Mastering HTML5: part 5

Animation can help to bring a website to life. Nik Rawlinson shows you how you can use the tools built into HTML 5 to achieve something flashy without the need for Flash

INTRODUCTION

In *Web Expert*, *Shopper* 284, we looked at how you can use text effects to draw visitors' attention to specific elements on your page, giving them extra prominence with the use of shadows. We also showed you how to rotate them to create more dynamic layouts.

In the final instalment of our guide to mastering HTML5, we'll consider CSS3's most advanced and ambitious feature: native animation. This allows you to achieve effects that were once open only to those who used Flash or a rival dedicated tool.

Nik Rawlinson
Web developer

letters@computershopper.co.uk

When used with care and moderation, animation is a useful tool. Its implementation can be purely decorative – as is the case with most animated web ads – or functional, controlling the behaviour of specific page elements.

To demonstrate how animation works, we'll be taking the functional approach with a practical example that you can implement on any site – a floating panel that slides in from the edge of the browser. Such panels are common on sites that encourage feedback from their visitors. In this instance, we're going to use our panel to provide the visitor with supplementary information; implementing a feedback form instead would be a simple matter of swapping out the panel content for the necessary form and applying a regular submit button tied to your form-handling routine (this will often be provided by your ISP as part of your hosting package).

Our panel is a simple flat graphic – a GIF created in a standard photo editor and saved with a transparent background. The transparency is important, as we want the body of our page to remain visible behind the empty areas of the panel when it slides into position, giving the pull-down tab a greater sense of realism. The only parts of the panels that should be opaque are the background on which we'll place the content and the tab that will always be visible, poking down from the top of the browser window.

Our graphic is 300 pixels wide and 242 pixels tall, but yours should be large enough to fit whatever content you need it to accommodate.

BUILDING YOUR PAGE

Our page is deceptively simple. We're coding it using HTML5 so that we can take advantage of the simpler header structure but, beyond that, the remainder of the page is the same as it would be in HTML4.01.

We have just two elements: the primary content, which is organised in a layer called 'mainbody', and the palette itself, which is split into two parts – one that defines the physical dimensions encompassing the panel and the protruding tab, called 'menutab', and one for the contents that sit on top of the panel, called 'menutabcontents'.

For the time being, we have filled the body of our page with dummy text, and positioned an image, some text and a button on the tab, as follows:

```
<!DOCTYPE html>
<head>
<meta http-equiv="X-UA-
Compatible" content="IE=9" />
<title>Slider</title>
<link href="slider.css"
rel="stylesheet"
 type="text/css">
</head>
<body>
<div id="menutab">
<div id="menutabcontents">
<p><b>More info</b></p><img src="lamb.jpg"
 align="left" width="150" height="100"
 style="margin-right: 10px;"/>Here is
 all of the further information you need
 to know. You can even include images
 if you want.<br /><br /><form><input
 type="button" value="You can include
 forms. Here's a button"></form>
</div>
</div>
<div id="mainbody"><p>Lorem ipsum dolor
 sit amet, consectetur adipiscing elit.
 Aliquam eleifend convallis leo sit amet
 vestibulum. Nunc eu faucibus risus. Duis
 dictum, dolor a ultrices dictum, lorem
 nisl viverra nisl, a aliquet sapien
 lectus et felis.
</div>
```

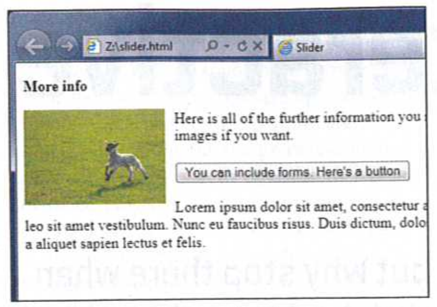
If you preview the page in your browser, you'll see that while the image sits to the left of the panel contents, everything else on the page flows on from what precedes it. We want the panel and its contents to slide into view, but there's no 'intelligence' on this page yet, and nothing to tell the browser how to handle the panel animation.

You could conceivably add animation using JavaScript, but it's far more efficient – in terms of the amount of bandwidth consumed when displaying the effect, and in the amount of code required to implement it – to look to CSS instead and define its behaviour at the same time as the page layout.

DOING THE MATHS

We have designed our panel with a tab on the bottom as it will drop down from the top of the browser window. This tab will remain visible at all times, but when the mouse isn't hovering over it, the remainder of the panel will be kept hidden. To achieve this, we need to position the main portion of the panel beyond the browser boundaries. We'll do this by setting a negative upper margin in our CSS code. The top of the tab, where it joins the main body of the panel, is 215 pixels from the top of the image canvas, so the upper margin will be set at -215px.

▲ So many sites use the dead browser space around their page designs to display buttons and tabs, visitors will already be familiar with the concept



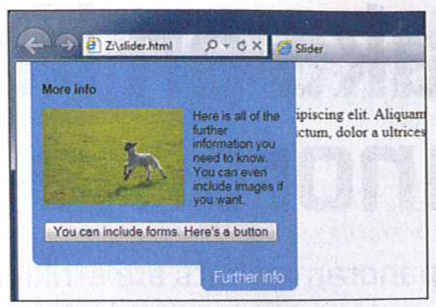
More info

Here is all of the further information you images if you want.

You can include forms. Here's a button

Lorem ipsum dolor sit amet, consectetur leo sit amet vestibulum. Nunc eu faucibus risus. Duis dictum, dolo a aliquet sapien lectus et felis.

