctrl+Shift+S/Shift+Cmd+S. In the Save As dialog box, navigate to the birds_phpcs5 folder, and save the file with the same name (style.css). When asked if you want to update links, click No.

5 Close the original style.css, and make sure you're working in the version in the birds_phpcs5 folder. The file path should be visible in the Browser Navigation toolbar.



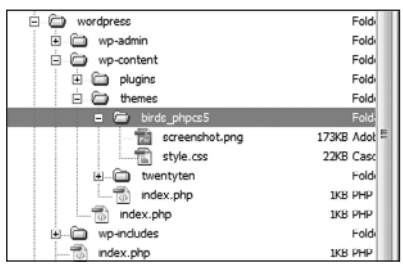
Alternatively, click the Open Documents icon  at the top of the Coding toolbar to reveal the file path.

6 The Theme Name comment must contain a unique name, which cannot consist only of numbers (that's why it's "Twenty Ten," not "2010"). When creating a child theme, you need to specify the parent theme as the child theme's template. Without these two changes, WordPress won't recognize your child theme. Changes to the remaining comments are optional. Amend the comments at the top of style.css like this:

```
/*
Theme Name: Birds of a Feather
Template: twentyten
Theme URI: http://foundationphp.com/phpcs5/
Description: Adaptation of the 2010 default theme for WordPress.
Author: David Powers, but mainly the WordPress team
Version: 1.0
Tags: purple, pink, blue, white, two-columns, fixed-width, custom-header,
➥ custom-background, theme-options, threaded-comments, sticky-post,
➥ translation-ready, microformats, rtl-language-support, editor-style
*/
```

The parent is identified by `Template` followed by a colon and its folder name.

7. Save style.css and close it.
8. This is sufficient for WordPress to identify the new theme, but it's a good idea to add an image to distinguish it from others in your Dashboard. The image should be about 300 pixels wide and must be called screenshot.png. Copy screenshot.png from lesson04/start/images to the birds_phpcs5 folder. The child theme folder should now contain two files.



You'll add more files later, but that's sufficient for now.

TIP: If you create a mockup of your final design in a graphics program, such as Fireworks or Photoshop, you can create screenshot.png by scaling the mockup and exporting it as a `.png` file. If it can't find screenshot.png, WordPress displays a text description of the theme.

Activating the child theme

The child theme needs to be activated before you can style it in Dreamweaver.

1. In your browser, log into the WordPress Dashboard. Depending on how you set up your testing environment, the URL should be one of the following:
   - **Virtual host.** http://phpcs5/wordpress/wp-admin/
   - **Localhost.** http://localhost/phpcs5/wordpress/wp-admin/

2 Expand the Appearance section in the column on the left of the Dashboard, and select Themes. The new Birds of a Feather theme should be displayed in the Available Themes section.

3 Click the Activate link for the Birds of a Feather theme. After a few seconds, Birds of a Feather is displayed as the Current Theme and Twenty Ten moves down to the Available Themes section.

4 Click the Widgets link in the Appearance menu to open the widget settings screen. The child theme inherits the default settings of the Twenty Ten theme. You can study all the options later. For now, remove the Recent Comments and Meta widgets from the Primary Widget Area by dragging them from the sidebar on the right back to the Available Widgets area.

5 Drag the Categories widget to move it just below the Search widget. The Primary Widget Area should now look like this:

6 Click the Background link in the Appearance menu. This allows you to set a background image and color for the site. However, it does this by generating a `<style>` block in the head of the page. It's better to use style.css to handle this, so leave this screen unchanged.

7 Click the Header link in the Appearance menu. This is one of the cleverest parts of the new Twenty Ten theme. It offers a choice of eight header images for your site. You can also upload your own images. The problem is that if your image isn't exactly the same size as used by Twenty Ten (940 × 198 pixels), you're prompted to crop it. WordPress doesn't crop your original, but instead makes a copy. If your image's height is less than 198 pixels, it's stretched. The result is often unsatisfactory.

The height and width of the header images are controlled by a custom function in the Twenty Ten theme. To change the default values, you need to override that function. Leave your browser open at the current page, and return to Dreamweaver.

