**Design Skill Set: A necessity**

-the technical skills and the creative skil;

### Technical Skills

**Graphic and Web Design tools**

**-**how to use graphic software applications that are very essential for a web designer

These graphic software tools generally demonstrate the following common features making them a “must-learn” especially among graphic and web designers:

* User-friendly GUI (Graphical User Interface)
* Supports Win32/Win64 and MAC platforms
* Online support from the software developers and among online forums and blogs
* Several online tutorials on their basic and advance usage
* Intuitive features making tasks easier and less time-consuming

**Adobe Photoshop**

-best graphic manipulation tool;

Adobe Photoshop is presently in version CS5 Extended.

Among the new features that are extremely helpful for designers include:

* ***Content-aware Fill***.  Remove any object or image detail and Content-Aware Fill feature magically fills in the space left behind with an astonishing blend of existing color and texture that looks as if the removed object never really existed.
* ***3D extrusions with Adobe Repoussé***. Design 3D logos, web buttons, and artwork from any text layer, selection, path, or layer mask, and then twist, rotate, extrude, bevel, and inflate your designs.
* ***Amazing painting effects***. Achieve super natural and realistic painting effects using the Mixer Brush, which offers on-canvas color blending; Bristle Tips, which let you design lifelike, textured brush strokes; an on-screen color picker; and a lot more.
* ***Simplified complex selections***. Select intricate image objects and elements, such as hair, for refinements, compositing, or placing in layout, with no difficulty. Easily eliminate background color around selection edges, and automatically vary selection edges and perfect masks using the enhanced refinement tools.
* ***Puppet Warp***. Precisely warp or stretch graphics, text, or image elements to create unique new looks for your web designs.
* ***Enhance d HDR imaging***. Create photo-realistic or surreal HDR images with unparalleled speed, control, and accuracy. Get superior-quality results by using automatic ghost removal and greater control with tone mapping and adjustments. You can even give single-exposure photos the look of HDR.
* ***Superior 3D realism and rich materials***. Easily enhance 3D objects and designs using simple shadow creation, image-based lights, and materials like glass and chrome. Explore the unlimited design possibilities using various focus points in your 3D scene by adjusting the range for depth of field.
* ***Enhanced workflow***. Boost your creativity and productivity using dozens of features and enhancements requested by Photoshop users. Straighten images automatically, choose colors from the onscreen color picker, adjust the opacity of many layers at once, and a lot more.
* ***Enhanced media management***. Manage your media easily with more flexible batch renaming, and access your assets in the context of what you’re working on by using the customizable Adobe Mini Bridge panel.
* ***High-speed performance across platforms***. Speed up day-to-day imaging tasks and process very large images up to ten times faster by taking advantage of cross-platform 64-bit support.

**Adobe Flash**

Since Adobe Flash is a part of Adobe’s Design Suite, it is also currently running on version CS5 Extended. This version is packed with a lot more features making the Adobe Flash software a real essential web design tool. Among these features include:

* ***New text engine***
* ***Print-quality typographic control***
* ***Global language support***
* ***Advanced inline text editing***
* ***Advanced rendering support***
* ***Delivers contents to mobile and consumer devices***
* ***Delivers contents to desktop computers***
* ***XML-based FLA source files***
* ***Code snippets panel.*** Flash Professional CS5 provides a new way to quickly include code for things like timeline navigation, actions, animation, audio and video, and event handlers with the Code Snippets panel. By injecting prebuilt code into projects, you can work faster, freeing up more time for creativity.
* ***Enhanced ActionScript editor.*** Boost development speed with an improved ActionScript editor, including custom class code hinting and code completion, and reference your own code or external code libraries more efficiently.
* ***Enhanced Creative Suite integration***
* ***Flash Builder integration.*** Rather than choosing between one integrated development environment (IDE) and another, you can now write your ActionScript code in Flash Builder and then test, debug, and publish your content in Flash Professional.
* ***Video improvements.*** With integrated support for content created with Adobe Premiere® Pro or Adobe After Effects software, it’s easy to work with video content directly in Flash Professional CS5.
* ***New deco drawing tools.*** Flash Professional CS5 extends the functionality of the dynamic Deco brush tools, enabling you to easily draw shapes and add advanced animation effects.

**Adobe Illustrator**

**-**it is used by design professionals worldwide.

With Adobe Illustrator, you have countless design opportunities at hand. With the right knowledge and know-how of the software, you can:

* Design with powerful creative tools
* Work smoothly and efficiently
* Deliver virtually anywhere

Packed with the latest version of the software, Adobe Illustrator CS5, are bunch of rich features like:

* Perspective drawing
* Beautiful strokes
* Variable width strokes
* Dashed line control
* Precise arrowheads
* Stretch control for brushes
* Brushes with corner style controls
* Bristle brush
* And a lot more

**Adobe Dreamweaver**

Adobe Dreamweaver is basically a web design tool that lets you design, develop, and maintain standards-based websites and web applications.

Using the software’s user-friendly and intuitive Graphical User Interface (GUI), you can work visually or directly in code, develop with dynamic third-party application frameworks like WordPress, Joomla!, or Drupal, and troubleshoot efficiently with CSS inspection tools. You can simplify advanced website development with custom PHP code hinting. Work more securely in a team environment with enhanced support for Subversion® software.

Adobe Dreamweaver CS5, the current version of the software, gives you a bunch of rich and essential features which include:

* Support for PHP-based CMS
* Dynamically related files
* Live View navigation
* Enhanced CSS inspection tools
* PHP custom class code hinting
* Site-specific code hints
* Enhanced CSS Starter layout
* Simplified site setup
* And a lot more

o date, the most important languages that every web designer should learn about include HTML, XML, and CSS.

***HTML***  
The Hypertext Markup Language is the internet’s native language. As the name implies, it is consist of an array of markup commands that instruct the web browser how to render the online content. HTML, like other programming languages, follows coding patterns known as syntax.

Today, HTML recently came up with version 5, known as HTML5. This is basically the fifth major revision of the core language of the World Wide Web.

***CSS***  
One major breakthrough in the field of internet technology is the formulation and implementation of the Cascading Style Sheets (CSS). As the name suggests, CSS is basically a special purpose style sheet language that helps in defining the format and the actual presentation of a web content written in a markup language like HTML.

In creating a design for the web, employing CSS is basically useful especially in controlling the overall look and feel of the site. Besides, using CSS in controlling how a web browser renders your web pages gives the following benefits:

* ***Improved content accessibility***. This normally translates to a better search engine optimization since contents are laid out purely as content eliminating inline styles.
* ***Better flexibility***. Since style is entirely separated from contents, designer can definitely play with his or her imagination in outputting the pages in various ways possible.
* ***Presentation control***. With CSS, the designer can control the overall layout and presentation of the site without affecting and risking the valuable content therein.

**TOP DESIGNER MISTAKES**

## 1 – Designing Blind

What do I mean? Working without any kind (or an extremely vague) briefing from a client. I can’t stress how much of a recipe for absolute chaos this is. It’s so simple that it almost sounds stupid. Going into a project completely blind is almost certainly going to fail. You can also check [Informative Details for Organizing Your Graphic Design Process](http://www.graphicmania.net/informative-details-for-organizing-your-graphic-design-process/).

## 2 – Designing with Only You in Mind

Design is of course a deeply personal form of creative expression. But it’s important to learn quickly that each product is designed for a target audience, and letting your own preferences and thoughts clouding this process can compromise the work you’re doing. It’s fair to say that sometimes the client can let their tastes influence their judgment over what would be better for the project. Learning to see from the audience’s point of view is a hard lesson but it’s better to nail it quickly.

## 3 – Missing What Makes a Client Unique

Every client is going to have a unique selling point (UPS). These vary greatly and are a crucial element to keep in mind when designing. Failing to locate a client’s UPS is a common logo design mistake. Keeping it in mind and then subtly molding your design around it (obviously you don’t want to ram it down people’s throats) is a sure fire way to tap into the correct design.

