CSS Primer

2017

Part 4 is a primer on CSS. It cannot cover all topics or all situations. It will give you a good start on learning about CSS styles and how they work. Consider this a springboard to continue learning. [Release Date 04.12.17]

Developer Guide [Part 4]

CONTENTS

LEARN MORE - DESIGN BETTER

Creating an online presence for your new business or redesigning an existing site has never been more exciting. Our templates allow you to achieve professional results that you will be proud to display to your visitors. The tutorials and information within this guide are offered in good faith.

Whether you are a new business or a well-established one looking to refresh your image, templates are an economical and convenient way to create a professional looking site. A well-designed template gives a great first impression that conveys trust, dependability, and professionalism.

CSS – LEARN THE BASICS

This tutorial section is designed to teach you the basics of CSS. It cannot cover all topics or all situations. It will give you a good start on learning about CSS styles and how they work. Consider this a springboard to continue learning.

WHY CSS?

Imagine that you are a web designer and have just completed a 40-page site for your client. He is thrilled with your work, but wants to make a few minor changes. Instead of using Arial as the default font, he would like for you to change it to Verdana. It's a simple request.

You look at your coding and see that there are font markup tags that specify the font as Arial. You have to change each and every <font> tag. At approximately 100 tags per page times the 40 pages, you have to change 4,000 items. Doing them one at a time is now a major headache and will take hours.

However, if you had defined your font using CSS (rather than using deprecated font tags), you would make the change on only one page and the changes would be made automatically on every page in the site. You've just cut the job down from hours to only a few seconds.

While covering valid CSS, remember that nothing is guaranteed to work in all browsers all of the time. Don't obsess over pixel perfection across the different browsers. If something doesn't work, look for other alternatives. Yes, you may need to rethink your design or the approach you are taking.

A BASIC PAGE

It all begins with the most basic page structure. It's just three tags, each which has an opening and closing tag.

HTML

```
<html>
 <head>
  <title>Your Site</title>
 </head>
 <body>
  All of your content goes here
 </body>
</html>
```

The whole page is wrapped in the <html> ... </html> tag pair. Nested inside is the <head> ... </head> tags. Information inside of the head tags does not display on the page. The <body> ... </body> pair encloses all of the content of your web page such as text, images, different layout sections, etc.

The HTML tags provide the **structure**. The CSS defines the **presentation** of how that structure will look. Simple concept!

RELAX AND HAVE FUN

The terminology is probably foreign to you, but the concepts are quite logical and simple. Take it one step at a time and don't try to rush through every topic in one sitting. There's a lot of information to absorb and put to use.

CSS RULES

PART 1: LEARNING THE RULES

It's all in the details, and rules are important. CSS is the acronym for Cascading Style Sheets. CSS is an extension to basic HTML that allows you to "style" how the content within your web pages will look. With CSS you can set up rules (or definitions) to tell specific HTML tags how to display content, or you can create rules and apply them to tags when you need them.

Each browser has a set of parameters for how it reads paragraphs, headings, tables, etc., and then renders it on your screen. Each of the tags can be defined so that you can tell the browser how to display it.

WITH THIS IN MIND, YOU CAN THINK OF YOUR CSS FILE AS A SORT OF DICTIONARY WITH A CLEAR DEFINITION OF HOW EACH ITEM IS DISPLAYED ON THE SCREEN.

HTML SELECTORS

All HTML tags have default properties. With CSS you can define any HTML tag to change the defaults.

CSS

```
p {
 line-height: 14px;
}
```

You can define a style for multiple tags at one time by separating each html tag name with a comma.

CSS

```
h1, h2, h3 {
 color: #c00000;
}
```

CLASS SELECTORS

A class is a rule that can be applied to an HTML tag. You can give a class any name you want with the exception of a few reserved words.

CSS

```
.myclass {
 font: bold 14px Verdana;
}
```

Classes have an advantage in that you can create a number of different classes and apply them to different paragraphs, headings, divs, etc.

ID SELECTORS

ID rules are similar to a class, but they can only be applied once on a page to a particular HTML tag. IDs are commonly used in CSS positioned layouts since items like a header or footer are only going to appear once on a page. An ID rule looks like the following:

CSS

```
#myid {
 background-color: #ffffff;
 height: 40px;
 padding: 10px 20px;
}
```

PART 2: PARTS OF A CSS RULE

Each of the rule examples above (html selector, class, and ID), consist of three distinct parts. The first part of the rule is the type of selector whether it be an html selector, a class, or an ID. The selector is followed by a space and then a beginning curly brace. Within the curly brace will be the property. It is followed by a colon.

After the property comes the value of that property. Note that you can have more than one property and value in a rule. The value is followed by a semicolon. To end the rule, place the closing curly brace. You cannot have a property without a value nor a value without a property.

Below is the syntax for a CSS rule:

CSS

```
selector {
 property: value;
 property: value;
}
```

PART 3: WHERE YOU CAN PLACE A CSS RULE

You can also choose where to place your CSS rules. You can place them in an external style sheet (preferred), as an embedded style, or as an inline style. Note that classes and IDs cannot be applied or placed in an inline style.

