The Personal Disquiet of

MARK BOULTON

Five simple steps to designing grid systems - Part 5

– *September 19th, 2005* –

It’s been a while, but this is the final part in my series ‘Five Simple Steps to designing Grid Systems’.

Flexible vs Fixed. Which one to choose? Why choose one over the other? Well you won’t find the answers to those questions here. What I’m aiming to do with this article is to investigate how the theory of grid design can be applied to a flexible web page.

Lets’s start by briefly examining Fixed and Flexible, or Fluid designs.

They both have their merits.

Fixed width designs are, well, just easier to produce. The designer has control over the measure, and therefore the legibility (until the user increases or decreases the font size that is).

Flexible width designs scale to the user’s resolution, and therefore the browser window. There is less empty space, typically at the side of fixed width designs.

However, they both also have the down sides such as fixed layouts generally scale badly and flexible layouts tend to look very wide and short. But, this article is not about the good and the bad and it’s not really the place for that type of discussion, that argument is going on elsewhere and will continue to do so for quite some time.

Flexible grids

As discussed the first few parts of this series, grid system design deals in fixed measurements - the media size, the type size and ultimately the grid size. They are all fixed. So, along comes the Web and challenges theory which has been around for generations. All of a sudden the reader can resize the media, they can change the font size, they can do all this stuff that designers didn't have to think about before. Designers have had to adapt, both to the technology and to the opportunities that media offers.

I've been giving flexible width a lot of thought over the past few weeks in preparation for writing this article. I can see the merits in both, but I've been trying to base my recent thought within the realms of purest grid theory. How does that theory translate to flexible grid design? and I think the answer is, quite well actually.

Adaptive Grid Systems

Ideally grid systems should be designed around the type size. Column widths, and therefore the Measure, should be constructed in such a way to maximise legibility based on the number of characters (you can read more about that in my Five Simple Steps to Better Typography). This is all fine if the units of measurements are fixed, but what if they can change? But what does that mean?

1. The user can change the font size
2. The user can resize the browser window
3. The user can change their resolution

The user can of course do this with all design, fixed or flexible, but the key to flexible grid systems is the *grid must adapt* to those changes. After a bit of thought I think the key components of an adaptive grid are:

1. The grid elements adapt to the user's changes, and
2. The grid must retain it's orginal proportions

I've never been comfortable with the desciptions of flexible grids -

flexible, elastic etc. What I'm trying to convey with *Adaptive Grid Systems or Adaptive Layouts* is the thought process behind the grid design which reacts to the *user's choices*. I think it appeals to the purist in me! ;)

It's a question of ratios. Again.

In the first two parts of this series, I talked about ratios, both rational and irrational, in the construction of grid system design. But, for those who haven't read them, or have forgotten, here's a quick overview.

Grid systems can be constructed from ratios. Simple ratios such as 1:3, or 2:1 are called *rational* ratios. More complex ratios, such as those based on the Golden Section (1: 1.618) , are called *irrational* ratios.

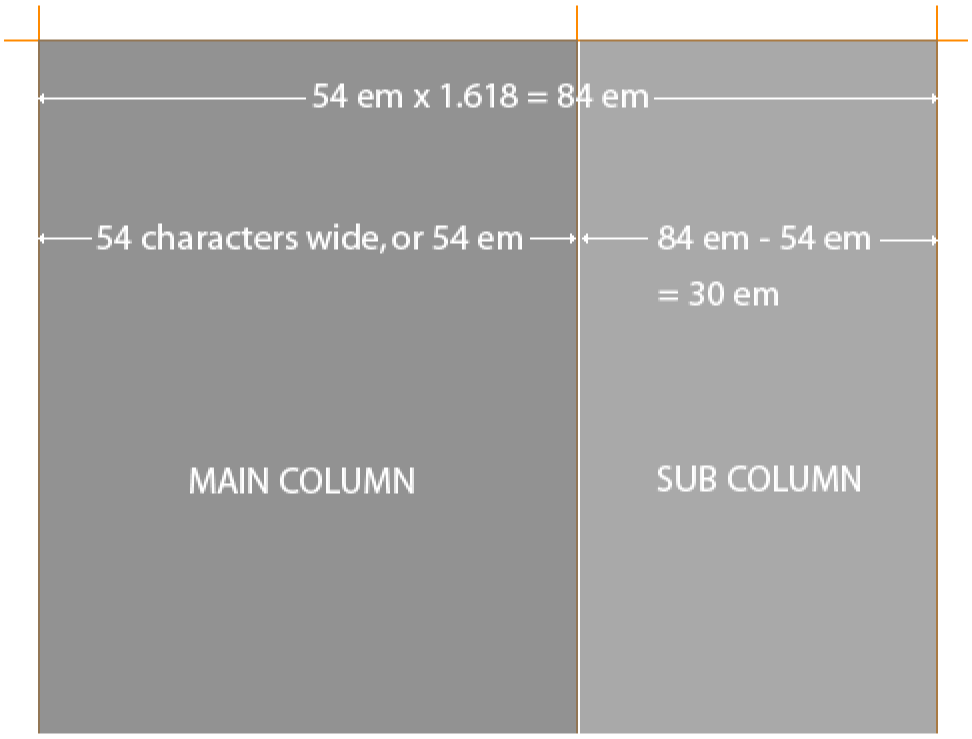
Ratios are just the job for constructing adaptive grid systems because they are independent of any unit of measurement. They are just a ratio to the whole, whatever the whole may be. This whole, be a browser window or whatever, can change and therefore so does the ratio or the grid.

