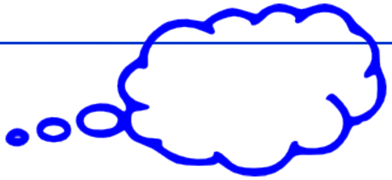
CSS

Cascading Style Sheets

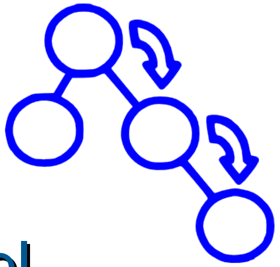
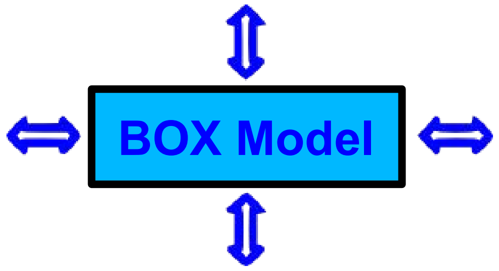
Lukáš Bařinka
barinkl@fel.cvut.cz

department of computer science and engineering

Content

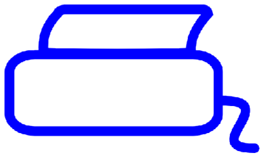
- Basic principles 
- Syntax

div#headder h1.title a:hover { color: blue; }

- Selectors
- Inheritance 
- Formating (box) model 

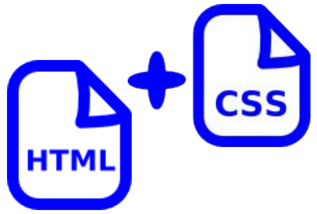
- Generated content

Figure 7.3:

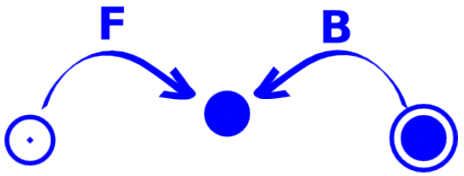
- Media, UI 
- Miscellaneous

Basic principles I

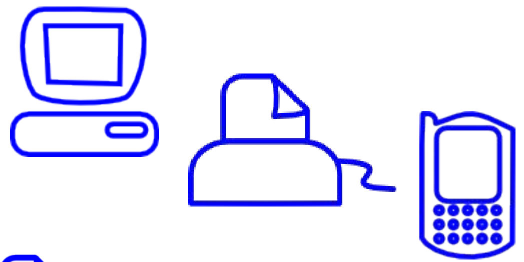
- Extension for structured document (enrichment)
  ! separation of structure and presentation !



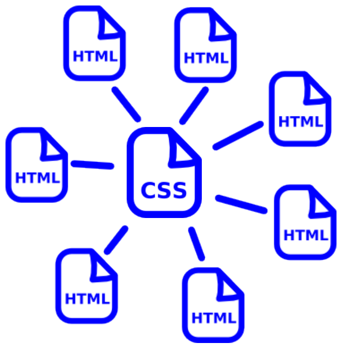
- Forward and backward compatibility



- Platform and device independency



- Maintainability (shareability)



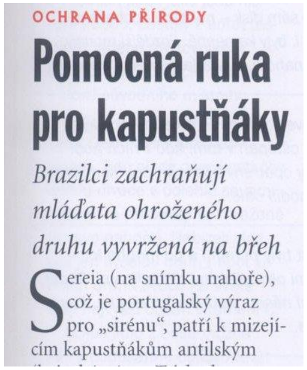
- Simplicity **color: blue**

Basic principles II

- Net-traffic saving

```
<td align="left">
<td align="left">   } ⇨ td { text-align: left; }
<td align="left">                ⇩
                     td { text-align: center; }
```

- Flexibility



OCHRANA PŘÍRODY

Pomocná ruka pro kapustňáky

Brazilci zachraňují mláďata ohroženého druhu vyvržená na břeh

Sereia (na snímku nahoře), což je portugalský výraz pro „sirénu“, patří k mizejícím kapustňákům antilským

