



## Clearing Floats, the Easy Way

By: [John Gallant](#) , [Holly Bergevin](#) ,

This article deals with floats, so if you don't already understand the basic concepts, you may bone up by reading [Float: The Theory](http://www.communitymx.com/abstract.cfm?cid=A46708877E1AC008) (<http://www.communitymx.com/abstract.cfm?cid=A46708877E1AC008>), which will familiarize you with the flexible, fun, and frustrating **float**.

### The Dirty Little Secret Of Floating

One of the biggest problems people have when learning CSS is understanding how floats really work, and why they seem to work differently when viewed in Internet Explorer for Windows versus other browsers. While many of the variances among browsers are indeed just variances, some are actual browser bugs. Sometimes though, the problem stems from a failure on the part of the coder to **clear** his or her contained floats.

While more people are beginning to use floats in their layouts, "**float clearing**" is still a mysterious and misunderstood subject. Remember, floats are not actually contained within a block box, even though they may be nested there in the source code. While the float's screen starting point is determined by its nested location, after that it just drapes over the containing box, as do absolutely positioned elements. Only line boxes containing inline elements (like text) will care where the float is displayed.

When a float is nested within a container box that has a visible border or background, that float does *not* automatically force the container's bottom edge down as the float is made taller. Instead the float is ignored by the container, and the float will hang down out of the container bottom like a flag if the float is tall enough.

### How Float Clearing Really Works

Those familiar only with Explorer for Windows may scratch their heads and say, "That's not right!" True, IE/Win does enclose a float within a container "automatically," but only if the container element has a stated dimension. Unfortunately, this is a violation of the [W3C float specification](http://www.w3.org/TR/CSS21/visuren.html#floats) (<http://www.w3.org/TR/CSS21/visuren.html#floats>). This improper behavior can also be "toggled" on and off by links within the container in IE, if "hovering" alters the link background, or by several other properties.

In other browsers, and in IE when the container has no dimension, the standard method of making an outer container appear to "enclose" a nested float is to place a **clearing element** inside the container, and after all other content there.

```

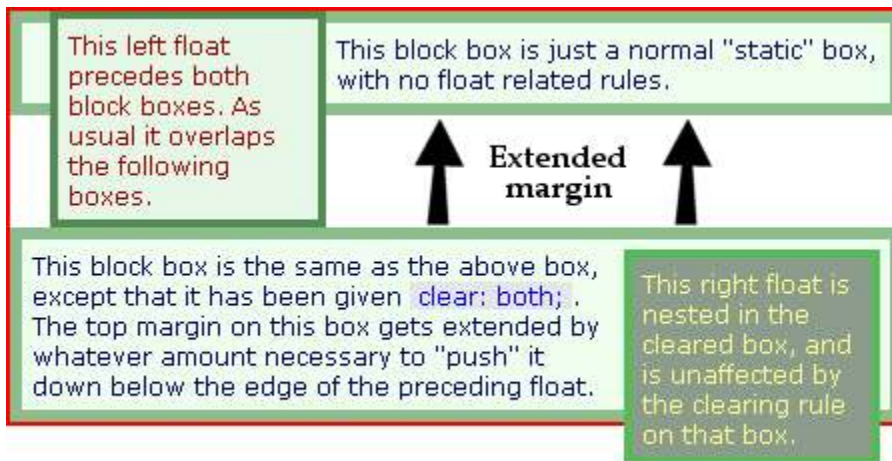
<div> <!-- float container -->
  <div style="float:left; width:30%;">
    <p>
      Some content which will be taller
      than the content outside the float
    </p>
  </div>
  <p>
    A paragraph in the float
    container outside the float
  </p>
  <div style="clear:both;"></div> <!-- Here is the clearing element -->
</div>

```

### Code Block 1

Since that **clearing div** is not floated, the container must recognize it and enclose it. The **clearing div** "pulls" the bottom edge of the container down below the bottom edge of the float.

The method shown in **Code Block 1** uses an otherwise wasted div just to make the float look like it is inside the container border. Of course it is perfectly allowable to have actual content inside the cleared element, as shown in the following graphic from the [Float article](http://www.communitymx.com/abstract.cfm?cid=A46708877E1AC008) (<http://www.communitymx.com/abstract.cfm?cid=A46708877E1AC008>):



**Image 1:** How a box may clear below a float

The **red border** represents the **containing box**, within which are nested all the other boxes. The float with the red text in the upper left would normally protrude out the bottom of the red bordered container, but it can't because the lower full-width box is not floated and has been given `{clear: both;}`. This automatically extends the top margin on the cleared element by just the amount needed to keep it below the upper float. Since the cleared box is not itself a float, it must be kept inside the red border of the container box, effectively "pulling" the red border down around the previous float. So in effect, such a cleared box cannot be at the same horizontal level as a preceding float. It must appear just below that level.

Notice how the float that is nested within the lower cleared box in the graphic above *does* protrude outside the container? That is because *that nested float does not come before the cleared box*. A cleared box is only required to stay below **previous** floats, and that nested float does not qualify as previous to its cleared parent container.