## 4 – Ignoring Brand Positioning

[Brand positioning](http://www.graphicmania.net/how-to-build-a-corporate-identity/) is something that reaches way beyond a logo or a brand identity; it’s deeply ingrained in the genetics of a product. In essence, it’s where a brand is perceived to sit when compared to other brands selling a similar (or the same) product. For example, Mars bars are down around the low price but not low value end of the chocolate spectrum whereas Lindt chocolates are perceived (and present themselves) as high value and fairly high price, and their marketing logos reflect this. Being aware of roughly where a client’s brand sits in comparison to others around them is integral to getting the correct tone for a logo when designing.

## 5. Lacking Research

Make sure you thoroughly understand every aspect of the client you’re designing for. It’s more than likely that the business that’s hired you may not fully understand the potential of a good logo design and may have omitted information that they think unnecessary. But of course, they don’t know how to differentiate, from a design standpoint, what is and isn’t needed. You don’t want to try and relay on luck when designing, you want to fully understand what makes your design good for a client. Learning everything about them is integral to this. Check [Examples of Creative Logo Design](http://www.graphicmania.net/20-examples-of-creative-logo-design/)

## 6 – Limited Application

It’s an easy trap to fall into: only designing with one surface in mind, a website or a t-shirt for example. But what if the client wants to take your logo and put it on a host of different things? You need to be conscious of this possibility and design something that can easily be applied to various different places; a perfect example is the Nike “tick” logo.

## 7 – Too Much Choice

This is something that will evolve from experience. It’s not always for the best to present a client with a vast array of potential designs. Ultimately, they are only going to choose one from you and instead of focusing your time and energy into one or two that are exceptionally good, you’ve spread yourself thin on a host of lesser ideas. Clients will likely ask to see several examples, but remember, it’s you who’s educating them on why your design is right for their brand, not vice versa. Check [Making Sense of Clients’ Inputs](http://www.graphicmania.net/making-sense-of-clients%e2%80%99-inputs/).

## 8 – Does It Work in Its Simplest Form?

If you take away all the digital wizardry that can so easily get piled on top of a logo and you find that it’s still a good piece of work, chances are you’ve produced a strong piece of design. A weak design can be made to look stronger through the infinite possibilities of Photoshop, but in the long run this is a bad idea. You want something that stands tall in its most simple version.

## 9 – Giving a Good “Why”

It’s a hideous feeling when a client asks why something has been designed this way, and the only answer that springs to mind is “I thought it looked nice.” Bring your personal tastes and feelings into a logos explanation is a serious no-no; every element needs to be thought out and backed up with a strong “why.” Every single pixel in your design must have a concrete reason for being there.

Keeping all this in mind is a creative way to avoid some common logo design mistakes. However, there is no stronger or quicker learning curve than going out and actually learning from your own errors. Once you’ve personally fumbled through something in the wrong manner, there’s no way you’re going to do it again. Screwing up is a good thing, embrace it.

# Understanding the Website Design Trends

## Headers

This is the one of the most significant trends in web design today. Small headers with multiple click options are a thing of the past. Many designers have started making large headers that take up more than half the page. This makes the website attractive and makes a permanent impression on the visitor. This trend is so powerful that visitors tend to remember the website, just for its huge header! It also gives an opportunity for the designer to bring out his creativity.

## Fonts

During the last decade, standard sans-serif fonts like Arial and Verdana were widely used. This was due to rendering factors of Windows XP and the browsers. But now, browser technology has improved greatly and it can support all kinds of fonts. So, web designers are looking to explore how different fonts can be used to make a web page attractive to visitors. Serif fonts like Georgia and Palatino are expected to rule for the next few years.

## Illustrations

Another popular trend is the increased use of illustrations on the page. The saying – “A picture speaks a thousand words”, is the crux of this trend. They are more visual and attractive than plain text. The illustrations have to be clean and clear to impress the visitor. Sketches and hand drawn images are also a good idea because they highlight the designer’s creativity and make the web page unique. It can make a website a lot more personalized. This uniqueness is important because Internet is all set to explode in this decade with more and more third-world countries getting connected to the web. The number of blogs is also expected to rise exponentially and the only way to make a web page unique is by having your own stamp and signature.

## Minimalistic Websites

The idea of simplicity has come back, but with a twist. A web page can be simple and yet attractive. This can be achieved by the use of bold colors, layout and fonts. Minimalism does not necessary mean black lettering on white spaces. It can be spruced up with a splash of color or even a revolutionary layout. It provides an opportunity to bring in some fresh colors and ideas to make the web page pleasing to the eye.

## Modal boxes

Modal boxes are going to stay for a long time. This because of the numerous social networking sites like Facebook, Digg, Twitter and so on. As and when a user likes a post, he or she can share it with the rest of their contacts. To do so, they should be able to log into their account right from the website where the article is posted and this is done through modal boxes. They are a smoother and an improved version of the traditional pop-up windows. Though pop-up windows proved to be a frustrating design, these modal boxes are not only utility-based, but also attractive.

Also, some businesses choose to incorporate live chat in their website so that a representative can answer the questions of the user. This is also accomplished through modal boxes. Web designers have to come up with cool designs to blend the modal boxes with the overall design of the web page and at the same time, the usability aspect of these modal boxes should also be considered.

## CSS3 and HTML5

CSS3 and HTML5 are the next generation techniques in website design and development. There are a whole lot more things that one can do with these technologies and  most of the popular  browsers have the capacity to support these technologies. This combination is perfect for web design and the future is going to see more and more websites based on these two technologies.

## Introduction Boxes

It has become popular in the last couple of years to have an introduction box about the owner of the website. This is a way of introducing oneself to the visitor. It is a simple and yet effective way of telling the world a little about the owner of a blog or website. Web designers are experimenting with the placement of these introduction boxes in different parts of the page. New and innovative messages are also being used instead of the regular “Hello, I am so and so.” Lot more innovation is expected in this segment in this coming decade.

## Single Page Layout

Single page layouts make it easier to cut out unnecessary and boring information. People are in a hurry and many of them take only a quick glance at the page before moving on to the next. With the explosion of blogs and information, every visitor just wants some quick information. One page layouts are the solution to this need. These pages are more like business cards that gives a gist of what the owner is involved in and the contact details including the social media profile. This trend is more applicable for personal profiles rather than businesses.

## Conclusion

In short, the next decade is going to see some revolutionary changes in web design to suit the changing needs of users and businesses. Web designers should keep these trends in mind while designing future web sites to remain unique and competitive.

# Dos and Don’ts of Website Design

## Dos

### Maintain Consistency

The website should have the same design through all its pages. It is unprofessional to use different fonts or colors on different pages. It is confusing for the reader and gives a negative impression about the company. Every page should have the same color, font, font-size and layout. You can however change the images, content and the way it is laid out. For example, the contact us page will mostly have a form which is way different from the about us page which will contain text about the company and maybe some images. However, both the pages should contain the same font, font-size, font-color, background, header and footer.

### Viewable Resolution

The pages should have a normal resolution and it should be compatible with all browsers. The normal resolution is 1024 \* 768 and the designer should stick to this number. This is important to ensure that everyone is able to view the pages the same way. When a designer has a resolution of higher than 1280 \* 1024 his computer, he tend to design a website that matches that resolution.

### Site Map

A site map is a useful tool on any website because it gives the readers a good idea of what is contained in the pages and more importantly the exact location of the information. It is just a web page that contains text links to the content in other web pages of the site. This helps the site to better organize the information and it can be a great boon for users. Most of them will use a site map if there is one.

### Safe Colors and fonts

There are certain colors and fonts that are called safe because they can be viewed by anyone with a basic computer and a browser. While designing websites, its important you keep in mind these users and not the ones with the latest software. So, you have to use simple colors and standard fonts in your website so that it can be seen by everyone. The simpler, the better it is to suit the needs of all kinds of viewers.

### Focus on the purpose

The website should convey a clear message to the user about the business or the product and the best way to do is to focus on the core business. Design the layout and choose colors and fonts that are in tune with the business. For example, if you are designing a doctor’s website, keep the layout simple and open while for a rock band, you can use bold colors. Also, concentrate on the message that is being conveyed. The below image shows the site of WOO themes and the focus is clearly on wordpress plugins. It is obvious at the first glance and it conveys a crisp and clear message to the user.

### Simple content and layout

The content should be the mainstay of your website. Use words that are easy to read and understand without being complex. It should be appealing and readable to any user with any level of knowledge in English. When you keep it simple, the chances for it to be universally popular is high.  
Other than the content, the layout should also be simple. When you use too many boxes with too many colors, it tends to get confusing and can make the page look cluttered. So, use a simple layout that is appealing and powerful.

