This book has been prepared exclusively for

VEIS COMPUTER EDUCATION

CSS- Cascading Style Sheets

CSS Introduction

What is CSS?

- **CSS** stands for **C**ascading **S**tyle **S**heets
- Styles define **how to display** HTML elements
- Styles were added to HTML 4.0 **to solve a problem**
- **External Style Sheets** can save a lot of work
- External Style Sheets are stored in **CSS files**

Cascading Style Sheets (CSS) is a fantastic tool to add layout to your websites. It can save you a lot of time and it enables you to design websites in a completely new way. CSS is a must for anyone working with web design. A simple text editor is ideal for learning HTML and CSS. CSS is a style language that defines layout of HTML documents. For example, CSS covers fonts, colours, margins, lines, height, width, background images, advanced positions and many other things. HTML can be used to add layout to websites. But CSS offers more options and is more accurate and stylish. CSS is supported by all browsers today.

Difference between CSS and HTML:

HTML is used to structure content. CSS is used for formatting structured content.

World Wide Web, the language HTML was only used to add structure to text. CSS was invented to provide web designers with sophisticated layout opportunities supported by all browsers. At the same time, separation of the presentation style of documents from the content of documents, makes site maintenance a lot easier.

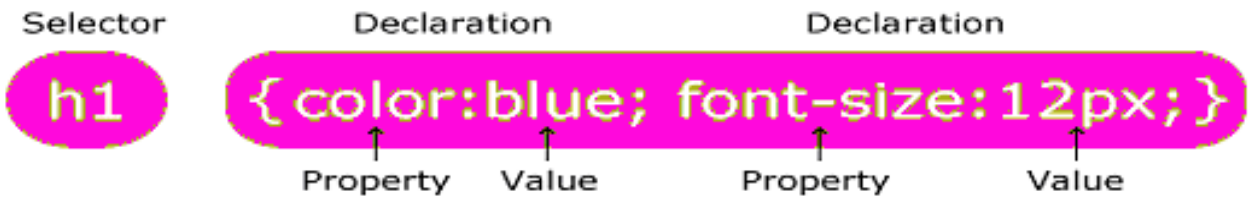
CSS benefits:

CSS was a revolution in the world of web design. The concrete benefits of CSS include:

- control layout of many documents from one single style sheet;
- more precise control of layout;
- apply different layout to different media-types (screen, print, form etc.);
- advanced, stylish and sophisticated techniques.

CSS Syntax

A CSS rule set consists of a selector and a declaration block:



The selector points to the HTML element you want to style. The declaration block contains one or more declarations separated by semicolons. Each declaration includes a property name and a value, separated by a colon.

CSS Example

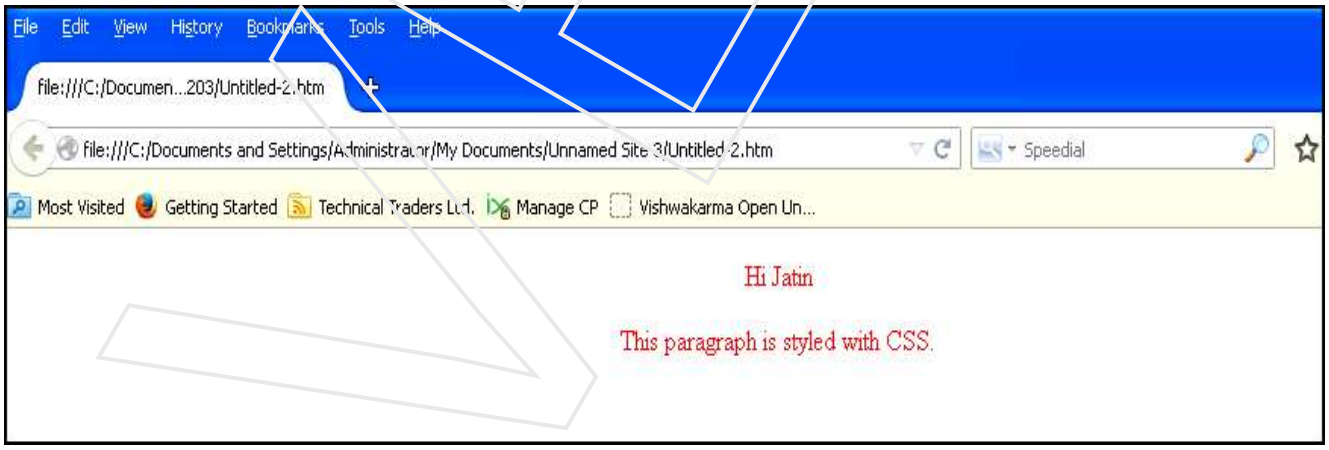
A CSS declaration always ends with a semicolon, and declaration groups are surrounded by curly brackets: p {color:red;text-align:center;}

To make the CSS more readable, you can put one declaration on each line, like this:

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
p
{
color:red;
text-align:center;
}
</style>
</head>

<body>
<p>Hi Jatin</p>
<p>This paragraph is styled with CSS.</p>
</body>
</html>
```

Result:



Applying CSS to an HTML document

There are three ways you can apply CSS to an HTML document. Detail of these methods are given below. We recommend that you focus on the third method i.e. **external.**

Method 1: In-line (the attribute style)

One way to apply CSS to HTML is by using the HTML attribute `style`. Building the red background color, it can be applied like this:

```
<html>
  <head>
        <title>Example</title>
  </head>
  <body style="background-color: #FF0000;">
        <p>This is a red page</p>
  </body>
</html>
```

Method 2: Internal (the tag style)

Another way is to include the CSS codes using the HTML tag `<style>`. For example like this:

```
<html>
  <head>
        <title>Example</title>
        <style type="text/css">
          body {background-color: #FF0000;}
        </style>
  </head>
  <body>
        <p>This is a red page</p>
  </body>
</html>
```

Method 3: External (link to a style sheet)

The recommended method is to link to a so-called external style sheet. An external style sheet is simply a text file with the extension **.css**. Like any other file, you can place the style sheet on your web server or hard disk.
For example, let's say that your style sheet is named **style.css** and is located in a folder named **style**. The situation can be illustrated like this: Open Notepad and create two files - an HTML file and a CSS file - with the following contents:

default.htm

```
<html>
  <head>
        <title>My document</title>
        <link rel="stylesheet" type="text/css" href="style.css" />
  </head>
  <body>
        <h1>My first stylesheet</h1>
  </body>
</html>
```

style.css

```
body {
  background-color: #FF0000;
}
```

Now place the two files in the same folder. Remember to save the files with the right extensions (respectively ".htm" and ".css")

Open **default.htm** with your browser and see how the page has a red background. Congratulations, You have made your first style sheet!

CSS Comments

Comments are used to explain your code, and may help you when you edit the source code at a later date. Comments are ignored by browsers.
A CSS comment starts with /* and ends with */. Comments can also span multiple lines:

Example

```
/*This is a multiple
lines comment*/
p
{
color:red;
/*This is another comment*/
text-align:center;
}
```

Program

```
<head>
<style>

/*This is a multiple
lines comment*/

p
{
color:red;
/*This is another comment*/
text-align:center;
}
</style>
</head>

<body>
<p>Hello World!</p>
<p>This paragraph is styled with CSS.</p>
<p>CSS comments are not shown in the output.</p>
</body>
</html>
```

Result:

Hello World!

This paragraph is styled with CSS.

CSS comments are not shown in the output.

CSS Selectors

CSS selectors allow you to select and manipulate HTML element(s).
CSS selectors are used to "find" (or select) HTML elements based on their id, classes, types, attributes, values of attributes and much more.

The element Selector

The element selector selects elements based on the element name.
You can select all <p> elements on a page like this: (all <p> elements will be center-aligned, with a red text color)

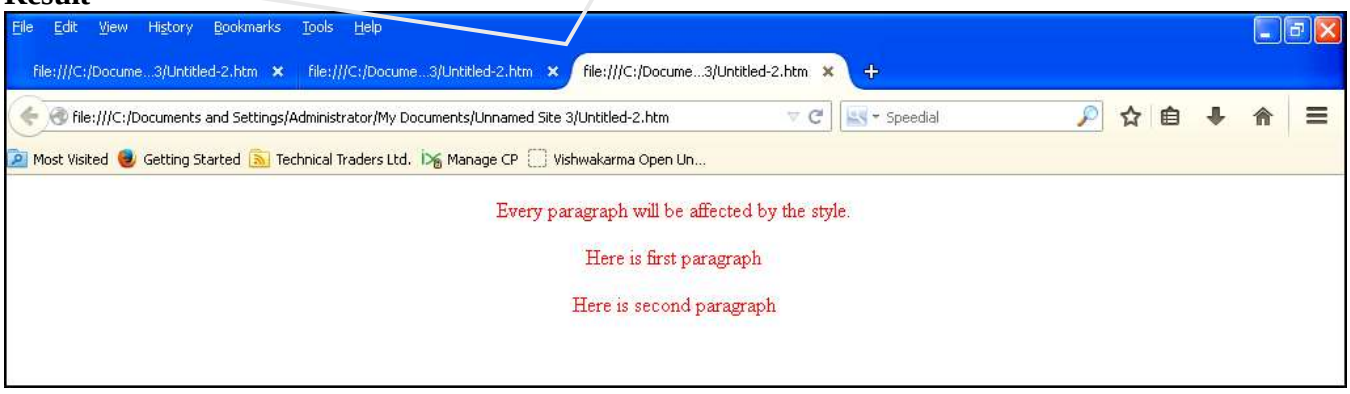
Example

```
p
{
text-align:center;
color:red;
}
```

Program

```
<!DOCTYPE html>
<html>
<head>
<style>
p
{
text-align:center;
color:red;
}
</style>
</head>

<body>
<p>Every paragraph will be affected by the style.</p>
<p id="para1">Here is first paragraph</p>
<p> Here is second paragraph</p>
</body>
</html>
```

Result



The id Selector

The id selector uses the id attribute of an HTML tag to find the specific element.
An id should be unique within a page, so you should use the id selector when you want to find a single, unique element.
To find an element with a specific id, write a hash character, followed by the id of the element.
The style rule below will be applied to the HTML element with id="para1":

Example

```
#para1
{
text-align:center;
color:red;
}
```

Program

```
<!DOCTYPE html>
<html>
<head>
<style>
#para1
{
text-align:center;
color:red;
}
</style>
</head>

<body>
<p id="para1">Hi Jatin How are YOU</p>
<p>This paragraph is not affected by the style, About me</p>
</body>
</html>
```

Result:

The class Selector

The class selector finds elements with the specific class.
The class selector uses the HTML class attribute.
To find elements with a specific class, write a period character, followed by the name of the class:
In the example below, all HTML elements with class="center" will be center-aligned:

Example

```
.center
{
text-align:center;
color:red;
}
```

You can also specify that only specific HTML elements should be affected by a class.
In the example below, all p elements with class="center" will be center-aligned:

Example

```
p.center
{
text-align:center;
color:red;
}
```

Note: Do **NOT** start a class name with a number!

Grouping Selectors

In style sheets there are often elements with the same style:

```
h1
{
text-align:center;
color:red;
}

h2
{
text-align:center;
color:red;
}
p
{
text-align:center;
color:red;
}
```

To minimize the code, you can group selectors. To group selectors, separate each selector with a comma.
In the example below we have grouped the selectors from the code above:

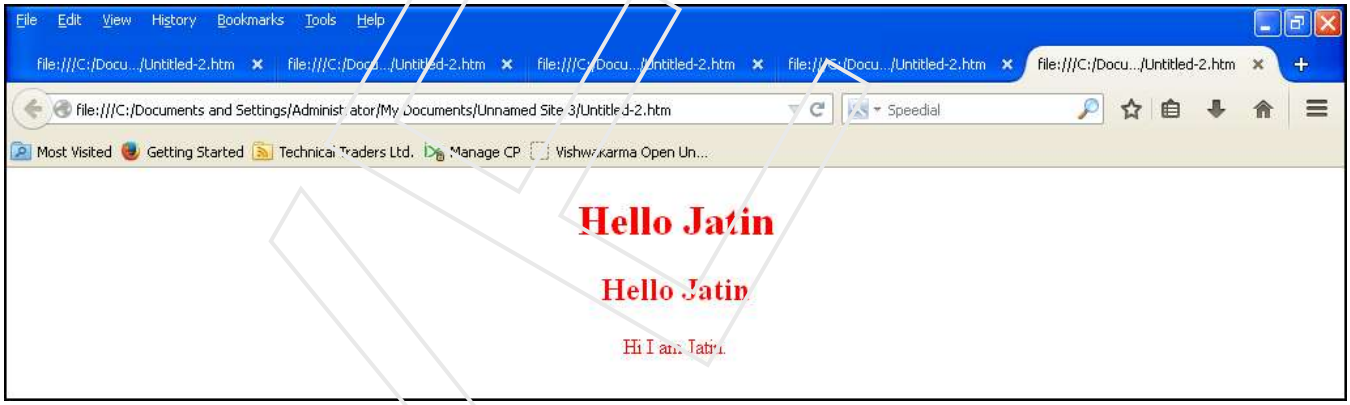
Example

```
h1,h2,p
{
text-align:center;
color:red;
}
```

program

```
<!DOCTYPE html>
<html>
<head>
<style>
h1,h2,p
{
text-align:center;
color:red;
}
</style>
</head>

<body>
<h1>Hello Jatin</h1>
<h2>Hello Jatin</h2>
<p>Hi I am Jatin.</p>
</body>
</html>
```

Result



CSS Background

CSS background properties are used to define the background effects of an element.
CSS properties used for background effects:

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

Foreground color: the 'color' property

The `color` property describes the foreground color of an element.

For example, imagine that we want all headlines in a document to be dark red. The headlines are all marked with the HTML element `<h1>`. The code below sets the color of `<h1>` elements to red.

```
h1 {
        color: #ff0000;
}
```

Result

This heading is in red

Colors can be entered as hexadecimal values as in the example above (#ff0000), or you can use the names of the colors ("red") or rgb-values (rgb(255,0,0)).

The 'background-color' property

The `background-color` property describes the background color of elements.
The element `<body>` contains all the content of an HTML document. Thus, to change the background color of an entire page, the background-color property should be applied to the `<body>` element.
You can also apply background colors to other elements including headlines and text. In the example below, different background colors are applied to `<body>` and `<h1>` elements.

```
body {
        background-color: #FFCC66;
}

h1 {
        color: #990000;
        background-color: #FC9804;
}
```

Notice that we applied two properties to `<h1>` by dividing them by a semicolon.

Result

Background Image

The background-image property specifies an image to use as the background of an element.
By default, the image is repeated so it covers the entire element.
The background image for a page can be set like this:

Example

body {background-image:url("image.gif");}

Repeat background image [background-repeat]

In the example above, did you notice that by default the image was repeated both horizontally and vertically to cover the entire screen? The property `background-repeat` controls this behaviour.

The table below outlines the four different values for `background-repeat`.

Value	Description
`background-repeat: repeat-x`	The image is repeated horizontally
`background-repeat: repeat-y`	The image is repeated vertically
`background-repeat: repeat`	The image is repeated both horizontally and vertically
`background-repeat: no-repeat`	The image is not repeated

For example, to avoid repetition of a background image the code should look like this:

```
body {
        background-color: #FFCC66;
        background-image: url("image.gif");
        background-repeat: no-repeat;
}

h1 {
        color: #990000;
        background-color: #FC9804;
}
```

Lock background image [background-attachment]

The property `background-attachment` specifies whether a background picture is fixed or scrolls along with the containing element.
A fixed background image will not move with the text when a reader is scrolling the page, whereas an unlocked background image will scroll along with the text of the web page.
The table below outlines the two different values for `background-attachment`. Click on the examples to see the difference between scroll and fixed.

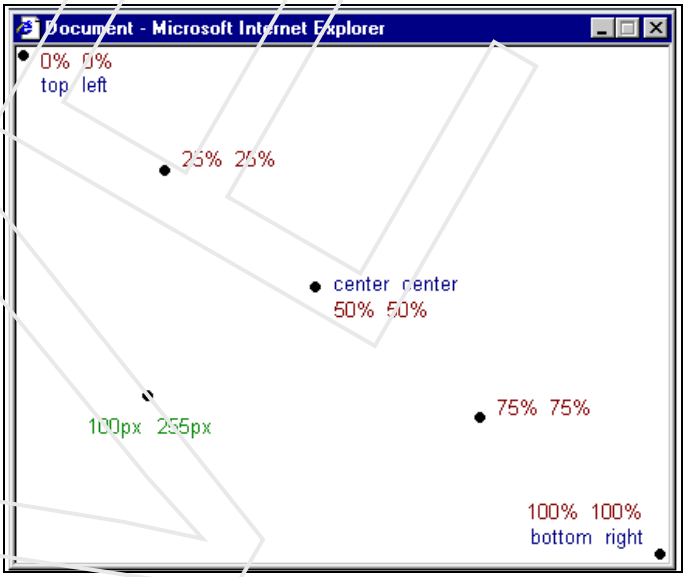
Value	Description
`Background-attachment: scroll`	The image scrolls with the page - unlocked
`Background-attachment: fixed`	The image is locked

For example, the code below will fix the background image.

