

# /CSS/flexible fixed layouts



Changes in screen size can be a nightmare for web developers, but it doesn't have to be this way. Craig Grannell reveals the art of creating fixed layouts for multiple screen sizes

**Knowledge needed** Intermediate CSS and HTML

**Requires** A text editor

**Project time** 20 minutes

Technology, as ever, marches on, and the current situation for web developers is another big screen-size changeover. Those of you who've been in the game for a while now will recall the headaches caused by the gradual shift from 640x480 to 800x600, with many sites leaving it until the last possible moment to resize. The same sort of thing's happening today, with 1,024x768 now the dominant screen resolution, but with 800x600 clinging on by its fingernails, and still accounting for anything up to a fifth of users.

In this month's tutorial, we're going to show you one method for creating a layout that works well at both of the most popular screen sizes, despite the design being a fixed width. It results in a flexible design that's suitable for many applications, including corporate sites, portfolios and blogs. Essentially, careful use of 'height' attributes and floated divs creates a web page with a fixed main content area and two columns of navigation links at the right-hand side. If the browser window width drops too much below 1,024 pixels, the navigation columns reposition themselves in a linear fashion, one beneath the other.

## Set things up

Copy the 'flexible-layout' folder from the CD to your hard drive. Open 'flexible-layout.html' and look at the 'head' section of the document. You'll see a 'style' element that imports 'columns.css' and a conditional comment for attaching the IE-specific style sheet. (See my tutorial in .net 157 for more on conditional comments.)

```
<style type="text/css" media="screen">
/*  */
@import url(columns.css);
/*  */
</style>
<!--[if lte IE 6]>
<link rel="stylesheet" type="text/css" href="ie-hacks.css" media="screen" />
<![endif]->
```

The structure of the web page is simple enough, and with all content removed, the page's 'body' looks like the following code block.

```
<div id="wrapper">
<div id="mainContentWrapper">
<div id="mainContent">
</div>
</div>
<div id="firstNavColumn">
</div>
<div id="secondNavColumn">
</div>
</div>
```

As you can see, the page's content is contained in a wrapper div, and there are three divs within it for the content blocks: mainContentWrapper, firstNavColumn and secondNavColumn. Within mainContentWrapper is a div called mainContent. Coding purists may be irked by this nesting of divs for the main content area, but it's little additional markup and provides flexibility when it comes to styling the main content area later on.

The content of the mainContent div is, in the example page, just headings, paragraphs and a list. The content of the two navigation columns is formed from blocks of content that comprise a heading, a paragraph (to introduce the links), a list of links (structured semantically using a HTML list), and a horizontal rule that serves as a simple visual and structural method for separating the blocks of navigation. The following code shows the first of the navigation content blocks:

```
<h1>:: Site sections ::</h1>
<p>Sed aliquam, nunc eget euismod ullamcorper, lectus</p>
<ul>
<li><a href="#">Home page</a></li>
```



**Completed web page** Here's the end result, as shown on a screen size of 1,024x768. As you can see, it largely fills the space, and provides immediate access to all of the navigation

## Expert tip Liquid vs fixed

The increasing popularity of monitors with screen sizes larger than 1,024x768 is used by many as an argument that you should always work with liquid web page layouts. In some cases, such layouts aren't workable – and not just for graphic

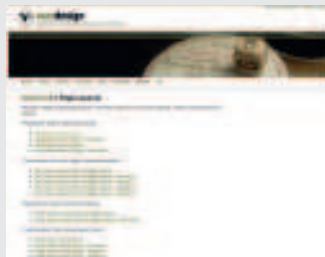
design reasons. For example, for sites that use a lot of text, liquid designs become problematic when widened, because long lines of text are hard to read (which is why magazines and newspapers tend to use fairly narrow column widths).

Resources Find out more online



[www.upsdell.com/BrowserNews/stat.htm](http://www.upsdell.com/BrowserNews/stat.htm)

The Browser News website provides a useful overview of statistics and trends related to web browser usage, including information about monitor resolution. (Such statistics should be used as a guideline only.)



