

Weaving a better Web page

A well-designed CSS framework saves time and effort

Skill Level: Intermediate

[Martin Streicher](mailto:martin.streicher@gmail.com) (martin.streicher@gmail.com)

Freelance software developer
Pixel, Byte, and Comma

23 Jun 2009

A Cascading Style Sheets (CSS) framework expedites and simplifies the design and development of Web pages. Moreover, a CSS framework produces more standardized results in all browsers. Here's a look at two CSS frameworks, each with a unique philosophy.

Around 1950, after building scores of his archetypal, L-shaped, Usonian homes, architect Frank Lloyd Wright introduced the *Usonian Automatic*, a novel system of pre-fabricated parts and simple techniques that would allow a layperson to construct most of the structure of a new house. To Wright, the Usonian Automatic house was affordable, purposeful, attractive, and stood to transform housing.

At the core of Wright's system was a standardized 4x12x24-inch concrete or glass block with concave edges. Building a wall, for instance, took three steps. First, blocks were stacked without guides or mortar. Next, transverse and vertical steel rods were inserted between the blocks to form a skeleton. Finally, mortar was pumped or poured into the edge voids to embed the steel, bond the blocks, and solidify the structure. Thus, beyond laying the foundation, the majority of the work could be accomplished by any determined do-it-yourselfer.

While clever, the Usonian Automatic system proved difficult and expensive in practice, much to Wright's dismay. Counter to Wright's goal, many homeowners turned to professional contractors for construction. Nonetheless, Wright's system was a worthy attempt to provide an entire infrastructure— materials, technique, and philosophy—to transform the economics of owning a comfortable home. The Usonian Automatic was quite literally a framework.

Software development frameworks

Software development can also benefit from frameworks. Like the Usonian Automatic, a software framework simplifies construction of code, allowing developers to focus less effort on assembly and more on purpose.

For example, Apple®'s well-known Cocoa framework provides tools, class libraries, technologies, and philosophies that embody the company's approach to application development. By adopting Cocoa, Mac developers need not reinvent the click-wheel. Rather, they can spend time on unique, valuable, and marketable features.

In addition to Cocoa, many other software frameworks are widely available. Zend offers its namesake Zend Framework for PHP development. Django, Rails, and Catalyst provide a Model-View-Controller (MVC)-based framework for Python, Ruby, and Perl applications, respectively. Nokia's Qt is a cross-platform (including Mac OS® X, Linux®, and Microsoft® Windows®, among others) framework for graphical application development.

Indeed, almost every facet of software development proffers at least one framework. This is especially true of any programming art in its second or later generation. Early trial and error leads to better practices and enlightened inventions, subsequently captured in a next-generation framework.

As evidenced by Django and Rails, server-side Web development has certainly matured over iterations. So, too, has browser-centric application development. SproutCore is a rich JavaScript framework for browser-based applications, and Flex and Silverlight each offer a complete application stack.

Oddly, though, Web page construction—a crucial, fundamental task for any online endeavor—has been slow to keep pace. Or, *had* been slow to stay in stride. Over the last two years or so, many Web page frameworks have popped up to make Hypertext Markup Language (HTML) and CSS construction simpler, faster, and more predictable—even standardized—across notoriously fickle browsers.

This article looks at two frameworks for Web page construction: Blueprint and the Yahoo!® User Interface (YUI) Grid. Both frameworks are free to use and permute, according to the terms of the liberal Massachusetts Institute of Technology (MIT) License and Berkeley Software Distribution (BSD) License, respectively. This article explores the novelties and capabilities of each and works through a series of examples. You can choose the option that works best for you.

Introducing Blueprint

Because HTML is a standard and describes structure not style, the Blueprint

framework is entirely realized as CSS. To use Blueprint, you design your Web site with Blueprint in mind and then author or generate HTML as always, attaching Blueprint CSS styles to HTML elements. In fact, because Blueprint provides "shrinkwrap" CSS, you can design your Web pages in HTML rather than using a graphic design program, such as photoshop, to emulate the final page. In a sense, Blueprint provides true WYSIWYG (what you see is what you get) Web page authoring, because your prototype uses the same code as the eventual production site.

