

/CSS/create scrollable page content



Craig Grannell shows you how to emulate iframes: discover how to place scrollable content within your page designs without moving away from current web standards

- Knowledge needed: Intermediate CSS and HTML
- Requires: A text editor
- Project time: 15 minutes

If you've been immersed in the world of web design for a while, you'll know that it's fluid in many ways. Design styles come and go, the market share of individual browsers peaks and troughs (or, in Netscape's case, leads the market and then spontaneously combusts due to one particularly disastrous release), and software wows the market one minute, only to be rapidly replaced by the next big thing. Under the hood, things are just as changeable: new technologies arrive, develop and evolve, while others are quickly put out to pasture. The technical term for the latter of those things, when it comes to web standards is "deprecated", which essentially means "marked for removal from the standards." In plain English, that's "stop using this right now, because future support is in no way guaranteed. Otherwise, we'll get terribly cross with you."

A quick trawl through any hefty HTML tome you have handy will confirm that a surprising number of commonly used elements and attributes are on seriously borrowed time, including the stupefyingly popular "target" for anchors. If you're using a transitional XHTML DOCTYPE, you can still get away with using the odd deprecated element, but designers and applications are increasingly moving towards XHTML Strict, making that final break away from presentation-oriented code within markup.

Perhaps one of the most surprising elements marked for deprecation is "iframe". Although frames-based websites have pretty much gone the way of the dinosaur, iframes have a specific purpose that can be useful when it comes to web design: they enable you to create areas of scrollable page content, making it possible to fit a large chunk of content into a limited space. Although you shouldn't do this with content that users need to be able to see and access immediately, it's a useful device for displaying things like a news archive that shows the most recent items, but provides quick access to several days' worth of posts (see pixelsurgeon.com, pictured below, for a good example), or for embedding terms and conditions within an online store page, rather than resorting to pop-up windows.

The overflow property

The key to emulating an iframe in these standards-compliant times is the CSS overflow property. By default, if you set the dimensions of an element (such as a div) to values that are too small to contain the element's content, one of two things happens, depending on the browser: in older versions of Internet Explorer (pre-7), the element's height stretches in order to show all of the content. In every other browser, the content spills out of the element, so the element's background only shows within the defined dimensions. In either case, the space taken up is not the size defined. By using the overflow property in your style sheet, you can amend this behaviour. Take, for example, the following HTML:

```
<div id="scrollable">
[lots of content goes here]
</div>
```

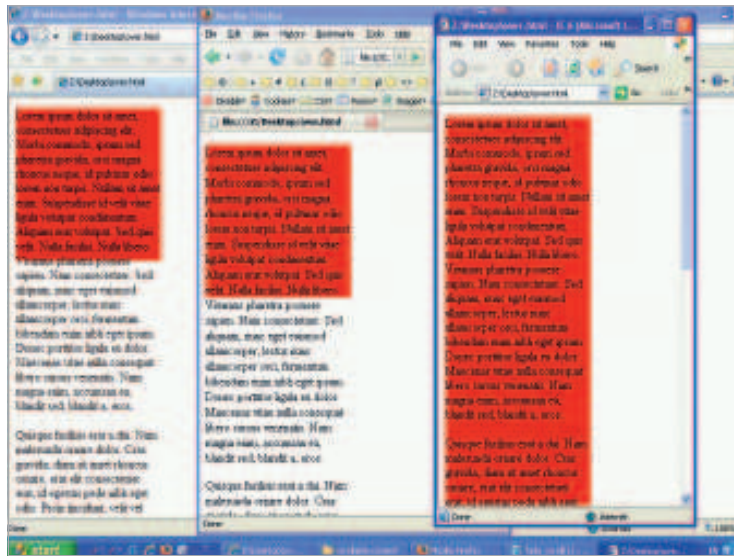
By styling the div using the following rule in CSS, you'll see a 200-pixel-square grey background in Internet Explorer 7, Opera, Safari and Firefox, with content



The final result This tutorial's web page, with its right-floated div offering scrollable content that fades into the background, thanks to two cunningly positioned divs with PNGs



Enhancing usability The front page of pixelsurgeon.com uses a scrollable area to house news items, providing immediate access to new posts and fast access to the recent archive



Comparing browsers On the right: how IE6 deals with content that overflows the defined dimensions of an element. On the left: IE7 and Firefox show how it should be done

continuing down the page, and a long, narrow, grey box in Internet Explorer 6 with the content inside it.

```
#scrollable {
background: #666666;
height: 200px;
width: 200px;
}
```

However, add "overflow: auto;" and the div gains a scroll bar. Just like with an iframe, the content stays within the defined dimensions, and you use a scroll bar to move through it. (By changing the auto value to scroll, you can force scroll bars at all times. However, by using 'auto', only relevant scroll bars are shown. For example, you won't see a horizontal scroll bar if the content fits within the horizontal confines of the box it's in.) Note that the scroll bar is placed within the defined width of the box – it's not added to the outside of it. Therefore, if you have a 200-pixel-wide image in a 200-pixel-wide div that has the overflow value set to auto, you'll see a horizontal scroll bar.

