Breaking Out of the Box

By Jina Bolton

May 16th 2007

Reader Rating: 9.2

One of the most commonly used arguments regarding CSS [1] that I hear from designers who don't like web standards is, "CSS designs are so boring. They're too boxy." I can't help but find this statement to be a bit silly. To me, that's like saying buildings built with wood are boxy. An architect can create a boring-looking building. But, using the same tools, an architect can create a stunning work of art. It all depends on creativity and experimentation.

It's true that CSS is heavily reliant upon a grid -- everything flows on x and y axes (and can be positioned as such, much like designs built in Photoshop). However, this doesn't mean that your design has to be boring. If you understand how the grid works, you can fracture or abstract that grid to make your layout more dynamic and interesting. In achieving this goal (while supporting flexibility and maintainability), CSS designs have so much more to offer than table-based layouts.

Before you begin, be sure to download the files [2] that I used to produce these examples, so you can follow along.

Grids and Wireframes

"You have to know the rules before you can break them."

I heard this common saying throughout my design and art classes when I was studying. If you've taken any classes in typography or layout, you know that the grid is one of the most important essentials in design. However, if you aren't familiar with this concept, here's a very brief overview:

- A grid provides organization. A grid can act as your page layout's "blue plan." Well-planned
grids can be aesthetically pleasing, as well as providing organization to your content.

- **A grid can define more information.** A well-organized layout can help provide consistency from page to page, and define what type of information is being provided at various points (for example: a sidebar can be used to provide secondary information; copyright and legal information, etc. might be placed in the page footer, and so on).

- **A grid can unify a series.** Having a planned guide for the placement of logos, headlines, images, and/or text in any medium (business cards, advertisements, web site layouts, etc.) can help "tie" everything together. Therefore, a grid can be a very important element of a brand.

For web design, wireframes are used to plan a site's grid. [Wireframes](http://www.sitepoint.com/print/breaking-out-of-the-box) are a basic visual guide used to suggest the layout and placement of fundamental design elements within the interface.

Wireframes are a great way to design your site's layout without actually worrying about the artistic details. They allow you to focus purely on the placement of your content (navigation, body copy, headlines, etc.). This technique is practiced in print design as well, but for this article, we're going to focus on the Web.

Wireframes also provide a guide for your site's markup and style sheet. For example, if your site has a multi-column layout (a content area, and one or two sidebars), you can use markup (usually a `<div>` tag, though this can vary depending on the content) to define those areas, styling them into columns, rows, or whatever is determined by your wireframe. You can take the same approach to style your header, or masthead, and footer.

Knowing the "setup" on the front end will give you more insight into how the site should be structured (which is a good thing to have in the back of your mind when you do start applying artistic detail).

Now, normally, in the typical design workflows with which I've been involved, the design is finalized after the wireframe is created. The content is then created before any coding begins. However, for the purposes of this article, the content is being created on a whim, so the code will come into play early on, just for learning purposes. Once this process comes more naturally, a "regular" workflow can be established. But for now, let's just explore the power of CSS. (Wow, that last bit sounded dramatic!)

### Starting our Design

Here's an example of a grid that you may typically see on a web site:
I decided to take advantage of the width of 960px: I divided by five, which gave me 192px. I used this as my grid unit -- columns will be this width, and all other design elements will be proportional to it. The header is 192px tall as well.

The markup and style for this layout could be pretty straightforward; it would probably look something like this HTML (to which I've added dummy copy):

```html
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en">
<head>
<title>Example Layout</title>
<meta http-equiv="content-type" content="text/html; charset=utf-8" />
<link type="text/css" rel="stylesheet" href="style.css" />
</head>
<body>
<div id="page">
  <div id="header">
    <p>Header content</p>
  </div>
  <div id="content-primary">
    <p><strong>Primary content.</strong> Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Suspendisse pharetra, arcu in dapibus sagittis, felis leo rhoncus erat, non porttitor urna libero vitae nibh. Ut convallis eros sed magna.</p>
    <p>Aenean enim purus, adipiscing a, vehicula ut, tempor vitae, justo. Nam laoreet mauris vitae elit. Vivamus eros quam,


&lt;/div&gt;
&lt;div id="content-secondary"

&lt;p&gt;&lt;strong&gt;Secondary content.&lt;/strong&gt; Lorem ipsum dolor sit amet, consectetur adipiscing elit. Suspendisse pharetra, arcu in dapibus sagittis, felis leo rhoncus erat, non porttitor urna libero vitae nibh. Ut convallis eros sed magna. Aenean enim purus, adipiscing a, vehicula ut, tempor vitae, justo. Nam laoreet mauris vitae elit. This is interesting information. Ut erat nibh, aliquet ut, congue at, tempus in, nunc. Ut aliquet leo et lectus volutpat molestie. Vivamus nunc. &lt;a href="#"&gt;Read more&rarr;&lt;/a&gt;&lt;/p&gt;

&lt;/div&gt;
&lt;div id="footer"

&lt;p&gt;Footer&lt;/p&gt;

&lt;/div&gt;
&lt;/body&gt;
&lt;/html&gt;

Here's the CSS:

* {
      margin: 0;
      padding: 0;
}
body {
      text-align: center;
The margins and padding on all elements are reset using an asterisk (*). This sets the styles to be applied to everything. Read more on this technique, as documented by Andrew Krespanis [8].

This example uses the standard method of centering the design: I’ve set text-align: center for the body, margin: 0 auto, a width (in this case, 960px), and text-align: left for the containing div (which I’ve called "page").

Next, the content divs (content-primary and content-secondary) float to the right. content-primary appears first in the markup, so this is the direction needed to align the columns properly next to each other. Finally, the columns are cleared in the footer.

So far, this is what we have:
Adding the Backgrounds

Let's turn on some backgrounds so we can see the boundaries a little more clearly. Note that, for this example, I'm using the faux-column technique [9] to create the columns. This technique basically takes a background image and repeats it vertically to create the first two column colors (and avoid uneven heights):

The CSS that makes this happen can be found in the #page selector:

```
#page {
  margin: 0 auto;
  width: 960px;
  background: url(page.gif) repeat-y;
  text-align: left;
}
```

Now that we have the basic structure, we'll add some margins and padding to the columns to create some whitespace.

The width of our page -- 960px -- divides evenly by 12, so we'll use this value as the basis for our margins and padding (and later we'll use it for font size). To provide a little more breathing room, let's double it and use 24px for our margins. Whitespace is important. We'll then subtract 48px from each column to accommodate our new margins. Here's the CSS:

```
* {
  margin: 0;
  padding: 0;
}
```
body {
    background: #eee;
    text-align: center;
}
#page {
    margin: 0 auto;
    width: 960px;
    background: #fff url(page.gif) repeat-y;
    text-align: left;
}
#header {
    height: 192px;
    background: #966;
}
#header p {
    padding: 24px;
}
#content-primary {
    float: right;
    margin: 24px;
    width: 720px;
}
#content-secondary {
    float: right;
    margin: 24px;
    width: 144px;
}
#footer {
    clear: right;
    background: #333;
}
#footer p {
    padding: 24px;
}
.clear {
    clear: both;
}

So far, this is what we have.
As you can see, we essentially have our wireframe in place -- with added colors and content. Now we can move beyond the wireframe and get a little more experimental with the design.

**Dropping in Content**

Let's go ahead and add some images, a menu, headings, and so on, so that we really have more to work with. This isn't an advanced tutorial, though, so we'll keep things fairly simple. Here's the HTML:

```html
<div id="page">
  <div id="header">
    <p>Header content</p>
    <ul>
      <li><a href="#">Link</a></li>
      <li><a href="#">Link</a></li>
      <li><a href="#">Link</a></li>
    </ul>
  </div>
  <div id="content-primary">
    <h1>Welcome to my awesome website.</h1>
    <h2>Indeed, this is an awesome website.</h2>
    <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Suspendisse pharetra, arcu in dapibus sagittis, felis leo rhoncus erat, non porttitor urna libero vitae nibh. Ut convallis eros sed magna.</p>
    <img src="photo.jpg" alt="" />
    <p>Aenean enim purus, adipiscing a, vehicula ut, tempor vitae, justo. Nam laoreet mauris vitae elit. Vivamus eros quam, euismod bibendum, pellentesque ut, tristique ut, nisl. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Suspendisse non metus. Nullam
```
Here, a menu (or navigation system) has been added. We've also added some headlines and a photograph. Now, I have to admit, the default typographical settings and colors are driving me a little mad, so let's go ahead and add some aesthetic styling to make things look a little nicer. Take a look at this CSS:

```css
* {
  margin: 0;
  padding: 0;
}

body {
  /* Your styles here */
}
```

<h2>Oooh, more copy&hellip;</h2>

<div id="content-secondary">
<h3>Important Information</h3>
<p>Secondary content. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Suspendisse pharetra, arcu in dapibus sagittis, felis leo rhoncus erat, non porttitor urna libero vitae nibh. Ut convallis eros sed magna. Aenean enim purus, adipiscing a, vehicula ut, tempor vitae, justo. Nam laoreet mauris vitae elit. This is interesting information. Ut erat nibh, aliquet ut, congue at, tempus in, nunc. Ut aliquet leo et lectus volutpat molestie. Vivamus nunc. <a href="#">Read more &rarr;</a></p>
</div>

<p>Photograph by <a href="http://jinabolton.com">Jina Bolton</a>.</p>

</div>

<div id="footer">
</div>
Besides setting type styles, we've also adjusted the spacing to give things more breathing room. The footer was set to a height of 192px (not just for consistency -- I like fat footers). I also gave the `img` negative margins so that the padding and borders would extend outside the boundaries, lining it up more nicely.

Let's take a look at our progress.
Now, if all CSS designs looked like this page, I could understand a designer’s concern about CSS layouts being boring. However, as we’re about to see, it doesn’t take much to make things much more interesting.

**Fracturing the Grid**

My design teacher was affectionately called the "Grid Nazi." He would measure everything and make sure all the pieces of a design matched and were lined up. However, though The Grid was very important to him, he stressed that knowing how to fracture the grid in the right way could really bring energy to a design.

So how do you fracture the grid the right way? Well, it's important to keep in mind the fact that proportion is a beautiful thing. I like to break columns up a little by offsetting objects outside of a margin. For this example, we can use the offset image, and also apply a left-side margin to the primary content (to get some more of that whitespace we love). Take a look at this wireframe, in which the image extends beyond the boundaries of the area created by our wireframe:
This feature is easy to accomplish using a negative margin in CSS:

```css
#content-primary {
  float: right;
  margin: 12px 24px 60px 24px;
  width: 528px;
}
#content-primary img {
  float: left;
  margin: -6px 18px 0 -114px;
  padding: 3px;
  border: 3px solid #eee;
  background: #fff;
}
#content-secondary {
  float: right;
  margin: 30px 216px 60px 24px;
  width: 144px;
  font-size: 11px;
  color: #300;
}
```

First, I applied a few rules that exist for aesthetic purposes only -- you can decide to skip this part if you like. I decided I wanted the **baselines** [12] of both the columns to line up properly. To achieve this, I
adjusted the top margins of 

```html
#content-primary
```

and

```html
#content-secondary
```

until the baselines lined up as intended. Then I set a larger margin (of 60px) on the bottoms of these both columns -- I prefer the bottom spacing on the designs I create to be a little larger than expected.

Next, I reduced the width of

```html
#content-primary
```

as is shown in the wireframe image. Now we can give the image a negative margin of 114px, which will allow the image to extend halfway into the margin.

Since we made the

```html
#content-primary
```

smaller, we'll need to give

```html
#content-secondary
```

a larger right margin so that it will sit to the left as intended. I also wanted to extend the border of the

```html
h1
```

out into the margin, just to balance things out a little. You can do this using negative margins, too:

```css
h1 {
  margin-bottom: 11px;
  margin-left: -108px;
  padding: 15px 0 21px 108px;
  border-bottom: 1px solid #ccc;
  font: normal italic 1.5em/18px Georgia, serif;
  color: #633;
}
```

The left-side padding was set to equal the negative margin, so the text will stay in place, but the border extends out as intended.

Another nice technique you can use to break up the "boxy" look is to apply an ornamental background image that can also extend into the margin. We create this look using roughly the same approach we used above:

```css
h2 {
  margin-left: -204px;
  padding: 10px 0 20px 204px;
  background: url(h2_ornament.gif) 3px 0 no-repeat;
  font-size: 1.25em;
  text-transform: uppercase;
  color: #999;
}
```

Little changes like this can be simple and subtle, yet they have the potential to really make a difference and break up straight lines.
While "fracturing the grid" definitely helps, this layout still could use something more.

### Applying Abstraction

Antoni Gaudi [13] believed that the straight line belonged to man, and the curved line belonged to God. If you want to see abstract, take a look at his work.

In this project, we're not exactly going all-out abstract, but it's fun to have the work of the masters in the back of your mind when you're using abstraction. Whether it's used along with a grid, or used on its own (in the vein of Gaudi), abstraction can add that little "something" that's needed to make a design really interesting.

For our layout, the `#header` and `#footer` are quite simple, so let's add some snazzy background images to these sections. Whether it's everyone's favorite rounded corner, a diagonal line, or a swirl, the background image is up to you ... be creative with it.

Let's start with the footer. (There's no particular reason the footer should be first -- I just felt like choosing it.) Let's add a background image that will "decorate" the top of the footer:

The rest of the footer will be filled with the dark gray color. Here's the relevant CSS:
The text inside the footer now should appear pushed down, so that it stays in the dark gray area:

```css
#footer p {
  padding: 132px 24px 0 24px;
}
```

We can create an illusion that the footer extends outside the boundaries of the containing div (#page) by creating a background image for the body. This image will extend the gray area, so it can be a simple, horizontally repeating image:

```css
body {
  background: #eee url(bg.gif) 0 100% repeat-x;
  font: 12px/18px "Lucida Grande", Verdana, sans-serif;
  text-align: center;
  color: #333;
}
```

So now we have a footer that is visually catchier:
Let's do the same for the header. First, we'll add a background image that breaks up the straight-line appearance of the layout:

This image will be positioned toward the bottom of the header, and we'll use the background color to fill the rest of the space:

```css
#header {
  height: 192px;
  background: #966 url(header.gif) bottom no-repeat;
}
```

Next, let's play with the padding for the header content to make it display more nicely around the background image. The approach you take here will vary depending on the content in your header:

```css
#header p {
  float: left;
  padding: 60px 24px 24px 126px;
  font-size: 1.5em;
  line-height: 1em;
  color: #fff;
}
```

We can use this method of applying an additional background image to make the header appear to extend outside of `#page`. To do so easily, we can wrap a div around the page, and use that when we apply the background image (since body is already using an image for the `#footer`). Here's the final HTML:

```html
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
  <head>
    <title>Example Layout</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <link type="text/css" rel="stylesheet" href="style.css"/>
  </head>
  <body>
    <div id="wrap">
      <div id="page">
        <div id="header">
          <p>Header content</p>
        </div>
      </div>
    </div>
  </body>
</html>
```
Welcome to my awesome website.

Indeed, this is an awesome website.

Primary content. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Suspendisse pharetra, arcu in dapibus sagittis, felis leo rhoncus erat, non porttitor urna libero vitae nibh. Ut convallis eros sed magna.

Aenean enim purus, adipiscing a, vehicula ut, tempor vitae, justo. Nam laoreet mauris vitae elit. Vivamus eros quam, euismod bibendum, pellentesque ut, tristique ut, nisl.


Oooh, more copy...


Important Information

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Suspendisse pharetra, arcu in dapibus sagittis, felis leo rhoncus erat, non porttitor urna libero vitae nibh. Ut convallis eros sed magna. Aenean enim purus, adipiscing a, vehicula ut, tempor vitae, justo. Nam laoreet mauris vitae elit. This is interesting information. Ut erat nibh, aliquet ut, congue at, tempus in, nunc. Ut aliquet leo et lectus volutpat...
Then we can add the CSS needed:

```css
#wrap {
  background: url(wrap.gif) repeat-x;
}
```

And here's the final CSS:

```css
* {
  margin: 0;
  padding: 0;
}
body {
  background: #eee url(bg.gif) 0 100% repeat-x;
  font: 12px/18px "Lucida Grande", Verdana, sans-serif;
  text-align: center;
  color: #333;
}
```

```css
a:link, a:visited {
  color: #633;
}
```

```css
a:hover {
  color: #966;
}
```

```css
h1 {
  margin-bottom: 11px;
  margin-left: -108px;
  padding: 15px 0 21px 108px;
  border-bottom: 1px solid #ccc;
  font: normal italic 1.5em/18px Georgia, serif;
  color: #633;
}
```

```css
h2, h3{
```
#header li {
    display: inline;
    margin: 0 12px;
}
#header li a {
    font-size: 1.25em;
    line-height: 1em;
    text-decoration: none;
    color: #fff;
}
#header li a:hover {
    color: #300;
}
#content-primary {
    float: right;
    margin: 12px 24px 60px 24px;
    width: 528px;
}
#content-primary p {
    padding-bottom: 18px;
}
#content-primary img {
    float: left;
    margin: -6px 18px 0 -114px;
    padding: 3px;
    border: 3px solid #eee;
    background: #fff;
}
#content-secondary {
    float: right;
    margin: 30px 216px 60px 24px;
    width: 144px;
    font-size: 11px;
    color: #300;
}
#content-secondary p {
    padding-bottom: 18px;
}
#footer {
    clear: right;
    height: 192px;
    background: #333 url(footer.gif) top no-repeat;
    font-size: 10px;
    line-height: 1em;
    text-shadow: 1px 1px 1px #333;
Here's the result:

![Image]

**Taking it Further**

A lot more can be done to "break out of the box". You can get much more experimental with your background images, or go a little crazy with margins -- both of which are explored in detail in *The Art & Science of CSS* [14], on which I was a co-author. While what we've discussed here was definitely a minimal approach, hopefully it will get you thinking of fun things you can try (and move beyond the thinking that CSS designs have to be boring). So have fun with it. Check out the book, as well as some of my favorite sites that use fun box-breaking techniques:

**Apple iPod Shuffle** [15]: I really like the iPod shuffle and headphones that appear to be "peeking" out from behind the container.
Tingsek [16]: I'm a big fan of horizontal "bars" that fill the entire screen -- I think it's a great way to avoid a totally contained, boxy feeling. The post-it note's angle also helps break the design up a bit, along with the background photo of Tingsek.

CSS Zen Garden, Entry #194 [17]: Background images really help this layout to avoid feeling boxy, even though it follows a basic grid.
Breaking Out of the Box

Back to SitePoint.com