



Design to Theme in Five

By Emma Jane Hogbin
emma@designtotheme.com
www.designtotheme.com



Design to Theme in Five

© 2010 by Emma Jane Hogbin

440-13th Street East

Owen Sound, Ontario

N4K 1W6

www.designtotheme.com

License

Creative Commons: Attribution-Noncommercial-No Derivative Works 2.5 Canada

Credits

Portions of this ebook are adapted from *Front End Drupal* by Emma Jane Hogbin and Konstantin Kaefer. They are included here by permission of Pearson Education.

Table of Contents

Introduction.....	4
Skills Summary.....	4
Page Elements and HTML Frameworks	5
Design Tools.....	5
Creating Designs.....	5
Designing for Drupal.....	5
Task Summary.....	6
A Basic Drupal Page	7
Define Your Theme.....	7
Template Variables.....	7
Theme Regions.....	9
Content.....	9
Adding Style Sheets.....	10
Upload and Test Your Theme.....	10
Task Summary.....	10
Inserting Navigation.....	11
Conditional Output.....	12
Base (or Starter) themes.....	13
Evaluating starter themes.....	13
Zen Theme.....	14
960 Grid System.....	15
Task summary.....	15
Template Files	16
The Template File node.tpl.php.....	16
Node Template Variables.....	17
Displaying Content with More Precision.....	18
Accessing Content in the \$node Object.....	19
Additional Template Files.....	20
Task Summary.....	21
Customizing Blocks.....	22
Theming Blocks.....	22
Collapsible Regions.....	23
Sharing Your Theme.....	24
Licensing.....	24
Selling Themes.....	24
Contributing Themes to the Drupal Project.....	24
Summary.....	26
Appendix A: Designing with Grids	27
Grid Resources.....	27
Task Summary.....	27
Appendix B: node-workshop.tpl.php file.....	29

INTRODUCTION

In this five-part workbook you will learn how to transform a Photoshop file (or Illustrator or GIMP) into a Drupal theme. No more relying on programmers. You're now in control. Whether you want to build and sell your own designs, or you're a newly hired designer at a Drupal Web development shop, this course will give you the confidence to transform your imagination into a working Web site.

Skills Summary

The goal of this workbook is to enable you to:

1. Build designs optimized for transformation to Drupal themes using common design software.
2. Convert Photoshop files into basic Web templates using a text editor.
3. Create and apply a new Drupal theme to a Web site using a text editor and simple Web tools such as FTP.
4. Evaluate common base themes and know when to choose between several popular base themes.
5. Create a new Drupal theme by extending a base theme.
6. Develop common template files (tpl.php) necessary to theme pages and nodes using a text editor.

Page Elements and HTML Frameworks

Over the course of the next five weeks you'll be converting a design file to a Drupal theme. You may already have a design file ready to convert, or you may be starting from scratch. It doesn't matter which design tool you're using for this course.

Design Tools

When I use the word "Photoshop" you can safely substitute this word for any tool you're using to design your site. I don't actually use Photoshop for my work. I use a free equivalent called GIMP for raster drawings and Inkscape for vector drawings. (Walter the cardinal was made in Inkscape.)

Creating Designs

For this course I'm going to assume that you're comfortable with Web design software. If you are a little bit new to Web design as well you can find some good design tutorials online by searching for "PSD webdesign tutorial." A few of the ones I've book marked include:

- foliofocus <http://foliofocus.com/category/psd/>
- graphicriver <http://blog.graphicriver.net/web-design/30-best-psd-website-templates-on>
- grafpedia <http://www.grafpedia.com/tutorials/create-business-wordpress-psd-layout>

With your design skills (re)freshed, it's time to take a look at Designing for Drupal!

Designing for Drupal

Designers that I've worked with in the past have typically run into problems when they try to design a Web site without a clear understanding of how Drupal is going to assemble all of the pieces. Unlike a static HTML page, Drupal has several puzzle pieces that fit together to form a single page. If you haven't read *Front End Drupal* please read the first page of ten of the online version of Chapter 4: The Drupal page (<http://www.informit.com/articles/article.aspx?p=1336146>). You may read to the end of this chapter if you like, but it gets a little more complicated than what you need for this week's assignment.

In Chapter 4 you learn that there are several elements, or templates, that make up a single page. To be successful at creating Drupal themes you need to be able to think in terms of the entire page but also in terms of each of the templates that make up that page. If you already have a design file take a close look at it and see if you can find repeating elements that will become individual template files.

For today's second task find in your design template the components that will make up: blocks, nodes, header region, footer region and the rest of the page. You may have more unique elements than this. Where possible group "related" items into a single pattern. For example: do your sidebar "blocks" all have the same size fonts for the headings? If you are working with a design program that supports layers, group design elements that will be in the same template files to help

organize your thoughts in preparation for the transition to HTML.

Once I have my template organized I generally start building an HTML file while looking at my design file. By using design grids I can quickly put together a shell for my site. Does your site have Four columns? Two columns with sub-columns? Once the general shape is in place I start filling in details by extracting visual elements from my design file.

