CSS Flyouts - Part One

By: John Gallant , Holly Bergevin ,

This tutorial is meant for those familiar with basic CSS syntax and usage. If this is not you, the following discussion will make more sense after you have studied these articles:

- Under the Hood — The Basics of HTML: Part Three (http://www.communitymx.com/abstract.cfm?cid=AB474C9E02102677)
- The Link and Dynamic Pseudo-classes (http://www.communitymx.com/abstract.cfm?cid=B0934)
- Flowing and Positioning: Two Page Models (http://www.communitymx.com/abstract.cfm?cid=EA1A82A6F65A33F5)
- Absolute Positioning and the Top Property (http://www.communitymx.com/abstract.cfm?cid=AAA7C45E7CD65D33)

To be prepared for this tutorial, you should already have read the earlier **CSS Tooltips** series:

- Part One (http://www.communitymx.com/content/article.cfm?cid=4E2C0)
- Part Two (http://www.communitymx.com/content/article.cfm?cid=52428)
- Part Three (http://www.communitymx.com/abstract.cfm?cid=48095)
- Part Four (http://www.communitymx.com/abstract.cfm?cid=202A2)
- Part Five (http://www.communitymx.com/abstract.cfm?cid=A863B)
- Part Six (http://www.communitymx.com/abstract.cfm?cid=AA826)

Flyouts and Their Uses

The generic terms "navigation flyout" and "flyout menu" refer to any page menu that visually generates submenus when links are moused over, or "hovered." Therefore, a flyout is a screen effect triggered by use of the mouse, and sometimes by tabbing through the links.

A common reason for wanting to use a flyout menu rather than a typical "static" navigation system is to compress a large set of navigation menus into a small space on the window, and then display the submenus only when the user is hovering or tabbing the navigation itself.

For the majority of users, flyout menus work well, but some users can have difficulties with such flyouts for one reason or another. Accessibility and usability issues can often be made less of a problem by employing various strategies, some of which will be discussed in the course of this series. However, be aware that all styled menus, including flyouts, can be inherently less usable and accessible than a plain unstyled link list, though some navigation systems may approach this goal very closely.

There are many users who do use a mouse, but who dislike flyout navigation menus, usually because negotiating a badly designed flyout menu can be an exercise in frustration. We've all seen these menus,

haven't we? One skinny flyout leads to another, and another, finally ending in the one slender link you want. After threading the maze successfully, sweat beading on your forehead, you try to click on the link, only to move the pointer a *hair's width* off the link. Whoosh! The entire flyout system is gone! You lose, try again.

Flyout menus that use scripting to generate the submenus often have some sort of timing preset that hopefully makes the flyout submenu persist long enough to get your pointer back where it belongs if you overshoot a link. But CSS flyouts lack such a feature, unless it is added with scripting. The next part in this series will cover "sticky hovering" the CSS way, with no scripting required. Stay tuned, folks, this will be a CMX scoop!

All that being said, many clients *demand* flyout (or dropdown) menus, both to save space and to add what they feel is a little pizzazz to their web sites. As a busy web developer, you are sure to be called on to do such flyout menus at some point in your career, so it's smart to know how to do them. At this time, the scripted flyout or dropdown menu is somewhat superior in terms of apparent visual function, except when scripting is disabled in a browser.

When scripting is disabled, the pure CSS navigation menu we will describe continues to function, *except in Explorer for Windows*, which needs scripting to parse the Jscript we are using to "fix" IE's weak CSS support. But as Firefox and other standards-compliant browsers continue to gain in popularity, the pure CSS nav will begin to look better and better since it does not depend on scripting to work.

In any case, it's nice to have a fair alternative to the scripted flyout navigation menu, so let's get started, shall we?

As with the CSS Tooltip series (http://www.communitymx.com/content/article.cfm?cid=4E2C0) , we have a demonstration page (http://www.communitymx.com/content/source/55A69/flyout1-demo.html) for you to view, and to study as we proceed with the method description. Take a look at that demo now and become familiar with it. You might also examine its source code. Although you may not understand everything just yet, you soon will.