### Optimize loading time

The amount of time it takes for your page to load should be less. This is because your website is going to be viewed by people in all parts of the world and many of them may have a slower Internet connection, If you have too many heavy objects in your page, then it can take considerable time to load. Some people may not have the patience to wait ten or even fifteen minutes for a page to load and this can reduce the viewership.  
These are some of the do’s that you should consider while designing a website.

## Don’ts

### No page counters

Page counters are a boring feature that does not add any value to the page. Readers are not interested in knowing how many people have visited your website before. It can also mess with your design and so it is a good idea to avoid it completely.

### No Flashing text

Flashing text can take the attention of people away from the content and it is plainly irritating for most users to have neon signs on the page. They are probably used only in casinos and bars and not in websites any more. So, if you want to turn your viewer’s attention to something important, make sure you come up with some innovative ways to do it.

### No forced downloads

While designing a website, make sure your user does not need anything more to view your website. Avoid the use of any feature that requires any special browser plug-in. With CSS and HTML, you can make everything possible today and so use these technologies to the optimum extent. Also, make sure you never ask your user to download a file or an image or anything else, unless they want to.

### No Background Music

Background music is a bad idea for most sites, unless you have a music-oriented website like MTV or Channel V. Some designers tend to think that music can set the mood for the information. Unfortunately, nothing is farther from the truth because of two reasons. The first reason is music files are large and typically take a longer time to load and the second is that it can sometimes cause unnecessary noise that can be irritating to the ears. If at all you want to have music, make it an optional feature that the user can turn on or off, depending on his or her preferences. You can also create a link to a MP3 file or You tube video.

### No Pop-ups

Pop-ups are a complete turn-off for most people. It takes users away from the core content of the main page and it is a good design to exclude them. Come up with creative ways to say all that you want to your user right on the main page and avoid pop-up windows completely.

### No keyword stuffing

Keyword stuffing is a bad design idea because more often than not it backfires. It not only makes the content badly written and difficult to read, but it can also make the search engine ignore the page. The main idea of keywords is to help the search engine identify your page when a user searches for the related words. When you have too many keywords in your content, the search engine can consider it a spam and so the purpose fails. Instead, have a density of no more than 1.5% in your page and this can make it appealing to read as well as search engine friendly.

### No Unnecessary Ads

Though ads can contribute to the revenue to some extent, too many ads can be an eye-sore. The page should not contain more ads than content because people will not even bother to look at such pages. So, you will have no viewership and whole idea of a website is lost.

## Conclusion

In short, the above-mentioned points can go a long way in making your website attractive and also appealing to viewers. This simply means more revenue for the company and more business for you. So, make use of them today to build a lucrative career.

Responsive websites formula

Nothing fancy: some paragraphs set in 16px Helvetica, an unordered list that’s been slightly downsized to 14px, and an h1 at the top in 24px Georgia. Sexy, no?

What’s doubly sexy is that one simple rule allows us to [get most of this in place](http://alistapart.com/d/fluidgrids/examples/type/initial.html):

body {

font: normal 100% Helvetica, Arial, sans-serif;

}

With a font-size of 100%, all the elements in our page are sized relative to the browser’s default type size, which in most cases is 16px. And thanks to the browser’s default stylesheet, the h1 is big, bold, and beautiful—but still in Helvetica, and much too large. So while it’d be easy enough to slap on a font-family to fix the header’s Helvetica problem, how do we size the text to 24 pixels? Or accurately reduce the size of that list?

With ems, it’s easily done. We take the target value for each element’s font-size in pixels and divide it by the font-size of its container (that is, its context). We’re left with the desired font-size, expressed in relative, em-friendly terms. Or to put it more succinctly:

target ÷ context = result

If we assume the body’s default type size to be 16px, we can plug each desired font-size value into this formula. So to properly match our header to the comp, we divide the target value (24px) by the font-size of its container (16px):

24 ÷ 16 = 1.5

So the header is 1.5 times the default body size, or 1.5em, which we can plug directly into our stylesheet.

h1 {

font-family: Georgia, serif;

font-size: 1.5em; /\* 24px / 16px = 1.5em \*/

}

To size the list to the em-equivalent of 14px, we can use the same formula. Assuming again that the body’s font-size is roughly 16px, we simply divide that target by the context:

14 ÷ 16 = 0.875

And we’re left with a value of 0.875em, which we can again drop into our CSS.

ul {

font-size: 0.875em; /\* 14px / 16px = 0.875em \*/

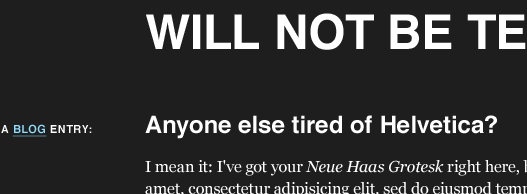
}

With those two rules, [our sample page](http://alistapart.com/d/fluidgrids/examples/type/sizes.html) is looking a lot closer to the comp, and will be practically pixel-perfect [after some slight cleanup](http://alistapart.com/d/fluidgrids/examples/type/polished.html). All with the help of our target ÷ context = result formula.

So after a few hours spent cleaning up relative type styling for our client, I realized I’d stumbled upon the answer. If we could treat font sizes not as pixels, but as *proportions* measured against their container, we could do the same with the different elements draped across our grid.

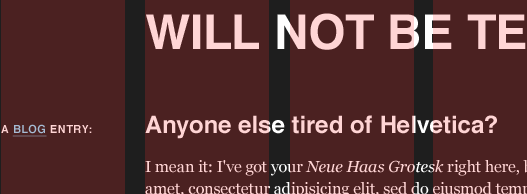
**After all, it’s not “The Golden Pixel”**

As before, let’s start with a fairly straightforward layout:

[](http://alistapart.com/d/fluidgrids/img/comp-full.gif)

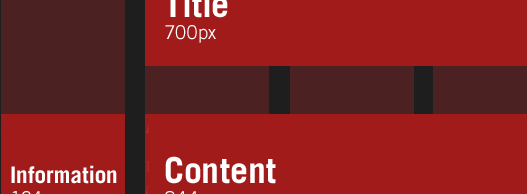
[Our basic page layout.](http://alistapart.com/d/fluidgrids/img/comp-full.gif)

Sure, [our “design”](http://alistapart.com/d/fluidgrids/img/comp-full.gif) is pretty modest. But those simple styles are draped over [a well-defined grid](http://alistapart.com/d/fluidgrids/img/comp-grid.gif): namely, seven columns of 124px each, separated by 20px-wide gutters, all of which totals up to a width of 988px. But hey, let’s forget about those nasty pixels. Proportions are the new black, right? Let’s get fluid, baby.

[](http://alistapart.com/d/fluidgrids/img/comp-grid.gif)

[Our basic page, with the grid overlaid upon it.](http://alistapart.com/d/fluidgrids/img/comp-grid.gif)

To start, let’s treat our comp like any other, fixed or fluid: before we start coding, let’s look at the design, and assess [the different content areas](http://alistapart.com/d/fluidgrids/img/comp-areas.gif). Thankfully, it’s a pretty short inventory.

