

Fractal CSS Design as a Row Primary Technique

The Impact of Design on Document Parsing

Dr. E. Garcia
admin@miislita.com

Published: January 6, 2010. Last Update: March 1, 2010

Copyright © E. Garcia

Keywords: fractal css design, row primary, document parsing, relevance perception mismatch

Abstract: This paper describes fractal css design as a row primary technique. It is demonstrated that this approach helps to minimize the user-machine relevance perception mismatch. The impact of design on the parsing of documents is demonstrated. It is shown that replacing table-based layouts with CSS tableless design not necessarily produces equivalent layouts.

Introduction

In a recent work (1) we introduced an algorithmic approach for generating complex Web page layouts from simple rules. Our treatment consists in identifying a motifs and its source code (CSS and HTML instructions) and then iterating the motif using fractal theory principles.

This is a *row primary* design technique where the result is a pattern that spans horizontally as a partitioned row. Each of these row layouts can then be placed one after another to resemble a larger table-like layout; i.e., a grid with cells, rows, and columns.

This is a convenient design technique since document parsing is a row primary activity. Accordingly, the purpose of this article is to discuss how these types of layouts, in particular iterated layouts, might impact the linearization and parsing of documents. Our approach should work with modern browsers regardless of support to the CSS Table Model which, by the way, is also a row primary model (2).

Background

When a Web page is linearized, for instance when being parsed by a search engine or screen reader, its source code is processed, one code line at a time. Essentially, the document is interpreted as a text stream (3). The order in which content is positioned in this text stream can be different from the one rendered by a browser or perceived by a user. This might lead to a user-machine relevance perception mismatch (4).

Frequently, what causes this mismatch is that for users the assessment of relevance is a visual experience while for search engines this is a parsing activity. For instance, Web users are prone to assume that text decorated in a particular style (e.g., bolded, underlined, or italicized) or rendered in large letters convey a special meaning or weights more.

Users also rely on visual clues. For instance, many assume that text surrounded by borders or white space and rendered as 'islands' separated from the main content weights more or is a good descriptor of the document. The use of textures, graphics, and sound might also affect their relevance judgments.

Thus, relying on visual clues for judging Web document relevancy is the result of human-computer interactions; i.e., the result of users interacting with the very same technology that creates the documents. For instance, it is customary to assume that text in blue and underlined has to be a link.

a table-based design with a CSS tableless design not necessarily produces equivalent layouts. All the layouts to be discussed were tested with the latest versions of Internet Explorer and Firefox.

Design Techniques

Consider the following row motif.



Figure 1. A 3-column layout.

It resembles a 3-column partitioned layout with width ratios arbitrarily set to 1-3-1 or 20%-60%-20%. This motif can be constructed in different ways; for instance, with the following techniques:

- Design Technique 1. Using HTML and table elements
- Design Technique 2. Using HTML, CSS, and by absolute positioning block-level elements
- Design Technique 3. Using HTML, CSS, and floated block-level elements
- Design Technique 4: Using HTML, CSS, floated block-level elements, and negative margins

The source code instructions for implementing these techniques are given below.

Design Technique 1. Using HTML and table elements

<pre>HTML <?xml version="1.0" encoding="UTF-8"?> <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd"> <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" > <head> <title>3-column HTML table layout example</title> <meta name="keywords" content="table,layout" /> <meta name="description" content="Example of a layout designed with an HTML table." /> </head> <body bgcolor="#ffffff" text="#000000" topmargin="0em" leftmargin="0em" bottommargin="0em" rightmargin="0em"> <table cellpadding="0em" width="100%"> <tr valign="top"> <td width="20%" bgcolor="#cccccc">Left Content ...</td> <td width="60%" bgcolor="#ffffcc">Center Content ...</td> <td width="20%" bgcolor="#cccccc">Right Content ...</td> </tr> </table> </body> </html></pre>
--

