Web Programming with CSS

Christophe Lecoutre
lecoutre@cril.fr

IUT de Lens - CRIL CNRS UMR 8188
Université d'Artois
France

Département SRC - 2010/2011

MCours.com

Outline

Outline

Bibliography

Books

- John Duckett.
  Beginning Web Programming with HTML,
  XHTML, and CSS,
  2nd Edition, Wrox. 2008.

- ...

Sites

- W3 Specifications at `http://www.w3.org/Style/CSS/#specs`
- W3School-CSS at `http://www.w3schools.com/css`
- W3C Validator at `http://jigsaw.w3.org/css-validator`
  _____

- SelfHtml-CSS at `http://fr.selfhtml.org/css/`
- Zen Garden at `http://www.mezzoblue.com/zengarden/resources/`
- DevGuru at
  `http://www.devguru.com/technologies/css2/CSS_PDF181.zip`

Objective

XHTML is appropriate to structure the content of your documents. CSS (Cascading Style Sheets) takes control of the style of your pages, including:

- the colors and sizes of fonts,
- the width and colors of lines,
- the amount of white space between items
- ...

Several versions of CSS:

- CSS1 (CSS level 1), the first CSS specification to become an official W3C Recommendation,
- CSS2 (CSS level 2), published as a Recommendation in May 1998,
- CSS3, under development since December 2005.

Warning

Do not use elements and/or attributes of HTML to control how a document should appear.

Rules

Rules are associated with the elements that appear in the document. These rules govern how the content of those elements should be rendered. Each rule is made of two parts:

- a selector: indicates which element(s) the declaration applies to by means of a comma-separated list of elements;
- a declaration: set outs between curly brackets how the elements should be styled by means of a semicolon-separated list of property:value pairs where
    - the property refers to the selected element(s)
    - the value is a specification for the property

Rules
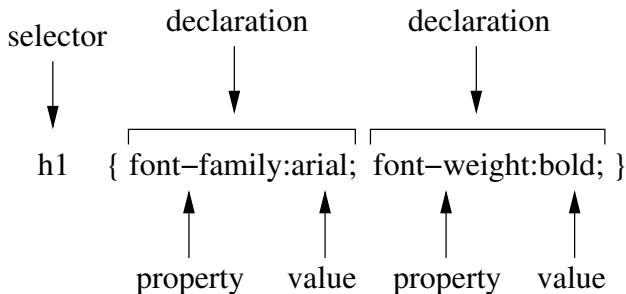
```
h1, h2, h3 {
    font-weight:bold;
    font-family:arial, verdane, sans-serif;
    color:#00000;
    background-color:#FFFFFF;
}
```

In this example, the content of all elements <h1>, <h2>, <h3> present in the document will be:

- written in bold Arial Font (or bold Verdane font, if Arial is not installed, or bold sans-serif font if both Arial and Verdane are not installed);
- written in black color with a white background color.

Remark

Always end each property:value pair with ; even if this is the last one in the sequence.

External Style Sheet

Tu use a separate file containing the different rules, you have to use an empty `<link>` element inside the `<head>` element of the document. When used with style sheets, the `<link>` element must carry three attributes:

- `rel` whose value must be "stylesheet";
- `type` whose value must be "text/css";
- `href` whose value must be an absolute or relative URL corresponding to the auxiliary file.

Example

Suppose we have a file *mySheet.css* in a subdirectory *css* whose content is:
```
em { color:red; }
```
In a XHTML document, we can use it as follows:

```
<head>
  ...
  <link rel="stylesheet" type="text/css"
        href="css/mySheet.css" />
  ...
```

External Style Sheet

The <link> element also accepts the media attribute. This attribute allows us to associate different style sheets to different output devices. Some of tts values are:

Value	Use
screen	Computer screens (this is default)
tv	Television type devices (low resolution, limited scroll ability)
handheld	Handheld devices (small screen, limited bandwidth)
print	Print preview mode/printed pages
aural	Speech synthesizers

Example

```
<link rel="stylesheet" type="text/css" href="theme.css" />
<link rel="stylesheet" type="text/css" href="print.css"
      media="print"/>
```

Remark

Comments in CSS files are put between /* and */

Internal Style Sheet

When the style sheet rules are held inside a `<style>` element contained in the `<head>` element, they are reffered to as an internal style sheet. The `<style>` element must carry the type attribute whose value must be "text/css".