This is different from how I built sites before Drupal. Before Drupal I would slice an entire design and use the images to bump the HTML structure around so that everything lined up. I no longer use this method because I find that Drupal presents too many layers to easily bump images together. You may want to think of building your HTML file as starting with an HTML wireframe and then adding more and more layers to the design until it is complete. If you've never worked this way before, be brave and give it a shot for this course. You may find that you hate it, but at least you'll be speaking from experience.

Once you have the wireframe for your theme you need to start thinking about each of the page elements you identified and grouped in your design. This is where theming gets a little bit conceptual. If you create a whole HTML version of your design file you will need to extract elements for each of the template files. For this week: go ahead and create a whole HTML page with all of the elements from your design...AND leave yourself comments in the HTML to say where each element begins and ends. For example: `<!-- HTML GOES HERE -->`.

Task Summary

1. Read Chapter 4 of *Front End Drupal* (<http://www.informit.com/articles/article.aspx?p=1336146>)
2. Find page elements in your design templates. Organize design elements into groups according to the template files they will be appearing in (e.g. nodes and blocks).
3. Create an HTML wireframe for your new theme.
4. To your HTML wireframe, fill in each of your page elements. Don't forget to add HTML comments to show you where they begin and end.
5. Bonus Task: Read Appendix A on design grids and complete the tasks outlined in it.

A Basic DRUPAL Page

In the first section of this guide you created a simple HTML mockup for your design. Now you're going to follow a few basic steps and convert that mockup into a theme! I'll give you step-by-step instructions to ensure things go smoothly.

At this point you cannot proceed if you don't have a single HTML page which holds all page elements from your design. Have you completed each of the tasks from the previous section?

Define Your Theme

Your theme will need a container for all of the associated files you are about to create. Create a new folder using only lowercase letters and numbers (no spaces!). For example: you could name your theme: bluebird or sk8tergrll.

In this folder create a new text file named `YOURTHEME.info`. For example: `bluebird.info` or `sk8tergrll.info`. This info file is a programmerly glossary which tells Drupal about your theme. In it you need to include the following text:

```
name = Your Theme Name
description = A description which will appear in the theme selector.
core = 6.x
engine = phptemplate
```

To find out more about this info file, please read: the drupal page (<http://drupal.org/node/171205>).

With the folder you just created and an info file you now have a theme! It will use all of the default templates provided by core. It won't even look like Garland...it'll just be plain, plain, plain!

Template Variables

Each theme has template files which allow you to create the HTML structure that you want for your page. Complete the following steps to get Drupal to use your HTML wire frame. This is the last time that you'll be able to look at your HTML page without also running it through Drupal.

Copy your HTML page into the theme directory and rename it: `page.tpl.php`.

Open your renamed HTML file and replace static content with the appropriate variable names for `page.tpl.php`. Each variable must be added to the template as follows:

```
<?php
  print $VARIABLE_NAME;
?>
```

Head Variables

Start with the following variables inside the HTML tag `<head>`:

- `$head_title` A modified version of the page title, for use in the TITLE tag.
- `$head` Markup for the HEAD section (including meta tags, keyword tags, and so on).
- `$styles` Style tags necessary to import all CSS files for the page.
- `$scripts` Script tags necessary to load the JavaScript files and settings for the page.

For example the `$head_title` HTML snippet would be as follows:

```
<title><?php print $head_title; ?></title>
```

The default `page.tpl.php` file can be viewed online at <http://api.drupal.org/api/drupal/modules--system--page.tpl.php/6/source>

Body Variables

In the body of your page add HTML divs with a unique id for the following body variables:

- `$logo` use this only if you want to upload your logo via the theme's administration section
- `$search_box` only displays if you've enabled search in your theme's administration
- `$help` Dynamic help text, mostly for admin pages.
- `$messages` HTML for status and error messages. Should be displayed prominently.

Another really handy variable that Drupal creates is the variable `$body_classes`. It contains CSS classes which give each page on your Web site a unique ID. You can take advantage of these to change background images and headers from your style sheets. To add this to your template use the following snippet:

```
<body class="<?php print $body_classes; ?>">
```

The sample output from the front page and the user profile page is as follows:

```
<body class="front logged-in page-node one-sidebar sidebar-right">  
<body class="not-front logged-in page-user one-sidebar sidebar-right">
```

Additional page template variables are listed at: <http://api.drupal.org/api/drupal/modules--system--page.tpl.php/6>

Closure

A special variable is needed at the bottom of your page. This variable is used by modules such as Google Analytics. To add this variable put the following snippet just before the end body tag on your theme.

```
<?php
  print $closure;
?>
```

Theme Regions

There are three regions available to you by default: left, right and footer. To enable these regions in your theme you must print them using the same technique described above. For example:

```
<?php
  print $left;
?>
```

If your HTML design that has wrapper divs that are used OUTSIDE of the region, add them now. For example you may have `<div id='left'><?php print $left; ?></div>`. If you have HTML elements that go INSIDE the region (i.e. the headings for each block) this will need to be moved to the template file `block.tpl.php`. Usually it is sufficient to use CSS to change the way a block will look though. Think hard about whether you really need custom HTML inside your blocks. Themes that rely on the default HTML are a LOT easier to maintain.

