Why Hack CSS?

The theory of Cascading Style Sheets (CSS) is a means to an end: better, more efficient Web site design. In the real world, however, CSS does not provide a perfect, clear-cut path to that goal. To achieve the promise of CSS, working designers have employed a series of workarounds known collectively as hacks. At the most basic level, a CSS hack is a modification to the standard CSS code. Like any deviation from the norm, the use of CSS hacks has both its supporters and detractors: Some designers feel CSS hacks are an absolute necessity and others are fervently opposed to them.

To figure out why the Web design community is divided over CSS hacks—and which camp you should be in—you’ll need a little background on the emergence of CSS.

The Cascading Style Sheets Promise

When work was begun in 1995 on the first CSS specification, the Web was one giant kludge. Hypertext Markup Language (HTML) tags were being pressed into service to handle chores they were never intended for. Tables, for example, meant to contain structured data were largely used for layout. But missapplied tags were the least of the designer’s woes when it came to working with HTML.

HTML is perfectly suited for its original design: to represent scientific papers and other documents that adhered to a highly structured format. A structured document is formatted with headings and, where necessary, subheadings, for all titles along with standard paragraphs for all body text. HTML hit a major stumbling block when the Internet was eclipsed by the World Wide Web—and graphic design came to the Web.

Designers used every trick in the book, and invented quite a few along the way, to reproduce their designs with HTML. Presentation tags, such as font, were inextricably entwined into the content—which meant sitewide style changes required a Herculean effort. To alter the typeface for all of a Web site’s primary headings, you had to either modify every single tag instance by hand or cross your fingers and perform an all-encompassing (and terrifying) search-and-replace operation. If you wanted to repurpose Web content for print or any other media, you had one choice and one choice only: redesign the site, page-by-page. Pages were top-heavy with dense mark-up code: a
real structural jungle that designers had to hack through to make the smallest change. Moreover, any hope of HTML working with assistive technologies such as screen readers was completely off the table.

The original drafters of the CSS specification hoped to cut away all the clutter brought by styling Web pages with HTML. CSS was conceived with numerous key advantages in mind:

- Separate presentation from content
- Flexible design model
- Faster loading times
- Easy, instant maintenance
- Portability
- Advanced design possibilities
- Enhanced user control
- Accessibility

The following sections examine each of these in detail so you grasp completely what CSS can do.

**Separate Presentation from Content**

The first and foremost mission was to disconnect the tight stranglehold that HTML style tags brought to a Web page. By isolating the control of a page’s look-and-feel from the content, a clear pathway to building structured pages opens up where you can still get the design you want. Additionally, both content and design benefit in terms of accessibility. Search engines can get at the content easier for indexing, while designers have hands-on control of their presentation. The core concept of separating presentation from content leads to many other benefits of the CSS model.

**Flexible Design Model**

Even with the most basic implementation of text styling, CSS runs rings around HTML. Whereas font tags are limited to seven browser-dependent sizes, CSS offers both absolute, number-based systems in the measurement unit of your choice (points, pixels, ems, percentages, and many others) and relative keyword-based systems (that is, small, medium, large, smaller, larger, and so on). Design considerations common in print publishing (such as line spacing) are impossible in HTML but a snap in CSS.

Aside from the specific properties available, the CSS methodology of assigning those properties is wonderfully robust. With CSS, you can re-style existing HTML tags or create custom styles in a variety of ways with classes, IDs, and through selectors. Selectors (whether as common as a descendent selector or as rare as adjacent-sibling selectors) encourage structural Web coding while delivering enhanced design control.
Chapter 1 — Why Hack CSS?

**Faster Loading Times**

To the casual Internet user, a Web page is completely virtual with no real substance or weight. Web designers, however, are very aware of the weight of their pages; the more code that’s in a page, the heavier it is and the longer it takes to load. For example, here’s the minimum code it takes to place a sentence on the page in the common HTML container, a `font` tag:

```
<table>
    <tr>
        <td>
            Welcome to my world.
        </td>
    </tr>
</table>
```

Now, here’s the same content in the standard CSS container, a `div` tag:

```
<div>
    Welcome to my world.
</div>
```

Multiply that doubled-difference many, many times for a single page—and then again for an entire Web site—and you have some idea of CSS’s edge in speed.

