

School of Information and Computer Technology  
Sirindhorn International Institute of Technology  
Thammasat University  
ITS351 Database Programming Laboratory

---

*Laboratory #1: CSS and User Interface*

---

## 1 What is CSS?

Cascading Style Sheets, or CSS was introduced in late 1996, as an elegant cousin to HTML that promised several things, including:

- more precise control than ever before over layout, fonts, colors, backgrounds, and other typographical effects;
- a way to update the appearance and formatting of an unlimited number of pages by changing just one document;
- compatibility across browsers and platforms; and
- less code, smaller pages, and faster downloads.

CSS represents the World Wide Web Consortium's (W3C's) effort to improve on the tag and attribute based style of formatting. The idea is that style sheets provide a way of customizing a whole page and even whole sites at one time and in much richer detail than the simple use of tags and attributes.

Plus, CSS is sort of like the cornerstone of Dynamic HTML, which is another subject that you should investigate. So CSS is a technology that gives you, as a webmaster, much more control over all these things that you would ever have in HTML alone.

## 2 Why we need CSS?

Let's suppose for a minute that you wanted all your paragraphs in a multi-page document to be center-aligned. If you omit the attribute and value for the <p> tag, what do you think the browser will do? It will simply use its default setting, which is left-aligned. So, every time you create a new paragraph, you have to remember to type in the attribute and the value. If you forget, your document won't look like you want it to look.

Now that would be no big deal if you only had to deal with one or two pages and one or two tags, but what if you are dealing with a website that has 50 pages and hundreds of different tags. Now we are talking about hours and hours and hours of work. That is when and where CSS becomes invaluable, and it is only one place that this technology is a real time and headache saver.

## 3 How it works?

A CSS (cascading style sheet) file allows you to separate your web sites HTML content from its style. As always you use your 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.

### 3.1 **Internal Style Sheet**

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

Content written by Ben Partch with contributions from Paul O'Brien & Vinnie Garcia.

Last Updated: 16/08/15

```

<head>
  <title>...</title>
  <style type="text/css">
    CSS Content Goes Here
  </style>
</head>
<body>

```

With this method each 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.

### 3.2 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". A CSS file contains no 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 HTML file you want to style with the CSS file.

```

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

```

By using an external style sheet, all of your HTML files link to one CSS file which is used 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

### 3.3 CSS Syntax

The syntax for CSS is different than that of HTML markup. It consists of only 3 parts.

```

selector { property: value }

```

1. The selector is the HTML element that you want to style.
2. The property is the actual property title.
3. 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 separated with a colon and contained within curly brackets. Multiple properties are separated by a semi colon. Multiple values within a property are separated 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, the color and the font-family properties are separated with a semi-colon. Specified font names are separated with commas. The final result sets the body color to light grey (#eeeeee), and sets the font to ones that most users will have installed on their computer.

We have changed the way we layout our code, but you can arrange it in one line if you choose. We find that it is more readable if we spread each property to a separate line.

### 3.4 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 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;}
```

### 3.5 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, the entire header elements are grouped into one selector. Each one is separated by a comma. The final result of the above code sets all headers to green (#009900) and to the specified font. If the user does not have the first font "Georgia", it will go to the next font "sans-serif" the user has installed on their computer.

### 3.6 **Comment**

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 CSS comment. Just like in C. */
```

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).

### 3.7 **CSS Classes**

The class selector allows you to style items within the same HTML element differently. Except with classes the style can be overwritten by changing out style sheets. You can use the same class selector again and again within an HTML file.

Let's say the following style of <p> is defined.

```
p {
  font-size: small;
  color: #666666
}
```

The resulting text will be rendered small and gray. Pretty simple, but let's say that we want to change the word "sentence" to green bold text, while leaving the rest of the sentence untouched. We would do the following to our 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 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** you 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, you can name it whatever you want. Though we do encourage you to **use selector names that are descriptive**, you can reuse the "greenboldtext" class as many times as you want.

