

Cross Browser Issues: CSS Hacks explained, Tips, Tricks and Fixes

Jennifer Sullivan Cassidy

2006



Anyone who creates Cascading Style Sheets knows that Internet Explorer is a headache of a browser to build for because of the way it handles CSS. In this article, Jennifer goes into detail about how to deal with some of the most common bugs, and notes which ones may be fixed in IE 7.

Cross Browser Issues: CSS Hacks Explained, Tips, Tricks and Fixes

Anyone who creates Cascading Style Sheets knows that Internet Explorer is a headache of a browser to build for because of the way it handles CSS. In this article, Jennifer goes into detail about how to deal with some of the most common bugs, and notes which ones may be fixed in IE 7.

In the previous article (which you can read here), I covered some cross browser compatibility issues that web developers and designers who use Cascading Style Sheets, or CSS, may run across. We touched on the differences among the top browsers, and talked briefly about why they do what they do in regards to CSS. In this article, I want to go a step further, and be specific about particular cross browser hacks and fixes, and to give you a few ways to overcome these particular issues.

I do want to point out that while I can understand the necessity for these workarounds, I always advocate code that is 100% compliant. When I write CSS, I try to make it as simple as possible, as I have found that the simpler the CSS, the less chance I have of design issues in Internet Explorer. Sometimes, however, I just can't achieve the look I want with the simple code. Hence, I do at times use workarounds.

In this article, you're going to find that most of these workarounds deal with Internet Explorer, for the reason that we outlined in the first article. Essentially, IE is a browser that "thinks" it knows what you are trying to tell it, instead of taking the directives in your CSS literally like Mozilla and Netscape based browsers. If IE doesn't quite understand something, like sloppy code or unsupported features, it attempts to do something anyway, instead of simply ignoring them, like Mozilla and Netscape based browsers. Unfortunately, this results in major headaches for CSS developers. Microsoft has attempted to help "dumb down" coding issues, resulting in more developers that write sloppy code, by having the browser anticipate the design. There are obvious problems with this concept. In my humble opinion, CSS code should be written, validated, and 100% compliant with the standards.

2

However, there are some developers that like to write their CSS code to work (albeit incorrectly) in Internet Explorer, then provide hacks for Mozilla, Opera, Netscape and Firefox browsers. The reason I don't like this approach of writing code for IE then implementing workarounds for the other browsers that tend to be more compliant, is that many of these coders will have to go running back to their code to change it once IE decides it's going to use the CSS the way it was designed: literally. And trust me, they will eventually; IE 7 is said to have done this much better. But just for argument's sake, I will show a few examples of the most common workarounds.

Note: using too many hacks will cause your code to break once IE 7 comes out. It seems that a few of the bugs that some of these hacks relate to have been fixed, as stated on MSDN's IE Blog. Even so, there will be a large number of non-compliant or quirky browsers out there that could still use them, as long as you are aware of the risk. During the writing of this article, Microsoft released a beta version of IE 7. In the next article, I will cover the CSS standards with which IE 7 is currently compliant. I have also gone back and made a note in each of the workarounds which have been said to be fixed in IE 7.

Cross Browser Issues: CSS Hacks Explained, Tips, Tricks and Fixes - The Use of JavaScript and CSS

There have been some coders that have found a way to incorporate JavaScript into CSS in order to avoid using hacks for browser compatibility. While this may sound great, there are still several browsers that don't support JavaScript in the CSS files, like Firefox, Opera, and Netscape. Fortunately, most of those browsers don't need to support the JavaScript in the CSS, since the modern versions of them handle CSS rather well. It's usually IE you have to worry about, but you are still going to run into a problem for those who disable JavaScript in their browsers.

Still, the following example demonstrates the use of JavaScript in an external style sheet:

```
document.tags.H1.color = "blue"
document.tags.H1.fontSize = "20pt";

document.tags.H2.textAlign = "left";
```

This method isn't new, however. It is really a JavaScript Style Sheet, or JSSS, which was actually submitted to the W3C in 1996 by Netscape as a method of offering an alternative to CSS, which at that time wasn't widely accepted. It has severe limitations as far as positioning, but other features like dynamic calculations and conditional processing could be pretty powerful. JSSS is no longer supported in Netscape 6 and higher. JSSS style sheets may also seem less friendly than CSS style sheets to users without programming backgrounds.