**Easy, Instant Maintenance**

As noted earlier, it’s a nightmare to change an HTML style across a site because styles are all applied at the lowest level, the tag. Not only must all pages with all the styles be altered one at a time, each page must be re-saved and then re-put to the server. With well-structured CSS, on the other hand, your styles are kept in a separate file where they can be modified in any text editor. Once published, a style change is immediately seen by anyone who views an affected page within the site.

**Portability**

Although the Internet may at times seem pervasive, it’s just one of many media. For example, many sites strive to have their Web pages available for print as well. With HTML-styled pages, the only viable route is to redesign the page with print in mind—a terrible chore to do it once and a never-ending time-suck if the site is updated frequently. CSS turns the HTML model on its head and allows you to simply specify a different style sheet for print—and, if desired, one for speech synthesizers, projectors, and hand-held devices, among others—and you’re done.

**Advanced Design Possibilities**

As defined, CSS is highly interactive and throws open the door to a multitude of design options. Just a few of the advanced text options were mentioned earlier; in addition to advanced sizing and line spacing, CSS also provides more robust alignment and far more specific margin and padding options. Text is not the only element to gain a power surge under CSS. The capability to control the position and tiling of background images (see Figure 1-1) is reason enough to use CSS in and of itself.
Another key element in the CSS toolchest is the `div` tag, commonly referred to in Web authoring programs like Macromedia Dreamweaver as *layers*. Content within a `div` tag can be placed anywhere on a page or made to flow in the context of a document. A `div` tag, like `span`, is a non-semantic tag that is used as a generic container; `div` tags are nothing more than block elements that enable you to mark up broad sections of a document. From a dynamic point of view, `div` tags can be programmatically hidden, revealed, change style, and even move across (or off) the page.

**Enhanced User Control**

While CSS provides a great deal of design-time flexibility, it also opens up the run-time options for Web page visitors. The entire notion of the cascade in Cascading Style Sheets stems from the originators' desire to blend the designer's style sheet with the user's. The end result is a cascade of several style sheets all coming together to render the page optimally. Some CSS-savvy designers have taken this a step further and designed their sites with multiple CSS styles attached to each page. Modern browsers include a style switching command that lists available style sheets.
One of the most commonly adjusted user settings is font size. The smallish text that looks good on the design spec may be too tiny to be read by a particular visitor—and that’s okay, if the page is styled properly with CSS and the text can expand to a readable size, as shown in Figure 1-2. When designed correctly, the text grows and the layout flows: site designer, site visitor, and site owner are all happy.

**Accessibility**

Adjustable text size is just one aspect of a vital trend in Web design: accessibility. Fueled by the Federal Rehabilitation Act, a U.S. law mandating that all government-run Web sites follow the guidelines established in Section 508 of that act, accessibility is on every designer’s watch list. The very core of Cascading Style Sheets—separating presentation from content—makes the content within the pages more available.

This openness, or accessibility, is immediately noticeable when you listen to software screen readers. Screen readers are a crucial assistive technology. If you ever want to demonstrate the benefit of CSS, just point a screen reader–enabled browser to a page in a CSS-based layout—and then visit the same page in a table-based layout, especially those with deeply nested tables. You won’t believe your ears when you hear what a difference CSS makes.

![Figure 1-2: Define your font sizes correctly with CSS, and text is easily rescaled with no sacrifice of design integrity.](image-url)
One CSS 2.1 specification goes to the next level in aiding the visually impaired to browse the Web by carving out a new media type: speech. Speech is a separate media type (just like print or hand-held devices) that allows designers to control how CSS classes and other selectors sound, just like the screen media type controls how CSS selectors look. Support for the speech media type is pretty much nonexistent at this time, but a much fuller implementation already on the table for CSS 3 bodes well for this much-needed functionality.

