HOW TO CREATE HORIZONTAL SCROLLING USING DISPLAY:TABLE-CELL

BY EZEQUIEL BRUNI • HOW TO • FEB 25, 2014

Horizontal layouts of all kinds have intrigued me ever since I found out you *could* do that. I don't know exactly why I'm fascinated with them… maybe it's just my inner rebel talking. Whatever the reason, I'm just about ready to declare vertical scrolling obsolete, passé, and soooo last millennium.

Okay, that's not really what I mean. Still, with the sudden increase in the amount of touch screens lying around, it becomes more difficult to assert that "up and down" are our best options. "Right and left" have become viable directions for content placement, as long as you're not dealing with substantial volumes of text.

I never bothered to really build any horizontal layouts, though. The technical

problems and limitations always seemed to outweigh any stylistic or navigational benefits there might be. That was before, however; and this is now…

I came across the technique described in this article the way I usually come across things: by trying to do something else entirely. I was attempting (you can laugh) to create a CSS grid framework based on display: table-cell (okay, stop laughing now).

Well, for reasons that now seem obvious, it didn't work. You try making a responsive image grid with the table-cell property. Go ahead, I'll wait.

Simply put, table cells are designed to form a single, horizontal row. (I said stop laughing!) That's what they do, and they don't like it when you try to make them do anything else. I gave up on that project. A few weeks later, though, I was considering redesigning my portfolio again.

I thought it would be nice to put all of my projects on one page. I considered several organizational solutions for displaying my web, writing, and photography projects, and came up with this: I want to display these three categories as horizontally scrolling rows of thumbnails.

That's when it hit me: "Table cells would be perfect for that. Also, you can vertically center things inside them! I'm so smart it hurts!" [Some dramatization here.]

I haven't gone and redesigned my site yet, instead, I coded up the two examples of my technique that are in the .zip file linked to at the bottom of this article.

MAKING IT WORK

So, to give you a visual, here's a **demo I've worked up.**

Here's how each row is marked up:

```
<div class="horizontal">
```

```
    <div class="table">
        <article>
            <h3>Project Title</h3>
        </article>
        <!-- Repeat this part as many times as necessary. -->
    </div>
</div>
```

From there, the CSS required to make it work is simple enough:

```
// This container element gives us the scrollbars we want.
div.horizontal {
    width: 100%;
    height: 400px;
    overflow: auto;
}

// table-layout: fixed does a lot of the magic, here. It makes s
.table {
    display: table;
    table-layout: fixed;
    width: 100%;
}

// Arranging your content inside the "cells" is as simple as usi
article {
    width: 400px;
    height: 400px;
    display: table-cell;
    background: #e3e3e3;
    vertical-align: middle;
    text-align: center;
}

// Some styling for contrast.
article:nth-child(2n+2)
{
```

```
    background: #d1d1d1;
}
```

Some horizontal layout techniques require the container element (*div.horizontal,* in this case) to have a defined pixel width equal to the combined width of the elements it contains. Other techniques require display: inline-block; I'm not a fan of this technique. With table-cell, just keep adding elements whenever you need to, and you're good to go — it's perfect for use with a CMS.

MAKING IT FULL-SCREEN

Okay, the other kind of horizontal layout is the full-screen horizontal layout. Creating this with the *table-cell* property requires some JavaScript. I used jQuery to speed things up. The JS requirement might make this technique more situationally useful, but it's still cool.

Here's a **working demo.**

The markup is similar:

```
<div class="horizontal">
    <div class="table">
        <section>
            <h1>Full-Screen Horizontal Layouts</h1>
            <p>Made with <code>display: table-cell;</code></p>
            <p>By Ezequiel Bruni</p>
        </section>
        <!-- Repeat this part as necessary. -->
    </div>
</div>
```

Here, however, it's just one "row" that's been made to fit the size of the browser window. Each *<section>* has, in this case, also been made to fit the browser window.

Here's the CSS:

```
// Don't touch this part. It helps.
html, body {
    width: 100%;
    height: 100%;
    overflow: hidden;
}

// In this case, I didn't want a scrollbar, so I used overflow: h
div.horizontal {
    display: block;
    width: 100%;
    height: 100%;
    overflow: hidden;
    position: static;
}

.table {
    display: table;
    table-layout: fixed;
    width: 100%;
    height: 100%;
}

.table > section {
    width: 1600px; // The width is based on my monitor. It's repl
    height: 100%;
    display: table-cell;
    background: #e3e3e3;
    vertical-align: middle;
    text-align: center;
}
```

As stated above, percentage widths do not work. Pixel widths are required. If you want to make each section fit your window dimensions, you'll need to do it with JavaScript:

```
$(window).load(function() {
    var vWidth = $(window).width();
    var vHeight = $(window).height();
    $('.table > section').css('width', vWidth).css('height', vHe
});

$(window).resize(function() {
    var vWidth = $(window).width();
    var vHeight = $(window).height();
    $('.table > section').css('width', vWidth).css('height', vHe
});
```

You'll notice that I also added the height. Well, that's for Firefox. Firefox doesn't play nice with percentage heights on the table-cell elements (incidentally, Firefox also throws a hissy fit if you make cells relatively positioned, and place absolutely positioned elements inside them).

Well, that's my technique for horizontally placing content. You can**download the source files here.**