**HTML and CSS: the Good Parts**

**Author: Ben Henick**

**Publisher: O’Reilly Media, Inc.**

**Pub Date: 2/18/2010**

**CSS: The Missing Manual**

**Author: David Sawyer McFarnald**

**Publisher: O’Reilly Media, Inc.**

**Pub Date 8.24.2009**

In this class we will use a combination of HTML, CSS and Javascript to build web pages. The goal of this class is to develop WYSIWIG skills for HTML/CSS/Javascript pages.

The presentation or display of an Web Page is constructed as a composition of different types of browser objects. The simplest web page consists of HTML Elements such as text and Image objects such as .bmp, .png, .gif, or .jpg objects.

The properties of these objects such as the size, and color of the fonts used to display HTML Text can be modified by CSS style rules.

There are 3 existing standards which define current “HTML” elements you will see in webpages today.

1. HTML 4.01: <http://www.w3.org/TR/html401/> Browsers uniformly support this standard starting with <html><body></body></html> tags for a minimum HTML 4.01 document.

**CLASS DEMO 1 HTML PAGE**

1. XHTML 1.0 <http://www.w3.org/TR/xhtml1/> XHTML 1.0 is the extendible version of HTML using XML notation and practices. In addition to adding support for more HTML tags, XHTML requires the inclusion of a DTD and is where you will see a DOCTYPE at the start of the webpage.

**CLASS DEMO 2 XHTML PAGE WITH VALIDATION**

XHTML and HTML documents can be validated using the W3C validator: [http://validator.w3.org/#validate\_by\_upload+with\_options](http://validator.w3.org/" \l "validate_by_upload+with_options)

1. HTML 5: <http://dev.w3.org/html5/spec/Overview.html> . Same story as above, adding more tags. We aren’t going to focus on every small technical difference between the 3 standards. We are going to start reverse engineering existing web pages to develop pixel placement skills.

(redo the current assignments using HTML 5 elements. See if the new standard makes anything we are currently doing “easier”)

**HTML Elements and Styles**

An HTML element consists of a tag and attribute. A tag is something like <h2> or one of these: <http://www.w3schools.com/tags/default.asp> . An HTML tag can contain an attribute which is an additional property to modify the display of the HTML Tag. An example of 2 attribute and values for the attribute FORM is: <FORM action="http://www.google.com" method="post"> or <p style=”color:red”>.

Composing a simple webpage. We need to rethink if the 10 HTML demos below are a good way to teach HTML and CSS in one pass. These demos should teach the student HTML and style rules to build webpages and to reverse engineer the CSS Zen Garden websites.

**10 Demos CLASS DEMO 3**

* **Divs**

A “div” or HTML Division element is probably the most widely used HTML element for building column layouts. Any column based layout like what you see on the news page like the nytimes is generated using divs, not tables.

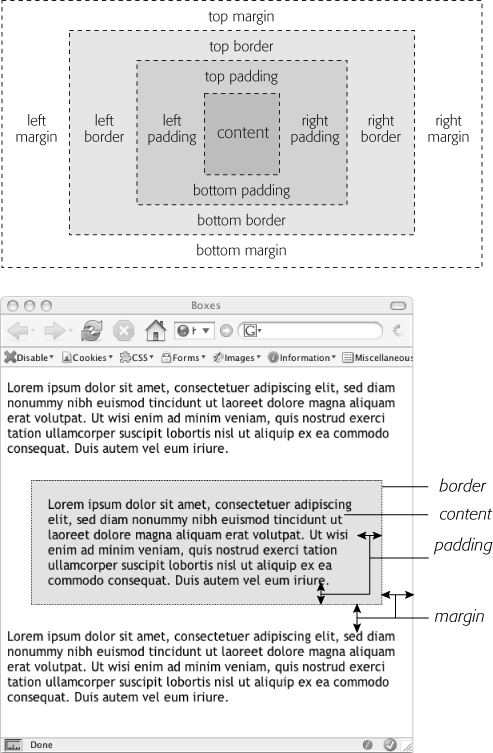
* **Span**

A span tag is used to apply specific style rules to text inside the span tag.

* **Box Model for HTML elements**.

HTML Block level elements (<http://htmlhelp.com/reference/html40/block.html> ) are placed starting on a new line. Each HTML element is surrounded by a border which is invisible on default. Space between the HTML element and the inside of the border is called padding. Space outside of the border is called margin. The border can be made visible in one of the four segments, top, bottom, right, left.

* + Margin, Top, Bottom, Left, Right, (trouble) ; adding space between HTML element content and outside of border. You can use margin-top to separate HTML elements or you can use a div spacer. You can control the width of the div spacer but not the HTML Element width which often determines whether you use margins vs. divs to position HTML elements. To span a webpage you many need to add an invisible div spacer.
  + Border, border-style, border-width, border-color. Borders and Paddings are used for adding space between the HTML Element and borders.
  + Height
  + Width



When working with layouts of HTML elements, boundaries are not always enclosed within the outer margin envelope.