Moreover, because Blueprint is designed to mirror the look of a printed page, designing interactively with Blueprint is likely to feel much like working with QuarkXPress or Adobe InDesign. Blueprint styles are based on pixels and an 18-pixel baseline grid. Without much effort or design talent, you can craft professional-looking pages.

Blueprint's (approximately 250) CSS styles are divided into three functional groups, which are described in [Table 1](#).

Table 1. Blueprint's CSS style categories

Category	Description
Reset	Clears all assumptions your browser might make about padding, margins, and type style, among other characteristics. Think of reset as a clean slate: The only styles and effects you see are those Blueprint or you explicitly define. Listing 1 is Blueprint's CSS code for reset.
Grid	Defines a topmost container, a variety of fixed column widths, and many other modifiers to move individual columns left and right, add borders, build attractive forms, and more. One of the grid styles even colors the underlying columns to ease design and debugging. Most of Blueprint's styles are implemented as a CSS class, so you can mix and match effects by assigning one or more classes to an HTML element.
Typography	Controls the appearance of type on the page. HTML type elements are defined in ems, allowing the look of the page to remain coherent and cohesive even if the type is resized. The typography CSS also sets the vertical alignment of all elements to <code>baseline</code> (the last line of CSS code in Listing 1), so that type in all boxes, no matter how deeply nested, aligns.

Listing 1. Blueprint's CSS code for reset

```
margin: 0;
```

```
padding: 0;
border: 0;
font-weight: inherit;
font-style: inherit;
font-size: 100%;
font-family: inherit;
vertical-align: baseline;
```

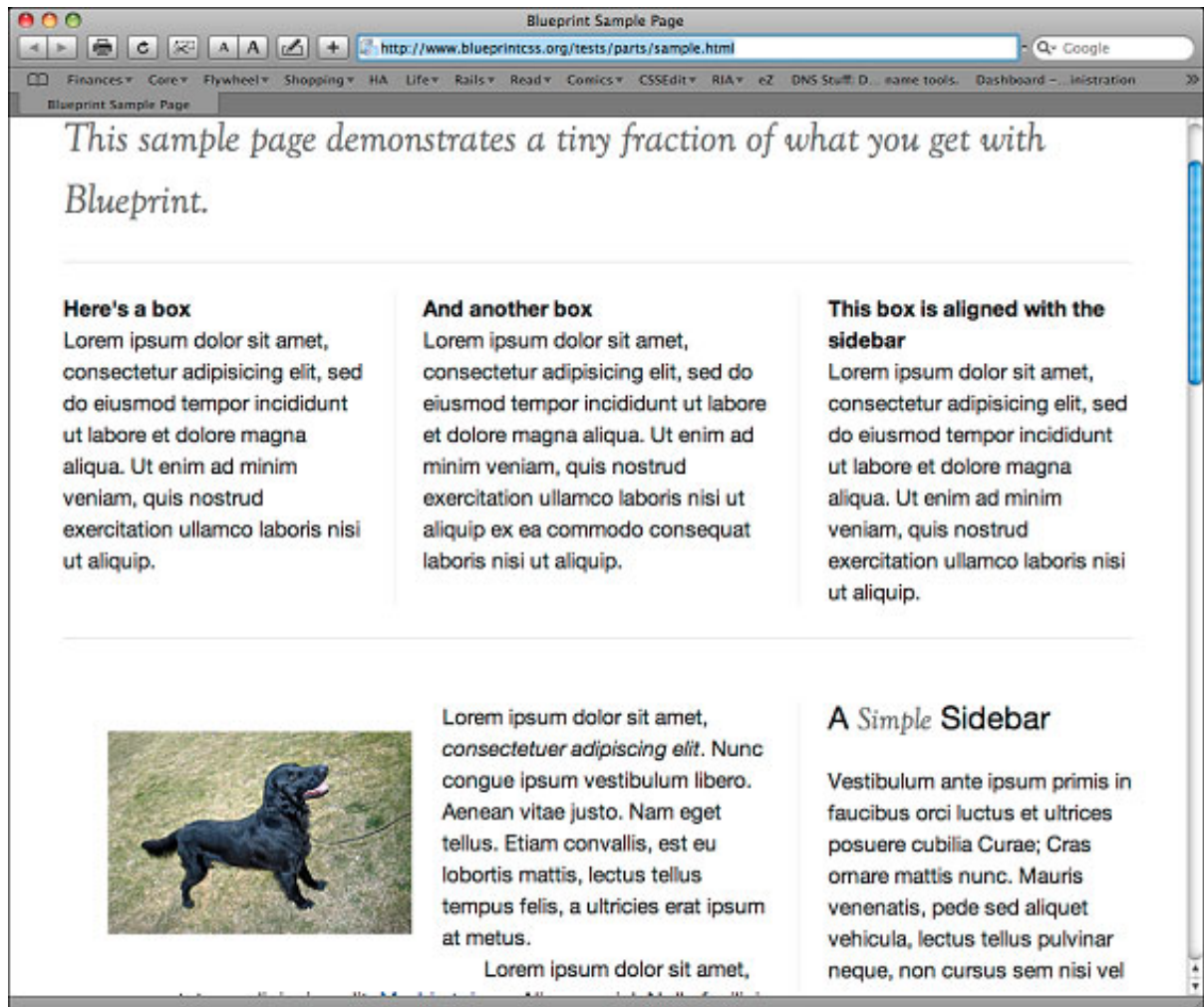
Figure 1 is an example of what Blueprint can achieve. This example is available online (see Resources).

