CSS Basics.com

Chapter: 1 - Introduction to CSS

A CSS (cascading style sheet) file allows you to separate your web sites (X)HTML content from it's style. As always you use your (X)HTML file to arrange the content, but all of the presentation (fonts, colors, background, borders, text formatting, link effects & so on...) are accomplished within a CSS.

At this point you have some choices of how to use the CSS, either internally or externally.

Internal Stylesheet

First we will explore the internal method. This way you are simply placing the CSS code within the <head></head> tags of each (X)HTML file you 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>
<body>
```

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

External Stylesheet

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 (X)HTML, only CSS. You simply save it with the .css file extension. You can link to the file externally by placing one of the following links in the head section of every (X)HTML file you want to style with the CSS file.

```
<link rel="stylesheet" type="text/css" href="Path To
stylesheet.css" />
```

Or you 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 your (X)HTML files link to one CSS file in order to style the pages. This means, that if you need to alter the design of all your pages, you only need to edit one .css file to make global changes to your entire website.

Here are a few reasons this is better.

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

Are you getting the idea? It's really cool.

Cascading Order

In the previous paragraphs, I have explained how to link to a css file either internally or externally. If you understood, than I am doing a good job. If not don't fret, there is a long way to go before we are finished. Assuming you have caught on already, you are probably asking, well can I do both? The answer is yes. You can have both internal, external, and now wait a minute a third way? Yes inline styles also.

Inline Styles

I have not mentioned them until now because in a way they defeat the purpose of using CSS in the first place. Inline styles are defined right in the (X)HTML file along side the element you want to style. See example below.

```
<p style="color: #ff0000;">Some red text</p>
```

Some red text

Inline styles will NOT allow the user to change styles of elements or text formatted this way

So, which is better?

So with all these various ways of inserting CSS into your (X)HTML files, you may now be asking well which is better, and if I use more than one method, in what order do these different ways load into my browser?

All the various methods will cascade into a new "pseudo" stylesheet in the following order:

1. Inline Style (inside (X)HTML element)
2. Internal Style Sheet (inside the <head> tag)
3. External Style Sheet

As far as which way is better, it depends on what you want to do. If you have only one file to style then placing it within the <head></head> tags (internal) will work fine. Though if you are planning on styling multiple files then the external file method is the way to go.

Choosing between the <link related=> & the @import methods are completely up to you. I will mention that the @import method may take a second longer to read the CSS file in Internet Explorer than the <link related=> option. To combat this see Flash of unstyled content

Users with Disabilities

The use of external style sheets also can benefit users that suffer from disabilities. For instance, a user can turn off your stylesheet or substitute one of there own to increase text size, change colors and so on. For more information on making your website accessible to all users please read Dive into accessibility

Power Users

Swapping stylesheets is beneficial not only for users with disabilities, but also power users who are particular about how they read Web documents.

Browser Issues

You will discover as you delve farther into the world of CSS that all browsers are not created equally, to say the least. CSS can and will render differently in various browsers causing numerous headaches.

Chapter 2 - CSS Syntax

The syntax for CSS is different than that of (X)HTML markup. Though it is not too confusing, once you take a look at it. It consists of only 3 parts.

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

The selector is the (X)HTML element that you want to style. The property is the actual property title, and the value is the style you apply to that property.

Each selector can have multiple properties, and each property within that selector can have independent values. The property and value are seperated with a colon and contained within curly brackets. Multiple properties are seperated by a semi colon. Multiple values within a property are sperated by commas, and if an individual value contains more than one word you surround it with quotation marks. As shown below.

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

As you can see in the above code I have seperated the color from the font-family with a semi-colon, seperated the various fonts with commas and contained the "Trebuchet MS" within quotations marks. The final result sets the body color to light grey, and sets the font to ones that most users will have installed on there computer.

I have changed the way I layout my code, but you can arrange it in one line if you choose. I find that it is more readable if I spread each property to a seperate line, with a 2 space indention.

Inheritance

When you nest one element inside another, the nested element will inherit the properties assigned to the containing element. Unless you modify the inner elements values independently.

For example, a font declared in the body will be inherited by all text in the file no matter the containing element, unless you declare another font for a specific nested element.

```
body {font-family: Verdana, serif;}
```

Now all text within the (X)HTML file will be set to Verdana.

If you wanted to style certain text with another font, like an h1 or a paragraph then you could do the following.

```
h1 {font-family: Georgia, sans-serif;}
p {font-family: Tahoma, serif;}
```

Now all <h1> tags within the file will be set to Georgia and all <p> tags are set to Tahoma, leaving text within other elements unchanged from the body declaration of Verdana.

There are instances where nested elements do not inherit the containing elements properties.

For example, if the body margin is set to 20 pixels, the other elements within the file will not inherit the body margin by default.

```
body {margin: 20px;}
```

Combining Selectors

You can combine elements within one selector in the following fashion.

```
h1, h2, h3, h4, h5, h6 {
   color: #009900;
   font-family: Georgia, sans-serif;
}
```

As you can see in the above code, I have grouped all the header elements into one selector. Each one is seperated by a comma. The final result of the above code sets all headers to green and to the specified font. If the user does not have the first font I declared it will go to another sans-serif font the user has installed on there computer.

Comment tags

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

```
/* This is a comment */
```

You 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).

Chapter 3: CSS Classes

The class selector allows you to style items within the same (X)HTML element differently. Similiar to what I mentioned in the introduction about inline styles. Except with classes the style can be overwritten by changing out stylesheets. You can use the same class selector again and again within an (X)HTML file.

To put it more simply, this sentence you are reading is defined in my CSS file with the following.

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

Pretty simple, but lets say that I 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 (X)HTML file.

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

Then in my CSS file I 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** you are reading is styled in my CSS file by the following.

Please note that a class selector begins with a (.) period. The reason I named it "greenboldtext" is for example purposes, you can name it whatever you want. Though I do encourage you to use selector names that are descriptive. You can reuse the "greenboldtext" class as many times as you want.

Chapter 4: CSS IDs

IDs are similar to classes, except once a specific id has been declared it cannot be used again within the same (X)HTML file.

I generally use IDs to style the layout elements of a page that will only be needed once, whereas I use classes to style text and such that may be declared multiple times.

The main container for this page is defined by the following.

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

I 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 my CSS file I have the following:

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

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

Chapter 5: CSS Divisions

Ok so you have finished the first 4 chapters in my series. You have learned the very basics of CSS, how the syntax works and a bit about classes and IDs. Now we are gonna take a quick break from CSS and focus on the (X)HTML side of using it.

Divsions

Divisions are a block level (X)HTML element used to define sections of an (X)HTML file. A division can contain all the parts that make up your website. Including additional divisions, spans, images, text and so on.

You define a division within an (X)HTML file by placing the following between the <body></body> tags:

```
<div>
Site contents go here
</div>
```

Though most likely you will want to add some style to it. You can do that in the following fashion:

```
<div id="container">
Site contents go here
</div>
```

The CSS file contains this:

```
#container{
  width: 70%;
  margin: auto;
  padding: 20px;
  border: 1px solid #666;
  background: #ffffff;
}
```

Now everything within that division will be styled by the "container" style rule, I defined within my CSS file. A division creates a linebreak by default. You can use both classes and IDs with a division tag to style sections of your website.

Chapter 6: CSS Spans

Spans are very similar to divisions except they are an inline element versus a block level element. No linebreak is created when a span is declared.

You can use the span tag to style certain areas of text, as shown in the following:

```
<span class="italic">This text is italic</span>
```

Then in my CSS file:

```
.italic{
  font-style: italic;
}
```

The final result is: *This text is italic.*

The purpose of the last 2 chapters was to provide you with a basis for using CSS in an (X)HTML file. For a more detailed explaination of XHTML please visit W3Schools

Chapter 7: CSS Margins

Inherited: No

As you may have guessed, the margin property declares the margin between an (X)HTML element and the elements around it. The margin property can be set for the top, left, right and bottom of an element. (see example below)

```
  margin-top: length percentage or auto;
  margin-left: length percentage or auto;
  margin-right: length percentage or auto;
  margin-bottom: length percentage or auto;
```

As you can also see in the above example you have 3 choices of values for the margin property

- length
- percentage
- auto

You can also declare all the margins of an element in a single property as follows:

```
  margin: 10px 10px 10px 10px;
```

If you declare all 4 values as I have above, the order is as follows:

1. top
2. right
3. bottom
4. left

If only one value is declared, it sets the margin on all sides. (see below)

```
margin: 10px;
```

If you only declare two or three values, the undeclared values are taken from the opposing side. (see below)

```
margin: 10px 10px; /* 2 values */
margin: 10px 10px 10px; /* 3 values */
```

You can set the margin property to negative values. If you do not declare the margin value of an element, the margin is 0 (zero).

```
margin: -10px;
```

Elements like paragraphs have default margins in some browsers, to combat this set the margin to 0 (zero).

```
p {margin: 0;}
```

Note: You do not have to add px (pixels) or whatever units you use, if the value is 0 (zero).

You can see in the example below, the elements for this site are set to be 20px (pixels) from the body

```
body{
  margin: 20px;
  background: #eeeeee;
  font-size: small;
  font-family: Tahoma, Arial, "Trebuchet MS", Helvetica, sans-
serif;
  text-align: left;
}
```

Chapter 8: CSS Padding

Inherited: No

Padding is the distance between the border of an (X)HTML element and the content within it.

Most of the rules for margins also apply to padding, except there is no "auto" value, and negative values cannot be declared for padding.

```
padding-top: length percentage;
padding-left: length percentage;
padding-right: length percentage;
padding-bottom: length percentage;
```

As you can also see in the above example you have 2 choices of values for the padding property

- length

- percentage

You can also declare all the padding of an element in a single property as follows:

```
padding: 10px 10px 10px 10px;
```

If you declare all 4 values as I have above, the order is as follows:

1. top
2. right
3. bottom
4. left

If only one value is declared, it sets the padding on all sides. (see below)

```
padding: 10px;
```

If you only declare two or three values, the undeclared values are taken from the opposing side. (see below)

```
padding: 10px 10px; /* 2 values */
padding: 10px 10px 10px; /* 3 values */
```

If you do not declare the padding value of an element, the padding is 0 (zero).

Note: You do not have to add px (pixels) or whatever units you use, if the value is 0 (zero).

You can see in the example below, the main container for this site has 30px (pixels) of padding between the border and the text.

```
#container{
  width: 70%;
  margin: auto;
  padding: 30px;
  border: 1px solid #666;
  background: #ffffff;
}
```

Chapter 9: CSS Text Properties

Inherited: Yes

Color

You can set the color of text with the following:

```
color: value;
```

Possible values are

- color name - example:(red, black...)
- hexadecimal number - example:(#ff0000, #000000)
- RGB color code - example:(rgb(255, 0, 0), rgb(0, 0, 0))

Letter Spacing

You can adjust the space between letters in the following manner. Setting the value to 0, prevents the text from justifying. You can use negative values.

```
letter-spacing: value;
```

Possible values are

- normal
- length

Example:

T h e s e   l e t t e r s   a r e   s p a c e d   a t   5 p x .

Text Align

You 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

You 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 Indent

You can indent the first line of text in an (X)HTML element with the following:

```
text-indent: value;
```

Possible values are

- length
- percentage

Examples:

This text is indented 10px pixels.

Text Transform

You can control the size of letters in an (X)HTML element with the following:

```
text-transform: value;
```

Possible values are

- none
- capitalize
- lowercase

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.

White Space

You can control the whitespace in an (X)HTML element with the following:

```
white-space: value;
```

Possible values are

- normal
- pre
- nowrap

Word Spacing

You can adjust the space between words in the following manner. You can use negative values.

```
word-spacing: value;
```

Possible values are

- normal
- length

Example:

These words are spaced at 5px.

Chapter 10: 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

You 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 you set a family name it is best to also add the generic family at the end. As this is a priortized 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

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

```
font-size: value;
```

There are alot 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, I am not even going to try to explain it. Actually there are already some great resources on how to size your text. (see below)

- What size text should I use in my css by Paul O'B
- Dive into accessibility - Font Sizes

Font Style

You can set the style of text in a element with the font-style property

```
font-style: value;
```

Possible values are

- normal
- itailc
- oblique

Font Variant

You can set the variant of text within an element with the font-variant property

```
font-variant: value;
```

Possible values are

- normal
- small-caps

Font Weight

You can control the weight of text in an element with the font-weight property:

```
font-weight: value;
```

Possible values are

- lighter
- normal
- 100
- 200
- 300
- 400
- 500
- 600
- 700
- 800
- 900
- bold

- bolder

Chapter 11: CSS Anchors, Links and Pseudo Classes

Below are the various ways you can use CSS to style links.

```
a:link {color: #009900;}
a:visited {color: #999999;}
a:hover {color: #333333;}
a:focus {color: #333333;}
a:active {color: #009900;}
```

Now lets take a look at what each one of the above link styles actually does.

```
a:link {color: #009900;}
```

The first on the list sets the color of a link when no event is occuring

```
a:visited {color: #999999;}
```

The second sets the color a link changes to, when the user has already visited that url

```
a:hover {color: #333333;}
```

The third sets the color a link changes to as the user places their mouse pointer over the link

```
a:focus {color: #333333;}
```

The fourth is primarilly for the same purpose as the last one, but this one is for users that are not using a mouse and are tabbing through the links via there keyboards tab key, it sets the color a link changes to as the user tabs through the links

```
a:active {color: #009900;}
```

The fifth on the list sets the color a link changes to as it is pressed.

Lets look at an example: Google

If your last visit to Google is not stored in your cache than the above link to google is blue, if you have already been to google then the link should be grey. if you mouseover or tab through the links, the link will change to dark grey, and last but not least if you click and hold the link without releasing it you will see it return back to the original blue color.

You must declare the a:link and a:visited before you declare a:hover. Furthermore, you must declare a:hover before you can declare a:active.

Using the above code will style all links on your web page, unless you declare a seperate set of link styles for a certain area of your webpage

Pseudo Classes

You can set links contained in different parts of your web page to be different colors by using the pseudo class. For example, lets say you want your links in the content area to have a different color then the links in the left or right column of your webpage.

You can do this in the following fashion:

```
#content a:link {color: #009900;}
#content a:visited {color: #999999;}
#content a:hover {color: #333333;}
#content a:focus {color: #333333;}
#content a:active {color: #009900;}
```

Now assuming that you have your main content in a division named "content" all links within that division will now be styled by this new style selector. Should your selector have a different name, just change the #content selector to match your division name.

Then for the links in a column you could use the following:

```
#column a:link {color: #009900;}
#column a:visited {color: #999999;}
#column a:hover {color: #333333;}
#column a:focus {color: #333333;}
#column a:active {color: #009900;}
```

Once again, this assumes the name of the column division, just change the name to match yours.

This same method can be accomplished by declaring a class instead of an id.

```
a.column:link {color: #009900;}
a.column:visited {color: #999999;}
a.column:hover {color: #333333;}
a.column:focus {color: #333333;}
a.column:active {color: #009900;}
```

Though in this case you will need to add a class to each link

```
<a class="column" href="" title="">some link text</a>
```

But, there is still yet an easier way

```
.column a:link {color: #009900;}
.column a:visited {color: #999999;}
.column a:hover {color: #333333;}
.column a:focus {color: #333333;}
.column a:active {color: #009900;}
```

Then in the (X)HTML file

```
<div class="column">
```

```
<a href="" title="">some link text</a>
</div>
```

There are other properties that can be added to links other than color, I was just trying to keep it simple. Almost any property that can be used to style text and fonts can be used to style links also

Chapter 12: CSS Backgrounds

Inherited: No

Background

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

```
background: #ffffff url(path_to_image) top left no-repeat fixed;
```

Values:

- attachment
- color
- image
- position
- repeat

Or you can set each property individually

Background Attachment

If you are using an image as a background. You can set whether the background scrolls with the page or is fixed when the user scrolls down the page with the background-attachment property

```
background-attachment: value;
```

Values:

- fixed
- scroll

Background Color

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

```
background-color: value;
```

Values:

- color name
- hexadecimal number
- RGB color code
- transparent

Background Image

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

```
background-image: url(path_to_image);
```

Values:

- url
- none

Background Position

You can position an image used for the background of an element using the background-position property.

```
background-position: value;
```

Values:

- top left
- top center
- top right
- center left
- center center
- center right
- bottom left
- bottom center
- bottom right
- x-% y-%
- x-pos y-pos

Background Repeat

You can set if an image set as a background of an element is to repeat (across=x and/or down=y) the screen using the background-repeat property.

```
background-repeat: value;
```

Values:

- no-repeat

- repeat
- repeat-x
- repeat-y

Chapter 13: CSS Borders

Inherited: No

Border

You can set the color, style and width of the borders around an element in one declaration by using the border property.

```
border: 1px solid #333333;
```

Values:

- color
- style
- width

Or you can set each property individually

Border Color

You can set the color of a border independently with the border-color property.

```
border-color: value;
```

Values:

- color name
- hexadecimal number
- RGB color code
- transparent

Border Style

You can set the style of a border independently with the border-style property.

```
border-style: value;
```

Values:

- dashed
- dotted
- double

- groove
- hidden
- inset
- none
- outset
- ridge
- solid

Border Width

You can set the width of a border independently with the border-width property.

```
border-width: value;
```

Values:

- Length
- Thin
- Medium
- Thick

Or you can set the elements for each borders side individually

Border Bottom

You can set the color, style and width of the bottom border around an element in one declaration with the border-bottom property.

```
border-bottom: 1px solid #333333;
```

Values:

- color
- style
- width

Or you can set each value individually

Border Bottom Color

You can set the color of the bottom border around an element with the border-bottom-color property.

```
border-bottom-color: value;
```

Border Bottom Style

You can set the style of the bottom border around an element with the border-bottom-style property.

```
border-bottom-style: value;
```

Border Bottom Width

You can set the width of the bottom border around an element with the border-bottom-width property.

```
border-bottom-width: value;
```

Border Left

You can set the color, style and width of the left border around an element with the border-left property.

```
border-left: 1px solid #333333;
```

Values:

- color
- style
- width

Or you can set each value individually

Border Left Color

You can set the color of the left border around an element with the border-left-color property.

```
border-left-color: value;
```

Border Left Style

You can set the style of the left border around an element with the border-left-style property.

```
border-left-style: value;
```

Border Left Width

You can set the width of the left border around an element with the border-left-width property.

```
border-left-width: value;
```

Border Right

You can set the color, style and width of the right border around an element in one declaration with the border-right property.

```
border-right: 1px solid #333333;
```

Values:

- color
- style
- width

Or you can set each value individually

Border Right Color

You can set the color of the right border around an element with the border-right-color property.

```
border-right-color: value;
```

Border Right Style

You can set the style of the right border around an element with the border-right-style property.

```
border-right-style: value;
```

Border Right Width

You can set the width of the right border around an element with the border-right-width property.

```
border-right-width: value;
```

Border Top

You can set the color, style and width of the top border around an element in one declaration with the border-top property.

```
border-top: 1px solid #333333;
```

Values:

- color
- style
- width

Or you can set each value individually

Border Top Color

You can set the color of the top border around an element with the border-top-color property.

Border Top Style

You can set the style of the top border around an element with the border-top-style property.

```
border-top-style: value;
```

Border Top Width

You can set the width of the top border around an element with the border-top-width property.

```
border-top-width: value;
```

Chapter 14 - CSS Ordered & Unordered Lists

Inherited: Yes

List Style

You can control the appearance of ordered and unordered lists in one declaration with the list-style property

```
list-style: value value;
```

Values:

- image
- position
- type

Or you can control them individually

List Style Image

You can use an image for the bullet of unordered lists with the list-style property

```
list-style-image: url(path_to_image.gif, jpg or png);
```

If you 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

You can control the position of ordered and unordered lists with the list-style-position property

```
list-style-position: value;
```

Values

- inside
- outside

List Style Type

You can control the type of bullet ordered and unordered lists use with the list-style-type property

```
list-style-type: value;
```

Values

- disc
- circle
- square
- decimal
- lower-roman
- upper-roman
- lower-alpha
- upper-alpha
- none

Chapter 15 - CSS Width and Height Properties

Inherited: No

Height

You can control the height of an element with the height property

```
height: value;
```

Values:

- auto
- length
- percentage

Line Height

You can control the height between lines with the line-height property

```
line-height: value;
```

Values:

- normal
- number
- length
- percentage

Max Height

You can control the maximum height of an element with the `max-height` property

```
max-height: value;
```

Values:

- none
- length
- percentage

Min Height

You can control the minimum height of an element with the `min-height` property

```
min-height: value;
```

Values:

- length
- percentage

Width

You can control the width of an element with the `width` property

```
width: value;
```

Values:

- auto
- length
- percentage

Max Width

You can control the maximum width of an element with the `max-width` property

```
max-width: value;
```

Values:

- none
- length
- percentage

Min Width

You can control the minimum width of an element with the min-width property

```
min-width: value;
```

Values:

- length
- percentage

Chapter 16 - CSS Classification

Inherited: No

Clear

You can control if an element allows floated elements to its sides with the clear property

```
clear: value;
```

Values:

- none
- both
- left
- right

Now, what does all that mean?

None

This is the default setting, floated elements can appear on either side of the element set to clear: none;

Both

Setting the value to both, causes no floated elements to appear on either side of the element set to clear: both;

Left

Setting the value to left, causes no floated elements to appear to the left side of the element set to clear: left;

Right

Setting the value to right, causes no floated elements to appear to the right side of the element set to clear: right;

Clip

You can control how much of an element is visible with the clip property

```
clip: value;
```

Values:

- auto
- shape

Currently the only shape recognized by the clip property is rect (rectangle)

```
clip: rect(10px, 10px, 10px, 10px);
```

Cursor

You can control the style of cursor to be used in an element with the cursor property

```
cursor: value;
```

Values:

- auto
- crosshair
- default
- help
- move
- pointer
- text
- url
- wait
- e-resize
- ne-resize
- nw-resize
- n-resize
- se-resize
- sw-resize
- s-resize

- w-resize

If you choose to use a custom cursor, it is always a good idea to declare a generic one after the custom value.

```
cursor: url("image.cur"), default;
```

Display

You can control how an element is displayed with the display property

```
display: value;
```

Values:

- block
- inline
- list-item
- none

Now, what does all that mean?

Block

Creates a line break before and after the element

Inline

No line break is created

List Item

Creates a line break before and after the element and adds a list item marker

None

Makes an element not display on the page

Float

The float property changes how text and or images within an element are displayed

```
float: value;
```

Values:

- left
- right
- none

Now, what does all that mean?

Left

The image/text is displayed to the left of the parent element

Right

The image/text is displayed to the right of the parent element

None

There is no change in the way the image/text is displayed

Overflow

You can control what an elements contents will do if it overflows it boundaries with the overflow property

```
overflow: value;
```

Values:

- auto
- hidden
- visible
- scroll

Overflow Example

As you can see, with this property you can mimic an iframe. This box is set to an overflow value of "auto". Meaning that if the contents of the element break the boundaries it should add a scrollbar.

Here is what I have in my CSS file.

```
#overflow_box {width:200px; height:200px; border-top: 1px solid
#eee; border-left: 1px solid #eee; border-bottom: 1px solid
#eee; padding: 10px; overflow: auto;}
```

Then in the (X)HTML file I have this:

```
<div id="overflow_box">Contents</div>
```

Visibility

You can control if an element is visible or not with the visibility property

```
visibility: value;
```

Values:

- hidden
- visible

Z-Index

You can control the layer order of positioned elements with the z-index property

```
z-index: value;
```

Values:

- auto
- number

The higher the number the higher the level. Negative numbers are allowed

Chapter 17 - CSS Positioning

Inherited: No

Position

The position property (as you may have guessed) changes how elements are positioned on your webpage.

```
position: value;
```

Values:

- static
- relative
- absolute
- fixed

Now, what does all that mean?

Static

Static positioning is by default the way an element will appear in the normal flow of your (X)HTML file. It is not necessary to declare a position of static. Doing so, is no different than not declaring it at all.

```
position: static;
```

Relative

Positioning an element relatively places the element in the normal flow of your (X)HTML file and then offsets it by some amount using the properties left, right, top and bottom. This may cause the element to overlap other elements that are on the page, which of course may be the effect that is required.

```
position: relative;
```

Absolute

Positioning an element absolutely, removes the element from the normal flow of your (X)HTML file, and positions it to the top left of it's nearest parent element that has a position declared other than static. If no parent element with a position other than static exists then it will be positioned from the top left of the browser window.

```
position: absolute;
```

Fixed

Positioning an element with the fixed value, is the same as absolute except the parent element is always the browser window. It makes no difference if the fixed element is nested inside other positioned elements.

Furthermore, an element that is positioned with a fixed value, will not scroll with the document. It will remain in it's position regardless of the scroll position of the page.

At this time IE6 (Internet Explorer 6) does not support the fixed value for the positioning of an element. Thus it will not position fixed elements correctly and will still scroll with the page. To see this effect in action you will need to use a standards compliant browser, such as Firefox 1.0

```
position: fixed;
```

When positioning elements with relative, absolute or fixed values the following properties are used to offset the element:

- top
- left
- right
- bottom

```
position: absolute; top: 10px; right: 10px;
```

Chapter 18 - CSS Pseudo Elements

The Syntax

The syntax for pseudo elements is a bit different than that of regular CSS, but it's real close. If you have already read chapter 11 then you are slightly ahead of the game.

```
selector:pseudo-element {property: value}
```

As you can see the only difference is that you place the pseudo element after the selector, and divide the 2 with a (:) colon.

Or you can assign a class to a pseudo element as follows

```
selector.p:pseudo-element {property: value}
```

Using the above code would style all paragraphs within the declared selector with the pseudo element.

The elements:

- first-line
- first-letter

First Line

The first-line pseudo element styles the first line of text in a block level element.

```
p{font-size: small;}
p:first-line {font-size: medium; color: #ff0000;}
```

As you can see in the above example paragraphs are set to be a small font size, but the p:first-line is set to be a medium size and a red color. The result is that the first line of all paragraphs will be red in color and a bit larger than the rest of the paragraph.

Though lets say you only want to style a certain paragraph of text with the first-line element. Thats where declaring a class to the pseudo element comes into play.

first-line with class

```
p.special:first-line {font-size: medium; color: #ff0000;}
```

I have declared a class of special within my css file.

First-Line Example

This is a special sentence I wrote to demonstrate the use and look of the first-line pseudo element. As you can see the first line of this paragraph is styled differently than the rest of the text within it. All of this was done by simply adding class="special" to the opening <p> tag for this paragraph.

```
<p class="special">the content</p>
```

Where the first-line ends depends on the width of the browser window or containing element, you can resize this page and see that it adjusts as you change the size of the browser window.

The following properties can be assigned to the first-line pseudo element:

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

First Letter

The first-letter pseudo element styles the first letter of text in a block level element.

```
p{font-size: small;}
p:first-letter {font-size: medium; color: #ff0000;}
```

As you can see in the above example paragraphs are set to be a small font size, but the p:first-letter is set to be a medium size and a red color. The result is that the first letter of all paragraphs will be red in color and a bit larger than the rest of the paragraph.

Though lets say you only want to style a certain paragraph of text with the first-letter element. Thats where declaring a class to the pseudo element comes into play.

first-letter with class

```
p.special_letter:first-letter {font-size: x-large; font-weight: bold; color: #ff0000;}
```

I have declared a class of special_letter within my css file.

First-Letter Example

This is a special sentence I wrote to demonstrate the use and look of the first-letter pseudo element. As you can see the first letter of this paragraph is styled differently than the rest of the characters within it. All of this was done by simply adding class="special_letter" to the opening <p> tag for this paragraph.

```
<p class="special_letter">the content</p>
```

The following properties can be assigned to the first-letter pseudo element:

- background
- border
- clear
- color
- float
- font
- line-height
- margin
- padding
- text-decoration
- text-transform
- word-spacing