Web Design, Level 3

Cascading Style Sheets
This handout accompanies classroom instruction provided by IT Services Technology Training, Stanford University. While it is not intended as a stand-alone tutorial, it may be helpful in reviewing the topics covered in class.

Copyright IT Services Technology Training
© 2008 Stanford University
All rights reserved
This page is intentionally blank.
The goal of this course is to provide an overview of Cascading Style Sheets (CSS).

The objectives of this course include your being able to:

- Understand what CSS is and why you should use CSS in your web sites.
- Understand the three methods of using CSS.
- Create a simple style sheet.
- Link a web page to a style sheet.
- Use some of the most commonly used CSS rules.
- Define positioning and create page layouts using CSS.
This page is intentionally blank.
What are Cascading Style Sheets?

The Cascading Style Sheets (CSS) language is made up of rules that control the look, or style, of web pages or other XML files. These rules are placed in a central location, or sheet, and determine how the HTML (Hyper-Text Markup Language) or XML (eXtensible Markup Language) tags are interpreted by browsers.

Why the term “cascading”? In CSS, multiple styles can be applied to a particular document (usually a web page or XML file). The browser interprets these styles in a specific top-down order.

All matching rules for a particular selector will be applied, except where they conflict with each other. If rules are in conflict, the last rule to be declared is applied. In the following example, <h2> tags would be displayed in red and italics (but not blue):

```
h2 {font-style: italic;}
h2 {color: darkblue;}
h2 {color: red;}
```

To properly see the effects of CSS, visitors to your site will need to use a web browser that is version 4.0 or newer. Fortunately, viewing web pages with CSS in an older browser does not cause the web pages to break. But, the styles won’t appear as defined. Since most people use Internet Explorer 6, Netscape 7, Firefox 1 or a newer browser, most pages with CSS will be properly displayed.

CSS-aware browsers apply their own stylesheet for every HTML element as the first set of rules in the cascade. This set of rules forms the default display for every element. For example, most browsers treat the <p> tag as a block element, as though there were the explicit declaration `p {display: block;}`. By using CSS, you override these implicit styles with an explicit declaration. For more on the block display, see Inline vs. Block display, page 23.

By using CSS, you can also:

- Control text formatting and location on the page.
- Eliminate the need for tables as a layout tool.
- Create logos merely using text, instead of having to rely on graphics.

These changes make pages more accessible to a wider audience.
The World Wide Web Consortium (W3C) is the organization that establishes the CSS rules and specifications. The latest specifications for CSS can be found at the following sites:

- **CSS 1**: [http://www.w3.org/TR/REC-CSS1-961217.html](http://www.w3.org/TR/REC-CSS1-961217.html)
- **CSS 2**: [http://www.w3.org/TR/CSS2/](http://www.w3.org/TR/CSS2/)
- **CSS 2.1**: [http://www.w3.org/TR/CSS21/](http://www.w3.org/TR/CSS21/)

For detailed, technical explanations of the differences between CSS 1, CSS 2, and CSS 2.1, go to the following:

- Between CSS 1 and CSS 2: [http://www.w3.org/TR/CSS2/changes.html](http://www.w3.org/TR/CSS2/changes.html)
- Between CSS 2 and CSS 2.1: [http://www.w3.org/TR/CSS21/changes.html](http://www.w3.org/TR/CSS21/changes.html)

**Pros and Cons of Using CSS**

**Pros**

- Greater designer control of the appearance of the page.
- Easier management of site-wide changes.
- Greater accessibility to web sites by non-graphical browsers and webpage-reading software.

**Cons**

- Different browsers may interpret CSS in different ways.
- Some styles may not be seen at all on some browsers.
The Cascading Style Sheets (CSS) Language

CSS contains rules. Each rule consists of a selector and a declaration (which in turn is made up of a property and a value).

Under standard HTML, to create a web site with <h2> tags that have the standard features of a Header tag (that is, their own paragraph, bold, with a size change) and also are dark blue, you have to code each one as follows:

```
<h2><font color="darkblue">This is a darkblue H2 tag</font></h2>
```

That’s a lot of information to type every time you want to use a dark blue <h2> tag. Using CSS, all you need to do is type a regular <h2> tag. The style information will be included in the Style Sheet as follows:

```
h2 { color: darkblue;}
```

To change the color of ALL <h2> tags from darkblue to green, in the CSS, simply change the declaration from `color: darkblue;` to `color: green;`. The next time anyone sees the site, all the <h2> tags on all the pages will display as green instead of darkblue.