If your design has more regions than this that you would like to place blocks into, you will need to let Drupal know about them. In a text editor open up your theme's info file and add your region definitions to the bottom of the file using the structure:

```
regions[your_new_region_name] = Name of region listed on the blocks page
```

For example:

```
regions[left_one] = Left sidebar, column 1
regions[left_two] = Left sidebar, column 2
regions[right] = Right sidebar
regions[footer] = Footer
regions[header] = Header
```

For more information about regions, please read <http://drupal.org/node/171224>.

Content

Finally we need to add the variables for placing content onto your page. This may be a list of content (e.g. the front page of your site) or a single node of content. To make this work, you will need to add the following variables to your template file:

- `$title` the content title (i.e. the stuff that appears at the top of a node)
- `$tabs` Tabs linking to any sub-pages beneath the current page (e.g., the view and edit tabs when displaying a node).

- `$content` The main content of the current Drupal page. Later you will learn how to break `$content` into its component parts using a `node.tpl.php` file.

Adding Style Sheets

If you already have the CSS files ready for your design, great! You can include all relevant CSS and JavaScript files automatically by registering the files in your theme's `.info` file. Open up your text editor and add the following to the bottom of your file:

```
; Include the following CSS files automatically
stylesheets[all][] = my_theme_styles.css

; Include the following javascript files automatically
scripts[] = myscript.js
```

You may add as many lines as you have CSS and Javascript files. For more information about adding CSS files please read <http://drupal.org/node/171209>. For more information about adding JS files, please read <http://drupal.org/node/171213>.

Upload and Test Your Theme

It's now time to enable your theme! Go ahead and upload this entire folder to your Drupal site. It should be placed into the sites directory. If you want it to be available to all sites installed on this server upload it to: `sites/all/themes/bluebird` or `sites/all/themes/sk8tergrrl` (as appropriate). If you want it to be accessible to a single domain upload the theme folder to the following location: `sites/domainname.com/themes/MY_THEME_NAME`.

You're now ready to test your theme for the first time and see how it works!

Fix the markup so that you have enough containers for each element that you want to style. Take a look at the mark that Drupal has provided you for blocks and content areas. Can you see how you would adjust your theme's style using only CSS? Making adjustments using only style sheets will result in a theme that is a lot easier to maintain over time.

Task Summary

1. Create a theme folder and an info file for your theme.
2. Add variables to your HTML template to make it into a `page.tpl.php` Drupal theme.
3. Add variables to your `page.tpl.php` for regions, define new regions if necessary.
4. Add the variables for content display.
5. Add the closure variable.
6. If you have style sheets, add them now. If you don't have CSS ready, go ahead and create the CSS you need based on the markup you have. Use the "live" theme to see what selectors you need to use to style each element.

INSERTING NAVIGATION

The previous section had a lot of variables to add to your page template. Two of the variables we didn't add were for primary and secondary navigation. These variables are actually special kinds of variables called arrays that need some extra work before they can be printed to the page. An array is a bit like a filing cabinet with multiple drawers of information instead of a simple file folder. The primary and secondary links need to be run through a special Drupal theming function which converts the filing cabinet of information into a series of file folders. At this time you may also add a CSS class to your navigation.

This is how the default page template correctly inserts primary links:

```
<?php
print theme('links', $primary_links,
  array('class' => 'links primary-links'));
?>
```

The CSS classes that are added are “links” and “primary-links.” To change the CSS classes added alter the text that appears in bold in the previous code example.

To print the secondary links, copy the code above and replace the two instances of the word “primary” with the word “secondary.”

CONDITIONAL OUTPUT

Sometimes you want HTML to appear, and sometimes (especially if a region has no content) you don't. In the core page template file you will see a number of "if" statements. Please take a look at the page as you read this next bit: (<http://api.drupal.org/api/drupal/modules--system--page.tpl.php/6/source>)

Here is some sample output from that file:

```
<?php if (!empty($left)): ?>
  <div id="sidebar-left" class="column sidebar">
    <?php print $left; ?>
  </div> <!-- /sidebar-left -->
<?php endif; ?>
```

The first line reads as follows: if the left region is NOT empty, print the following HTML.

You may have also seen a second way to add conditional statements to your template files:

```
<?php
  if (!empty($left)) {
    // do stuff here
  }
?>
```

This uses { brackets } to open and close the conditional statement. In this second example I have also not switched back and forth between HTML and PHP words. The entire snippet is PHP. Look for <?php open and ?> close statements in both examples and find the lines which are HTML output and PHP output.

Using the `page.tpl.php` file as an example, add at least two conditional statements to your own page template.