**Collapsing Margin Problem**

Source: <http://www.complexspiral.com/publications/uncollapsing-margins/> This is the best written article on the subject.

* **Floats**

A HTML page layout goes from left to right. The browser places objects starting from the upper left hand corner. The next object starts immediately below the first object. When we add 2 divs we see them stacked on top of each other.

<style type=*"text/css"*>

*#firstdiv*{

border-style:*solid*;

border-width:*1px*;

border-color:*black*;

height:*20px*;

width:*50px*;

background-color:*#FE0*;

}

*#seconddiv*{

border-style:*solid*;

border-width:*1px*;

border-color:*black*;

height:*20px*;

width:*50px*;

background-color:*#0F0*;

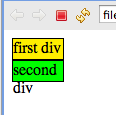
}

</style>

<body>

<div id=*"firstdiv"*>first div</div>

<div id=*"seconddiv"*>second div</div>



Note in the above case the 2 divs get stacked on top of each other and the extra “div” text is on the line below not to the right of the green div box.

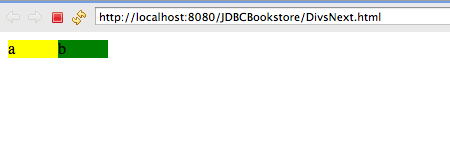
This brings up a couple issues:

1. how do we form text around an image where the “div” text below the green box is to the right of the green box. Use float to float the image to the left.

To get text to float around.

1. how do we stack the yellow div next to the green div like in a column layout?

Encapsulate in left and right divs into another dif, float left and float right, add into single div. Add width 50% into smaller divs.



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

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"*>

<title>Insert title here</title>

</head>

<style type=*"text/css"*>

*#holdColumns*{

background-color:*red*;

width:*100px*;

}

*#column*{

width:*50%*;

}

</style>

<body>

<div id=*"holdColumns"*>

<div id=*"column"* style="background-color:*yellow*; float:*left*">a</div>

<div id=*"column"* style="background-color:*green*; float:*right*">b</div>

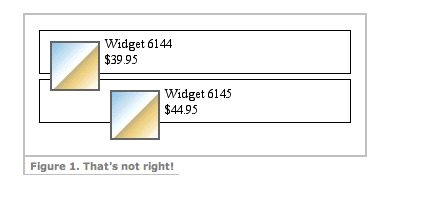
</div>

</body>

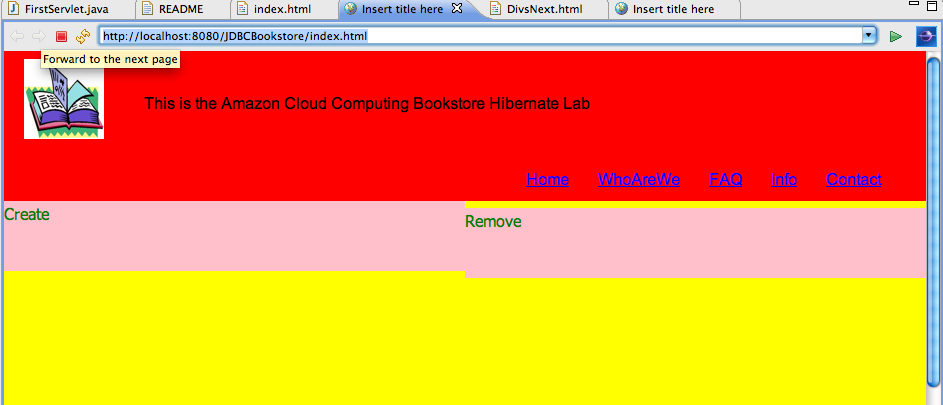
</html>

1. When the floated element, an image in the code below is **taller** than div, the image sticks out of the div as shown here: <http://www.complexspiral.com/publications/containing-floats/>

This is a common problem as you can have a nav bar which also sticks out of the div. It is better to prevent div “stickouts” so you don’t have to hardcode offsets into the rest of the page elements.



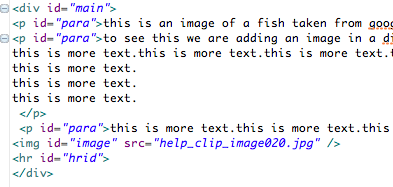
Here is an example of a navbar with div “stickout,” even though this isn’t visible. After adding color to the background you can see the right Column titled Remove sticks about 7px down from the top. This is because the nav bar has some “stickout.”