In the example below, h2 is the selector, color is the property, and darkblue is the value. When used with web pages, selectors are usually HTML tags.

```
h2 { color: darkblue;}
```

Syntax for a CSS rule:

```
selector { property: value; }
```
Grouping Styles and Selectors

- **Multiple styles** – Each rule can include *multiple styles*, using semicolons to separate them.
  
  ```
  h2 {color: darkblue; font-style: italic;}
  ```

- **Multiple selectors**. Additionally, *multiple selectors* that have the same styles can be grouped, using commas to separate them.
  
  ```
  h1, h2, h3 {color: darkblue; font-style: italic;}
  ```

- **Contextual selectors**. *Contextual selectors* allow you to specify that something will occur only when it is used in conjunction with something else. In the style below, `em` will display in red, but only when it occurs within `li` within `ul`:
  
  ```
  ul li em {color: red;}
  ```
  
  Elements being modified by contextual selectors need not appear immediately inside one another. For example, using the style above with the HTML below, `blah` would still be red text:
  ```
  <ul><li><strong><em> blah </em></strong></li></ul>
  ```

- **Direct child selectors**. *Direct child selectors* allow you to specify that something will change, but only when immediately inside another element. With the style below, only those `strong` elements that are directly inside `h1` will be purple; no `strong` tags deeper within the sheet will be purple:
  
  ```
  h1 > strong {color: purple;}
  ```
  
  Elements being modified by adjacent selectors appear immediately after one another.

  Using the style above, this link would be green:
  ```
  ```

  But this link would not:
  ```
  ```

- **Adjacent selectors**. *Adjacent selectors* allow you to specify that something will change only when preceded by something else. In the style below, only those links that are preceded by an `h2` will be green:
  
  ```
  h2 + a {color: green;}
  ```

  Elements being modified by adjacent selectors appear immediately after one another.

  Using the style above, this link would be green:
  ```
  ```

  But this link would not:
  ```
  ```

- **By attribute**. You can also group selectors *by attribute*. In the example below, text that is inside centered `h2` tags (`<h2 align="center">`) will be surrounded by a dotted border:
  
  ```
  h2[align="center"] {border: dotted;}
  ```
CSS Locations

Style information can be located:

- External to the pages in a site (the CSS rules are listed in a separate file and linked to the HTML files).
- Internal to each page.
- Inline with individual tags.

Generally, creating an external style sheet file is the preferred method. To take full advantage of CSS, the Style Sheet for a site should be in an external file, so that any changes made there will apply throughout the site. This also means that only one style document has to be downloaded for a single site, making the web pages load faster.

CSS: External location

The most common place to put style information is in an external document that each page of a web site points to directly. Any changes made to this single document will then be applied throughout the entire web site as each page is accessed by users. External Style Sheets have a .css extension.

When linking to an external style sheet, you can also specify separate style sheets by media type:

<table>
<thead>
<tr>
<th>Media Type</th>
<th>Intended for Use With</th>
</tr>
</thead>
<tbody>
<tr>
<td>all</td>
<td>All devices</td>
</tr>
<tr>
<td>aural</td>
<td>Speech synthesizers</td>
</tr>
<tr>
<td>braille</td>
<td>Braille tactile feedback devices</td>
</tr>
<tr>
<td>embossed</td>
<td>Paged braille printers</td>
</tr>
<tr>
<td>handheld</td>
<td>Handheld devices (typically small screen, monochrome, limited bandwidth)</td>
</tr>
<tr>
<td>print</td>
<td>Paged, opaque material and for documents viewed on the screen in print preview mode</td>
</tr>
<tr>
<td>projection</td>
<td>Projected presentations</td>
</tr>
<tr>
<td>screen</td>
<td>Computer screens, primarily</td>
</tr>
<tr>
<td>tty</td>
<td>Media using a fixed pitch character grid, such as teletypes, terminals, or portable devices with limited display capabilities</td>
</tr>
<tr>
<td>tv</td>
<td>Television-type devices</td>
</tr>
</tbody>
</table>
There are two methods you can use to link web pages to an external style sheet: `<link>` and `@import`.