You may also download the demo page and associated files by clicking on the **support files icon** for this tutorial, and unzipping to a folder. Remember, though, that IE might not allow proper hovering unless the Jscript file (csshover.htc) is correctly served with the MIME type **text/x-component** . MIME types are under the control of the server administrator, not the browser or the user.

Basic Construction

We begin with a valid nested list set, *where each sublist (submenu) is entirely contained within one of the "top level" list items*. These top level items are **Home**, **About Us**, **Galleries**, and **Articles**. Note that the links within these top level list items are directly followed by the complete sublist that will become the flyout submenu for that link. Then, *and only then*, does the top level list item close, and another one begin. It is crucial that nested lists obey this format. Remember, the main list items must each enclose complete sublists, and the same construction must be followed for any additional levels of nested lists.

The HTML:

```
<div class="navholder">
 <ul>
  <li><a href="#">Home</a>
   <ul>
    <li><a href="#">Table of Contents</a></li>
    <li><a href="#">Site Map</a></li>
    <li><a href="#">Guestbook</a></li>
   </ul>
  </li> <!-- end "Home" list item -->

  <li><a href="#">About Us</a>
   <ul>
    <li><a href="#">Resume</a></li>
    <li><a href="#">Portfolio</a></li>
    <li><a href="#">Future Goals</a></li>
   </ul>
  </li> <!-- end "About Us" list item -->

  <li><a href="#">Galleries</a>
   <ul>
    <li><a href="#">People</a></li>
    <li><a href="#">Places</a></li>
    <li><a href="#">Things</a></li>
   </ul>
  </li> <!-- end "Galleries" list item -->

  <li><a href="#">Articles</a>
   <ul>
    <li><a href="#">Current</a></li>
    <li><a href="#">Favorites</a></li>
    <li><a href="#">Archive</a></li>
   </ul>
  </li> <!-- end "Articles" list item -->
 </ul>
</div>
```

Code Block 1

Image 1: *The default display of a nested list set.*

So far, we have a typical nested list set, and by itself it would look like **Image 1** to the left. Our goal is to use CSS to convert that list set into the pretty flyout navigation seen in the demo page.

The Trouble with Z-index

If you have been following along in the CSS Tooltips series you will recall how a popup element is **absolutely positioned** (AP). You will also recall how the parent element of that popup can be either **relatively positioned** or **absolutely positioned** so that the AP child can be positioned in relation to that parent when the popup appears. Using AP for the parent element works well in a design like the map demonstration (http://www.communitymx.com/content/source/A863B/castlerock.html) we used in CSS Tooltips Part Five and Part Six, where the popups and their parents are well spaced out, and we preferred that the list items take up no computed height, allowing the map image to fill the window. However, a flyout navigation menu is not widely spaced, it's all packed into a tight group. This presents new challenges.

For the menu we're constructing in this tutorial, the top level list items are **relatively positioned** elements, so they remain in the flow of the page. But their AP children are all going to occupy separate **z-index** levels. Since only one flyout appears at any one time, different flyouts can't interfere with each other. But the flyout *does need to overlap its parent*, or at least make direct contact with an edge of the parent box. Otherwise, you cannot move the mouse pointer from the parent to the flyout element without seeing the flyout vanish before you get there.

If the flyouts are to appear in contact with the side of the parent, then there is no need to worry about overlaps. But getting a perfect alignment like that can be tricky, and it gets even trickier when older IE versions with their faulty box model must be supported. We have chosen to have the displayed flyouts overlap their parent elements slightly. This works very well, looks good, and allows for slight variances with the box model. It also introduces a new issue we must address, concerning **z-index**.

The graphic to the right demonstrates this **z-index** issue, showing the problem that occurs when the flyouts are allowed to overlap the parent list. The parent for the displayed flyout is the **About Us** list item, the second list item in the top-level list. You can see how the borders of the third and fourth top level list items (Galleries and Articles) are overlapping the second LI's flyout. If the background colorings were applied to the list items rather than to the ULs, then the overlapping would be even worse.