Let me put this into practice now with a working grid.

Divine measurements

If you've been following these articles, you should know by now that I like to start simple. Following on from several articles I've written about the golden section, I'll continue with that and construct this adaptive grid using the Golden Section which is an irrational ratio - 1:1.618.

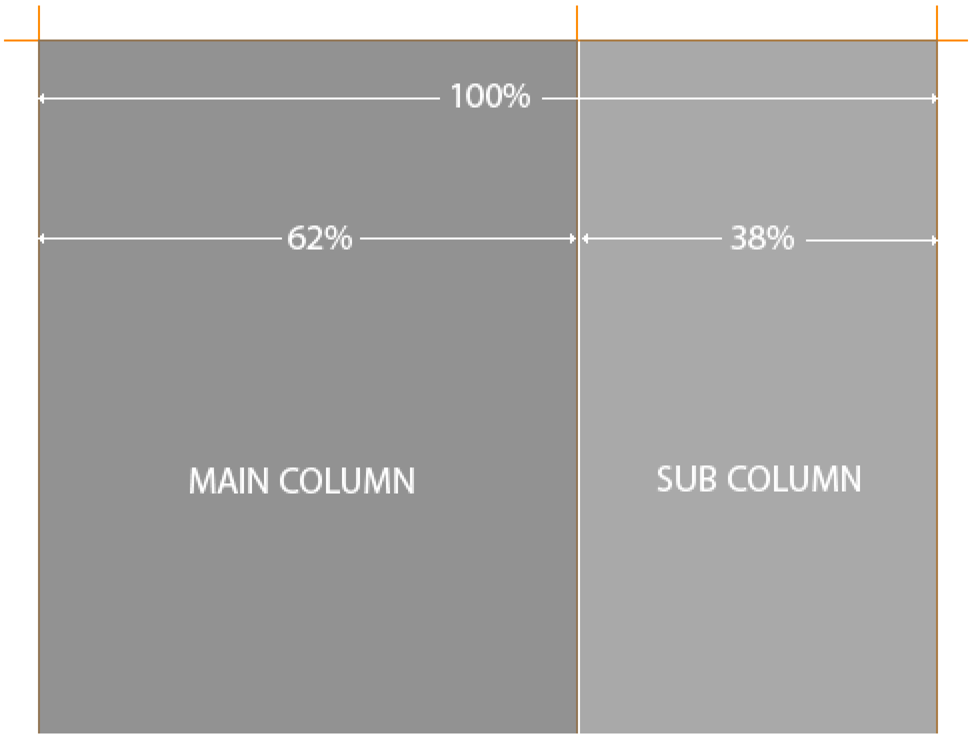
So first off we construct a simple two column grid system with the content areas being defined by the Golden Section ratio.



Getting the right units

In order for a grid to be adaptive, we have to use scalable units of measurement such as 100% or Ems. Just a reminder: An Em (pronounced ‘m’ NOT ‘e, m’) is a typographic measurement equal to the point size of the typeface you are using. We also use percentages.

To give us our column width I convert the ratio’s to percentages, which gives us 61.8% for the main column and 38.2% for the right column.



That's our grid as determined by percentages. Pretty easy really. Now, on to the implementation.

Constructing the grid in CSS

Before I move headlong into implementing this using CSS, I just wanted to point out that this tutorial is about grid design and not about web standards. XHTML and CSS just happens to be the tools I use to realise this design. You can of course use tables if you want (but you'd be wrong!). Oh, and I'm just not clever enough to stuff this example full of hacks for IE this, Mac that, weird linux browser version 0.6.1 etc. etc. This example is a Best Case Scenario example. If you want to add all that stuff to it, let me know and I'll add it. Like I said, it's the grid that interests me. Disclaimer over.