The `<link>` method was introduced with CSS 1, and specifically links an HTML document to a CSS file. With the `<link>` method, you can specify the media type using the `media="type"` attribute. (See below for an example.)

The `@import` method was introduced with CSS 2, and is used to import style sheets into web pages, XML files, or even other style sheets. With the `@import` method, you can specify the media type by adding an `@media` rule. (See below for an example.)

If you want to attach a style sheet to a web page, either method will work.

**EXAMPLE:** A web site with two web pages, using two external style sheets (4 separate files – when viewed on-screen, the `basic.css` rules are used; when printed, the `print.css` rules are used)

---

**Text that appears in the `basic.css` style sheet document:**

```css
h2 {font-family: Arial, sans-serif; font-style: italic;}
p {font-family: Courier, monotype; font-style: bold; }
```

**Text that appears in the `print.css` style sheet document:**

```css
h2 {font-family: Book Antiqua, serif; font-style: italic; }
p {font-family: Courier, monotype; font-style: bold; }
```

**HTML document 1, using the `<link>` tag method:**

```html
<head>
<link rel="stylesheet" type="text/css" href="basic.css" media="all">
<link rel="stylesheet" type="text/css" href="print.css" media="print">
</head>
```

**HTML document 2, using the `@import` and `@media` method:**

```html
<head>
<style type="text/css">
<!--[if --]
@import url("basic.css") all;
@media url("print.css") print;
-->
</style>
</head>
```
CSS: Internal location

Style information can also be included in the `<head>` section of an individual web page. This tends to work best when a single page needs to have a slightly different look than the rest of the site. If you plan to apply these styles to more than one web page, it is more useful to create an external style sheet and link the web pages to that style sheet.

EXAMPLE: A web page using internal styles

```html
<head>
  <style type="text/css">
    h1 { font-family: Arial; 
          font-style: italic; 
          color: green; 
    }
  -->
  </style>
</head>

<body>
  <h1>This is sooooooooo cool!</h1>
  <p>Nothing cool here.</p>
  <h1>This one's cool, too!</h1>
</body>
```

This is sooooooooo cool!

Nothing cool here.

This one's cool, too!
CSS: Inline location

For extreme control, style information can be included in an individual tag. The style effects only that tag and no others in the document. This option is most useful for those rare occasions when a single tag needs to have a slightly different style.

EXAMPLE: A web page using inline styles

```html
<body>
<h1 style="font-family:Arial; font-style:italic; color:green;">This is sooooooo cool!</h1>
<p>Nothing cool here.</p>
<h1>This isn't cool, either.</h1>
</body>
</html>
```

This is sooooooo cool!

Nothing cool here.

This isn't cool, either.
Font and Text Styling

When choosing a font, there are several things to keep in mind.

- Not everyone has the same set of fonts.

- If you use a font that the visitor doesn't have, the page will display in the default font (usually Times), unless you provide more choices. To do this, add more than one font in your declaration, and always end with the font family (serif, sans-serif, or monospace):
  font-family: Verdana, Arial, Helvetica, sans-serif

- Printed documents tend to look better in Serif fonts (Times New Roman, Georgia, Book Antiqua, etc.).

- Documents to be viewed on-screen tend to look better in Sans-serif fonts (Verdana, Arial, Helvetica, etc.).

To apply a font to the entire web page, modify the <body> tag in the CSS. For example, this CSS rule changes the default font for the web page to Verdana:
body {font-family: Verdana;}

To apply a font to a specific letter, word, or series of words, see Span and Div on page 19.
Using !important

Normally, the last rule listed in the cascade will take precedence over previous rules.

In this example, the body font will be Verdana, not Times:
```css
body {font-family: Times;
     font-family: Verdana;}
```

However, by entering `!important` in a rule, that rule will take precedence, regardless of its location.

In this example, the body font will be Times, not Verdana:
```css
body {font-family: Times !important;
     font-family: Verdana;}
```

**Note:** `!important` does not work with all properties in *Internet Explorer.*
Classes and IDs

HTML has two attributes that make CSS even more useful: class and ID. They make it easy to apply style to just about any tag.

