Module I CSS Part I

CSS

Cascading Style Sheets (CSS) is a style sheet language used for describing the look and formatting of a document written in a markup language. CSS was developed by theW3C. 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. The most often used to style web pages and interfaces written in HTML and XHTML, the language can be applied to any kind of XML document, including plain XML, SVG and XUL (Scalable Vector Graphics and XML User Interface Language).

A new feature being added to HTML that gives both Web site developers and users more control over how pages are displayed. With CSS, designers and users can create style sheets that define how different elements, such as headers and links, appear. These style sheets can then be applied to any Web page. The term cascading derives from the fact that multiple style sheets can be applied to the same Web page.

Internal Style Sheet

First we will explore the internal method. This way we are simply placing the CSS code within the <head></head> tags of each HTML or XHTML file we want to style with the CSS. The format for this is shown in the example below.

```
<head>
<title>--------- <title>
<style type=”text/css”>
CSS Content Goes Here
</style>
</head >
```

With this method each HTML or XHTML file contains the CSS code needed to style the page. Meaning that any changes we want to make to one page will have to be made to all. This method can be good if we need to style only one page, or if we want different pages to have varying styles.

External Style Sheet

Next we will explore the external method. An external CSS file can be created with any text or HTML editor such as “Notepad” or “Dreamweaver”. A CSS file contains no HTML or XHTML, only CSS. We simply save it with the **.**css file extension. We can link to the file externally by placing one of the following links in the head section of every HTML or XHTML file we want to style with the CSS file.

<link rel=”stylesheet” type=”text/css” href=“Path To stylesheet.css” />

Or we can also use the @import method as shown below

<style type=”text/css”>@import url(Path To stylesheet.css)</style>

Either of these methods are achieved by placing one or the other in the head section as shown in example below.

```
<head><title><title>
<link rel=”stylesheet” type=”text/css” href=”style.css” />
</head>
<body>
```

OR

<head><title></title>
<style type="text/css"> @import url(Path To stylesheet.css)</style>
</head><body>

By using an external style sheet, all of our HTML or XHTML files links to one CSS file in order to style the pages. This means, that if we need to alter the design of all our pages, we only need to edit one **.**css file to make global changes to our entire website. Here are a few reasons this is better (Advantages).

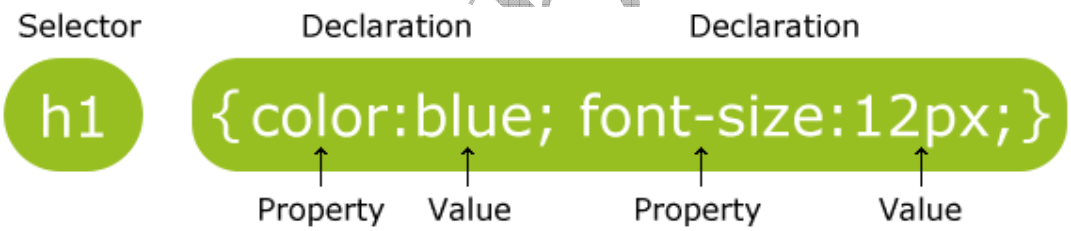
- Easier Maintenance
- Reduced File Size
- Reduced Bandwidth
- Improved Flexibility

CSS Syntax

A CSS rule set consists of a selector and a declaration block: The selector points to the HTML element we want to style. The declaration block contains one or more declarations separated by semicolons. A CSS statement consists of only 3 parts.

Selector { property: value }

The selector is the HTML or XHTML element that we want to style. The property is the actual property title, and the value is the style that we apply to that property.



