Theme HowTos

The following section provides a collection of 'How-to' articles on subjects relevant to theme developers. For a selection of useful code samples, see the Theme Snippets section.

Convert any website layout or template into a Drupal theme - easily!

Theming for Drupal is not actually as hard as you may think. Certainly there are more advanced areas of theming possible with Drupal, which may or may not be necessary depending on the needs of your site or whether there are subtle defaults which you wish to change (and which you can learn about if and when they become necessary for your site). Creating the "overall theme" for your site though is very simple.

A Drupal theme is nothing more than a standard website layout, with a few bits of PHP code added in certain places, which you can simply copy/paste into place in order to call up Drupal's "dynamic" content and features in those spots.

If you learn better visually, a video is available which covers much of the same information that will be illustrated here: Static Theme Conversion.

Contributed/downloaded themes are more complex than "your" theme needs to be

You may have looked at various pre-made themes which come with Drupal core, or are available for download, and sighed with frustration at seeing how confusing the code of some of those themes looks.

Contributed themes such as those are created to handle all kinds of "possible" scenarios about how the theme might need to work "for everyone", allowing you to dynamically add various columns and other aspects of the layout, and supporting custom theming for many features and modules which you might not even need on your site. These themes usually include a large amount of PHP code that essentially says *"If this setting is true, then show this part this way, or else show that."* This makes the theme very flexible, but also very complicated.

However, "your" theme doesn't need to work for everyone. It only needs to do the things your own site requires and nothing more. All that you need are a few bits of PHP which you can simply copy and paste into your own HTML template, which tell Drupal to load its dynamic content in those spots.

You might also find it useful to have a look at Drupal's "most basic" core template files (those used when a theme doesn't provide any alternative template files to override them). To view these basic templates, look in the */modules/system* folder of your Drupal installation (in particular, look at *page.tpl.php* and *node.tpl.php*).

Just a few simple & basic elements make up a Drupal theme

The "main" template file in Drupal is called *page.tpl.php*. Make a copy of your website layout/template and call it *page.tpl.php*. Then, simply paste the following bits of code into your layout, in the locations noted (all the code mentioned in this lesson goes in the *page.tpl.php* file).

The <head> section of your site

```
<head>
  <title><?php print $head_title; ?></title>
  <?php print $head; ?>
  <?php print $styles; ?>
  <?php print $scripts; ?>
</head>
```

The side bars or columns of your layout

Most websites have a left and/or right column next to the "main" larger content area. These side bars usually contain various features and information (for instance a log-in form, navigation, recent comments, etc). In Drupal these features/information are called "Blocks", and you can customize where they show up on your page by going to **Administer > Site building > Blocks** *(admin/build/block)*. In order for Drupal to know *where* to put the Blocks though, you need to add some "placeholders" to your theme (Drupal calls these placeholders "Regions"). There are more of these placeholders available (and you can create your own custom ones too) but it's important to add the side bar one(s) at first, since the admin menu appears in them. Add the below Region(s) to your side bar:

If you're using Drupal 5:

```
<?php print $sidebar_left ?>

...and/or....

<?php print $sidebar_right ?>
```

If you're using Drupal 6:

```
<?php print $left ?>

...and/or....




<?php print $right ?>
```

Menu system / navigation

Drupal's default menu for your site's content is called **Primary Links**. There is also a **Secondary Links** menu which can show related sub-pages under the Primary menu. When stripped of all images and styles, a menu in Drupal is nothing more than a nested unordered list, in plain HTML, with links for each list item. You can use CSS to style a menu in any way you want; changing how it looks, whether it is horizontal (like tabs) or vertical, etc.

To make the Primary and Secondary Links menus appear on your site, paste the following into either the top area of your theme (for instance, if your menu is going to be styled as tabs or buttons), or in the sidebar area (for instance if you plan to have your menu show vertically in the sidebar).

Simple way:

Here's the simple version to use, if you're already sure you want to use "both" Primary and Secondary Links (or feel free to delete the Secondary Links portion if you don't need it):

```
<div id="primary">
  <?php print theme('links', $primary_links); ?>
</div> <!-- /#primary -->

<div id="secondary">
  <?php print theme('links', $secondary_links); ?>
</div> <!-- /#secondary -->
```

More flexible way:

If you're not sure that you will have both Primary and Secondary Links, you could use the following code instead (it's the same, but just does a "check" to see if the menus are enabled, and only shows them if they are):

```
<?php if ($primary_links): ?>
  <div id="primary">
    <?php print theme('links', $primary_links); ?>
  </div> <!-- /#primary -->
<?php endif; ?>

<?php if ($secondary_links): ?>
  <div id="secondary">
    <?php print theme('links', $secondary_links); ?>
  </div> <!-- /#secondary -->
<?php endif; ?>
```

The main "dynamic" content of the page

At this point, you've added all of the "framework" of your site's layout which surrounds every page, and is generally similar or the same throughout the entire site. Now though, it's time to include the area where actual **Content** will be displayed in the template (different content depending on the page being viewed, of course).

Simply locate the spot in your template where the main content is to be displayed (usually the center of the page, after the header and between any sidebars), and paste the below code in that spot (if you'd like to re-order things feel free, but otherwise you can simply paste this as-is):

```
<?php if ($breadcrumb): ?><div id="breadcrumb"><?php print $breadcrumb; ?></div><?php
endif; ?>
<?php if ($mission): ?><div id="mission"><?php print $mission; ?></div><?php endif; ?>

<div id="content">
<?php if ($title): ?><h1 class="title"><?php print $title; ?></h1><?php endif; ?>
<?php if ($tabs): ?><div class="tabs"><?php print $tabs; ?></div><?php endif; ?>
<?php if ($messages): print $messages; endif; ?>
<?php if ($help): print $help; endif; ?>

<?php print $content; ?>
</div>
```

Footer: Add a final tag at the bottom of your theme

Paste the following at the end of your *page.tpl.php* file. You *must* have this tag at the bottom, which will dynamically include any scripts that need to appear at the bottom of the page, as well as close up any loose

ends and tags that might have been left open:

```
<?php print $closure ?>
</body>
</html>
```

For Drupal 6: Create a YourThemeName.info file

Lastly, in Drupal 6, the one final element is to create a *YourThemeName.info* file (where *YourThemeName* is the name of your theme, with no spaces). Simply make a blank file and paste the following code into it, customize it with the appropriate name/description, and save it in the same directory along with the rest of your theme's template files:

```
name = YourThemeName
description = Description of YourTheme
core = 6.x
engine = phptemplate
stylesheets[all][] = style.css
regions[left] = Left sidebar
regions[right] = Right sidebar
regions[content] = Content
regions[header] = Header
regions[footer] = Footer
```

The "regions" portion after *style.css* includes all of Drupal's "standard" regions, and is recommended so that you can easily remove regions you don't want to use, or add custom regions. If you want to add custom regions, you must manually include any of the default ones you wish to keep as well or the defaults will disappear from your theme (simply copy/paste them from this page or another theme). If you want to add a custom region, simply copy one of the *region* lines and rename it.

That's all - Paste the above bits of code into ANY site layout or template, and you have a basic, functional Drupal theme.

Learning more about Drupal theming

There are quite a few more elements you could use in your theme if you want to. Here are a number of excellent resources for learning the more detailed aspects of theming in Drupal:

- Anatomy of a Drupal theme
- Full listing of available Drupal 5 theme variables to choose from for page.tpl.php.
- Full listing of available Drupal 6 theme variables to choose from for page.tpl.php.
- Drupal theming for designers - this is a fantastic article, including screenshots and diagrams to help you clearly understand how Drupal's theming system works.
- Theme guide (Drupal 6) - Official theming guide on Drupal.org

Convert XHTML template to Drupal theme in 5

See http://drupal.org/node/213530 (screencast plus separate worked example).

Custom ul list-style

Difficulty level: intermediate to advanced.

Example 1: The basics

For this example we will create a unordered list (ul) that will show a disc next to a list item (li) only in a particular area, and will hide them in the rest of the website. You can then decide which lists you want to style in your website.

Note:
Make sure you do this in a development website, and in any case have a backup of the files you are going to work on.

We will take a div and name it "content". The name of your div may be different. In it we will place the ul list to style:

```
<div id="content">
<ul class="unordered_list_in_content">
<li>list item 1</li>
<li>list item 2</li>
<li>list item 3</li>
</ul>
</div>
```

In your stylesheet you would use:

```
ul { list-style-type:none } /* This will turn off all list-style-types in your theme
*/

#content ul.unordered_list_in_content { list-style-type: disc }
```

You should now see your list with discs in the area that you chose, and not in the rest of the website.

Example 2: Let's have some fun with lists styles

To turn off all list-style in a theme use:

```
ul {list-style:none}
```

instead of

```
ul {list-style-type:none}
```

You can do do something like this:

```
ul {list-style:disc inside}
```

instead of

```
ul {list-style-type:disc}
```

So your final stylesheet would look like:

```
ul {list-style:none}

#content ul.unordered_list_in_content { list-style: disc inside}
```

Note: If you have more than one stylesheet in your theme you will have to make sure there are no list-styles overriding your css, otherwise it still might not show.

Customize the search results page

Do you need to change the content shown in the results of the Drupal search? It is imperative to do this in the Drupal way, and it is most helpful to understand the thought process of doing such a procedure. Following is an example of how to customize the Drupal search results from more than just a technical aspect but also from the thought process required. This same thinking can be done when having to customize other components within Drupal framework.

The first step is to find out what is driving the Drupal search. In the modules directory you will see the search folder which is a pretty good indication that this is what drives the Drupal search. Now, open the search module folder to see what is inside and what will be there to customize, keeping in mind that you will not actually be doing any changes to any files in the search module itself.

Inside the search module you will find the standard **.info**, **.install**, **.module** and a **.css** file. In this case you will be looking at the search.module itself. What you are looking for is something you can hook into or theme. The functions you are looking for are theme_ functions and in the search module you will find theme_search_theme_form, theme_search_block_form, theme_search_item and theme_search_page. You will be able to customize any one of these functions in a way that will not touch the actual Drupal core module. Great!

In this example you are going to change the look of the search result page (or srp to all those seo gurus). We are going to **remove** the line of info that tells **who wrote the node, when it was written, comments etc...** From first glance it may be a good assumption that the theme_search_page function is the one to customize from within my own theme. For this example we will use the zen theme that would be located in your /sites/all/themes/zen folder. Here is the theme_search_page which is nicely commented so you can see the parameters, and purpose of this function.

```
/**
* Format the result page of a search query.
*
* Modules may implement hook_search_page() in order to override this default
* function to display search results. In that case it is expected they provide
* their own themeable functions.
*
* @param $results
*   All search result as returned by hook_search().
* @param $type
*   The type of item found, such as "user" or "node".
*
* @ingroup themeable
*/
function theme_search_page($results, $type) {
  $output = '<dl class="search-results">';

  foreach ($results as $entry) {
    $output .= theme('search_item', $entry, $type);
  }
  $output .= '</dl>';
  $output .= theme('pager', NULL, 10, 0);

  return $output;
}
```

After looking more closely you will notice that this function calls the **theme_search_items** function by way of theme('search_item', $entry, $type) so you now see that this is actually the function that you want to manipulate. This function looks like the following:

```
function theme_search_item($item, $type) {
  $output = ' <dt class="title"><a href="'. check_url($item['link']) .'">'.
check_plain($item['title']) .'</a></dt>';
  $info = array();
  if ($item['type']) {
    $info[] = check_plain($item['type']);
  }
  if ($item['user']) {
    $info[] = $item['user'];
  }
  if ($item['date']) {
    $info[] = format_date($item['date'], 'small');
  }
  if (is_array($item['extra'])) {
    $info = array_merge($info, $item['extra']);
  }
  $output .= ' <dd>'. ($item['snippet'] ? '<p>'. $item['snippet'] .'</p>' : '') .'<p
class="search-info">'. implode(' - ', $info) .'</p></dd>';
  return $output;
}
```

So what you will do next is to copy all of the above function directly into my **template.php** file that is found in the /sites/all/themes/zen folder. The template.php file is what you can use in order to indirectly manipulate core Drupal. Not just for search, but for all core modules. Then **change** the name of the **theme**_search_item($item, $type) function to **zen**_search_item($item, $type). After saving the file, Drupal no longer calls search modules function but now calls your newly created theme zen_search_item function. Simply awsome, and now when you have to do a core drupal update, your changes in the theme will not be over written when a core search module update is performed.

Don't forget to change the name of your template.php function or you may get the following error. "Fatal error: Cannot redeclare theme_search_item() (previously declared in....)" I just saying ... this didn't happen to me ;)

Now, lets get to the really good stuff. Because this new **zen_search_item()** function is in your theme, you are pretty free to do whatever you need in order to get the **results you desire**. So, my quick change of removing the extra info mentioned previously can be accomplished by removing the code that contains the extra data that you don't need. Your new function in your template.php file now becomes:

```
function zen_search_item($item, $type) {
  $output = ' <dt class="title"><a href="'. check_url($item['link']) .'">'.
check_plain($item['title']) .'</a></dt>';
  $output .= ' <dd>'. ($item['snippet'] ? '<p>'. $item['snippet'] .'</p>' : '');
  return $output;
}
```

So, you can see that you removed a lot of info but at this point you can go both ways. You can add content, change the elements or do whatever you desire, without editing core Drupal!

Displaying random images

This is made possible by a script found at www.alistapart.com called *rotate.php*.

Procedure:

1. Create a directory under the web server's document root. I placed it under my theme's directory.
2. Populate that directory with images.
3. Download script and put it in a nice resting place.
4. Edit script so it can find your image directory. To avoid editing you can just place the script in the same directory as the images.
5. Under your theme's config, for custom logo, instead of a path to an image use the path to the script.
6. You're done.

HowTo: Add Regions to your Frontpage

DEPRECATED: This method will NOT work for Drupal 6.x. Please see this page for a solution.

Note that this is for use with PHPTemplate themes (the default theme engine in Drupal.) There is a page for Plain PHP themes to do something similar.

In this example we have a custom theme, called 'mytheme', and we are going to add three new regions to the frontpage. We will call them frontpage top, frontpage center and frontpage bottom.

Replace all occurrences of 'mytheme' with the name of your custom theme.