Classes can describe a generic style that can be applied to any HTML element, or can be created for specific elements. When defining a style for elements with a particular class attribute in the Style Sheet, declare a rule using a dot (.) followed by the class name. To limit the style to a particular element with that class attribute, use a selector combining the tag name with a dot followed immediately by the class name.

The following rule would apply to any element with the attribute class="shade":

```
.shade { background: yellow; }
```

The following rule would apply only to paragraph tags with the class shade:

```
<p class="shade">
...</p>
```

```
p.shade { background: red; }
```

IDs are similar to classes, but IDs are unique. They can only be used with one instance of an element within a document. When defining a CSS rule using an ID-based selector, use a number sign (#) followed by the style name. To limit the style to a particular element with that ID attribute, use a selector combining the tag name with a # and then the ID name.

The following rule would apply to any element with the attribute id="intro":

```
#intro { font-size: 2em; }
```

The following rule would apply only to heading 1 tags with the ID intro:

```
<h1 id="intro">
...</h1>
```

```
h1#intro { color: green; }
```
EXAMPLE: A web page with an internal CSS style with a class called "highlight"

```html
<head>
  <style type="text/css">
    .highlight {
      background-color: #CCCCCC;
    }
  </style>
</head>

<body>
  <p>This paragraph is not highlit.</p>
  <p class="highlight">This paragraph is highlit.</p>
</body>
</html>
```
Span and Div

There are two tags that are particularly useful when using CSS: `<span>` and `<div>`. They are both container tags that have minimal formatting associated with them.

Using `<span>` and `<div>` tags in conjunction with classes and IDs allows for great flexibility in creating pages.

The `<span>` tag is an inline element that simply holds text without doing anything special to it.

The `<div>` tag is a block element and causes the text it encloses to start on a new line.

For example, to apply a font to a specific section of text, create a class, and use the span tag with that class:

In the CSS:
```
.neatstuff {font-family: Comic Sans MS;}
```

In the HTML:
```
<span class="neatstuff">This is in Comic Sans</span>
```
EXAMPLE: An example of a web page using CLASS, ID, SPAN, and DIV

```html
<head>
  <style type="text/css">
    <!--
    #BrightText {color: #00ff00; font-weight: bolder;}
    .PeriwinkleBlueText {color: #6699ff;}
    -->
  </style>
</head>

<html>
  <body>
    Baa, baa black sheep <br />
    Have you any wool? <br />
    <div id="BrightText">
      Yes sir! Yes sir!
    </div>

    Three bags full. <br />
    <span class="PeriwinkleBlueText">One</span> for my master. <br />
    <span class="PeriwinkleBlueText">One</span> for my dame. <br />
    <span class="PeriwinkleBlueText">One</span> for the little boy, <br />
    Who lives down the lane.
  </body>
</html>
```

Baa Baa Black Sheep
Have you any wool?
Yes sir! Yes sir!
Three bags full.
One for the master
One for the dame
One for the little boy
Who lives down the lane.
Pseudo-elements and Pseudo-classes

Pseudo-elements and pseudo-classes are special, predefined groupings used in CSS to deal with special situations that do not exist in standard HTML.

Pseudo-elements
Under standard HTML, there is no way to automatically change the look and feel of the first letter or line of a paragraph. But the pseudo-element :first-letter allows you to specify a style that affects the first letter, and the pseudo-element :first-line allows you to specify a style that affects the first line.

EXAMPLES:
```css
p:first-line { font-size: 200%; color:red;}
li:first-letter { border: dotted;}
```

Pseudo-classes
Under standard HTML, there is no mechanism to deal with mouse movements. But with CSS, the pseudo-class :hover can be used to change the style of a link. In the example below, a:hover is used to change the link color to red and have the underlining disappear whenever a mouse hovers over links.
```css
a:hover {color: #ff0000; text-decoration: none;)
```

To change the style of links, use the pseudo-class :link
```css
a:link {color: #00f; font-weight: bold;)
```

To change the style of visited links, use the pseudo-class :visited
```css
a:visited {color: purple; border: groove;}
```
Unit Measurements

In CSS, you can measure units either in absolute values or in relative values.

**Absolute values** are fixed, specific values. Since they are exact measurements, they allow the designer complete control over the display of the web pages.