The selector points to the HTML element we 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 braces:

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

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

Example

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

- h1 {
  color: #36CFFF;
  }

- body {
  background: #eeeeee;
  font-family: “Trebuchet MS”, Verdana, Arial, serif;
  }

Example


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

<p>Hello World!</p>
<p>This paragraph is styled with CSS.</p>

</body>
</html>
```

Comment tags

Comments can be used to explain why we added certain selectors within our css file. So as to help others who may see our file, or to help us remember what we are thinking at a later date. Comments are ignored by browsers. We can add comments that will be ignored by browsers in the following manner.

/* This is a comment */

We will note that it begins with a / (forward slash) and than an * (asterisks) then the comment, then the closing tag which is just backward from the opening tag * (asterisks) then the / (forward slash).

Example

- p {
  color: red;
  /* This is a single-line comment */
  text-align: center;
  }

  /* This is
  a multi-line
  comment */

CSS Classes

The class selector allows us to style items within the same HTML or XHTML element differently. Similar to what we mentioned in the introduction about inline styles. Except with classes the style can be overwritten by changing out style sheets. We can use the same class

selector again and again within an HTML or XHTML file. To put it more simply, this sentence we are reading is defined in my CSS file with the following.

```
p {
  font-size: small;
  color: #333333
}
```

Pretty simple, but lets say that we wanted to change the word “sentence” to green bold text, while leaving the rest of the sentence untouched. I would do the following to my HTML or XHTML file.

```
<p>
To put it more simply, this <span class=”greenboldtext”>sentence</span> we are reading is styled in my CSS file by the following.
</p>
```

Then in my CSS file we would add this style selector:

```
.greenboldtext{
  font-size: small;
  color: #008080;
  font-weight: bold;
}
```

The final result would look like the following:

To put it more simply, this sentence we are reading is styled in my CSS file by the following. Please note that a class selector begins with a (.) period. The reason we named it “greenboldtext” is for example purposes, we can name it whatever we want. Though I do encourage us to use selector names that are descriptive. We can reuse the “greenboldtext” class as many times as we want.

CSS IDs

IDs are similar to classes, except once a specific id has been declared it cannot be used again within the same HTML or XHTML file. We generally use IDs to style the layout elements of a page that will only be needed once, whereas we use classes to style text and such that may be declared multiple times. Do NOT start an ID name with a number! The main container for this page is defined by the following.

```
<div id=”container”>
everything within my document is inside this division.
</div>
```

We have chosen the id selector for the “container” division over a class, because I only need to use it one time within this file. Then in our CSS file we have the following:

```
#container{
  width: 80%;
  margin: auto;
  padding: 20px;
  border: 1px solid #666;
  background: #ffffff;
}
```

We will notice that the id selector begins with a (#) number sign instead of a (.) period, as the class selector does.

Example


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

<p id="para1">Hello World!</p>
<p>This paragraph is not affected by the style.</p>

</body>
</html>
```

In the CSS, a class selector is a name preceded by a full stop (“.”) and an ID selector is a name preceded by a hash character (“#”).
So the CSS might look something like:

- #top {
  background-color: #ccc;
  padding: 20px
  }

- .intro {
  color: red;
  font-weight: bold;
  }

The HTML refers to the CSS by using the attributes id and class. It could look something like this:

```
<div id="top">
<h1>Chocolate curry</h1>
<p class="intro">This is my recipe for making curry purely with chocolate</p>
<p class="intro">Mmm mm mmmmm</p>
</div>
```

The difference between an ID and a class is that an ID can be used to identify one element, whereas a class can be used to identify more than one. We can also apply a selector to a specific HTML element by simply stating the HTML selector first, so p.jam { /* whatever */ } will only be applied to paragraph elements that have the class “jam”.

Background

We can style the background of an element in one declaration with the background property.

background: color name or RGB value, url(path_to_image), top, left, no-repeat, fixed. The values are attachment, color, image, and position, repeat. OR we can set each property individually.

Background Color

We can specifically declare a color for the background of an element using the background-color property.

background-color: value;

The values are color name, hexadecimal number, RGB color code, transparent

Background Image

We can set an image for the background of an element using the background-image property.

background-image: url(path_to_image);

The values are url, none

CSS Text Properties

Inherited: Yes

TextColor

We can set the color of text with the following:

color: value;

Possible values are colorname – example:(red, black…)
hexadecimal number – example:(#ff0000, #000000)
RGB color code – example:(rgb(255, 0, 0), rgb(0, 0, 0))

Text Align

We can align text with the following:

text-align: value;

Possible values are left, right, center, justify

Examples:

This text is aligned left.
This text is aligned in the center.
This text is aligned right.
This text is justified.

