

70 Expert Ideas For Better CSS

Coding

May 5th, 2007,

www.smashingmagazine.com/?p=116

CSS isn't always easy to deal with. Depending on your skills and your experience, CSS coding can sometimes become a nightmare, particularly if you aren't sure, which selectors are actually being applied to document elements. An easy way to minimize the complexity of the code is as useful as not-so-well-known CSS attributes and properties you can use to create a semantically correct markup.

We've taken a close look at some of the **most interesting and useful CSS tricks, tips, ideas, methods, techniques and coding solutions** and listed them below. We also included some basic techniques you can probably use in every project you are developing, but which are hard to find once you need them.

And what has come out of it is an overview of **over 70 expert tips, which can improve your efficiency of CSS coding.** You might be willing to check out the list of references and related

articles in the end of this post. You might be interested in reading our article [53 CSS-Techniques You Couldn't Live Without](#), which should provide you with a basic toolbox for CSS-based techniques you might use in your next projects.

We'd like to **express sincere thank to all designers** who shared their ideas, techniques, methods, knowledge and experience with their readers. Thank you, we, coders, designers, developers, information architects - you name it - really appreciate it.

1.1. Workflow: Getting Started

- **After you have a design, start with a blank page of content.** “Include your headers, your navigation, a sample of the content, and your footer. Then start adding your html markup. Then start adding your CSS. It works out much better.” [[CSSing](#)]
- **Reset your CSS-styles first.** “You can often eliminate the need to specify a value for a property by taking advantage of that property’s default value. Some people like doing a [Global white space reset](#) by zeroing both margin and padding for all elements at the top of their stylesheets. Eric Meyer’s [Global Reset](#), [Christian Montoya’s initial CSS file](#), [Mike Rundle’s initial CSS file](#), [Ping Mag’s initial CSS file](#). [[Roger Johansson](#)]
- **Use a master stylesheet.** “One of the most common mistakes I see beginners and intermediates fall victim to when it comes to CSS is not removing the default browser styling. This leads to inconsistencies in the appearance of your design across browsers, and ultimately leaves a lot of designers blaming the browser. It is a misplaced blame, of

course. Before you do anything else when coding a website, you should reset the styling.”
[[Master Stylesheet: The Most Useful CSS Technique](#)], [[Ryan Parr](#)]

1. **master.css**

2. `@import url("reset.css");`
3. `@import url("global.css");`
4. `@import url("flash.css");`
5. `@import url("structure.css");`

1. `<style type="text/css" media="Screen">`
2. `/**/@import url("css/master.css");/**/`
3. `</style>`

- **Keep a library of helpful CSS classes.** Useful for debugging, but should be avoided in the release version (separate markup and presentation). Since you can use multiple class names (i.e. `<p class="floatLeft alignLeft width75">...</p>`), make use of them debugging your markup. (*updated*) [[Richard K. Miller](#)]

1. CSS:
2. `.width100 { width: 100%; }`

```
3. .width75 { width: 75%; }
4. .width50 { width: 50%; }
5. .floatLeft { float: left; }
6. .floatRight { float: right; }
7. .alignLeft { text-align: left; }
8. .alignRight { text-align: right; }
```

1.2. Organize your CSS-code

- **Organize your CSS-styles, using master style sheets.** “Organizing your CSS helps with future maintainability of the site. Start with a master style sheet. Within this style sheet import your `reset.css`, `global.css`, `flash.css` (if needed) and `structure.css` and on occasion a typography style sheet. Here is an example of a “master” style sheet and how it is embedded in the document:”

```
1. h2 { }
2.     #snapshot_box h2 {
3.         padding: 0 0 6px 0;
4.         font: bold 14px/14px "Verdana", sans-serif; }
```

```
5. #main_side h2 {
6.     color: #444;
7.     font: bold 14px/14px "Verdana", sans-serif; }
8. .sidetagselection h2 {
9.     color: #fff;
10.    font: bold 14px/14px "Verdana", sans-serif; }
```

- **Organize your CSS-styles, using flags.** “Divide your stylesheet into specific sections: i.e. Global Styles – (body, paragraphs, lists, etc), Header, Page Structure, Headings, Text Styles, Navigation, Forms, Comments, Extras. [[5 Tips for Organizing Your CSS](#)]

```
1. /* -----*/
2. /* ----->>> GLOBAL <<<-----*/
3. /* -----*/
```

