Refacing your 68classifieds site V1.0
By: Larry Hotchkiss

The goal of this tutorial is to help you design, layout and finally apply a whole new look to your 68classifieds site and is geared toward 68classifieds ver 3.0.x. The graphics manipulation program I am using is Photoshop 7 but if you are familiar with another package you should still be able to follow along. I also assume you have some basic knowledge of Photoshop so I wont necessarily show you what/where every tool is that I use. Also, a basic understanding of HTML and cascading stylesheets is also assumed.

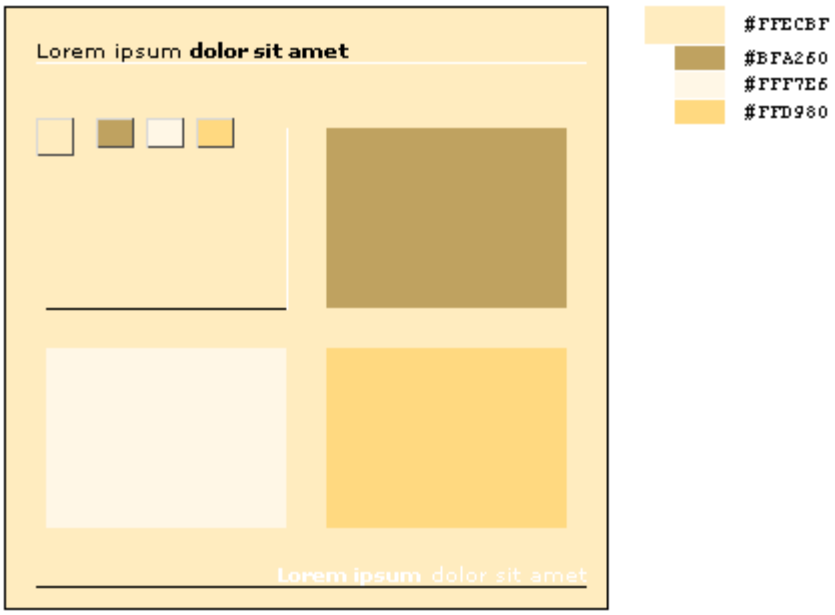
Additionally, if you don't already have it I would highly recommend you get the firefox browser and install the web designer plugin for it. It will make this whole process much easier. Lastly, keep in mind I dont consider myself a web designer, artist or anything resembling either. This is just an exercise to get you started. Its was rather basic exercise to help you understand the 68 Classifieds template layout and is only one of the many ways you can modify the look of your own site.

I like to start by choosing colors for the site. One problem I have always had was choosing colors that went well together. It was often time consuming placing some squares on a page and filling them in endlessly trying to find colors that work well together. To help simplify this task I found this neat little tool on the web.

http://wellstyled.com/tools/colorscheme2/index-en.html

Its pretty self explanatory, there is a color wheel, an image to show the output and a series of buttons where you can alter the colors. If you have a starting color in mind you can even enter your RGB number and get complimentary colors returned.

For this tutorial I have chosen the following colors.

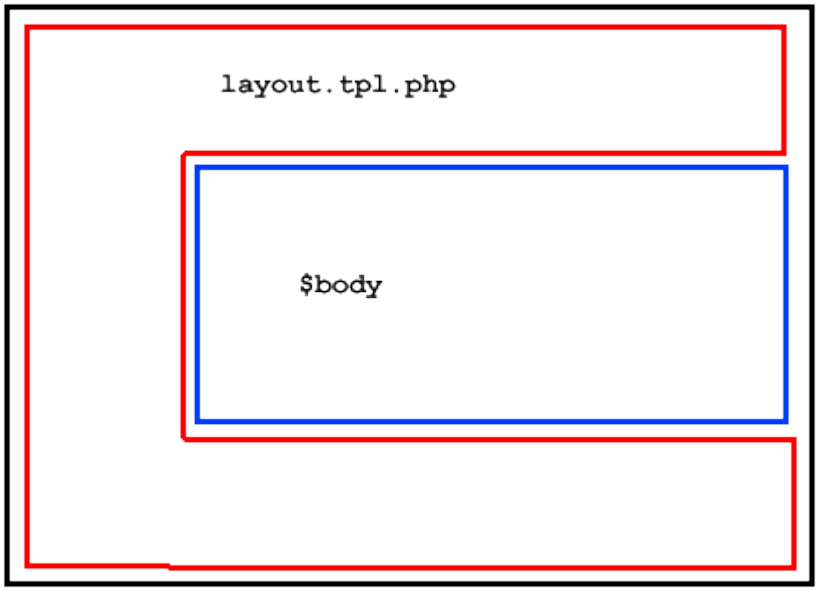


Now, before we start with our new design we have to do a little planning. To keep things simple I am going to stick with the default classifieds content layout where we have a header, left side navigation and a footer.

With that in mind we need to understand just how the default templates work.68 Classifieds uses Smarty templates. These templates represent the presentation layer of your site, in other words they contain the HTML that visitors to you site will see. More specifically, they, in conjunction with the templates style sheet, control the images, fonts and overall look of the site. The underlying PHP scripts handle the dynamic data and control which template to call and what dynamic data to pass to the templates being used.

When a visitor points a web browser to your classified site by default they will land on **index.php**. This script will in turn connect to the database, request some information and assign this information to variables that it will pass along to the templates which control how this information is to be displayed. Generally speaking each page the user sees is made up of 2 template files, **layout.tpl.php** and what ever other template is called to display the body of the page. **layout.tpl.php** is in essence a frame and within it a second template is displayed.

The base script that you point your browser to will display the **layout.tpl.php**, assign misc variables to be displayed and also assign another template file to the variable called **$body**. Below is a simple picture to represent a typical page. The black outline represent the whole page, the red represents the **layout.tpl.php** section of the page and the blue represents the area handled by an additional template file specified by the underlying script and passed to the **layout.tpl.php** template as the variable called **$body.**



Ok, so now that we have a very basic understanding of the **layout.tpl.php** we need to take a peek at the HTML source to see how the page is presented to the web browser. This will give us a better understanding of how the underlying HTML is organized and the page layout is handled. This can be done by pointing a browser to you classified site and once the page loads, select “view -> source” from your browsers menu.

By looking at the HTML source or using firefox with the web developer extension we can see that the page is basically a table where one data cell contains the navigation and another the body. Next we need to determine what kind of restraints are in place for the main tables as well as the nested tables.

For starters, we can see that the page is contained within a ID selector called “Wrapper”. If we take a peek at the stylesheet we see that this wrapper sets the width to 90%. What this means is the site should autosize to a degree based on the resolution of your visitors system. In other words if your visitor has their systems resolution set to 1024x768 pixels the site content will be 90% of that width or 921pixels wide. Alternatively if your visitor has their resolution set to 800x600 pixels the classifieds page will be 720pixels wide. The ability for the site to autosize to accommodate your visitors is a very nice touch but can add
a lot of complexity, especially when you want to start adding banner ads etc to your site. One example would be if you have a banner ad in the header of your site. A standard size for banner ads is 468x60 pixels and this may fit great when viewing your site in 1024x768 resolution but when viewed in 800x600 resolution that same banner can hang way off the right of you page, or if you used CSS to position it based on the right side of the page it can shift to the left covering up your logo etc.

To keep things simple this tutorial will be based around a static sized site, one that does not auto size when viewed at a lower resolution. This means that if being viewed on a system running 800x600 resolution you will not only have the vertical scroll bar but a horizontal one as well. According to recent statistics 74% of all internet users are running 1024x768 resolution or higher so this shouldn't be a huge concern.

Since we are going to create a site that is horizontally static in size the “wrapper” selector currently in the style sheet for the default templates can in essence be disregarded. We simply need to chose a size that will display nicely and make sure we allow enough space for our navigation links. If we look just under the header section contained in the “wrapper” selector we can see the sites main table is defined with a width of 100%. Keep in mind however that for the default template this is within the “wrapper” selector so its in essence using 100% of the 90% defined by “wrapper”.

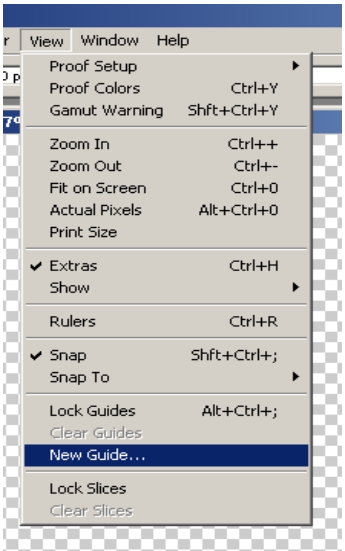
Here is the top few lines of body HTML as seen when viewing source from a browser. As we can see just under the main table markup we have a table row and then within that row we have a data cell defined as 200pixels wide. This 1st data cell contains a table for the navigation section of our site. This 200 pixel width is probably a good number to use when designing our layout since we know the currently defined nav links will fit in it.

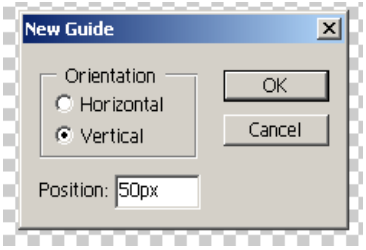
```
---------------- start view source ---------------------------------
<body>
<div id="wrapper">
<!-- // Header // -->
<div id="header"><a href="index.php"><img src="templates/default(2)/
images/header.gif"
width="633" height="93" border="0" alt="68 Classifieds" /></a></div>
<!-- // Header // -->
<!-- // Top Bar // -->
<div class="topbar">
<div id="menu"><strong>
</strong></div>
</div>
<!-- // Top Bar // -->
<table width="100%" class="tableWhite">
<tr>
<td width="200" valign="top">
<!-- // Navigation // -->
<table border="0" cellpadding="0" cellspacing="0" class="navigation">
---------------- end view source -----------------------------------
```

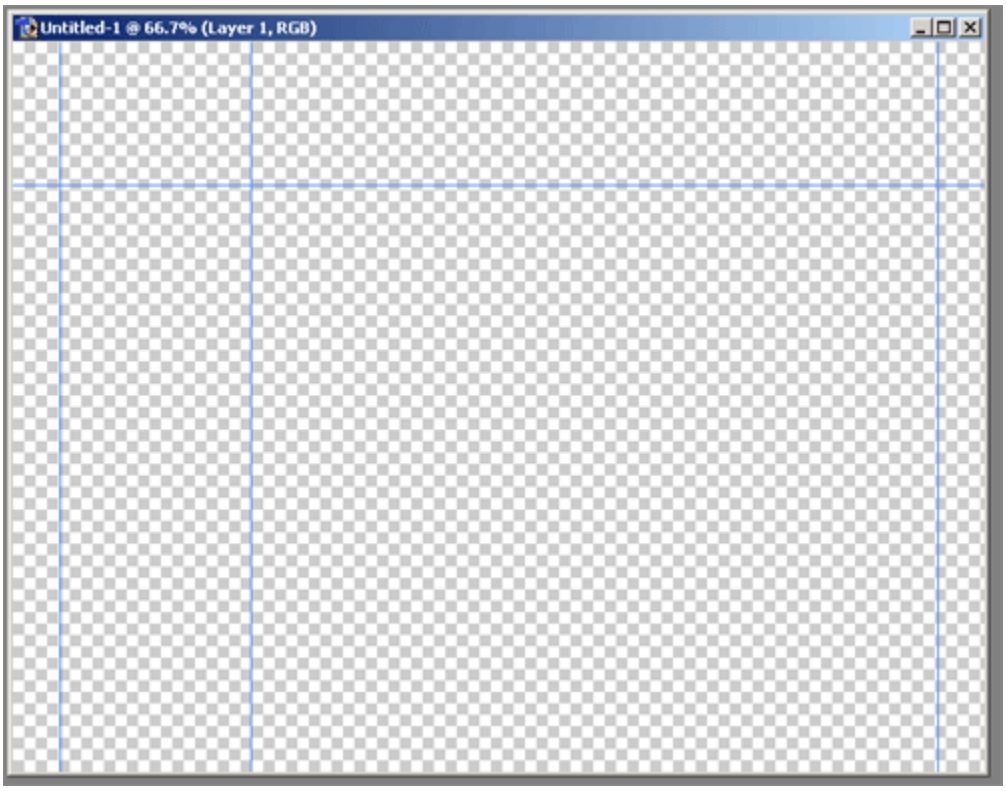
Next if we scroll down the viewed source a bit father to where the navigation table ends and the data cell holding the table is closed. On the same row we can see that another data cell is created and in it we have the text for our home page as well as a table which has our categories in it and then below that in the same call is another table which has our featured ads in it. These tables aren't limited in width and by default will just use what space is left over after the navigation table uses its 200 pixel width.

Now that we have some initial information on how the default template is laid out we can move over to photoshop and start to put together a new design.

Lets start by opening a new document thats 1024x768 pixels. You can have the background set to what ever you like. Next, lets set up some grid lines to help us in laying out the site. I want to leave a little space on the left and right and have the site centered on the page so I am going to place a couple guides, one on right and one on left. The first will be a vertical at 50px which will place a grid line 50 pixels in from my left side and the next will be a vertical at 974px which will set the second grid 50 pixels in on the right side. Next I will place a third vertical grid at 250px which will represent the width of my navigation space. Then I will place a horizontal grid at 150px which will act as a rough guide for the bottom of my sites header. You should end up with something similar to this.