Design Technique 2. Using HTML, CSS, and absolute positioning block-level elements

<pre>CSS html,body{font-family:sans-serif,arial,Helvetica;background:#fff,color:#000;vertical-align:top;margin:0em;padding:0em;font-size:100%;line-height:1.125em;height:100%;} div{border:1px solid #000;} div.link-list{width:20%;position:absolute;top:0;padding-left:1%;padding-right:1%;margin-left:0;margin-right:0;} div.main{margin-left:22%;margin-right:22%;padding-left:1%;padding-right:1%;} div.list1{left:0;} div.list2{right:0;}</pre>
<pre>HTML <?xml version="1.0" encoding="UTF-8"?> <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd"> <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" > <head> <title>3-column CSS layout example with absolute positioning</title> <meta name="keywords" content="css,tableless,layout,absolute positioning" /></pre>

```

<meta name="description" content="Example of a tableless 3-column layout with absolute positioning." />
<link rel="stylesheet" type="text/css" href="fracstyle.css" />
</head>
<body>
<div class="main">
Center Content
</div>
<div class="link-list list1">
Left Content
</div>
<div class="link-list list2">
Right Content
</div>
</body>
</html>

```

Design Technique 3. Using HTML, CSS, and floating block-level elements

CSS
<pre> html,body{font-family:sans-serif,arial,Helvetica;background:#fff;color:#000;vertical-align:top;margin:0em;padding:0em;font-size:100%;line-height:1.125em;height:100%;} div.wrapper,div.left,div.right{margin:0em;padding:0em;} div.left{float:left;width:20%;background:#ccc;} div.right{float:right;width:20%;background:#ccc;} div.center{margin:0em auto;padding:0em;width:60%;background:#ffc;} </pre>
HTML
<pre> <?xml version="1.0" encoding="UTF-8"?> <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd"> <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"> <head> <title>3-column CSS layout with floating divisions</title> <meta name="keywords" content="CSS layout, floating divisions" /> <meta name="description" content="Example of a 3-column CSS layout with floating divisions." /> <link rel="stylesheet" type="text/css" href="fracstyle.css" /> </head> <body> <div class="left"><div class="wrapper"> Left Content </div></div> <div class="right"><div class="wrapper"> Right Content </div></div> <div class="center"><div class="wrapper"> Center Content </div></div> </body> </html> </pre>

Design Technique 4: Using HTML, CSS, floating block-level elements, and negative margins

CSS
<pre> html,body{font-family:sans-serif,arial,Helvetica;background:#fff;color:#000;vertical-align:top;margin:0em;padding:0em;font-size:100%;line-height:1.125em;height:100%;} div.wrapper{float:left;width:100%;margin-left:-20%;margin-right:-20%;} div.left{float:left;width:20%;background:#ccc;} div.right{float:right;width:20%;background:#ccc;} div.center{margin-right:20%;margin-left:20%;background:#ffc;} </pre>
HTML
<pre> <?xml version="1.0" encoding="UTF-8"?> <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd"> <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"> <head> <title>3-column CSS layout with floating divisions and negative margins</title> <meta name="keywords" content="CSS layout, floating divisions, negative margins" /> <meta name="description" content="Example of a 3-column CSS layout with floating divisions and negative margins" /> <link rel="stylesheet" type="text/css" href="fracstyle.css" /> </pre>

```

</head>
<body>
<div class="left">
Left Content
</div>
<div class="wrapper">
<div class="center">Center Content</div>
</div>
<div class="right">
Right Content
</div>
</body>
</html>

```

Comparative Analysis

Design Technique 1 is based on using HTML tables. CSS is not required. Design Technique 2 consists in using HTML, CSS, and absolute positioning of division elements. This technique is featured at the W3C site (3).

Design Technique 3 is different from these two. It is based on HTML and CSS, but absolute positioning is not used. Division elements are positioned using the float property.