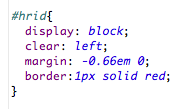


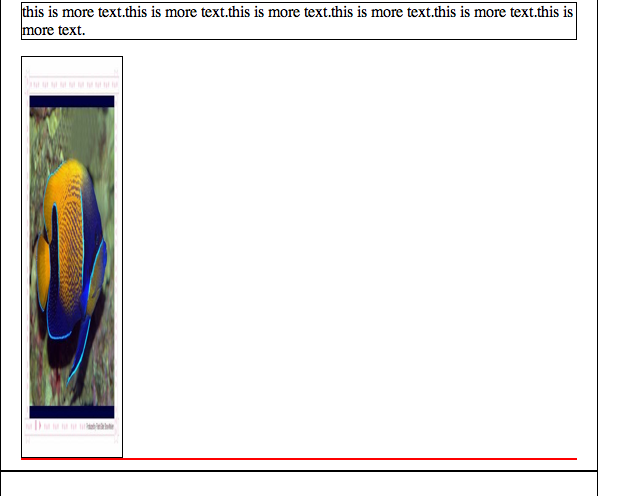
Multiple Solutions:

* + Add a block level element <hr>, make it invisible and add it below the image to extend the div to cover the image. You can use a <hr> or <br>. Below we use a <hr> element after the image. Make it invisible( using visibility:hidden in the style rule).

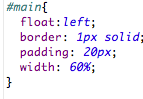
We made this red so we could see where it is in relation to the image.



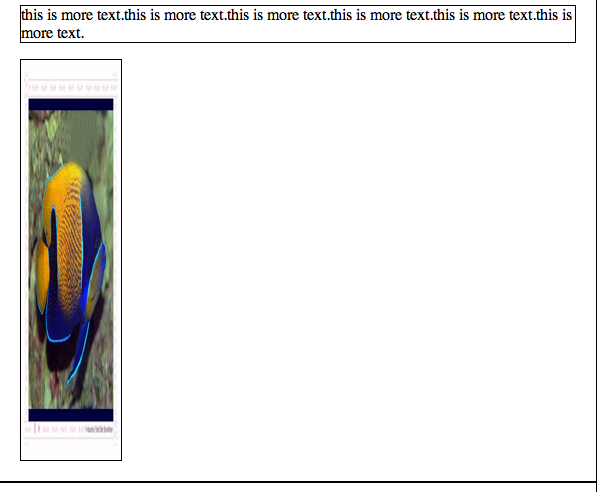


Note: the red line indicating the <hr> element below the image. 

* + Second method is to add float;left to the div containing the image. This automatically causes the div to include the image and not allow it to protrude.



Note in image below the div holds the image, we removed the <hr> element and only added the float:left to the main div.



* Position properties along with floats are used to create multicolumn layouts. The CSS position properties are
  + position:static
  + position:relative
  + position:absolute

The default rendering position for all HTML elements is position:static. The element is not positioned and layout occurs as part of the normal flow.

Position:relative has parameters top, bottom, left, right to move the html element as a relative offset to the original position.

Position:absolute **removes** the element from the layout flow and places it in a **separate absolute coordinate system in reference to the parent** HTML element. This element never moves in relation to other elements. The element is fixed.

Relative Position Demo: The following code moves the div down 100px and to the right 100px.

<style type=*"text/css"*>

*#relative*{

position:*relative*;

top:*100px*;

left:*100px*;

style: *solid 1px red*;

background-color:*green*;

width:*100px*;

height:*100px*;

}

</style>

<body>

<div id=*"relative"*>

this is relative div

</div>

</body>



Fixe

* FORMS, textboxes, textarea, BUTTONS submit GET/POST, Href link

Install the Tomcat server and run a basic servlet to verify the server is working. We are going to create forms which communicate with the server.

* Li, ul for lists, horizontal and vertical. Emphasize Horizontal Lists for navbars. Cover highlighting and selection.
* Table rows, columns, td, tr, tbody, colgroup borders around cells
* Tooltips using <title>.
* **Placeholder for other demos….**
* Dir/font/strike/center (deprecated HTML elements using style instead)
* Text
  + Word-spacing (ems)

word-spacing: 2px;

* + Letter-spacing

letter-spacing: -1px;

font-family: Arial; Serif(with tail,for reading paragraphs), San-serif(headlines).

font-family: courier (program code, java, js, etc…)

font-weight: bold;

font-size: 1em;

font-size: 10px;

font-size: large; xx-small, x-small, small, medium, large, x-large, and xx-large

font-size: 200%;

A “em” is a typographic convention related to the size of the capital letter M for a base reference size. Scale up/down from this capital letter. Use em, inches, pixels, mm, cm.

* + Line-height:

line-height: 1.5;

There is no unit for line-height. 1.5 is the multiplied by font-size in pixels to determine the spacing between lines. For a font with pixel height 10px, line height is 15px.