Text Decoration

We can decorate text with the following:

text-decoration: value;

Possible values are: none, underline, overline, line through, blink

Examples:

This text is underlined.
This text is overlined.
This text has a line through it.
This text is blinking (not in internet explorer).

Text Transform

We can control the size of letters in an HTML or XHTML element with the following:

text-transform: value;

Possible values are: , none, capitalize, lowercase, uppercase

Examples:

This First Letter In Each Word Is Capitalized, Though It Is Not In My File.
THIS TEXT IS ALL UPPERCASE, THOUGH IT IS ALL LOWERCASE IN MY FILE.
this text is all lowercase. though it is all uppercase in my file.

Text Indent

We can indent the first line of text in an HTML or XHTML element with the following:

text-indent: value;

Possible values are: length, percentage

Examples:

This text is indented 10px pixels.

CSS Font Properties

Inherited: Yes

Font

The font property can set the style, weight, variant, size, line height and font:

font: italic bold normal small/1.4em Verdana, sans-serif;

The above would set the text of an element to an italic style a bold weight a normal variant a relative size a line height of 1.4em and the font to Verdana or another sans-serif typeface.

Font-Family

We can set what font will be displayed in an element with the ***font-family*** property.
There are 2 choices for values: family-name, generic family.

If we set a family name it is best to also add the generic family at the end. As this is a prioritized list. So if the user does not have the specified font name it will use the same generic family. (see below)

font-family: Verdana, sans-serif;

Font Size

We can set the size of the text used in an element by using the font-size property.

font-size: value;

There are a lot of choices for values: xx-large, x-large, larger, large, medium, small, smaller, x-small, xx-small, length, % (percent)

There is quite a bit to learn about font sizes with CSS so, we are not even going to try to explain it. Actually there are already some great resources on how to size our text.

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, we 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 we 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 us full control over the text size:

Example

```
h1 {
   font-size: 40px;
}

h2 {
   font-size: 30px;
}

p {
   font-size: 14px;
}
```

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
   font-size: 40px;
}

h2 {
   font-size: 30px;
}

p {
   font-size: 14px;
}
</style>
</head>
<body>

<h1>This is heading 1</h1>
<h2>This is heading 2</h2>
<p>This is a paragraph.</p>
<p>Specifying the font-size in px allows Internet Explorer 9, Firefox, Chrome, Opera, and Safari to resize the text.</p>
<p><b>Note:</b> This example does not work in IE, prior version 9.</p>

</body>
```

```
</html>
```

The example above allows Internet Explorer 9, Firefox, Chrome, Opera, and Safari to resize the text.
Note: *The example above does not work in IE, prior version 9.*
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 */
}
```

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
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 */
}
</style>
</head>
<body>

<h1>This is heading 1</h1>
<h2>This is heading 2</h2>
```

This is a paragraph.
Specifying the font-size in em allows all major browsers to resize the text.
Unfortunately, there is still a problem with older versions of IE. When resizing the text, it becomes larger/smaller than it should.

</body>
</html>

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;
}
```

examples

```
<!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</h1>
<h2>This is heading 2</h2>
<p>This is a paragraph.</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>
```

CSS LISTS

Inherited: Yes

List Style

We can control the appearance of ordered and unordered lists in one declaration with the list-style property. The Values are image, position, type.

List Style Image

We can use an image for the bullet of unordered lists with the list-style property. If we use an image, it is a good idea to declare the list-style-type also in case the user has images turned off.

List Style Position

We can control the position of ordered and unordered lists with the liststyle-position property.

border-top-style: value;
border-top-width: value;
list-style: value value;
list-style-image: url(path_to_image.gif, jpg or png);
list-style-position: value;

The values are inside, outside.

List Style Type

We can control the type of bullet ordered and unordered lists use with the list-style-type property. The values are: disc, circle, square, decimal, lower-roman, upper-roman, lower-alpha, upper-alpha, none.

CSS List

The CSS list properties allow us 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

In HTML, there are two types of lists:

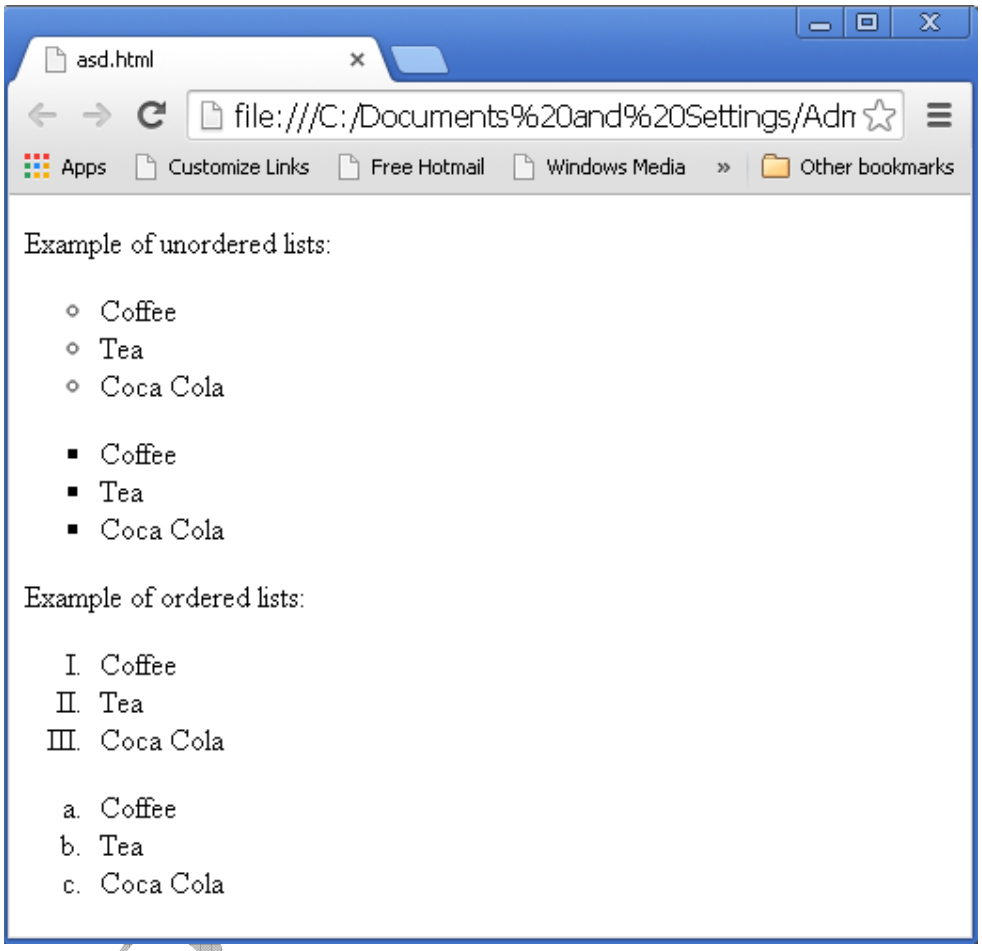
- unordered lists - the list items are marked with bullets
- ordered lists - the list items are marked with numbers or letters

With CSS, lists can be styled further, and images can be used as the list item marker. Some of the values are for unordered lists, and some for ordered lists. The type of list item marker is specified with the list-style-type property:

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
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;
}
</style>
</head>
<body>

<p>Example of unordered lists:</p>
<ul class="a">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ul>

<ul class="b">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ul>

<p>Example of ordered lists:</p>
<ol class="c">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ol>