INLINE RULES

An inline rule is a style that is attached to one specific tag. There may be times when you need a rule that you don't plan to use anywhere else. The syntax for an inline style rule is different from that of embedded or external styles:

HTML & CSS

```
<h1 style="color: #c00000"> ... </h1>
```

EMBEDDED RULES

An embedded rule is a style that is attached to one specific web page. Embedded styles are placed within the <head> section of the HTML page and the rules affect the entire page. An embedded rule would look like the following:

HTML & CSS

```
<style type="text/CSS">
.style1 {
 text-align: center;
 font: "Times New Roman";
 color: #800000;
}
</style>
```

EXTERNAL CSS FILE

An external style sheet is the file containing CSS information that can be linked with any number of pages. This is a very convenient way of formatting the entire site as well as restyling it by editing just one file.

External style sheets are text documents with a .CSS file extension. The syntax for linking to an external style sheet is:

HTML & CSS

```
<link ref="stylesheet" type="text/CSS" href="style.CSS" />
```

AS A RECAP

- We've learned that there are three types of CSS rules: HTML, Class, and ID
- We've learned there are three parts to a CSS rule: selector {property: value;}
- And we've learned there are three places to apply a CSS rule: Inline, Embedded, and External

Congratulations! Take a break and, when you are ready, move on to the next lesson.

CSS BACKGROUNDS

There are five background properties that can be used to set the background of an element. The background properties are listed in order as follows:

- background-color
- background-image
- background-repeat
- background-position
- background-attachment

BACKGROUND-COLOR

This specifies the background color of an area such as a page background, a table cell, a div block, a heading, etc. You can use a hex value (#ffffff), a named color (blue), an RGB value written as rgb(00,00,00), or use a value of transparent. You can apply a background color to an entire page by redefining the body tag or to other elements where you want a background color to appear.

The example below sets a white background color within the body tag using a hex color value:

CSS

```css
body {
 background-color:#ffffff;
}
```

BACKGROUND-IMAGE

This setting allows you to set an image as a background. There are only two settings for this attribute: none and url.

Setting the value to none is normally used when you want to remove an image that was previously set.

Using URL requires that you specify the file location of your image as shown below:

CSS

```css
body {
 background-image: url('imagename.jpg');
}
```

BACKGROUND-REPEAT

The default setting for the background-repeat value creates a tiled effect so that your image repeats both horizontally and vertically to fill the space in which it is placed.

The settings include repeat, no-repeat, repeat-x, and repeat-y. No-repeat causes the image to appear only once at the position you set. Repeat -x makes the image tile horizontally while repeat-y makes it tile vertically

BACKGROUND-POSITION

This property sets the background position of the image relative to the element. Position values are as follows: top left, top center, top right, center left, center center, center right, bottom left, bottom center, and bottom right.

A style rule using the background-position property would look like this:

CSS

```css
body {
 background-image: url('imagename.jpg');
 background-position: top right;
}
```

BACKGROUND-ATTACHMENT

Values for this property are scroll and fixed. You can choose whether or not to have your background remain in a fixed position when the page scrolls.

- Fixed instructs the browser not to scroll the background with the rest of the elements.
- Scroll is the default setting and instructs the background to scroll along with the other page elements.

SHORTHAND PROPERTIES

Some properties, like the background property, are made up of several values. To reduce the size of your CSS file and also save some keystrokes as well as bandwidth, they can all be specified as one shorthand property.

For example, the background properties listed below...

CSS

```
body {
 background-color: #ffffff;
 background-image: url('images/image.jpg');
 background-repeat: repeat-x;
}
```

...can also be written as

CSS

```
body {
 background: #ffffff url('images/image.jpg') repeat-x;
}
```

A USEFUL THING TO KNOW

Shorthand properties exist for many items such as background, font, border, margin, and padding controls. It is important, however, that the values be listed in their specific order.

In the shorthand rule example, the background-color is first, followed by the background-image. The background-repeat is set to repeat-x so it will tile horizontally. There was no need to set a background-position since the default is top left. Because this background will scroll with the rest of the page elements, there was no need for a background-attachment value.

Any property that is omitted will use the default value.

FONTS AND THEIR PROPERTIES

No other style will have as much impact on your site visitors than your font controls. The appropriate fonts, colors, and sizes have a lot of impact on how user-friendly your site is to others. Keep in mind that a web page is not the same as a printed page in a magazine.

NOTE: HOW TO SPECIFY A FONT SIZE IS AN EVOLVING ISSUE. IN PRINT, YOU CAN CHOOSE AN EXACT SIZE IN POINTS AND IT WILL ALWAYS BE THE SAME SIZE AND WILL ALWAYS TAKE UP THE SAME AMOUNT OF SPACE. IN A WEB BROWSER, FONT SIZES CAN VARY DEPENDING ON THE INDIVIDUAL COMPUTER, THE OPERATING SYSTEM, THE BROWSER USED, AND WHETHER THE COMPUTER IS WINDOWS OR MAC. WE PREFER TO USE A BASE FONT SIZE OF 100%, AND THEN USE PIXELS (USUALLY FOR HEADINGS) AND EMS FOR MOST EVERYTHING ELSE, BUT THIS WILL VARY DEPENDING ON THE PARTICULAR TEMPLATE.