3

Multiple Style Sheets for Multiple Browsers

While it may be your choice to only offer a single style sheet for all browsers, you may decide you wish to serve a separate style sheet for each type of browser, especially due to the vast differences in the way browsers support CSS. In this case, you'll need a way to detect the visitor's browser type and version, and then show the correct style sheet based on the results.

A simple JavaScript script can detect a browser type and version, and then choose the style sheet that was written for that browser:

4

```
<script language="JavaScript"><!--  
  
version=parseInt(navigator.appVersion);  
  
if (navigator.appVersion.indexOf('5.')>-1){  
  
version=5  
  
};  
  
if (navigator.appVersion.indexOf('6.')>-1){  
  
version=6  
  
};  
  
if (navigator.appVersion.indexOf('7.')>-1){  
  
version=7  
  
};  
  
browser='OTHER';  
  
if (navigator.appName=='Netscape'){  
  
browser='NS'+version;  
  
}  
  
if (navigator.appName=='Microsoft Internet Explorer'){  
  
browser='MSIE'+version;  
  
}  
  
if (navigator.appVersion.indexOf('MSIE 4')>0) {  
  
browser='MSIE4';  
  
}  
  
if(browser == 'NS5'){  
  
browser='NS6'  
  
};  
  
//Internet Explorer  
  
if (browser=='MSIE4') {  
  
document.write('<link rel="stylesheet" type="text/css"
```

```
href="/css/msie4.css">');  
  
}  
  
if (browser=='MSIE5') {  
  
document.write('<link rel="stylesheet" type="text/css"  
href="/css/msie5.css">');  
  
}  
  
if (browser=='MSIE6') {  
  
document.write('<link rel="stylesheet" type="text/css"  
href="/css/msie6.css">');  
  
}  
  
//Netscape  
  
if (browser=='NS4') {  
  
document.write('<link rel="stylesheet" type="text/css"  
href="/css/nn4.css">');  
  
}  
  
if (browser=='NS6') {  
  
document.write('<link rel="stylesheet" type="text/css"  
href="/css/nn6.css">');  
  
}  
  
if (browser=='NS7') {  
  
document.write('<link rel="stylesheet" type="text/css"  
href="/css/nn7.css">');  
  
}  
  
//everyone else  
  
if (browser=='OTHER') {  
  
document.write('<link rel="stylesheet" type="text/css"  
href="/css/other.css">');  
  
}  
  
// --></script>
```

Obviously, you will have to create a separate style sheet for each browser in your script.

Cross Browser Issues: CSS Hacks Explained, Tips, Tricks and Fixes - Hiding CSS from Buggy Browsers

It may be in your best interest for several reasons, to simply hide CSS files from certain browsers. One reason to do this would be that providing browser support for many browsers takes a lot of time, especially if you change your site often.

There are a few hacks to enable you to do this. Here is one that basically allows you to place the CSS code that all browsers understand in your current style sheet, yet hide the more complex, separate style sheet from the browsers that don't understand the directives contained in it, particularly IE 3 and Netscape 3:

```
/* CSS style sheet - hide from IE3 and Netscape 3 */
```

```
@import url(compliant_browsers.css);
```

Because the chances are slim that people are still using these browsers anymore, you will probably want something that deals with newer browsers:

```
/* CSS style sheet - hide from IE4 */
```

```
@import "compliant_browsers.css";
```

Or, for hiding individual properties that render oddly in IE 4 and 5:

```
color/*This comment is to hide this property from IE4 and IE5 */: red
```

Once you've decided whether to use a single style sheet for your CSS design purposes, or particular ones for each browser type, or to simply hide them from outdated browsers that don't support CSS, it's time to look at what the workarounds may be for the browser issues we talked about.

Cross Browser Issues: CSS Hacks Explained, Tips, Tricks and Fixes - Workarounds for Internet Explorer