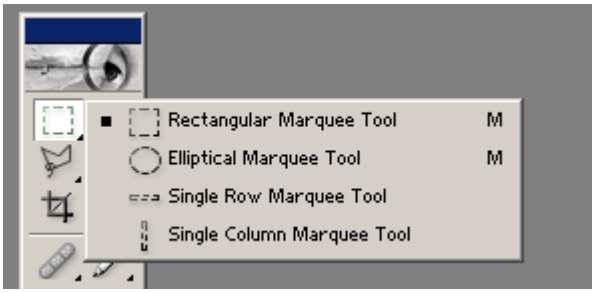




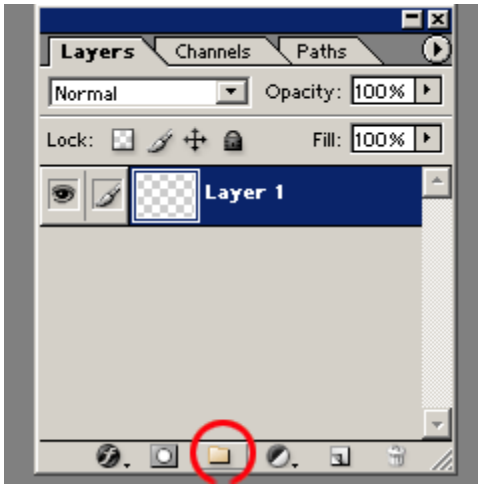


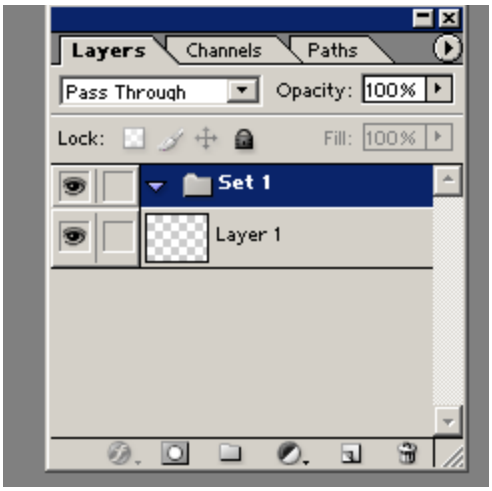
Ok, so now we have a rough guide as to where sections of our layout should start and stop we can start to drop stuff into place. This is where you want to let your creativity shine but for this tutorial I am going to keep things rather simple.

For making shapes I rely mainly on the marquee tools. You can alter the selection type to add to and take away from existing selections to make a variety of shapes.

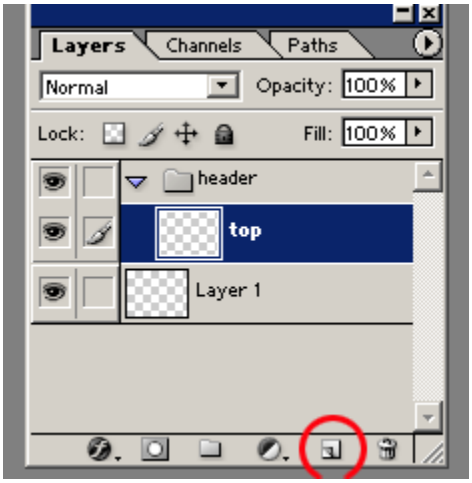


Lets start off by making the header. To help separate different elements in the layout, I like to keep the various image layers in “sets” and this can be done by selecting the “Create a new set” button at the bottom of the layers panel (left image below, red circle). Then you can right click on the new set and select “Layer set properties” to give it a name or simply double click the default set name and it should allow you to rename it, call it “header” to help identify whats in the set.





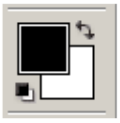
Now with the header set highlighted, click the "Create a new layer button" (2 buttons to the right of the create a new set button) and you will create a new blank layer within the new "header" set. If you forget somewhere along the way and inadvertently add a layer to the wrong set, you can click, hold and drag layers around within the layer palette. You can name the layer just like you did the layer set. Im simply going to call it "Top" since this piece will be the top piece of our header. You should end up with a layer pallet that looks like this. (the add layer button is circled in red im following image).

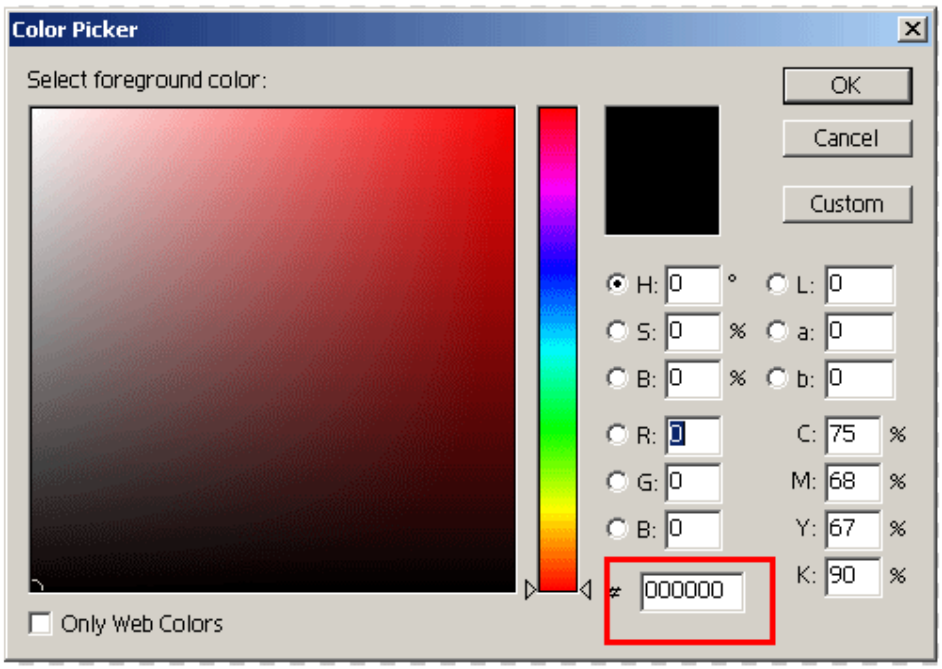


Now that we have our "top Layer" created select that layer by single left clicking it with your mouse and it will highlight blue as in the image to the left.

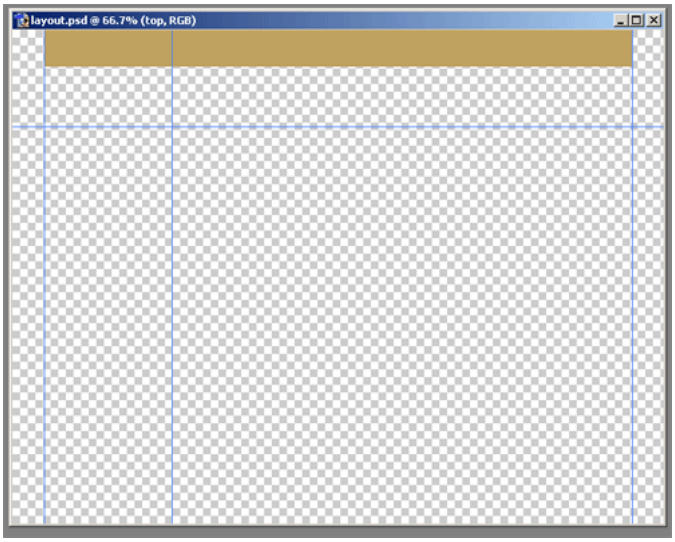
A couple words before we get started making our shapes. Generally speaking straight lines will give us the cleanest edges. Curved shapes and circles, even when anti aliased have a tendency to give you a more jagged edge. So, for simplicities sake I am going to keep curves to a minimum.

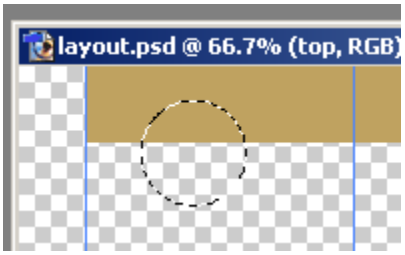
Im going to start by single left clicking on the current foreground color in our toolbar which will cause the "Color Picker" to open. Im going to start with the darkest of the 4 main colors I have chosen which is RGB value of BFA260 and you can enter this value in the color picker window (see red box on image below) and press the "OK" button to accept the color.





Now with our intended color selected go up to your toolbar and select the rectangular marquee tool and then drag out a rectangle. I used the guides we placed earlier as right and left boundaries for my rectangle and made it about 1/3 of the height we had specified for our header area. Next, select the paint bucket tool and fill in the rectangular selection you just made and you should have something that looks like the following.



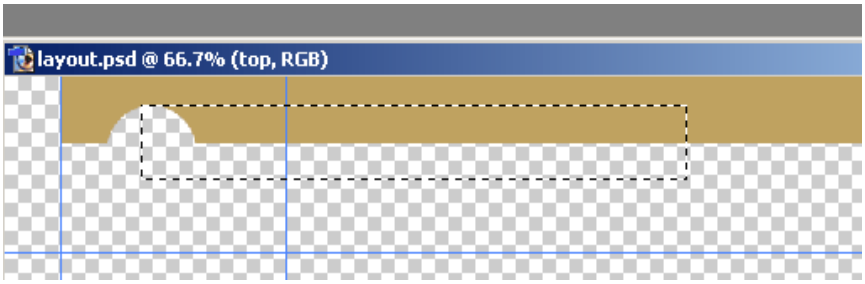


Next, lets right click on the rectangular marquee tool to get the popup window with other marquee tool options and select the “Elliptical Marquee tool”. After selecting the Elliptical Marquee there will be a style dropdown box on the top menu bar. Click on the dropdown and select “fixed aspect ratio”. This will make sure we end up with a nice circle instead of an oval. Make a selection on the upper left so the circle about half overlaps the colored bar we just created. It should look something like this.

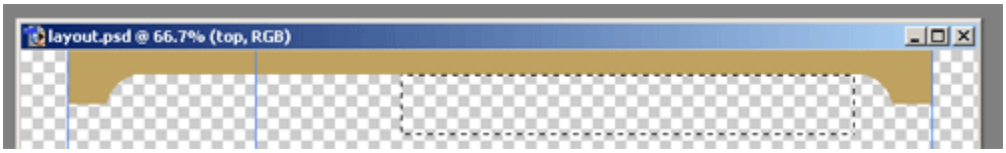
Once we have the circle selection created, press the “delete key” to remove a circular portion of our rectangle. Now you can use your arrow keys on your keyboard and move the circle selection to the right, place it about the same distance from your right guide as the original was from the left guide. When you get the circle selection where you want it press the delete key to remove another piece of the rectangle. Your starting rectangle should now look something like this.

After deleting the second piece of the rectangle press the “CTRL+D” key combination to clear your circle selection marquee. Or you can goto the top menu, press “Select:” and then “De select” from the dropdown menu.

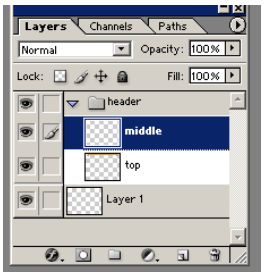
Now what we need to do is create another rectangle marquee to use to remove the space between the two circles we cut out. Make the rectangle about the size of our original rectangle but only about half as wide, something like this.



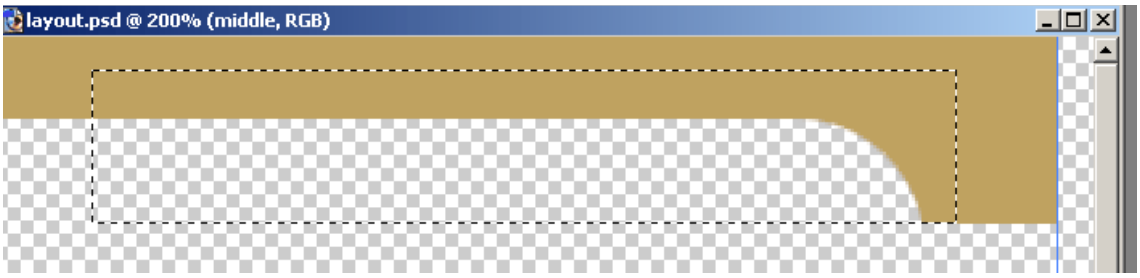
The goal here is to use this new rectangle selection to delete the bar between the two circles. We want to try and align the top corner of the rectangular selection with the very top of the circle section we removed so we have a seamless transition. You may need to zoom in to do this. Once you have it aligned, press the “delete key” to remove another chunk of our starting rectangle. Now just use your arrow keys to move the rectangular selection to the right and line it up with the circular deletion we made on the right. Once aligned, press “delete” to remove the rest of the top bar and you should have something like this.



Once again press “CTRL+D” to release the rectangular selection. Now create a second layer in your “header” layer set. Name it something like “middle” and this is where we will place another color bar that makes up part of our header. Select the rectangular marquee tool once again and make sure your new layer is selected in the layer pallet.