- **Organize your CSS-styles, making a table of contents.** At the top of your CSS document, write out a table of contents. For example, you could outline the different areas that your CSS document is styling (header, main, footer etc). Then, use a large, obvious section break to separate the areas. [[5 Steps to CSS Heaven](#)]
- **Organize your CSS-styles, ordering properties alphabetically.** “I don’t know where I

got the idea, but I have been alphabetizing my CSS properties for months now, and believe it or not, it makes specific properties much easier to find.” [[Christian Montoya](#)]

```
1. body {  
2.     background:#fdfdfd;  
3.     color:#333;  
4.     font-size:1em;  
5.     line-height:1.4;  
6.     margin:0;  
7.     padding:0;  
8. }
```

- **Separate code into blocks..** “This might be common sense to some of you but sometimes I look at CSS and it’s not broken down into “sections.” It’s easy to do an it makes working with code weeks, months, or years later much easier. You’ll have an easier time finding classes and elements that you need to change. Examples: /* Structure */, /* Typography */ etc.” [[CSS Tips and Tricks](#)]
- **Hook, line, and sinker.** Once you have your CSS and sections in place start considering where your selector “hooks” will live by using structural hooks in your mark up. This is your saving grace for future editing and maintenance of the site. This will also give you

strength in your document.” [[Ryan Parr](#)]

- **Break your style sheet in separate blocks.** “I break down my style sheet into three separate blocks. The first is straight element declarations. Change the body, some links styles, some header styles, reset margins and padding on forms, and so on. [...] After element declarations, I have my class declarations; things like classes for an error message or a callout would go here. [...] I start by declaring my main containers and then any styles for elements within those containers are indented. At a quick glance, I can see how my page is broken down and makes it easier to know where to look for things. I’ll also declare containers even if they don’t have any rules.” [[Jonathan Snook](#)]

1.3. Workflow: Handling IDs, Classes, Selectors, Properties

- **Keep containers to a minimum.** “Save your document from structural bloat. New developers will use many div’s similar to table cells to achieve layout. Take advantage of the many structural elements to achieve layout. Do not add more div’s. Consider all options before adding additional wrappers (div’s) to achieve an effect when using a little nifty CSS can get you that same desired effect.” [[Ryan Parr](#)]
- **Keep properties to a minimum.** “Work smarter, not harder with CSS. Under this rule, there are a number of subrules: if there isn’t a point to adding a CSS property, don’t add it; if you’re not sure why you’re adding a CSS property, don’t add; and if you feel like

you've added the same property in lots of places, figure out how to add it in only one place." [\[CSSing\]](#)

- **Keep selectors to a minimum.** "Avoid unnecessary selectors. Using less selectors will mean less selectors will be needed to override any particular style — that means it's easier to troubleshoot." [\[Jonathan Snook\]](#)
- **Keep CSS hacks to a minimum.** "Don't use hacks unless its a known and documented bug. This is an important point as I too often see hacks employed to fix things that aren't really broken in the first place. If you find that you are looking for a hack to fix a certain issue in your design then first do some research (Google is your friend here) and try to identify the issue you are having problems with. [\[10 Quick Tips for an easier CSS life\]](#)
- **Use CSS Constants for faster development.** "The concept of constants – fixed values that can be used through your code [is useful]. [..] One way to get round the lack of constants in CSS is to create some definitions at the top of your CSS file in comments, to define 'constants'. A common use for this is to create a 'color glossary'. This means that you have a quick reference to the colors used in the site to avoid using alternates by mistake and, if you need to change the colors, you have a quick list to go down and do a search and replace." [\[Rachel Andrew\]](#)

1. # /*
2. # Dark grey (text): #333333
3. # Dark Blue (headings, links) #000066
4. # Mid Blue (header) #333399

