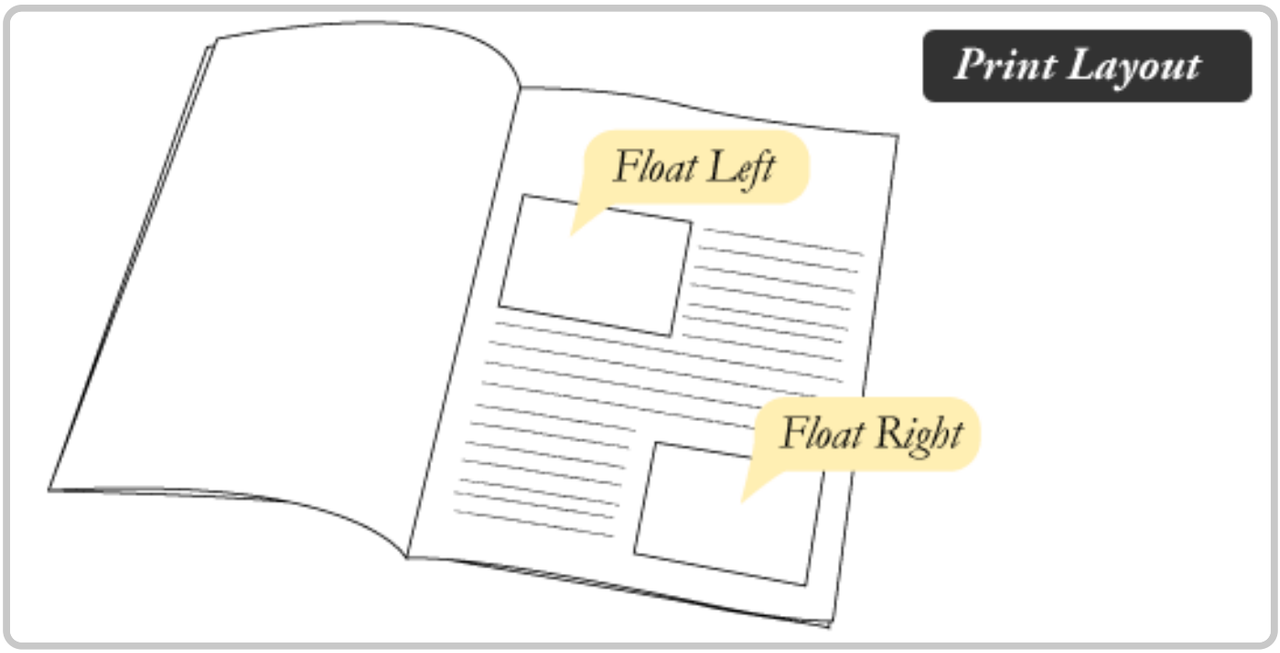
Chris Coyier Mar 25, 2021

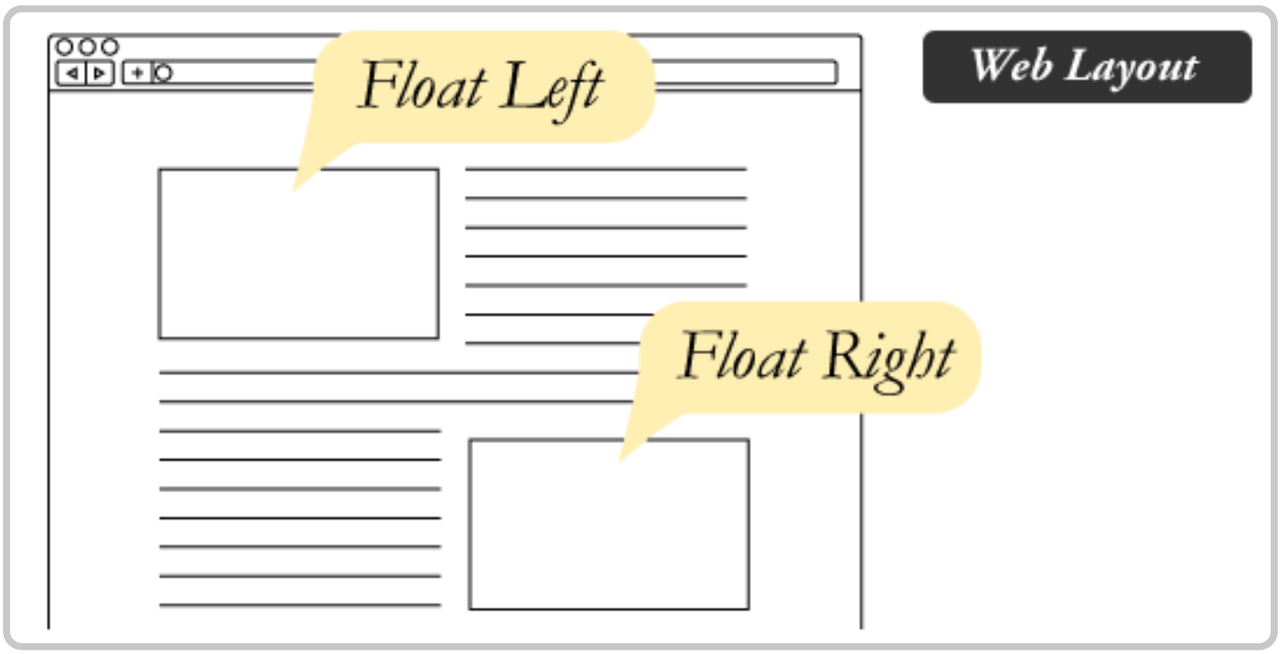
(#aa-what-is-float) What is “Float”?

Float is a CSS positioning property. To understand its purpose and origin, we can look to print design. In a print layout, images may be set into the page such that text wraps around them as needed. This is commonly and appropriately called “text wrap”. Here is an example of that.



In page layout programs, the boxes that hold the text can be told to honor the text wrap, or to ignore it. Ignoring the text

wrap will allow the words to flow right over the image like it wasn't even there. This is the difference between that image being part of the *flow* of the page (or not). Web design is very similar.