Example

```
<link rel="stylesheet" type="text/css" href="mySheet.css" />
<style type="text/css">
   h1 { color:red; }
</style>
```

Remark

An internal style sheet may be appropriate to include just a few extra rules that do not apply to the other documents that share the same style sheet.

Inline Style Rules

When `style` attributes are used on XHTML elements, they are known as *inline style rules*. The style is directly applied to the element that carries the `style` attribute.

Example

```
<td style="font-family:courier; padding:5px;"> 10 </td>
<td style="border-style:solid; border-width:1px;"> 20 </td>
```

Warning

Do not use inline style rules !!
The `style` attribute is not allowed in strict XHTML 1.0

Outline

- a simple *type* selector specifies the name of a single element type; e.g.,

    p { }

- a *type* selector specifies several element types in a comma-delimited list; the rule applies to all element types of the list; e.g.,

    h1, h2, p { }

- the *universal* selector * governs all element types in the document; e.g.,

    * { }

- a *class* selector governs all elements carrying a class attribute with the value specified in the selector (preceded by a period); e.g.,

    .rouge { }

- a class selector can be combined with a type selector to limit the rule to the elements of the given type; e.g.,

    p.rouge { }

- an id selector governs the unique element carrying a id attribute with the value specified in the selector (preceded by #); e.g.,

    #idTable1 { }

- a *child* selector governs an element type that is a direct child of another; a child selector is made up of two or more selectors separated by >, e.g.,

      td > strong { }

- a *descendent* selector governs an element type that is a descendent of another; a descendent selector is made up of two or more selectors separated by a whitespace, e.g.,

      table strong { }

- an *adjacent sibling* selector governs an element type that is the next sibling of another; an adjacent sibling selector is made up of two types separated by $+$, e.g.,

      h1 + p { }

Remark

The child and adjacent sibling selectors add a lot of flexibility to how you style documents (and may avoid adding classes).

Attribute Selectors

- an *existence* selector [att] governs all elements carrying an attribute att
- an *equality* selector [att="val"] governs all elements carrying an attribute att whose value is exactly "val"
- a *space* selector [att~="val"] governs all elements carrying an attribute att whose value is a list of space-separated words, one of which is exactly "val"
- a *hyphen* selector [att|="val"] governs all elements carrying an attribute att whose value is exactly "val" or begins with "val" and is followed by a hyphen

Example

```
h1[title] { background-color: gray; }
a[href="http://www.w3.org/"] { color: blue; }
a[rel~="copyright"] { color:red; }
*[lang="fr"] { display:none; }
```

CSS introduces the concepts of pseudo-elements and pseudo-classes to permit formatting based on information that lies outside the document tree.

- Pseudo-elements create abstractions about the document tree beyond those specified by the document language. For instance, document languages do not offer mechanisms to access the first letter or first line of an element's content. Pseudo-elements may also provide style sheet designers a way to assign style to content that does not exist in the source document (e.g., the :before and :after pseudo-elements give access to generated content).
- Pseudo-classes classify elements on characteristics other than their name, attributes or content; in principle characteristics that cannot be deduced from the document tree. Pseudo-classes may be dynamic, in the sense that an element may acquire or lose a pseudo-class while a user interacts with the document. The exceptions are :first-child, which can be deduced from the document tree, and :lang(), which can be deduced from the document tree in some cases.

Pseudo-classes are allowed anywhere in selectors while pseudo-elements may only be appended after the last simple selector.

Pseudo-classes

- `:first-child`: this pseudo-class matches an element that is the first child element of some other element
- `:link`: this pseudo-class applies for a link that has not yet been visited
- `:visited`: this pseudo-class applies for a link that has been visited
- `:hover`: this pseudo-class applies while the user designates an element (with some pointing device), but does not activate it
- `:active`: this pseudo-class applies while an element is being activated
- `:focus`: this pseudo-class applies while an element has the focus

Example

```
div > p:first-child { text-indent:0px; }
p:first-child em { font-weight:bold; }
a.external:visited { color:blue; }
a:focus:hover { background-color:white; }
```

Warning

`a:hover` must be placed after the `a:link` and `a:visited` rules.