```
body {
        background-color: #FFCC66;
        background-image: url("image.gif");
        background-repeat: no-repeat;
        background-attachment: fixed;
}

h1 {
        color: #990000;
        background-color: #FC9804;
}
```

Place background image [background-position]

By default, a background image will be positioned in the top left corner of the screen. The property `background-position` allows you to change this default and position the background image anywhere you like on the screen.
There are numerous ways to set the values of `background-position`. However, all of them are formatted as a set of coordinates. For example, the value '100px 200px' positions the background image 100px from the left side and 200px from the top of the browser window.
The coordinates can be indicated as percentages of the browser window, fixed units (pixels, centimetres, etc.) or you can use the words top, bottom, center, left and right. The model below illustrates the system:



Example:

The code example below positions the background image in the bottom right corner:

```
body {
        background-color: #FFCC66;
        background-image: url("image.gif");
        background-repeat: no-repeat;
        background-attachment: fixed;
        background-position: right bottom;
```

```
}

h1 {
        color: #990000;
        background-color: #FC9804;
}
```

Do it yourself

Compiling [background]

The property `background` is a short hand for all the background properties listed in this lesson.

With `background` you can compress several properties and thereby write your style sheet in a shorter way which makes it easier to read.

For example, look at these five lines:

```
background-color: #FFCC66;
background-image: url("image.gif");
background-repeat: no-repeat;
background-attachment: fixed;
background-position: right bottom;
```

Using `background` the same result can be achieved in just one line of code:

```
background: #FFCC66 url("image.gif") no-repeat fixed right
bottom;
```

The list of order is as follows:

```
[background-color]|[background-image]|[background-repeat]|
[background-attachment]|[background-position]
```

If a property is left out, it will automatically be set to its default value. For example, if `background-attachment` and `background-position` are taken out of the example:

```
background: #FFCC66 url("image.gif") no-repeat;
```

These two properties that are not specified would merely be set to their default values which as you know are scroll and top left.

All CSS Background Properties

Property	Description
background	Sets all the background properties in one declaration
background-attachment	Sets whether a background image is fixed or scrolls with the rest of the page
background-color	Sets the background color of an element
background-image	Sets the background image for an element
background-position	Sets the starting position of a background image
background-repeat	Sets how a background image will be repeated

CSS Text

Text Color

The color property is used to set the color of the text.
With CSS, a color is most often specified by:

- a HEX value - like "#ff0000"
- an RGB value - like "rgb(255,0,0)"
- a color name - like "red"

The default color for a page is defined in the body selector.

Example

```
body {color:blue;}
h1 {color:#00ff00;}
h2 {color:rgb(255,0,0);}
```

Note: For W3C compliant CSS: If you define the color property, you must also define the background-color property.

Text Alignment

The text-align property is used to set the horizontal alignment of a text. Text can be centered, or aligned to the left or right, or justified.

When text-align is set to "justify", each line is stretched so that every line has equal width, and the left and right margins are straight (like in magazines and newspapers).

Example

```
h1 {text-align:center;}
p.date {text-align:right;}
p.main {text-align:justify;}
```

Text Decoration

The text-decoration property is used to set or remove decorations from text.

The text-decoration property is mostly used to remove underlines from links for design purposes:

Example

```
a {text-decoration:none;}
```

It can also be used to decorate text:

Example

```
h1 {text-decoration:overline;}
h2 {text-decoration:line-through;}
h3 {text-decoration:underline;}
```

Text Transformation

The text-transform property is used to specify uppercase and lowercase letters in a text. It can be used to turn everything into uppercase or lowercase letters, or capitalize the first letter of each word.

Example

```
p.uppercase {text-transform:uppercase;}
p.lowercase {text-transform:lowercase;}
p.capitalize {text-transform:capitalize;}
```

Text Indentation

The text-indent property is used to specify the indentation of the first line of a text.

Example

```
p {text-indent:50px;}
```

All CSS Text Properties

Property	Description
color	Sets the color of text
direction	Specifies the text direction/writing direction
letter-spacing	Increases or decreases the space between characters in a text
line-height	Sets the line height
text-align	Specifies the horizontal alignment of text
text-decoration	Specifies the decoration added to text
text-indent	Specifies the indentation of the first line in a text-block
text-shadow	Specifies the shadow effect added to text
text-transform	Controls the capitalization of text
unicode-bidi	Used together with the direction property to set or return whether the text should be overridden to support multiple languages in the same document
vertical-align	Sets the vertical alignment of an element
white-space	Specifies how white-space inside an element is handled
word-spacing	Increases or decreases the space between words in a text

CSS Font Families

In CSS, there are two types of font family names:

- **generic family** - a group of font families with a similar look (like "Serif" or "Monospace")
- **font family** - a specific font family (like "Times New Roman" or "Arial")

Generic family	Font family	Description
Serif	Times New Roman Georgia	Serif fonts have small lines at the ends on some characters
Sans-serif	Arial Verdana	"Sans" means without - these fonts do not have the lines at the ends of characters
Monospace	Courier New Lucida Console	All monospace characters have the same width

Note: ***On computer screens, sans-serif fonts are considered easier to read than serif fonts.***

Difference Between Serif and Sans-serif Fonts

Font Family

The font family of a text is set with the font-family property.

The font-family property should hold several font names as a "fallback" system. If the browser does not support the first font, it tries the next font.

Start with the font you want, and end with a generic family, to let the browser pick a similar font in the generic family, if no other fonts are available.

Note: If the name of a font family is more than one word, it must be in quotation marks, like: "Times New Roman".

More than one font family is specified in a comma-separated list:

Example

```
p{font-family:"Times New Roman", Times, serif;}
```

Font Style

The font-style property is mostly used to specify italic text.

This property has three values:

- normal - The text is shown normally
- italic - The text is shown in italics
- oblique - The text is "leaning" (oblique is very similar to italic, but less supported)

Example

```
p.normal {font-style:normal;}
p.italic {font-style:italic;}
p.oblique {font-style:oblique;}
```

Font Size

The font-size property sets the size of the text. Being able to manage the text size is important in web design. However, you should not use font size adjustments to make paragraphs look like headings, or headings look like paragraphs. Always use the proper HTML tags, like <h1> - <h6> for headings and <p> for paragraphs. The font-size value can be an absolute, or relative size.

Absolute size:

- Sets the text to a specified size
- Does not allow a user to change the text size in all browsers (bad for accessibility reasons)
- Absolute size is useful when the physical size of the output is known

Relative size:

- Sets the size relative to surrounding elements
- Allows a user to change the text size in browsers

Note: If you do not specify a font size, the default size for normal text, like paragraphs, is 16px (16px=1em).

Set Font Size With Pixels

Setting the text size with pixels gives you full control over the text size:

Example

```
h1 {font-size:40px;}
h2 {font-size:30px;}
p {font-size:14px;}
```

The text can be resized in all browsers using the zoom tool (however, this resizes the entire page, not just the text).

Set Font Size With Em

To avoid the resizing problem with older versions of Internet Explorer, many developers use em instead of pixels.

The em size unit is recommended by the W3C. 1em is equal to the current font size. The default text size in browsers is 16px. So, the default size of 1em is 16px. The size can be calculated from pixels to em using this formula: *pixels*/16=*em*

Example

```
h1 {font-size:2.5em;} /* 40px/16=2.5em */
h2 {font-size:1.875em;} /* 30px/16=1.875em */
p {font-size:0.875em;} /* 14px/16=0.875em */
```

In the example above, the text size in em is the same as the previous example in pixels. However, with the em size, it is possible to adjust the text size in all browsers.

Unfortunately, there is still a problem with older versions of IE. The text becomes larger than it should when made larger, and smaller than it should when made smaller.

Use a Combination of Percent and Em

The solution that works in all browsers, is to set a default font-size in percent for the <body> element:

Example

```
body {font-size:100%;}
h1 {font-size:2.5em;}
h2 {font-size:1.875em;}
p {font-size:0.875em;}
```

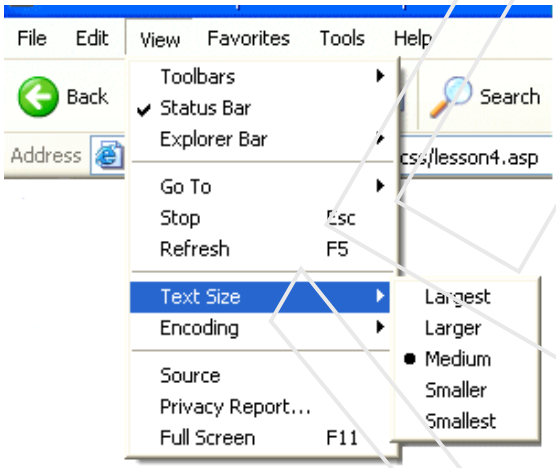
Program

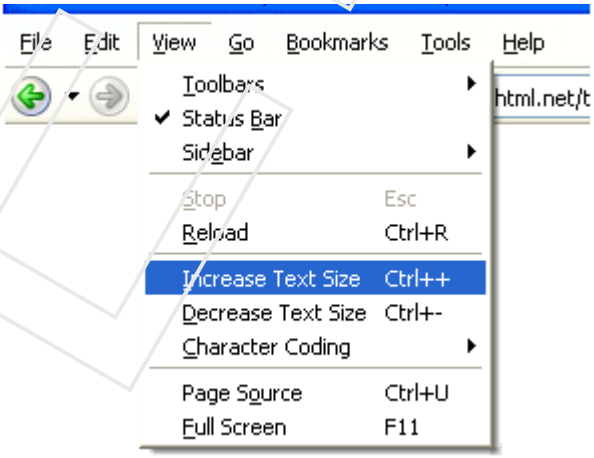
```
<!DOCTYPE html>
<html>
<head>
<style>
body {font-size:100%;}
h1 {font-size:2.5em;}
h2 {font-size:1.875em;}
p {font-size:0.875em;}
</style>
</head>
<body>

<h1>This is heading 1 Hi Jatin</h1>
<h2>This is heading 2 Hi Jatin</h2>
<p>This is a paragraph. Hi Jatin</p>
<p>Specifying the font-size in percent and em displays the same size in all
major browsers, and allows all browsers to resize the text!</p>

</body>
</html>
```

Result

This code shows the same text size in all browsers, and allows all browsers to zoom or resize the text!

Below you can see an illustration of how to adjust the text size in Mozilla Firefox and Internet Explorer. Try it yourself - neat feature, don't you think?





All CSS Font Properties

Property	Description
font	Sets all the font properties in one declaration
font-family	Specifies the font family for text
font-size	Specifies the font size of text
font-style	Specifies the font style for text
font-variant	Specifies whether or not a text should be displayed in a small-caps font
font-weight	Specifies the weight of a font

CSS Links

Links can be styled in different ways.

Styling Links

Links can be styled with any CSS property (e.g. color, font-family, background, etc.). In addition, links can be styled differently depending on what **state** they are in.

The four links states are:

- a:link - a normal, unvisited link
- a:visited - a link the user has visited
- a:hover - a link when the user mouses over it
- a:active - a link the moment it is clicked

Example

```
a:link {color:#FF0000;}      /* unvisited link */
a:visited {color:#00FF00;}  /* visited link */
a:hover {color:#FF00FF;}  /* mouse over link */
a:active {color:#0000FF;}  /* selected link */
```

When setting the style for several link states, there are some order rules:

- a:hover MUST come after a:link and a:visited
- a:active MUST come after a:hover

Common Link Styles

In the example above the link changes color depending on what state it is in.

Lets go through some of the other common ways to style links:

Text Decoration

The text-decoration property is mostly used to remove underlines from links:

Example

```
a:link {text-decoration:none;}
a:visited {text-decoration:none;}
a:hover {text-decoration:underline;}
a:active {text-decoration:underline;}
```

Background Color

The background-color property specifies the background color for links:

Example

```
a:link {background-color:#B2FF99;}
a:visited {background-color:#FFFF85;}
a:hover {background-color:#FF704D;}
a:active {background-color:#FF704D;}
```

Note: a:hover MUST come after a:link and a:visited in the CSS definition in order to be effective.

Note: a:active MUST come after a:hover in the CSS definition in order to be effective.

CSS Lists

The CSS list properties allow you to:

- Set different list item markers for ordered lists
- Set different list item markers for unordered lists
- Set an image as the list item marker

List

In HTML, there are two types of lists:

- unordered lists - the list items are marked with bullets
- ordered lists - the list items are marked with numbers or letters

With CSS, lists can be styled further, and images can be used as the list item marker.

Different List Item Markers

The type of list item marker is specified with the list-style-type property:

Example

```
ul.a {list-style-type: circle;}
ul.b {list-style-type: square;}

ol.c {list-style-type: upper-roman;}
ol.d {list-style-type: lower-alpha;}
```

Some of the values are for unordered lists, and some for ordered lists.

An Image as The List Item Marker

To specify an image as the list item marker, use the list-style-image property:

Example

```
ul
{
list-style-image: url('sqpurple.gif');
}
```

The example above does not display equally in all browsers. IE and Opera will display the image-marker a little bit higher than Firefox, Chrome, and Safari.

If you want the image-marker to be placed equally in all browsers, a crossbrowser solution is explained below.

Crossbrowser Solution

The following example displays the image-marker equally in all browsers:

Example

```
ul
{
list-style-type: none;
padding: 0px;
margin: 0px;
}
ul li
{
background-image: url(sqpurple.gif);
background-repeat: no-repeat;
background-position: 0px 5px;
padding-left: 14px;
}
```

Example explained:

- For ul:
  - Set the list-style-type to none to remove the list item marker
  - Set both padding and margin to 0px (for cross-browser compatibility)
- For all li in ul:
  - Set the URL of the image, and show it only once (no-repeat)
  - Position the image where you want it (left 0px and down 5px)
  - Position the text in the list with padding-left

List - Shorthand property

It is also possible to specify all the list properties in one, single property. This is called a shorthand property.

The shorthand property used for lists, is the list-style property:

Example

```
ul
{
list-style: square url("sqpurple.gif");
}
```

When using the shorthand property, the order of the values are:

- list-style-type
- list-style-position (for a description, see the CSS properties table below)
- list-style-image

It does not matter if one of the values above are missing, as long as the rest are in the specified order.

All CSS List Properties

Property	Description
list-style	Sets all the properties for a list in one declaration
list-style-image	Specifies an image as the list-item marker
list-style-position	Specifies if the list-item markers should appear inside or outside the content flow
list-style-type	Specifies the type of list-item marker

CSS Tables

The look of an HTML table can be greatly improved with CSS:

Table Borders

To specify table borders in CSS, use the border property.
The example below specifies a black border for table, th, and td elements:

Example

```
table, th, td
{
border: 1px solid black;
}
```

Notice that the table in the example above has double borders. This is because both the table and the th/td elements have separate borders.
To display a single border for the table, use the border-collapse property.

Collapse Borders

The border-collapse property sets whether the table borders are collapsed into a single border or separated:

Example

```
table
{
border-collapse:collapse;
}
table, th, td
{
border: 1px solid black;
}
```

Table Width and Height

Width and height of a table is defined by the width and height properties.
The example below sets the width of the table to 100%, and the height of the th elements to 50px:

Example

```
table
{
width:100%;
}
th
{
height:50px;
}
```

Table Text Alignment

The text in a table is aligned with the text-align and vertical-align properties.
The text-align property sets the horizontal alignment, like left, right, or center:

Example

```
td
{
text-align:right;
}
```

The vertical-align property sets the vertical alignment, like top, bottom, or middle:

Example

```
td
{
height:50px;
vertical-align:bottom;
}
```

Table Padding

To control the space between the border and content in a table, use the padding property on td and th elements:

Example

```
td
{
padding:15px;
}
```

Table Color

The example below specifies the color of the borders, and the text and background color of th elements:

Example

```
table, td, th
{
border:1px solid green;
}
th
{
background-color:green;
color:white;
}
```

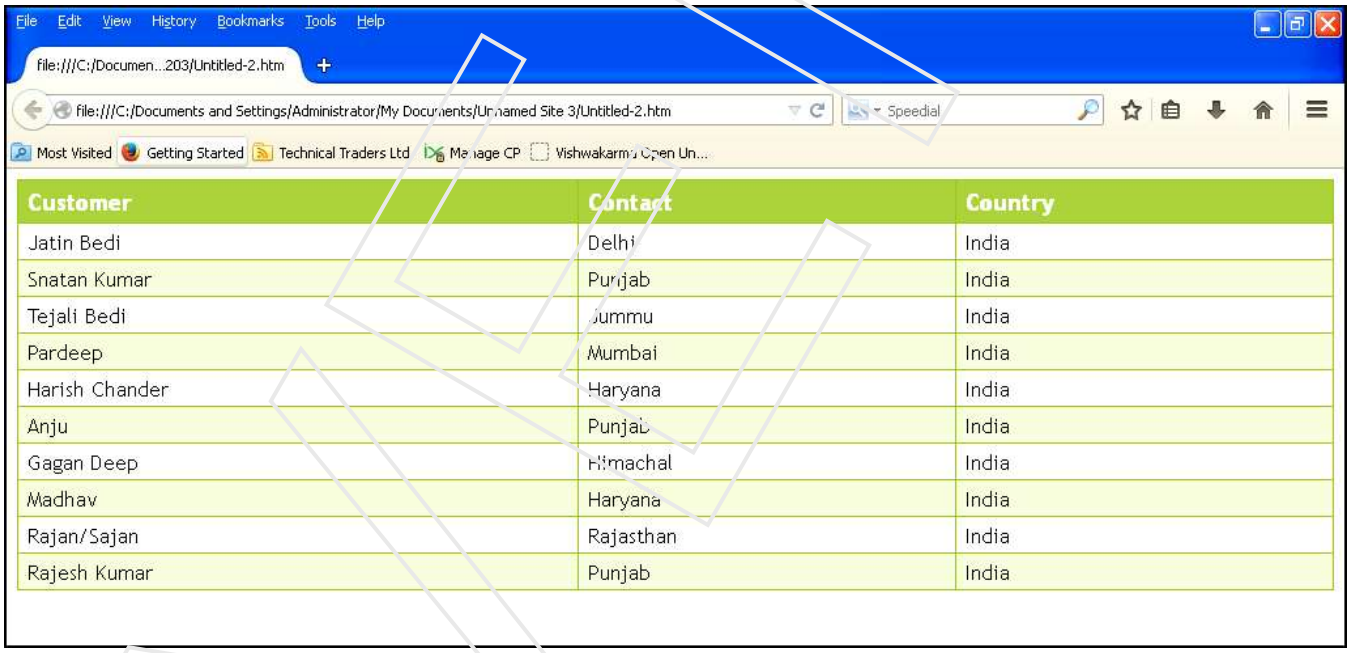
Example

```
<!DOCTYPE html>
<html>
<head>
<style>
#customers
{
font-family:"Trebuchet MS", Arial, Helvetica, sans-serif;
width:100%;
border-collapse:collapse;
}
#customers td, #customers th
{
font-size:1em;
border:1px solid #98bf21;
padding:3px 7px 2px 7px;
}
#customers th
{
font-size:1.1em;
text-align:left;
padding-top:5px;
padding-bottom:4px;
background-color:#A7C942;
color:#ffffff;
}
#customers tr.alt td
```

```
{
color:#000000;
background-color:#EAF2D3;
}
</style>
</head>

<body>
<table id="customers">
<tr>
  <th>Customer</th>
  <th>Contact</th>
  <th>Country</th>
</tr>
<tr>
<td>Jatin Bedi</td>
<td>Delhi</td>
<td>India</td>
</tr>
<tr class="alt">
<td>Snatan Kumar</td>
<td>Punjab</td>
<td>India</td>
</tr>
<tr>
<td>Tejali Bedi</td>
<td>Jummu</td>
<td>India</td>
</tr>
<tr class="alt">
<td>Pardeep</td>
<td>Mumbai</td>
<td>India</td>
</tr>
<tr>
<td>Harish Chander</td>
<td>Haryana</td>
<td>India</td>
</tr>
<tr class="alt">
<td>Anju</td>
<td>Punjab</td>
<td>India</td>
</tr>
<tr>
<td>Gagan Deep</td>
<td>Himachal</td>
<td>India</td>
</tr>
<tr class="alt">
<td>Madhav</td>
```