1. Inside of themes/mytheme/template.php (create the file if it doesn't exist), add the following:
```
<?php
/** Define the regions **/
function mytheme_regions() {
  return array(
        'left' => t('left sidebar'),
        'right' => t('right sidebar'),
        'content' => t('content'),
        'header' => t('header'),
        'footer' => t('footer'),
        'frontpage_top' => t('frontpage top'),
        'frontpage_center' => t('frontpage center'),
        'frontpage_bottom' => t('frontpage bottom'),
   );
}
?>
```
2. Now that the regions are defined, go to page.tpl.php (within your themes/mytheme folder). Place the following code where you would like to display the appropriate regions:
```
    <?php if ($is_front || strstr($_GET['q'], 'admin/block')) : ?>
    <div id="frontpage_top" class="frontpage">
      <?php print $frontpage_top ?>
    </div>
    <?php endif; ?>
```
3. Repeat this procedure for all three frontpage regions. You can combine two regions within one *if()* statement to improve performance.
4. To add blocks to these new regions, simply go to Administer > Site Building > Blocks

Now with some more HTML (tables) or CSS, you can place these regions anywhere on your frontpage, make them look nicer, etc.

If you do not want the default content to print on the frontpage, you should surround the `<?php print $content ?>` with an *if (!$is_front)*.

Visit the blocks repository for nice blocks, or create them with views so that you can put them on your frontpage.

Front page blocks for plain PHP themes

I was able to add a front page block by editing the drupal.theme file in my plain PHP theme. This means if you are using a Chameleon clone (the one with the chameleon.theme file) all you have to do is add the new blocks reference to the yourtheme_regions() function, the block theme reference to the yourtheme_page function:

```
  // Get blocks before so that they can alter the header (JavaScript, Stylesheets
etc.)
  $blocks_left = theme_blocks('left');
  $blocks_right = theme_blocks('right');
  $blocks_home_center = theme_blocks('home_center');
```

then add the following code where you want your block to appear.

```
  $output .= "\n<!-- begin content -->\n";

  if ( $show_blocks && !empty($blocks_home_center) && drupal_is_front_page() ){
    $output .= "<div id=\"home_center\" class=\"home-center
\">$blocks_home_center</div>";
  }
```

As you can see I put mine just after the begin content marker. For left and right blocks the areas in the theme file are clearly marked.

If this is a center block, you may also need to add a line to common.css after .block

```
.block {
  width: 180px;
}
#block-block-11{
width: 100%;
}
```

You will need to publish the block or hover over the configure in admin blocks to get the block number

HowTo: Create a custom user login bar

In this how-to, I will show you how to create a user bar, that:

- Displays a compact, nice user login form that can fit in one line.
- Once user is logged in, It will display a welcome message and a couple of helpful links.

This is the final result we will be trying to achieve:

As you can see, this how-to is written for Garland theme, the default theme for Drupal 5. But I'm pretty sure it can be easily used in any other theme.

Decide on the approach..

Drupal is very powerful, it's so flexible that you have many ways to do things. That's why a 5 minutes of thinking is very recommended before we get our hands dirty.

The solution I thought of will work like this:

- We will write a separate function, will call it garland_user_bar(). It decide, and return HTML for the login form, or the welcome message. It will contain the necessary code to check which one it should return.
- In page.tpl.php, we will place a call to this function somewhere where it gets displayed in the top.
- Back to the function, If it should show a login form, then it will just borrow the default login form, we will not try to create our own. Why? Because likely, the default form, while it looks rectangular in shape and fits in many lines, by checking view-source I could tell that we can style it using CSS just like in the screenshots without changing anything in the form itself.
- If the user is already logged in, the function will instead return a nice welcome message, and some helpful links.
- Now we have the logic, after that we will write some CSS to style it, to change how it looks.

Oh, let's place the function in page.tpl.php first

Let's do this first, let's place a call to garland_user_bar() (even if it doesn't exist yet), that's better so while we are working on garland_user_bar() we can hit 'refresh' in our browser to see the result any time we want to.

1. Open `themes/garland/page.tpl.php`, at around line 16 you will find:
   ```
   <div id="navigation"></div>
   ```

This `<div>` is the container for the top bar in Garland theme, so let's add our call to garland_user_bar() there..

```
<div id="navigation"><?php print garland_user_bar() ?></div>
```

Write the necessary code, garland_user_bar()

Before giving you the whole piece of code, I will explain how we go for writing this..

1. Open `themes/garland/template.php`, at the of the file, let's start an empty function.. we will add code inside it in the next steps..
   ```
   function garland_user_bar() {
   }
   ```
2. We will need access to the global $user variable, which Drupal makes available for us, and it's going to tell us whether a user is logged in already or not. So first line we add inside the function is going to be:
   ```
   global $user;
   ```
3. I will also define a new variable called `$output`, so I can keep adding whatever I want to that variable, and at the end, I will return its contents to be placed on where we placed the function itself in page.tpl.php. Alright, So that's extra line..
   ```
   $output = '';
   ```
4. I want to check if the user is logged in or not. We do this by checking if `$user->uid` exists or not, because when Drupal logs a user in, it will automatically change $user->uid to reflect the user id, which is always > 0. That's what we need to add..
   ```
   if (!$user->uid) { // if user is NOT logged in..
   ```

   Notice the `!` here, it translates to NOT, so this code means "if not $user->uid" which means if the user is NOT logged in already...
5. Alright, if the user is NOT logged in.. do what? Give him/her a login form! Remember what we concluded in the deciding step? We said that the default login form is okay, it just needs a bit of CSS love, no need to write another form. So let's just use that form.. this is how..
   ```
   $output .= drupal_get_form('user_login_block');
   ```

   `drupal_get_form()` is a very handy function, you give it the name of a form, and it returns it to you, ready for user consumption.
6. Okay, now we handles the case when user is not logged in, but what if they are already logged in?? let's continue our code to handle this case.. add:
   ```
   }
   else {
   ```

   This allows us to handle the other case.
7. If user is logged in already, we want to display a welcome message, and a few helpful links. Let's see how we display the welcome message first..
   ```
   $output .= t('<p class="user-info">Hi !user, welcome back.</p>', array('!user' =>
   theme('username', $user)));
   ```

   `t()` function encapsulates the whole message, so it can be translated. `theme('username', $user)` is what you should always use for displaying usernames.
8. Well, now the links.. I want two links, "Your accont" and "Sign out". Drupal has a function called theme_item_list() that you can give a list of items, and it will return a HTML `<ul>` list out of them. "Your account" should link to `user/<uid>`, and "Sign out" should link to `logout`. These are standard URLs in Drupal. Alright, let's do this..
   ```
   $output .= theme('item_list', array(
     l(t('Your account'), 'user/'.$user->uid, array('title' => t('Edit your
   ```

```
account'))),
  l(t('Sign out'), 'logout')));
}
```

Oh well, new function here, `l()`, this one is used to generate links. You simply pass it the title and the URL, and it will generate a sound and safe `<a href="..." title="..">....</a>`.

9. Now one more last thing before we exit and return the output, I want to encapsulate all this in a `<div id="user-bar">...</div>`. Why? because it will make it easier for us to write the CSS that will apply to this bar. How I do this? Easy, using the . (dot) operator. Like this:
   ```
   $output = '<div id="user-bar">' . $output . '</div>';
   ```
10. Oh, now everything is ready, let's return the HTML we generated..
    ```
    return $output;
    ```

So what's the final overall function? this is it..

```
<?php
function garland_user_bar() {
  global $user;
  $output = '';

  if (!$user->uid) {
    $output .= drupal_get_form('user_login_block');
  }
  else {
    $output .= t('<p class="user-info">Hi !user, welcome back.</p>', array('!user' =>
theme('username', $user)));

    $output .= theme('item_list', array(
      l(t('Your account'), 'user/'.$user->uid, array('title' => t('Edit your
account'))),
      l(t('Sign out'), 'logout')));
  }

  $output = '<div id="user-bar">'.$output.'</div>';

  return $output;
}
?>
```

Styling time!

Now that login form and welcome message are displayed correctly for users, we only need to make them look better. That's where CSS comes in.

1. First, let's create a new file to place all our user bar CSS into, that way we keep organized. So create a new file `themes/garland/user_bar.css`.
2. Now the file exists, but nobody knows about it, we should modify page.tpl.php to include it. How? Open `themes/garland/page.tpl.php` and AFTER line 11 add this:
   ```
   <style type="text/css" media="all">@import "<?php print base_path() .
   path_to_theme() ?>/user_bar.css";</style>
   ```

   That will include our user_bar.css file.
3. Let's write the magical CSS inside user_bar.css
   Notice that it's not too big really. It's just the comments consuming too much space :p.
   ```
   /*
   this will expand the default garland bar, make it bigger so our form and message
   can fit in.
   */
   #navigation {
   ```

```
  height: 3em;
}

/*
by default, the default form adds some surrounding space, this cancels it
*/
#navigation div.form-item,
#navigation div.content {
  margin: 0; padding: 0;
}

/*
this adds some space in top and bottom, so anything inside can look vertically
centered
*/
#user-bar {
  padding: .65em 0;
}

/*
by default, fields labels tries to reserve a whole line for itself, this
cancels that and and sends it to the left.
it also adds some space on the right and left of the label to look easy on
the eye.
*/
#user-bar label {
  float: left;
  margin-left: 10px;
  margin-right: 2px;
}

/*
inputs too, they try to reserve a whole line for itself, this
cancels that and sends it to the left
*/
#user-bar input {
  float: left;
}

/*
I don't like the required * (asterisks), so I hide them.
*/
#user-bar span.form-required {
  display: none;
}

/*
the form submit button, it's so tight so we expand it a bit, and give it some
free space around.
*/
#user-bar input.form-submit {
  margin-top: -1px;
  margin-left: 10px;
  padding: 0 .5em;
}

/*
now this is for the links list, lists by default tries to reserve a whole line
also they add space surrounding them. we cancel all that and send the list
to the right
*/
#user-bar div.item-list ul {
  float: right;
  margin: 0; padding: 0;
  margin-right: 10px;
```

```
}

/*
remember, stylign above was for the whole list, now for each item,
we all know each item in the list by default exists on a separate line, also
has that bullet on the left. we cancel all that. and makes all items sit beside
each other
*/
#user-bar div.item-list ul li {
  float: left;
  background: none;
  margin: 0 5px;
  padding: 0 10px;
  border: 1px solid #b8d3e5;
}

/*
this is the "Hi user, welcome back message".
by default <p> tries to exist on a separate line, we cancel that.
also by default <p> has some surrounding space, we cancel that too, and give it
only space on the left.
*/
#user-bar p.user-info {
  float: left;
  padding: 0;
  margin: 0 0 0 10px;
}
```

The end..

That was it, A solution that works and doesn't override or modify any existing code. And since it doesn't have any custom hardcoded HTML for forms, that means it will rarely need to be maintained after new releases of Drupal.
Post any improvements you can think of in the comments, and I will add them here.

HowTo: Emulate "preprocess" theme functions in Drupal 5

Drupal 6 introduced the concept of a theme preprocess function. In short, it allows themes, theme engines, and even modules to step into the theming process and manipulate the variables that get sent to a template. It replaces the older, clunkier _phptemplate_variables() function from Drupal 5's phptemplate templating engine, but serves much the same purpose.

While we can't backport all of that new functionality, it is possible to greatly simplify _phptemplate_variables() in Drupal 5 in a way that looks a lot like Drupal 6. Specifically, we can break up _phptemplate_variables() into separate functions that act like a theme's preprocess functions in Drupal 6.

To get that simplification, simply specify your _phptemplate_variables() function in template.php like so:

```
<?php
function _phptemplate_variables($hook, $vars) {

  $function = 'phptemplate_preprocess_'. str_replace('-', '_', $hook);
  if (function_exists($function)) {
    $function($vars);
  }
```

```
  return $vars;
}
?>
```

First, we ensure that the theme hook is in a format that can be used as part of a function name, and build the name of what the preprocessing function would be called. Then if it exists, we call it with the $vars array. Note that we do not accept a return value from the preprocessing function. Instead, in the called function, we accept the parameter by reference like so:

```
<?php
function phptemplate_preprocess_page(&$vars) {
  // Manipulate the $vars array here.
}
?>
```

Now we can, in the theme, step into any template file's variable building process add, remove, or modify variables to be sent to the template just by declaring the appropriate function. Although it doesn't actually give us any new functionality, it is easier to read and maintain than a giant switch statement in _phptemplate_variables() and will also be easier to upgrade to Drupal 6 later.

Separate var files when the template function starts to become unmanageable

great technique that we use when the template function starts to become unmanageable, separate them into their own vars files:

```
<?php
/**
* Intercept template variables
*
* @param $hook
*   The name of the theme function being executed
* @param $vars
*   A sequential array of variables passed to the theme function.
*/
function _phptemplate_variables($hook, $vars = array()) {
  global $user;

  $vars['user'] = $user;
  $vars['path'] = base_path() . path_to_theme() .'/';
  $vars['admin'] = $user->roles[3] || $user->uid == 1 ? TRUE : FALSE;
  $vars['moderator'] = $user->roles[4] ? TRUE : FALSE;
  $vars['editor'] = $user->roles[5] ? TRUE : FALSE;
  $vars['authenticated'] = $user->uid ? TRUE : FALSE;
  $vars['theme'] = 'my_theme_name';

  // Include broad variables for each hook.
  if (file_exists($vars['directory'] .'/'. $hook .'.vars.php')) {
    include_once $vars['directory'] .'/'. $hook .'.vars.php';
    $function = $vars['theme'] .'_variables_'. str_replace('-', '_', $hook);
    $vars = $function($vars);
  }

  // Specific variables for node types.
  if ($hook == 'node') {
    if (file_exists($vars['directory'] .'/node-'. $vars['node']->type .'.vars.php')) {
      include_once $vars['directory'] .'/node-'. $vars['node']->type .'.vars.php';
      $function = $vars['theme'] .'_variables_node_'. $vars['node']->type;
```

```
      $vars = $function($vars);
    }
  }

  return $vars;
}
?>
```

Note: Thanks Nate! (modified slightly so specify theme name in a var)

Further enhancements

Adding another convenience to it:

```
<?php
function phptemplate_preprocess_node(&$vars) {
  $node = $vars['node'];
  $type = $node->type;

  // general node variables processing

  // preprocess by node type
  if (function_exists($function = "phptemplate_preprocess_node_$type")) {
    $function($node, $vars);
  }
}
?>
```

which let us use a different preprocess function per node type, for example:

```
<?php
function phptemplate_preprocess_node_story($node, &$vars) {
  // specific story variable processing
}
?>
```

HowTo: Have "first" and "last" classes on blocks

Sometimes, the first or last block in a region needs to be styled different than the rest. To get 'first' and 'last' classes on the first and last block in a region, you need to override two theme functions: theme_block() and theme_blocks().