[www.maxdesign.com.au/presentation/page\\_layouts/](http://www.maxdesign.com.au/presentation/page_layouts/)

If you're not familiar with the concepts behind column-based layouts built with CSS, or if you just want to brush up on your technique, the Max Design website offers a number of useful starting points.

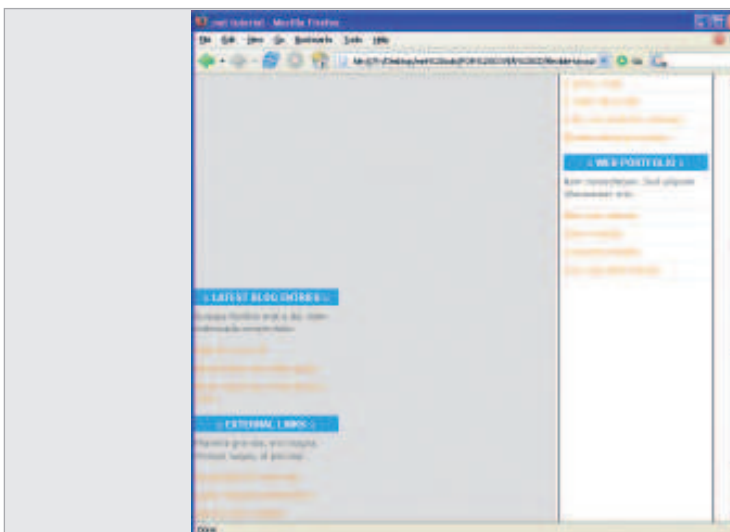
```
<li><a href="#">About the company</a></li>
<li><a href="#">Design for print</a></li>
<li><a href="#">Design for web</a></li>
<li><a href="#">Contact details</a></li>
</ul>
<hr />
```

Styling the structural elements

Open 'columns.css'. The first rule, which uses the universal selector ('\*'), zeroes margins and padding for all elements on the page. In the following code block, the first and third rules force the height of the design to the height of the browser window. The background property/value pair within the body rule adds a background colour and vertically tiled image to the page. The image itself provides a grey background for the content area, vertical separators for the columns, and a blue/green area to the right of the second column, to make the page content stand out from any background shown, should the browser window be wide.

```
html, body {
height: 100%;
}

body {
background: #005b7f url(background.gif) repeat-y;
}
```



**Misalignment** If content doesn't stretch past the first navigation column's height and its content is higher than the initial visible area, misalignment of the second column may occur



**Max design** The Floatutorial section at [css.maxdesign.com.au/floatutorial](http://css.maxdesign.com.au/floatutorial) provides a number of useful articles for newcomers who want to get to grips with the CSS float property

```
#wrapper {
height: 100%;
}
```

Next in the CSS are three rules that style the divs that house the main page content and the two sets of navigation blocks. All three divs are floated left and given set widths, which means they stack horizontally, rather than being displayed in a linear fashion. The 'margin-right' settings place space between the divs, ensuring they are correctly positioned over the background image.

```
#mainContentWrapper {
width: 500px;
min-height: 100%;
float: left;
margin-right: 11px;
}

#firstNavColumn {
float: left;
width: 200px;
margin-right: 11px;
}

#secondNavColumn {
float: left;
width: 200px;
padding-bottom: 20px;
}
```

As you can see, a 'min-height' setting is defined for #mainContentWrapper. This forces that div's height to that of the browser window, regardless of how much content is within. (Note that 'min-height' is used rather than 'height' because the latter makes strange things happen in Safari; see later for an IE-specific fix, required because that browser, pre version 7, doesn't correctly deal with 'min-height'.) The padding-bottom setting in #secondNavColumn is also important – it provides a gap under the second navigation column, should the navigation columns be displayed linearly on a small display and, in combination, be taller than the main content div. This avoids the navigation content hugging

1,024x768 is now the dominant screen resolution, but 800x600 is clinging on by its fingernails

the bottom edge of the browser window. The next rule, #mainContent, styles the div nested within #mainContentWrapper, which houses the page's main content. The margin setting provides spacing around the div, while the padding setting provides spacing within, ensuring the div's content doesn't hug its edges. Note the use of a two-pixel orange border – this makes the main content area more prominent, so it stands out from other content on the page.

```
#mainContent {
background: #ffffff;
margin: 10px;
padding: 10px;
border: 2px solid #e58200;
}
```

