



# Present and layout with Cascade StyleSheets (CSS)

Release: 2010-07-20

## Table Of Contents

- [Webpage Presentation](#)
- [Attaching StyleSheets to HTML and XML Documents](#)
- [CSS](#)
  - ◆ [Selectors](#)
  - ◆ [Values And Units](#)
  - ◆ [At Rules \(@\)](#)
  - ◆ [Basic Styles](#)
  - ◆ [The Display Property](#)
  - ◆ [CSS Tables](#)
  - ◆ [Box Model](#)
  - ◆ [box-sizing](#)
  - ◆ [Floating Objects](#)
  - ◆ [Fonts](#)
  - ◆ [Transparency](#)
  - ◆ [border-radius](#)
  - ◆ [Shadows](#)
  - ◆ [Media Queries](#)
  - ◆ [Multiple Background Images](#)
  - ◆ [2D Transforms](#)
  - ◆ [Gradients](#)
- [Epilogue](#)

## ***Webpage Presentation***

In order to realistically present and layout information on webpages it is recommended that Cascade StyleSheets (CSS) are used on Markup documents such as HTML and XML including XHTML, SVG and MathML. This articlette is not exhaustive.

There are many types of media that StyleSheets could be used to provide different styles and layout for such as [screen](#): the full scope of visual and interactive CSS level 3 can be applied to webpages for a full user experience on desktop web browsers.

[print](#): used for hardcopy presentation of the webpage. Can have a different layout due to the paper size or if it is a left page, right page or even just for the first page. A lot of web browsers are configured to not print out backgrounds (possibly including element box borders). Can be viewed on the screen by Print Preview.

[projection](#): is used for displaying on projectors or printing to OHP projector sheets and even displaying in [Opera Show](#).

[handheld](#): including the Mobile Profile allows a view of the webpage on smart phones and PDAs.

[tv](#): this media, including the TV Profile, is for web-enabled television set-top boxes.

Other CSS Media Types include [braille](#), [embossed](#) for text-to-braille devices, [aural](#), [speech](#) for text-to-speech devices and software including Screen Readers.

If you have separate StyleSheet files for each media, some web browsers don't automatically change to the appropriate StyleSheet for that media - such as when you go from normal screen to print preview and the print styles are in a separate StyleSheet to the screen styles. If your web browser lists StyleSheets then just select the appropriate StyleSheet and then go to the media environment. For instance choose the [print](#) StyleSheet and then go to Print Preview or Print; select the [projection](#) StyleSheet before going to [Opera Show](#) (F11 in Opera Web Browser).

It maybe beneficial to put all the media styles into one StyleSheet file but separated by [@media](#) groups (see below).

## ***Attaching StyleSheets to HTML and XML Documents***

In addition to the [style](#) element in HTML and XHTML you can attach an external

## Present and Layout with Cascade Stylesheets (CSS) - Legend Scrolls

StyleSheet to apply to multiple webpages via the `link` void element in the `head` element as follows:

```
<link rel="stylesheet" type="text/css" href="mestyles.css" title="Main style">
```

**Figure 1:** Reference an external StyleSheet using HTML's link void element.

As you can see, Cascade StyleSheets' MIME Media Type is `text/css`. A `media` attribute is one of the ways to specify which media the stylesheet is for. You can refer to alternative StyleSheets that could be switched to from a web browser menu by prepending the `alternate` keyword to the `rel` attribute's value as:

```
<link rel="stylesheet" type="text/css" href="meprint.css" media="print" title="A Print StyleSheet">
```

```
<link rel="alternate stylesheet" type="text/css" href="meprintland.css" media="print" title="Print StyleSheet for Landscape">
```

**Figure 2:** More options and alternate StyleSheets.

The `title` attribute provides a name to be used for instance in a web browser's styles menu list.

StyleSheets can be attached to XML Documents (including Native XHTML) via a Processing Instruction (PI):

```
<?xml-stylesheet type="text/css" href="mestylesheet.css" title="Main Stylesheet"?>
```

**Figure 3:** Referencing an external StyleSheet with XML's xml-stylesheet PI.

This goes after the XML Declaration but before any other elements. The attribute `alternate` can be used to specify if the StyleSheet is an Alternate StyleSheet with the value `'yes'` or `'no'` (default).

```
<?xml-stylesheet type="text/css" href="differentColourScheme.css" media="screen" alternate="yes" title="Different Colour Scheme"?>
```

**Figure 4:** More options and alternative StyleSheets with xml-stylesheet PI.

You can have as many StyleSheet Links or XML StyleSheet PIs in a document.