1) Override theme_blocks()

```
<?php
function phptemplate_blocks($region) {
  $output = '';

  if ($list = block_list($region)) {
    $blockcounter = 1; // there is at least one block in this region
    foreach ($list as $key => $block) {
      // $key == <i>module</i>_<i>delta</i>
      $block->extraclass = ''; // add the 'extracclass' key to the $block object
      if ($blockcounter == 1){ // is this the first block in this region?
        $block->extraclass .= ' first';
      }
      if ($blockcounter == count($list)){ // is this the last block in this region?
        $block->extraclass .= ' last';
      }
```

```
      $output .= theme('block', $block);
      $blockcounter++;
    }
  }

  // Add any content assigned to this region through drupal_set_content() calls.
  $output .= drupal_get_content($region);

  return $output;
}
?>
```

2) Override theme_block()

```
<?php
function phptemplate_block($block) {
  $output  = "<div class=\"block block-$block->module $block->extraclass\"
id=\"block-$block->module-$block->delta\">\n"; // in this line, the extraclass value
is added as class
  $output .= " <h2 class=\"title\">$block->subject</h2>\n";
  $output .= " <div class=\"content\">$block->content</div>\n";
  $output .= "</div>\n";
  return $output;
}
?>
```

That's it!

HowTo: Have "first" and "last" classes on list view LI's

Much like HowTo: Have "first" and "last" classes on menu blocks sometimes you need to be able to address the first and last elements of lists, this howto allows you to have that for your LI's.

The processing of lists is done by `function theme_item_list($items = array(), $title = NULL, $type = 'ul', $attributes = NULL)` which can be over ridden in template.php, it only takes a few small changes to get this to work (adding $item_key to the foreach and checking it against count and 0 to determine if to apply the class).

Paste the code below into your template.php and your done.

```
<?php

function phptemplate_item_list($items = array(), $title = NULL, $type = 'ul',
$attributes = NULL) {
  $output = '<div class="item-list">';
  if (isset($title)) {
    $output .= '<h3>'. $title .'</h3>';
  }

  if (!empty($items)) {
    $output .= "<$type" . drupal_attributes($attributes) . '>';
    foreach ($items as $item_key=>$item) {
      $attributes = array();
      $children = array();
      if (is_array($item)) {
        foreach ($item as $key => $value) {
          if ($key == 'data') {
```

```
            $data = $value;
          }
          elseif ($key == 'children') {
            $children = $value;
          }
          else {
            $attributes[$key] = $value;
          }
        }
      }
      else {
        $data = $item;
      }
      if (count($children) > 0) {
        $data .= theme_item_list($children, NULL, $type, $attributes); // Render
nested list
      }

      if($item_key == 0) {
        $attributes['class'] = (isset($attributes['class'])? $attributes['class'] .= '
first' : 'first');
      } elseif($item_key == count($items)-1){
        $attributes['class'] = (isset($attributes['class'])? $attributes['class'] .= '
last' : 'last');
      }

      $output .= '<li' . drupal_attributes($attributes) . '>'. $data .'</li>';
    }
    $output .= "</$type>";
  }
  $output .= '</div>';
  return $output;
}
?>
```

Bonus: Zebra

Add the following before `$output .= '<li' . drupal_attributes($attributes) . '>'. $data .'</li>';` to add a zebra class to alternate list items (not the php tags).

```
<?php
if ($item_key % 2) { $attributes['class'] = (isset($attributes['class'])?
$attributes['class'] .= ' zebra' : $attributes['class'] = 'zebra'); }
?>
```

HowTo: Have "first" and "last" classes on menu blocks

Often, one wants to style the first and last items in a list differently than the rest. CSS 2 provides a simple selector to do that dynamically but sadly it's not supported by Internet Explorer so most designers add a "first" and "last" class to list items where needed so that they can target them with CSS definitions.

Drupal 5 added automatically-generated "first" and "last" classes to primary and secondary links (yay!), but unfortunately did not do so for menus displayed in blocks (boo!). Fortunately, the theme system allows us to do so ourselves.

The menu is actually generated by the `menu_tree()` function, which is not overridable by a theme.

However, that function is itself called from a theme function, so we can override that. To do so, copy and paste the following code into your template.php file:

```
<?php
function phptemplate_menu_tree($pid = 1) {
  if ($tree = phptemplate_menu_tree_improved($pid)) {
    return "\n<ul class=\"menu\">\n". $tree ."\n</ul>\n";
  }
}

function phptemplate_menu_tree_improved($pid = 1) {
  $menu = menu_get_menu();
  $output = '';

  if (isset($menu['visible'][$pid]) && $menu['visible'][$pid]['children']) {
    $num_children = count($menu['visible'][$pid]['children']);
    for ($i=0; $i < $num_children; ++$i) {
      $mid = $menu['visible'][$pid]['children'][$i];
      $type = isset($menu['visible'][$mid]['type']) ? $menu['visible'][$mid]['type'] :
NULL;
      $children = isset($menu['visible'][$mid]['children']) ? $menu['visible'][$mid]
['children'] : NULL;
      $extraclass = $i == 0 ? 'first' : ($i == $num_children-1 ? 'last' : '');
      $output .= theme('menu_item', $mid, menu_in_active_trail($mid) || ($type &
MENU_EXPANDED) ? theme('menu_tree', $mid) : '', count($children) == 0,
$extraclass);
    }
  }

  return $output;
}

function phptemplate_menu_item($mid, $children = '', $leaf = TRUE, $extraclass = '') {
  return '<li class="'. ($leaf ? 'leaf' : ($children ? 'expanded' : 'collapsed')) .
($extraclass ? ' ' . $extraclass : '') . '">'. menu_item_link($mid, TRUE, $extraclass)
. $children ."</li>\n";
}
?>
```

The first function differs from the default only in that it directs the system to the alternate version of `menu_tree()`, `phptemplate_menu_tree_improved()`. That function works exactly like its core counterpart except that it passes an extra class to our alternate menu item theme function. (Yes, you can add parameters to a theme override function.) `phptemplate_menu_item()` simply tacks on the extra class if specified, or does nothing different if it isn't. Voila, first and last classes on any menu block.

Note that this code adds the first/last class to the list item, not to the link itself. That is how Drupal generally handles such classes, as it allows a CSS rule to target either the list item or the link, depending on which is needed, while putting the class on the `<a>` element itself would only allow a themer to style the link, not the list item.

Single menu items

Sometimes you have a (sub)menu with only one item. The above code only adds the class "first" to a menu item, regardless of the fact if it is the first item among many, or the first item of only one. This may break a theme, therefore there are 2 possibilities to cover the case, that there might only be one menu item:

#1:
Add this code
`$extraclass .= $num_children == 1 ? ' last' : '';`

below
```
$extraclass = $i == 0 ? 'first' : ($i == $num_children-1 ? 'last' : '');
```
This will just add a "last" class, so you might have a menu item with the class "first last".

#2:
Add this code
```
$extraclass = $num_children == 1 ? 'single-item' : $extraclass;
```
below
```
$extraclass = $i == 0 ? 'first' : ($i == $num_children-1 ? 'last' : '');
```
This will add a class "single-item", instead of "first".

Isolate a specific menu item with $mid

This is an additional tweak to the HowTo: Have "first" and "last" classes on menu blocks code.

Sometimes I have the need to style just a single item in a menu (use a different color or background for that item specifically). By adding the $mid into the class array for each list item, I have complete control over this with CSS.

My modified function looks like this:

```
<?php
function phptemplate_menu_item($mid, $children = '', $leaf = TRUE, $extraclass = '') {
  return '<li class="'. ($leaf ? 'leaf' : ($children ? 'expanded' : 'collapsed')) .
($extraclass ? ' ' . $extraclass : '') . ' mid-' . $mid . '">'. menu_item_link($mid,
TRUE, $extraclass) . $children ."</li>\n";
}
?>
```

All I added is ". ' mid-' . $mid", and didn't bother with conditionals since $mid is always defined.

In the CSS I use a selector like #header .mid-123 to isolate the menu item as used in the header, etc. Works great.

HowTo: Hide the node title on a page

You can hide node titles by specific node type by adding some code to your template.php file.

Here's the snippet, in this example it is ignoring the title on any node that is of the type "page" or "story". Note that we're also saving the title to another variable for use in breadcrumbs if desired.

```
<?php
//titles are now ignored by specific node type when they are anomalous in the design
$vars['breadcrumb_title'] = $vars['title'];
if (arg(0) == 'node' && is_numeric(arg(1))) {
  $node = node_load(arg(1));
  if (in_array($node->type, array('page', 'story'))) {
    $vars['title'] = '';
  }
}
?>
```

Here's an example with the code where it belongs in the _phptemplate_variables() function

```
<?php
function _phptemplate_variables($hook, $vars = array()) {
  switch ($hook) {
    case 'page':
    //titles are now ignored by specific node type when they are anomalous in the
design
    $vars['breadcrumb_title'] = $vars['title'];
    if (arg(0) == 'node' && is_numeric(arg(1))) {
      $node = node_load(arg(1));
      if (in_array($node->type, array('page', 'story'))) {
        $vars['title'] = '';
      }
    }
    break;
  }

  return $vars;
}
?>
```

HowTo: Replace node title with a related image

I took over a static site, designed with Dreamweaver. The site design and layout were beautiful, and one of my goals for converting the site to Drupal was to maintain as much of the site's visual appeal as possible. One of the visual elements were sharp graphic titles on each page. And so began my quest to figure out how to replace drupal's text title with these image versions. I tried several methods before coming across the excellent article above (this page's parent), which gave me the following idea (much of the code is lifted directly from that article).

Overview: basic concept

A node on your site has a title, like "Bob's Antiques & Junk", and you want to display the related image file "bobs_antiques_junk.gif" *instead of* this text title when displaying the node as a page. Furthermore, you want the original title as alt text on the image, and you want "graceful degradation", so the original text title is displayed if the "related image" file does not exist.
What this does is essentially make the Title field of the nodes into an special image selector - if you use a title with an associated image, the image will appear, otherwise, the text title will appear - sweet!

Requirements:

- You are using a template theme.
- You are not averse to a little php.

All code goes into your template.php file.
NOTE: Ignore the opening and closing PHP tags in here. They are just for display and should not be added to your file.
For background info on template.php and the phptemplae_variables function, see:

- template.php: Overriding other theme functions http://drupal.org/node/11811
- Adding function call to _phptemplate_variables http://drupal.org/node/207841
- HowTo: merge multiple _phptemplate_variables functions http://drupal.org/node/152426

Step 1: convert title to related filename

```
<?php
// Prepare some general title text for use as a file name.
// Remove special HTML characters, trim whitespace, convert to lower-case
// repace spaces with underscores.
function clean_title($string)
{
    $cleanString = htmlspecialchars_decode( strtolower(trim($string)), ENT_QUOTES );
    $cleanString = str_replace(array("& ", "'"), '', $cleanString);
    return str_replace(' ', '_', $cleanString );;

}
?>
```

This handy little function will take a title, like "Bob's Antiques & Junk" and convert it to a nice base filename: "bobs_antique_junk". Since Drupal will convert quotes and ampersands to "special characters" (e.g., $#039; or $amp;), it strips those out first. Then does a simple replace to get rid of apostrophe's and ampersands - common in titles. Finally, it replaces remaining spaces with underscores.
This function can be customized to convert titles to filenames however you see fit. Depending on the titles on your site, you may want to extend the substitutions made here. For example, use a regex to match and replace patterns (e.g., preg_replace("/[^a-zA-Z0-9]/", "", $string)) - this is a little slower, but more general purpose.

Step 2: making the substitution - code snippets

We need both the file system path (to check if the file exists) and the URL (to form the image tag) for the title image. Note: substitute the correct extension and path to your title images - mine are '.gif' files that live in "files/images/titles".

```
<?php
        // convert title to suitable filename and derive both file system path and
URL.
        $titleFile = clean_title($vars['title']) . '.gif';
        $titleImage = base_path() . "files/images/titles/" . $titleFile;
        $titleURL = "http://" . $_SERVER['SERVER_NAME'] . $titleImage;
        $titlePath = $_SERVER['DOCUMENT_ROOT'] . $titleImage;
?>
```

Only perform the substitution if the "related image" can be found. Use $vars['title'] as the alt text for the image tag, and then put the image tag back in to $vars['title'].

```
<?php
        // Determine if a suitable replacement image for title can be found, and make
substitution.
        if ( file_exists($titlePath) ) {
          $newTitle = '<img alt=\'' . $vars['title'] .'\' src=\'' . $titleURL .'\'
/>';
          $vars['title'] = $newTitle;
        }
      }
?>
```

A few refinements:
Restrict application of the substitution - see the parent article, which explains this nicely.

```
<?php
    // Substitute title only for nodes...
    if (arg(0) == 'node' && is_numeric(arg(1))) {
      $node = node_load(arg(1));
      // ... of one of the listed types.
      if (in_array($node->type, array('page'))) {
```

```
        // convert title to suitable filename and derive both file system path and
URL.
        ...
?>
```

Save original title text for use in head and breadcrumb.

```
<?php
    $vars['breadcrumb_title'] = $vars['title'];
    $vars['head_title'] = $vars['title'];
?>
```

Step 3: putting it all together

```
<?php
function _phptemplate_variables($hook, $vars = array()) {
  switch ($hook) {
    case 'page':
    // substitute node title with an image, if a suitable replacement can be found.
    // save original title text for use in head and breadcrumb.
    $vars['breadcrumb_title'] = $vars['title'];
    $vars['head_title'] = $vars['title'];
    // Substitute title only for nodes...
    if (arg(0) == 'node' && is_numeric(arg(1))) {
      $node = node_load(arg(1));  // (expensive, unless arg(1) is already loaded,
which is should be at this point.)
      // ... of one of the listed types - add node types to process to the array.
      if (in_array($node->type, array('page'))) {
        // convert title to suitable filename and derive both file system path and
URL.
        $titleFile = clean_title($vars['title']) . '.gif';
        $titleImage = base_path() . "files/images/titles/" . $titleFile;
        $titleURL = "http://" . $_SERVER['SERVER_NAME'] . $titleImage;
        $titlePath = $_SERVER['DOCUMENT_ROOT'] . $titleImage;
        // $vars['title'] = $titleURL;   // DEBUG - see what's being produced

        // Determine if a suitable replacement image for title can be found, and make
substitution.
        if ( file_exists($titlePath) ) {
          $newTitle = '<img alt=\'' . $vars['title'] .'\' src=\'' . $titleURL .'\'
/>';
          $vars['title'] = $newTitle;
        }
      }
    }
    break;
  }

  return $vars;
}
?>
```

I hope this helps someone - and if anyone with better PHP skills and Drupal knowledge than I (which means most ;-), has some suggestions for improving this script, please let me know.

