

# Advanced CSS Training

## CSS Background Tricks

## Lesson 1, Activity 2: Rounded Corner Boxes

There are many different methods of creating rounded-corner boxes using CSS. Most methods use background images and involve extra non-semantic HTML markup. Unfortunately, until [CSS3](#) is released and supported, designers who want rounded corners may be forced to compromise on semantic HTML markup.

For more information about rounded corners in CSS3, see [The border-endings properties of CSS Backgrounds and Borders Module Level 3](#).

## Fixed-width Rounded Corner Boxes

To create a fixed-width rounded-corner box with variable height, you need to use at least two background images, one for the top and one for the bottom, and sometimes a third for the middle. Each image must be the same width as the box, which is usually a `div` marked up like this:

## Syntax

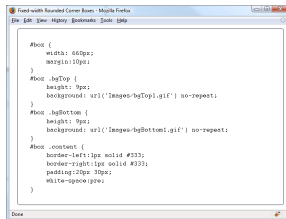
```
<div id="box">
<div class="bgTop"><!--for IE6--></div>
<div class="content">
content goes here
</div>
<div class="bgBottom"></div>
</div>
```

In IE6, the `height` property is ignored on an element has no content in it. By adding the `<!--for IE6-->` comment, we make sure IE6 honors `height` property.

Our first example, [CssBackgroundTricks/Demos/FixedRoundCornerBox.html](#), uses images to cap the top and bottom of a container. It uses the `CSS border` property for the left and right borders.

The background images, which are 9 pixels high by 660 pixels wide, are shown below:

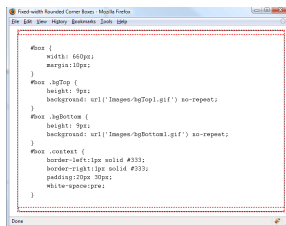
And here is the result in the browser with the CSS shown in the box itself:



To make it easier to see what's going, in the screenshot below we've added a dashed border around the `div`s using this code:

```
div { border:1px dashed #f00; }
```

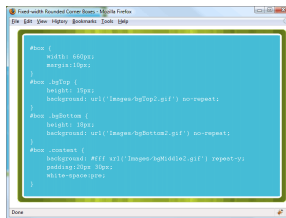
## The result:



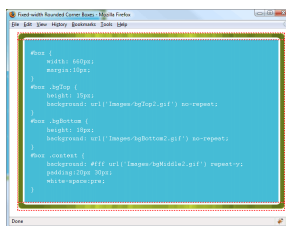
In the next example, the HTML markup is the same, but his one uses a background image instead of the `border` property to style the left and right edges of the container. We cannot use the `border` property because the border here is fancier than any options provided by CSS.

The images used are shown below:

And here is the result in the browser with the CSS shown in the box itself:



And here it is with the dashed border to show how the sections are broken out:



## Flexible-width Rounded Corner Boxes

Creating rounded corner boxes with *flexible* widths is more complicated because images have *fixed* widths. Instead of using images to cap the top and bottom, we use an image for each corner. The markup looks like this:

## Syntax

```
<div id="box">
<span class="topLeft bg"><!--for IE6--></span>
<span class="topRight bg"><!--for IE6--></span>
<span class="bottomLeft bg"><!--for IE6--></span>
<span class="bottomRight bg"><!--for IE6--></span>
<div class="content">content goes here</div>
</div>
```

Notice that we have assigned two classes to each of the `span` tags: one to capture its position (e.g. `topLeft`) and the other to indicate that it is a background (`bg`).

We could use four separate images; however, by taking advantage of background positioning, we can accomplish this with just a single image (magnified below):



The checkered portion of this image is transparent. Each corner of the image shown will serve as the background for the analogous corner of our box. Here's the code:

If the white space between the corners of the blue circle image could be removed, you would get a circle. You could do that and this sample would still work. We've separated the corners for illustrative purposes.