## CSS

Within StyleSheets it may be required to add comments to clarify things:

```
/* This is a CSS Comment */
```

**Figure 5:** A CSS comment.

## Present and Layout with Cascade Stylesheets (CSS) - Legend Scrolls

The general layout of style code is to state what element(s) or similar you are applying the styles to - these are called the Selectors. Then wrapped in braces, { and }, you have one or more style properties of the form `propertyname: propertyvalue;`:

```
h2 {
    font-size: 125%;
    color: #0000ff;
    text-decoration: underline;
}
div.className {
    background-color: #003300;
    color: #ffff00;
    font-style: italic;
    width: 50%;
    margin-left: 5%;
}
span#me {
    font-weight: bold;
    color: #ff0000;
    line-height: 2em;
}
*.spaced {
    letter-spacing: 4px;
}
```

**Figure 6:** Examples of CSS properties.

The first applies the **h1 element's text size to be 125% of normal**, the **h1's text color to be blue** and the **h1's text to be underlined**.

Then any div element with a class attribute (a class selector) with the value of 'className' to have its **text background colour dark green and text colour yellow**, *style of the text to be italic*, the width of the div element's box to be 50% of the horizontal screen (whatever the physical size of the web content area and the screen) and the left side of the box to be 5% away from the normal position.

On the third line we attach styles to a span element with an id attribute (an id selector) with the value 'me' making the **text bold, and coloured red** and the line height be 2 times the current font size.

Last but not least we apply `spacing of 4 pixels between each letter` to any element with a class attribute with a value of 'spaced'.

You can increase the priority or importance to a CSS property by adding the **'!important'** keyword just after the value but before the semi-colon:

```
p.emph {
    font-style: italic !important;
}
```

**Figure 7:** The !important keyword.

## Selectors

In addition to the Element Selector (`p`) and Group Element Selectors (`p, h1, a`), Class Selectors (`p.myclass`) and ID Selectors (`p#theid`) shown above; you can apply styles to specific elements that are children or grandchildren or descendants of specific elements with the Descendant Selector such as all links within paragraphs (with class of 'news') within a div container:

```
div p.news a {
    color: blue;
}
```

**Figure 8:** Class Selectors.

An element that is a direct child such as a quote within any list items:

```
li > q {
    font-style: italic;
}
```

**Figure 9:** Direct Child Selector.

A direct sibling such as a paragraph with a heading as a direct sibling:

```
h1 + p {
    margin-top: 0;
}
```

**Figure 10:** Direct Sibling Selector.

Styling elements with attributes in general can be done using the various Attribute Selectors such as a paragraph with a `lang` attribute:

```
p[lang] {
    background-color: cyan;
    color: black;
}
```

**Figure 11:** Attribute Selector.

An element with an attribute with a specific value such as a text box (input element with a `type` attribute with the value 'text'):

```
input[type="text"] {
    font-size: 100%;
}
```

**Figure 12:** Attribute Value Selector.

Matching one of the space separated attribute values in `<p class="mainstyle substyle uberstyle"></p>` can be achieved by the following:

## Present and Layout with Cascade Stylesheets (CSS) - Legend Scrolls

```
p[class~="substyle"] {  
    line-height: 1.5em;  
}
```

**Figure 13:** Space-separated Attribute Value Selector.

And matching the first part of a dash separated attribute value such as `<p lang="en-GB"></p>` can apply styles by:

```
p[lang|="en"] {  
    font-family: Verdana;  
}
```

**Figure 14:** Dash-separated Attribute Value Selector.

Pseudo Selectors add the ability to style elements in particular conditions such as when the pointer is hovering over it:

```
p:hover {  
    color: blue;  
    font-weight: bold;  
}
```

**Figure 15:** Hover Selector.

Or the element has the current focus:

```
p:focus {  
    border-width: 2px;  
    border-style: outset;  
    border-color: #0000ff;  
}
```

**Figure 16:** Focus Selector.

A normal hyperlink:

```
a:link {  
    color: #ff0000;  
}
```

**Figure 17:** Link Selector.

A visited link:

```
a:visited {  
    color: #006600;  
}
```

**Figure 18:** Visited Link Selector.

An element with a `lang="en-GB"` attribute:

```
p:lang(en-GB) {  
    background-color: blue;  
    color: white;  
}
```

**Figure 19:** Language Selector.

## Present and Layout with Cascade Stylesheets (CSS) - Legend Scrolls

To style the first paragraph:

```
p:first-child {  
    font-weight: bolder;  
}
```

**Figure 20:** First Child Selector.

Attribute Selectors are also expanded to allow Starts With:

```
a[href^="https"] {  
    background-color: yellow;  
}
```

**Figure 21:** Starts With Attribute Value Selector.

Ends With:

```
a[href$=".html"] {  
    background-color: green;  
}
```

**Figure 22:** Ends With Attribute Value Selector.

Contains Substring:

```
input[type*="date"] {  
    background-color: aqua;  
}
```

**Figure 23:** Contains Substring Attribute Value Selector.

Trident 3 based browsers like Internet Explorer 7 support those selectors above except `:focus` and `:lang()` but Internet Explorer 8 Standards Mode (Trident 4 Standards Mode) supports these two.