[](http://alistapart.com/d/fluidgrids/img/comp-areas.gif)

[Defining our different content areas.](http://alistapart.com/d/fluidgrids/img/comp-areas.gif)

[On the highest level](http://alistapart.com/d/fluidgrids/img/comp-areas.gif), we’ve got a title at the top, a content area that spreads across six columns, and some contextual information in the leftmost column. From this diagram, we can flesh out some skeleton markup that keys into our content inventory, both structurally and semantically:

<div id="page">

<h1>The Ratio Revolution Will Not Be Televised</h1>

<div class="entry">

<h2>Anyone else tired of Helvetica?</h2>

<h3 class="info">A <a href="#">Blog</a> Entry:</h3>

<div class="content">

<div class="main">

<p>Main content goes here. Lorem ipsum etc., etc.</p>

</div><!-- /end .content -->

<div class="meta">

<p>Posted on etc., etc.</p>

</div><!-- /end .meta -->

</div><!-- /end .main -->

</div><!-- /end .entry -->

</div><!-- /end #page -->

And with some type rules applied, we’ve got [a respectable-looking starting point](http://alistapart.com/d/fluidgrids/examples/grid/initial.html). However, the #page container doesn’t have any constraints on it, so our content will simply reflow to match the width of the browser window. Let’s try to rein in those long line lengths a bit:

#page {

margin: 40px auto;

padding: 0 1em;

max-width: 61.75em; /\* 988px / 16px = 61.75em \*/

}

We’ve used [margins and padding to ventilate our design](http://alistapart.com/d/fluidgrids/examples/grid/bounded.html) a bit, and establish a gutter between it and the window edges. But in the last line of our rule, we’re using a variant of our font-size formula to define the maximum width of our design. By dividing the comp’s width of 988px by our base font-size of 16px, we can set a max-width in ems to approximate the pixel-based width from our mockup, which will prevent the page from exceeding our ideal of 988px. But since we’ve used ems to set this upper limit, the max-width will scale up as the user increases her browser’s text size—a nifty little trick that even works in older versions of Internet Explorer, if [a small CSS patch](http://www.cameronmoll.com/archives/000892.html) is applied.

So with our design properly cordoned off, let’s begin working on each element in our design inventory, beginning with the page’s title. In the comp, it spans five columns and their four gutters, with a total width of 700px. It’s also removed from the left-hand edge of the page by one column/gutter pair, making for a nice 144px offset. And if we were working in a fixed-width design, our job would be pretty straightforward:

h1 {

margin-left: 144px;

width: 700px;

}

Since we’re working in a fluid context, though, fixed measurements don’t quite cut it. And as I was working on relative font sizing, that’s when it hit me: every aspect of the grid—and the elements laid upon it—can be expressed as a proportion relative to its container. In other words, as in our type resizing exercise, we’re looking not just at the desired size of the element, but also at **the relationship of that size to the element’s container**. This will allow us to convert our design’s pixel-based widths into percentages, and keep the proportions of our grid intact as it resizes.

In short, we’ll have a fluid grid.

**Everything old is new again**

So, how do we begin?

target ÷ context = result

That’s right: it’s the return of our trusty type formula. We can use the same proportional analysis to transform pixel-based column widths into percentage-based, *flexible* measurements. So we’re working from a target value of 700px for the page’s title—but it’s contained within a designed width of 988px.



Converting our pixel-based title to percentages.

As a result, we simply divide 700px (the target) by 988px (the context) like so:

700 ÷ 988 = 0.7085

And there it is: 0.7085 translates into 70.85%, a width we can drop directly into our stylesheet:

h1 {

width: 70.85%; /\* 700px / 988px = 0.7085 \*/

}

Can we do the same with our target margin of 144px? Oh, I do so love a leading question:

144 ÷ 988 = 0.14575

Once again, we can take that 0.14575, or 14.575%, and add that directly to our style rule as a value for the title’s margin-left:

h1 {

margin-left: 14.575%; /\* 144px / 988px = 0.14575 \*/

width: 70.85%; /\* 700px / 988px = 0.7085 \*/

}

And [voilà](http://alistapart.com/d/fluidgrids/examples/grid/header.html). By measuring the title’s margin and width in relation to its container, we’ve successfully translated the ratios from our grid into CSS-friendly percentages. The title’s proportions will always remain intact, even as it reflows to fit the size of the browser window.

We can even perform the same simple division to wrap up the layout for the entry itself, sized at 844px in our comp, with some 124px-wide marginalia to the left of it. For the entry:

844 ÷ 988 = 0.85425

And for the informational column:

124 ÷ 988 = 0.12551

These two quick divisions net us some percentages that we can drop into our stylesheet, fleshing out our layout even more:

.entry h2,

.entry .content {

float: right;

width: 85.425%; /\* 844px / 988px = 0.85425 \*/

}.entry .info {

float: left;

width: 12.551%; /\* 124px / 988px = 0.12551 \*/

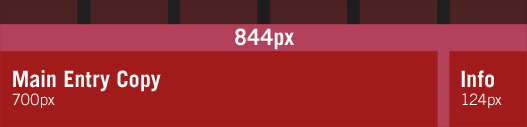
}

And with that, [our fluid grid shapes up a bit further](http://alistapart.com/d/fluidgrids/examples/grid/entry.html).

**Changing the context**

So far we’ve got the big content areas sorted, but we’ve yet to touch the inner area. Currently, the blog entry’s main copy and its contextual info occupy the full width of the entry, and are stacked on top of each other. But in [our initial comp](http://alistapart.com/d/fluidgrids/img/comp-full.gif), the main copy inside the blog entry only spanned five columns, with the ancillary info slotted neatly into the rightmost column.

Sharp readers will have noticed that, as it’s currently designed, the entry’s body is the same width as the page’s title (700px), and the marginalia is the same width as the leftmost column we styled earlier (124px). So while we’re working with some dimensions we’ve previously calculated, we can’t reuse the same formulas: the context has changed.



Since we’re working inside a new container, we need to use its width as our context.

Whereas before we were calculating percentages relative to the 988px-wide #page, we’re currently working within .entry .content, which is noticeably smaller. So as a result, we need to redefine our context, and work off the designed width of .entry .content as our reference point. So to define the percentage-based width of the main copy, we take its designed width of 700px, and divide it by 844px:

700 ÷ 844 = 0.82938

And for our 124px-wide column on the right, we can use the same reference point:

124 ÷ 844 = 0.14692

We can now take each of these measurements, and plug them into our CSS:

.entry .main {

float: left;

width: 82.938%; /\* 700px / 844px = 0.82938 \*/

}.entry .meta {

float: right;

width: 14.692%; /\* 124px / 844px = 0.14692 \*/

}

And with that we’ve finished our work, [our fluid grid complete](http://alistapart.com/d/fluidgrids/examples/grid/final.html).

**A note on rounding**

As you may guess from the lack of CSS patches above, I’ve had very few cross-browser issues with this technique.  I would highly recommend John Resig’s excellent article on [*Sub-Pixel Problems in CSS*](http://ejohn.org/blog/sub-pixel-problems-in-css/). It explains how different browsers handle percentage-based widths, and the mechanics by which they reconcile sub-pixel measurements.

As John explains in his article, if modern browsers are presented with four 25%-wide elements within a 50px-wide container, they can’t actually render the elements at 12.5px; instead, most will round the columns down or up as best fits the layout. Internet Explorer, as it happens, will simply round all of those sub-pixel values up, which breaks layouts.

If you’re working with sufficiently generous margins in your grid, this shouldn’t be an issue. But if IE causes undue wrapping with your percentage-based columns, reducing the *target* value by one pixel can help. So if, for example, our left-hand marginalia was too wide for IE (Internet Explorer), you might change your calculation from:

124 ÷ 988 = 0.12551

to a lower target of 123px:

123 ÷ 988 = 0.12449

Plug that width of 12.449% into your IE-specific stylesheet, and your layout woes should clear right up.

**A grid for all seasons**

And finally, I don’t pretend that design is easy, whether it’s fixed or fluid. But given all that we’ve achieved over the past few years—moving past tables, evangelizing standards in our companies and in our shared industry, demanding better standards of our browsers and our peers—I do wish we’d bend some of that ingenuity to break out of our reliance on “minimum screen resolution.” After all, our users’ browsing habits aren’t as fixed as our comps would suggest. I hope the promise of fluid grids has fired your imagination, and I’m excited to see how you improve on the technique. Our users will be, too.