Base (or Starter) Themes

You do not need to rely exclusively on your imagination to create both the visual design and the code for a truly excellent theme. The default theme, Garland, has received a good scrub and can now be used as inspiration for your new theme. Several other starter kits are available as well. These kits, which typically have very little style, are intended to serve as a launching point for building your own unique Drupal theme. Each kit uses a CSS, table-free layout and has been tested in a variety of browsers. The documentation explaining how to implement each starter kit ranges from poor to excellent. Following are the sources for some popular starter kits:

- Zen <http://www.drupal.org/project/zen>
- 960.gs <http://drupal.org/project/ninesixty>
- Genesis <http://www.drupal.org/project/genesis>
- ATCK <http://www.drupal.org/project/atck>
- Basic <http://www.drupal.org/project/basic>
- Beginning <http://www.drupal.org/project/beginning>
- Blueprint <http://www.drupal.org/project/Blueprint>
- Clean <http://www.drupal.org/project/clean>
- Flexible 2 <http://www.drupal.org/project/flexible>
- Foundation <http://www.drupal.org/project/foundation>
- Framework <http://www.drupal.org/project/framework>
- Hunchbaque <http://www.drupal.org/project/hunchbaque>
- Tendu <http://www.drupal.org/project/tendu>

(And there are probably more too.)

Among the many advantages of using a base theme is that there are many short cuts prepared for you. For example: the Zen Theme converts the array for primary links into a variable that can be printed immediately.

Base themes can be applied at any time by updating your info file with the following snippet:

```
base theme = themename
```

Replace “`themename`” with the folder name of the base theme. Your theme does not need to be a sub-folder of its parent, but it does need to be available within your Drupal installation.

Evaluating starter themes

There are a lot of starter themes! If you are already using a specific CSS grid framework you may be immediately drawn to one of the starter themes listed above. I encourage you to download them all and look at the documentation to see which may be the most appropriate for you. Questions you may want to ask yourself include:

- does this theme support simplified layouts for print and/or mobile?
- how is the documentation for this theme?

- what is the developer community like? how often are updates made? how many unsolved issues are in the queue?
- does the theme use a CSS framework that I know about and/or already use and like?

I typically use one of three different base themes: Zen, 960.gs and BeginningW2. I use Zen when I have a completely custom design; 960.gs when I have a unique design that fits easily into a grid; and BeginningW2 when I just want to put together a quick Web site that doesn't look horrid (the main Design to Theme Web site uses BeginningW2).

Zen Theme

Touted as being “the ultimate starting theme for Drupal,” the Zen starter kit provides an example of how to implement all of the basics. Its markup is clean and extensively documented. Whether you use it as a reference guide, pulling out the elements you need, or use the actual files as a base for your own theme, Zen will help you to produce cleaner themes, faster. Even a simple design is worthy of beginning with a starter kit like Zen. Although it is a simple template The Memory Garden Retreats (www.memorygardenretreats.com) Web site was built using Zen. Going from a graphic file in GiMP to a deployed theme took less than a day—all because of Zen. The headaches of figuring out cross-browser, CSS-based design were completely eliminated.

You can download the Zen theme from drupal.org (<http://www.drupal.org/project/zen>). Unpackage the files and place the entire directory (and subdirectories) into the themes folder. The Zen theme will help you to identify each area of a Drupal installation that can be customized and themed.

The steps outlined below assume that you are creating a new theme. Go ahead and complete these steps. You may decide to use Zen as the base for your theme (or perhaps future themes); however, at this point start with the only the design that Zen provides and see what this theme looks like unstyled.

1. Create your own Zen subtheme by making a copy of the STARTERKIT folder with your theme name.
2. Rename the folder to your new theme name; rename the `.info` file contained in the copied STARTERKIT folder to the same name as the folder it is contained in. For example, if you named your directory blackbird, the corresponding `.info` file would be named `blackbird.info`. Adjust the contents of the info file according to the steps in last week's lessons.
3. Copy the CSS file `zen.css` from the zen folder into your theme's folder.
4. Copy the liquid, or fixed-width, CSS file from the main zen folder into your theme's folder.
5. Copy the print CSS file from the main zen folder into your theme's folder.
6. Edit the `template.php` and `theme-settings.php` files and change all instances of STARTERKIT to your theme's name (“blackbird,” in this example).
7. Enable your new theme in Drupal's administrative Web interface by navigating to Administer, Site building, Themes.
8. Continue to adjust each of your new theme's settings and styles to suit your needs. In addition to the Zen project, a series of default templates are available for many different Drupal components. Most of these files correspond to specific modules and can be copied directly from your Drupal installation. For example, three template files are available within the modules/comment folder: `comment-folded.tpl.php`, `comment-wrapper.tpl.php`, and

`comment.tpl.php`.

The online documentation for Zen is comprehensive. To learn more about it, please read the relevant pages (<http://drupal.org/node/193318>).

You may find the Zen Theme Garden (<http://groups.drupal.org/taxonomy/term/5171>) and Zen sub-theme page (<http://drupal.org/node/340837>) to be of particular interest.

960 Grid System

If you haven't tried the 960.gs before, now is the time. Complete the following steps:

1. Download the Drupal ninesixty theme here (<http://drupal.org/project/ninesixty>) and unpack it.
2. Read the README.txt file. There are even more tutorials listed in this file on working with grid systems.
3. Create a new theme and list its base theme as ninesixty.

That's it! There is no need to copy any files from the ninesixty folder as you did for the Zen theme. All of the CSS classes which enable 960 layouts can now be used in your theme as well. If your HTML template is based on the 960.gs you can add it as your base theme and remove all of the 960-specific CSS grid files from your theme's directory.

