An Introduction to CSS

Prof. Ing. Andrea Omicini
Ingegneria Due, Università di Bologna a Cesena
andrea.omicini@unibo.it

2006-2007

Web Style Sheets

- Style sheets for the Web
- Aims
  - describing how elements in a document must be presented
    - on different media types, as paper print, video, audio, medium for people with disabilities, etc.
  - separating style's description from content and its structure
- See **http://w3c.org/Style/**
  - Many specifications: CSS1, CSS2, XPath, XSLT, XSL-FO
  - Two languages: CSS & XSL

Why two languages?

- CSS
  - can be used with HTML and XML
  - but it has its own syntax, and it's not general enough to be a transformational language
- XSL (union of XSLT / XSL-FO / XPath)
  - it's a transformational language
    - e.g., it can be used to transform an XML page in HTML/CSS
  - featuring an XML syntax
  - but it can be used with XML only, not with HTML
- Indeed, they share the same "formatting model"...
- ...and they can be used together

Dynamic HTML

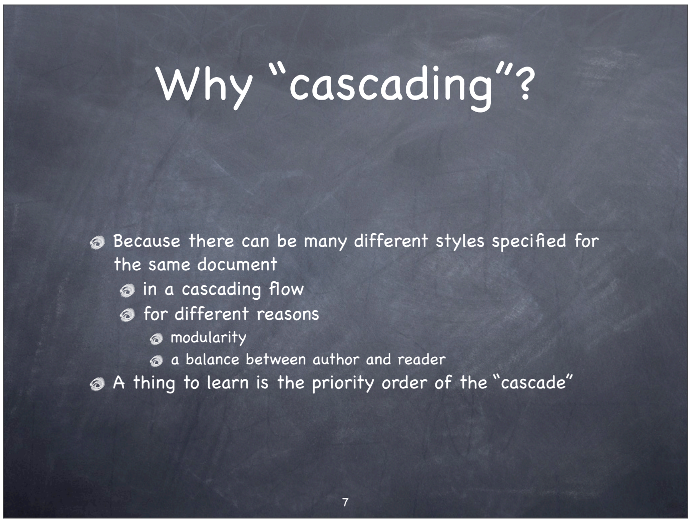
- HTML pages with dynamic content
- composed using three technologies
  - HTML / XHTML
  - CSS
  - JavaScript / ECMAScript
- sharing the DOM
  - Document Object Model
    - which describes the conceptual general structure of a DHTML document
  - which is referenced by browsers
    - which feature their own detailed DOM specifications
      - which we have to know and avoid

AJAX

- Asynchronous JavaScript And XML
  - goal: improve interaction between browsers & servers
- composed using three technologies
  - a combination of:
    - XHTML / HTML & CSS
    - JavaScript for DOM manipulation
    - XMLHttpRequest object
      - to exchange data asynchronously with server
    - usually, XML for data transfer
- example: changing a portion of a web page according to some user interaction without reloading a whole page

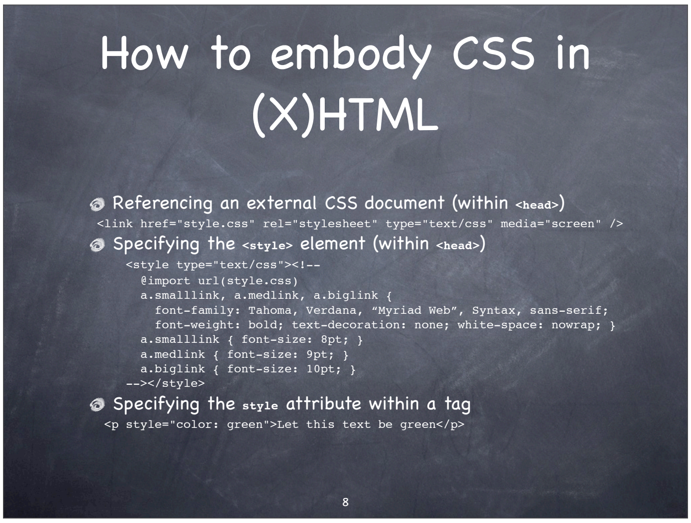
CSS Specifications

- CSS1, CSS2, and above
  - CSS3 under development
- We focus our work on CSS1
  - study CSS1 besides tutorials
    - see **http://www.w3c.org/TR/REC-CSS1**
  - because questions in the exam will be based on that specification
    - so you'll benefit from learning how to quickly search needed information in that document



Why “cascading”?