<ol class="d">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ol>

</body>
</html>
```



An Image as The List Item Marker

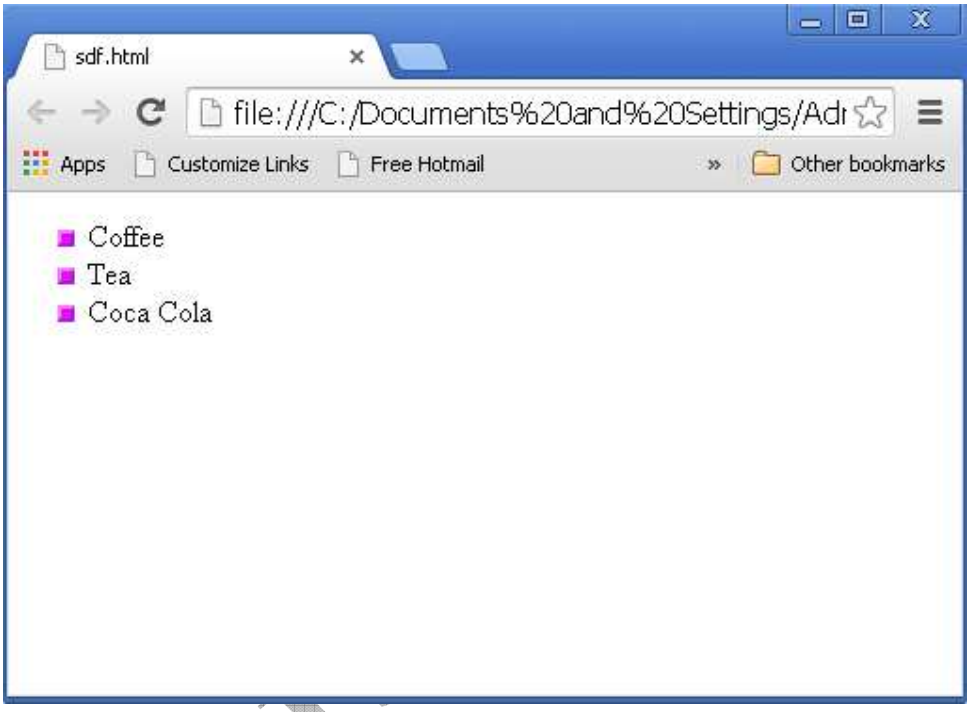
To specify an image as the list item marker, use the list-style-image property: This example does not display equally in all browsers. IE and Opera will display the image-marker a little bit higher than Firefox, Chrome, and Safari. Example explained:

Example

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

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
ul {
   list-style-image:
url('sqpurple.gif');
}
</style>
</head>
<body>

<ul>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ul>

</body>
</html>
```



CSS TABLES

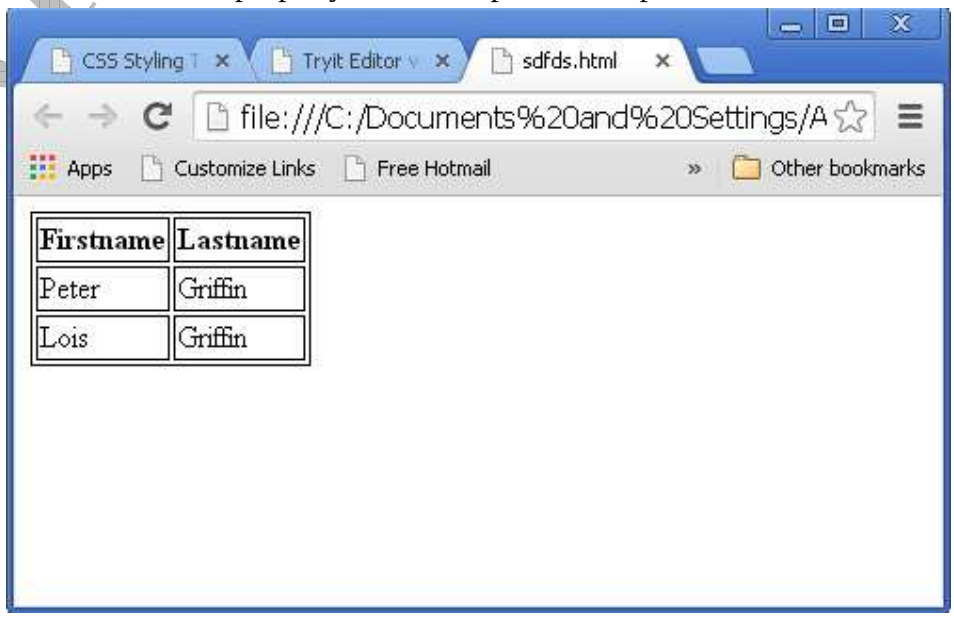
Table Borders

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

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

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
table, th, td {
 border: 1px solid blue;
}
</style></head><body>
<table>
  <tr>
    <th>Firstname</th>
    <th>Lastname</th>
  </tr>
```