In web design, page elements with the CSS `float` property applied to them are just like the images in the print layout where the text flows around them. Floated elements *remain a part of the flow of the web page.* This is distinctly different than page elements that use absolute positioning. Absolutely positioned page elements are **removed** from the flow of the webpage, like when the text box in the print layout was told to ignore the page wrap. Absolutely positioned page elements will not affect the position of other elements and other elements will not affect them, whether they touch each other or not.

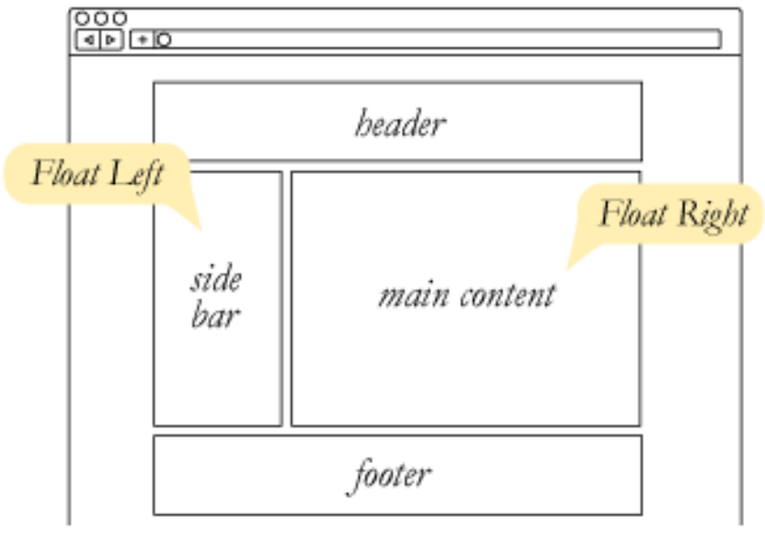
Setting the float on an element with CSS happens like this:

```css
#sidebar {
  float: right;
}
```

There are four valid values for the float property. **Left** and **Right** float elements those directions respectively. **None** (the default) ensures the element will not float and **Inherit** which will assume the float value from that elements parent element.