Figure 1. A sample of a Blueprint-styled HTML page



Figure 2 is the same page, albeit with much larger type. The grid remains intact and identical because the dimensions are specified in pixels.

Figure 2. The same HTML page as Figure 1, but with type enlarged via a browser preference



Blueprint's default grid is 950 pixels wide, divided into twenty-four 30-pixel columns separated by 10-pixel gutters: $[(24 \text{ columns} * 30 \text{ pixels/column}) + (23 \text{ gutters} * 10 \text{ pixels/gutter}) = 950 \text{ pixels}]$. If you prefer or require a wider or narrower grid or different column widths, Blueprint offers a Ruby tool to re-generate Blueprint to *your* specifications. The Ruby tool also creates an image of the grid for you to use as a reference in Photoshop, for instance, and it "minifies," or compresses, the resulting CSS to minimize file size, thereby reducing transfer time and bandwidth.

Creating Web pages with Blueprint

To help you get a feel for Blueprint, this article recreates a portion of the sample page shown in [Figure 1](#).

The first step is to include Blueprint's CSS files in your HTML page, as shown in [Listing 2](#). This snippet of HTML assumes you have the Blueprint files in a subdirectory of your Web root named `css/blueprint`.

Listing 2. Include Blueprint's CSS files in your HTML page

```
<head>
<!--[if IE]>
<link media="screen" rel="stylesheet" type="text/css"
      href="css/blueprint/ie.css" />
<![endif]-->
<link media="screen" rel="stylesheet" type="text/css"
      href="css/blueprint/screen.css" />
<link media="print" rel="stylesheet" type="text/css"
      href="css/blueprint/print.css" />
<link rel="stylesheet" type="text/css"
      href="css/custom.css" />
  . . .
</head>
```

There are three standard Blueprint files, which are described in [Table 2](#).

Table 2. Standard Blueprint CSS files

File	Description
ie.css	Specialized code to reset Internet Explorer
screen.css	Holds styles for screen display
print.css	Declares a nice set of default styles for print

In general, you shouldn't edit the Blueprint CSS files. Instead, define your own styles in a separate file and override the Blueprint code as needed. That's the purpose of the last file, `css/custom.css`, which you create and maintain as part of your own code.