- Language richness

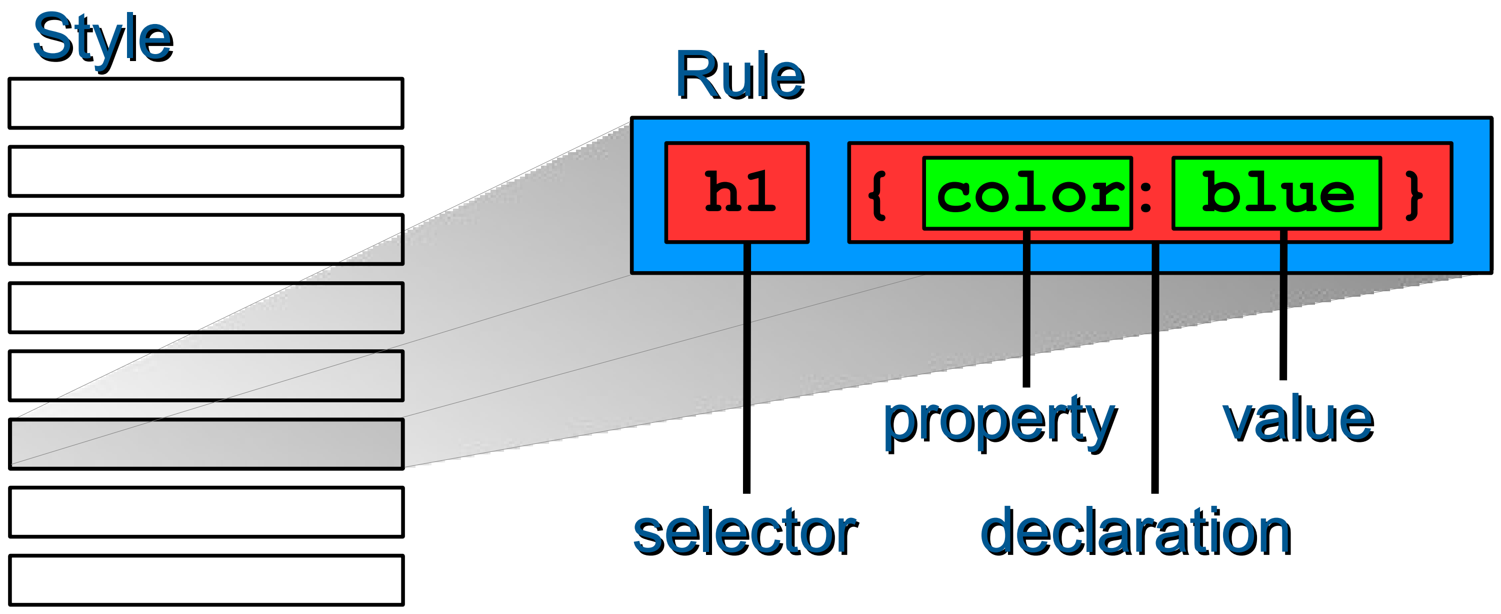
- Cooperation with others languages

JavaScript → CSS

- Accessibility

Syntax I

- The style is a set of rules
- These rules specify look and feel of the document



Syntax II – Integration CSS into HTML a

plain HTML

```
<html>
   <head>
      <title>title</title>
   </head>
   <body>
      <h1>headline is blue</h1>
      <p>while the paragraph is green.
      <p>this paragraph is yellow
      <p>while the paragraph is green.
   </body>
</html>
```

department of computer science and engineering

Syntax II – Integration CSS into HTML

linked CSS

```
<html>
   <head>
      <title>title</title>
      <link rel="stylesheet" type="text/css"
            href="http://style.com/cool" title="cool">
   </head>
   <body>
      <h1>headline is blue</h1>
      <p>while the paragraph is green.
      <p class="motto"> this paragraph is yellow
      <p id="mypar">while the paragraph is green.
   </body>
</html>
```

Syntax II – Integration CSS into HTML c

imported CSS

```
<html>
   <head>
      <title>title</title>
      <style type="text/css">
         @import url("http://style.com/basic");
      </style>
   </head>
   <body>
      <h1>headline is blue</h1>
      <p>while the paragraph is green.
      <p class="motto"> this paragraph is yellow
      <p id="mypar">while the paragraph is green.
   </body>
</html>
```