(#aa-what-are-floats-used-for)
What are floats used for?

Aside from the simple example of wrapping text around images, floats can be used to create **entire web layouts**.

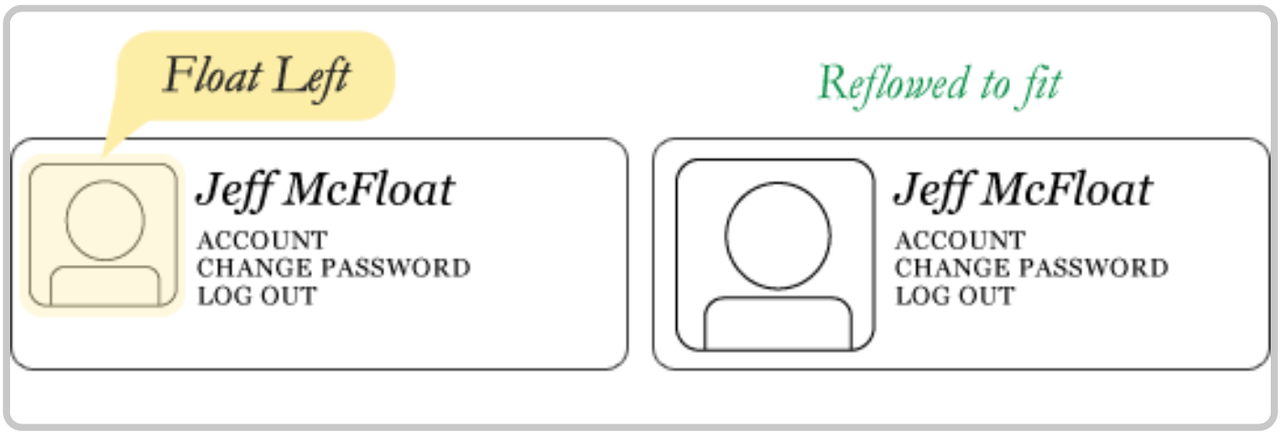


While floats can still be used for layout, these days, we have much stronger tools for creating layout on web pages. Namely, Flexbox (https://css-tricks.com/snippets/css/a-guide-to-flexbox/) and Grid

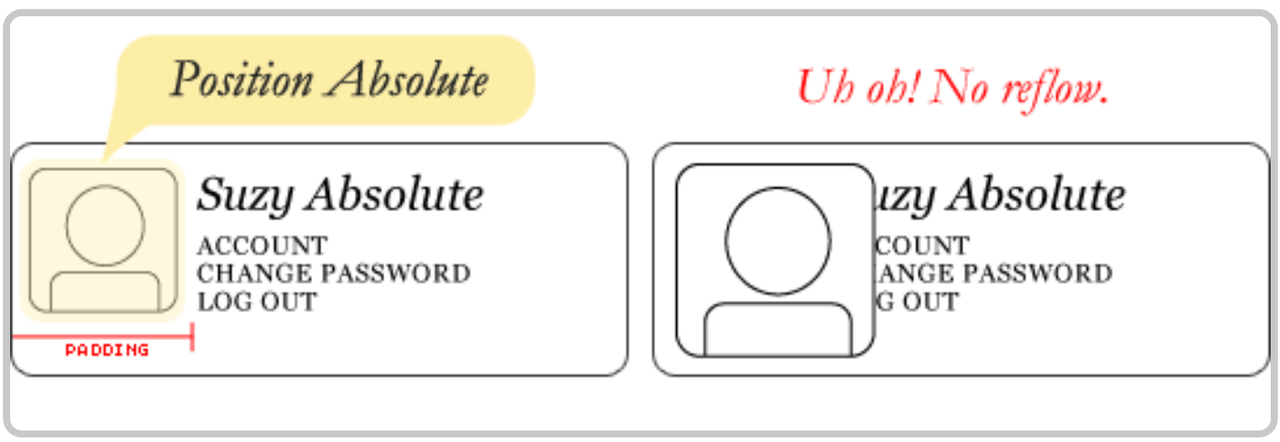
Hey!

