SMASHING MAGAZINE

The Mystery Of The CSS Float Property

By Louis Lazaris

Published on October 19th, 2009 in CSS, Essentials with 101 Comments

Years ago, when developers first started to make the transition to HTML layouts without tables, one **CSS** property that suddenly took on a very important role was the `float` property. The reason that the `float` property became so common was that, by default, block-level elements will not line up beside one another in a column-based format. Since columns are necessary in virtually every CSS layout, this property started to get used — and even overused — prolifically.

The **CSS `float` property** allows a developer to incorporate table-like columns in an HTML layout without the use of tables. If it were not for the CSS `float` property, CSS layouts would not be possible except using absolute and relative positioning — which would be messy and would make the layout unmaintainable.

In this article, **we'll discuss exactly what the `float` property is and how it affects elements** in particular contexts. We'll also take a look at some of the differences that can occur in connection with this property in the most commonly-used browsers. Finally, we'll showcase a few practical uses for the CSS `float` property. This should provide a well-rounded and thorough discussion of this property and its impact on CSS development.

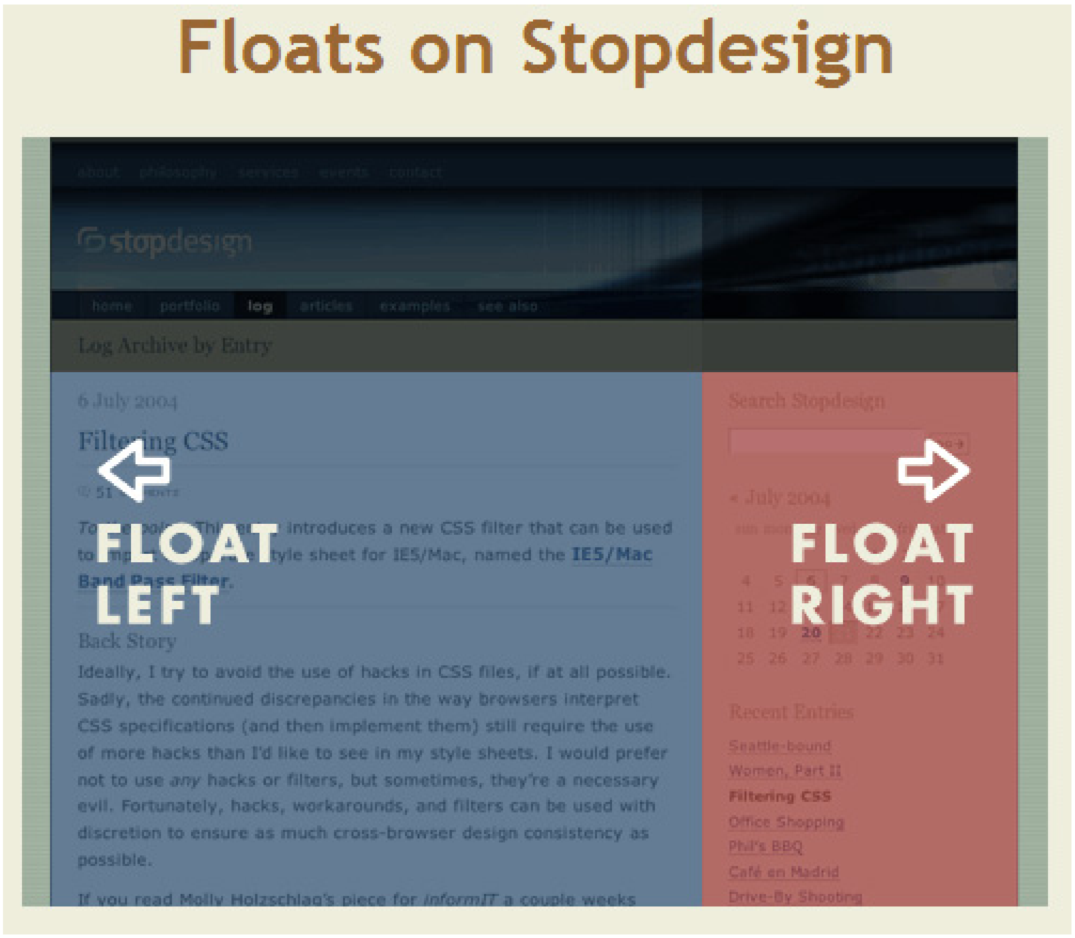
Definition and Syntax

The purpose of the CSS `float` property is, generally speaking, to push a block-level element to the left or right, taking it out of the flow in relation to other block elements. This allows naturally-flowing content to wrap around the floated element. This concept is similar to what you see every day in print literature, where photos and other graphic elements are aligned to one side while other content (usually text) flows naturally around the left- or right-aligned element.

Flickr photo by presentday[1]

The photo above shows the "Readers' sites" section of .net magazine[2], which displays 3 readers' photos each aligned left in their respective columns with text wrapping around the aligned images. That is the basic idea behind the `float` property in CSS layouts, and demonstrates one of the ways it has been used in table-less design.

The effectiveness of using floats in multi-columned layouts was explained by Douglas Bowman in 2004 in his classic presentation No More Tables[3]:



4

No More Tables[5]

Bowman explained the principles behind table-less design, using Microsoft's old site as a case study. Floats were used prominently in his mock overhaul of the Microsoft layout.

SYNTAX

The `float` CSS property can accept one of 4 values: `left`, `right`, `none`, and `inherit`. It is declared as shown in the code below.

```
#sidebar {
  float: left;
}
```

The most commonly-used values are `left` and `right`. A value of `none` is the default, or initial `float` value for any element in an HTML page. The value `inherit`, which can be applied to nearly any CSS property, does not work in Internet Explorer versions up to and including 7.

The `float` property does not require the application of any other property on a CSS element for `float` to function correctly, however, `float` will work more effectively under specific circumstances.

Generally, a floated element **should have an explicitly set width** (unless it is a replaced element[6], like an image). This ensures that the float behaves as expected and helps to avoid issues in certain browsers.

```
#sidebar {
  float: left;
  width: 350px;
}
```

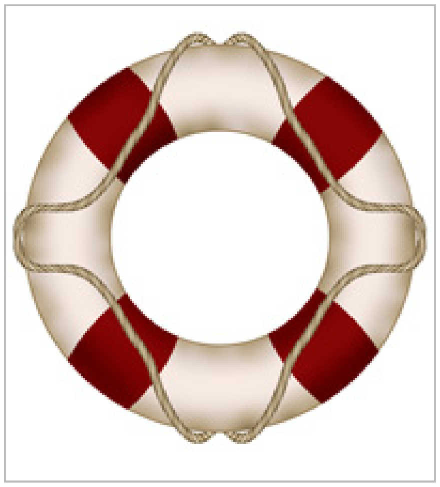
SPECIFICS ON FLOATED ELEMENTS

Following is a list of exact behaviours of floated elements according to CSS2 Specifications[7]:

- A left-floated box will shift to the left until its leftmost margin edge (or border edge if margins are absent) touches either the edge of the containing block, or the edge of another floated box
- If the size of the floated box exceeds the available horizontal space, the floated box will be shifted down
- Non-positioned, non-floated, block-level elements act as if the floated element is not there, since the floated element is out of flow in relation to other block elements
- Margins of floated boxes do not collapse with margins of adjacent boxes
- The root element (`<html>`) cannot be floated
- An inline element that is floated is converted to a block-level element

Float in Practice

One of the most common uses for the `float` property is to float an image with text wrapping it. Here is an example:



Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet quam egestas semper.

Aenean ultricies mi vitae est. Mauris placerat eleifend leo. Quisque sit amet est et sapien ullamcorper pharetra. Vestibulum erat wisi, condimentum sed, commodo vitae, ornare sit amet, wisi. Aenean fermentum, elit eget tincidunt condimentum, eros ipsum rutrum orci, sagittis tempus lacus enim ac dui.