While the scrollable content is cosmetically akin to an iframe, and functions in the same way, there are key differences. The most obvious is that the content for the scrollable area is within the same document, unlike an iframe's, which is in an entirely separate HTML file. However, should you want to keep the content of a scrollable div external to the page it's on, you can always use a PHP include to keep the content separate – at least from a file-management standpoint.

Of course, a scrollable div is just the same as any other, and it can be styled. Therefore, the rest of this tutorial will show you how to put the page together in the 'scrollable-content' folder on the CD.

Adding some style

Open the 'scrollable-content' folder on the CD and you'll find a selection of files. Drop the HTML document on to a browser and you'll see the page. It's fairly basic stuff – a heading and a few paragraphs of text, and a floated box that has

Expert tip Horizontal and vertical overflow

Unfortunately, there are no fully supported means of specifying different values for horizontal and vertical overflow. Although the overflow-x and overflow-y properties exist (indeed, overflow is actually a shorthand property), and happily work in Firefox and Internet Explorer, both fail at the time of writing in Opera and Safari. While Firefox and Internet Explorer have the lion's share of the browser market, ditching five per cent or more of your users isn't a great move, so it's best to avoid using overflow-x and overflow-y until Safari and Opera catch up.

Hiding overflow content

Is IE6 misbehaving again? Here's the solution

One of the overflow property values that hasn't been mentioned elsewhere in this tutorial is "hidden". As you might expect, this crops any content that's too large for the defined dimensions of a container, and it should therefore be used with extreme caution, since a lack of scroll bars means that users will find it hard to access hidden content.

However, the hidden value has other uses, notably when Internet Explorer 6 is being unruly regarding CSS-based layouts. Sometimes, the browser will stretch divs to fit unbroken content that's too large, and by setting overflow to hidden, that content won't affect the layout (for example, floats may otherwise appear one under the other, rather than being stacked horizontally). This is primarily of use for non-text-based content. When you find Internet Explorer misbehaving regarding text content, stretching a container to display the content, you can instead define "word-wrap: break-word;" in an IE-only style sheet, and IE will break the text string(s), reverting the layout to how it should be. That said, on those occasions, you may want to edit your text or amend the design, because long strings will stick out of their containers in more standards-compliant browsers, which doesn't look that great, either.

The content for the scrollable area is within the same document, unlike an iframe's

scrollable content. As an added touch, the scrollable content fades (top and bottom) into the background of the main content area. This requires more div elements than a typical scrollable-content div would, and while the markup strays slightly from extremely tight semantics, it's still valid and doesn't create problems.

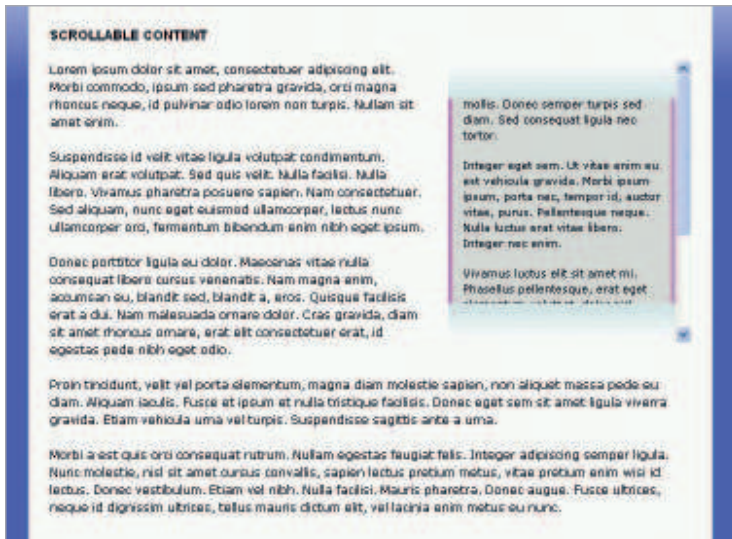
Open 'scrollable-page-content.html' in a text editor and examine the structure of the web page's body section. The content is within a wrapper div, which has a level-one heading, a div with a class of scContainer ("sc" cunningly standing for "scrollable content" in this tutorial), and a bunch of paragraphs. The scContainer div has nested within it two divs for the fade effect (with class values of fadeTop and fadeBottom, respectively) and a div to house the actual content, whose class value is scContent.

In the style sheet, 'scrollable-page-content.css', the first half-dozen rules concentrate on dealing with the basics for the page layout. The '*' rule's property values remove padding and margins from all elements, and the html and body rules set the default font size to 62.5 per cent (enabling subsequent font sizes to be defined in ems, using values that are a tenth of the target pixel size), the latter also applying a background image and colour to the page (the light-blue-to-dark-blue gradient). The #wrapper rule defines the wrapper div's width, padding, borders, background colour and alignment (the margin values centre the page horizontally in the browser window), while the h1 and p rules style the level-one heading and paragraphs, respectively.

Styling the scrollable content

All subsequent rules in the style sheet are for the scrollable content container and its content. The first, .scContainer p, resizes the text within the container, making it easier to distinguish from other page text. The next two rules style the container and the div that holds the container's content:

```
.scContainer {
position: relative;
float: right;
width: 250px;
margin: 0 0 20px 20px;
}
.scContent {
padding: 20px 15px;
width: 220px;
height: 250px;
}
```



PNG problems IE6 splutters upon encountering a PNG with alpha transparency, so make alternative arrangements by way of an override rule in a conditional comment

```
>> overflow: auto;
background: #dddddd url(sc-background.gif);
}
```

By setting position to relative for the container, absolute positioning values for nested elements are taken from the top-left of the container rather than the page as a whole – in other words, “top: 0;” for an absolute-positioned nested div refers to the top of the .scContainer div, rather than the top of the browser window view area. Other property values float the container div right, assign it a width and then set bottom and left margins, so that the div doesn’t hug other page content.

For the content container (the scContent div), dimensions are assigned, along with a background colour and image – the image being two vertical stripes to further accentuate the box’s content – and overflow is set to auto. Padding is assigned, and the top and bottom padding values (20px in this case) are particularly important, because if they were omitted, the top of the text may appear underneath the more “solid” areas of the PNG (and the same at the bottom when the div is fully scrolled).

Which brings us to the final rules in the style sheet. These two style the divs that create the fade effect: .fadeTop and .fadeBottom. Both contain very similar declarations, so only .fadeTop is shown in the following code block.

```
.fadeTop {
position: absolute;
width: 233px;
}
```

Resources Find out more online



The box model

All you ever wanted to know (but were afraid to ask) about the box model can be found on the W3 website, in the page excitingly entitled “CSS3 module: The box model. W3C Working Draft”. www.w3.org/TR/css3-box



CSS positioning

If “position: absolute” makes you go blank, you may want to brush up a bit on your CSS positioning properties. W3Schools has a page that enables you to do just that. www.w3schools.com/css/css_positioning.asp

```
height: 40px;
top: 0;
left: 0;
background: url(white-gradient-top.png);
z-index: 1;
}
```

The div’s background is a PNG file that has a white-to-transparent gradient. When text passes under it, this creates the illusion of text fading into the page’s background. The other properties position the div in an absolute manner, provide a height value equal to the PNG’s, a width equal to that of the container but minus the scroll bar, and then set the div’s position to the top-left of its parent. (The z-index value is required for Safari to correctly position the divs over the other content, but this doesn’t adversely affect other browsers.)

The differences in the .fadeBottom rule are that it uses a different background image (‘white-gradient-bottom.png’), and “top: 0;” is replaced by “bottom: 0;”, which positions the div at the bottom of its parent.

Tweaks and hacks

There are some issues with this design, which would only be apparent if you’d built it from scratch. However, if you temporarily move the files ‘ie-lt7-hacks.css’ and ‘mac.js’ from the folder the tutorial files are in (‘scrollable-content’), you’ll be able to see the issues for yourself.

The first one affects the Mac, whose scroll bars are narrower than those on Windows. Because of this, the background stripes for the scContent div don’t align correctly, the right-hand one being a couple of pixels to the left of the scroll bar. This isn’t a total disaster, but it’s nice to fix (and simple enough, too) – the JavaScript document ‘mac.js’ is attached to the web page, and this script loads the style sheet ‘mac-over-rides.css’ if the user is surfing with a Mac. This style sheet has two rules, the first of which sets a different scContent div background image (with the right-hand stripe further to the right), and the second of which adds two pixels to the width value of the two fade divs, thereby lining everything up nicely.

```
.scContent {
background-image: url(sc-background-mac.gif);
}
.fadeTop, .fadeBottom {
width: 235px;
}
```

But more important is Internet Explorer’s lack of support – prior to version 7 of Microsoft’s browser – for PNG alpha transparency. This omits the neat fade effect, instead displaying the two PNGs as solid blocks of doom. There are various workarounds for this. You could use IE-proprietary properties for forcing the alpha transparency, or you could keep things simple, like this:

```
.fadeTop, .fadeBottom {
background: none;
}
```

Here, I’ve used a conditional comment to link to an IE hacks style sheet, which handily removes the background images for the two divs in question. ●

When text passes under it, this creates the illusion of text fading into the page’s background



About the author

Name | Craig Grannell
Site | www.snubcommunications.com
Areas of expertise | Information architecture, site concepts, graphics, interface and front-end design
Clients | Swim~, Rebellion, IDG
If I were a kitchen implement, I’d be ... a blender