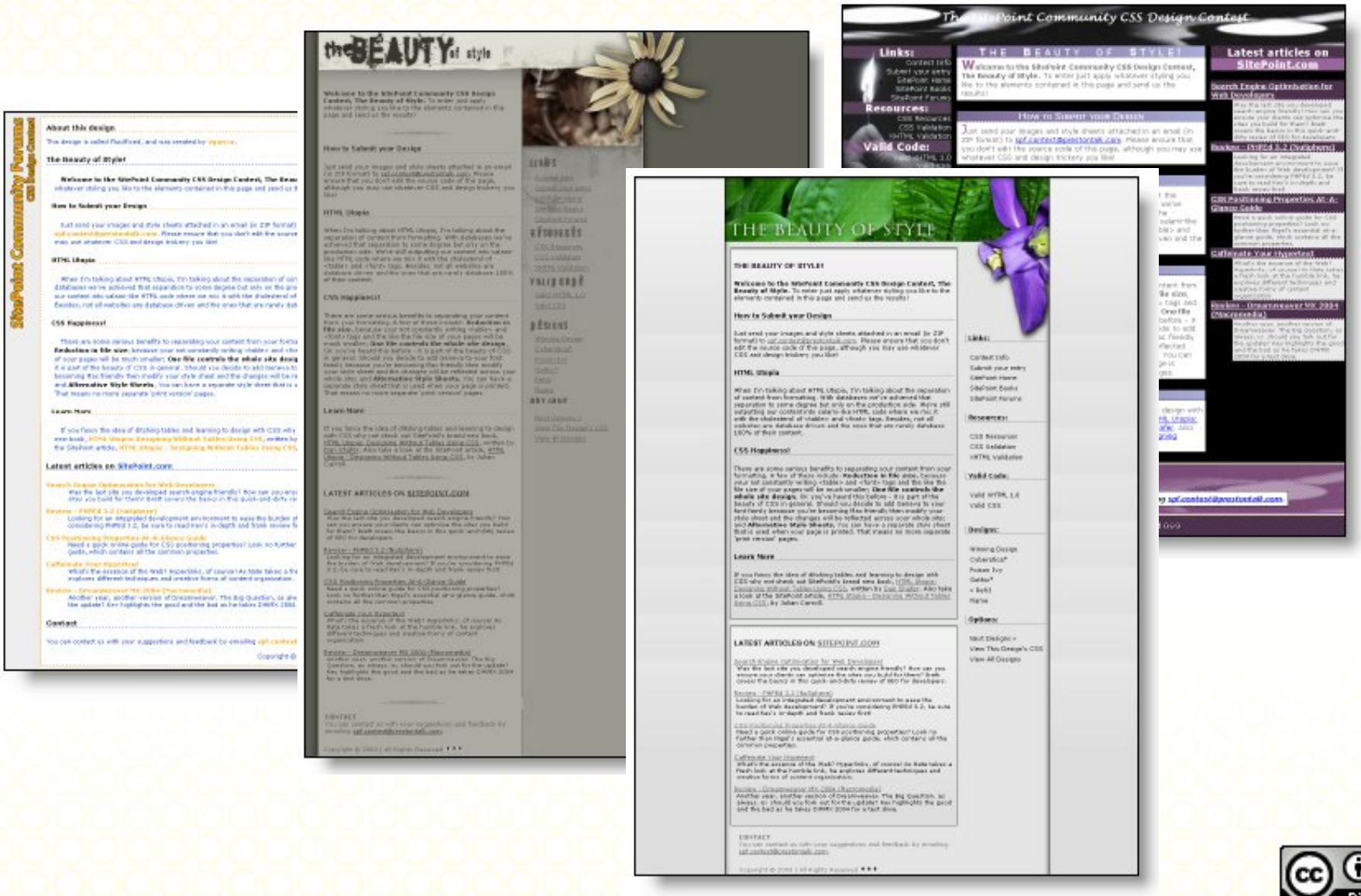
SGG 3643

Computer Programming III

Cascading Style Sheets (CSS)

Ivin Amri Musliman

The beauty of Style




CSS ZEN GARDEN

the road to enlightenment
the beauty of css design


ZEN GARDEN
The Beauty of CSS Design
Select a Design
The Road to Enlightenment
So What is This About?
Archives
Resources
Participation
Benefits
Requirements


CSS Zen Garden
Select a Design
The Road to Enlightenment
So What is This About?

the Road to Enlightenment
So What is This About?
Participation
Benefits
Requirements

Web References

- The Layout Reservoir, http://www.bluerobot.com/web/layouts/
- Web Page Reconstruction with CSS, http://www.digital-web.com/tutorials/tutorial_2002-06.shtml
- Using Style Sheets, http://www.brainjar.com/css/using/
- Little Boxes, http://www.thenoodleincident.com/tutorials/box_lesson/boxes.html
- CSS Layout Techniques: for Fun and Profit, http://glish.com/css/
- Css/edge, http://www.meyerweb.com/eric/css/edge/
- CSS FAQ, http://www.mako4css.com/CSSFAQ.htm
- CSS Support Charts, http://devedge.netscape.com/library/xref/2003/css-support/
- http://devedge.netscape.com/central/css/
- http://www.cssbook.com/resources/css/index.html

Resources

- Cascading Style Sheets, level 2
  CSS2 Specification: http://www.w3.org/TR/REC-CSS2/
- http://www.glish.com/css/
- http://www.glish.com/css/7.asp

Stylesheets: Principle

- *A la WinWord*: self defined format definitions (physical markup)
- If you must assign many passages in the text with a certain combination of format instructions, it is worthwhile to define these instructions in a Stylesheet in order to have to assign all formatting instructions in a step afterwards.

What we will learn

- What style sheets are
- Use the style element
- Link to separate style sheets
- Set page margins
- Set left and right and first-line indents
- Set the amount of whitespace above and below
- Set the font type, style and size
- Add borders and backgrounds
- Set colors with named or numeric values

What is CSS?

- CSS stands for Cascading Style Sheets
- Styles define how to display HTML or XML elements
- Styles are normally stored in Style Sheets .
- Styles were added to HTML 4.0 to solve a problem.
- External Style Sheets can save you a lot of work.
- External Style Sheets are stored in CSS files.
- Multiple style definitions will cascade into one.

Source: http://www.w3schools.com

Stylesheets: Advantages

- free and flexible design of complete websites,
- precise positioning of single elements of web pages,
- cascading, modular character
- reduced number of HTML tags in single web pages
  - Less work for the author
  - Style Sheets Can Save a Lot of Work!
  - Reduced download times.
- *Easier to update* site designs
- *Structured content* in your Web documents.
- *More control* over the design of your Web pages.
- Permits you to design *how Web pages look when printed.*

Where are the styles?

- Style Sheets allow style information to be specified in many ways.
  - Styles can be specified inside a single HTML element,
  - inside the <head> element of an HTML page, or
  - in an external CSS file.
- Even multiple external Style Sheets can be referenced inside a single HTML document.

Cascading: Multiple Styles Will Cascade Into One

- Generally speaking: All the styles will "cascade" into a new "virtual" Style Sheet by the following rules, where number four has the highest priority:

1. Browser default
2. External Style Sheet
3. Internal Style Sheet (inside the <head> tag)
4. Inline Style (inside HTML element)

- So, an inline style (inside an HTML element) has the highest priority, which means that it will override every style declared inside the <head> tag, in an external style sheet, and in a browser (a default value).