Pseudo Selectors also expand with styling the last child such as the last paragraph:

```
p:last-child {  
    border-right: 1px solid #000off;  
    border-bottom: 1px solid #000off;  
}
```

**Figure 24:** Last Child Selector.

You can style the selected text or object (note the double colon):

```
div::selection {  
    background-color: black;  
    color: white;  
}
```

**Figure 25:** Selection Selector.

Mozilla based browsers like Firefox support an experimental version of `::selection` as `::-moz-selection`:

```
div::-moz-selection {  
    background-color: black;  
    color: white;  
}
```

**Figure 26:** Mozilla-based Selection Selector.

`:root` will select the top element such as the `html` element for HTML and XHTML webpages, the `svg` element for SVG Images and the `feed` element for Atom Feeds.

`:empty` selects the elements that have no content including text.

`:not()` provides a negation such as all list items except the first one:

```
li:not(:first-child) {  
    font-family: georgia, serif;  
}
```

**Figure 27:** Negation Selector.

You can even target say the fifth paragraph:

```
p:nth-child(5) {  
    font-style: italic;  
}
```

**Figure 28:** Nth Child Selector.

Or every second list item:

```
li:nth-child(2n) {  
    font-style: italic;  
}
```

**Figure 29:** Nth Child Selector variant 1.

Or even the first paragraph then every third paragraph after that:

```
p:nth-child(3n+1) {  
    font-style: italic;  
}
```

**Figure 30:** Nth Child Selector variant 2.

So to do alternate colours for lists items or other multiple row content:

```
li:nth-child(2n+1) {  
    background-color: aqua;  
    color: darkblue;  
}  
li:nth-child(2n) {  
    background-color: darkblue;  
    color: aqua;  
}
```

**Figure 31:** Nth Child Selector variant 3.

## Present and Layout with Cascade Stylesheets (CSS) - Legend Scrolls

Negative values allows more flexibility such as the sixth child and every second sibling before it:

```
li:nth-child(-2n+6) {
    font-family: Verdana, sans-serif;
}
```

**Figure 32:** Nth Child Selector with negatives.

Or even the last child and every second sibling before it:

```
li:nth-child(-2n-1) {
    color: darkblue;
}
```

**Figure 33:** Nth Child Selector with more negatives.

`:nth-last-child()` is similar but instead of initially from the top it is from the bottom of the list.

`:nth-of-type()` is the same as `:nth-child()` except siblings should also have the same name as the element in the selector or the same selector before `:nth-of-type()` such as:

```
p.important:nth-of-type(2n+1) {
    border: 1px dashed red;
}
```

**Figure 34:** Nth Of Type Selector.

`:first-of-type` and `:last-of-type` match the first or last of the same named element or same selector such as:

```
dt:first-of-type {
    border: 1px solid blue;
}
```

**Figure 35:** First Of Type Selector.

`:only-child` matches the element that is the only child of its parent.

`:only-of-type` matches the element that is the only named child of its parent.

`:enabled` and `:disabled` selectors style form controls that are not disabled or are disabled respectively.

`:target` applies styles to an element that has an id attribute value the same as the current # part of an URI. For instance 'mepage.html#me' containing `<p id="me"></p>` will apply styles to this particular paragraph with:

```
p:target {
    font-style: italic;
    font-weight: bold;
}
```

**Figure 36:** Target Selector.

Mozilla 1.9.1 based web browsers like Firefox 3.5 do support `:nth-*` or `*-type`

selectors. Trident 5 based (Internet Explorer 9) will support the `:nth-*`, `*-type`, `:root`, `:not()`, `:enabled`, `:disabled`, `:checked`, `:target` or any other modern useful selector when it comes out but is currently still in development.

## Values And Units

Numeric values can be of these relative units: percentages (%), relative to the font size (em), relative to the height of the font's lowercase x character (ex).

Or could be an absolute unit: character point size (pt), centimeters (cm), millimeters (mm), picas (pc).

The pixels unit (px), according to the CSS specification, is a relative unit as it is relative to the resolution: the higher the resolution, the higher the amount of pixels per inch, the smaller the objects become.

But not all devices allow you to change the resolution, such as mobile phones, and 99.9% of the time you won't be changing the resolution of your computer screen or digital TV so practically the pixel unit is an absolute unit. Most web designers actually use the pixel unit as an absolute unit for fixed width layouts.

These days Liquid Layout or Elastic Layout is preferred to Fixed Layout and in version 2 of the Web Content Accessibility Guidelines (WCAG), the pixel unit is not counted as a relative unit because of the fact that in some devices the unit is effectively an absolute unit.

Some values need to refer to external files such as images would use the url function: `url('path/to/image')` (or `url("path/to/image")`).

Colour values may be in one of several forms such as a hexadecimal value or an rgb function (red, green, blue) or a HTML4/SVG keyword such as `#ff0000` or `rgb(255,0,0)` or `red`.