## Code Sample:

[CssBackgroundTricks/Demos/FlexibleRoundCornerBox.html](#)

```
--- CODE --- O M I T T E D ---
<style type="text/css">
body{
margin:10px;
font-family:Courier;
font-size:1.1em;
color:#000;
}
```

```
#box {
margin:10px;
position:relative;
height:180px;
#box .bg {
position:absolute;
height:120px;
width:120px;
background:#fff url(images/blueCircle.gif) no-repeat;
#box .topLeft {
top:10px;
left:10px;
background-position:top left;
#box .topRight {
top:10px;
right:10px;
background-position:top right;
#box .bottomLeft {
bottom:10px;
left:10px;
background-position:bottom left;
#box .bottomRight {
bottom:10px;
right:10px;
background-position:bottom right;
#box .content {
border:4px solid #00f;
padding:10px 30px;
</style>
<!--Flexible-width Rounded Corner Boxes-->
</head>
<body>


Things to notice:



- The box itself gets relative positioning. That's so we can position the spans absolutely within it.
- Each span, via its bg class, has the following properties:



```
position:absolute;
height:120px;
width:120px;
background:#fff url(images/blueCircle.gif) no-repeat;
```



This will turn them into positionable 12 pixel squares with a background. If we were to change the height and width to 50 pixels, the box would look like this:



- Each span is then positioned within the box and the background image is positioned so that the appropriate corner is revealed.
- We put a 4-pixel blue border on the "content" div to meet up with the four corners.
- We add a height: 1% declaration to our rule for #box. This is necessary for early versions of Internet Explorer to render the box correctly. For a good explanation on this, see http://www.cotransistz.de/css/ondynamiclayout.html



Here the result shown with two different widths:



Unfortunately, as the screenshot below shows, Internet Explorer 6 has a problem with this method for creating rounded corners:



Notice the right border where the two arrows are pointing. These extra lines are a result of the right border assigned to the "content" div. They don't always show up. It depends on the actual width of the box, which of course is dynamic.



It's possible to create rounded-corner boxes with flexible widths that do work in Internet Explorer 6 by having the borders fade into the background like this:



```
content goes here content goes
here content goes here content
goes here content goes here
content goes here content goes
here content goes here content
goes here content goes here
```



The code for this, which is in CsaBackground\Tricks\Demos\FlexibleRoundCornerBox2.html, is almost identical. The only differences are



1. The magnified background image looks like this (notice the corners are filled in). The center is transparent:
2. The body and the "content" div have background colors (blue and white, respectively).
3. In the bg class, the background color of all the spans has been set to "transparent". This makes it possible to change the background color of the "content" div without having to adjust anything else.



#### Mountaintop Corners



In an A List Apart article, Dan Cederholm introduced another way to create rounded corners, which he dubbed "Mountaintop Corners."



The idea is similar to the last example, but doesn't involve using borders on the box. Instead, you cover the corners of the box with a concave image of the same color as the background leaving the concave portion transparent. You could use four images like this:



But we can put these all together and use just a single image like this:



As you can see, it looks like a hole so we named it hole.gif in the Demos\Images folder. That image is 18x18 pixels, so each corner is 9x9 pixels as you can see in the diagram below:



One nice feature about this method is that you can freely style the boxes without considering the color of the border or the rounded corners. The screenshot below shows the output for two boxes with different background colors:



And here is the code:



Code Sample:



CsaBackground\Tricks\Demos\mountaintop.html



For use only by Tim Dempsey (landsurvival@hotmail.com). Not to be distributed.



2 of 8


