Web-based Applications

Lecture 3

Cascading Style Sheets

Bartosz Walter

<Bartek.Walter@man.poznan.pl>

Catalog of Design Patterns

Foundations of CSS

- **1992 – Time Berners-Lee defines a basic set of tags defining the document structure**
- **1996 – introduction of CSS 1.0**
- **Next versions: CSS 2.0 (IX 1998), CSS 2.1 (I 2001)**
- **Goals:**
  - more sophisticated page design
  - separating the content from layout
  - accessibility/multiple interpreting devices

What is CSS?

- **text file defining styling rules**
- **no header**
- **.css extension**

```
body {text-align: justify}

.note {color: white; background-color: grey}
```

CSS standards

Browser	CSS1	CSS2
Microsoft Internet Explorer	acceptable	good
Netscape 4.x	partial	none
Netscape 7.x/Mozilla 1.x	extensive	good
Opera	extensive	good

according to

How CSS work?

- **series of statements**
- **statement**
  - **selector:** an HTML element to select
  - **declaration:** how to draw the element
  - **property:** a specific rendering instruction

```
body                {text-align: justify}
```

by the Gang of Four

CSS syntax

- **statement = selector { declaration }**
- **declaration = property1; property2; property3**
- **property = name: value;**
-

```
body {
   text-align: justify;
   font-family: Verdana;
   font-size: 14px
}
```

Embedding CSS into HTML

Style-sheet is directly embedded within HEAD element

```
<head>
    <style>
    /* css style */
    </style>
</head>
```

by the Gang of Four

Linking CSS with HTML

Style-sheet is linked to a document in HEAD element

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

Statements

@import

the browser should import an external stylesheet and use it together with the present one

CSS2 allows for conditional imports of stylesheets specific

for different media

```
@import
   url(http://www.cs.put.poznan.pl/res/style1.css);
@import
   "http://www.cs.put.poznan.pl/res/style1.css";
```

Statements (cont.)

@media

the browser should apply the stylesheet only if specific output medium is used

media types: all, aural, braille, handheld, print, projection, screen, tty, tv

```
@media print {
   body {background-color: white;}
}

@import url(monochrome.css) print, handheld
```

Statements (cont.)

@page

introduced for better control over printed pages
media types: all, aural, braille, handheld, print, projection, screen, tty, tv

```
@page {margin: 10px}

@page:left {}

@page:right {}
```

Type selectors

Select any element that matches the selector

syntax

the tag name without < and >

applications

changing appearance of any element on page

```
body  {font-family: Arial}
```

Types of elements

Block elements

- separated from surrounding elements in HTML
- define a separate block in a page
- examples: P, H1

There are:

** Paragraph one **

** Paragraph two **

** Paragraph three **

There are:

Paragraph one

Paragraph two

Paragraph three

Types of elements (cont.)

Inline elements

- flow directly from their surrounding elements in HTML
- are displayed just as text appears on a page
- examples: IMG, A

```
There are:
<a href="a.html">A doc</a>
<p>Paragraph two</p>
<p>Paragraph three</p>
```

There are: A doc

Paragraph two

Paragraph three

Types of elements (cont.)

List item elements

- similar to block elements
- are preceeded with a bullet, dash, number etc.
- example: LI (but not UL or OL!)

There are:

```
<ol>
<li>Paragraph one</li>
<li>Paragraph two</li>
<li>Paragraph three</li>
</ol>
```

There are:

2. Paragraph one
3. Paragraph two
4. Paragraph three

Class selectors

Select any element that belongs to a class

syntax

.class {...}
element.class {...}

applications

define new classes of elements
class attribute in HTML 4.0 elements

```
.note       {font-family: Arial}

p.note      {font-family: Arial}
```

ID selectors

Select exactly one element with the ID

syntax

 # id {...}
element.# id {...}

applications

CSS positioning
ID attribute in HTML 4.0 elements

```
#general-note      {font-family: Arial}

p#general-note     {font-family: Arial}
```

Descendant selectors

Select appropriately nested elements

syntax

p em {...}
h1 img {...}

applications

finer control over the appearance of a page
easy changing the default display of entire documents
without altering the HTML

```
p em                    {font-family: Arial}
```