FONT ATTRIBUTES

There are five attributes that define fonts in a web document. They are listed in order as follows:

- font-style
- font-variant
- font-weight
- font-size
- font-family

FONT-STYLE

There are three values for the font-style property: normal, italic, and oblique. Normal refers to standard type. Italic and oblique will render the text as italicized.

In the shorthand property, you do not have to specify a font-style. The default is normal.

FONT-VARIANT

This property has two values: small-caps and normal. Small-caps displays the text in capital letters with the first letter being normal sized but the other letters smaller than normal. The small-caps setting should be reserved for headings because it may be difficult to read at smaller sizes. In the shorthand property you do not have to specify a font-variant. The default is normal.

FONT-WEIGHT

The font-weight property refers to letter thickness. You can choose a value of bold that makes the text darker, a value of bolder that makes the text even darker, and a value of lighter that makes the text less bold than the bold setting. You can also specify a value range from 100 (the lightest) to 900 (the darkest), in increments of 100.

FONT-SIZE

Font-size values can be an exact size such as pixels or they can be a relative size set as a percentage or 'em'. 1.2em is the same as 120%. Font size values can also use keywords such as xx-large, x-large, large, medium, small, x-small, and xx-small. Larger and smaller keywords are set relative to the size of the parent element.

FONT-FAMILY

The font-family value allows you to specify more than one font setting. If the first font in the list is not available on the reader's computer, the browser will select the next font listed. It is recommended that you use a generic font type such as serif or sans-serif as the last font. You separate font names with commas. Fonts that have spaces in their names, such as MS Comic Sans, must be enclosed in quotation marks.

FONT SHORTHAND PROPERTIES

The font properties also have shorthand properties. For example, the heading tag rule shown below...

CSS

```
h1 {
font-weight: bold;
font-size: 2.10em;
font-family: Georgia, "Times New Roman", serif;
}
```

...can also be written as

CSS

```
h1 {
font: bold 2.10em Georgia, "Times New Roman", serif;
}
```

TEXT CONTROLS

Text controls do not have shorthand properties, but there are a lot of properties you can use to control the look of your text.

TEXT ATTRIBUTES

- text-align
- letter-spacing
- text-indent
- line-height
- text-transform
- text-decoration

Finding all of the rules just a bit difficult to remember? If you are just beginning to learn about CSS, you might want to search for a CSS cheat sheet that has a lot of the selectors along with their properties and values on a handy sheet.

- Simple CSS Cheat Sheet 1
- CSS Version 2

TEXT-ALIGN

The text-align property sets the horizontal alignment of your text on the screen. Your value choices are left, right, center, or justify. Left orients the text to the left, right orients the text to the right, and center orients the text to the center of the container block. Justify aligns the text so the left and right edges are aligned. An example of the text-align rule looks like the following:

CSS

```
.myclass {
 text-align: left;
}
```

LETTER-SPACING

The letter-spacing property sets the amount of spacing between letters. It's a great way to add a bit of pizzazz to things like headings or small amounts of text. While a little bit of spacing can add a great look, remember that a little goes a long way and can sometimes make text more difficult to read. The default for this property is "none", so if you want to add a bit of spacing you can specify the length in ems or pixels. An example of a letter-spacing rule would look like this:

CSS

```
h1 {letter-spacing: 3px;}
```

TEXT-INDENT

The text-indent property will indent the first line of a paragraph. The value can be either a fixed length value such as 20px or it can be set as a percentage such as 10%. A text-indent rule would look something like this:

CSS

```
.indent {text-indent: 20px;}
```

Caution: The text-indent value can also accept a negative length, however, there must be enough padding to accommodate it. If your containing block has a padding of 10px, you cannot set a negative value to more than 10px.

LINE-HEIGHT

The line-height property controls the amount of spacing between each line of text. The value can be set as a number or as a percentage, or the default of none. For example, a value of 20px would set twenty pixels of space between each line. A percentage value would calculate the height relative to the font size. An example of a line-height rule would look like this:

CSS

```
.tall {line-height: 20px;}
```

The line-height property can also be used within a font shorthand property. It needs to be placed after the font size value and is separated by a forward slash with no space added. An example would look like this:

CSS

```
p {font: 12px/20px Verdana, Arial, sans-serif;}
```

TEXT-TRANSFORM

The text-transform property changes text from lowercase to uppercase. It can also change text from uppercase to lowercase. Text-transform values are capitalize, uppercase, lowercase, and none. An example of a text-transform rule would look like this:

CSS

```
h1 {text-transform: uppercase;}
```

TEXT-DECORATION

The text-decoration property has these values: underline, overline, line-through, and none. An example would look like:

CSS
```
h1 {text-decoration: underline;}
```

The **underline** value adds an underline to a word or group of words. The **overline** value is most commonly used along with the underline value and will add a line above your words.

Hyperlinks are underlined by default, so use caution if you wish to underline words that are not hyperlinks. This can be confusing to your visitors who may expect the results to be an actual link. Because links default to having the underline, in order to not have an underline the value of none must be specified. An example is shown below:

CSS
```
a {text-decoration: underline;}
a:hover {text-decoration: none;}
```

The **line-through** value will draw a line through the word or groups of words. You have probably seen this on catalog pages to show the difference between a regular price and a sales price as in the example below:

CSS
```
.price {
 text-decoration: line-through;
 color: #ff0000;
}
```

Results in this: ~~$12.95~~ $10.95

In our line-through example, the html coding looks like this:

HTML
```
<p>Results in this:
<span class="price">$12.95</span> $10.95</p>
```

THE HANDY <SPAN> TAG

Sometimes you may want to apply a style to specific elements, like words, within a tag. This is where you can use the <span> ... </span> tags. The span tag has no inherited properties and serves as a blank slate for creating your own inline effects.

