



Articles » Web Development » HTML / CSS » General

# Fluid, Multi-column, Vertically Ordered List using CSS



Richard Atkins

20 Nov 2019 CPOL

Create a fluid, multi-column, vertically ordered list using nested, floating divs

## Introduction

This article describes a novel solution to the problem of creating a dynamic, multi-column, vertically ordered list that fluidly wraps according to the available width while preserving the ordering of items.

## Background

Creating a single-column vertical list using HTML is a doddle, requiring nothing more than the use of the **UL** and **LI** tags or even **BR** tags. This can be expanded to a fixed number of columns using a little server-side coding that breaks the list into equally sized chunks and then inserts it into either a multi-column table or a series of **divs**. However, we recently had a requirement to create a fluid, multi-column vertical list that makes efficient use of the available space by wrapping itself as required. Searching the web for answers produced suggestions that we either use JavaScript or wait for wider support for the CSS3 columns feature! We didn't want to do either, so we came up with what we believe to be a novel way of doing this.

## Using the Code

One solution to this problem that works with horizontally ordered lists is the use of fixed-width **floats**.

```
<div style="float:left;width:120px;">Archery<br />Ballooning<br />Biking</div>  
<div style="float:left;width:120px;">Bungee Jumping<br />Coasteering<br />Dancing</div>  
<div style="float:left;width:120px;">Driving<br />Gliding<br />Motor Racing</div>  
<div style="float:left;width:120px;">Murder Mystery<br />Off-Roading<br />Paintballing</div>
```

With enough **width** (indicated by the dotted rectangle), this code would render as follows:

Archery	Bungee Jumping	Driving	Murder Mystery
Ballooning	Coasteering	Gliding	Off-Roadng
Biking	Dancing	Motor Racing	Paintballing

However, as the available **width** becomes less, the list wraps in such a way that although horizontal ordering might be preserved, vertical ordering is not preserved until full wrapping has been achieved.

Archery	Bungee Jumping	Driving
Ballooning	Coasteering	Gliding
Biking	Dancing	Motor Racing
Murder Mystery		
Off-Roadng		
Paintballing		

Archery	Bungee Jumping
Ballooning	Coasteering
Biking	Dancing
Driving	Murder Mystery
Gliding	Off-Roadng
Motor Racing	Paintballing

Furthermore, if some of the floated **divs** are different heights, even the horizontal ordering can break down as the wrapping **floats** are 'intelligently' placed.

The problem is that we have no way of controlling the wrapping order of **floats**. If we could somehow tell the second **float** to wrap under the first before the fourth wraps under the first, second, and third, we would be fine. However, it's always the last **float** that gets wrapped.

Our solution to this problem is to use nested **floats** and to set the **floats'** **width** and **min-width** attributes, in order to constrain the order in which the blocks of the list are wrapped.

Our prototypical **float** now looks like this:

```
<div style="float:left;width:50%;min-width:110px;"> CONTENT </div>
```

These are nested together such that each pair of adjacent list items or blocks of items, each in a floated **div**, is contained within a superordinate floated **div**.

```
<div style="float:left;width:50%;min-width:110px;">
  <div style="float:left;width:50%;min-width:110px;">Archery
    <br />Ballooning<br />Biking</div>
  <div style="float:left;width:50%;min-width:110px;">Bungee Jumping
    <br />Coasteering<br />Dancing</div>
</div>
<div style="float:left;width:50%;min-width:110px;">
  <div style="float:left;width:50%;min-width:110px;">Driving
    <br />Gliding<br />Motor Racing</div>
  <div style="float:left;width:50%;min-width:110px;">Murder Mystery
    <br />Off-Roading<br />Paintballing</div>
</div>
```

This gives us a DOM structure as below:

As the available **width** decreases, the percentage **widths** result in each column shrinking until a column hits its lower limit **width** as set by the **min-width** attribute. The **divs** that hit this first are the inner **divs** (in red) so as we pass a critical threshold, the outer **divs** (blue) become narrow enough that they are no longer able to hold the inner **divs** without them wrapping, so we see the following:

As you can see, the vertical ordering is preserved.