8 Custom functions for WordPress themes are stored in a file called functions.php. If you attempt to redefine an existing function, PHP throws a fatal error, but WordPress overcomes this problem with a simple conditional statement. All the functions in the parent theme's functions.php file are wrapped in a conditional statement that checks whether a function of the same name has already been defined. If it hasn't, the parent theme defines the function. Otherwise, it uses the one defined by the child theme.

Choose File > New. Set Page Type to PHP, set Layout to <none>, and click Create. Switch to Code view, and delete all the HTML code inserted by Dreamweaver. You should have a completely blank page.

9 Add an opening PHP tag at the top of the new page, and save it as **functions.php** in the birds_phpcs5 folder.

10 Double-click functions.php in the twentyten folder to open it in the Document window. The file contains extensive comments that help you understand what the functions are for and how to override them.

11 Scroll down to around line 47 to locate the following code:

```
if ( ! isset( $content_width ) )
  $content_width = 640;
```

This defines the width of the main content `<div>` in the Twenty Ten theme. As you can see, the conditional statement sets the value to 640 (pixels) only if `$content_width` hasn't already been defined. So, to change the width to a different value, you need to add `$content_width` to functions.php in the child theme. Otherwise, this value is used.

12 The header image for Birds of a Feather is 20 pixels wider than the Twenty Ten images, so you can expand the content by the same amount.

Switch to the empty functions.php file you created for the child theme, and add the following code after the opening PHP tag:

```
<?php
$content_width = 660;
```

13 Switch back to the Twenty Ten version of functions.php, and scroll down to locate the following line of code (around line 53):

```
if ( ! function_exists( 'twentyten_setup' ) ):
```

This conditional statement checks whether a function called `twentyten_setup()` has already been defined. If it hasn't, it creates the function, which—as the name suggests—defines the default settings for the Twenty Ten theme.

NOTE: This conditional statement ends with a colon rather than an opening curly brace. This is an alternative syntax for control structures. See http://docs.php.net/manual/en/control-structures.alternative-syntax.php.

14 To create your own default settings for the child theme, you need to copy the function definition to the version of functions.php in birds_phpcs5. The function definition begins like this (around line 75):

```
function twentyten_setup() {

  // This theme styles the visual editor with editor-style.css to match
  ➥ the theme style.
  add_editor_style();
```

The final section of the function definition looks like this (around lines 171–178):

```
    'sunset' => array(
      'url' => '%s/images/headers/sunset.jpg',
      'thumbnail_url' => '%s/images/headers/sunset-thumbnail.jpg',
      /* translators: header image description */
      'description' => __( 'Sunset', 'twentyten' )
    )
  ) );
}
```

Select the entire function description, and copy it to your clipboard.

15 Paste the function definition into functions.php in the birds_phpcs5 folder. If you copied and pasted the code correctly, Dreamweaver should display "No syntax errors" in the Info Bar at the top of the Document window.

16 Scroll down to locate this code (around line 35):

```
define( 'HEADER_IMAGE', '%s/images/headers/path.jpg' );
```

This defines the default header image for the Twenty Ten theme (the tree-lined path). To display a different image, change the filename like this:

```
define( 'HEADER_IMAGE', '%s/images/headers/cormorants.jpg' );
```

You'll add this and other images to the relevant folder shortly.

17 The next section of code defines the width and height of the header image. Change the width from 940 to **960** and the height from 198 to **150** like this:

```
define( 'HEADER_IMAGE_WIDTH', apply_filters(
➥ 'twentyten_header_image_width', 960 ) );
define( 'HEADER_IMAGE_HEIGHT', apply_filters(
➥ 'twentyten_header_image_height', 150 ) );
```

18 About 20 lines farther down is a long section of code that begins like this:

```
register_default_headers( array(
  'berries' => array(
  'url' => '%s/images/headers/berries.jpg',
  'thumbnail_url' => '%s/images/headers/berries-thumbnail.jpg',
  /* translators: header image description */
  'description' => __( 'Berries', 'twentyten' )
  ),
```

This passes a multidimensional array to `register_default_headers()`, a function new to WordPress 3.0, which defines the choice of header images offered by the theme. The default Twenty Ten images are all 940 pixels wide and 198 pixels high, so they won't fit the child theme.