For example, let's say you create a highlight class as follows:

CSS

```
.highlight {background-color: #ffff80;}
```

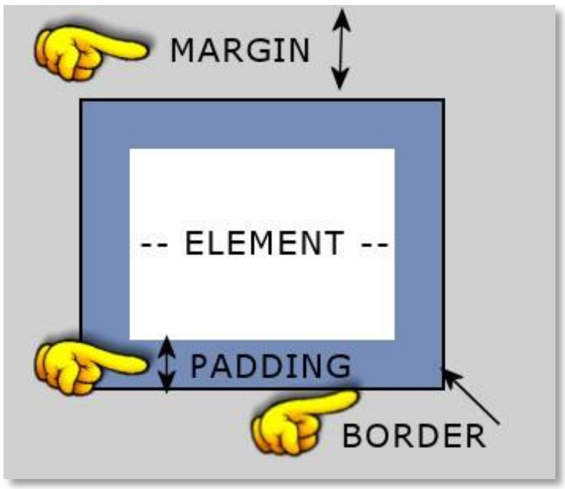
The class is then applied to a few words within a paragraph as follows:

HTML

```
<p>A paragraph to which <span class="highlight">I apply a
highlight</span> to words I want to emphasize.</p>
```

This is the result:
A paragraph to which I apply a highlight to words I want to emphasize.

THE BOX MODEL CONCEPT



Before we can move on to discuss margin and padding values, it is important to first understand the Box Model and how it affects CSS.

WHAT IT IS

This is an example of a box with different color areas showing the various regions that can be formatted.

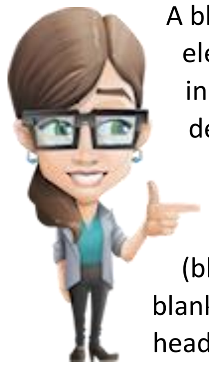
Each block level tag has four sides that can be specified in order going clockwise beginning at the top. Every box element can have a width and height defined, as well as a margin, border, and padding to all four sides.

Width and height properties accept a length value, a percentage value, or auto (which leaves the dimensions up to the browser.) An example is as follows:

```
CSS
.myarea {
 width: 300px;
 height: 100px;
}
```

COMMON BLOCK-LEVEL PAIRS

```
<aside> ... </aside>
<blockquote> ...</blockquote>
<body> ... </body>
<button> ... </button>
<div> ... </div>
<dl> ... </dl>
<fieldset> ... </fieldset>
<figure> ... </figure>
<footer> ... </footer>
<form> ... </form>
<h1> ... </h1>
<h2> ... </h2>
<h3> ... </h3>
<h4> ... </h4>
<h5> ... </h5>
<h6> ... </h6>
<head> ... </head>
<html> ... </html>
<iframe> ... </iframe>
<img> <-- does not have a closing tag.
<layer> ... </layer>
<legend> ... </legend>
<ol> ... </ol>
<p> ... </p>
<select> ... </select>
<table> ... </table>
<ul> ... </ul>
```



A block-level element may contain other block-level elements, or it may contain inline elements. But inline elements cannot contain block-level elements, only data or other inline elements. So a block-level element is "bigger" or "higher" in the hierarchy. It defines a larger section of the page than an inline element does.

Block-level elements generally begin on new lines. In practice, most browsers insert a blank line between any two block-level elements. So if you have a paragraph (block-level), and you end that paragraph and begin another paragraph, there will be blank line between them. Same thing between a header and a paragraph, between two headers, between a paragraph and a list, between a list and a table, etc. This is a useful clue as to what a block-level element is.

Inline elements do not start on a new line. If you type out a paragraph, and decide to make one of the words italic, does the browser jump down a line for the italic word? No, it doesn't and this is a clue as well. The italic tag is an inline tag. So is bold, strong, em, a href, etc. None of these tags causes the browser to start a new line, so these are all inline elements.

DO THE MATH

If you are wondering why this discussion on block-level elements is important, you will understand after reading through this example:

Assume that you have purchased a template and the page layout is contained within a <div> called "wrapper" or "container". This div is set to a width of 1200 pixels. A div is a block-level element.

Inside of this wrapper are two other divs, one for the sidebar and one for the content. You decide you need for the sidebar to be just a bit narrower and the content to be a bit wider. How do you make this change?

Here's a peek inside the CSS file (don't worry if you don't understand it all at this point):

CSS

```
.container {
 padding: 0;
 margin: 0 auto;
 width: 1200px;
}
```

CSS

```
.sidebar {
 float: left;
 width: 28%;
 padding: 10px 0;
 margin: 0 2% 0 0;}
```

CSS

```
.content {
 float: right;
 width: 68%;
 padding: 10px 0;
 margin: 0 0 0 2%;
}
```

WE WANT THE SIDEBAR AND CONTENT AREAS TO TAKE UP 100% OF THE CONTAINER SIZE.