Now lets place another rectangle in our header. Im going to make a smaller selection on the right side underneath the upper bar. Since this is going to be in essence behind the upper bar we don't have to be terribly concerned with how tall the rectangle is but we do want to make sure the bottom edge doesn't go lower than the current top bar, you may need to zoom in and use the arrow keys to get it perfectly aligned. You should try for something similar to this.



To fill this box in I want to use one of the other colors from our color scheme, which you use is totally up to you, but for this example im going to use FFD980. Just like above you can click on the foreground color in your toolbar to get the color picker window and enter the RGB value of FFD980. Select your paint bucket and fill in rectangle.

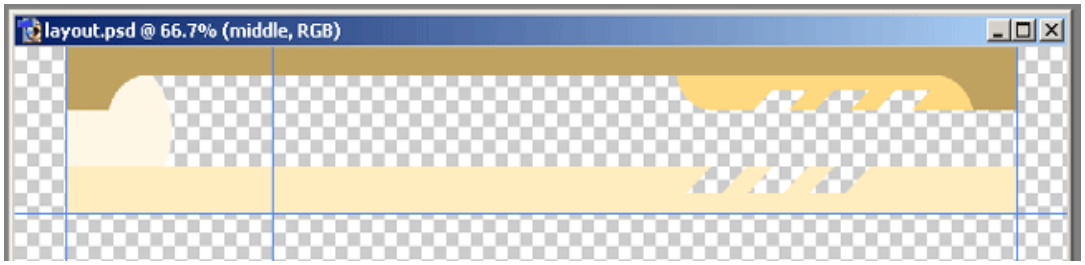
“But wait” your saying “that is in front of the top image we created”, yes it is but thats simple enough to change. Before I do that however I would like to round the lower inside corner of our newly created rectangle. I want the rounded corner to match as closely as possible the cutout size on our top bar so what I will do is select the elliptical marquee tool and on the top bar change the style to “fixed aspect ratio” so its stays as a perfect circle. Then I will use the left hand cutout as a guide for making my new circle. It may take a few tries to get the right size circle. After making one you can use the arrow keys to nudge it over to compare its size. Alternatively you can chose “Fixed” for the style and try entering various pixel sizes until you get it right. If you make a selection and its off, just press CTRL+D to delete the selection and try again. Once you got a circle close to the

right size, move it right so that it overlaps our latest rectangle so that the very bottom edge of the circle lines up exactly with our rectangle. Once lined up grab the paint bucket and fill it in with the same color used for the smaller rectangle.

Once thats filled in, go down to your layer pallet, click and hold on the “middle” layer and drag it down slightly and let off the mouse button. That should move the layer behind our first shape so you should have something like this.

Instead of outlining every single step I take I am just going to make a few more shapes and plop them in our “Header” layer set using various colors from our list. As you have seen, you can use various marquee shapes to cut sections out of filled in shapes etc. You can not only use this technique to create neat shapes but can also use it to trim shapes so they fit within the boundaries of our site. Also keep in mind its a good idea to use a new layer for each shape. If you mess something up or want to change something it makes it much easier.

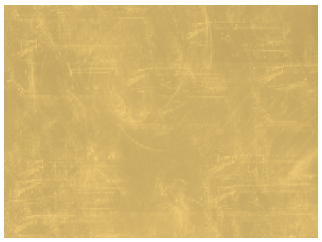
So after adding a few more shapes and making a couple cutouts in existing ones I ended up with the following.

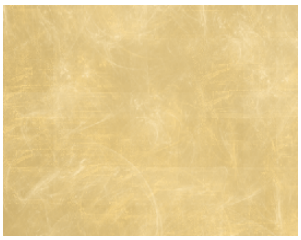


Now I want to fill in the background with a modern kind of grungy pattern. To do this I will create a new image with the same dimensions as our original, 1024x768 with a transparent background. Create a new layer and fill the whole layer with our darkest color, BFA260. Next create a new layer and select the next darkest color, FFD980.

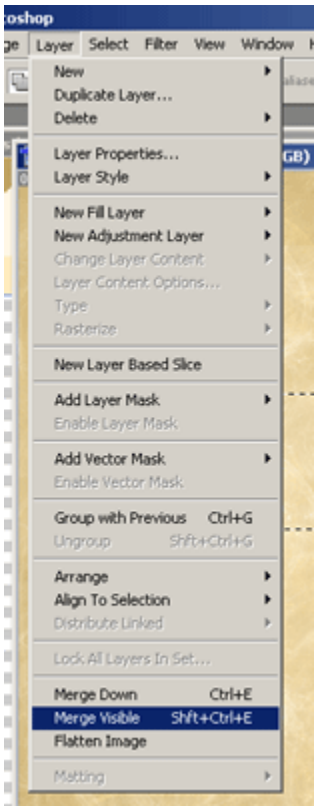
Next your going to have to be a little creative, what I did was a google search for some free photoshop brushes on the net. Brushes come in a variety of sizes and patterns. I picked a large brush (ie 300+) that had a wild pattern to it and then just moved it all around my new layer clicking randomly to add bit of color here and there. I then created

another layer, selected another color from our swatch above and repeated and after that I repeated it again on a new layer with the last and lightest of our 4 colors. Here are some pics after each layer was complete.





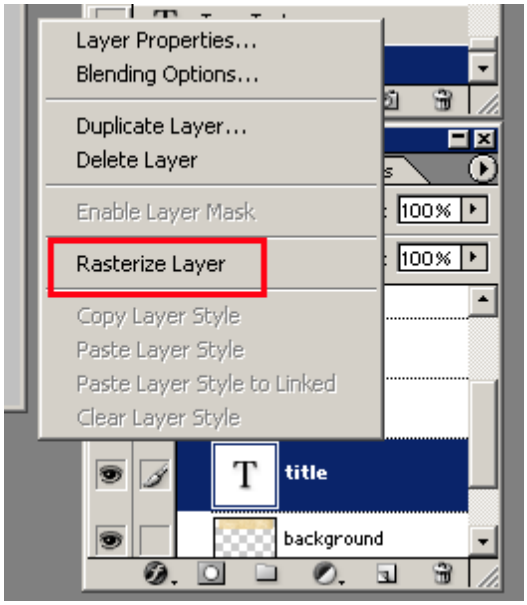
Now what we need to do is flatten the layers. This is done by going to the top menu and selecting “Layers -> Merge Visible” and what this will do is combine all the layers into one.



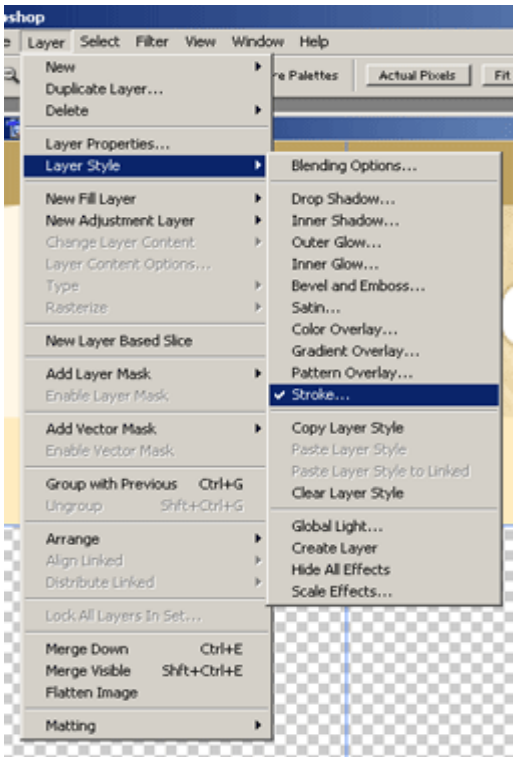
Next, grab the rectangular marquee tool and make a selection that is approximately the size of our header (ie 924x150px) or once you have the rectangular marquee tool chosen you can set the style for it to “fixed size” and simply enter the dimensions. Once you have the selection go ahead and use the arrow keys to move it to a spot on your larger canvas that has a pattern you like the most. Once you have it positioned, press the CTRL+C keys to copy the selected area.

Now go back to your original canvas and create another layer in your header set. Call it something like “background”. Make sure that layer is highlighted and then press CTRL+V to paste the selected section of your image into the new layer. You can use the “Move” tool and your mouse or the arrow keys to align you header background. If its not already there you will want to drag your “background layer” to the bottom of the header layer set so the other layer will appear over the top of it. If you would like, create a new layer, below all the others but above the background layer to place your sites name.

Letters are typically vectors so once you have the text typed out and positioned over your background right click the layer that you created your text on and from the popup menu select “Rasterize Layer”.

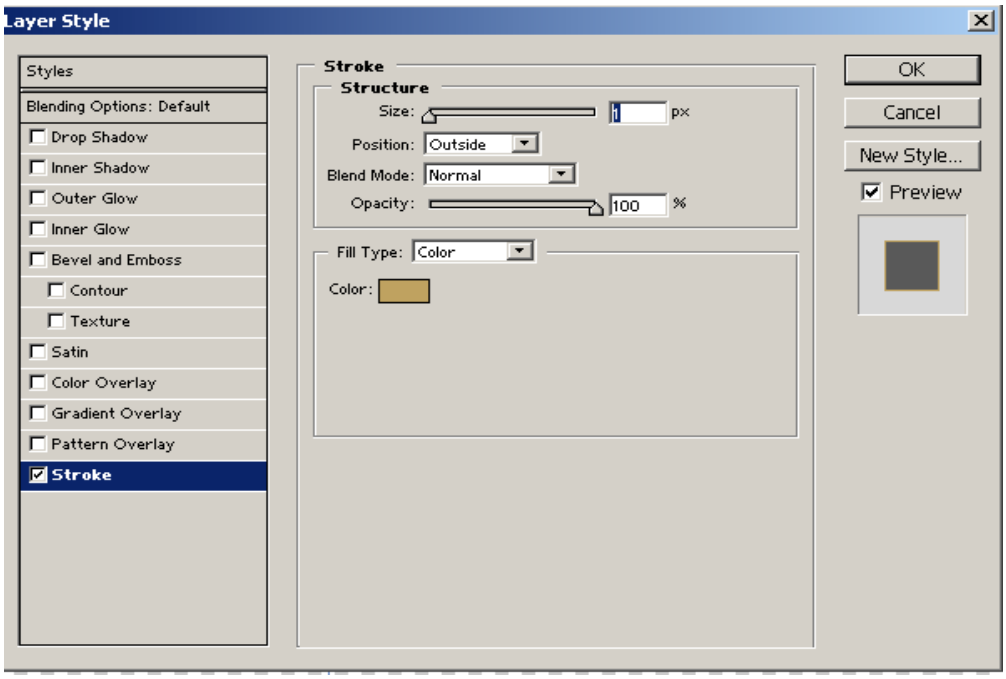


Here is what I came up with after dropping the background in behind the header and adding some text.
Not too bad but a little flat. To add a little depth lets start off with our text

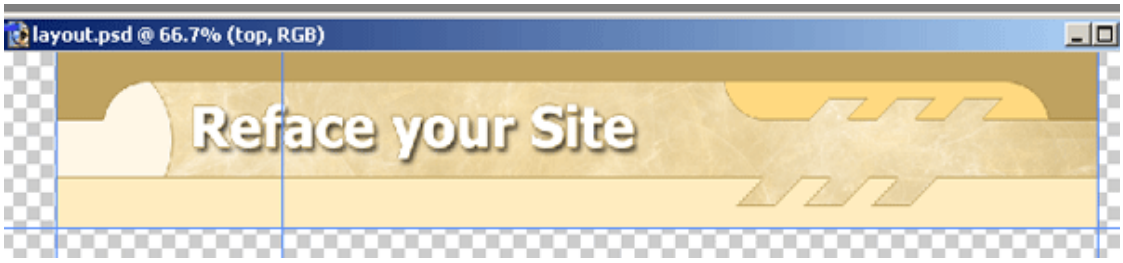


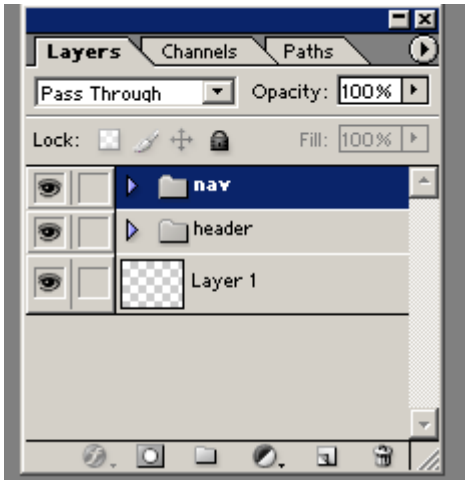
layer. Start by clicking the text layer so that its highlighted. Then goto the top menu and select “Layer -> Layer style - > Stroke”.

In the layer style window (see below) select a “size” of 1 and click in the little box of color and select a color for the stroke that will appear around your image. What I have found to work well is for all your light colors, make the color of the stroke the darkest color you have for your theme and then when “stroking” your darkest color try and find a color that has the same tint but is just a tad darker.

