

# Style Sheets

# The Presentation Issue

- ❑ The HTML markup elements allow to specify
  - ❖ The **logic structure** of the document (paragraph, titles, etc.)
  - ❖ Some **aspects of the presentation** (colors, fonts to be used to render text, etc.)
- ❑ All other presentation choices (space between lines, space between characters, etc.) are left to **browser implementation**
- ❑ The necessity to better control presentation issues led to complexity and incompatibility issues:
  - ❖ Proprietary extensions of HTML
  - ❖ Massive use of tables, images as spaces, code

# Style Sheet

- ❑ Introduced with HTML 4

- ❖ Allow to better define the style and presentation aspects
- ❖ Allow separation between content and presentation

- ❑ In one word:

A **style** allows to define the physical structure (e.g., layout) of a document starting from its logical structure

# Why using Style Sheets?

- ❑ Separate structure from presentation
- ❑ Do not tweak the document structure just for presentation purposes
- ❑ Ensure coherency across pages
  - ❖ All pages in a web site can refer to the same style sheet (file)
- ❑ Improve accessibility
- ❑ Possibility to change style depending on device or user
- ❑ Reduce the size of web pages

# Style Sheet

- ❑ The description of the presentation style be written
  - ❖ Using different languages (the w3c one is **CSS**: Cascading Style Sheet)
  - ❖ In the HTML document itself or in separate files. If the language allows it, hierarchies of style files can be created (cascading)
- ❑ Different presentation styles can be used for different devices (e.g., screen, print)

# CSS Timeline and Resources

- ❑ CSS 1: W3C recommendation (Dec 1996)
- ❑ CSS 2.1: W3C Candidate Recommendation (July 2007)
- ❑ CSS 3: Working Draft
  
- ❑ Resources:
  - ❖ CSS2.1 standard, <http://www.w3.org/TR/CSS21/>
  - ❖ W3C CSS Tutorial,
  - ❖ <http://www.w3.org/Style/Examples/011/firstcss>

# Specifying the Language

- ❑ To use a style sheet it is necessary to specify the language used by the style sheet
- ❑ It can be done in a META element or in the HTTP headers

```
<META http-equiv="Content-Style-Type"  
      content="text/css">
```

```
Content-Style-Type: text/css
```

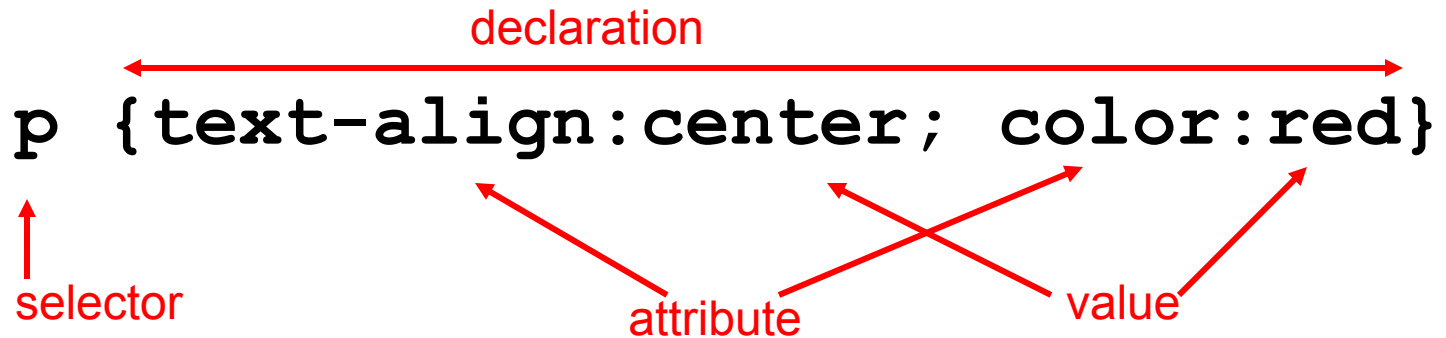
# CSS Language

- ❑ CSS is based on rules
- ❑ A rule is a statement about one stylistic aspect of one or more elements
- ❑ A style sheet is a set of one or more rules that apply to an HTML document
  - ❖ A CSS file is a set of rules



# CSS Rule

## □ Generic form of a CSS rule

  
The diagram shows a CSS rule: `p {text-align:center; color:red}`. A red double-headed arrow labeled "declaration" spans the curly braces and their contents. A red arrow labeled "selector" points to the `p`. Two red arrows labeled "attribute" point to `text-align` and `color`. Two red arrows labeled "value" point to `center` and `red`.