Syntax II – Integration CSS into HTML d

included CSS

```
<html>
   <head>
      <title>title</title>
      <style type="text/css">
         h1 { color: blue }
         p.motto  { color: yellow }
         #mypar { color: green }
      </style>
   </head>
   <body>
      <h1>headline is blue</h1>
      <p>while the paragraph is green.
      <p class="motto"> this paragraph is yellow
      <p id="mypar">while the paragraph is green.
   </body>
</html>
```

department of computer science and engineering

Syntax II – Integration CSS into HTML e

inlined CSS

```
<html>
   <head>
      <title>title</title>
      <style type="text/css">
         h1 { color: blue }
         p.motto  { color: yellow }
         #mypar { color: green }
      </style>
   </head>
   <body>
      <h1>headline is blue</h1>
      <p style="color: green">while the paragraph is green.
      <p class="motto"> this paragraph is yellow
      <p id="mypar">while the paragraph is green.
   </body>
</html>
```

Syntaxe II – Overview

```
<html>
   <head>
      <title>title</title>
      <link rel="stylesheet" type="text/css"
            href="http://style.com/cool" title="cool">
      <style type="text/css">
         @import url("http://style.com/basic");
         h1 { color: blue }
         p.motto { color: yellow }
         #colg { color: green }
      </style>
   </head>
   <body>
      <h1>headline is blue</h1>
      <p style="color: green">while the paragraph is green.
      <p class="motto"> this paragraph is yellow
      <p id="colg">while the paragraph is green.
   </body>
</html>
```

Syntax III - Features

- Case insensitive (excluding parts out of CSS – e.g. selector)
- Key-word (`red`) / String (`"red"`)
- Block enclosed by "`{`" "`}`"
- Rules in block separated by ";"
- Grouping rules

```
h1 { color: red }
h2 { color: red }
h1, h2 { color: red }
```

```
h1 { font-size: 12pt }
h1 { font-weight: bold }
h1 { font-family: "Helvetica" }
h1 { font: bold 12pt "Helvetica" }
```

- Comments inside "`/*`" and "`*/`"
  (HTML comment tags "`<!--`", "`-->`" allowed, but do not delimit CSS comment)
- Encoding in CSS: `@charset "ISO-8859-2"`

Syntax IV – Values 1

- Lengths
  - RELATIVE
    - em – font size
    - ex – height of letter “x“
    - %
  - ABSOLUTE
    - px – pixel element
    - in
    - cm
    - mm
    - pt – point (1/72 in)
    - pc – 12pt (pica)

Child elements do not inherit the relative values specified for their parent; they inherit the computed values !

Syntaxe V – Values 2

- URI
  - RELATIVE
    - `url(˝bg.jpg˝)`
  - ABSOLUTE
    - `url(˝http://faraon.felk.cvut.cz/img/bg.jpg˝)`
- Colors

– `red`	[key-word]
– `#f00`	[#rgb]
– `#ff0000`	[#rrggbb]
– `rgb(255,0,0)`	
– `rgb(100%,0%,0%)`	

- Background / Image
  - `url(˝bg.jpg˝)`
- Text
  - `"Helvetica"`
  - `serif`
  - `sans-serif`
  - `cursive`
  - `fantasy`
  - `monospace`

Selectors

*	Universal selector
E	Type selector
E F	Descendant selector
E > F	Child selector (direct descendant)
E + F	Adjacent selector (sibling)
E:first-child	First child pseudo-class
E:link	Link pseudo-class
E:hover	Dynamic pseudo-class
E:lang(x)	Language pseudo-class
E[atr="val"]	Atribute selector (= ~= |=)
E.class	Class selector
E#id	ID selector

Pseudo-classes

- Child
  - `:first-child`
- Link (anchor)
  - `:link`
  - `:visited`
- Dynamic
  - `:hover`
  - `:active`
  - `:focus`
- Language
  - `:lang`

```
a {
   color: blue;
   text-decoration: none;
}

a:hover {
   color: red;
   border: 1px dotted black;
}
```

department of computer science and engineering

Pseudo-elements

- **:first-line**

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

- **:first-letter**

```
p:first-letter { text-transform: uppercase;
                  font-size: 200%;
                  font-style: italic;
                  font-weight: bold;
                  float: left;
                  padding-right: 0.2em;
                }
```