### 3.8 CSS IDs

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

For example, the main container for the 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 we only need to use it once** 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;
}
```

The final result would look like the following:

```
Everything within my document is inside this division.
```

Notice that:

- The id selector begins with a (#) number sign.
- The class selector begins with a (.) period.

### 3.9 CSS Divisions

Now we will take a quick break from CSS and focus on the HTML side of using it. Divisions are a block level HTML element used to define sections of an 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 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. **A division creates a line break by default.** You can use both classes and IDs with a division tag to style sections of your website.

### 3.10 CSS Spans

**Spans are very similar to divisions except they are an inline element** as oppose to divisions which are a block level element. **No line break 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 the CSS file:

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

The final result is:

*This text is italic*

### 3.11 CSS Margins

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

```
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 (e.g., 10px)
- Percentage (e.g., 40%)
- 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 20px; /* margin-bottom is 10px */  
margin: 10px 20px 30px; /* margin-left is 20px */
```

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;}
```

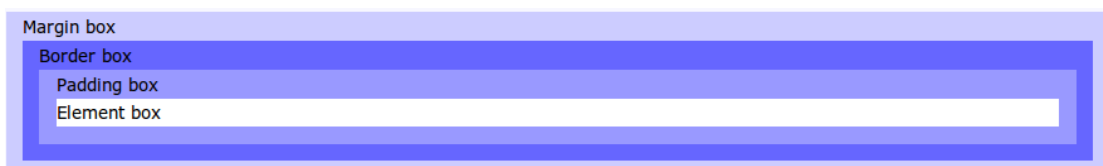
Note: You do not have to add px (pixels) or whatever units you use, if the value is 0 (zero).

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;  
}
```

### 3.12 CSS Padding

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



Content written by Ben Partch with contributions from Paul O'Brien & Vinnie Garcia.

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 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 just like in the case of margin. (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).**

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;  
}
```



### 3.13 CSS Text Properties

#### 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. This text is justified. This text is justified. This text is justified.  
This text is justified. This text is justified. This text is justified. This text is justified.

#### Text Decoration

You can decorate text with the following:

Content written by Ben Partch with contributions from Paul O'Brien & Vinnie Garcia.

```
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 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 HTML element with the following:

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

Possible values are

- none
- capitalize
- lowercase

**Examples:**

This First Letter In Each Word Is Capitalized.

THIS TEXT IS ALL UPPERCASE.

this text is all lowercase.

## **White Space**

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

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

Possible values are

- **normal:** Sequences of whitespace will collapse into a single whitespace. Text will wrap when necessary.
- **pre:** Whitespace is preserved by the browser. Text will only wrap on line break.
- **nowrap:** Sequences of whitespace will collapse into a single whitespace. Text will only wrap on <br>.

## 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.

## 3.14 CSS Font Properties

### 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: serif, sans-serif, monospace, cursive, fantasy

If you 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.

```
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 a lot of choices for values:

- xx-large
- x-large
- larger
- large
- medium
- small
- smaller
- x-small
- xx-small
- length
- % (percent)

## Font Style

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

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

Possible values are

- normal
- italic
- 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 ( 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

### 3.15 CSS Anchors, Links and Pseudo Classes

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

```
a:link {color: #009900;} /* link when no event is occurring*/
a:visited {color: #999999;} /* link already visited*/
a:hover {color: #333333;} /*link when there is a mouse over*/
a:focus {color: #333333;} /*in-focus link (when navigating with keys)*/
a:active {color: #009900;} /*link being pressed down*
```

#### **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, let's say you want your links in the content area to have a different color than 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. If your selector has a different name, just change the #content selector to match your division name.

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 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. Almost any property that can be used to style text and fonts can also be used to style links.

### 3.16 CSS Backgrounds

#### **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 e.g., `background-color: rgb(255, 0, 255);`
- 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-% (relative to top-left corner)
- x-pos y-pos (relative to top-left corner)

## **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:

- repeat
- no-repeat
- repeat-x
- repeat-y

## **3.17 CSS Borders**

### **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:        

- hidden
- none

## Border Width

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

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

Values:

- Length
- Thin
- Medium
- Thick

## Border Collapse

The border-collapse property sets whether the table borders are collapsed into a single border or detached as in standard HTML.

```
border-collapse: value;
```

Values:

- collapse – Borders are collapsed into a single border.
- separate – Borders are detached (default)
- inherit

### 3.18 One-sided Border

Properties of border on each side of an element can be set separately. The syntax is

```
border-<direction>: 1px solid #333333; /*width, style, color*/
```

Where <direction> can be

- top
- bottom
- left
- right

Or you can set each property (i.e., width, style, color) individually by

```
border-<direction>-width: 1px solid #333333;  
border-<direction>-style: solid;  
border-<direction>-color: #333333;
```

For example, to set the left border to have width of 15px, dotted style, and red color, we use

```
border-left-width:15px;  
border-left-style:dotted;  
border-left-color:#ff0000;
```

### 3.19 CSS Ordered & Unordered Lists

#### **List Style**

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

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

Values:

- image
- position
- type

Or you can set each property 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

