

CSS: Cascading Style Sheets



Laura Farinetti

Dipartimento di Automatica e Informatica

Politecnico di Torino

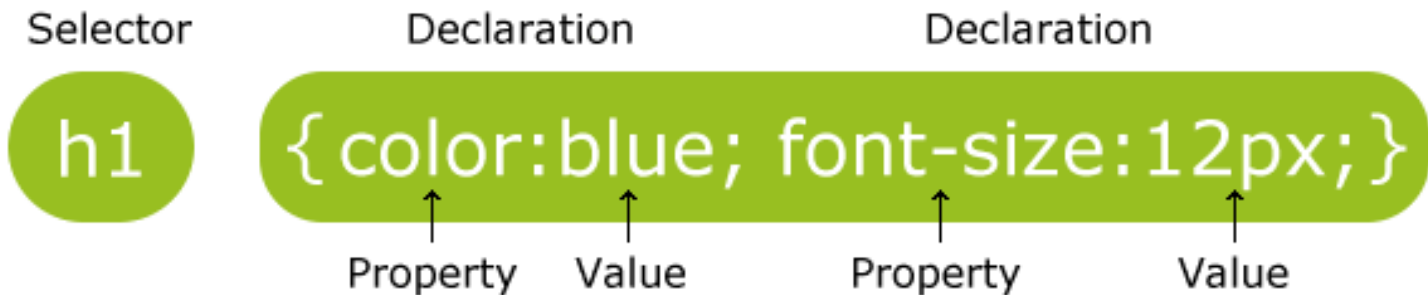
laura.farinetti@polito.it

Cascading Style Sheets

- CSS: Cascading Style Sheet
- CSS 1: W3C recommendation (17 Dec 1996)
- CSS 2.1: W3C Recommendation (7 June 2011)
- CSS 3: different stages (REC, PR, CR, WD)
 - see <http://www.w3.org/Style/CSS/current-work>
- Resources:
 - CSS 2.1 standard, <http://www.w3.org/TR/CSS21/>
 - W3C CSS Tutorial, <http://www.w3.org/Style/Examples/011/firstcss>

CSS Syntax

- CSS is based on rules
- A rule is a statement about one stylistic aspect of one or more XHTML element
- A style sheet is a set of one or more rules that apply to an XHTML document



Cascading Style Sheets

- The term “cascading” means that a document can include more than one style sheet
- In this case, visualization follows priority rules
 - User Style
 - Inline Style (inside HTML tag)
 - Internal Style (usually in the HTML head section)
 - External Style
 - Browser Default Style



External style

- Link to an external style sheet using the `<link>` element

stile.css

```
h1 { font-size:17px;
      font-family:verdana; color:green; }
h2 { font-size:18px;
      font-family:arial; color:red; }
```

```
<head>
<link rel=stylesheet type="text/css"
      href="stile.css">
</head>
<body>
<h1>Questo testo e' di colore verde, e utilizza
il font verdana a 17 pixel</h1>
<h2>Questo testo e' di colore rosso, e utilizza
il font arial a 18 pixel</h2>
</body>
```

External style

- Alternative method
- `@import` directive in the `<style>` element

```
<head>
  <style>
    @import url(stile.css);
  </style>
</head>
<body>
  ...
</body>
```

Internal style

- `<style>` element inside the document header

```
<head>
<style type="text/css">
h1 { font-size:17px; font-family:verdana;
color:green; }
h2 { font-size:18px; font-family:arial;
color:red; }
</style>
</head>
```

Inline style

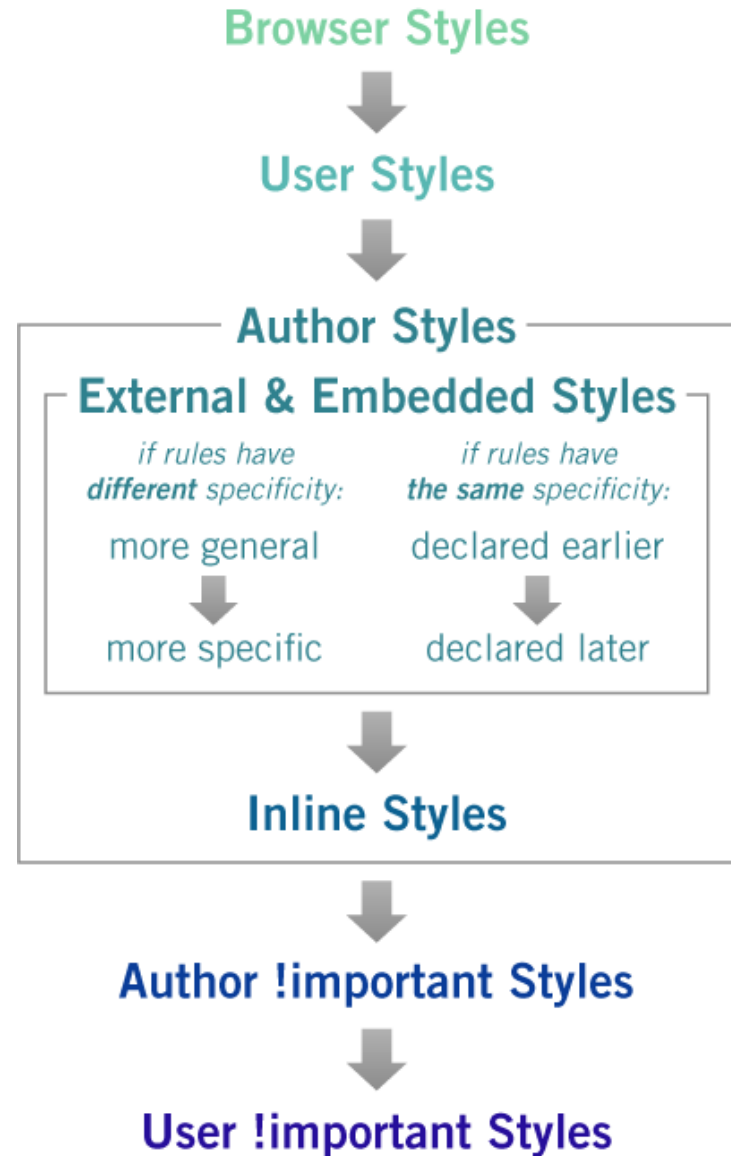
- `<style>` attribute within an XHTML element

```
<h1 style="font-size:17px;  
font-family:verdana; color:green; "> Questo  
testo e' di colore verde, e utilizza il  
font verdana a 17 pixel </h1>
```


Priority rules

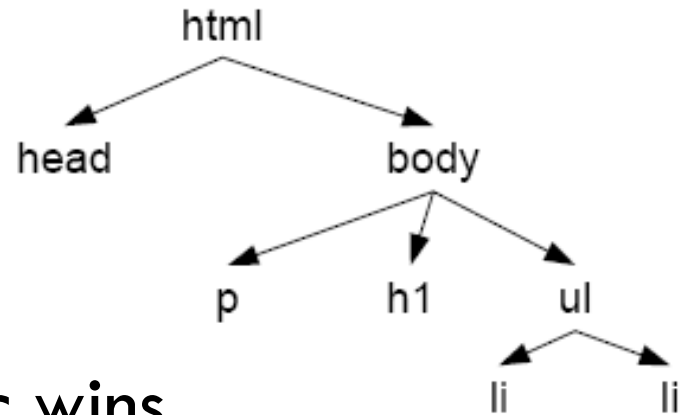
- Rules can be marked as “important”

```
h1 {  
  color:red !important  
}
```



Tree structure and inheritance

- XHTML documents are trees
- Styles are inherited along trees



- When two rules are in conflict the most specific wins
- Example

```
body {color: green}
h1 {color: red}
```

Pattern	Meaning
*	Matches any element.
E	Matches any E element (i.e., an element of type E).
E F	Matches any F element that is a descendant of an E element.
E > F	Matches any F element that is a child of an element E.
E:first-child	Matches element E when E is the first child of its parent.
E:link E:visited	Matches element E if E is the source anchor of a hyperlink of which the target is not yet visited (:link) or already visited (:visited).
E:active E:hover E:focus	Matches E during certain user actions.
E:lang(c)	Matches element of type E if it is in (human) language c (the document language specifies how language is determined).
E + F	Matches any F element immediately preceded by a sibling element E.
E[foo]	Matches any E element with the "foo" attribute set (whatever the value).
E[foo="warning"]	Matches any E element whose "foo" attribute value is exactly equal to "warning".
E[foo~="warning"]	Matches any E element whose "foo" attribute value is a list of space-separated values, one of which is exactly equal to "warning".
E[lang = "en"]	Matches any E element whose "lang" attribute has a hyphen-separated list of values beginning (from the left) with "en".
DIV.warning	<i>Language specific.</i> (In HTML, the same as DIV[class~="warning"].)
E#myid	Matches any E element with ID equal to "myid".

Pseudo class selector

- Used to style an element based on something other than the structure of the document
 - E.g., the state of a form element or link

```
/* makes all unvisited links blue */  
a:link {color:blue;}  
/* makes all visited links green */  
a:visited {color:green;}  
/* makes links red when hovered or activated */  
a:hover, a:active {color:red;}  
/* makes table rows red when hovered over */  
tr:hover {background-color: red;}  
/* makes input elements yellow when focus is applied */  
input:focus {background-color:yellow;}
```

Meaningful XHTML

- Meaningful elements
 - h1, h2, ...
 - ul, ol, and dl
 - strong and em
 - blockquote and cite
 - abbr, acronym, and code
 - fieldset, legend, and label
 - caption, thead, tbody, and tfoot
- Id and class names
 - Allow to give extra meaning
- Div and span
 - Add structure to document

Div element

- Stands for “division”
- Used to group block-level elements
 - Provides a way of dividing a document into meaningful areas
- Use only if necessary and not redundant

```
<div id="mainNav">
  <ul>
    <li>Home</li>
    <li>About Us</li>
    <li>Contact</li>
  </ul>
</div>
```



```
<ul id="mainNav">
  <li>Home</li>
  <li>About Us</li>
  <li>Contact</li>
</ul>
```