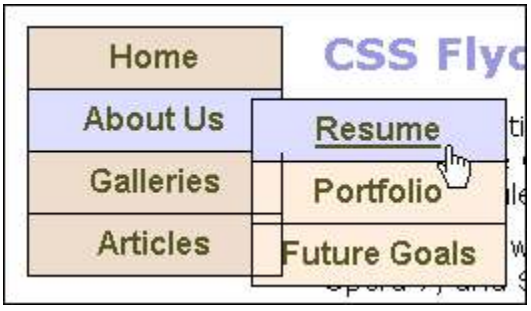

Image 2: *The default display of a flyout before appropriate z-indexes have been assigned.*

This is all very interesting, but what causes it? Well, the **About Us** flyout UL shares the same **default z-index value** as its parent list item, and that value is lower than the z-index of the next two list items in the main

list. By default, browsers will assign different z-index values to each positioned element on a page, increasing the value for elements coming later in the source. The flyout in question shares the z-index of the second top-level list item, and is therefore "lower" (farther from the viewer) than the third and fourth top-level list items which come later in the source. Naturally, those later LIs cover the flyout where the overlap occurs. In this case the LIs don't have backgrounds, so only their borders cover the flyout.

Base Over Apex

The solution to this problem is simple, just *reverse the z-index ordering on the main list items!* This is done by taking away the browser's control over z-index, and explicitly reordering the z-index values for each LI in the main list. Take a look:

The HTML:

```
<div class="navholder">
 <ul>
  <li style="z-index: 10;"><a href="#">Home</a>
   <ul>
    <li><a href="#">Table of Contents</a></li>
    <li><a href="#">Site Map</a></li>
    <li><a href="#">Guestbook</a></li>
   </ul>
  </li>
  <li style="z-index: 9;"><a href="#">About Us</a>
   <ul>
    <li><a href="#">Resume</a></li>
    <li><a href="#">Portfolio</a></li>
    <li><a href="#">Future Goals</a></li>
   </ul>
  </li>
  <li style="z-index: 8;"><a href="#">Galleries</a>
   <ul>
    <li><a href="#">People</a></li>
    <li><a href="#">Places</a></li>
    <li><a href="#">Things</a></li>
   </ul>
  </li>
  <li style="z-index: 7;"><a href="#">Articles</a>
   <ul>
    <li><a href="#">Current</a></li>
    <li><a href="#">Favorites</a></li>
    <li><a href="#">Archive</a></li>
   </ul>
  </li>
 </ul>
</div>
```

Code Block 2

Notice that you are not required to assign **{z-index: 1;}** to the last top level list item. As long as the first item in the list has the highest value and the last has the lowest, that's all that matters. You can start with **{z-index: 10;}** as we have, or with **{z-index: 10000;}** if you prefer. Also, you may choose to have the z-index values very far apart, or as a tight numeric sequence as in this demo. Just make sure the ordering is *opposite* to the default "last-in-source is highest" browser ordering.

So, we have given the top-level list items the reverse z-indexing we need to get the overlap effect we desire. The result is seen in **Image 3** to the right. Excellent!

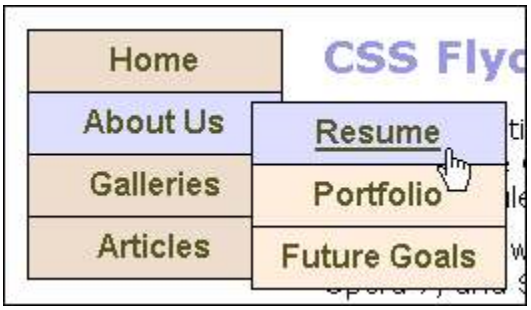

Image 3: *The display of a flyout after appropriate z-indexs have been assigned to the parent list items.*

While the solution for this problem was not particularly difficult, if this z-index issue is not fully understood you can get in a lot of trouble with more complex designs. Just remember this rule: *If your flyouts are not overlapping the way you want, the z-index values probably need reversing somewhere*. It does get a bit more involved when you add extra nested flyouts, but happily the z-index reversing is performed in exactly the same way for each sublevel, so once you understand how it works, you won't have too much difficulty applying it.