Another function to provide colour is `hsl()`. Rather than red, green and blue values it has Hue (the main colour in degrees), Saturation, in a percentage (main colours have this as 100%) and Lightness, also in a percentage (main colours have this as 50%). The higher the percentage of lightness, the lighter the colour and a lower percentage represents a darker form of the colour.

Such as the red HSL colour:

```
color: hsl(0, 100%, 50%);
```

**Figure 37:** HSL Colour function.

Most web browsers support `hsl()`. Naturally Microsoft Trident based like Internet Explorer does not.

Most style properties can have a value of `inherit` that takes the parent elements value for that style property.

## At Rules (@)

As styles can be for different media like colour screens, printed material, small screens or speech and some styles are only used for a particular media then you can use @media rules:

```
@media screen, projection {
    thebox#handle {
        background-color: #ff0000;
        color: #ffff00;
        font-weight: bold;
    }
} /* End core styles for Screen and Projection Media */

@media screen {
    thebox#handle {
        line-height: 2em;
    }
} /* End extra styles for Screen Media */

@media print {
    thebox#handle {
        line-height: 1.5em;
        font-style: italic;
    }
} /* End styles only for Print Media */
```

**Figure 38:** Media At Rules.

## Basic Styles

For simple text stylings like size, boldness, italics, lining and colour you have these style properties:

**font-size** can have percentage values (50%, 125%, 3%, etc) or more specific numbered values like 16px for 16 pixels or 14pt for 14 point. It can have these relative terms **xx-small**, **x-small**, **small**, **medium**, **large**, **x-large** and **xx-large**. It is best to use percentages, relative terms, ex or em for font sizes.

**font-weight** has values of **normal**, **bold**(to replace **b** elements), **bolder**, **lighter** and **100**, **200**, **300**, **400**, **500**, **600**, **700**, **800** and **900**.

**font-style** values are **italic**(to replace **i** elements), **normal** and **oblique**.

**text-decoration** includes **underline** (to replace **u** elements), **overline** and **line-through** (to replace **s** and **strike** elements) as well as **none**.

**color** uses any of the colour hex / rgb / keyword for the HTML4 / SVG colours and also can have **none** or **transparent**.

Elements including the webpage body have properties to change the

## Present and Layout with Cascade Stylesheets (CSS) - Legend Scrolls

background including colour, position, repetition and to set a background image via either the collection of `background-color`, `background-position`, `background-attachment`, `background-repeat` and `background-image` or the compact `background` property.

`background-color` takes the same values as `color`. `background-image` takes a URI (using the `url` function) to reference an image. The relative path to the image is relative from the StyleSheet. `background-position` has a pair of % or length values where the first is of the horizontal positioning and the second is the vertical positioning of the background image. You can use tokens such as `left`, `center` or `right` for the first value and `top`, `center`, `bottom` for the second.

To fix the background so it doesn't move when you scroll the page you can use `background-attachment: fixed`; or use `background-attachment: scroll`; to return it to the default. Background images can be repeated with `background-repeat: repeat`;

`background-repeat: repeat-x`; allows you to just repeat across and `background-repeat: repeat-y`; only repeats downwards. `background-repeat: no-repeat`; forces no repetition.

```
body {
    background-color: #ffffff;
    background-image: url('../images/fadedlogobkgrnd.png');
    background-repeat: no-repeat;
    background-attachment: fixed;
    background-position: right bottom;
}
div.patterned {
    background: transparent url('../images/pattern024.png');
}
```

**Figure 39:** Background properties.

## The Display Property

You can specify that elements are of a specific layout type or not displayed at all using the `display` style property:

To hide elements use the `none` value. Other values include `inline`, `block`, `inline-block`, `list-item`, `table`, `table-row`, `table-cell`, `table-column` and `table-caption`:

```
fulltextel {
    display: block;
    font-size: 110%;
    color: #0000cc;
}
emphit {
    display: inline;
    font-style: italic;
}
listit {
```

```
    display: list-item;
    list-style-type: square;
    list-style-image: url('../images/listit.png');
}
cell.standard {
    display: table-cell;
    width: 35%;
}
span.msiecell {
    display: inline-block;
    width: 35%;
}
```

**Figure 40:** Display property.

The default value of `display` is `inline` which allows the element to be embedded in a line amongst other inline elements and text. Some positioning styles cannot work on inline elements. These elements tend to have one or more line boxes (see below).

Most elements are block level (`display: block;`) and so put themselves on their own line. These types of elements can have multiple lines within its box with other block level, inline level elements and text.

The value `inline-block` provides an element to be embedded in a line like inline elements but the element itself is a block element.

To style lists you need to have a `display: list-item;` as one of the style properties along with `list-style-type` which formats a shaped list marker such as `square`, `disc`, `circle`, etc and optionally the `list-style-image` takes a URI of an image to represent the list marker. If the environment does not support image list markers or it can't find the image then it will fallback to the shaped marker.