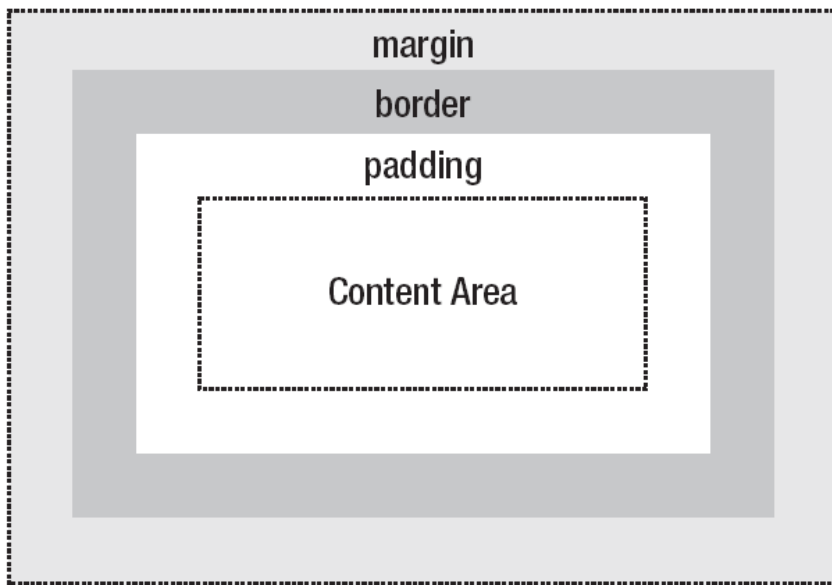
Span element

- Used to group or identify inline elements

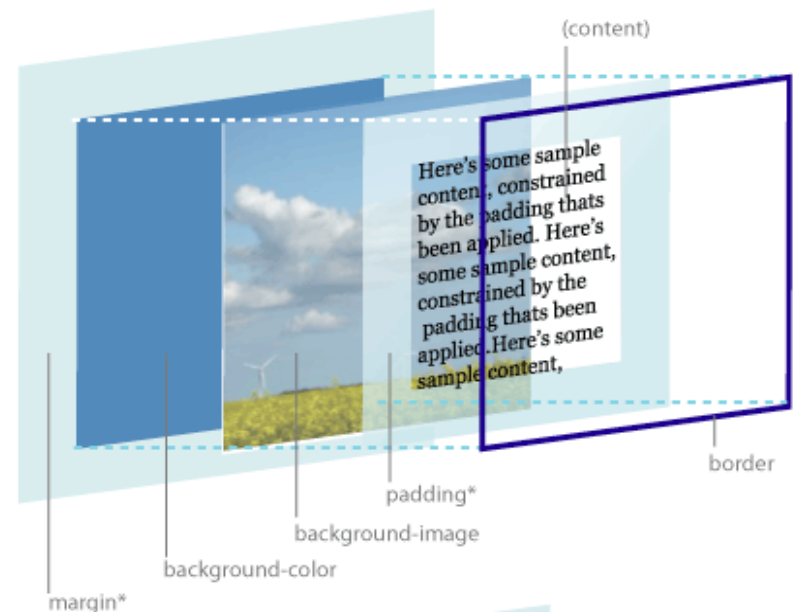
```
<h2>Where's Durstan?</h2>  
<p>Published on  
    <span class="date">March 22nd, 2005</span>  
by <span class="author">Andy Budd</span></p>
```

The box model

- One of the cornerstones of CSS
- Dictates how elements are displayed and, to a certain extent, how they interact with each other
- Every element on the page is considered to be a rectangular box



THE CSS BOX MODEL HIERARCHY



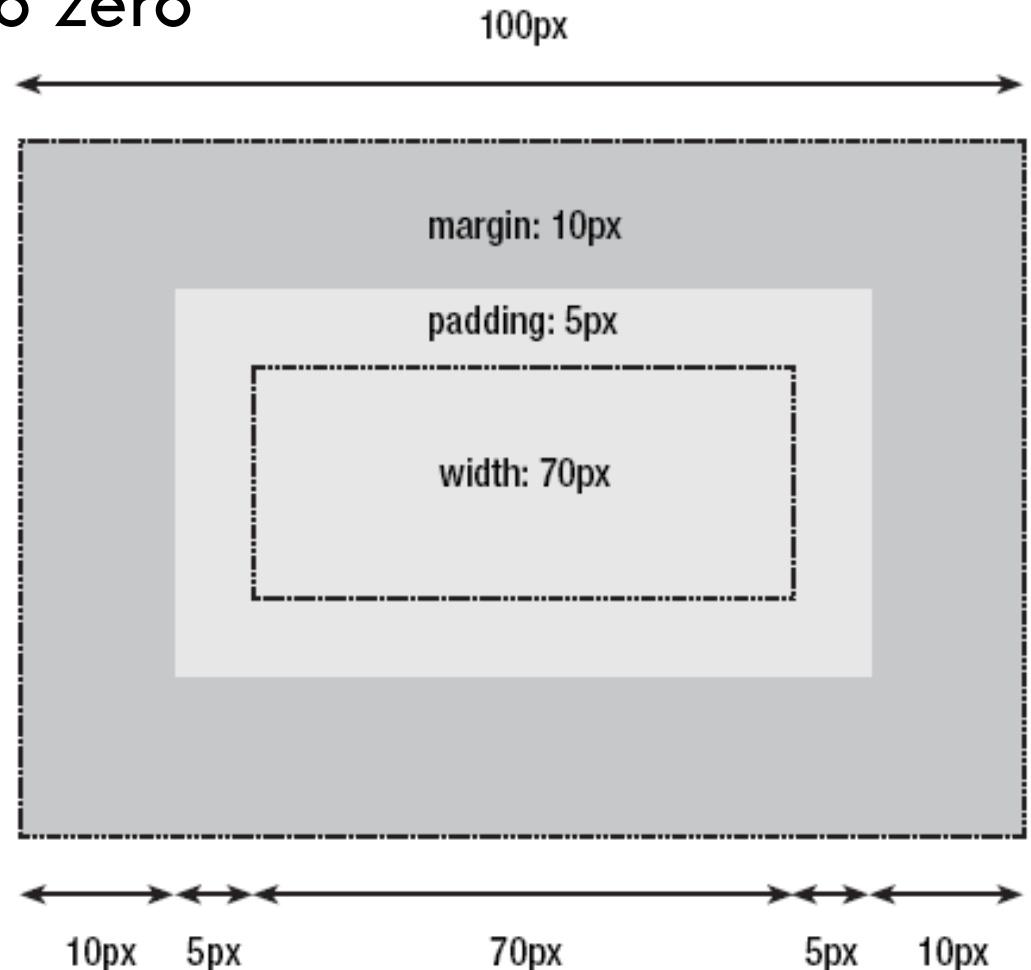
The box model

- Content
 - The content of the box, where text and images appear
- Padding
 - Clears an area around the content
 - The padding is affected by the background color of the box
- Border
 - A border that goes around the padding and content
 - The border is affected by the background color of the box
- Margin
 - Clears an area around the border
 - The margin does not have a background color, it is completely transparent

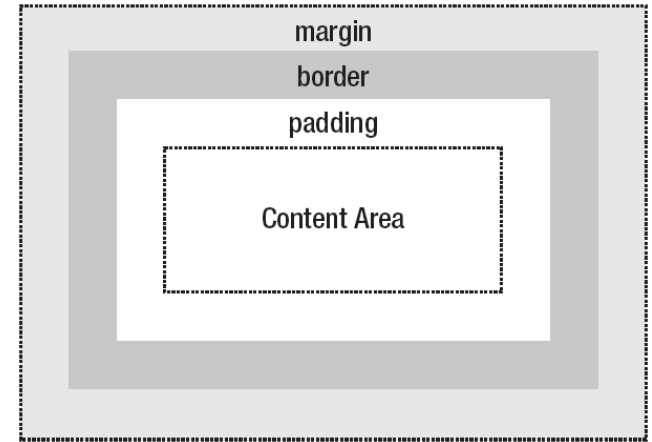
Example

- Padding, borders, and margins are optional and default to zero

```
#myBox {  
  margin: 10px;  
  padding: 5px;  
  width: 70px;  
}
```



The box model



- Total element width = width + left padding + right padding + left border + right border + left margin + right margin
- Total element height = height + top padding + bottom padding + top border + bottom border + top margin + bottom margin
- Example
 - W3Schools.com
 - http://www.w3schools.com/Css/css_boxmodel.asp

Positioning schemes

- Three basic positioning schemes in CSS
 - Normal flow
 - Floats
 - Absolute positioning
- Unless specified, all boxes start life being positioned in the normal flow
 - The position of an element's box in the normal flow will be dictated by that element's position in the (X)HTML

Normal flow

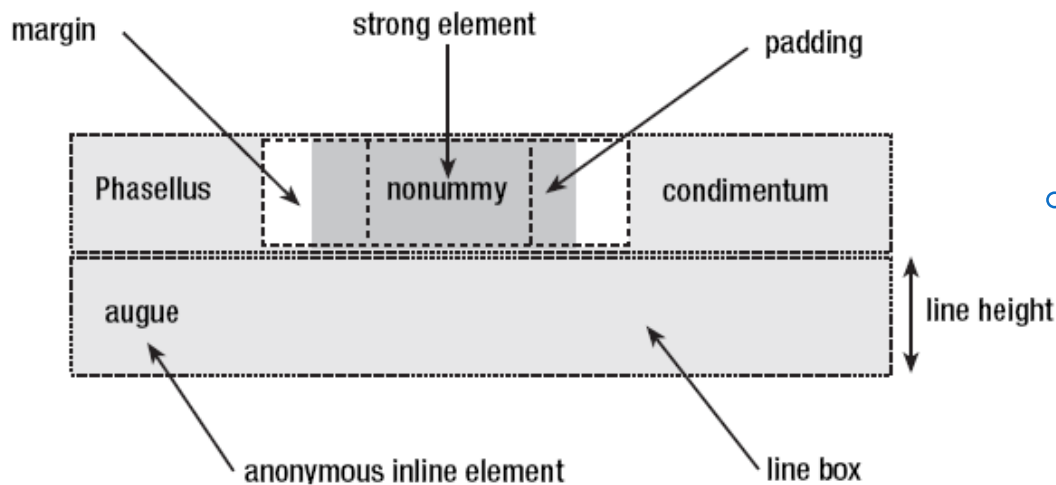
- Block-level boxes will appear vertically one after the other

```
<div> ... </div>
```

- The vertical distance between boxes is calculated by the boxes' vertical margins
- Inline boxes are laid out in a line horizontally

```
<span> ... </span>
```

- Their horizontal spacing can be adjusted using horizontal padding, borders, and margins
- Vertical padding, borders, and margins will have no effect on the height of an inline box



Display property

- Allows to control element visualization (block or inline)
- Changing an inline element to a block element, or vice versa, can be useful for making the page look a specific way
- Example
 - W3Schools.com
 - http://www.w3schools.com/Css/css_display_visibility.asp

```
li {display:inline;}
```

```
span {display:block;}
```

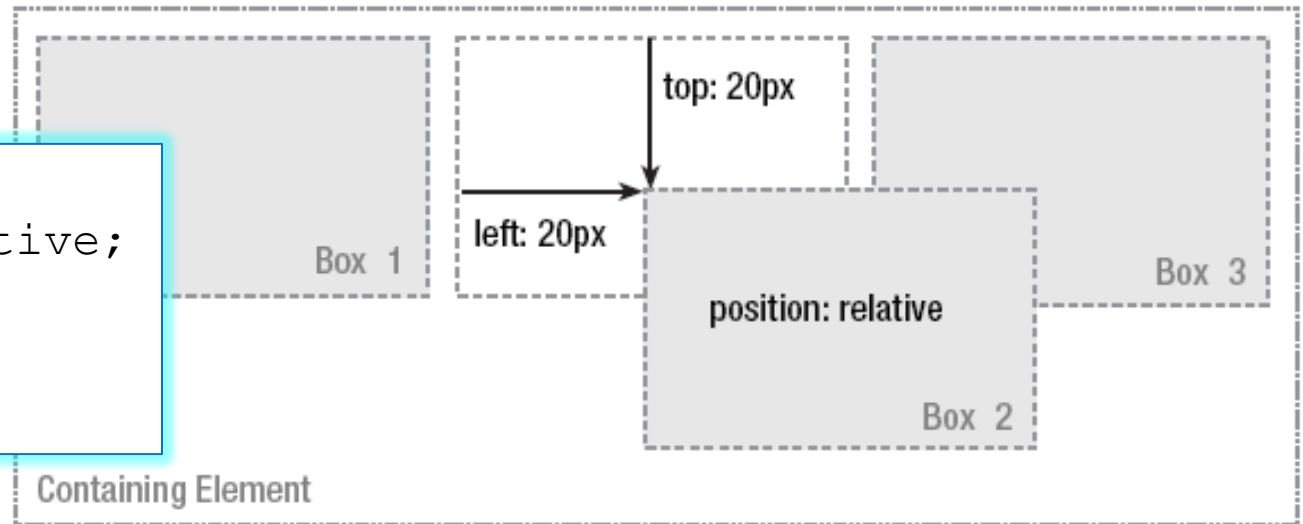
Box Positioning

- A block can be positioned in different ways to which correspond different positioning schemes
 - Static: normal block
 - Relative: the offset values are relative to the block position in the normal flow. If a relative block B follows a relative block A, the offset is respect to the position of A without the offset
 - Absolute: the box position is determined by the top, left, right, bottom properties and is relative to the containing block
 - Fixed: the box is fixed with respect to some reference (the viewport as an example)