Donec non enim in turpis pulvinar facilisis. Ut felis. Praesent dapibus, neque id cursus faucibus, tortor neque egestas augue, eu vulputate magna eros eu erat. Aliquam erat volutpat. Nam dui mi, tincidunt quis, accumsan porttitor, facilisis luctus, metus.

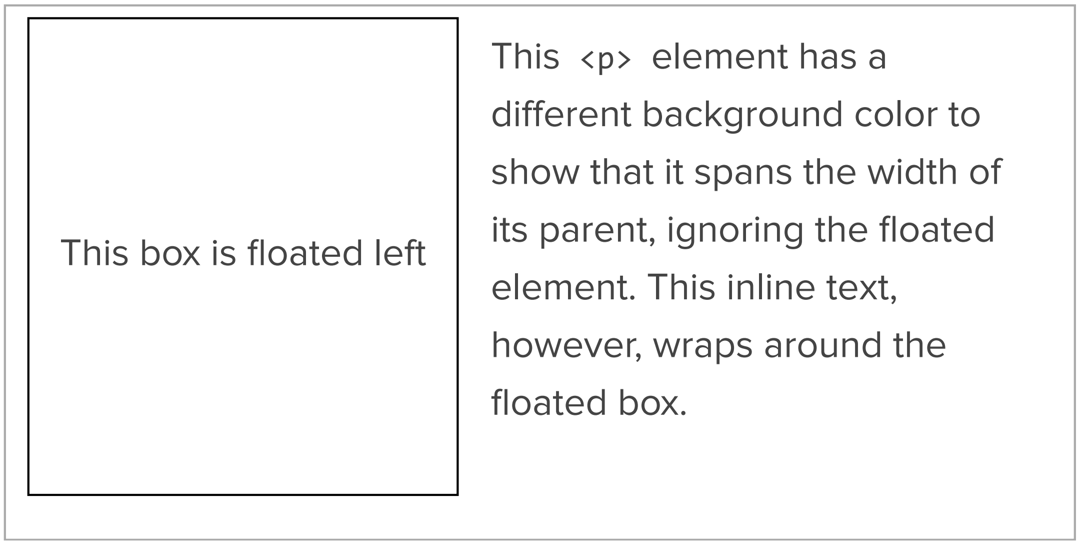
Lifesaver Image from stock.xchng[8]

The CSS applied to the image in the box above is as follows:

```
img {
  float: left;
  margin: 0 15px 5px 0;
  border: solid 1px #bbb;
}
```

The only property required to make this effect work is the `float` property. The other properties (margin and border) are there for aesthetic reasons. The other elements inside the box (the `<p>` tags with text inside them) do not need any styles applied to them.

As mentioned earlier, floated elements are taken out of flow in relation to other block elements, and so other block elements remain in flow, acting as if the floated element is not even there. This is demonstrated visually below:



This `<p>` element has a different background color to show that it spans the width of its parent, ignoring the floated element. This inline text, however, wraps around the floated box.

In the above example, the `<p>` element is a block-level element, so it ignores the floated element, spanning the width of the container (minus padding). All non-floated, block-level elements will behave in like manner.

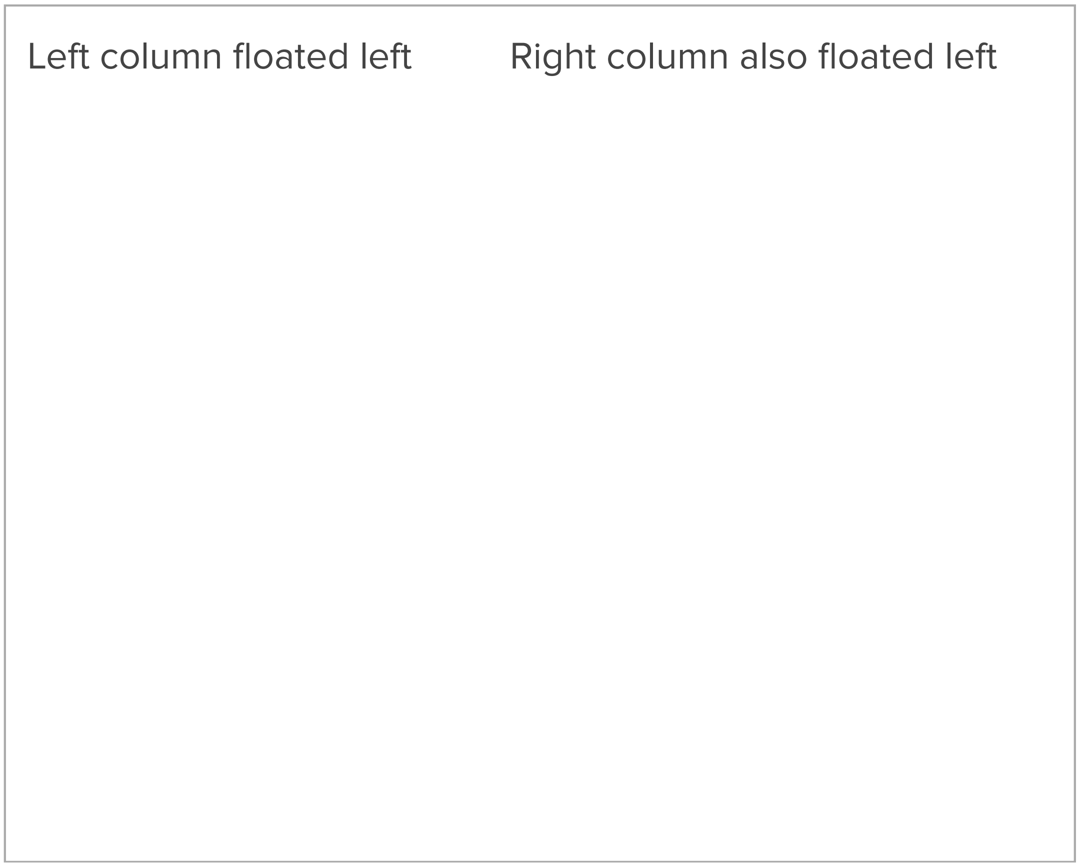
The text in the paragraph is inline, so it flows around the floated element. The floated box is also given margin settings to offset it from the paragraph, making it visually clear that the paragraph element is ignoring the floated box.

Advertisement

Clearing Floats

Layout issues with floats are commonly fixed using the CSS `clear` property, which lets you “clear” floated elements from the left or right side, or both sides, of an element.

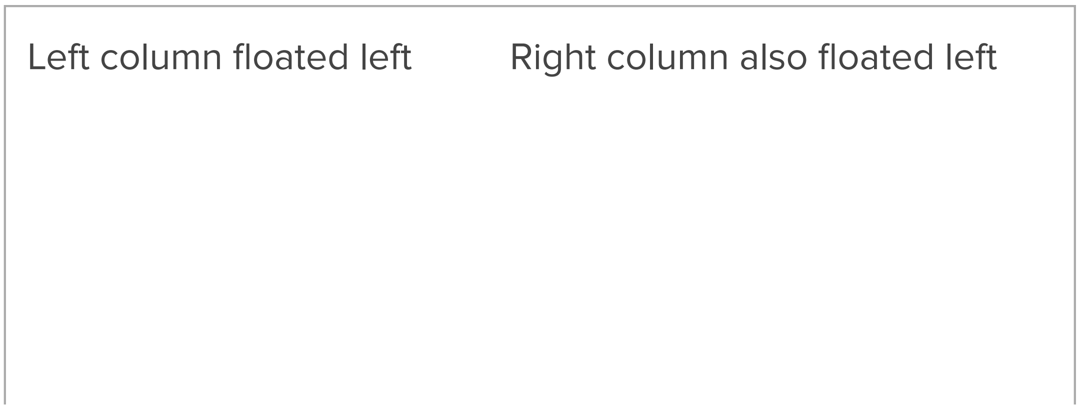
Let’s take a look at an example that commonly occurs — a footer wrapping to the right side of a 2-column layout:


Left column floated left Right column also floated left

If you view this page in IE6 or IE7, you won’t see any problems. The left and right columns are in place, and the footer is nicely tucked underneath. But in Firefox, Opera, Safari, and Chrome, you’ll see the footer jump up beside the left column. This is caused by the float applied to the columns. This is the correct behaviour, even though it is a more problematic one. To resolve this issue, we use the aforementioned `clear` property, applied to the footer:

```
#footer {
  clear: both;
}
```

The result is shown below:

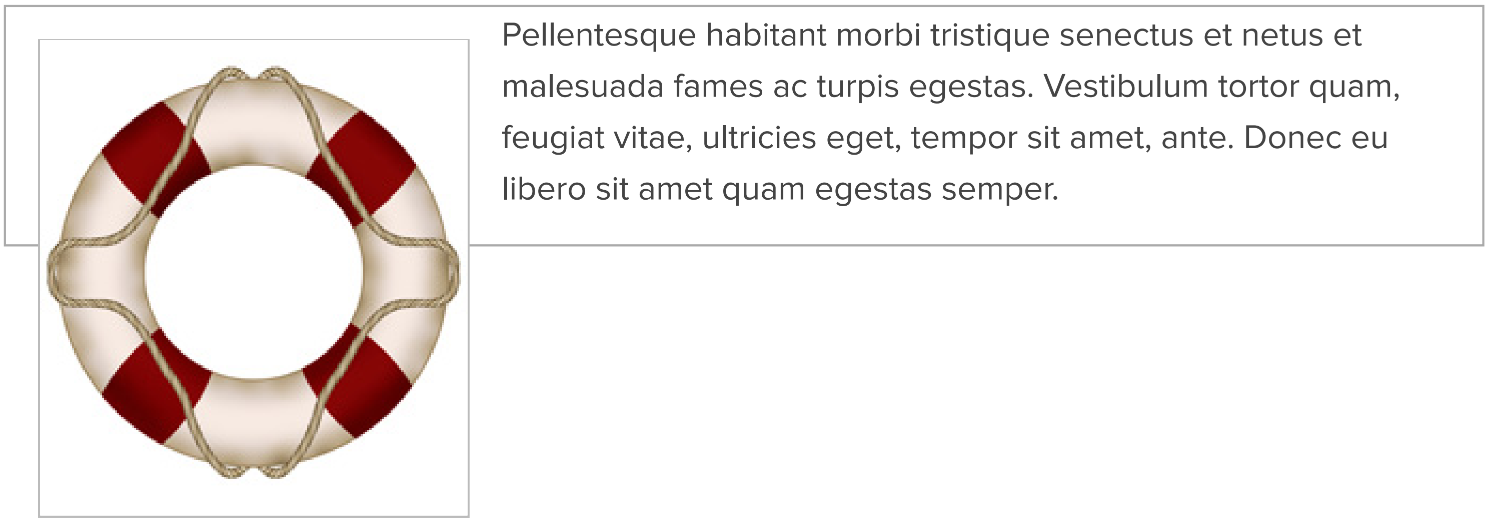

Left column floated left Right column also floated left

The `clear` property will clear only floated elements, so applying `clear: both` to both columns would not cause the footer to drop down, because the footer is not a floated element.

So use `clear` on non-floated elements, and even occasionally on floated elements, to force page elements to occupy their intended space.

Fixing the Collapsed Parent

One of the most common symptoms of float-heavy layouts is the “collapsing parent”. This is demonstrated in the example below:

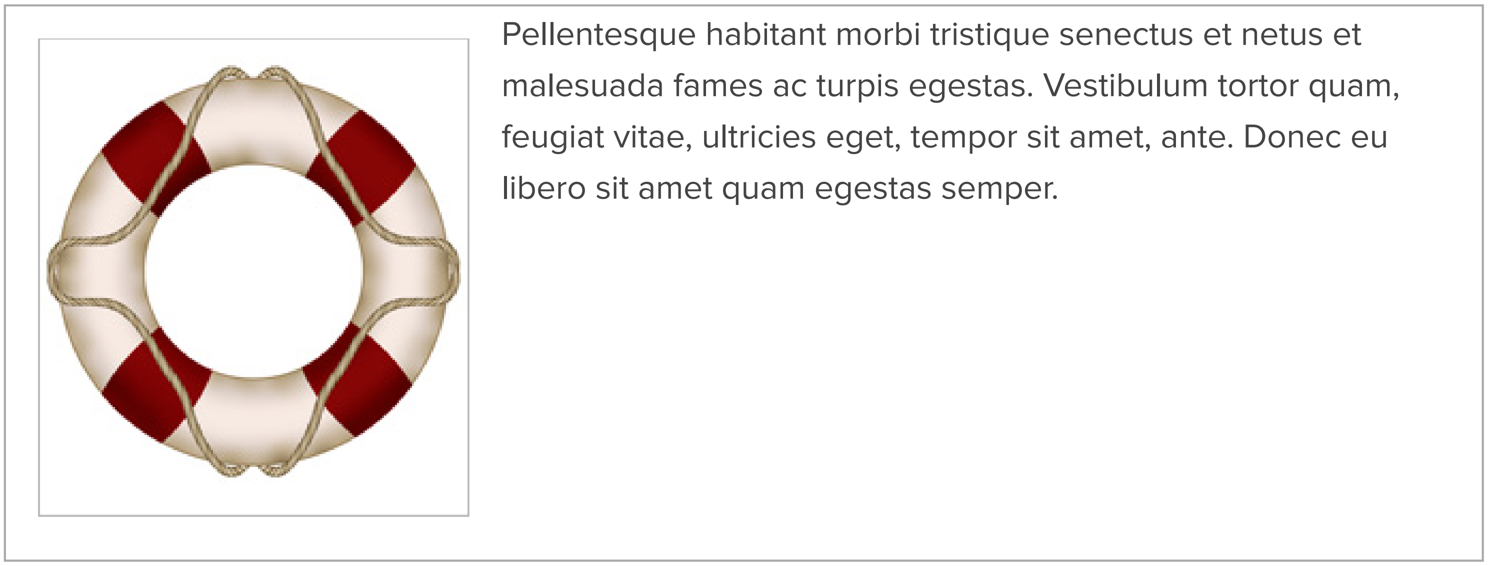


Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet quam egestas semper.

Notice that the bottom of the floated image appears outside its parent. The parent does not fully expand to hold the floated image. This is caused by what we discussed earlier: the floated element is out of the flow in relation to other block elements, so all block elements will render as if the floated element is not even there. This is not a CSS bug; it's in line with CSS specifications. All browsers render the same in this example. It should be pointed out that, in this example, adding a width to the container prevents the issue from occurring in IE, so this would normally be something you would have to resolve in Firefox, Opera, Safari, or Chrome.

SOLUTION 1: FLOAT THE CONTAINER

The easiest way to fix this problem is to float the containing parent element:



Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet quam egestas semper.

Now the container expands to fit all the child elements. But unfortunately this fix will only work in a limited number of circumstances, since floating the parent may have undesirable effects on your layout.

SOLUTION 2: ADDING EXTRA MARKUP

This is an outdated method, but is an easy option. Simply add an extra element at the bottom of the container and "clear" it. Here's how the HTML would look after this method is implemented:

```
<div id="container">
<img src="https://media-mediatemple.netdna-ssl.com/wp-
content/uploads/2009/10/lifesaver.jpg" width="200" height="222" alt="" />
<p>Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac
turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit
amet, ante. Donec eu libero sit amet quam egestas semper.</p>

<div class="clearfix"></div>
</div>
```

And the resulting CSS applied to the new element:

```
.clearfix {
  clear: both;
}
```

You could also do this by means of a `<br />` tag with an inline style. In any case, you will have the desired result: the parent container will expand to enclose all of its children. But this method is not recommended since it adds nonsemantic code to your markup.