(https://css-tricks.com/snippets/css/complete-guide-grid/) . Floats are still useful to know about because they have some abilities entirely unique to them, which is all covered in this article.

Floats are also helpful for layout in smaller instances. Take for example this little area of a web page. If we use float for our little avatar image, when that image changes size the text in the box will reflow to accommodate:

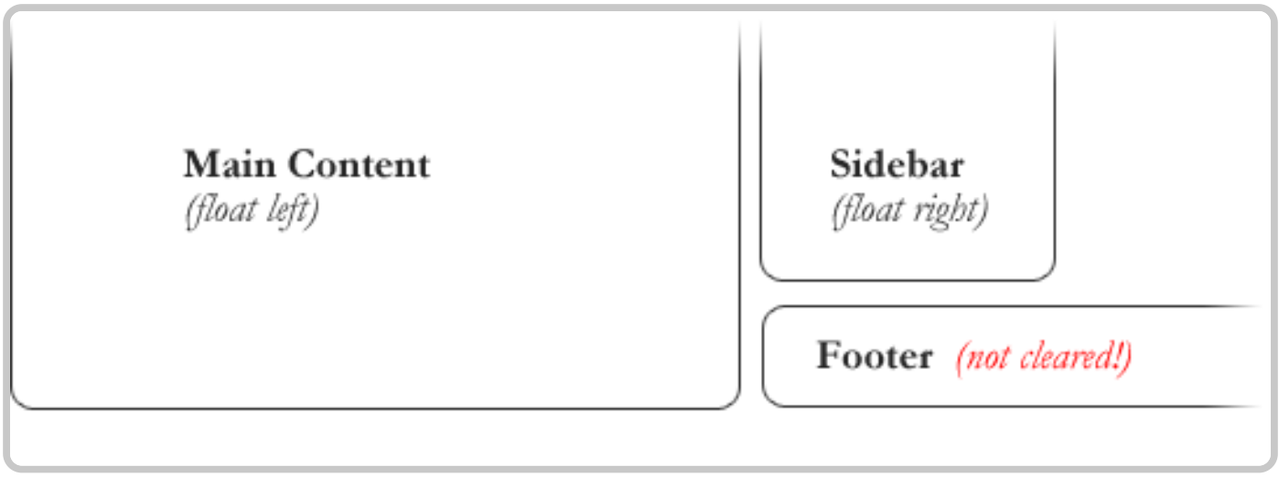


This same layout could be accomplished using relative positioning on container and absolute positioning on the avatar as well. In doing it this way, the text would be unaffected by the avatar and not be able to reflow on a size change.



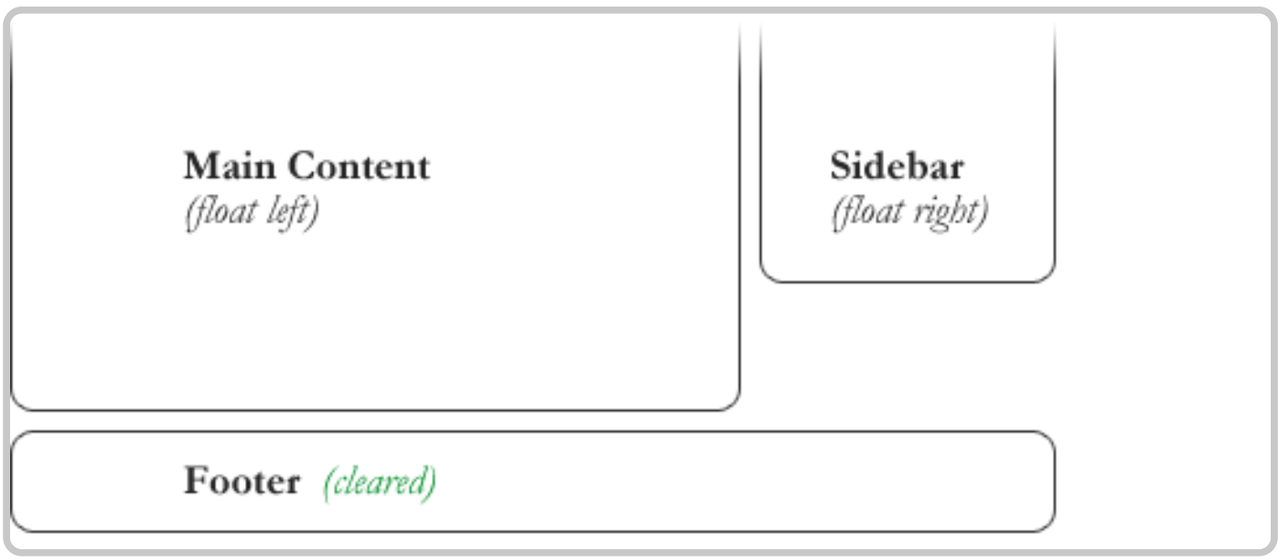
(#aa-clearing-the-float) Clearing the Float