The sidebar has a width of 28% and the content has a width of 68%. For the math wizards, you will see that this does not add up to 100%! It's only 96%.

Now look at the padding values and you will see there are some percentage values. Remember that values are listed clockwise starting at the top. All you are concerned about are values affecting the width.

The sidebar has a 2% value added to the right side. Add this 2% value to the sidebar's width and you have 30%. The content a 2% value added to the right side, so its total width 70%.

Add the two together and you come up with 100%. It's a perfect fit.

To change both the sidebar and content div width, make adjustments so that all width values still equal 100%.

CHANGING THE BOX MODEL

Sometimes there may be a circumstance where it is difficult to determine the correct percentages for margins and paddings. You can overwrite the way the box model works with a bit of CSS. To maintain a specific width with margins or paddings, set width and add:

CSS
```
box-sizing: border-box;
```

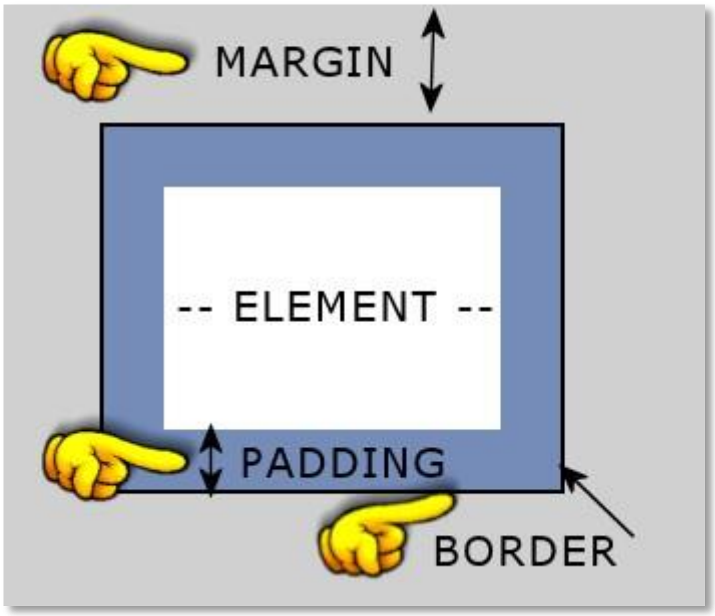
This works very well, but only has one disadvantage if you are using Expression Web or Dreamweaver. The box-sizing is only rendered in the browser and not in Design View of your web editor. It causes the content blocks to distort with some dropping below rather than sitting beside another block. For more advanced users it isn't a problem. For the novice or casual user who relies on a more accurate Design View, it is confusing.

Your mileage may vary so be sure to preview often to make sure you are getting the desired results. But it's a handy CSS tip to know.

MARGINS, PADDING AND BORDERS

All three of these properties follow the box model as they affect space surrounding an element.

MARGIN CONTROLS



Margins define the space surrounding an element. In the example above, the margin is the gray area. You can set each margin individually as shown in the example below:

CSS
```
.myclass {
 margin-top: 20px;
 margin-right: 30px;
 margin-bottom: 20px;
 margin-left: 10px;
}
```

You can also use the margin shorthand property. In the example below, a 20 pixel margin will be added to all four sides:

CSS
```
.myclass {margin: 20px;}
```

If you use two values, the first value applies to the top and bottom and the second value applies to the right and left sides. An example would look like the following:

CSS
```
.myclass {margin: 20px 10px;}
```

If you wish to use all four values, you can apply each value separately as follows:

CSS

```
.myclass {margin: 20px 10px 20px 30px;}
```

Note that there are no commas, only a single space separates each value. You can also use percentages in place of pixels.

BORDER CONTROLS

Border controls enable you to add a visible border around an area. You can specify the border's style, width, color, and you can set each individual side of an element.

Using the border shorthand properties, the attributes are listed as follows:

- border-width
- border-color
- border-style

A shorthand property rule might look something like this, which applies a 1 pixel solid dark gray border:

CSS

```
.myclass {border: 1px #666666 solid;}
```

Border-width: The border-width property accepts a length value such as 1px or a relative-size keyword such as thin, medium, or thick. A border-width of 0 will prevent a border from appearing.

Border-style: The border-style attribute refers to the type of border such as solid or dashed. note that not all values are recognized in all browsers. If a browser does not recognize a value, it will usually either be ignored or will display as a solid border. The different values are: none, dotted, dashed, solid, double, groove, ridge, inset, outset.

Border-color: The border-color property accepts a value in either hex or RGB. It will also accept a color name, but can have unexpected results. We prefer using a hex value.

Each border has four separate sides and you can define each side individually with the following properties:

- border-top
- border-right
- border-bottom
- border-left

It is possible to have a block with each side having a different width, color and style. (Although why would you ever want to?)

CSS

```
.mycrazyblock {
 border-top: 5px blue dashed;
 border-right: 3px #606060 double;
 border-bottom: 1px #000000 solid;
 border-left: 2px #ff0000 dotted;
}
```

PADDING CONTROLS

Padding sets the amount of space inside of a block element. In the box model image above, the padding is shown in blue. You can set each side's padding individually as shown in the example below:

CSS

```
.myclass {
 padding-top: 5px;
 padding-right: 20px;
 padding-bottom: 10px;
 padding-left: 15px;
}
```

Just as with margin properties detailed above, the padding properties can also use shorthand properties setting all sides the same, in pairs, or differently.