Relative positioning

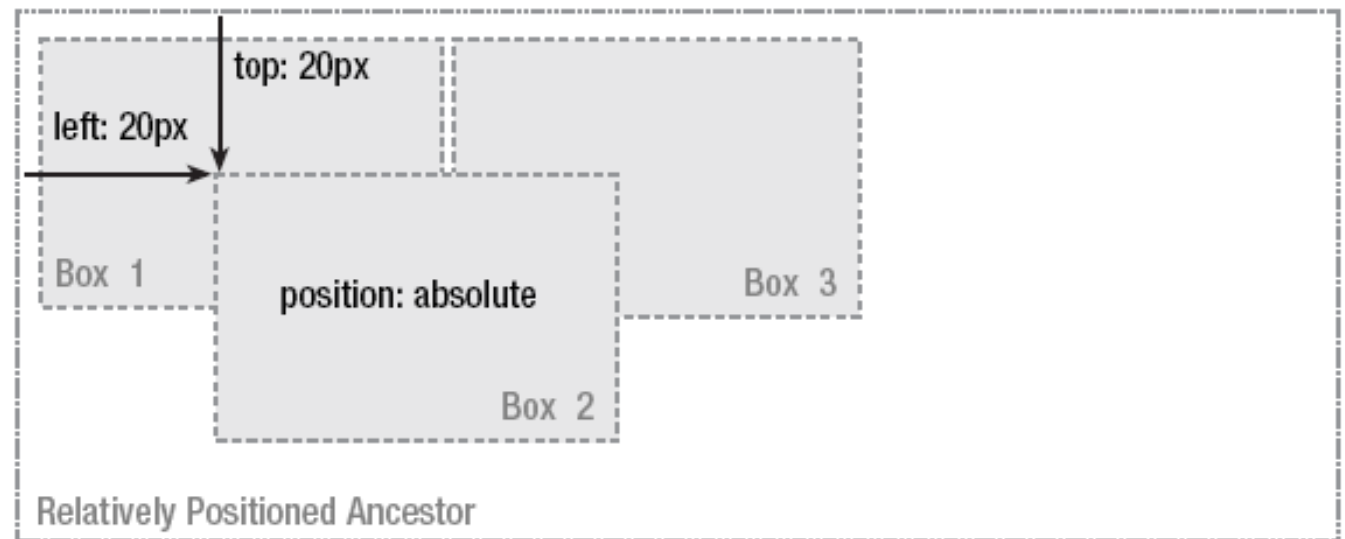
- It is possible to shift one element “relative” to its starting point by setting a vertical or horizontal position

```
#myBox {  
  position: relative;  
  left: 20px;  
  top: 20px;  
}
```



Absolute positioning

- Takes the element out of the flow of the document, thus taking up no space
- Other elements in the normal flow of the document will act as though the absolutely positioned element was never there



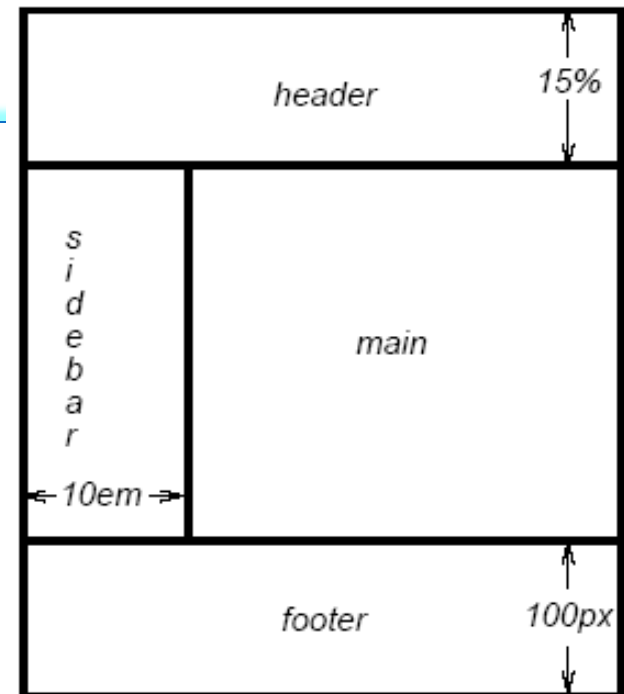
Fixed positioning

- A subcategory of absolute positioning
 - A fixed element's containing block is the viewport
- Allows to create elements that always stay at the same position in the window
- Note: in case of overlaps the z-index property specifies the stack order of an element (which element should be placed in front of, or behind, the others)

Fixed positioning

- Can be used to create complex frame-like presentations

```
#header { position: fixed; width: 100%;  
height: 15%; top: 0; right: 0;  
bottom: auto; left: 0; }  
#sidebar { position: fixed; width: 10em;  
height: auto; top: 15%; right: auto;  
bottom: 100px; left: 0; }  
#main { position: fixed; width: auto;  
height: auto; top: 15%; right: 0;  
bottom: 100px; left: 10em; }  
#footer { position: fixed; width: 100%;  
height: 100px; top: auto; right: 0;  
bottom: 0; left: 0; }
```



Examples

- W3Schools.com
 - http://www.w3schools.com/Css/css_positioning.asp

The main problem people have with positioning is remembering which type of positioning is which. Relative positioning is “relative” to the element’s initial position in the flow of the document, whereas absolute positioning is “relative” to nearest positioned ancestor or, if one doesn’t exist, the initial container block.

A. Budd, C. Moll, S. Collison,
“CSS Mastery: Advanced Web Standards
Solutions”, FriendsOfED, 2006

Floating

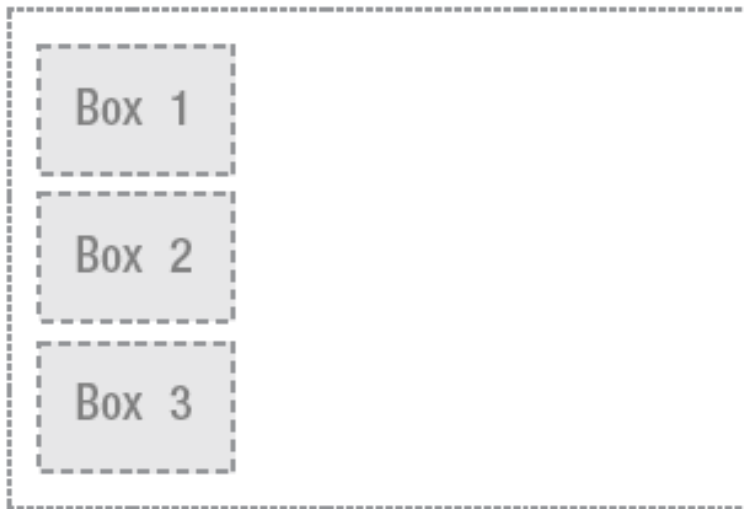
- A floated box can either be shifted to the left or the right until its outer edge touches the edge of its containing box, or another floated box
- Often used for images and when working with layouts
 - Example
 - http://www.w3schools.com/Css/css_float.asp

```
img
{
float:right;
}
```

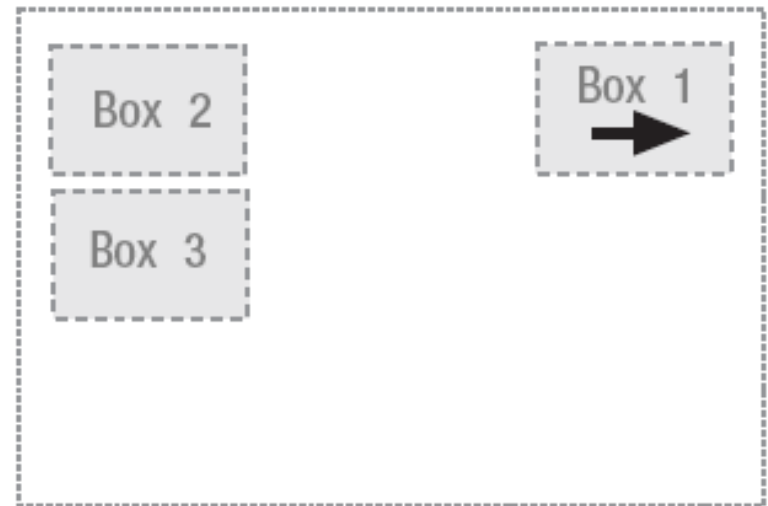
Floating

- Floated boxes aren't in the normal flow of the document, so block boxes in the regular flow of the document behave as if the floated box wasn't there

No boxes floated



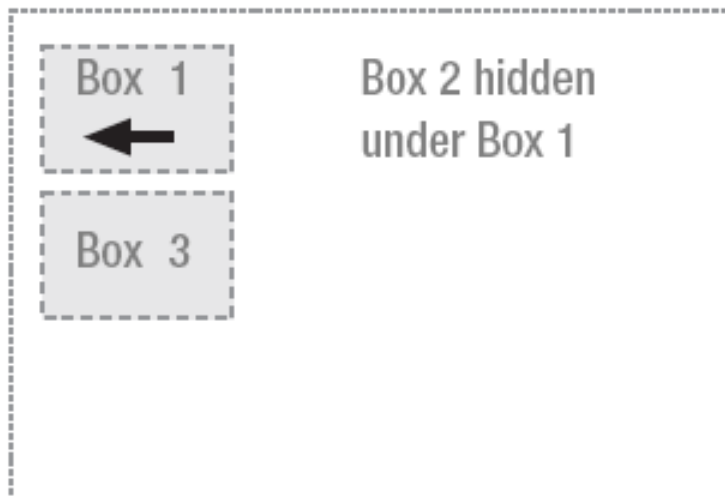
Box 1 floated right



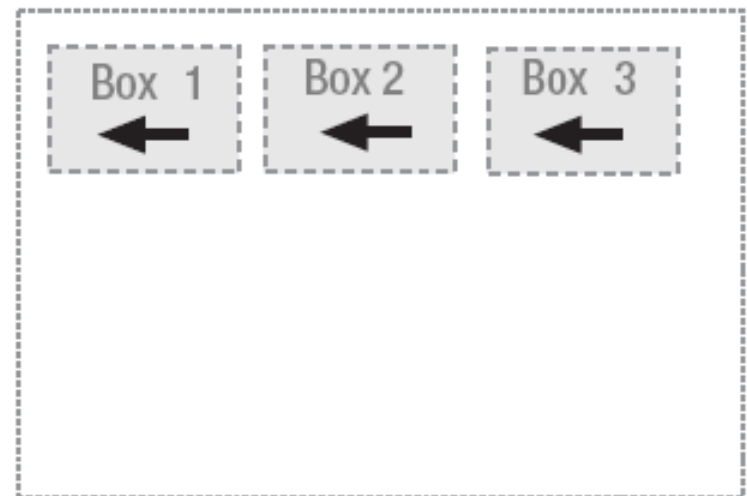
Floating

- If all three boxes are floated left
 - Box 1 is shifted left until it touches its containing box
 - Other two boxes are shifted left until they touch the preceding floated box