- **:before**

```
h1:before { content: "Special! \""; }
```

- **:after**

```
h1:after { content: '"'; }
```

Inheritance I

- Value is derived from:
  - Value as result of cascade (rules)
  - Else inherited value (from parent)
  - Else initial value
- Some values are inherited, some values don't
  - `body { color: black;` value inherits
    `background: white;` value does not inherit
    `}`
  - `h1 { font-height: 120% }` 120% of parent value
- Styles can be defined by
  - Author
  - User
  - UA (user agent) – e.g. web browser

Inheritance II

To find cascade result style is necessary to follow order:

1. Find all declarations that apply to the element and property
   (for the target media type)
2. Sort by importance and origin
   UA < user < author < author !important < user !important
3. Sort by specificity of selector
   more specific selectors precede more general ones
4. Sort by order specified
   latter specified is used

To affect importance of rule can be used **`!important`**

```
p { font: normal 12pt sans-serif !important }
```

Inheritance - Computation

a – style atribute

b – ID atribute count

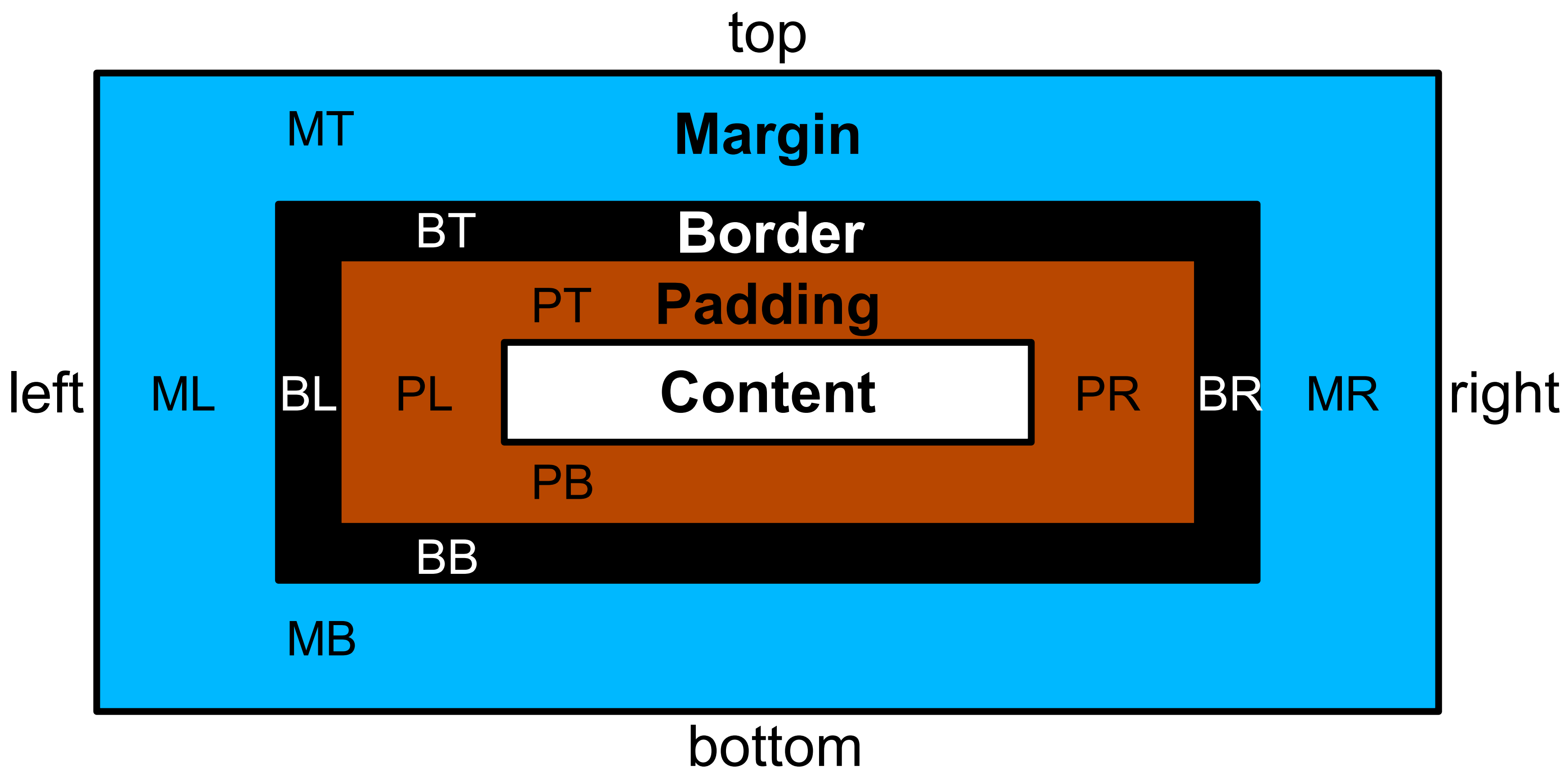
c – others atribute and pseudo-class count