### Why CSS Is Broken

Cascading Style Sheets certainly were intended to be the Web designer’s promised land. Unfortunately, the first time you attempt to implement a CSS solution for a site, you’ll quickly realize that the promise has not been kept.

How bad can it be? Take a look at a typical CSS problem shown in Figures 1-3 and 1-4. Figure 1-3 displays the page as designed in Macromedia Dreamweaver MX 2004, whereas Figure 1-4 renders the same page in Internet Explorer 6. Look carefully at the model’s head in both figures and you’ll see that in Internet Explorer, the top of her head has come off and is shifted to the left by a number of pixels. What’s happening is that the design requires that the head image be sliced and placed in two CSS-styled `div` tags and Internet Explorer is adding several pixels to the bottom `div`. It’s enough to make CSS designers lose their minds—if not their heads.

![Figure 1-3: Dreamweaver gets it right, and the model’s head looks as it should.](image-url)
FIGURE 1-4: Internet Explorer 6 is flawed when rendering floated div tags—and the top of the model’s head is noticeably off.

You can find two different approaches to fixing the Internet Explorer pixel shift in Chapter 3, “Hiding CSS from Newer Browsers,” and Chapter 4, “Applying Conditional Comments.”

So, what went wrong with CSS? Although some errors have appeared in the CSS recommendations themselves, the major problem has been spotty, inconsistent, or downright wrong browser implementation. The reasons for the browser inconsistencies are as varied as the browsers themselves. For example, one of the biggest ongoing CSS headaches has been working with Netscape 4.x browsers. The primary problem with this version was one of timing: the CSS recommendations were finalized while Netscape 4 was in the final stages of its development cycle. Consequently, only a fraction of CSS specifications were enabled—and not all of them well.

Even the same browser version from the same company can differ wildly. Take, for example, Internet Explorer 5. On the PC, Internet Explorer 5 supported much larger portions of the CSS specification than ever before. When Internet Explorer 5 was released for the Macintosh, the design community was stunned to see that CSS support was even better—not just from Netscape’s latest release, but also from the PC version of the same browser. This development further complicated life for the Web designer, who often developed sites on a Macintosh, only to see them break on the PC.

The fact that CSS works as well as it does is pretty amazing. Think of it: You’re working with a standard developed over a long period of time by many large groups of independent thinkers, which is then implemented by another assortment of organizations (of varying resources and expertise) who are expected to create identical results from within their own. It’s like handing the blueprints of the Taj Mahal to 10 architects in 10 different countries with 10 different cultures and the full spectrum of economic standing and material and saying, “Go for it.”
Naturally, there are going to be differences in design, as well as omissions and unrequested and market-confusing enhancements.

In recent years, the latest generation of browsers have been focused squarely on getting CSS right. Browsers from Mozilla.org (including Firefox and Mozilla) along with the latest releases of Apple’s Safari browser have made tremendous strides in correctly interpreting the recommendations of the World Wide Web Consortium (W3C). And yet, they are all still rife with inconsistencies and contradictory behavior. To some, the details of their differences are fairly minor, but to designers with a perfectionist eye and a mandate to build universally accessible Web sites, details matter.

To Hack or Not to Hack

So, the situation, in brief, is this: Web designers have in their hands a wonderful technology with loads of benefits across the board, but it doesn’t work as well as it should. Or rather, CSS doesn’t work as well as it could—with a little help. And help is available, an amazing amount of help, in fact, in the form of CSS hacks and filters uncovered by a legion of working Web designers.

