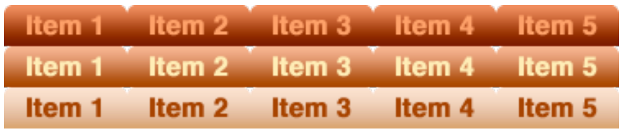
WEB DESIGN

GRAPHICAL TABS USING CSS AND SPRITES

Tabbed navigation is very popular—it's clean and easy to use. Tabs can be created a number of ways, from simple HTML + CSS methods to more complex graphics-controlled-by-javacript techniques. In this tutorial I present another method that falls somewhere in between: it uses graphics with rollover effects, but it controls them entirely with CSS. When they're finished they will look something like this:

Part One: Create the Tabs

Tabbed navigation requires at least two, but usually three "states" per tab: a "normal" or "up" state (how the tab usually appears), an "over" state (how the tab appears when it's rolled-over), and a "current" state (how the tab appears when you are on that particular page). Some methods require you to create a separate graphic for each state of each button, but with this method a single graphic is created for every state of every tab. Such graphics that contain multiple images that are used separately are often referred to as **sprites** (a name that comes from the video game development world). Using one large graphic instead of numerous small ones can save download time. For every image on a page, the browser has to send an http request to the server to get that image. If instead you use only one image, the browser has to make only one http request, speeding the download.



To begin, I created a sprite containing all of the tabs states. The top row is "normal," the second row is "over" and the bottom "current." I also made sure that that each row was exactly 1/3 the height of the image. In this case, each row is 25 pixels tall and each column 75 pixels wide. Thus the entire graphic is 375 pixels by 75 pixels. You can create your own or use this one—simply right-click on it and choose "save image," or whatever option your browser gives you.

I chose to save the graphic as a PNG file. Why use PNGs? They are better than JPEG files for flat graphics and text and also allow for transparency (so I could leave the rounded corners transparent). And they're better than GIF files in terms of quality.

Saved the file as "**tabs.png**" in a folder called "**images**."

Part Two: Create the HTML/CSS Page

1. Launch Dreamweaver (or some other text editor). Open a new file and save it.

2. Write the following markup in the body:
```
<div id="mainNav">
    <ul>
        <li id="item1Tab"><a href="#"><span></span>Item 1</a></li>
        <li id="item2Tab"><a href="#" id="current"><span></span>Item 2</a></li>
        <li id="item3Tab"><a href="#"><span></span>Item 3</a></li>
        <li id="item4Tab"><a href="#"><span></span>Item 4</a></li>
        <li id="item5Tab"><a href="#"><span></span>Item 5</a></li>
    </ul>
</div>
```
   Note the addition of id="current" on the second <a> tag; you can move it to any of the other <a> tags if you prefer (but only use one per menu). This id is used to indicate the page the user is currently on; so if your 're creating the page for "Item 4," you would move the id="current" to the <a> tag for tab 4.

3. Create a cascading style sheet section in your head. Remember, **all** of your styles must be placed between these tags:
```
<style type="text/css">
```

```
<!--

-->
</style>
```

4. Within the <style> tags write the following style rules:
```
* {
    margin:0;
    padding:0;
}
#mainNav ul li{
    float:left;
    position:relative;
    list-style-type:none;
    width:75px;
    height:25px;
}
```
The first style rule uses the universal selector to removes all default margins and padding; this is not always appropriate for every Web page, but I find it very useful. The next style rule turns the vertical list column into a horizontal row, removes the bullets, and gives each list item a **relative position**. The positioning has no effect of their appearance, but we need it to absolutely position the graphic images on top, which we will do next...

5. Now we need to write the style rules for the individual tabs. Each tab will have three style rules, one for the "normal" state of the tab, one for the "over" or "hover" state, and one for the "current" state. These of course correspond to the three vertical sections of the graphic. We'll apply these rules to the <span> tags inside of the links:
```
#mainNav #item1Tab a span {
    position:absolute;
    width:75px;
    height:25px;
    top:0;
    let:0;
    background-image: url(images/tabs.png);
    background-repeat: no-repeat;
    background-position: 0px 0px;
}
#mainNav #item1Tab a:hover span {
    background-position: 0px -25px;
}
#mainNav #item1Tab a#current span {
    background-position: 0px -50px;
}
```
The first rule positions the <span> absolutely, pulling it out of the markup and dropping it **on top** of the link. It then applies the tab image as a background —but since the <span> item is on top of the text link, the background covers it up. Note also that the span is given a width and height equivalent to the size of a single "tab" within the larger sprite graphic. If an element is smaller than its background image, only a portion of that image will be shown—we need to determine which portion of that image is visible. The **background-position** property does just that; in this case it positions the background at 0, 0 (the top-left corner of the graphic).

The next two rules reposition the tab graphic slightly depending on its state—note the changes to the background-position line: the "hover" state shifts the background up 25 pixels, which reveals the middle third of the graphic. Thus when one rolls-over the tab it changes, essentially 'sliding' up and down into the proper position. The final style, the "current" state, shows the bottom third of the graphic by shifting the sprite up 50 pixels.

6. That takes care of the first tab; what about the others? Well they use nearly identical code, you just have to change a couple of small details. Here is the code for the second tab; note the differences:
```
#mainNav #item2Tab a span {
    position:absolute;
    width:75px;
    height:25px;
    top:0;
    let:0;
    background-image: url(images/tabs.png);
    background-repeat: no-repeat;
    background-position: -75px 0px;
}
#mainNav #item1Tab a:hover span {
    background-position: -75px -25px;
}
#mainNav #item1Tab a#current span {
    background-position: -75px -50px;
}
```
   Catch the changes? First make sure you change the id reference to `#item2Tab`. Then change the **y** position for the background-images to -75px; this will shift the sprite 75 pixels to the left, revealing the second column. Of course, if you're using your own graphics with different widths, you'll have to adjust the position accordingly.

7. Write style rules for the rest of the tabs, changing both the **id** names and **y** positions. Since all of these tabs are the same width, you have to subtract 75 from the **y** each time (-75, -150, -225, -300). If you use tabs of varying widths, you'd have to adjust the y amount accordingly.

8. Try it. Do they work?