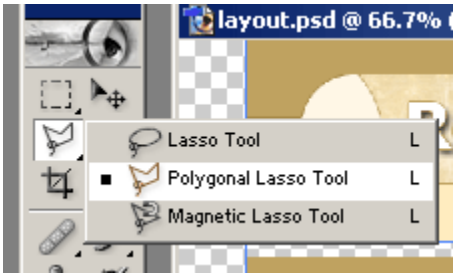


Now go ahead and add a stroke to all of the layers in your header set. You will notice that after adding a stroke to your layers you will have additional little icons in your layer pallet on each layer. You can double click the little symbol on each layer to access the various layer effects available. For the text layer I chose to also add a black drop shadow to the text and here is what I ended up with.

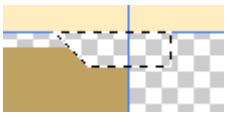




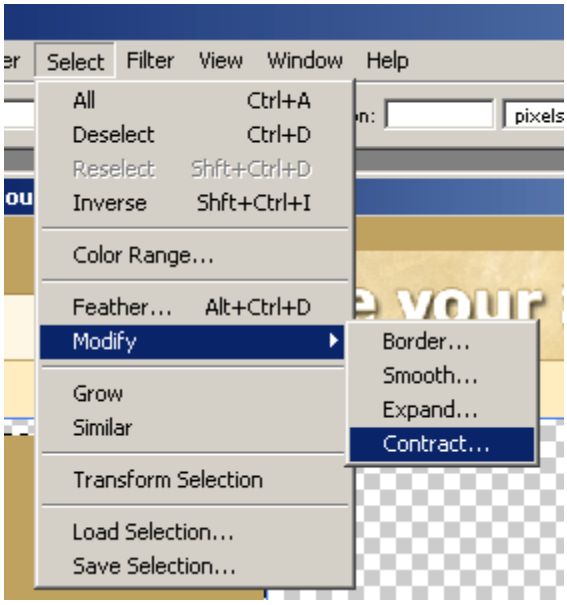
Next lets work on the body area of the site. Start by creating another layer set and call it “nav”. And within that layer set create a new layer called “nav” and click on it so its highlighted, thats the layer we want to work in next. You can collapse the sets by clicking the small arrow to the left of the folder image.

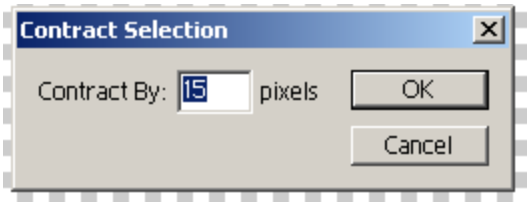


For the graphic to hold the navigation elements I am going to start by making a rectangle. The actual height of the rectangle isn't all that important but to get a better idea of how the site will look you may want to make it fill the height of our canvas but don't forget to leave room for the footer image. Im going to fill this rectangle with the color BFA260. Next I'll press CTRL+D to remove the selection and then im going to grab my “Polygonal Lasso tool”. This tool is great for cutting corners off images or removing intricate parts of images. Every time you click on you canvas this tool will place a small anchor point and draw a line connecting it to the previous anchor point. To complete a selection you need to have the final anchor point on top of the original anchor point and if you look closely a small circle will appear on the mouse pointer when you have the pointer close enough to the original anchor to complete the selection. Additionally, when using this tool you can constrain the angles allowed by holding down the shift key.

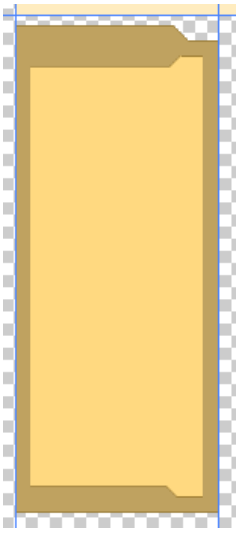


This will allow you to draw lines based on 45 degree increments. Im going to use this to cut some corners off of our navigation box to add a little character to it. Then I am going to clear the selection pressing CTRL+D. Now what I would like to do is select the whole navigation image. This is done by pressing and holding down the CTRL key and then single left clicking with your mouse on the layer that the image is in. This should give you the little dashed line all the way around the nav image you created. Since the color we chose is a bit dark I want to add a lighter element to the box as well. To accomplish this Goto the upper menu bar and select the “Select -> Modify -> Contract” and enter a value of 15 or so.



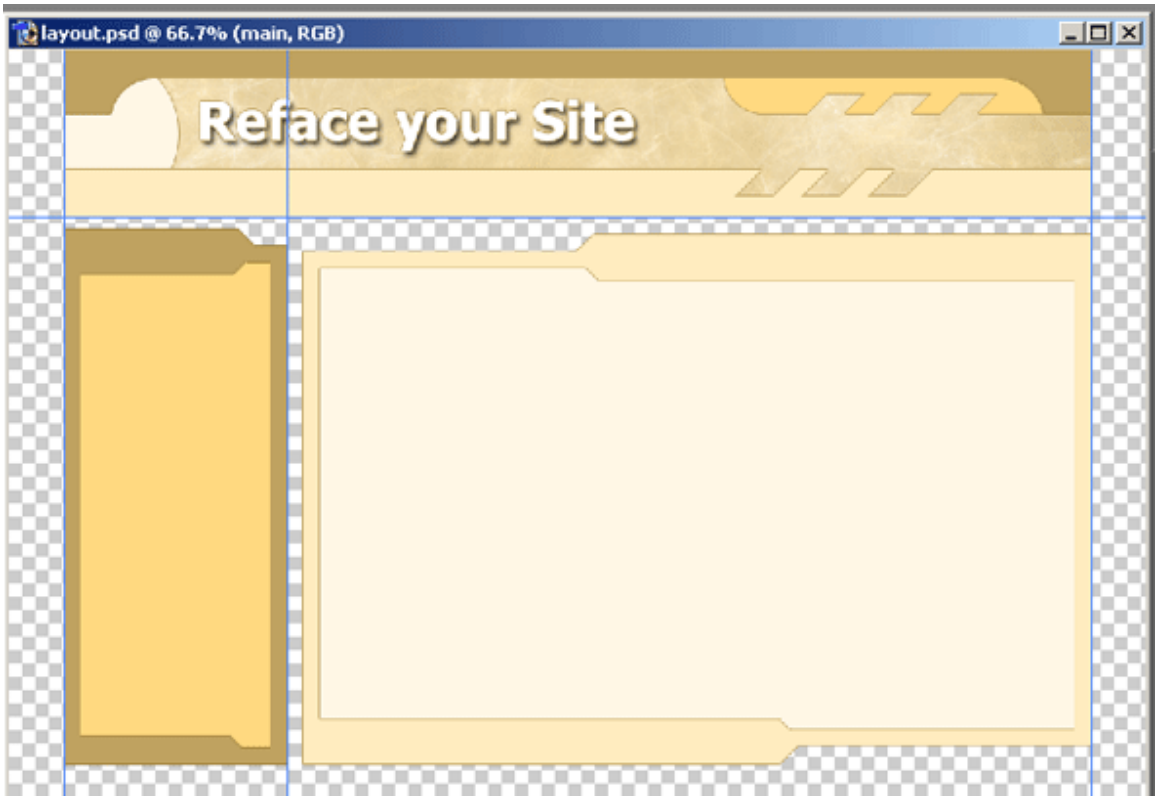


Now, I want to create another layer in the “nav” set, make sure its highlighted and then fill in our current selection, I chose to use color FFD980. Now I am going to go ahead and cut a couple corners on this box to add some additional character to the nav box. When done creating your shapes, go ahead and add a stroke to each of the layers in your nav set. Keep in mind that in this case used my darkest color as the base color for the nav box so when stroking it and the layer above it your going to need to experiment a little and chose a color thats darker to give it some depth.

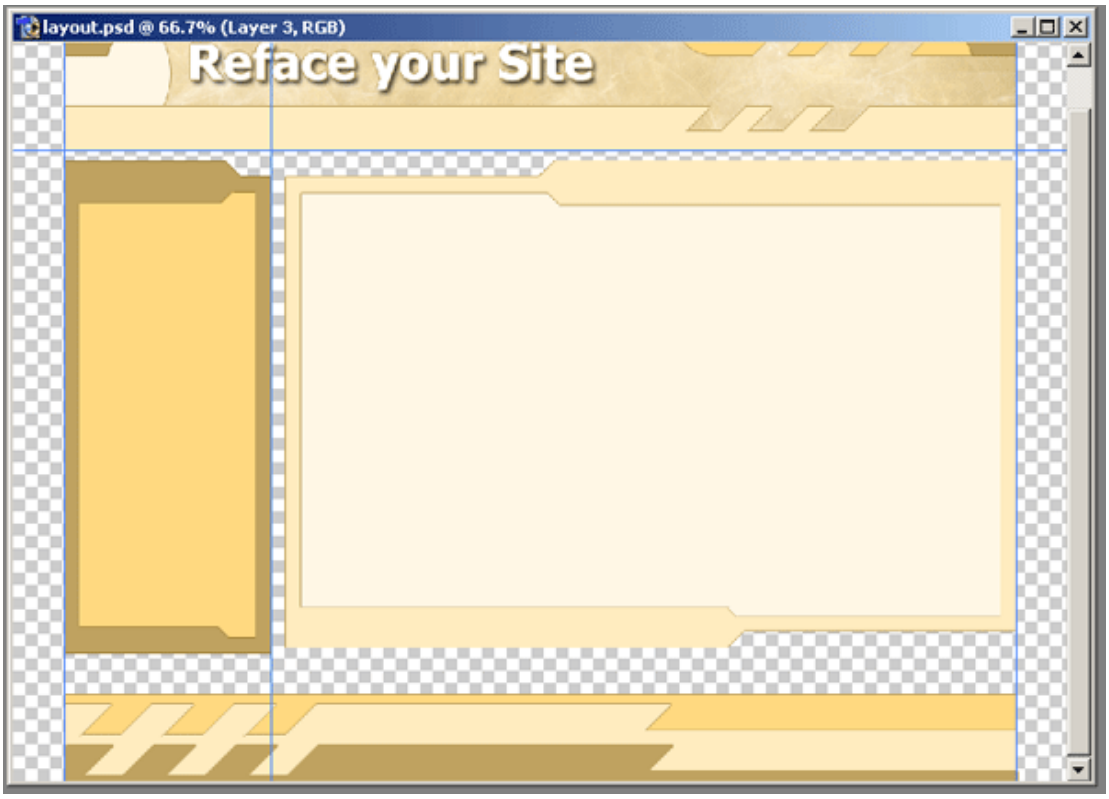


I

Now we can move over to the body section. Go ahead and create a new layer set and call it body and then create a new layer in that set to start working with. Just like with our nav section lets once again start with the rectangle marquee tool and create a rectangle that will contain the main body of our site. Once again, the height isnt all that important but its a good idea to fill up the canvas so you can get a better idea how it will look. I created a rectangle and filled it in with color FFECBF. Then I cleared the selection by pressing CTRL+D and then I used the polygonal lasso tool to cut some pieces to add a little character. Then, like on the nav image, I will select the image by pressing and holding the CTRL key while I click the layer holding the body image. Then I contacted the selection by 15 pixels and filled it in with color FFF7E5. I then released the section with CTRL+D and made a couple more cuts for character. Follow that up by adding a stroke to each layer and here is what I have.



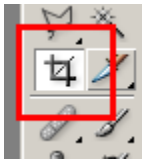
Now we are ready to build the footer and then start slicing the image into pieces. As with the other sections go ahead and create a new layer set and add a layer to it to start work in. Its always a good idea to name the set something to help you identify it if its collapsed so I chose the name footer. Then I will name my sections of the footer accordingly. Im just going to make a few shapes, some cutouts and stroke each layer like I have for all the earlier sections of the site, go ahead and experiment on your own but here is my final product. Make sure to save your image as a PSD file so it retains all its layer info.



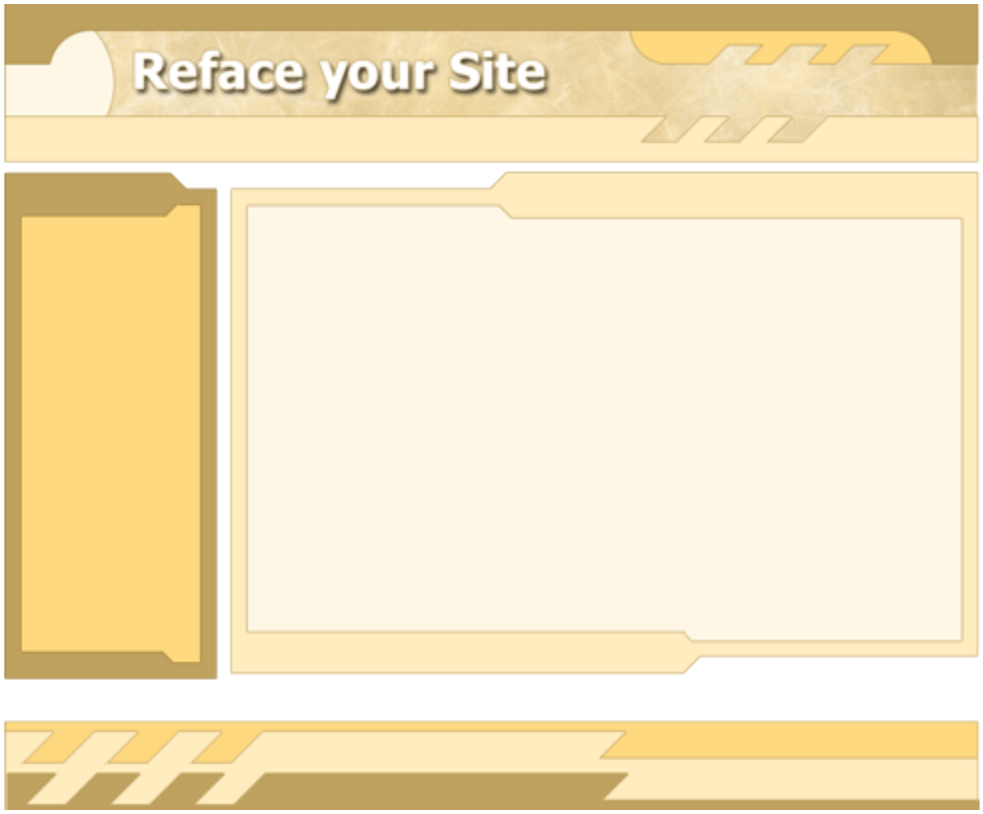
Now the fun part begin!! We get to cut the image up into smaller pieces so that we can use them as the background images in the HTML table that makes up our site. For those that are adventurous you can do the same using CSS as well, but to keep things simple we will use HTML tables for controlling the image in this tutorial.