## CSS Tables

The term 'CSS Tables' refers to the collection of table values of the `display` property: `table`, `inline-table`, `table-column-group`, `table-column`, `table-header-group`, `table-footer-group`, `table-row-group`, `table-row`, `table-cell` and `table-caption`.

These provide a tabular layout similar to (X)HTML Tables but provide fine-grained style and flexibility due to the very nature of CSS; as opposed to the rough concrete layout of (X)HTML Tables which display no matter what device you are using even if the device does not have enough view to successfully display the layout table. (X)HTML Tables are developed only for semantic (meaningful) tabular data and the web browser requires processing a highly complex algorithm to layout (X)HTML Tables. CSS Tables provide layout and does not use the (X)HTML Table algorithm.

```
PageBox {
  display: table;
  width: 90%;
  border: 1px solid blue;
}
PageBox div {
  display: table-cell;
  font-size: 1.2em;
}
```

**Figure 41:** Tabular layout with CSS Tables.

As in the example above we specify a PageBox element to display like a table box and any div elements within that will be like table cells. An anonymous table row box will be processed around the table-celled elements. Even if we removed the PageBox styling then an anonymous table box and table row box will surround the table-celled elements.

## Box Model

Markup elements have a kind of invisible box around them that structures a margin, border, padding and content model. Margins can be altered by setting the percentage or other numeric length to either any of these style properties: [margin-top](#), [margin-right](#), [margin-bottom](#), [margin-left](#) or by using the compact style property [margin](#) providing one to four of the values:

If just one value then it refers to all four sides; if two values then the first refers to top and bottom sides and the second refers to the left and right sides. For three values the first refers to the top, second to the left and right and the bottom side is handled by the third value. The four values respectively refer to top, right, bottom and left.

```
Boxemall {
  margin: 2px;
}
quoteTheBlock {
  margin: 2px 6px;
}
spacebucket {
  margin: 2px 6px 4px;
}
variablesides {
  margin: 2px 6px 4px 10px;
}
```

**Figure 42:** Margins.

## Present and Layout with Cascade Stylesheets (CSS) - Legend Scrolls

Borders can be styled using these three properties: `border-width` using `thin` or `medium` or `thick` or size values to state the border thickness.

`border-style` can be `none` to hide any border or any of `solid`, `dashed`, `dotted`, `double`, `ridge`, `groove`, `inset` or `outset` to present the border.

`border-color` states the colour of the border.

Or you can use a compact version: `border` that has the `border-width` value, a space, `border-style` value, a space and the `border-color` value such as `border: 1px solid #000000;` for a simple black border.

Padding is kind of the same principle as margins except it is for between the border and the actual content - so its a kind of buffer. `padding-top`, `padding-right`, `padding-bottom`, `padding-left` and `padding` have the same kind of value forms as the margin properties.

```
padaway {
    padding-left: 6px;
}
pademall {
    padding: 2px;
}
padTheQuote {
    padding: 2px 6px;
}
padbucket {
    padding: 2px 6px 4px;
}
variablepads {
    padding: 2px 6px 4px 10px;
}
```

**Figure 43:** Padding.

You can set the `width` and `height` of these invisible boxes too using style properties of the same name.

```
div.dsb {
    width: 7.81em;
    height: 12.50em;
    border-width: 1px;
    border-style: solid;
    border-color: #ff0000;
}
```

**Figure 44:** Width and Height.

produces ([this example is more accurate on the CSS webpage](#)):

The World

Wide Web  
Consortium  
(W3C) develops  
interoperable  
technologies...

**Figure 45:** Box styled with CSS Width and Height.

`top`, `left`, `right` and `bottom` are style properties that can push the box to a specific location on the webpage. By default the values are relative to the surrounding elements or the web content area. But this can be changed with the `position` style property providing `relative` value or `absolute` value which makes the location relative to the web content area and the box is out of the normal flow of the webpage. A value of `fixed` can make the element appear like a watermark - totally fixed in place no matter if you scroll the webpage or not.

## box-sizing

`box-sizing: content-box;` makes the sizes of `width`, `height`, `min-width`, `max-width`, `min-height`, `max-height` not include padding and border sizes. Such as:

```
p.content {  
    box-sizing: content-box;  
    width: 100px;  
    padding: 2px;  
    border-width: 2px;  
    border-style: solid;  
    border-color: #ff0000;  
}
```

**Figure 46:** Borders and padding are not included in box sizes.

This paragraph's full width is 104 pixels.

Whereas `box-sizing: border-box;` makes the sizes of `width`, `height`, `min-width`, `max-width`, `min-height`, `max-height` include padding and border sizes. Such as:

```
p.border {  
    box-sizing: border-box;  
    width: 100px;  
    padding: 2px;  
    border-width: 2px;  
    border-style: solid;  
    border-color: #ff0000;  
}
```

**Figure 47:** Borders and padding are included in box sizes.

This paragraph's full width is 100 pixels and the content's width is calculated to

be 96px.