The media files for this lesson contain two header images called cormorants.jpg and seagulls.jpg, together with two smaller versions called cormorants-thumbnail.jpg and seagulls-thumbnail.jpg. Change all instances of berries in the multidimensional array to **cormorants**, and cherryblossom(s) to **seagulls**. There are only two header images, so you need to delete the other six subarrays. When you have finished, the final section of functions.php should look like this:

```
register_default_headers( array(
  'cormorants' => array(
    'url' => '%s/images/headers/cormorants.jpg',
```

```
      'thumbnail_url' => '%s/images/headers/cormorants-thumbnail.jpg',
      /* translators: header image description */
      'description' => __( 'Cormorants', 'twentyten' )
    ),
    'seagulls' => array(
      'url' => '%s/images/headers/seagulls.jpg',
      'thumbnail_url' => '%s/images/headers/seagulls-thumbnail.jpg',
      /* translators: header image description */
      'description' => __( 'Seagulls', 'twentyten' )
    )
  ) );
}
```

19 Save functions.php and copy cormorants.jpg, cormorants-thumbnail.jpg, seagulls.jpg, and seagulls-thumbnail.jpg from lesson04/start/images to twentyten/images/headers. The images must go in the parent theme's folder because that's where `register_default_headers()` expects to find them.

20 Return to the Header page in the WordPress administrative area. Click the Background link in the Appearances menu, and then click Header to reflect the changes you have made. The Custom Header section should now display the two Birds of a Feather header images, and the text in the Upload Image section should show the new default dimensions of 960 × 150 pixels.

If necessary, compare your code with lesson04/completed/functions.php.

Styling the child theme

All that remains now is to adjust the styles to give the theme its own look. Most of the remaining tasks are done in Live View and the CSS Styles panel.

1 Close functions.php if it's still open, and create a new folder called **images** in the birds_phpcs5 folder. Copy birds_bg_gradient.jpg from lesson04/start/images to the new folder. You'll use this later as a background image to the new theme.

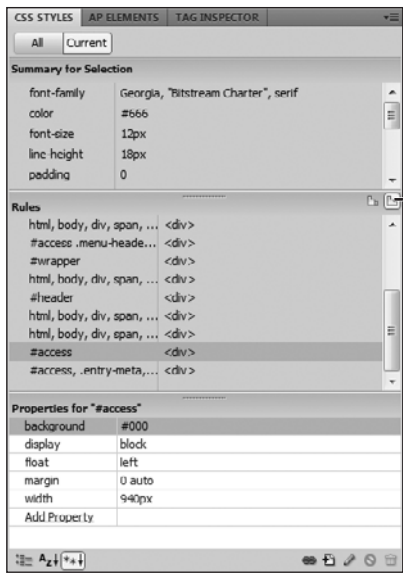
2 Double-click index.php in the main wordpress folder to open it in the Document window. Switch to Design view if necessary, and click Live View. The Birds of a Feather site should display with the new default header and the edited sidebar.

3 The header image is now wider than the menu bar. To fix that, click the Inspect button in the Document toolbar. As you move your pointer over Live View, you'll see different sections of the page highlighted. The content of an element is light blue or aqua, padding is mauve, and margins are yellow.

Notice that as you move from element to element, the currently highlighted element is also selected in the Tag selector at the bottom of the Document window. When your pointer is over the black menu bar below the header image, you should see `<div#access>` highlighted in the Tag selector. Click the menu bar to select it.

4 Selecting an element turns off the Inspect button, allowing you to move your pointer without highlighting other elements. Open the CSS Styles panel by clicking its tab or by choosing Window > CSS Styles. On Windows, you can also use the keyboard shortcut Shift+F11 (there is no Mac shortcut).

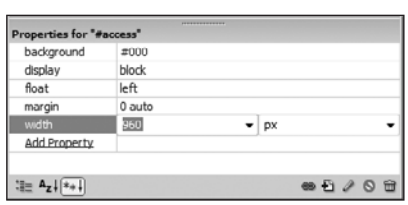
Make sure the Current button is selected at the top of the CSS Styles panel and that the Rules pane is visible in the middle section. If the middle section is titled About, click the Cascade icon as indicated in the following screen shot. You might need to close other panels and drag the panes inside the CSS Styles panel to see the rules and properties listed.