* + Text-align:

text-align: center;

text-align: left;

text-align: right;

* + Vertical-align
  + Text-indent
  + Text-decoration, controls if an item is underlined. Very useful in lists to remove the default underline. Blink, overline not commonly used.

text-decoration: none;

text-decoration: underline overline;

* + Text-transform: capitalize, uppercase, lowercase(different from smallcaps)

text-transform: uppercase;

text-transform: none;

text-transform: lowercase;

font-variant: small-caps;

**DOM(move to later for JS section)**

Browsers store HTML page in a tree like structure called a Document Object Model. You can see this by installing Firebug and clicking on the DOM tab.

<DEMO HELLO WORLD WEBPAGE>

**CSS Style Rules**

CSS Style rules are applied to HTML Elements. This was part of a design philosophy to separate code from presentation. HTML was the code and CSS would define presentation. In theory the same code could be used to create many different “looks” or presentations for the same HTML page. <http://www.mezzoblue.com/zengarden/alldesigns/> shows the same HTML code with different CSS rules applied to them.

There are 4 ways to add style rules to HTML

1. using the style attribute inside the HTML element. This isn’t recommended for production. The format of a style attribute inside an HTML element is: ***style=”property:value”*** We see this in the div code examples below.

<div id=*"div1"* style="height:*100px*;width:*100px*;background-color:*blue*">this is div 1</div>

where height, weight, background-color are the name of the attributes and values of the attribute are assigned with colons and separated using semicolons. This creates a blue box. Divs are probably the most widely used HTML element used to create columns and layouts.

We can also next divs within each other. Most HTML elements do not allow this. Here is a div inside another div:

<div id=*"outerdiv"* style="background-color:*orange*; height:*200px*; width:*200px*">this is the outer div

<div id=*"innerdiv"* style="background-color:*yellow*; height:*100px*; width:*100px*">inner div</div>

</div>

* All dimensions in CSS require a dimension except for 0.

body {

margin:0;

height:0;

}

* Colors: colors can be expressed using English color names for one of the 16 primary HTML colors. (list the color names here).
* Creating custom colors outside these 16 color names requires calculating a numeric color value representing a mixture of RGB. The values are between 0-255 converted to hex. (Show website for decimal to hex conversion). Color: #ADFF2F is a mixture of R=, G=, B= ;
* 16 bit custom colors You can do a shorthand simplifying the RGB to single digit numbers where the range is 0-15 for RGB instead.
* Color Percentages: you can specify percentages in RGB: (100%, 20%, 50%)
* Most web sites use hexadecimal 0-255 colora ganes in RGB format.
* Background color: this is transparent by default which means what is underneath shows through.
  + Background-image
  + Background-repeat: default is repeat unless you turn this off.
  + Background-position
  + Background-attachment
  + Background…

1. Use a style rule inside style tags where the <style> tags are inserted before the </head> tag. The format of a style rule in this mode is

selector { property:”value”; property:”value”;}. Note: make sure you add the semicolon to the last value. There are 3 fiormats for the selectors

* Use an HTML element as a selector

p { color:”blue”} sets all the text in paragraph tags to blue .

* #firstID { color:”blue”} sets all the HTML Element tags with the id=firstID to blue, if there is an attribute named color for the element. Note the # notation specifies value of id.
* .firstClass { color:”blue”} sets all the HTML Element tags with the class=firstClass to blue, if there is an attribute named color for the element. Note the . notation specifies value of class.

<style>

#div1{

color:”red”;

}

</style>

<div id=’div1’>

or

<style>

.div1{

color:”red”;

}

</style>

<div class=’div1’>

1. Internal CSS Style sheets, insert the style rules using the @import statement.
2. External CSS Style sheets. Insert the style rules in a separate file using the <link rel=””> element to indicate where the css file is.

<link rel="stylesheet" type="text/css" href="css/test.css"/>

You can also use the @import statement inside <style> tags. This isn’t as common because of speed : <http://www.stevesouders.com/blog/2009/04/09/dont-use-import/>

CSS style rules can be applied to elements with an id attribute or class attribute. An id designation is more specific than a class assignment. Id’s and class attributes don’t have to be unique. We can reuse the same id and class values.

<p style="color:red">This is a test</p>

<div id="div1" style="border-width:thin;border-style:solid;height:100px;width:100px">this is div 1</div>

<div class="div2">this is div2</div>

<div id="div1">this is another div1</div>

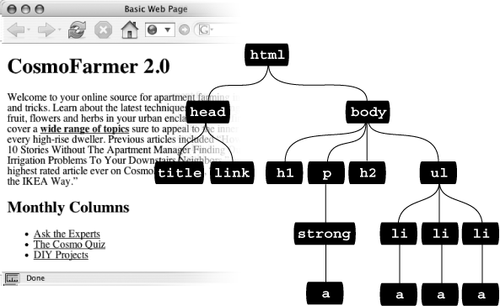
<div class="div2">this is another div2</div>

An \* or asterick rule is the universal rule where the style rule is applied to all HTML elements.