Stylesheets definition and use I

- A style sheet is a collection of style attributes and their values
- The single style attributes are separated by semi-colons:

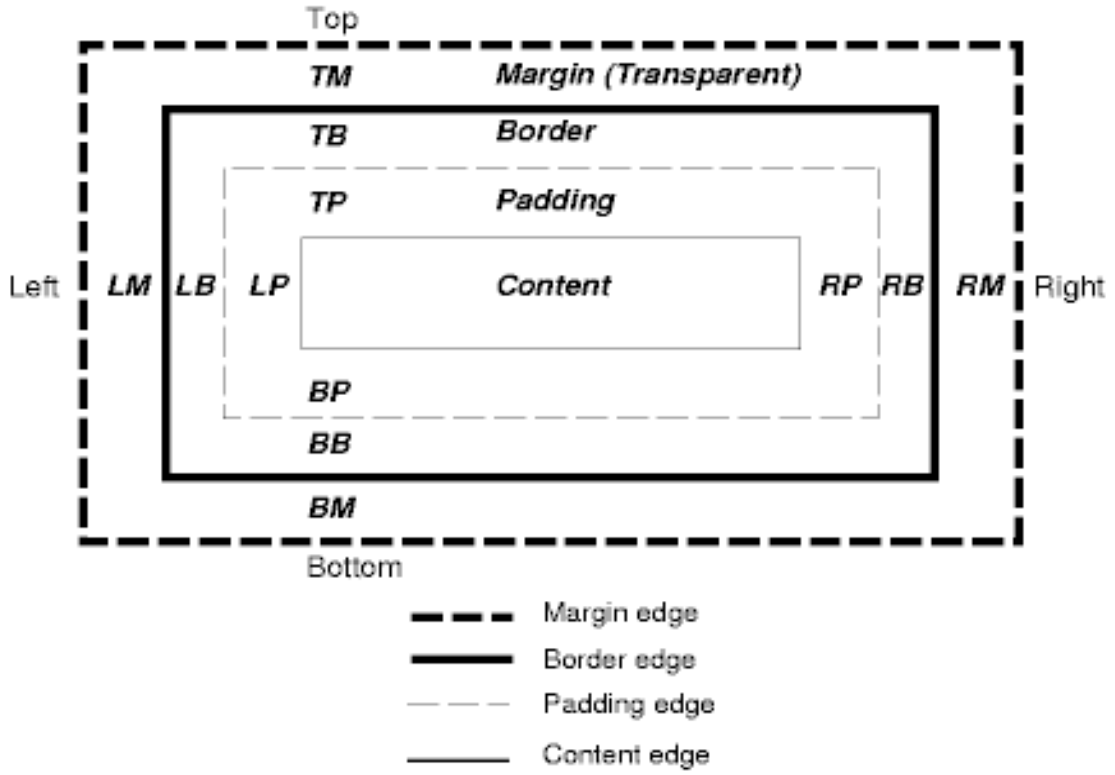
```
attribute1: value; attribute2: value; ...
```

- To make styles more readable you can write every attribute in a single line:

```
text-align: center;
color: black;
font-family: arial
```

The box model

- Each box has a content area (e.g., text, an image, etc.) and optional surrounding padding, border, and margin areas; the size of each area can be specified by properties.



Source: http://www.w3.org/TR/REC-CSS2/box.html

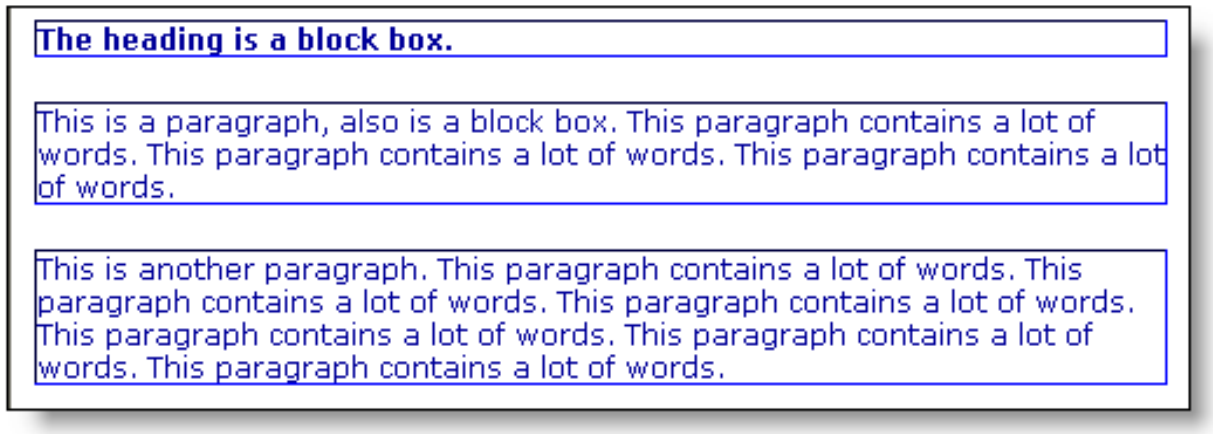
Box Types

- Boxes come in a number of different types. The only types we are concerned with are
  - block boxes and
  - inline boxes.
- Properties can be defined using units like %, em, px, ex, pt, mm, cm, in or pc.
- Use relative sizes if possible!

Block Boxes

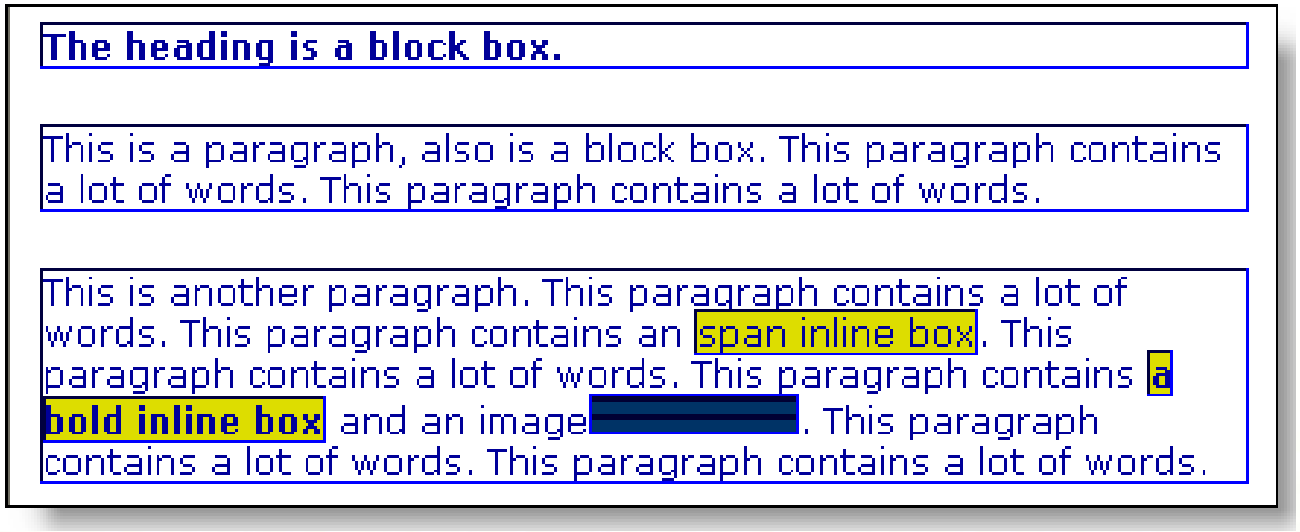
- http://www.w3.org/TR/REC-CSS2/visuren.html#q5

- Block boxes are the types of boxes generated (by default) by elements like p and div. A black box has space around it, and elements that come after start below the box instead of next to it.