5. # Light blue (top navigation) #CCCCFF

6. # Mid grey: #666666

7. # */

- **Use a common naming system.** Having a naming system for id's and classes saves you a lot of time when looking for bugs, or updating your document. Especially in large CSS documents, things can get a big confusing quickly if your names are all different. I recommend using a `parent_child` pattern. [[10 CSS Tips](#)]
- **Name your classes and IDs properly, according to their semantics.** “We want to avoid names that imply presentational aspects. Otherwise, if we name something `right-col`, it's entirely possible that the CSS would change and our “right-col” would end up actually being displayed on the left side of our page. That could lead to some confusion in the future, so it's best that we avoid these types of presentational naming schemes. [[Garrett Dimon](#)]
- **Group selectors with common attributes.** “Group selectors. When several element types, classes, or id:s share some properties, you can group the selectors to avoid specifying the same properties several times. This will save space – potentially lots of it.” [[Roger Johansson](#)]
- **Isolate single properties that you are likely to reuse a lot.** “If you find yourself using a single property a lot, isolate it to save yourself repeating it over and over again and also enabling you to change the display of all parts of the site that use it.” [[5 Steps to CSS Heaven](#)]

- **Move ids and class naming as far up the document tree as you can.** Leverage [contextual selectors](#) as much as possible. Don't be afraid to be verbose in your selectors. Longer selectors can make css documents easier to read while also cutting down the chances of developing class- or [divitis](#). [[Chric Casciano](#)]
- **Learn to exploit the cascading nature of CSS.** “Say you have two similar boxes on your website with only minor differences - you could write out CSS to style each box, or you could write CSS to style both at the same time, then add extra properties below to make one look different.” [[5 Steps to CSS heaven](#)]
- **Use Your Utility Tags: <small>, and .** “Many times you'll have a section in your design that calls for various typographical weights/looks all on the same line, or very close to each other. drop in random divs and classes because I feel they're not semantic and defeat the purpose of your nice XHTML everywhere else.” Instead, use semantic tags. [[Mike Rundle's 5 CSS Tips](#)]

1.4. Workflow: Use shorthand notation

- **Shorten hexadecimal colour notation.** “In CSS, when you use hexadecimal colour notation and a colour is made up of three pairs of hexadecimal digits, you can write it in a more efficient way by omitting every second digit: #000 is the same as #000000, #369 is the same as #336699 [[Roger Johansson](#)]
- **Define pseudo classes for links in the LoVe/HAtE-order:** Link, Visited, Hover, Active. “To ensure that you see your various link styles, you're best off putting your styles in the

order “link-visited-hover-active”, or “LVHA” for short. If you’re concerned about focus styles, they may go at the end– but wait until you’ve read this explanation before you decide.” [[Eric Meyer](#)]

```
1. a:link { color: blue; }
2. a:visited { color: purple; }
3. a:hover { color: purple; }
4. a:active { color: red; }
```

- **Define element’s margin, padding or border in TRouBled-order:** Top, Right, Bottom, Left. “When using shorthand to specify an element’s margin, padding or border, do it clockwise from the top: Top, Right, Bottom, Left.” [[Roger Johansson](#)]
- **You can use [shorthand properties](#).**
“Using shorthand for margin, padding and bottom properties can save a lot of space.

```
1. CSS:
2.     margin: top right bottom left;
3.     margin: 1em 0 2em 0.5em;
4.     (margin-top: 1em; margin-right: 0; margin-bottom: 2em;
       margin-left: 0.5em;)
```

1. CSS:

2. `border:width style color;`

3. `border:1px solid #000;`

1. CSS:

2. `background: color image repeat attachment position;`

3. `background:#f00 url(background.gif) no-repeat fixed 0 0;`

1. CSS:

2. `font: font-style (italic/normal) font-variant (small-caps)
font-weight font-size/line-height font-family;`

3. `font: italic small-caps bold 1em/140% "Lucida
Grande",sans-serif;`

1.5. Workflow: Setting Up Typography

- **To work with EMs like with pxs, set font-size on the body-tag with 62.5%.** Default-value of the font-size is 16px; applying the rule, you'll get one Em standing for roughly ten pixels ($16 \times 62.5\% = 10$). "I tend to put a font-size on the body tag with value: 62.5%. This allows you to use EMs to specify sizes while thinking in PX terms, e.g. 1.3em is approximately 1.3px." [[Jonathan Snook](#)]