▲ The contents of our panel and our page are the same thing. We just need to split them on to separate layers



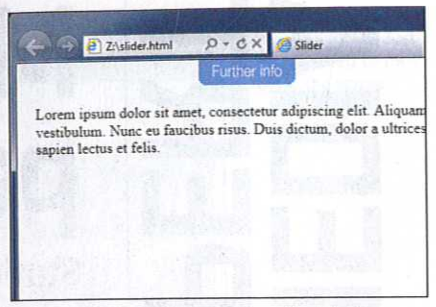
More info

Here is all of the further information you need to know. You can even include images if you want.

You can include forms. Here's a button

Further info

▲ Hover the mouse over the tab and it drops down the full panel; this becomes an active area so the visitor can move their mouse around within it



Further info

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam vestibulum. Nunc eu faucibus risus. Duis dictum, dolor a ultrices sapien lectus et felis.

▲ When the mouse is away from the tab, the panel remains out of view, so it doesn't interfere with the page content

WRITING THE CSS

To write the CSS for this page, we need to think in three dimensions. We need to specify the horizontal and vertical positions of the elements on our page (we can usually leave these alone, unless we need to specifically anchor objects within the layout). In addition, we need to consider their relative heights, as though they were coming out of the screen towards us.

We want our panel to slide down over the main page body, so we need to tell the browser to position it there using the z-index attribute. Z-index describes the third axis on a page (x-axis, y-axis, z-axis), which in this case comes vertically out from the page. Each object that you want to stack is given a value, with higher-valued objects appearing further up the stack. To position our panel higher than the body of our page, we can either give the menutab layer a z-index of 2 while giving the mainbody layer a z-index of 1 or, as we'll do here, give the mainbody layer a value of -1 while leaving the menutab on layer 1.

Z-index works in conjunction with the position element. If you don't tell the browser whether your positioning is relative, static, absolute, fixed or inherited, your stack won't be rendered properly. Therefore, we need to decide what the browser should use as the point of reference when positioning each element.

In the case of the mainbody layer, it's the document body. We're going to set this to start at the very top of the browser window with a top margin of 0px; by setting the mainbody layer to 'absolute' and its top to 30px, we can be absolutely certain that the mainbody will leave a gap of exactly 30 pixels at the top of the browser window. This is sufficient room to display the tab of our panel, which is always visible, even when the panel is hidden – without the tab fouling the page contents.

We want to position the panel in relation to the browser window and not the page. If we did position it in relation to the page, it would disappear as soon as the page was scrolled to read any excess body content that didn't fit within the browser window. By anchoring the panel to the browser window, it will ignore any scrolling.

The 'position' attribute for the menutab is therefore set to 'fixed', and we have also called in the graphic that we created earlier to be displayed as its background. Our initial CSS, which is saved in an attached file called 'slider.css', as referenced in the header of the HTML page, looks like this:

```
body {
  margin-top: 0px;
  overflow: scroll;
}
#menutab {
  width: 300px;
  height: 242px;
  background: url(tab.gif) no-repeat;
  margin-top: -215px;
  z-index: 1;
  font: 0.8em Arial;
  position: fixed;
}
#menutabcontents {
  padding: 10px 20px 10px 20px;
}
#mainbody {
  z-index: -1;
  position: absolute;
  top: 30px;
  left: 20px;
}
```

As you can see, the menutabcontents layer has been padded to keep the contents away from the very edge of the panel and improve its appearance.

CHANGING YOUR POSITION

The page is starting to take shape. We can now see the panel tab hanging down below the top edge of the browser, and its contents are hidden because they have been moved up by the negative top margin value to a position above the browser's viewable area.

To get the tab to roll down, we need to use the :hover class variable. This is often used to define how a link should change when a site visitor hovers their mouse across it. You might, for example, want to remove the underlining from the links on your pages to make them less obvious, but have them reappear when the visitor hovers their mouse across them.

In this case, we don't want to change the styling of an element; we just want to change its position. So we'll add a new :hover-specific class that's attached to the menutab layer that normalises its position:

```
#menutab:hover {
  margin-top: 0px;
}
```

Reload the page and move your mouse over the protruding tab, and you'll see that the whole panel now pops out, overlaying the main body of the page without disturbing the main body's position, and that you can also move your pointer around within the popped-out panel without the panel disappearing, which it does when we move our mouse away.

We're halfway there, but the result could be better. Because of the shape of the tab, which mimics closely that of the panel's main body, it may look like it's being magnified rather than sliding into position. We can fix this by slowing down the transition using the transition-duration element and its -moz, -ms and -webkit variants to target Firefox (-moz), IE (-ms) and both Chrome and Safari (-webkit).

We're doing a bit of forward-planning here as far as Internet Explorer is concerned, because version 9 doesn't support transition durations fully, although work is underway to integrate it into future releases. Still, any work done at this stage will put you in good stead as the browser is upgraded.

We don't need to slow the animation down much to achieve the result we're after. In this case, we're going to set a duration of half a second on all browsers by adding the following lines to the styling of the menutab layer:

```
-webkit-transition-duration: 0.5s;
-moz-transition-duration: 0.5s;
-ms-transition-duration: 0.5s;
transition-duration: 0.5s;
```

Save your changes and then reload your page. When you move your pointer over the tab and away again, you'll see that the panel animation is now a smooth slide into view, then back off the edge of the browser window again.

APPLYING EFFECTS ELSEWHERE

The key to achieving a realistic result in all animation tasks is to control the speed at which the motion completes. When using this technique elsewhere, the one consideration that should remain uppermost in your mind, beyond the start and end position of your assets, is the speed at which the assets should transition from one state to the other. Master this, and you will afford your site a degree of polish that puts it several steps ahead of the competition. CS

Next month

GET STARTED WITH JQUERY
Create dynamic, interactive web pages with ease using jQuery.