Click to display the cascade of rules for the selected tag

The Rules pane displays all the style rules that affect the current selection. Sometimes you need to examine several rules before finding the right one, but on this occasion, it should be the one selected by Dreamweaver. It's the `#access` rule shown in the preceding screen shot.

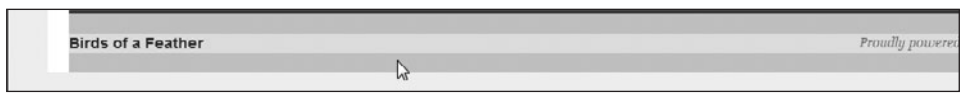
Select `#access` in the Rules pane. This reveals that the `width` property is set to `940px`. Click the value to edit it, and change the number to **960**. The px unit is controlled by a separate menu, so you don't need to change it.



As soon as you press Enter/Return to confirm the edit, the menu bar in Live View expands to match the width of the header image.

5 Although the header image and menu bar are now the same length, there's a gap of about 20 pixels of white space on the left of both elements. Finding the cause of the gap is a process of elimination, but if you look farther down the page, you'll see there's a similar gap on both sides of the horizontal line above the footer.

Click the Inspect button again, position the pointer over the footer so that the full width below the horizontal line is highlighted, and click to select it.



6 In the Rules pane of the CSS Styles panel, `#colophon` is selected. Examining the Properties pane reveals nothing to help eliminate the gap on the left and right, so start moving up the cascade of rules in the Rules pane. The next property begins with `#access .menu-header` and has a `width` property of `940px`. Change the number to **960** as you did in step 4.

As soon as you press Enter/Return, the white background expands to create the same gap on both sides of the header image, menu bar, and the horizontal line above the footer. This is progress, but the final design calls for the gap to be eliminated.

7 Click `<div.hfeed#wrapper>` in the Tag selector at the bottom of the Document window to reveal its properties in the CSS Styles panel. You'll see that the `padding` property is set to `0 20px`. This adds 20 pixels of padding to both sides of the wrapper `<div>`.

Remove the padding by selecting it in the Properties pane of the CSS Styles panel and clicking the trash can icon at the bottom right of the panel.

The left and right sides of the heading image, menu bar, and horizontal line above the footer are now flush with the white background of the wrapper.

8 There's a large gap between the white background and the top of the page. It's caused by the `margin-top` property, which is set to `20px`. Select `margin-top` in the Properties pane for `#wrapper` and click the trash can icon to delete it.

The entire contents of the page move up to eliminate the gap, leaving the white background flush with the top of the Document window.

9 With `#wrapper` still selected in the Rules pane of the CSS Styles panel, change the `background` property from `#fff` (white) to **#FAF2EF** (light pink).

TIP: Hexadecimal values for colors are case insensitive. The Twenty Ten style sheet uses a mixture of uppercase and lowercase, indicating that it's almost certainly the work of more than one person. Color values can also be shortened to three characters if each even character is the same as the preceding odd one. Thus, `#ffffff` can be shortened to `#fff`, but `#FAF2EF` cannot be shortened.

10 The next step is to change the background of the whole page. Begin by selecting `<body.home blog>` in the Tag selector. The Rules pane selects the `body, input, textarea` style rule, which covers too many elements, so start moving up the list of rules. The next one, `body`, defines the `background` property, which is the one you need to change.

The `background` shorthand CSS property is difficult to define directly in the Properties pane of the CSS Styles panel, so select the property and click the Edit Rule icon at the bottom right of the panel to open the CSS Rule Definition dialog box.

11 The CSS Rule Definition dialog box should automatically select the Background category.

Change the value of Background-color from `#f1f1f1` to **#E1DFE0**.

Click the Browse button next to the Background-image text box, navigate to the birds_phpcs5/images folder, and select birds_bg_gradient.jpg.

Set the Background-repeat menu to **repeat-x**.

When you click OK, the page background should change from light oatmeal to a vertical gradient that fades from light purple to a light gray.

12 Now comes a little bit of CSS magic—swapping the sidebar from right to left. The default style rule for the sidebar floats it to the right in a margin created by the main content. You can move the sidebar to the other side of the page by floating it left and giving it a large enough negative left margin to sit on the opposite side of the main content. But first, you need to adjust the margins of the main content.