d – pseudo-/element's name count

```
*                 { … }   /* a=0 b=0 c=0 d=0 -> specificity = 0,0,0,0 */
 li               { … }   /* a=0 b=0 c=0 d=1 -> specificity = 0,0,0,1 */
 li:first-line    { … }   /* a=0 b=0 c=0 d=2 -> specificity = 0,0,0,2 */
 ul li            { … }   /* a=0 b=0 c=0 d=2 -> specificity = 0,0,0,2 */
 ul ol+li         { … }   /* a=0 b=0 c=0 d=3 -> specificity = 0,0,0,3 */
 h1 + *[rel=up]   { … }   /* a=0 b=0 c=1 d=1 -> specificity = 0,0,1,1 */
 ul ol li.red     { … }   /* a=0 b=0 c=1 d=3 -> specificity = 0,0,1,3 */
 li.red.level     { … }   /* a=0 b=0 c=2 d=1 -> specificity = 0,0,2,1 */
 #x34y            { … }   /* a=0 b=1 c=0 d=0 -> specificity = 0,1,0,0 */
 style=" … "      { … }   /* a=1 b=0 c=0 d=0 -> specificity = 1,0,0,0 */
```

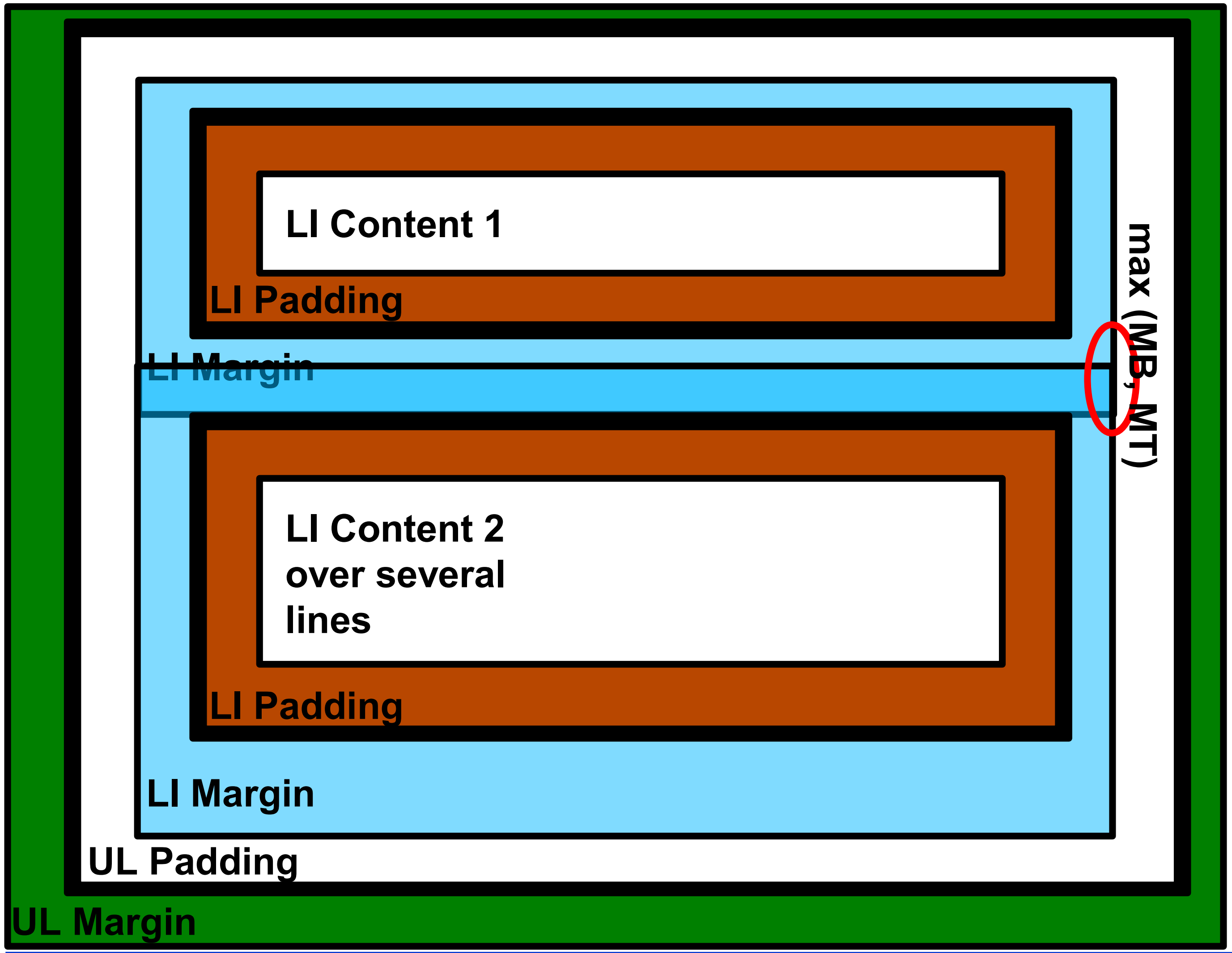
department of computer science and engineering