## "You're saying we are *forced* to use an element just to clear a float?"

Yes, that is what we are saying you have to do, if you want the container to appear to enclose the nested float. This concept is pretty darned tricky, and it's no wonder so many new coders fail to properly understand it.

**Absolutely positioned** elements may also escape their containers like this, but at least the display is consistent across browsers if this happens. Not so with floats. Because IE/Win happens to "auto-clear" contained floats when the containing box has either a width or height, many coders don't realize their floats need a cleared element until they check in a non-IE browser and find that something appears to be wrong. Then they usually blame the non-IE browser for "breaking" their layout.

The "IE-wannabe" browser, **Opera 7**, will also enclose a float when the container is dimensioned, but that fact does not adversely impact any of the following arguments so it may be safely ignored, except for optional rueful head shaking.

## A New Way To Clear Floats

It so happens that the W3C has provided us with a [CSS 2 method](http://www.w3.org/TR/REC-CSS2/generate.html#x5) (<http://www.w3.org/TR/REC-CSS2/generate.html#x5>) to add content either at the start or at the end of an element, without actually having that content appear in the source document. These are the **:before** and **:after** pseudo-elements, and the syntax is as follows:

```
div:after {content: "This text comes last in the element";}
```

### Code Block 2

As shown in **Code Block 2**, the text string value given in quotes for the **content** property causes that text string to be appended at the end of any div within the source document. Typically you won't want the text string to appear at the end of every div, so the "div" in that code block needs to be replaced by something more selective, such as a class name.

Such **:after generated content** cannot be affected by certain CSS properties, including **position**, **float**, **list** properties, and **table** properties. However, the **clear** property *will* affect this **CSS generated content**. Are you beginning to see where we are going with this?

## Generated Clearing

Imagine that we use **:after** to insert a simple character like a "period," and then give that **generated element** **{clear: both;}**. Will that cleared period serve to clear any previous float, just as if it were a normal cleared element? Yes, indeed, it sure will! Isn't that interesting? CSS allows you to clear a float container directly from the style sheet. This means that the extra clearing element we've had to add to our HTML up to now is no longer necessary!

That period is all you really need to do the clearing. But no one wants an odd character messing up the end of their clean container box, so we also use **{height: 0;}** and **{visibility: hidden;}** to keep our period from showing. Just tie these rules to a class name like **clearfix**, and that class becomes a magic wand you can apply to any container that might need clearing.

#### The HTML:

```
<div class="clearfix"> <!-- float container -->
  <div style="float:left; width:30%;">
    <p>
      Some content which will be taller
      than the content outside the float
    </p>
  </div>
  <p>
    A paragraph in the float
    container outside the float
  </p>
</div>
```

#### The CSS:

```
.clearfix:after {
  content: "."; /* This dot is the actual appended content */
  clear: both;
  height: 0;
  visibility: hidden;
  display: block;
}
```

### Code Block 3

Notice that `{display: block;}` is also applied to the `:after` pseudo-element. If this isn't done, then that element defaults to `{display: inline;}`, and the generated element cannot then receive the `clear` property.

### Does Explorer Support This Pseudo-element Business?

Well, no it does not. But remember that Explorer does "auto-enclose" a float if the container is given a width or height, so we can take advantage of that bug to bring IE into the picture along with the other browsers.

In many cases it is undesirable to use either a height or a width on the container, but happily the **Holly hack** comes to our rescue. This hack lets IE/Win, and *only* IE/Win, see a simple 1% height for the container. How does that help? Well, IE/Win happens to have another specification violation that causes all boxes to expand and enclose all content, regardless of any stated dimensions that may be smaller! So, that `{height: 1%;}` will just be ignored by IE, and the box will be expanded to contain whatever content may be nested inside. Meanwhile, the mere act of applying a dimension to the container will trigger the auto-float-enclosing behavior for IE/Win anyway. Very nice.

#### The Holly Hack

```
/* Hides from IE-mac */
* html .clearfix {height: 1%;}
/* End hide from IE-mac */
```

### Code Block 4

For all the details on the Holly hack, read this page in [How To Attack An Internet Explorer \(Win\)](#)

[Display Bug \(http://www.communitymx.com/content/article.cfm?page=2&cid=C37E0\)](http://www.communitymx.com/content/article.cfm?page=2&cid=C37E0) here at CMX.

As a side benefit, the Holly hack also prevents several other major IE/Win float bugs. However, should this container box be placed following a previous external float, the IE dimensional fix will trigger Microsoft's proprietary and improper [Float Model \(http://www.communitymx.com/abstract.cfm?cid=AF85B1B881347F29\)](http://www.communitymx.com/abstract.cfm?cid=AF85B1B881347F29), so watch out for that.

## A Word About Floats In Floats

You may have noticed that modern browsers always let float containers enclose other floats, without need for a clearing element. This is true for all version 5 browsers and up, although IE/Mac may possibly have problems with floats inside floats. An upcoming article will discuss how to work around the IE/Mac float issues that cause the most problems, so stay tuned.