- **Use universal character set for encoding.** “[..] The answer is to use a single universal character set that’s able to cover most eventualities. Luckily one exists: UTF-8, which is based on Unicode. Unicode is an industry standard that’s designed to enable text and symbols from all languages to be consistently represented and manipulated by computers. UTF- 8 should be included in your web page’s head like this. [[20 pro tips](#)]

```
1. <meta http-equiv="content-type" content="text/ html; charset=utf-8" />
```

- **You can change capitalisation using CSS.** If you need something written in capitals, such as a headline, rather than rewriting the copy, let CSS do the donkey work. The following code will transform all text with an h1 attribute into all capitals, regardless of format”. [[20 pro tips](#)]

```
1. h1 {  
2.     text-transform: uppercase;  
3. }
```

- **You can display text in small-caps automatically.** The font-variant property is used to display text in a small-caps font, which means that all the lower case letters are converted to uppercase letters, but all the letters in the small-caps font have a smaller font-size compared to the rest of the text.

```
1. h1 {
2.     font-variant: small-caps;
3. }
```

- **Cover all the bases - define generic font-families.** “When we declare a specific font to be used within our design, we are doing so in the hope that the user will have that font installed on their system. If they don’t have the font on their system, then they won’t see it, simple as that. What we need to do is reference fonts that the user will likely have on their machine, such as the ones in the font-family property below. It is important that we finish the list with a generic font type. [[Getting into good coding habits](#)]

```
1. p {
2.     font-family: Arial, Verdana, Helvetica, sans-serif;
3. }
```

- **Use 1.4em - 1.6em for line-height.** “line-height:1.4” for readable lines, reasonable line-lengths that avoid lines much longer than 10 words, and colors that provide contrast without being *too* far apart. For example, pure black on pure white is often too strong for bright CRT displays, so I try to go with an off-white (#fafafa is a good one) and a dark gray (#333333, another good one).” [[Christian Montoya](#)]

- **Set 100.01% for the html-element.** This odd 100.01% value for the font size compensates for several browser bugs. First, setting a default body font size in percent (instead of em) eliminates an IE/Win problem with growing or shrinking fonts out of proportion if they are later set in ems in other elements. Additionally, some versions of Opera will draw a default font-size of 100% too small compared to other browsers. Safari, on the other hand, has a problem with a font-size of 101%. The current “best” suggestion is to use the 100.01% value for this property.” [[CSS: Getting into good habits](#)]

1.6. Workflow: Debugging

- **Add borders to identify containers.** “Use plenty of test styles like extra borders or background colors when building your documents or debugging layout issues. `div { border:1px red dashed; }` works like a charm. There are also [bookmarklets that apply borders](#) and do other things for you.” You can also use `* { border: 1px solid #ff0000; }`. [[Chric Casciano](#)]. Adding a border to specific elements can help identify overlap and extra white space that might not otherwise be obvious. [[CSS Crib Sheet](#)]

```
1. * { border: 1px solid #f00; }
```

- **Check for closed elements first when debugging.** “If you ever get frustrated because it seemed like you changed one minor thing, only to have your beautiful holy-grail layout

break, it might be because of an unclosed element. [[10 CSS Tips](#)]

2.1. Technical Tips: IDs, Classes

- **1 ID per page, many classes per page.** “Check your IDs: Only one element in a document can have a certain value for the id attribute, while any number of elements can share the same class name. [...] Class and id names can only consist of the characters [A-Za-z0-9] and hyphen (-), and they cannot start with a hyphen or a digit (see CSS2 syntax and basic data types).” [[Roger Johansson](#)]
- **Element names in selectors are case sensitive.** “Remember case sensitivity. When CSS is used with XHTML, element names in selectors are case sensitive. To avoid getting caught by this I recommend always using lowercase for element names in CSS selectors. Values of the class and id attributes are case sensitive in both HTML and XHTML, so avoid mixed case for class and id names.” [[Roger Johansson](#)]
- **CSS classes and IDs must be valid.** “I.e. [beginning with a letter](#), not a number or an underscore. IDs must be unique. Their names should be [generic](#), describe functionality rather than appearance.” [[CSS Best Practices](#)]
- **You can assign multiple class names to a given element.** “You can assign multiple class names to an element. This allows you to write several rules that define different properties, and only apply them as needed.” [[Roger Johansson](#)]

2.2. Technical Tips: Use the power of selectors

Roger Johansson has written an **extremely** useful series of articles about [CSS 2.1 Selectors](#). These articles are **highly recommended** to read - some useful aspects can be found in the list below. Note that selectors '>' and '+' aren't supported in IE6 and earlier versions of Internet Explorer (*updated*).