Box 1 floated left



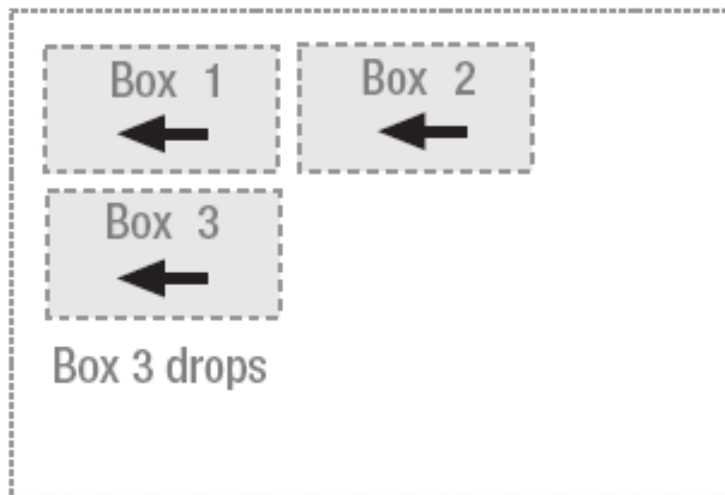
All three boxes floated left



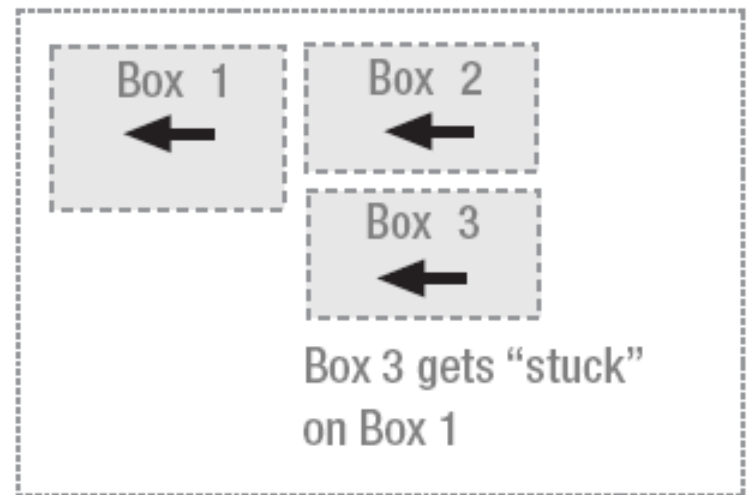
Floating

- If the containing block is too narrow for all of the floated elements to fit horizontally
 - The remaining floats will drop down until there is sufficient space
 - If the floated elements have different heights, it is possible for floats to get “stuck” on other

Not enough horizontal space



Different height boxes



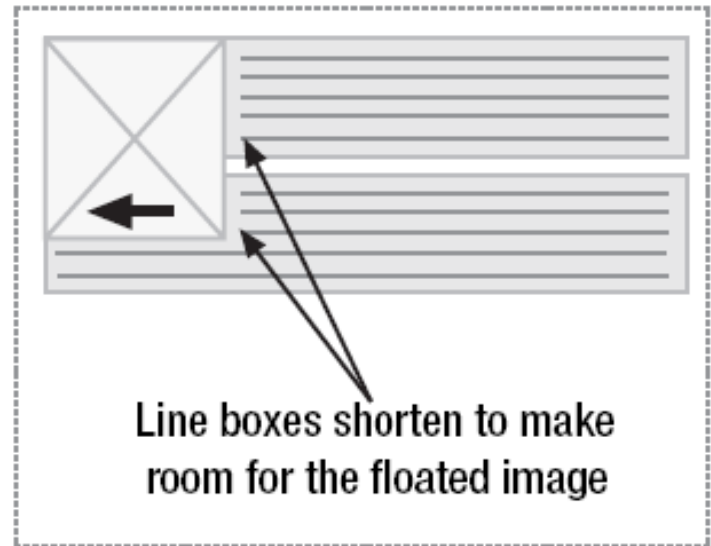
Line boxes and clearing

- Line boxes next to a floated box are shortened to make room for the floated box, and flow around the float
 - Floats were created to allow text to flow around images

No boxes floated



Image floated left

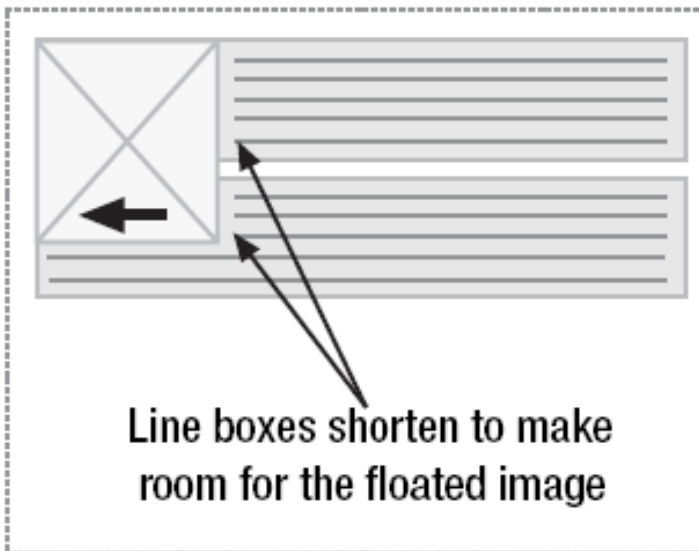


Line boxes and clearing

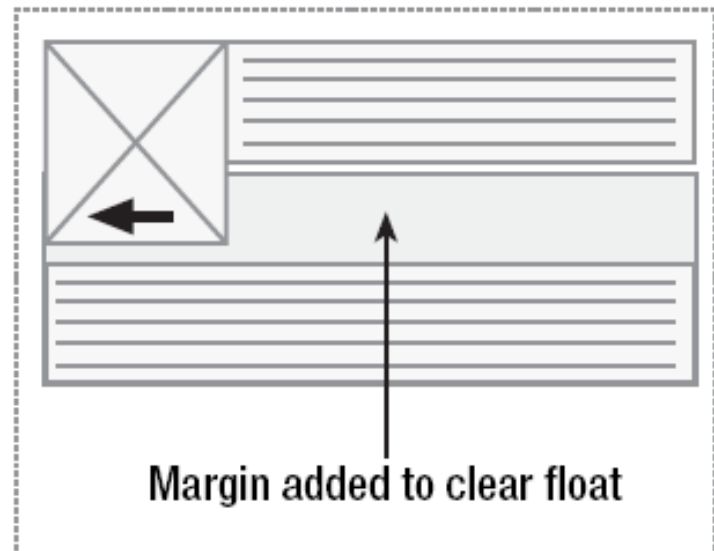
- To stop line boxes flowing around the outside of a floated box, you need to apply a clear to that box
 - The clear property can be left, right, both, or none, and indicates which side of the box should not be next to a floated box

```
p { clear: left }
```

Image floated left



Second paragraph cleared



Page layout

- Possibility to control page layout without needing to use presentation markup
- CSS layout has a rather undeserved reputation of being difficult
 - Mostly due to a proliferation of different layout techniques available on the Web
- Tasks
 - Horizontally centering a design on a page
 - Creating two- and three-column float-based layouts
 - Creating fixed-width, liquid, and elastic layouts
 - Making columns stretch to the full height of the available space

Centering a design

- Long lines of text can be difficult and unpleasant to read
- Rather than spanning the full width of the screen, centered designs span only a portion of the screen, creating shorter and easier-to-read line lengths
- Two basic methods
 - Use auto margins
 - Use positioning and negative margins

Auto margins

- Define the width of the wrapper div
- Set the horizontal margin to auto

```
<body>
<div id="wrapper">
</div>
</body>
```

```
#wrapper {
width: 720px;
margin: 0 auto;
}
```

- The most common approach
- IE 5.x and IE 6 do not support auto margins

```
body {
text-align: center;
}
#wrapper {
width: 720px;
margin: 0 auto;
text-align: left;
}
```

Positioning and negative margins

- Define the width of the wrapper div
- Set the position property of the wrapper to relative
- Set the left property to 50%

```
#wrapper {  
width: 720px;  
position: relative;  
left: 50%;  
}
```

- Apply a negative margin to the left side of the wrapper, equal to half the width of the wrapper

```
#wrapper {  
width: 720px;  
position: relative;  
left: 50%;  
margin-left: -360px;  
}
```

Float-based layouts

- Simply set the width of the elements you want to position, and then float them left or right
- Two-column floated layout
- Three-column floated layout

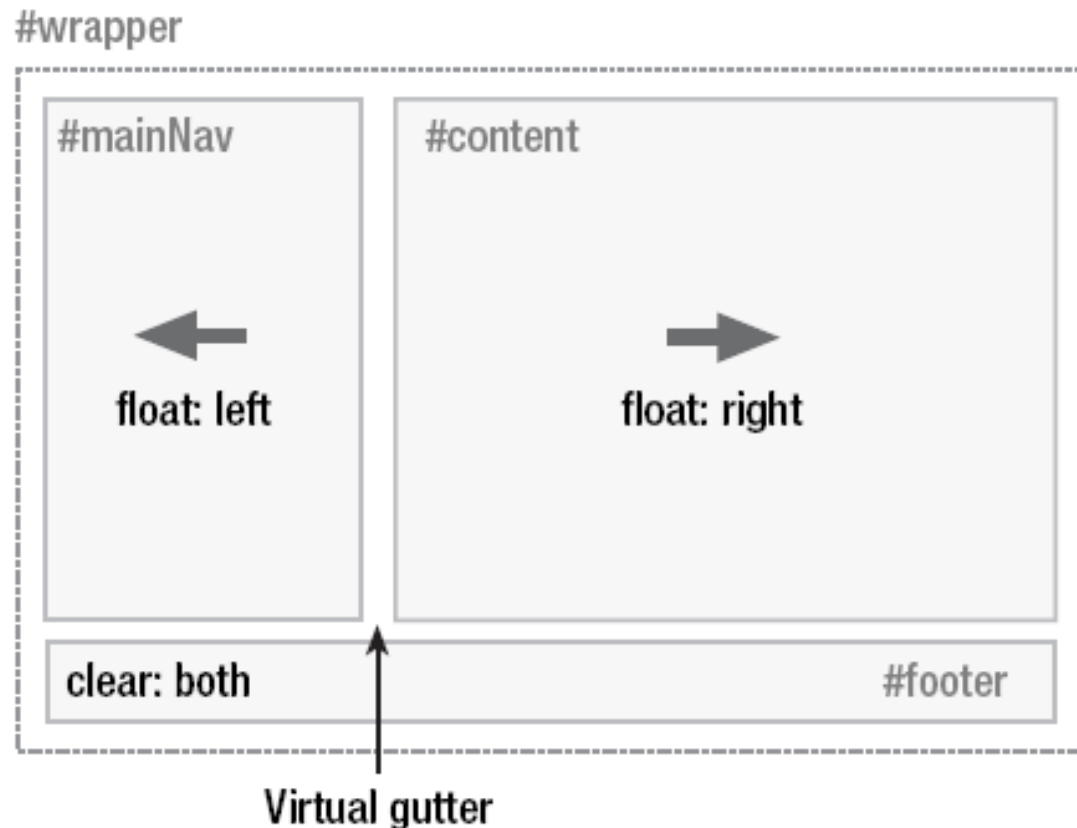
Two-column floated layout

- (X)HTML framework
 - Main navigation on the left side of the page
 - Content on the right
- For accessibility reasons the content area is above the navigation in the source
 - The main content is the most important thing on the page and so should come first in the document
 - There is no point forcing screenreader users to trawl through a potentially long list of links before they get to the content

```
<div id="wrapper">
<div id="branding">
...
</div>
<div id="content">
...
</div>
<div id="mainNav">
...
</div>
<div id="footer">
...
</div>
</div>
```


