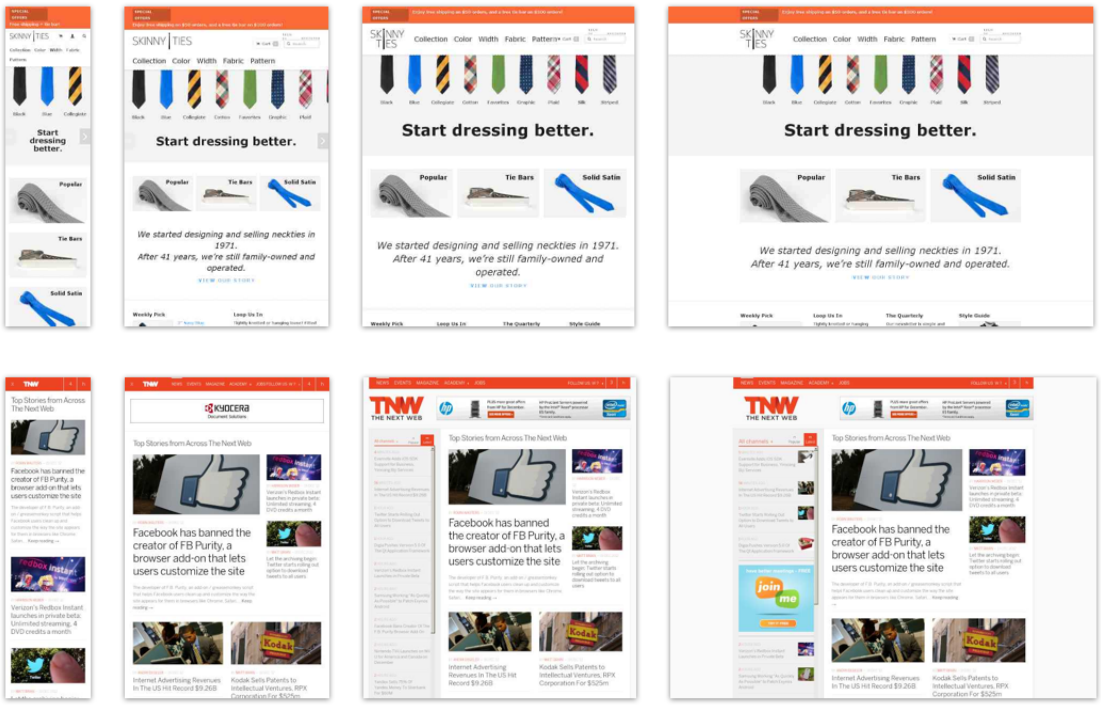
Responsive Web Design: overview

nigelbuckner 2014

Responsive web design refers to a web page adapting to the screen size of the device being used to view it, whether that device is a desktop, tablet or mobile. In other words, the page responds to the user's environment.



Responsive design illustrating how content can be re-ordered

There are two main ways to achieve responsive web design:

- Creating a separate build for mobile devices and desktops
- Using *media query* breakpoints for different size screens

Separate build

This is where there is more than one physical version of a web site, for instance, a desktop version and a *jQuery* mobile version. The separate build method requires more work. This could be used where a single set of content is not appropriate or not adaptable to differing screen sizes. Strictly speaking this method is not responsive in the true sense in that one set of content does not adapt to the viewing environment. This method uses a script, such as JavaScript or PHP, to detect the device used to access the site.

Media query

Media query is the method by which true responsive design is achieved. A media query is a statement in CSS that is called into effect by the size of the browser and then applies appropriate styles for that size. *Breakpoints* are defined screen widths such as 320, 640 and 960px. Styles are created to arrange content for display at these sizes. The styles are defined either within a single CSS document or as separate CSS documents. In either case, *changes in the layout are achieved by employing the cascade*. That is, the

same selectors are used in the CSS for each breakpoint but, through use of the cascade, the properties are adapted to alter the effect on layout.

Examples of media queries

Example 1

The following is an example of a media query that is included within the main CSS document for the page:

```
@media only screen and (max-device-width: 480px) {    }
```

The styles for the specific device width are then entered between the curly braces like so:

```
@media only screen and (max-device-width: 480px) {
        #wrapper {
                width: auto;
        }

        #header {
                background-image: url(img/image-file.jpg);
                height: 93px;
        }

        #header h1 {
                font-size: 140%;
        }

        #content {
                float: none;
                width: 100%;
        }

        #navigation {
                float:none;
                width: auto;
        }
}
```

Example 2

The following is an example of a link to a separate CSS:

```
<link rel="stylesheet" type="text/css" media="only screen and (max-device-width: 480px)"
href="small-device.css" />
```

Here, the media query is included in the link i.e. max-device-width: 480px

Example 3

Dreamweaver *CS6* uses a method that employs a CSS document as a media query, which then links to separate CSS documents for particular sizes. The statements in the media query document look like this:

```
@import url("tablet.css") only screen and (min-width:640px) and (max-width:951px);
@import url("desktop.css") only screen and (min-width:960px);
```

Designing for a responsive layout
At the design stage (prior to build), there is an accepted philosophy that says design for mobile content upward. Content at the smallest size is likely to be in one column. Content can be rearranged into further columns for tablet and desktop display.

At the build stage, create the phone version first and then create the larger versions at the required breakpoints.

In regard to sizing for main structural container elements such as <divs>, depending on the requirements of the layout, a general approach is that for desktop you can use fixed widths and for the smaller breakpoint layouts you can use percentage or *width: auto***.**

Think about making image and media content responsive in size in order to expand or contract as required. Any large background images in the desktop version, such as ones that may be used in headers, can be substituted with smaller versions from the CSS or removed completely. This will help with bandwidth on a mobile device.

In responsive design the:

- Content can be resized
- Number of columns changed
- Content re-ordered

Text sizing
An accepted good practice for text sizing is to set the default size to 100% and use *ems* for line height, e.g. body { font-size: 100%; line-height: 1.5em; }. Then, if you need to alter text in a particular region in the page use *ems*.

This approach is based upon the possibility of an end user altering the default font size. In this case, larger default text will not ‘break’ the layout. The default of 100% will be what the browser dictates and the *em* measurements work in proportion to that.

A browser default of 16px is equivalent to 1em. The line height in the example above is equivalent to 24px –
24 ÷ 16 = 1.5.
A font size of 0.75em is equivalent to 12px –
12 ÷ 16 = 0.75 *or* 0.75 x 16 = 12