Inline boxes

- Inline elements (<b>, <img>, , <pre> are displayed next to each other.

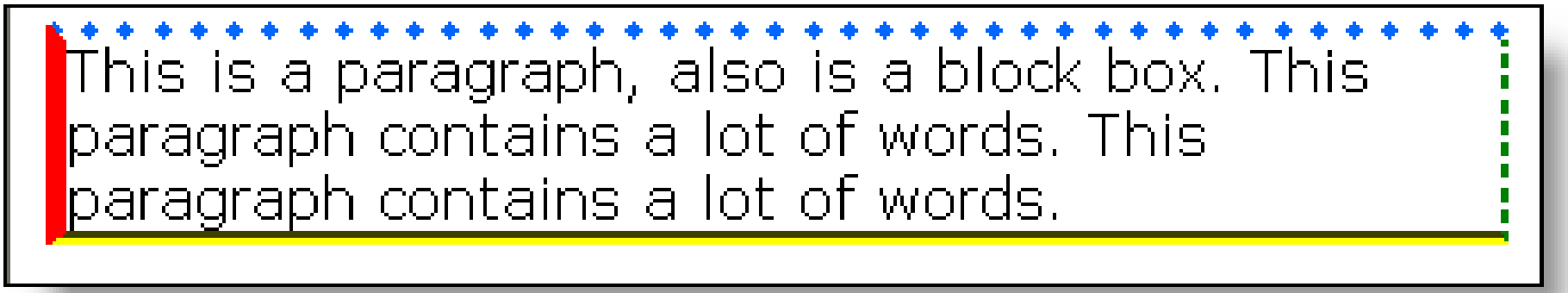


Properties of Border

- border-width thin | medium | thick | <length>]{1,4}
- border-color
- border-style none | dotted | dashed | solid | double | groove | ridge | inset | outset
- border-top <border-top-width> || <border-style> || <color>
- border-right, border-bottom, border-left
- border: combination of the mentioned properties.

Properties of Border - Example

```
p {
border-top: 5px dotted #0066FF;
border-bottom: 2px groove YELLOW;
border-bottom-width: medium;
border-left: 6px solid RED;
border-right: 2px dashed GREEN;
}
```



Underline / overline

```
h2
{
  border-width: 0.5em 0em 0.5em 0em;
  border-style: solid; border-color: maroon
}
...
<h2>A heading</h2>
```

Style definition

- Definition of style can be done in several ways:
  - Inline-Styles
  - embedded stylesheets
  - External Stylesheets (recommended!)

Deprecated in XHTML!

Inline-Style

- To start you can define style attribute for single HTML tags in the <body>-section of your web page:

```
<p style="color: red; font-family: sans-serif; text-
align: right">This paragraph uses a font without serifs
(normally Arial), appears in red color and is right
oriented</p>

<p>Dieser Absatz erscheint in Standardschrift. Lediglich
<span style="text-decoration: underline">diese
Textpassage</span> ist durch Unterstreichung
hervorgehoben.</p>
```

Imbedded Style Definitions

- Stylesheet definitions can be included with

  <style type="text/css"> ... </style>

  in the <head> section of your HTML page.

- Alternative definition of the style specification:

```
<meta http-equiv="content-style-type"
content="text/css">
```

Media types and Stylesheets I

- Example:

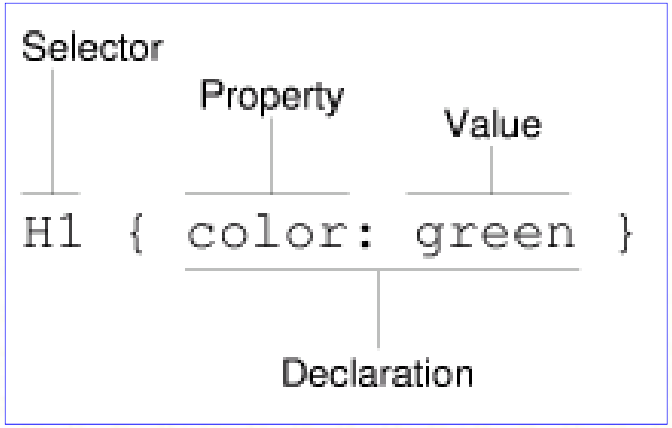
```
...<link rel=stylesheet media="screen"
href="website.css"><link rel=stylesheet media="print"
href="printer.css"><link rel=stylesheet media="aural"
href="speaker.css">...
```

Media types and Stylesheets II

- Currently the following media types are defined:
  - screen (normal screen),
  - tty (Terminal),
  - tv (TV),
  - projection (Projector),
  - handheld (small screens like PalmPilots),
  - print (printer),
  - Braille (for blind persons), aural (with spoken output) and
  - all.
- In current browsers not completely supported.

Defining styles for HTML elements: Selectors

- Three types of selectors:
  - HTML-Tag-Selectors
  - class-Selectors and
  - ID-Selectors
- The selectors are written before the style attributes.
- In this way style definitions get something like a name.
- Using this name, style definitions get a "name" by which the styles are applied to HTML elements.



Pseudo class Selectors

- Dedicated definition of style properties for some instances of specific HTML-Tags
- Arbitrary name for a style definition, preceded by a dot:

```
.mystylename {color: rgb(255,0,255); font-family: Arial}
```

- Use of this definition using the class attribute of a HTML element:

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

Type selectors

- A *type selector* matches the name of a document language element type. A type selector matches every instance of the element type in the document tree.
- HTML based type selectors have the same name as the corresponding HTML-Tags (i.e. <p>, <h1>, ..., but without brackets)!
- Examples:

```
<style type="text/css">
h1 { color: rgb(255,0,255) }
p { margin-left: 10%; margin-right: 10%}
</style>
```

Pseudo-classes

- Pseudo-classes classify elements on characteristics other than their name, attributes or content; in principle characteristics that cannot be deduced from the document tree.
- The exception is ':first-child', which *can* 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.

Class-Selectors: Example

```
<head>...
  <title>The Title</title>
  <style type="text/css">
  <!--
  p.normal { font-size:10pt; color:black; }
  p.big { font-size:12pt; color:black; }
  p.small { font-size:8pt; color:black; }
  all.red { color:red; }
  .blue { color:blue; }
  //-->
  </style>
  </head>
  <body>
  <p class="normal">Normal text</p>
  <p class="big">Big text</p>
  <p class="small">Small text</p>
  <p class="red">Red text</p>
...
```

ID Selectors

- Document languages may contain attributes that are declared to be of type ID. What makes attributes of type ID special is that no two such attributes can have the same value; whatever the document language, an ID attribute can be used to uniquely identify its element.
- In HTML all ID attributes are named "id";
- XML applications may name ID attributes differently, but the same restriction applies.
- The ID attribute of a document language allows authors to assign an identifier to one element instance in the document tree. CSS ID selectors match an element instance based on its identifier.
- A CSS ID selector contains a "#" immediately followed by the ID value.