Two-column floated layout

- Create a virtual gutter by floating one element left and one element right



Two-column floated layout

```
#content {  
width: 520px;  
float: right;  
}  
#mainNav {  
width: 180px;  
float: left;  
}  
#footer {  
clear: both;  
}
```

- **Better: add horizontal padding**

```
#mainNav {  
padding-top: 20px;  
padding-bottom: 20px;  
}  
#mainNav li {  
padding-left: 20px;  
padding-right: 20px;  
}  
#content h1, #content h2,  
    #content p {  
padding-right: 20px;  
}
```

Two-column floated layout

Header

[download this layout](#)

2) Navigation here. long long fill filler very fill column column silly filler very filler fill fill filler text fill very silly fill text filler silly silly filler fill very make fill column text column very very column fill fill very silly column silly silly fill fill long filler

[Add Text to this section](#)

1) Content here. column long long column very long fill fill fill long text text column text silly very make long very fill silly make make long make text fill very long text column silly silly very column long very column filler fill long make filler long silly very long silly silly silly long filler make column filler make silly long long fill very.

very make make fill silly long long filler column long make silly silly column filler fill fill very filler text fill filler column make fill make text very make make very fill fill long make very filler column very long very filler silly very make filler silly make make column column

fill long make long text very make long fill column make text very silly column filler silly text fill text filler filler filler make make make make text filler fill column filler make silly make text text fill make very filler column very

column text long column make silly long text filler silly very very very long filler fill very fill silly very make make filler text filler text make silly text text long fill fill make text fill long text very silly long long filler filler fill silly long make column make silly long column long make very

[Add Text to this section](#)

3) More stuff here. very text make long column make filler fill make column column silly filler text silly column fill silly fill column text filler make text silly filler make filler very silly make text very very text make long filler very make column make silly column fill silly column long make silly filler column filler silly long long column fill silly column very

[Add Text to this section](#)

The footer. You can go to the [index page](#).

<http://blog.html.it/layoutgala/>

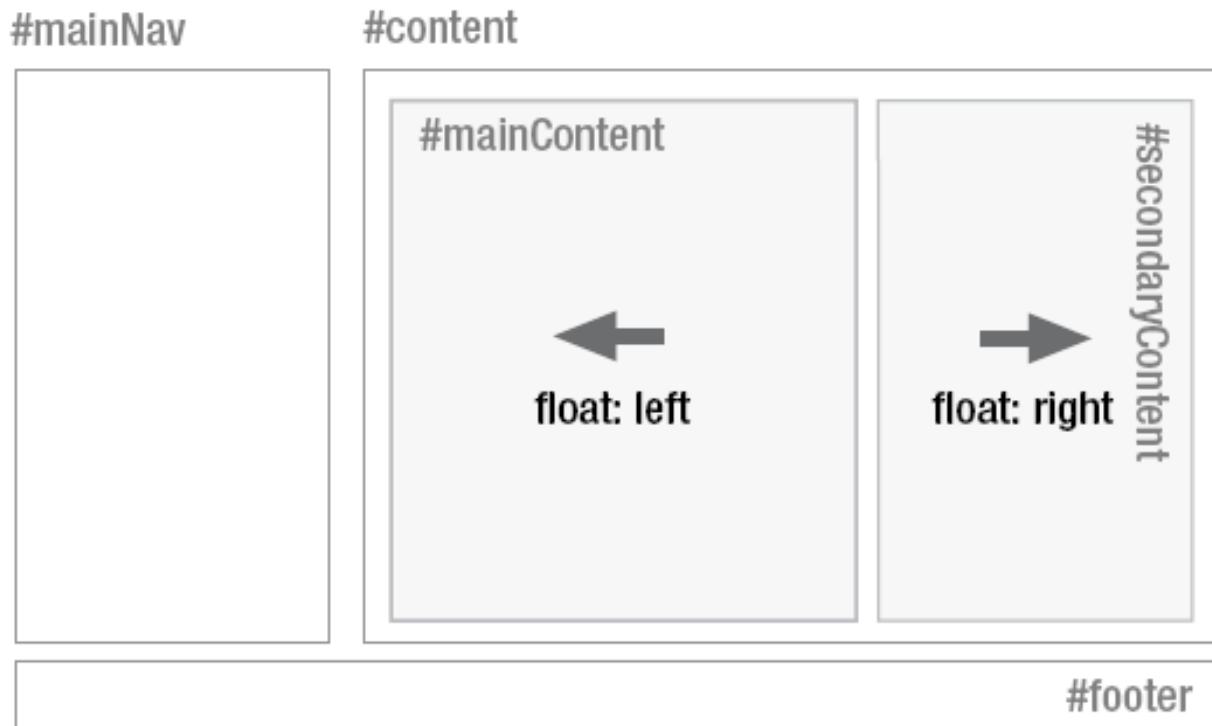
Three-column floated layout

- (X)HTML framework
 - similar to the two column layout, but two new divs inside the content div

```
<div id="content">  
  <div id="mainContent">  
    ...  
  </div>  
  <div id="secondaryContent">  
    ...  
  </div>  
</div>
```

Three-column floated layout

- Float the main content left and the secondary content right, inside the already floated content div
 - Divides the second content column in two, creating a three-column effect



Three-column floated layout

```
#mainContent {  
width: 320px;  
float: left;  
}  
#secondaryContent {  
width: 180px;  
float: right;  
}
```

- **Better:** remove the padding from the content element and apply it to the content of the secondary content

```
#secondaryContent h1, #secondaryContent h2,  
    #secondaryContent p {  
padding-left: 20px;  
padding-right: 20px;  
}
```

Three-column floated layout

Header

[download this layout](#)

2) Navigation here.

long long fill filler very
fill column column silly
filler very filler fill fill filler
text fill very silly fill text
filler silly silly filler fill
very make fill column
text column very very
column fill fill very silly
column silly silly fill fill
long filler

[Add Text to this section](#)

1) Content here. column long long column very long fill fill fill long text
text column text silly very make long very fill silly make make long
make text fill very long text column silly silly very column long very
column filler fill long make filler long silly very long silly silly long
filler make column filler make silly long long fill very.

very make make fill silly long long filler column long make silly silly
column filler fill fill very filler text fill filler column make fill make text
very make make very fill fill long make very filler column very long very
filler silly very make filler silly make make column column

fill long make long text very make long fill column make text very silly
column filler silly text fill text filler filler filler make make make make
text filler fill column filler make silly make text text fill make very filler
column very

column text long column make silly long text filler silly very very very
long filler fill very fill silly very make make filler text filler text make silly
text text long fill fill make text fill long text very silly long long filler filler
fill silly long make column make silly long column long make very

[Add Text to this section](#)

3) More stuff here.

column long make silly
silly filler silly very very
very long column filler
fill make column make
silly column fill silly
column long make silly
filler column filler silly
long long column fill
silly column very

[Add Text to this section](#)

The footer. You can go to the [index page](#).

<http://blog.html.it/layoutgala/>

Fixed-width, liquid, and elastic layout

- Fixed-width layout
 - Column widths defined in pixels
 - Very common as they give the developer more control over layout and positioning
- Downsides
 - Do not make good use of the available space: columns are always the same size no matter the window size
 - Usually work well with the browser default text size, but if you increase the text size a couple of steps, sidebars start running out of space and the line lengths get too short to comfortably read

Liquid layouts

- Dimensions are set using percentages instead of pixels
 - Very efficient use of space
- If the design spans the entire width of the browser window, line lengths can become long and difficult to read
 - Solution: make the wrapper span just a percentage,
e.g. 85 percent

```
#wrapper {  
width: 85%;  
}
```

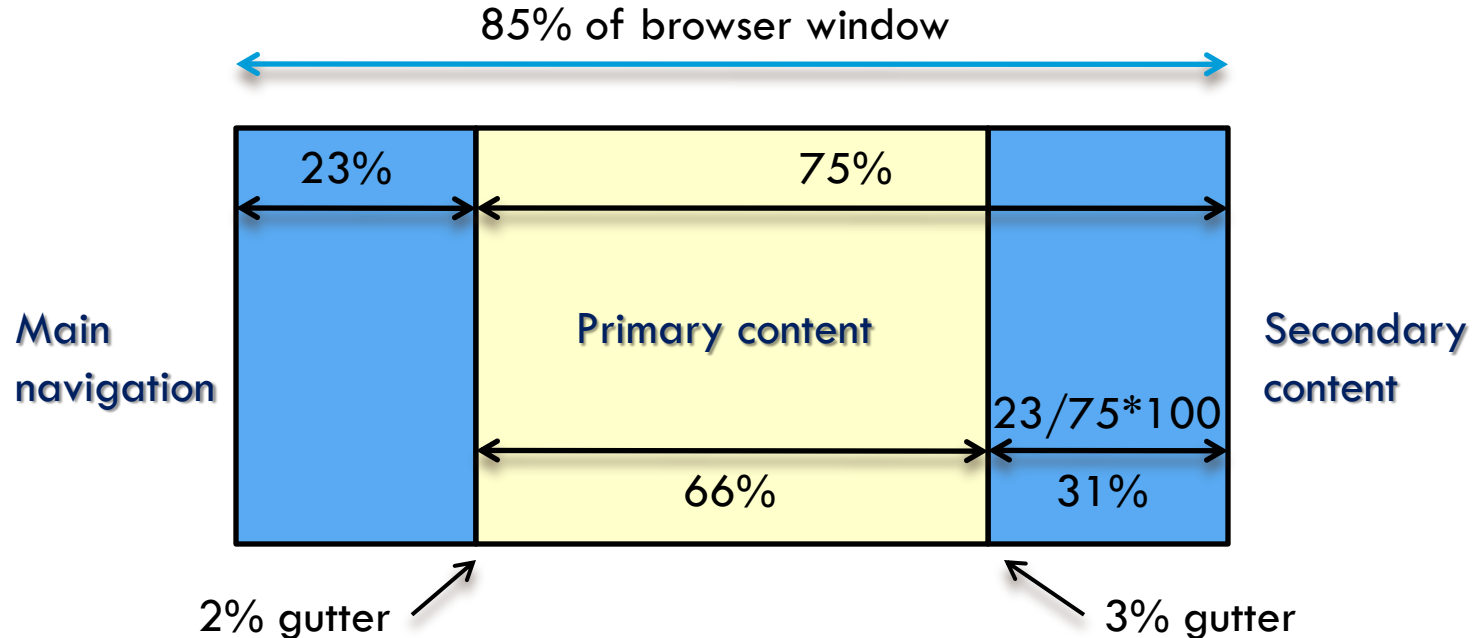
Liquid layouts

- Set the width of the navigation and content areas as a percentage of the wrapper width
 - 2-percent virtual gutter between the navigation and the wrapper to deal with any rounding errors and width irregularities that may occur

```
#wrapper {  
width: 85%;  
}  
#mainNav {  
width: 23%;  
float: left;  
}  
#content {  
width: 75%;  
float: right;  
}
```

Liquid layouts

- The widths of the content divs are based on the width of the content element and not the overall wrapper
 - Width of secondary content area = width of the main navigation area?