- Selector: the part before the left curly bracket
  - ❖ It specifies what elements are affected by the declaration
- Declaration: the part within the curly brackets
  - ❖ It sets what the effect will be

# Selectors

- ❑ There are several kind of selectors
- ❑ The **type** selector is the simplest kind of selector
- ❑ The selector **P** in the previous slide is a type selector and is based on an HTML element type
- ❑ The final effect of the previous rule is
  - ❖ Center the text in all paragraphs, and make it red

```
p {text-align:center; color:red}
```

# ID Selector

- ❑ All elements with id="para" will be formatted accordingly
- ❑ Use the «#» syntax

```
#para{ text-align:center; color:red}
```

.....

```
<p id="para">Hello world!</p>
```

```
<p>Part with different style</p>
```

# Class Selector

- Useful to define styles as in a word processor
- Use the «.» syntax

```
.centered {text-align:center}
```

.....

```
<h1 class="centered">Centered  
  title</h1>
```

```
<p class="centered">Centered  
  paragraph</p>
```

# Mix of type and other selectors


- It is possible to mix a type and class selector

```
<p class="author">Mario Rossi</p>
```

```
p.author {font-style: italic; font-  
color: red;}
```

# Style hierarchies

A style can be specified by means of:

- 
1. An external file (**external style**)
  2. A **<style>** tag inside the HTML document (**internal style**)
  3. Specifying *style* elements in any tag of the document (**in-line style**)

inside

In case of conflicting rules, the innermost rule is applied

# External Style Specifications

- ❑ Included in external files. Ex. : mystyle.css

```
H1.myclass {border-width: 1; border: solid;
text-align: center}
P.special {color: green; border: solid red}
```

- ❑ Associated to the document using LINK elements in the header section:

```
<HEAD>
<LINK href="mystyle.css" rel=stylesheet
type="text/css">
</HEAD>
<BODY>
  <H1 class="myclass"> Title with style
  applied</H1>
  <P class="special"> Paragraph with style
  applied
</BODY>
```

# Internal Style Specifications

## □ Inside the HTML FILE:

**<HEAD>**

**<STYLE type="text/css">**

**H1.myclass {border-width: 1;  
border: solid; text-align: center}  
</STYLE>**

**</HEAD>**

**<BODY>**

**<H1 class="myclass"> Title with style  
applied</H1>**

**<H1> Title without style applied</H1>**

**</BODY>**



# Inline Style Specifications

- They can be included in the elements, using the attribute «style»:

```
<P style="font-size: 12pt; color: red">  
paragraph with text size 12 points in  
red</P>
```

# Other Useful Attributes

## □ background-color

❖ Color can be specified as:

- name ex. red
- Decimal RGB ex. `rgb(255,0,0)`
- Hex RGB ex. `#FF0000`

```
h1 {background-color:#6495ed;}
```

```
p {background-color:#e0ffff;}
```

```
div {background-color:#b0c4de;}
```

### CSS background-color example!

This is a text inside a div element.

This paragraph has it's own background color.

We are still in the div element.

