CSS and modern web interface

Maciej Grzenda
Warsaw University of Technology
Faculty of Mathematics and Information Science
M.Grzenda@mini.pw.edu.pl
http://www.mini.pw.edu.pl/~grzendam

Motivation

- Although HTML tags and attributes enable a limited display settings (fonts, colours, alignment, spacing,…), HTML should describe the CONTENT, not the PRESENTATION of the document.
- It is not advisable to define all display settings in HTML files:
  - Imagine a large web site of hundreds of documents to be prepared,
  - Problems:
    - After final acceptance some visual settings of all the documents have to be changed
    - How to maintain consistent settings in hundreds of documents?

CSS

- Cascading Style Sheets (CSS) – a standard defined by W3ORG (http://www.w3.org) that allows to:
  - Set 100+ visual attributes of HTML documents,
  - Separate display definition (how? -> CSS) from content definition (what? -> HTML)
- Vendor-independent, still differences in web browsers can be observed
- Detailed specification can be found at: http://www.w3.org
- Tutorials: http://www.w3schools.com

Standards

- Current finished standard: CSS 2.1: http://www.w3.org/TR/CSS2/
- New standard (under development): CSS 3
- Status of ongoing development works: http://www.w3.org/Style/CSS/current-work
- *CSS Level 3 builds on CSS Level 2 module by module, using the CSS2.1 specification as its core. Each module adds functionality and/or replaces part of the CSS2.1 specification.* ***The CSS Working Group intends that the new CSS modules will not contradict the CSS2.1 specification: only that they will add functionality and refine definitions****. As each module is completed, it will be plugged in to the existing system of CSS2.1 plus previously-completed modules.*

The discussion of CSS 3 vs. CSS2.1 cited above comes from http://www.w3.org/TR/CSS/#css3

CSS – basic rules

- CSS attributes serve to define display and behaviour of HTML items e.g. tables, paragraphs, images
- CSS definitions can be placed in:
  - HEAD section of HTML document,
  - As STYLE attribute of any HTML tag,
  - In separate CSS documents

CSS – first example

```
…
<head>
   <style type="text/css">
        h3 {color:red;}
        p  {font-size:20pt;margin-left:20pt;}
   </style>
</head>
<body>
   <h3>Sample header</h3>
   <br />
   <p>Sample paragraph starts here
   …
```

All H3 tags will be displayed in red. All paragraphs will be affected by font size and margin settings.

CSS in action

In this case, font sizes, backgrounds, margins, location of elements and borders are defined.

CSS rule syntax

- Style section is defined by a number of selectors combined with style settings:

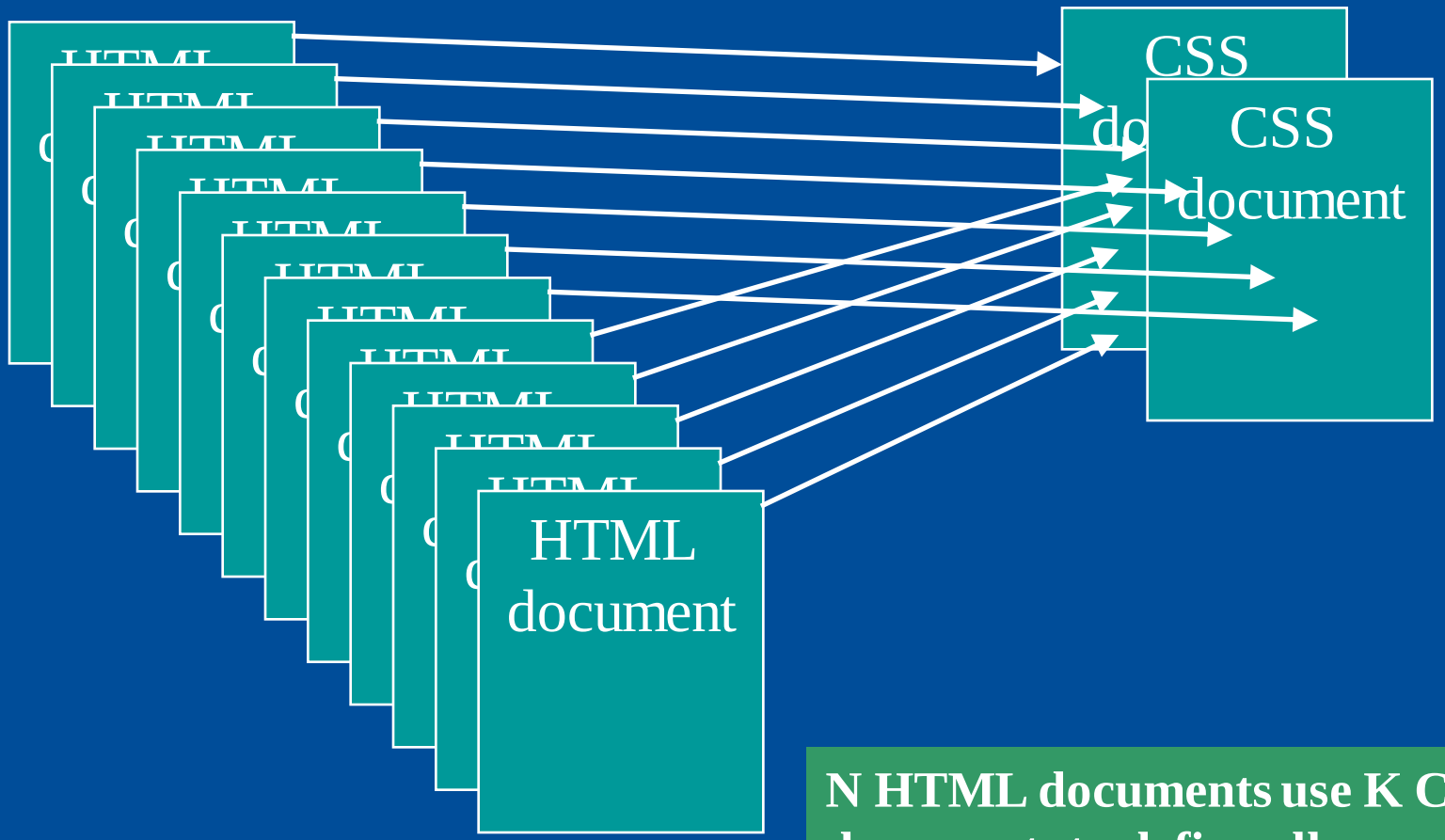
  Selector {*Declaration*}
- Declaration takes the following form

  CSSAttribute:CSSAttributeValue;…
- Each CSS rule can be defined in:
  - STYLE tag in HEAD section of HTML document
  - Separate file linked in HEAD section by

    `<link rel="stylesheet" type="text/css" href="myCSSFile.css">`
  - **`style`** attribute

    Example: **`<p style="font-size:20pt;">`** …

CSS – suggested approach



N HTML documents use K CSS documents to define all visual settings. K << N!

Why separate CSS files?

- Centralised display definition:
  - Whenever there is a need to change visual attributes of some HTML objects (e.g. images), it can be done through the modification of a single document
  - Limited risk of inconsistent settings. Such inconsistencies are inevitable if visual attributes are defined in each HTML document separately

CSS and classes

- Problem:
  - What if different settings have to be applied for the paragraphs of the same HTML document?
  - Is STYLE attribute the only solution?
- Solution:
  - Class definitions

Class selectors

- Example:

```
p.marketing {font-size:20pt;margin-left:20pt;}
p.notes  {font-size:10pt;}
…
<p class="marketing">…</p>
```

- In general: for each HTML tag a number of subclasses can be defined
- Also:

  `.marketing` {….} – all HTML tags with this class will be affected
- Arbitrary number of classes can be defined
- Thus, both numerous display settings can co-exist in one HTML document and centralised CSS settings can be applied
- Any modern HTML web site will use both CSS and CSS classes

Class selectors - remarks

- Classes should be self-documenting and their names should rather reflect the logical meaning of the tag than its physical appearance.
- Thus, instead of `class="italic"` `class="copyrightnotice"` should rather be used.
- Class names can not contain spaces in their names and should be in lowercase – please consult CSS specification for details
- One HTML element can have many classes (e.g. `class="copyrightnotice important"`)