Task summary

1. Add primary and secondary links to your page template.
2. Compare the two examples of conditional statements and find the lines written in PHP and HTML..
3. Add your own conditional statements to your page template.
4. Create a new theme using Zen as your starter theme.
5. Create a new theme using 960.gs as your starter theme.
6. Find a starter theme that makes sense for your projects. Apply it to your theme and adjust CSS classes and variables as necessary to take advantage of your base theme.
7. Download the sample sub-themes attached to this page. Examine their contents and copy and relevant CSS or tpl.php files to your own theme.

Template Files

In the previous section you cleaned up your template file by adding some new variables and choosing a base theme. Depending on the base theme you may have added a CSS grid framework (960.gs) or a series of helper functions (Zen theme). This section does not technically build on the content from the previous section; however, I do recommend completing each of the activities before proceeding.

The Template File `node.tpl.php`

The node template controls how each unit of content is displayed within the larger page template. By default, this includes everything between the editing “tabs” of the content down to (but not including) the orange RSS feed icon.

To create a custom node template, you must create a new file named `node.tpl.php` in your theme directory.

Compared to a page template with a full HTML framework for multiple regions and headers, the default node template contains very little markup. The default node template is shown in this section. It includes the default CSS classes that can be used to provide sophisticated and customized context-sensitive designs. Two items of note: If the node being displayed is in “teaser” mode, the node title links to the full page of content. The second item to note is that the page template is also configured to display a title. For example, when the Blog module is enabled, an additional title, “Blogs,” appears on `example.com/blog`. The second title is part of the page template and is set by the module Blog. Note: The editing tabs for a node are actually configured within the page template with the variable `$tabs`.

```
<div id="node-<?php print $node->nid; ?>" class="node
<?php if ($sticky) { print ' sticky'; } ?>
  <?php if (!$status) { print ' node-unpublished'; } ?> clear-block">
<?php print $picture ?>
<?php if (!$page) { ?>
  <h2><a href="<?php print $node_url ?>" title="
    <?php print $title ?>"><?php print $title ?></a></h2>
<?php } ?>
  <div class="meta">
    <?php if ($submitted) { ?>
      <span class="submitted"><?php print $submitted ?></span>
    <?php } ?>
    <?php if ($terms) { ?>
      <div class="terms terms-inline"><?php print $terms ?></div>
    <?php } ?>
  </div>
  <div class="content">
    <?php print $content ?>
  </div>
  <?php print $links; ?>
</div>
```

The node module has, by default, two types of node templates: `node.tpl.php`, for all nodes that do not have a more specific template, and `node-contenttypename.tpl.php`, for each type of content. The template file used to theme the workshop registration pages on Design to Theme is available in Appendix B.

Node Template Variables

You may print any of the following variables into your node template file:

- `$content` Node body or teaser depending on the contents of the variable
- `$teaser` This variable contains all of the display information for the node stuffed into a single variable. Later in this lesson you will learn how to use the `$node` variable to pick out individual fields for display.
- `$terms` A list of themed links for each of the relevant Drupal categories for this content.
- `$links` A list of themed links related to modules other than taxonomy. The links may include “Read more” (which is displayed on the teaser) and “Add new comment” (which is displayed based on the comment settings).
- `$node_url` The URL of the current node. This link is useful for “permalinks” to the page. It is commonly used to link the date a blog entry was created to the full node.
- `$submitted` The full themed submission information (for example, “Submitted by Wiarion Willie on 2 February 2009—9:46 am”). This information can be broken into the variable `$date` and the user data (see the last point in this list).
- `$date` The formatted creation date. This variable uses the Drupal “short” date setting. You can configure formatting of the date by navigating to Administer -> Site configuration -> Date and time. To display a custom-formatted date, you may use the variable `$created` and the Drupal function `format_date()`.
- `$picture` (the author’s profile photo), `$name` (username of node author), and `$uid` (the author’s ID).

The variable `$title` is available in the node template as well and should be output for lists of nodes; however, you may need to adjust your templates to distinguish between the title displayed in the page template and the title displayed in the node template (especially on pages that display lists of content).

In addition to the variables that you print to the page and are visible to the Web site visitor, many other variables contain information about the node that may help you to format the page appropriately. These variables can be classified into several categories.

Additional Information about the Content.

These variables contain information about the node that is being displayed. They include variables that are relevant to a single node display page as well as variables that are relevant to a page containing a list of teasers to several different nodes.

- `$type` The content type of the node (for example, story, page, blog, and custom content types).
- `$teaser` The variable that announces Drupal is requesting the “teaser” view for the content. When the page is displayed in full, the related variable `$page` returns true.
- `$readmore` If the teaser content of the node does not contain the full content, a “read more”

link is displayed.

- `$zebra` Displays either “even” or “odd”; to be used for zebra striping in teaser listings.
- `$id` Position of the node within a list of nodes (for example, on the front page). This variable is incremented each time a node is output.
- `$node` The full array containing all data for the node. This variable may contain “unsafe” data and should be used with caution.

Information about the Site Visitor

These variables are related not to the content, but rather to the visitor who is viewing the content. They can be useful when you are customizing the options for authenticated users and administrators.