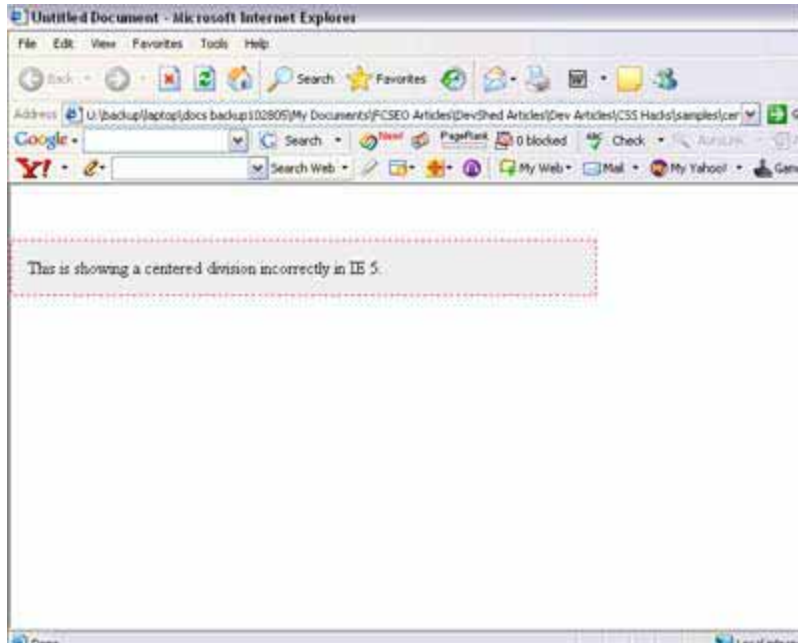
I use the term CSS hacks and workarounds interchangeably, because the word "hack" almost brings a negative connotation to mind. But in essence, many of these hacks are simply patches, which is why I use the term this way. Keep in mind, however, that some of the more laborious hacks may not allow your CSS files to validate, especially the ones that utilize some JavaScript. Below will be some common problems experienced in Internet Explorer and their hacks. This, of course, is not an all-inclusive guide, and will probably miss a few you like, so please forgive this ahead of time.

Centering: The Auto-Width Margin Hack (IE 5 for Windows)

A box can be horizontally centered with CSS by setting its right and left margin widths to "auto." This is the preferred way to accomplish horizontal centering with CSS, and works very well in most browsers with CSS2 support; IE 5 for Windows has problems with this. NOTE: This IE bug is fixed in IE 7.

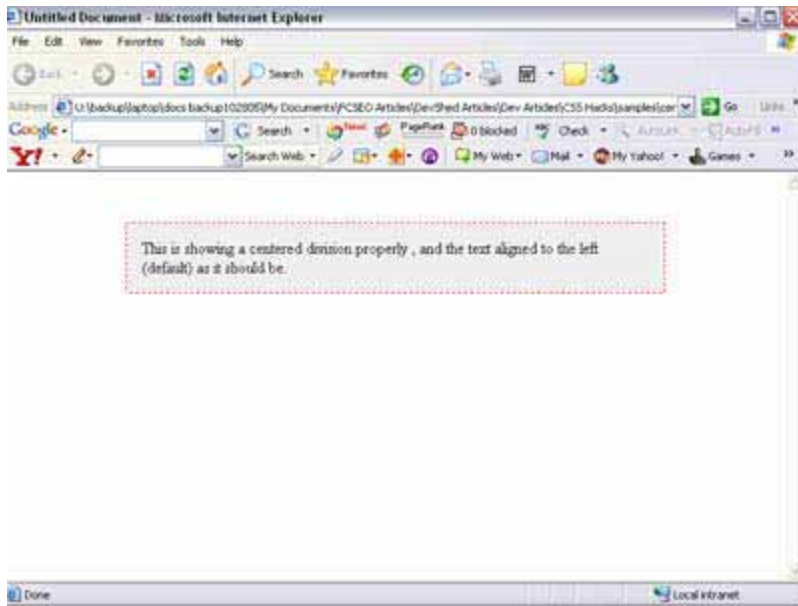
IE 5 incorrectly applies the CSS "text-align" attribute to block-level elements. Using "text-align:center" for the containing block-level element, sometimes the BODY element, horizontally centers the box in IE5/Win. But there is a side effect of this workaround: the CSS "text-align" attribute is inherited, centering all inline content. It is often necessary to then set the "text-align" attribute for the centered box, counteracting the effects of the workaround. Here's an example:

Before the hack:



After the hack:

7



The CSS fix (this CSS will validate.):

```
<style>

body {

    margin:50px 0px; padding:0px;

    text-align:center;

}

#Content {

    width:500px;

    margin:0px auto;

    text-align:left;

    padding:15px;

    border:1px dashed #FF0000;

    background-color:#eee;

}

</style>
```

Double Float Margin Hack (all versions of IE)

You want to put a left float into a container box, and use a left margin on the float to push it away from the left side of the container. In IE, the left float margin is doubled.

```
.floatbox {

float: left;

width: 150px;

height: 150px;

margin: 5px 0 5px 100px;

}
```