Absolute values include:
- mm, cm, in, pt, pc, xx-small, x-small, small, medium, large, x-large, xx-large

**Relative values** have no fixed, specific values, and are calculated in comparison to something else (usually the size of the default font or line size). Relative values tend to be a better choice, because different computers have different video cards and screen sizes, and users have differing eyesight abilities. They give the designer less absolute control but often create a better experience for the visitor.

Relative values include:
- em, ex, px, larger, smaller, num%

Examples of unit measurements in a CSS file:

```css
body { font-size: 12px; }

h1, h2, h3 { line-height: 200%;}
```

**Warning:**

If you use percentages and nest one element inside another, the percentages will be cumulative.

```html
<head>
<title>Untitled Document</title>
<style type="text/css">
ul { font-size: 70%}
</style>
</head>
<body>
A list:
<ul>
  <li>Item
    <ul>
      <li>Sub item</li>
    </ul>
  </li>
</ul>
</body>
```
Inline vs. Block Display

All HTML elements (tags) are assigned a display property value of either *inline* or *block*.

Inline elements display in browsers horizontally.

```
[INLINE ELEMENT 1] [INLINE ELEMENT 2] [INLINE ELEMENT 3]
```

Examples of inline elements:

```
<a>  <img>  <strong>  <em>  <span>
```

Block elements display in browsers vertically (stacked one on top of the other).

```
[BLOCK ELEMENT 1]
[BLOCK ELEMENT 2]
[BLOCK ELEMENT 3]
```

Examples of block elements:

```
<p>  <h1-h6>  <div>  <hr>  <table>  <ul>  <ol>
```

Using CSS, you can change the inherent display property to:

- Force a block display, by using the declaration `display: block;`
- Force an inline display, by using the declaration `display: inline;`
- Force a list, by using the declaration `display: list-item;`
- Hide elements, by using the declaration `display: none;`
Using \texttt{display: block;}

Normally, \texttt{<a>} tags display inline.

\begin{verbatim}
<body>
  <a href="link1.html">Link 1</a> <a href="link2.html">Link 2</a> <a href="link3.html">Link 3</a>
</body>
\end{verbatim}

\textbf{Link 1 Link 2 Link 3}

But, if you add the style \texttt{a \{display: block;\}}, they will display as a vertical navigation menu:

\begin{verbatim}
<style type="text/css">
  \--
  a \{ display: block; \}
  \--
</style>
</head>
<body>
  <a href="link1.html">Link 1</a> <a href="link2.html">Link 2</a> <a href="link3.html">Link 3</a>
</body>
\end{verbatim}

\textbf{Link 1 Link 2 Link 3}
Using `display: inline;`

Normally, the heading tags display in block format.

```html
<body>
<h1>Heading 1</h1><h2>Heading 2</h2><h3>Heading 3</h3>
</body>
```

But, to have them display inline, add the style `h1,h2,h3 {display: inline;}.`

```html
<style type="text/css">
!---
h1,h2,h3 { display: inline;}
-->
</style>
</head>
<body>
<h1>Heading 1</h1><h2>Heading 2</h2><h3>Heading 3</h3>
</body>
```
Modifying List Elements

In HTML, by default, unordered lists (<ul>) appear as bullets and ordered lists (<ol>) appear as numbers.

Using CSS, you can modify how list items appear.

**Note:** Internet Explorer only recognizes the **bolded** values listed below.

- **Properties:**
  - list-style, list-style-type, list-style-image, list-style-position

- **Values:**
  - disc, circle, square, decimal, decimal-leading-zero, lower-roman, upper-roman, lower-alpha, upper-alpha, lower-greek, lower-latin, upper-latin, hebrew, armenian, georgian, cjk-ideographic, hiragana, katakana, hiragana-iroha, katakana-iroha, none, url("graphic.gif"), inside, outside

Examples:

```css
ul { list-style: disc; }
ol { list-style: upper-roman; }
li { list-style: url("blackball.gif");}
ul li { list-style-position: inside; }
```