- Because there can be many different styles specified for the same document
  - in a cascading flow
  - for different reasons
    - modularity
    - a balance between author and reader
- A thing to learn is the priority order of the “cascade”



How to embody CSS in (X)HTML

- Referencing an external CSS document (within **<head>**)

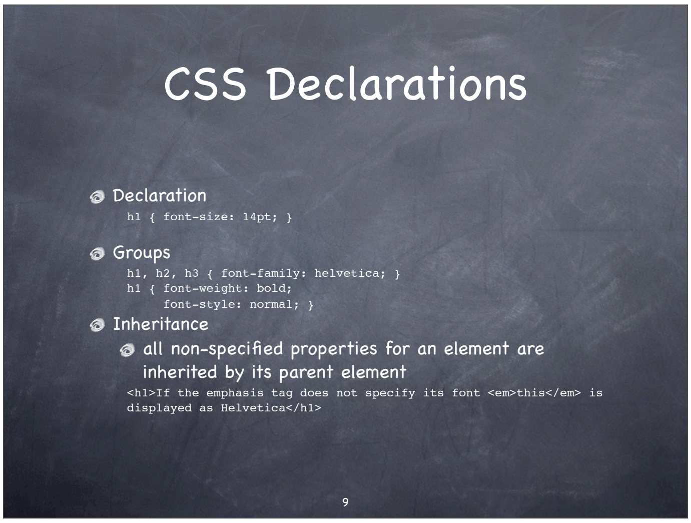
```
<link href="style.css" rel="stylesheet" type="text/css" media="screen" />
```

- Specifying the **<style>** element (within **<head>**)

```
    <style type="text/css"><!--
      @import url(style.css)
      a.smalllink, a.medlink, a.biglink {
        font-family: Tahoma, Verdana, “Myriad Web”, Syntax, sans-serif;
        font-weight: bold; text-decoration: none; white-space: nowrap; }
      a.smalllink { font-size: 8pt; }
      a.medlink { font-size: 9pt; }
      a.biglink { font-size: 10pt; }
    --></style>
```

- Specifying the **style** attribute within a tag

```
 <p style="color: green">Let this text be green</p>
```



CSS Declarations

- Declaration

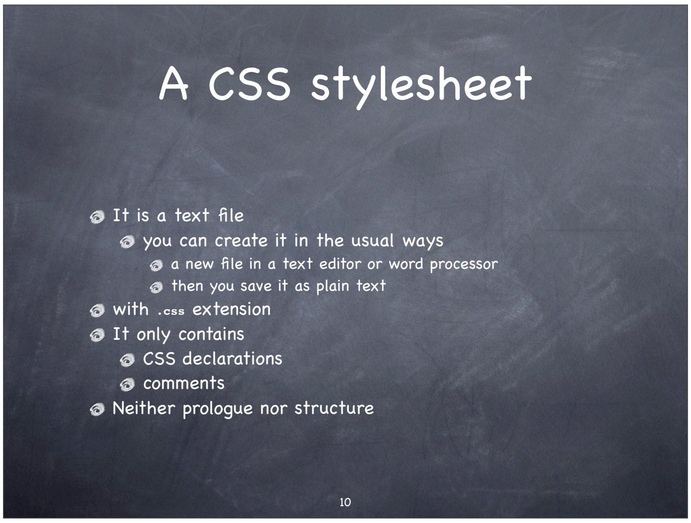
```
h1 { font-size: 14pt; }
```

- Groups

```
h1, h2, h3 { font-family: helvetica; }
h1 { font-weight: bold;
     font-style: normal; }
```