- `:first-line`: this pseudo-element applies special styles to the contents of the first formatted line of a paragraph
- `:first-letter`: this pseudo-element selects the first letter of the first line of a block, if it is not preceded by any other content (such as images or inline tables) on its line. The :first-letter pseudo-element may be used for "initial caps" and "drop caps", which are common typographical effects
- `:before`: this pseudo-element can be used to insert generated content before an element's content
- `:after`: this pseudo-element can be used to insert generated content after an element's content

Example

```
p.pageOne:first-letter { font-size:42px; width:200px; }
p.pageOne:first-line { font-weight:bold; }
h1:before { content:url(images/penguin.jpeg); }
h2:after { content:"(you need to complete the story)"; }
```

The content property is used with the :before and :after pseudo-elements, to insert generated content. Some possible values are:

- a string (containing no quotes)
- a URL
- a counter to number elements on the page
- open-quote and close-quote: inserts the appropriate opening and closing quotes
- no-open-quote and no-close-quote: introduces no content, but increments (decrements) the level of nesting for quotes.

Example

```
body { counter-reset:chapter; }  /* Set chapter to 0 */
h1:before { content:"Chapter " counter(chapter) ": ";
  counter-increment:chapter; }   /* Add 1 to chapter */
h1 { counter-reset:section; }    /* Set section to 0 */
h2:before { content:counter(chapter) "." counter(section) ": ";
  counter-increment:section; }   /* Add 1 to section */
```

CSS @ Rules

There are some additional rules:

- @import imports another style sheet into the current one
- @charset indicates the character set of the style sheet
- @font-face describes a font face
- !important applied to a property indicates that that the property will always be applied, no matter what are the other rules.

Example

```
@import "mySheet1.css"      /* first way */
@import url("mySheet2.css") /* second way */
@charset "iso-8859-1"
p { font-size:18px !important; }
```

Warning

The @import and @charset rules should appear right at the start of the document (without even a space before them).

Outline

MCours.com

The following table shows the main properties available to you from CSS1 and CSS2.

FONT	FONT (ct.)	TEXT (ct.)	TEXT (ct.)
font	font-variant	text-align	white-space
font-family	font-weight	text-decoration	word-spacing
font-size	TEXT	text-indent	BACKGROUND
font-size-adjust	color	text-shadow	background
font-stretch	direction	text-transform	background-attachment
font-style	letter-spacing	unicode-bidi	background-color

BACKGROUND (ct.)	BORDER (ct.)	DIMENSIONS (ct.)	TABLE (ct.)
background-image	border-top-style	min-width	table-layout
background-position	border-top-width	width	LIST and MARKER
background-repeat	border-width	POSITIONING	list-style
BORDER	MARGIN	bottom	list-style-image
border	margin	clip	list-style-position
border-bottom	margin-bottom	left	list-style-type
border-bottom-color	margin-left	overflow	marker-offset
border-bottom-style	margin-right	right	CONTENT
border-bottom-width	margin-top	top	content
border-color	PADDING	vertical-align	counter-increment
border-left	padding	z-index	counter-reset
border-left-color	padding-bottom	OUTLINES	quotes
border-left-style	padding-left	outline	CLASSIFICATION
border-left-width	padding-right	outline-color	clear
border-right	padding-top	outline-style	cursor
border-right-color	DIMENSIONS	outline-width	display
border-right-style	height	TABLE	float
border-right-width	line-height	border-collapse	position
border-style	max-height	border-spacing	visibility
border-top	max-width	caption-side	
border-top-color	min-height	empty-cells	

Many properties use lengths as values. Lengths can be measured in three ways:

- absolute units
  - ► pt (point); 1 pt is equivalent to $1/72$ of an inch
  - ► pc (pica); 1 pc is equivalent to 12 points (or $1/12$ of an inch)
  - ► in (inch); 1 in is equivalent to 2.54 centimeters
  - ► cm (centimeter)
  - ► mm (miliimeter)
- relative units
  - ► px (pixel); this is the smallest unit of resolution on a screen; it is then relative to the resolution of the viewing device;
  - ► em; an em unit corresponds directly to the font size of the reference element which is either that element or the containing element
  - ► ex; the ex unit should be the height of a lowercase x in the font of the element
- percentages; a percentage gives a value in relation to another value

Remark

Relative units and percentages can adjust size with the kind of media that the document is shown on.

Colors

Some properties use colors as values. A color can be defined by:

- a hex code, i.e, a six-digit code (in hexadecimal) representing the amount of reg, green and blue that make up the color, preceded by #
- a rgb expression of the form $rgb(x, y, z)$ where x, y and z are an integers between 0 and 255 inclusive
- a rgb expression of the form $rgb(x\%, y\%, z\%)$ where x, y and z are numbers between 0 and 100 inclusive
- a color name in: aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, white , and yellow.