- `$logged_in` Returns true when the current user is a logged-in member.
- `$is_admin` Returns true when the current user is an administrator.

Information about the Comments Related to This Node

These variables are related to the comments for this node, including whether comments are enabled for this node.

- `$comment` Indicates whether comments are enabled for this node.
- `$comment_count` Number of comments for this node.

Status of This Content

These variables are related to the “Publishing options” for a given node.

- `$promote` Identifies whether this node should be displayed on the front page. (`$is_front` allows you to check whether the page being displayed is the front page.)
- `$sticky` Identifies whether this node ought to be displayed at the top of lists of content.
- `$status` Identifies whether this page is “published.”

A full list of these variables is available here (<http://api.drupal.org/api/file/modules/node/node.tpl.php/6>). You may also refer to the default node template file within your own Drupal system files. The default template can be found within your Drupal core files at `modules/node/node.tpl.php`.

Displaying Content with More Precision

The variable `$content` contains both the field labels and the content values for all display information for each content type. The variable `$content` comes prepackaged, so you cannot use it if you want to insert markup around individual content fields or add explanatory text between fields. To accomplish these kinds of customizations, you must use the variable `$node` instead. The variable `$node` is an object containing all node-related data for the node you are viewing. It includes everything from the node ID to the categories for the node. By carefully selecting the right part of the variable `$node`, you may use any of this information in your template.

Deciphering the Object \$node

There are two ways to view the contents of `$node`. If you have the Devel module installed and enabled, you may use the function `dsm()` to create an easy-to-read display of the information. In

your node template file (`node.tpl.php`), add the following snippet:

```
<?php
  dsm($node);
?>
```

If you do not have the Devel module enabled, you may use the PHP function `print_r` instead with the following snippet:

```
<?php
  print "<pre>";
  print_r($node);
  print "</pre>";
?>
```

Using the HTML tags `<pre>` helps you to see each of the indents that represent the structure of the object. You may also choose to print the variable without the `<pre>` tags and view the source of the rendered Web page to see the indents.

Each of the items listed in the printed output can be accessed by referencing `$node->object_property`. For example, if you wanted to print the node ID (`nid`) content type (`type`) to the page, you would include the following snippet in your node template file:

```
<?php print $node->nid; ?>

<?php print $node->type; ?>
```

This node object contains four subsections:

- `body`: contains the same data as the variable `$content`
- `content`: a full listing of the data to be printed to the page as well as formatting instructions
- `links`: the contents of the variable `$links` separated into its component parts

Within these subsections, the `body` contains text and the last three subsections contain arrays.

This content type contains only one extra field. Thus, if five fields were added to the content type, there would be a total of eight subsections in the node object.

Accessing Content in the \$node Object

Content is contained in multiple places within the node object. For example, the `body` field for the node can be accessed from the variable as follows: `$node->body` and `$node->content['body']['#value']`. It makes sense to use the shortest variable name to access the content you need; however, if you need content that is buried deep within the node object, it can be a frustrating task to find the right variable name if you are not comfortable with complex array structures. To show you how to retrieve information from any point in the node object, the next example retrieves the `body` field from the pit of despair (`$node->content['body']['#value']`).

```
The array contains the following code:
content (Array, 5 elements)
  body (Array, 5 elements)
    #weight (Integer) 0
    #value (String, 25 characters ) <p>With some content</p>
    #title (NULL)
    #description (NULL)
    #printed (Boolean) TRUE
  #title (NULL)
  #description (NULL)
  #children (String, 25 characters ) <p>With some content</p>
  #printed (Boolean) TRUE
```

To access the contents of the body field, you would start with the node object and then walk through each of the nested arrays to build the variable: `$node->content['body']['#value']`.

To access information for custom fields in CCK content types, you can use the content variable (shown above). Much of the same information also appears as a sub-section of the node object. The shortcut reveals the content of the field, but not the label. If you need to display the label text as well, you will need to use the contents of the array `$node->content`. For example, to print the contents of a custom field called “Extra text,” you could use either of the following variables:

```
$node->field_extratext[0]['safe']
$node->content['field_extratext']['field']['items'][0]['#item']['safe']
```

To access the label for this field, you could use only the following variable:

```
$node->content['field_extratext']['field']['#title']
```

Additional Template Files

In addition to the node template file there are also template files available for blocks, comments, user profiles and many other modules. To find the template files that are available for the Drupal core modules check this out (<http://api.drupal.org/api/search/6/.tpl.php>). For a list of `tpl.php` files that are available for contributed modules, take a peek inside the modules folder and look for files ending in `tpl.php`.

Some of the commonly customized template files include:

- Search form (<http://api.drupal.org/api/drupal/modules--search--search-theme-form.tpl>)
- User profile (<http://api.drupal.org/api/drupal/modules--user--user-profile.tpl.php/6>)
- Book navigation (<http://api.drupal.org/api/drupal/modules--book--book-navigation.tpl.php/6>)
- Blocks (<http://api.drupal.org/api/drupal/modules--system--block.tpl.php/6>)

Task Summary

1. Create and customize the template file which controls the output of each node on a page using only the variables provided by Drupal core.
2. Extract content from the node object and print it to your node template. Examples are given in the text as well as the sample tpl.php file. Note: this is typically only done when you are using CCK to create additional fields of content.
3. Examine each of the core template files that are available for theming. If appropriate add one or more custom template files to your theme. In many cases this is *not* required as you may control a lot of output using only CSS; however, go ahead and theme as many files from the complete list so that you understand what you would need to do if you had to.
4. Create a custom template file for a specific content type. (For example: the workshop file attached.) If you do not have a custom content type you can still change the style for either Page or Story by using the template naming conventions described in the lesson.

