CSS Positioning

To use CSS for layout effectively, it helps to know how it's used to position page content. This article gives an overview of the methods and rules that govern visual rendering in the CSS2 specification. It also points out some things to watch out for.

Although the specification applies to any device for displaying web pages, this article focuses on how it works in browsers. Many details are left out for the sake of simplicity. For a definitive reference, see http://www.w3.org/TR/css3-box/.

It's important to remember that a given browser may not support a given feature or may even implement it incorrectly. Also, there is some leeway provided within the standards where individual browsers are free to deal with situations as they please. Where appropriate these inconsistencies are noted.

The Box Model

To understand positioning in CSS you must first understand the box model. For display purposes, every element in a document is considered to be a rectangular box which has a content area surrounded by padding, a border and margins. The illustration below shows these various parts.

```
content content content content content content content content
content content content content content content content content
content content content content content content content content
content content content content content content content content
```

Margin  Border  Padding  Content

Margins are always transparent. Borders come in various styles. Background settings for an element apply to the the area just inside the borders which includes both the padding and content areas. For purposes of illustration however, the padding area is shown in a slightly darker color.

When referring to boxes throughout this article, the term "margin edge," "border edge", etc. means the the outer boundary of the corresponding box area as shown above.

Margins, borders and padding are all optional but for purposes of calculating positions and sizes they are given a default width of zero if not specified. Different widths can be set for each individual side (top, right, bottom and left) if desired. Margins can even have negative values.
The width and height of each box is equal to the width and height of the outer margin box. Note that this is not necessarily the same as the width and height of the content area.

A box can contain any number of other boxes, creating a hierarchy of boxes that corresponds to the nesting of page elements. The browser window serves as the root element for this hierarchy.

Box Types

There are two basic types of boxes, block and inline. Block boxes are generated by elements such as P, DIV or TABLE. Inline boxes are generated by tags such as B, I or SPAN and actual content like text and images.

The box type may also be set using the display property. Setting a value of block on an inline element, for example, will cause it to be treated as a block element. Note that if you set the display to none, no box is created. The browser acts as if the element did not exist. Likewise, any nested elements are ignored as well, even if they specifically declare some other display value.

There are other types of boxes which apply to special elements like lists and tables. However, these are ultimately treated as block or inline boxes for positioning purposes. As such, these other box types not covered here.

Containing Blocks

Block boxes act as a containing block for any boxes within them. For example, in this code:

```html
    <div>
        This is the first sentence.
        <p>This is the second sentence.</p>
    </div>
```

the DIV element establishes a containing block for both the first string of text and the P element. The P element in turn creates a containing block for the second text string.

It’s interesting to note that while the text of the first sentence in the above example generates an inline box, there is considered to be block box surrounding it. These "anonymous" block boxes are used to simplify the positioning process. The result is that a block box will only contain either all inline boxes or all block boxes, even if some of those block boxes only act as a wrapper for an inline box.

This containing block is used in determining both the position of the boxes within it and in some cases, the dimensions of those boxes. For example, if an element has a style setting of width: 50%; its width will be set to half the width of its containing block.
For any element that is not absolutely positioned, the containing block is considered to be the content edge of its most recent, block-level ancestor. If none exists, the browser window serves as the containing block. Absolutely positioned elements are discussed separately.

DIV Based Layout with CSS - The DIV tag

According to the O'Reilly HTML Reference, “the DIV element gives structure and context to any block-level content in the document. Each DIV element becomes a generic block-level container for all content within the required start and end tags.” As we can see, the DIV tag is a powerful generic element well suited for being used as a container within our Web page. This turns it into a good candidate for creating sections or divisions (hence "DIV") of Web documents.

The concept is very intuitive. Instead of using tables as layout elements, we are going to take advantage of DIVS as excellent content containers to build our page layout. In conjunction with several CSS declarations, DIVS based layout are pretty easy to get.

Before we explain any further, two general categories used for Web page design must be clearly differentiated: “fixed” and “liquid” design. When referring to “fixed” design, we are talking about a Web page that has “fixed” dimensions. Widths (and optionally, heights) for each container element are assigned normally in pixels.

The final result of this approach is that Web pages are displayed with a determined “fixed” size, and they don’t “expand” to cover all the space in the computer monitor. Many “big” websites currently use this technique for achieving a consistent look and size across several user screen resolutions.

On the other hand, Web pages built over a “liquid” design principle will display as “elastic” documents, expanding as needed according to different monitor resolutions. Normally, dimensions for any container element are expressed in a percentage. “Liquid design,” when used properly, looks much more professional and takes one step further the definition that conceives a Web page as an “elastic” element, not a desktop document.

With all of these basic concepts out of the way, we will see different approaches for page layouts, depending on whether we follow a “fixed” design or a “liquid” design pattern.
Centering lines of text

The most common and (therefore) easiest type of centering is that of lines of text in a paragraph or in a heading. CSS has the property 'text-align' for that:

```css
p { text-align: center }
h2 { text-align: center }
```

renders each line in a P or in a H2 centered between its margins, like this:

The lines in this paragraph are all centered between the paragraph's margins, thanks to the value 'center' of the CSS property 'text-align'.

Centering a block or image

Sometimes it is not the text that needs to be centered, but the block as a whole. Or, phrased differently: we want the left and right margin to be equal. The way to do that is to set the margins to 'auto'. This is normally used with a block of fixed width, because if the block itself is flexible, it will simply take up all the available width. Here is an example:

```css
p.blocktext {
    margin-left: auto;
    margin-right: auto;
    width: 6em
}
...
<p class="blocktext">This rather...
```

This rather narrow block of text is centered. Note that the lines inside the block are not centered (they are left-aligned), unlike in the earlier example. This is also the way to center an image: make it into block of its own and apply the margin properties to it. For example:

```css
img.displayed {
    display: block;
    margin-left: auto;
    margin-right: auto
}
...
<img class="displayed" src="stone_face.jpg" alt="Stone Face">
```

The following image is centered: