

/CSS/cater for screen resolutions



Does your site look great on all monitors, from the small to the ridiculously big? Craig Grannell shows you how to create a layout for multiple screen resolutions

Knowledge needed Intermediate

Requires A text editor or the code view of a web design application

Project time 30 minutes

Back in the darkest depths of time (well, the 1990s), there was a common screen resolution: 640x480. For a while, it ruled the roost, and all websites were made to happily fit within it. Then 800x600 came along to spoil the party. Web designers had to decide whether to abandon the smaller size, create fixed-width sites that worked in both common resolutions, or fashion liquid sites that would stretch to fit any resolution. Back then, the liquid-design route was an eminently sensible one: with the difference in screen sizes being relatively small, and few users running resolutions higher than 800x600, it was fairly easy to create stretchy designs that worked well on the majority of set-ups.

Things have changed. Although 640x480 is but a dim and distant memory, up to a fifth of users are clinging on to 800x600. And although 1,024x768 has rapidly become the most popular screen resolution, plenty of users run larger resolutions, thanks to the affordability of massive monitors. The upshot of all this is a problem for web designers. Design for the lowest common denominator and you may end up with a layout that looks strange on a much larger screen. However, design a liquid layout and you could end up with something that's unusable if someone surfs full-screen on a 24in display.

Of course, in some cases, you'll find that a fixed layout is the best choice to use anyway. Some designs only work well if you can position elements precisely where you want them, and if you're working on something like a blog, you want to ensure that your text columns remain narrow, otherwise the wonderful advice you're imparting to your readers will be hard to read. However, there's a useful compromise in the liquid vs fixed war: a semi-liquid layout, which stretches to a maximum point and then stays put, no matter how much wider you make the browser window. This enables you to create a layout that works well at 800x600, stretches to fit nicely within 1,024x768, but then stops getting any wider, so it doesn't end up looking bizarre on the monitors of people who clearly have far too much money.

This kind of design is made possible by the little-used `max-width` property, which has remained little-used due to IE's lack of support for it. Version 7 deals with it correctly, and it's possible to cater well for older versions of IE by way of a second style sheet attached using a conditional comment. Due to various other bugs that IE is afflicted with, the site in this tutorial will work rather differently in IE6 and below, often not resizing without a refresh when the browser window is narrowed. This, according to tests, is because of the margin settings applied to the wrapper. Removing these (resulting in a left-aligned site) removes this shortcoming, but as those users with small monitors will likely be surfing full-screen anyway, this minor problem isn't really all that much of a concern.

Set things up

Copy the 'files-to-use' folder from the CD. You'll find the following files: 'max-width.html' (the web page that you'll be styling), 'max-width.css' and 'lte-ie6.css' (the main and IE-specific style sheets), plus 'masthead.jpg' (a background image for the masthead). Open 'max-width.html' in your code editor. As you can see, the file is an XHTML 1.0 Strict document. Within the head section, you'll find a meta element for defining the page's content type, a title element, a style element for importing the main CSS document, and a conditional comment with nested link element, to attach a style sheet that will deal with some of IE's shortcomings (pre-IE7).

```
<meta http-equiv="content-type" content="text/html; charset=utf-8" />
<title>PWD37</title>
<style type="text/css" media="screen">
/*  */
@import url(max-width.css);
/*  */
</style>
<!--[if lte IE 6]>
<link rel="stylesheet" type="text/css" href="lte-ie6.css" media="screen" />
<![endif]-->
```

The web page's structure is simple. All of the content is placed within a wrapper div. Nested within are `masthead` and `content` divs, one under the other, to house the masthead/navigation and content respectively. In each of these is a further nested div; within the `masthead` div is a `navigation` div to enable the navigation bar to be positioned in a specific location. And within the `content` div is a `sidebar` div, which houses supplementary content.



Looking good in IE7 In Internet Explorer 7 and other compliant browsers, the site works much as you'd expect. Here, it's in a maximised window on a 1,024x768 display

Resources Find out more online



W3C Markup Validator

The online Markup Validation Service is essential for checking that your code is compliant, and testing takes no time at all.

validator.w3.org



W3 Schools

This site has thorough HTML and CSS sections, enabling you to learn about elements, attributes, rules, properties and values.

www.w3schools.com

```
<div id="wrapper">
<div id="masthead">
  <div id="navigation">
  </div>
</div>
<div id="content">
  <div class="sidebar">
  </div>
</div>
</div>
```

As always, content is marked up as simply as possible. An unordered list is used to structure the navigation bar. In the **content** div (aside from the **sidebar** div) are headings and paragraphs, and that's it.

Open 'max-width.css' and add the two rules below. The first zeros margins and padding for all of the web page's elements, and the second sets the page's **body** element's width to 100 per cent (the full browser window width).

```
* {
margin: 0;
padding: 0;
}
```

```
body {
width: 100%;
margin: 20px 0;
color: #000000;
background-color: #493c30;
}
```

The containing **wrapper** div is styled using the rule below. The margin settings centre the div, and the addition of the border makes sure that the site content stands out from the page background. The **max-width** setting is perhaps the most important one, though. In compliant browsers (everything bar pre-IE7), it ensures the layout is liquid until the **wrapper** div reaches a width of 950 pixels, whereupon it becomes fixed. This enables you to cater for smaller screens, optimise a design for the most popular current screen resolution, and ensure that your design doesn't stretch too wide on large screens.

```
#wrapper {
margin: 0 auto;
```

Max-width ensures the layout is liquid until the wrapper div reaches a width of 950 pixels



Looking bad in IE6 Because Internet Explorer 6 doesn't understand the **max-width** property value in the **p** rule, the paragraphs don't stop short of the sidebar

```
border: 3px solid #6a964a;
max-width: 950px;
background-color: #ffffff;
}
```

Style structural elements

The **masthead** is styled using the following rule. It applies the background image 'masthead.jpg', defines a height for the div (which is the same as the image's height), and adds a bottom margin to make sure that subsequent content doesn't hug the masthead background image.

```
#masthead {
background: url(masthead.jpg) 0 0 no-repeat;
height: 150px;
margin-bottom: 20px;
}
```

The **content** div's **style** rule adds internal padding, so that the page's main content doesn't hug the div's borders. White space is crucial. Without it, your work may look cramped, and text that butts up against a border is hard to read. The settings in the following rule, defined in shorthand, apply no padding to the top of the div (spacing between this and the masthead having already been taken care of in the **#masthead** rule), set 20 pixels of padding at the left- and right-hand edges of the div, and 10 pixels at the bottom.

```
#content {
padding: 0 20px 10px;
}
```

The sidebar needs to be a standalone block of content that can be viewed at the same time as the main content, and yet also be distinct from it. So, via the following rule, it's floated to the right of the page, given a left-hand border (to act as a very obvious visual separator) and margins at the left and bottom edges (to make sure the sidebar doesn't hug the main content). The width setting is visually pleasing, and later you'll see that for the full-size layout – when the page is at its maximum width – this will work nicely with the **maximum-width** setting for the paragraphs.

```
.sidebar {
float: right;
width: 200px;
border-left: 1px solid #aaaaaa;
padding-left: 20px;
margin-left: 20px;
margin-bottom: 20px;
}
```





Test your liquids Always try liquid layouts in a range of browser window widths, or you may end up accidentally covering important content when a window is resized

When creating a site like this, where the width can be reduced dramatically, you need to be mindful of how the various elements will interact. As you can see if you're working through the tutorial and you now preview your web page, the masthead contains a grass-munching sheep and a title, which is more or less vertically centred. When the design is narrowed, the navigation shouldn't cross in front of the important elements of the masthead background (the sheep and the site's name). Therefore, it makes sense to style it as a horizontal bar that can slot in above the title when the browser window is narrowed (see the right-hand image, above).

```
#navigation {
font: 1.2em Arial, sans-serif;
float: right;
background: #ffffff;
margin: 10px 10px 0 0;
padding-left: 20px;
border-top: 3px solid #b8e296;
border-bottom: 3px solid #b8e296;
}
```

The previous rule floats the navigation div right, adds margins to the top and right edges, assigns a background colour and borders (to make the div stand out), and sets the font to Arial. The padding-left setting adds a gap to the

Expert tips Readable text and future-proofing

Make blocks of text distinct

When you have a text-heavy site, it's important to ensure that the eye won't get confused and jump from one block of text to another. Use font and colour variations to make each area of content distinct.

Future-proof your sites

Compatibility with old browsers is all very well, but don't compromise on compatibility with forthcoming releases. For example, many CSS hacks fail in IE7, so stick to using conditional comments to deal with the shortcomings of Microsoft's browser.



Rise above When the browser window is narrowed, as shown here, the navigation bar appears above the title and the body copy wraps underneath the sidebar

left of the first link within the navigation bar, which is necessary to do here, because other gaps will be taken care of when styling list items within the div. Following on from the previous rule, the list, list items and anchors within the navigation div must be styled. The style rule for the list itself is simple, merely removing the default bullet points by way of list-style-type: none.

```
#navigation ul {
list-style-type: none;
}
```

Next, the list items are styled. The display: inline property/value pair makes the list items display as inline elements rather than block elements, resulting in the list being displayed horizontally. The margin-right setting provides spacing between each of the list items and to the right of the final one (note that spacing to the left of the first item was set in the previous step). Finally, the line-height value is used to add some height to the list, but not in a restrictive way. Had a pixel height been defined, the navigation bar's height wouldn't change when a user altered the size of the text.

```
#navigation li {
display: inline;
margin-right: 20px;
line-height: 1.7em;
}
```

The following two rules style the links within the navigation div. The first styles the links' default state, turning off the default underline, and sets the text to bold and green (taken from the background image's grass).

```
#navigation a {
text-decoration: none;
font-weight: bold;
color: #5f7c3b;
}
```

Secondly, the underline value of text-decoration brings back the underline on the hover state, providing visual feedback that the links are clickable.



Logical layout Because of the simplicity of this website's markup, the web page remains noticeably easy to navigate and use when the CSS is disabled

```
#navigation a:hover {
text-decoration: underline;
}
```

Set up default text styles

Add the following two rules to set the font-size to 62.5 per cent. This enables text to be sized using ems, with values that are a tenth of the target pixel size (1.2em is 12px). Text sized this way can be resized using IE's built-in commands.

```
html {
font-size: 100%;
}
```

```
body {
font-size: 62.5%;
}
```

The main header is styled using the following rule. The text is bold, upper case, 1.6em and Arial, and the margin-bottom setting ensures that it looks distinct.

```
h1 {
font: bold 1.6em/100% Arial, sans-serif;
margin-bottom: 10px;
text-transform: uppercase;
}
```

Now style up the sidebar heading. The text is smaller than the main heading, the colour value is lighter than the body copy, and it's in sentence case.

```
.sidebar h1 {
font-size: 1.4em;
text-transform: none;
color: #444444;
}
```

The other headings on the web page are styled using the following rule:

```
h2 {
font: bold 1.3em/1.4em Arial, sans-serif;
```

Set the default font-size to 62.5 per cent. This enables text to be sized using ems

```
margin-top: 1.5em;
margin-bottom: 2px;
}
```

Now add the following three rules to style the paragraphs. The p rule sets the standard paragraph text to 1.1em Verdana and adds a bottom margin, while .sidebar p overrides this, making the sidebar paragraphs smaller and lighter, differentiating them from the main text. Note the max-width setting for p.

This creates a comfortable maximum width for the paragraphs, and when the browser window is close to maximised on 1,024x768 (or wider), it stops short of the sidebar, making the layout look neater.

```
p {
font: 1.1em/1.4em Verdana, Arial, sans-serif;
margin-bottom: 1em;
max-width: 60em;
}
```

```
.sidebar p {
font-size: 1.0em;
color: #444444;
}
```

Although IE7 deals with the CSS defined so far, earlier versions of Microsoft's browser fare less well. Importantly, anything below version 7 doesn't correctly deal with the max-width property. Open 'lte-ie6.css' and add this rule:

```
#wrapper {
width: expression(document.body.clientWidth>950?"950px":"auto");
}
```

This defines a width setting for IE6 and earlier, based on the width of the browser window. In plain English, the expression states: "If the browser window is wider than 950px, set the wrapper div to 950px, otherwise give it a width value of auto." Because of further bugs in IE, even this fix doesn't make the design work exactly as expected. It fails to correctly shrink when the browser window is narrowed. However, a refresh gets it all working properly.

Because the navigation div is floated right and has a margin-right value, IE6 doubles the margin size, which is corrected by adding this override:

```
#navigation {
margin-right: 5px;
}
```

The other issue in IE affects version 5.x: the site isn't centred. This is fixed by adding the two rules below.

```
body {
text-align: center;
}
```

```
#wrapper {
text-align: left;
}
```

Strictly speaking, these should be added to a style sheet for IE5.x (you duplicate the existing conditional comment in the HTML, change lte IE6 to lt IE6 and provide a new value for the href attribute). However, as the rules don't affect IE6 adversely, in this case there's no need to add another style sheet and conditional comment to the head section of the web page. ●



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