- **You can use child selectors.** “A child selector targets an immediate child of a certain element. A child selector consists of two or more selectors separated by a greater than sign, “>”. The parent goes to the left of the “>”, and whitespace is allowed around the combinator. This rule will affect all strong elements that are children of a div element. [\[Roger Johansson\]](#)

```
1. div > strong { color:#f00; }
```

- **You can use adjacent sibling selectors.** An adjacent sibling selector is made up of two simple selectors separated by a plus sign, “+”. Whitespace is allowed around the adjacent sibling combinator. The selector matches an element which is the next sibling to the first element. The elements must have the same parent and the first element must

immediately precede the second element. [[Roger Johansson](#)]

```
1. p + p { color:#f00; }
```

- **You can use attribute selectors.** Attribute selectors match elements based on the presence or value of attributes. There are four ways for an attribute selector to match:

```
1. [att]
```

```
2.    Matches elements that have an att attribute, regardless of its  
      value.
```

```
3. [att=val]
```

```
4.    Matches elements that have an att attribute with a value of exactly  
      "val".
```

```
5. [att~=val]
```

```
6.    Matches elements whose att attribute value is a space-separated  
      list that contains "val". In this case "val" cannot contain spaces.
```

```
7. [att|=val]
```

```
8.    Matches elements whose att attribute value is a hyphen-separated  
      list that begins with "val". The main use for this is to match  
      language subcodes specified by the lang attribute (xml:lang in
```

XHTML), e.g. “en”, “en-us”, “en-gb”, etc.

9.

- The selector in the following rule matches all `p` elements that have a `title` attribute, regardless of which value it has:

```
1. p[title] { color:#f00; }
```

- The selector matches all `div` elements that have a `class` attribute with the value `error`:

```
1. div[class=error] { color:#f00; }
```

- Multiple attribute selectors can be used in the same selector. This makes it possible to match against several different attributes for the same element. The following rule would apply to all `blockquote` elements that have a `class` attribute whose value is exactly “quote”, and a `cite` attribute (regardless of its value):

```
1. blockquote[class=quote][cite] { color:#f00; }
```

- **You should use descendant selectors.** “Descendant selectors can help you eliminate

many class attributes from your markup and make your CSS selectors much more efficient.” [[Roger Johansson](#)]

2.3. Technical Tips: Styling Links

- **Be careful when styling links if you’re using anchors.** “If you use a classic anchor in your code () you’ll notice it picks up :hover and :active pseudo-classes. To avoid this, you’ll need to either use id for anchors instead, or style with a [slightly more arcane](#) syntax: :link:hover, :link:active” [[Dave Shea](#)]
- **Define relationships for links.** “The rel attribute is supposed to indicate a semantic link relationship from one resource to another.

```
1. a[rel~="nofollow"]::after {
2.     content: "\2620";
3.     color: #933;
4.     font-size: x-small;
5. }
6. a[rel~="tag"]::after {
7.     content: url(http://www.technorati.com/favicon.ico);
```

8. }

- “These make use of the attribute selector for space separated lists of values. Any a element with a relationship containing those values will be matched. Links with the nofollow relationship will be followed by a dark red skull and crossbones (?) and those with the tag relationship will be followed by the Technocrati icon.” [[Handy CSS](#)]
- **You can mark external links automatically.** Many people make use of the non-standard `rel="external"` relationship to indicate a link to an external site. However, adding that to each and every link is time consuming and unnecessary. This style rule will place an north east arrow after any link on your site to an external site. [[Handy CSS](#)]

```
1. a[href^="http://"]:not([href*="smashingmagazine.com"]):after {  
2.     content: "\2197";  
3. }
```

- **You can remove dotted links with `outline: none;`**. To [remove dotted links](#) use `outline: none;`

```
1. a:focus {
```

```
2. outline: none;
3. }
```