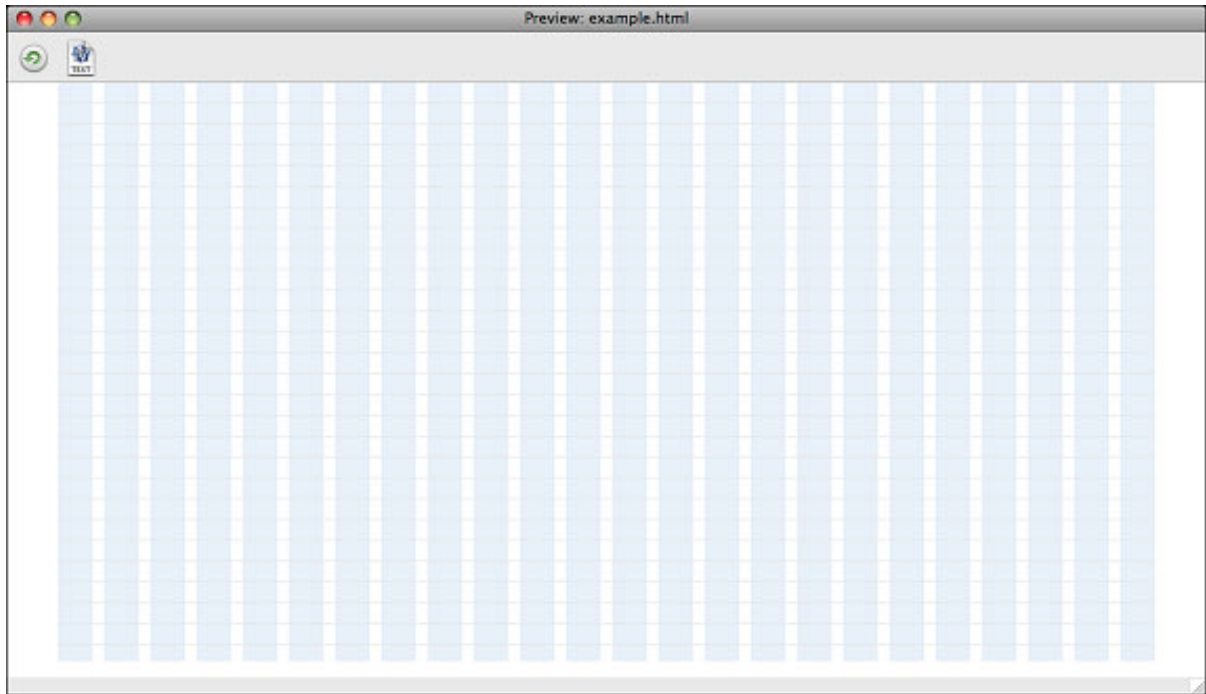
To begin, all Blueprint pages must be contained in a `div` aptly tagged `class="container"`. If you want to see the page's underlying grid, add the `showgrid` class, as shown in [Listing 3](#).

Listing 3. Add the showgrid class

```
<body>
  <div class="container showgrid" style="height:500px;">
  </div><
</body>
```

[Figure 3](#) shows the default Blueprint grid.

Figure 3. The default Blueprint grid



There are two simple rules to follow when you design with Blueprint:

1. Keep the grid in mind.
Blueprint predefines a large number of styles based on multiples of columns. For instance, the `.span-4` style is four columns wide, including the intervening gutters, or 150 pixels. You can attach this style to any kind of HTML element—such as a `textarea`—to make it four columns wide. Obviously, you can apply `.span-4` to a `div`, but be mindful that a `div.span-n`, where n is 1 to 24, floats left and has a right margin of one gutter's width.
2. Keep the baseline in mind.
Blueprint sets the line height for the entire page to 18 pixels, so every image and variant of text must be a multiple of 18 pixels.

To continue with the example (again, referring to [Figure 1](#)), the first and second paragraph on the page are accomplished easily with the `alt` class, shown in [Listing 4](#). The `alt` class is a custom style defined in `custom.css`.

Listing 4. The `alt` class

```
<h1>A simple sample page<
/h1>

<hr><h2 class="alt">This sample page demonstrates
a tiny fraction of what you get with Blueprint.</h2>
<hr>
```

The next portion of the page, the row of three text boxes, is also achieved handily given some of the spans predefined by Blueprint and shown in [Listing 5](#).

Listing 5. Spans predefined by Blueprint

```
<div class="span-7 colborder">
  <h6>Here's a box</h6>
  <p>Lorem ...</p>
</div>

<div class="span-8 colborder">
  <h6>And another box</h6>
  <p>Lorem ...</p>
</div>

<div class="span-7 last">
  <h6>This box is aligned with the sidebar</h6>
  <p>Lorem ...</p>
</div>
<hr>
<hr class="space">
```

As mentioned earlier, a `div` tagged as a `span-n` spans *n* columns, floats left, and has a right margin of one gutter width. The `last` style should be amended to the final column in any series. Surprisingly, `last` does not force a clear to reposition any elements that follow; it merely sets the right margin to 0, which overrides the default right margin of 10 pixels in `span-n`. Instead, the element `hr` forces a clear, precluding any elements that follow from wrapping. `hr class="space"` draws a white line, effectively rendering the rule invisible, but providing a standard vertical blank space.