```
  <tr>
    <td>Peter</td>
    <td>Griffin</td>
  </tr>
  <tr>
    <td>Lois</td>
    <td>Griffin</td>
  </tr>
</table></body>
</html>
```

Border Style

The border-style property specifies what kind of border to display. None of the border properties will have ANY effect unless the **border-style** property is set. The values are: none, dotted, dashed, solid, double, groove, ridge, inset and outset.

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

Example 1:

```
table {
   border-style: solid;
   border-width: 5px;
}

table {
   border-style: dotted;
   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"

We can also set the border color to "transparent". The "border-color" property does not work if it is used alone. Use the "border-style" property to set the borders first.

Example 2:

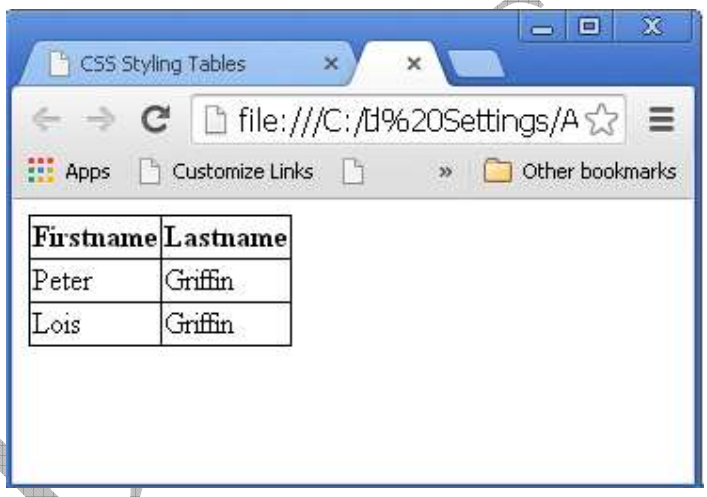
```
table {
   border-style: ridge;
   border-color: red;
}

table {
   border-style: double;
   border-color: #98bf21;
}
```

Border Individual Sides

In CSS it is possible to specify different borders for different sides: the values are left, right, top and bottom.

Example 3:

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



Collapse Borders

Notice that the table in the example above has double borders. This is because both the table and the th or td elements have separate borders. To display a single border for the table, use the border-collapse property. The border-collapse property sets whether the table borders are collapsed into a single border or separated:

```
table {
   border-collapse: collapse;
}
```