Formating model – Box model



- Background below content and padding (up to border)
- Margins are always transparent

Formating model II



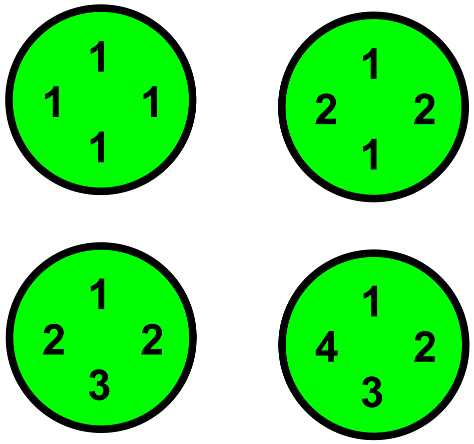
```
<ul>
   <li>
      LI 1
   </li>
   <li>
      LI 2
      over...
   </li>
</ul>
```

Borders

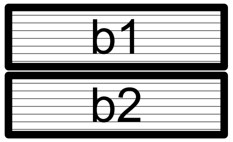
- **border-width**
  - <length>
  - thin
  - medium
  - thick
- **border-color**
  - <color>
  - transparent
- **border-style**
  - none
  - dotted
  - double
  - ridge
  - outset
  - hidden
  - solid
  - groove
  - inset

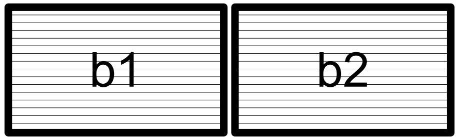
```
h1 { border: 0.5em solid silver }
```

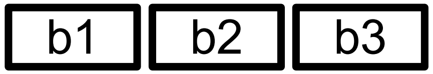
```
                         1      2     3      4
h1 { border-width: thin }
h1 { border-width: thin thick}
h1 { border-width: thin thick medium }
h1 { border-width: thin thick medium 5px}
```

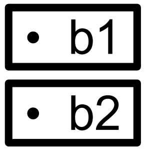


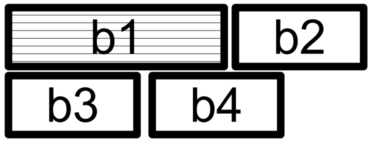
Visual formating I

- property **display**
  - **block**
    

  - **inline-block**
    

  - **inline**
    

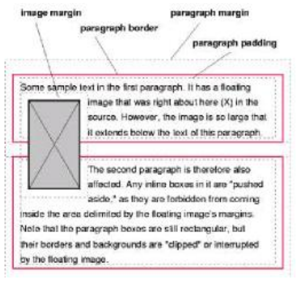
  - **list-item**
    

  - **none**
  - **run-in**
    

  - **table**
  - **inline-table**
  - **table-column**
  - **table-caption**
  - **table-row**
  - **table-cell ...**