The `colborder` style creates the gray line between the first and second and second and third boxes. To realize the line, the style expands the right padding to 24 pixels, draws a one-pixel line (in the middle of the column), and extends the right margin to 25 pixels. Because Blueprint is based on pixels, maintaining alignment to the grid is essential.

The next row of content is similar to the previous, but the first column is pushed slightly off the grid with the `prepend-1` modifier, shown in [Listing 6](#).

Listing 6. The `prepend-1` modifier

```
<div class="span-15 prepend-1 colborder">
  <p>
    Lorem ...
  </p>

  <blockquote>
    <p>Integer cursus ornare mauris. ...</p>
  </blockquote>
  ...

  <hr>
  <div class="span-7 colborder">
    <h6>This is a nested column</h6>
```

```
<p>Lorem ipsum ...
</p>
</div>
<div class="span-7 last">
  <h6>This is another nested column</h6>
  <p>Lorem ipsum ...</p>
</div>
</div>
```

The `prepend-v` styles, where `v` is 1 through 23, prepend `v` columns and `v-1` gutters worth of pixels, using `padding-left`.

The image of the dog is 180 pixels tall. Because its height is a multiple of 18, the text displayed to the right of the image and the text displayed below the image remain equidistant on the designated baseline.

The best way to discover all of Blueprint's tricks is to read the `screen.css` file from top to bottom. Here are some highlights and tips:

- The `hide` style class hides any element using `display: none;`.
- The `added` and `removed` style classes are useful in Asynchronous JavaScript + XML (Ajax) applications to highlight elements added and elided from the page, respectively. There is also a class named `highlight` that colors the background of any element yellow.
- You can use `push-i` and `pull-j` to move a column right `i` columns or left `j` columns, respectively.

Again, Blueprint's approach is unique. It enforces a regimented grid usually found in print design. However, that's not necessarily a detractor. The precision gives you great control over how your site looks in any browser.

Introducing YUI Grids

In contrast, the YUI Grids provide CSS for both fixed-width pages and fluid-width pages, and it provides for arbitrarily deep nesting of columns. You can also position and move columns arbitrarily, so you can quickly and simply reorder the content on the page with CSS. For instance, you can move navigation from left to right with one CSS change. Further, you can augment YUI Grids with Yahoo's own JavaScript library to add interactivity. You can read more about the YUI JavaScript library separately. This article focuses on Grids' stylesheets, which can work independent of a JavaScript library.

YUI Grids is similar in many ways to Blueprint:

- It cleans the slate with a set of predictable, baseline styles.

- It defines styles for typography.
- It provides a number of predefined page layouts suitable for most Web applications.

YUI Grids also defines a set of columnar grids that you can nest to create arbitrarily complex pages.

