COLIN | BLOG (http://colintoh.com/blog)

The Anti-hero of CSS Layout - "display:table"

Redeeming the maligned reputation of CSS Table

October 27, 2014

◆

> **Anti-hero**
> - a central character in a story, movie, or drama who lacks conventional heroic attributes.

The topic of table usage in layouts is a sensitive one. In general, web developers consider table-based layout a taboo. Although the reasons against it are well-documented (http://phrogz.net/css/WhyTablesAreBadForLayout.html), most developers are unable to provide a sound background for decrying table-base layout except for, "tables are bad".

The momentum from the early anti-HTML table movement was strong. It managed to brainwash many generation of developers into thinking that **any usage of table** is evil.

Admittedly, I am one of those developers who avoided table layout, even for displaying tabular data. I had even chided my fellow developers when they used `display:table` for dashboard-style layout (or the Holy Grail Layout (http://alistapart.com/article/holygrail)).

In retrospect, it was mind-numbingly stubborn of me to spent a ridiculous amount of time on hacking the CSS.

Two types of table-layout

There are two ways that you can use table in layout - **HTML Table** and **CSS Table**.

HTML Table refers to the usage of table with the native `<table>` tag while **CSS Table** mimics the same table model as **HTML Table** but with CSS properties.

Full source: W3C Table (http://www.w3.org/TR/CSS2/tables.html)

```
table    { display: table }
tr       { display: table-row }
thead    { display: table-header-group }
tbody    { display: table-row-group }
tfoot    { display: table-footer-group }
```

```
col      { display: table-column }
colgroup { display: table-column-group }
td, th   { display: table-cell }
caption  { display: table-caption }
```

view raw (https://gist.github.com/colintoh/0a92a25bb3f3481544e9/raw/ea8f268fd241ce9fc775ce50c52f73a6483fe5f4/table.css)
table.css (https://gist.github.com/colintoh/0a92a25bb3f3481544e9#file-table-css) hosted with ♥ by **GitHub (https://github.com)**

There is a key difference

As an individual who learnt his craft from CSS Zen Garden (http://www.csszengarden.com/), I detest HTML table-layout. Unknowingly, I was falling for the "illusory correlation bias" (http://en.wikipedia.org/wiki/Illusory_correlation) by overestimating the relationship between **HTML Table** and **CSS Table**.

> If it looks,works and sounds like a table, it must be a table? **Wrong!**

In actual fact, **CSS Table** has a key differentiation over **HTML Table**. It can choose not to be a table by just adjusting its CSS properties. Something that **HTML Table** is incapable of. And with that, you are able to unlock a whole lot of possibilities by cherry-picking the layout goodies you get from table.

Below are some `display:table` examples that you might find useful:

Dynamic Vertical Center-Alignment

Click on the button to add more lines.

HTML LESS JS Result Edit on

Double Line
Double Line

Add more line

This is perhaps the most common use-case for `display:table`. With it, you can achieve a true vertical alignment (right in the middle) for elements with dynamic height.

There is another shorter way of vertical aligning a element that might interest you:

This gist is labeled this way for a reason. But that's another story for another day.

```
1  .element {
2    position: relative;
3    top: 50%;
4    transform: translateY(-50%);
5  }
```

raw s://gist.github.com/colintoh/62c78414443e758c9991/raw/5bfddd1491f355585cd6e12586d2357862854d17/douchebag-cal-align.css)
douchebag-vertical-align.css (https://gist.github.com/colintoh/62c78414443e758c9991#file-douchebag-vertical-align-css) hosted with ♥ by **GitHub (https://github.com)**

Dynamic Horizontal Center-Alignment

To center-align a dynamic element horizontally, you can set the element to be `display:inline-block`. Then set `text-align:center` on the outer-wrapper of that element. The disadvantage here is the side-effect of text-alignment. All the child element within the outer-wrapper will inherit the `text-align: center` property, causing potential overwriting.

Thanks to @mojtabaseyedi (https://twitter.com/mojtabaseyedi), I found a new way to horizontally center-align a dynamic element with no side effects. Apply `display:table` and `margin: auto` to the dynamic element.

The tweet that inspired this section (https://twitter.com/mojtabaseyedi/status/527418083221454850)

HTML CSS Result Edit on

Dynamic horizontal centering

With display:table

Home About Clients Contact Us

With display: inline-block

Home About Clients Contact Us

Responsive Layout

Drag below 480px to see the responsiveness in action

HTML LESS Result Edit on

Box 1 Box 2 Box 3

As mentioned above, a **CSS Table** can choose not to behave like a table when it want to. By switching the element's `display` property from `table-cell` to `block`, we are able to stack the element.