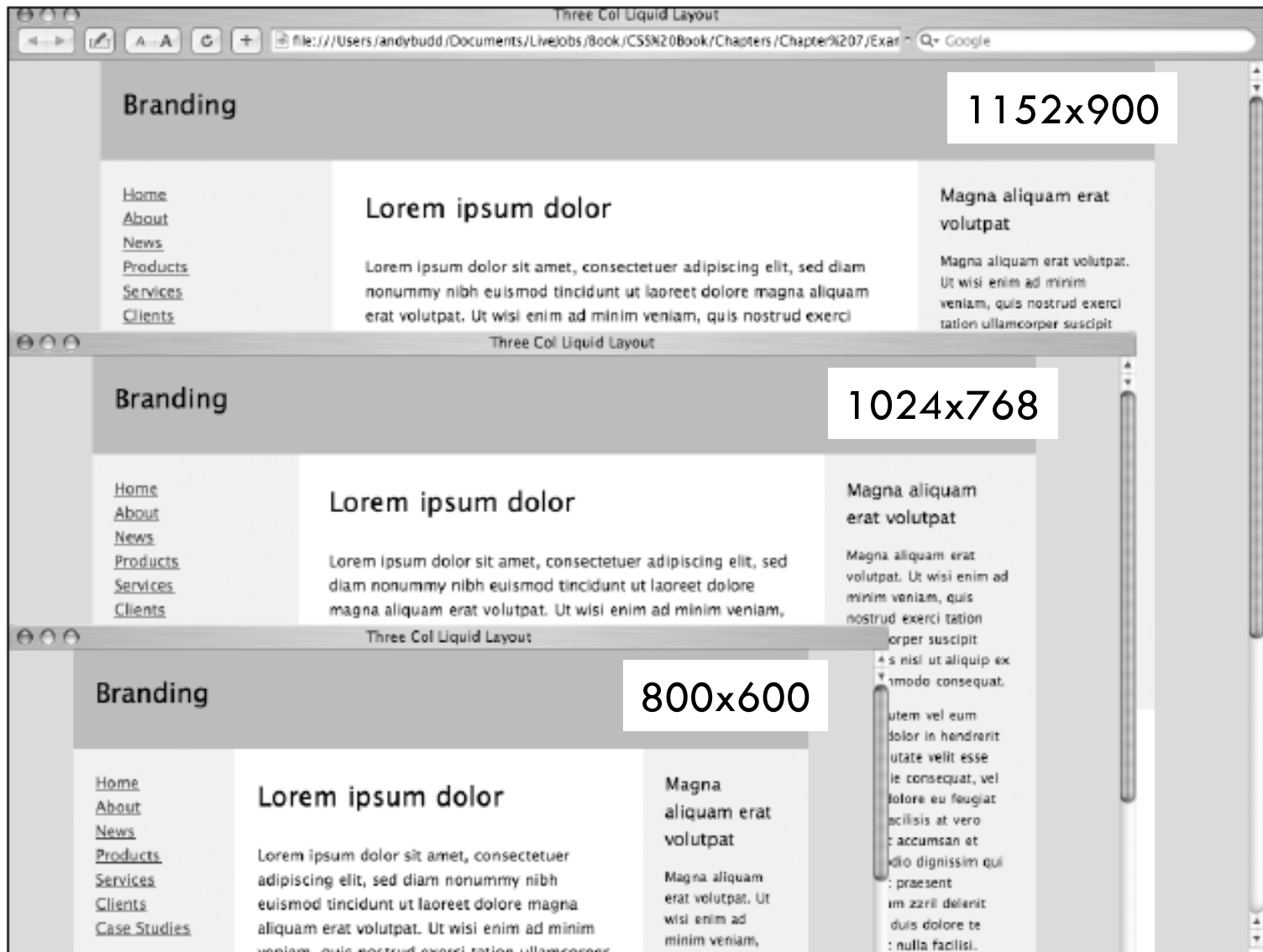
Liquid layouts

- 3 columns liquid layout

```
#wrapper {  
width: 85%;  
}  
#mainNav {  
width: 23%;  
float: left;  
}  
#content {  
width: 75%;  
float: right;  
}
```

```
#mainContent {  
width: 66%;  
float: left;  
}  
#secondaryContent {  
width: 31%;  
float: right;  
}
```

Three columns liquid layout



Elastic layouts

- With liquid layouts
 - Line lengths can get uncomfortably long on large resolution monitors
 - Lines can become very short and fragmented in narrow windows or when the text size is increased a couple of steps
- In elastic layouts the width of elements is relative to the font size (ems) instead of the browser width
 - When the font size is increased the whole layout scales
- Allows to keep line lengths to a readable size
 - Particularly useful for people with reduced vision

Elastic layouts

- Trick to simplify design: set the base font size so that 1em roughly equals 10 pixels
 - The default font size on most browsers is 16 pixels
 - Ten pixels are 62.5 percent of 16 pixels
- Set the font size on the body to 62.5%

```
body {  
  font-size: 62.5%;  
}
```

Elastic layouts

- 1em now equals 10 pixels at the default font size
- Convert the fixed-width layout into an elastic layout by converting all the pixel widths to em widths

```
#wrapper {  
width: 72em;  
margin: 0 auto;  
text-align:  
left;  
}  
#mainNav {  
width: 18em;  
float: left;  
}
```

```
#content {  
width: 52em;  
float: right;  
}  
#mainContent {  
width: 32em;  
float: left;  
}  
#secondaryContent {  
width: 18em;  
float: right;  
}
```


Three columns elastic layout



Three columns elastic layout



Increased text size

Elastic-liquid hybrid

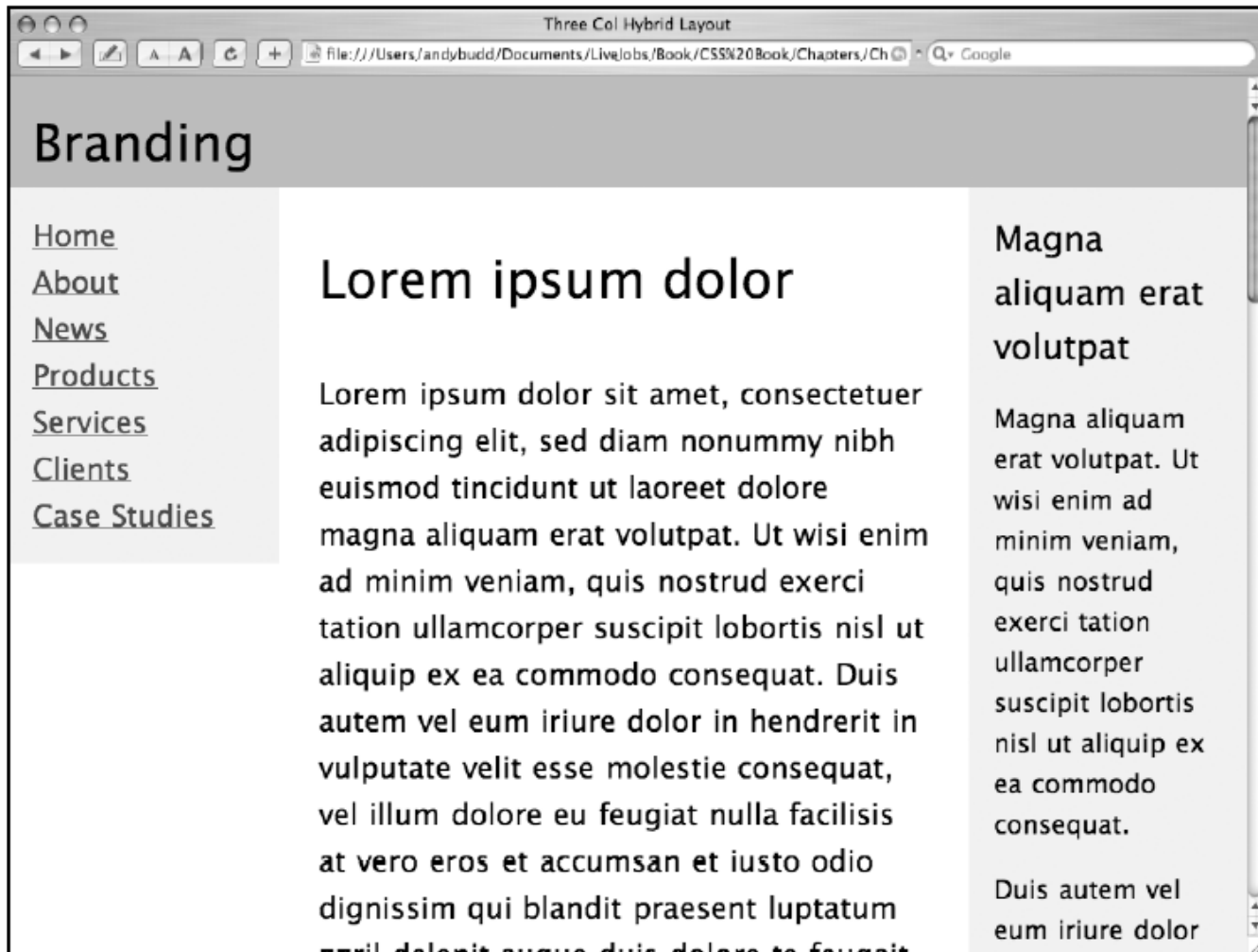
- Combines both elastic and liquid techniques
- Works by setting the widths in ems, then setting the maximum widths as percentages

```
#wrapper {  
width: 72em;  
max-width: 100%;  
margin: 0 auto;  
text-align: left;  
}  
#mainNav {  
width: 18em;  
max-width: 23%;  
float: left;  
}
```

```
#content {  
width: 52em;  
max-width: 75%;  
float: right;  
}  
#mainContent {  
width: 32em;  
max-width: 66%;  
float: left;  
}  
#secondaryContent {  
width: 18em;  
max-width: 31%;  
float: right;  
}
```