```
<td>Haryana</td>
<td>India</td>
</tr>
<tr>
<td>Rajan/Sajan</td>
<td>Rajasthan</td>
<td>India</td>
</tr>
<tr class="alt">
<td>Rajesh Kumar</td>
<td>Punjab</td>
<td>India</td>
</tr>
</table>
</body>
</html>
```

Result



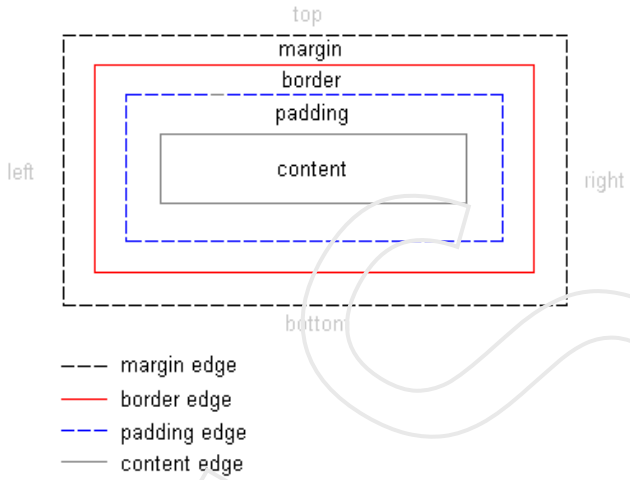
The CSS Box Model

All HTML elements can be considered as boxes. In CSS, the term "box model" is used when talking about design and layout.

The CSS box model is essentially a box that wraps around HTML elements, and it consists of: margins, borders, padding, and the actual content.

The box model allows us to place a border around elements and space elements in relation to other elements.

The box model image:



Explanation of the different parts:

- **Margin** - Clears an area around the border. The margin does not have a background color, it is completely transparent
- **Border** - A border that goes around the padding and content. The border is inherited from the color property of the box
- **Padding** - Clears an area around the content. The padding is affected by the background color of the box
- **Content** - The content of the box, where text and images appear

In order to set the width and height of an element correctly in all browsers, you need to know how the box model works.

Width and Height of an Element

When you set the width and height properties of an element with CSS, you just set the width and height of the **content area**. To calculate the full size of an element, you must also add the padding, borders and margins.

The total width of the element in the example below is 300px:

```
width:250px;
padding:10px;
border:5px solid gray;
margin:10px;
```

Let's do the math:
250px (width)
+ 20px (left + right padding)
+ 10px (left + right border)
+ 20px (left + right margin)
= 300px

Assume that you had only 250px of space. Let's make an element with a total width of 250px:

Example

width:220px;
padding:10px;
border:5px solid gray;
margin:0px;
The total width of an element should be calculated like this:
Total element width = width + left padding + right padding + left border + right border + left margin + right margin
The total height of an element should be calculated like this:
Total element height = height + top padding + bottom padding + top border + bottom border + top margin + bottom margin

Browsers Compatibility Issue

IE8 and earlier versions of IE, included padding and border in the width property.
To fix this problem, add a <!DOCTYPE html> to the HTML page.

CSS Border Properties

The CSS border properties allow you to specify the style and color of an element's border.

Border Style

The border-style property specifies what kind of border to display.
Note: None of the border properties will have ANY effect unless the **border-style** property is set

Border-style values:

none: Defines no border

dotted: Defines a dotted border

dashed: Defines a dashed border

solid: Defines a solid border

double: Defines two borders. The width of the two borders are the same as the border-width value

groove: Defines a 3D grooved border. The effect depends on the border-color value

ridge: Defines a 3D ridged border. The effect depends on the border-color value

inset: Defines a 3D inset border. The effect depends on the border-color value

outset: Defines a 3D outset border. The effect depends on the border-color value

Border Width

The border-width property is used to set the width of the border.

The width is set in pixels, or by using one of the three pre-defined values: thin, medium, or thick.

Note: The "border-width" property does not work if it is used alone. Use the "border-style" property to set the borders first.

Example

```
p.one
{
border-style:solid;
border-width:5px;
}
p.two
{
border-style:solid;
border-width:medium;
}
```

Border Color

The border-color property is used to set the color of the border. The color can be set by:
name - specify a color name, like "red"
RGB - specify a RGB value, like "rgb(255,0,0)"
Hex - specify a hex value, like "#ff0000"
You can also set the border color to "transparent".
Note: The "border-color" property does not work if it is used alone. Use the "border-style" property to set the borders first.

Example

```
p.one
{
border-style:solid;
border-color:red;
}
p.two
{
border-style:solid;
border-color:#98bf21;
}
```

Border - Individual sides

In CSS it is possible to specify different borders for different sides:

Example

```
p
{
border-top-style:dotted;
border-right-style:solid;
border-bottom-style:dotted;
border-left-style:solid;
}
```

The example above can also be set with a single property:

Example

```
border-style:dotted solid;
```

The border-style property can have from one to four values.

- **border-style:dotted solid double dashed;**
  - top border is dotted
  - right border is solid
  - bottom border is double
  - left border is dashed
- **border-style:dotted solid double;**
  - top border is dotted
  - right and left borders are solid
  - bottom border is double
- **border-style:dotted solid;**
  - top and bottom borders are dotted
  - right and left borders are solid
- **border-style:dotted;**
  - all four borders are dotted

The border-style property is used in the example above. However, it also works with border-width and border-color.

Border - Shorthand property

As you can see from the examples above, there are many properties to consider when dealing with borders.

To shorten the code, it is also possible to specify all the individual border properties in one property. This is called a shorthand property.

The border property is a shorthand for the following individual border properties:

- border-width
- border-style (required)
- border-color

Example

```
border:5px solid red;
```

All CSS Border Properties

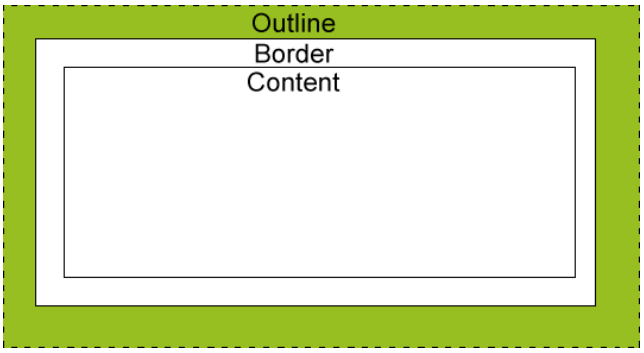
Property	Description
border	Sets all the border properties in one declaration
border-bottom	Sets all the bottom border properties in one declaration
border-bottom-color	Sets the color of the bottom border
border-bottom-style	Sets the style of the bottom border
border-bottom-width	Sets the width of the bottom border
border-color	Sets the color of the four borders
border-left	Sets all the left border properties in one declaration
border-left-color	Sets the color of the left border
border-left-style	Sets the style of the left border
border-left-width	Sets the width of the left border
border-right	Sets all the right border properties in one declaration
border-right-color	Sets the color of the right border
border-right-style	Sets the style of the right border
border-right-width	Sets the width of the right border
border-style	Sets the style of the four borders
border-top	Sets all the top border properties in one declaration
border-top-color	Sets the color of the top border
border-top-style	Sets the style of the top border
border-top-width	Sets the width of the top border
border-width	Sets the width of the four borders

CSS Outlines

An outline is a line that is drawn around elements (outside the borders) to make the element "stand out". The outline properties specify the style, color, and width of an outline.

CSS Outline

An outline is a line that is drawn around elements (outside the borders) to make the element "stand out". However, the outline property is different from the border property.
The outline is not a part of an element's dimensions; the element's total width and height is not affected by the width of the outline.



All CSS Outline Properties

Property	Description	Values
outline	Sets all the outline properties in one declaration	*outline-color* *outline-style* *outline-width* inherit
outline-color	Sets the color of an outline	*color_name* *hex_number* *rgb_number* invert inherit
outline-style	Sets the style of an outline	None dotted dashed solid double groove ridge inset outset inherit
outline-width	Sets the width of an outline	Thin medium thick *length* inherit

CSS Margin

The CSS margin properties define the space around elements.

Margin

The margin clears an area around an element (outside the border). The margin does not have a background color, and is completely transparent.
The top, right, bottom, and left margin can be changed independently using separate properties. A shorthand margin property can also be used, to change all margins at once.

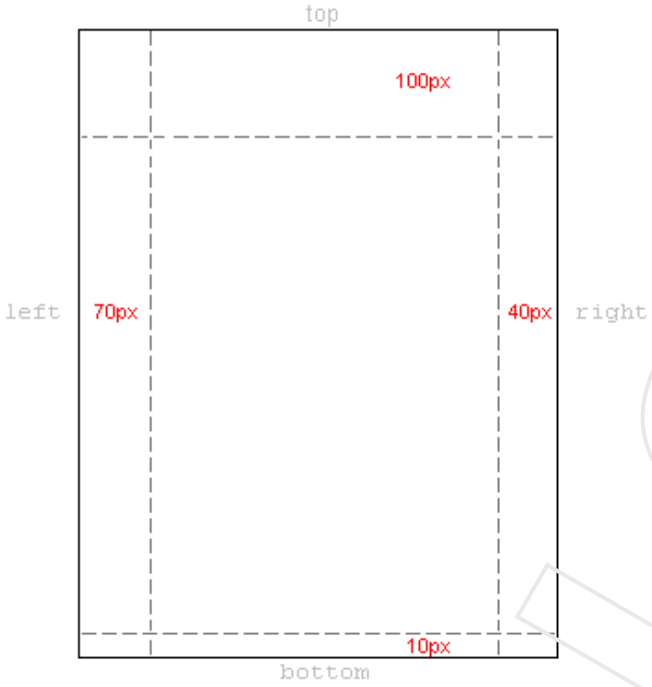
Possible Values

Value	Description
auto	The browser calculates a margin
length	Specifies a margin in px, pt, cm, etc. Default value is 0px
%	Specifies a margin in percent of the width of the containing element
inherit	Specifies that the margin should be inherited from the parent element

Note: It is also possible to use negative values, to overlap content.

Set the margin in an element

An element has four sides: right, left, top and bottom. The `margin` is the distance from each side to the neighboring element (or the borders of the document). The illustration below shows how we want the margins in our pages to be.



The CSS code for this would look as follow:

```
body {
        margin-top: 100px;
        margin-right: 40px;
        margin-bottom: 10px;
        margin-left: 70px;
}
```

Or you could choose a more elegant compilation:

```
body {
        margin: 100px 40px 10px 70px;
}
```

You can set the margins in the same way on almost every element. For example, we can choose to define margins for all of our text paragraphs marked with <p>:

```
body {
        margin: 100px 40px 10px 70px;
}

p {
        margin: 5px 50px 5px 50px;
}
```

Margin - Individual sides

In CSS, it is possible to specify different margins for different sides:

Example

```
margin-top:100px;
margin-bottom:100px;
margin-right:50px;
margin-left:50px;
```

Margin - Shorthand property

To shorten the code, it is possible to specify all the margin properties in one property. This is called a shorthand property.

The shorthand property for all the margin properties is "margin":

Example

```
margin:100px 50px;
```

The margin property can have from one to four values.

- **margin:25px 50px 75px 100px;**
  - top margin is 25px
  - right margin is 50px
  - bottom margin is 75px
  - left margin is 100px
- **margin:25px 50px 75px;**
  - top margin is 25px
  - right and left margins are 50px
  - bottom margin is 75px
- **margin:25px 50px;**
  - top and bottom margins are 25px
  - right and left margins are 50px
- **margin:25px;**
  - all four margins are 25px

All CSS Margin Properties

Property	Description
margin	A shorthand property for setting the margin properties in one declaration
margin-bottom	Sets the bottom margin of an element
margin-left	Sets the left margin of an element
margin-right	Sets the right margin of an element
margin-top	Sets the top margin of an element

CSS Padding

The CSS padding properties define the space between the element border and the element content.

Padding

The padding clears an area around the content (inside the border) of an element. The padding is affected by the background color of the element.

The top, right, bottom, and left padding can be changed independently using separate properties. A shorthand padding property can also be used, to change all paddings at once.

Possible Values

Value	Description
length	Defines a fixed padding (in pixels, pt, em, etc.)
%	Defines a padding in % of the containing element

Set padding in an element

Padding can also be understood as "filling". This makes sense as padding does not affect the distance of the element to other elements but only defines the inner distance between the border and the content of the element.
The usage of padding can be illustrated by looking at a simple example where all headlines have background colors:

```
h1 {
      background: yellow;
}

h2 {
      background: orange;
}
```

By defining padding for the headlines, you change how much filling there will be around the text in each headline:

```
h1 {
      background: yellow;
      padding: 20px 20px 20px 80px;
}

h2 {
      background: orange;
      padding-left:120px;
}
```

Padding - Individual sides

In CSS, it is possible to specify different padding for different sides:

Example

```
padding-top:25px;
padding-bottom:25px;
padding-right:50px;
padding-left:50px;
```

Padding - Shorthand property

To shorten the code, it is possible to specify all the padding properties in one property. This is called a shorthand property.

The shorthand property for all the padding properties is "padding":

Example

```
padding:25px 50px;
```

The padding property can have from one to four values.

- **padding:25px 50px 75px 100px;**
  - top padding is 25px
  - right padding is 50px
  - bottom padding is 75px
  - left padding is 100px
- **padding:25px 50px 75px;**
  - top padding is 25px
  - right and left paddings are 50px
  - bottom padding is 75px
- **padding:25px 50px;**
  - top and bottom paddings are 25px
  - right and left paddings are 50px
- **padding:25px;**
  - all four paddings are 25px

All CSS Padding Properties

Property	Description
padding	A shorthand property for setting all the padding properties in one declaration
padding-bottom	Sets the bottom padding of an element
padding-left	Sets the left padding of an element
padding-right	Sets the right padding of an element
padding-top	Sets the top padding of an element

CSS Dimension

The CSS dimension properties allow you to control the height and width of an element.

All CSS Dimension Properties

Property	Description	Values
height	Sets the height of an element	Auto *length* % inherit
max-height	Sets the maximum height of an element	None *length* % inherit
max-width	Sets the maximum width of an element	None *length* % inherit
min-height	Sets the minimum height of an element	*Length* % inherit
min-width	Sets the minimum width of an element	*Length* % inherit
width	Sets the width of an element	Auto *length* % inherit

CSS Display and Visibility

The display property specifies if/how an element is displayed, and the visibility property specifies if an element should be visible or hidden.

Hiding an Element - display:none or visibility:hidden

Hiding an element can be done by setting the display property to "none" or the visibility property to "hidden". However, notice that these two methods produce different results:

visibility:hidden hides an element, but it will still take up the same space as before. The element will be hidden, but still affect the layout.

Example

```
h1.hidden {visibility:hidden;}
```

display:none hides an element, and it will not take up any space. The element will be hidden, and the page will be displayed as if the element is not there:

Example

```
h1.hidden {display:none;}
```

CSS Display - Block and Inline Elements

A block element is an element that takes up the full width available, and has a line break before and after it.

Examples of block elements:

- <h1>
- <p>
-

An inline element only takes up as much width as necessary, and does not force line breaks.

Examples of inline elements:

-
- <a>

Changing How an Element is Displayed

Changing an inline element to a block element, or vice versa, can be useful for making the page look a specific way, and still follow web standards.

The following example displays list items as inline elements:

Example

```
li {display:inline;}
```

The following example displays span elements as block elements:

Example

```
span {display:block;}
```

Note: Setting the display property of an element only changes **how the element is displayed**, NOT what kind of element it is. So, an inline element with display:block is not allowed to have other block elements inside of it.

Positioning

The CSS positioning properties allow you to position an element. It can also place an element behind another, and specify what should happen when an element's content is too big.
Elements can be positioned using the top, bottom, left, and right properties. However, these properties will not work unless the position property is set first. They also work differently depending on the positioning method.

There are four different positioning methods.

Static Positioning

HTML elements are positioned static by default. A static positioned element is always positioned according to the normal flow of the page.

Static positioned elements are not affected by the top, bottom, left, and right properties.

Fixed Positioning

An element with fixed position is positioned relative to the browser window. It will not move even if the window is scrolled:

Example

```
p.pos_fixed
{
position:fixed;
top:30px;
right:5px;
}
```

Fixed positioned elements are removed from the normal flow. The document and other elements behave like the fixed positioned element does not exist. Fixed positioned elements can overlap other elements.

Relative Positioning

A relative positioned element is positioned relative to its normal position.

Example

```
h2.pos_left
{
position:relative;
left:-20px;
}
h2.pos_right
{
position:relative;
left:20px;
}
```

The content of relatively positioned elements can be moved and overlap other elements, but the reserved space for the element is still preserved in the normal flow.

Example

```
h2.pos_top
{
position:relative;
top:-50px;
}
```

Relatively positioned elements are often used as container blocks for absolutely positioned elements.

Absolute Positioning

An absolute position element is positioned relative to the first parent element that has a position other than static. If no such element is found, the containing block is <html>:

Example

```
h2
{
position:absolute;
left:100px;
top:150px;
}
```

Absolutely positioned elements are removed from the normal flow. The document and other elements behave like the absolutely positioned element does not exist.
Absolutely positioned elements can overlap other elements.

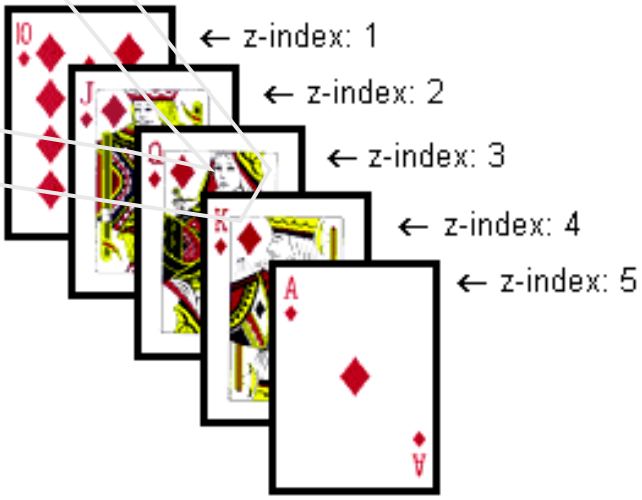
Overlapping Elements

When elements are positioned outside the normal flow, they can overlap other elements.
The z-index property specifies the stack order of an element (which element should be placed in front of, or behind, the others).

Layer on layer with z-index (Layers)

CSS operates in three dimensions - height, width and depth. We have seen the first two dimensions in previous lessons. In this lesson, we will learn how to let different elements become layers. In short, this means the order of which the elements overlap one another. For that purpose, you can assign each element a number (`z-index`). The system is that an element with a higher number overlaps an element with a lower number.

Let us say we are playing poker and have a royal flush. Our hand can be presented in a way where each card has got a `z-index`:



Note: An element can have a positive or negative stack order:

Example