With the crucial z-index issue covered, let's dive into the actual CSS rules for the flyouts. This won't be very different than what you have learned in the previous series (http://www.communitymx.com/content/article.cfm?cid=4E2C0) , but the styling is a bit more involved. There is also an old Explorer list bug that must be dealt with to keeps things looking good.

Details Of The Flyout CSS

Below is the minimal code set necessary to create the basic flyout menu:

The HTML:

```
 <ul>
  <li style="z-index: 10;"><a href="#">Home</a>
   <ul>
    <li><a href="#">Table of Contents</a></li>
    <li><a href="#">Site Map</a></li>
    <li><a href="#">Guestbook</a></li>
   </ul>
  </li>
 other LI's omitted to save space...
 </ul>
```

The CSS:

```
ul {
  width: 8em;
  background: #edc;
  border-top: 1px solid #000;
} /* Makes top border for 1st button */

li {
  list-style: none; /* turns off the list bullets */
  position: relative;
  border-left: 1px solid #000; /* side button borders */
  border-right: 1px solid #000; /* side button borders */
  border-bottom: 1px solid #000; /* bottom button borders */
  vertical-align: bottom; /* for IE5-Win */
}

ul a {
  display: block;
  padding: 5px 10px;
  vertical-align: bottom; /* needed for IE5-Win */
}

ul ul {
  background: #fed;
  position: absolute;
  left: -3000px;
} /* set flyout BG color and preset the positioning */

ul li:hover ul {
  left: 7em;
  top: 4px;
} /* make flyout appear when the LI is hovered */
```

Code Block 3

The first rule block sets a width, background, and top border on the main list and all the flyout lists as well. We chose to use EMs for the width so that when the user resizes text, the navigation menu will widen along with the text. The top border on the lists (ul selector), combined with bottom borders on all the list items (li selector) will take care of our top and bottom border needs.

You may recall in our previous series that the popup parents needed some kind of background before IE/Win would allow the hovering behavior. In this flyout example, however, the parent elements are relatively positioned rather than absolutely positioned, and they contain block links as well. So for both reasons the bug does not apply here. Later, when we introduce "sticky hovering," this IE/Win background

bug will return, but we'll easily deal with it then, since we already know how to handle that bug.

The next rule block (li) removes the default list bullets via the **list-style** property, sets the LI parent elements to **relative**, and assigns side and bottom borders that complete the needed border arrangements.

That leaves the **vertical-align:bottom;** rule, which is used on the link itself as well (ul a). Normally such a rule is not needed, but IE5/Win is so old and cranky that it sometimes needs reminding to do the correct thing. Oh well.

We then see the **ul ul {}** rule block, which gives any unordered list that is inside another unordered list a different background-color than the top-level list. It also makes them **absolute**, and sets a large negative **left** value to pull them far off-screen to the left. This block does not apply to the top-level list because it is not nested in another list, but the flyouts do fit that description and so will be selected by this rule block.

Finally we come to the really fun part, the **:hover** rule. Again we show the relevant code below:

The HTML:

```
 <ul>
  <li style="z-index: 10;"><a href="#">Home</a>
   <ul>
    <li><a href="#">Table of Contents</a></li>
    <li><a href="#">Site Map</a></li>
    <li><a href="#">Guestbook</a></li>
   </ul>
  </li>
 other LI's omitted to save space...
 </ul>
```

The CSS:

```
ul li:hover ul {
  left: 7em;
  top: 4px;
} /* make flyout appear when the LI is hovered */
```

Code Block 4

Take a look at the selector in **Code Block 4**. It starts with **ul**, then a space, then the hover portion, **li:hover**, another space and a final **ul**. This selector selects any UL when it is nested inside an LI that happens to be getting hovered, and is in turn nested inside another UL. If you examine the nested list structure in the HTML above you can see that the flyout list is nested in a list item, which is nested in the top-level list. As soon as that list item is hovered, the rules in this block will be applied to the flyout UL.

