# CSS Crash Course

*Sébastien Aperghis-Tramoni*
*sebastien@aperghis.net*

## 16.1 Why talking about CSS in a Perl conference?

Because CSS is a very nice technology, which allow for a great expressivity. Plus, CSS shares some common points with Perl: with its short and clear syntax (lazyness), you can express very powerful things you can be proud of (hubris). And you can see the result *now* (impatience).

Another reason is that I saw several talented Perl programmers very ignorant in CSS and seeing that as something reserved to a web designer, which is quite wrong as they could understand and write simple stylesheets with few efforts. Hence this talk, where I'll try to present several examples by starting from a HTML page and progressively add CSS in order to enhance the presentation. The rest of this abstract is a gentle introduction to CSS.

## 16.2 What are CSS anyway?

Cascading StyleSheet, a.k.a. CSS, is a standard edicted by the W3C (World Wide Web Consortium), the same that wrote and maintain widely used standards like HTML and XML (and many other more-or-less related things like XSLT and XML-RPC).

Current version (2.1) of the standard is supported by most of the modern browsers, except retarded ones (particularly one from a small Redmond-based software company, but I can't recall the name). The next version, 3.0, is currently still being written but some parts of it are already supported by many browsers.

CSS were first created in order to separate the presentation layer from the structural layer in HTML documents because all those `<FONT>` tags were really getting ugly and unmaintainable. Now, HTML, and particularly XHTML, is supposed to only contain structural markup and let the CSS do the work for the presentation part. In the process we gained a simple yet powerful language which allows to express richer typographic effects and different renderings depending upon the target medium (computer screen, mobile device, printer or even aural and braille device).

## 16.3 A kind of magic

Thanks to CSS, you can get rid of all the old and ugly tricks that were previously used to achieve graphical effects. No more layout tables, invisible GIF and `<FONT>` tag soup. You can even trash the JavaScript scripts used for creating menus, as they can be done in pure CSS. And writing

JavaScript is also easier, as you can make HTML elements appear and disappear by switching their CSS properties.

Making selected elements invisible is very useful for creating pages that gracefully degrade when viewed on browsers without CSS support. For example, a menu for quickly accessing parts of a documents, while useless in a graphical browser like Firefox, may be very useful in a text browser like Lynx. Why should you care about Lynx? Because this browser (or a similar low-tech HTML renderer) is typically used as a base for software designed to help impaired people. Think voice synthesis or braille "screen".

On the other hand, CSS can create content on-the-fly, purely for presentational mean. Think of the bulleted lists. The bullet are not part of the HTML, but generated by the renderer. Using CSS, you can change the bullet with any other character, string or image.

## 16.4 The compatibility problem

As said in the introduction, not all web browsers support the same amount or version of CSS. When dealing with CSS, you usually consider the following four engines:

- Gecko, the well-know engine that powers Mozilla, Firefox, Camino, Nautilus and a few others.
- Opera, the lightweight and multiplatorms browser.
- KHTML, the engine that powers Konqeror and Safari.
- Internet Explorer, a.k.a. MSIE, the security hole$\hat{W}\hat{W}$browser included with Windows.

The first three have a very good support for CSS/2.1, and the first two also support the finalized parts CSS/3. MSIE supports... CSS/1.0. Ugh! That mean that you can't write a single CSS and expect it to work everywhere, and especially to work on MSIE.

There are two ways to solve this problem:

- the web designer *"it must look the same everywhere"* way.
- the *"I so much don't and don't want to care"* way.

The first way means writing a CSS for each browser, and in some cases for each combination of browser+platform. The browser and platform are usually recognized by looking at its user-agent string, which can of course break in several ways. Another solution is to write only one CSS and isolate the parts dedicated to specific browsers using well-know bugs of the CSS parsers. This is extremely awful and can break with each release of any browser.

The second way would previously mean that people using the retarded MSIE could not see your web site. Now, thanks to CSS hacker Dean Edwards, you can more safely follow this way and even expect your CSS/2 stylesheet to work on MSIE. His JavaScript hackery IE7 being the thing that improve MSIE to make it support CSS/2.1, parts of CSS/3, and even correct long standing bugs like flaky PNG support.

## 16.5 A few links

The material corresponding to this presentation will be (hopefully) available on my web site, `http://www.maddingue.org/conferences/yapc-eu-2005/csscc/`

CSS/2.1 Specification: `http://www.w3.org/TR/CSS21/`

Dean Edwards IE7: `http://dean.edwards.name/ie7/`

Mozilla Firefox: `http://www.mozilla.org/products/firefox/`