Web

Master 1 IFI

Andrea G. B. Tettamanzi

Université de Nice Sophia Antipolis

Département Informatique

andrea.tettamanzi@unice.fr

Unit 2

A refresher on HTML, CSS, and the Document Object Model

Agenda

- Separation of Content and Presentation
- HTML
- DOM
- CSS

Separation of Content and Presentation

- The "**Separation of Concerns**" design principle
- Application to the authoring and presentation of content:
  - Content of a document: material and structure
  - Presentation: graphical design and style
- Use in computer-based typesetting
  - e.g.: LaTeX
- Use in Web Design
  - A best practice more than a rigid guideline
  - HTML is used to give structure to the core material of a page
  - CSS is used to define the graphical design and style

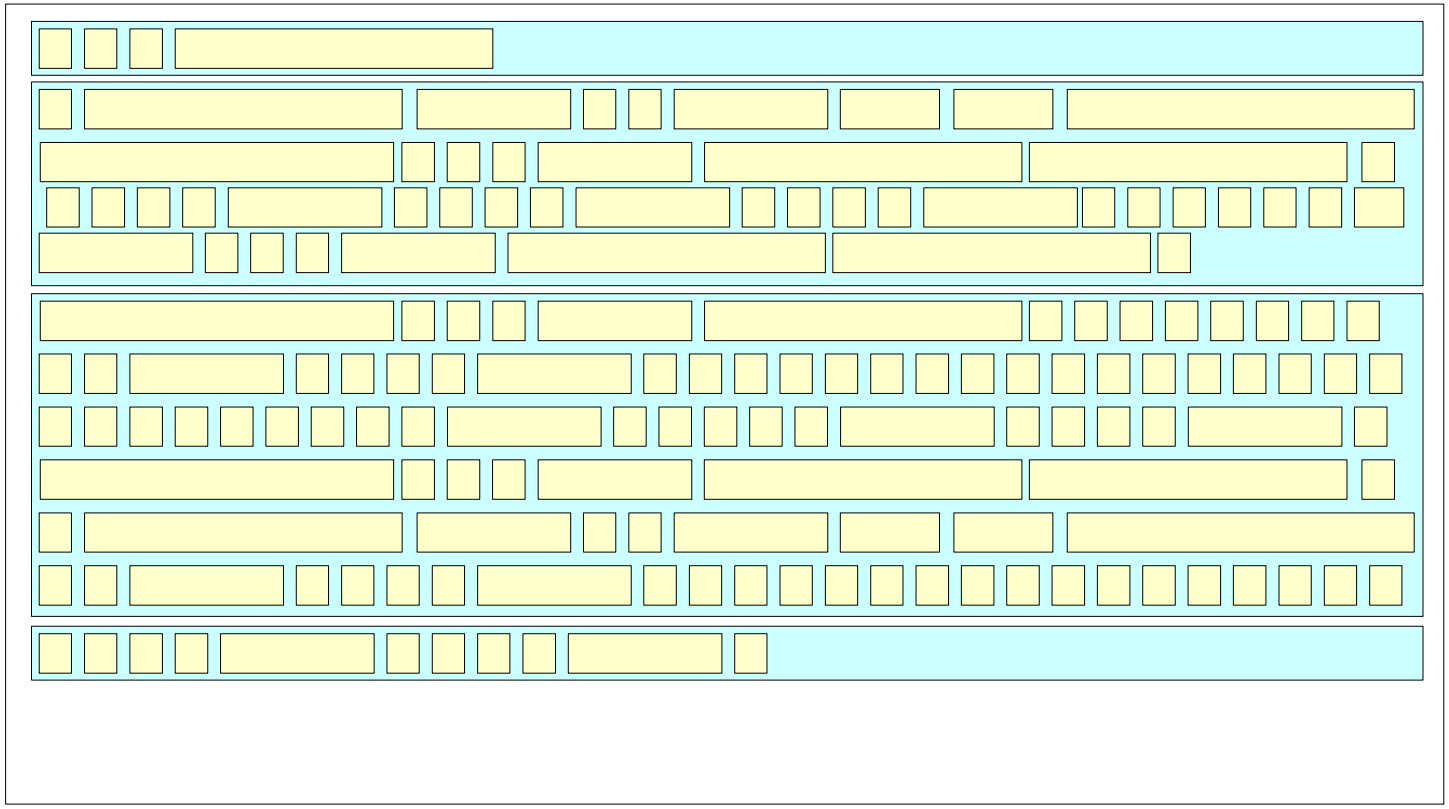
HTML

- Markup language
  - Any system for annotating a document in a way that is syntactically distinguishable from the text
  - In particular, for structuring and/or formatting
  - But also to add semantic annotations
- Tags (or Elements): <*tag_name attributes*> … </*tag_name*>
- Attribute: *attr_name*=”*value*”
- Types of tags:
  - structural, text, hyperlink, articulation, lists, tables, forms, media, script, frame, metadata,
  - Formatting (deprecated!)