HowTo: horizontal login block in the footer

It's quite simple to restyle the standard user login block so it appears all on one line. You can then put it your footer, and it's out of the way on sites where most users don't need to (or can't) log in.

Here's the code for your stylesheet:

```
/**
* user login block in footer
*/
#block-user-0 {
margin: 2em 0 1em;
}
#block-user-0 *
{
display: inline;
color: white; /* match the text colour of your theme here */
}
#edit-user-login-block {
display:none; /* rehide hidden form element */
}
#block-user-0 h2 {
font-size: 92%; /* match .content font size... change this for your theme */
}
#block-user-0 .form-item {
margin-left: 1em;
}
```

HowTo: merge multiple _phptemplate_variables functions

Many PHPTemplate theme snippets, and some contributed modules, require you to add a _phptemplate_variables function to your theme's template.php file. Unfortunately, if you want to use more than one such module or snippet, or if your theme already contains a _phptemplate_variables function, you can't simply add another one - you have to merge the two functions. If you don't know PHP very well, this may seem like a daunting task.

Here is a simple recipe for merging multiple _phptemplate_variables functions into one. You don't need to know PHP; just copy and paste, and change some names as described below.

1. Put all the necessary _phptemplate_variables functions into your template.php file, and change all their names from _phptemplate_variables to something else. For example, _phptemplate_variables_taxonomy_snippet, _phptemplate_variables_page_title_module etc. If there's already a _phptemplate_variables function in the theme, change its name too - e.g. _phptemplate_variables_garland_theme. Make sure not to give any two functions the same name.
2. Create a new _phptemplate_variables function. This will be the master function that will combine all the variables from the other functions. Here is what it should look like:

```
<?php
/**
* Call all our custom _phptemplate_variables_* functions and merge the results
into a single array.
*/
function _phptemplate_variables($hook, $vars) {
  $funcs = array(
    // These are the names of our custom functions.  Add more below as needed.
    '_phptemplate_variables_taxonomy_snippet',
    '_phptemplate_variables_page_title_module',
    '_phptemplate_variables_garland_theme',
  );
  foreach ($funcs as $func) {
```

```
    $vars = array_merge($vars, $func($hook, $vars));
  }
  return $vars;
}
?>
```

Copy the above code into your template.php and change the function names to the ones you actually used.

3. Later, if you want to add yet another _phptemplate_variables function, you can easily do so by giving it a new name, and adding that name to the list.

HowTo: Theme a CCK input form

(Originally posted at http://drupal.org/node/98253)

The reason I have come up with this step by step overview of themeing input forms was that I was working with CCK created content types myself and was trying to figure out how to change the order of the fields of the input form. What I found in the forums was very technical (pointers to the API in 'developer speak') or incomplete, or simply did not cover input forms. There was a lot of help for people looking to modify/theme output, but nothing clear and concise for theming input forms.

So, as a non-developer, semi-technical, marketing/business type person, I set out to discover how to 'theme' my input forms. (All those developers and others that are better informed than I, please feel free to correct me where I am wrong!) I found there are at least two different ways to theme input forms:

1. Using the Form API (creating new .module files programmed in php and using things like 'hook_form_alter')
2. Creating tpl.php files

I have found that it is far easier to use the second method. You actually end up having to do a little more than create a custom tpl.php file - you have to modify template.php, which requires a little bit of PHP knowledge as well as work with stylesheets (either edit style.css or create a new one and link it to your theme - more on this later).

First off, there are a couple people who are responsible for the bulk of the content in the tutorial and deserve proper credit: Lyal (http://www.harostreetmedia.com) - thanks for emailing code snippets in the middle of the night. Dublin Drupaller (http://www.dublindrupaller.com) - thanks for the rich depth of knowledge you share with others on the forums from which I have gleaned and repurposed much. jghyde - who has helped out others on this very same quest. Nick Lewis (http://www.nicklewis.org) - whose website is full of helpful and insightful Drupal information.

Now on with the show.

How do you theme a CCK input form?

Assumptions

- You understand a little CSS (how it works, how you use class= and id= in HTML tags to style the output)
- You understand a little PHP (nothing serious, at least the ability to understand what a PHP code snippet is doing so you can copy it and modify it a little)
- You have CCK installed and know how to create content types with it (although I am referring to

CCK input forms in this tutorial, I am pretty sure these are the same steps you would use for any custom node type)

Steps

Step 1: Create your CCK content type
Step 2: Create a [yourcontenttype].tpl.php file
Step 3: Modify template.php
Step 4: Modify style.css

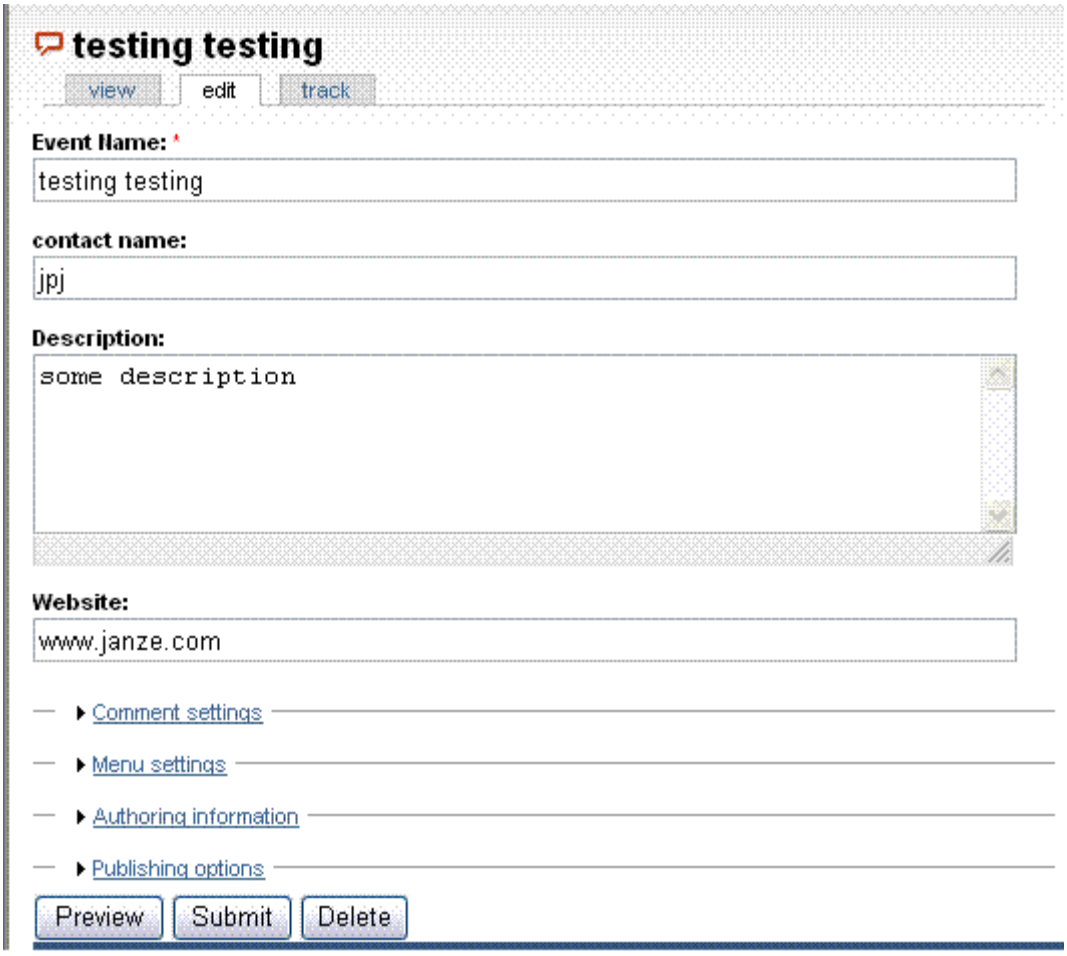
You should be able to modify the input form, pretty much anyway you want with this method. For this example, I am going to do the following: Not show some of the automatically generated 'location' fields (I have 'location' module installed and enabled for my CCK content type), then create my own 'collapsible' group and put the elements that I want to in that group, and put some of the input fields side by side instead of each one being on a new line.

Even with the method I am using, there are different ways to accomplish the same thing; for example, I am using tpl.php to order the fields and .css to style them. I think I could just as easily use .css to do everything (but I am no CSS guru) and only use the custom tpl.php file to assign new `<div>`'s.

Step 1: Create your CCK content type

Create a CCK content type called 'event_listing' and add some fields to it. I have added contact information, location, event information fields etc.

Create at least one node (create an event listing with your new content type).



Step 2: Create a [yourcontenttype].tpl.php file

Now create a text file called 'event_listing_edit.tpl.php' and put it in the directory of your selected theme (which is somewhere under the '/theme/' directory. For example, my template.php file is found here: /theme /internet_jobs/). You could call this pretty much anything, but it will make a lot more sense and be easier to manage if you call it something like this.

Now add the following PHP code snippet to the tpl.php file - this is simply sample text to prove that your 'next step' (modifying template.php) is actually working.

```
<?php
print("yahoo, it works!");
?>
```

Step 3: Modify template.php

Now the hard part - we need to open and edit your template.php file which is again in the directory of your selected theme (in my case it is located here: /theme/internet_jobs/template.php)

You will need to add two seperate functions in this file, one to call your custom event_listing_edit.tpl.php file when the form is being edited and another one to call it when a NEW node is to be created (you only need one tpl.php file, but as the template.php file modifications are overriding default functionality, it is necessary to instruct drupal to use the custom file for each case.

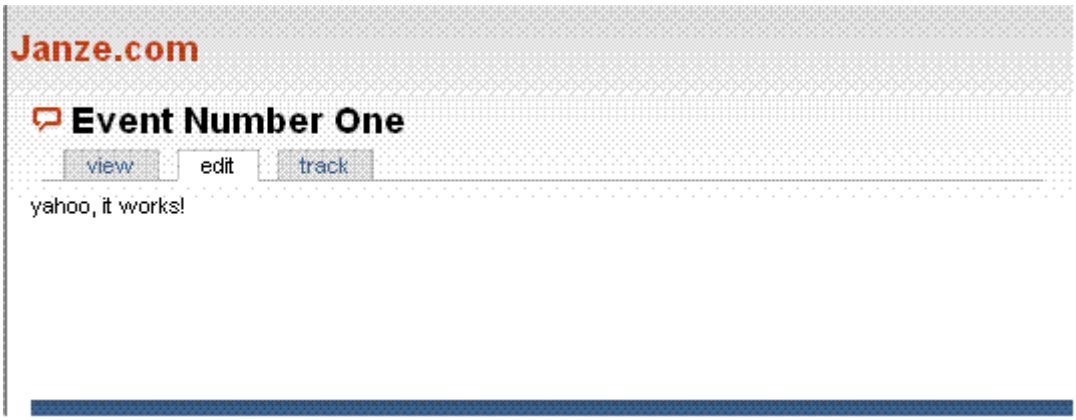
At the end of the template.php file, add the following snippets which you should be able to see is two seperate override functions:

```
<?php
// Add Form Start (jghyde)
if ((arg(0) == 'node') && (arg(1) == 'add') && (arg(2) == 'content_event_listing')){
  function phptemplate_node_form($form) {
    return _phptemplate_callback('event_listing_edit', array('user' => $user, 'form'
=> $form));
  }
}
// Add Form End

// Edit Form Start (Dublin Drupaller)
if ((arg(0) == 'node') && (arg(2) == 'edit')){
  $node = node_load(array('nid' => arg(1)));
  function phptemplate_node_form($form) {
    return _phptemplate_callback('event_listing_edit', array('user' => $user, 'form'
=> $form));
  }
}
// Edit Form End
?>
```

You can see from the snippets who actually wrote them (although I have modified them slightly from their original) - thank you jghyde and Dublin Drupaller (http://drupal.org/node/85908).

Now try to edit your event_listing node you should see the text 'yahoo, it works' on the page - see screenshot. If you don't get the 'yahoo' message, there a couple of 'suspects' to look at: go check out your naming convention in the template.php file, I tripped up and referred to my tpl.php file incorrectly a few times before I got it right.



The next step is to add all the fields you want back onto the input form. You can find out what fields are available to your form by putting the following line of code in your custom tpl.php file (Tip from goose2000 for outputting data into a nicer format)

```
<?php
print("yahoo, it works!<br>");
print("<pre>");
print_r(array_values($form));
print("</pre>");
?>
```

You will see a great big long list of information describing the fields available to your input form, the part you are interested in looks like this (there will be a section like this for every field and object on the form):

```
[field_website] => Array
  (
    [0] => Array
      (
        [value] => Array
          (
            [link] => www.janze.com
            [title] => Janze.com
            [attributes] =>
          )
      )
  )
```

In this example, if I want to refer to the website URL, I would refer to it like this:

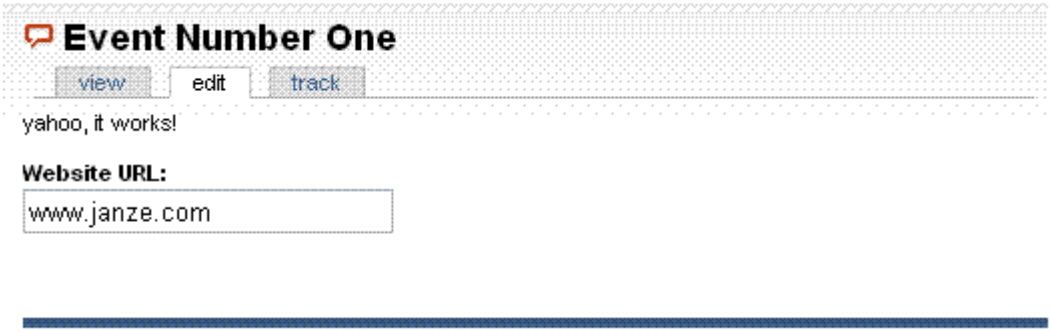
```
<?php
['field_website']['0']['value']['link']
?>
```

. So if I want to add the website URL to my input form I would add the following line to my tpl.php file:

```
<div id="jpj8"><?php print form_render($form['field_website']['0']['value']['link']);
?></div>
```

(I have wrapped the line in a 'div id=' tag so I can style it using CSS)

Now if I edit my event_listing node, I will get something that looks like this:



Don't forget to delete the test line that adds 'yahoo, it works!'.