What exactly is a CSS hack? Typically, a CSS hack is a slight modification to the CSS or HTML code developed to work around a particular CSS problem on a specific browser. Many CSS hacks act as filters, hiding one or more styles from a problematic browser. For example, suppose you have a style sheet that includes an absolutely positioned `div` area on the right edge of the screen. Unfortunately, Internet Explorer 5.x on the Mac doesn’t render this properly and, as shown in Figure 1-5, an unnecessary and unwanted scroll bar appears at the bottom of the browser window. You can fix this problem in Internet Explorer Mac by declaring a negative right margin for the `div` style—which, of course, breaks the page in all other browsers. To ensure the design of the page looks the way you want it to when viewed with this browser version and all others, you must first set the style rule so that it works in Internet Explorer 5 Mac, as shown here:

```css
div#rightEdge {
    position:absolute;
    top: 20px; right: 10px;
    margin: 0 -10px 0 0;
}
```

Next, you need to reset to properties so the area renders correctly for all browsers, but is hidden from the problem browser, as shown here:

```css
/* hide from IE mac */
div#rightEdge {
    right: 0;
    margin: 0
}
/* reveal to IE Mac */
```

The hack is contained within the two comments surrounding the style declaration. The key is escaping the end of the first comment with a backslash, `\*/`, which makes Internet Explorer Macintosh disregard the rest of the style until the second closing comment delimiter, `*/`, is encountered. In this case, two CSS-style comment tags (one slightly altered) comprise the hack.
A school of CSS usage experts has just been outraged. According to them, the change just made to this code is, on all levels, wrong. Coding a CSS hack goes against the very nature of a W3C-recommended standard and should be considered an affront to Web designers everywhere. The faithful application of standards (whether they govern XHTML, CSS, or any other) is absolute. To code in any other way diminishes the standard and concesses a victory to the chaotic nature of the Web standards that they are created to battle.

I regard myself as a practical Web designer and, to me, this argument against CSS hacks is purely academic. A good friend of mine, Massimo Foti (known in Dreamweaver circles as a “developer’s developer”) once said, “Web standards are suggestions, not religion.” I think he’s right.

Other arguments against the user of CSS hacks are more meaningful. Some coders point out that such hacks may not be forward-compatible and may break in the next round of browser releases. I agree that such an event is a possibility, but I’m not at all sure it’s a certainty. Moreover, the very nature of external style sheets means that correcting any such issues in the future is a relatively centralized action: You’re not updating hundreds of pages in a site, you’re modifying one or two style sheets.
Consider this real-world story. A designer friend asked me to consult on a Web site redesign for a major metropolitan public library. She’s an excellent designer, but new to CSS, especially when it comes to layout (a client requirement for this job). The mandate was to use CSS layout techniques coupled with full compliance with Section 508.

“Not a problem,” I said. “What are the target browsers?”

The answer was, as you might expect, “Everything.”

It seems that although the vast majority of people outside of the library system used Internet Explorer 6, all the computers inside the library branches throughout the city used Netscape 4. Budget cut after budget cut had prevented an administrative rollout of a more modern browser. To satisfy all of the client’s bottom-line goals (CSS layout and full cross-browser compatibility), the only recourse was to employ CSS hacks.

Designers are often perfectionists and, given that someone else is paying for their work, often need to be. The client isn't (and shouldn't be) concerned with the ins and outs of CSS. That’s not the client’s responsibility; that’s why the client hired you, the designer. Clients want a Web site designer. They don’t want a Web site design that charges according to who is looking at it on what browser or operating system.

Learning and implementing CSS is no trivial task. It takes a great deal of practice, study, and application. After struggling up the learning curve and designing their first CSS-based site, many designers rightfully say, “You mean to tell me that after all that work, it still isn't right?”

Complications in applying CSS are, in essence, a fact of life. Software engineers often shorten the phrase “fact of life” to FOL when referring to an unchangeable condition. Some bugs can be fixed, while others are FOL. The implication is, of course, that when some aspect of a technology is FOL, you're SOL.

Whether or not you’re incorporating CSS hacks is a choice every Web designer must make. For me, CSS hacks are a FOL—and they get the job done. To hack or not to hack: in a perfect world, no; in the real world, yes.