```

```

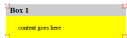
---- U O U E U N I T E D ----
<style type="text/css">
body {
margin:10px;
font:1.2em/1.1;
color:#000;
background:#fff;
}
h1 {
font-size:1.2em;
padding:5px 10px;
}
.box {
margin:10px;
position:relative;
height:150px;
}
.box .bg {
position:absolute;
width:100%;
height:100%;
background: transparent url('images/hole.gif') no-repeat;
border:1px solid red;
}
.box .topLeft {
top:-1px;
left:-1px;
background-position:top left;
}
.box .topRight {
top:-1px;
right:-1px;
background-position:top right;
}
.box .bottomLeft {
bottom:-1px;
left:-1px;
background-position:bottom left;
}
.box .bottomRight {
bottom:-1px;
right:-1px;
background-position:bottom right;
}
#box1 .content {
padding:2px 30px;
background:#fff;
}
#box1 h1 {
background:#ccc;
}
#box2 .content {
padding:20px 30px;
background:#0ff;
}
#box2 h1 {
background:#faa;
}
</style>
<title>Mountaintop Boxes</title>
</head>
<body>
<div class="box" id="box1">
<span class="topLeft bg"><!--for IE6--></span>
<span class="topRight bg"><!--for IE6--></span>
<span class="bottomLeft bg"><!--for IE6--></span>
<span class="bottomRight bg"><!--for IE6--></span>
<h1>Box 1</h1>
<div class="content">
content goes here
</div>
</div>
<div class="box" id="box2">
<span class="topLeft bg"><!--for IE6--></span>
<span class="topRight bg"><!--for IE6--></span>
<span class="bottomLeft bg"><!--for IE6--></span>
<span class="bottomRight bg"><!--for IE6--></span>
<h1>Box 2</h1>
<div class="content">
content goes here
</div>
</div>
</body>
</html>

```

Things to notice:

- The HTML code is very similar to what we saw in the previous example. We've used ids to identify the two different boxes and we have added h1 elements above the "content" div.
- We use -1px for the positioning to push the span tags 1 pixel outside of the containing box. This is to fix the bug we saw earlier in Internet Explorer 6. If you don't need to support IE6, you can set those values to 0px.
- We have added background colors to the h1 and "content" div.

Below is a shot of the top box with the corners outlined, which we did by adding this CSS code: span { border:1px dashed red; }.



There are other ways of creating boxes with rounded corners. For a slew of different methods, see <http://www.exquisite.com/75-rounded-corners-techniques-with-css>.

Lesson 1, Activity 4: Creating a Rooftop Panel

Duration: 20 to 30 minutes.

In this exercise you will use some of the techniques you have learned to create rounded corners to create a rooftop panel that looks like this:



1. Open [C:\BackgroundTricks\Exercises\rooftop.html](#) in your editor.
2. Fill in the three empty rules to create a rooftop panel as shown in the screenshot above.
3. You will need to make use of [rooftop.gif](#) in the [images](#) folder. The image is 350 x 20 pixels.

Solution:

[C:\BackgroundTricks\Solutions\rooftop.html](#)

```
----- C O D E . . O M I T T E D -----
<style type="text/css">
body {
margin:10px;
font: 1.2em/1.1;
background:#fff;
}
p {
margin:15px 20px;
}
.panel {
position:relative;
width:350px;
background-color:#cef;
padding-bottom:10px;
}
.panel .bg {
position:absolute;
height:20px;
width:100%;
top:0px;
left:0px;
background: transparent url('images/rooftop.gif') no-repeat;
}
.panel h2 {
padding-top:15px;
text-align:center;
font-size:2em;
font-variant: small-caps;
background-color:#add;
color:#006;
}
</style>
----- C O D E . . O M I T T E D -----
```

## Lesson 1, Activity 5: Tabbed Navigation

The screenshot below shows an example of a page that uses tabbed navigation:



To create the navigation bar, we combine some of the techniques we have learned thus far. Let's start by looking at the code:

**Code Sample:**

C:\BackgroundTricks\Demo\tabbedMenu.html

```

---- C O D E   O M I T T E D ----
<style type="text/css">
#wrapper {
width:846px;
margin:10px auto;
}
#content {
clear:both;
width:838px;
background-color:#eee;
border:4px solid #000808;
}
p {
margin:5px;
padding:5px;
}
#mainMenu {
width:846px;
font-family: "Trebuchet MS";
}
#mainMenu li {
position:relative;
display:block;
width:100px;
float:left;
margin-left:1px;
}
#mainMenu li:first-child, #mainMenu li.first {
margin-left:0px;
}
#mainMenu a {
display:block;
padding:10px;
text-decoration:none;
background:transparent url(Images/link.gif) repeat-x;
color:#fff;
text-align:center;
}
#mainMenu a:hover {
background:transparent url(Images/hover.gif) repeat-x;
color:#333;
}
#mainMenu a:active {
background:transparent url(Images/active.gif) repeat-x;
color:#fff;
}
</style>
<title>CSS Menu</title>
</head>
<body>
<div id="wrapper">
<ul id="mainMenu">
<li class="first"><a href="home.html">Home</a></li>
<li><a href="services.html">Services</a></li>
<li><a href="products.html">Products</a></li>
<li><a href="support.html">Support</a></li>
<li><a href="blog.html">Blog</a></li>
<li><a href="about.html">About</a></li>
<li><a href="contact.html">Contact</a></li>
</ul>
<div id="content">
---- C O D E   O M I T T E D ----
</div>
</body>
</html>