The following page gives the code for all list style types and then shows how the codes are rendered in Firefox and in Internet Explorer.
HTML / CSS Code

```html
<head>
<title>Lists</title>
<style type="text/css">
</style>
</head>
<body>

<li class="a">armenian</li>
<li class="b">circle</li>
<li class="c">cjk-ideographic</li>
<li class="d">decimal</li>
<li class="e">decimal-leading-zero</li>
<li class="f">disc</li>
<li class="g">georgian</li>
<li class="h">hebrew</li>
<li class="i">hiragana</li>
<li class="j">hiragana-iroha</li>
<li class="k">katakana</li>
<li class="l">katakana-iroha</li>
<li class="m">lower-alpha</li>
<li class="n">lower-greek</li>
<li class="o">lower-latin</li>
<li class="p">lower-roman</li>
<li class="q">none</li>
<li class="r">square</li>
<li class="s">upper-alpha</li>
<li class="t">upper-latin</li>
<li class="u">upper-roman</li>
<li class="v">url(blackball.gif)</li>
<li class="w">inside</li>
<li class="x">outside</li>
</ul>
</body>
</html>

Firefox

U. armenian
° circle
¥ cjk-ideographic
4. decimal
05. decimal-leading-zero
• disc
® georgian
† hebrew
hiragana
hiragana-iroha
katakana
katakana-iroha
m. lower-alpha
n. lower-greek
o. lower-latin
xvi. lower-roman
none
□ square
S. upper-alpha
T. upper-latin
XXI. upper-roman
• url(blackball.gif)
  • inside
  • outside

Internet Explorer

• armenian
o circle
cjk-ideographic
decimal
decimal-leading-zero
disc
georgian
hebrew
hiragana
hiragana-iroha
katakana
katakana-iroha
m. lower-alpha
n. lower-greek
o. lower-latin
xvi. lower-roman
none
□ square
S. upper-alpha
T. upper-latin
XXI. upper-roman
• url(blackball.gif)
  • inside
  • outside

Web Design Level 3: Cascading Style Sheets
August 2008
Specificity

When two (or more) CSS rules refer to the same element, the browser determines which rule relates more to that element (is more specific) and applies that rule. To calculate the specificity of a rule, browsers give IDs a score of 100, classes a score of 10, and HTML selectors a score of 1. The rule with the highest score is invoked.

Suppose you have the following CSS:

div#cool p { color: red; }
.nifty { color: green; }
#cool { color: purple; }
p { color: blue; }

And you also have the following HTML, and that the above CSS has been invoked:

```html
<div id="cool"><p class="nifty">What color am I?</p></div>
```

You might think the color of the text in the paragraph would be blue, since `p` was the last rule to be written. You might think the color would be purple, since the ID `#cool` defines color text to be purple. Or, you might think the color would be green, since the class `.nifty` defines color text to be green.

In actuality, the browser displays the paragraph in red text. The more specific a selector, the more preference it is given in regards to styles that are otherwise in conflict (and `div#cool p` is more specific than `p`, `.nifty`, or `#cool`).

So, in the example above, `p` has a specificity of 1 (1 HTML selector); `div p` has a specificity of 2 (2 HTML selectors); `.nifty` has a specificity of 10 (1 class selector); `div p.nifty` has a specificity of 12 (2 HTML selectors plus 1 class selector); and `#cool` has a specificity of 100 (1 ID selector).

Even though `div#cool p` is the first rule listed, it is the one invoked, since it has the highest score (102) in this group of styles:

<table>
<thead>
<tr>
<th>Specificity Score</th>
<th>CSS rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>102</td>
<td><code>div#cool p { color: red; }</code></td>
</tr>
<tr>
<td>100</td>
<td><code>#cool { color: purple; }</code></td>
</tr>
<tr>
<td>10</td>
<td><code>.nifty { color: green; }</code></td>
</tr>
</tbody>
</table>
p { color: blue; }
The Box Model

When a browser draws an object on a page, it places it into an invisible rectangular space called a “bounding box.” (For more on the bounding box, see page 36.)

You can specify the size, look, and feel of the margins, the padding, the border, and the content of that bounding box.

In CSS1, the width property is defined as the distance between the left and right edges of the bounding box that surrounds the element’s content.

Likewise, the height property is defined in CSS as the distance between the top and bottom edges of the bounding box.

Internet Explorer interprets CSS box styles differently than most other web browsers. In Internet Explorer, the width and height properties also include the border and padding belts that surround the element’s bounding box.
Positioning

Using CSS, you can place elements exactly on a page using a technique called “positioning.” Positioning is determined by an X axis and Y axis. To specify a point on the screen, you can use the X and Y coordinates for that point.

There are several ways to specify position in CSS: absolute, relative, fixed, inherit, and static.

The three most often used are absolute, relative, and fixed.

- **Absolute positioning** defines the position of a given bounding box from the top and left side margins of the web page. This not only allows objects to be placed in an exact location, it also allows objects to be placed one on top of another.

- **Relative positioning** defines positioning in such a way that elements are offset from the previous element in the HTML code. This allows objects to be placed in relation to one another.

- **Fixed positioning** defines the position of a given box relative to the window and remains in its specified location even as the content scrolls underneath it. This value does not work in Internet Explorer 6 or earlier. In Internet Explorer 7, the browser must be in “standards-compliance mode.”

- **Inherit positioning** explicitly sets the value to that of the parent. If the parent is position:absolute, the child will be position:absolute. If the parent is position:fixed, the child will be position:fixed.

- **Static positioning** is the default. It essentially defines the position of a given box as an unpositioned element. It flows in the normal rendering sequence of the web page.

**Note:** Internet Explorer 6 only recognizes absolute and relative positioning. Internet Explorer 7 recognizes fixed positioning in standards-compliance mode. To force standards-compliance mode, make sure your web pages are created using strict XHTML.
Using `position: absolute;`