ID Selectors

- Style assignment to *one* HTML element:

```
#pspecial { margin-left: 10%; margin-right: 10%}
...
<p id=" pspecial ">... </p>
```

ID Selectors

- In the following example, the style rule matches the element that has the ID value "z98y". The rule will thus match for the P element:

```
<HEAD> <TITLE>Match P</TITLE> <STYLE
type="text/css"> *#z98y { letter-spacing: 0.3em }
</STYLE> </HEAD> <BODY> <P id=z98y>Wide text</P>
</BODY>
```

- In the next example, however, the style rule will only match an H1 element that has an ID value of "z98y". The rule will not match the P element in this example:

```
<HEAD> <TITLE>Match H1 only</TITLE> <STYLE
type="text/css"> H1#z98y { letter-spacing: 0.5em }
</STYLE> </HEAD> <BODY> <P id=z98y>Wide text</P>
</BODY>
```

Child selectors

- A *child selector* matches when an element is the child of some element. A child selector is made up of two or more selectors separated by ">".
- The following rule sets the style of all P elements that are children of BODY:

```
BODY > P { line-height: 1.3 }
```

- The following example combines descendant selectors and child selectors:

```
DIV OL>LI P
```

It matches a P element that is a descendant of an LI; the LI element must be the child of an OL element; the OL element must be a descendant of a DIV. Notice that the optional whitespace around the ">" combinator has been left out.

Adjacent sibling selectors

- Adjacent sibling selectors have the following syntax: E1 + E2, where E2 is the subject of the selector. The selector matches if E1 and E2 share the same parent in the document tree and E1 immediately precedes E2.
- In some contexts, adjacent elements generate formatting objects whose presentation is handled automatically (e.g., collapsing vertical margins between adjacent boxes). The "+" selector allows authors to specify additional style to adjacent elements.
- Thus, the following rule states that when a P element immediately follows a MATH element, it should not be indented:

Adjacent sibling selectors II

- MATH + P { text-indent: 0 } The next example reduces the vertical space separating an H1 and an H2 that immediately follows it:
- H1 + H2 { margin-top: -5mm } The following rule is similar to the one in the previous example, except that it adds an attribute selector. Thus, special formatting only occurs when H1 has class="opener":
- H1.opener + H2 { margin-top: -5mm }

Attribute selectors

- CSS2 allows authors to specify rules that match attributes defined in the source document.
- Attribute selectors may match in four ways:
  - [att]: Match when the element sets the "att" attribute, whatever the value of the attribute.
  - [att=val]: Match when the element's "att" attribute value is exactly "val".
  - [att~=val]: Match when the element's "att" attribute value is a space-separated list of "words", one of which is exactly "val". If this selector is used, the words in the value must not contain spaces (since they are separated by spaces).

Attribute selectors II

- [att|=val]: Match when the element's "att" attribute value is a hyphen-separated list of "words", beginning with "val". The match always starts at the beginning of the attribute value. This is primarily intended to allow language subcode matches (e.g., the "lang" attribute in HTML) as described in RFC 1766 ([RFC1766]).

Attribute selectors: Examples

- For example, the following attribute selector matches all H1 elements that specify the "title" attribute, whatever its value:

```
H1[title] { color: blue; }
```

- In the following example, the selector matches all SPAN elements whose "class" attribute has exactly the value "example":

```
SPAN[class=example] { color: blue; }
```

The language pseudo-class: :lang

- If the document language specifies how the human language of an element is determined, it is possible to write selectors in CSS that match an element based on its language. For example, in HTML [HTML40], the language is determined by a combination of the "lang" attribute, the META element, and possibly by information from the protocol (such as HTTP headers). XML uses an attribute called XML:LANG, and there may be other document language-specific methods for determining the language.

The language pseudo-class: :lang Examples

- The following rules set the quotation marks for an HTML document that is either in French or German:
- HTML:lang(fr) { quotes: '« ' ' »' }
  HTML:lang(de) { quotes: '»' '«' '\2039' '\203A' }
  :lang(fr) > Q { quotes: '« ' ' »' }
  :lang(de) > Q { quotes: '»' '«' '\2039' '\203A' }

The :first-line and :first-letter pseudo-elements

- p:first-line { text-transform: uppercase }

THIS IS A PARAPRAPH. THIS PARAGRAPH CONTAINS A LOT OF words. This paragraph contains a lot of words. This paragraph contains a lot of words.

p:first-letter { font-size: 200%; font-style: italic; font-weight: bold; float: left }

This is another parapraph. This paragraph contains a lot of words. This paragraph contains a lot of words. This paragraph contains a lot of words. This paragraph contains a lot of words. This paragraph contains a lot of words. This paragraph contains a lot of words.

Selectors

- Type selectors: One style definition for all elements of one tag.
- Class selectors: One style definition elements of one or different tags.
- Id selectors: One style definition for one element.

Properties: Colors and Background

- Text color: color: `rgb(100%, 0%, 0%)`
- background-color: `background-color: gray;`
- Background image, background-position, background-repeat
  repeat | repeat-x | repeat-y | no-repeat
  ```
  <p style="background-image: url('saturn.jpg');
  background-position: 50% 50%; background-repeat:
  no-repeat">
  ```
- background-attachment scroll | fixed

Units of Measure in CSS

- Relative units:
  - **em**: relative to the size of the upper-case M in the font-size currently in use on this element
  - **ex**: relative to the 'x-height' (lower-case x) of the font-size currently in use on this element
  - **%**: relative to the font-size currently in use on this element.
- Absolute units:
  - **in**: inches
  - **cm**: centimeters
  - **mm**: millimeters
  - **px**: pixels
  - **pt**: points – the points used by CSS2 are equal to 1/72nd of an inch
  - **pc**: picas – 1 pica is equal to 12 points (6 picas in an inch)