# Backgrounds

## ❑ background-image

- ❖ Applicable to <body> tag only
- ❖ Specifies the URL to use for the background image

```
body{background-image:url("logo_poli.png");}
```

## ❑ background-repeat

- ❖ Allows to repeat background horizontally or vertically

```
body (background-image:url("logo_poli.png");  
    background-repeat:repeat-x;}
```

# Result



# DIV elements

- ❑ It is possible to group elements in a page into a `<DIV> </DIV>` block
- ❑ `id` and `class` attributes can be used to modify the presentation of the whole block



# Anchors

- The default colors for links can be changed using pseudo-classes (e.g. :visited)

```
a:link {color:#0000ff;} /* not  
visited link (blue) */
```

```
a:visited {color:darkred;} /*  
already visited link */
```

```
a:hover {color:#ff0000;} /* mouse  
over the link (red) */
```

```
a:active {color:#00ff00;} /*  
selected link (green) */
```

hover must follow link & visited

active must follow hover



# Other anchor examples

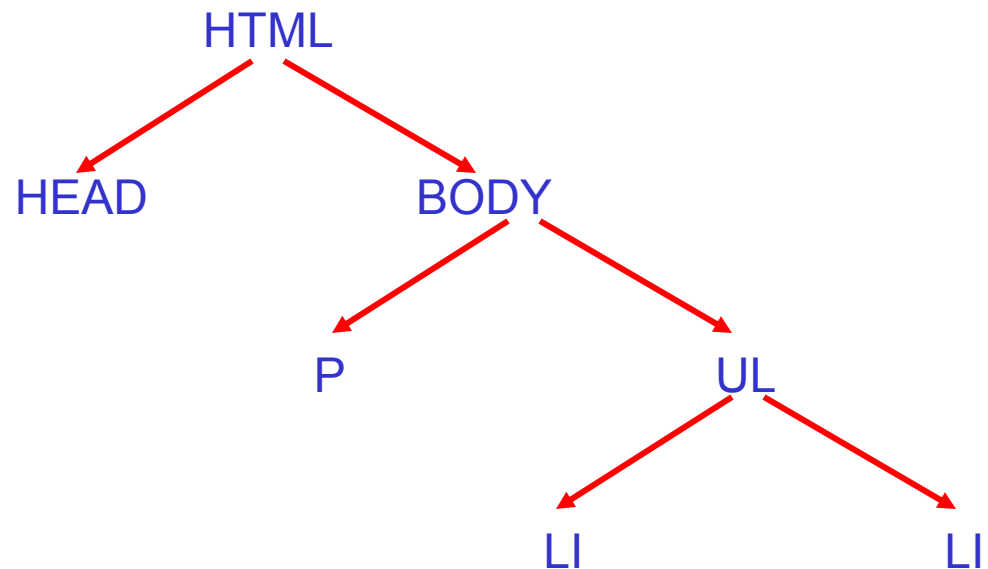
- Different backgrounds and underlining can be defined for the various link states

```
a:link {text-decoration:none;}
a:visited {text-decoration:none;}
a:hover {text-decoration:underline;}
a:active {text-decoration:underline;}

a:link {background-color:#B2FF99;}
a:visited {background-color:#FFFF85;}
a:hover {background-color:#FF704D;}
a:active {background-color:#FF704D;}
```

# Inheritance of style values

- HTML documents are structured as trees
- Style values are inherited by children
  - ❖ Some exceptions (e.g. background)





# Tag «block» and «inline»

- ❑ The text layout algorithm explores the tag tree
  - ❖ A visualization policy is associated to each tag by means of the «display» property attribute
- ❑ All tags of type «block» are put into a rectangle
  - ❖ The tag content is layed out recursively inside the rectangle
- ❑ Content of «inline» tags is layed out consecutively, moving to the next line if necessary