Design Technique 4 is discussed later. Table 1 shows that during document linearization, a search engine will parse these layouts by reading columns in the following order

Layout	Left Content ...	Center Content ...	Right Content ...
Design Technique	Parsing Order		
1	Left → Center → Right		
2	Center → Left → Right		
3	Left → Right → Center		
4	Left → Center → Right		

Table 1. Order in which a search will parse a 3-column row motif.

Table 2 reveals that if we iterate the corresponding motif by nesting a scaled-down copy into its middle column (Center), a search engine will end parsing the new document in the following order:

Layout	Left Content ...	Center Content ...	Right Content ...
Design Technique	Parsing Order		
1	Left → Center (Left → Center → Right) → Right		
2	Center (Center → Left → Right) → Left → Right		
3	Left → Right → Center (Left → Right → Center)		
4	Left → Center (Left → Center → Right) → Right		

Table 2. Effect of scaling on parsing.

Clearly, the first three techniques do not produce equivalent layouts. We conclude that while a user might perceive these as similar layouts, a search engine will be parsing three dissimilar text streams; i.e., non-equivalent layouts, probably achieving different relevance judgments.

In Design Technique 2, text rendered in the center of a page is the one that comes first in the code, which means that is the first to be parsed by a search engine. If one wants to render the main content of a document in the center of a Web page and allow a search engine to parse this content first, then

this is the prescribed layout. Absolute positioning, however, is an inflexible approach which difficults the design of iterated layouts, in particular if more than one iterations or motifs are used.

In Design Technique 3, text rendered in the center of a page comes last in the code which means that is the last to be parsed by a search engine. If one wants to render the main content of a document in the left side of a Web page and allow a search engine to parse this content first, then this is the prescribed layout. This technique is also suitable for content that goes at the left and right of a centered content, but that some search engines might elect to ignore. Such type of content could be ads, dynamic menus, scripts, encrypted content, etc.

If the goal is the replacement of a 3-column table-based layout with an equivalent tableless layout, Design Technique 4 should be used. This technique works only with browsers and screen readers that support negative margins. It can be used to design iterated layouts, but up to some extent.

Achieving Table-Equivalent Tableless Layouts through Scaling

Through scaling, we can achieve what Design Technique 4 does: a table-equivalent tableless layout. We refer to this as Design Technique 5. For rendering a 3-column layout we proceed as follows.

We first define a 2-column row motif. Let L_0 and $R_0 = 1 - L_0$ be the widths of the left and right columns of this motif, given as a fraction of the window width. Assume $L_0 = 0.20$ and $R_0 = 0.80$, or 20% and 80% of the total width.

By shrinking to 80% the initial motif, rotating it, and then placing it back to the right column of its original version, we obtain a 3-column row layout. However, the width of the right-most column will be 16% of the total width since it is 20% of 80%.

To compensate for this, we readjust the right-most column width to 25%, so that 25% of 80% is 20% of the total width. This is a rescaling operation. The 25% value is obtained by computing the L_0/R_0 ratio. The entire procedure is depicted in Figure 2 and described in the Appendix section.