CUSTOMIZING BLOCKS

In the previous section you created a custom template file for nodes on your site. You may have even created a few node template files for each of the different content types on your site. In this section we'll extend that information and customize the block template as well.

Theming Blocks

Blocks may contain a wide range of “things” from site navigation to plain text content to user login to just about anything else the core and contributed modules are capable of spitting out.

1. Copy the default file `block.tpl.php` from <http://api.drupal.org/api/drupal/modules--system--block.tpl.php/6/source> into a new file in your theme directory. This file should be named `block.tpl.php`.
2. Arrange the variables according to how you want all blocks to appear on your site. The variables available to you are listed at <http://api.drupal.org/api/drupal/modules--system--block.tpl.php/6>.

It's as simple as that!

Theming Only Some Blocks

But what if you want to change only SOME blocks? The first step is to decide which block(s) you want to change, then you'll need to create a corresponding template file which uses Drupal's rules for that specific type of block. Here are your options:

1. Theme all blocks within a specific region (e.g. the footer): `block-REGIONNAME.tpl.php`
2. Theme all blocks output from a specific module: `block-MODULENAME.tpl.php`
3. Theme one single block: `block-MODULENAME-DELTA.tpl.php`

The term “delta” refers to the unique identifier that Drupal has assigned that specific block created by that specific module. The combination of module name and delta (number) will be unique.

To find the unique identifier for your block:

1. Navigate to Admin, Build, Blocks.
2. Scroll down to the block you want to customize and hover over the link labeled “configure.” A module name and number will be revealed at the end of the URL.
3. Create a new block template file using the module name and number in the file name using the formula: `block-MODULENAME-DELTA.tpl.php`.

The configuration link will also provide you with additional options which allow you to customize under what conditions a block is displayed on your site.

Collapsible Regions

One of the most annoying things in a theme is an empty region that holds its place for blocks that don't exist. There are a number of themes that do an excellent job of collapsing their unused regions. The conditional statements you learned previously are just one way to collapse unused regions. If you want to see even more examples of themes which have collapsible regions take a look at:

- Acquia Marina (http://drupal.org/project/acquia_marina)
- LiteJazz (<http://drupal.org/project/litejazz>)
- Amity Island (http://drupal.org/project/amity_island)
- A3 Atlantis (smart columns) (http://drupal.org/project/a3_atlantis)

You may have other favourites as well.

SHARING YOUR THEME

Finally we get to the 5th step in the Design to Theme in Five workbook: sharing your theme with others. When you uploaded your theme to your Web server you probably FTPed (or SCPed) individual files. To share your theme with others you'll need to put them into some kind of wrapper. Typically a `.zip` or a `.tar.gz` file is sufficient. Most utilities capable of unzipping these archive formats can also create archives of this format. You will need to check the instructions for your specific application. If there are any "gotchas" for your theme you may also want to include a plain text file named, `README.txt` with the extra installation instructions. For example: if your theme is optimized for a specific content type you may want to include information on how to configure Drupal so that the theme works properly.

Whether you are sharing your theme with a client or the rest of the world there are a few extra considerations you'll need to be aware of.

Licensing

Drupal is released under the GPL (GNU Public License). Any projects hosted on drupal.org are licensed under the same conditions. The full text of the license is available from this site: (<http://www.gnu.org/copyleft/gpl.html>). Although you are encouraged to use this license for your themes, there is no obligation to do so if you're not hosting your theme on drupal.org. You may also want to use a Creative Commons license (<http://creativecommons.org/choose/>). When choosing your license you need to make sure you have the right to distribute all parts of your theme under the new license--including graphics and photographs.

Selling Themes

You may also choose to retain complete copyright over your theme and sell it on the Internet. Top Notch Themes (<http://www.topnotchthemes.com/>) is the gold standard for selling Drupal themes online. There are other companies that also sell themes. For more ideas on how to sell your themes online do a search for "drupal themes" and see how other businesses have set up their sites.

Contributing Themes to the Drupal Project

Of course you can upload your theme to your own Web site and have people download it from there. If, however, you'd like to have your theme included in the main Drupal Web site there are a few extra steps you'll need to go through. There are two guides you'll need to read. Yes they involve scary revision control information. Mostly we use this to scare off people who are only mildly interested in contributing their theme. It is worth the learning curve you'll have to go through to be able to promote your design on drupal.org!

- Starter guide on contributing to Drupal projects (themes and modules) (<http://drupal.org/node/7765>).