# Tag «block» and «inline» (2)

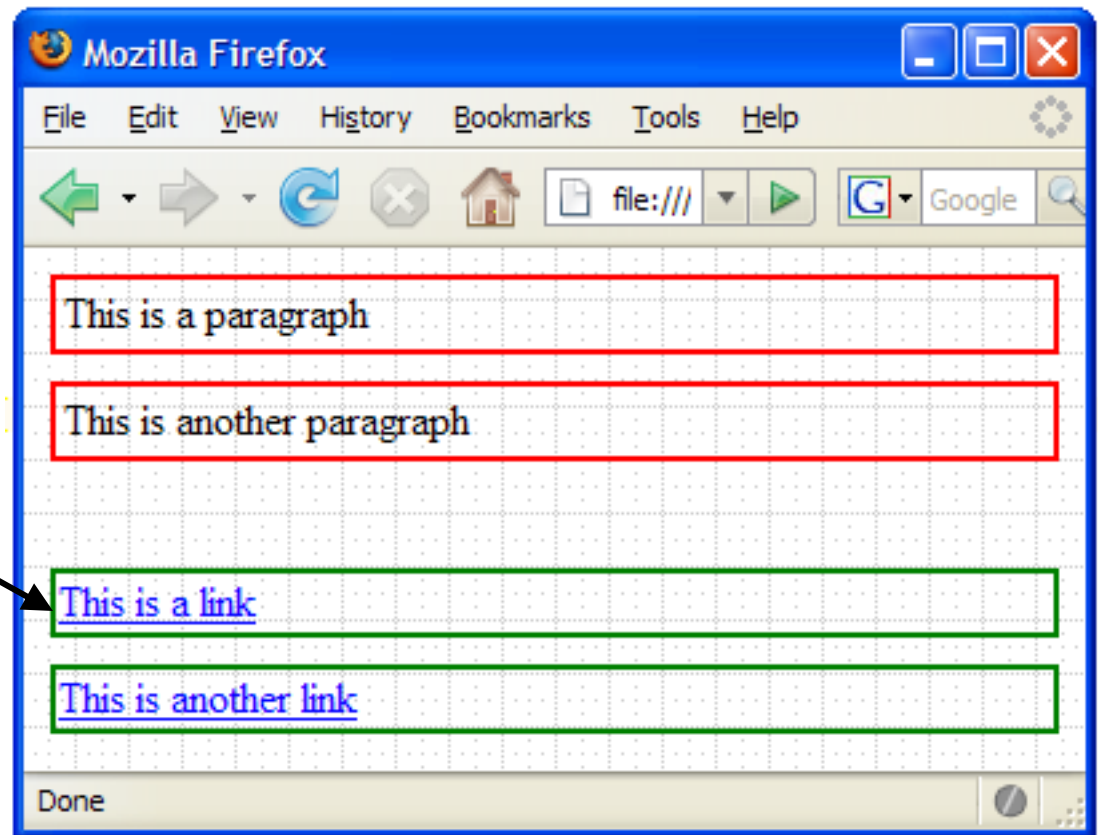
- ❑ Block-level elements are presented visually as blocks, i.e., they generate a line break after themselves (e.g., `<p>`, `<ul>`, `<div>`, ...)
  - ❖ Act as containing blocks for nested elements
- ❑ All other elements are inline elements (except if the «display» property attribute is set to block)
  - ❖ No line breaks before and after them (e.g., `<b>`, `<i>`, `<span>`, ...)
  - ❖ Cannot contain other block-level elements

# Example

```
body { background-image: url(baseimg.gif); margin: 0px; }
```

```
p {  
margin: 10px;  
border: solid;  
border-width: 2px;  
border-color: red;  
padding: 3px;  
}
```

```
a {  
display: block;  
margin: 10px;  
border: solid;  
border-width: 2px;  
border-color: green;  
padding: 1px;  
}
```