```
img
{
position:absolute;
left:0px;
top:0px;
z-index:-1;
}
```

An element with greater stack order is always in front of an element with a lower stack order.

Note: If two positioned elements overlap without a z-index specified, the element positioned last in the HTML code will be shown on top.

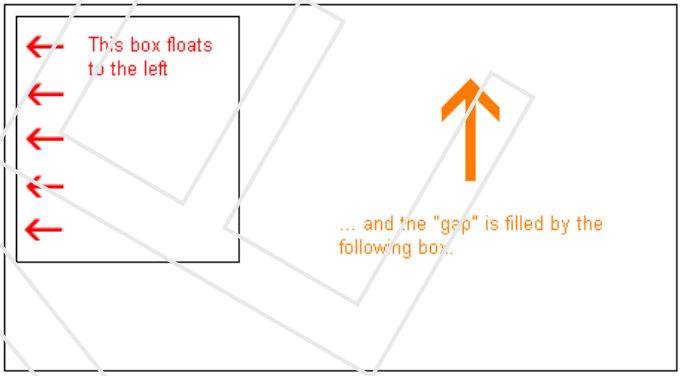
All CSS Positioning Properties

Property	Description	Values
bottom	Sets the bottom margin edge for a positioned box	auto *length* % inherit
clip	Clips an absolutely positioned element	*shape* auto inherit
cursor	Specifies the type of cursor to be displayed	*url* auto crosshair default pointer move e-resize ne-resize nw-resize n-resize se-resize sw-resize s-resize w-resize text wait help
left	Sets the left margin edge for a positioned box	auto *length* % inherit
overflow	Specifies what happens if content overflows an element's box	auto hidden scroll visible inherit

position	Specifies the type of positioning for an element	absolute fixed relative static inherit
right	Sets the right margin edge for a positioned box	auto *length* % inherit
top	Sets the top margin edge for a positioned box	auto *length* % inherit
z-index	Sets the stack order of an element	*number* auto inherit

Floating elements (floats)

An element can be floated to the right or to left by using the property `float`. That is to say that the box with its contents either floats to the right or to the left in a document (or the containing box). The principle of FLOAT is described below:



What is CSS Float?

With CSS float, an element can be pushed to the left or right, allowing other elements to wrap around it. Float is very often used for images, but it is also useful when working with layouts.

How Elements Float

Elements are floated horizontally, this means that an element can only be floated left or right, not up or down.
A floated element will move as far to the left or right as it can. Usually this means all the way to the left or right of the containing element.
The elements after the floating element will flow around it.
The elements before the floating element will not be affected.
If an image is floated to the right, a following text flows around it, to the left:

Example

```
img
{
float:right;
}
```

Floating Elements Next to Each Other

If you place several floating elements after each other, they will float next to each other if there is room.
Here we have made an image gallery using the float property:

Example

```
.thumbnail
{
float:left;
width:110px;
height:90px;
margin:5px;
}
```

Turning off Float - Using Clear

Elements after the floating element will flow around it. To avoid this, use the clear property.
The clear property specifies which sides of an element other floating elements are not allowed.
Add a text line into the image gallery, using the clear property:

Example

```
.text_line
{
clear:both;
}
```

All CSS Float Properties

Property	Description	Values
clear	Specifies which sides of an element where other floating elements are not allowed	left right both none inherit
float	Specifies whether or not a box should float	left right none inherit

CSS Horizontal Align

Aligning Block Elements

A block element is an element that takes up the full width available, and has a line break before and after it.
Examples of block elements:
<h1>
<p>

Center Aligning Using the margin Property

Block elements can be center-aligned by setting the left and right margins to "auto".

Note: Using margin:auto; will not work in IE8 and earlier, **unless a !DOCTYPE is declared.**

Setting the left and right margins to auto specifies that they should split the available margin equally. The result is a centered element:

Example

```
.center
{
margin-left:auto;
margin-right:auto;
width:70%;
background-color:#b0e0e6;
}
```

Tip: Center-aligning has no effect if the width is 100%.

Left and Right Aligning Using the position Property

One method of aligning elements is to use absolute positioning:

Example

```
.right
{
position:absolute;
right:0px;
width:300px;
background-color:#b0e0e6;
}
```

Note: Absolute positioned elements are removed from the normal flow, and can overlap elements.

Crossbrowser Compatibility Issues

When aligning elements like this, it is always a good idea to predefine margin and padding for the <body> element. This is to avoid visual differences in different browsers.
There is a problem with IE8 and earlier, when using the position property. If a container element (in our case) has a specified width, and the !DOCTYPE declaration is missing, IE8 and earlier versions will add a 17px margin on the right side. This seems to be space reserved for a scrollbar. Always set the !DOCTYPE declaration when using the position property:

Example

```
body
{
margin:0;
padding:0;
}
.container
{
position:relative;
width:100%;
}
.right
{
position:absolute;
right:20px;
width:300px;
background-color:#b0e0e6;
}
```

Left and Right Aligning Using the float Property

One method of aligning elements is to use the float property:

Example

```
.right
{
float:right;
width:300px;
background-color:#b0e0e6;
}
```

Cross Browser Compatibility Issues

When aligning elements like this, it is always a good idea to predefine margin and padding for the <body> element. This is to avoid visual differences in different browsers.

Example

```
body
{
margin:0;
padding:0;
```

```
}
.right
{
float:right;
width:300px;
background-color:#b0e0e6;
}
```

CSS Combinators

A combinator is something that explains the relationship between the selectors. A CSS selector can contain more than one simple selector. Between the simple selectors, we can include a combinator.

There are four different combinators in CSS3:

- descendant selector
- child selector
- adjacent sibling selector
- general sibling selector

Descendant Selector

The descendant selector matches all element that are descendants of a specified element.

The following example selects all <p> elements inside elements:

Example

```
div p
{
background-color:yellow;
}
```

Child Selector

The child selector selects all elements that are the immediate children of a specified element.

The following example selects all <p> elements that are immediate children of a element:

Example

```
div>p
{
background-color:yellow;
}
```

Adjacent Sibling Selector

The adjacent sibling selector selects all elements that are the adjacent siblings of a specified element. Sibling elements must have the same parent element, and "adjacent" means "immediately following".
The following example selects all <p> elements that are placed immediately after elements:

Example

```
div+p
{
background-color:yellow;
}
```

General Sibling Selector

The general sibling selector selects all elements that are siblings of a specified element.

The following example selects all <p> elements that are siblings of elements:

Example

```
div~p
{
background-color:yellow;
}
```

CSS Pseudo-classes

CSS pseudo-classes are used to add special effects to some selectors.

Syntax

The syntax of pseudo-classes:

selector:pseudo-class {property:value;}

CSS classes can also be used with pseudo-classes:

selector.class:pseudo-class {property:value;}

Anchor Pseudo-classes

Links can be displayed in different ways in a CSS-supporting browser:

Example

```
a:link {color:#FF0000;}      /* unvisited link */
a:visited {color:#00FF00;}   /* visited link */
a:hover {color:#FF00FF;}     /* mouse over link */
a:active {color:#0000FF;}    /* selected link */
```

Note: a:hover MUST come after a:link and a:visited in the CSS definition in order to be effective!!
a:active MUST come after a:hover in the CSS definition in order to be effective!!
Pseudo-class names are not case-sensitive.

Pseudo-classes and CSS Classes

Pseudo-classes can be combined with CSS classes:

a.red:visited {color:#FF0000;}

CSS Syntax

If the link in the example above has been visited, it will be displayed in red.

CSS - The :first-child Pseudo-class

The :first-child pseudo-class matches a specified element that is the first child of another element.

Note: For :first-child to work in IE8 and earlier, a <!DOCTYPE> must be declared

Match the first <p> element

In the following example, the selector matches any <p> element that is the first child of any element:

Example

```
<html>
<head>
<style>
p:first-child
{
color:blue;
}
</style>
</head>

<body>
<p>I am a strong man.</p>
<p>I am a strong man.</p>
</body>
</html>
```

Match the first <i> element in all <p> elements

In the following example, the selector matches the first <i> element in all <p> elements:

Example

```
<html>
<head>
<style>
p > i:first-child
{
color:blue;
```

```
}
</style>
</head>

<body>
<p>I am a <i>strong</i> man. I am a <i>strong</i> man.</p>
<p>I am a <i>strong</i> man. I am a <i>strong</i> man.</p>
</body>
</html>
```

Match all <i> elements in all first child <p> elements

In the following example, the selector matches all <i> elements in <p> elements that are the first child of another element:

Example

```
<html>
<head>
<style>
p:first-child i
{
color:blue;
}
</style>
</head>

<body>
<p>I am a <i>strong</i> man. I am a <i>strong</i> man.</p>
<p>I am a <i>strong</i> man. I am a <i>strong</i> man.</p>
</body>
</html>
```

CSS - The :lang Pseudo-class

The :lang pseudo-class allows you to define special rules for different languages.

In the example below, the :lang class defines the quotation marks for q elements with lang="no":

Example

```
<html>
<head>
<style>
q:lang(no) {quotes: "~" "~";}
</style>
</head>

<body>
<p>Some text <q lang="no">A quote in a paragraph</q> Some text.</p>
</body>
</html>
```

All CSS Pseudo Classes/Elements

Selector	Example	Example description
:link	a:link	Selects all unvisited links
:visited	a:visited	Selects all visited links
:active	a:active	Selects the active link
:hover	a:hover	Selects links on mouse over
:focus	input:focus	Selects the input element which has focus
::first-letter	p::first-letter	Selects the first letter of every <p> element
::first-line	p::first-line	Selects the first line of every <p> element
:first-child	p:first-child	Selects every <p> elements that is the first child of its parent
::before	p::before	Insert content before every <p> element
::after	p::after	Insert content after every <p> element
:lang(*language*)	p:lang(it)	Selects every <p> element with a lang attribute value starting with "it"

CSS Pseudo-elements

CSS pseudo-elements are used to add special effects to some selectors.

Syntax

The syntax of pseudo-elements:
selector::pseudo-element {property:value;}
CSS classes can also be used with pseudo-elements:
selector.class::pseudo-element {property:value;}

The ::first-line Pseudo-element

The ::first-line pseudo-element is used to add a special style to the first line of a text.
The ::first-line pseudo-element can only be applied to block-level elements.

Example

Format the first line of the text in p elements:

```
p::first-line
{
color:#ff0000;
font-variant:small-caps;
}
```

The following properties apply to the ::first-line pseudo-element:

- font properties
- color properties

- background properties
- word-spacing
- letter-spacing
- text-decoration
- vertical-align
- text-transform
- line-height
- clear

The ::first-letter Pseudo-element

The ::first-letter pseudo-element is used to add a special style to the first letter of a text.
The ::first-letter pseudo-element can only be applied to block-level elements.

Example

Format the first letter of the text in p elements.

```
p::first-letter
{
color:#ff0000;
font-size:xx-large;
}
```

The following properties apply to the ::first-letter pseudo- element:

- font properties
- color properties
- background properties
- margin properties
- padding properties
- border properties
- text-decoration
- vertical-align (only if "float" is "none")
- text-transform
- line-height
- float
- clear

Pseudo-elements and CSS Classes

Pseudo-elements can be combined with CSS classes:

```
p.article::first-letter {color:#ff0000;}

<p class="article">A paragraph in an article</p>
```

The example above will display the first letter of all paragraphs with class="article", in red.

Multiple Pseudo-elements

Several pseudo-elements can also be combined.
In the following example, the first letter of a paragraph will be red, in an xx-large font size. The rest of the first line will be blue, and in small-caps. The rest of the paragraph will be the default font size and color:

Example

```
p::first-letter
{
color:#ff0000;
font-size:xx-large;
}
p::first-line
{
color:#0000ff;
font-variant:small-caps;
}
```

CSS - The ::before Pseudo-element

The ::before pseudo-element can be used to insert some content before the content of an element.
The following example inserts an image before each <h1> element:

Example

```
h1::before
{
content:url(smiley.gif);
}
```

CSS - The ::after Pseudo-element

The ::after pseudo-element can be used to insert some content after the content of an element.
The following example inserts an image after each <h1> element:

Example

```
h1::after
{
content:url(smiley.gif);
}
```

Remove underline of links

You should consider carefully whether it is necessary to remove the underlining as it might decrease usability of your website significantly. People are used to blue underlined links on web pages and know that they can click on them The property `text-decoration` can be used to determine whether text is underlined or not. To remove underlining, simply set the value of `text-decoration` to none.

```
a {
      text-decoration:none;
}
```

Alternatively, you can set text-decoration along with other properties for all four pseudo-classes.

```
a:link {
      color: blue;
      text-decoration:none;
}

a:visited {
```

```
        color: purple;
        text-decoration:none;
}

a:active {
        background-color: yellow;
        text-decoration:none;
}

a:hover {
        color:red;
        text-decoration:none;
}
```

Identification and grouping of elements (class and id)

Sometimes you want to apply a special style to a particular element or a particular group of elements. In this lesson, we will take a closer look at how you can use `class` and `id` to specify properties for selected elements.
How can you color one particular headline differently than the other headlines on your website? How can you group your links into different categories and give each category a special style?

Grouping elements with class

Let's say that we have two lists of links of different grapes used for white juice and red juice. The HTML code could look like this:

```
<p>Grapes for white juice:</p>
<ul>
<li><a href="ri.htm">Richy</a></li>
<li><a href="ch.htm">Charm</a></li>
<li><a href="pb.htm">Pink Beuty </a></li>
</ul>

<p>Grapes for red juice:</p>
<ul>
<li><a href="cs.htm">Camel Sail</a></li>
<li><a href="me.htm">Merin</a></li>
<li><a href="pn.htm">Pizza Null</a></li>
</ul>
```

Then we want the white juice links to be yellow, the red juice links to be red and the rest of the existing links on the webpage to stay blue.
To achieve this, we divide the links into two categories. This is done by assigning a class to each link using the attribute `class`.
Let us try to specify some classes in the example above:

```
<p>Grapes for white juice:</p>
<ul>
<li><a href="ri.htm" class="whitejuice">Richy</a></li>
<li><a href="ch.htm" class="whitejuice">Charm</a></li>
```

```
<li><a href="pb.htm" class="whitejuice">Pink Beuty </a></li>
</ul>

<p>Grapes for red juice:</p>
<ul>
<li><a href="cs.htm" class="redjuice">Camel Sail</a></li>
<li><a href="me.htm" class="redjuice">Merin</a></li>
<li><a href="pn.htm" class="redjuice">Pizza Null</a></li>
</ul>
```

We can hereafter define special properties for links belonging to whitejuice and redjuice, respectively.

```
a {
      color: blue;
}

a.whitejuice {
      color: #FFBB00;
}

a.redjuice {
      color: #800000;
}
```

As shown in the example you can define the properties for elements which belong to a certain class by using **.classname** in the style sheet of the document.

Identification of element using id

In addition to grouping elements, you might need to identify one unique element. This is done by using the attribute `id`.
What is special about the attribute `id` is that there can not be two elements in the same document with the same `id`. Each `id` has to be unique. In other cases, you should use the `class` attribute instead.
Now, let us take a look at an example of a possible usage of `id`:

```
<h1>Page 1</h1>
...
<h2>Page 1.1</h2>
...
<h2>Page 1.2</h2>
...
<h1>Page 2</h1>
...
<h2>Page 2.1</h2>
...
<h3>Page 2.1.2</h3>
...
```

The above could be headings of any document split into Pages or paragraphs. It would be natural to assign an `id` to each Page as follows:

```
<h1 id="c1">Page 1</h1>
...
<h2 id="c1-1">Page 1.1</h2>
...
<h2 id="c1-2">Page 1.2</h2>
...
<h1 id="c2">Page 2</h1>
...
<h2 id="c2-1">Page 2.1</h2>
...
<h3 id="c2-1-2">Page 2.1.2</h3>
...
```

Let us say that the headline for Page 1.2 must be in red. This can be done accordingly with CSS:

```
#c1-2 {
      color: red;
}
```

As shown in the example above you can define the properties in a specific element by using #id in the stylesheet of the document.

All CSS Pseudo Classes/Elements

Selector	Example	Example description
:link	a:link	Selects all unvisited links
:visited	a:visited	Selects all visited links
:active	a:active	Selects the active link
:hover	a:hover	Selects links on mouse over
:focz-index	input:focus	Selects the input element which has focus
::first-letter	p::first-letter	Selects the first letter of every <p> element
::first-line	p::first-line	Selects the first line of every <p> element
:first-child	p:first-child	Selects every <p> elements that is the first child of its parent
::before	p::before	Insert content before every <p> element
::after	p::after	Insert content after every <p> element
:lang(*language*)	p:lang(it)	Selects every <p> element with a lang attribute value starting with "it"

CSS Navigation Bar

Demo: Navigation Bar

- Home
- News
- Articles
- Forum
- Contact
- About

Navigation Bars

Having easy-to-use navigation is important for any web site.

With CSS you can transform boring HTML menus into good-looking navigation bars.

Navigation Bar = List of Links

A navigation bar needs standard HTML as a base.

In our examples we will build the navigation bar from a standard HTML list.

A navigation bar is basically a list of links, so using the <ul> and <li> elements makes perfect sense:

Example

```
<ul>
<li><a href="default.asp">Home</a></li>
<li><a href="news.asp">News</a></li>
<li><a href="contact.asp">Contact</a></li>
<li><a href="about.asp">About</a></li>
</ul>
```

Now let's remove the bullets and the margins and padding from the list:

Example

```
ul
{
list-style-type:none;
margin:0;
padding:0;
}
```

Example explained:

- list-style-type:none - Removes the bullets. A navigation bar does not need list markers
- Setting margins and padding to 0 to remove browser default settings

The code in the example above is the standard code used in both vertical, and horizontal navigation bars.

Vertical Navigation Bar

To build a vertical navigation bar we only need to style the <a> elements, in addition to the code above:

Example

```
a
{
display:block;
width:60px;
}
```

Example explained:

- display:block - Displaying the links as block elements makes the whole link area clickable (not just the text), and it allows us to specify the width
- width:60px - Block elements take up the full width available by default. We want to specify a 60 px width

Horizontal Navigation Bar

There are two ways to create a horizontal navigation bar. Using **inline** or **floating** list items.
Both methods work fine, but if you want the links to be the same size, you have to use the floating method.

Inline List Items

One way to build a horizontal navigation bar is to specify the <li> elements as inline, in addition to the "standard" code above:

Example

```
li
{
display:inline;
}
```

Example explained:

- display:inline; - By default, <li> elements are block elements. Here, we remove the line breaks before and after each list item, to display them on one line

Floating List Items

In the example above the links have different widths.
For all the links to have an equal width, float the <li> elements and specify a width for the <a> elements:

Example

```
li
{
float:left;
}
a
{
display:block;
width:60px;
}
```

Example explained:

- float:left - use float to get block elements to slide next to each other
- display:block - Displaying the links as block elements makes the whole link area clickable (not just the text), and it allows us to specify the width
- width:60px - Since block elements take up the full width available, they cannot float next to each other. We specify the width of the links to 60px

CSS Image Gallery

CSS can be used to create an image gallery.

Add a description of the image here

Add a description of the image here

Add a description of the image here

Add a description of the image here

The following image gallery is created with CSS:

Example