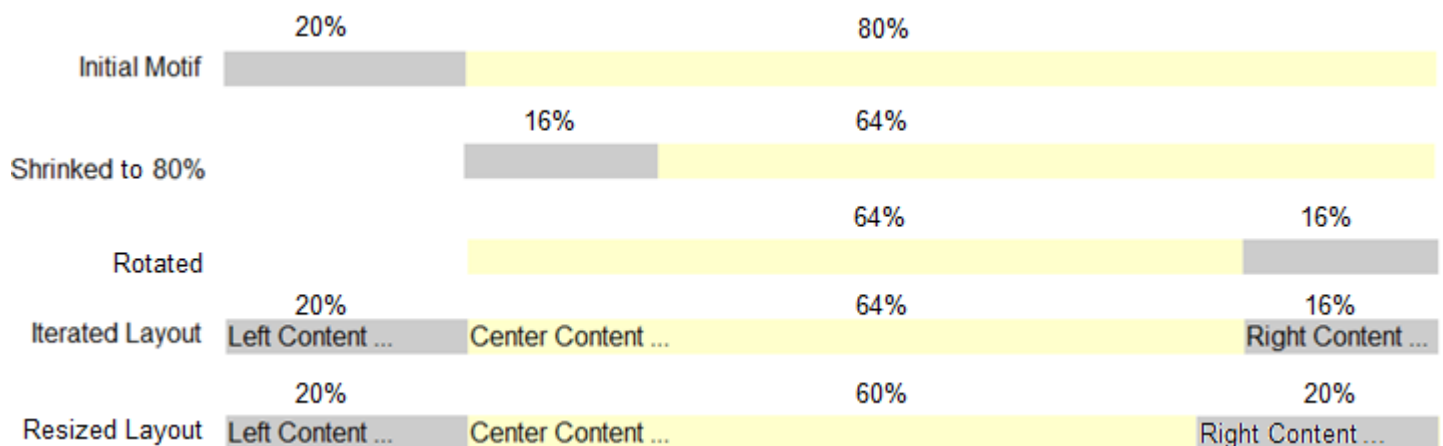


Figure 2. Iterated 2-column row motif.

The width of the middle column is now 75% or 60% of the total width since $0.75 \cdot 80 = 0.60$. This gives us a layout with the desired width ratios of 20%-60%-20%. In this case we need to name the new columns and add two classes to the CSS file (e.g., `div.left1` and `div.right1`) and specify the new widths. The required source code is given below.

Design Technique 5: Using HTML and CSS with iterated motifs

CSS
<pre>html,body{font-family:sans-serif,arial,Helvetica;background:#fff;color:#000;vertical-align:top;margin:0em;padding:0em;font-size:100%;line-height:1.125em;height:100%;} div.wrapper,div.left,div.right,div.left1,div.right1{margin:0em;padding:0em;} div.left{float:left;width:20%;background:#ccc;} div.right{float:right;width:80%;} div.left1{float:left;width:75%;background:#ffc;} div.right1{float:right;width:25%;background:#ccc;}</pre>
HTML
<pre><?xml version="1.0" encoding="UTF-8"?> <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd"> <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"> <head> <title>Iterated layout</title> <meta name="keywords" content="iterated,layout,example" /> <meta name="description" content="Example of an iterated layout" /> <link rel="stylesheet" type="text/css" href="fracstyle.css" /> </head> <body> <div class="left"><div class="wrapper"> Left Content ... </div></div> <div class="right"><div class="wrapper"> <div class="left1"><div class="wrapper"> Center Content ... </div></div> <div class="right1"><div class="wrapper"> Right Content ... </div></div> </div></div> </body> </html></pre>

While this design technique requires more lines of code, all the benefits derived from Design Technique 4 are obtained, without resorting to negative margins. Multiple iterations and motifs can be used, depending on the degree of complexity desired.

Since the new layout is based on scaling and on using resizable units (ems and %), it should readjusts nicely when the user resizes the browser window.

Looking back at Table 2, what Design Techniques 4 and 5 have in common is this: they both are row primary and they both allow the coding of content in the same order as it is displayed by the browser.

Considering that this is how document parsing works, the use of row primary design techniques should minimize a user-machine relevance perception mismatch. This is a convenient design approach.

Additional Work

In Design Technique 5 we did not make all columns of same height. This can be achieved in several ways. One way consists in using the viewport technique; i.e., by adding a `height:100%` property to the body and to the block-level elements, like this:

```
div.wrapper,div.left,div.right,div.left1,div.right1{margin:0em;padding:0em;height:100%;}
```

However as illustrated in Figure 3, column widths will be slaved to the browser viewport. A `div.wrapper{border:1px solid #000;}` property was added to help visualize the layout. In Figure 3, the layout at the right was obtained by iterating the layout at the left.

