

Do You Want To Do That With CSS? — Float Tricks

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

Float Tricks

So, you have read about Floats, and maybe you would like to try using them a little. The question is, what exactly can one *do* with floats? Sure, it's useful to float columns in a tableless layout, but that happens to be pretty advanced stuff, and the bugs can be very frustrating as well. However, floats can do things in a larger design that would be quite complicated to replicate using tables. It's just a matter of getting comfortable with the way floats work, and soon you will be floating all over the place, perhaps even inside good old table cells. It's fun!

In order to understand the following methods it's necessary to understand the basic rules that floats should follow. See [Float: The theory \(http://www.communitymx.com/abstract.cfm?cid=A46708877E1AC008\)](http://www.communitymx.com/abstract.cfm?cid=A46708877E1AC008) for a primer on floats.

Basic Floating

Authors commonly need to have small images appear inside a larger text block. The way this used to be done was to give the `` element an **align** attribute such as `align="right"`. Since **align** is now deprecated (but still valid in non-strict doctype pages), we use `{float: right;}` to do the same thing, but with the advantage that CSS floating can be used on any element and not just the limited number that can be "floated" using the align attribute. Note that floating is bi-directional and works either to the left or right. You can have different floats that go different directions within the same containing element.



The froggy is the first element in the above paragraph, and is floated right. It has both bottom and left side margins to hold the text away from the float. Using margins on the float is the *only* way to keep text from bumping up against the float. The code for this looks like:

The HTML:

```
<p>
  
  Lots of boring text, bla bla bla...
</p>
```

The CSS:

```
img {
  float: right;
  margin: 0 0 20px 40px;
}
```

The **align** attribute was used both to "float" images (and a few other elements) and also to align text within elements. However, you could not both float and alter the text alignment at the same time. To simplify the situation, the deprecated align function was split into two CSS properties, namely **float**, and **text-align**, which are applied, respectively, to either a floated element or an element containing text needing to be aligned (including floated elements). Goodbye, one confusing, dual-purpose attribute!

Floating out of the Middle

When you insert a floated element into the middle of a block of text, several things must be kept in mind. First, a float will start on a **new line**, so if your float is placed in the middle of the text, the float's upper edge will appear on the next line down.

That is **not** what the W3C float specifications say should happen, but that **is** what happens in both Explorer and Mozilla. The W3C requires that the top edge of a float be aligned with the top of the current line box, unless there isn't enough room for the float to display on that line, in which case it is shifted down. Inline content on the same line that the float occurs is to be shifted left or right depending on the float direction, with no indication that it occurs before or after the float. But only Opera 7 strictly obeys the rule. This browser disagreement is usually not very harmful to a layout, but someday such display variance could cause you difficulties, so keep it in mind.

All the modern browsers are capable of filling up a short text line above the float with "borrowed" text that comes after the float. This is as it should be, even with the incorrect implementations found in IE and Mozilla, since inline content should be "reflowed in the first available line on the other side of the float." (9.5 Floats (<http://www.w3.org/TR/CSS21/visuren.html#floats>)) Despite the fact that (currently) IE and

This live demo div (black bordered box) contains a paragraph with ordinary text. Right in the middle of all this text is an image that has been floated to the left. The floated element is actually located, **in the source**, between the following pair of square bracket characters. [] Now that the float has



come and gone, the text continues, running down along the side of the float. Note how the text appears to always complete a full line above the float, no matter what the browser width is. Once the text passes the bottom of the float it is able to again occupy the full width of the paragraph. By the way, this entire demo div has itself been floated to the right, proving that nested floats are no problem.

Mozilla drop floats to the next line before reflowing inline content to the proper side, they still fill potentially empty inline spaces as though they had followed the specifications correctly. This definitely helps prevent unsightly half-lines of text from appearing just above the float. Also be aware that floated images never **require** a `width` to be used because they have **intrinsic dimensions** that the browser can use to define the size of the float. This does not preclude the use of a width, however.

