CSS display: inline-Block: Why It Rocks, And Why It Sucks

Published on Wednesday, February 24, 2010

Usually when you want a horizontal list, you need to use `float` in the CSS code to make it work, with all its drawbacks. However, there is an alternative with `display: inline-block`.

Problems With float

The problem when you have `float` in your CSS code is that you need to take some precaution to make the surrounding element to encompass the floated elements, and also to avoid following elements in the code to sneak up next to it. Another problem is that if you have a floated list that will take up several rows (visually speaking) and the content is of varying height, you are in for a world of hurt.

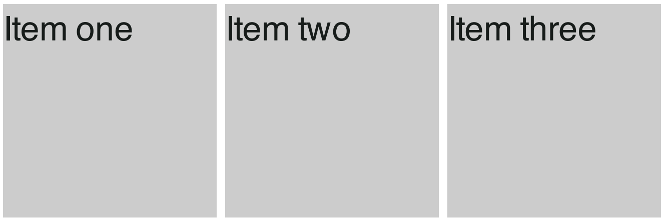
Enter display: inline-Block

This is where the magic value `inline-block` for the `display`property comes into play. Basically, it’s a way to make elements inline, but preserving their block capabilities such as setting width and height, top and bottom margins and paddings etc. A simple example will look like this:

```
<style>
    ul#display-inline-block-example,
    ul#display-inline-block-example li {
        /* Setting a common base */
        margin: 0;
        padding: 0;
    }

    ul#display-inline-block-example li {
        display: inline-block;
        width: 100px;
        min-height: 100px;
        background: #ccc;
    }
</style>

<ul id="display-inline-block-example">
    <li>Item one</li>
    <li>Item two</li>
```

```html
    <li>Item three</li>
</ul>
```

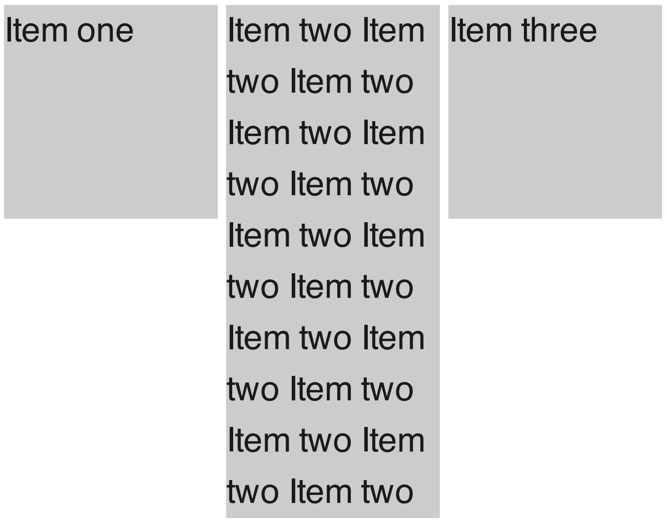
Rendered Results



As you can see, `display: inline-block;` helps us here to render three square gray boxes next to each other. Awesome, right? However, with varying content, we need to add the property `vertical-align: top` to make sure everything is aligned to the top.

```html
<style>
    ul#display-inline-block-example,
    ul#display-inline-block-example li {
        /* Setting a common base */
        margin: 0;
        padding: 0;
    }

    ul#display-inline-block-example li {
        display: inline-block;
        width: 100px;
        min-height: 100px;
        background: #ccc;
        vertical-align: top;
    }
</style>

<ul id="display-inline-block-example">
    <li>Item one</li>
    <li>Item two Item two Item two Item two Item two Item two Item two Item two Item two Item two Item two Item two Item two Item two</li>
    <li>Item three</li>
</ul>
```

Rendered Results



Item one

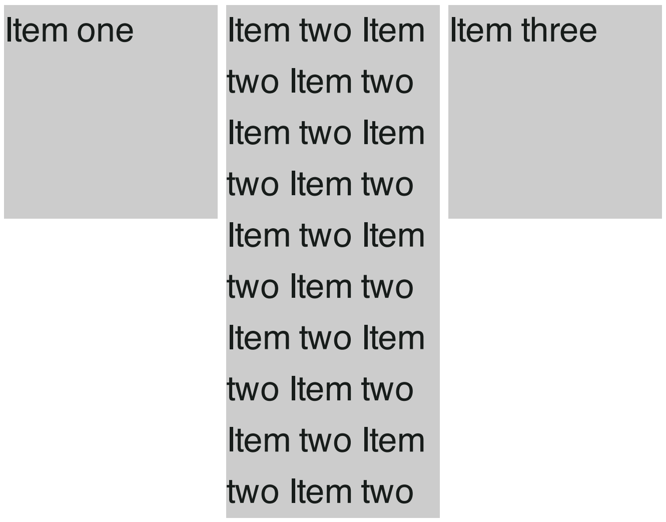
Item two Item two Item two Item two Item two Item two Item two Item two Item two Item two Item two