2.4. Technical Tips: CSS-Techniques

- **You can specify body tag ID.** “In most cases placing an ID in the body tag will allow you manipulate CSS presentational items and markup elements by page by page basis. Not only will you be able to organize your sections you will be able to create multiple CSS presentations without changing your markup from template to template or page to page.” [[Ryan Parr, Invasion of Body Switchers](#)]
- **You can create columns with equal heights with CSS.** [Equal Height Technique](#): a method to make all columns appear to be the same height. But without the need for faux column style background images. [Faux Columns](#): with background images.
- **You can align vertically with CSS.** “Say you have a navigation menu item whose height is assigned 2em. Solution: specify the line height to be the same as the height of the box itself in the CSS. In this instance, the box is 2em high, so we would insert line-height: 2em into the CSS rule and the text now floats in the middle of the box!” [[Evolt.org](#)]
- **You can use pseudo-elements and classes to generate content dynamically.** [Pseudo-classes and pseudo-elements](#). Pseudo-classes and pseudo-elements can be used to format elements based on information that is not available in the document tree. For example, there is no element that refers to the first line of a paragraph or the first letter of an element’s text content. You can use :first-child, :hover, :active, :focus, :first-line,

:first-letter, :before, :after and more.

- **You can set <hr> to separate posts beautifully.** “Restyling the horizontal rule (<hr>) with an image can be a beautiful addition to a web page. [[CSS: Best Practices](#)]
- **You can use the same navigation (X)HTML-code on every page.** “Most websites highlight the navigation item of the user’s location in the website. But it can be a pain as you’ll need to tweak the HTML code behind the navigation for each and every page. So can we have the best of both worlds?” [[Ten More CSS Tricks you may not know](#)]

1. XHTML:

2.

3. Home

4. About us

5. Contact us

6.

- Insert an id into the <body> tag. The id should be representative of where users are in the site and should change when users move to a different site section.

1. CSS:

2. #home .home, #about .about, #contact .contact

```
3. {  
4. commands for highlighted navigation go here  
5. }
```

- **You can use `margin: 0 auto;` to horizontally centre the layout.** “To horizontally centre an element with CSS, you need to specify the element’s width and horizontal margins.” [[Roger Johansson](#)]

```
1. XHTML:  
2. <div id="wrap">  
3. <!-- Your layout goes here -->  
4. </div>
```

```
1. CSS:  
2. #wrap {  
3. width:760px; /* Change this to the width of your layout */  
4. margin:0 auto;  
5. }
```

- **You can add CSS-styling to RSS-feeds.** “You can do a lot more with an XSL stylesheet (turn links into clickable links, etc), but CSS can make your feed look much less scary for the non-technical crowd. [[Pete Freitag](#)]

```
1. <?xml version="1.0" ?>
2. <?xml-stylesheet type="text/css" href="http://you.com/rss.css" ?>
3. ...
```

- **You can hide CSS from older browsers.** “A common way of hiding CSS files from old browsers is to use the @import trick. [[Roger Johansson](#)]

```
1. @import "main.css";
```

- **Always declare margin and padding in block-level elements.** [[10 CSS Tips](#)]
- **Set a width OR margin and padding.** “My rule of thumb is, if I set a width, I don’t set margin or padding. Likewise, if I’m setting a margin or padding, I don’t set a width. Dealing with the box model can be such a pain, especially if you’re dealing with percentages. Therefore, I set the width on the containers and then set margin and padding on the elements within them. Everything usually turns out swimmingly.” [[Jonathan Snook](#)]
- **Avoid applying padding/borders and a fixed width to an element.** “IE5 gets the box

model wrong, which really makes a mess of things. There are ways around this, but it's best to side-step the issue by applying the padding to the parent element instead of the child that gets a fixed-width. [\[CSS Crib Sheet\]](#)

- **Provide print styles.** “You can add a print stylesheet in exactly the same way that you would add a regular stylesheet to your page:

```
1. <link rel="stylesheet" type="text/css" href="print.css" media="print">  
2. or  
3. <style type="text/css" media="print"> @import url(print.css); </style>
```

- This ensures that the CSS will only apply to printed output and not affect how the page looks on screen. With your new printed stylesheet you can ensure you have solid black text on a white background and remove extraneous features to maximise readability. [More about CSS-based print-Layouts.](#) [\[20 pro tips\]](#)