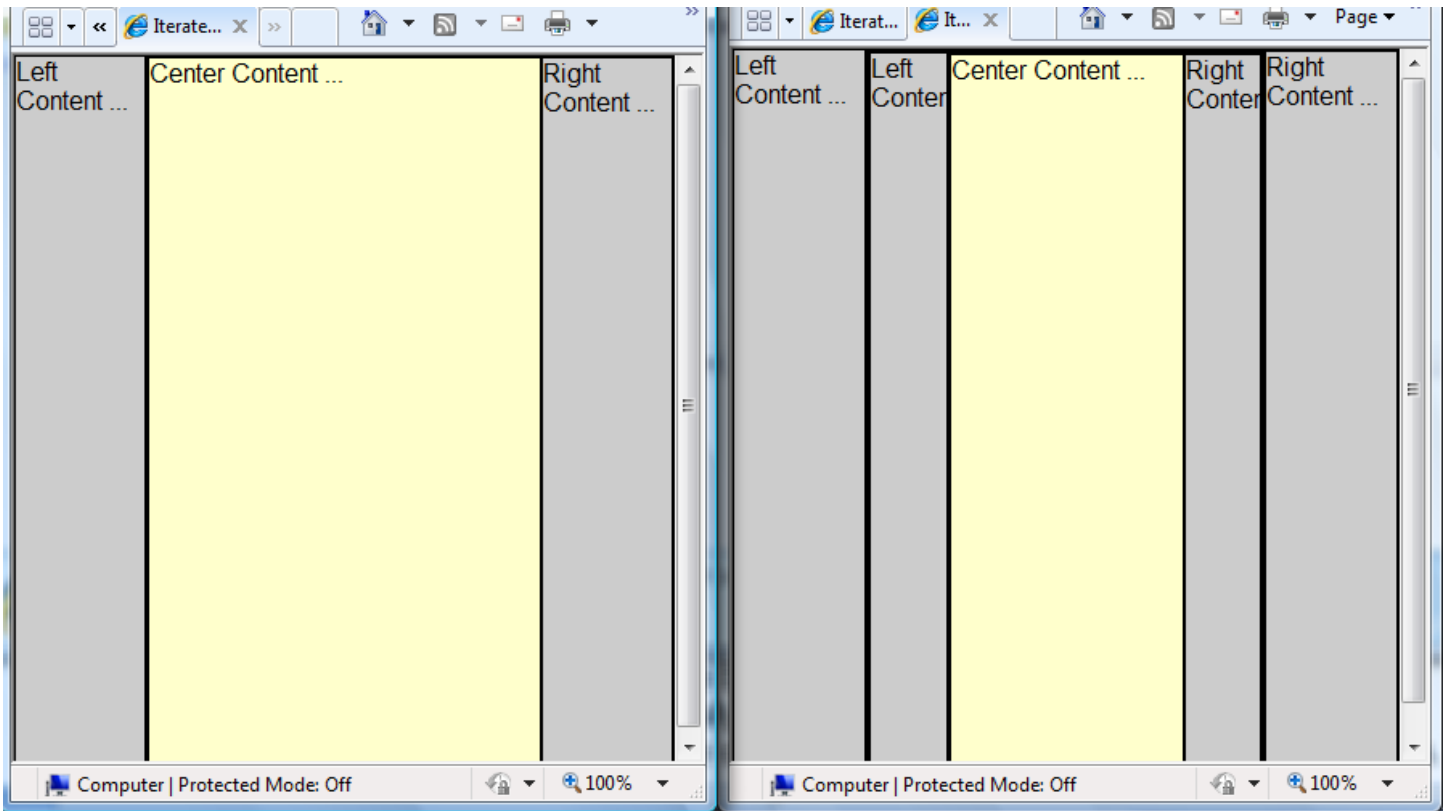


Figure 3. Same high column layouts constructed with Design Technique 5 using the view port technique.