Now go ahead and add the rest of the fields you want to the form.

If you want to make a group of fields that collapse, all you have to do is wrap those particular fields with `<fieldset class=" collapsible"></fieldset>` and if you want the Group to have a name add `<legend>Contact Info</legend>` right after the opening fieldset. Here is an example:

```
<div class="jpj8aa">
  <fieldset class=" collapsible">
    <legend>Contact Info</legend>
    <div id="jpj8"><?php print form_render($form['field_name']); ?></div>
    <div id="jpj8a"><?php print form_render($form['field_email']); ?></div>
    <div id="jpj8b"><?php print form_render($form['field_phone_0']); ?></div>
    <div id="jpj8c"><?php print form_render($form['field_website']['0']['value']
['link']); ?></div>
    <div id="jpj8e"><?php print form_render($form['field_description']); ?></div>
  </fieldset>
</div>
```

This will look like:

Interesting bits

It is a bit odd, but at the end of your tpl.php file you put the

```
<?php
print form_render($form);
?>
```

line which will render all remaining form objects (fields and buttons) that have not previously been referred to. That means it will now display all fields that you have not explicitly added in previous lines of code. Now, if you are like me, the reason you didn't explicity add them to the form is because you DON'T want them to be there!

So, to get rid of the fields that have now been automatically added back in, I have found two methods, one that worked for me (that I am sure is 'wrong' because it isn't very elegant) and another that is very elegant, but didnt' work for me. I would definitely try the 'elegant' route first.....which is, you have to override unwanted fields explicity in your template.php file. For example, if I do not want to display the website, I put this text

```
<?php
$form['field_website'] = '';
?>
```

in the template.php file, inside your override function (before the callback). Reading this line of code shows us that we identify the field on the form and set its value to '' - nothing. Here is what the function would look like:

```
<?php
// Add Form Start (jghyde)
if ((arg(0) == 'node') && (arg(1) == 'add') && (arg(2) == 'content_event_listing')){
  function phptemplate_node_form($form) {
    $form['field_website'] = '';
    return _phptemplate_callback('event_listing_edit', array('user' => $user, 'form'
=> $form));
  }
}
// Add Form End

// Edit Form Start (Dublin Drupaller)
if ((arg(0) == 'node') && (arg(2) == 'edit')){
  $node = node_load(array('nid' => arg(1)));
  if ($node->type == 'content_event_listing'){
    function phptemplate_node_form($form) {
      $form['field_website'] = '';
      return _phptemplate_callback('event_listing_edit', array('user' => $user, 'form'
=> $form));
    }
  }
}
// Edit Form End
?>
```

Note that I had to put the code snippet into BOTH the 'Add form' and 'Edit form' sections that I newly added.

The 'not so elegant' way to do it is to add the unwanted fields to your tpl.php file, just like the wanted ones and then simply use CSS 'display: none;' to style it away (see below).

Step 4: Modify style.css

OK, now you have a form with all the fields you want on it, none of the ones you don't want and you may even have grouped some of them together into a collapsible group, now what? Now you can style each of those elements using CSS. I am no CSS guru, so I am only doing a few simple things here: 1) I am showing an example of using CSS to hide a field, and 2) I am also putting two fields side by side.

To use CSS to hide a field, simply identify the `<div>` ID of the field that you want to hide, then create a style in your style.css file for that `<div>` (or in your custom, linked css file - I think it is recommended that you actually use your own CSS file so that any modifications you make will survive an upgrade).

Here is an example:

```
#jpj10 {
  display: none;
}
```

To use CSS to put to fields side by side, you need to wrap each field that you want side by side in `<span>` tags instead of `<div>` tags like this:

```
<span id="jpj8b"><?php print form_render($form['field_phone']); ?></span>
<span id="jpj8c"><?php print form_render($form['field_website']['0']['value']
['link']); ?></span>
```

Then create styles for them to float left like this:

```
#jpj8b, #jpj8c {
```

```
  float: left;
}
```

Now my two fields will display side by side - I leave it to you to make it look better - this is just the mechanics of getting it to work!

Additional information

FYI, for those playing around with **Drupal 5.x;** replace form_render() with drupal_render() - thanks to ericelbow.

HowTo: XHTML/CSS Prototyping with Drupal

XHTML/CSS Prototypes are a professional approach to solve client side display issues apart from content and application logic. This can be normally done by using static files. Since creating static files first and then converting the prototype into a Drupal Theme might work as well, it can make sense to integrate both steps with each other. This HowTo shows some settings and hints on how XHTML/CSS prototyping can be done directly in Drupal.

In general you need to enter XHTML content unfiltered for doing prototyping while using a CSS based theme like ZEN-CSS. In the standard setup Drupal filters output by adding additional XHTML Elements and removing others. This is unwanted in our case sothat Drupal has to be configured to allow the user to input XHTML unfiltered.

Adding the *AS-IS HTML* Input format

This can be achieved by adding an Input format. This is how you can create a new *Input Format* named **AS-IS HTML** that will not filter the output any longer:

1. Log in your site as administrator
2. Navigate to Administer › Site configuration › Input formats
3. Press **Add input format**
4. You are now on a page entitled **Add input format**
5. Enter `AS-IS HTML` in the *Name* textfield.
6. Ensure that any checkbox remains unchecked in both *frames* **Roles** and **Filters**.
7. Submit the Form by pressing the **Save configuration** *Button*.

Now a new *Input Format* should have been created. You should be redicted to the *page* **Input format** and in the new name should be listed.

Using unfiltered XHTML in Themes

This input format can now be used to output XHTML as needed through Druapl sothat it can be used while creating a theme. You can use it with many content types and you can use it while adding blocks. The only thing you need to do is to allow the user which is doing the theme-work to use this filter.

Any User who is allowed to **administer filters** can use all filters directly, so therefore the administrator can directly. If you create a user only for XHTML/CSS-prototyping you should add this permission to the role.

Do the prototyping

Now the user who is doing the prototyping can create content and blocks with this filter and do the theming work. This should lead to a faster theme development while keeping things apart from development. Additionally this shortcut will help developers and themers to understand shortcommings.

Look and feel

The following articles provide general information about altering the look and feel of your site using styles and JavaScript.

For information about specific website implementations, see the Site Recipes section.

For information about themes, see the Theme Developers' Guide

Adding a collapsible fieldset to your nodes

Editor's note: *This page attempts to demonstrate how to make a collapsible form fieldset appear on a page without actually using a form. While the results are pretty, semi-functional, and even slightly cool, this method is a no-no for semantics, usability, and even breaks the basic rules of HTML. A better way to do this would be to use jQuery to toggle the collapse/expand of a styled div. When someone posts a HowTo on doing exactly that, this page will be deleted and redirected to the better, safer jQuery method.*

This method has not been tested in Drupal version 5.x, but will most likely work.

The Collapsible Fieldset is a drop-down area you see underneath a titled section. For example, on the Settings page (Administer » Settings) you see the titled sections such as:

- General settings
- Error handling
- Cache settings
- etc...

When you mouse click on any of these titles, the area expands below displaying the information corresponding to the section.

I found this feature useful to incorporate on node pages that are long. Instead of using a page break, a teaser, or read more option, you can use the Expandable / Collapsible Fieldset Feature. This method also works well when you want to initially display half of a table.

Want a demonstration? Click either of the 'read more' links on this web page.

This example uses the Multiple Expandable / Collapsible Fieldsets as explained in the second section below.

To use the Collapsible Fieldset feature, do the following:

1. Copy the following CSS code to a new file and save it as **collapse.css**. This code was extracted from the original drupal.css file version 4.7.5 and should work with later drupal versions. It was modified to not display the fieldset border by adding the "Removes Fieldset Border" CSS parameters shown in the first lines of code.
   collapse.css file

```
/*
* $Id: Copied from the drupal.css file,v 1.147.2.8 2006/12/14 20:20:21 killes Exp
$
*
* Collapsing fieldsets
*/

/* Removes Fieldset Border */
html.js fieldset {
border: 0; padding: 0;
}

html.js fieldset.collapsed {
  border-bottom-width: 0;
  border-left-width: 0;
  border-right-width: 0;
  margin-bottom: 0;
}

html.js fieldset.collapsed * {
  display: none;
}

html.js fieldset.collapsed table *,
html.js fieldset.collapsed legend,
html.js fieldset.collapsed legend * {
  display: inline;
}

html.js fieldset.collapsible legend a {
  padding-left: 15px;
  background: url(menu-expanded.png) 5px 50% no-repeat;
}

html.js fieldset.collapsed legend a {
  background-image: url(menu-collapsed.png);
}

/* Note: IE-only fix due to '* html' (breaks Konqueror otherwise). */
* html.js fieldset.collapsible legend a {
}
```

2. Upload the collapse.css file to your site's /misc/ directory. (This is the same directory that contains the drupal.css file.) *Note: the collapse.js file is a core file and is already in your /misc/ directory.*
3. Copy the following code to your node page.

   Single Expandable / Collapsible Fieldset:

```
<link rel=stylesheet type="text/css" href="/misc/collapse.css">
<script type="text/javascript" src="/misc/collapse.js"></script>
```

```
******* Enter the first part of your text here. *******

<form>
<fieldset class="collapsible collapsed">
<legend>More Information</legend>
<div>

******* Enter the last part of your text your text here. *******

</div>
</fieldset>
</form>
```

4. Replace the:

   ******* Enter the first part of your text here. *******

   with the first half of your text.

5. Next, replace the:

   ******* Enter the last part of your text your text here. *******

   with the remaining last half of the text that will be initially hidden. This is the text section that is expandable and collapsible.

To use multiple Expandable / Collapsible Fieldset, use the following code:

Multiple Expandable / Collapsible Fieldsets:

```
<link rel=stylesheet type="text/css" href="/misc/collapse.css">
<script type="text/javascript" src="/misc/collapse.js"></script>

******* Enter the first part of your text here. *******

<!-- *** Begin collapsible 1 *** -->
<form>
<fieldset class="collapsible collapsed">
<legend>Read More &gt;&gt; Section 1</legend>
<div>

******* Enter the second part of your text your text here. *******

</div>
</fieldset>
</form>
<!-- *** End collapsible 1 *** -->

<!-- *** Begin collapsible 2 *** -->
<form>
<fieldset class="collapsible collapsed">
<legend>Read More &gt;&gt; Section 2</legend>
<div>

******* Enter the third (last) part of your text your text here. *******

</div>
</fieldset>
</form>
<!-- *** End collapsible 2 *** -->
```

That's it!

Change the size of a CCK input field

Overriding the default width of a CCK input field is best done within a custom module. Use the hook_form_alter() function to target your form's ID. If you don't know the ID of the form you want, use `<?php print($form_id); ?>` at the top of your custom module, and Drupal will show you the ID of all the forms it knows about. Now that you have the form ID, use form_alter to change the #size element of your CCK field:

```
<?php
function nameOfYourModuleGoesHere_form_alter($form_id, &$form) {
  if ($form_id == 'idOfYourFormGoesHere') {
    $form['nameOfYourCCKfieldGroup']['nameOfYourCCKinputField']['#size'] = 40;
  }
}
?>
```

Use `<?php print_r($form); ?>` if you need to see those CCKfieldGroup names or CCKinputField names.

Setting the #size attribute of the form field in this way is better than using CSS to override the width of a form's input class, since CSS won't target the "file" fields in a form due to browser security restrictions. This form_alter method has the added bonus of working on forms when CSS support is disabled, such as some of those early handheld browsers. Enjoy!

For the Zen Theme

Add this to your sub-theme template.tpl.php:

```
<?php
function zen_file($element) {
  _form_set_class($element, array('form-file'));
  return theme('form_element', $element, '<input type="file" name="'.
$element['#name'] .'"'. ($element['#attributes'] ? ' '.
drupal_attributes($element['#attributes']) : '') .' id="'. $element['#id'] .'"
size=\"40\"' ."\" />\n");
}
?>
```

CSS Tips, Tricks, and Techniques

In this section of the handbook you can show off your favorite tips, tricks, and techniques to make your site pretty with nothing but CSS. If your tip is theme-specific, be sure to specify to which theme it applies.

These are to be CSS-only tips, please do not include anything that also requires any coding in a page or in a module. There is a modules snippets section for those tips.

Feel free to add new tips or to enhance ones that are already posted. In fact, please try to reduce duplication as much as possible. We'd rather improve an existing page, then add another one.

PLEASE NOTE! The following tips are user-submitted. Apply them at your own risk! In general, CSS will

not "break" your site, but it can cause unexpected formatting or lack of user control.

If you are developing a brand new theme, see the Theme developer's guide.

Javascript snippets can be found in the Javascript section.

Block quotes

If you would like to display multi-paragraph quotes with quotation marks as characters, rather than using images, you could use this;-

```
blockquote p::before {
  content: open-quote;
}

blockquote p::after {
  content: close-quote;
  visibility: hidden;
}

blockquote p:last-child::after {
  visibility: visible;
}
```

Of course, you can add indenting *etc.* as required.

Markup is simply;–

```
<blockquote>
  <p>paragraph one</p>
  <p>paragraph two</p>
  <p>paragraph three</p>
  <p>paragraph four</p>
</blockquote>
```

NB Only tested on Firefox.