Before we move on however we need to do a little planning. To make things simple think of the above image as an HTML table. At the top of the table we will have a single row and in that row a single data cell and the background for that cell will be our header image. With that in mind we simply need to cut off the top of our image.

Below that we have the navigation and body section of the layout. These will occupy the second row in the table but each will be in its own data cell. To add additional control over each side not only will they be in separate data cells but each will be in its own table nested within those data cells. Then lastly we will have a 3rd table row and within it our data cell that has a background image of our footer graphic. To begin the process of cutting up our image, Photoshop comes with a tool called Image Ready.



The last thing we want to do before sending the image over to Image Ready is to crop it. This is done by selecting the crop tool and much like the Marquee tool we select a point and drag a rectangle around the part we want to keep. Don't worry if you don't get it perfect the first time, after you drag out the rectangle and release the mouse button the outlined area will have little square anchors. You can click and drag these with your mouse to adjust the selected area. If you get close to our blue guides however it will snap to them. If you crop on the blue guide however it will cut off the stroke we added on the outer edges so you will want to grab the little anchors and move the sides out just a tad before doing the crop. After cropping my final image in Photoshop looks like this, you should just have a small margin of space on either side..

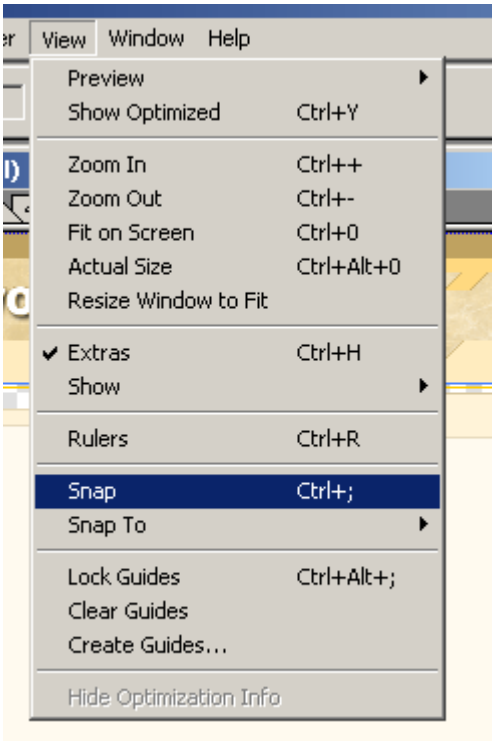


Now, we can send the currently selected image directly to image ready by clicking the “Jump to ImageReady” button at the bottom of the toolbar.

Once Image Ready opens we should see our new interface in all its glory. Image Ready looks a lot like Photoshop and has many of the same tools but is more geared toward....well...making your Image Ready for the web.

We have two main goals here in Image Ready, the first is to cut up or slice our image. The second is to save the image slices in a nice neat table that when viewed will look like our complete image.

To start the slicing process, select the "Slice Tool" from your tool bar. When over the image the mouse cursor will turn into a little knife. Using the slice tool is very much like using the Marquee tool in Photoshop. You click on a starting point and drag out a rectangle which will represent a slice of our image. As you do this you will notice the current slice will be blue and it will snap to the grid lines we set up in photoshop. Just like when we did the crop however be careful where you make your cuts. The goal for our first cut is to select with our slice tool the whole header of the page. We want its bottom border to be just below our header but above our navigation and body images. What I usually do is start in the upper left and drag down to the right until I get the selection im looking for. If you find that your cut is snapping to your grid lines and is preventing you from placing the cut in between images you can goto the top menu bar and select "View -> snap" to turn on/off the snap to grid function.



In my case the nav and body image are fairly close to the header so I had to turn off snap so I could position my cut between them.

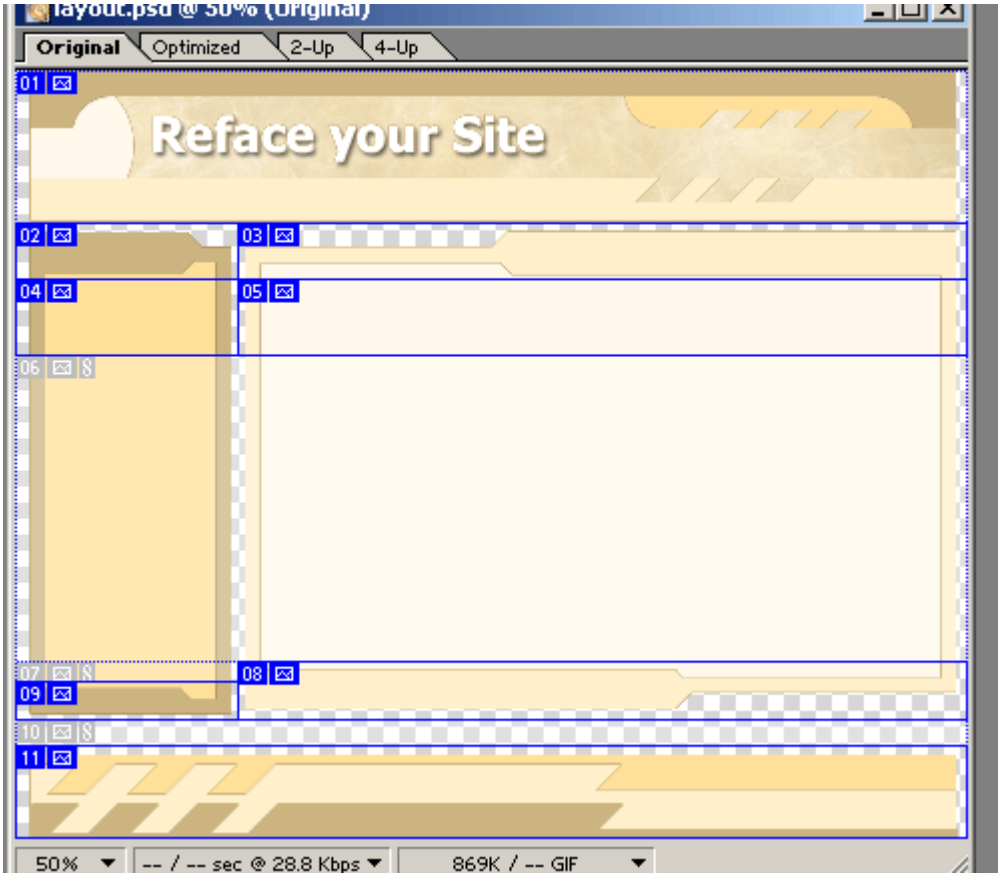
As you make your cuts you will notice that in the upper left corner of each slice there is a number and as you make cuts, Image Ready may make additional slices automatically. This needs to be done to keep your image in tact when it is cut up and redisplayed via HTML in a table.

After we make our first slice which should encompass the whole header of our site we need to consider our next move. The headers of the navigation image and the body image only need to be displayed once. But what happens if the content in these images is smaller than the current image size of much longer?

Whats going to happen is we are going to use the various image slices as backgrounds to our HTML data cells. The default behavior is to tile if a static size for the data cell is not specified in the HTML that makes up the table. With that in mind we are going to want to make additional slices, one for the nav header, one for the body header, one for the nav body, another for the body of the body and finally one for the footer of the nav and body images.

When making your slices if after placing one you find that it needs some adjusting you can move the sides of it much like we did with the crop tool. Its a good idea to make sure each slice is set perfect before you move on to the next. If for some reason you do need to adjust a slice you already made, you can do so by holding down the CTRL key and click on the slice you want to make the active one, then you can adjust it as needed. Keep in mind if you do this you may need to adjust the slices that border it or Image Ready will make those small gaps into new slices automatically.

Here is my final sliced up image.



Ok lets take a look at my slices, you will notice each has been given a number, some are blue and some are grey. Blue slices are ones that I made and the grey slices are ones automatically done by Image Ready to make tableizing your image easier.

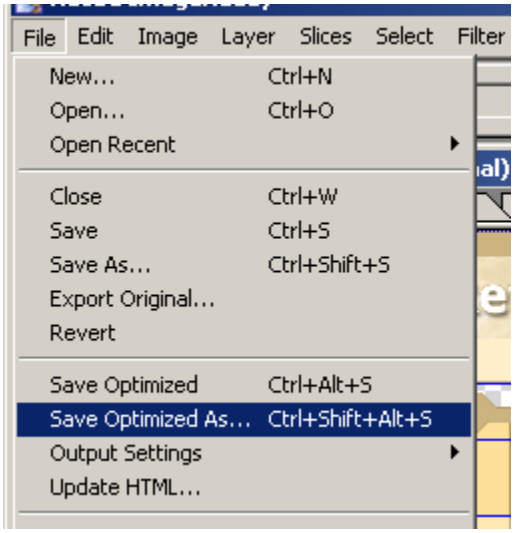
Now, you may be asking why I have the body section of the nav and body image broken into 2 slices. Number 4 and 5 are the slices I made while number 8 was done automatically. As I mentioned above the background image in data cells will wrap or tile So really all we need is one section of the image from the body portion of the nav and body images. I could have used a larger slice but in terms of data transfer, there is no need if we are only going to need a 20 pixel high image in a table cell holding a link why would we want to load an image that is 100pixels high?

In essence what is going to happen is that in the final site layout, I wont be using the number 6 slice at all. For every link I place in the navigation image for example I will have a new data cell and simply reuse the number 4 slice as the back ground. They will stack seamlessly. That means that the nav image will grow and shrink in height based on how many links or items are in it. The same applies to the body image.

One last thing to note is that when I created the images for the nav and body, the bottoms didn't line up. When creating the slices for the bottoms of each I wanted to make sure I got the whole bottom of each image with just a sliver of space above it. This means that the two slices are different heights and Image Ready automatically added slice number 8, which is a slice I will not be using in the final product.

Now that you have your image all sliced up its time to save it. For this step I recommend creating a directory/folder somewhere on your local drive then saving it to that folder. Image ready is going to cut up your image and turn it into multiple images and will also create and HTML file containing a table that holds these images so when you view the HTML file you will see what appears to be a single image.

To save your image goto the top menu, click “Save-> Save Optimized As” and you will get a popup window asking for the location to save to. Select the folder you created to save your files in and hit “save” Make sure to note the HTML filename being saved, by default it is layout.html.



Now to see your handy work go ahead and open your browser. On your menu across the top of your browser select “File -> Open File” to get the file browser window. Navigate to the folder you saved your slices to in Image Ready and select layout.html file that Image Ready created. What you should get is a nice clean page containing your interface. It looks like one solid image but in actuality its an HTML table that contains the slices of your image. Go ahead and use your browsers “View Source” feature to see the HTML that makes up the page.

Now comes the tricky part, integrating this new image into our 68classifieds templates. What you will want to do is within your templates directory holding your 68classifieds script is create a new subdirectory. For this example I am going to call mine "orange". The end result will be if your looking at the file/directory structure for your 68classifieds you will have a directory called /templates and within that 2 directories, one will be the default template directory and then your new orange directory.

/templates/default
/templates/orange

You may have additional template directories if you have played around some. What we want to do now is copy the contents of an unmolested /default template into our orange directory. If you only have the default template on your web host and you have made changes to it I highly recommend you copy the contents of /templates/default from your original zip file and copy that to your new orange directory so you have a clean template to start with.

Once thats done we are going to be working with the files in the /templates/orange directory. For starters lets rename the file layout.tpl.php to something like layout.tpl.php.bak. Once that file is renamed, rename your local file "layout.html" that was created by Image Ready. We now want to call it layout.tpl.php and once renamed, upload it into your /templates/orange directory. Next we need to take all of the images created by Image Ready and upload them to our templates image directory /templates/orange/images. These files are located in a subdirectory called /images on your local drive where you told Image Ready to save them.