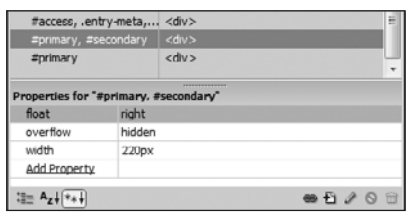
Click the Inspect button and select the main content on the left of the page. The style rule that controls its margins is applied to the container `<div>`, so click `<div#container>` in the Tag selector at the bottom of the Document window. The Properties pane of the CSS Styles panel reveals that its `margin` property is set to `0 -240px 0 0`. In other words,

a space of 240 pixels has been created on the right for the sidebar. You need to move the space to the opposite side. Change the margin property to **0 0 0 240px**. This moves the main content to the right of the page, but pushes the sidebar below it.

TIP: The Twenty Ten style sheet's use of a negative right margin on the container `<div>` is rather unconventional. It's needed because the `width` property of the `<div>` is set to `100%`. Using a negative value reduces the width and makes room for the sidebar. If you need to brush up on your knowledge of CSS, take a look at my book, *Getting StartED with CSS* (friends of ED, 2009, ISBN: 978-1-4302-2543-0).

13 Click the Inspect button again, and move your pointer until the whole of the sidebar is highlighted. Then select it. This selects `<ul.xoxo>` in the Tag selector. The rule that you want to edit is the next one up the page hierarchy. Click `<div.widget-area#primary>` in the Tag selector.

The Properties pane of the CSS Styles panel displays no styles, so move up the cascade in the Rules pane. The next one—`#primary, #secondary`—displays the rules that you need to change.



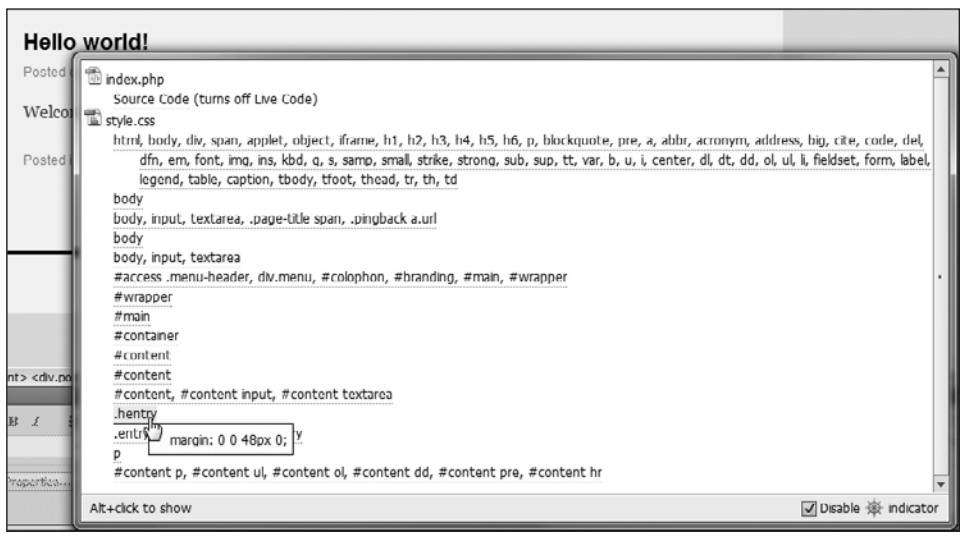
14 Change the value of `float` from right to **left**. The sidebar jumps to the left of the page but still below the main content.

15 Click Add Property, type **margin-left**, and set the value to **-1180px**. (That's *minus* 1180 pixels.) The sidebar is after the main content in the underlying HTML markup, but a combination of the left float and the large negative margin allows it to leapfrog over the main content and move into the correct position in the margin on the left.

16 Let's add a touch of CSS3 coolness to the main content. Click the Inspect button, move the pointer over the main content until it's highlighted, and click to select it. The Tag selector shows you have selected `<div.post-1 post type-post hentry category-uncategorized#post-1>`. Whew! That's a complex CSS selector. Fortunately, the Rules pane selects the lowest part of the cascade, the style rule for the `hentry` class, which currently sets only the `margin` property.