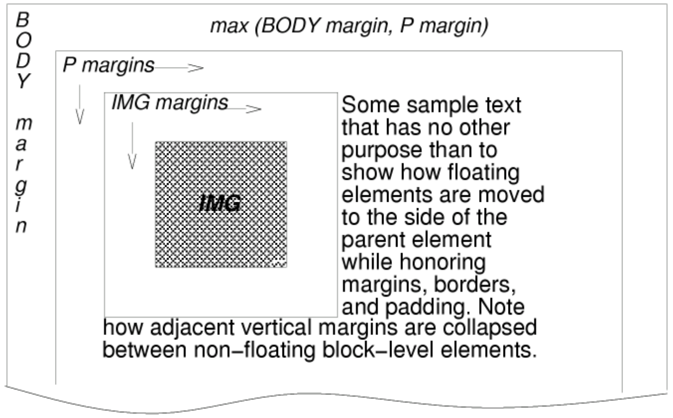
Visual formating II

- property `position`
  - `static`
  - `relative`
  - `absolute`
  - `fixed`
- offset (except static)
  - `top`
  - `right`
  - `bottom`
  - `left`
- float
  - `left`
  - `right`
  - `none`



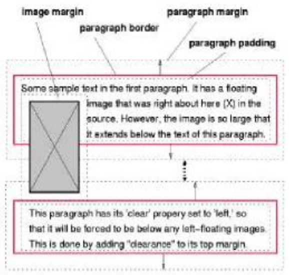
Some sample text in the first paragraph. It has a floating image that was right about here (X) in the source. However, the image is so large that it extends below the text of this paragraph.

The second paragraph is therefore also affected. Any inline boxes in it are "pushed aside," as they are forbidden from coming inside the area delimited by the floating image's margins. Note that the paragraph boxes are still rectangular, but their borders and backgrounds are "clipped" or interrupted by the floating image.



Some sample text that has no other purpose than to show how floating elements are moved to the side of the parent element while honoring margins, borders, and padding. Note how adjacent vertical margins are collapsed between non–floating block–level elements.

- property `clear`
  - `left`
  - `right`
  - `both`
  - `none`



Some sample text in the first paragraph. It has a floating image that was right about here (X) in the source. However, the image is so large that it extends below the text of this paragraph.

This paragraph has its 'clear' propery set to 'left,' so that it will be forced to be below any left–floating images. This is done by adding "clearance" to its top margin.

- level `z-index`
  - `auto`
  - `<level>`
- text direction `direction`
  - `ltr` (left to right)
  - `rtl` (right to left)

Visual formating III

- `width, min-width, max-width`
- `height, min-height, max-height`
- `line-height, text-indent, text-align`
- `text-decoration, letter-spacing, word-spacing`
- `vertical-align (baseline, middle, sub, super, text-top, text-bottom, top, bottom)`
- `overflow (visible, hidden, scroll, auto)`
- `clip (auto, rectl(T,R,L,B))`
- `visibility (visible, hidden, collapse)`

Content generating

- Pseudo-element usage `:before` and `:after`
- Property `content`
  - `normal`
  - `<string>`
  - `<uri>`
  - `counter`
  - `open-quote, close-quote`
  - `no-open-quote, no-close-quote`
  - `attr(X)`
- Property `quotes`
  - `none`
  - `[ <string1> <string2> ]+`

Content generating II – Counters and lists

- Property `counter-reset`
- Property `counter-increment`
- It is possible to nest counters (e.g. lists) – `counters`
- Usage:
  - `counter(name)`
  - `counter(name, style)`
    - style: (disc, circle, square, none, upper-latin, upper-roman, hebrew)
- Property `list-style-type`
    - (disc, circle, square, decimal, decimal-leading-zero, lower/upper-roman, georgian, armenian, lower/upper-latin, lower/upper-alpha, lower-greek)
- Property `list-style-image`
- Property `list-style-position`
    - (outside, inside)