```
<html>
<head>
<style>
div.img
  {
  margin:5px;
  padding: 5px;
  border:1px solid #0000ff;
  height:auto;
  width:auto;
  float:left;
  text-align:center;
  }
div.img img
  {
  display:inline;
  margin:5px;
  border:1px solid #ffffff;
  }
div.img a:hover img
  {
  border:1px solid #0000ff;
  }
div.desc
  {
  text-align:center;
  font-weight:normal;
  width:120px;
  margin:5px;
  }
</style>
</head>
<body>
<div class="img">
  <a target="_blank" href="Image_big.htm">
  <img src="image_small.jpg" alt="Image" width="110" height="90">
  </a>
  <div class="desc">Add a description of the image here</div>
</div>
<div class="img">
  <a target="_blank" href="Image2_big.htm">
  <img src="image2_small.jpg" alt="Image" width="110" height="90">
  </a>
  <div class="desc">Add a description of the image here</div>
</div>
<div class="img">
  <a target="_blank" href="Image3_big.htm">
  <img src="image3_small.jpg" alt="Image" width="110" height="90">
  </a>
```

```
  <div class="desc">Add a description of the image here</div>
</div>
<div class="img">
  <a target="_blank" href="Image4_big.htm">
  <img src="image4_small.jpg" alt="Image" width="110" height="90">
  </a>
  <div class="desc">Add a description of the image here</div>
</div>

</body>
</html>
```

CSS Image Opacity / Transparency

Creating transparent images with CSS is easy.

Example 1 - Creating a Transparent Image

The CSS3 property for transparency is **opacity**.

First we will show you how to create a transparent image with CSS.

Regular image:

The same image with transparency:

Look at the following CSS:

```
img
{
opacity:0.4;
filter:alpha(opacity=40); /* For IE8 and earlier */
}
```

IE9, Firefox, Chrome, Opera, and Safari use the property **opacity** for transparency. The opacity property can take a value from 0.0 - 1.0. A lower value makes the element more transparent.

IE8 and earlier use **filter:alpha(opacity=x)**. The x can take a value from 0 - 100. A lower value makes the element more transparent.

Example 2 - Image Transparency - Hover Effect

Mouse over the images:

The CSS looks like this:

```
img
{
opacity:0.4;
filter:alpha(opacity=40); /* For IE8 and earlier */
}
img:hover
{
opacity:1.0;
filter:alpha(opacity=100); /* For IE8 and earlier */
}
```

The first CSS block is similar to the code in Example 1. In addition, we have added what should happen when a user hover over one of the images. In this case we want the image to NOT be transparent when the user hover over it.

The CSS for this is: **opacity=1**.

IE8 and earlier: **filter:alpha(opacity=100)**.

When the mouse pointer moves away from the image, the image will be transparent again.

Example 3

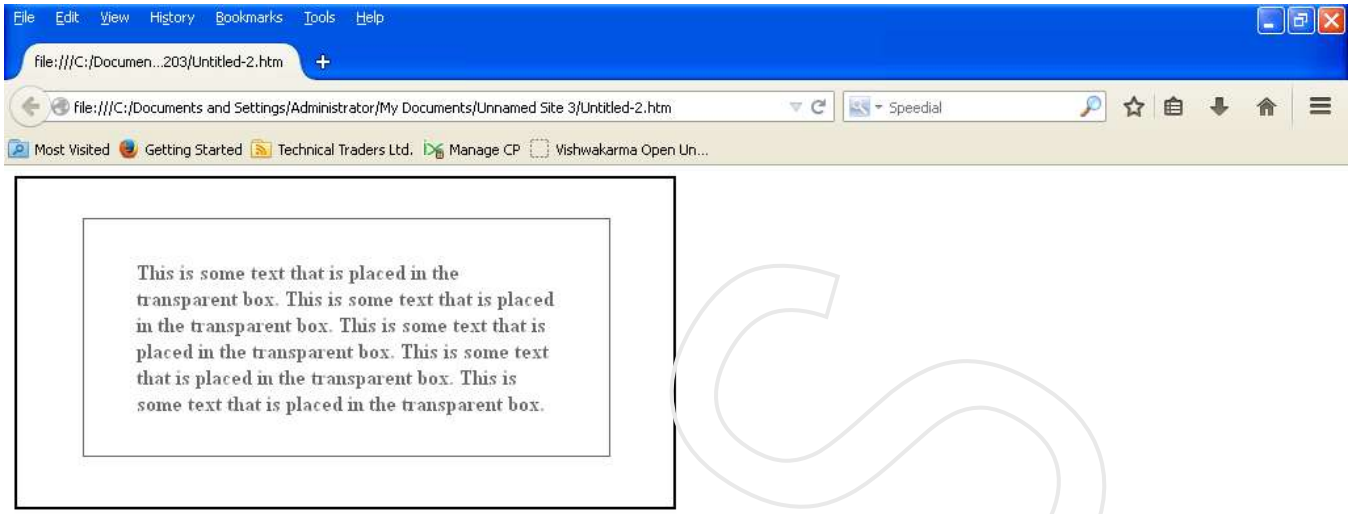
Text in Transparent Box

This is some text that is placed in the transparent box. This is some text that is placed in the transparent box. This is some text that is placed in the transparent box. This is some text that is placed in the transparent box. This is some text that is placed in the transparent box.

The source code looks like this:

```
<html>
<head>
<style>
div.background
  {
  width:500px;
  height:250px;
  background:url(Image.jpg) repeat;
  border:2px solid black;
  }
div.transbox
  {
  width:400px;
  height:180px;
  margin:30px 50px;
  background-color:#ffffff;
  border:1px solid black;
  opacity:0.6;
  filter:alpha(opacity=60); /* For IE8 and earlier */
  }
div.transbox p
  {
  margin:30px 40px;
  font-weight:bold;
  color:#000000;
  }
</style>
</head>

<body>

<div class="background">
<div class="transbox">
<p>This is some text that is placed in the transparent box.
This is some text that is placed in the transparent box.
This is some text that is placed in the transparent box.
This is some text that is placed in the transparent box.
This is some text that is placed in the transparent box.
</p>
</div>
</div>

</body>
</html>
```

Result



First, we create a div element (class="background") with a fixed height and width, a background image, and a border. Then we create a smaller div (class="transbox") inside the first div element. The "transbox" div have a fixed width, a background color, and a border - and it is transparent. Inside the transparent div, we add some text inside a p element.

CSS Image Sprites

Image Sprites

An image sprite is a collection of images put into a single image.

A web page with many images can take a long time to load and generates multiple server requests.

Using image sprites will reduce the number of server requests and save bandwidth.

Image Sprites - Simple Example

Instead of using three separate images, we use this single image ("img_icon.gif"):

With CSS, we can show just the part of the image we need.

In the following example the CSS specifies which part of the "img_icon.gif" image to show:

Example

```
img.home
{
width:46px;
```

```
height:44px;
background:url(img_icon.gif) 0 0;
}
```

Example explained:

- <img class="home" src="img_trans.gif"> - Only defines a small transparent image because the src attribute cannot be empty. The displayed image will be the background image we specify in CSS
- width:46px;height:44px; - Defines the portion of the image we want to use
- background:url(img_icon.gif) 0 0; - Defines the background image and its position (left 0px, top 0px)

This is the easiest way to use image sprites, now we want to expand it by using links and hover effects.

Image Sprites - Create a Navigation List

We want to use the sprite image ("img_icon.gif") to create a navigation list.
We will use an HTML list, because it can be a link and also supports a background image:

Example

```
#navlist{position:relative;}
#navlist li{margin:0;padding:0;list-style:none;position:absolute;top:0;}
#navlist li, #navlist a{height:44px;display:block;}

#home{left:0px;width:46px;}
#home{background:url('img_navsprites.gif') 0 0;}

#prev{left:63px;width:43px;}
#prev{background:url('img_icon.gif') -47px 0;}

#next{left:129px;width:43px;}
#next{background:url('img_icon.gif') -91px 0;}
```

Example explained:

- #navlist{position:relative;} - position is set to relative to allow absolute positioning inside it
- #navlist li{margin:0;padding:0;list-style:none;position:absolute;top:0;} - margin and padding is set to 0, list-style is removed, and all list items are absolute positioned
- #navlist li, #navlist a{height:44px;display:block;} - the height of all the images are 44px

Now start to position and style for each specific part:

- #home{left:0px;width:46px;} - Positioned all the way to the left, and the width of the image is 46px
- #home{background:url(img_icon.gif) 0 0;} - Defines the background image and its position (left 0px, top 0px)
- #prev{left:63px;width:43px;} - Positioned 63px to the right (#home width 46px + some extra space between items), and the width is 43px.
- #prev{background:url('img_icon.gif') -47px 0;} - Defines the background image 47px to the right (#home width 46px + 1px line divider)

- #next{left:129px;width:43px;}- Positioned 129px to the right (start of #prev is 63px + #prev width 43px + extra space), and the width is 43px.
- #next{background:url('img_icon.gif') -91px 0;} - Defines the background image 91px to the right (#home width 46px + 1px line divider + #prev width 43px + 1px line divider)

Image Sprites - Hover Effect

Now we want to add a hover effect to our navigation list.

The :hover selector is used to select elements when you mouse over them.

Tip: The :hover selector can be used on all elements, not only on links.

Our new image ("img_icon_hover.gif") contains three navigation images and three images to use for hover effects:

Because this is one single image, and not six separate files, there will be **no loading delay** when a user hovers over the image.

We only add three lines of code to add the hover effect:

Example

```
#home a:hover{background: url('img_icon_hover.gif') 0 -45px;}
#prev a:hover{background: url('img_icon_hover.gif') -47px -45px;}
#next a:hover{background: url('img_icon_hover.gif') -91px -45px;}
```

Example explained:

- #home a:hover{background: transparent url(img_icon_hover.gif) 0 -45px;} - For all three hover images we specify the same background position, only 45px further down

CSS Media Types

By using the @media rule, a website can have a different layout for screen, print, mobile phone, tablet, etc.

Media Types

Some CSS properties are only designed for a certain media. For example the "voice-family" property is designed for aural user agents. Some other properties can be used for different media types. For example, the "font-size" property can be used for both screen and print media, but perhaps with different values. A document usually needs a larger font-size on a screen than on paper, and sans-serif fonts are easier to read on the screen, while serif fonts are easier to read on paper.

The @media Rule

The @media rule allows different style rules for different media in the same style sheet.

The style in the example below tells the browser to display a 14 pixels Verdana font on the screen. But if the page is printed, it will be in a 20 pixels font, and in a red color. Notice that the font-weight is set to bold, both on screen and on paper:

```
<html>
<head>
<style>
@media screen
  {
  p.test {font-family:verdana,sans-serif;font-size:14px;}
  }
@media print
  {
  p.test {font-size:20px;color:red;}
  }
@media screen,print
  {
  p.test {font-weight:bold;}
  }
</style>
</head>

<body>
....
</body>
</html>
```

See it yourself ! Print this page (or open Print Preview), and you will see that the paragraph under "Media Types" will be displayed in a larger font size, and in red color.

Other Media Types

Media Type	Description
all	Used for all media type devices
aural	Used for speech and sound synthesizers
braille	Used for braille tactile feedback devices
embossed	Used for paged braille printers
handheld	Used for small or handheld devices
print	Used for printers
projection	Used for projected presentations, like slides
screen	Used for computer screens
tty	Used for media using a fixed-pitch character grid, like teletypes and terminals
tv	Used for television-type devices

CSS Attribute Selectors

Style HTML Elements With Specific Attributes

It is possible to style HTML elements that have specific attributes, not just class and id.

CSS [attribute] Selector

The [attribute] selector is used to select elements with the specified attribute.

The following example selects all <a> elements with a target attribute:

Example

```
a[target]
{
background-color:yellow;
}
```

CSS [attribute=value] Selector

The [attribute=value] selector is used to select elements with the specified attribute and value.

The following example selects all <a> elements with a target="_blank" attribute:

Example

```
a[target="_blank"]
{
background-color:yellow;
}
```

CSS [attribute~=value] Selector

The [attribute~=value] selector is used to select elements with an attribute value containing a specified word.

The following example selects all elements with a title attribute that contains a space-separated list of words, one of which is "flower":

Example

```
[title~="flower"]
{
border:5px solid yellow;
}
```

The example above will match elements with title="flower", title="summer flower", and title="flower new", but not title="my-flower" or title="flowers".

CSS [attribute|=value] Selector

The [attribute|=value] selector is used to select elements with the specified attribute starting with the specified value.

The following example selects all elements with a class attribute value that begins with "top":

Note: The value has to be a whole word, either alone, like class="top", or followed by a hyphen(-), like class="top-text"!

Example

```
[class|="top"]
{
background:yellow;
}
```

CSS [attribute^=value] Selector

The [attribute^=value] selector is used to select elements whose attribute value begins with a specified value.

The following example selects all elements with a class attribute value that begins with "top":

Note: The value does not have to be a whole word!

Example

```
[class^="top"]
{
background:yellow;
}
```

CSS [attribute$=value] Selector

The [attribute$=value] selector is used to select elements whose attribute value ends with a specified value.

The following example selects all elements with a class attribute value that ends with "test":

Note: The value does not has to be a whole word!

Example

```
[class$="test"]
{
background:yellow;
}
```

CSS [attribute*=value] Selector

The [attribute*=value] selector is used to select elements whose attribute value contains a specified value.

The following example selects all elements with a class attribute value that contains "te":

Note: The value does not has to be a whole word!

Example

```
[class*="te"]
{
background:yellow;
}
```

Styling Forms

The attribute selectors can be useful for styling forms without class or ID:

Example

```
input[type="text"]
{
width:150px;
display:block;
margin-bottom:10px;
background-color:yellow;
}
input[type="button"]
{
width:120px;
margin-left:35px;
display:block;
}
```

CSS3 Introduction

CSS3 Modules

CSS3 has been split into "modules". It contains the "old CSS specification" (which has been split into smaller pieces). In addition, new modules are added.
Some of the most important CSS3 modules are:

- Selectors
- Box Model
- Backgrounds and Borders
- Image Values and Replaced Content
- Text Effects
- 2D/3D Transformations
- Animations
- Multiple Column Layout
- User Interface

CSS3 Recommendation

The CSS3 specification is still under development by W3C.
However, many of the new CSS3 properties have been implemented in modern browsers.

CSS3 Borders

With CSS3, you can create rounded borders, add shadow to boxes, and use an image as a border - without using a design program, like Photoshop.
In this Page you will learn about the following border properties:

- border-radius
- box-shadow
- border-image

CSS3 The border-radius Property - Rounded Corners

Adding rounded corners in CSS2 was tricky. We had to use different images for each corner. In CSS3, creating rounded corners is easy. In CSS3, the border-radius property is used to create rounded corners:

This box has rounded corners!

Example

Add rounded corners to a div element:

```
div
{
border:2px solid;
border-radius:25px;
}
```

CSS3 The box-shadow Property

In CSS3, the box-shadow property is used to add shadow to boxes:

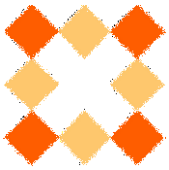
Example

Add a box-shadow to a div element:

```
div
{
box-shadow: 10px 10px 5px #888888;
}
```

CSS3 The border-image Property

With the CSS3 border-image property you can use an image to create a border:
The border-image property allows you to specify an image as a border!
The original image used to create the border above:



Example

Use an image to create a border around a div element:

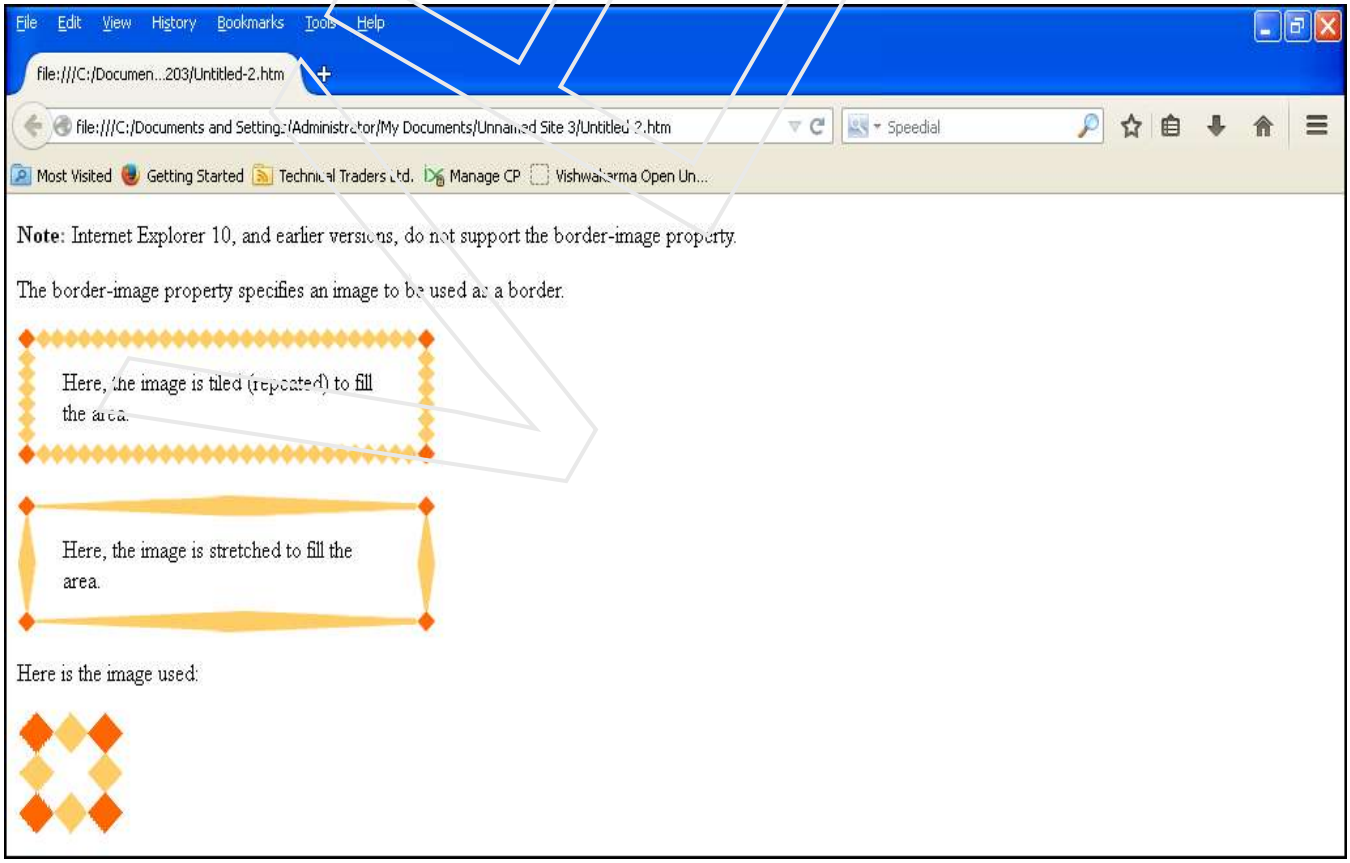
```
div
{
-webkit-border-image:url(border.png) 30 30 round; /* Safari 5 */
-o-border-image:url(border.png) 30 30 round; /* Opera 10.5-12.1 */
border-image:url(border.png) 30 30 round;
}
```

Program