FLOAT

The float property is most commonly applied to images and divs and accepts the value of right, left, or none. If you have a need to display your images in various ways, you can define a class and apply the class to a single image. In this site, you will see our small icon images are floated to either the left of the right. Here are some examples:

This example has the image floated to the left without an image border. It sits to the left side of the containing block and the text will flow around the image. The coding is as follows:

CSS

```
.img-left {margin: 0 10px 0 0; float: left;}
```

This is the same image, but it is floated to the right. Again the image will sit in its position and the text will flow around it. The coding is as follows:

CSS

```
.img-right {margin: 0 0 0 10px; float: right;}
```

CSS LIST PROPERTIES

Need a bulleted list and want to control what the bullet looks like or a numbered list that shows roman numerals rather than plain numeric ones? Control it all through CSS.

There are three different types of lists: **Unordered lists** which typically display some type of bullet, **Ordered lists** which follow an alpha or numeric sequence, and **Definition lists** that consist of a term and its definition.

UNORDERED LISTS

Unordered lists begin with opening <ul> tag with the individual list items <li> .... </li> contained within and ends with the closing </ul> tag. An example of the coding for an unordered list looks like this:

HTML

```
<ul>
    <li>This is the first item</li>
    <li>This is the second item</li>
    <li>This is the third item</li>
</ul>
```

The unordered list properties allow you to control the appearance of your lists by choosing a "marker" (such as a disc, circle, or square) or by setting a small image. You can also then determine where the marker is placed.

There are three properties used to control the appearance of an unordered list:

- list-style-image
- list-style-type
- list-style-position

It is important to note that both unordered and ordered lists use the same <li> tag within it. When you define a style for lists, you must define the both the type of list as well as the list item. If you don't, then all list items would look the same whether they are in an ordered or unordered list.

List-style-image: The list-style-image property lets you use a small graphic image in place of a bullet in a list. A typical rule would look like this:

CSS

```
ul li {list-style-image: url('images/bullet.gif');
```

Because list items can have up to three levels of bullets, you can define a different bullet for each level. For second and third level bullets, your style coding would look similar to this:

CSS

```
ul li {list-style-image: url('bullet.gif');
ul li li {list-style-image: url('bullet2.gif');}
ul li li li {list-style-image: url('bullet3.gif');}
```

List-style-type: The list-style-type property controls the type of bullet used for list items. The possible values are disk, circle, square, and none. You can also define a different type of bullet for all three levels as follows:

CSS

```
ul li {list-style-type: disc;}
ul li li {list-style-type: circle;}
ul li li li {list-style-type: square;}
```

List-style-position: The list-style-position property controls the alignment of text with respect to the list item bullet. The two possible values are inside or outside. Inside aligns subsequent lines of text with the bullet. An outside value causes subsequent lines of wrapped text to line up with the first letter in the first line of text

ORDERED LISTS

If you have ever had to do an outline for a school paper, you are already familiar with ordered lists. These follow a numerical and/or alphabetical sequence such as decimal (1, 2, 3), lower roman (i, ii, iii), upper roman (I, II, III), and upper alpha (A, B, C).

Ordered lists also use the list-style-type:

CSS

```
ol li {list-style-type: upper-roman;}
```

As with unordered lists, ordered lists can also be defined for up to three levels:

CSS

```
ol li {list-style-type: upper-roman;}
ol li li {list-style-type: upper-alpha;}
ol li li li {list-style-type: decimal;}
```

DEFINITION LISTS

A definition list consists of three HTML parts: the definition list container <dl>, the definition term <dt>, and the definition description <dd>. All three parts can be styled, but only the definition description <dd> can contain block level items like paragraph <p> tags.

If you are unfamiliar with definition lists, consider the advantages of styling your FAQ page. Typically you have a question (the definition term) followed by an answer (the definition description). The following is an example:

CSS

```
dt {
 font-weight: bold;
 letter-spacing: 2px;
 color: #bf2c22;
 padding: 15px 0;
}
dd {
 color: #5454d4;
}
```

If you are curious about the possibilities of Definition Lists, I recommend you visit the sites below (will open in a new window).

http://www.maxdesign.com.au/presentation/definition/
http://www.CSSplay.co.uk/menus/

THE DOCTYPE DECLARATION

Every HTML document should have a document type declaration. The "doctype" begins the web page and tells the browser how to interpret the coding on the page. It also tells a validator which version of HTML to use in checking the document's syntax.

When you use a CSS file, omitting the document type will throw the browser into "quirks" mode and can give you very unpredictable results. Since CSS is designed to tell the browser how to display the content, why make the browser guess? Add a doctype!

HTML5

All of our templates are coded using HTML5. HTML5 is latest major revision of the HTML standard. It's really the HTML we're all used to, just with more elements.

But HTML5 isn't one thing. It is a whole collection of features and technologies. While not all of these features are cross-browser compatible, the good news is many parts are ready for prime time so we can use HTML5 right now.

For instance, you can change your doctype to the HTML5 one (<!doctype html>). Your document is now HTML5. Because the HTML5 spec was based on a lot of work figuring out what browsers already do, things like this work. So, if you prefer the HTML5 syntax, feel free to do that now.