This can be extended to any number of initial columns as long as it is a power of 2, e.g., 4, 8, 16, or 32 columns, which would respectively require 2, 3, 4, and 5 levels of nesting where each pair of 50% width floated **div**s is contained by a superordinate 50% width floated **div**.

An example of this method in action with 8 columns is available ~~here~~ (link to dead site removed by author).

## Server Side Requirements

The main point of this article was to demonstrate the above 'trick' to building multi-column vertical lists that adapt to the available space. To illustrate the server side requirements, we include a sample method in VB.NET which does the required rendering for a list with 8 columns.

```
Public Function HTMLMultiColumnList(ByVal listitems As List(Of String), _
                                     ByVal widthpixels As Integer) As String
    Dim output As New StringBuilder
    '
    ' Calculate the number of items required in each block, and the number of blocks
    ' that will have an extra item to cope with the remainder.
    '
    Dim itemsperblock As Integer = CInt(Math.Floor((listitems.Count) / 8))
    Dim blockswithextraitems As Integer = listitems.Count - (itemsperblock * 8)
    '
    ' Build the sublists
```

```

'
Dim sublists(8) As List(Of String)
Dim counter As Integer = 0
For i As Integer = 0 To 8 - 1
    sublists(i) = New List(Of String)
    Dim itemsthisblock As Integer = itemsperblock
    If i < blockswithextraitems Then itemsthisblock += 1
    For k As Integer = 0 To itemsthisblock - 1
        If counter < listitems.Count Then
            sublists(i).Add(listitems(counter))
            counter += 1
        End If
    Next
Next
'
' Render the sublists with nested float divs
'
Dim divopen As String = String.Format( _
    "<div style=""float:left;width:50%;min-width:{0}px;"">",
    widthpixels.ToString)
Dim divclose As String = "</div>"

For i As Integer = 0 To 7
    If i Mod 4 = 0 Then output.Append(divopen)
    If i Mod 2 = 0 Then output.Append(divopen)
    output.Append(divopen)
    output.Append(Join(sublists(i).ToArray, "<br />"))
    output.Append(divclose)
    If i Mod 2 = 1 Then output.Append(divclose)
    If i Mod 4 = 3 Then output.Append(divclose)
Next

Return output.ToString
End Function

```

## Points of Interest

The one problem with this method is that the wrapping occurs using modulo 2, i.e., an 8 column list wraps to 4 columns, then 2 columns, then a single column. However, at each stage, the width used by the bottom level **divs** containing the list items is always exactly equal to or greater than 50% of the available width. While it would be nice to have a solution that wrapped 8 into 7 columns, etc., from a visual perspective, this method works rather well.

**N.B.:** Please note that the margins and colored borders in the illustrations above should not be used. The wrapping **divs** must have no borders or margins, or the 50% width value will no longer be correct. If you want to 'decorate' the blocks of list items, then do so using further markup within the innermost floated **divs** however any horizontal decoration would rather defeat the object of having the separate blocks wrap under each other to produce the effect of continuous vertical lists.

## License

This article, along with any associated source code and files, is licensed under [The Code Project Open License \(CPOLE\)](#)

## About the Author



### Richard Atkins




Zero D Ltd.

United Kingdom 

Having enjoyed reasonably successful careers in medical statistics, psychology, lecturing, computer programming and project management, I've finally settled down to just one thing and am running my own software/webservices business (i.e. director, accountant, manager, sales and marketing, senior programmer, DBA, network administrator, IT support engineer, receptionist, administrator, secretary, cleaner, maintenance man, and person who puts the rubbish bags out on Tuesday evenings).

## Comments and Discussions

 **13 messages** have been posted for this article Visit <https://www.codeproject.com/Articles/38964/Fluid-Multi-column-Vertically-Ordered-List-using-C> to post and view comments on this article, or click [here](#) to get a print view with messages.

[Permalink](#)  
[Advertise](#)  
[Privacy](#)  
[Cookies](#)  
[Terms of Use](#)

Article Copyright 2009 by Richard Atkins  
Everything else Copyright © [CodeProject](#),  
1999-2019

Web04 2.8.19119.1