NOVEMBER 07, 2003 No. 162

Suckerfish Dropdowns

by PATRICK GRIFFITHS, DAN WEBB

Published in: Accessibility, Browsers, CSS, HTML and XHTML, Scripting, User Interface Design

"DHTML" dropdown menus have notoriously involved nasty big chunks of JavaScript with numerous browser-specific hacks that render any otherwise neat, semantic HTML quite inaccessible. Oh, the dream of a lightweight, accessible, standards-compliant, cross-browser-compatible method! Enter Suckerfish Dropdowns.

Meet the markup

To start, we should use the best method for defining a navigation menu — a list. For this example, we will work on a simple HTML unordered list. {Line wraps are marked ». *–Ed.*}

```
<ul>
  <li>Sunfishes
    <ul>
      <li><a href="">Blackbanded»
        sunfish</a></li>
      <li><a href="">Shadow bass</a></li>
      <li><a href="">Ozark bass</a></li>
      <li><a href="">White crappie</a></li>
                </ul>
        </li>

  <li>Grunts
    <ul>
      <li><a href="">Smallmouth grunt
        </a></li>
      <li><a href="">Burrito</a></li>
      <li><a href="">Pigfish</a></li>
    </ul>
  </li>

  <li>Remoras
    <ul>
      <li><a href="">Whalesucker</a></li>
      <li><a href="">Marlinsucker</a></li>
      <li><a href="">Ceylonese remora</a></li>
      <li><a href="">Spearfish remora</a></li>
      <li><a href="">Slender suckerfish</a></li>
    </ul>
  </li>
</ul>
```

Quite straightforward really — nice and neat HTML that, as a result, is highly accessible. But now we want to transform this into a dynamic list — the first level of list items will make up a horizontal menu bar from which the second level lists will drop down.

Styling it

To get started, all of the lists need to be jigged around a bit — namely, the padding and margin set to zero and the list-style set to none:

```
ul {
  padding: 0;
  margin: 0;
  list-style: none;
  }
```

Now we need to transform the first-level list into a horizontal menu bar. There are a number of methods to do this, discussed in detail elsewhere (http://alistapart.com/articles/taminglists/) . We could display the list-items inline (display: inline**), but for this example, we are going to float them to the left.**

```
li {
  float: left;
  position: relative;
  width: 10em;
  }
```

The position has been set to relative **because we want the position of the second-level, nested lists to be relative to the first-level list items and the width has been set to space it out a bit. The dropdown menu is coming together.**

The next step is to tackle the second-level lists that will be the dropdowns themselves:

```
li ul {
  display: none;
  position: absolute;
  top: 1em;
  left: 0;
  }
```

This positions the second-level lists absolutely (pulling them out of the flow of HTML into a world all of their own) and sets their initial state to not be displayed. If you substitute display: none **with** display: block**, you will see the need for the top and left properties in Internet Explorer, because without them, IE will align the second-level lists to the top right of their relative parent rather than the bottom left. Unfortunately, this IE fix will mess things up in browsers like Opera, so add the following CSS to reset the top and left properties on all but IE browsers:**

```
li > ul {
        top: auto;
        left: auto;
        }
```

And now, making the sucker work. To make a second-level list appear when its parent list item is “rolled over,” we simply need to add the following:

```
li:hover ul { display: block; }
```

Which says that any list that is nested in a list item that has the cursor hovering over it should be displayed.

Finally, because the lists are floated left, the content underneath it needs to be set free of the floating by applying clear: left **to it.**

Hold on a minute!

“This dropdown malarkey doesn't work!” I hear 102.6% (or the latest percentage being thrown about) of you cry. I am, as some might have guessed, talking about Internet Explorer users. The more you use and develop with browsers such as Mozilla the more you realize how pathetic Internet Explorer can be when it comes to web standards. The :hover **pseudo class should work with any element, but in Internet Explorer it only works with links. So. What’s the use in a dropdown menu when it only works on -2.6% of browsers? Not much, to be honest. We need to apply a little bit more magic.**

DOM-based scripting to the rescue

We’ve established IE’s lack of support for the :hover **pseudo class, but by using the Document Object Model, we can attach mouseover and mouseout events to any element. This is good news for us because it means that with a simple snippet of JavaScript we can effectively patch IE’s** :hover **problems.**

Because IE is blind we need to find another way to identify the properties of the :hover **pseudo class. With JavaScript, we know that we can manipulate the className property of an element so what we are going to do first is alter the CSS:**

```
li:hover ul{ display: block; }
```

becomes:

```
li:hover ul, li.over ul{ display: block; }
```

Now we can invoke the :hover **CSS rules by adding the class** over **to the desired element. We also need a way to tell IE which of the UL elements on the page we actually want to be our dropdown menus. We can do this by giving an** id **to our root** ul **element:**

```
<ul>
```

becomes:

```
<ul id="nav">
```

Now that we have a means of identifying the root ul **element of our dropdown list, we can grab this element and loop through all of its child elements, attaching mouseover and mouseout events to all the** li **elements nested within it. And this is how it’s done:**

```
startList = function() {
if (document.all&&document.getElementById) {
navRoot = document.getElementById("nav");
for (i=0; i<navRoot.childNodes.length; i++) {
node = navRoot.childNodes[i];
if (node.nodeName=="LI") {
```

```
node.onmouseover=function() {
this.className+=" over";
  }
  node.onmouseout=function() {
  this.className=this.className.replace(" over", "");
   }
   }
  }
 }
}
window.onload=startList;
```

On page load, the startList **function is invoked. The function determines if the browser is actually IE 5 or greater by checking for the existence of the** document.all **object and** document.getElementById **function. This is a bit of a crude way of doing it but it's short and sweet — and since we are trying to make a compact solution, this will do. It then loops through, enabling mouseover and mouseout events which add and remove the** over **class from the** className **property of the element.**

There you have it. If you got lost anywhere, have a look at a commented, bare-bones example (http://www.htmldog.com/articles/suckerfish/bones/) in action.

GILLS, FINS, SCALES...

So far things are a little bare. The idea has been to show the basic workings of the Suckerfish Dropdown, but CSS can make things look a lot prettier (http://www.htmldog.com/articles/suckerfish/example/) . An obvious starting point would be to apply a background color to the second-level lists.

After resetting the top and left properties as described earlier (#resettop) , dropdowns in the pretty (http://www.htmldog.com/articles/suckerfish/example/) example appear directly below menu labels in most modern browsers, but unfortunately not in all. In Safari 1.0, they still drop down from the top left edge of the screen. {Check the discussion forum (http://alistapart.com/discuss/dropdowns/) for workarounds. *–Ed.*}

FURTHER USABILITY AND ACCESSIBILITY

Making links out of the first-level list items will allow tab-stopping for readers who don't use pointing devices. Pointing those links to higher-level pages than the links in the dropdowns would be even better.

Learn More

Related Topics: Accessibility, Browsers, CSS, HTML and XHTML, Scripting, User Interface Design

About the Authors

Patrick Griffiths is a freelance webmaker based in London who has a penchant for soul music, evolution and walking his pet website, HTML Dog (http://www.htmldog.com) . He sometimes prefers the moniker PTG (http://www.htmldog.com/ptg/) , depending on what mood he's in.

Dan Webb (www.danwebb.net) is a web developer and wannabe DJ. His recent work includes implementing standards-based, accessible sites and web applications for UK government bodies and making people dance around in murky London bars.

ISSN: 1534-0295