* {color:green}

**Descendant Selector Rules:**

In most cases using ids and classes to distinguish HTML elements is not enough. We can specify a rule for a particular element by specifying it’s location using parents of the element.



From: CSS: The Missing Manual

In the diagram above we can specify the <a> elements with the following patterns:

* ul li a
* body li a
* html body a
* html li a
* etc….

You can select any combination of parent elements as long as the selection is not ambiguous to the CSS interpreter.

**Demo Lab #4 Descendant Selectors: make the links in the href appear in the color of your choice for the html below:**

<!-- testing descendant selectors -->

<h1>This is h1</h1>

<p>this is the p in h1<a href="/plink.html">plink</a></p>

<h2>this is h2</h2>

<ul>

<li id="firstline" class="lines"><a href="/first.html">first item</a></li>

<li id="secondline" class="lines"><a href="/second.html">second item</a></li>

<li id="thirdline" class="lines"><a href="/third.html">third item</a></li>

</ul>

Soln:

html ul li a{

color:yellow;

}

**//more selector practice css lines css from zen garden**

**Javascript:** The browser stores HTML in a tree model similar to the black node tree in the figure above. This DOM or document object model resides in memory and is created after the HTML page has loaded into the browser. When the DOM is initialized by the browser and it exists in memory, we can modify the properties of objects in the DOM using JavaScript.

Javascript code is entered inside <script> tags. A simple example for Javascript is to modify the contents of a Header or paragraph using ids, element names or class names to identify HTML tag elements as we did using CSS rules. The result is the same as using CSS rules but the implementation is different.

**Demo Lab #5: Use JS to change the i**

**Link Styling:**

* **a:link** selects any link that your guest hasn't visited yet, while the mouse isn't hovering over or clicking it. This style is your regular, unused web link.
* **a:visited** is a link that your visitor has clicked before, according to the web browser's history. You can style this type of link differently than a regular link to tell your visitor, "Hey, you've been there already!"
* **a:hover** lets you change the look of a link as your visitor passes the mouse over it. The rollover effects you can create aren't just for fun—they can provide useful visual feedback for buttons on a navigation bar.

You can also use the *:hover* pseudo-class on elements other than links. For example, you can use it to highlight text in a <p> or <div> when your guests mouse over it. In that case, instead of using *a:hover* (which is for links) to add a hover effect, you can create a style named *p:hover* to create a specific effect when someone mouses over any paragraph. If you just want to style tags with a specific class of *highlight* applied to them, then create a style named *.highlight:hover*.

***Note:***

In Internet Explorer 6 and earlier, *:hover* works only on links. For a JavaScript workaround, see the box on [**Section 3.6.3.3**](javascript:moveTo('first-child');). (If IE 7 isn't in standards mode ([**Section 1.3**](http://my.safaribooksonline.com/9780596806736/the_importance_of_the_doctype#the_importance_of_the_doctype))—that is, the page is missing the proper doctype—it also won't obey *:hover* on anything but links.)

* **a:active** lets you determine how a link looks *as* your visitor clicks. In other words, it covers the brief nanosecond when someone's pressing the mouse button, before releasing it.

Add advanced selectors and the selector tutorial . very good. Do this and redo…

<http://gallery.theopalgroup.com/selectoracle/>

**HTML/CSS Multi column Layout with Header and Footer:**

This is the most common form of layout used for company webpages.

This is comprised of the following elements:

**Header:**

Consists of a logo floated on left and navbar on right.

Navigation bar using horizontal list, consists of 4 items

* Outer nav div container
* <ul> element with navli style rules
* <li> list element with list style rules for font style and font size
* 2 sets of rules for removing the underline in the href link and producing the visual effect when the mouse hovers over the element.

*#nav*{

float:*right*;

padding-top:*25px*;

margin:*0px*;

padding-right:*40px*;

}

*#nav* **li** {

display:*inline*;

padding-left:*15px*;

}

*#nav* **a**{

color:*blue*;

font-size:*10px*;

padding:*5px*;

}

*#nav* **a***:hover*{

color:*gray*;

text-shadow: *0 1px #334*;

background:*#373a41*;

border: *ridge 1px #262930*;

-moz-border-radius: *3px*;

-khtml-border-radius: *3px*;

-webkit-border-radius: *3px*;