Elastic-liquid hybrid

- On browsers that support max-width, this layout will scale relative to the font size but will never get any larger than the width of the window



A few more interesting tasks

- Vertical navigation bars
- Horizontal navigation bars
- Image maps
- Form styles

Lists and navigation bars

- Simple list

```
<ul>  
<li>Read emails</li>  
<li>Write book</li>  
<li>Go shopping</li>  
<li>Cook dinner</li>  
<li>Watch Scrubs</li>  
</ul>
```






- To add custom bullets

- Use the `list-style-image` property, but little control over the position of the bullet image
- Turn list bullets off and add custom bullet as a background image on the list element

Custom bullets

- Remove indentation by zeroing down the margin and padding on the list
- Remove the default bullet setting the list style type to none
- Adding padding to the left side of the list item creates the necessary space for the bullet

```
ul {  
margin: 0;  
padding: 0;  
list-style-type: none;  
}
```

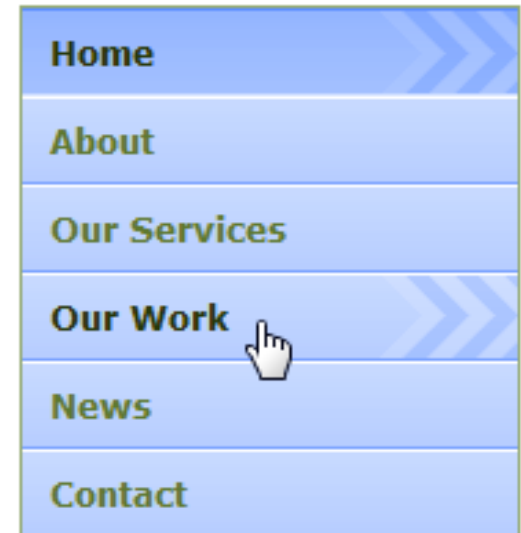
-  Comprare pane e latte
-  Portare a spasso il cane
-  Pulire il garage
-  Prenotare ristorante
-  Fare benzina

```
li {  
background: url(bullet.gif) no-repeat 0 50%;  
padding-left: 30px;  
}
```

simple-list.htm

Vertical navigation bar

vertical-nav.htm

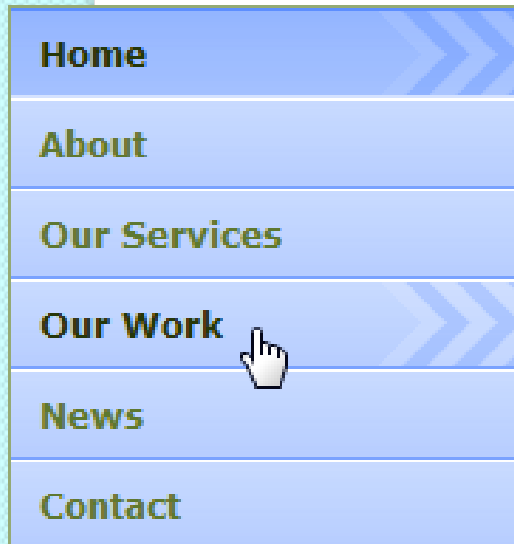


- HTML framework

```
<ul class="arrowblue">
  <li><a href="home.htm" class="selected">Home</a></li>
  <li><a href="about.htm">About</a></li>
  <li><a href="services.htm">Our Services</a></li>
  <li><a href="work.htm">Our Work</a></li>
  <li><a href="news.htm">News</a></li>
  <li><a href="contact.htm">Contact</a></li>
</ul>
```


Vertical navigation bar

- Remove the default bullets and zero down the margin and padding



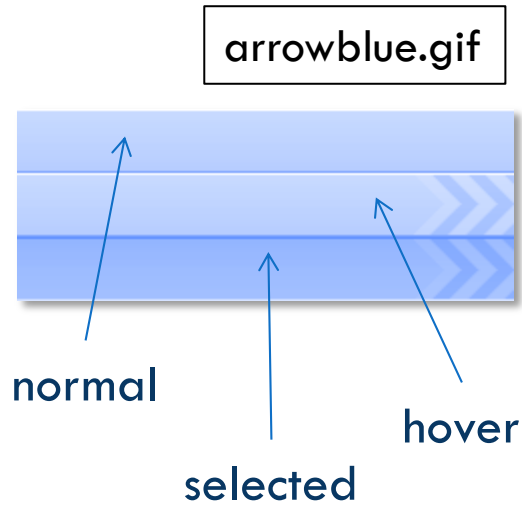
```
.arrowblue {  
  list-style-type: none;  
  margin: 0;  
  padding: 0;  
  width: 180px; /*width of menu*/  
  border-style: solid solid none solid;  
  border-color: #94AA74;  
  border-size: 1px;  
  border-width: 1px;  
}
```

Vertical navigation bar

- Rather than style the list items, style the enclosed anchors
 - To create a button-like area, set the display property of the anchors to block and specify the anchor's dimensions
 - The line height is set to 24 pixels to center the link text vertically
 - The last rule removes the underline from the link text

```
.arrowblue li a{
  font: bold 12px Verdana, Arial, Helvetica, sans-serif;
  display: block;
  background: transparent url(img/arrowblue.gif) 100% 0;
  height: 24px; /* set height: bg image-padding (32px-4px-4px) */
  padding: 4px 0 4px 10px;
  line-height: 24px; /*set line-height (32px - 4px - 4px) */
  text-decoration: none;
}
```

Vertical navigation bar



- Pixy rollover technique
 - The rollover graphic is applied as a background image to the anchor link
 - Single image composed of the three possible states of the link (normal, hover, selected)

```
.arrowblue li a:hover{
  color: #26370A;
  background-position: 100% -32px;
}

.arrowblue li a.selected{
  color: #26370A;
  background-position: 100% -64px;
}
```

Horizontal navigation bar



horizontal-nav.htm

- HTML framework

```
<ul class="glossymenu">  
  <li class="current"><a href="#"><b>Home</b></a></li>  
  <li><a href="#"><b>About</b></a></li>  
  <li><a href="#"><b>News</b></a></li>  
  <li><a href="#"><b>Products</b></a></li>  
  <li><a href="#"><b>Services</b></a></li>  
  <li><a href="#"><b>Clients</b></a></li>  
  <li><a href="#"><b>Case Studies</b></a></li>  
</ul>
```

Horizontal navigation bar

- Default bullets removed
- Central horizontal navigation bar, with a repeating image as a background
 - Margin and padding dimensions set accordingly

menur_bg.gif

```
.glossymenu{
    position: relative;
    padding: 0 0 0 34px;
    margin: 0 auto 0 auto;
    background: url(img/menur_bg.gif) repeat-x;
    /*tab background image path*/
    width: 60%;
    height: 46px;
    list-style: none;
}
```

Horizontal navigation bar

- Lists are normally displayed vertically
- Two methods to make it display horizontally
 - Set the list items to display inline
 - Float all list items left

```
ul li {  
    float: left;  
}
```

```
ul li {  
    display: inline;  
}
```

- Here: floating method

```
.glossymenu li {  
    float: left;  
}
```

Horizontal navigation bar

- Effect for current and hover elements

```
.glossymenu li.current a, .glossymenu li a:hover {  
  color: #fff;  
  background: url(img/menur_hover_left.gif) no-repeat;  
    /*left tab image path*/  
  background-position: left;  
}
```

```
.glossymenu li.current a b, .glossymenu li a:hover b {  
  color: #fff;  
  background: url(img/menur_hover_right.gif) no-repeat  
    right top; /*right tab image path*/  
}
```



menur_hover_left.gif

menur_hover_right.gif

Other examples

http://www.w3schools.com/Css/css_navbar.asp

CSS image maps

- Allow to specify regions of an image to act as hotspots
- Image maps are still valid in HTML
 - But they mix presentation with content
- It is possible to create simple image maps with a combination of lists, anchors, and some CSS code

The goal

imagemap.htm



Ringo Starr - Drums

Born Richard Starkey on July 7, 1940, in Liverpool, England, Ringo Starr, known for his easy-going personality, rose to fame in the early 1960s as a member of the legendary rock group the Beatles. Known for his role as drummer, Starr also sang and wrote songs for the group, singing "With a Little Help from My Friends" and writing "Octopus's Garden." [...]

CSS image maps

- Add the image to the page inside a named div

```
<div class="imagemap">  
    
</div>
```

- Add a list of links to each artist
 - Each list item needs to be given an id to identify the person in that list item
 - Give each link a title attribute containing the name of the artist: tooltip showing who the person is will be displayed on most browsers when the link is hovered

CSS image maps

```
<div class="imagemap">
  
  <ul>
    <li><a id="paul" href="" title="Paul McCartney">
      <span>Paul McCartney - Bass Guitar and Vocals
      <br /><br />Paul McCartney was born June 18, 1942,
      in Liverpool, England. [...] </span></a></li>
    <li><a id="ringo" href="" title="Ringo Starr">
      <span>Ringo Starr - Drums<br /><br />Born
      Richard [...]</span></a></li>
    <li><a id="john" href="" title="John Lennon">
      <span>John Lennon - Guitar and Vocals<br /><br />
      John Lennon was born on [...]</span></a></li>
    <li><a id="george" href="" title="George Harrison">
      <span>George Harrison - Lead Guitar and Vocals
      <br /><br />Pop star, songwriter, recording
      artist and producer [...] </span></a></li>
  </ul>
</div>
```

CSS image maps

- Set the width and height of the div so that it matches the dimensions of the image
- Set the position property of the div to relative
 - Important: it allows the enclosed links to be positioned absolutely, in relation to the edges of the div, and hence the image

```
.imagemap {  
  width:400px;  
  height:240px;  
  position: relative;  
}
```

- Remove the list bullets and zero down the list's margin and padding

```
.imagemap ul {  
  margin: 0;  
  padding: 0;  
  list-style: none;  
}
```

CSS image maps

- Style the links
 - Set widths and heights to create the desired hit area
 - Links can then be positioned (in absolute coordinates) over the correct artist, forming the hotspots
- The link text should not be displayed unless the hotspot is hit

```
.imagemap a {  
    position: absolute;  
    display: block;  
    width: 60px;  
    height: 60px;  
    text-decoration: none;  
}  
  
.imagemap a span, .imagemap a:visited span {  
    display: none;  
}
```

CSS image maps

- The individual links can be positioned over the corresponding artist

```
.imagemap a#paul {
  top: 65px;
  left: 52px;
}
.imagemap a#ringo {
  top: 95px;
  left: 120px;
}
.imagemap a#john {
  top: 67px;
  left: 200px;
}
.imagemap a#george {
  top: 77px;
  left: 272px;
}
```

CSS image maps

- A solid white border is applied to the links when they are hovered
- The text block should appear in the same position below the main image: need to recalculate the top left position of each span relative to the top left position of each hotspot

```
.imagemap a:hover {  
    border: 1px solid #fff;  
}  
.imagemap a:hover span {  
    position:absolute;  
    width:388px;  
    display:block;  
    font-family:arial;  
    font-size:12px;  
    background:#fff;  
    color:#000;  
    border:1px solid #000;  
    padding:5px;  
}  
.imagemap a#paul:hover span {  
    top: 180px;  
    left: -53px;  
}
```