SOLUTION 3: THE :AFTER PSEUDO-ELEMENT

The `:after` pseudo-element adds an element to the rendered HTML page. This method has been used quite extensively to resolve float-clearing issues. Here is how the CSS looks:

```
.clearfix:after {
    content: ".";
    display: block;
    height: 0;
    clear: both;
    visibility: hidden;
}
```

The CSS class “clearfix” is applied to any container that has floating children and does not expand to enclose them.

But this method does not work in Internet Explorer up to version 7, so an IE-only style needs to be set with one of the following rules:

```
.clearfix {
  display: inline-block;
}

.clearfix {
  zoom: 1;
}
```

Depending on the type of issue you're dealing with, one of the above two solutions will resolve the issue in Internet Explorer. It should be noted that the `zoom` property is a non-standard Microsoft proprietary property, and will cause your CSS to become invalid.

So, because the `:after` pseudo-element solution does not work in IE6/7, is a little bit bloated code-wise, and requires additional invalid IE-only styles, this solution is not the best method, but is probably the best we've considered so far.

SOLUTION 4: THE OVERFLOW PROPERTY

By far the best, and easiest solution to resolve the collapsing parent issue is to add `overflow: hidden` or `overflow: auto` to the parent element. It's clean, easy to maintain, works in almost all browsers, and does not add extra markup.

You'll notice I said "almost" all browsers. This is because it does not work in IE6. But, in many cases, the parent container will have a set width, which fixes the issue in IE6. If the parent container does not have a width, you can add an IE6-only style with the following code:

```
// This fix is for IE6 only
.clearfix {
  height: 1%;
  overflow: visible;
}
```

In IE6, the `height` property is incorrectly treated as `min-height`, so this forces the container to enclose its children. The `overflow` property is then set back to "visible", to ensure the content is not hidden or scrolled.

The only drawback to using the `overflow` method (in any browser) is the possibility that the containing element will have scrollbars or hide content. If there are any elements with negative margins or absolute positioning inside the parent, they will be obscured if they go beyond the parent's borders, so this method should be used carefully. It should also be noted that if the containing element has to have a specified `height`, or `min-height`, then you would definitely not be able to use the `overflow` method.

So, there really is no simple, cross-browser solution to the collapsing parent issue. But almost any float clearing issue can be resolved with one of the above methods.

Float-Related Bugs in Internet Explorer

Over the years, there have been numerous articles published online that discuss common bugs in connection with floats in CSS layouts. All of these, not surprisingly, deal with problems specific to Internet Explorer. Below, you'll find a list of links to a number of articles that discuss float-related issues in-depth:

- The Internet Explorer Guillotine Bug[9]
- The IE5/6 Doubled Float-Margin Bug[10]
- IE7 Bottom Margin Bug[11]
- The IE Escaping Floats Bug[12]
- The IE6 Peekaboo Bug[13]
- The IE6 "Ghost Text" Bug[14]
- The IE6 Expanding Box Problem[15]
- The IE6 3-pixel Gap[16]

Changing the Float Property with JavaScript

To change a CSS value in JavaScript, you would access the `style` object, converting the intended CSS property to “camel case”. For example, the CSS property “margin-left” becomes `marginLeft`; the property `background-color` becomes `backgroundColor`, and so on. But with the `float` property, it’s different, because `float` is already a reserved word in JavaScript. So, the following would be incorrect:

```
myDiv.style.float = "left";
```

Instead, you would use one of the following:

```
// For Internet Explorer
myDiv.style.styleFloat = "left";

// For all other browsers
myDiv.style.cssFloat = "left";
```

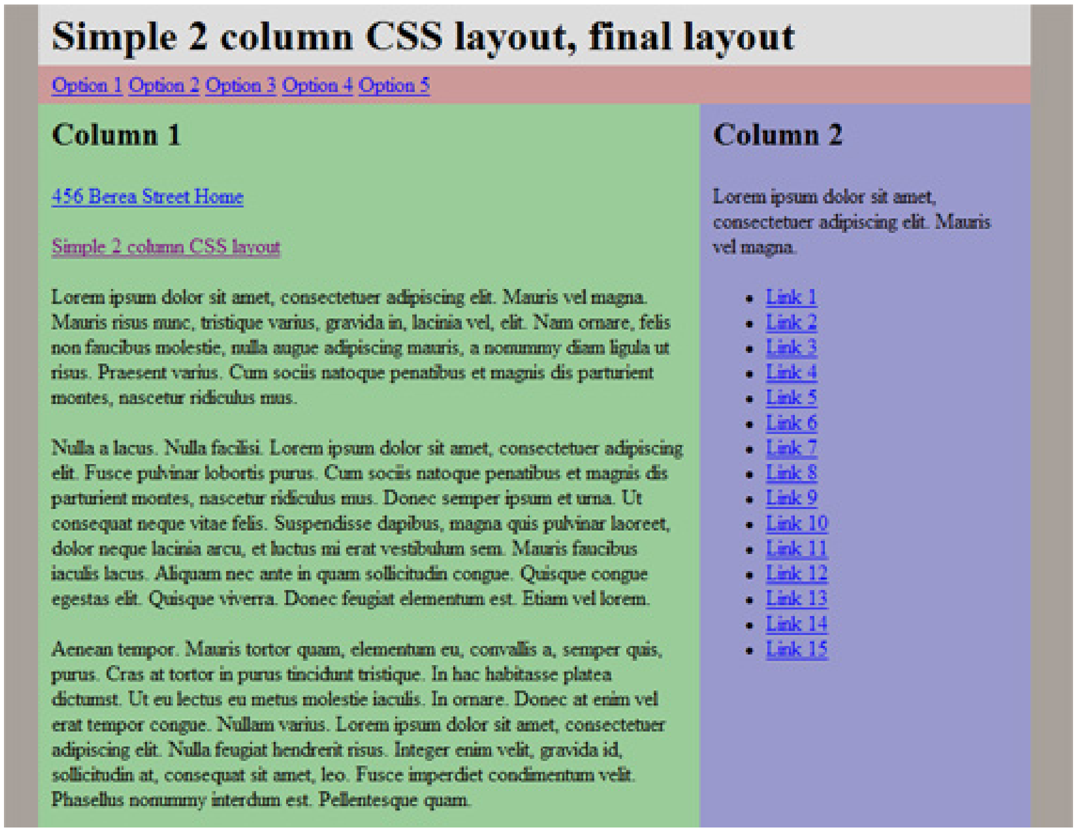
Practical Uses for Float

Floats can be used to resolve a number of design challenges in CSS layouts. Some examples are discussed here.

2-COLUMN, FIXED-WIDTH LAYOUT

Roger Johansson of 456 Berea Street[17] outlines an 8-step tutorial to create a simple, cross-browser, 2-column, horizontally centered layout. The `float` property is integral to the chemistry of this layout.

> *“The layout consists of a header, a horizontal navigation bar, a main content column, a sidebar, and a footer. It is also horizontally centered in the browser window. A pretty basic layout, and not at all difficult to create with CSS once you know how to deal with the inevitable Internet Explorer bugs.”*



Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Mauris vel magna. Mauris risus nunc, tristique varius, gravida in, lacinia vel, elit. Nam ornare, felis non faucibus molestie, nulla augue adipiscing mauris, a nonummy diam ligula ut risus. Praesent varius. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus.

Nulla a lacus. Nulla facilisi. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Fusce pulvinar lobortis purus. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec semper ipsum et urna. Ut consequat neque vitae felis. Suspendisse dapibus, magna quis pulvinar laoreet, dolor neque lacinia arcu, et luctus mi erat vestibulum sem. Mauris faucibus iaculis lacus. Aliquam nec ante in quam sollicitudin congue. Quisque congue egestas elit. Quisque viverra. Donec feugiat elementum est. Etiam vel lorem.

Aenean tempor. Mauris tortor quam, elementum eu, convallis a, semper quis, purus. Cras at tortor in purus tincidunt tristique. In hac habitasse platea dictumst. Ut eu lectus eu metus molestie iaculis. In ornare. Donec at enim vel erat tempor congue. Nullam varius. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Nulla feugiat hendrerit risus. Integer enim velit, gravida id, sollicitudin at, consequat sit amet, leo. Fusce imperdiet condimentum velit. Phasellus nonummy interdum est. Pellentesque quam.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Mauris vel magna.