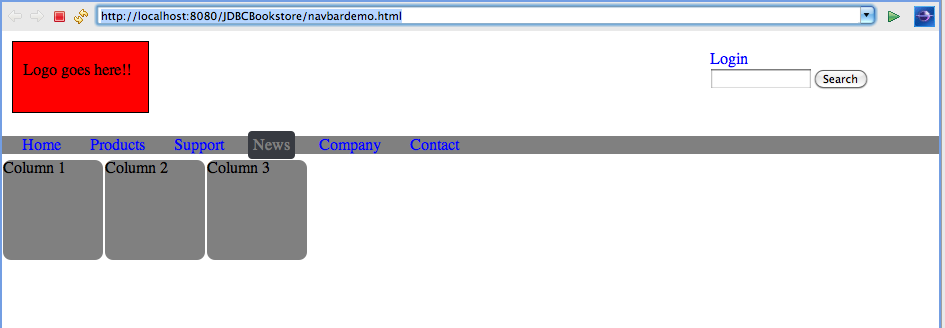
-border-radius: *3px*;

}

**Body:** The body div is the div containing the 3 columns.

* Use Width to separate the columns. 33% for 3 columns, 50% for 2 columns
* Add a header separator between the header and body. Make this the same color as the body background
* Add a color to the columns

Here are some examples of a 3 column layouts using width:



* HTML for columns:

<div id=*"body"*>

<div id=*"column"* >

Column 1

</div>

<div id=*"column"* >

Column 2

</div>

<div id=*"column"* >

Column 3

</div>

</div>

* CSS rules for column and body

/\* start column/body \*/

*#body*{

}

/\* want rounded corners \*/

/\* float all the columns to the left

/\* hard code a height and width because we have no content to put in there \*/

*#column* {

float:*left*;

height:*100px*;

width:*100px*;

background-color:*gray*;

margin:*1px*;

-moz-border-radius: *8px*;

-khtml-border-radius: *8px*;

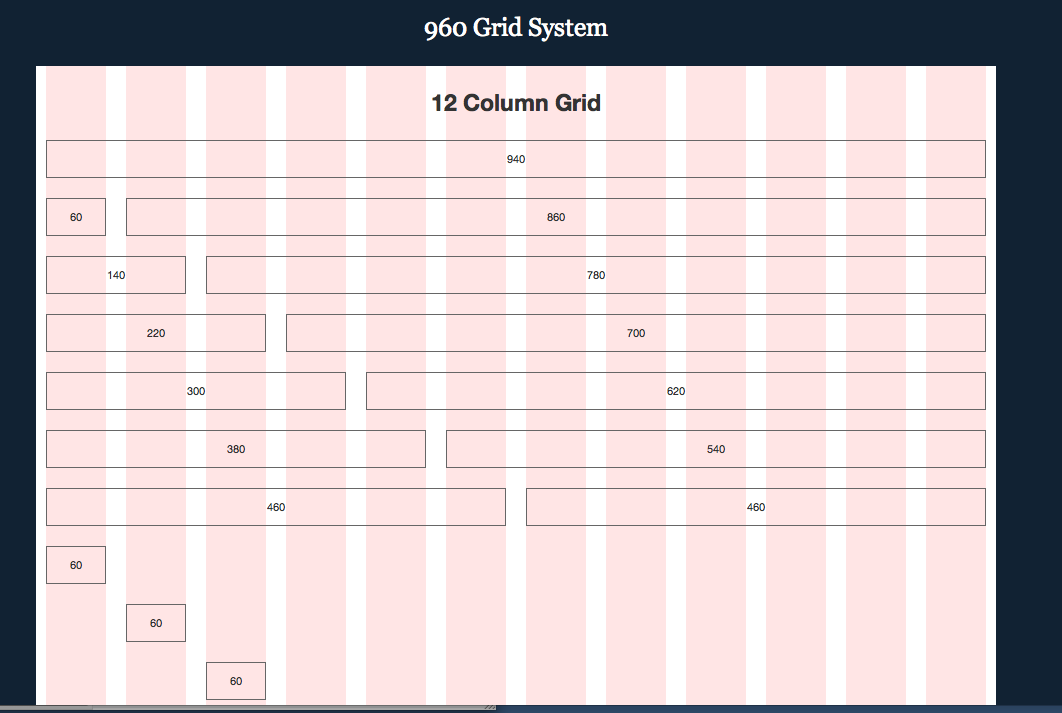
-webkit-border-radius: *8px*;

-border-radius: *8px*;

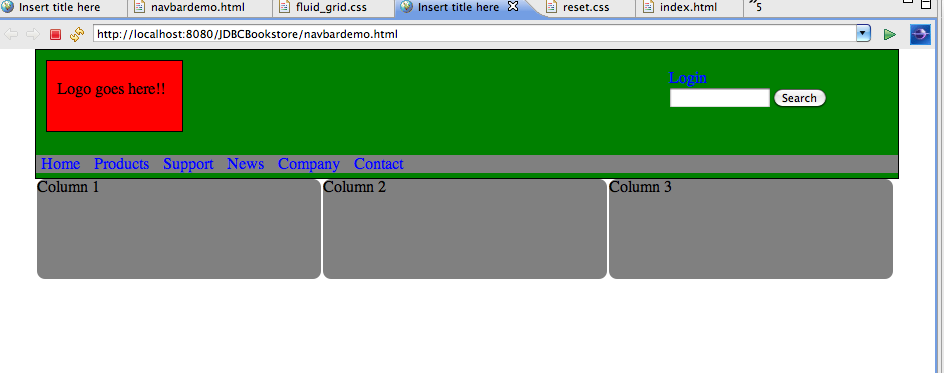
}

Another way to define columns is to use a predefined template, like the one from: <http://960.gs/demo.html>

The 960 predefined column style sheets work like traditional graphics design programs where the canvas is divided into grids. You can choose the size of the grid and place where you need them. Each black rectangle shows the relative size of the grids with a numeric pixel size which you can use to find the name of the style rule.