Contextual selectors

- These provide the rule: if ELEMENT2 is a descendent of ELEMENT1, then the given properties apply to those ELEMENT2s, e.g.

```
STRONG EM {text-transform:uppercase};
```

  Which says: EM inside STRONG block will be uppercase
- Other examples:

```
ul li {background-color: black;
color: silver}

ol li {color: yellow; }
```

Pseudo-class selectors - examples

- These are so called because the behaviour is as if the element was specially marked as that class.
- The support for such selectors may strongly vary – depending on the browser

Selector	Meaning	Example
`:link`	Unvisited links	`A:link{color:red}`
`:visited`	Visited links	`A:visited{color:purple}`
`:hover`	*Applies when a mouse is over an element*	*`P:hover{text-decoration : underline}`*
`:focus`	When the element has the focus	`A:focus...`
`:active`	*Applies when element is being activated*	*`INPUT:active{color:red}`*

Other pseudo-classes proposed in CSS3 can be found at http://www.w3.org/TR/css3-selectors/

Inheritance and CSS

- The CSS definitions can naturally follow the tree structure of HTML. Basically by defining the properties of a certain level of the HTML tree one can affect all the tags in the child sub-tree, for instance the definition BODY {color:green} will affect the headers as well.
- As usual, when inheritance takes place the settings can be overridden as well, for instance, by defining:

```
body {color:green;}
h1   {color:blue;}
```

All the text but the header1 (H1) will be displayed in green

Common tasks

- **CSS and backgrounds**
- Among the most common tasks performed by CSS setting backgrounds plays an important role.

  Example:

  `BODY`

  `{background : url('texture.gif ')}`
  or

  `background: #RGB` (in RGB notation)

Common tasks: fonts

Property	Sample setting
Font-style	Italic, normal
Font-weight	Bold, normal,bolder,lighter
Font-family	Serif
Font-size	36pt,xx-small,x-small,small,medium,large,x-large,xx-large,1cm etc.
Text-transform	Capitalize, uppercase,lowercase,none
Text-align	Left,right,justify,center
Text-decoration	Underline,none,line-through,blink

Block and inline elements

- *Certain HTML elements that may appear in BODY are said to be "block-level" while others are "inline" (also known as "text level"). The distinction is founded on several notions:*
  - ***Content model*** *Generally, block-level elements may contain inline elements and other block-level elements. Generally, inline elements may contain only data and other inline elements. Inherent in this structural distinction is the idea that block elements create "larger" structures than inline elements.*
  - ***Formatting*** *By default, block-level elements are formatted differently than inline elements.* ***Generally, block-level elements begin on new lines, inline elements do not.***

This material comes from http://www.w3.org/TR/html401/struct/global.html Further data on block and inline elements can be found at the same address.

DIV, SPAN and CSS

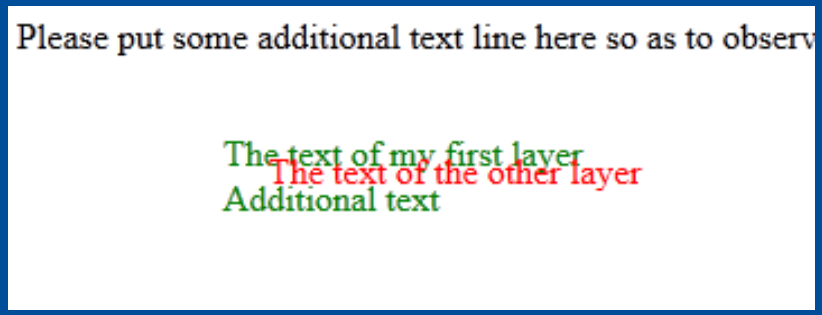
- DIV and SPAN are HTML elements that have no a priori meaning, unlike most HTML tags.
- DIV element is a block element, SPAN is an inline element.
- Both elements take no HTML attributes and are applied when their content can not be semantically described by other HTML elements.
- In fact, the main motivation is to apply the same CSS-based formatting to their content

DIV positioning

- The use of DIVs is strongly coupled with CSS positioning attributes. These can be used to define bottom, top, left and right attributes. Either auto or the actual position measured in pixels or % can be used e.g. left:75px stands for 75 pixels from the left,
- position attribute is used to define the interpretation of the positioning attributes. The following settings can be used:
  - **`static`** – the position is based on standard HTML,
  - **`absolute`** – the position is based on absolute coordinates,
  - **`fixed`** – the position is based on absolute coordinates of the browser window,
  - **`relative`** – the coordinates define the position relative to default - resulting from standard HTML rules.

