Horizontal menu bars with CSS

This article will show you, how to build horizontal menu bars with WebYep. In your career as a web designer, you have probably seen many different stylings for navigation menues. Back in the dark millennium, it was considered good practise to take an innocent table tag and stuff it with links and images.

To achieve those fancy new rollover effects, one would have added things like `onMouseOver="MM_swapImage('Image12','',imageo3,1)` to the links (and of course another for `onMouseOut`). The result was a bloated monster, unaccessible by anything but sight and a nightmare to maintain. If you've ever had to redesign such a menu, you'll know what I'm talking about. Those were the days...

Today there really is no need to abuse tables for menues, because the support for CSS has grown among all modern browsers. This article will show you, how to implement menues in web pages with HTML and CSS alone.

Simple menu bars

As for menues, it has become widely accepted to use unordered lists and this is what WebYep does.

Let's start with a simple example. Consider the following markup:

```
<ul>
  <li><a href="#">Menu 1</a></li>
  <li><a href="#">Menu 2</a></li>
  <li><a href="#">Menu 3</a></li>
</ul>
```

This will define a simple structure with 3 entries. Throughout this tutorial, we'll wrap the unordered list containing the menu in a `<div>` with an `id='myMenu'`, so we can easily address the `<ul>` by its parents id and wont mess up other `<ul>` elements on the page.

The first thing we want to change, is the default behaviour and appearance of this `<ul>`.

```
#myMenu ul {
  list-style-type: none;
  position: relative;
  margin: 0;
  padding: 0;
}
#myMenu ul li {
  float: left;
}
```

By floating the `<li>` items left, the `<ul>` will no longer go from top down, but will rather look like a horizontal bar. Now we have to take care of the actual links. In order to give the whole thing the look and feel of an actual bar, we'll display the links as blocks, so we can give them dimensions, as well as padding.

```
#myMenu ul li a {
  display: block;
  padding: 3px 10px 1px;
  background-color: #444;
  color: #fff;
  font-family: sans-serif;
  text-decoration: none;
}
```

Now this looks more like it, but we'd like to have a different style when hovering the mouse over an entry. For this we'll use the CSS pseudo class `:hover`, which applies only if the mouse is over the specified item. We dont need to specify every style rule again, just the ones that are supposed to change when the mouse is over the link.

```
#myMenu ul li a:hover {
  color: #0b0;
}
```

So far, we've only taken care of the menu itself, but since it's content is floated, we need to make sure, that content after the menu doesn't wrap itself around the floating menu. In our example pages, we've put the page content in another `<div>` with an `id='myContent'` and cleared the previous float rule.

```
#myContent {
  clear: both;
  padding: 10px 0;
}
```

More complex

Now that we have a vertical menu bar, we'd like to have submenues as well. This is achieved by nesting unordered lists, so we take our previous example and extend it:

```
<ul>
  <li><a href="#">Menu 1</a>
    <ul>
      <li><a href="#">Submenu 1.1</a></li>
      <li><a href="#">Submenu 1.2</a></li>
    </ul>
  </li>
  <li><a href="#">Menu 2</a>
    <ul>
      <li><a href="#">Submenu 2.1</a></li>
      <li><a href="#">Submenu 2.2</a></li>
    </ul>
  </li>
  <li><a href="#">Menu 3</a>
    <ul>
      <li><a href="#">Submenu 3.1</a></li>
      <li><a href="#">Submenu 3.2</a></li>
    </ul>
  </li>
</ul>
```

Each list item has now got an additional unordered list with 2 list items. At this point, notice how lean the code still is. Everything is still easily readable, editing is no problem and the markup is not cluttered with things only good for decoration.

Of course, we don't want the submenues to be displayed all the time, so we add another rule to our style sheet:

```
#myMenu ul ul {
  display: none;
  position: absolute;
}
```

But how do we display the submenues now? Again, the `:hover` pseudoclass comes in handy and since we don't want the submenues to be vertical bars as well, we also quit floating subsequent list items.

```
#myMenu li:hover ul {
  display: block;
}
#myMenu li ul li {
  float: none;
}
```

Going fancy

We now have a working menu bar with submenues that pop up, when the mouse hovers over them. Now we want to give them the little extra, that will distinguish our site from the rest.

Popping up is no longer good enough - we want our submenues to elegantly fade in! This will be possible with the new generation of CSS, but since not all browsers support this, we'll have to use JavaScript for now.

For this example we'll use the popular library jQuery. Simply include the library and the program code for the effect in the `head` section of your page:

```
<script type="text/javascript" src="jquery-1.5.1.min.js"></script>
<script type="text/javascript">//<![CDATA[
$(document).ready(function(){
  $('#myMenu ul ul').css({display:'none'});
  $('#myMenu > ul > li').hover(
    function(){$(this).find('ul').stop(true,true).fadeIn(300);},
    function(){$(this).find('ul').stop(true,true).fadeOut(300);}
  );
});
//]]></script>
```

Also remove the style rules that show/hide the submenues, since they will interfere with the animation:

```
#myMenu ul ul {
  display: none;
  position: absolute;
}
#myMenu li:hover ul {
  display: block;
}
```

Conclusion

As you can see, there really is no need to abuse tables for menues nowadays. With a little knowledge of CSS, such menues are easy to create. Download the packed examples and toy with the styles or the script code. For example, try to replace `fadeIn` and `fadeOut` in the jQuery code with `slideDown` and `slideUp`. After all, the best way to get experience with something, is to actually do it.