Joni Korpi on...

Adaptive static design

Joni Korpi explains how he built a grid system for creating multiple layouts that are as consistent as possible

Responsive web design, a method coined by Ethan Marcotte, has been getting lots of attention lately. In its simplest form, it uses a flexible grid to create a fluid layout, and media queries to spot-fix any problems that crop up when the structure gets too wide or narrow. It's great, and is really paving the way for more adaptive websites.

But responsive design is all about fluid layouts. I still prefer set widths. That's fine, but we static designers haven't quite updated our tactics to deal with the myriad screen sizes in use today: 960px isn't good enough any more.

Media queries are the key elements of adaptive websites. We could simply use them to create an optimised layout for all the most common screen sizes in use today, each with their own pixel-based grid. The result would likely be a set of layouts that would look good by themselves but that would be out of keeping with one another.

For example, say we made layouts for 320px, 768px, 1,024px and 1,280px. The optimal grids for each of these would require different column widths, which would mean slight changes to the dimensions of elements, creating the feeling that each design is slightly different from the other. This kind of deviation is bad.

Also, who really wants to construct four grids for a single website? Instead, why not create a system that makes it efficient to form multiple layouts and keeps them as consistent as possible.

Now imagine that we're designing a grid for a brand. It would have to retain its proportions despite the size of the media it's projected on: business cards, stationery, billboards and so forth. It would most likely be scaled up and down to fit, and the content inside it would simply change size with it as required.

In website layouts, however, we can't just scale whatever's in our grids whenever the screen size alters. Generally, we want all the text and element measurements to remain the same, ensuring a feeling of familiarity across each design.

Composing the grid system

We've established that column widths must remain the same across each layout. So must the gaps between columns, since they would change the dimensions of elements that span multiple columns. We're left with two things to fiddle around with: the margins around the grid and the number of columns.

I'm going to use Less Framework (lessframework.com) – a CSS blueprint I've built – as an example. In it, I use a column width of 60px, and 24px gaps between columns. I've found these dimensions to be remarkably flexible. With them, I can create layouts with three columns and 46px margins (320px), five columns and 42px margins (480px), eight columns and 60px margins (768px) and 13 columns with 72px margins. There'll still be space left over for browser chrome (1,212px), and it's easy to add an 11-column option with 42px margins (984px) if it's needed.

In a system like this, elements created for one layout are often reusable in others. For example, a block of text three columns wide will fit into any of the layouts as it is, and an image six columns wide can be scaled down to 50 per cent to slot into the three-column layout, taking advantage of the iPhone 4's double resolution.

Implementation is simple. First, pick a default layout. This will be served to all browsers that are incompatible with media queries. I use the 768px one, but 984px is a safe bet. Write the CSS for this layout normally, with no media queries applied. Then write the other layouts inside an inline media query. They'll be used by all modern desktop and mobile browsers and inherit styles from the default layout, so you won't have to write too much extra CSS; you'll just be overwriting as necessary.

Browsers that are incompatible with media queries – most notably Internet Explorers 6-8 – will ignore every style declaration inside an inline media query, using the default layout. However, you can use JavaScript to add support here. At the time of writing, **css3-mediaqueries.js** seems best.

Joni Korpi is a Finnish web designer and the author of Less Framework. *Find out more about him by visiting jonikorpi.com or following @jonikorpi on Twitter.*

"Responsive design is all about fluid layouts. I still prefer set widths" Joni Korpi