Pseudo-class selectors

Select an A element according to its state

syntax

a:link {...}
a:visited {...}
a:hover {...}
a:active {...}

applications

changes appearance of a link in different states
beware the order: a, a:link, a:visited, a:hover, a:active

```
a:visited              {background-color: red}
```

Pseudo-element selectors

Select a special-meaning part of an element

syntax

```
:first-line {...}
:first-letter {...}
:before {content: "xyz"}
:after {content: "abc"}
```

applications

changes appearance of part of an element
allows to append additional text to elements

```
p:first-letter    {color: red}
```

Group selectors

Select every element of an enumeration

syntax

p.note, p.definition {...}

applications

shorten the style sheets

```
p.note, p.definition    {color: gray}
```

Language selectors

Select elements encoded in a given language

syntax

:lang(pl) {...}

applications

render the content depending on the language

```
p.lang(pl)  {color: white; background-color: red}
```

Child selectors

Select directly nested elements

syntax

p>em {...}

applications

similar to descendant selectors, but more specific

```
p>em  {color: blue}
```

Adjacent selectors

Select sibling elements

syntax

p + table {...}

applications

format the elements that immediately follow other elements

```
table + p   {indent: 6px}
```

Attribute selectors

Select elements with specific attributes

syntax

```
img[name] {...}              /* attribute is set */
img[nam~="pic1"] {...}       /* attribute is set to a given value */
img[nam~="pic1"] {...}       /* attribute contains a given value */
img[nam|="pic1"] {...}       /* attribute contains a given value */
```

applications

partial support for XSLT-like transformations

```
img[name]    {color: red}
```

Values

length

em, ex, pica (pc), point (pt), pixel (px), mm (mm), cm (cm), in (in)