Although you could add the next set of CSS properties through the CSS Styles panel, it's a lot easier to work directly in the style sheet. The problem is that the style sheet contains more than 1,000 lines. A quick way to locate the correct rule is to use the Code Navigator.

Hold down the Alt key on Windows or Cmd+Opt on a Mac, and click anywhere in the "Welcome to WordPress" default post to invoke the Code Navigator, a context-sensitive tool for investigating styles that affect the area you clicked. Many rules affect this area, but you should find `.hentry` listed near the bottom of the panel, as shown in the following screen shot.



Click the `.hentry` link in the Code Navigator to open the style sheet in Split view. Using this technique positions the insertion point directly inside the `.hentry` style rule.

17 Amend the style rule like this:

```
.hentry {
  margin: 0 0 48px 0;
  padding:5px 5px 5px 15px;
  background-color: #FFF;
  -moz-border-radius: 20px;
  -webkit-border-radius: 20px;
  border-radius: 20px;
  -moz-box-shadow: 10px 10px 5px #888;
  -webkit-box-shadow:10px 10px 5px #888;
  box-shadow: 10px 10px 5px #888;
  border: 1px solid #eee;
}
```

The properties beginning with `-moz` and `-webkit` are browser-specific implementations of the CSS3 `border-radius` and `box-shadow` properties. Putting the properties in this order ensures that the effects will be maintained when the official properties are implemented by browsers.

If you refresh Live View by pressing F5 or clicking anywhere in Live View, you'll see the post now has a white background with rounded corners. In the Mac version of Dreamweaver, you'll also see a drop shadow as shown in the following screen shot.

NOTE: The Windows version of Live View fails to render the drop shadow correctly. However, if you view the finished WordPress site in a recent version of Safari, Google Chrome, or Firefox, the drop shadow is rendered correctly on both Windows and Mac OS X. Internet Explorer 8 and earlier ignores the CSS3 properties and just displays a rectangular white background.

18 Save style.css.

Hopefully, by now you've got the picture. You use the Inspect button or Code Navigator in Live View to identify elements on the page and inspect the style rules that govern their display. The Tag selector at the bottom of the Document window shows you where the element resides in the document hierarchy. Most CSS properties are inherited, so you often need to go back up the hierarchy to find the element where a specific rule has been applied. The CSS Styles panel in Current mode also allows you to work back up the cascade of rules affecting the selected element.

Styling a web page requires patience and skill. If you have a good command of CSS, you'll find working in Live View with these tools make styling WordPress, Joomla!, or Drupal very similar to working with a static HTML page. This lesson has concentrated on styling the front page of a WordPress site, but Live View is navigable in Dreamweaver CS5. Just hold down Ctrl/Cmd while clicking a link, and you can inspect and style all pages and views within a CMS.

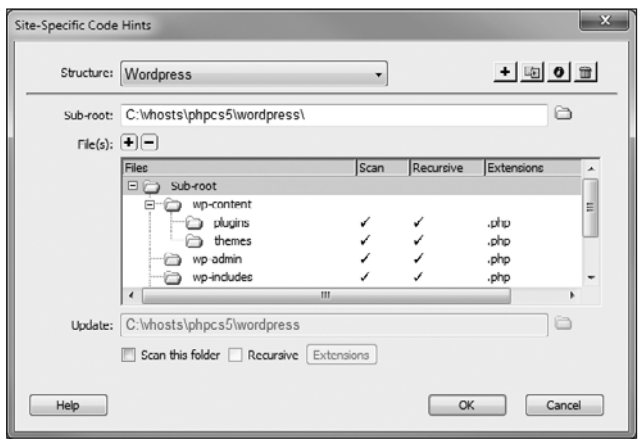
To round out this lesson, let's take a look at a WordPress template and make a slight change so that the text heading can be hidden from visual browsers. When editing CMS templates it's a good idea to set up site-specific code hints.

Enabling site-specific code hints for WordPress, Drupal, and Joomla!

WordPress, Drupal, and Joomla! use many custom functions to generate the content for each page. For example, WordPress provides the `bloginfo()` function (http://codex.wordpress.org/Template_Tags/bloginfo) to display information about your site. Dreamweaver uses code introspection to generate hints for your chosen CMS, speeding up editing page templates or functions.