Try dragging this article's window width wider and narrower to see how the froggy stays below the brackets (or beside them in Opera), while some text after the bracketed float is often "borrowed" and placed above the float to fill out the pre-float line. Neat, eh? Here's the code:

The HTML:

```
<p>
  Lots of boring text, bla bla bla...
  []
  Lots more boring text, bla bla bla...
</p>
```

The CSS:

```
img {
  float: left;
  margin-right: 5px;
}
```

If the float were placed at the start of the paragraph, like with the first frog, it would naturally be in the upper left or right corner of the paragraph. Sadly, you cannot float anything into a lower corner, although careful construction can make it look that way. The rules of floating do allow you to locate the float *before* the start of the paragraph and still have the float seem to be inside the paragraph. But, keeping the float within the paragraph element (or div, or list item, or blockquote, etc.) will greatly reduce the chances of encountering an Explorer float bug.

Speaking of IE/Win float bugs, they usually occur when two CSS conditions are present. The primary **bug trigger** is the lack of any stated `width` or `height` on the parent element containing the float. This alone can cause bugs, but matters get much worse when a "cleared" element following the float comes in direct contact with the bottom of the float. Apparently, when IE/Win must activate its "clearing mechanism" it can't do the job correctly unless it is aware of at least one dimension applied to the surrounding container element. If you are not sure what "clearing" is, please read [Float: The Theory](http://www.communitymx.com/abstract.cfm?cid=A46708877E1AC008) (<http://www.communitymx.com/abstract.cfm?cid=A46708877E1AC008>) for the complete explanation.

Since it's not always possible to stop a cleared element from contacting a float bottom (that *is* what cleared elements are for, after all), it means that the containing element should have a dimension to insure against bugs in Explorer for Windows. This can easily be done using the Holly hack, as explained in the article [How To Attack An Internet Explorer Display Bug](http://www.communitymx.com/content/article.cfm?cid=C37E0) (<http://www.communitymx.com/content/article.cfm?cid=C37E0>). "Holly hacking" the element that directly surrounds the float will prevent the worst float bugs in IE/Win.

Due to the small float sizes and the large text blocks found in this article, no cleared elements were needed. Cleared elements are not required when floating unless some element following the float must be kept below the float on the screen.

Pull Quotes

When only images need to be floated, not much is gained by switching from "align" to "float." But, unlike aligning, the float property can be applied to any element on a page, even text related elements such as `` or ``. These **inline** elements, when floated, are automatically converted into **block** elements, so that all floats are *always* "block," just as paragraphs and divs are.

The W3C has in the past required a stated width on all floats, but it has become clear that letting the length of a text line determine a float's width is very useful for things like horizontal button rows. Therefore all modern browsers (other than IE/Mac) will "shrink-wrap" a float when that float has no stated width.

This means that a float containing text will be as wide as the longest line of text in the float. Normally, multiple text lines in a float will be roughly equal in length, but in the case of lines shortened by (hard coded) line breaks, the longest of the lines determines the float width. Most modern browsers have no problem with the display of "widthless" floats. However, IE5/Mac makes widthless floats as wide as their parent container, potentially causing layout problems.

For example, consider those floated menu buttons in a horizontal navigation bar. Not declaring widths on these multiple floats should result in side by side buttons that are only as large as they need to be for the text and padding they enclose. This is the case in most modern browsers, but IE5/Mac makes these widthless floats as wide as possible, and stacks them vertically instead, making necessary either layout changes, or hacked accommodations for that browser. So, if possible, it's wise to define a width on any float lacking an intrinsic width, to avoid problems in IE5/Mac.