## A Small IE Toggling Problem

When IE/Win auto-encloses a float within a box that has a dimension, the links inside may cause the buggy enclosing effect to "toggle" on and off if `a:hover` is used to change the background or several other properties. If this toggling behavior happens in your layout, it can be easily corrected simply by wrapping the problem links with containing elements that have had the Holly hack fix applied to them as well. This will induce the IE float model problem, but assuming that the link container is allowed to remain transparent, the actual appearance will not be any different.

Any block element can serve as a wrapper for those buggy links. Yes, this means another element may possibly be needed, but unlike a clearing div, such links really ought to have an inner "semantic" wrapper element such as a paragraph or list. Just remember that when using the `.clearfix` class on a float container, always use the Holly hack fix on an inner block element wrapper around your links, and you will be fine. We'll see how this can be done in the next code example.

To see the IE float toggle and learn more about it, go to the [Guillotine Bug demo page \(http://www.communitymx.com/content/article.cfm?page=4&cid=A47F91175DB7863B\)](http://www.communitymx.com/content/article.cfm?page=4&cid=A47F91175DB7863B) here at CMX.

## Putting It Together

So far we have CSS rules that provide clearing for non-IE browsers, and a bug induced fix that takes care of Explorer. Here is the code up to this point. We've switched the HTML to match the link example just discussed:

#### The HTML:

```
<div class="clearfix">
  <div style="float:left; width:30%;">
    <ul>
      <li><a href="#">link</a></li>
      <li><a href="#">link</a></li>
    </ul>
  </div>
  <p><a href="#">link</a></p>
</div>
```

#### The CSS:

```
.clearfix:after {
  content: "."; /* This dot is the actual appended content */
  clear: both;
  height: 0;
  visibility: hidden;
  display: block;
}

/* Hides from IE-mac */
* html .clearfix {height: 1%;}
/* End hide from IE-mac */
```

### Code Block 5

Note that the Holly hack has a second selector group after the comma, which is exactly the same except for the additional star on the right end. This **universal selector** causes the Holly hack to be applied to any element within a container that has a class of **clearfix**, while the first selector still applies the hack to the container itself. This second selector group is only there to guard against IE link toggling, and can be omitted if toggling is not a problem.

## IE5/Mac Strikes Back

All this is wonderful, but unfortunately Internet Explorer for the Mac does not "auto-enclose" floats as IE/Win does, nor does it support the **:after** pseudo-element, and so is left out of the clearing party. What is to be done?

### Taming the IE5/Mac Float Clearing Problem

Basically, the fix for IE5/Mac is just a matter of applying **{display: inline-block;}** to the **.clearfix** class, and then hiding that property from all other browsers. That's it! We can easily do this with our existing code, only slightly modified.

```
.clearfix:after {
  content: "."; /* This dot is the actual appended content */
  clear: both;
  height: 0;
  visibility: hidden;
  display: block;
}

.clearfix {display: inline-block;} /* Fixes IE/Mac */

/* Hides from IE-mac */
* html .clearfix {height: 1%;}
.clearfix {display: block;}
/* End hide from IE-mac */
```

## Code Block 6

The `.clearfix {display: inline-block;}` is seen by all browsers and fixes IE5/Mac. Then, inside the rule set that is hidden from IE5/Mac, the display property is "reset" to `{display: block;}` for all browsers other than IE5/Mac. In this way everyone is kept happy.

## Conclusion

Using the new method is quite simple. Just stick the CSS code into your CSS file, and use `.clearfix` on any box that has to contain a sizable float. However, watch out for potential problems involving the [IE Float Model \(http://www.communitymx.com/abstract.cfm?cid=AF85B1B881347F29\)](http://www.communitymx.com/abstract.cfm?cid=AF85B1B881347F29), as mentioned earlier. If these or other bugs do arise and can't be easily worked around, the old **cleared element** method is always available for use.

We have created a separate demonstration page in the support files, to show this technique in action. Download the support zip to a folder and view the HTML file. The page contains two actual demos, one with no clearing, and the other with this clearing technique applied to it. Have fun!

## Web Credit

We would like to say that these neat tricks are our invention, but we can't. They have been developed by several different folks, each working to make the web a better place. [Tony Aslett \(http://www.csscreator.com/attributes/containedfloat.php\)](http://www.csscreator.com/attributes/containedfloat.php) started the ball rolling, and [Mark Hadley \(http://archivist.incutio.com/viewlist/css-discuss/46048\)](http://archivist.incutio.com/viewlist/css-discuss/46048) recently found the fix for IE5/Mac. Many others have contributed with testing and refinements, and now the results are yours to enjoy. The web can indeed be a great place when people are pulling in the same direction.

## Keywords

CSS, float, floats, float clearing, clearing element, containing floats, IE, float bugs, holly hack, pseudo-element, :after, :before, generated content, Guillotine bug, IE Mac

All content ©CommunityMX 2002-2006. All rights reserved.