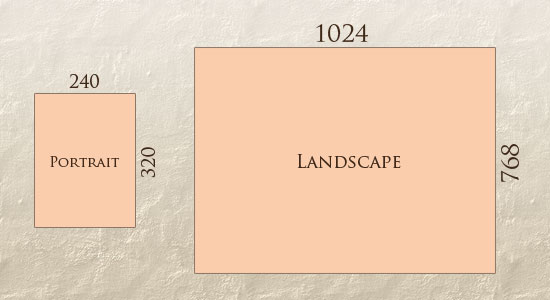
**Additional reading**

As you may have gathered from my introductory digression, two passions of mine are fluid web design and, more recently, the importance of a well-considered grid. Both of these have been fueled by the following, though this isn’t an exhaustive list:

* John Allsopp, [*A Dao of Web Design*](http://www.alistapart.com/articles/dao/)
* Mark Boulton, [*Feeling your way around grids*](http://www.markboulton.co.uk/)
* David Emery, [*More Width*](http://de-online.co.uk/2006/11/27/more-width)
* Molly Holzschlag, [*Thinking Outside the Grid*](http://www.alistapart.com/articles/outsidethegrid)
* Jeremy Keith, [*The unpushed envelope*](http://adactio.com/journal/1149)
* Jeffrey Zeldman, [*Rules-based design*](http://www.zeldman.com/daily/0403b.shtml#ap3003)

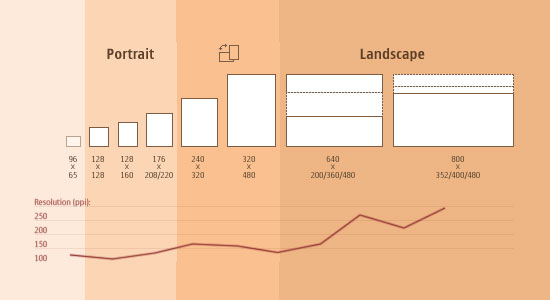
### Adjusting Screen Resolution [Link](http://www.smashingmagazine.com/2011/01/guidelines-for-responsive-web-design/#adjusting-screen-resolution)

With more devices come varying screen resolutions, definitions and orientations. New devices with new screen sizes are being developed every day, and each of these devices may be able to handle variations in size, functionality and even color. Some are in landscape, others in portrait, still others even completely square. As we know from the rising popularity of the iPhone, iPad and advanced smartphones, many new devices are able to switch from portrait to landscape at the user’s whim. How is one to design for these situations?



In addition to designing for both landscape and portrait (and enabling those orientations to possibly switch in an instant upon page load), we must consider the hundreds of different screen sizes. Yes, it is possible to group them into major categories, design for each of them, and make each design as flexible as necessary. But that can be overwhelming, and who knows what the usage figures will be in five years? Besides, [many users do not maximize their browsers](http://www.456bereastreet.com/archive/200704/poll_results_504_of_respondents_maximise_windows/), which itself leaves far too much room for variety among screen sizes.

Morten Hjerde and a few of his colleagues [identified statistics on about 400 devices](http://sender11.typepad.com/sender11/2008/04/mobile-screen-s.html) sold between 2005 and 2008. Below are some of the most common:

[](http://sender11.typepad.com/sender11/2008/04/mobile-screen-s.html)

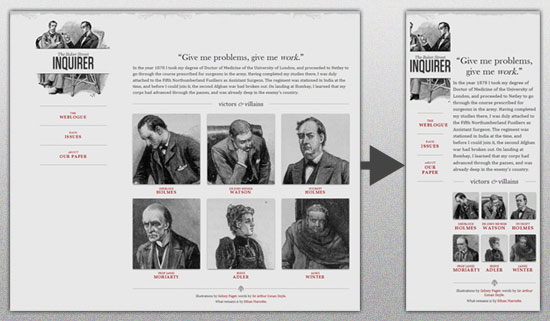
Since then even [more devices have come out](http://www.quirksmode.org/mobile/mobilemarket.html). It’s obvious that we can’t keep creating custom solutions for each one. So, how do we deal with the situation?

#### Part of the Solution: Flexible Everything [Link](http://www.smashingmagazine.com/2011/01/guidelines-for-responsive-web-design/#part-of-the-solution-flexible-everything)

A few years ago, when flexible layouts were almost a “luxury” for websites, the only things that were flexible in a design were the layout columns (structural elements) and the text. Images could easily break layouts, and even flexible structural elements broke a layout’s form when pushed enough. Flexible designs weren’t really that flexible; they could give or take a few hundred pixels, but they often couldn’t adjust from a large computer screen to a netbook.

Now we can make things more flexible. Images can be automatically adjusted, and we have workarounds so that layouts never break (although they may become squished and illegible in the process). While it’s not a complete fix, the solution gives us far more options. It’s perfect for devices that switch from portrait orientation to landscape in an instant or for when users switch from a large computer screen to an iPad.

In Ethan Marcotte’s article, he created a sample Web design that features this better flexible layout:

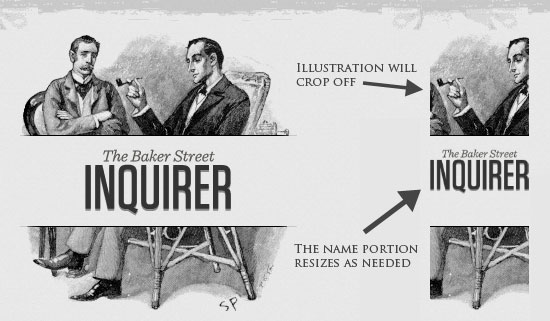
[](http://www.alistapart.com/d/responsive-web-design/ex/ex-site-flexible.html)

The entire design is a lovely mix of [fluid grids](http://www.alistapart.com/articles/fluidgrids/), [fluid images](http://unstoppablerobotninja.com/entry/fluid-images) and smart mark-up where needed. Creating fluid grids is fairly common practice, and there are a number of techniques for creating fluid images:

* [Hiding and Revealing Portions of Images](http://zomigi.com/blog/hiding-and-revealing-portions-of-images/)
* [Creating Sliding Composite Images](http://zomigi.com/blog/creating-sliding-composite-images/)
* [Foreground Images That Scale With the Layout](http://zomigi.com/blog/foreground-images-that-scale-with-the-layout/)

For more information on creating fluid websites, be sure to look at the book “Flexible Web Design: Creating Liquid and Elastic Layouts with CSS” by Zoe Mickley Gillenwater, and download the sample chapter “[Creating Flexible Images](http://www.flexiblewebbook.com/bonus.html).” In addition, Zoe provides the following extensive list of tutorials, resources, inspiration and best practices on creating flexible grids and layouts: “[Essential Resources for Creating Liquid and Elastic Layouts](http://zomigi.com/blog/essential-resources-for-creating-liquid-and-elastic-layouts/).”

While from a technical perspective this is all easily possible, it’s not just about plugging these features in and being done. Look at the logo in this design, for example:

[](http://www.alistapart.com/d/responsive-web-design/ex/ex-site-flexible.html)

If resized too small, the image would appear to be of low quality, but keeping the name of the website visible and not cropping it off was important. So, the image is divided into two: one (of the illustration) set as a background, to be cropped and to maintain its size, and the other (of the name) resized proportionally.

<h1 id="logo"><a href="#"><img src="site/logo.png" alt="The Baker Street Inquirer" /></a></h1>

Above, the h1 element holds the illustration as a background, and the image is aligned according to the container’s background (the heading).

This is just one example of the kind of thinking that makes responsive Web design truly effective. But even with smart fixes like this, a layout can become too narrow or short to look right. In the logo example above (although it works), the ideal situation would be to not crop half of the illustration or to keep the logo from being so small that it becomes illegible and “floats” up.

### Flexible Images [Link](http://www.smashingmagazine.com/2011/01/guidelines-for-responsive-web-design/#flexible-images)

One major problem that needs to be solved with responsive Web design is working with images. There are a number of techniques to resize images proportionately, and many are easily done. The most popular option, noted in Ethan Marcotte’s article on [fluid images](http://unstoppablerobotninja.com/entry/fluid-images/) but first experimented with by [Richard Rutter](http://clagnut.com/sandbox/imagetest3/), is to use CSS’s max-width for an easy fix.

img { max-width: 100%; }

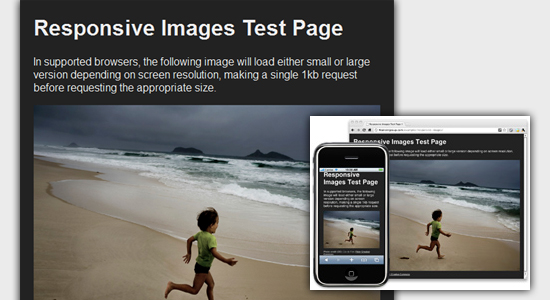
As long as no other width-based image styles override this rule, every image will load in its original size, unless the viewing area becomes narrower than the image’s original width. The **maximum width** of the image is set to 100% of the screen or browser width, so when that 100% becomes narrower, so does the image. Essentially, as Jason Grigsby [noted](http://www.cloudfour.com/css-media-query-for-mobile-is-fools-gold/), “The idea behind fluid images is that you deliver images at the maximum size they will be used at. You don’t declare the height and width in your code, but instead let the browser resize the images as needed while using CSS to guide their relative size”. It’s a great and simple technique to resize images beautifully.