```html
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Pink Box</title>
<style type="text/css">
<!--
#pinkbox {
    position: absolute;
    width: 200px;
    background-color: pink;
    left: 0px;
    top: 150px;
    height: 200px;
}
-->
</style>
</head>

<body>
<div id="pinkbox">This is a 200 by 200 pixel pink box located 150 pixels from the top of the web page.</div>
</body>
</html>
```
Using position: relative;

```html
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Offset example</title>
<style type="text/css">
!--
 .offset {
    position: relative;
    top: 150px;
    left: 50px;
   }
-->
</style>
</head>

<body>
<p>This text will flow normally across the screen. The next line will be offset from the end of this line.
<spam class="offset">This text is offset from the line above. Neat, huh?</spam> </p>
</body>
</html>
```

This text will flow normally across the screen. The next line will be offset from the end of this line.

This text is offset from the line above. Neat, huh?
Using position: fixed;

```html
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
<title>Untitled Document</title>
<style type="text/css">

#links {
    position: fixed;
    border: dotted;
    border-color: #000000;
    width: 20%;
    height: 100%;
    z-index: 1;
    left: 0px;
    top: 0px;
    background-color: #FFFFFF;
}

#main {
    position: absolute;
    left: 25%;
    top: 50%;
    width: 75%;
}
</style>
</head>
<body>

<div id="main">


</div>

</body>
</html>
```

Fusce purus lectus, ultricies nec, aliquam at, facilisis id, arcu. Vestibulum quis mi


Layers and the Bounding Box

The Box Model
When the browser draws an object on a page, it places it into an invisible, rectangular space called a bounding box. You can set the box’s exact location on the page or offset it from other objects on the page. As mentioned in the previous pages, you can also specify the size of the box. For more information, see the previous section on the box model (page 28).

With CSS, these boxes can be stacked one on top of another as layers. Horizontal and vertical positioning happen along the X and Y axes, and the layered positioning happens along the Z axis.

The z-index Style
The Z axis is set using the CSS style `z-index`, which allows you to specify which layer appears on top of the others. By setting the `z-index` higher or lower, an object can move up and down a stack. The higher the `z-index`, the more “on top” it is.
Layering Example 1

```html
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Untitled Document</title>
<style type="text/css">
  .over {
    font-family: "Courier New", Courier, mono;
    font-size: 200%;
    color: #000000;
    background-color: #FFFF00;
    border: thick dotted #990000;
    position: absolute;
    left: 40px;
    top: 5px;
    z-index: 2;
  }
  .under {
    font-family: Arial, Helvetica, sans-serif;
    font-size: 75%;
    font-weight: bold;
    color: #FFFFFF;
    background-color: #009900;
    border: thin dashed #0000FF;
    position: absolute;
    left: 20px;
    top: 20px;
    z-index: 1;
  }
</style>
</head>

<body>
<p class="under">Row, row, row your boat, gently down the stream.</p>
<p class="over">Merrily, merrily, merrily, merrily, life is but a dream!</p>
</body>
```