percentage

p {width: 75%} – relative to the direct container

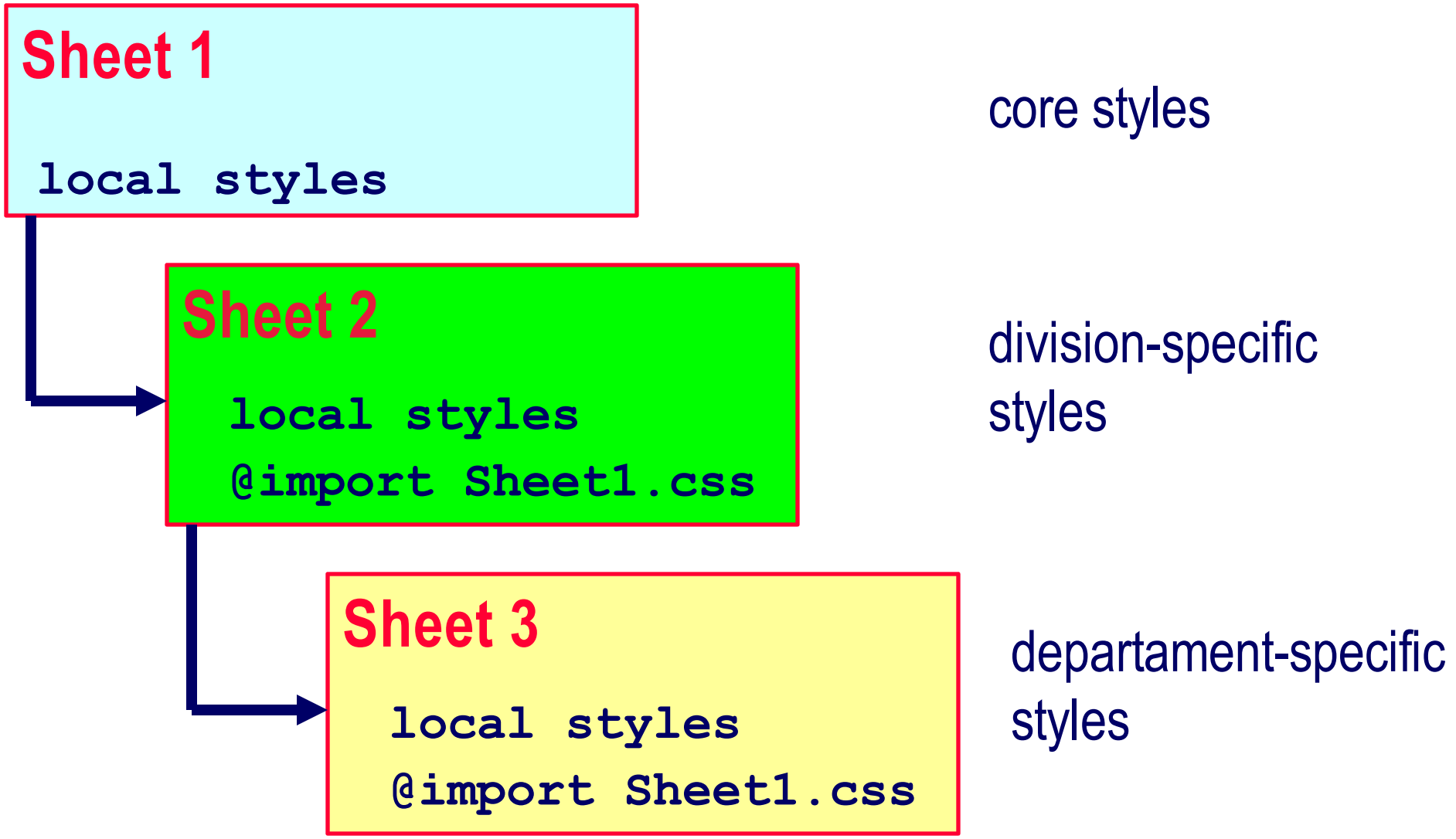
color values

keywords (green), hexadecimal (#ab03ce), rgb (rgb(255, 255, 255))

URLs

url(http://www.put.poznan.pl/), url(../../style.css)

Style-sheet cascade



Inheritance (*)

Select every element of an enumeration

syntax

p.note, p.definition {...}

applications

shorten the style sheets

```
p.note, p.definition    {color: gray}
```

Specificity

1. ! important
2. *author's style* defeates *reader's style*
3. common rule:
    a. 100 * number of ID attrs in a selector
    b. 10 * number of CLASS attrs in a selector
    c. 1 * number of HTML tags in a selector
4. last definition wins

```
#id1 {xxx}                  /* specificity = 100 */
UL UL LI.red {xxx}          /* specificity = 013 */
LI.red {xxx}                /* specificity = 011 */
LI {xxx}                    /* specificity = 001 */
```

Printing with CSS

@page, @page:left, @page:right

page properties

margin, margin-left, margin-right, margin-bottom, margin-top

page-breaking properties

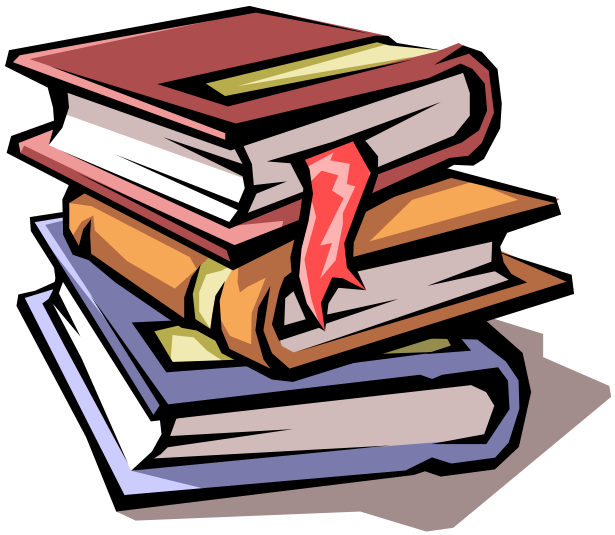
orphans, widows

break-after, break-before, break-inside

```
@page:left {margin: 20px}

P.important {
  break-inside: avoid;
  orphans: 3;
}
```

Readings



1. Gamma E. et al., *Design Patterns. Elements of Reuseable Object-Oriented Software.* Addison-Wesley, 1995
2. Eckel B., *Thinking in patterns*. http://www.bruceeckel.com
3. Cooper J., *Java. Wzorce Projektowe.* Helion, 2001
4. Shalloway A., Trott J., *Projektowanie zorientowane obiektowo. Wzorce projektowe.* Helion, 2001

Q & A