Color	Hexadecimal Code	rgb (form 1)	rgb (form 2)
blue	#0000FF	rgb(0,0,255)	rgb(0%,0%,100%)
olive	#808000	rgb(128,128,0)	rgb(50%,50%,0%)
silver	#C0C0C0	rgb(192,192,192)	rgb(75%,75%,75%)

Example

```
p  { color:gray; background-color:rgb(120,45,203); }
td { color:#01E2C1; }
```

Controlling Fonts

The following properties allow you to directly affect the font and its appearance.

Property	Purpose
font	Allows you to combine several of the following properties into one
font-family	Specifies the family of font to be used (the user must have this installed on his or her computer)
font-size	Specifies the size of a font
font-weight	Specifies whether the font should be normal, bold, or bolder than the containing element
font-style	Specifies whether the font should be normal, italic, or oblique (an oblique font is the normal font on a slant rather than a separate italic version of the font)
font-stretch	Allows you to control the width of the actual letters in a font (not spaces between them)
font-variant	Specifies whether the font should be normal or small caps
font-size-adjust	Allows you to alter the aspect ratio of the size of characters of the font

A font is not the same thing as a typeface:

- a typeface is a family of fonts that are either serif fonts or sans-serif fonts; e.g. Times and Arial are two typefaces;
- a font is a specific member of a typeface family, such as Arial 12-point bold.

SERIF FONTS

SERIFS

img

SANS-SERIF FONTS

img

Times New Roman
Book Antiqua
Garamond
Times Family

Arial
Gill Sans
Verdana
Century Gothic

Comic Sans MS
Monotype Corsiva
COPPERPLATE
ALGERIAN

SERIF FONTS
FOR RESUME TEXT

SANS-SERIF FONTS
FOR HEADINGS, NAME

CURSIVE, DECORATIVE
FONTS – AVOID

Proportional

Monospace

Serif

The small decorative
pieces on the ends of
each character are called
Serifs

San Serif

The `font-family` Property

The `font-family` property can hold several font names as a "fallback" system. If the browser does not support the first font, it tries the next font, etc. There are two types of font family names:

- the name of a font-family, like "times", "courier", "arial", etc.
- the name of a generic-family, like "serif", "sans-serif", "cursive", "fantasy", "monospace".

Remark

If a font name contains spaces, such as `times new roman`, you have to place the name in double quotation marks.

Example

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

The font-size Property

The font-size enables you to specify a size for a font in different ways:

- absolute size: "xx-small", "x-small", "small", etc.
- relative size: "smaller", "larger"
- length: expressed in px, em, ex, pt, in, cm or mm
- percentage (in relation to the parent element)

Example

```
p.one { font-size:xx-small; }
p.twelve {font-size:12px; }
p.thirtheen {font-size:3pc; }
p.fourteen { font-size:10%; }
```

Other Font Properties

Among other font properties, we find:

- font-weight: "normal", "bold", "bolder", "lighter", "100", "200", etc.
- font-style: "normal", "italic", "oblique"
- font-variant: "normal", "smallcaps"

Example

```
p.one { font-weight:bold; }
p.twelve {font-style:italic; }
p.thirtheen {font-weight:bold; font-variant:smallcaps; }
```

Formatting text

The following properties allow you to directly affect the appearance or formatting of your text.

Property	Purpose
color	Specifies the color of the text
text-align	Specifies the alignment of the text within its containing element
vertical-align	Vertical alignment of text within containing element and in relation to containing element
text-decoration	Specifies whether the text should be underlined, overlined, strikethrough, or blinking text
text-indent	Specifies an indent from the left border for the text
text-transform	Specifies that the content of the element should all be uppercase, lowercase, or capitalized
text-shadow	Specifies that the text should have a drop shadow
letter-spacing	Controls the width between letters (known to print designers as kerning)
word-spacing	Controls the amount of space between each word
white-space	Specifies whether the white space should be collapsed, preserved, or prevented from wrapping
direction	Specifies the direction of text (similar to the dir attribute)
unicode-bidi	Allows you to create bidirectional text

Other Text Properties

Among other text properties, we find:

- text-align: "left", "right", "center", "justify"
- text-decoration: "underline", "overline", "line-through", "blink"
- text-indent: a unit of length to indent the first line of text within an element
- text-transform: "none", "capitalize", "uppercase", "lowercase"

Example

```
p { text-align:justify; }
p.twelve { text-decoration:underline; }
p.thirtheen { text-indent:3em; }
td { text-transform:uppercase; }
```

The Box Model

Every element gets treated as a box in CSS, and every box has three properties:

- `border`: even if you cannot see it, it exists;
- `margin`: this is the space between the border and the box next to it;
- `padding`: this is the space between the content of the box and its border .