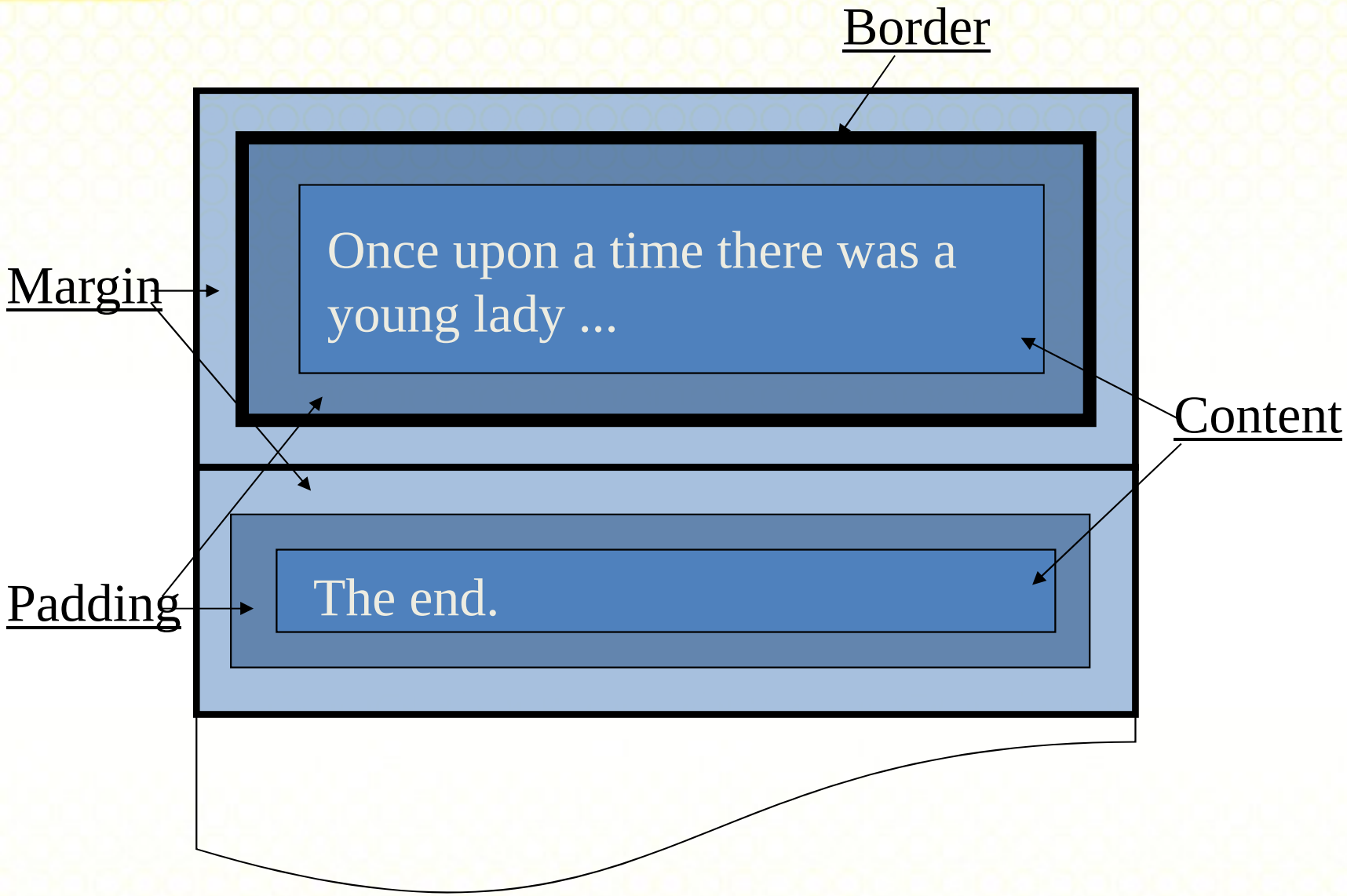
Units of Measure in CSS

- Not all are supported by all browsers
- Those that are supported are not always displayed the same way among the browsers
- For the best likelihood of cross-browser/cross-platform consistency, use **%** or **em**.

About em and points

- The em is a very useful unit as it scales with the size of the font. One em is the height of the font. By using em's you can preserve the general look of the Web page independently of the font size. This is much safer than alternatives such as pixels or points, which can cause problems for users who need large fonts to read the text.

- Points are commonly used in word processing packages, e.g. 10pt text. Unfortunately the same point size is rendered differently on different browsers. What works fine for one browser will be illegible on another! Sticking with em's avoids these problems.


Border
Margin
Once upon a time there was a young lady ...
Content
Padding
The end.

General page layout

- There exist specific style attribute for page layout:
  - margin-top
  - margin-left
  - margin-right
  - margin-bottom
  - margin (for all pages)

Setting the page margins

- Web pages look a lot nicer with bigger margins. You can set the left and right margins with the "margin-left" and "margin-right„ properties, e.g.

```
<style type="text/css">
  body { margin-left: 10%; margin-right:
  10%; }
</style>
```

- This sets both margins to 10% of the window width, and the margins will scale when you resize the browser window.

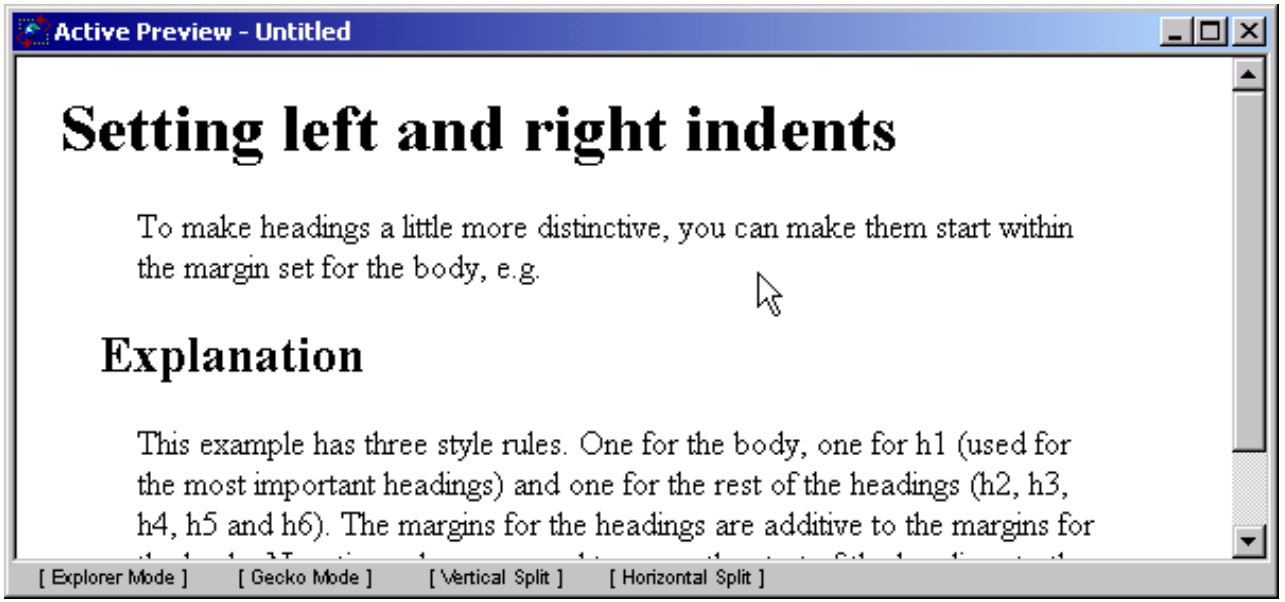
Setting left and right indents

- To make headings a little more distinctive, you can make them start within the margin set for the body, e.g.

```
body { margin-left: 10%; margin-right:
  10%; }
h1 { margin-left: -8%;}
h2,h3,h4,h5,h6 { margin-left: -4%; }
```

- This example has three style rules. One for the body, one for h1 (used for the most important headings) and one for the rest of the
- headings (h2, h3, h4, h5 and h6).
- The margins for the headings are additive to the margins for the body. Negative values are used to move the start of the headings to the left of the margin set for the body.

Setting left and right indents: Example



Controlling the white space above and below

- Browsers do a pretty good job for the white space above and below headings and paragraphs etc. Two reasons for taking control of this yourself are: when you want a lot of white space before a particular heading or paragraph, or when you need precise control for the general spacings.

- The "margin-top" property specifies the space above and the "margin-below" specifies the space below. To set these for all h2 headings you can write:

```
h2 { margin-top: 8em; margin-bottom: 3em; }
```

- When a heading is followed by a paragraph, the value for margin-bottom for the heading isn't added to the value for margin-top for the paragraph. Instead, the maximum of the two values is used for the spacing between the heading and paragraph. This subtlety applies to margin-top and margin-bottom regardless of which tags are involved!