The following instructions show how to enable site-specific code hints for WordPress in the phpcs5 site. The procedure is identical for Drupal and Joomla!

1. Make sure that the phpcs5 site is selected in the Files panel, and that the active document is from the same site or that all documents are closed.
2. Choose Site > Site-Specific Code Hints. As long as you haven't previously set up site-specific code hints for the same site, the Site-Specific Code Hints dialog box automatically recognizes not only which CMS is installed, but also the correct folder.



3. All that's necessary is to check that the Sub-root text field points to the folder that contains the top level of the CMS. If it doesn't, click the folder icon next to the Sub-root text field, and select the correct folder.
4. After checking the Sub-root text field, just click OK. That's all there is to it.
5. Dreamweaver inserts a file called dw_php_codehinting.config into the site root and automatically cloaks it to prevent it from being uploaded to your remote server when you use site synchronization to update your files. This file is used only in your local development environment.

Editing a WordPress page template

The first time you dive into a WordPress template can be a baffling experience, but the Twenty Ten theme is well commented. If you understand the HTML structure of the template you're working with, it's not too difficult to work out where to add custom features, such as a static paragraph, or remove elements that you don't want. As long as you make the changes to a template in a child theme, you can always delete the template and use the parent theme's default version if you make a mistake.

1. Open index.php in the top-level wordpress folder and click Live View to display the front page of the Birds of a Feather site.
2. Click Live Code to display in Split view the underlying HTML output generated by WordPress. Adjust Split view so you can see the code and "Just another WordPress site" at the top right of the page (you might need to scroll horizontally in Live View).
3. Click between the words "Just" and "another" in Live View. Depending on the size of your monitor, this should scroll Code view so that the equivalent HTML output is in the center with the insertion point between the two words, as shown on line 24 in the following screen shot:

```
18              <div id="branding" role="banner">
19                                  <h1 id="site-title">
20                      <span>
21                          <a href="http://phpcs5/wordpress/" title="Birds of a
Feather" rel="home">Birds of a Feather</a>
22                      </span>
23                  </h1>
24                  <div id="site-description">Just another WordPress site</div>
25
26                                      <img src=
"http://phpcs5/wordpress/wp-content/themes/twentyten/images/headers/cormorants.
jpg" width="960" height="150" alt="">
27                                  </div><!-- #branding -->
```

You can't edit the output in Live Code, because it's dynamically generated by WordPress, but inspecting the output here gives you a good idea of what to look for when you open the WordPress template.

The text you clicked is in a `<div>` that has the ID `site-description`. You can also see on line 19 that the main heading has the ID `site-title`, and that the text is wrapped in a link that returns to the front page of the WordPress section of the site.

The final point to notice is that the `alt` attribute of the header image on line 26 is empty. The HTML specification requires all images to have alternative text (in the `alt` attribute), but it's recommended to use an empty string when the image is purely decorative. This redesign incorporates text into the image, so the same text should be inserted into the `alt` attribute in case the image is not displayed for any reason.

4 In the Files panel, double-click twentyten/header.php to open it in the Document window. Save the file as **header.php** in the birds_phpcs5 folder, and click No when asked if you want to update the links. Close the original version of header.php. You want to edit the copy in the child theme.

5 Scroll down to locate the following (it should be around line 54):

```
<div id="site-description"><?php bloginfo( 'description' ); ?></div>
```

This uses the WordPress `bloginfo()` function to display the site description ("Just another WordPress site").

6 Delete the entire line, and save header.php.

7 Switch back to index.php, and press F5 or click the Refresh icon in the Document toolbar (not the Files panel) to update Live View. After a few moments, the text disappears from the top right of the page, and the corresponding code is removed from Live Code.