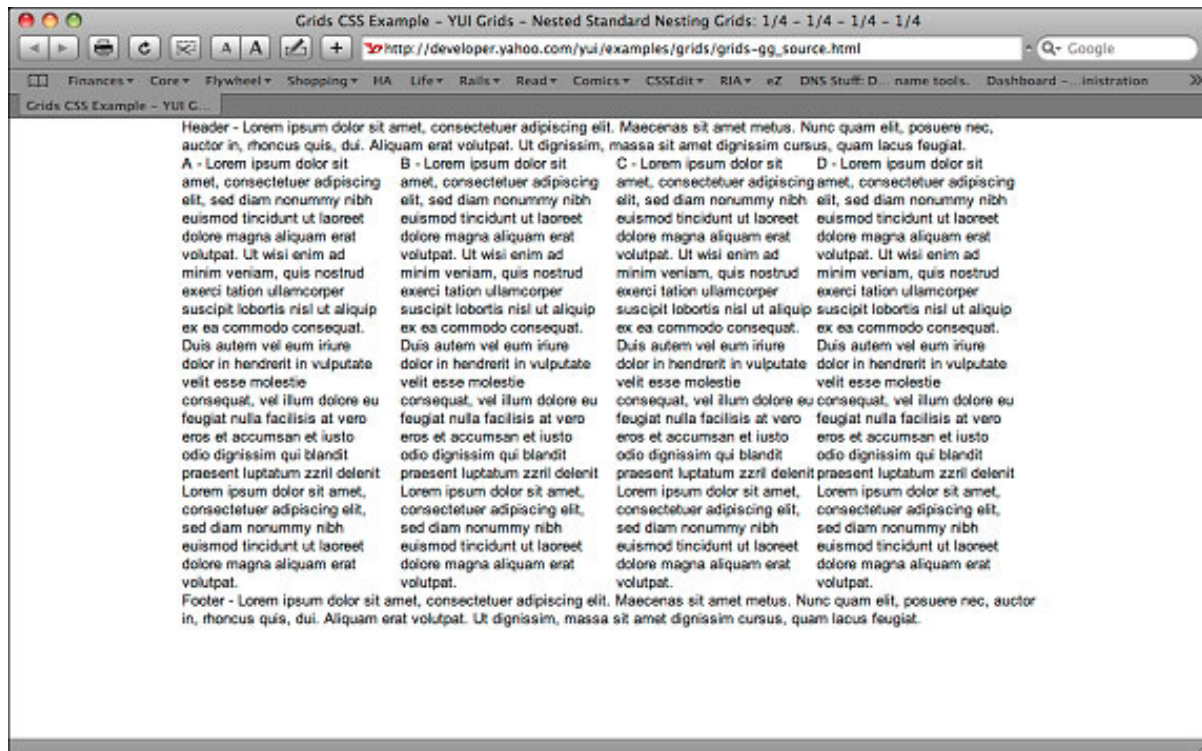
The roughly 100 YUI Grids CSS styles are divided into three categories, which are described in [Table 3](#).

Table 3. The YUI Grids CSS style categories

Category	Description
Document	Define the width of the page.
Template	Describe a variety of popular, two-column page formats. Typically, one column is wider than the other, providing a main content area and a navigation "rail." The template styles also provide variants to swap the order of the two columns.
Grid	Control how two or more <code>div</code> elements appear in a row. The nesting grid sets the width of each embedded <code>div</code> , enables float, and controls when to begin and end a row. Unlike Blueprint, the nesting grid styles use percentages to describe the width of components. Thus, as the page grows in width, so do the entire grid and the individual elements in the grid. However, the components of the grid always maintain relative size, such as 66% and 33%.

[Figure 4](#) captures a four-column layout composed via nesting. There is a header at top and a footer at bottom. The content area itself is a two-column grid, with each of those columns further subdivided into a two-column grid.

Figure 4. A YUI Grids layout with nested diptych grids



Building Web pages with YUI Grids

To use YUI Grids, Yahoo recommends strict behavior in Web browsers that offer multiple rendering modes. You can request strict mode with a leading line in your HTML file:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
```

Next, you must include the YUI Grids CSS. Yahoo hosts a minimized stylesheet for your convenience.

```
<link rel="stylesheet" type="text/css"
href="http://yui.yahooapis.com/2.6.0/build/reset-fonts-grids/reset-fonts-grids.css">
```

(You can also download the file and keep it in your own Web root, or download each of its constituent parts individually if you want to use only a portion of its features. One portion implements reset; another describes typography; and the third defines the grid elements.)

Like Blueprint, a page built with YUI Grids must have standard components, as shown in [Listing 7](#).

Listing 7. The YUI Grids' standard page components

```
<div id="doc">
  <div id="hd"><!-- header --></div>
  <div id="bd"><!-- body --></div>
  <div id="ft"><!-- footer --></div>
</div>
```

The ID of the outermost `div` determines the width of the page. There are four styles, `doc1` through `doc4`. Specify `id="doc1"` for a narrow, 750-pixel page; `id="doc2"` for 950 pixels; and `id="doc4"` for the wider and increasingly common 974-pixel page. `id="doc3"` is a fluid width of 100%. All the page width styles center content.

Site content should be placed within the `div` with ID `bd`.

Content is further divided into two "blocks," one main and one secondary, as shown in [Listing 8](#). (Blocks can be subdivided as needed, as shown in [Figure 4](#).)

Listing 8. Main and secondary blocks