Once you get your layout.html file copied to /templates/orange and renamed to layout.tpl.php and your images moved over to /templates/orange/images you should be able to point your browser to layout.tpl.php and see your new interface. Keep in mind that you will need to use the complete path to the file since its not being called by a php script in the root of your classified site. It would be something like this if your classifieds are installed in the root of your web host.

http://mydomain.com/templates/orange/layout.tpl.php

Now its time to merge the original template.tpl.php with our new one from Image Ready. The 1st thing we need to do is trim some fat from our new layout.tpl.php file so go ahead and open that file in your HTML editor. Its all pretty straight forward and is basically an HTML table with a bunch of images in it. For starters we need to go through and instead of having the images inside the cells we need to actually have them a background for the cell.

If we look at our very 1st table row we will see something like the following.

```
---------------------------------------- html ----------------------------------------
<tr>
        <td colspan=2>
        <img src="images/layout_01.gif" width=950 height=152 alt=""></td>
</tr>
----------------------------------- end html -----------------------------------
```

We need to move the images from within the actual cell and make them background images for the table data cell itself. That is done by using the background attribute for the data cell. We want to change the above HTML to the following and save our changes.

```
 ---------------------------------------- html ----------------------------------------
<tr>
        <td colspan=2 background="images/layout_01.gif" width=950 height=152 alt="">
        </td>
</tr>
----------------------------------- end html -----------------------------------
```

After making the above change you should be able to once again look directly at your layout.tpl.php file as we did above and not see any difference, however if you look at the source via your browsers “View -> source” feature you should see that the first row in the table is now a background image.

Now what we want to do is go through the rest of the file and move the image from the cell contents to the cell itself just as we did above, when done save the file and try and view it again. It should look the same but once again viewing the source should show the images in the cell backgrounds.

Once thats done and your new interface still looks in order when viewed in a browser, its time to start moving over our template information. Start by replacing the header info, go ahead and open up the original layout file, layout.tpl.php, and copy the whole top of the file all the way down to the </head> closing tag. Place this into your new layout.tpl.php replacing the existing HTML.

Next we will need to add our nested tables to the HTML file created by Image Ready. Since we want both the navigation image and body image to dynamically size based on content and independently of each other we need to nest 2 tables in the second table row of our new layout.tpl.php file.

Looking at the body section of the html created by Image Ready we have the following HTML that we modified so the cells contained background images.

```
-----------------------------------layout.tpl.php ------------------------------------------------
<body bgcolor=#FFFFFF leftmargin=0 topmargin=0 marginwidth=0 marginheight=0>
<!-- ImageReady Slices (layout.psd) -->
<table width=952 border=0 cellpadding=0 cellspacing=0>
	<tr>
		<td colspan=2 background="images/layout_01.gif" width=952 height=152  alt="">
		</td>
	</tr>
	<tr>
		<td background="images/layout_02.gif" width=221 height=56  alt=""></td>
		<td background="images/layout_03.gif" width=731 height=56  alt=""></td>
	</tr>
	<tr>
		<td background="images/layout_04.gif" width=221 height=76 alt=""></td>
		<td background="images/layout_05.gif" width=731 height=76 alt=""></td>
	</tr>
	<tr>
		<td colspan=2>
			<img src="images/layout_06.gif" width=952 height=306 alt=""></td>
	</tr>
	<tr>
		<td>
			<img src="images/layout_07.gif" width=221 height=20 alt=""></td>
		<td rowspan=2>
			<img src="images/layout_08.gif" width=731 height=58 alt=""></td>
	</tr>
	<tr>
		<td>
			<img src="images/layout_09.gif" width=221 height=38 alt=""></td>
	</tr>
	<tr>
		<td colspan=2>
			<img src="images/layout_10.gif" width=952 height=26 alt=""></td>
	</tr>
	<tr>
		<td colspan=2>
			<img src="images/layout_11.gif" width=952 height=94 alt=""></td>
	</tr>
</table>
<!-- End ImageReady Slices -->
---------------------------------- End layout.tpl.php ---------------------------------------------
```

Ok, now if you remember I said we were not going to use certain images in our layout, with that in mind, we can go ahead and trim out the images, cells and rows that hold them. If we look back at the sliced up image the grey slices are ones that will not be used, these are slice 6, 7 and 10.

If we look at the source for image slice layout_06.gif we see that it is in its own row, go ahead and remove the row, and the data cell within it. Next lets look for layout_07.gif in the HTML. As you can see layout_07.gif and layout_08.gif are both in the same table row. We wont be using 7 but we want to keep 8 so we cant get rid of the whole row. Simply get rid of the cell that makes up slice 7. Lastly look for layout_10.gif which is another slice we wont be using. It resides in its very own row so we can go ahead and cut out that row and its data cell.

Your HTML should now look like this.

```
---------------------------------- layout.tpl.php ----------------------------------------------
<!-- ImageReady Slices (layout.psd) -->
<table width=952 border=0 cellpadding=0 cellspacing=0>
	<tr>
		<td colspan=2 background="images/layout_01.gif" width=952 height=152  alt="">
		</td>
	</tr>
	<tr>
		<td background="images/layout_02.gif" width=221 height=56  alt=""></td>
		<td background="images/layout_03.gif" width=731 height=56  alt=""></td>
	</tr>
	<tr>
		<td background="images/layout_04.gif" width=221 height=76 alt=""></td>
		<td background="images/layout_05.gif" width=731 height=76 alt=""></td>
	</tr>

	<tr>
		<td rowspan=2 background="images/layout_08.gif" width=731 height=58
alt=""></td>
	</tr>
	<tr>
		<td background="images/layout_09.gif" width=221 height=38 alt=""></td>
	</tr>
	<tr>
		<td colspan=2 background="images/layout_11.gif" width=952 height=94
alt=""></td>
	</tr>
</table>
<!-- End ImageReady Slices -->
---------------------------------- end layout.tpl.php ----------------------------------------------
```

If you once again look at your layout.tpl.php file in your browser you will see the header looks fine and the footer looks fine but the body portion is all messed up. This is because we got rid of layout_07.gif and its cell, but that will be all straighted out once we create our nested tables.

From looking at our current HTML we can tell that the first table row contains our header image and the second row is where our nav and body images start. It is in this second row that we want to create our 2 new tables each table will reside in the data cell on the second row.

As it sits now our second table row looks like this.

```
--------------------------- html source -------------------------------------------------------
<tr>
	<td background="images/layout_02.gif" width=221 height=56  alt=""></td>
	<td background="images/layout_03.gif" width=731 height=56  alt=""></td>
</tr>
--------------------------- end html source -------------------------------------------------------
```

We need to change it by pulling our background images from the data cells but we want to leave the width specification for the data cell in tact, at the same time its good to specify valign. Then within each cell create our tables and in each table, we want to have

3 rows for starters, one for the head of each image, one for the body and one for the footer. Make sure to specify you want no padding, no borders and a static width that matches the data cell its in is also a good idea. When filling in the image filenames for the new data cells in our nested tables feel free to look back at your sliced up image for reference on which image is where.

What you should end up with is similar to this.

```
---------------------------- html source ------------------------------------------------------------
<tr>
 <td width=221 valign="top">
  <table border="0" cellpadding="0" cellspacing="0" width="221">
   <tr>
    <td background="images/layout_02.gif" width=221 height=56 alt="">
    </td>
   </tr>
   <tr>
    <td background="images/layout_04.gif" width=221 height=76 alt="">
    </td>
   </tr>
   <tr>
    <td background="images/layout_09.gif" width=221 height=38 alt="">
    </td>
   </tr>
  </table>
 </td>
 <td width=731 valign="top">
  <table border="0" cellpadding="0" cellspacing="0" width="731">
   <tr>
    <td background="images/layout_03.gif" width=731 height=56 alt="">
    </td>
   </tr>
   <tr>
    <td background="images/layout_05.gif" width=731 height=76 alt="">
    </td>
   </tr>
   <tr>
    <td background="images/layout_08.gif" width=731 height=58 alt="">
    </td>
   </tr>
  </table>
 </td>
</tr>
---------------------------- end html source ------------------------------------------------------
```

After inserting the two nested tables holding the body of the page into the second table row of the main table we can go ahead and get rid of the rest of the tables rows with the exception of the last one which is our footer. So your final main table should look like this.

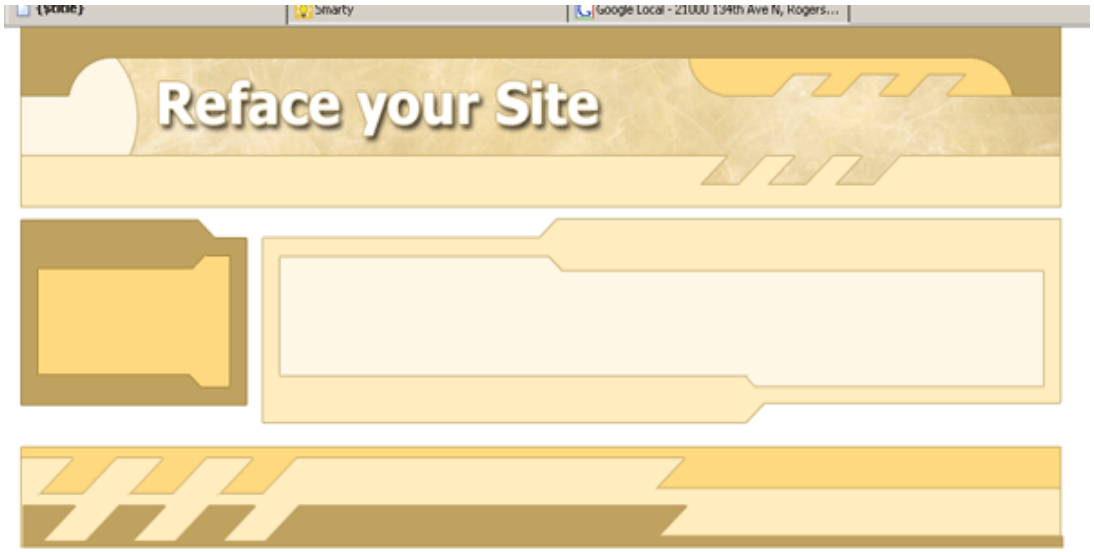
```
----------------- end html source -----------------------------------
<table width=952 border=0 cellpadding=0 cellspacing=0>
        <tr>
                <td colspan=2 background="images/layout_01.gif" width=952 height=152  alt="">
                </td>
        </tr>
```

```
	<tr>
		<td width=221 valign="top">
			<table border="0" cellpadding="0" cellspacing="0" width="221">
				<tr>
					<td background="images/layout_02.gif" width=221
height=56 alt="">
					</td>
				</tr>
				<tr>
					<td background="images/layout_04.gif" width=221
height=76 alt="">
					</td>
				</tr>
				<tr>
					<td background="images/layout_09.gif" width=221
height=38 alt="">
					</td>
				</tr>
			</table>
		</td>

		<td width=731 valign="top">
			<table border="0" cellpadding="0" cellspacing="0" width="731">
				<tr>
					<td background="images/layout_03.gif" width=731
height=56 alt="">
					</td>
				</tr>
				<tr>
					<td background="images/layout_05.gif" width=731
height=76 alt="">
					</td>
				</tr>
				<tr>
					<td background="images/layout_08.gif" width=731
height=58 alt="">
					</td>
				</tr>
			</table>
		</td>
	</tr>

	<tr>
		<td colspan=2>
			<img src="images/layout_11.gif" width=952 height=94 alt=""></td>
	</tr>
</table>
---------- end html source ----------------------------------
```

Now save the file and once again view it in your browser. It should have all lined back up nice and pretty like this.



The only difference is that our nav and body images are much shorter than they were in the initial image we created. As I mentioned previously this wont be a problem. The body section of the navigation and the body image is made up of a single cell and for each item we want to appear in these locations we will specify the same background image so that each of these will grow or shrink as needed based on content.