Controlling the white space II

- To specify the space above only particular heading, you should create a named style for the heading.

```
h2.subsection { margin-top: 8em; margin-
bottom: 3em; }
```

- The rule starts with the tag name, a dot and then the value of the class attribute. Be careful to avoid placing a space before or after the dot. There are other ways to set the styles for a particular element but the class attribute is the most flexible.

- You use this with the class attribute in the markup, e.g.

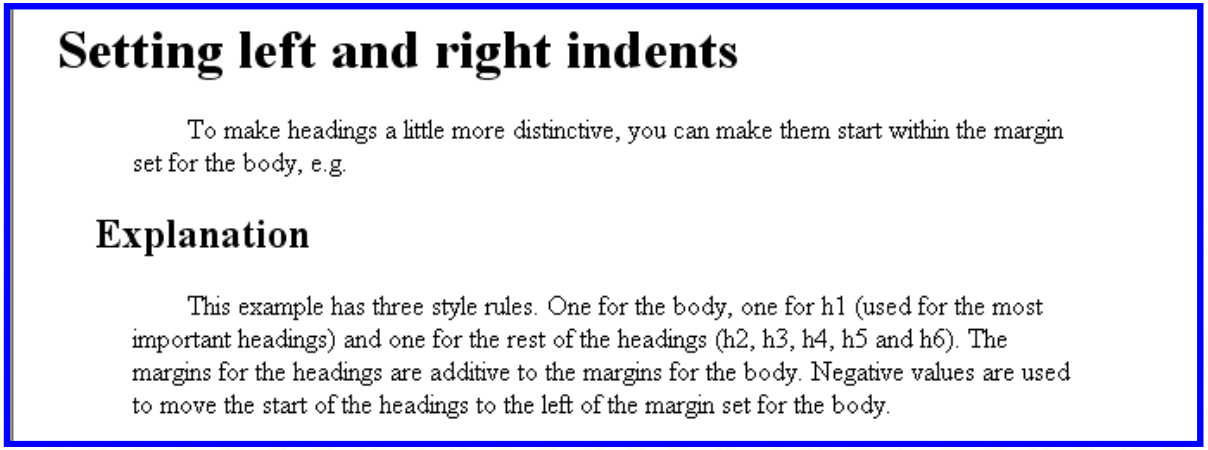
```
<h2 class="subsection">Getting started</h2>
```

First-line indent

- Sometimes you may want to indent the first line of each paragraph. The following style rule emulates the traditional way paragraphs are rendered in novels:

```
p {text-indent: 2em; margin-top: 0;
  margin-bottom: 0;}
```

- It indents the first line of each paragraph by 2 em's and suppresses the inter-paragraph spacing.



Setting left and right indents

To make headings a little more distinctive, you can make them start within the margin set for the body, e.g.

Explanation

This example has three style rules. One for the body, one for h1 (used for the most important headings) and one for the rest of the headings (h2, h3, h4, h5 and h6). The margins for the headings are additive to the margins for the body. Negative values are used to move the start of the headings to the left of the margin set for the body.

Controlling the font

- The Font properties allow you to change the font family, boldness, size, and the style of a text.
  - Font styles
  - Setting the font size
  - Setting the font family
  - Avoid problems with fonts and margins

Font styles

- font-family: A prioritized list of font family names and/or generic family names for an element.
- At the end of the enumeration you should use always one of the following generic font names:serif, sans-serif, cursive, fantasy, monospace.

```
body { font-family: Verdana, sans-serif; }
h1,h2 { font-family: Garamond, "Times New Roman", serif; }
```

- The legibility of different fonts generally depends more on the height of lower case letters than on the font size itself. Fonts like Verdana are much more legible than ones like "Times New Roman" and are therefore recommended for paragraph text.

Font styles II

- The font-style property sets the style of a font.

```
font-style:  normal | italic | oblique
```

- The font-weight property sets how thick or thin characters in text should be displayed.
- font-weight:
  - bold = thick characters.
  - bolder = thicker characters .
  - lighter = lighter characters.
  - 100 (extra-light) up to 900 (extra-thick).
  - normal = normal characters.

Font formatting with a formatting file

- You define the font like defining the font-family property..
- But you specify directly the resource (URL) of the font.

```
...<title></title>
  <style type="text/css">
   @font-face { font-family:Garamond;
   src:url(http://www.xyz.de/fonts/garamond.eot),
   url(http://www.xyz.de/fonts/ garamond.pfr); }
  </style>
  </head>
  <body>
  <p style="font-family:Garamond">Text in Garamond</p>
...
```

Font formatting with a formatting file

- The most common styles are to place text in italic or bold. Most browsers render the em tag in italic and the strong tag in bold.
- Let's assume you instead want em to appear in bold italic and strong in bold uppercase:

```
em { font-style: italic; font-
  weight: bold; }
strong { text-transform: uppercase;
  font-weight: bold; }
```

- If you feel so inclined, you can fold headings to lower case as follows:

```
h2 { text-transform: lowercase; }
```

Setting the font size

- Most browsers use a larger font size for more important headings. If you override the default size, you run the risk of making the text too small to be legible, particularly if you use points.
- You are therefore recommended to specify font sizes in relative terms.

- This example sets heading sizes in percentages relative to the size used for normal text:

```
h1 { font-size: 200%; }
h2 { font-size: 150%; }
h3 { font-size: 100%; }
```

Style properties Text II

- The font-size property sets the size of a font:

```
font-size:  <size in em, pt, px ,ex oder %> |
larger | smaller | xx-small | x-small | small |
medium | large |  x-large | xx-large
```

CSS Text properties

- Text properties allow you to control the appearance of text. It is possible to change the color of a text, increase or decrease the space between characters in a text, align a text, decorate a text, indent the first line in a text, and more.
  - text-decoration: none | [underline || overline || line-through || blink]
  - text-indent: <bigger in em, pt, px ,ex, cm, mm or %>
  - text-shadow: none | [<color> || <length> <length> <length>? ,]* [<color> || <length> <length> <length>?]
  - text-transform: capitalize | uppercase | lowercase | none

The text-align property

- The text-align property aligns the text in an element.
- It can only be applied to certain elements, called *block-level elements*. Normally you'll apply it to a paragraph.

```
p { text-align: center}
```

- Value Description

left	Aligns the text to the left
right	Aligns the text to the right
center	Centers the text
justify	

The text color property

- The color property sets the color of a text.

```
body { color: rgb(255,255,0) }
```

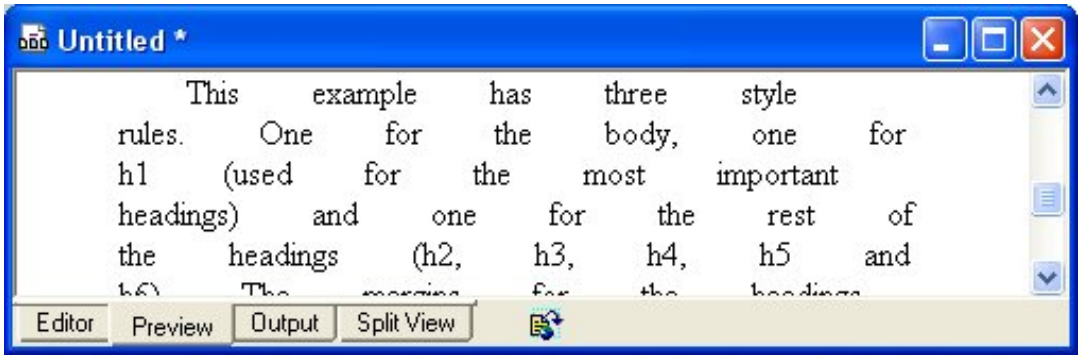
- The color value can be
  - a color name: `red`
  - a rgb value: `rgb(255,0,0)`
  - a percentage rgb value: `rgb(80%,80%,80%)`
  - or a hex number: `#ff0000`