```

This code should be pretty straightforward. It simply combines the list navigation techniques you learned in CSS Lists as Hierarchical Navigation with the background image techniques we've learned here.

Open the page in your browser to see that the three states of the menu items. The magnified background images used are shown below:



Each image is 6 pixels wide by 37 pixels high.

## Lesson 1, Activity 7: Tabbed Navigation: Rounding the Corners

Duration: 15 to 25 minutes.

In this exercise, you will modify the tabbed menu we just showed so that the top corners of each menu item are rounded like this:



1. Open [C:\BackgroundTricks\Solutions\tabbedMenu.html](#).
2. Modify the code so that it uses [hole.gif](#) from the [images](#) folder to create the rounded corners shown in the screenshot above.

**Solution:**[C:\BackgroundTricks\Solutions\tabbedMenu.html](#)

```
<!DOCTYPE HTML>
<html>
<head>
  <meta charset="UTF-8">
  <link type="text/css" rel="stylesheet" href="../reset-meyer.css">
  <style type="text/css">
    #wrapper {
      width:846px;
      margin:10px auto;
    }
    #content {
      clear:both;
      width:838px;
      background-color:#eee;
      border:4px solid #000000;
    }
    p {
      margin:5px;
      padding:1px;
    }
    #mainMenu {
      width:846px;
      font-family: "Trebuchet MS";
    }
    #mainMenu li {
      position:relative;
      display:block;
      width:120px;
      float:left;
      margin-left:1px;
    }
    #mainMenu li:first-child,#mainMenu li:first {
      margin-left:0px;
    }
    #mainMenu a {
      display:block;
      padding:10px;
      text-decoration:none;
      background:transparent url(Images/link.gif) repeat-x;
      color:#fff;
      text-align:center;
    }
    #mainMenu a:hover {
      background:transparent url(Images/hover.gif) repeat-x;
      color:#333;
    }
    #mainMenu a:active {
      background:transparent url(Images/active.gif) repeat-x;
      color:#fff;
    }
    #mainMenu .bgLeft {
      position:absolute;
      top:-1px;
      left:-1px;
      height:9px;
      width:9px;
      background: transparent url('Images/hole.gif') no-repeat top left;
    }
    #mainMenu .bgRight {
      position:absolute;
      top:-1px;
      right:-1px;
      height:9px;
      width:9px;
      background: transparent url('Images/hole.gif') no-repeat top right;
    }
  </style>
</head>
<body>
  <div id="wrapper">
    <div id="mainMenu">
      <div id="first"><span class="bgLeft"><!--for IE6--></span><span class="bgRight"><!--for IE6--></span><a href="home.html">Home</a></div>
      <div id="second"><span class="bgLeft"><!--for IE6--></span><span class="bgRight"><!--for IE6--></span><a href="services.html">Services</a></div>
      <div id="third"><span class="bgLeft"><!--for IE6--></span><span class="bgRight"><!--for IE6--></span><a href="products.html">Products</a></div>
      <div id="four"><span class="bgLeft"><!--for IE6--></span><span class="bgRight"><!--for IE6--></span><a href="support.html">Support</a></div>
      <div id="fifth"><span class="bgLeft"><!--for IE6--></span><span class="bgRight"><!--for IE6--></span><a href="contact.html">Contact</a></div>
    </div>
    <div id="content">
      <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.</p>
    </div>
  </div>
</body>
</html>
```

## Lesson 1, Activity 8: Drop Shadows

A drop shadow effect can be created by nesting one `div` inside of another `div`. Using relative positioning, the nested `div`, which will contain the content, can be pushed from the bottom right, thus allowing the outer `div` to show through. The outer `div` is then given a background color or image to create the drop shadow effect.