# 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



**NB: only basic concepts are presented here**

# Box Elements

## ❑ 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

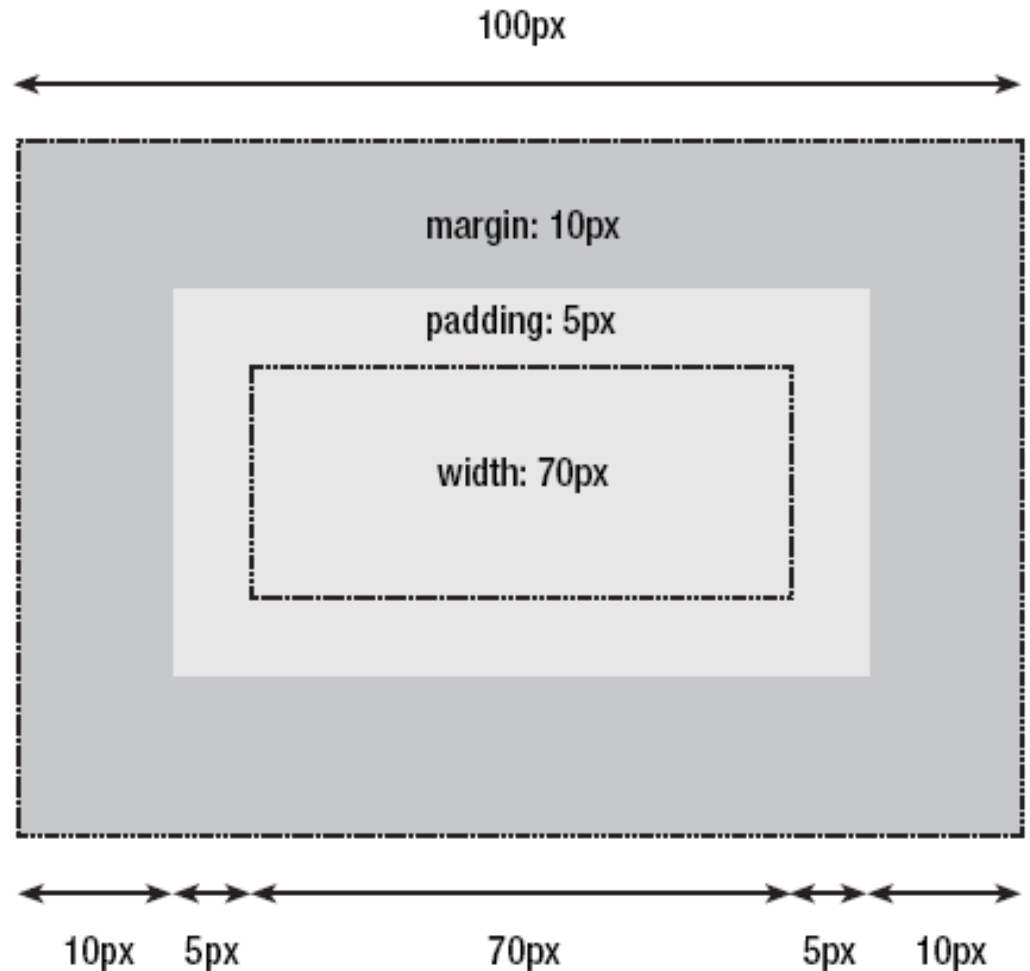
## ❑ **Padding, borders, and margins are optional and default to zero**

[See also [http://www.w3schools.com/Css/css\\_positioning.asp](http://www.w3schools.com/Css/css_positioning.asp) ]