> Happy quoting.

Highlight the expandable section of a menu

This tip changes the background of an expanded menu section, highlighting that you are in it. It works really well with my tip on highlighting the actual item.

Another thing I do here is "shrink" the menu item font size, but slightly increase the size of the parent item when it is expanded. I also set the background color of the whole menu<.p>

```
.menu .leaf {font-size:0.9em; background:#FFDAB9;}
.menu .leaf .active {border:1px solid #D2691E;}
.menu .collapsed {font-size:0.9em; background:#FFDAB9;}
.menu .expanded {font-size:1.0em; background:#FFEBCD;}
#sidebar-left, #sidebar-right {background-color: #FFDAB9;}
```

How to use style.css to theme CCK

Step 1. Find out the cck field names you want to theme:
/admin/content/types/fields
For example, I want to define the following two field names:

field_author
field_photo

Step 2. In style.css, at end add:

```
.field-field-author {
color: blue;
}

.field-field-photo {
float: right;
}
```

You can also define field type.
For example, I define all my "text" type:

```
.field-type-text{
font-style: italic;
}
```

For example, I define all my "image" type:

```
.field-type-image{
margin-top:1em;
}
```

Iconification : adding UI icons using (mostly) CSS

It is currently fairly easy to insert icons into the interface of your Drupal5 site using only CSS. However, there are one or two pitfalls and limitations along the way, which I hope to try and document. As ever, this is a work in progress, so please do share your thoughts and ideas by adding comments, or even new pages!

Important note for iconification of Garland themes: If you have color.module enabled, do not add your iconification CSS code to *style.css*! Otherwise, when you resubmit your color.module settings, color.module redirects all the icon file URL paths off into to the files/color/#colorschemecode/icons folder :-P If your icon images are very theme-specific (ie they only work within a given colour-scheme) this might be an advantage, but my approach is to try and use more generic icons in order to save lots of work :-) So ...

Groundwork: We need to do some filesystem groundwork before we get going - it's very easy, and the advantage of creating new (custom) files and folders is that they will not get overwritten when you update your Drupal installation files.

In your theme folder (eg Drupal5/Themes/Garland/icons.css):

1. create a new file called **icons.css:**. this keeps (almost) all our iconification code in one place.

2. next, edit the *page.tpl.php* file to append the following line of code to the style header calls:

```
<style type="text/css" media="all">@import "<?php print base_path() . path_to_theme()
?>/icons.css";</style>
```

3. create a new *folder* called 'icons' inside your theme folder (eg Drupal5/Themes/Garland/icons) to keep all your icon images together, and your filepaths simple.

adding icons to module operator links

Prepending icons to module function labels is very easy to do with simple CSS. Because the CSS hooks are created by module functions they are relatively theme-safe. But this only works if the module is generating proper CSS hooks. If a particular module is not doing this properly, the best thing to do is raise it in that module's issue queue.

For this example, the comment.module generates a link to 'Add new comment'. the HTML looks like this:

```
<div class="links">
  <ul class="links inline">
    <li class="first last comment_add">
      <a href="/test_site/comment/reply/62#comment-form" title="Share your thoughts
and opinions related to this posting." class="comment_add">Add new comment</a>
    </li>
  </ul>
</div>
```

All we need is the *comment_add* class attribute (which happens to be attributed to a list item). So all we do is add the following code to our *icons.css* file:

```
/** prefix 'Add new comment' link with icon **/
.comment_add a {
    padding-left: 20px;
    padding-bottom: 2px;
    background:url(icons/comment.png)  no-repeat;
}
```

If you want to show the icon but hide the 'Add new comment' text, you can use this code instead, but beware that a CSS *block* gets an automatic linebreak before and after the element (so it doesn't work well on $links ...)

```
/** replace 'Add new comment' link with icon **/
.comment_add a {
    display: block;
    text-indent: -10000px;
    background:url(icons/comment.png)  no-repeat;
}
```

This works for most of the iconficiation you are likely to need, as long as you can find the right CSS tag (be it a class or and id attribute) for the label, you can re-use the code snippets editing only the `.comment_add a` and `comment.png` references to suite.

prefixing Block Titles with an icon

Unfortunately there's not currently a simple `<div class="block_x_title">` tag, but we can target the block title by interpreting contextual tags. Currently Drupal is quite strict about generating consistent tags for blocks so this method should be fairly reliable!

All blocks have a **class** attribute denoting the module that generated the block, so specifying the class will apply the same icon to all the blocks generated by that module. For example, both *Login* and *Navigation* blocks come from the (core) user.module :

```
/** iconify Navigation & Log-in block titles **/
.block-user h2 {
    padding-left: 20px;
    background:url(icons/user.png)  no-repeat;
    background-position: center left;
}
```

NB: in order to work `background-position: ... ;` must follow `no-repeat;`.

Some modules (eg *Views*!) create more than one block, so the Block **id** attribute includes a $delta (eg id="block-user-0") which will specify different icons for individual blocks :

```
/** iconify Log-in block title only **/
#block-user-0 h2 {
    padding-left: 20px;
    background:url(icons/login.png)  no-repeat;
    background-position: center left;
}

/** iconify Navigation block title only **/
#block-user-1 h2 {
    padding-left: 20px;
    background:url(icons/user.png)  no-repeat;
    background-position: center left;
}
```

Note on using the `<h2>` tag: To identify the block title we target the `<h2>` tag which is set in the theme's *block.tpl.php*. If your theme changes this (eg to `<h3>`) it breaches coding standards for structural mark-up; `<h2>` defines a second-level heading. If you want to change the size of an `<h2>` element, you should do this with CSS: eg

```
h2 {
  font-size: 160%;
  line-height: 130%;
}
```

prefixing the Current Node Title with an icon

Unfortunately the node title does not carry a handy CSS tag, and trying to select it by context is not very reliable either because it doesn't (eg Garland) have a parent `<div>` container that distinguishes it from other `<h2>` elements in page content (eg the comments header).

The quick solution is to add a tag using #yourtheme's *page.tpl.php*; eg Garland:
(note we use an `id` tag because there should only ever be one instance of this title on a given page.)

```
- <?php if ($title): print '<h2'. ($tabs ? ' class="with-tabs"' : '') .'>'. $title
.'</h2>'; endif; ?>
+ <?php if ($title): print '<span id="node_title"><h2'. ($tabs ? ' class="with-tabs"'
: '') .'>'. $title .'</h2></span>'; endif; ?>
```

and then in icons.css add:

```
/** iconify node-title only **/
#node_title h2 {
```

```
    padding-left: 20px;
    background:url(icons/user.png)  no-repeat;
    background-position: center left;
}
```

IE Conditional Comments

Browser detection using JavaScript is a touchy subject these days.

Setting IE's font size to match the rest of the world's browsers is no easy task. Or is it? Many themers feel a need to adjust the naturally gargantuan tendency of IE's fonts, or use a CSS hack to force IE to comply with a proper box model. Unfortunately for Palm Pilot users, Blackberries, screen readers, and even older versions of Opera, turning to JavaScript browser detection is a failed proposition. There are so many different browsers and screen sizes out there that it becomes impossible to feed each one its own unique style sheet.

The savvy themer will no doubt be asking, "Why not use one style sheet for all browsers, and then use JavaScript to feed IE its own settings?" Good point, but there's a better way than scripting to get this done.

Microsoft has enabled CSS designers to send Internet Explorer its very own set of CSS rules using a modified HTML comment tag. This comment is not a valid tag according to HTML standards, it's just a comment. All browsers should simply ignore it and move on. IE, however, will read this unique tag and follow every instruction inside it. The Conditional Comment is wrapped around a <link> tag that contains IE's very own stylesheet. Here's how it looks:

```
<!--[if IE]>
    <link href="screenStyle4IE.css" rel="stylesheet" type="text/css" media="screen" />
<![endif]-->
```

As always with Drupal, there are many ways to achieve this change to the head of every themed html page. Probably the best way is to use the new Conditional Stlyes Module (http://drupal.org/project/conditional_styles) and modify your theme's .info file to include conditional styles, thus:

```
; Set the conditional stylesheets that are processed by IE.
conditional-stylesheets[if lt IE 7][all][] = ie6-and-below.css
conditional-stylesheets[if IE 7][all][] = ie7.css
```

Note: This module only affects the way the theme registry is built. You will need to clear the theme registry before you will see any changes.

Another way is to place this tag *only* within the <head> of the page. Once the <link> tag is parsed by IE, any CSS in the screenStyle4IE stylesheet will take over. Remember that CSS is a cascading ruleset, so any overrides to previous rules must necessarily come *after* the normal CSS include link. Add a Conditional Comment to Garland's page.tpl.php file like this:

```
<head>
  <title><?php php print $head_title ?></title>
  <?php print $head ?>
  <?php print $styles ?>
  <?php print $scripts ?>
  <style type="text/css" media="print">@import "<?php print base_path() .
path_to_theme() ?>/print.css";</style>
   style type="text/css" media="screen">@import "<?php print base_path() .
path_to_theme() ?>/screen.css";</style>
   style type="text/css" media="handheld">@import "<?php print base_path() .
```

```
path_to_theme() ?>/handheld.css";</style>
  <!--[if lt IE 7]>
    <style type="text/css" media="all">@import "<?php print base_path() .
path_to_theme() ?>/fix-ie-layout.css";</style>
  <![endif]-->
  <!--[if IE ]>
  <style type="text/css" media="all">@import "<?php print base_path() .
path_to_theme() ?>/fix-ie-font-sizes.css";</style>
  <![endif]-->
</head>
```

This flexibility is astounding! It is now possible to feed a class for layout divs to all browsers, and then override the size, z-index, float, or margins of that div specifically for IE.

Font sizes, you say? But of course! IE's fonts are always one size larger than other browsers. Tame them by specifying style rules in fix-ie-font-sizes.css that are one size smaller that the corresponding rule in your regular stylesheet. For instance, if p{font-size:normal}, then fix-ie-font-sizes would spec p{font-size:small}. This keeps all browsers in somewhat of a uniformity without having to rely on font percentage hacks.

Oh, and if you want to hone in on specific versions of IE, refer to the previous code block for an example. <!--[if lt IE 7]> means "All instances of IE that are less than IE7". For more info on "lt IE6", "lte IE7" and so on, refer to Microsoft's Conditional Comments workshop

You can also create CSS and JS files that are read by everyone except for IE, using a slightly different construction, since in this case the declarations must be *outside* comments:

```
<!--[if !IE]>-->
    <style type="text/css" media="all">@import "<?php print base_path() .
path_to_theme() ?>/ie-wont-see-this.css";</style>
<!--<[endif]-->
```

The original forum post on this subject can still be viewed at http://drupal.org/node/16173

Note: Using *internal, or inline* stylesheets can lead to problems with some browsers (notably Opera and Safari), since they don't always ignore the CSS inside the comments. Using external stylesheets, as described above, seems to resolve this issue.

Include style sheets for specific browsers

How many hours have you spent debugging CSS to get it to work in most browsers?

Ever wish you could just specify CSS for that ONE version of IE that doesn't agree with the rest of the siblings in the collective, or any other misbehaving browser?

This shows how to specify separate stylesheet 'overrides' based on the user agent identification. A stylesheet specific to any browser? Bliss!

So here's how to:

Insert the following line into page.tpl.php, just below the tag that outputs the system styles:

```
<?php print browser_specific_sheet($directory); ?>
```

My header looks like this:

```
<head>
  <?php print $head_title; ?>
  <?php print $head; ?>
  <?php print $styles; ?>
  <?php print browser_specific_sheet($directory); ?>
  <?php print $scripts; ?>
</head>
```

This simply outputs an extra style tag, generated by the following function that you insert into your themes' template.php:

```
<?php
function browser_specific_sheet($directory=''){
  $user_agent = ( isset( $_SERVER['HTTP_USER_AGENT'] ) ) ? strtolower(
$_SERVER['HTTP_USER_AGENT'] ) : '';
  $prepend = '<style type="text/css" media="all">@import "/'.$directory.'/';
  $append = '";'."</style>\n";

  switch (true){
    case strchr($user_agent, 'msie 5.0'):
      return $prepend.'ie_5.0.css'.$append;
    case strchr($user_agent, 'msie 5.5'):
      return $prepend.'ie_5.5.css'.$append;
    case strchr($user_agent, 'msie 6.0'):
      return $prepend.'ie_6.0.css'.$append;
  }
  return '<!-- '.$user_agent.' -->';
}
?>
```

These cases are specific to IE 5, 5.5 and 6, you can add or remove case statements for any prefered browser, as long as you pick something unique to that browser's agent identification.

For IE 5 the following tag is rendered:

```
  <style type="text/css" media="all">@import "/sites/all/themes/themename/ie_5.0.css";
</style>
```

All that's left to do is create the style sheets 'ie_5.0.css', 'ie_5.5.css', 'ie_6.0.css' in your theme directory. If there is no case statement for a browser (say Firefox in this case) the following HTML comment is rendered:

```
<!-- user agent: mozilla/5.0 (x11; u; linux i686; en-us; rv:1.8.1.6) gecko/20061201
firefox/2.0.0.6 (ubuntu-feisty)-->
```

If you wanted to specify a style sheet containing style overrides specifically for Mozilla Firefox, you'd add the following case statement to the browser_specific_sheet function:

```
<?php
  case strchr($user_agent, 'firefox/2.0.0.6'):
    return $prepend.'my_firefox_styles.css'.$append;
?>
```

Making your content look "fixed width" yet be fluid

Some people don't want their site to use the full width of the screen, but this can cause problem for users

who aren't using a full sized window. This tip "shrinks" the content area, yet remains fluid, so it works better with smaller windows. It does not change the header and footer size, so they still extend across the full width.

```
#content {margin-left:10%; margin-right:10%;}
```

Note, this also moves the side bars in. You will have "white space" (actually, your body background color) on both sides.

Showing which Menu item you're on

This tips allows you to highlight, or show the focus on, the menu item your user is currently on. By default, Drupal just turns the item black. This tip places a box around the item. One nice "extra" to it, is that it also works on the book navigation block as well.

```
.menu .leaf .active {border:1px solid #D2691E;}
```

I know this tip works in Bluemarine and Pushbutton, but should work in all themes.

Use a color that contrasts with, but compliments your menu color scheme.