Float's sister property is `clear`. An element that has the clear property set on it will not move up adjacent to the float like the float desires, but will move itself down past the float. Again an illustration probably does more good than words do.

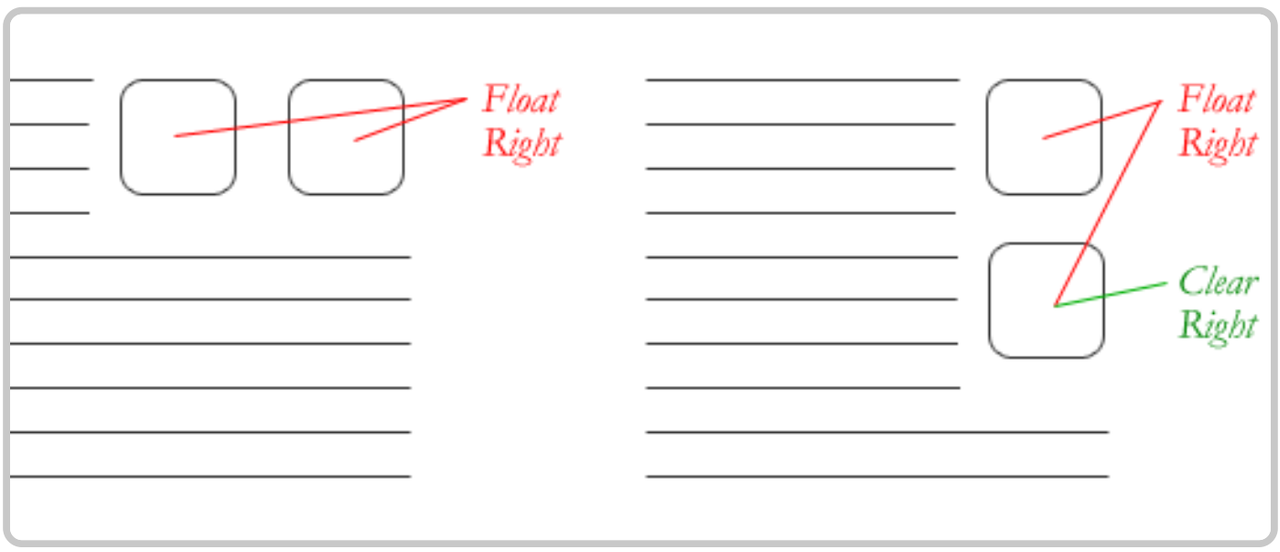


In the above example, the sidebar is floated to the right and is shorter than the main content area. The footer then is required to jump up into that available space as is required by the float. To fix this problem, the footer can be cleared to ensure it stays beneath both floated columns.