# Box Elements

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

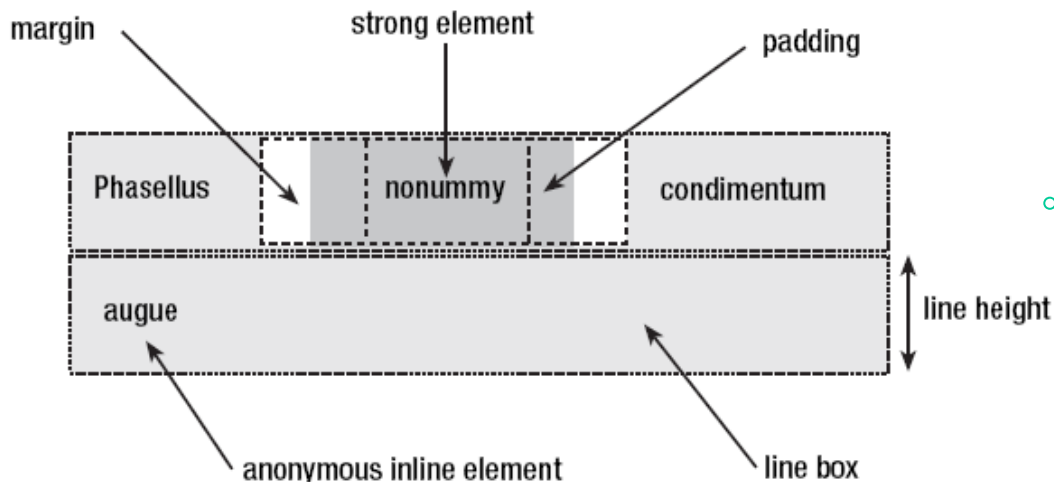


# Positioning Schemes

- ❑ 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 HTML

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

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





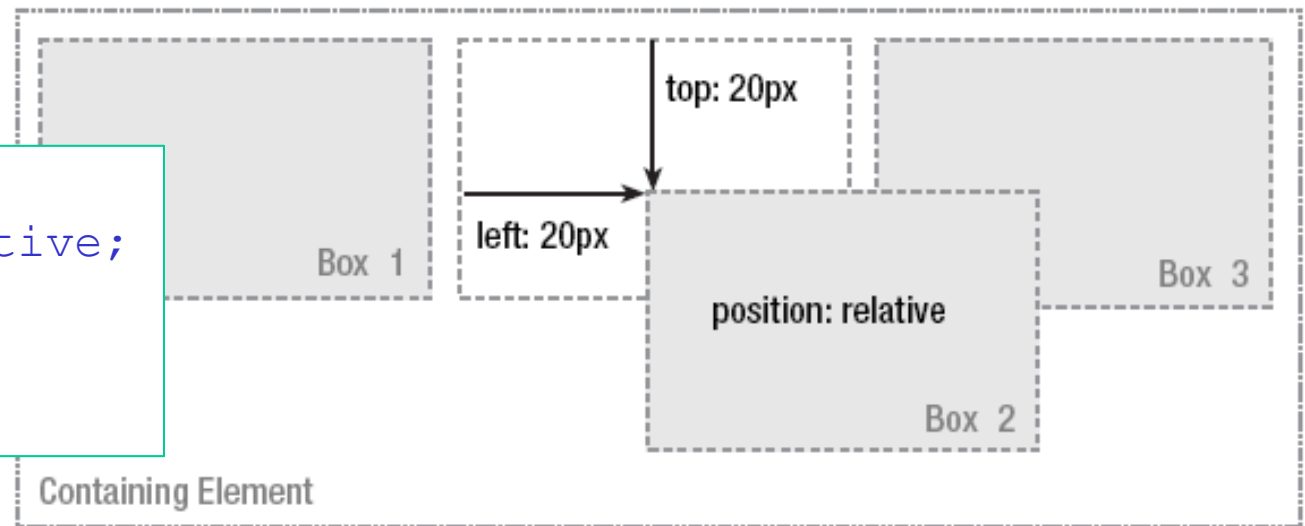
# Box Positioning

- ❑ 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

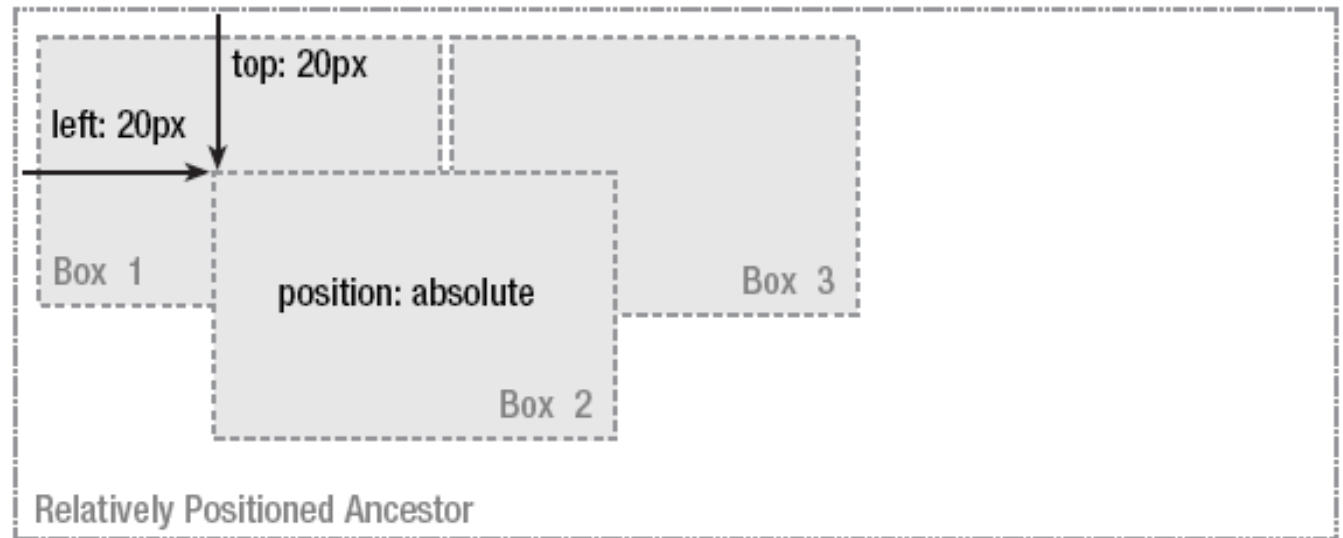
- ❑ It is possible to shift one element “relative” to its starting point by setting a vertical or horizontal position

```
#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