Oh, and there's a small change to the percentages of the columns to get round some browser bug or another, it's basically a couple of percentage smaller so the floats work correctly.

For those who can't be bothered going through this code, here's the example.

Here's the CSS, including all the global stuff such as links, typographic stuff and general body stuff which is applied to a pretty basic XHTML structure.

```
body {
margin: 0 auto;
padding: 0;
font-family: "Lucida Grande", verdana, arial,
helvetica, sans-serif;
font-size: 62.5%;
color: #333;
background-color: #f0f0f0;
text-align: center;
}

* {
padding: 0;
margin: 0;
}

/* Make sure the table cells show the right font
*/
td { font-family: "Lucida Grande", verdana,
arial,
helvetica, sans-serif; }

/*---------------------------------------
GLOBALS & GENERAL CASES
---------------------------------------*/

a {text-decoration: underline; padding: 1px; }
a:link { color: #03c; }
a:visited { color: #03c; }
a:hover { color: #fff; background-color: #30c;
text-decoration: none; }


/*---------------------------------------
```

```
TYPOGRAPHY
----------------------------------------*/
h1, h2, h3, h4, h5, h6 {
font-family: helvetica, arial, verdana, sans-
serif;
font-weight: normal;
}

/* approx 21px*/
h1 {
font-size: 2.1em;
margin-top: 2em;
}

/* approx 16px*/
h2 {
font-size: 1.6em;
margin-bottom: 1em;
}

/* approx 14px*/
h3 {
font-size: 1.4em;
}

/* approx 12px*/
h4 {
font-size: 1.2em;
}

/* approx 11/14 */
p {
font-size: 1.1em;
line-height: 1.4em;
padding: 0;
margin-bottom: 1em;
}
```

I then add the page structure css.

The columns are wrapped with a container div (called ‘container’), this is defined as being 90% wide, with a minimum width of 84 em. What’s that about? Well I got that number by doing this.

As mentioned earlier, our ideal minimum width for the measure is 52 em. This is the width of our main column and as determined by the Golden Section, the overall column widths combined is 1.62 multiplied by 52 em, which is 84 em. The right column is therefore 84 em minus 52 em, which is 32 em. Converting these to percentages gives us 62% and 38%, which is what you use in the CSS for each column.

There is also a minimum width of 84 em applied to the overall container, which when the user resizes the text, maintains the ratios.

Here’s the CSS for the columns:

```
#container {
width:90%;
margin:0 auto;
text-align: left;
min-width: 84em;
}

#contentframe {
margin: 2em 0 0 0;
padding: 2em 0;
width: 100%;
text-align: left;
float: left;
border: 1px solid #ccc;
background-color: #fff;
}
```

```
/* 2 column layout c1-c1-c2 */
.c1-c2 #c2 {
float: right;
width: 36.2%;
padding: 0 0 0 1em;
margin: 0;
}

.c1-c2 #c1 {
float: left;
width: 61.8%;
padding: 0 0 0 1em;
margin: 0;
}
```

So, there we have it. An Adaptive Grid System based on the Golden Section.

Please feel free to abuse the hell out of this layout. Push it, stretch it, batter it into submission. Please let me know though what your findings are. Are Adaptive Layouts the way forward in flexible grid design for the web. Like I said, I'm interested in the grid, not necessarily in the implementation.

That wraps up another series

Well, there we are. Another series finished with. Hope you liked it, and thanks for all the comments and feedback. I've got a feeling this one's going to be interesting…

This is the fifth, and final, installment of this "Simple Steps…" series.

1. Subdividing ratios
2. Ratios and complex grid systems
3. Grid systems for web design: Part 1
4. Grid systems for web design: Part 2 Fixed

5. Grid systems for web design: Part 3 Fluid

Filed in: design, typography.

Further reading

My Handbook – Environment– June 5th, 2014

Collaborative Moodboards– April 1st, 2014

Responsive Web Design – Defining The Damn Thing– March 10th, 2014

Recently: Nizlopi and the JCB SongPreviously: Guidelines

SEARCH THE SITE

Search Go

Browse the archives by date:
2014,2013, 2012, 2011, 2010, 2009, 2008, 2007, 2006, 2005, 2004, 2003

ELSEWHERE

I work for Monotype. Previously I founded Mark Boulton Design and co-founded Five Simple Steps. You can follow me on Twitter.

THANKS TO

BUY MY BOOK!

A Practical Guide to Designing for the Web Design your website using the principles of graphic design.