The Box Model
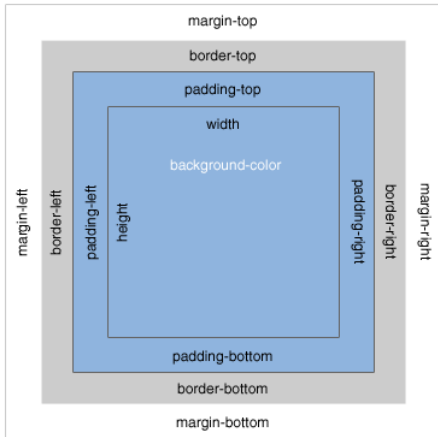
The box around inline elements flows within its containing element. The box around block level elements takes up the full width of the page.

Example

```
body, h1, p, img, b {
  border-style:solid;
  border-width:2px;
  border-color:#000000;
  padding:2px;
}
```

Remark

When a bottom margin of one element meets the top margin of another, only the larger of the two will show.

The Border properties

The border properties are:

- `border-color`: you can use hex codes and color names, but also rgb expressions of the form rgb(x,y,z) or rgb(a%, b%, c%) where x, y, z are values between 0 and 255, and a, b, c are values between 0 and 100.
- `border-style`: "none", "solid", "dotted", "dashed", "double", "groove",...
- `border-width`: "thin", "medium", "thick", or a length (not a percentage)

Remark

You can individually change the property values of the right, bottom, left and top borderx. For example, you can use the properties `border-color-right`, `border-color-bottom`, `border-color-left`, and `border-color-top`.

Remark

The `border` property allows you to spcify color, style and width in one property:
```
p { border:  4px solid red; }
```

The border properties are:

- `padding`: a length, a percentage (of the containing box) or "inherit" (the padding value of the parent element)
- `margin`: a length, a percentage (of the containing box) or "inherit" (the padding value of the parent element)
- `border-width`: "thin", "medium", "thick", or a length (not a percentage)

Remark

You can individually change the property values of the right, bottom, left and top borderx. For example, you can use the properties `border-color-right`, `border-color-bottom`, `border-color-left`, and `border-color-top`.

Remark

The `border` property allows you to spcify color, style and width in one property:

```
p { border:  4px solid red; }
```

The dimensions of a box are controlled by the following attributes:

- `height` and `width`: a length, a percentage or "auto" (the default value)
- `line-height`: allows you to increase the space between lines of text
- `min-width` and `max-width`: specifies a minimum and maximum width for a box; these properties will prevent a box from being so wide or so narrow that it is hard to read
- `min-height` and `max-height`: specifies a minimum and maximum height for a box; because of these properties, the content can overflow out of the box
- `overflow`: specifies what to do when overflow happens; its value may be:
  - ▶ "visible": the overflow is not clipped; it renders outside the element's box (default value)
  - ▶ "hidden": the overflow is clipped, and the rest of the content will be invisible
  - ▶ "scroll": the overflow is clipped, but a scroll-bar is added to see the rest of the content; the scroll-bar is always present
  - ▶ "auto": if overflow is clipped, a scroll-bar should be added to see the rest of the content
  - ▶ "inherit"

Background

The following properties allow you to specify how the background of either the whole browser window or any individual box should appear.

Property	Purpose
background-color	Specifies a color that should be used for the background of the page or box
background-image	Sets an image to be in the background of a page or box
background-repeat	Indicates whether the background image should be repeated across the page or box
background-attachment	Indicates a background image should be fixed in one position on the page, and whether it should stay in that position when the user scrolls down the page or not
background-position	Indicates where an image should be positioned in either the window or the containing box
background	A shorthand form that allows you to specify all of these properties

- `background-image`: allows you to add an image to the background of any box; the syntax is:

    `background-image=url(imageFilename)`

  This property overrides the `background-color` property. By default, the background-image property repeats an image both horizontally and vertically.
- `background-repeat`: its possible values are "repeat", "repeat-x", "repeat-y", and "no-repeat"
- `background-position`: its possible values are "x% y%", "x y", "center", "top", ...
- `background-attachment`: allows you to specify that the image must stay in the same position even when the page is being scrolled; its possible values are "scroll", "fixed", and "inherit"
- `background`: allows you to specify all the properties in any order

Example

```
background-image:url("images/penguin.jpeg");
background-attachment:fixed; background-position:center;
background-repeat:repeat-x; background-color:#eaeaea;
```

The list properties are shown in the following table:

Property	Purpose
list-style-type	Allows you to control the shape or appearance of the marker (bullet point or number)
list-style-position	Specifies whether a long item that takes up more than one line of text and therefore wraps to a second line should align with the first line or start underneath the start of the marker
list-style-image	Specifies an image for the marker rather than a bullet point or number
list-style	Serves as shorthand for the preceding properties
marker-offset	Specifies the distance between a marker and the text in the list

Controlling Lists

list.html

- list-style-type: allows you to control the shape of the marker:

Value	Marker	Example
For unorderered lists		
none	none	
disc	filled-in circle	●
circle	empty circle	○
square	filled-in square	■
For ordered lists		
decimal	number	1,2,3
decimal-leading-zero	0 before the number	01, 02, 03
lower-alpha	lowercase alphanumeric characters	a, b, c
upper-alpha	uppercase alphanumeric characters	A, B, C
lower-roman	lowercase Roman numerals	i, ii, iii
upper-roman	uppercase Roman numerals	I, II, III

- list-style-position: "inside", "outside" or "inherit"
- list-style-image: an URL, "none" or "inherit"

Tables

Property	Purpose
border-collapse	Indicates whether the browser should control the appearance of adjacent borders that touch each other or whether each cell should maintain its style
border-spacing	Specifies the width that should appear between table cells
caption-side	Specifies which side of a table the caption should appear on
empty-cells	Specifies whether the border should be shown if a cell is empty
table-layout	Allows browsers to speed up layout of a table by using the first width properties it comes across for the rest of a column (rather than having to load the whole table before rendering it)

Remark

If you want to explictly hide or show borders for empty cells, use the
`empty-cells` property because some browsers treat empty cells differently.

Outlines

Outilnes are attached to boxes and are useful to highlight some aspects of a page. An outline does not take up space; it is almost as if the outline style sits on top of the page after it has been rendered.

Property	Purpose
outline-width	Specifies the width of the outline
outline-style	Specifies the line style for the outline
outline-color	Specifies the color of the outline
outline	Shorthand for above properties

Remark
The outline is always the same on all sides.

- `cursor`: specifies the type of mouse cursor to display when pointing on an element.
- `display`: specifies the type of box an element should generate like `"inline"` or `"block"`; this allows us to force an element to be a different type; `"none"` can be used to ignore the element (and so, not to display it).
- `visibility`: allows you to hide a box from view, although it still affects the layout of the page; two possible values are `"visible"` and `"hidden"`

Remark

You may wish to change the cursor from a pointer to a hand whenever someone hovers over an image that is a submit button.

Warning

Do not use the `visibility` property to hide confidential information.

Many of the CSS properties when applied to an element are inherited by child elements. When a more specific rule than an inherited property comes along, it overrides the inherited property.

Example

```
body { font-size:16pt; }
td { font-size:10pt; }
```

Remark

Inheritance saves you from having to write out rules for each element and makes for a more compact style sheet.

Warning

Not all properties can be inherited.

Outline

By default, block-level elements within a page will flow from top to bottom and inline elements will flow from left to right (inside a block). This is called *normal flow*. If you want to control the position of elements, you must use the following properties:

- `position` allows you to specify a position scheme for a box
- the box offset properties `top`, `right`, `bottom` and `left`
- `z-index` sets the stack order of an element
- `float` specifies whether or not a box (an element) should float (out of normal flow)
- `clear` is used to give control of boxes that appear after a floated box

Remark

Avoid using tables for positionning.

The values for the position element are:

- static: this is the same as normal flow and is the default value
- relative: the element is positioned relative to its normal position (when considering normal flow)
- absolute: An absolute position element is positioned relative to the first parent element that has a position other than static; if no such element is found, the containing block is <html>
- fixed: the element is positioned relative to a fixed point (e.g., the top-left corner of a browser window); it will not move even if the window is scrolled

When a box has a position property whose value is "relative", "absolute" or "fixed", box offset properties, whose values are a length, a percentage or "auto", are used to indicate where it should be positioned. They are:

- top: offset position from the top of the containing element
- right: offset position from the right of the containing element
- bottom: offset position from the bottom of the containing element
- left: offset position from the left of the containing element

Relative positionning takes an element and positions it in relation to where it would otherwise sit in normal flow. It is displaced from that position by an amount given by the box offset properties.