Using the container\_12 as a wrapper for the internal divs we can define a scaled inner region of 940 pixels. The container\_12 has a green background:

****

The advantage of this approach is no style rules have to be added when using the standard templates. The 3 columns are generated using width:33%.

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

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"*>

<title>Insert title here</title>

</head>

<link href=*"css/reset.css"* type=*"text/css"* rel=*"stylesheet"* >

<link href=*"css/fluid\_grid.css"* type=*"text/css"* rel=*"stylesheet"* >

<style type=*"text/css"*>

/\* bad programming style to add height. The contained elements should take care of this\*/

/\* begin header \*/

*#header*{

height:*100px*;

}

*#logoheader*{

}

*#logoheadernav*{

float:*right*;

padding-right:*50px*;

}

*#logoheadernav* **li**{

display:*inline*;

padding-left:*5px*;

}

*#logoheadernav* **a**{

color:*blue*;

font-size:*1em*; /\* standard size, not too small, not too large \*/

padding:*5px*; /\* separaton between teh horizontal list elements \*/

text-decoration: *none*; /\*remove underline for hrefs \*/

}

*#logoheadernav* **a***:hover*{

color:*gray*;

text-shadow: *0 1px #334*;

background:*#373a41*;

border: *ridge 1px #262930*;

-moz-border-radius: *3px*;

-khtml-border-radius: *3px*;

-webkit-border-radius: *3px*;

-border-radius: *3px*;

}

*#logo*{

float:*left*;

padding-left:*10px*;

padding-top:*20px*;

margin:*10px*;

background-color:*red*;

border-style:*solid*;

border-width:*1px*;

height:*50px*;

width:*125px*;

}

/\* design notes: unless you have href links, use divs instead of horizontal list \*/

/\* search boxes don't work well in a horizontal list because you can't nest html elements\*/

*#login*{

padding-top:*20px*;

padding-right:*20px*;

}

*#search*{

margin-left:*4px*; /\*vertically align search box with login \*/

}

/\* end header \*/

/\* begin navbar \*/

*#nav*{

margin-top:*5px*;

margin-bottom:*5px*;

margin-right:*0px*;

margin-left:*0px*;

padding-top:*1px*;

padding-bottom:*1px*;

background-color:*gray*;

}

*#nav* **li** {

display:*inline*; /\*horizontal list \*/

/\*padding-left:15px;\*/

}

*#nav* **a**{

color:*blue*;

font-size:*1em*; /\* standard size, not too small, not too large \*/

padding:*5px*; /\* separaton between teh horizontal list elements \*/

text-decoration: *none*; /\*remove underline for hrefs \*/

}

*#nav* **a***:hover*{

color:*gray*;

background:*#373a41*;

border:*hidden 1px*;

-moz-border-radius: *4px*;

-khtml-border-radius: *4px*;

-webkit-border-radius: *4px*;

-border-radius: *4px*;

}

/\* end navbar \*/

/\* start column/body \*/

*#body*{

}

/\* want rounded corners \*/

/\* float all the columns to the left

/\* hard code a height and width because we have no content to put in there \*/

*#column* {

float:*left*;

width:*33%*;

height:*100px*;

background-color:*gray*;

margin:*1px*;

-moz-border-radius: *8px*;

-khtml-border-radius: *8px*;

-webkit-border-radius: *8px*;

-border-radius: *8px*;

}

</style>

<body>

<div id=*"pageWrapper"* class=*"container\_12"*>

<div id=*"header"*>

<div id=*"logoheader"*>

<div id=*"logo"*>

Logo goes here!!