So how do you fix it? You could just not use floats in a container, or you can use a workaround for IE.

```
.floatbox {  
  
float: left;  
  
width: 150px;  
  
height: 150px;  
  
margin: 5px 0 5px 100px;  
  
display: inline;  
  
}
```

Floats automatically become "block" elements, no matter what they were before becoming floats. By using the "display: inline;" directive, it somehow triggers IE to stop doubling the float's margin. (This also works on the weird text indent bug.) Because the standards for display on a float say this should be ignored, all browsers show no changes in the float when this is done, including IE. Further, the CSS will validate. NOTE: This IE bug is said to be fixed in IE 7.

Cross Browser Issues: CSS Hacks Explained, Tips, Tricks and Fixes - More Workarounds for Internet Explorer

9

Min-Width and Max-Width Hacks (all versions of IE)

IE is the only browser that doesn't support the CSS2 properties min-width and max-width, which have been a standard since 1997. These are useful for controlling larger areas of type, particularly when setting line length limits to aid in readability.

IE treats width as if it were min-width. By exploiting the fact that IE doesn't support child selectors, we can create some CSS code that doesn't allow a browser window to shrink below a certain size:

```
body {  
  
width: 800px;  
  
}  
  
html>body {  
  
width: auto;  
  
min-width: 800px;  
  
}
```

The max-width hack does require using some JavaScript in the CSS, which will cause your CSS not to validate.

```
width:expression(some javascript goes here);
```

Here's an example of its use:

```
border:1px solid red;
width:expression(
  document.body.clientWidth > (500/12) *
  parseInt(document.body.currentStyle.fontSize)?
  "30em":
  "auto" );
```

The key is using Internet Explorer's relatively little known `expression()` property. What this does is make IE check whether a value is over a certain threshold. If it is, then it makes IE use 30em as the box width; if not, auto is used, making the box shrink. Fortunately, Mozilla, Firefox, Opera, Netscape 6 and other compatible browsers just ignore this because, when a browser that isn't IE comes across something it doesn't understand, it simply disregards it.

Utilizing the Asterisk HTML Selector Bug (all versions of IE)

The following selectors are all incorrectly interpreted by Internet Explorer as if the first universal selector (asterisk) does not exist:

```
* html /* basic form */
* * body /* white space between asterisks is significant */
* html body /* higher specificity than either of above */
```

IE treats the above as if they were, correspondingly:

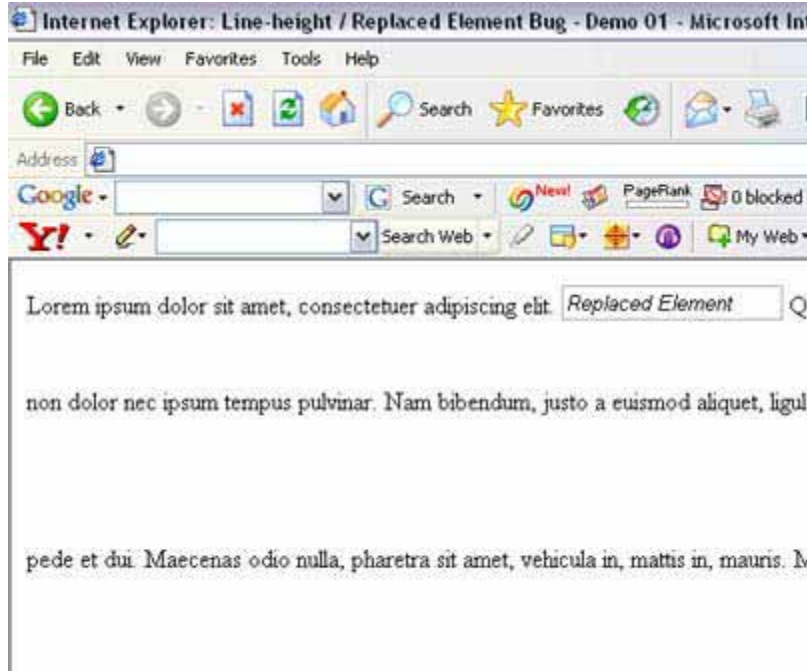
```
html
* body
html body
```

In valid HTML and XHTML documents, `html` is always the root element and `body` is always a child of the root element, and never a grandchild (or great grandchild). Therefore, the first three CSS selectors above should not match any element. Nonetheless, they are valid selectors. By using any of the above erroneous selectors we can specify CSS rules that are meant only for IE.