Stack Ordering

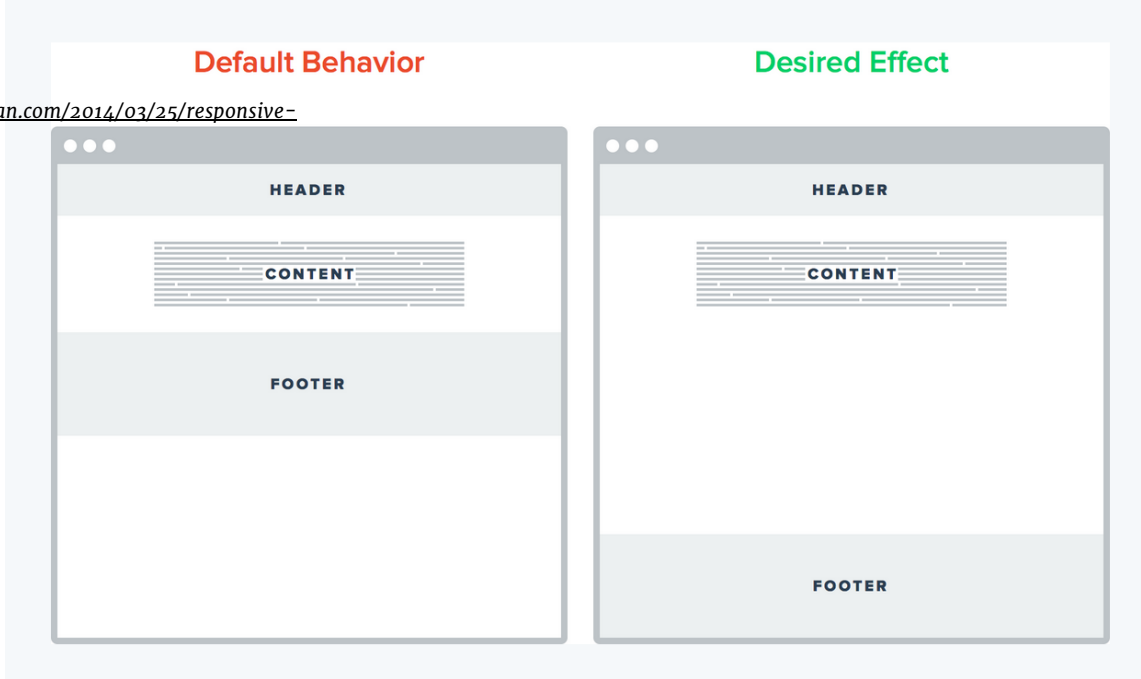
Ordering of the stack have changed from 1,2,3 to 2,3,1

HTML LESS Result Edit on

Box 1 Box 2 Box 3

You can even change the order of the stack. You can read up more about the technique here (http://www.iandevlin.com/blog/2013/06/css/css-stacking-with-display-table).

Dynamic Sticky Footer



Source: galengidman.com (http://galengidman.com/2014/03/25/responsive-flexible-height-sticky-footers-in-css/)

A sticky footer needs to meet these two criteria:

1. Footer needs to stick to the bottom of the page when the content of the main body is insufficient to exceed page height.
2. Footer will continue to flow as per normal once the content of the body exceeds the page height