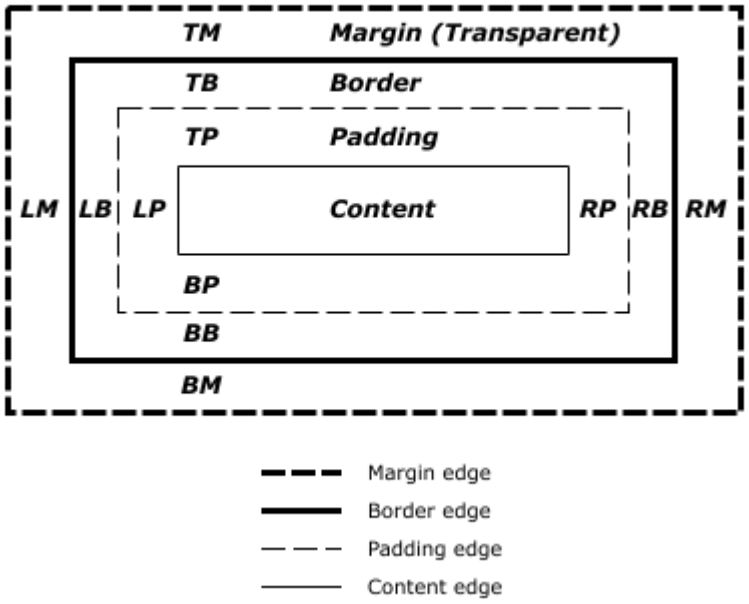
Structural Tags

- **html**: defines a document
  - one head, one body
- **head**: the part consisting of headers and metadata
  - A mandatory <title> … </title>
  - Optional “**meta**”, “**script**”, “**style**”, etc. tags
- **title**: the name of the document (and of the window)
  - Contains some text
- **body**: the document body
  - Core material of the document (the content)

Block and Inline Elements



Box Model


TM
Margin (Transparent)
TB
Border
TP
Padding
LM
LB
LP
Content
RP
RB
RM
BP
BB
BM
Margin edge
Border edge
Padding edge
Content edge

Block Elements

- Generic grouping: div | section | fieldset | summary
- Paragraphs: p | pre
- Titles: h1 | h2 | h3 | h4 | h5 | h6
- Lists : ol | ul | dl
- Tables: table
- Forms: form
- Miscellaneous: noscript | blockquote | hr | address
- Floats: aside | figcaption | figure | header | footer | main | mark | nav | details |