```
<!DOCTYPE html>
<html>
<head>
<style>
div
{
border:15px solid transparent;
width:250px;
padding:10px 20px;
}

#round
{
-webkit-border-image:url(border.png) 30 30 round; /* Safari 5 */
-o-border-image:url(border.png) 30 30 round; /* Opera 10.5-12.1 */
border-image:url(border.png) 30 30 round;
}
```

```
#stretch
{
-webkit-border-image:url(border.png) 30 30 stretch; /* Safari 5 */
-o-border-image:url(border.png) 30 30 stretch; /* Opera 10.5-12.1 */
border-image:url(border.png) 30 30 stretch;
}
</style>
</head>
<body>

<p><strong>Note:</strong> Internet Explorer 10, and earlier versions, do not support the border-image property.</p>

<p>The border-image property specifies an image to be used as a border.</p>

<div id="round">Here, the image is tiled (repeated) to fill the area.</div>
<br>
<div id="stretch">Here, the image is stretched to fill the area.</div>

<p>Here is the image used:</p>
<img src="border.png">

</body>
</html>
```

Result



CSS3 Border Properties

Property	Description	CSS
border-image	A shorthand property for setting all the border-image-* properties	3
border-radius	A shorthand property for setting all the four border-*-radius properties	3
box-shadow	Attaches one or more drop-shadows to the box	

CSS3 Backgrounds

CSS3 contains several new background properties,
which allow greater control of the background element.

CSS3 The background-size Property

The background-size property specifies the size of the background image.
Before CSS3, the background image size was determined by the actual size of the image. In CSS3 it is possible to specify the size of the background image, which allows us to re-use background images in different contexts.
You can specify the size in pixels or in percentages. If you specify the size as a percentage, the size is relative to the width and height of the parent element.

Example 1

Resize a background image:

```
div
{
background:url(img_flwr.gif);
background-size:80px 60px;
background-repeat:no-repeat;
}
```

Example 2

Stretch the background image to completely fill the content area:

```
div
{
background:url(img_flwr.gif);
background-size:100% 100%;
background-repeat:no-repeat;
}
```

CSS3 The background-origin Property

The background-origin property specifies the positioning area of the background images.

The background image can be placed within the content-box, padding-box, or border-box area.

Example

Position the background image within the content-box:

```
div
{
background:url(img_flwr.gif);
background-repeat:no-repeat;
background-size:100% 100%;
background-origin:content-box;
}
```

CSS3 Multiple Background Images :

CSS3 allows you to use several background images for an element

Example

Set two background images for the body element:

```
body
{
background:url(img_tree.gif),url(img_flwr.gif);
background-size:100% 100%;
background-repeat:no-repeat;
}
```

CSS3 Background Properties

Property	Description	CSS
background-clip	Specifies the painting area of the background images	3
background-origin	Specifies the positioning area of the background images	3
background-size	Specifies the size of the background images	3

CSS3 Gradients

CSS3 gradients let you display smooth transitions between two or more specified colors.

Earlier, you had to use images for these effects. However, by using CSS3 gradients you can reduce download time and bandwidth usage. In addition, elements with gradients look better when zoomed, because the gradient is generated by the browser.

CSS3 defines two types of gradients:

- **Linear Gradients (goes down/up/left/right/diagonally)**
- **Radial Gradients (defined by their center)**

CSS3 Linear Gradients

To create a linear gradient you must define at least two color stops. Color stops are the colors you want to render smooth transitions among. You can also set a starting point and a direction (or an angle) along with the gradient effect.

Example of Linear Gradient:

Syntax

background: linear-gradient(*direction, color-stop1, color-stop2, ...*);

Linear Gradient - Top to Bottom (this is default)

The following example shows a linear gradient that starts at the top. It starts red, transitioning to blue:

Example

A linear gradient from top to bottom:

```
#grad
{
background: -webkit-linear-gradient(red, blue); /* For Safari 5.1 to 6.0 */
background: -o-linear-gradient(red, blue); /* For Opera 11.1 to 12.0 */
background: -moz-linear-gradient(red, blue); /* For Firefox 3.6 to 15 */
background: linear-gradient(red, blue); /* Standard syntax */
}
```

Linear Gradient - Left to Right

The following example shows a linear gradient that starts from the left. It starts red, transitioning to blue:

Example

A linear gradient from left to right:

```
#grad
{
background: -webkit-linear-gradient(left, red , blue); /* For Safari 5.1 to 6.0 */
background: -o-linear-gradient(right, red, blue); /* For Opera 11.1 to 12.0 */
background: -moz-linear-gradient(right, red, blue); /* For Firefox 3.6 to 15 */
background: linear-gradient(to right, red , blue); /* Standard syntax */
}
```

Linear Gradient - Diagonal

You can make a gradient diagonally by specifying both the horizontal and vertical starting positions.

The following example shows a linear gradient that starts at top left (and goes to bottom right). It starts red, transitioning to blue:

Example

A linear gradient that starts at top left (and goes to bottom right):

```
#grad
{
background: -webkit-linear-gradient(left top, red , blue); /* For Safari 5.1 to 6.0 */
background: -o-linear-gradient(bottom right, red, blue); /* For Opera 11.1 to 12.0 */
background: -moz-linear-gradient(bottom right, red, blue); /* For Firefox 3.6 to 15 */
background: linear-gradient(to bottom right, red , blue); /* Standard syntax */
}
```

Using Angles

If you want more control over the direction of the gradient, you can define an angle, instead of the predefined directions (to bottom, to top, to right, to left, to bottom right, etc.).

Syntax

background: linear-gradient(*angle*, *color-stop1*, *color-stop2*);

The angle is specified as an angle between a horizontal line and the gradient line, going counter-clockwise. In other words, 0deg creates a bottom to top gradient, while 90deg generates a left to right gradient.

The following example shows how to use angles on linear gradients:

Example:

A linear gradient with a specified angle:

```
#grad
{
background: -webkit-linear-gradient(180deg, red, blue); /* For Safari 5.1 to 6.0 */
background: -o-linear-gradient(180deg, red, blue); /* For Opera 11.1 to 12.0 */
background: -moz-linear-gradient(180deg, red, blue); /* For Firefox 3.6 to 15 */
background: linear-gradient(180deg, red, blue); /* Standard syntax */
}
```

Using Multiple Color Stops

The following example shows how to set multiple color stops:

Example

A linear gradient from top to bottom with multiple color stops:

```
#grad
{
background: -webkit-linear-gradient(red, green, blue); /* For Safari 5.1 to 6.0 */
background: -o-linear-gradient(red, green, blue); /* For Opera 11.1 to 12.0 */
background: -moz-linear-gradient(red, green, blue); /* For Firefox 3.6 to 15 */
background: linear-gradient(red, green, blue); /* Standard syntax */
}
```

The following example shows how to create a linear gradient with the color of the rainbow and some text:

Example

```
#grad
{
/* For Safari 5.1 to 6.0 */
background: -webkit-linear-gradient(left,red,orange,yellow,green,blue,indigo,violet);
/* For Opera 11.1 to 12.0 */
background: -o-linear-gradient(left,red,orange,yellow,green,blue,indigo,violet);
/* For Fx 3.6 to 15 */
background: -moz-linear-gradient(left,red,orange,yellow,green,blue,indigo,violet);
/* Standard syntax */
background: linear-gradient(to right, red,orange,yellow,green,blue,indigo,violet);
}
```

Using Transparency

CSS3 gradients also support transparency, which can be used to create fading effects.

To add transparency, we use the rgba() function to define the color stops. The last parameter in the rgba() function can be a value from 0 to 1, and it defines the transparency of the color: 0 indicates full transparency, 1 indicates full color (no transparency).

The following example shows a linear gradient that starts from the left. It starts fully transparent, transitioning to full color red:

Example

A linear gradient from left to right, with transparency:

```
#grad
{
background: -webkit-linear-gradient(left,rgba(255,0,0,0),rgba(255,0,0,1)); /*Safari 5.1-6*/
background: -o-linear-gradient(right,rgba(255,0,0,0),rgba(255,0,0,1)); /*Opera 11.1-12*/
background: -moz-linear-gradient(right,rgba(255,0,0,0),rgba(255,0,0,1)); /*Fx 3.6-15*/
background: linear-gradient(to right, rgba(255,0,0,0), rgba(255,0,0,1)); /*Standard*/
}
```

Repeating a linear-gradient

The repeating-linear-gradient() function is used to repeat linear gradients:

Example

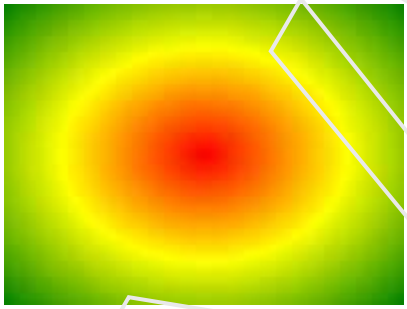
A repeating linear gradient:

```
#grad
{
/* Safari 5.1 to 6.0 */
background: -webkit-repeating-linear-gradient(red, yellow 10%, green 20%);
/* Opera 11.1 to 12.0 */
background: -o-repeating-linear-gradient(red, yellow 10%, green 20%);
/* Firefox 3.6 to 15 */
background: -moz-repeating-linear-gradient(red, yellow 10%, green 20%);
/* Standard syntax */
background: repeating-linear-gradient(red, yellow 10%, green 20%);
}
```

CSS3 Radial Gradients

A radial gradient is defined by its center.
To create a radial gradient you must also define at least two color stops. You can also specify the gradient's center, shape (circle or ellipse) as well as its size. By default, center is center, shape is ellipse, and size is farthest-corner.

Example of Radial Gradient:



Syntax

background: radial-gradient(*center, shape size, start-color, ..., last-color*);

Radial Gradient - Evenly Spaced Color Stops (this is default)

Example

A radial gradient with evenly spaced color stops:

```
#grad
{
background: -webkit-radial-gradient(red, green, blue); /* Safari 5.1 to 6.0 */
```

```
background: -o-radial-gradient(red, green, blue); /* For Opera 11.6 to 12.0 */
background: -moz-radial-gradient(red, green, blue); /* For Firefox 3.6 to 15 */
background: radial-gradient(red, green, blue); /* Standard syntax */
}
```

Radial Gradient - Differently Spaced Color Stops

Example

A radial gradient with differently spaced color stops:

```
#grad
{
background: -webkit-radial-gradient(red 5%, green 15%, blue 60%); /* Safari 5.1-6.0 */
background: -o-radial-gradient(red 5%, green 15%, blue 60%); /* For Opera 11.6-12.0 */
background: -moz-radial-gradient(red 5%, green 15%, blue 60%); /* For Firefox 3.6-15 */
background: radial-gradient(red 5%, green 15%, blue 60%); /* Standard syntax */
}
```

Set Shape

The shape parameter defines the shape. It can take the value circle or ellipse. The default value is ellipse.

Example

A radial gradient with the shape of a circle:

```
#grad
{
background: -webkit-radial-gradient(circle, red, yellow, green); /* Safari */
background: -o-radial-gradient(circle, red, yellow, green); /* Opera 11.6 to 12.0 */
background: -moz-radial-gradient(circle, red, yellow, green); /* Firefox 3.6 to 15 */
background: radial-gradient(circle, red, yellow, green); /* Standard syntax */
}
```

Use of Different Size Keywords

The size parameter defines the size of the gradient. It can take four values:

- **closest-side**
- **farthest-side**
- **closest-corner**
- **farthest-corner**

Example

A radial gradient with different size keywords:

```
#grad1
{
```

```
/* Safari 5.1 to 6.0 */
background: -webkit-radial-gradient(60% 55%, closest-side,blue,green,yellow,black);
/* For Opera 11.6 to 12.0 */
background: -o-radial-gradient(60% 55%, closest-side,blue,green,yellow,black);
/* For Firefox 3.6 to 15 */
background: -moz-radial-gradient(60% 55%, closest-side,blue,green,yellow,black);
/* Standard syntax */
background: radial-gradient(60% 55%, closest-side,blue,green,yellow,black);
}

#grad2
{
/* Safari 5.1 to 6.0 */
background: -webkit-radial-gradient(60% 55%, farthest-side,blue,green,yellow,black);
/* Opera 11.6 to 12.0 */
background: -o-radial-gradient(60% 55%, farthest-side,blue,green,yellow,black);
/* For Firefox 3.6 to 15 */
background: -moz-radial-gradient(60% 55%, farthest-side,blue,green,yellow,black);
/* Standard syntax */
background: radial-gradient(60% 55%, farthest-side,blue,green,yellow,black);
}
```

Repeating a radial-gradient

The repeating-radial-gradient() function is used to repeat radial gradients:

Example

A repeating radial gradient:

```
#grad
{
/* For Safari 5.1 to 6.0 */
background: -webkit-repeating-radial-gradient(red, yellow 10%, green 15%);
/* For Opera 11.6 to 12.0 */
background: -o-repeating-radial-gradient(red, yellow 10%, green 15%);
/* For Firefox 3.6 to 15 */
background: -moz-repeating-radial-gradient(red, yellow 10%, green 15%);
/* Standard syntax */
background: repeating-radial-gradient(red, yellow 10%, green 15%);
}
```

CSS3 Text Effects

CSS3 contains several new text features.

Now you will learn about the following text properties:

- text-shadow
- word-wrap

CSS3 Text Shadow

In CSS3, the text-shadow property applies shadow to text.

You specify the horizontal shadow, the vertical shadow, the blur distance, and the color of the shadow:

Example

Add a shadow to a header:

```
h1
{
text-shadow: 5px 5px 5px #FF0000;
}
```

CSS3 Word Wrapping

If a word is too long to fit within an area, it expands outside:

This paragraph contains a very long word: thisisaveryveryveryveryveryverylongword. The long word will break and wrap to the next line.

In CSS3, the word-wrap property allows you to force the text to wrap - even if it means splitting it in the middle of a word:

This paragraph contains a very long word: thisisaveryveryveryveryveryverylongword. The long word will break and wrap to the next line.

Example

Allow long words to be able to break and wrap onto the next line:

```
p {word-wrap:break-word;}
```

CSS3 Text Properties

Property	Description	CSS
hanging-punctuation	Specifies whether a punctuation character may be placed outside the line box	3
punctuation-trim	Specifies whether a punctuation character should be trimmed	3
text-align-last	Describes how the last line of a block or a line right before a forced line break is aligned when text-align is "justify"	3
text-emphasis	Applies emphasis marks, and the foreground color of the emphasis marks, to the element's text	3

text-justify	Specifies the justification method used when text-align is "justify"	3
text-outline	Specifies a text outline	3
text-overflow	Specifies what should happen when text overflows the containing element	3
text-shadow	Adds shadow to text	3
text-wrap	Specifies line breaking rules for text	3
word-break	Specifies line breaking rules for non-CJK scripts	3
word-wrap	Allows long, unbreakable words to be broken and wrap to the next line	3

CSS3 Web Fonts - The @font-face Rule

Web fonts allow Web designers to use fonts that are not installed on the user's computer.
When you have found/bought the font you wish to use, just include the font file on your web server, and it will be automatically downloaded to the user when needed.
Your "own" fonts are defined within the CSS3 @font-face rule.

Different Font Formats

TrueType Fonts (TTF)

TrueType is a font standard developed in the late 1980s, by Apple and Microsoft. TrueType is the most common font format for both the Mac OS and Microsoft Windows operating systems.

OpenType Fonts (OTF)

OpenType is a format for scalable computer fonts. It was built on TrueType, and is a registered trademark of Microsoft. OpenType fonts are used commonly today on the major computer platforms.

The Web Open Font Format (WOFF)

WOFF is a font format for use in web pages. It was developed in 2009, and is now a W3C Recommendation. WOFF is essentially OpenType or TrueType with compression and additional metadata. The goal is to support font distribution from a server to a client over a network with bandwidth constraints.

SVG Fonts/Shapes

SVG fonts allow SVG to be used as glyphs when displaying text. The SVG 1.1 specification define a font module that allows the creation of fonts within an SVG document. You can also apply CSS to SVG documents, and the @font-face rule can be applied to text in SVG documents.

Embedded OpenType Fonts (EOT)

EOT fonts are a compact form of OpenType fonts designed by Microsoft for use as embedded fonts on web pages.

Using The Font You Want

In the CSS3 @font-face rule you must first define a name for the font (e.g. myFirstFont), and then point to the font file.
To use the font for an HTML element, refer to the name of the font (myFirstFont) through the font-family property:

Example

```
<style>
@font-face
{
font-family: myFirstFont;
src: url(sansation_light.woff);
}

div
{
font-family:myFirstFont;
}
</style>
```

Using Bold Text

You must add another @font-face rule containing descriptors for bold text:

Example

```
@font-face
{
font-family: myFirstFont;
src: url(sansation_bold.woff);
font-weight:bold;
}
```

The file "sansation_bold.woff" is another font file, that contains the bold characters for the Sansation font.
Browsers will use this whenever a piece of text with the font-family "myFirstFont" should render as bold.
This way you can have many @font-face rules for the same font.

CSS3 Font Descriptors

The following table lists all the font descriptors that can be defined inside the @font-face rule:

Descriptor	Values	Description
font-family	*name*	Required. Defines a name for the font
src	*URL*	Required. Defines the URL of the font file
font-stretch	normal condensed	Optional. Defines how the font should be stretched. Default is "normal"

	ultra-condensed extra-condensed semi-condensed expanded semi-expanded extra-expanded ultra-expanded	
font-style	normal italic oblique	Optional. Defines how the font should be styled. Default is "normal"
font-weight	normal bold 100 200 300 400 500 600 700 800 900	Optional. Defines the boldness of the font. Default is "normal"
unicode-range	*unicode-range*	Optional. Defines the range of UNICODE characters the font supports. Default is "U+0-10FFFF"

CSS3 Transforms

With CSS3 transforms we can move, scale, turn, spin, and stretch elements.
A transformation is an effect that lets an element change shape, size and position.
You can transform your elements using 2D or 3D transformation.

CSS3 2D Transforms

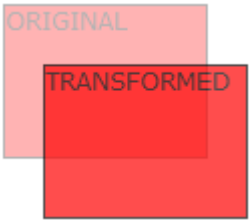
Now you will learn about the 2d transform methods:

- translate()
- rotate()
- scale()
- skew()
- matrix()

Example

```
div
{
-ms-transform: rotate(30deg); /* IE 9 */
-webkit-transform: rotate(30deg); /* Chrome, Safari, Opera */
transform: rotate(30deg);
}
```

The translate() Method



With the translate() method, the element moves from its current position, depending on the parameters given for the left (X-axis) and the top (Y-axis) position:

Example