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

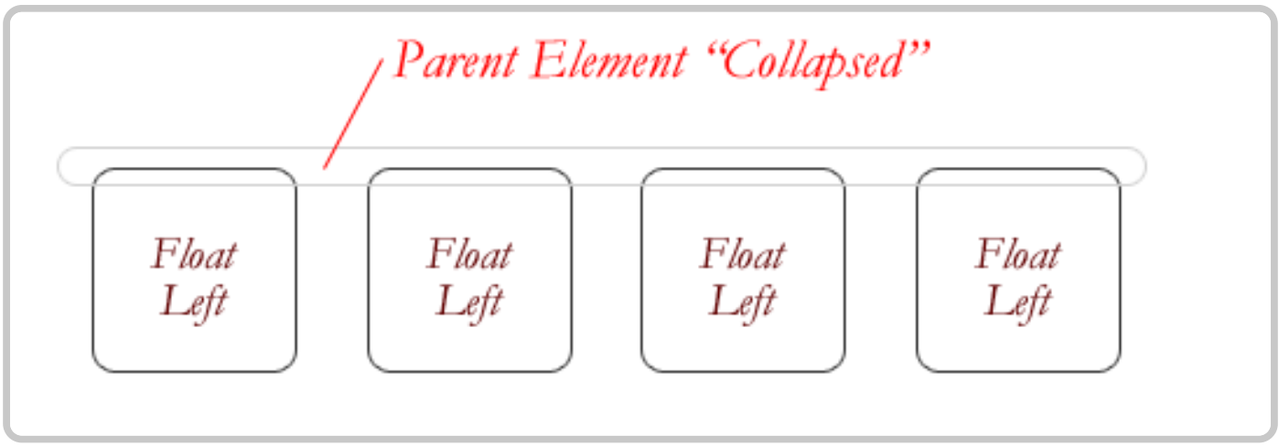


Clear has four valid values as well. **Both** is most commonly used, which clears floats coming from either direction. **Left** and **Right** can be used to only clear the float from one direction respectively. **None** is the default, which is typically unnecessary unless removing a clear value from a cascade. **Inherit** would be the fifth, but is strangely not supported in Internet Explorer. Clearing only the left or right float, while less commonly seen in the wild, definitely has its uses.



(#aa-the-great-collapse) The Great Collapse

One of the more bewildering things about working with floats is how they can affect the element that contains them (their “parent” element). If this parent element contained nothing but floated elements, the height of it would literally collapse to nothing. This isn’t always obvious if the parent doesn’t contain any visually noticeable background, but it is important to be aware of.



As anti-intuitive as collapsing seems to be, the alternative is worse. Consider this scenario:

If the block element on top were to have automatically expanded to accommodate the floated element, we would have an unnatural spacing break in the flow of text between

paragraphs, with no practical way of fixing it. If this were the case, us designers would be complaining much harder about this behavior than we do about collapsing.

Collapsing almost always needs to be dealt with to prevent strange layout and cross-browser problems. We fix it by clearing the float **after** the floated elements in the container but **before** the close of the container.

(#aa-techniques-for-clearing-floats) Techniques for Clearing Floats

If you are in a situation where you always know what the succeeding element is going to be, you can apply the `clear: both;` value to that element and go about your business. This is ideal as it requires no fancy hacks and no additional elements making it perfectly semantic. Of course things don't typically work out that way and we need to have more float-clearing tools in our toolbox.

- **The Empty Div Method** is, quite literally, an empty div. `<div style="clear: both;"></div>` Sometimes you'll see a `<br>` element or some other random element used, but div is the most common because it has no browser default styling, doesn't have any special function, and is

unlikely to be generically styled with CSS. This method is scorned by semantic purists since its presence has no contextual meaning at all to the page and is there purely for presentation. Of course in the strictest sense, they are right, but it gets the job done right and doesn't hurt anybody.