```
Merrily, merrily,
merrily, merrily,
life is but a dream!
```
Layering Example 2

```html
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>Layering Example 2</title>
    <style type="text/css">
      .over {
        font-family: "courier new", courier, mono;
        font-size: 200%;
        color: #000000;
        background-color: #FFFFFF;
        border: thick dotted #990000;
        position: absolute;
        left: 40px;
        top: 5px;
        z-index: 1;
      }
      .under {
        font-family: Arial, Helvetica, sans-serif;
        font-size: 75%;
        font-weight: bold;
        color: #000000;
        background-color: #009900;
        border: thin dashed #0000FF;
        position: absolute;
        left: 20px;
        top: 20px;
        z-index: 2;
      }
    </style>
  </head>
  <body class="under">
    Row, row, row your boat, gently down the stream.
    Merrily, merrily, merrily, merrily, life is but a dream!
  </body>
</html>
```

Row, row, row your boat, gently down the stream.
Merrily, merrily, merrily, merrily, life is but a dream!
Float

If you want to wrap content around other content (such as text around a picture), you can use the `float` property.

The `float` property determines on which side of the bounding box the element aligns, so that the other content wraps around it.

**Using float: right;**

```html
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>&quot;float: right&quot; example</title>
<style type="text/css">
  .floatright { float: right; }
</style>
</head>
<body>
<div id="floatright"><img src="down_seal3.gif" width="122" height="118" alt="Stanford Seal" /></div>
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam mattis, erat at tempus porta, velit dui tempus metus, sit amet ullamcorper nibh nibh sed nulla. Nam consectetur, metus id vulputate tincidunt, libero risus vestibulum leo, eu dictum augue diam in risus. Vestibulum eleifend nisi congue pede. Curabitur non nulla quis pede aliquam accumsan. &lt;/p&gt;
</p>
</body>
</html>
```
Using `float: left;`

```html
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>"float: left" example - Microsoft Inte... - ...</title>

<style type="text/css">
    .floatleft { float: left; }
</style>

</head>

<body>
    <div id="floatleft"><img src="down_seal3.gif" width="122" height="118" alt="Stanford Seal" /></div>
    <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam mattis, erat at tempus porta, velit dui tempus metus, sit amet ullamcorper nibh nibh sed nulla. Nam consectetur, metus id vulputate tincidunt, libero risus vestibulum leo, eu dictum augue diam in risus. Vestibulum eleifend nisi congue pede. Curabitur non nulla quis pede aliquam accumsan.</p>
</body>
</html>
```
Resources

- **A List Apart**: articles on practical issues and suggestions for working with CSS correctly  
  [http://www.alistapart.com/topics/code/css](http://www.alistapart.com/topics/code/css)

- **Sample XHTML Pages, with and without the CSS Style Sheet**  
  [http://techbriefings.stanford.edu/web_standards/example1.html](http://techbriefings.stanford.edu/web_standards/example1.html)  
  [http://techbriefings.stanford.edu/web_standards/example2.html](http://techbriefings.stanford.edu/web_standards/example2.html)  
  [http://techbriefings.stanford.edu/web_standards/example.css](http://techbriefings.stanford.edu/web_standards/example.css)

- **Tech Training Services Cascading Style Sheets reference web site**  
  [http://techtraining.stanford.edu/css](http://techtraining.stanford.edu/css)

- **The CSS Zen Garden** shows some of the most advanced uses of CSS  

- **CSS in the real world: ajc.com’s ‘News Break’**  

- **Microsoft’s CSS Information:**  

- **Microsoft’s Style Sheet Demonstrations**  

- **W3C Style Examples**  
  [http://www.w3.org/Style/Examples/007](http://www.w3.org/Style/Examples/007)

- **W3C CSS 2.1 Specifications**  
  [http://www.w3.org/TR/CSS21/](http://www.w3.org/TR/CSS21/)

- **W3Schools CSS Tutorial**  
  [http://www.w3schools.com/css](http://www.w3schools.com/css)

- **W3Schools CSS Reference**  
  [http://www.w3schools.com/css/css_reference.asp](http://www.w3schools.com/css/css_reference.asp)

- **Webmonkey’s Cascading Style Sheet Guide**  

- **Brian Wilson’s Cascading Style Sheet Reference Guide**  