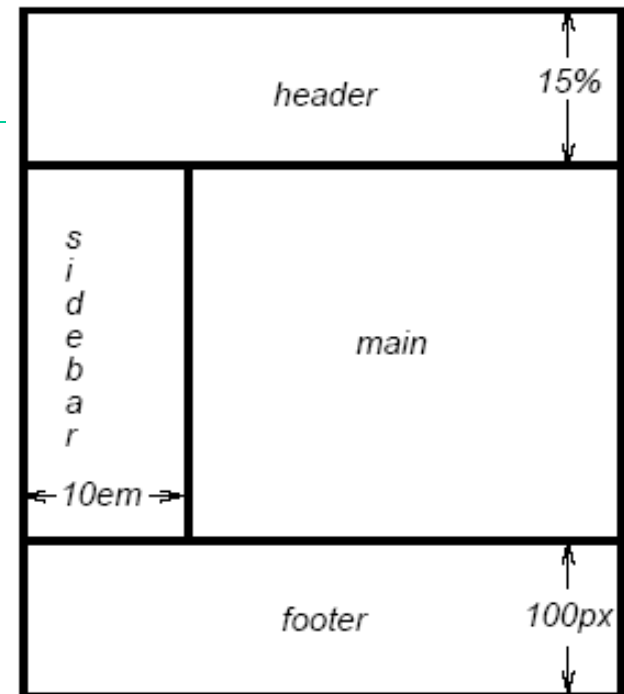
# Fixed Positioning

- ❑ 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 Example

- ❑ 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; }
```



# Other Examples

❑ W3Schools.com

❑ [http://www.w3schools.com/Css/css\\_positioning.asp](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

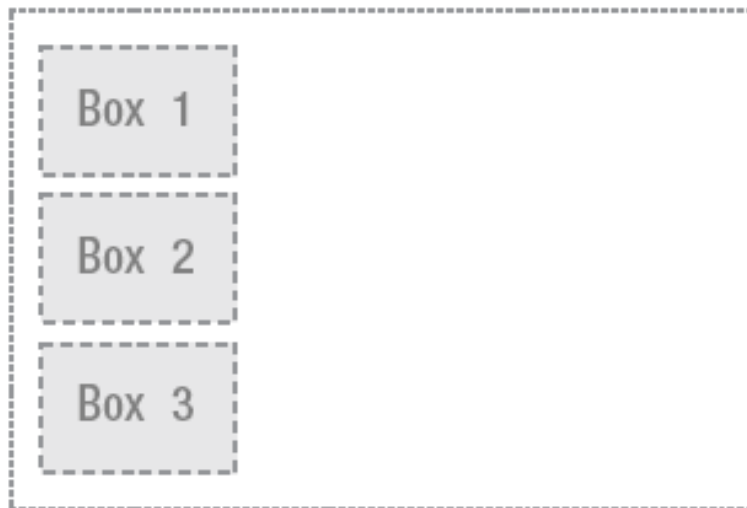
- ❑ A floated box can either be shifted to the left or the right until its outer edge touches the edge of its containing box, or another floated box
- ❑ Often used for images and when working with layouts
- ❑ [http://www.w3schools.com/Css/css\\_float.asp](http://www.w3schools.com/Css/css_float.asp)