HTML

```
<!DOCTYPE html>
<html>
```

Note that unknown HTML elements are displayed as inline by all browsers, so we've added display: block; for new block-level elements in the main CSS file for older browsers

CSS

```
header, footer, nav, article, section, aside, figure,
figcaption {display: block;}
```

CSS MEDIA QUERIES

LEARN MORE - MEDIA QUERIES

If the mere mention of "responsive design" and "media queries" makes your eyes roll back in your head, we're going to unlock a bit of the mystery. A media query is simply a set of CSS rules that are only used when a device screen is within a specific size range.

CSS

```
@media screen and (max-width: 320px) {
  ...rules go here...
}
```

The example above can be translated as "When the viewer's screen is 320 pixels or less, display the content according to these CSS rules..." This is actually pretty cool, because you can tweak, resize, move, hide, and change almost anything so it displays well on a variety of mobile devices and screen sizes.

PUTTING IT TOGETHER

Want to play? In this scenario, the footer area has four divs that sit side by side on the desktop view. The last div holds a simple calendar, but you would rather not display that content to your visitors who are using a cell phone or a tablet.

The main CSS rules look like this:

CSS

```css
.footerBoxa {float: left; width: 23%; padding: 0; margin: 0 2% 0 0;}
.footerBoxb {float: left; width: 23%; padding: 0; margin: 0 1% 0 1%;}
.footerBoxc {float: left; width: 23%; padding: 0; margin: 0 1% 0 1%;}
.footerBoxd {float: left; width: 23%; padding: 0; margin: 0 0 0 2%;}
```

Since a cell phone screen is pretty small, we need the footer boxes to sit on top of each other rather than side by side. So we are going to target any screen that is smaller than 481 pixels, resize the footer boxes, remove the float property on some, and hide the last one:

MEDIA QUERY CSS

```css
@media screen and (max-width: 480px) {
.footerBoxa {float: none; width: 100%; margin: 0;}
.footerBoxb {float: none; width: 100%; margin: 0;}
.footerBoxc {float: none; width: 100%; margin: 0;}
.footerBoxd {visibility: hidden; display: none;}
}
```

But visitors who are on tablets have a larger screen. You would like for them to see the first two boxes sitting side by side, have the third box display below the top two but be wider, and still hide the fourth box:

MEDIA QUERY CSS

```css
@media screen and (min-width: 481px) and (max-width: 800px) {
.footerBoxa {float: left; width: 49%; margin: 0 1% 0 0;}
.footerBoxb {float: left; width: 49%; margin: 0 0 0 1%;}
.footerBoxc {float: none; width: 100%; margin: 0;}
.footerBoxd {visibility: hidden; display: none;}
}
```

Yes, this is a simplified example, but hopefully it will give you an idea of how the media queries work to serve your content to devices with smaller screens.

CSS COMMENTS

You can insert comments in CSS to explain your code. Like HTML comments, CSS comments will be ignored by the browser. A CSS comment begins with "/*", and ends with "*/". Comments can appear before or within rule sets as well as across multiple lines. They can also be used to comment out entire rules or individual declarations. Here are some examples:

CSS - BEFORE A RULE

```
/* ======== HEADER AREA ======== */
header {
   position: relative;
   width: 100%;
   height: 160px;
   text-align: center;
}
```

CSS - WITHIN A RULE

```
#nav ul li {
   width: 200px; /* you can increase or decrease width of submenu */
   float: left;
   padding: 0;
   margin: 0;
   border-radius: 0; /* removes radius set previously */
   font-size: 0.90em;
}
```

CSS - MULTIPLE LINES

```
/* ==================================== */
/* ==== CELL PHONE - PORTRAIT VIEW ==== */
/* ==================================== */
@media screen and (max-width: 320px) {
   #wrapper {
      width: 96%;
      margin: 0 2%;
   }
....
}
```

BORDER RADIUS

The CSS3 border-radius property is an easy way to give an element rounded corners, without having to use corner images or multiple div tags. We'll go through some examples of how to use the border-radius property on an image:

CSS

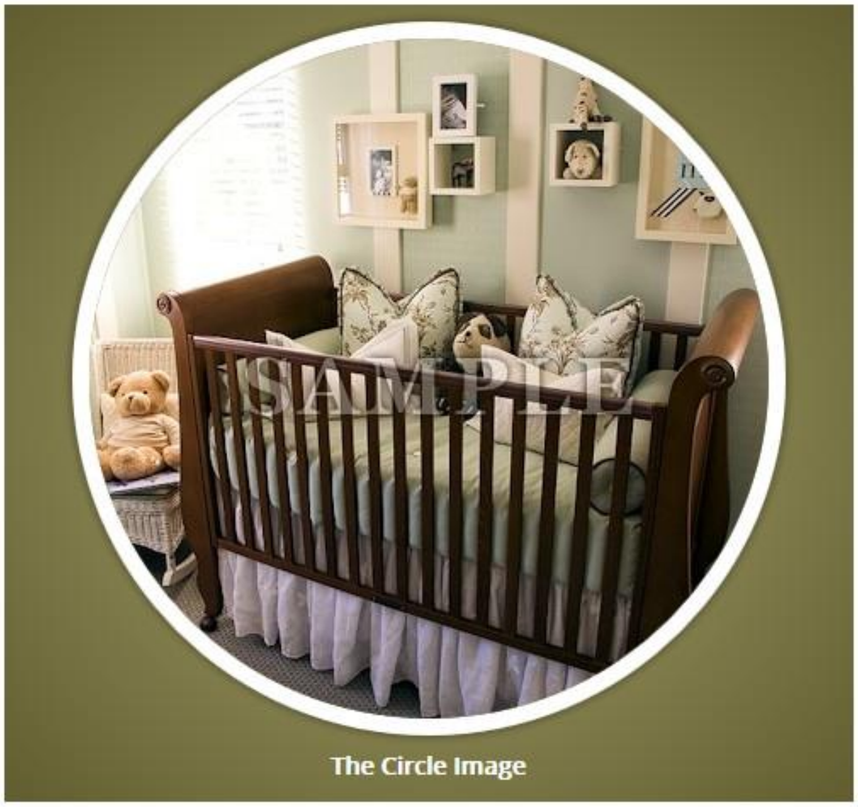
```
.rounded1 {
border-radius: 20px;
}
```

HTML

```
<p class="center"><img class="rounded1" alt=""
src="images/4720lg.jpg"></p>
```

THE BORDER-RADIUS PROPERTY CAN BE USED ON ANY BLOCK-LEVEL ELEMENT

Now let's style an image to look like a circle. This is the same principal as shown avoce, but by setting a radius of 50%, it will make a square image appear round. We'll also add a border and a nice drop shadow. Our image is 400px x 400px. Because we also want this image to contract in size on smaller devices, we first have to place the image into a <div> where we can control the width of the box.



CSS FOR THE BOX

```
.photoblock {
   display: block;
   padding: 10px;
   margin: 10px auto; /* to center the box */
   max-width: 500px; /* set the maximum width of the box */
   text-align: center; /* centers the box content */
   background-color: #aba873;
   background: radial-gradient(#aba873, #625d30);
}
```

CSS FOR THE IMAGE

```
.photoblock img {
   max-width: 100%; /* a percentage will contract on smaller devices
*/
   height: auto; /* keeps the height proportional to the width */
   border: 10px solid #fff;
   box-shadow: 0 0 5px rgba(0, 0, 0, 0.5);
   /* below the corners are rounded to half of the image width to make
the circle*/
   border-radius: 50%;
}
```

HTML

```
<div class="photoblock">
  <img alt="" src="images/mainimage01.jpg">
</div>
```

THERE ARE DOZENS OF FUN THINGS YOU CAN DO WITH A TOUCH OF CSS STYLING. FROM GRADIENT COLORS TO DROP SHADOWS, FROM TEXT EFFECTS TO MINIMUM/MAXIMUM WIDTHS AND HEIGHTS, FROM DROP-CAPS TO FANCY HORIZONTAL RULES - - THE SKY'S THE LIMIT ON WHAT YOU CAN DO.

BE SURE TO READ THROUGH YOUR TEMPLATE'S INSTRUCTIONS AND TYPOGRAPHY PAGES TO LEARN HOW WE STYLE MANY OF THE ITEMS WE USE ON THE VARIOUS PAGES OF YOUR WEB.

CSS RESOURCES

If you made it through the previous CSS lessons, you should have some basic knowledge about how CSS works....but the learning doesn't end here. You've taken the first baby steps to get a grasp of CSS1. Now you can move on to CSS2 and then into CSS3. Each step doesn't get harder, but it does expand on what you have learned.

WRITING YOUR CSS

A lot of how you write and organize your CSS file is just a matter of personal preference.

You can write your rules with properties and values on one line...

CSS

```css
.myclass {font-weight: bold; text-align: left;}
```

...or you can place each property and its value on a separate line.

CSS

```css
.myclass {
   font-weight: bold;
   text-align: left;
}
```

It does save time, however, if you stay organized in the external CSS file. I will normally place all of my HTML selectors first, starting with the body rule. Next I will organize any IDs in the order they are being used on the page. After the IDs, I will list the classes, again in the order they are being used. Last, I will place any miscellaneous classes, etc. that may not be used on every page.

To help you stay organized, you can add comments within your CSS file to help you recognize what a section is for. A CSS comment looks like this:

CSS

```css
/* -- Anything written here will not be read ----------- */
/* -- This is where I write important notes to you ----- */
/* -- Look for comments in your CSS file. It helps. ---- */
```

CSS LINKS

CSS School: The best teaching CSS site available. http://www.w3schools.com/CSS/

CSS Validator: Validate your CSS file using one of three different methods. http://jigsaw.w3.org/CSS-validator/

Markup Validation Service: Validate your HTML or XHTML coding at http://validator.w3.org/

THIS **DEVELOPER GUIDE** IS PRESENTED BY ROUND THE BEND WIZARDS FOR THE CONVENIENCE OF OUR CLIENTS AND CUSTOMERS. IF YOU HAVE QUESTIONS REGARDING A TEMPLATE PROBLEM OR AN ISSUE RELATING TO HOW TO BEST HANDLE A SITUATION WHERE YOU ARE USING ONE OF OUR TEMPLATES, WE WOULD BE HAPPY TO HELP YOU.