HTML LESS JS Result Edit on

This is the main body

Add content to main body

Add content to footer

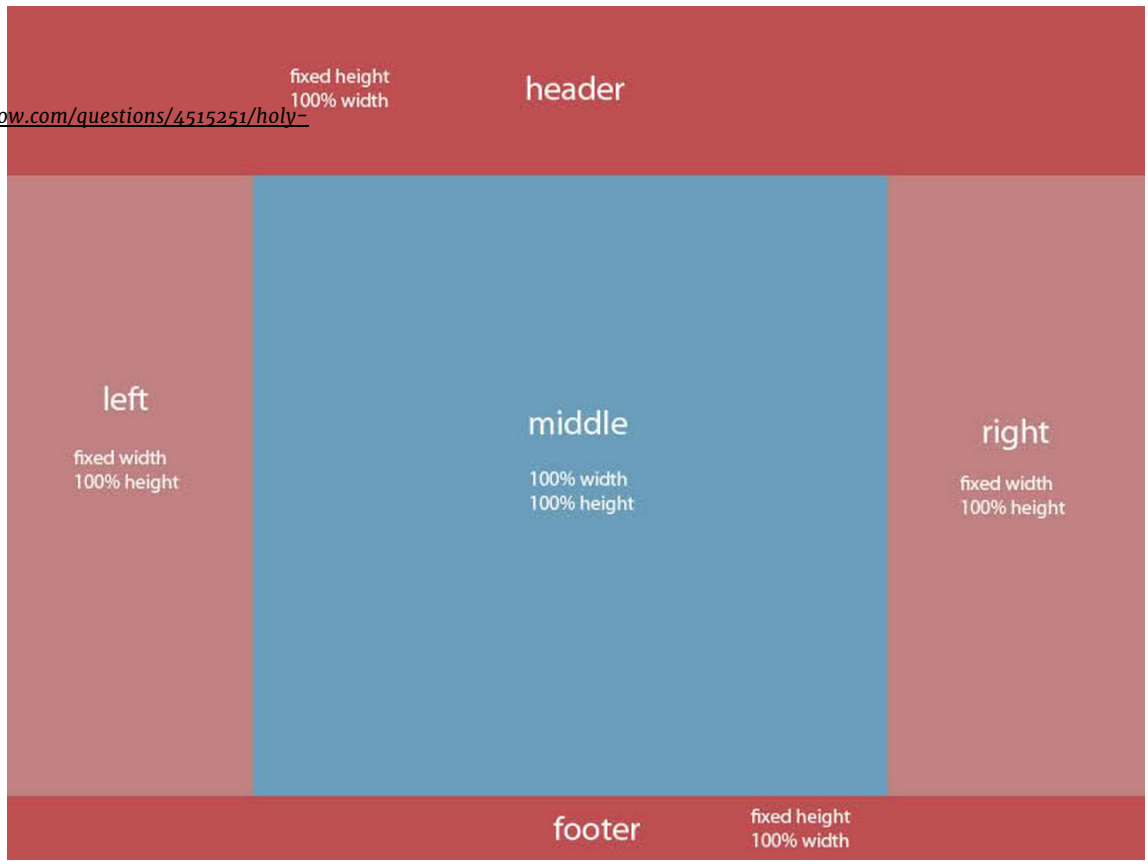
This is a footer

If you ever search for a sticky footer solution, you might have encounter these useful snippets by Chris Coyier (http://css-tricks.com/snippets/css/sticky-footer/) or Ryan Fait (http://ryanfait.com/sticky-footer/).

Their solutions work really well but there is only one disadvantage - the footer must be of fixed height. You can solve this problem with javascript but I will prefer to solve it with CSS. With `display:table`, you are able to create a sticky footer with dynamic height.

Holy Grail Layout

From alistapart (http://alistapart.com/article/holygrail), the Holy Grail layout is a page with a header, 3 equal height columns(2 fixed sidebar and a fluid center) and a sticky footer.



Source: stackoverflow.com (http://stackoverflow.com/questions/4515251/holy-grail-layout-with-100-height)

As much as possible, the Holy Grail layout should achieves these following goals:

1. have a fluid center with fixed width sidebars

2. allow the center column to appear first in the source
3. allow any column to be the tallest

Holy Grail Layout with `display:table`

The above codepen recreates the *Holy Grail layout*. You can resize the window to the fluid middle column and also add more content to test the sticky footer. The only goal that it didn't manage to satisfy is

2) allow the center column to appear first in the source

I'm guessing this goal is for SEO purpose. Hence, if SEO is not of utmost importance, `display:table` will solve this *Holy Grail layout* relatively easily.

Are you kidding? CSS Flex can solve everything!

Indeed it can. Check out Phillip Walton's flex solution (http://philipwalton.github.io/solved-by-flexbox/) to my above examples. However, I'm not rushing into it anytime soon. IE8 and IE9 still make up 32% of the desktop browser market share (http://www.netmarketshare.com/browser-market-share.aspx?qprid=2&qpcustomd=0) and that is a lot to give up if I was to revert to the flex solution. Unless the website serves purely mobile traffic, which I highly doubt so, I will still stick to my `display: table`.

Update: I'm working for a consultancy company thus, desktop compatibility concerns me. But if you only need to cater for mobile browsers, please feel free to flex away.

Conclusion

With the above examples, I hope I'm able to reveal the qualities of the much-maligned `display: table`. However, I have to emphasize that *CSS Table* is not the **silver bullet** for layouts (Read: quirky (https://bugzilla.mozilla.org/show_bug.cgi?id=63895) bugs (https://code.google.com/p/chromium/issues/detail?id=103543)). Use them at the right context and you will save yourself the agony of many late-night CSS hacking.

Update: This article was featured on the front page of Hacker News. For more discussion: https://news.ycombinator.com/item?id=8514717 (https://news.ycombinator.com/item?id=8514717)

Colin Toh | @p0larboy (https://twitter.com/p0larBoy)

Full-stack Javascript Developer at 2359 Media (http://2359media.com/)