Mozilla based web browsers use the experimental `-moz-box-sizing` property, WebKit based like Safari and Chromium based like Google Chrome use `-webkit-box-sizing` property. Presto based like Opera, KHTML based like Konqueror and MS Trident 4 Standards Mode based such as Internet Explorer 8 Standards Mode uses the `box-sizing` property.

## Floating Objects

Wrapping elements to the left or right of another element, possibly to create a multiple column effect, or just to have text flow around an image, you can use the `float` property with either `left` or `right` or even `none`. But some floating elements may affect the positioning of other elements. In this case you have the `clear` property with `left`, `right`, `both` or `none` to push the element below as if it were a block element (if it is or not).

## Fonts

`font-family` style property provides access to font types for the text. The value is a comma separated list of font names - if spaces are in the font names then you should surround the name in either double or single quotes:

```
div.s1 {  
    font-family: 'Times New Roman', Helvetica, sans-serif;  
}
```

**Figure 48:** Specifying a font.

## Transparency

To make an element partially transparent you can use either a transparent colour function `rgba()` or `hsla()` in some web browsers such as Firefox, Safari 3, Google Chrome and Konqueror 4; or the `opacity` property from CSS level 3 color module in most web browsers such as Mozilla Firefox, Opera, Konqueror, Google Chrome and Apple Safari. Opera 10 now has support for `rgba()` and `hsla()`.

The functions are like the non-transparent versions except they have a fourth value for opacity such as `rgba(255, 0, 0, .4)` or `hsla(0, 100%, 50%, .4)` for a 60% transparent red colour.

The fourth value for the functions and the value of the `opacity` property is from `1.0`, not transparent, through to `0.0`, fully transparent.

```
InfoBox.fancy {  
    width: 200px;
```

```
float: right;
border: 1px solid #0000ff;
opacity: 0.75;
}
```

**Figure 49:** Transparency with opacity.

This will make a floating box a quarter transparent. Trident 4 and under based like Internet Explorer 8 and under do not support the functions or `opacity` property but does support an incompatible syntax as:

```
InfoBox.fancy {
width: 200px;
float: right;
border: 1px solid #0000ff;
filter: alpha(opacity=75);
}
```

**Figure 50:** Microsoft's Alpha Filter (Boo Hiss).

But this is not CSS Compliant so in Internet Explorer 8 (IE8) Standards Mode it supports this and a new CSS Vendor Specific Mechanism version as:

```
InfoBox.fancy {
width: 200px;
float: right;
border: 1px solid #0000ff;
-ms-filter: "alpha(opacity=75)";
}
```

**Figure 51:** Microsoft's Workaround Alpha Filter (Boo Hiss).

One drawback with Microsoft's filter is that the element that the styles are to be applied requires some sort of explicit layout. Such as you must specify a `width` or `height` or `position: absolute`. In Trident 5 (IE9), Microsoft has implemented the actual CSS3 `opacity` property.

## border-radius

You can round border corners using CSS3's `border-radius` property which takes a length as its value:

```
.bubbleit {
border-radius: 3px;
}
```

**Figure 52:** Rounding your corners.

You can specify particular corners with the `border-top-left-radius`, `border-top-right-radius`, `border-bottom-left-radius` and `border-bottom-right-radius` properties.

Currently no web browser supports these but Mozilla based like Firefox do support an experimental version as `-moz-border-radius`, `-moz-border-radius-topleft`, `-moz-border-radius-topright`, `-moz-border-radius-bottomleft` and `-moz-border-radius-bottomright`.

WebKit based such as Safari and Chromium based such as Google Chrome supports `-webkit-border-radius`, `-webkit-border-top-left-radius`, `-webkit-border-top-right-radius`, `-webkit-border-bottom-left-radius` and `-webkit-border-bottom-right-radius`.

Plus KHTML 4.2 based like Konqueror 4.2 supports `-khtml-border-radius`, `-khtml-border-top-left-radius`, `-khtml-border-top-right-radius`, `-khtml-border-bottom-left-radius` and `-khtml-border-bottom-right-radius`.

Trident has no equivalent but Trident 5 Standards Mode (IE9 Standards Mode) will have the actual `border-radius`, `border-top-left-radius`, `border-top-right-radius`, `border-bottom-left-radius` and `border-bottom-right-radius` set of properties.