Note that max-width is **not supported in IE**, but a good use of width: 100% would solve the problem neatly in an IE-specific style sheet. One more issue is that when an image is resized too small in some older browsers in Windows, the rendering isn’t as clear as it ought to be. There is a JavaScript to fix this issue, though, found in [Ethan Marcotte’s article](http://unstoppablerobotninja.com/entry/fluid-images/).

While the above is a great quick fix and good start to responsive images, image resolution and download times should be the primary considerations. While resizing an image for mobile devices can be very simple, if the original image size is meant for large devices, it could significantly slow download times and take up space unnecessarily.

#### Filament Group’s Responsive Images [Link](http://www.smashingmagazine.com/2011/01/guidelines-for-responsive-web-design/#filament-groups-responsive-images)

This technique, presented by the Filament Group, takes this issue into consideration and not only resizes images proportionately, but shrinks image resolution on smaller devices, so very large images don’t waste space unnecessarily on small screens. Check out [the demo page here.](http://filamentgroup.com/examples/responsive-images/)

[](http://filamentgroup.com/lab/responsive_images_experimenting_with_context_aware_image_sizing/)

This technique requires a few files, all of which are available on [Github](https://github.com/filamentgroup/Responsive-Images). First, a JavaScript file (rwd-images.js), the .htaccess file and an image file (rwd.gif). Then, we can use just a bit of HTML to reference both the larger and smaller resolution images: first, the small image, with an .r prefix to clarify that it should be responsive, and then a reference to the bigger image using data-fullsrc.

<img src="smallRes.jpg" data-fullsrc="largeRes.jpg">

The data-fullsrc is a custom HTML5 attribute, defined in the files linked to above. For any screen that is wider than 480 pixels, the larger-resolution image (largeRes.jpg) will load; smaller screens wouldn’t need to load the bigger image, and so the smaller image (smallRes.jpg) will load.

The JavaScript file inserts a base element that allows the page to separate responsive images from others and redirects them as necessary. When the page loads, all files are rewritten to their original forms, and only the large or small images are loaded as necessary. With other techniques, all higher-resolution images would have had to be downloaded, even if the larger versions would never be used. Particularly for websites with a lot of images, this technique can be a great saver of bandwidth and loading time.

This technique is fully supported in modern browsers, such as **IE8+, Safari, Chrome and Opera, as well as mobile devices that use these same browsers** (iPad, iPhone, etc.). Older browsers and Firefox degrade nicely and still resize as one would expect of a responsive image, except that both resolutions are downloaded together, so the end benefit of saving space with this technique is void.

#### Stop iPhone Simulator Image Resizing [Link](http://www.smashingmagazine.com/2011/01/guidelines-for-responsive-web-design/#stop-iphone-simulator-image-resizing)

One nice thing about the iPhone and iPod Touch is that Web designs automatically rescale to fit the tiny screen. A full-sized design, unless specified otherwise, would just shrink proportionally for the tiny browser, with no need for scrolling or a mobile version. Then, the user could easily zoom in and out as necessary.

There was, however, one issue this simulator created. When responsive Web design took off, many noticed that images were still changing proportionally with the page even if they were specifically made for (or could otherwise fit) the tiny screen. This in turn scaled down text and other elements.

[](http://thinkvitamin.com/design/responsive-design-image-gotcha/)  
(Image: [*Think Vitamin*](http://thinkvitamin.com/design/responsive-design-image-gotcha/) | Website referenced: [*8 Faces*](http://8faces.com/))

Because this works only with Apple’s simulator, we can use an Apple-specific meta tag to fix the problem, placing it below the website’s <head> section. Thanks to [Think Vitamin’s article on image resizing](http://thinkvitamin.com/design/responsive-design-image-gotcha/), we have the meta tag below:

<meta name="viewport" content="width=device-width; initial-scale=1.0">

Setting the initial-scale to 1 overrides the default to resize images proportionally, while leaving them as is if their width is the same as the device’s width (in either portrait or lanscape mode). Apple’s documentation has a lot more information on the [viewport meta tag](http://developer.apple.com/library/safari/#documentation/appleapplications/reference/safarihtmlref/Articles/MetaTags.html).

### Custom Layout Structure [Link](http://www.smashingmagazine.com/2011/01/guidelines-for-responsive-web-design/#custom-layout-structure)

For extreme size changes, we may want to change the layout altogether, either through a separate style sheet or, more efficiently, through a CSS media query. This does not have to be troublesome; most of the styles can remain the same, while specific style sheets can inherit these styles and move elements around with floats, widths, heights and so on.

For example, we could have one main style sheet (which would also be the default) that would define all of the main structural elements, such as #wrapper, #content, #sidebar, #nav, along with colors, backgrounds and typography. Default flexible widths and floats could also be defined.

If a style sheet made the layout too narrow, short, wide or tall, we could then detect that and switch to a new style sheet. This new child style sheet would adopt everything from the default style sheet and then just redefine the layout’s structure.

Here is the **style.css (default) content:**

/\* Default styles that will carry to the child style sheet \*/

html,body{

background...

font...

color...

}

h1,h2,h3{}

p, blockquote, pre, code, ol, ul{}

/\* Structural elements \*/

#wrapper{

width: 80%;

margin: 0 auto;

background: #fff;

padding: 20px;

}

#content{

width: 54%;

float: left;

margin-right: 3%;

}

#sidebar-left{

width: 20%;

float: left;

margin-right: 3%;

}

#sidebar-right{

width: 20%;

float: left;

}

Here is the **mobile.css (child)** content:

#wrapper{

width: 90%;

}

#content{

width: 100%;

}

#sidebar-left{

width: 100%;

clear: both;

/\* Additional styling for our new layout \*/

border-top: 1px solid #ccc;

margin-top: 20px;

}

#sidebar-right{

width: 100%;

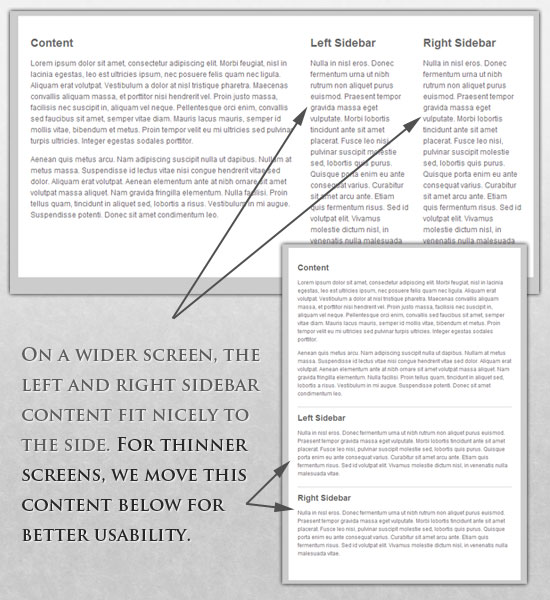
clear: both;

/\* Additional styling for our new layout \*/

border-top: 1px solid #ccc;

margin-top: 20px;

}



#### Media Queries [Link](http://www.smashingmagazine.com/2011/01/guidelines-for-responsive-web-design/#media-queries)

CSS3 supports all of the same media types as CSS 2.1, such as screen, print and handheld, but has added dozens of new media features, including max-width, device-width, orientation and color. New devices made after the release of CSS3 (such as the iPad and Android devices) will definitely support media features. So, calling a media query using CSS3 features to target these devices would work just fine, and it will be ignored if accessed by an older computer browser that does not support CSS3.

In Ethan Marcotte’s article, we see an example of a media query in action:

<link rel="stylesheet" type="text/css"

media="screen and (max-device-width: 480px)"

href="shetland.css" />

This media query is fairly self-explanatory: if the browser displays this page on a screen (rather than print, etc.), and if the width of the screen (not necessarily the viewport) is 480 pixels or less, then load shetland.css.