Red, green, blue

The word-spacing property

- The word-spacing property increases or decreases the white space between words.
- Negative values are allowed.

  p{ word-spacing: 30px}
  p{word-spacing: -0.5px}



The letter-spacing property

- The letter-spacing property increases or decreases the white space between characters.
- A positive value makes letters farther apart, a negative value makes letters close together or overlapping.

  letterspacing : normal | <size>

- Negative values are allowed.

```
p{letter-spacing: -0.5px;}
```

The text direction property

- The direction property sets the text direction.

- ltr Text direction is left-to-right
- rtl Text direction is right-to-left

```
div{direction: rtl}
```

line-height

- With the **line-height** property you can control the spacing in between lines of text.
- You can use, for example, to make your text double spaced (as might want to format an essay):

```
BODY { line-height: 1.2 }
```

Horizontal Rules

```
{
.rule {border-top-width: 1px;
      border-top-style: solid;
       border-color: #FF0000;
       margin: 0px 2%;
}
```

Recommendations of Dave Raggett (W3C staff)

Use <p>!

- My first rule is to avoid text at the body level that isn't wrapped in a block level element such as p. For instance:

```
<h1>Setting left and right indents</h1>
To make headings a little more distinctive,
you can make them start within the margin set
for the body.
```

- The text following the heading runs the risk on some browsers of being rendered with the wrong font and margins. You are therefore advised to generally enclose all such text in a paragraph, e.g.

-
```
<h1>Setting left and right indents</h1>
<p>To make headings a little more
distinctive, you can make them start within
the margin set for the body.</p>
```

Recommendations of Dave Raggett II

- My second rule is to set the font family for pre elements, as some browsers forget to use a fixed pitch font when you set the font size or other properties for pre.

```
pre { font-family: monospace; }
```

- In the same way, you can define properties for the <bold> tag:

```
bold { font-weight: 700; }
```

Recommendations of Dave Raggett III

- My third rule is to set the font family on headings, p and ul elements if you intend to set borders or backgrounds on elements such as div.
- This is a work-around for a bug where the browser forgets to use the inherited font family, instead switching to the default font as set by the browser preferences.

```
h1,h2,h3,h4,h5,p,ul { font-family: sans-
  serif; }
```

Table properties

```
td.left   {color:#0000FF; font-family:Arial,sans-serif; border-
   width:1pt;
    border-style:solid; border-color:#999999; background-
   color:#FFFFE0; }
td.right  {color:#0000FF; font-family:Arial,sans-serif; border-
   width:1pt;
    border-style:solid; border-color:#999999; background-
   color:#FFFFE0; }
```

External Stylesheets integration

- The LINK tag should be placed in the document's head. The rel attribute must be set to the value "stylesheet" to allow the browser to recognize that the href attribute gives the Web address (URL) for your style sheet:

```
<link rel=StyleSheet type="text/css"
href="externalsheet.css" />
```

```
<style>
 <!--
   @import url(externalsheet.css);
   #margin { margin-left: 10%; margin-right: 10%}
 -->
 </style>
```

Style for Links

- There are predefined pseudo-elements:

```
a:link { color:BLUE; text-decoration:underline; }
a:visited { color:#772200; text-decoration:underline; }
a:active { color:#000000; text-decoration:none; }
```

Comments in style sheet

```
<style type="text/css">
  <!--
    p { color:blue; } /* blue text */
  //-->
  </style>
```

Principles of Positioning in HMTL

- In the visual formatting model, each element in the document tree generates zero or more boxes according to the box model. The layout of these boxes is governed by:
  - box dimensions and type.
  - positioning scheme (normal flow, float, and absolute).
  - relationships between elements in the document tree.
  - external information (e.g., viewport size, intrinsic dimensions of images, etc.).

http://www.w3.org/TR/REC-CSS2/visuren.html#box-gen

Block-level, Floating and Inline

- There are three different types of elements, and each will be formatted differently.
  - A *block-level* element cannot share space on a line with other elements. An example is a paragraph, or a list.
  - A *floating* element is positioned outside of the normal flow, to the left or right of the parent element.
  - An *inline* element can share space on a line (like words or an image).

Positioning Schemes

- Normal Flow
- Relative Positioning
- Float Positioning
- Absolute Positioning
- Fixed Positioning

Normal Flow

- Positioning is set according to the 'normal' rules that have been around for nearly a decade.

- http://www.w3.org/TR/REC-CSS2/visuren.html#normal-flow

Relative Positioning

- Relatively positioned elements are positioned according to the normal flow and then moved.
- Elements that come after a relatively-positioned element behave as if the relatively-positioned element was in its 'normal flow' position, even if this means they occupy the same screen space.
- http://www.w3.org/TR/REC-CSS2/visuren.html#relative-positioning

Float Positioning

- Boxes positioned using float are positioned using normal flow and then moved left or right as far as they can go.
- Elements that appear after them will move up to fill any gap left behind, but will flow around the box -- they will not occupy the same screen space.

- http://www.w3.org/TR/REC-CSS2/visuren.html#floats

Absolute Positioning

- Absolutely-positioned elements are not affected by normal flow. They are positioned by specifying precise distances that will exist between their sides and their containing block.

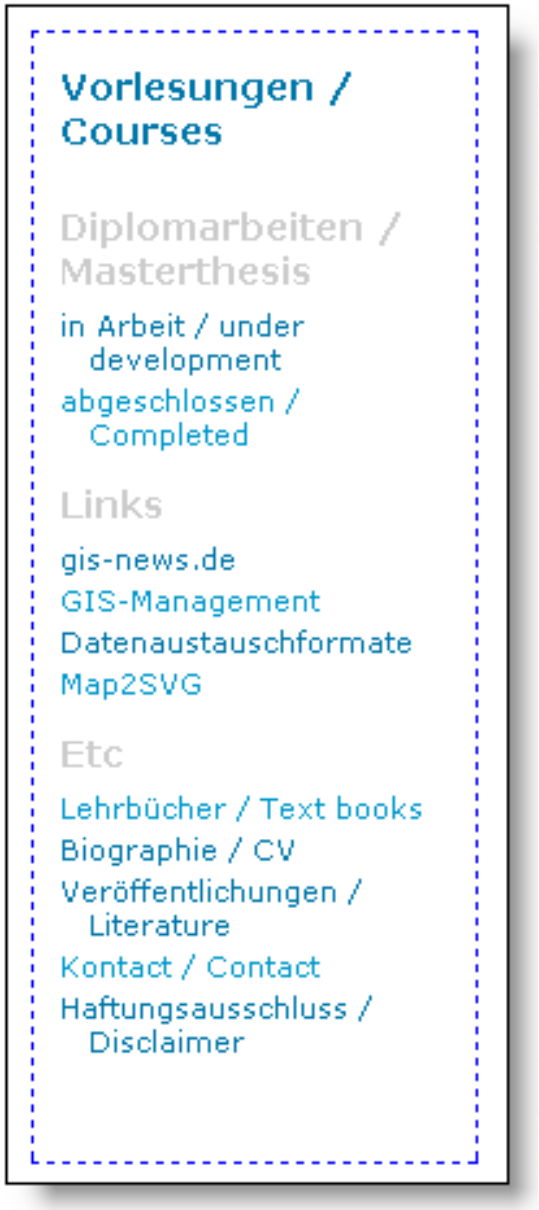
- http://www.w3.org/TR/REC-CSS2/visuren.html#absolute-positioning

Fixed Positioning

- Fixed-position elements work in the same way as absolutely-positioned elements, except their position is relative to the Viewport. As such, they don't move if the page is scrolled -- they stay relative to the Viewport.