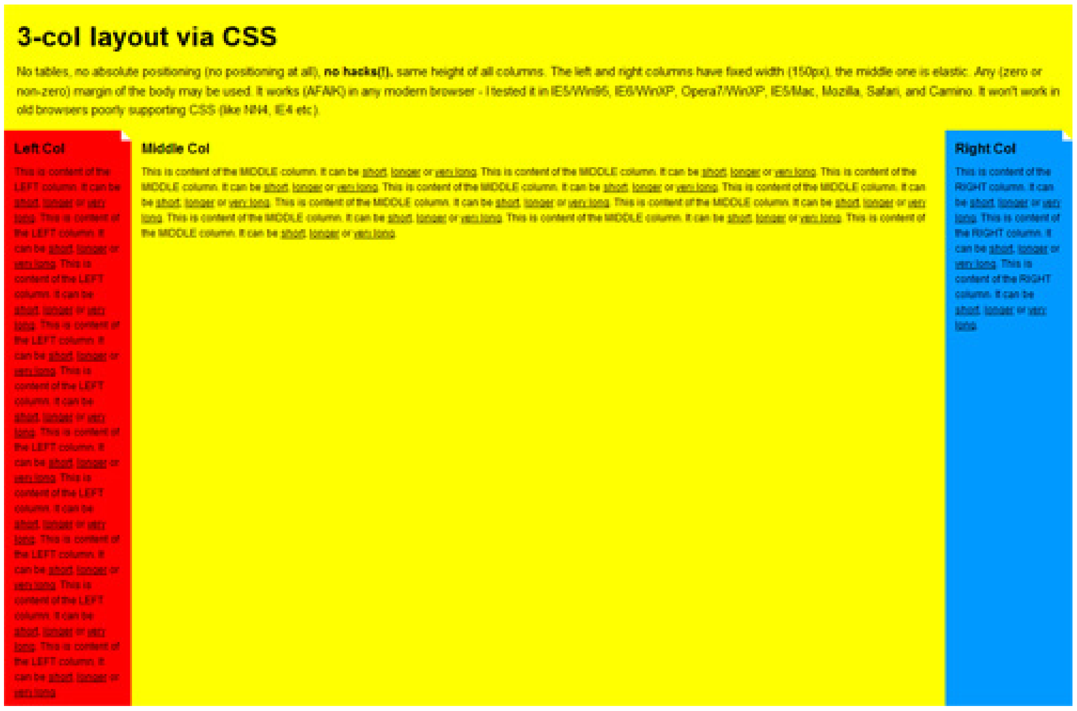
18

Simple 2 column CSS layout[19]

3-COLUMN, EQUAL-HEIGHT LAYOUT

Petr Stanicek of pixy.cz[20] demonstrates a cross-browser 3-column layout, again using `float`:

> *"No tables, no absolute positioning (no positioning at all), no hacks(!), same height of all columns. The left and right columns have fixed width (150px), the middle one is elastic."*



3-col layout via CSS

No tables, no absolute positioning (no positioning at all), **no hacks(!)**, same height of all columns. The left and right columns have fixed width (150px), the middle one is elastic. Any (zero or non-zero) margin of the body may be used. It works (AFAIK) in any modern browser - I tested it in IE5/Win95, IE6/WinXP, Opera7/WinXP, IE5/Mac, Mozilla, Safari, and Camino. It won't work in old browsers poorly supporting CSS (like NN4, IE4 etc).

Left Col

This is content of the LEFT column. It can be short, longer or very long. This is content of the LEFT column. It can be short, longer or very long. This is content of the LEFT column. It can be short, longer or very long. This is content of the LEFT column. It can be short, longer or very long. This is content of the LEFT column. It can be short, longer or very long. This is content of the LEFT column. It can be short, longer or very long. This is content of the LEFT column. It can be short, longer or very long. This is content of the LEFT column. It can be short, longer or very long. This is content of the LEFT column. It can be short, longer or very long. This is content of the LEFT column. It can be short, longer or very long.

Middle Col

This is content of the MIDDLE column. It can be short, longer or very long. This is content of the MIDDLE column. It can be short, longer or very long. This is content of the MIDDLE column. It can be short, longer or very long. This is content of the MIDDLE column. It can be short, longer or very long. This is content of the MIDDLE column. It can be short, longer or very long. This is content of the MIDDLE column. It can be short, longer or very long. This is content of the MIDDLE column. It can be short, longer or very long. This is content of the MIDDLE column. It can be short, longer or very long. This is content of the MIDDLE column. It can be short, longer or very long. This is content of the MIDDLE column. It can be short, longer or very long.

Right Col

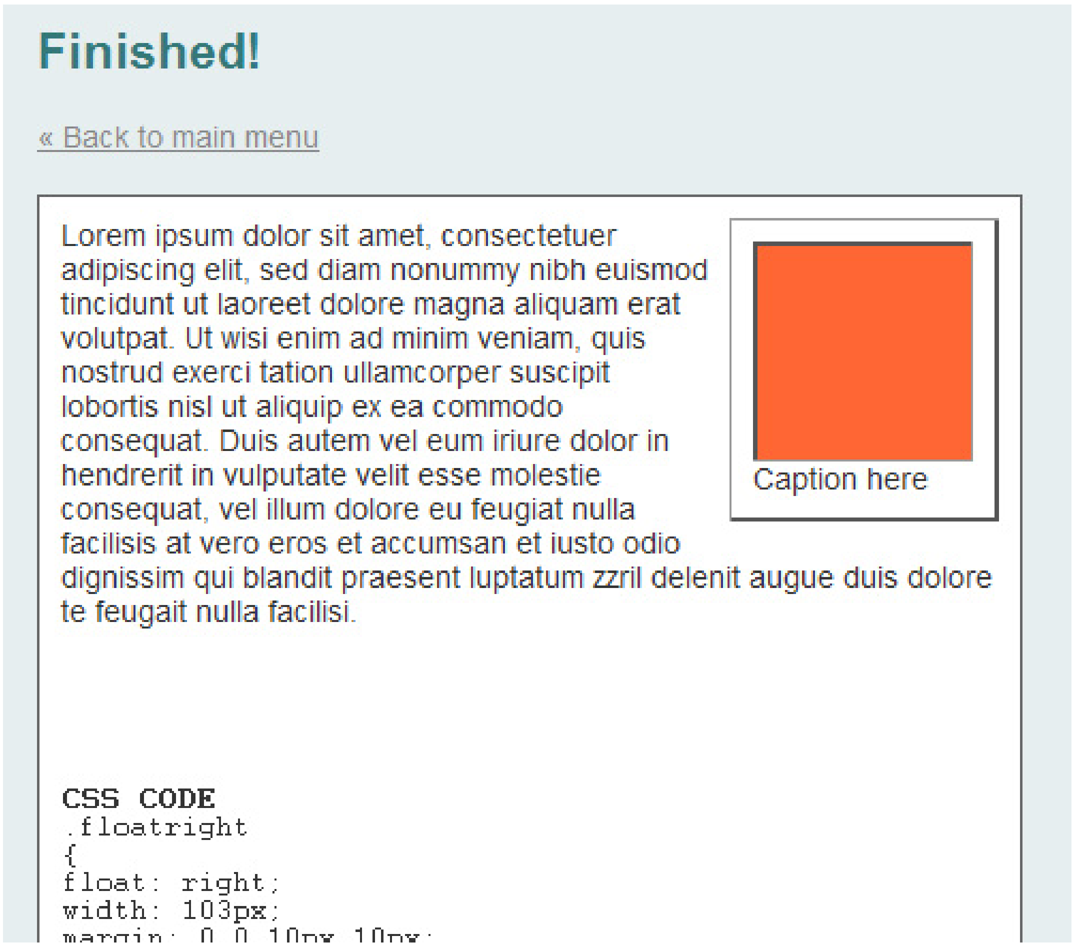
This is content of the RIGHT column. It can be short, longer or very long. This is content of the RIGHT column. It can be short, longer or very long. This is content of the RIGHT column. It can be short, longer or very long.