Sample DIV-based solution

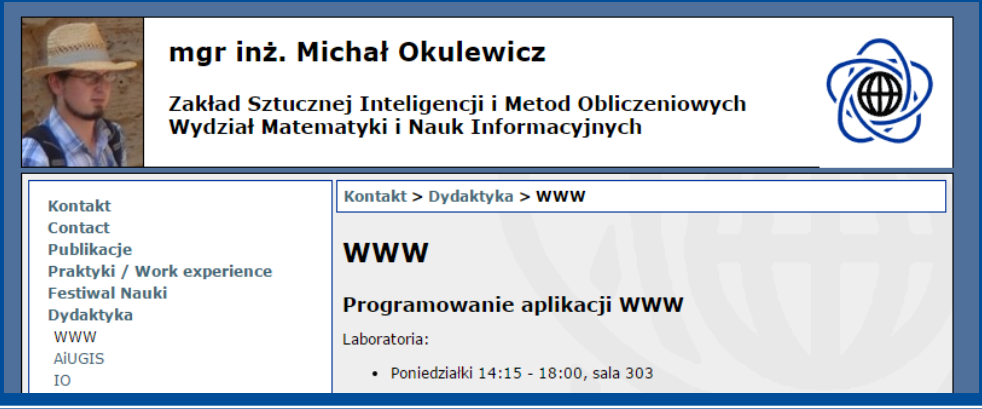
```
…
<style type="text/css">
div.mylayer1 { position:absolute; left:100px; top:60px;}
div.mylayer2 { position:absolute; left:120px; top:68px;}
</style>
</head>
<body>

Please put some additional text line here so as to observe the way
   layers co-exist in the document

<div class="mylayer1" style="color:green;">
The text of my first layer <br />
Additional text
</div>

<div class="mylayer2" style="color:red;">
The text of the other layer
</div>…
```



Layers may overlap with each other and the main document content. By switching them on and off dynamic display of menus, tabs in page frames etc. can be obtained

CSS and Responsive Web Design

- CSS might also help with RWD
  - `@media screen and (max-device-width: 768px) and (max-device-height: 768px) {`
    `/* Style for smaller devices */`
    `}`
  - `@media screen and (min-device-width: 768px) and (min-device-height: 768px) {`
    `/* Style for larger devices */`
    `}`



Comments in CSS

- Comments begin with the characters "/*" and end with the characters "*/". This is de facto the only way of commenting in CSS.
- Comments may not be nested.
- Notice: HTML (SGML-like) comments will be ignored. For example, the H1 and H2 rules WILL affect display of the document:

```
<style type="text/css">
/* h3 {color:blue;background-color: #0F0;}  */
  <!--
   h1 {color:blue;background-color: #0F0;}
   h2 {color:green;background-color: #F00;}
   -->
</style>
```

1. **Based on CSS specification (http://www.w3.org/TR/CSS2/)**
2. **In the example, only H3 rule will be ignored.**

Colors and units in CSS

- All size related styles must use one of the predifined units (e.g. px,pt,%,em,vw,vh,...)
- http://www.w3.org/Style/Examples/007/units.pl.html
- Colors can be specfied in the following way:
  - #RRGGBB — #FF0000
  - #RGB — #F00
  - color name — red
  - rgb(red, green, blue) — rgb(255,0,0)
  - rgba(red, green, blue, alpha) — rgba(255,0,0,1.0)
  - hsl(hue, saturation, lightness) — hsl(0,100%,50%)
  - hsla(hue, saturation, lightness, alpha) — hsla(0,100%,50%,1.0)

To sum up

- Modern web sites do use CSS
- Separate CSS files containing CSS rules are suggested
- Pseudo-classes may provide dynamic display settings
- Large number of attributes suggests the use of CSS specification rather than learning all possible attributes and their settings