To avoid slaving column heights to the viewport, we can use the Faux Columns technique (8), which consists in adding background images, placed in additional division containers. See Reference 1.

Another alternative is the Equal Height Method (9). In this method, also known as the bottom padding method, an unusually large padding and margin is added to the bottom of the page. This method does not require of background images, but has some drawbacks and workarounds (10).

Conclusion

We have discussed a row primary design technique for iterated Web page layouts. Our approach helps to minimize the user-machine relevance perception mismatch since document parsing itself is a row primary activity. The effect of iterated layouts on parsing has been demonstrated.

In a recent article (11), we demonstrated how our row-primary approach can be used to construct fractal movies, CSS-only backgrounds, and two-column iterated layouts. An entire sub-site on this topic is available at Mi Islita.com (<http://www.miislita.com>).

Appendix – Notes on Scaling and Power Laws

Let L_0 and R_0 be the widths of the left and right columns of a 2-column row motif, given as a fraction of the total window width. Then after iteration i , the left, right, and total widths of the motif are given by the following power laws:

$$\text{Left Column: } L_i = L_0 * (1 - L_0)^i$$

$$\text{Right Column: } R_i = (1 - L_0)^{1+i}$$

$$\text{Width of scaled motif: } W_i = L_i + R_i$$

If $L_0 = 0.20$ (or 20%), Table 3 shows that after few iterations

i	L_i	R_i	L/R_i	$W_i = L_i + R_i$	$(L/R_i) * W_i$
1	0.16	0.64	0.25	0.8	0.2
2	0.128	0.512	0.25	0.64	0.16
3	0.1024	0.4096	0.25	0.512	0.128
4	0.08192	0.32768	0.25	0.4096	0.1024

Table 3. Iterated widths for a 2-column row motif with $L_0 = 0.20$.

Thus when $L_0 = 0.20$, the L_i/R_i ratio will always be 0.25 and is said to be scale-invariant. Multiplying this ratio by W_i resizes L_i as a fraction of the previous scaled motif width. This is how we scaled back widths in Design Technique 5. So, a width of 25% is 20% of the previous width.

References

1. Garcia, E. (2009). Fractal CSS Design.
<http://www.miislita.com/fractals/fractal-css-design.pdf>
2. W3C. Tables.
<http://www.w3.org/TR/CSS21/tables.html#table-display>
3. Garcia, E. (2005). Document Linearization Tutorial
<http://www.miislita.com/information-retrieval-tutorial/indexing.html>
4. Garcia, E. (2008). Users-Machines Perception of Relevance
<http://irthoughts.wordpress.com/2007/05/02/users-machines-perception-of-relevance/>
5. Nielsen, J. (2006). F-Shaped Pattern For Reading Web Content
http://www.useit.com/alertbox/reading_pattern.html
6. Shrestha, S. and Lenz, K. Eye Gaze Patterns while Searching vs. Browsing a Website Usability News, (2007). Vol. 9 Issue 1.
<http://psychology.wichita.edu/surl/usabilitynews/91/eyegaze.asp>
7. Garcia, E. (2005). The Keyword Density of Non-Sense.
<http://www.miislita.com/fractals/keyword-density-optimization.html>
8. Dan Cederholm, D. (2004). Faux Columns
<http://www.alistapart.com/articles/fauxcolumns/>
9. Robinson, A (2005). Equal Height Columns
<http://www.positioniseverything.net/articles/onetrue/layout/equalheight>
10. Robinson, A. (2005) Problems with the Equal Height Columns Method
<http://www.positioniseverything.net/articles/onetrue/layout/appendix/equalheightproblems>
11. Garcia, E. (2010) Fractal Movies, CSS-only Backgrounds, and Two-Column Layouts
<http://www.miislita.com/fractals/fractal-movies.pdf>