Thus, hovering any top-level LI causes the nested list inside to obey a **left** value of **7em** rather than the **-3000px** we gave it before. With the **{left: 7em;}** rule applied, the nested list starts its left edge 7em from the left edge of its parent LI, and since that LI has the same 8em width as the ULs, there is a 1em overlap between the parent LI and the flyout list. We could have had the overlap be just a fraction of an EM, but why get even more complicated for a mere demo?

Then we added **{top:4px;}** just to make it look nicer. Shazzam! Our flyout menu is ready to go. Well, almost. The fact is, there are some problems to clean up before Mr. Navigation Menu is ready for the big bad Web. The most important one concerns the way older IE/Win browser versions handle bullets, or the

lack thereof, in lists.

This Space Reserved

Thankfully, IE6 handles list styling more or less like other modern browsers, but IE5.5 and IE5 have some pretty strange issues when it comes to the location where you would normally see a bullet. For this navigation menu we have removed the list bullet in the CSS, but those old IEs only partially comply with the rule. While they do remove the visible bullet marker, they seem to still reserve a 16px wide "zone" for the bullet anyway.

Because of some other IE bugs, we will have to apply the **Holly hack (http://www.communitymx.com/content/article.cfm?page=2&cid=C37E0)** to the ULs, the LIs, and the links too. Several pesky IE bugs are fixed in this way, by applying a height to those elements, and in the case of the list items, the bullet zone issue changes somewhat. Fortunately, the change actually improves and simplifies things, so much so that all we need to complete the fix is to apply a negative 16px left margin on the list items.

However, IE6 does not need this negative margin, so we must make sure only the older IE/Win versions will see this fix. Since we already need a **conditional comment** to hold our **csshover.htc Jscript**, we can just put our IE-only rules in there as well. In addition to the bullet fix and the .htc behavior call, we will add a **<noscript>** element that holds a style block containing just one special rule. All this will be explained shortly, but first view the complete conditional comment:

```
<!--[if IE]>

  <style type="text/css">
    body {behavior: url(csshover.htc);}

    ul a, ul, li {height: 1%;} /* Holly hack fix for IE bugs */

    li {
    margin-left: -16px;
    mar\gin-left: 0;
    }
  </style>

  <noscript>
    <style type="text/css">
      ul ul {position: static;}
    </style>
  </noscript>

<![endif]-->
```

Code Block 5

This special HTML comment hides everything inside it from all browsers except IE/Win. The first thing in the comment is a style block with three rule sets inside. The first is the behavior call for the Jscript, followed by the **{height:1%;}** for the ULs, LIs, and links. The final rule set in the style block targets the list items, applying **{margin-left:-16px;}**. Directly after that is same property, but with a different value **{mar\gin-left:0;}**. This second rule overrides the first margin rule, resetting the margin to zero. However, the presence of the escape slash (****) within the property name prevents the older IE versions from reading

the property name, so they ignore the rule. IE6 has no problems with such legal escape characters, so it *does* read the second margin rule, which is what we want since IE6 does not need the negative margin.

Using escapes like this to hide from the old IEs is great, but be sure that the escape is always inside the property name, and never place the escape directly before any character that is one of the first six in the English alphabet (a, b, c, d, e, f). If you do this, it's treated as a "hex code" and IE6 (and other browsers) will also fail to read the rule.

Make sure you don't use HTML comments inside the conditional comment, as comment nesting is strictly forbidden!