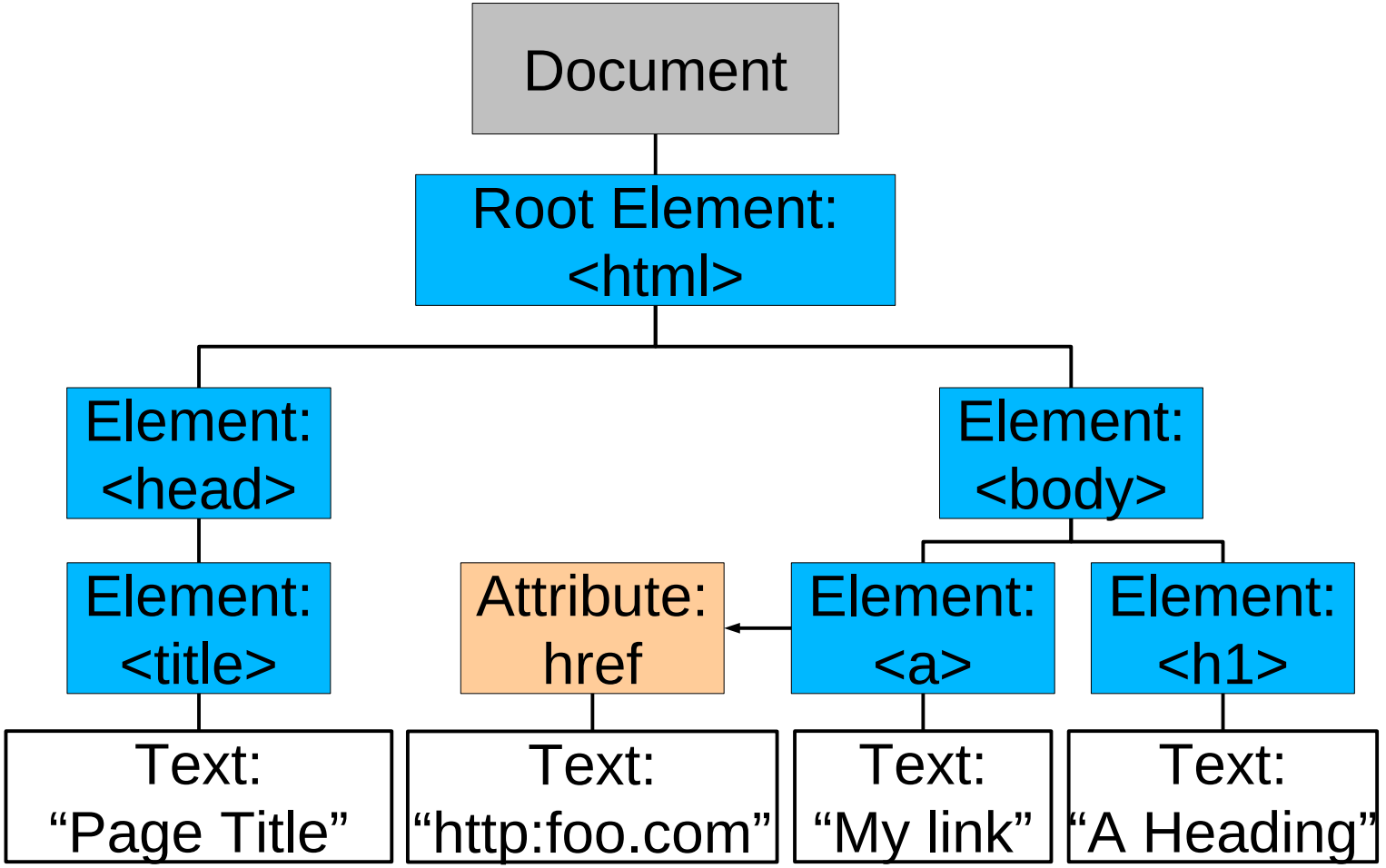
Inline Elements

- Text (PCDATA)
- Phrase: em | strong | dfn | code | samp | kbd | var | cite | abbr | acronym | time
- Media: img | object | audio | video | track | embed | source
- Special: a | br | wbr | script | map | q | sub | sup | span | bdo | bdi
- Form: input | select | textarea | label | button | datalist | keygen | output | progress
- Font Style (deprecated): tt | i | b | big | small

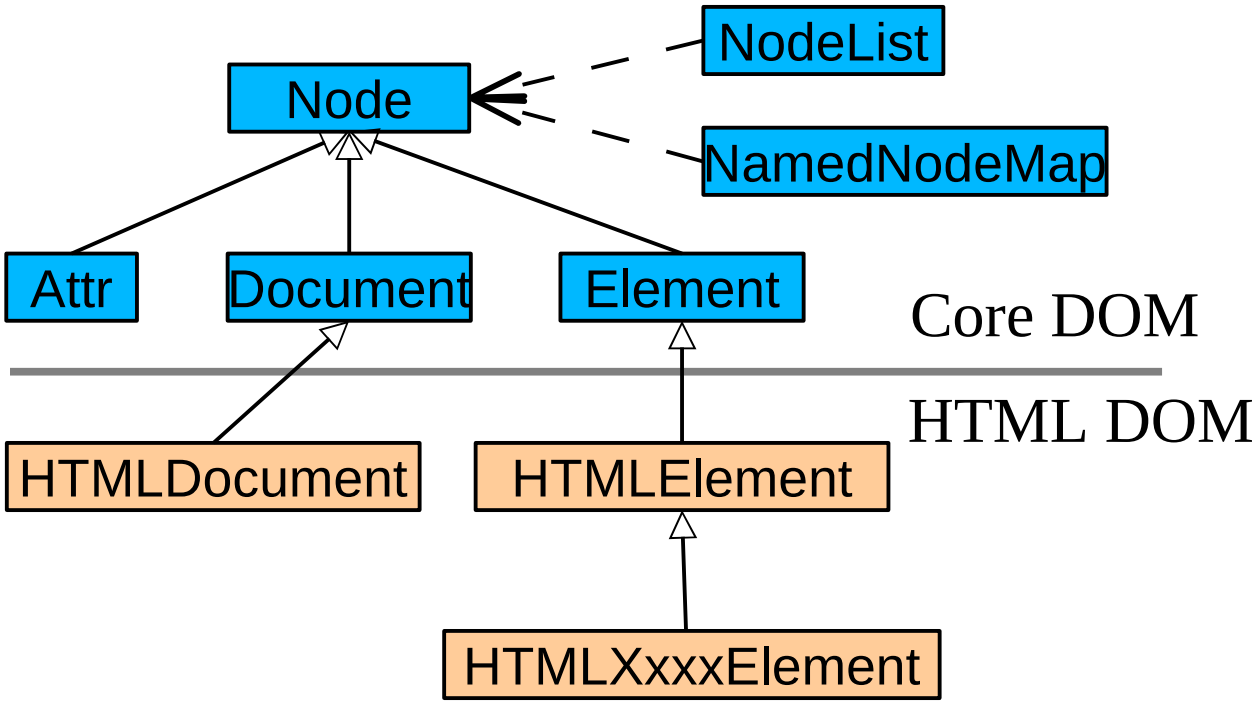
Forms

```
<form action="http://somesite.com/prog/adduser" method="post">
   <P>
   <label for="firstname">First name: </label>
            <input type="text" id="firstname"><br>
   <label for="lastname">Last name: </label>
            <input type="text" id="lastname"><br>
   <label for="email">email: </label>
            <input type="text" id="email"><br>
   <input type="radio" name="sex" value="Male"> Male<br>
   <input type="radio" name="sex" value="Female"> Female<br>
   <input type="submit" value="Send"> <input type="reset">
   </p>
</form>
```