Item three

This will render just fine in Firefox, Safari, Google Chrome and IE 8. However, for older versions of Internet Explorer, we need to triggerhasLayout and also use a little hack to set the `display` to `inline` (note that this code could be in a separate stylesheet for IE, included viaconditional comments – it's inline here out of simplicity for this example):

```
<style>
    ul#display-inline-block-example,
    ul#display-inline-block-example li {
        /* Setting a common base */
        margin: 0;
        padding: 0;
    }

    ul#display-inline-block-example li {
        display: inline-block;
        width: 100px;
        min-height: 100px;
        background: #ccc;
        vertical-align: top;

        /* For IE 7 */
        zoom: 1;
        *display: inline;
    }
</style>

<ul id="display-inline-block-example">
    <li>Item one</li>
    <li>Item two Item two Item two Item two Item two Item two Item two Item two Item two Item two Item two Item two</li>
    <li>Item three</li>
</ul>
```

Rendered Results That Will Work In IE 7 Too



Item one

Item two Item two Item two Item two Item two Item two Item two Item two Item two Item two Item two Item two Item two Item two Item two

Item three

The Enormous Drawback

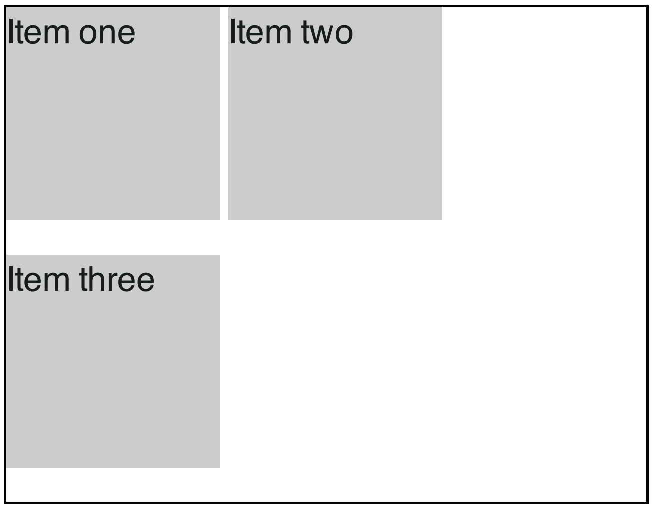
I hope so far that I have managed to get you to realize the potential and simplicity of this approach. However, there's one giant drawback, that might or might not apply to your use case. That is, since the elements become rendered inline, white-space in your HTML code will affect the rendering. That means, if we want perfect size and alignment, but we have space between the `LI` elements in our code example, it will render a 4 pixel margin to the right of each element.

Case in point, let's make it more clear by giving the surrounding `UL`element a fixed width, wherein the `LI` elements *should* fit:

```
<style>
    ul#display-inline-block-example,
    ul#display-inline-block-example li {
        /* Setting a common base */
        margin: 0;
        padding: 0;
    }

    ul#display-inline-block-example {
        width: 300px;
        border: 1px solid #000;
    }

    ul#display-inline-block-example li {
        display: inline-block;
        width: 100px;
        min-height: 100px;
        background: #ccc;
        vertical-align: top;

        /* For IE 7 */
        zoom: 1;
        *display: inline;
    }
```

```
</style>

<ul id="display-inline-block-example">
    <li>Item one</li>
    <li>Item two</li>
    <li>Item three</li>
</ul>
```

Rendered Results



As you can see, the third item will now fall down to the next row. Which sucks! Hard! However, if we were to place the `LI` elements directly after each other, they would all be rendered in the same line:

```
<ul id="display-inline-block-example">
    <li>Item one</li><li>Item two</li><li>Item three</li>
</ul>
```

Conclusion

So, the conclusion is that white-space dependent rendering blows! But, there are also great possibilities with `display: inline-block`, so I advise you to try it out, play around with it and see if it works fine in your specific context.