## Shadows

Text shadows can be applied using `text-shadow` with either the value 'none' or one or more groups of a number referring to the horizontal direction, positive number is right and negative number is left. The second number is the vertical direction: positive for downwards and negative for upwards. An optional third number provides how much blur is applied to the shadow. Also an optional colour value will state the colour of the shadow. Such as:

```
h2 {
    text-shadow: 3px 4px 3px #009900;
}
```

**Figure 53:** Make your text float.

Mozilla 1.9.1, Presto, WebKit, Chromium 3 and KHTML based web browsers support text shadows. Microsoft Trident based does not but uses an incompatible buggy filter syntax as:

```
h2 {
    width: 99%;
    filter: Shadow(Color: #009900 Direction: 135 Strength: 4);
}
```

**Figure 54:** Microsoft's Shadow (Shudder).

The Direction is in degrees and the Strength is how far the shadow will go. This filter syntax is not CSS Compliant but in IE8 Standards Mode supports this as well as a CSS Vendor Compliant version:

```
h2 {
    width: 99%;
    -ms-filter: "Shadow(Color: #009900 Direction: 135 Strength: 4)";
}
```

**Figure 55:** Microsoft's Workaround Shadow (Shudder).

Trident's Shodow filter is very buggy and also requires some layout on the element it is styling.

Shadows can be applied to element boxes too:

```
h2 {
```

```
    box-shadow: 3px 4px 3px #009900;
}
```

**Figure 56:** Make your boxes float.

Mozilla 1.9.1 supports the experimental `-moz-box-shadow` and WebKit and Chromium based support the `-webkit-box-shadow` property. Trident's Shadow filter applies to boxes and images as well.

## Media Queries

As an extension to the `@media` rules, Media Queries adds the ability to apply groups of styles to media with various sub details such as not just to a screen but a colour screen as:

```
@media screen and (color) {
    /* ... */
}
```

**Figure 57:** Querying colour screens.

For a range you can use `min-` and `max-` prefixes to any of the new terms which include `color`, `width`, `height`, `device-width`, `device-height`, `aspect-ratio`, `device-aspect-ratio`, `color-index`, `monochrome`, `orientation` and `resolution`:

```
@media screen and (max-width: 400px), handheld and (max-width: 400px) {
    /* ... */
}
```

**Figure 58:** Querying 400 pixel maximum large and small screens.

Apply to a 24-bit colour screen:

```
@media screen and (color: 24) { /* ... */ }
```

**Figure 59:** Querying 24-bit and higher colour screens.

The difference between `width` and `device-width` is that `width` is the width of the webpage (or webpage on paper) and `device-width` is the width of the screen (or paper).

`aspect-ratio`:

```
@media screen and (min-aspect-ratio: 16/9) { /* ... */ }
@media screen and (aspect-ratio: 1280/720) { /* ... */ }
```

**Figure 60:** Querying screen aspects.

`color-index` refers to the number of colours supported in the device's colour table (just like there are 256 colours in a GIF or 8-bit PNG image).

`monochrome` and `orientation` respectively provide the number of bits in a monochrome display and whether it is a landscape or portrait display.

Applying styles to a device with a certain amount of dots per inch resolution can be accomplished by:

```
@media print and (resolution: 300dpi) { /* ... */ }
```

**Figure 61:** Querying print resolution.

Mozilla 1.9.1 and Presto supports Media Queries while WebKit and Chromium based support these query features except [resolution](#).

## Multiple Background Images

In CSS 3 you can apply multiple background images with a single declaration using the same [background-image](#), [background-position](#) and [background-repeat](#) properties by using a comma separated list of values each referring to a different background layer. The first value is the top most layer, the second value is the layer under that, third value is for the layer under that and so on.

```
.compositbkgd {  
    background-image: url('../images/logo.png'), url('../images/bird.png'),  
    url('../images/bird.png'), url('../images/pattern.png');  
    background-repeat: no-repeat, no-repeat, no-repeat, repeat-x;  
    background-position: 5px 0, 20px 5px, 23px 10px, 0 0;  
}
```

**Figure 62:** A box composite background

So far only Mozilla 1.9.2 based like Firefox 3.6, Apple WebKit 525 based like Safari 3, Chromium based like Google Chrome and KHTML 4 based like Konqueror support CSS 3 Multiple Background Images.

## 2D Transforms

Objects on the webpage or web application, including text, can be transformed such as scaled, rotated, skewed (stretched) and translated using the [transform](#) property.

A [transform-origin](#) property with a space separated x value and y value can change the center of the object transformation so that it is not necessarily the very center of the object. Each value can be in typical CSS units including pixels and percentages. Alternatively the x value could be [left](#) or [center](#) or [right](#) and the y value could be [top](#) or [center](#) or [bottom](#). Such as [transform-origin: 25% 35%](#); making the transform origin at 25% by 35% from the left, top of the object (the default is [50% 50%](#)).

The value of the [transform](#) property is usually either one transform function or a space separated list of transform functions.

- Translating: specifies a new location for the origin of the object – effectively moving the content. The [translate\(\)](#) function takes an x,y coordinate with units. For instance [transform: translate\(125px, 125px\)](#); changes the origin to 125 pixels by 125 pixels from the previous origin. The second number is optional and if omitted will be zero. Plus a [translateX\(\)](#) function just moves the horizontal origin and the [translateY\(\)](#) function just moves the vertical origin.