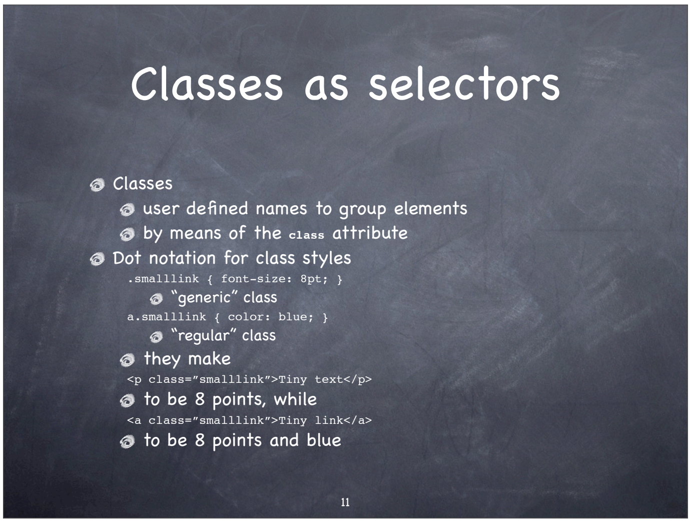
- Inheritance
  - all non-specified properties for an element are inherited by its parent element

```
<h1>If the emphasis tag does not specify its font <em>this</em> is
displayed as Helvetica</h1>
```



A CSS stylesheet

- It is a text file
  - you can create it in the usual ways
    - a new file in a text editor or word processor
    - then you save it as plain text
- with **.css** extension
- It only contains
  - CSS declarations
  - comments
- Neither prologue nor structure



Classes as selectors

- Classes
  - user defined names to group elements
  - by means of the **class** attribute
- Dot notation for class styles

```
.smalllink { font-size: 8pt; }
```

    - “generic” class

```
a.smalllink { color: blue; }
```

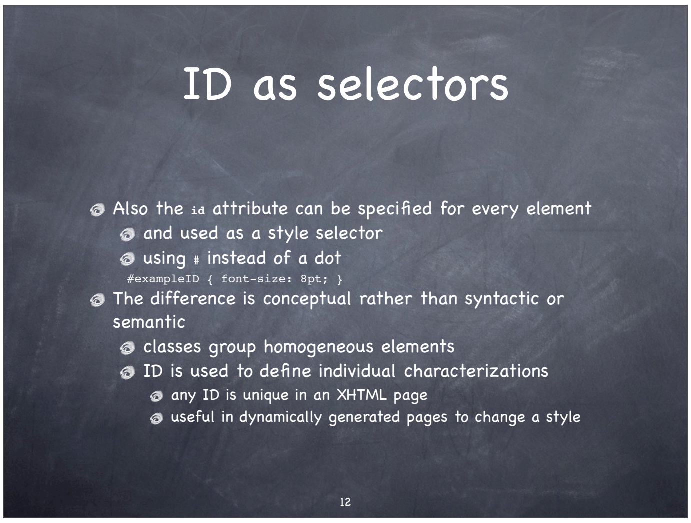
    - “regular” class
  - they make

```
<p class=”smalllink”>Tiny text</p>
```