Line-Height and Replaced Element Hack (all versions of IE)

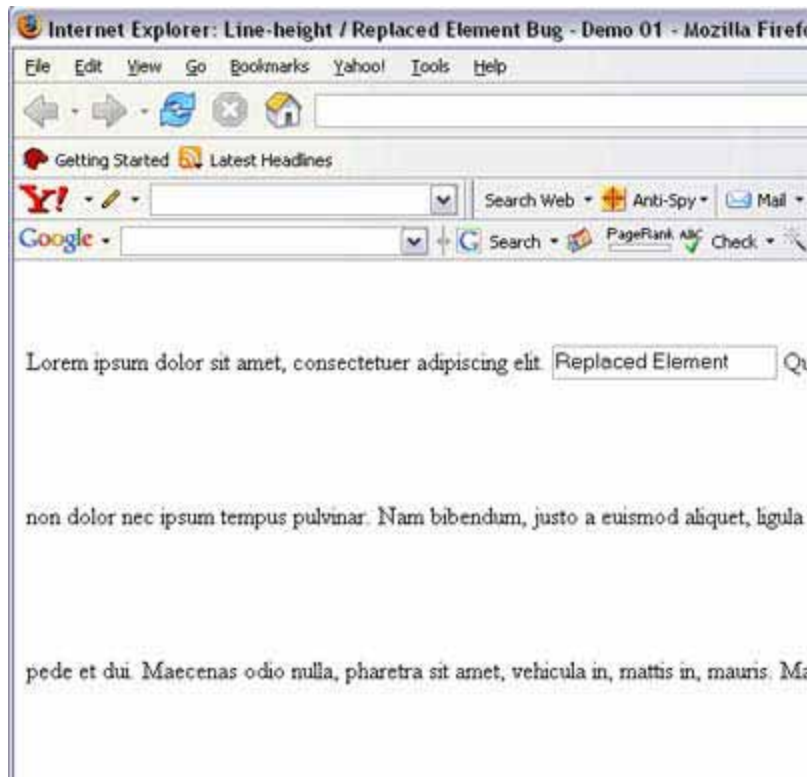
In IE, line-height is incorrectly rendered if the line contains a replaced element. If the default line-height is changed to a larger value, the line spacing above and below any text line that contains a replaced element, such as a small image, will be half of the spacing seen elsewhere in the text. This bug is triggered just by the simple presence of any replaced element: `IMG`, `INPUT`, `TEXTAREA`, `SELECT` and `OBJECT`.

Line-height in IE:



11

Line-height in Firefox:



By calculating and applying a top and bottom margin to the replaced element, you can force the line-height to be what was intended. Utilize a hack that gives specific instructions to IE, like the asterisk hack above, and then apply your top and bottom margin fixes. You will need to know the height of the replaced element, and the height of all the replaced elements of the same type needs to be almost equal if you want to make use of the same CSS classes, otherwise, you'll have to create new classes for each replaced element.

```
* html img {  
  
margin: 45px 0;  
  
vertical-align: middle;
```

The vertical-align property is needed to position it correctly. NOTE: This IE bug is supposed to be fixed in IE 7.

Workarounds for Mozilla, Firefox, Netscape and Opera

Generally, you aren't going to see too many hacks for these types of browsers, unless you are talking about older versions of Netscape and Opera, for the sheer fact that they support CSS very well; why write hacks for all of the browsers instead of the one that's erring? But you have a lot of people that use IE, and the school of thought here is to try and look good to the majority of the public when the cross-browser compatibility issues emerge, then go back later and put in the hack to "fix" the other minority browsers.

I don't really feel I need to go into any fixes for the compliant browsers, just so that your code in IE works in everything else; I believe this is the absolute incorrect approach. It not only makes things more difficult, but the likelihood of your code breaking in later versions of IE is extremely high.

Writing compliant CSS code is always a good idea; the simpler the better. Once in a while, you will want to use a workaround to get your design to look right in IE, and hopefully you'll be able to use these for those times. There is no excuse for writing sloppy code, but then again, there really is no excuse for IE not to be compliant with the CSS standards. A great philosopher once said, "You can't always get what you want; but sometimes you can get what you need." Workarounds might not be what you want, but you can sometimes get what you need.