### Styling the navigation

The 'a' rule defines the default colour for links on the web page. The next two rules style the lists and the list items within the navigation columns. The first of those rules (see the following code block) removes the default bullet points from the lists, explicitly sets the left-hand margin to zero, and sets a dotted border along the top edge. The second rule sets a dotted bottom border on all list items and some bottom padding. These definitions result in list items having dotted lines above and below, making each individual link distinct.

```
#firstNavColumn ul, #secondNavColumn ul {
list-style-type: none;
margin-left: 0;
border-top: 1px dotted #cccccc;
}

#firstNavColumn li, #secondNavColumn li {
border-bottom: 1px dotted #cccccc;
padding-bottom: 0.3em;
}
```

The last two rules in the section, with the selectors #firstNavColumn a, #secondNavColumn a and #firstNavColumn a:hover, #secondNavColumn a:hover, turn off underlines for links in the navigation columns and turn on the underline for the hover state of those links, respectively.

### Defining font styles

In the fonts section of the CSS document, the html and body rules enable you to use ems to size fonts by using values a tenth of the target size in pixels (for example, 1.2em is equivalent to 12px). The next two rules, shown in the following code block, style the level-one headings. The first rule defines the default style, setting the headings to bold 2.0em Arial in upper case. (Remember 2.0em equates to 20px.) The second rule provides overrides for the level-one headings in the navigation columns, reducing



Vertical stacking When the browser window is narrowed (as it would be on a screen size of 800x600), the navigation columns stack vertically, but the layout remains usable



CSS disabled The site works well enough with CSS disabled, but the navigation comes after the content. If using this system on a live site, investigate creating a 'skip to navigation' link

their size, increasing their line-height setting, aligning the text centrally, adding a background, changing the text colour to white and adding a top margin. The headings in the navigation columns will have centred text on a blue background.

```
h1 {
font: bold 2.0em/1.0em Arial, sans-serif;
text-transform: uppercase;
margin-bottom: 5px;
}

#firstNavColumn h1, #secondNavColumn h1 {
font-size: 1.3em;
font-weight: normal;
line-height: 1.8em;
text-align: center;
background: #00a4e5;
margin-top: 0.8em;
color: #ffffff;
}
```

The next few rules define styles for the level-two headings (used for crossheads in the main content area), the default style for paragraphs, and an override for paragraphs in the navigation columns. Note the margin-top setting for the 'h2' rule. Because margins collapse, this effectively adds an extra 0.5em above level-two headings, rather than the gap between them and previous content being the standard 1.0em that's applied to the bottom of paragraphs and lists. Also, the 'color' setting for the navigation column paragraphs is set to a dark grey to differentiate these paragraphs from the ones in the main content area.

```
h2 {
font: bold 1.5em/1.0em Arial, sans-serif;
margin-top: 1.5em;
margin-bottom: 5px;
}

p {
color: #333333;
}
```

**Expert tip Larger screen sizes**

Although up to a fifth of web users are still, for whatever reason, clinging to an 800x600 screen size, roughly the same amount are using sizes larger than 1,024x768. Some argue that web designs should be created for every eventuality, but it would be tricky to create a liquid design fit for purpose on both 800x600 and 1,920x1,200. Ultimately, there has to be a cut-off point, although it should be noted that the majority of users with very large screens don't tend to have browser windows maximised anyway.

# Browser window resizing

## Set up your browser to change dimensions

If you're a designer with a shiny new monitor, it's often easy to forget what everyone else is lumbered with, and to design something that's unusable on smaller screen sizes. Because of this, it pays to be able to rapidly resize your browser window to common screen dimensions for testing purposes. One way of doing this is by adding 'favelets' or 'bookmarklets' to your browser's toolbar. These can be set to resize your browser window to any width and height, and several examples are available at [www.tantek.com/favelets](http://www.tantek.com/favelets). Note, however, that it's worth creating a number of alternatives, because browser

chrome varies wildly from browser to browser, and so the viewing area in one browser will be different to that in another. (Also, users often install multiple additional browser toolbars, so be mindful of them.)