21

3-Column Layout with CSS[22]

FLOATED IMAGE WITH CAPTION

Similar to what we discussed earlier under “Float in Practice”, Max Design[23] describes how to float an image with a caption, allowing text to wrap around it naturally.



Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi.

CSS CODE

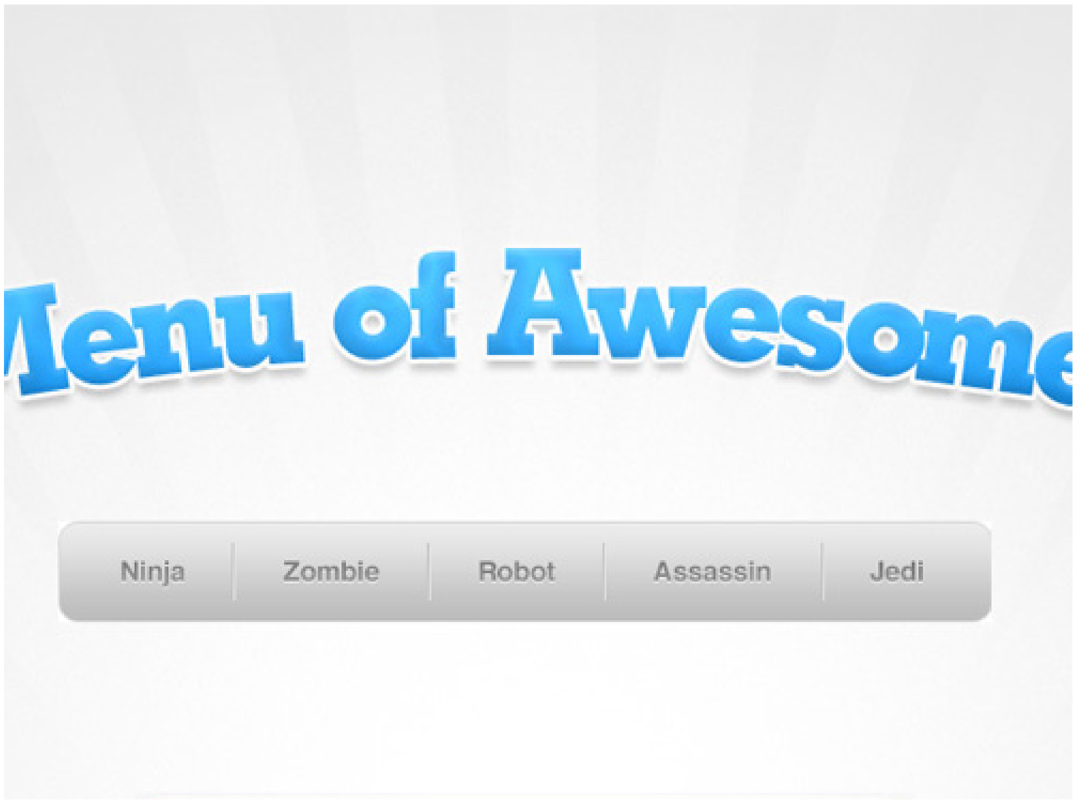
```
.floatright
{
float: right;
width: 103px;
```

24

Floating an Image and Caption[25]

HORIZONTAL NAVIGATION WITH UNORDERED LISTS

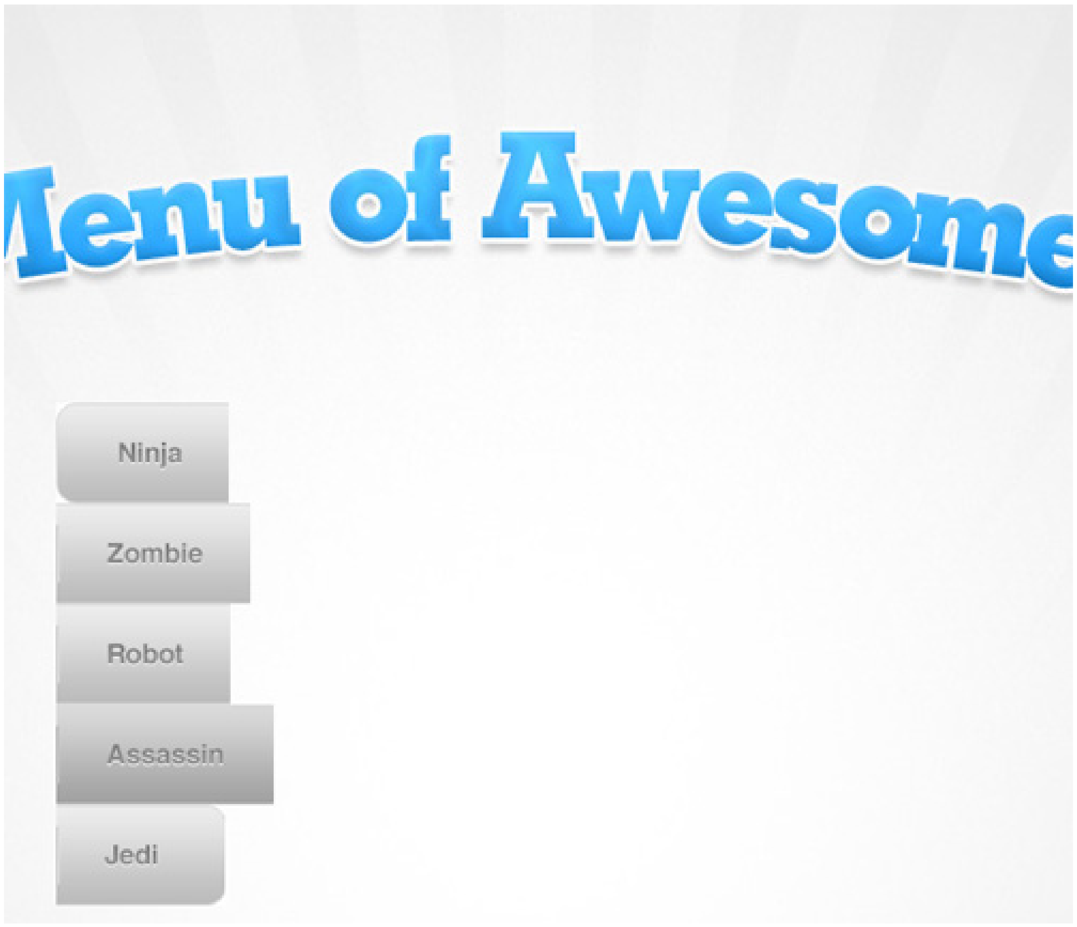
The `float` property is a key ingredient in coding sprite-based horizontal navigation bars. Chris Spooner of Line25[26] describes how to create a Menu of Awesomeness[27], in which the `<li>` elements that hold the navigation buttons are floated left:



28

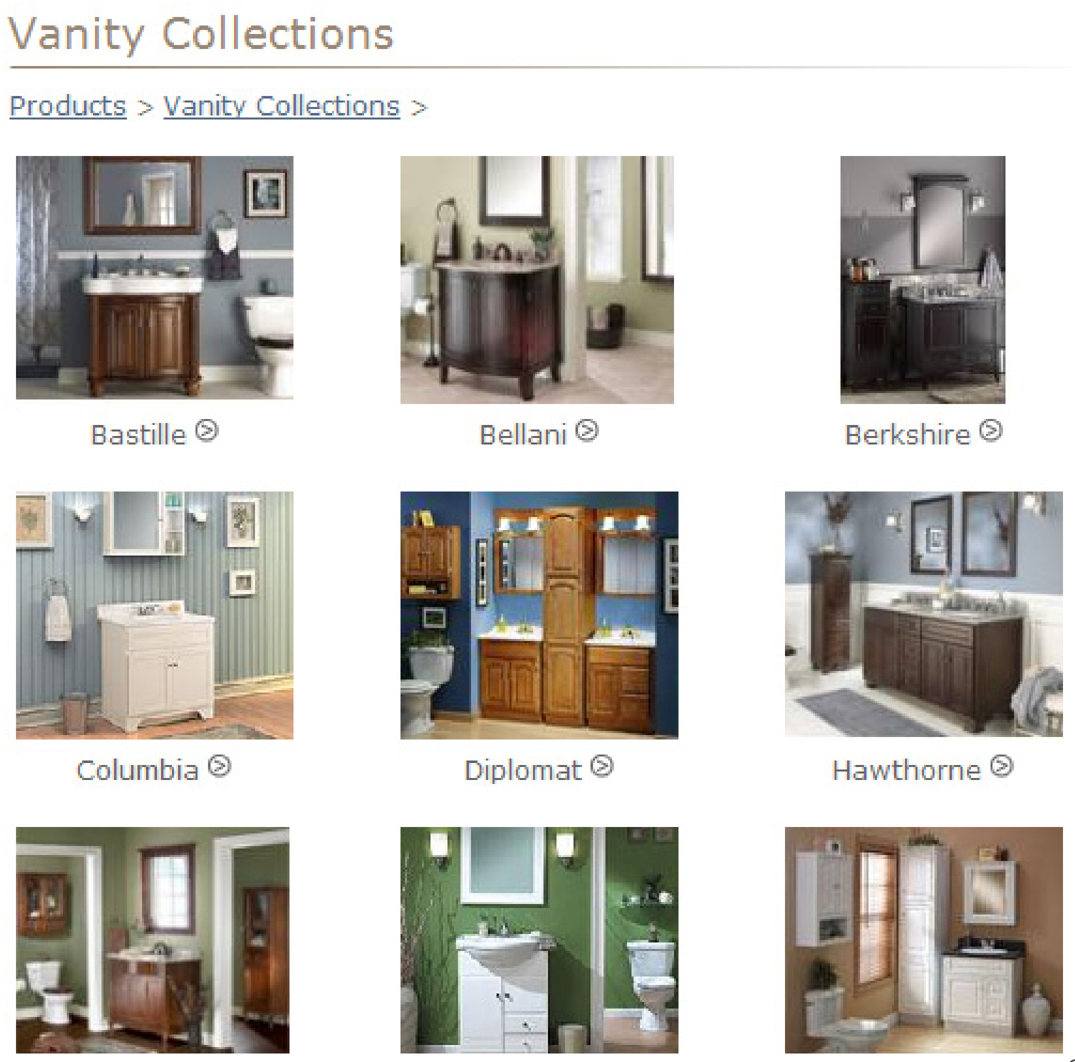
How to Create a CSS Menu Using Image Sprites[29]

To demonstrate the importance of the `float` property in this example, here is a screen shot of the same image after using firebug to remove the `float: left`:



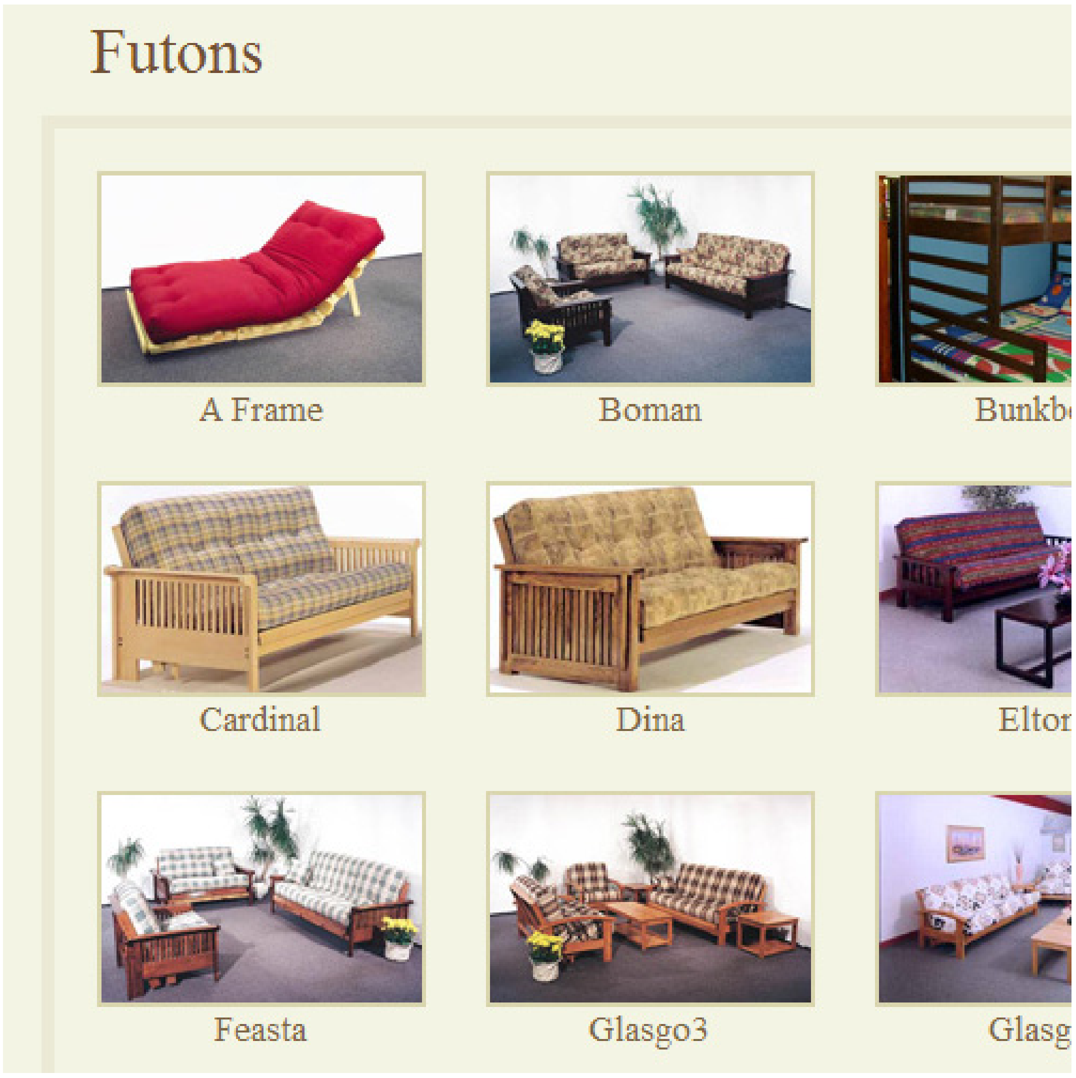
GRID-BASED PHOTO GALLERIES

A simple use for the `float` property is to left-float a series of photos contained in an unordered list, which gets the same result as what you would see in a table-based layout.