Absolute positionning removes an element from normal flow. It is displaced from the containing element by an amount given by the box offset properties.

Example

```
p.two { position:relative; left:40px; top:-40px; }
p.twoBis { position:absolute; left:50px; top:-25px; }
```

Remark

You should specify only a left or right offset and only a top or bottom offset.

Warning

Unless you set a background color for a box, it will be transparent by default, making any overlapping text an unreadbale mess.

Fixed positionning removes an element from normal flow, and besides, the box is not moved when the user scrolls down the page.

When you have boxes that are positionned using relative, absolute or fixed, you can control which of the boxes appears on top using the z-index property. The value of the z-index property is a number, and the higher the number the nearer the top that element should be displayed.

Example

```
p.two { position:absolute; left:30px; top:120px; z-index:3; }
p.three { position:absolute; left:10px; top:140px; z-index:2; }
```

The `float` element allows you to take an element out of normal flow and place it as far to the left or right of a containing box as possible within that element's padding. The floated box will be aligned with the top of the containing element.

- `left`: the box is floated to the left of the containing element and the content of the containing element will flow to the right of it.
- `right`: the box is floated to the right of the containing element and the content of the containing element will flow to the left of it.
- `none`: the box is not floated and remains positioned in normal flow.
- `inherit`: the box takes the same property as its containing element.

Remark

Block-level elements as well as inline elements can be floated.

Warning

Whenever you specify a `float` property, you should also set a `width` property indicating the width of the containing box that the floating box should take up; otherwise, it will automatically take up 100 percent.

To control elements that may flow around the content of floated elements, use the
clear property. Its possible values are:

- left: the content of the element with the clear property is cleared from the
  left side of a float (it cannot appear to the left of a floating element).
- right: the content of the element with the clear property is cleared from
  the right side of a float (it cannot appear to the right of a floating element).
- both: the content of the element with the clear property is cleared from
  either side of a float (it cannot appear to either side of a floating element).
- none: allows floating on either side.

Liquid Design vs Fixed-width Design

Considering browser statistics and screen resolution statistics that can be found at:

- http://www.w3counter.com/globalstats.php
- http://www.w3schools.com/browsers/browsers_stats.asp
- http://www.w3schools.com/browsers/browsers_display.asp

It is a good idea (in 2010) to design web pages whose content is around 980 pixels wide while relevant information about the page can be found in the top 550 pixels of the screen.

There are two kinds of web pages design:

- Liquid design is design of web pages that stretch to fit the whole screen
- Fixed-width design forces pages to a certain width or height

To follow it, you specify proportions of a page using percentage values.

Example

```
#page {
  width:95%; margin-left:auto; margin-right:auto;
  background-color:#ffffff;
  border-style:1px solid #666666; padding:20px;
  font-size:12px;
}

<body>
  <div id="page">
    <!-- content of page goes here -->
  </div>
</body>
```

To follow it, you specify lengths to indicate the dimension of the pages.

Example

```
#page {
  width:800px; margin-left:auto; margin-right:auto;
  background-color:#ffffff;
  border-style:1px solid #666666; padding:20px;
  font-size:12px;
}

<body>
  <div id="page">
    <!-- content of page goes here -->
  </div>
</body>
```

For example, one column and three rows: the first contains the company name or logo, the second contains the navigation and the third contains the content.

Example

```
<body>
  <div class="page">
    <div class="header">
        COMPANY NAME/LOGO goes here
    </div>
    <div class="nav">
       <!-- NAVIGATION goes here -->
    </div>
    <div class="content">
      <!-- PAGE CONTENT goes here -->
    </div>
 </div>
</body>
```

Keep the same code, but change the CSS rules.

Example

```
.page {
  width:700px; margin-left:auto;  margin-right:auto;
  background-image:url(images/2columnbackground.gif);
  background-repeat:repeat-y;
  ...
}
.nav { float:left; width:100px; ... }
.content { margin-left:100px; ... }
...
```

MCours.com

Remark

The image must be 1 pixel tall.