  - to be 8 points, while

```
<a class=”smalllink”>Tiny link</a>
```

  - to be 8 points and blue

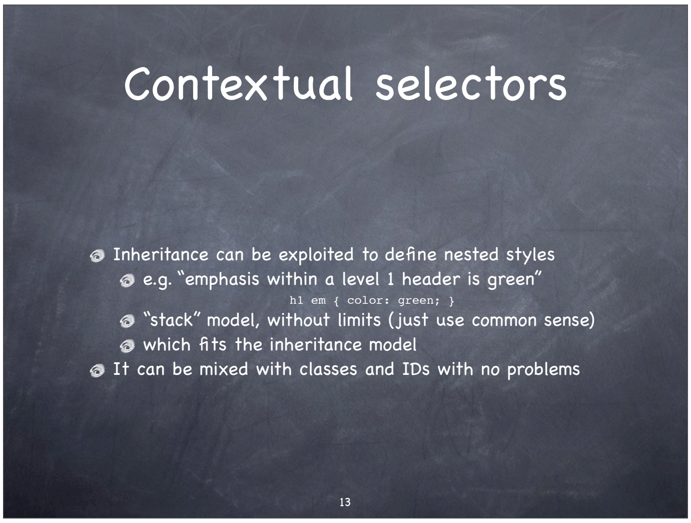


ID as selectors

- Also the **id** attribute can be specified for every element
  - and used as a style selector
  - using **#** instead of a dot

```
#exampleID { font-size: 8pt; }
```

- The difference is conceptual rather than syntactic or semantic
  - classes group homogeneous elements
  - ID is used to define individual characterizations
    - any ID is unique in an XHTML page
    - useful in dynamically generated pages to change a style



Contextual selectors

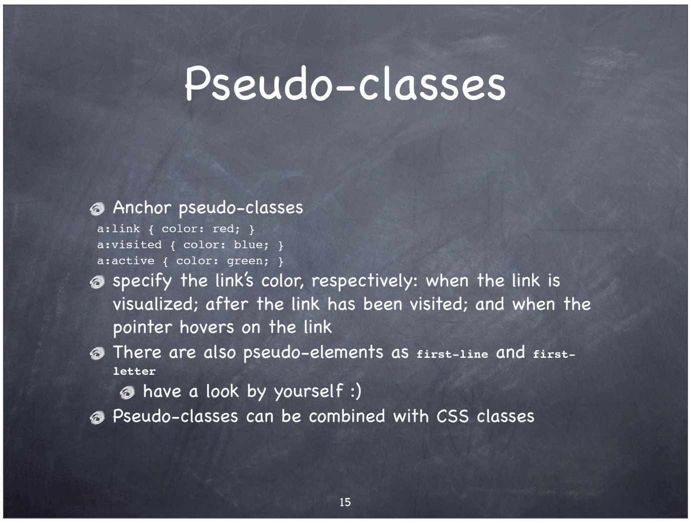
- Inheritance can be exploited to define nested styles
  - e.g. "emphasis within a level 1 header is green"

```
h1 em { color: green; }
```

  - "stack" model, without limits (just use common sense)
  - which fits the inheritance model
- It can be mixed with classes and IDs with no problems

Comments

```
/* This is a comment */
```

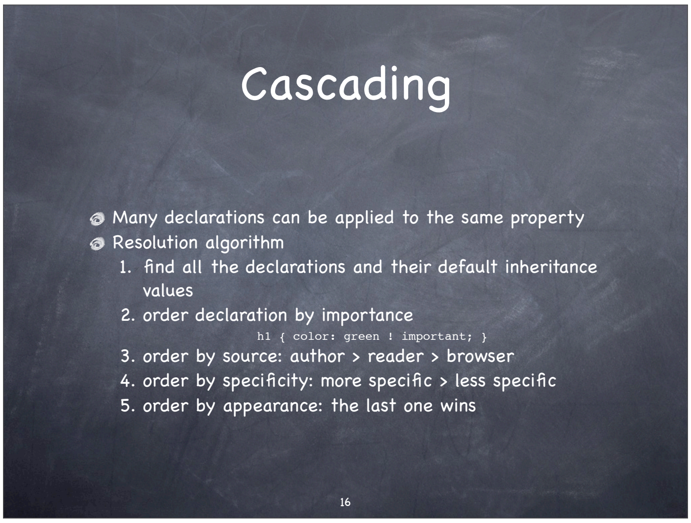


Pseudo-classes

- Anchor pseudo-classes

```
a:link { color: red; }
a:visited { color: blue; }
a:active { color: green; }
```