New CSS3 features also include orientation (portrait vs. landscape), device-width, min-device-width and more. Look at “[The Orientation Media Query](http://www.quirksmode.org/blog/archives/2010/04/the_orientation.html)” for more information on setting and restricting widths based on these media query features.

One can create multiple style sheets, as well as basic layout alterations defined to fit ranges of widths — even for landscape vs. portrait orientations. Be sure to look at the section of Ethan Marcotte’s article entitled “[Meet the media query](http://www.alistapart.com/articles/responsive-web-design/)” for more examples and a more thorough explanation.

Multiple media queries can also be dropped right into a single style sheet, which is the most efficient option when used:

/\* Smartphones (portrait and landscape) ----------- \*/

@media only screen

and (min-device-width : 320px)

and (max-device-width : 480px) {

/\* Styles \*/

}

/\* Smartphones (landscape) ----------- \*/

@media only screen

and (min-width : 321px) {

/\* Styles \*/

}

/\* Smartphones (portrait) ----------- \*/

@media only screen

and (max-width : 320px) {

/\* Styles \*/

}

The code above is from a free template for multiple media queries between popular devices by Andy Clark. See the differences between this approach and including different style sheet files in the mark-up as shown in the post “[Hardboiled CSS3 Media Queries](http://stuffandnonsense.co.uk/blog/about/hardboiled_css3_media_queries).”

#### CSS3 Media Queries [Link](http://www.smashingmagazine.com/2011/01/guidelines-for-responsive-web-design/#css3-media-queries)

Above are a few examples of how media queries, both from CSS 2.1 and CSS3 could work. Let’s now look at some specific how-to’s for using CSS3 media queries to create responsive Web designs. Many of these uses are relevant today, and all will definitely be usable in the near future.

The **min-width and max-width** properties do exactly what they suggest. The min-width property sets a minimum browser or screen width that a certain set of styles (or separate style sheet) would apply to. If anything is below this limit, the style sheet link or styles will be ignored. The max-width property does just the opposite. Anything above the maximum browser or screen width specified would not apply to the respective media query.

Note in the examples below that we’re using the syntax for media queries that could be used all in one style sheet. As mentioned above, the most efficient way to use media queries is to place them all in one CSS style sheet, with the rest of the styles for the website. This way, multiple requests don’t have to be made for multiple style sheets.

@media screen and (min-width: 600px) {

.hereIsMyClass {

width: 30%;

float: right;

}

}

The class specified in the media query above (hereIsMyClass) will work only if the browser or screen width is above 600 pixels. In other words, this media query will run only if the **minimum width is 600 pixels** (therefore, 600 pixels or wider).

@media screen and (max-width: 600px) {

.aClassforSmallScreens {

clear: both;

font-size: 1.3em;

}

}

Now, with the use of max-width, this media query will apply only to browser or screen widths with a maximum width of 600 pixels or narrower.

While the above min-width and max-width can apply to either screen size or browser width, sometimes we’d like a media query that is relevant to device width specifically. This means that even if a browser or other viewing area is minimized to something smaller, the media query would still apply to the size of the actual device. The **min-device-width and max-device-width** media query properties are great for targeting certain devices with set dimensions, without applying the same styles to other screen sizes in a browser that mimics the device’s size.

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

.classForiPhoneDisplay {

font-size: 1.2em;

}

}

@media screen and (min-device-width: 768px) {

.minimumiPadWidth {

clear: both;

margin-bottom: 2px solid #ccc;

}

}

There are also other tricks with media queries to target specific devices. Thomas Maier has written two short snippets and explanations for targeting the iPhone and iPad only:

* [CSS for iPhone 4 (Retina display)](http://thomasmaier.me/2010/06/css-for-iphone-4-retina-display/)
* [How To: CSS for the iPad](http://thomasmaier.me/2010/03/howto-css-for-the-ipad/)

For the iPad specifically, there is also a media query property called **orientation**. The value can be either landscape (horizontal orientation) or portrait (vertical orientation).

@media screen and (orientation: landscape) {

.iPadLandscape {

width: 30%;

float: right;

}

}

@media screen and (orientation: portrait) {

.iPadPortrait {

clear: both;

}

}

Unfortunately, this property works only on the iPad. When [determining the orientation for the iPhone](http://www.thecssninja.com/css/iphone-orientation-css) and other devices, the use of max-device-width and min-device-width should do the trick.

There are also many media queries that **make sense when combined**. For example, the min-width and max-width media queries are combined all the time to set a style specific to a certain range.

@media screen and (min-width: 800px) and (max-width: 1200px) {

.classForaMediumScreen {

background: #cc0000;

width: 30%;

float: right;

}

}

The above code in this media query applies only to screen and browser widths between 800 and 1200 pixels. A good use of this technique is to show certain content or entire sidebars in a layout depending on how much horizontal space is available.

Some designers would also prefer to **link to a separate style sheet** for certain media queries, which is perfectly fine if the organizational benefits outweigh the efficiency lost. For devices that do not switch orientation or for screens whose browser width cannot be changed manually, using a separate style sheet should be fine.

You might want, for example, to place media queries all in one style sheet (as above) for devices like the iPad. Because such a device can switch from portrait to landscape in an instant, if these two media queries were placed in separate style sheets, the website would have to call each style sheet file every time the user switched orientations. Placing a media query for both the horizontal and vertical orientations of the iPad in the same style sheet file would be far more efficient.

Another example is a flexible design meant for a standard computer screen with a resizable browser. If the browser can be manually resized, placing all variable media queries in one style sheet would be best.

Nevertheless, organization can be key, and a designer may wish to define media queries in a standard HTML link tag:

<link rel="stylesheet" media="screen and (max-width: 600px)" href="small.css" />

<link rel="stylesheet" media="screen and (min-width: 600px)" href="large.css" />

<link rel="stylesheet" media="print" href="print.css" />

#### JavaScript [Link](http://www.smashingmagazine.com/2011/01/guidelines-for-responsive-web-design/#javascript)

Another method that can be used is JavaScript, especially as a back-up to devices that don’t support all of the CSS3 media query options. Fortunately, there is already a pre-made JavaScript library that makes older browsers (IE 5+, Firefox 1+, Safari 2) support CSS3 media queries. If you’re already using these queries, just grab a copy of the library, and include it in the mark-up: [css3-mediaqueries.js](http://code.google.com/p/css3-mediaqueries-js/).

In addition, below is a sample jQuery snippet that detects browser width and changes the style sheet accordingly — if one prefers a more hands-on approach:

<script type="text/javascript" src="http://ajax.googleapis.com/ajax/libs/jquery/1.4.4/jquery.min.js"></script>

<script type="text/javascript">

$(document).ready(function(){

$(window).bind("resize", resizeWindow);

function resizeWindow(e){

var newWindowWidth = $(window).width();

// If width width is below 600px, switch to the mobile stylesheet

if(newWindowWidth < 600){ $("link[rel=stylesheet]").attr({href : "mobile.css"}); } // Else if width is above 600px, switch to the large stylesheet else if(newWindowWidth > 600){

$("link[rel=stylesheet]").attr({href : "style.css"});

}

}

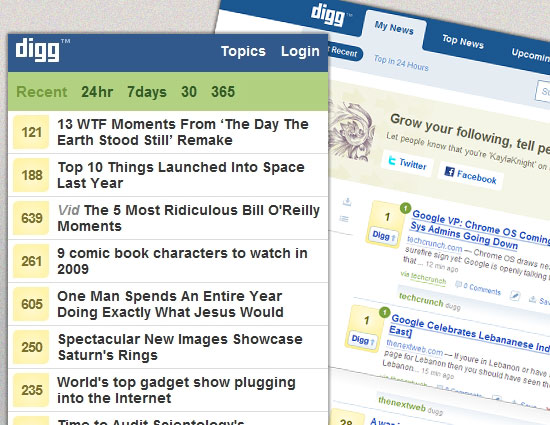
});

</script>

There are many solutions for pairing up JavaScript with CSS media queries. Remember that media queries are not an absolute answer, but rather are fantastic options for responsive Web design when it comes to pure CSS-based solutions. With the addition of JavaScript, we can accomodate far more variations. For detailed information on using JavaScript to mimic or work with media queries, look at “[Combining Media Queries and JavaScript](http://www.quirksmode.org/blog/archives/2010/08/combining_media.html).”