8 Switch back to header.php, and examine the code immediately above the line that you removed in step 6. It looks like this:

```
<?php $heading_tag = ( is_home() || is_front_page() ) ? 'h1' : 'div'; ?>
  <<?php echo $heading_tag; ?> id="site-title">
    <span>
      <a href="<?php echo home_url( '/' ); ?>" title="<?php echo esc_attr(
      ➥ get_bloginfo( 'name', 'display' ) ); ?>" rel="home"><?php
      ➥ bloginfo( 'name' ); ?></a>
    </span>
</<?php echo $heading_tag; ?>>
```

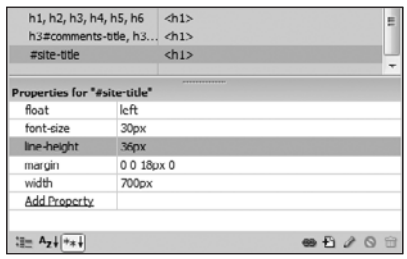
This code checks whether the current page is the site root or the front page of the WordPress section. If it is one of these, `$heading_tag` creates a pair of `<h1>` tags. Otherwise, it creates a `<div>`. It then creates a link to the front page of the WordPress site, which is wrapped around the site name.

9 The site name is already in the header image, so you don't want it displayed twice. However, you should leave it in the underlying code for search engines and screen readers for the blind. Edit the code like this to remove the link and `<span>` tags:

```
<?php $heading_tag = ( is_home() || is_front_page() ) ? 'h1' : 'div'; ?>
  <<?php echo $heading_tag; ?> id="site-title">
    <?php bloginfo( 'name' ); ?>
</<?php echo $heading_tag; ?>>
```

10 Save header.php, and refresh Live View in index.php. The text heading is still there, but it's no longer a link.

11 Click the Inspect button, select the Birds of a Feather text in Live View, and open the CSS Styles panel in Current mode. In the Rules pane, `#site-title` should be automatically selected. You know this is the style rule you want to change, because it's the ID selector for the element you just edited in header.php.



12 Select each of the properties in turn and click the trash can icon at the bottom right of the CSS Styles panel to delete them. Then click Add Property to add the `position` property, and set it to `absolute`.

Click Add Property again to add the `top` property, and set it to `-1000px` (minus 1000). The heading disappears from visual browsers but remains accessible to search engines and screen readers.

13 There's still a large gap at the top of the page, so click the Inspect button again, and select the area above the header image.

The `#header` rule is selected in the CSS Styles panel. Delete the `padding` property to remove the gap.

14 Click the header image in Live View to select the `#branding img` style rule, and delete the `border-top` property to remove the thick black border at the top of the page.

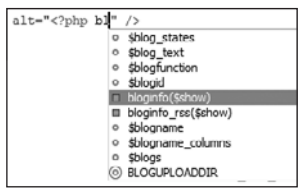
Save style.css to preserve the changes.

15 Return to header.php, and locate the following code, which inserts the header image (around line 63):

```
<img src="<?php header_image(); ?>" width="<?php echo HEADER_IMAGE_WIDTH;
➥ ?>" height="<?php echo HEADER_IMAGE_HEIGHT; ?>" alt="" />
```

You could hard code the value of the `alt` attribute, but that would mean changing the template if you decide to use the same theme for different sites. It makes more sense to use the WordPress API.

Position the insertion point between the quotation marks of the `alt` attribute, type an opening PHP tag followed by a space, and then press Ctrl+spacebar to bring up code hints. Type **bl**. Dreamweaver should select `bloginfo()`.



Press Enter/Return to autocomplete the function name. Dreamweaver automatically inserts the opening parenthesis. Complete the code by typing **'name'** followed by a closing parenthesis, semicolon, and closing PHP tag. The `alt` attribute should look like this:

```
alt="<?php bloginfo('name'); ?>"
```

16 Save header.php, and refresh Live View in index.php. When Live Code reloads, you should see "Birds of a Feather" in the `alt` attribute.

You can check your code against style.css and header.php in lesson04/completed.

What You Have Learned

In this lesson, you have:

- Examined the basic structure of Drupal, Joomla!, and WordPress (page 112)
- Installed WordPress 3.0 in your local testing environment (pages 113–119)
- Created a child theme based on the default WordPress Twenty Ten theme (pages 119–127)
- Used Live View, the CSS Styles panel, and Code Navigator to style WordPress (pages 128–134)
- Enabled site-specific code hints for WordPress (page 135)
- Edited a WordPress template (pages 136–139)

Index

D

E

F

G

H

I

N

O

Q

R

S

T

U

V

W

X

Y

Z