```
<div id="bd">
  <div id="yui-main">
    <div class="yui-b">
    </div>
  </div>

  <div class="yui-b">
  </div>
</div>
```

`yui-b` stands for "block", and there should always be two blocks in a YUI Grids page. By convention, `yui-main` displays on the left, unless you add a class style to the outermost `div` to change the order of the two columns. For example, `<div id="doc" class="yui-t2">` puts the 180-pixel (otherwise rightmost) rail on the left instead.

Given the blocks, you embed a grid in each. In turn, each grid can contain one or more sub-grids or one or more "units." [Listing 9](#) is the HTML and associated CSS to build the four-column layout pictured in [Figure 4](#).

Listing 9. HTML and associated CSS to build the four-column layout in Figure 4

```
<div id="outermost" class="yui-g">
  <div id="firstgrid" class="yui-g first">
    <div class="yui-u first">
    </div>
    <div class="yui-u">
    </div>
  </div>

  <div id="secondgrid" class="yui-g">
```

```
<div class="yui-u first">
</div>
<div class="yui-u">
</div>
</div>
</div>
```

`yui-g` stands for "grid," while `yui-u` stands for "unit." Here, the outermost grid (named simply to differentiate it for this discussion) contains two other grids: `firstgrid` and `secondgrid`. The `first` style indicates the start of a new "row" or sequence of either grids or units. Actual content appears within each unit.

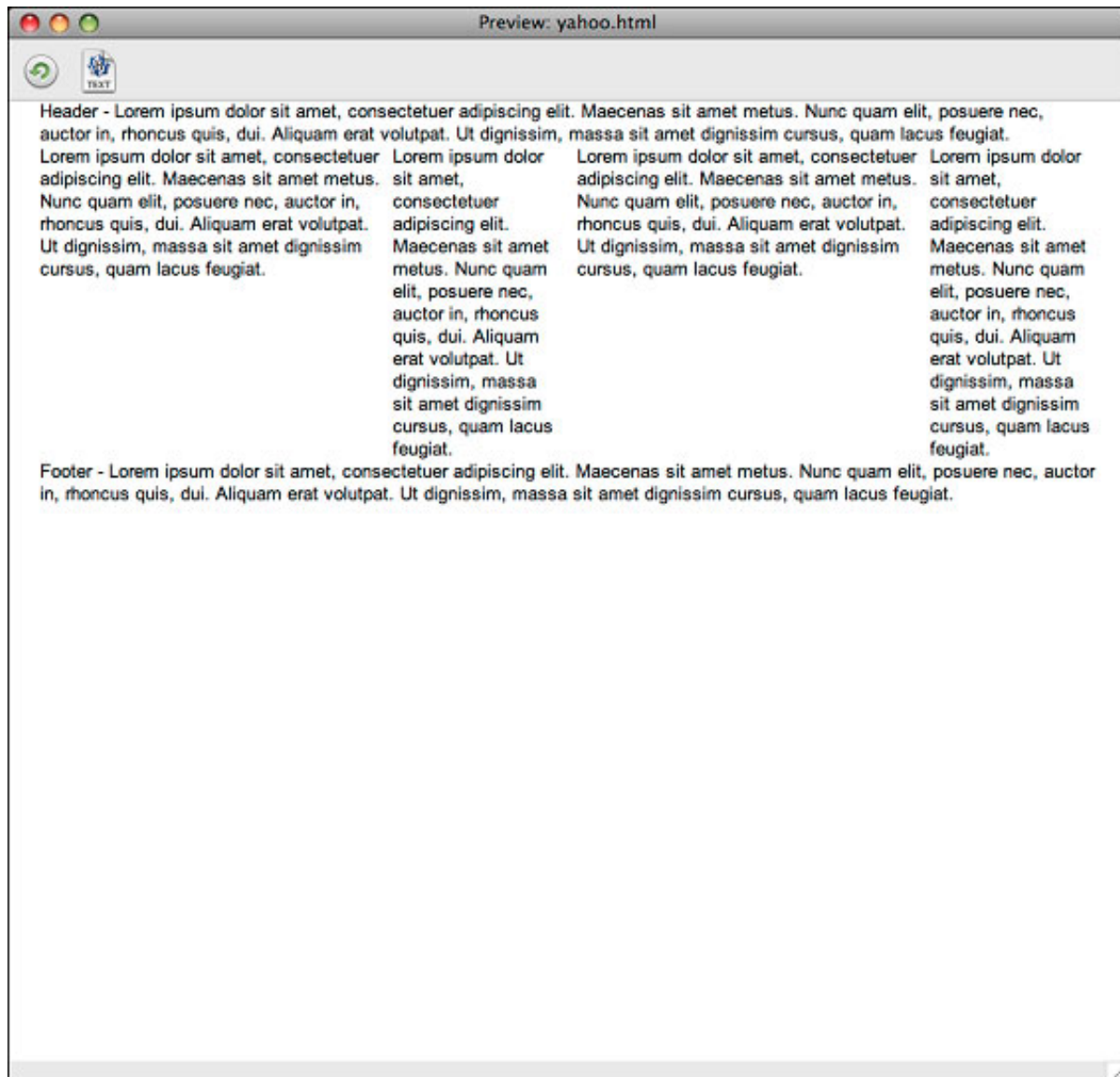
By default, a grid—that is, a `yui-g`—is apportioned equally between two units. You can change this by using a special class to replace `yui-g`. For instance, if you want to subdivide each of the two grids in the previous listing into two units where one unit is twice as wide as the other, you write the code shown in [Listing 10](#).

Listing 10. Subdividing two grids into two units where one is twice as wide as the other