### Showing or Hiding Content [Link](http://www.smashingmagazine.com/2011/01/guidelines-for-responsive-web-design/#showing-or-hiding-content)

It is possible to shrink things proportionally and rearrange elements as necessary to make everything fit (reasonably well) as a screen gets smaller. It’s great that that’s possible, but making every piece of content from a large screen available on a smaller screen or mobile device isn’t always the best answer. We have best practices for mobile environments: simpler navigation, more focused content, lists or rows instead of multiple columns.

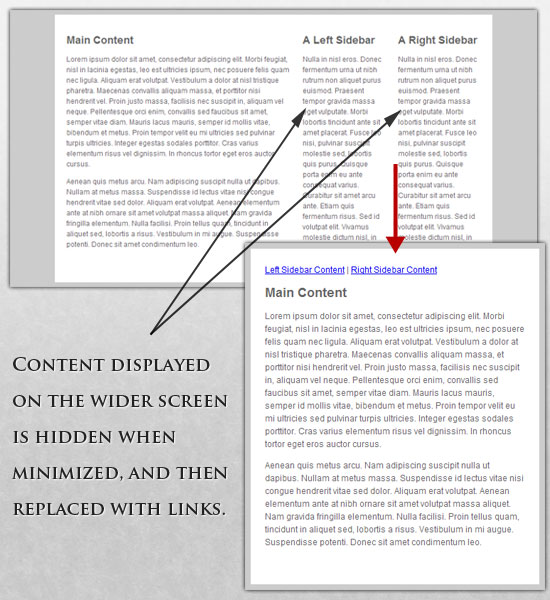
[](http://digg.com/)

Responsive Web design shouldn’t be just about how to create a flexible layout on a wide range of platforms and screen sizes. It should also be about the user being able to pick and choose content. Fortunately, CSS has been allowing us to show and hide content with ease for years!

display: none;

Either declare display: none for the HTML block element that needs to be hidden in a specific style sheet or detect the browser width and do it through JavaScript. In addition to hiding content on smaller screens, we can also hide content in our default style sheet (for bigger screens) that should be available only in mobile versions or on smaller devices. For example, as we hide major pieces of content, we could replace them with navigation to that content, or with a different navigation structure altogether.

Note that we haven’t used visibility: hidden here; this just hides the content (although it is still there), whereas the display property gets rid of it altogether. For smaller devices, there is no need to keep the mark-up on the page — it just takes up resources and might even cause unnecessary scrolling or break the layout.



Here is **our mark-up**:

<p class="sidebar-nav"><a href="#">Left Sidebar Content</a> | <a href="#">Right Sidebar Content</a></p>

<div id="content">

<h2>Main Content</h2>

</div>

<div id="sidebar-left">

<h2>A Left Sidebar</h2>

</div>

<div id="sidebar-right">

<h2>A Right Sidebar</h2>

</div>

In our default style sheet below, we have hidden the links to the sidebar content. Because our screen is large enough, we can display this content on page load.

Here is the **style.css (default)** content:

#content{

width: 54%;

float: left;

margin-right: 3%;

}

#sidebar-left{

width: 20%;

float: left;

margin-right: 3%;

}

#sidebar-right{

width: 20%;

float: left;

}

.sidebar-nav{display: none;}

Now, we hide the two sidebars (below) and show the links to these pieces of content. As an alternative, the links could call to JavaScript to just cancel out the display: none when clicked, and the sidebars could be realigned in the CSS to float below the content (or in another reasonable way).

Here is the **mobile.css (simpler)** content:

#content{

width: 100%;

}

#sidebar-left{

display: none;

}

#sidebar-right{

display: none;

}

.sidebar-nav{display: inline;}

With the ability to easily show and hide content, rearrange layout elements and automatically resize images, form elements and more, a design can be transformed to fit a huge variety of screen sizes and device types. As the screen gets smaller, rearrange elements to fit mobile guidelines; for example, use a script or alternate style sheet to increase white space or to replace image navigation sources on mobile devices for better usability (icons would be more beneficial on smaller screens).

Below are a couple of relevant resources:

* [Mobile Web Design Trends For 2009](http://www.smashingmagazine.com/2009/01/13/mobile-web-design-trends-2009/)
* [7 Usability Guidelines for Websites on Mobile Devices](http://www.webcredible.co.uk/user-friendly-resources/web-usability/mobile-guidelines.shtml)

#### Touchscreens vs. Cursors [Link](http://www.smashingmagazine.com/2011/01/guidelines-for-responsive-web-design/#touchscreens-vs-cursors)

Touchscreens are becoming increasingly popular. Assuming that smaller devices are more likely to be given touchscreen functionality is easy, but don’t be so quick. Right now touchscreens are mainly on smaller devices, but many laptops and desktops on the market also have touchscreen capability. For example, the [HP Touchsmart tm2t](http://www.shopping.hp.com/webapp/shopping/store_access.do?template_type=series_detail&category=notebooks&series_name=tm2t_series&aoid=51320&keyword=hp+touchsmart+tm2&tafcjnef=fy10&DS_KWID=p117477087) is a basic touchscreen laptop with traditional keyboard and mouse that can transform into a tablet.

[](http://www.flickr.com/photos/rrrrred/5134202846/)

Touchscreens obviously come with different design guidelines than purely cursor-based interaction, and the two have different capabilities as well. Fortunately, making a design work for both doesn’t take a lot of effort. Touchscreens have no capability to display CSS hovers because there is no cursor; once the user touches the screen, they click. So, don’t rely on CSS hovers for link definition; they should be considered an additional feature only for cursor-based devices.

Look at the article “[Designing for Touchscreen](http://www.whatcreative.co.uk/blog/tips/designing-for-touch-screen/)” for more ideas. Many of the design suggestions in it are best for touchscreens, but they would not necessarily impair cursor-based usability either. For example, sub-navigation on the right side of the page would be more user-friendly for touchscreen users, because most people are right-handed; they would therefore not bump or brush the navigation accidentally when holding the device in their left hand. This would make no difference to cursor users, so we might as well follow the touchscreen design guideline in this instance. Many more guidelines of this kind can be drawn from touchscreen-based usability.

### A Showcase Of Responsive Web Design [Link](http://www.smashingmagazine.com/2011/01/guidelines-for-responsive-web-design/#a-showcase-of-responsive-web-design)

Below we have a few examples of responsive Web design in practice today. For many of these websites, there is more variation in structure and style than is shown in the pairs of screenshots provided. Many have several solutions for a variety of browsers, and some even adjust elements dynamically in size without the need for specific browser dimensions. Visit each of these, and adjust your browser size or change devices to see them in action.

### Further Resources [Link](http://www.smashingmagazine.com/2011/01/guidelines-for-responsive-web-design/#further-resources)

* [Responsive Web Design](http://www.alistapart.com/articles/responsive-web-design/), A List Apart
* [CSS Media Query for Mobile is Fool’s Gold](http://www.cloudfour.com/css-media-query-for-mobile-is-fools-gold/), Cloud Four
* [Designing for a Responsive Web with Heuristic Methods](http://designreviver.com/articles/designing-for-a-responsive-web-with-heuristic-methods/), Design Reviver
* [Examples Of Flexible Layouts With CSS3 Media Queries](http://zomigi.com/blog/examples-of-flexible-layouts-with-css3-media-queries/), Zoe Mickley Gillenwater
* [The Big Web Show #9: Responsive Web Design](http://5by5.tv/bigwebshow/9), 5by5 Studios
* [How to Use CSS3 Media Queries to Create a Mobile Version of Your Website](http://www.smashingmagazine.com/2010/07/19/how-to-use-css3-media-queries-to-create-a-mobile-version-of-your-website/), Smashing Magazine
* [Application: Rapid Prototyping of Adaptive CSS and Responsive Design](http://protofluid.com/), ProtoFluid
* [Handcrafted CSS: More Bulletproof Web Design](http://www.amazon.com/exec/obidos/ASIN/0321643380/hivelogic-20), Dan Cederholm (printed book)
* [Flexible Web Book](http://www.flexiblewebbook.com/), Zoe Mickley Gillenwater (printed book)