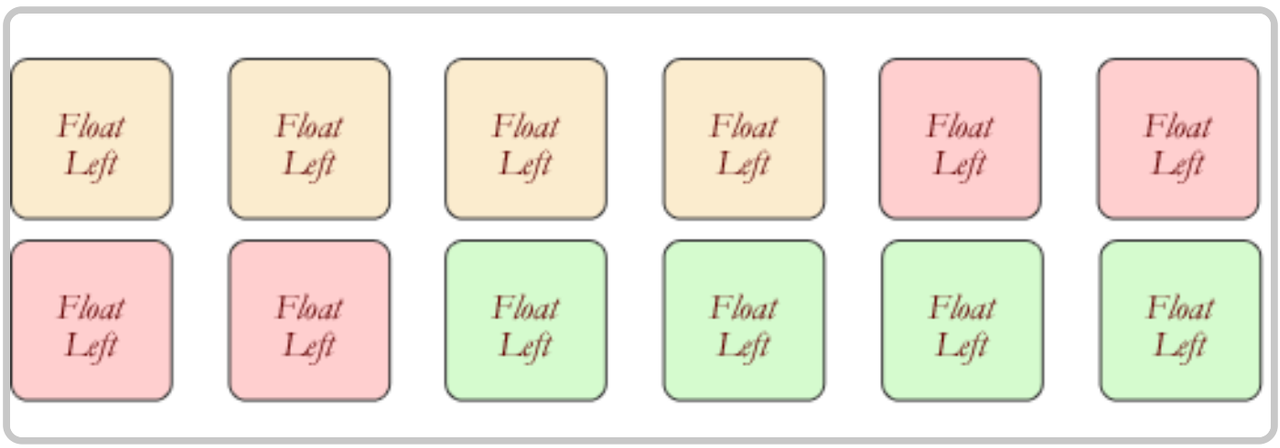
- **The Overflow Method** relies on setting the overflow CSS property on a parent element. If this property is set to auto or hidden on the parent element, the parent will expand to contain the floats, effectively clearing it for succeeding elements. This method can be beautifully semantic as it may not require additional elements. However if you find yourself adding a new div just to apply this, it is equally as non-semantic as the empty div method and less adaptable. Also bear in mind that the overflow property isn't specifically for clearing floats. Be careful not to hide content or trigger unwanted scrollbars.
- **The Easy Clearing Method** uses a clever CSS pseudo selector (`:after`) to clear floats. Rather than setting the overflow on the parent, you apply an additional class like "clearfix" to it. Then apply this CSS:

CSS

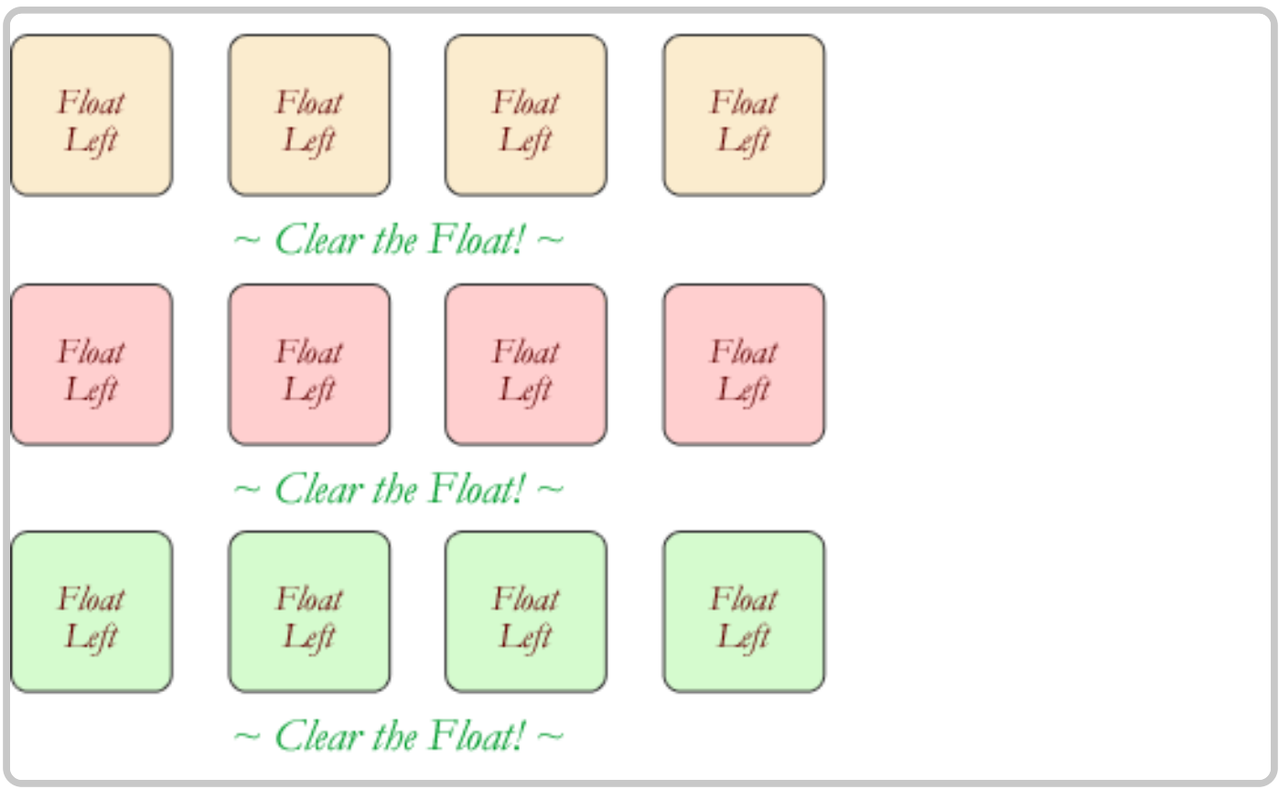
```
.clearfix:after {
   content: ".";
   visibility: hidden;
   display: block;
   height: 0;
   clear: both;
}
```