### 3.20 CSS Width and Height Properties

#### **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

### **3.21 CSS Classification**

## **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**

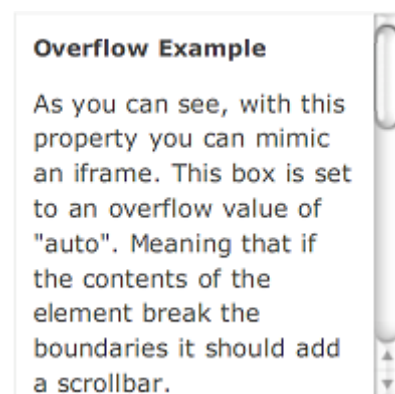
You can control what an elements contents will do if it overflows it boundaries with the overflow property

```
overflow: value;
```

Values:

- auto
- hidden
- visible
- scroll

Overflow Example



Here is what we have in our 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;}
```

Content written by Ben Partch with contributions from Paul O'Brien & Vinnie Garcia.

Last Updated: 16/08/15

Then in the HTML file, we 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 means the higher the level. Negative numbers are allowed.

## **3.22 CSS Positioning**

### **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 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 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 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;
```

## **3.23 CSS Pseudo Elements**

### **The Syntax**

The syntax for pseudo elements is a bit different than that of regular CSS, but it is close.

```
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 let's say you only want to style a certain paragraph of text with the first-line element. That is where declaring a class to the pseudo element comes into play.

### **first -line with class**

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

We have declared a class of special within our css file.

### **First-Line Example**

This is a special sentence we 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. That is 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;}
```

We have declared a class of special\_letter within our css file.

### **First-Letter Example**

**T**his 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

## **4 CSS 3**

So far, the mentioned properties are mostly of CSS 1 and CSS 2. Recent development of CSS has already gone to version 3. Even version 4 is already in its working draft stage. Here, we will touch upon cool features offered by CSS3 a little bit. Although they may not work in all major web browsers now, they will all work soon.

As a side note, when W3C sets a specification of a new CSS, most mainstream web browsers (i.e, Firefox, Internet Explorer, Chrome, Safari) will try to implement them as soon as possible. However, immediately making them work as specified in the standard can be challenging. So, when those web browsers test their new CSS implementations, they will prefix those new not-fully-supported properties with their own keywords, so that users know they are not yet fully supported.

- Firefox uses the prefix: -moz-
- Chrome and Safari use: -webkit-
- Internet Explorer uses: -ms-
- Opera uses: -o-

For example, you may need to use "-moz-box-orient" in Firefox for the new "box-orient" property in CSS3. When it is fully supported, "-moz-" will be no longer needed. To be safe, it is recommended that you specify "box-orient" and all of its variations if you need to use this experimental property.

The following CSS3 properties are ones which should be supported by all major browsers. If you use an older browser, prefix the property with "-moz-" and/or "-webkit-" as already mentioned. In this course, **everyone is strongly recommended to use Firefox**. Internet explorer is not recommended since it is specific for Windows.

## border-radius

This property allows you to add rounded borders to elements. Variations for specific sides also exist.

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

Since the borders are round, text inside the box may overflow. So, it is common to use **border-radius** with **padding**. For example,

```
div{
    border:2px solid #a1a1a1;
    padding:10px 40px;
    width:300px;
    border-radius:25px;
}
```

This CSS code sets round borders to all <div> tags.

The border-radius property allows you to add rounded corners to elements.

## text-shadow

The text-shadow property attaches one or more shadows to text. Currently, it is supported in all major browsers, except Internet Explorer.

text-shadow: *h-shadow v-shadow blur color*;

Value	Description
<i>h-shadow</i>	Required. The position of the horizontal shadow. Negative values are allowed
<i>v-shadow</i>	Required. The position of the vertical shadow. Negative values are allowed
<i>blur</i>	Optional. The blur distance
<i>color</i>	Optional. The color of the shadow.

Content written by Ben Partch with contributions from Paul O'Brien & Vinnie Garcia.

For example, the following code sets <h1> heading to have a red shadow.

```
h1 { text-shadow:2px 4px 3px #FF0000; }
```

The result looks like

**Text-shadow effect**

## resize

The resize property specifies whether or not an element is resizable by the user. You may want to use it with "overflow" property to control how content is handled with overflowing. The property is usually used in a <div>.

resize: none|both|horizontal|vertical

Value	Description
none	The user cannot resize the element
both	The user can adjust both the height and the width of the element
horizontal	The user can adjust the width of the element
vertical	The user can adjust the height of the element