- specify the link's color, respectively: when the link is visualized; after the link has been visited; and when the pointer hovers on the link
- There are also pseudo-elements as `first-line` and `first-letter`
  - have a look by yourself :)
- Pseudo-classes can be combined with CSS classes

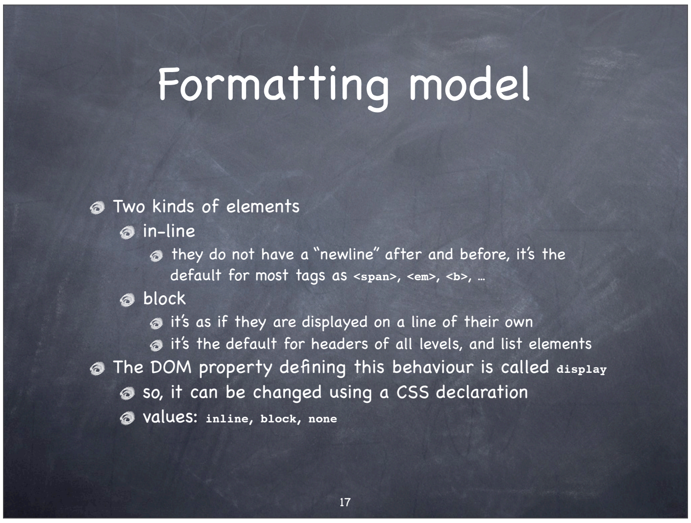


Cascading

- Many declarations can be applied to the same property
- Resolution algorithm
  1. find all the declarations and their default inheritance values
  2. order declaration by importance

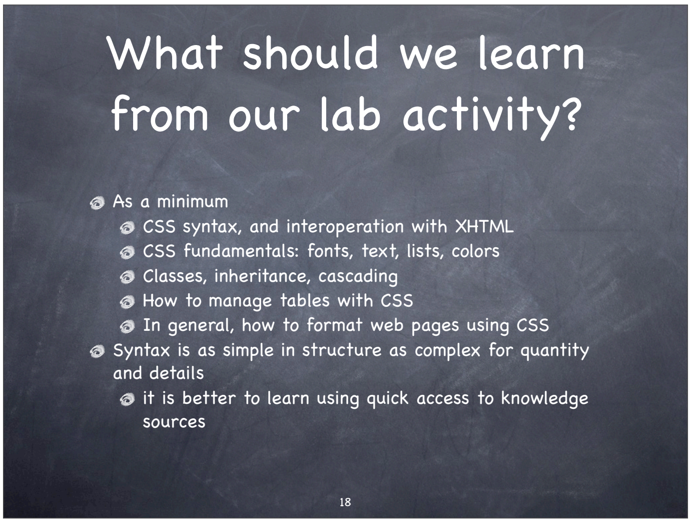
```
h1 { color: green ! important; }
```

  3. order by source: author > reader > browser
  4. order by specificity: more specific > less specific
  5. order by appearance: the last one wins



Formatting model

- Two kinds of elements
  - in-line
    - they do not have a "newline" after and before, it's the default for most tags as `<span>`, `<em>`, `<b>`, …
  - block
    - it's as if they are displayed on a line of their own
    - it's the default for headers of all levels, and list elements
- The DOM property defining this behaviour is called `display`
  - so, it can be changed using a CSS declaration
  - values: `inline`, `block`, `none`



What should we learn from our lab activity?

- As a minimum
  - CSS syntax, and interoperation with XHTML
  - CSS fundamentals: fonts, text, lists, colors
  - Classes, inheritance, cascading
  - How to manage tables with CSS
  - In general, how to format web pages using CSS
- Syntax is as simple in structure as complex for quantity and details
  - it is better to learn using quick access to knowledge sources