2.5. Technical Tips: IE Tweaks

- **You can force IE to apply transparency to PNGs.** “In theory, PNG files do support varied levels of transparency; however, an Internet Explorer 6 bug prevents this from working cross-browser.” [\[CSS Tips, Outer-Court.com\]](#)

```
1.   #regular_logo
2.  {
3.   background:url('test.png'); width:150px; height:55px;
4.  }
5.  /* \ */
6.  * html #regular_logo
7.  {
8.   background:none;
9.   float:left;
10.  width:150px;
11.  filter:progid:DXImageTransform.Microsoft.AlphaImageLoader(src='test.
    png', sizingMethod='scale');
12. }
13. /* */
```

- **You can define min-width and max-width in IE.** You can use Microsoft's dynamic expressions to do that. [[Ten More CSS Trick you may not know](#)]

```
1. #container
2. {
3. min-width: 600px;
4. max-width: 1200px;
5. width:expression(document.body.clientWidth < 600? "600px" :
   document.body.clientWidth > 1200? "1200px" : "auto");
6. }
```

- **You can use Conditional Comments for IE.** “The safest way of taking care of IE/Win is to use [conditional comments](#). It feels more future-proof than CSS hacks – is to use Microsoft’s proprietary conditional comments. You can use this to give IE/Win a separate stylesheet that contains all the rules that are needed to make it behave properly.” [[Roger Johansson](#)]

```
1. <!--[if IE]>
2.     <link rel="stylesheet" type="text/css" href="ie.css" />
3.     <![endif]-->
```

Workflow: Get Inspired

- **Play, experiment with CSS.** “Play. Play with background images. Play with floats.” [[Play with positive and negative margins](#)]. Play with inheritance and cascading rules. Play. [[Chric Casciano](#)]
- **Learn from others.** Learn from great sites built by others. Any site’s HTML is easily accessible by viewing a page’s source code. See how others have done things and apply their methods to your own work. [[20 pro tips](#)]

Sources and Related Posts

- [CSS Tips and Tricks](#) by *Roger Johansson*
- [\(The Only\) Ten Things To Know About CSS](#) by *John Manoogian*
- [CSS Crib Sheet](#) by *Dave Shea*
- [My Top Ten CSS Tricks \[CSS Tutorials\]](#) by *Trenton Moss*
- [CSS Tips](#) by *Philipp Lenssen*
- [Top CSS Tips](#) by *Jonathan Snook*
- [Ten CSS tricks — corrected and improved](#) by *Tantek Çelik*
- [Ten More CSS Trick you may now know](#) by *Trenton Moss*
- [CSS techniques I use all the time](#) by *Christian Montoya*
- [CSS Tip Flags](#) by *Douglas Bowman*

- [My 5 CSS Tips](#) by *Mike Rundle*
- [5 Steps to CSS Heaven](#) by *Ping Mag*
- [Handy CSS](#) by *Lachlan Hunt*
- [Erratic Wisdom: 5 Tips for Organizing Your CSS](#) by *Thame Fadiel*
- [15 CSS Properties You Probably Never Use \(but perhaps should\)](#) by *SeoMoz*
- [10 CSS Tips You Might Not Have Known About](#) by *Christopher Scott*
- [A List Apart: Articles: 12 Lessons for Those Afraid of CSS and Standards](#) by *Ben Henick*
- [Tips for a better design review process](#) by *D. Keith Robinson*
- [20 pro tips - .net magazine](#) by *Jason Arber*
- [CSS Best Practices](#) by *Richard K Miller*
- [10 Quick Tips for an Easier CSS Life](#) by *Paul Ob*
- [10 CSS Tips from a Professional CSS Front-End Architect](#) by *72 DPI in the shade team blog*
- [Web Design References: Cascading Style Sheets](#) by *Laura Carlson*
- [Getting Into Good Coding Habits](#) by *Adrian Senior*

Visit Smashingmagazine.com for more!



we smash you with information, which will make your life easier. really.

by Vitaly Friedman, Sven Lennartz, www.smashingmagazine.com, 23.05.2007

about: <http://www.smashingmagazine.com/about/>

e-mail: office@smashingmagazine.com

advertise with us: advertising@smashingmagazine.com (Michael Dobler)