```
<div id="outermost" class="yui-g">
  <div id="firstgrid" class="yui-gc first">
    <div class="yui-u first">
</div>
    <div class="yui-u">
</div>
  </div>
  <div id="secondgrid" class="yui-gc">
    <div class="yui-u first">
</div>
    <div class="yui-u">
</div>
  </div>
</div>
</div>
```

`yui-gc` makes one unit 66% and the other unit 33% of the available width. The result is shown in [Figure 5](#). There are other variants, too, which you can find in the YUI Grids documentation. One interesting exception is grid style `yui-gb`, which divides space equally among *three* embedded units.

Figure 5. You can alter the layout of units within a YUI Grids CSS grid



YUI Grids is a little verbose, but it works well in all browsers and performs adequately on mobile devices. Yahoo claims YUI Grids is used widely within its own site. If so, it's surely well vetted and compatible with all popular browsers.

Save time, look better

In addition to Blueprint and YUI Grids, you can scan the Web to find ten to twelve other frameworks to accelerate the design and development of Web pages. Some of the frameworks are "heavy," dictating the organization of your project and other conventions. Others are "lightweight." In fact, one is just a collection of CSS snippets with explanations. To find all available options, type `CSS framework` into a search engine, peruse the results, and measure if your project can benefit from one or

more.

No matter what framework you choose—and yes, you should choose a framework if you do not already have one—it's sure to save you time and effort.

Wright's Usonian Automatic may not have succeeded, but its spirit lives on.

Resources

Learn

- The Frank Lloyd Wright Archive and About.com ignored multiple requests for images of the block system and one of the well-known Usonian Automatic homes. However, you can find a description of the block and the specialized construction technique in [Concrete Construction](#). You can see snapshots of the Toufic Khalil Usonian Automatic home at [About.com: Architecture](#).
- View the [Blueprint sample page](#) shown in Figures 1 and 2 in this article.
- The [developerWorks Web development zone](#) is packed with tools and information for Web 2.0 development.
- Browse the [technology bookstore](#) for books on these and other technical topics.
- Personalize your developerWorks experience with [My developerWorks](#).

Get products and technologies

- Learn more about [Blueprint](#).
- Delve in to the [Yahoo User Interface library and YUI Grids](#).
- Download [IBM product evaluation versions](#) and get your hands on application development tools and middleware products from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.

About the author

Martin Streicher

Martin Streicher is a freelance Ruby on Rails developer and the former Editor-in-Chief of [Linux Magazine](#). Martin holds a Master of Science degree in computer science from Purdue University and has programmed UNIX-like systems since 1986. You can reach Martin at martin.streicher@gmail.com.