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

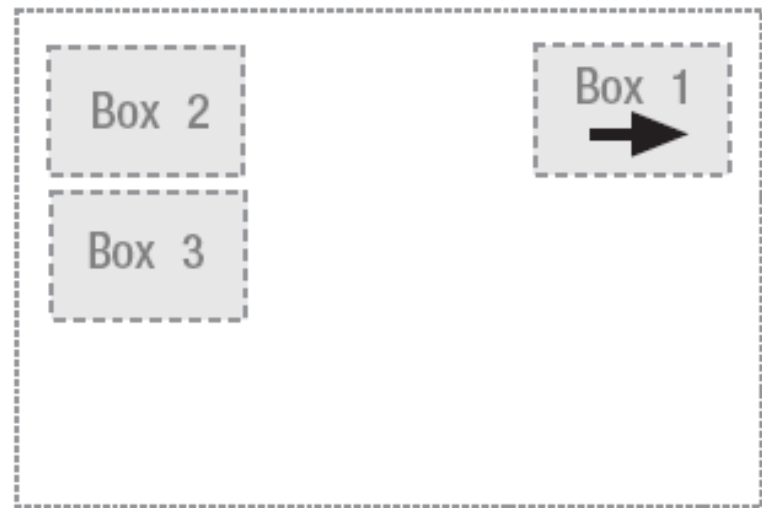
# Floating

- ❑ 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





# Summary

- ❑ 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
- ❑ Using the CSS box model parameters very flexible and «adaptive» layouts can be created
  - ❖ Also text around boxes can be managed (e.g., figures surrounded by text)
- ❑ No need of TABLE for formatting purposes, just use e.g. DIV tags + CSS

# Practical Advice

- ❑ As in any programming activity, it is often simpler to start from an example than from scratch
- ❑ CSS files are resources as HTML, images, etc: they can be downloaded

# CSS3

- ❑ Old syntax still ok. Addition w.r.t. CSS 2.1:
  - ❖ Can target almost any element in the page with a wide range of selectors
  - ❖ New selectors
    - Relational selectors (child, sibling, etc.)
    - Attribute selectors (elements with specific attribute)
  - ❖ Colors with transparency and other color models
  - ❖ **Rounded corners**, drop **shadows**, text shadows,
  - ❖ **Gradients** (linear, radial), multiple background images
- ❑ Some **features** very much used were already possible in previous versions, but required several «tricks» (images etc.)

# CSS3

- ❑ Other addition with respect to CSS 2.1:
  - ❖ Transforms (scaling, rotation, skew), transitions, animations  
(e.g., see: <http://css3.bradshawenterprises.com/cfimg/> )
  - ❖ Loading and using fonts different from the ones installed in the system
  - ❖ Multicolumn layout
- ❑ Conclusion: CSS3 allows to build much nicer interfaces both for web sites and web applications
  - ❖ especially when coupled with code that reacts to events and modify properties on-the-fly)

# CSS Considerations

- ❑ Proprietary extensions available in certain browsers (or better, “layout engine components” of the browsers, for example webkit)
  - ❖ Especially for new, experimental features
- ❑ Standard CSS does not allow definition of variables
  - ❖ Extension languages have been developed (e.g., SASS - “Syntactically Awesome Style Sheets”) to expand CSS functionalities
  - ❖ Those languages can be “compiled” into standard CSS