The excellent Web Developer toolbar for Firefox (available from [addons.mozilla.org/firefox/60](http://addons.mozilla.org/firefox/60)) provides another means of resizing a browser window, via the suitably named Resize menu – 800x600 is a built-in size, but the Custom Size/Resize Window option (depending on which version you're using) enables you to set any dimensions for the browser window's width and height.



**Colly Logic** Simon Collison's website has a similar layout to the one shown in this tutorial, although his method uses JavaScript to deal with the positioning of the two 'moveable' columns

```
p {
font: 1.1em/1.6em Verdana, Arial, sans-serif;
margin-bottom: 1em;
}
```

```
#firstNavColumn p, #secondNavColumn p {
color: #666666;
}
```

The last three rules in the style sheet deal with styling list items and horizontal rules. The first CSS rule, 'ul', applies margins to the left and bottom of the unordered lists. The 'margin-left' setting ensures bullet points are aligned with other body copy, rather than the content of the list items being aligned with body copy, and the bullets themselves being positioned left of the main content area.

```
ul {
margin-left: 2em;
margin-bottom: 1em;
}
```

```
li {
font: 1.1em/1.6em Verdana, Arial, sans-serif;
margin-bottom: 3px;
}
```

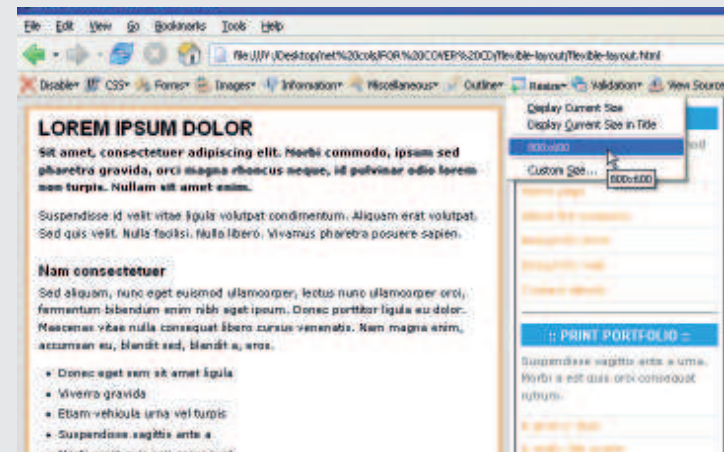
```
hr {
background-color: #666666;
color: #666666;
border: 0;
height: 1px;
}
```

It's impossible to get the horizontal rule element exactly the same across browsers. By using the settings in the previous code block – defining both background-color and color properties as the same colour, removing the border and setting height to 1px – the element at least looks similar cross-browser. The main difference is that Internet Explorer puts larger margins above and below the element.

### Hacks and caveats

From the CSS used so far, there's only one thing IE6 and earlier can't correctly deal with: the min-height setting for #mainContentWrapper. Open up 'ie-hacks.css' and you'll see it has a single rule (shown below) that deals with this problem.

```
#mainContentWrapper {
height: 100%;
}
```



**Web Developer toolbar** The astonishingly useful Web Developer toolbar for Firefox enables you to rapidly change the size of your browser window

Test the web page in a browser that has a window width greater than about 920 pixels, and you'll see a three-column layout. On the left is the main content area, within its orange border. Next is the first of the navigation columns (with three blocks of content), and to the right of that is the second navigation column (with two blocks of content). Make the browser window narrower and the second column repositions itself underneath the first, meaning the layout is suitable for 800x600 displays, too.

There is one caveat: because of the way the page is structured and styled, you have to take care with the lengths of the navigation columns. If the content in the main content area doesn't stretch past the first navigation column content's height and the content of the navigation column is higher than the initial visible area within the browser window, misalignment of the second navigation column occurs, placing it at the far left, under both the main content area and the first navigation column. But this is easy enough to avoid, and the design works well in most instances, especially for text-heavy sites such as blogs and news outlets. ●



### About the author

Name Craig Grannell  
Site [www.snubcommunications.com](http://www.snubcommunications.com)  
Areas of expertise Information architecture, site concepts, graphics, interface and front-end design  
Clients Swim~, Rebellion, IDG  
Favourite 80s band The Cure