Now that we have our nested tables taken care of its time to start adding or sites navigation. Looking at just the nested table that makes up our navigation we can see its made up of three images.

layout_02.gif --> the navigation image headers
layout_04.gif --> the navigation body
layout_09.gif --> the navigation footer

For starters lets once again refer back to the original layout.tpl.php file that we renamed to layout.tpl.php.bak and lets find the very first navigation item. You can find it by looking for this line.

```
<!-- // Navigation // -->
```

Just under it will be the navigation table tag and below that the first row and cell which is the navigation header which looks like this.

```
<tr>
  <td class="navTitle">{$smarty.const.LANG_TPL_NAVIGATION}</td>
</tr>
```

what we need to do is apply the class navTitle to our current nav data cell and add the smarty constant {$smarty.const.LANG_TPL_NAVIGATION} to our data cell. Doing so should look like this.

```
<tr>
  <td class="navTitle" background="images/layout_02.gif" width=221 height=56 alt="">
  {$smarty.const.LANG_TPL_NAVIGATION}
  </td>
</tr>
```

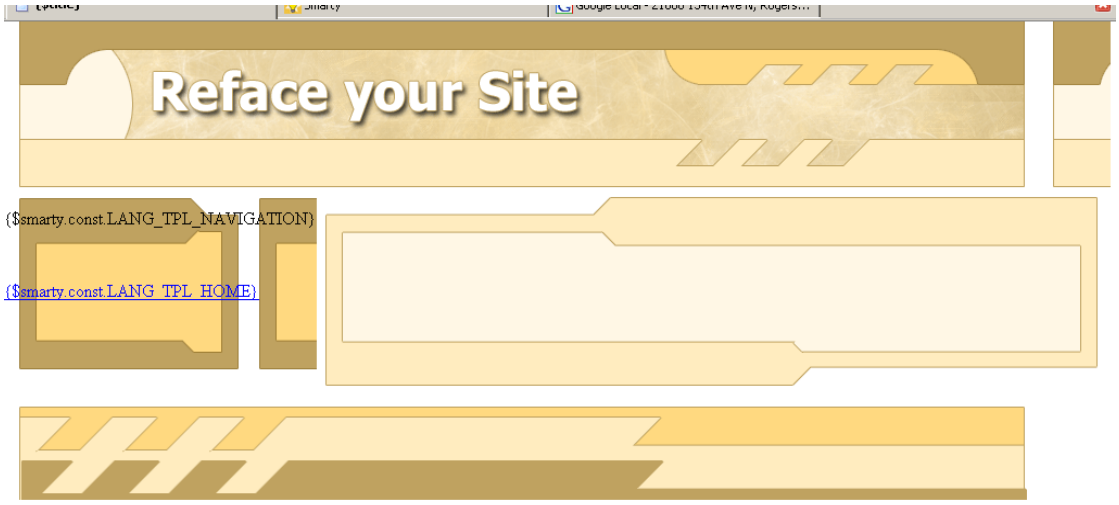
Now, if we view our page in a web browser again we will see that across the top of our navigation image we have {$smarty.const.LANG_TPL_NAVIGATION} and it likely stretched out the cell and messed things up. When we point a browser to our site and in turn a php script that in will call this layout file the above smarty variable will be replaced by the word “Navigation” for the language you have set via your sites admin settings.

Lets move on and finish filling in the nav table with the rest of our fields so refer back to our original template file and we can see that it also has a CSS class associated with it so lets start by adding that to our table data cell. Next we can see that cell content is an image and text which is also a link, so go ahead and add that to our nav table as well. Our nested navigation table should now look like this.

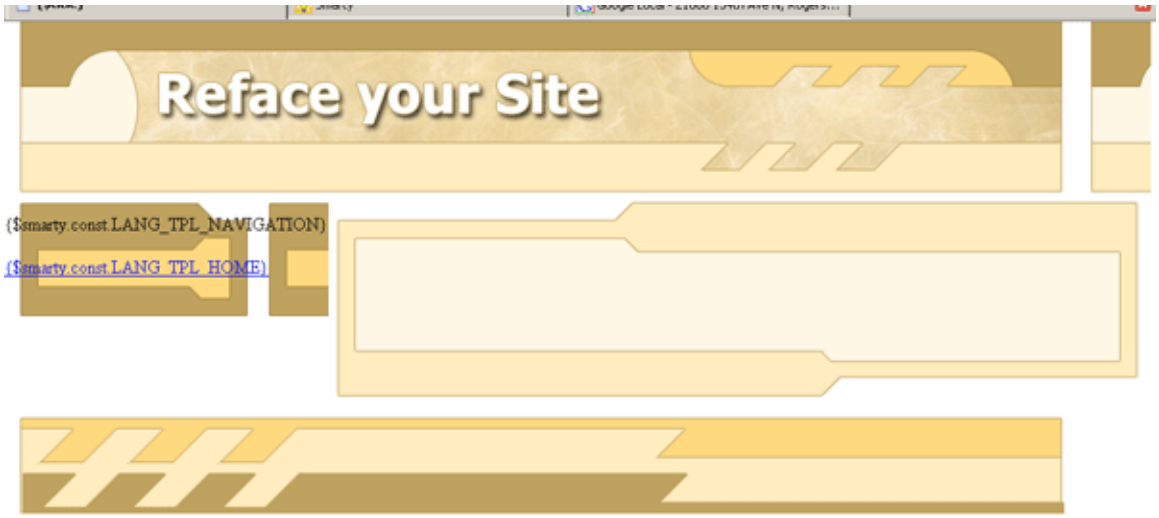
```
<table border="0" cellpadding="0" cellspacing="0" width="221">
 <tr>
  <td class="navTitle" background="images/layout_02.gif" width=221 height=56 alt="">
  {$smarty.const.LANG_TPL_NAVIGATION}
  </td>
 </tr>
 <tr>
  <td class="navlist" background="images/layout_04.gif" width=221 height=76 alt="">
  <img src="images/bullet.gif" alt="" /><a href="{$smarty.const.URL}/index.php">
{$smarty.const.LANG_TPL_HOME}</a>
  </td>
 </tr>
 <tr>
  <td background="images/layout_09.gif" width=221 height=38 alt="">
  </td>
 </tr>
</table>
```

Once again go ahead and save your layout.tpl.php file and point a browser to it. You should now see the smarty variable for the header and the smarty variable for the first nav link on the page.

It will likely look a bit out of whack still simply due to the length of the text placed in the data cell. This should straighten out when the smarty variables are substituted for the actual value they represent and we start to play with the style sheet to keep everything in line. Here is what mine looks like now.

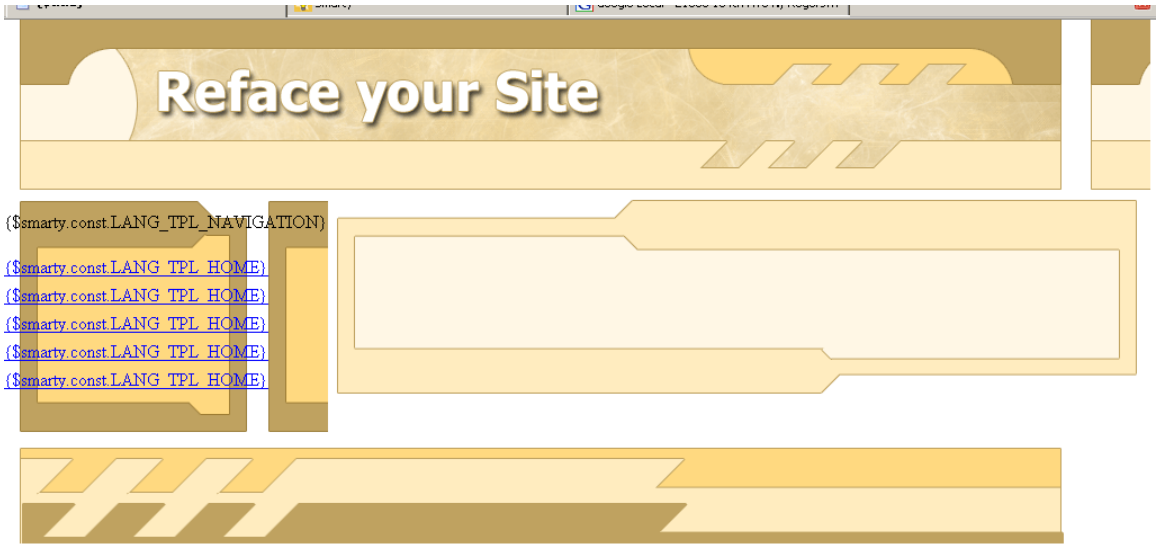


Not much to look at currently but what I wanted to point out was how thick (or tall if you will) the current Nav image is. The nav header and footer image due to their having various shapes in them, should maintain their width and height settings However, since the middle cell holding our Home link is a basic image we can adjust its size in the table. Currently it has its height specified as 76 pixels. Go ahead and play with its height setting and see how it changes the height of the Nav image. Ideally you should pick a height that will give you a nice space above and below the text you place in the cell. I went in and changed just the height setting for the middle cell in the nav table to 25 and here is how it looks now.



As we can see, the height of the nav table shrunk considerable. Now go back to our original backup of the layout file and count how many more links there are. You will have at least 5, maybe more if you have added pages via Admin or selected existing ones to be published.

Lets just start with the 5 we know we will have. The easiest way to create these 5 new rows and cells would be to copy (ie highlight, CTRL+C) the current Home row and data and then simply paste it in below the current row 5 more times. Now go ahead and save your layout.tpl.php file and refresh your browser again. You can see the nav table is seamless and has grown to accommodate our 5 new links. You should have something like this.



Now simply refer to the original layout.tpl.php file that we renamed and cut/paste the proper links into your 5 new nav cells. When done your nav HTML table source should look like this.

```
---------------------------------  nav table source ------------------------------------------
<table border="0" cellpadding="0" cellspacing="0" width="221">
  <tr>
    <td class="navTitle" background="images/layout_02.gif" width=221 height=56 alt="">
    {$smarty.const.LANG_TPL_NAVIGATION}
    </td>
  </tr>
  <tr>
    <td class="navlist" background="images/layout_04.gif" width=221 height=25 alt="">
    <img src="images/bullet.gif" alt="" /><a href="{$smarty.const.URL}/index.php">
    {$smarty.const.LANG_TPL_HOME}</a>
    </td>
  </tr>
  <tr>
    <td class="navlist" background="images/layout_04.gif" width=221 height=25 alt="">
    <img src="images/bullet.gif" alt="" /><a href="{$smarty.const.URL}/search.php">
{$smarty.const.LANG_TPL_SEARCH}</a>
    </td>
  </tr>
```

```
  <tr>
    <td class="navlist" background="images/layout_04.gif" width=221 height=25 alt="">
    <img src="images/bullet.gif" alt="" /><a href="{$smarty.const.URL}/category.php">
{$smarty.const.LANG_TPL_BROWSE}</a>
    </td>
  </tr>
  <tr>
    <td class="navlist" background="images/layout_04.gif" width=221 height=25 alt="">
    <img src="images/bullet.gif" alt="" /><a href="{$smarty.const.URL}/
toplistings.php?pg=featured">{$smarty.const.LANG_TPL_FEATURED}</a>
    </td>
  </tr>
  <tr>
    <td class="navlist" background="images/layout_04.gif" width=221 height=25 alt="">
    <img src="images/bullet.gif" alt="" /><a href="{$smarty.const.URL}/toplistings.php?pg=new">
{$smarty.const.LANG_TPL_NEW}</a>
    </td>
  </tr>
  <tr>
    <td class="navlist" background="images/layout_04.gif" width=221 height=25 alt="">
    <img src="images/bullet.gif" alt="" /><a href="{$smarty.const.URL}/toplistings.php?pg=top">
{$smarty.const.LANG_TPL_TOP}</a>
    </td>
  </tr>
  <tr>
    <td background="images/layout_09.gif" width=221 height=38 alt="">
    </td>
  </tr>
</table>
------------------------------ end nav table source ---------------------------------------------
```

That covers the basic nav links that appear all the time. If you look below them in the original layout template you will see we now have a smarty {foreach} statement. What this does is it checks to see if you have created or set any additional pages from within admin to have their links displayed in the nav menu. To add this to our new nav menu you can once again just cut/past the foreach statement and its closing tag and place it above the last row in our nav table. Remember we want it above the last row because the last row is actually the nav footer. Then, cut and past the second to the last row/cell details from our current nav table and place it in between the smarty opening {foreach} and closing {/foreach} tags, then modify the cell contents to mirror what is in the original renamed template file. You should end up with something like this just above your last row of the nav table.

```
{foreach from=$templatepages item="entry"}
  <tr>
    <td class="navlist" background="images/layout_04.gif" width=221 height=25 alt="">
    <img src="images/bullet.gif" alt="" /><a href="{$smarty.const.URL}/pages.php?page=
{$entry.pageID}">{$entry.pPageTitle}</a>
    </td>
  </tr>
{/foreach}
```

If we refresh our browser page again its going to show the foreach statement above the actual nav table because the smarty variables are not being parsed, don't be alarmed, this will clean itself up when the template is assigned from the script.

If we look at the original template file again, just below the {foreach} statement we can see that there is one more link for “contact us” so go ahead and once again copy/past one of the existing table rows/cells and paste it below the {/foreach} and above the last row in the nav table, then update its contents with what is in our original template file like you did with all the previous links.

Now, if we look back at the original template file again we can see that after the “contact us” link we have another row and cell that has a CSS class of navTitle and is used as a header and it has the member option links after it. Im going to stray from the original layout slightly and actually create a new table for the member section. This will be done by closing the existing nav table and creating a new one directly under it and before the main tables closing data tag. To make things easy, you can actually just copy the whole current nav table and place it directly under the current one. Since the member section doesn't have the smarty {foreach} {/foreach} in it why dont you go ahead and prune that out of your newly pasted table but you can leave the row/cell between the smarty tags in tact.

Now simply refer back to the original layout file and copy the contents of the first data cell and place it into the top cell of our newly pasted table. Now looking back at the original layout template again we see just below the member section title we have a smarty {if} statement. This does a check to see if a member is logged in and if they are it displays the nav links between the opening {if} and the closing {/if} tag. Within there is also another if statement to see if the site is set to allow users to save favorites and also an {else} section as well. At this point all we need to do is simply take things one HTML row at a time copying over the smarty tags and placing them between the rows as we go and update the contents of our new layouts cells with the info from the original layout file.

That means we will have smarty {if}{/if} tags around the second cell which holds a link to user favorites. Then we have 4 more rows/cells with usercheckout, userbrowselistings, userstore, and logout links respectively followed by an smarty {else} tag. Following the {else} tag we have three more rows/cells and finally a closing smarty {/if} statement. After that we should have one last row/cell that contains the footer image and finally a closing table tag.