```
div
{
-ms-transform: translate(50px,100px); /* IE 9 */
-webkit-transform: translate(50px,100px); /* Chrome, Safari, Opera */
transform: translate(50px,100px);
}
```

The value translate(50px,100px) moves the element 50 pixels from the left, and 100 pixels from the top.

The rotate() Method

With the rotate() method, the element rotates clockwise at a given degree. Negative values are allowed and rotates the element counter-clockwise.

Example

```
div
{
-ms-transform: rotate(30deg); /* IE 9 */
-webkit-transform: rotate(30deg); /* Chrome, Safari, Opera */
transform: rotate(30deg);
}
```

The value rotate(30deg) rotates the element clockwise 30 degrees.

The scale() Method

With the scale() method, the element increases or decreases the size, depending on the parameters given for the width (X-axis) and the height (Y-axis):

Example

```
div
{
-ms-transform: scale(2,4); /* IE 9 */
-webkit-transform: scale(2,4); /* Chrome, Safari, Opera */
transform: scale(2,4);
}
```

The value scale(2,4) transforms the width to be twice its original size, and the height 4 times its original size.

The skew() Method

With the skew() method, the element turns in a given angle, depending on the parameters given for the horizontal (X-axis) and the vertical (Y-axis) lines:

Example

```
div
{
-ms-transform: skew(30deg,20deg); /* IE 9 */
-webkit-transform: skew(30deg,20deg); /* Chrome, Safari, Opera */
transform: skew(30deg,20deg);
}
```

The value skew(30deg,20deg) turns the element 30 degrees around the X-axis, and 20 degrees around the Y-axis.

The matrix() Method

The matrix() method combines all of the 2D transform methods into one.

The matrix method take six parameters, containing mathematic functions, which allows you to: rotate, scale, move (translate), and skew elements.

Example

How to rotate a div element 30 degrees, using the matrix method:

```
div
{
-ms-transform:matrix(0.866,0.5,-0.5,0.866,0,0); /* IE 9 */
-webkit-transform:matrix(0.866,0.5,-0.5,0.866,0,0); /* Chrome, Safari, Opera */
transform:matrix(0.866,0.5,-0.5,0.866,0,0);
}
```

CSS3 Transform Properties

The following table lists all the transform properties:

Property	Description
transform	Applies a 2D or 3D transformation to an element
transform-origin	Allows you to change the position on transformed elements

2D Transform Methods

Function	Description
matrix(*n*,*n*,*n*,*n*,*n*,*n*)	Defines a 2D transformation, using a matrix of six values
translate(*x*,*y*)	Defines a 2D translation, moving the element along the X- and the Y-axis
translateX(*n*)	Defines a 2D translation, moving the element along the X-axis
translateY(*n*)	Defines a 2D translation, moving the element along the Y-axis
scale(*x*,*y*)	Defines a 2D scale transformation, changing the elements width and height
scaleX(*n*)	Defines a 2D scale transformation, changing the element's width
scaleY(*n*)	Defines a 2D scale transformation, changing the element's height
rotate(*angle*)	Defines a 2D rotation, the angle is specified in the parameter
skew(*x-angle*,*y-angle*)	Defines a 2D skew transformation along the X- and the Y-axis
skewX(*angle*)	Defines a 2D skew transformation along the X-axis
skewY(*angle*)	Defines a 2D skew transformation along the Y-axis

CSS3 3D Transforms

CSS3 allows you to format your elements using 3D transforms.
Now you will learn about some of the 3D transform methods:

- rotateX()
- rotateY()

Click on the elements below, to see the difference between a 2D transform and a 3D transform:

2D rotate
3D rotate

The rotateX() Method

With the rotateX() method, the element rotates around its X-axis at a given degree.

Example

```
div
{
-webkit-transform: rotateX(120deg); /* Chrome, Safari, Opera */
transform: rotateX(120deg);
}
```

The rotateY() Method

With the rotateY() method, the element rotates around its Y-axis at a given degree.

Example

```
div
{
-webkit-transform: rotateY(130deg); /* Chrome, Safari, Opera */
transform: rotateY(130deg);
}
```

CSS3 Transform Properties

The following table lists all the transform properties:

Property	Description
transform	Applies a 2D or 3D transformation to an element
transform-origin	Allows you to change the position on transformed elements
transform-style	Specifies how nested elements are rendered in 3D space
perspective	Specifies the perspective on how 3D elements are viewed
perspective-origin	Specifies the bottom position of 3D elements
backface-visibility	Defines whether or not an element should be visible when not facing the screen

3D Transform Methods

Function	Description
matrix3d (*n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n*)	Defines a 3D transformation, using a 4x4 matrix of 16 values
translate3d(*x,y,z*)	Defines a 3D translation
translateX(*x*)	Defines a 3D translation, using only the value for the X-axis
translateY(*y*)	Defines a 3D translation, using only the value for the Y-axis
translateZ(*z*)	Defines a 3D translation, using only the value for the Z-axis
scale3d(*x,y,z*)	Defines a 3D scale transformation
scaleX(*x*)	Defines a 3D scale transformation by giving a value for the X-axis
scaleY(*y*)	Defines a 3D scale transformation by giving a value for the Y-axis
scaleZ(*z*)	Defines a 3D scale transformation by giving a value for the Z-axis
rotate3d(*x,y,z,angle*)	Defines a 3D rotation
rotateX(*angle*)	Defines a 3D rotation along the X-axis
rotateY(*angle*)	Defines a 3D rotation along the Y-axis
rotateZ(*angle*)	Defines a 3D rotation along the Z-axis
perspective(*n*)	Defines a perspective view for a 3D transformed element

CSS3 Transitions

With CSS3, we can add an effect when changing from one style to another, without using Flash animations or JavaScripts.

Browser Support

The numbers in the table specifies the first browser version that fully supports the property.
Numbers followed by -webkit-, -moz-, or -o- specifies the first version that worked with a prefix.

What Are CSS3 Transitions?

CSS3 transitions are effects that let an element gradually change from one style to another.

To do this, you must specify two things:

- the CSS property you want to add an effect to
- the duration of the effect

Example

Add a transition effect on the width property, with a duration of 2 seconds:

```
div
{
-webkit-transition: width 2s; /* For Safari 3.1 to 6.0 */
transition: width 2s;
}
```

Note: If the duration part is not specified, the transition will have no effect, because the default value is 0.

The transition effect will start when the specified CSS property changes value. A typical CSS property change would be when a user mouse-over an element:

Example

Specify :hover for elements:

```
div:hover
{
width:300px;
}
```

Note: When the cursor mouse out of the element, it gradually changes back to its original style.

Multiple Changes

To add transition effects for more than one CSS property, separate the properties with a comma:

Example

Add transition effects on width, height, and transformation:

```
div
{
-webkit-transition: width 2s, height 2s,-webkit-transform 2s;  /* For Safari 3.1 to 6.0 */
transition: width 2s, height 2s, transform 2s,
}
```

More Transition Examples

The example below uses all the four transition properties:

Example

```
div
{
/* For Safari 3.1 to 6.0 */
-webkit-transition-property:width;
-webkit-transition-duration:1s;
-webkit-transition-timing-function:linear;
-webkit-transition-delay:2s;
/* Standard syntax */
transition-property: width;
transition-duration: 1s;
transition-timing-function: linear;
transition-delay: 2s;
}
```

The same transition effects as the example above. However, here we are using the shorthand transition property:

Example

```
div
{
-webkit-transition:width 1s linear 2s; /* For Safari 3.1 to 6.0 */
transition: width 1s linear 2s;
}
```

CSS3 Transition Properties

The following table lists all the transition properties:

Property	Description	CSS
transition	A shorthand property for setting the four transition properties into a single property	3
transition-delay	Specifies when the transition effect will start	3
transition-duration	Specifies how many seconds or milliseconds a transition effect takes to complete	3
transition-property	Specifies the name of the CSS property the transition effect is for	3
transition-timing-function	Specifies the speed curve of the transition effect	3

CSS3 Animations

With CSS3, we can create animations which can replace Flash animations, animated images, and JavaScripts in existing web pages.

CSS3 @keyframes Rule

The @keyframes rule is where the animation is created. Specify a CSS style inside the @keyframes rule and the animation will gradually change from the current style to the new style.

CSS3 Animation

When an animation is created in the @keyframe rule, you must bind it to a selector, otherwise the animation will have no effect.
Bind the animation to a selector (element) by specifying at least these two properties:
the name of the animation
the duration of the animation

Example

Bind the "myfirst" animation to the div element. Animation duration: 5 seconds:

```
div
{
-webkit-animation: myfirst 5s; /* Chrome, Safari, Opera */
animation: myfirst 5s;
```

```
}

/* Chrome, Safari, Opera */
@-webkit-keyframes myfirst
{
from {background: red;}
to {background: yellow;}
}

/* Standard syntax */
@keyframes myfirst
{
from {background: red;}
to {background: yellow;}
}
```

Note: If the duration part is not specified, the animation will have no effect, because the default value is 0.

CSS3 Animations

An animation lets an element gradually change from one style to another.
You can change as many properties you want, as many times you want.
You can specify when the change will happen in percent, or you can use the keywords "from" and "to" (which represents 0% and 100%).
0% represents the start of the animation, 100% is when the animation is complete.

Example

Change the background color when the animation is 25%, and 50%, and again when the animation is 100% complete:

```
/* Chrome, Safari, Opera */
@-webkit-keyframes myfirst
{
0%   {background: red;}
25%  {background: yellow;}
50%  {background: blue;}
100% {background: green;}
}

/* Standard syntax */
@keyframes myfirst
{
0%   {background: red;}
25%  {background: yellow;}
50%  {background: blue;}
100% {background: green;}
}
```

Example

Change the background color and the position when the animation is 25%, 50%, 75%, and again when the animation is 100% complete:

```
/* Chrome, Safari, Opera */
@-webkit-keyframes myfirst
{
0%   {background: red; left:0px; top:0px;}
25%  {background: yellow; left:200px; top:0px;}
50%  {background: blue; left:200px; top:200px;}
75%  {background: green; left:0px; top:200px;}
100% {background: red; left:0px; top:0px;}
}

/* Standard syntax */
@keyframes myfirst
{
0%   {background: red; left:0px; top:0px;}
25%  {background: yellow; left:200px; top:0px;}
50%  {background: blue; left:200px; top:200px;}
75%  {background: green; left:0px; top:200px;}
100% {background: red; left:0px; top:0px;}
}
```

More Animation Examples

Example

```
div
{
/* Chrome, Safari, Opera */
-webkit-animation-name: myfirst;
-webkit-animation-duration: 5s;
-webkit-animation-timing-function: linear;
-webkit-animation-delay: 2s;
-webkit-animation-iteration-count: infinite;
-webkit-animation-direction: alternate;
-webkit-animation-play-state: running;
/* Standard syntax */
animation-name: myfirst;
animation-duration: 5s;
animation-timing-function: linear;
animation-delay: 2s;
animation-iteration-count: infinite;
animation-direction: alternate;
animation-play-state: running;
```

The same animation effect as the example above (except the animation-play-state property). However, here we are using the shorthand animation property:

Example

```
div
{
-webkit-animation: myfirst 5s linear 2s infinite alternate; /* Chrome, Safari, Opera */
animation: myfirst 5s linear 2s infinite alternate; /* Standard syntax */
}
```

CSS3 Animation Properties

The following table lists the @keyframes rule and all the animation properties:

Property	Description	CSS
@keyframes	Specifies the animation	3
Animation	A shorthand property for setting all the animation properties, except the animation-play-state and the animation-fill-mode property	3
animation-delay	Specifies when the animation will start	3
animation-direction	Specifies whether or not the animation should play in reverse on alternate cycles	3
animation-duration	Specifies how many seconds or milliseconds an animation takes to complete one cycle	3
animation-fill-mode	Specifies what styles will apply for the element when the animation is not playing (when it is finished, or when it has a "delay")	3
animation-iteration-count	Specifies the number of times an animation should be played	3
animation-name	Specifies the name of the @keyframes animation	3
animation-play-state	Specifies whether the animation is running or paused	3
animation-timing-function	Specifies the speed curve of the animation	3

CSS3 Multiple Columns

With CSS3, you can create multiple columns for laying out text - like in newspapers!
Now you will learn about the following multiple column properties:

- column-count
- column-gap
- column-rule

CSS3 Create Multiple Columns

The column-count property specifies the number of columns an element should be divided into:

Example

Divide the text in a div element into three columns:

```
div
{
-webkit-column-count:3; /* Chrome, Safari, Opera */
-moz-column-count:3; /* Firefox */
column-count:3;
}
```

CSS3 Specify the Gap Between Columns

The column-gap property specifies the gap between the columns:

Example

Specify a 40 pixels gap between the columns:

```
div
{
-webkit-column-gap:40px; /* Chrome, Safari, Opera */
-moz-column-gap:40px; /* Firefox */
column-gap:40px;
}
```

CSS3 Column Rules

The column-rule property sets the width, style, and color of the rule between columns.

Example

Specify the width, style and color of the rule between columns:

```
div
{
-webkit-column-rule:3px outset #ff00ff; /* Chrome, Safari, Opera */
-moz-column-rule:3px outset #ff00ff; /* Firefox */
column-rule:3px outset #ff00ff;
}
```

CSS3 Multiple Columns Properties

The following table lists all the multiple columns properties:

Property	Description	CSS
column-count	Specifies the number of columns an element should be divided into	3
column-fill	Specifies how to fill columns	3
column-gap	Specifies the gap between the columns	3
column-rule	A shorthand property for setting all the column-rule-* properties	3
column-rule-color	Specifies the color of the rule between columns	3
column-rule-style	Specifies the style of the rule between columns	3
column-rule-width	Specifies the width of the rule between columns	3
column-span	Specifies how many columns an element should span across	3
column-width	Specifies the width of the columns	3
columns	A shorthand property for setting column-width and column-count	3

CSS3 User Interface

In CSS3, some of the new user interface features are resizing elements, box sizing, and outlining. Now you will learn about the following user interface properties:

- resize
- box-sizing
- outline-offset

CSS3 Resizing

In CSS3, the resize property specifies whether or not an element should be resizable by the user.
This div element is resizable by the user (in Firefox, Chrome, and Safari).
The CSS code is as follows:

Example

Specify that a div element should be resizable by the user:

```
div
{
resize:both;
overflow:auto;
}
```

CSS3 Box Sizing

The box-sizing property allows you to define certain elements to fit an area in a certain way:

Example

Specify two bordered boxes side by side:

```
div
{
-moz-box-sizing:border-box; /* Firefox */
box-sizing:border-box;
width:50%;
float:left;
}
```

CSS3 Outline Offset

The outline-offset property offsets an outline, and draws it beyond the border edge.
Outlines differ from borders in two ways:
Outlines do not take up space
Outlines may be non-rectangular
This div has an outline 15px outside the border edge.
The CSS code is as follows:

Example

Specify an outline 15px outside the border edge:

```
div
{
border:2px solid black;
outline:2px solid red;
outline-offset:15px;
}
```

CSS3 User-interface Properties

The following table lists all the user-interface properties:

Property	Description	CSS
appearance	Allows you to make an element look like a standard user interface element	3
box-sizing	Allows you to define certain elements to fit an area in a certain way	3
icon	Provides the author the ability to style an element with an iconic equivalent	3
nav-down	Specifies where to navigate when using the arrow-down navigation key	3
nav-index	Specifies the tabbing order for an element	3
nav-left	Specifies where to navigate when using the arrow-left navigation key	3
nav-right	Specifies where to navigate when using the arrow-right navigation key	3
nav-up	Specifies where to navigate when using the arrow-up navigation key	3
outline-offset	Offsets an outline, and draws it beyond the border edge	3
resize	Specifies whether or not an element is resizable by the user	3

The "CSS" column indicates in which CSS version the property is defined (CSS1, CSS2, or CSS3).

Color Properties

Property	Description	CSS
color	Sets the color of text	1
opacity	Sets the opacity level for an element	3

Background and Border Properties

Property	Description	CSS
background	Sets all the background properties in one declaration	1
background-attachment	Sets whether a background image is fixed or scrolls with the rest of the page	1
background-color	Sets the background color of an element	1

background-image	Sets the background image for an element	1
background-position	Sets the starting position of a background image	1
background-repeat	Sets how a background image will be repeated	1
background-clip	Specifies the painting area of the background	3
background-origin	Specifies the positioning area of the background images	3
background-size	Specifies the size of the background images	3
border	Sets all the border properties in one declaration	1
border-bottom	Sets all the bottom border properties in one declaration	1
border-bottom-color	Sets the color of the bottom border	1
border-bottom-left-radius	Defines the shape of the border of the bottom-left corner	3
border-bottom-right-radius	Defines the shape of the border of the bottom-right corner	3
border-bottom-style	Sets the style of the bottom border	1
border-bottom-width	Sets the width of the bottom border	1
border-color	Sets the color of the four borders	1
border-image	A shorthand property for setting all the border-image-* properties	3
border-image-outset	Specifies the amount by which the border image area extends beyond the border box	3
border-image-repeat	Specifies whether the image-border should be repeated, rounded or stretched	3
border-image-slice	Specifies the inward offsets of the image-border	3
border-image-source	Specifies an image to be used as a border	3
border-image-width	Specifies the widths of the image-border	3
border-left	Sets all the left border properties in one declaration	1
border-left-color	Sets the color of the left border	1
border-left-style	Sets the style of the left border	1
border-left-width	Sets the width of the left border	1
border-radius	A shorthand property for setting all the four border-*-radius properties	3
border-right	Sets all the right border properties in one declaration	1
border-right-color	Sets the color of the right border	1
border-right-style	Sets the style of the right border	1
border-right-width	Sets the width of the right border	1
border-style	Sets the style of the four borders	1
border-top	Sets all the top border properties in one declaration	1
border-top-color	Sets the color of the top border	1
border-top-left-radius	Defines the shape of the border of the top-left corner	3
border-top-right-radius	Defines the shape of the border of the top-right corner	3
border-top-style	Sets the style of the top border	1
border-top-width	Sets the width of the top border	1
border-width	Sets the width of the four borders	1

box-decoration-break	Sets the behaviour of the background and border of an element at page-break, or, for in-line elements, at line-break.	3
box-shadow	Attaches one or more drop-shadows to the box	3

Basic Box Properties

Property	Description	CSS
bottom	Specifies the bottom position of a positioned element	2
clear	Specifies which sides of an element where other floating elements are not allowed	1
clip	Clips an absolutely positioned element	2
display	Specifies how a certain HTML element should be displayed	1
float	Specifies whether or not a box should float	1
height	Sets the height of an element	1
left	Specifies the left position of a positioned element	2
overflow	Specifies what happens if content overflows an element's box	2
overflow-x	Specifies whether or not to clip the left/right edges of the content, if it overflows the element's content area	3
overflow-y	Specifies whether or not to clip the top/bottom edges of the content, if it overflows the element's content area	3
padding	Sets all the padding properties in one declaration	1
padding-bottom	Sets the bottom padding of an element	1
padding-left	Sets the left padding of an element	1
padding-right	Sets the right padding of an element	1
padding-top	Sets the top padding of an element	1
position	Specifies the type of positioning method used for an element (static, relative, absolute or fixed)	2
right	Specifies the right position of a positioned element	2
top	Specifies the top position of a positioned element	2
visibility	Specifies whether or not an element is visible	2
width	Sets the width of an element	1
vertical-align	Sets the vertical alignment of an element	1
z-index	Sets the stack order of a positioned element	2

Flexible Box Layout

Property	Description	CSS
align-content	Specifies the alignment between the lines inside a flexible container when the items do not use all available space.	3
align-items	Specifies the alignment for items inside a flexible container.	3
align-self	Specifies the alignment for selected items inside a flexible container.	3
display	Specifies how a certain HTML element should be displayed	1
flex	Specifies the length of the item, relative to the rest	3

flex-basis	Specifies the initial length of a flexible item	3
flex-direction	Specifies the direction of the flexible items	3
flex-flow	A shorthand property for the flex-direction and the flex-wrap properties	3
flex-grow	Specifies how much the item will grow relative to the rest	3
flex-shrink	Specifies how the item will shrink relative to the rest	3
flex-wrap	Specifies whether the flexible items should wrap or not	3
justify-content	Specifies the alignment between the items inside a flexible container when the items do not use all available space	3
margin	Sets all the margin properties in one declaration	1
margin-bottom	Sets the bottom margin of an element	1
margin-left	Sets the left margin of an element	1
margin-right	Sets the right margin of an element	1
margin-top	Sets the top margin of an element	1
max-height	Sets the maximum height of an element	2
max-width	Sets the maximum width of an element	2
min-height	Sets the minimum height of an element	2
min-width	Sets the minimum width of an element	2
order	Sets the order of the flexible item, relative to the rest	3

Text Properties

Property	Description	CSS
hanging-punctuation	Specifies whether a punctuation character may be placed outside the line box	3
hyphens	Sets how to split words to improve the layout of paragraphs	3
letter-spacing	Increases or decreases the space between characters in a text	1
line-break		3
line-height	Sets the line height	1
overflow-wrap		3
tab-size	Specifies the length of the tab-character	3
text-align	Specifies the horizontal alignment of text	1
text-align-last	Describes how the last line of a block or a line right before a forced line break is aligned when text-align is "justify"	3
text-indent	Specifies the indentation of the first line in a text-block	1
text-justify	Specifies the justification method used when text-align is "justify"	3
text-transform	Controls the capitalization of text	1
white-space	Specifies how white-space inside an element is handled	1
word-break	Specifies line breaking rules for non-CJK scripts	3
word-spacing	Increases or decreases the space between words in a text	1
word-wrap	Allows long, unbreakable words to be broken and wrap to the next line	3

Text Decoration Properties

Property	Description	CSS
text-decoration	Specifies the decoration added to text	1
text-decoration-color	Specifies the color of the text-decoration	3
text-decoration-line	Specifies the type of line in a text-decoration	3
text-decoration-style	Specifies the style of the line in a text decoration	3
text-shadow	Adds shadow to text	3
text-underline-position		3

Font Properties

Property	Description	CSS
font	Sets all the font properties in one declaration	1
font-family	Specifies the font family for text	1
font-feature-setting		3
@font-feature-values		3
font-kerning		3
font-language-override		3
font-synthesis		3
font-variant-alternates		3
font-variant-caps		3
font-variant-east-asian		3
font-variant-ligatures		3
font-variant-numeric		3
font-variant-position		3
font-size	Specifies the font size of text	1
font-style	Specifies the font style for text	1
font-variant	Specifies whether or not a text should be displayed in a small-caps font	1
font-weight	Specifies the weight of a font	1
@font-face	A rule that allows websites to download and use fonts other than the "web-safe" fonts	3
font-size-adjust	Preserves the readability of text when font fallback occurs	3
font-stretch	Selects a normal, condensed, or expanded face from a font family	3

Writing Modes Properties

Property	Description	CSS
direction	Specifies the text direction/writing direction	2
text-orientation		3
text-combine-horizontal		3
unicode-bz-indexidi	Used together with the direction property to set or return whether the text should be overridden to support multiple languages in the same	2

	document	
writing-mode		3

Table Properties

Property	Description	CSS
border-collapse	Specifies whether or not table borders should be collapsed	2
border-spacing	Specifies the distance between the borders of adjacent cells	2
caption-side	Specifies the placement of a table caption	2
empty-cells	Specifies whether or not to display borders and background on empty cells in a table	2
table-layout	Sets the layout algorithm to be used for a table	2

Lists and Counters Properties

Property	Description	CSS
counter-increment	Increments one or more counters	2
counter-reset	Creates or resets one or more counters	2
list-style	Sets all the properties for a list in one declaration	1
list-style-image	Specifies an image as the list-item marker	1
list-style-position	Specifies if the list-item markers should appear inside or outside the content flow	1
list-style-type	Specifies the type of list-item marker	1

Animation Properties

Property	Description	CSS
@keyframes	Specifies the animation	3
animation-delay	Specifies when the animation will start	3
animation	: A shorthand property for all the animation properties below, except the animation-play-state property	3
animation-duration	Specifies how many seconds or milliseconds an animation takes to complete one cycle	3
animation-fill-mode	Specifies what values are applied by the animation outside the time it is executing	3
animation-iteration-count	Specifies the number of times an animation should be played	3
animation-name	Specifies a name for the @keyframes animation	3
animation-timing-function	Specifies the speed curve of the animation	3
animation-play-state	Specifies whether the animation is running or paused	3

Transform Properties

Property	Description	CSS
backface-visibility	Defines whether or not an element should be visible when not facing the screen	3

perspective	Specifies the perspective on how 3D elements are viewed	3
perspective-origin	Specifies the bottom position of 3D elements	3
transform	Applies a 2D or 3D transformation to an element	3
transform-origin	Allows you to change the position on transformed elements	3
transform-style	Specifies how nested elements are rendered in 3D space	3

Transitions Properties

Property	Description	CSS
transition	A shorthand property for setting the four transition properties	3
transition-property	Specifies the name of the CSS property the transition effect is for	3
transition-duration	Specifies how many seconds or milliseconds a transition effect takes to complete	3
transition-timing-function	Specifies the speed curve of the transition effect	3
transition-delay	Specifies when the transition effect will start	3

Basic User Interface Properties

Property	Description	CSS
box-sizing	Allows you to define certain elements to fit an area in a certain way	3
content	Used with the :before and :after pseudo-elements, to insert generated content	2
cursor	Specifies the type of cursor to be displayed	2
icon	Provides the author the ability to style an element with an iconic equivalent	3
ime-mode		3
nav-down	Specifies where to navigate when using the arrow-down navigation key	3
nav-index	Specifies the tabbing order for an element	3
nav-left	Specifies where to navigate when using the arrow-left navigation key	3
nav-right	Specifies where to navigate when using the arrow-right navigation key	3
nav-up	Specifies where to navigate when using the arrow-up navigation key	3
outline	Sets all the outline properties in one declaration	2
outline-color	Sets the color of an outline	2
outline-offset	Offsets an outline, and draws it beyond the border edge	3
outline-style	Sets the style of an outline	2
outline-width	Sets the width of an outline	2
resize	Specifies whether or not an element is resizable by the user	3
text-overflow	Specifies what should happen when text overflows the containing element	3

Multi-column Layout Properties

Property	Description	CSS
break-after		3
break-before		3
break-inside		3
column-count	Specifies the number of columns an element should be divided into	3
column-fill	Specifies how to fill columns	3
column-gap	Specifies the gap between the columns	3
column-rule	A shorthand property for setting all the column-rule-* properties	3
column-rule-color	Specifies the color of the rule between columns	3
column-rule-style	Specifies the style of the rule between columns	3
column-rule-width	Specifies the width of the rule between columns	3
column-span	Specifies how many columns an element should span across	3
column-width	Specifies the width of the columns	3
columns	A shorthand property for setting column-width and column-count	3
widows	Sets the minimum number of lines that must be left at the top of a page when a page break occurs inside an element	2

Paged Media

Property	Description	CSS
orphans	Sets the minimum number of lines that must be left at the bottom of a page when a page break occurs inside an element	2
page-break-after	Sets the page-breaking behavior after an element	2
page-break-before	Sets the page-breaking behavior before an element	2
page-break-inside	Sets the page-breaking behavior inside an element	2

Generated Content for Paged Media

Property	Description	CSS
marks	Adds crop and/or cross marks to the document	3
quotes	Sets the type of quotation marks for embedded quotations	2

Filter Effects Properties

Property	Description	CSS
filter		3

Image Values and Replaced Content

Property	Description	CSS
image-orientation	Specifies a rotation in the right or clockwise direction that a user agent applies to an image	3

image-rendering		3
image-resolution		3
object-fit		3
object-position		3

Masking Properties

Property	Description	CSS
mask		3
mask-type		3

Speech Properties

Property	Description	CSS
mark	A shorthand property for setting the mark-before and mark-after properties	3
mark-after	Allows named markers to be attached to the audio stream	3
mark-before	Allows named markers to be attached to the audio stream	3
phonemes	Specifies a phonetic pronunciation for the text contained by the corresponding element	3
rest	A shorthand property for setting the rest-before and rest-after properties	3
rest-after	Specifies a rest or prosodic boundary to be observed after speaking an element's content	3
rest-before	Specifies a rest or prosodic boundary to be observed before speaking an element's content	3
voice-balance	Specifies the balance between left and right channels	3
voice-duration	Specifies how long it should take to render the selected element's content	3
voice-pitch	Specifies the average pitch (a frequency) of the speaking voice	3
voice-pitch-range	Specifies variation in average pitch	3
voice-rate	Controls the speaking rate	3
voice-stress	Indicates the strength of emphasis to be applied	3
voice-volume	Refers to the amplitude of the waveform output by the speech synthesises	3

Marquee Properties

Property	Description	CSS
marquee-direction	Sets the direction of the moving content	3
marquee-play-count	Sets how many times the content move	3
marquee-speed	Sets how fast the content scrolls	3
marquee-style	Sets the style of the moving content	3

CSS Selectors

In CSS, selectors are patterns used to select the element(s) you want to style.
Use our CSS Selector Tester to demonstrate the different selectors.
The "CSS" column indicates in which CSS version the property is defined (CSS1, CSS2, or CSS3).

Selector	Example	Example description	CSS
.class	.intro	Selects all elements with class="intro"	1
#id	#firstname	Selects the element with id="firstname"	1
*	*	Selects all elements	2
element	p	Selects all <p> elements	1
element,element	div,p	Selects all elements and all <p> elements	1
element element	div p	Selects all <p> elements inside elements	1
element>element	div>p	Selects all <p> elements where the parent is a element	2
element+element	div+p	Selects all <p> elements that are placed immediately after elements	2
element1~element2	p~ul	Selects every element that are preceded by a <p> element	3
[*attribute*]	[target]	Selects all elements with a target attribute	2
[*attribute=value*]	[target=_blank]	Selects all elements with target="_blank"	2
[*attribute~=value*]	[title~=flower]	Selects all elements with a title attribute containing the word "flower"	2
[*attribute|=value*]	[lang|=en]	Selects all elements with a lang attribute value starting with "en"	2
[*attribute^=value*]	a[src^="https"]	Selects every <a> element whose src attribute value begins with "https"	3
[*attribute$=value*]	a[src$=".pdf"]	Selects every <a> element whose src attribute value ends with ".pdf"	3
[*attribute**=*value*]	a[src*="w3schools"]	Selects every <a> element whose src attribute value contains the substring "w3schools"	3
:active	a:active	Selects the active link	1
::after	p::after	Insert content after every <p> element	2
::before	p::before	Insert content before the content of every <p> element	2
:checked	input:checked	Selects every checked <input> element	3
:disabled	input:disabled	Selects every disabled <input> element	3
:empty	p:empty	Selects every <p> element that has no children (including text nodes)	3
:enabled	input:enabled	Selects every enabled <input> element	3
:first-child	p:first-child	Selects every <p> element that is the first child of its parent	2
::first-letter	p::first-letter	Selects the first letter of every <p> element	1
::first-line	p::first-line	Selects the first line of every <p> element	1
:first-of-type	p:first-of-type	Selects every <p> element that is the first <p> element	3

		of its parent	
:focus	input:focus	Selects the input element which has focus	2
:hover	a:hover	Selects links on mouse over	1
:in-range	input:in-range	Selects input elements with a value within a specified range	3
:invalid	input:invalid	Selects all input elemets with an invalid value	3
:lang(*language*)	p:lang(it)	Selects every <p> element with a lang attribute equal to "it" (Italian)	2
:last-child	p:last-child	Selects every <p> element that is the last child of its parent	3
:last-of-type	p:last-of-type	Selects every <p> element that is the last <p> element of its parent	3
:link	a:link	Selects all unvisited links	1
:not(*selector*)	:not(p)	Selects every element that is not a <p> element	3
:nth-child(*n*)	p:nth-child(2)	Selects every <p> element that is the second child of its parent	3
:nth-last-child(*n*)	p:nth-last-child(2)	Selects every <p> element that is the second child of its parent, counting from the last child	3
:nth-last-of-type(*n*)	p:nth-last-of-type(2)	Selects every <p> element that is the second <p> element of its parent, counting from the last child	3
:nth-of-type(*n*)	p:nth-of-type(2)	Selects every <p> element that is the second <p> element of its parent	3
:only-of-type	p:only-of-type	Selects every <p> element that is the only <p> element of its parent	3
:only-child	p:only-child	Selects every <p> element that is the only child of its parent	3
:optional	input:optional	Selects input elements with no "required" attribute	3
:out-of-range	input:out-of-range	Selects input elements with a value outside a specified range	3
:read-only	input:read-only	Selects input elements with the "readonly" attribute specified	3
:read-write	input:read-write	Selects input elements with the "readonly" attribute NOT specified	3
:required	input:required	Selects input elements with the "required" attribute specified	3
:root	:root	Selects the document's root element	3
::selection	::selection	Selects the portion of an element that is selected by a user	
:target	#news:target	Selects the current active #news element (clicked on a URL containing that anchor name)	3
:valid	input:valid	Selects all input elemets with a valid value	3
:visited	a:visited	Selects all visited links	1

CSS Colors

Colors in CSS can be specified by the following methods:

- Hexadecimal colors
- RGB colors
- RGBA colors
- HSL colors
- HSLA colors
- Predefined/Cross-browser color names

Hexadecimal Colors

Hexadecimal color values are supported in all major browsers.
A hexadecimal color is specified with: #RRGGBB, where the RR (red), GG (green) and BB (blue) hexadecimal integers specify the components of the color. All values must be between 0 and FF.
For example, the #0000ff value is rendered as blue, because the blue component is set to its highest value (ff) and the others are set to 0.

Example

Define different HEX colors:

```
#p1 {background-color:#ff0000;} / * red */
#p2 {background-color:#00ff00;} / * green */
#p3 {background-color:#0000ff;} / * blue */
```

RGB Colors

RGB color values are supported in all major browsers.
An RGB color value is specified with: rgb(red, green, blue). Each parameter (red, green, and blue) defines the intensity of the color and can be an integer between 0 and 255 or a percentage value (from 0% to 100%).
For example, the rgb(0,0,255) value is rendered as blue, because the blue parameter is set to its highest value (255) and the others are set to 0.
Also, the following values define equal color: rgb(0,0,255) and rgb(0%,0%,100%).

Example

Define different RGB colors:

```
#p1 {background-color:rgb(255,0,0);} / * red */
#p2 {background-color:rgb(0,255,0);} / * green */
#p3 {background-color:rgb(0,0,255);} / * blue */
```

RGBA Colors

RGBA color values are supported in IE9+, Firefox 3+, Chrome, Safari, and in Opera 10+.
RGBA color values are an extension of RGB color values with an alpha channel - which specifies the opacity of the object.
An RGBA color value is specified with: rgba(red, green, blue, alpha). The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (fully opaque).

Example

Define different RGB colors with opacity:

```
#p1 {background-color:rgba(255,0,0,0.3);} / * red with opacity */
#p2 {background-color:rgba(0,255,0,0.3);} / * green with opacity */
#p3 {background-color:rgba(0,0,255,0.3);} / * blue with opacity */
```

HSL Colors

HSL color values are supported in IE9+, Firefox, Chrome, Safari, and in Opera 10+.
HSL stands for hue, saturation, and lightness - and represents a cylindrical-coordinate representation of colors.
An HSL color value is specified with: hsl(hue, saturation, lightness).
Hue is a degree on the color wheel (from 0 to 360) - 0 (or 360) is red, 120 is green, 240 is blue.
Saturation is a percentage value; 0% means a shade of gray and 100% is the full color. Lightness is also a percentage; 0% is black, 100% is white.

Example

Define different HSL colors:

```
#p1 {background-color:hsl(120,100%,50%);} / * green */
#p2 {background-color:hsl(120,100%,75%);} / * light green */
#p3 {background-color:hsl(120,100%,25%);} / * dark green */
#p4 {background-color:hsl(120,60%,70%);} / * pastel green */
```

HSLA Colors

HSLA color values are supported in IE9+, Firefox 3+, Chrome, Safari, and in Opera 10+.
HSLA color values are an extension of HSL color values with an alpha channel - which specifies the opacity of the object.
An HSLA color value is specified with: hsla(hue, saturation, lightness, alpha), where the alpha parameter defines the opacity. The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (fully opaque).

Example

Define different HSL colors with opacity:

```
#p1 {background-color:hsla(120,100%,50%,0.3);} / * green with opacity */
#p2 {background-color:hsla(120,100%,75%,0.3);} / * light green with opacity */
#p3 {background-color:hsla(120,100%,25%,0.3);} / * dark green with opacity */
#p4 {background-color:hsla(120,60%,70%,0.3);} / * pastel green with opacity */
```

Web-standards and validation

W3C is the World Wide Web Consortium, which is an independent organization that manages code standards on the web (e.g. HTML, CSS, XML and others). Microsoft, The Mozilla Foundation and many others are a part of W3C and agree upon the future developments of the standards.
If you have been working just a bit with web design, you probably know that there can be a big differences in how a webpage is presented across different browsers. It can be very frustrating and time-consuming to create a webpage which can be viewed in Mozilla, Internet Explorer, Opera and all the rest of the existing browsers.

The idea of having standards is to agree upon a common denominator on how to use web technologies. This means that by observing the standards, a web developer has a certainty that what he or she does will work in a more appropriate manner across different platforms.
We therefore recommend that you back up the work carried out by the W3C and validate your CSS in order to observe the standard.

CSS validator

To make it easier to observe the CSS standard, W3C has made a so-called validator which reads your stylesheet and returns a status listing errors and warnings, if your CSS does not validate.
To make it easier for you to validate your stylesheet, you can do it directly from this webpage. Simply replace the URL with the URL of your stylesheet below and click to validate. You will then be informed by the W3C site if there are any errors found.

Predefined/Cross-browser Color Names as in HTML