Media

- CSS are platform/device independent, but it is possible to define different styles for different media
- Definition using block **`@media`** or **`@import`**
  - **`@import url(˝screen.css˝) screen;`**
  - **`@media print {   ...   }`**
    - all, braile, embossed, handheld, print, projection, screen, speech, tty, tv
- Paged media definition using **`@page`**
  - **`@page:left, @page:right, @page:first`**
- Page break using
  - **`page-break-before/after/inside`**
    - auto, always, avoid, left, right
  - **`orphans`**
  - **`widows`**

    minimum number of lines of a paragraph that must be left at the bottom/top

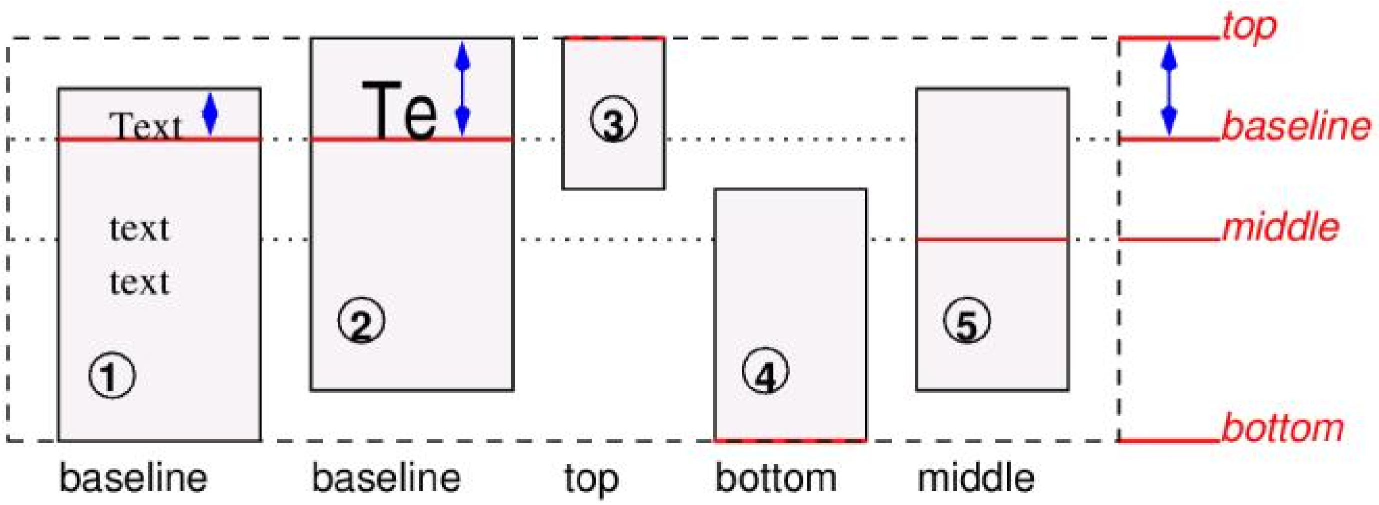
User Interface

- Cursor specification using **`cursor`**
  - **auto, crosshair, default, pointer, move, e/ne/nw/n/...-resize, text, wait, progress, help, <url>**

- Reusage of user interface colors
  - **reference to UI color by name (e.g. ButtonFace, ButtonText, Menu, Scrollbar, Window, …)**

Miscellaneus – Tables

- Tables
  - `table-layout` (auto, fixed)
  - `border-collapse` (collapse, separate)
  - `border-spacing` (<length> <length>)
  - `empty-cells` (show, hide)
  - `vertical-align` (baseline, top, bottom, middle, sub, super, text-top, text-bottom, <length>, %)



Miscellaneus II

- Property `white-space`
  - normal, pre, nowrap, pre-wrap, pre-line
- Font specification
  - `font-family`
  - `font-style` (normal, italic, oblique)
  - `font-variant` (normal, small-caps)
  - `font-weight` (100-900)
  - `font-size`
- Text specification
  - `text-indent`
  - `text-align` (left, right, center, justify)
  - `text-decoration` (none, underline, overline, line-through, blink)
  - `text-transform` (capitalize, uppercase, lowercase, none)

Thank you for your attention

Lukáš Bařinka

http://webing.felk.cvut.cz

department of computer science and engineering