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

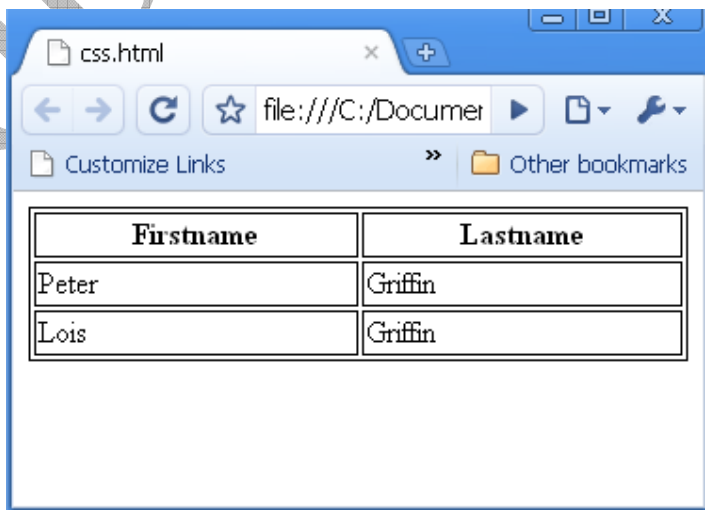


Table Width and Table Height

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;
}
```

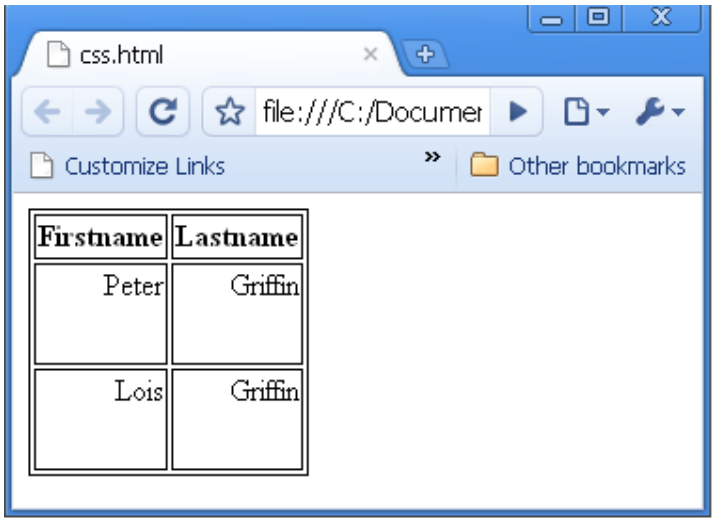


Table Text Alignment

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:

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

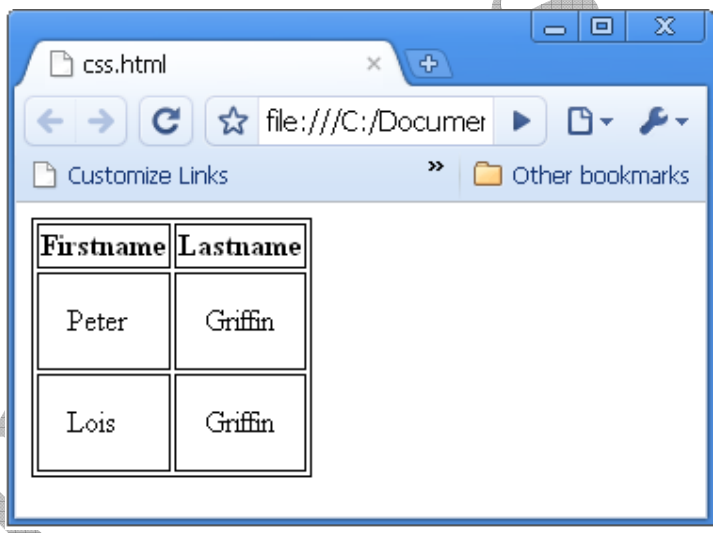


Table Padding

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;
}
```

CSS POSITIONING

The CSS positioning properties allow us 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.

1. **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.

2. Fixed Positioning

An element with fixed position is positioned relative to the browser window. It will not move even if the window is scrolled: IE7 and IE8 support the fixed value only if a !DOCTYPE is specified

Example

```
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.

3. 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.

4. 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).
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. If two positioned elements overlap without a z-index specified, the element positioned last in the HTML code will be shown on top.

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. The 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 we 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;
}
```

HORIZONTAL ALIGN

In CSS, several properties are used to align elements horizontally.

Center Aligning Using The Margin Property

Block elements can be center-aligned by setting the left and right margins to "auto". 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. Center-aligning has no effect if the width is 100%. For Example

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

Left and Right Aligning Using the position Property

One method of aligning elements is to use absolute positioning: for example

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

Left And Right Aligning Using The Float Property

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

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

IMAGE GALLERY

CSS can be used to create an image gallery.

IMAGE OPACITY/TRANSPARENCY

The CSS3 property for transparency is **opacity**.
First we will show you how to create a transparent image with CSS.
Example 1:
Regular image:

The same image with transparency:

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:
Before mouse over the images:

After mouse over the first image:

After mouse over the second image:

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 */
}
```

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_navsprites.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_navsprites.gif" image to show:

Example

```
img.home {
   width: 46px;
   height: 44px;
   background: url(img_navsprites.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_navsprites.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_navsprites.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;
   background: url('img_navsprites.gif') 0 0;
}

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

#next {
   left: 129px;
   width: 43px;
   background: url('img_navsprites.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_navsprites.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_navsprites.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_navsprites.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. The hover selector can be used on all elements, not only on links. Our new image ("img_navsprites_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_navsprites_hover.gif') 0 -45px;
}

#prev a:hover {
   background: url('img_navsprites_hover.gif') -47px -45px;
}

#next a:hover {
   background: url('img_navsprites_hover.gif') -91px -45px;
}
```