Take a look at the code below:

**Code Sample:**

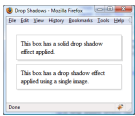
[CssBackgroundTricks/Demos/DropShadows.html](#)

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="utf-8">
<title>>Drop Shadows</title>
<style type="text/css">
body {
padding: 10px;
}
p {
margin: 0;
padding: 10px;
}
.box {
width: 95%;
margin: 10px;
}
.box .content {
position: relative;
background-color: #fff;
border: 1px solid #ccc;
bottom: 3px;
right: 3px;
}
.shadow1 {
background: #ddd;
}
.shadow2 {
background: url('Images/bg-dropShadow.gif') bottom right;
}
</style>
</head>
<body>
<div class="box shadow1">
<div class="content">
<p>This box has a solid drop shadow effect applied.</p>
</div>
</div>
<div class="box shadow2">
<div class="content">
<p>This box has a drop shadow effect applied using a single image.</p>
</div>
</div>
</body>
</html>
```

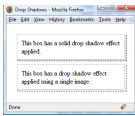
Things to notice:

- The HTML markup is quite simple: one `div` inside of another.
- The nested "content" `div` has relative positioning and is pushed up and to the left. It also takes a background color (white) and a 1 pixel grey border.
- The shadow of the first box is created with a background color.
- The shadow of the second box is created with a background image.

The result is shown below:



In the screenshot below, we put a dashed border around the shadow `div`s to show that they are largely covered by the "content" `div`s:



That's it. As you can see, creating drop shadow effects is relatively easy.



Lesson 1, Activity 9: CSS Sprites

A CSS sprite is an image file that contains several graphics used on a web page. By showing different parts of the sprite in different locations, it appears that there are several different images, but they are all contained in a single file, which translates to a single download.

As a simple example, let's look at the images we used for tabbed navigation earlier. Here they are again (magnified):



Instead of having three different images, let's combine these into a single image (magnified below):



Now let's see how we can play with the background positioning to make use of this single [tabSprite.gif](#) image rather than the three separate images.

Code Sample:

[CssBackgroundTricks/Demos/tabbedMenu-sprite.html](#)

```
--- C O D E   O M I T T E D ---
#mainMenu a {
display:block;
padding:10px;
text-decoration:none;
background:transparent url(Images/tabSprite.gif) 0px 0px repeat-x;
color:#fff;
text-align:center;
}

#mainMenu a:hover {
color:#333;
background-position:0px -37px;
}

#mainMenu a:active {
color:#fff;
background-position:0px -74px;
}
--- C O D E   O M I T T E D ---
```

Things to notice:

- In the rule for #mainMenu a, we have changed the background image to [Images/tabSprite.gif](#) and added positioning of 0px 0px. That's because we want the link state to show the top portion of our [tabSprite](#) image.
- In the rule for #mainMenu a:hover, we have replaced the rule for background with background-position:0px -37px; That's because we inherit the background image from the #mainMenu a, but we want to bump it up 37 pixels so that it shows the middle portion of the image.
- We do the same thing with the rule for #mainMenu a:active, but this time we bump it up 74 pixels.

The more graphic heavy a page is, the more it can benefit from using CSS sprites. Consider this small example we have just shown. The table below shows the file sizes for the the three original graphics and then for the sprite:

Comparing CSS Sprite File Size  
with Separate Images

File	Size
link.gif	590 bytes
hover.gif	612 bytes
active.gif	378 bytes
Total:	1,580 bytes
tabSprite.gif	817 bytes
Difference:	763 bytes

That savings of 763 bytes translates to a savings in file size of 48%. Add to that the overhead associated with each http download for the separate images and you'll find the savings can be significant.

Open [CssBackgroundTricks/Demos/sprite.html](#) for a more complex example that uses a single image to create a library of thumbnails that highlight when the user hovers over them. Check out the code and you'll see that only a single graphic is used.