## Present and Layout with Cascade Stylesheets (CSS) - Legend Scrolls

- **Scaling:** gracefully resizes the object. The `scale()` function takes an x,y coordinate. For instance `transform: scale(2, 2);` increases to twice the size or `transform: scale(.5, .5);` decreases to half the size. The second number is optional and if omitted then is the same as the first. Plus `scaleX()` only handles the horizontal and `scaleY()` only handles the vertical sizing or zooming.
- **Rotation:** rotates the object. The `rotate()` function takes a number in degrees and the `deg` unit. For instance `transform: rotate(-8deg);` rotates 8 degrees anti-clockwise or `transform: rotate(18deg);` rotates 18 degrees clockwise.
- **Stretching:** stretches the object. The `skew()` function takes an x,y coordinate with the `deg` units to stretch both horizontal and vertical directions of the object. Plus `skewX()` function takes an angle to stretch horizontally and the `skewY()` function takes an angle to stretch vertically. For instance `transform: skewX(45deg);` or `transform: skewY(15deg);` or even `transform: skew(45deg, 8deg);`.

There is a `matrix()` transform function with 6 values that is the core, advanced transform that will do translations, scaling, stretching and rotation but most authors use the `translate()`, `scale()`, `skew()` and `rotate()` functions instead.

## Gradients

For the value of `background-image` and `list-style-image` properties you can specify CSS Colour Gradients. Gradients can be defined as either linear or radial.

Linear gradients are specified via the `linear-gradient()` function. In this function you can have an optional background position value or an angle value for the direction of the gradient. If not used then the position is `'top'` for top to bottom by default.

Then you need at least two colour stops, all separated with commas. Each colour stop may have, in addition to the colour value, a space then a percentage or length value to specify the colour stops position with the gradient line.

```
background-image: linear-gradient(white, blue);  
background-image: linear-gradient(top left, white, cyan, blue);  
background-image: linear-gradient(45deg, #226622, #ffff00 46%, magenta 90%,  
black);
```

**Figure 63:** CSS linear gradients

Radial gradients are specified via the `radial-gradient()` function. In this function you can also have an optional background position value or an angle value for the direction and the center of the gradient. If not used then the position is `'center'` for the center of the background area by default.

Followed by an optional group of a comma and an optional shape then an

## Present and Layout with Cascade Stylesheets (CSS) - Legend Scrolls

optional space and size. This shape can be a 'circle' or by default is an 'ellipse'. This size can be 'closest-side', 'closest-corner', 'furthest-side', 'furthest-corner', 'contain' or 'cover'.

Then you need at least two colour stops, all separated with commas and may have percentages or lengths for each colour stop.

```
background-image: radial-gradient(circle, white, blue);
background-image: radial-gradient(left bottom, circle, white, cyan 10%, blue 25%,
green 52%, red 75%, black);
background-image: radial-gradient(45deg, ellipse, #226622, #ffff00 46%, magenta
90%, black);
```

**Figure 64:** CSS radial gradients

So far no web browser supports proper CSS Gradients. Mozilla based supports the same syntax but with the `-moz-linear-gradient()` and `-moz-radial-gradient()` functions.

Apple WebKit and Google Chromium based support an alternative syntax with the single `-webkit-gradient()` function. The first parameter states whether it is 'linear' or 'radial'.

Then two pairs of positioning, almost like the background position but without the 'center' keyword and only the percentage value will have the percent unit. If the type is 'radial' then each position, in addition, will have a comma and a number for the radius.

Afterwards are at least two colour stops. Each colour stop is represented as a `color-stop()` function with a parameter stating the colour stop's position between 0.0 and 1.0 or a percentage; and a parameter of the colour itself. You can also use the `from()` function with a colour parameter, instead of the first colour stop function and use the `to()` function with a colour parameter, instead of the last colour stop function.

```
background-image: -webkit-gradient(linear, left bottom, right top, from(green),
color-stop(90%, red), to(rgba(0,0,0,0.2)));
```

```
background-image: -webkit-gradient(radial, 50 50, 10, 50 50, 30, from(green),
color-stop(90%, red), to(rgba(0,0,0,0.2)));
```

**Figure 65:** WebKit and Chromium's CSS gradients

## ***Epilogue***

For more information about Cascade StyleSheets (CSS) you can visit <http://www.w3.org/standards/techs/css>.

## Present and Layout with Cascade Stylesheets (CSS) - Legend Scrolls

This article and others are available online and in other document formats at:

<http://www.legendscrolls.co.uk/webstandards/>

Author: Peter Davison from [Legend Scrolls](#).

Copyright ©2005-2010 Legend Scrolls and Peter Davison.  
All rights reserved.