Styling more usable form buttons

To try and prevent clicking on destructive buttons, it might be a good practice to make the Delete button less important, yet still available. To do that, you can simply style each of the button types Drupal puts out—edit, submit and delete. Here's an example making the submit button boldest, and removing the button style of delete.

```
#edit-preview {
  border: 2px solid #CACC00;
  color: #A6A800;
}
#edit-submit {
  border: 2px solid #769405;
  color: #668004;
}
#edit-delete {
  background: none;
  border: none;
  color: #999;
}
```

Generate tabs for blocks using jQuery

The following will allow your site to generate a tab for every block assigned to a region. It requires jQuery and the tabs plugin from stilbuero.

stilbuero's demo page

Each block from Drupal will have its' own tab for a designated region. It's an efficient way to save space and clear out the clutter from your sidebars or wherever else you have more than one block. -*look below*

for live example in a drupal site.

The main problem is the way blocks are structured out of Drupal. The subjects need to be grouped together in an unordered list with a link pointing to a block it's related to. Drupal outputs the subject right above the block content which is difficult to override server side.

Drupal's output:

```
<code>
<div id="block-1" class="block">
  <h2 class="title">subject 1</h2>
  <div class="content">
    ...
  </div>
</div>

<div id="block-2" class="block">
  <h2 class="title">subject 2</h2>
  <div class="content">
    ...
  </div>
</div>

<div id="block-3" class="block">
  <h2 class="title">subject 3</h2>
  <div class="content">
    ...
  </div>
</div>
```

desired results:

```
<code>
<div id="tab-container">

  <ul class="tab-title-group">
    <li class="tab-title"><a href="#block-1">subject 1</a></li>
    <li class="tab-title"><a href="#block-2">subject 1</a></li>
    <li class="tab-title"><a href="#block-3">subject 1</a></li>
  </ul>

  <div id="block-1" class="block tab-block">
    <div class="content">
      ...
    </div>
  </div>

  <div id="block-2" class="block tab-block">
    <div class="content">
      ...
    </div>
  </div>

  <div id="block-3" class="block tab-block">
    <div class="content">
      ...
    </div>
  </div>

</div>
```

One way to achieve this is to use jQuery's DOM manipulation. A big bonus of this is that the tabs will not

break the site if someone is browsing without javascript. It will just be another block to them. Your **block.tpl.php** does have to output a unique id, a class of "block" and the subject must have a class of "title". *-look above for the assumed structure.*

First prepare a new region in your template or dedicate an existing one for the tabs. Inside your **page.tpl.php** file, look for a region and surround it with a unique div. Lets say for example that you have a region named *'tab-region'.''*The id of *'tab-container'*is what will be called to trigger the tab plugin.

```
<code>
<?php if ($tab-region) { ?>
<div id="tab-container">
  <?php print $tab-region ?>
</div>
<?php } ?>
```

Now you must include the jQuery library. If your using version 4.7, a patch is available. Be aware that it can cause complication down the line since it is *not supported!* It's part of Drupal core for version 5 though.

An easy alternative is to link to jQuery directly in the page.tpl.php. The biggest side effect is that it can conflict with the functions inside autocomplete.js and upload.js files. This is a known problem so you can try and modify core files or use the code below which checks for the files and disables itself if it is present. It's a cheap solution until you decide to patch 4.7 or upgrade to 5. Considering that your site shouldn't break when it's disabled, it's not an unreasonable tradeoff.

```
<code>
<!-- check for autocomplete.js -->
<?php if (!preg_match("/autocomplete.js/",$head) || !preg_match("/upload.js/",$head))
{ ?>

<link rel="stylesheet" href="<?php echo url($directory.'/tabs.css') ?>"
type="text/css" media="print, projection, screen" />

<!--[if lte IE 7]>
<link rel="stylesheet" href="<?php echo url($directory.'/tabs-ie.css') ?>"
type="text/css" media="projection, screen" />
<![endif]-->


<script type="text/javascript" src="<?php echo url($directory.'/scripts/jQuery.js')
?>"></script>

<script type="text/javascript" src="<?php echo url($directory.'/scripts
/jQuery.tabs.js') ?>"></script>


<script type="text/javascript">

  $(document).ready(function(){

    // new class is added for separate optional theming.
    $("#tab-container > .block").addClass("tab-block");

    // iterate through each block chaining functions.
    $("#tab-container > .tab-block").each(

      function() {

        $(".title",this)
          // wrap .title in li.
          .wrap('<li class="tab-title"></li>')
          // make link fragment from block id. Must be unique! $(this) = .tab-block or
```

```
.block
          .wrap('<a href="#' + $(this).id() + '"></a>');

        // copy inner text of .title into the link.
        $("li.tab-title > a",this).append($(".title",this).text());

        // delete old .title since markup associated with it doesn't make sense here.
        $("li.tab-title .title",this).remove();

      }

    );

    // add an unordered list container to the top of #tab-container.
    $("#tab-container").prepend('<ul class="tab-title-group"></ul>');

    // move all list items to unordered list container.
    $("#tab-container > .tab-title-group").prepend($("#tab-container .tab-title"));

    // trigger tabs with optional effects. More options available. Look at the
documentation.
    $("#tab-container").tabs({fxAutoheight: false, fxSlide: false, fxFade: true,
fxSpeed: 180});

  });

</script>


<?php
}?>
```

Place all the files into your theme folder including the css files included with the plugin. From there it should function but you must style the css files to match your theme. Use the new classes to style so the blocks don't look out of place when the script is disabled or when a visitor has javascript turned off.

Generated classes:

```
<code>
.tab-title-group {
  ...
}
.tab-title {
  ...
}
.tab-block {
  ...
}
```

Any blocks added to your "administration > blocks" page will have its' own tab now. You can reorder them, hide/show them as usual.

Take a look at stilbuero's demo page for usage information.

notes:

- this came out of a forum question http://drupal.org/node/88558.
- working example in a drupal site. -*scroll down* Theme isn't finished so it will break in Internet Explorer.
- If you know of a way to make it more efficient then please post. The author of this post is by no means an expert.

How can I change Drupal's character encoding? (UTF-8 and Unicode)

Several people have asked how to specify the character encoding that Drupal uses. The short answer is: you can't, but you don't have to.

Drupal uses UTF-8 for encoding all its data. This is a Unicode encoding, so it can contain data in any language. You no longer need to worry about language specific encodings for your website (such as Big5, GB2312, Windows-1251 or 1256, ...). Also, when Drupal imports external XML data (such as RSS or XML-RPC), it is automatically converted into UTF-8 (iconv support for PHP will be required for most encodings).

If you really want to change Drupal's encoding, you will experience a lot of troubles, because of the various ways Drupal can receive and send out data (web, e-mail, RSS, XML-RPC, etc).

How to change text-links to image links in the pager

After frantically searching for a solution I found out about the l(); function pestering me with plain text.

This already seemed to be a problem in Drupal 5 but no clear workaround was found. I overwrote the theme_pager() function, only for changing the $li_previous and $li_next variables.

Original:

```
$li_previous = theme('pager_previous', (isset($tags[1]) ? $tags[1] : t('‹ previous')),
$limit, $element, 1, $parameters);
```

Changed to:

```
$li_previous = theme('pager_previous', (isset($tags[1]) ? $tags[1] : theme('image',
'sites/all/themes/landrush/img/vorige.jpg', $alt = 'Vorige', $parameters, FALSE)),
$limit, $element, 1, $parameters);
```

By using the theme_image() function Drupal should nicely format the img tag and output it to your theme. Well, this works, but unfortunately somewhere in this theme_pager process a l(); is telling Drupal to only output plain text here.

The trick here is not to rewrite your whole pager functions. Have a look at http://api.drupal.org/api/function/theme_pager_link/6 .
This function is building all the links which are used by the pager functions. When you look closer to the l(); formatting the output:

```
return l($text, $_GET['q'], array('attributes' => $attributes, 'query' =>
count($query) ? implode('&', $query) : NULL));
```

you'll see that the options aren't setting the html output. So I placed the whole function in my template.php (changing theme_pager_link to landrush_pager_link) with a different output:

```
return l($text, $_GET['q'], array('attributes' => $attributes, 'html' => TRUE, 'query'
=> count($query) ? implode('&', $query) : NULL));
```

Note the 'html' => TRUE.

So now l(); is allowing me to print html tags (images!) within the pager function.

My complete code (in template.php):

```
// Override the ending l() function to allow html in the pager links (thanks Kaaskop &
Swentel)  :
function landrush_pager_link($text, $page_new, $element, $parameters = array(),
$attributes = array()) {
  $page = isset($_GET['page']) ? $_GET['page'] : '';
  if ($new_page = implode(',', pager_load_array($page_new[$element], $element,
explode(',', $page)))) {
    $parameters['page'] = $new_page;
  }

  $query = array();
  if (count($parameters)) {
    $query[] = drupal_query_string_encode($parameters, array());
  }
  $querystring = pager_get_querystring();
  if ($querystring != '') {
    $query[] = $querystring;
  }

  // Set each pager link title
  if (!isset($attributes['title'])) {
    static $titles = NULL;
    if (!isset($titles)) {
      $titles = array(
        t('« first') => t('Go to first page'),
        t('‹ previous') => t('Go to previous page'),
        t('next ›') => t('Go to next page'),
        t('last »') => t('Go to last page'),
      );
    }
    if (isset($titles[$text])) {
      $attributes['title'] = $titles[$text];
    }
    else if (is_numeric($text)) {
      $attributes['title'] = t('Go to page @number', array('@number' => $text));
    }
  }

  return l($text, $_GET['q'], array('attributes' => $attributes, 'html' => TRUE,
'query' => count($query) ? implode('&', $query) : NULL));
}

// Function borrowed from http://api.drupal.org/api/function/theme_pager/6 to display
less items
// in our smaller space (needs above function to allow html!)
//
// NOTE: I modified a lot in this function, if you notice a flaw please tell me,
haven't had the time to test it yet..
//
function landrush_pager($tags = array(), $limit = 10, $element = 0, $parameters =
array()) {
global $pager_page_array, $pager_total;

$pager_current = $pager_page_array[$element] + 1;
$pager_first = $pager_current - $pager_middle + 1;
$pager_last = $pager_current + $quantity - $pager_middle;
$pager_max = $pager_total[$element];
```

```
$i = $pager_first;
if ($pager_last > $pager_max) {
$i = $i + ($pager_max - $pager_last);
$pager_last = $pager_max;
}
if ($i <= 0) {
$pager_last = $pager_last + (1 - $i);
$i = 1;
}

$li_previous = theme('pager_previous', (isset($tags[1]) ? $tags[1] : theme('image',
'sites/all/themes/landrush/img/vorige.jpg', $alt = 'Vorige', $parameters, FALSE)),
$limit, $element, 1, $parameters);
$li_next = theme('pager_next', (isset($tags[3]) ? $tags[3] : theme('image', 'sites/all
/themes/landrush/img/volgende.jpg', $alt = 'Volgende', $parameters, FALSE)), $limit,
$element, 1, $parameters);

if ($pager_total[$element] > 1) {
if ($li_previous) {
$items[] = array(
'class' => 'pager-previous',
'data' => $li_previous,
);
}

for (; $i <= $pager_last && $i <= $pager_max; $i++) {
if ($i > $pager_current) {
$items[] = array(
'class' => 'pager-item',
'data' => theme('pager_next', $i, $limit, $element, ($i - $pager_current),
$parameters),
);
}
}
if ($li_next) {
$items[] = array(
'class' => 'pager-next',
'data' => $li_next,
);
}
return theme('item_list', $items, NULL, 'ul', array('class' => 'pager'));
}
}
```

Remove "Login to post comments" from Drupal 6 sites

By default, at the end of an article Drupal displays the text "Login to post comments".To remove this text, allow commenting for anonymous users or just disable the Comment module.

Mod of Garland background

Quick tutorial based on what I did at www.shelfanger.com

First as with all learning – before you start make a backup of your site and database in case of any problems

Now go to Admin > Themes > Garland > Configure. Chose the standard colourset that most closely matches your desired end result. In this case I am choosing Greenbean. Turn off [Logo], [Site name], [Site Slogan] and [Mission statement] and click [Save configuration].

What you would (usually) now do is browse to your drupal site folder > files. > color > you will now find a new folder in color called [something like] garland-b3aeefcb > open this and find body.png.

If your site is online use ftp-filezilla (or whatever you use) to download this file to your hard disk. To save repeating myself at each stage when we work on this file locally IF your drupal site is online you will need to upload the new version via ftp to then browse the result. If working with a local server simply save and check in browser.

OK – open body.png in image editing program change its size from 1px by 280px to 1280px.by 280px or download and use the versions I have already created at http://82.69.23.179/shelfanger/files/color.zip

Save this elongated version overwriting the original (tip I always make extra copies as I go in case I want to go back a stage). If you go back to your site in a browser and refresh you should see no difference because all we have done, to date, is replace a background.png that is one pixel wide (but that repeats across the background) with a similar one that is 1028 pixels wide. Now go back to your image program > turn body.png negative > save > and check in browser and the background image should now be negative. If so we know we are working with the correct background image.

Back to the image editing program and reverse the last step, turning it back to a positive image and save. The best way forward is to create a new layer and then bring in your new photos, graphics, fancy text onto this new layer. As a test just bring in a very small photo Say 150pixels wide aand put it in the top left corner. Save and publish. (but keep it open if your editing prigramme so you can continue working. When you check again in your browser you should see the new graphic.

Waita minute... it appears to have mostly disappeared into the left hand margin!. Now some code adapting I'm afraid.

Go to files. > color > garland-b3aeefcb > open style.css in notepad and alter the line

background: #fbfaf7 url(body.png) repeat-x 50% 0;

to

background: #fbfaf7 url(body.png) repeat-x 0% 0;

Save file and and recheck in browser. However you have adapted the background in your editing programe should now be reflected in your site. If the new background elements crash into any site text or existing elements simply go back to the editing prog and resize or move them.

That should be enough for part one of this tutorial. - do give me your feedback

How to: Modify the Garland theme to include a custom image in the background

This is an update to the post "Simple but effective mod of Garland theme". This document clarifies some points and notes the differences to get it to work for Drupal 6.1.

Prepare
Backup your site files and database in case of any problems and try this on a test site.