This will apply a small bit of content, hidden from view, after the parent element which clears the float. This isn't quite the whole story (http://www.positioniseverything.net/easyclearing.html) , as additional code needs to be used to accommodate for older browsers.

Different scenarios call for different float clearing methods. Take for example a grid of blocks, each of different types.



To better visually connect the similar blocks, we want to start a new row as we please, in this case when the color changes. We could use either the overflow or easy clearing method if each of the color groups had a parent element. Or, we use the empty div method in between each group. Three wrapping divs that didn't previously exist or three after divs that didn't previously exist. I'll let you decide which is better.



(#aa-problems-with-floats)

Problems with Floats

Floats often get beat on for being *fragile*. The majority of this fragility comes from IE 6 and the slew of float-related bugs it has. As more and more designers are dropping support for IE 6, you may not care, but for the folks that do care here is a quick rundown.

- **Pushdown** is a symptom of an element inside a floated item being *wider than the float itself* (typically an image). Most browsers will render the image outside the float, but not have the part sticking out affect other layout. IE will expand the float to contain the image, often drastically affecting layout. A common example is an image sticking out of the main content push the sidebar down below.

Quick fix: Make sure you don't have any images that do this, use `overflow: hidden` to cut off excess.

- **Double Margin Bug** – Another thing to remember when dealing with IE 6 is that if you apply a margin in the same direction as the float, it will double the margin (http://www.cssnewbie.com/double-margin-float-bug/) . *Quick fix:* set `display: inline` on the float, and don't worry it will remain a block-level element.
- The **3px Jog** is when text that is up next to a floated element is mysteriously kicked away by 3px like a weird forcefield around the float. *Quick fix:* set a width or height on the affected text.
- In IE 7, the **Bottom Margin Bug** is when if a floated parent has floated children inside it, bottom margin on those children is ignored by the parent. *Quick fix:* using bottom padding on the parent instead.

(#aa-alternatives) Alternatives

If you need text wrapping around images, there really aren't any alternatives for float. Speaking of which, check out this rather clever technique (http://blog.ideashower.com/post/15139639050/css-text-wrapper) for wrapping text around irregular shapes. But for page layout, there definitely are choices. Eric Sol right here on A List Apart has an article on Faux Absolute Positioning (http://alistapart.com/articles/fauxabsolutepositioning) , which is a very interesting technique that in many ways combines the flexibility of floats with the strength of absolute positioning. CSS3 has the Template Layout Module (http://www.w3.org/TR/2009/WD-css3-layout-20090402/) that, when widely adopted, will prove to be the page layout technique of choice.

(#aa-video) Video

I did a screencast (https://css-tricks.com/video-screencasts/42-all-about-floats-screencast/) a while back explaining many of these float concepts.