Next we can start working on the body section, which is actually much easier and we already have the table itself created. All we need to do now is look at the original layour template and we can see that the body of the site is simply stated as

```
<!-- // Content // -->
        {* This includes the content portion *}
        {include file=$body}
<!-- // End Content // -->
```

Our new layout however is inside of an image so we need to simply copy/paste the line that includes the variable $body and place it inside the middle row of our body table. The whole table source for the body should now look like this.

```
---------- body table source ----------
<td width=731 valign="top">
  <table border="0" cellpadding="0" cellspacing="0" width="731">
    <tr>
      <td background="images/layout_03.gif" width=731 height=56 alt="">
      </td>
    </tr>
    <tr>
      <td background="images/layout_05.gif" width=731 height=76 alt="">
      {* This includes the content portion *}
      {include file=$body}
      </td>
    </tr>
    <tr>
      <td background="images/layout_08.gif" width=731 height=58 alt="">
      </td>
    </tr>
</table>
---------- end body table source ----------
```

Now we are almost ready to view it live and work on the CSS portion to get everything lined up but before we do that we need to go through our new layout.tpl.php and make sure the path to images etc are all correct when used with the actual script files.

68classifieds uses variables in its paths to images etc to allow us to easily switch between templates via the admin side of the site so to make sure our new layout file works when called from the underlying php script we can once again look at the original layout template. By default the bullets used for the nav items is stored in the images directory in the root of the install directory. Since these bullets dont really match our color scheme we can make our own and place them in the images directory within our template. This means that we will also need to adjust the path to those images as well. We can do this by taking advantage of the smarty path/template variable. To do this we need to go through our new layout.tpl.php file and prepend the following to each path where an image is loaded.

```
templates/{$smarty.const.MAIN_TEMPLATE}/
```

This means that the path for our background images will change from this...

```
background="images/layout_02.gif”
```

to this....

```
background="templates/{$smarty.const.MAIN_TEMPLATE}/images/layout_02.gif"
```

And the same holds true for the path to our bullet images, they will change from this....

```
src="images/bullet.gif"
```

to this .....

```
src="templates/{$smarty.const.MAIN_TEMPLATE}/images/bullet.gif"
```

Now, so we actually have a bullet image instead of a broken link why dont you go ahead and copy the bullet.gif file from /images to your template directory image folder / templates/orange/images. You can open this image up in photoshop and redesign it as you see fit.

At this point we are ready to go ahead and check out our new layout. Log into the admin side of your site, goto “Front End Settings” section and select “Main Settings”. In the page that appears you can select your “Site Template”, click the dropdown and select “Orange” and then hit the submit button at the bottom of the form. When done its a good idea to clear the contents of your template_c directory. Then go ahead and point a browser to your classifieds front end.

Don't be alarmed if you have some blue bars overlapping some of your nice interface, these are handled by the style sheet and now that we have our interface viewable its time to start making changes to the style.css file.

Once again, I can not stress enough how important using the firefox web browser with the web developer extension is, it can truly save you a ton of time and frustration. Firefox also allows you to select items on a web page and selectively view just the source for the highlighted section of the page.

Ok, lets start now by looking at our style.css sheet, go ahead and open the style.css file for the orange template in your editor of choice. This is also a good time to “View Source” from your browser. Lets start by looking at the HTML source to identify the class thats being used for the navigation box. It should be class="navTitle" and if we look through the style.css file we see that the default “navTitle” looks like this.

```
.navTitle {
	font-weight: bold;
	color: #FFFFFF;
	background: #5487BF url("images/fade.gif");
	border-bottom: 1px solid #5A89C0;
	text-indent: 5px;
	padding: 5px;
	margin: 5px;
}
```

Lets start by removing the background and the bottom border so it should end up like this.

```
.navTitle {
	font-weight: bold;
	color: #FFFFFF;
	text-indent: 5px;
	padding: 5px;
	margin: 5px;
}
```

This will get rid of the blue over the header and things start looking a bit better. Now lets look for the css that the table in the main body uses by looking at the browsers “View Source” again. We can see that the header for the category tables uses class="tableborder" so finding that in our style sheet we see this.

```
.tableborder {
	border: 1px solid #000;
	width: 100%;
	padding: 0px;
	margin:0px;
}
```

There is nothing there to indicate a blue color or background image so lets dig a little deeper. The HTML that the blue background is in is a table header or <TH> tag so lets read the css a little more closely. There is a section, with comments that states the following is for the main table and under it we see there is a definition for “th” which is a table header. Taking a closer look at it we see this.

```
th {
	font-weight: bold;
	color: #FFFFFF;
	background: #5487BF;
	border-bottom: 1px solid #5A89C0;
	text-indent: 5px;
	padding: 5px;
	margin: 5px;
	font-size: 10px;
}
```

If you still have photoshop open you can open the color picker and enter the RGB number of 5487BF to see what color that is and sure enough, its blue. We have a couple choices, we can get rid of the colored background altogether or what I would do is change the color to one that fits our color scheme. In the “th” section for the css im simply going to change the background from its current blue color of 5487BF to our orange color FFD980. Im also going to change the color of border-bottom to be our darkest brown/orange color BFA260.

Next it looks like we have some alignment issues and it appears everything is centered so lets see if we can get the contents of our nav table left justified.

If you will remember, when setting up our new layout file and creating the rows and cells our table data cells had a class of navlist, so lets find that in our style sheet next. You should have this for navlist.

```
.navlist{
	font-weight: bold;
	background-color:#F1F1F1;
	text-indent: 5px;
	padding: 3px;
}
```

There isn't anything there that would center the nav text so it must be coming from another element that the table data is in. You can work your way up through the various HTML tags that the table data tags are in until you finally get up to the body where you will find this.

```
body {
	font-family: Verdana, Arial, Helvetica, sans-serif;
	font-size: 10px;
	color: #000000;
	background-color: #F4F4F4;
	text-align: center;
}
```

go ahead and remove "text-align: center;" and refresh your page, all your text should now left justify, some of it however has actually gone too far to the left and is outside our boxes so next lets look at fixing that. Lets start with navTitle so find that in your style sheet. In navTitle you can see we have...

```
.navTitle {
	font-weight: bold;
	color: #FFFFFF;
	text-indent: 5px;
	padding: 5px;
	margin: 5px;
}
```

Lets increase the indent for it from 5px to 35px and see how that looks. If you have firefox with the web developer plug in you make make changes and see results real time. If not you will have to edit the css, save it and reload the web page. 35Px looks pretty good though so lets move on to navlist to see what we can do with the nav links themselves. In navlist we have this...

```
.navlist{
	font-weight: bold;
	background-color:#F1F1F1;
	text-indent: 5px;
	padding: 3px;
}
```

Once again lets go ahead and bump the text-indent from 5 to 35px. That seemed to nicely straighten out our Navigation nicely. Depending on just how wide you mad your navigation background image you may need to play with the text-indent number to get what works for you.

Now lets take a closer look at the body of our site. In the body of our site we have 3 different things to consider. The top text is pulled from the database via the “home page” page content section of the admin side of the site. Next we have our categories which are contained in “tableborder” and then lastly we have the featured listings which are created via a smarty function and displayed in the body of the site via the home.tpl.php template.

There are likely many ways to deal with the text at the top of the body but since it is already in paragraph tags I am simply going to add a new element to our stylesheet. 35Px indent worked well for our nav table so I am going to use the same for our “p” selector. Just below the part of our style sheet where #header is I am going to squeeze in a new selector that looks like this.

```
P{
	text-indent: 35px;
}
```

As you can tell it pushed our body text over nicely but feel free to experiment with other routes or indents as you see fit. Next lets address the category box. As we found earlier it has a class of tableborder and its section looks like this.

```
.tableborder {
	border: 1px solid #000;
	width: 100%;
	padding: 0px;
	margin:0px;
}
```

Its width is specified at 100% so lets start by shrinking its size a tad and drop it to 90%. That shrunk it up but its still a bit left too far so lets add a left margin for it as well. Yours will likely vary slightly depending on just how wide your main body graphic ended up being so you may need to adjust the width percentage or the left margin to get it nice and centered but for me the following yielded nice results.

```
.tableborder {
	border: 1px solid #000;
	width: 90%;
	padding: 0px;
	margin:0px;
	margin-left:33px;
}
```

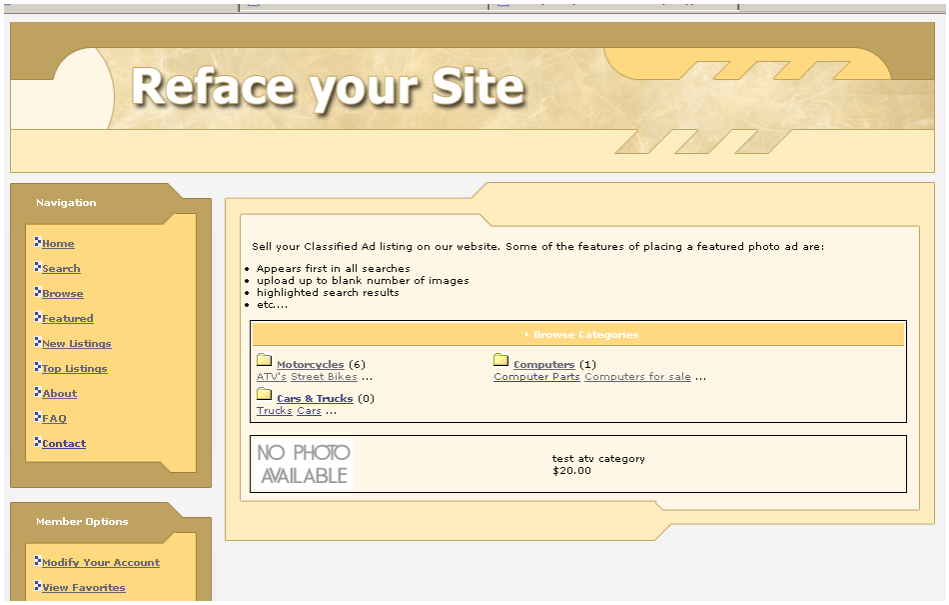
Lastly lets look at the featured listing section of the body by finding its HTML source. We can see that it has a “style” that is defined right in the HTML. I know from adjusting the section above that a 90% width worked well and that a left margin of 33px centered it nicely. Since the content and the table are created by a smarty function and the table properties are passed to this function as a parameter I need to open the template home.tpl.php and look to the bottom we can see where the function is called and it looks like this.

```
{feature_listings_horizontal number=8 cols=3 table_attr='width="100%" style="border: 1px solid #000;"' td_attr='valign="middle" align="left"'}
```

The tables attributes are passed along to the function via the “table_attr” parameter above and by looking at it we can see its currently specifying a width of 100% so im going to drop that to 90%. Additionally the style is also defined and passed along as a parameter to the function so I am going to add a left margin setting as well. What I ended up with looks like this.

```
{feature_listings_horizontal number=8 cols=3 table_attr='width="90%" style="margin-left:33px; border: 1px solid #000;"' td_attr='valign="middle" align="left"'}
```

Go ahead and save home.tpl.php again and refresh your browser to see how it looks. Here is how my site currently looks.



Next, go ahead and chose one of your categories and drill down so you see that table with your items listed. If your site is anything like mine this table also is too wide for our current body image. As we figured out above cutting the table width to 90% seems to work pretty well. Looking at the HTML source we see we have class="sortable" so go ahead and find that section in your style sheet. It should look like this, ive already changed my width to 90%..

```
table.sortable
{
	border: 1px solid #000;
	width: 90%;
	padding: 0px;
	margin:0px;
}
```

That shrunk the table nicely but its still to the left too far so lets change the margin setting, which applies to all sides, to a more specific left margin like so. For me 33px worked perfect, as with previous tables you may have to tweak yours to get it centered right.

```
table.sortable
{
	border: 1px solid #000;
	width: 90%;
	padding: 0px;
	margin-left:33px;
}
```

The next thing that really sticks out is the little pagination box. Its still blue and right justified so it hangs outside the right side of my body image. We can look at our page source to see it has class="pagination". Looking through our style sheet there is a whole section for the pagination but for starters we have this.

```
.pagination{
	text-align:right;
	background-color:#7088b0;
	color:#333333;
}
```

For our other tables we added a left margin to nudge them but in the case of the little pagination table we need to add a right margin to nudge it to the left, for me 40px worked great. Also you may want to change the background-color:#7088b0; to something more fitting. Im going to use our darkest color BFA260. In order for the color change to fully come out we will also need to remove the background color setting from .navigationBack. For me, I ended up with .pagination and . NavigationBack that look like the following.

```
.pagination{
	text-align: right;
	background-color: #BFA260;
	color: #333333;
	margin-right:40px;
}
.navigationBack{
	color: #FFFFFF;
	font: bold 11px tahoma, verdana, geneva, lucida, 'lucida grande', arial, helvetica,
sans-serif;
	padding: 3px 6px 3px 6px;
	white-space: nowrap;
}
```

At this point your new layout should be really taking shape. There is a lot more you can do, text colors to change etc but I will leave those little details up to you. I should have given you enough info to get you a good start on designing your own unique look.

If you made it this far give yourself a big pat on the back, this turned out to be a rather wordy, yet hopefully valuable, tutorial.

Larry.