- [CVS Guide for Theme Maintainers \(http://drupal.org/node/262432.\)](http://drupal.org/node/262432)

If you are IRC-friendly the easiest thing to do is to read through the documentation I've listed below and then hop into irc.freenode.net in the channel #drupal-contribute and let people know you're ready for help. This process involves getting a CVS account. If you are truly committed to uploading your theme to [Drupal.org](http://drupal.org) (and maintaining it!) you can jump start the help process by applying for a CVS account online¹.

1 <http://drupal.org/cvs-application/requirements>

Summary

The first step in creating a great Drupal theme is to find the page elements in your design file that will be controlled by Drupal template files. Your theme may include the following page elements:

- blocks contained in regions (aka sidebars)
- page content
- headings
- navigation

Next you converted your simple HTML page into a Drupal theme. This step made it possible for you to test your theme for the first time. You created the info file used by Drupal to identify parts of your theme and added Drupal variables to your modest HTML wireframe from the first section. The variables you added to your page template may have included:

- regions
- logo
- content

You may have also had additional CSS files to include in your theme. These would have been added as part of the second step of making your Drupal theme.

Your next step was to add navigation to your page template. You learned how to wrap conditional statements around things that you only wanted to be displayed “sometimes.” You also evaluated Base (or Starter) themes. If you completed each of the activities to this point you would have read the project pages for over ten different starter themes and tried at least one of the Zen, NineSixty and BeginningW2 starter themes.

Then it was time to get into the nitty gritty details of a Drupal theme: you created a template file to style the guts of every node and block that is displayed within the page template.

Finally you learned the basics of how to share your theme with others.

APPENDIX A: DESIGNING WITH GRIDS

Grid Based Design is good. CSS Frameworks are lovely. If you haven't spent time with them, dig in! Using CSS Frameworks will make the HTML and CSS part of designing for Drupal a LOT easier. There are lots of grid systems available for the Web. Raj Dash wrote a good article² on why you should use a CSS Grid Framework and gives a brief comparison of a few systems including his reasons for using Blueprint CSS³.

I quite like the 960 Grid System. You can download design templates and see sample sites at: www.960.gs. Four Kitchens has created a very extensive slidedeck on Accelerated Grid Theming Using NineSixty⁴.

Drupal has a number of starter themes available for the various CSS grid frameworks. Don't worry about picking the "wrong" one. You can always incorporate your CSS framework into any Drupal theme you create. (Although you can't mix and match frameworks...so do choose whether you want to work with 960.gs or Blueprint or...)

Grid Resources

- Grid Based Design: Six Creative Column Techniques Smashing Magazine <http://www.smashingmagazine.com/2008/03/26/grid-based-design-six-creative-column-techniques/>
- Outside the Grid A List Apart <http://www.alistapart.com/articles/outsidethegrid>
- Five Simple Steps to Designing Grid Systems by Mark Boulton http://v3.markboulton.co.uk/articles/detail/five_simple_steps_to_designing_grid_systems/
- 30 Blogs with Grid Based Design cssJuice <http://www.devwebpro.com/30-weblogs-with-grid-based-design/>
- Grid and Column Designs Web Designer Wall <http://www.webdesignerwall.com/trends/grid-and-column-designs/>

For truly advanced "bonus" material look up the difference between a "horizontal" and "vertical" grid then check out the references at Smashing Magazine on Designing with Grid-Based Approach⁵.

Task Summary

1. Download the design templates for the 960 Grid System (www.960.gs). You may choose from Fireworks, InDesign, Inkscape, Illustrator, OmniGraffle, Photoshop, Visio or Expression Design.
2. Open them in your design tool of choice.
3. Adjust your site's design to fit into the constraints of the grid.

2 <http://net.tutsplus.com/tutorials/html-css-techniques/which-css-grid-framework-should-you-use-for-web-design/>

3 <http://www.blueprintcss.org/>

4 <http://fourkitchens.com/sites/default/files/accelerated-grid-theming-2009-08-02.pdf>

5 <http://www.smashingmagazine.com/2007/04/14/designing-with-grid-based-approach/>

4. Create an HTML wireframe to hold your content. You may find the following tutorial useful <http://dabrook.org/blog/articles/tutorial-for-the-960-grid-system-css-framework>.
5. Add design components (and relevant CSS) to your static HTML page.

APPENDIX B: node-WORKSHOP.TPL.PHP FILE

```
<div id="node-<?php print $node->type; ?>" class="node clear-block">

<div id="product-info">
<h2 class="title"><?php print $node->title; ?> Registration</h2>
  <p>Dates:
<?php print date_format(date_create($field_date[0]['value']), "M j"); ?> to
<?php print date_format(date_create($field_date[0]['value2']), "M j"); ?></p>

  <p>Fee: $ <?php print number_format($sell_price, 2); ?></p>
<?php print $node->content['add_to_cart']['#value']; ?>

  <p class="cancellation">
    <a href="/cancellation-policy">Cancellation Policy</a>
  </p>
</div>

<div class="content">
  <?php if($node->content['image']['#printed']): ?>
    <?php print $node->content['image']['#value']; ?>
  <?php endif; ?>

  <div id="content-body">
    <?php print $node->content['body']['#value']; ?>
  </div>
</div>

<?php if ($links): ?>
  <div class="links"><?php print $links; ?></div>
<?php endif; ?>

</div>
```