When Flyouts Fail

The **<noscript>** block is seen only by IE/Win browsers because it is inside the conditional comment. If scripting is disabled, the code inside the **<noscript>** element will be used only by IE browsers. In that case, a browser will parse the style block it finds within the <noscript> element and apply the rule **ul ul {position: static;}**. This overrides the **ul ul {position: absolute;}** rule found in the regular style sheet, as long as the conditional comment is located later in the head of the document than the link to the style sheet. If the style sheet link falls later, the selector in the <noscript> element will have to be made more specific in order to make sure it is applied when scripting is disabled. See our article on the cascade and specificity (http://www.communitymx.com/abstract.cfm?cid=AE76913AF0D2E5D2) for more information on this topic.

By making the flyouts **static** when scripting is disabled, they will appear as the nested lists they really are, and will stack vertically down the page. Their other stylings will remain intact, though. This allows those IE users (without scripting enabled) to still click on every link in the navigation menu. It may not be as pretty as the dynamic flyout menu, but at least those few users won't be shut out of your site.

Image 4 to the right shows how the menu looks when expanded. This graphic was made using Firefox. If the menu is seen expanded in IE/Win it won't look quite as clean as this. However, it is possible to add more rules to the CSS in the <noscript> element for the purpose of styling the menu when it is in its expanded state.

Internet Explorer for Macintosh has a greater problem in that it does not support the flyouts at all. Therefore, it's necessary to expand the menu for IE5/Mac users as well, whether scripting is disabled or not. This can be done using a copy of the CSS within the <noscript> block; placing it outside the conditional comment where IE5/Mac can see it.

Since we don't want any browsers *other* than IE5/Mac to see it, we'll need a way to show it only to IE5/Mac. Placing this copy of the list expanding CSS rule in a special CSS comment that only IE5/Mac actually parses as valid CSS code will solve the problem. Here's the complete code set to use:

Image 4: *The display of the fully expanded list sets as shown in Firefox.*

The CSS for IE5/Mac:

```
/* \*/ /*/
  ul ul {position: static;}
/* */
```

Code Block 6

This code needs to go inside the main CSS style sheet, somewhere after the rule it will override. If it comes before the **ul ul {position: absolute;}** rule, this altered rule for will have no effect for IE5/Mac.

We call this construction the **Anti-Mac-hack**, because it reverses the normal action of the Mac hiding hack. What happens is that when IE5/Mac sees that escape (\ , colored red in Code Block 6), it fails to recognize the star directly after the escape (and thus the closing comment construction, */) so it thinks the CSS comment is still active. Then when it sees the **/*/** part, the last two characters there appear to end the comment for IE5/Mac. Thus, the browser parses the following style block, applies that rule, and recognizes the last part of the code as a simple empty comment.

On the other hand, other browsers are not fooled by the escape (\ ), so they end the first comment right where it should end. Then the **/*/** part is viewed, not as the end of the first comment, but as the beginning of another one! Thus these browsers ignore the style rule, and also ignore the beginning of the last line. Finally they see the last two characters, ***/**, and end the comment. Now all browsers are back to normal parsing, but IE5/Mac has been fed styles that no other browsers recognized. Very handy, indeed.

By providing these tricks, we have arranged for browsers which cannot handle the flyout menu to get a

usable expanded one instead. So all browsers later than Nav 4 will get some kind of navigation, and nearly all of them will get our pretty flyouts. Neat.

Even Nav 4 may be fed a dumbed down nav using similar tricks. See the Caio hack (http://www.communitymx.com/content/article.cfm?cid=A3E04D26B481EAB6) article for details on how to accomplish it.

Up Next

In the upcoming installment we will make certain code modifications so that the hovering will be more "sticky," making the menu more user friendly. Stay tuned!

Approximate download size: 128k

Keywords

CSS, flyout, flyouts, dropdown, menu, navigation, :hover, csshover.htc, Jscript, 'sticky hovering,' tooltips, list, lists, sublist, submenu, z-index, holly hack, conditional comment, Anti-Mac-hack, escape, noscript