</div>

<div id=*"logoheadernav"*>

<div id=*"login"*><a href=*"#"*>Login</a>

<div id=*"search"*><form><input type=*"text"* size=*"15"* name=*"searchquery"*><input type=*"submit"* value=*"Search"*></form></div>

</div>

</div>

</div>

</div>

<div id=*"nav"*>

<ul>

<li ><a href=*"#"*> Home</a> </li>

<li ><a href=*"#"*> Products</a></li>

<li > <a href=*"#"*>Support</a></li>

<li ><a href=*"#"*> News</a></li>

<li > <a href=*"#"*>Company</a></li>

<li ><a href=*"#"*> Contact </a></li>

</ul>

</div>

<div id=*"body"* >

<div id=*"column"* >

Column 1

</div>

<div id=*"column"* >

Column 2

</div>

<div id=*"column"* >

Column 3

</div>

</div>

</div>

</body>

</html>

**Horizontal List as NavBar:**

* There are several layers of style rules required to make a horizontal nav bar

CSS Rules for the Outer div, <nav> containing the list

CSS Rules for the <ul> list element

CSS Rules for the <li> elements inside list

CSS rules for the anchor element, <a>

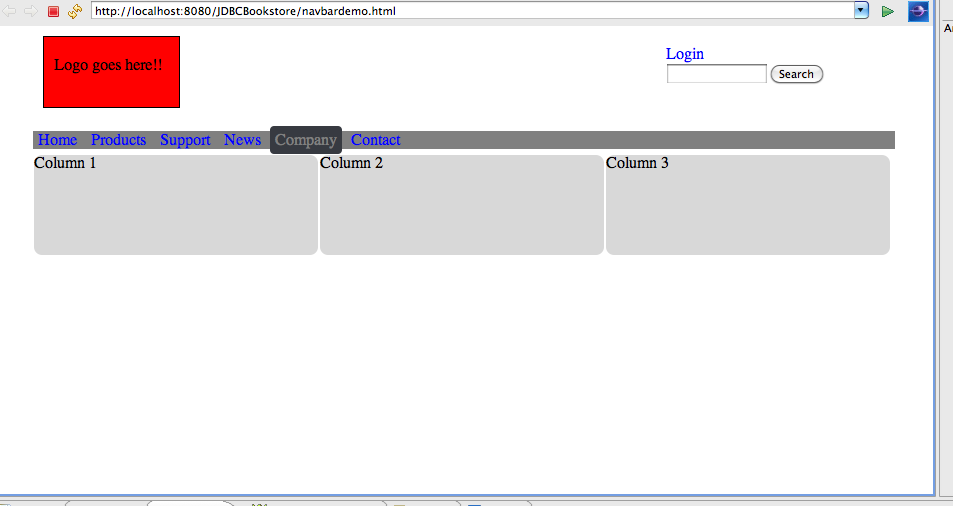
CSS rules for the hover acton <a:hover>

* Making list horizontal:

display:inline

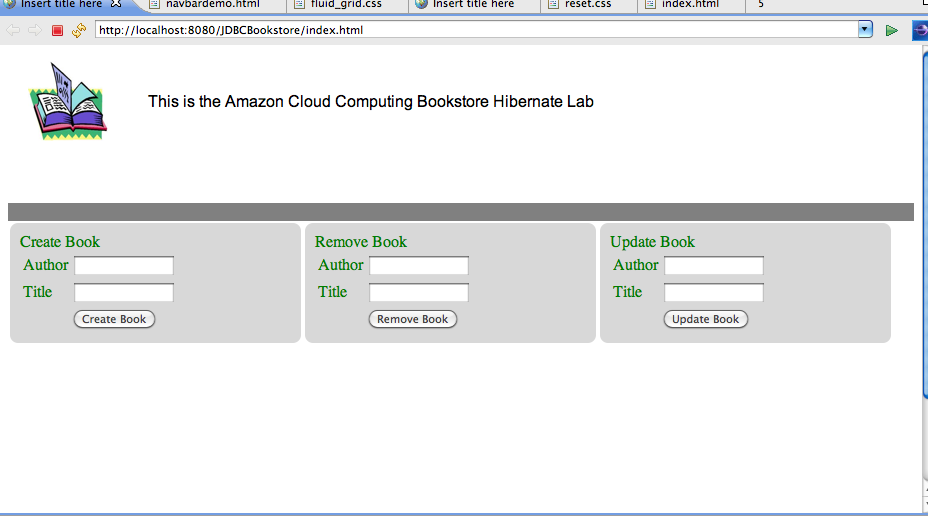
* Adjusting separation between the list elements, here this is Home, Products, Support, News, Company ad Contact.

padding: 5px;

****

**Color Theming:** Keep to one theme, like gray and do lighter and darker to draw the user to more important areas. This is the same webpage with extraneous information removed, keep the same design with a filler for the nav bar and lighter shades of gray.

Older browsers use addition to determine dimensions of HTML elements using borders, padding and gutters. The CSS2.1 specification reverses this and uses subtraction to determine HTML element footprint.



**CSS Reset:** HTML elements have different default values in different browsers. For example a div contains a default margin so if you want to create a header using a div you get a funky space between the div and edge of the browser. Font sizes have different px values in different browsers leading to different looking webpages in different browsers. It is common to reset all the default values back to 0 and then add your own fonts and sizes to keep the presentation consistent across browsers.