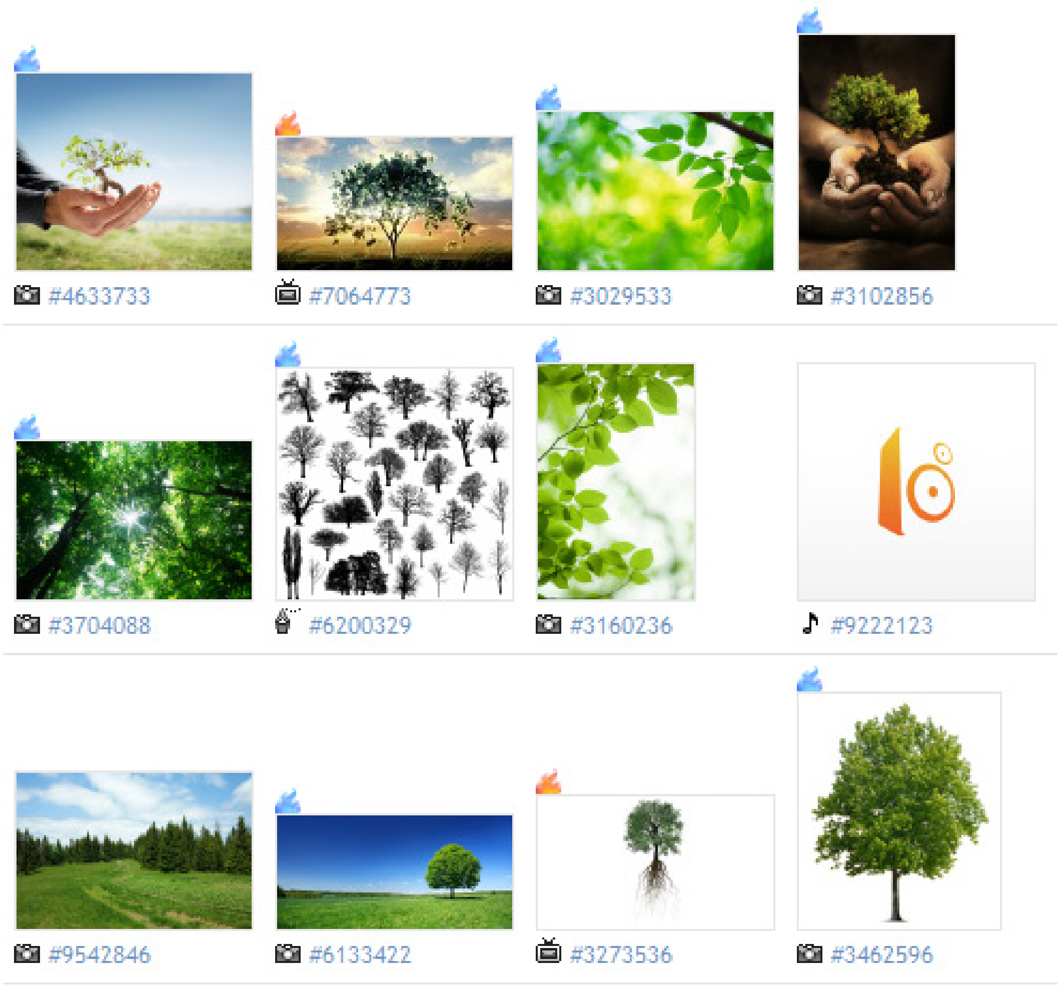
30

Foremost Canada's product page[31] (above) displays their products in grid-based format, next to a left-column sidebar. The photos are displayed in an unordered list with the float property for all `<li>` elements set to `float: left`. This works better than a table-based grid, because the number of photos in the gallery can change and the layout would not be affected.



32

Paragon Furniture's futons page[33] (above) is another example of a product page using an unordered list with floated list items.

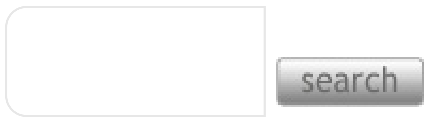


34

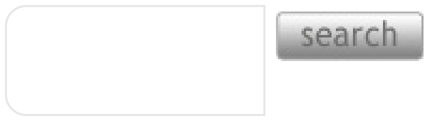
iStockphoto's search results page[35] (above) is a similarly-structured grid of photos, but this time the photos are contained in left-floated `<div>` elements, instead of `<li>` elements.

ALIGNING AN <INPUT> FIELD WITH A BUTTON

Default styling on form elements across different browsers can be a pain to deal with. Often times, in a single-field form like a search form, it is necessary to place the `<input>` element next to the submit button. Here is a simple search form, with an image used for the submit button:



In every browser, the result is the same: The button appears slightly higher than the input field. Changing margins and padding does nothing. The simple way to fix this is to float the input field left, and add a small right margin. Here is the result:



Conclusion

As was mentioned at the outset, without the CSS `float` property, table-less layouts would be, at worst, impossible, and, at best, unmaintainable. Floats will continue to be prominent in CSS layouts, even as CSS3 begins to gain prominence — even though there have been a few discussions about layouts without the use of floats[36].

Hopefully this discussion has simplified some of the mysteries related to floats, and provided some practical solutions to a number of issues faced by CSS developers today.

Further Reading

- Sitepoint CSS Reference: Float[37]
- All About Floats on CSS-Tricks[38]
- Float Layouts @ The Autistic Cuckoo[39]

- Simple Clearing of Floats[40]
- Visual Formatting Model: Floats[41]
- Floating and Clearing[42]

ABOUT THE AUTHOR

Louis Lazaris is a writer and freelance Web Developer based in Toronto, Canada. He has 9 years of experience in the web development industry and posts web design articles[43] and tutorials[44] on his blog, Impressive Webs[45]. You can follow Louis on Twitter[46] or contact him using this form[47].

FOOTNOTES

1 http://www.flickr.com/photos/presentday/2789469603/

2 http://www.netmag.co.uk/

3 http://www.stopdesign.com/present/2004/ddw-seattle/tables/

4 http://www.stopdesign.com/present/2004/ddw-seattle/tables/?no=26

5 http://www.stopdesign.com/present/2004/ddw-seattle/tables/?no=26

6 http://reference.sitepoint.com/css/replacedelements

7 http://www.w3.org/TR/CSS1/#floating-elements

8 http://www.sxc.hu/photo/1179935

9 http://css-class.com/articles/explorer/guillotine/index.htm

10 http://www.positioniseverything.net/explorer/doubled-margin.html

11 http://www.maratz.com/blog/archives/2006/11/11/ie-7-quirks-floats-and-margins/

12 http://www.positioniseverything.net/explorer/escape-floats.html

13 http://www.positioniseverything.net/explorer/peekaboo.html

14 http://www.impressivewebs.com/ie6-ghost-text-bug-with-multiple-solutions/

15 http://www.positioniseverything.net/explorer/expandingboxbug.html

16 http://www.positioniseverything.net/explorer/threepxtest.html

17 http://www.456bereastreet.com/

18 http://www.456bereastreet.com/lab/developing_with_web_standards/csslayout/2-col/

19 http://www.456bereastreet.com/lab/developing_with_web_standards/csslayout/2-col/

20 http://www.pixy.cz

21 http://www.pixy.cz/blogg/clanky/css-3col-layout/

22 http://www.pixy.cz/blogg/clanky/css-3col-layout/

23 http://www.maxdesign.com.au/

24 http://css.maxdesign.com.au/floatutorial/tutorial0211.htm

25 http://css.maxdesign.com.au/floatutorial/tutorial0211.htm

26 http://line25.com/

27 http://line25.com/tutorials/how-to-create-a-css-menu-using-image-sprites

28 http://line25.com/tutorials/how-to-create-a-css-menu-using-image-sprites

29 http://line25.com/tutorials/how-to-create-a-css-menu-using-image-sprites

30 http://www.foremostcanada.com/products.php?ctgyid=2

31 http://www.foremostcanada.com/products.php?ctgyid=2

32 http://www.getparagon.com/futons.php

33 http://www.getparagon.com/futons.php

34 http://www.istockphoto.com/file_search.php?text=trees&action=file

35 http://www.istockphoto.com/file_search.php?text=trees&action=file

36 http://tjkdesign.com/articles/float-less_css_layouts.asp

37 http://reference.sitepoint.com/css/float

38 http://css-tricks.com/all-about-floats/

39 http://www.autisticcuckoo.net/archive.php?id=2004/12/10/floating

40 http://www.sitepoint.com/blogs/2005/02/26/simple-clearing-of-floats/

41 http://www.w3.org/TR/CSS2/visuren.html#floats

42 http://reference.sitepoint.com/css/floatclear

43 http://www.impressivewebs.com/articles

44 http://www.impressivewebs.com/tutorials

45 http://www.impressivewebs.com

46 http://twitter.com/ImpressiveWebs

47 http://www.impressivewebs.com/contact

Louis Lazaris

Louis Lazaris is a freelance web developer and author based in Toronto, Canada. He blogs about front-end code on Impressive Webs and curates Web Tools Weekly, a weekly newsletter for front-end developers.