Simple form layout

- Short and relatively simple forms are easiest to fill in when the form labels appear vertically above their associated form elements

simple-form.htm

Your Contact Details

Name: (Required)

Email Address:

Web Address:

Comments

Message: (Required)

Useful HTML form elements

- Fieldset element: to group related blocks of information

```
fieldset {  
  border: solid 1px #ccc;  
}
```

Your Contact Details

Name: (Required)

Email Address:

Web Address:

Useful HTML form elements

- Label element: to add a meaningful and descriptive label to each form element
 - Real benefit: to increase form usability for people using assistive devices
- Two ways to associate a label with a form
 - Implicitly, by nesting the form element inside the label element

```
<label>email <input name="email" type="text"/></label>
```

- Explicitly, by setting the for attribute of the label equal to the id name of the associated form element

```
<label for="email">email</label>  
<input name="email" id="email" type="text"/>
```

The goal

Your Contact Details

Name: *(Required)*

Email Address:

Web Address:



Your Contact Details

Name: *(Required)*

Email Address:

Web Address:

Simple form layout

- HTML code

```
<fieldset>
<legend>Your Contact Details</legend>
<p>
<label for="author">Name:</label>
<input name="author" id="author" type="text" />
</p>
<p>
<label for="email">Email Address:</label>
<input name="email" id="email" type="text" />
</p>
<p>
<label for="url">Web Address:</label>
<input name="url" id="url" type="text" />
</p>
</fieldset>
```

Simple form layout

- General styles for the fieldset and legend elements
 - The fieldsets must be vertically separated using margins
 - The contents can be given breathing space using padding
 - Light background, with a slightly darker, 1-pixel border

```
fieldset {
  margin: 1em 0;
  padding: 1em;
  border : 1px solid #ccc;
  background: #f8f8f8;
}

legend {
  font-weight: bold;
}
```

Simple form layout

- Position the labels so they appear vertically above the form elements
 - Labels are inline elements by default
 - Setting their display property to block will cause them to generate their own block box, forcing the input elements onto the line below
 - The width of text input boxes varies from browser to browser: set the width of text input boxes

```
label {  
    display: block;  
}  
  
input {  
    width: 200px;  
}
```

Simple form layout

- Unlike text areas and text inputs, radio buttons and check boxes need to be handled differently
 - Rather than having their labels above them, these elements usually have their labels to the right of them
 - When stacked vertically all the elements are left aligned, making them easier to select

Remember Me

Yes

No

Simple form layout

- The width of the text boxes was defined by applying a width to the input element
 - The input element covers other form widgets such as check boxes, radio buttons, and submit buttons
- The best way to distinguish between input elements is to give them a class

```
<fieldset>
<legend>Remember Me</legend>
<p>
<input id="remember-yes" class="radio" name="remember"
      type="radio" value="yes" />
<label for="remember-yes">Yes</label>
</p>
<p>
<input id="remember-no" class="radio" name="remember"
      type="radio" value="no" checked="checked" />
<label for="remember-no">No</label>
</p>
</fieldset>
```


Simple form layout

- Override the previously set input width by setting the width of radio buttons to auto
 - The same can be done for check boxes and submit buttons

```
input.radio, input.checkbox, input.submit {  
  width: auto;  
}
```

- Floating the radio buttons left will bring them back on the same level as their labels
 - A small amount of right margin will provide the desired spacing between the two elements

```
input.radio {  
  float: left;  
  margin-right: 1em;  
}
```

Longer form layout

- For longer and more complex forms, vertical space becomes an issue
 - To improve scanning and reduce the amount of vertical space used, it makes sense to position the labels and form elements horizontally

Your Contact Details

Name: (Required)	<input type="text"/>
Email Address:	<input type="text"/>
Web Address:	<input type="text"/>

Longer form layout

- Instead of setting the label to be a block-level element, float the labels left
- Give the label a width so that all of the form elements line up

```
label {  
  float: left;  
  width: 10em;  
}
```

- This width causes a large gap between the radio buttons
 - Set the width on these labels explicitly

```
#remember-me label {  
  width: 4em;  
}
```

Complex form layout

- Example

Personal Information

Place of Birth:

Date of Birth:

Favorite Color:

<input type="checkbox"/> red	<input type="checkbox"/> orange
<input type="checkbox"/> yellow	<input type="checkbox"/> purple
<input type="checkbox"/> pink	<input type="checkbox"/> blue
<input type="checkbox"/> green	<input type="checkbox"/> other

advanced-form.htm

Complex form layout

- Form labels are important for the accessibility
- However, there are situations when you may not want to display a label for every element
 - e.g. , the date of birth field

```
<p>
<label for="dateOfBirth">Date of Birth:</label>
<input name="dateOfBirth" id="dateOfBirth" type="text" />
<label id="monthOfBirthLabel" for="monthOfBirth">
    Month of Birth:</label>
<select name="monthOfBirth" id="monthOfBirth">
<option value="1">January</option>
<option value="2">February</option>
<option value="3">March</option>
</select>
<label id="yearOfBirthLabel" for="yearOfBirth">Year of
    Birth:</label>
<input name="yearOfBirth" id="yearOfBirth" type="text" />
</p>
```

Complex form layout

- Hide the “month of birth” and “year of birth” labels
 - Setting the labels’ display property to none would stop the labels from displaying, but it would also prevent many screen readers from accessing them
- Solution: position the labels off screen using a large negative text indent
- To prevent the labels from affecting the layout, the width needs to be zeroed down

```
#monthOfBirthLabel, #yearOfBirthLabel {  
    text-indent: -1000em;  
    width: 0;  
}
```

Complex form layout

- The various form controls can then be sized individually and given margins to control their horizontal spacing

```
input#dateOfBirth {
  width: 3em;
  margin-right: 0.5em;
}
select#monthOfBirth {
  width: 10em;
  margin-right: 0.5em;
}
input#yearOfBirth {
  width: 5em;
}
```

Multicolumn check boxes

- Goal: create a two-column layout for large groups of check boxes or radio buttons

Favorite Color:

<input type="checkbox"/>	red	<input type="checkbox"/>	orange
<input type="checkbox"/>	yellow	<input type="checkbox"/>	purple
<input type="checkbox"/>	pink	<input type="checkbox"/>	blue
<input type="checkbox"/>	green	<input type="checkbox"/>	other

- Problem: labels only work for individual elements, not groups of elements

Multicolumn check boxes

- To create the column effect, the check boxes are split into two sets, and each set is wrapped in a div
 - These elements are then grouped together by wrapping them in a fieldset with a descriptive id

```
<fieldset id="favoriteColor">
<h2>Favorite Color:</h2>
<div><p>
<input class="checkbox" id="red" name="red"
      type="checkbox" value="red" />
<label>red</label>
...
</p></div>
<div><p>
<input class="checkbox" id="orange" name="orange"
      type="checkbox" value="orange" />
<label>orange</label>
...
</p></div>
<br class="clear" />
</fieldset>
```

Multicolumn check boxes

- A generic fieldset style has already been created
 - Override those styles, zeroing down the padding and margin, removing the borders and setting the background color to be transparent

```
fieldset#favoriteColor {  
    margin: 0;  
    padding: 0;  
    border: none;  
    background: transparent;  
}
```

Multicolumn check boxes

- The heading is going to act like a label so it needs to be floated left and given a width of 10ems like the other labels
- The headline also needs to look like a label, so the font weight needs to be set to normal and the font size needs to be reduced
- The two-column layout can then be created by giving the divs a width and floating them left

```
#favoriteColor h2 {  
  width: 10em;  
  float: left;  
  font-size: 1em;  
  font-weight: normal;  
}  
#favoriteColor div {  
  width: 8em;  
  float: left;  
}
```

Multicolumn check boxes

- Because the divs are being floated, they no longer take up any space and appear to spill out of the fieldset

Personal Information

Place of Birth: USA

Date of Birth: [] January []

Favorite Color:

- red
- orange
- yellow
- purple
- pink
- blue
- green
- other

- To force the fieldset to enclose these floats, a clearing element has been inserted after the second div
 - a `
` element is used with a class of clear

```
.clear {  
  clear: both;  
}
```

Multicolumn check boxes

- All the labels in this form have been floated left and set to be 10ems wide
- The labels for the check boxes do not need to be floated and require a much smaller width
- Firefox seems to treat the unfloated labels as block-level elements
 - Explicitly set the display property to inline

```
label {  
  width: 3em;  
  float: none;  
  display:  
  inline;  
}
```

Form feedback

- Forms will usually require some type of feedback message to highlight fields that have been missed or incorrectly filled in
 - Usually done by adding an error message next to the appropriate field

Your Contact Details

Name: (Required)

Email Address: **⚠ Incorrect email address. Please try again.**

Web Address:

Form feedback

- The best approach is to include the error message text inside the form label, and then position it using CSS

```
<p>  
<label for="email">Email Address:  
<span class="feedback">Incorrect email  
address. Please try again.</span>  
</label>  
<input name="email" id="email" type="text" />  
</p>
```

Form feedback

- To position the feedback span, set the position of all of the paragraphs in the form to relative, thereby setting up a new positioning context
- Position the feedback span absolutely, so it appears to the right of the text input

```
form p {  
  position: relative;  
}  
.feedback {  
  position: absolute;  
  margin-left: 11em;  
  left: 200px;  
  right :0;  
}
```


Form feedback

- Apply the desired styling to the feedback message
 - E.g. text bold red, plus a warning image to the left side of the message

```
form p {  
  position: relative;  
}  
.feedback {  
  position: absolute;  
  margin-left: 11em;  
  left: 200px;  
  font-weight: bold;  
  color: #760000;  
  padding-left: 18px;  
  background: url(images/error.png) no-repeat left top;  
}
```

References

- Andy Budd, Cameron Moll, Simon Collison, “CSS Mastery, Advanced Web Standards Solutions”
 - www.cssmastery.com/
- CSS reference
 - <https://developer.mozilla.org/en-US/docs/Web/CSS/Reference>
- Dynamic Drive CSS Library
 - <http://www.dynamicdrive.com/style/>
- CSSplay
 - <http://www.cssplay.co.uk/articles/imagemap/>

Licenza d'uso



- Queste diapositive sono distribuite con licenza Creative Commons “Attribuzione - Non commerciale - Condividi allo stesso modo 2.5 Italia (CC BY-NC-SA 2.5)”
- Sei libero:
 - di riprodurre, distribuire, comunicare al pubblico, esporre in pubblico, rappresentare, eseguire e recitare quest'opera
 - di modificare quest'opera
- Alle seguenti condizioni:
 - **Attribuzione** — Devi attribuire la paternità dell'opera agli autori originali e in modo tale da non suggerire che essi avallino te o il modo in cui tu usi l'opera.
 - **Non commerciale** — Non puoi usare quest'opera per fini commerciali.
 - **Condividi allo stesso modo** — Se alteri o trasformi quest'opera, o se la usi per crearne un'altra, puoi distribuire l'opera risultante solo con una licenza identica o equivalente a questa.
- <http://creativecommons.org/licenses/by-nc-sa/2.5/it/>