So, let's try floating a **pull quote** within a paragraph, just like the glossy magazines do it. We'll pull our quote right out of this very paragraph! Everybody has seen how magazines often make their quotes protrude to one side of the main text block, so let's do that by right floating the quote and then applying a **negative right margin** to the float. This has the effect of dragging that pull quote out to the right just, a little beyond the rest of the text. The pull quote could just as easily be put at one of the top corners of the paragraph, but we think it looks more "authentic" to have it part way down in the paragraph.

“We'll pull our quote right out of this very paragraph!”

The containing paragraph has been padded on the right side in order to create the space into which we pulled the span. The right padding on the paragraph is not necessary for the pull quote effect, but was used in order to avoid having the span extent out of the article page.

We don't have to stop with placement alone. The float is a separate element, so it can be styled differently than the paragraph proper. The text for our pull quote has been copied and pasted into a **span** which has then been **float**ed **right**. Just for fun, we've given it a larger font-size, italics, and even a different font-family and font color. With a little imagination, the pull quote could be made to appear in a variety of styles and positions. Here's the code for the above "pull quote":

The HTML:

```
<p>
  Lots of boring text, including the line to be quoted...
  <span>"Our wonderful pull quote!"</span>
  Lots more boring text, bla bla bla...
</p>
```

The CSS:

```
p {
  padding-right: 120px;
}

span {
  float: right;
  width: 50%;
  margin: 10px -80px 5px 20px;
  color: #a00;
  font-family: Georgia, serif;
  font-size: 1.7em;
  font-style: italic;
  line-height: 1em;
}
```

The reason we floated a span rather than a div or another paragraph is that paragraphs are not supposed to have divs or paragraphs inside them, and we want our code to be valid, don't we?

In our example, the span stayed within the paragraph itself, but if it had been negatively margined any further to the right it would have protruded beyond the paragraph, and it would appear to be **clipped off** in IE/Win!

In this paragraph, we have placed a red-bordered span with the text "**IE/Win negative margining bug.**" It is negatively margined a bit too far to the right. This causes it to extend beyond the containing paragraph where it is clipped in IE/Win. This **IE/Win negative margining bug** can be defeated by applying `{position: relative;}` to the float. Strangely, that seems to do the trick! Just remember that if you do this with a floated image, you also have to make the parent container "relative," or the image may be hidden behind any background that happens to be placed on the parent. Sigh...

IE/Win negative margining bug

One final IE/win bug must be mentioned before we go (yup, another one). When a float is margined so that it is pushed away from the inner side of its parent container, IE/Win likes to *double the length of the margin*. This used to be a serious problem, requiring a hack to feed a special margin value only to IE/Win. Isn't that special?

Well, we now know that this unhappy little doubling bug can be squelched simply by applying `{display: inline;}` to the float! If you'll recall, we did say that all floats are automatically made "block," so trying to make a float "inline" is at the very least a bit silly. The `{display: inline;}` rule must be ignored by all browsers, and sure enough it is ignored, even by IE/Win, but somehow, just *trying* to make the float "inline" actually fixes the doubling bug! Is this a weird browser or what? (For a more complete discussion

of this bug, see page two of

[Common Coding Problems — Part Five \(http://www.communitymx.com/abstract.cfm?cid=8AD92\)](http://www.communitymx.com/abstract.cfm?cid=8AD92)

By the way, the right-floated pull quote demos above use negative right margins which do touch the right side of the paragraphs, but the margin values *have not been doubled*. Compare IE to some other browser, and you'll see that the spacing is exactly the same. What gives? Well, since the float started out as an inline **span** element, that fact is (apparently) just as effective as adding `{display: inline;}` when it comes to fixing the bug. Thus, it appears that no floated inline element should ever be affected by the doubled margin bug.

That's it for now. Next time we will look at some more ways to usefully use floats in real pages, such as multiple floats and the effective use of clearing elements. Ciao!

Approximate download size: 175k

Keywords

CSS, float, pull quote, images, align attribute, IE float bug, negative margining bug, doubled float margin bug, HTML

All content ©CommunityMX 2002-2004. All rights reserved.