- http://www.w3.org/TR/REC-CSS2/visuren.html#fixed-positioning

```
#Menu {
        position: fixed;
        width:150px;
        top:70px; /*20px;*/
        right:10%;/*520px;*/
        width:150px;
        margin-top:38px;
        border-right: #ffffff 3px solid;
         padding-right: 10px;
        padding:0 10px 40px;
        border-color:blue;
        border-width:1px 1px 1px 1px;
        }
```



- Block elements (<h1>, <p>, etc.) are positioned according to their sequence in the HTML page..
- Inline elements (, <img>, etc.) are positioned according to their sequence in the surrounding element.
  If they can‘t be placed into one line the browser can insert a line break.

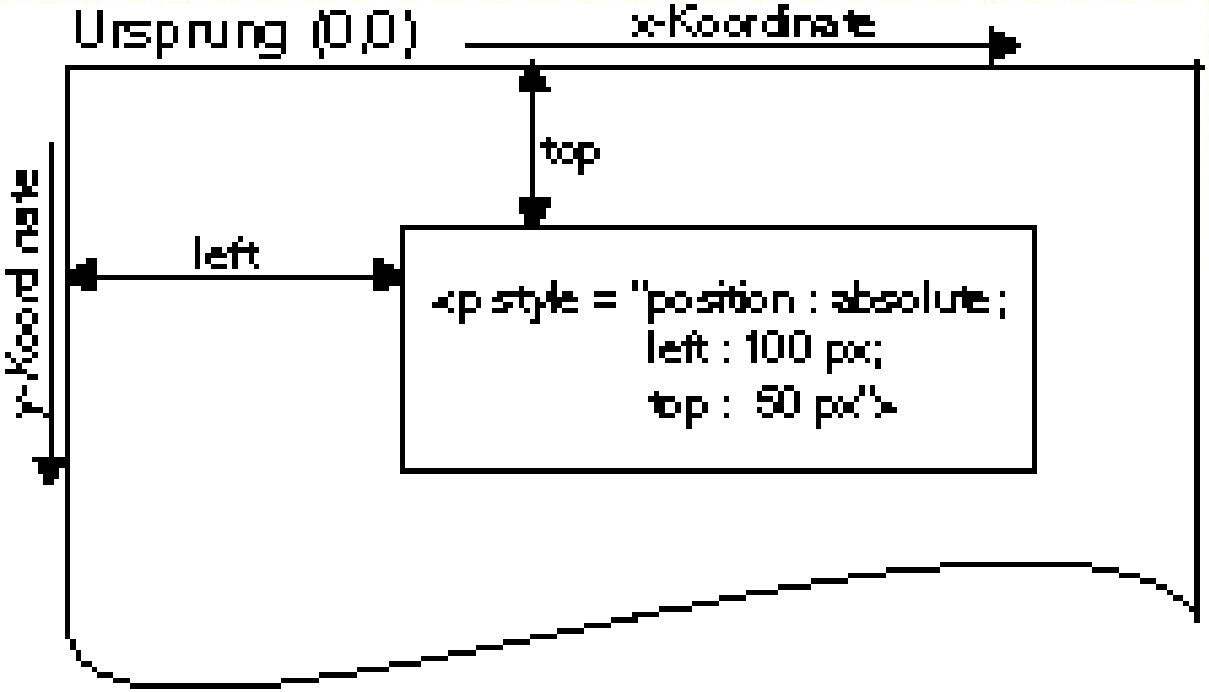
Choosing a positioning scheme: 'position' property

- **Static**: The box is a normal box, laid out according to the normal flow. The 'left' and 'top' properties do not apply.
- **Relative:** The box's position is calculated according to the normal flow (this is called the position in normal flow). Then the box is offset relative to its normal position. When a box B is relatively positioned, the position of the following box is calculated as though B were not offset.
- **Absolute**: The box's position (and possibly size) is specified with the 'left', 'right', 'top', and 'bottom' properties. These properties specify offsets with respect to the box's containing block. Absolutely positioned boxes are taken out of the normal flow. This means they have no impact on the layout of later siblings. Also, though absolutely positioned boxes have margins, they do not collapse with any other margins.

Choosing a positioning scheme: 'position' property

- **Fixed**: The box's position is calculated according to the 'absolute' model, but in addition, the box is fixed with respect to some reference. In the case of continuous media, the box is fixed with respect to the viewport (and doesn't move when scrolled). In the case of paged media, the box is fixed with respect to the page, even if that page is seen through a viewport (in the case of a print-preview, for example). Authors may wish to specify 'fixed' in a media-dependent way. For instance, an author may want a box to remain at the top of the viewport on the screen, but not at the top of each printed page. The two specifications may be separated by using an @media rule, as in:
  - @media screen { H1#first { position: fixed } } @media print { H1#first { position: static } }

CSS-Positioning: absolute




```
<img style="position: absolute; top: 20px; left: 10px" src=... />
```

Positioning Context I

```
...
<body>
<style type="text/css">
 #aussen {position: absolute;  top: 4cm; left: 3cm; width: 5cm; color: red}
 #innen {position: absolute; top: 4cm; left: 3cm; width: 5cm; font-weight:
bold}
</style>
<body>
<p>Jede Webseite verfügt über einen Standardpositionierungskontext, der an
das
<html>-Tag gebunden ist. Seine Koordinaten und Abmaße werden vom Browser
definiert
und können nicht verändert werden.

Sofern man von der CSS-Positionierung Gebrauch macht, gilt:
Wird aber für ein HTML-Element die Stileigenschaft position auf absolute oder
relative gesetzt, bildet dieses Element einen eigenen
Positionierungskontext, <b>der für alle eingebetteten Elemente gilt</b>.
<span id="aussen">Beginn der
 äußeren Textpassage. <span id="innen">Innerer Text.</span> Ende der äußeren
 Textpassage. </span></p>
</body>
</html>
```

Positioning Context II

Jede Webseite verfügt über einen Standardpositionierungskontext, der an das -Tag gebunden ist. Seine Koordinaten und Abmaße werden vom Browser definiert und können nicht verändert werden. Sofern man von der CSS-Positionierung Gebrauch macht, gilt: Wird aber für ein HTML-Element die Stileigenschaft position auf absolute oder relative gesetzt, bildet dieses Element einen eigenen Positionierungskontext, **der für alle eingebetteten Elemente gilt**.

Beginn der äußeren Textpassage. Ende der äußeren Textpassage.

Innerer Text.