The style.css Changes

Go to themes folder. Make copy of Garland folder and rename it garland_custom (or whatever name you like). Open custom folder and open style.css in notepad (or your favorite editor) and alter the lines:

(line 308)

```
background: #f7f1de url(bg-navigation.png) repeat-x 50% 100%;
```

to
```
background: #f7f1de url(bg-navigation.png) repeat-x 0% 100%;
```

and (line 351)

```
background: #fbf9f2 url(body.png) repeat-x 50% 0;
```

to
```
background: #fbf9f2 url(body.png) repeat-x 0% -37px;
```

and (line 465)

```
background: #ffffff url(bg-content.png) repeat-x 50% 0;
```

to
```
background: #ffffff url(bg-content.png) repeat-x -10px 0;
```

and (line 357)

```
max-width: 1270px;
```

to
```
max-width: 1920px;
```

Save and close style.css

For Drupal 6.1: The info file Changes
Rename garland.info file to match new theme name, garland_custom.info in my case. Edit garland_custom.info line 2. Change "name = " to something that makes sense. I changed it to:

```
name = Garland Custom
```

Save and close garland_custom.info

It is best to put your new folder (garland_custom) in sites/all/themes or if it is specific to one site in a multi-site setup put the garland_custom folder in sites/example.com/themes.

Configure your site via the Administer section

Using your browser go to your the Themes configuration page on you website (*Administer* > *Site building* > *Themes*). Enable the custom theme and set to default, then save the settings with the "Save Configuration" button at the bottom of the page. Now configure your new custom theme using the "configure" link and choose the color set that you want to base your custom theme on (for example

Belgium Chocolate, or if you are using a custom color set, enter your colors, it will make things easier later if you use a custom Base color, then picked a second color and used it as the same color for *both* Header top and Header bottom). Decide if you wish to use the [Logo] and [Site name] options, or if you are incorporating them directly into your background graphic and tick/untick accordingly. There is a home_logo.gif that works well that you can download. Save your settings with the "Save configuration" button.

Your site will look a bit strange now as the top of the shaded boxed don't really line up.

Make the graphical changes (using PaintShopPro)

1. Change the color set.

   Using Windows explorer, go to files > color > new folder something like "garland_custom-1e27bf52" view this folder as thumbnails. (This might be in sites/example.com/files/color; it depends on your set up.)

   Download drupalcustom.pspimage (and some other files if you want), which is a PaintShopPro v8 file, a trial version of PaintShopPro v8 is available for download. (This is a windows version, so either do this on a windows machine, or use an emulator on a Mac or *nix.)

   Open file - drupalcustom.pspimage in PaintShopPro, on right hand side of window you will see the layer box. In the group called GarlandColour most are inactive. Turn off pinkplastic (by clicking on the eye) and turn on say belgiumchocolate. You will see that the background has changed.

   (Or if you used a custom color set above then you have to create your own new background color layer! Make a copy of one of the colorset layers (aquamarine for example) by right clicking on aquamarine, then choose "duplicate". Then right click on "Copy of aquamarine" and choose "rename". Call the new layer "mycustomcolorset" or something. If you didn't use the gradient and used the same color for Header top and Header bottom when configuring your custom theme above, you can change the colors in your mycustomcolorset layer. Hide (by clicking on the eye icon), the layers for "shading" and "Group - mainimage". Make sure in the "GarlandColours" layer group your "mycustomcolorset" layer is visible, and the other colorsets are hidden. Then set your foreground color that you want for example by: opening a file that has the colors you want, and use the "dropper tool" of the left hand side to set the foreground color. Switch tools on the left hand side to the "flood fill tool" then do a "fill" in the gradient area almost at the top. You might have to click a very narrow strip that didn't fill the first time. Then use the "dropper tool" again to choose your other color and do a fill in the area at the very top and fill the large area at the bottom. Turn the "shading" and "Group -mainimage" layers back on by clicking on the eye next to them.)

2. Remove the "car" custom image and bring in your own custom image.

   Now in the layer dialog on the right hand side, within "Group - mainimage", choose "mainimage". The layer name in the Layer dialog should turn grey. Using the regular menus at the top of Paint Shop Pro, select Edit > Cut and the car should have gone away. Open a new file with: File > Open and bring in an image of your own (anything will do for now). Use the menu: Edit > Copy. And then go back to drupalcustom.pspimage and use the menu: Edit > Paste > Past As New Selection. Position it in the top left to suit your wishes. [Optional: right click on the "Floating Selection" layer in the Layer dialog box and "Promote Selection To Layer". Then right click on "Promoted Selection" and choose "rename" to rename your image. This is useful if you have several images to drop in and want to be able to work with them separately to fine tune things.] Then unselect it using the menu: Selections > Select None.

To recap, you can change the base color to whatever you want and brining an image in automatically blends it with the background.

3. Create new body.png

   Now revert in the menu: File > Revert and change color to belgiumchocolate as above. Save as drupalcustom.png. [Optional: Do not do the revert. Not sure what effect not Reverting makes.] Close this file and open up drupalcustom.png again (this ensures it is now a single layer flat image). Save it in the [garland_custom-1e27bf52] folder as body.png overwriting existing one. Check home page in browser to view change. (You may need to force it to reload images by holding down the shift key while clicking the reload button in your browser.) You are still not done yet, so while you should see your new image in the top left, it is still not quite right. You need to make the navigation, left and right slices.

4. Create new bg-navigation.png

   On right hand side of PaintShopPro window (you may need to reopen drupalcustom.pspimage) in Overview dialog, in the Preview tab, zoom to say 300%. Click Info tab. This will show your cursor position. Use the selection tool to select (x,y) 0,0 to 1920,37. Use menu: Edit > Copy, then Edit > Paste > Paste As New Image. Save in the [garland_custom-1e27bf52] folder as bg-navigation.png, overwriting existing one. Check home page in browser to view change (should see the change at the very top of the page).

5. Create a "working version" of bg-content.png

   Back in PaintShopPro drupalcustom.pspimage, go to position 220,117 and select to the bottom right corner 1920,524 (size of select is 1700x407). Use menu: Edit > copy and then menu: Edit > Paste > Paste As New Image. Save overwriting current bg-content.png. Check home page in browser to view change. (Things look worse temporarily if your window is wide, we will fix that in a little bit.)

6. Create new bg-content-left.png

   Now in PaintShopPro bg-content.png, select from top,left (0,0) to 50 wide x 352 deep (50x352). From menu: Edit > copy and then from menu: Edit > Paste > Paste As New Image. Save overwriting current bg-content-left.png. Check home page in browser to view change. (Now your custom image should be current in the upper left corner of the content area.)

7. Create new bg-content-right.png

   Now back in PaintShopPro bg-content.png go to position 488,0 and select to depth 352 and a width of 50 (cursor to position 538,352 this selects the right edge of the content area). From menu: Edit > Copy and then Edit > Paste > Paste As New Image. Save overwriting current bg-content-right.png. Check home page in browser to view change. (Depending on your changes you might not notice any.)

8. Fix new bg-content.png

   Back to PaintShopPro in the new bg-content.png (Maybe save a copy called bg-content-precrop200.png) and crop to 200 deep. (The bg-content.png before the crop is 1699x406. Crop it to 1699x200 keeping the top part.) Then go from 478,0 (grabbing a piece of the "middle" of the content area) to total depth and width 10px (so select from 478,0 to 488,200). From menu: Edit > Copy. Then select from 488,0 to bottom right corner (1699,200). From menu: Edit > Paste > Paste Into Selection. From menu: File > Save As bg-content.png. Check home page in browser. (This should fix

the temporary problem from the end of the previous bg-content step.)

Back up your new files

That is pretty much it. Except to say that you should backup your [garland_custom-1e27bf52] folder, because if you go into the theme settings for whatever reason it helpfully overwrites all your custom graphics with new ones! (The files you will need to recopy to the folder would be: bg-content.png, bg-content-left.png, bg-content-right.png, bg-navigation.png, body.png.

Overriding Theme Templates in Drupal 6

Adding additional preprocess functions inside existing preprocess functions

You may want to add additional preprocess functions inside your themes preprocess function or preprocess_hook function to create different variable in different situations. For instance, if you wanted to have a node-type template (node-story.tpl.php) you could create a function called MYTHEME_preprocess_node_story(), where you would set story specific variables. The title of the function is not called trough the theme API, so you could call it whatever you like, but this would allow you to setup your own API for calling all node-type templates with only a few lines of code.

```
<?php

function MYTHEME_preprocess_node(&$vars, $hook) {

//variables i want to do for every node

// node-type specific variables each with their own preprocess function
  $function = 'MYTHEME_preprocess_node'.'_'. $vars['node']->type;
  if (function_exists($function)) {
   $function($vars);
  }
}
?>
```

Recipe for two column block of recent comments

My site http://www.delriolive.com has rather large CSS "pods" to display certain blocks. One such block is recent comments. The default "Recent Comments" block is too narrow for my large block size, so I embarked to create two columns of recent columns with 10 recent comments in each column. Using the phptemplate technique, here's how I did it in Drupal 4.7:

1. Find the comment block function in modules/comment.module:

```
function theme_comment_block() {
  $result = db_query_range(db_rewrite_sql('SELECT c.nid, c.subject, c.cid, c.timestamp
FROM {comments} c INNER JOIN {node} n ON n.nid = c.nid WHERE n.status = 1 AND c.status
= %d ORDER BY c.timestamp DESC', 'c'), COMMENT_PUBLISHED, 0, 10);
  $items = array();
  while ($comment = db_fetch_object($result)) {
    $items[] = l($comment->subject, 'node/'. $comment->nid, NULL, NULL, 'comment-'.
$comment->cid) .'<br />'. t('%time ago', array('%time' => format_interval(time() -
$comment->timestamp)));
  }
  return theme('item_list', $items);
```

```
}
```

2. Cut-n-paste the entire function into theme/simplex/template.php (my theme in this example is "simplex." Replace your theme name with appropriate theme name). Then modify the mysql query gather 20 instead of 10 recent comments:

```
$result = db_query_range(db_rewrite_sql('SELECT c.nid, c.subject, c.cid, c.timestamp
FROM {comments} c INNER JOIN {node} n ON n.nid = c.nid WHERE n.status = 1 AND c.status
= %d ORDER BY c.timestamp DESC', 'c'), COMMENT_PUBLISHED, 0, 20);
```

3. Rename the function to correspond to the simplex theme:

```
function simplex_comment_block() {
  ...
}
```

4. Change the elements returned by the function to parse through phptemplate using the _phptemplatecallback function. Note that we are going to parse just the comments, so I renamed the array to be passed as 'comment_block,' NOT 'item_list.' Because of this, instead of creating an items_list.tpl.php file, I will create a comment_block.tpl.php and place it in the themes/simplex directory:

```
function simplex_comment_block() {
   ...
  return _phptemplate_callback('comment_block', array('items' => $items));
}
```

5. Create a comment_block.tpl.php file where we will format the 20 comments that are passed as an array called $items[]. To place the comments in two columns with 10 comments in each columns in CSS, I used the following code in comment_block.tpl.php:

```
<div class="leftfrontcommentblock">
<?php
$ni=0;
    foreach ($items as $aa) {
        if ($ni < 10) {
            print $aa . "<br />";
            $ni++;
            }
        elseif ($ni== 10) {
            ?>
</div>
<div class="rightfrontcommentblock">
<?php
            print $aa . "<br />";
            $ni++;
            } else {
            print $aa . "<br />";
            $ni++;
            }
        }
    ?>
</div>
```

6. The two columns of comments are inserted into an existing CSS DIV tag and need to lie side-by-side to make the two columns. Here is the CSS that accompanies the output of comment_block.tpl.php:

```
.leftfrontcommentblock {
  float: left;
  width: 192px;
  height:260px;
  margin: 5px 5px 0 0;
```

```
  padding: 0;
  background: #fff;
  font-size: 9px;

}
.rightfrontcommentblock {
  float: right;
  width: 192px;
  height:260px;
  margin: 5px 5px 0 0;
  padding: 0;
  background: #fff;
  font-size: 9px;

}
```

If you have any comments, or better way to do this, please post it here! Thanks! Joe Hyde joeghyde at gmail dot com.

Tips for designing themes in Dreamweaver, GoLive etc.

An Xtemplate-in-GoLive how-to is available at http://drupal.org/node/6634

For notes on configuring various tools, including Dreamweaver see the Development Tools section of the handbook.

Totally customize the LOOK of your front page

If you want to lay out your front page dramatically differently from the rest of your site, you can define a whole other template just for that.
This could be handy if you have

- A highly graphic front page.
- A working mock-up from a designer that can't be changed much.
- A flash blob as a landing page.
- A splash screen, or any other special requirement.

... and working that into a generic Drupal theme template is just not practical.

Any Drupal page can be themed specifically, using the template-suggestions process used in phptemplate theme engine. This is powerful, learn about it.

At its simplest, you can

1. Create a page called `page-front.tpl.php` in your active theme directory.
2. Paste in solid, static HTML, with appropriate file path fixes.
3. Done.

Note that using *relative paths* to images (or swf) directly from that page will not work to find images stored nearby in your theme directory. *Placing those images in your theme directory is the correct thing to do*, but you will have to rewrite your HTML to either use:

`<img src="/sites/default/themes/yourtheme/images/yourimage.gif">` (which is bad, but quick) or `<img src="<?php print $directory; ?>/images/yourimage.gif">` (which is much better - $directory always returns the current active theme path)

To do a custom front page that is *mostly* unique, yet merges seamlessly with Drupal, you can

1. Create a page called `page-front.tpl.php` in your active theme directory.
2. Paste in solid, static HTML, with appropriate file path fixes.
3. **Compare** your page with the themes normal page.tpl.php and copy as much of the code tags as is logical into your new file in appropriate places.
4. You can put `$content` in the middle, sidebars in a useful place for you to call blocks into later, and the $footer in your footer zone, etc. Try to copy as much as possible - most of it can be useful, and is the real reason for using a CMS.

Under Drupal 6, new templates may not always be recognized or used immediately. Visit the admin-themes page and submit once to refresh the themes cache the first time you add the *.tpl.php

There are also several module designed to give you advanced control of the page without too much template work, such as Panels or maybe Node Style

See the tutorials for advice on how to change the *content* of the front page and replace the "River of news" or the "Welcome to Drupal" with a specific page.