Document Tree



DOM Objects


NodeList
Node
NamedNodeMap
Attr
Document
Element
Core DOM
HTML DOM
HTMLDocument
HTMLElement
HTMLXxxxElement

CSS

- CSS = Cascading Style Sheets
- Rule-based: element selector → attribute value assignments
- The presentation of a Web page is the combined result of
  - An HTML document (structure, content)
  - One or more style sheets (formatting)
- The document is parsed and transformed into a (DOM) tree
  - A style sheet consists of a number of rules
- The document-tree is visited
  - For each node (element), the matching rules are applied
  - In case of conflict, rules are prioritized based on specificity
- The result is a “decorated” tree

Syntax

- Two types of instruction:
  - Directives (*at-rules*), « @ » + <identifier>
    - @import, @charset, @media, …
    - Terminated by “;” or by a declaration block
  - Rule sets, <selector> + <declaration block>
    - Declaration Block: “{“ + <declarations> + “}”
    - Declaration: <property> : <value> ;
    - Selector: an expression involving tags and pseudo-tags
- Comments : /* … */
- Any invalid expression is silently ignored

Examples

```
  h2 { color: green }

                        @import subs.css

h3:before {
   content: counter(sectno, upper-roman) " - ";
   float: left;
   margin-left: -2cm;
}
```

Including a CSS Style Sheet

- External style sheet (referenced by an URI):
  - <link rel="stylesheet" type="text/css" media="screen" href="http://foo.org/file.css" />.
  - The **link** tag must appear between <head> and </head>
- @import "fichier.css", within a <style>…</style> tag
- Within the document (always in the head) by using a **style** tag:
  - <style type="text/css"> h1 { color: green ; }</style>.
- Embedding a declaration block in a tag via the **style** attribute:
  - <li style="color:red;"> …

Selectors (1/4)

- **element** pattern (type selector)
  - Matches the specified element, which will appear with the style defined in the declaration block, within { }
  - Example: p { color: white ; background-color: black; }
- **,** pattern (grouping)
  - Groups elements
  - Example: H1, H2, H3 { color: blue; }
- * pattern (universal selector)
  - Matches any element
  - Example: * { background-color: rgb(230,255,230); }

Selectors (2/4)

- **element$_1$ element$_2$** pattern (descendant selector)
  - Matches element$_2$ tags having element$_1$ as an ancestor
  - Example: p span { color: red ; background-color: orange; }

- **element.classname** pattern (class selector)
  - Matches the specified element of the specified class
  - Example: p.special { color: white ; background-color: black; }

- **.classname** pattern (class selector)
  - Matches all elements of the specified class
  - Example: .special {color: white ; background-color: black;}

Selectors (3/4)

- **#id** pattern (ID selector)
  - Matches the element with the specified ID
  - Example:

    #special { background-color:#ffffef; border:blue 2px solid; }
- **element$_1$ > element$_2$** pattern (child selector)
  - Matches element$_2$ tags having element$_1$ as parent
  - Example: p > span { background-color:#ffefff; border: solid; }
- **element$_1$ + element$_2$** pattern (adjacent sibling selector)
  - Matches element$_2$ tags having element$_1$ as previous sibling
  - Example: p + ul { background-color:#efffff; border-color:red; }

Selectors (4/4)

- **element[attribute]** pattern (attribute selector)
  - Matches the specified element if its specified attribute is defined (independently of its value)
  - Example: p[id] { color: white; background-color: black; }

- **element[attribute="value"]** pattern (attribute selector)
  - Matches the specified element if its specified attribute has the specified value

- **element[attribute~="value"]** pattern (attribute selector)
  - Matches the specified element if its specified attribute contains the specified value (as a substring)

Pseudo Classes

- **element:first-child**
  - Matches the first child of the specified element
- For an anchor: **a:link** and **a:visited**
  - Matches not yet visited and already visited links, respectively

- Dynamic pseudo-classes **:active** , **:hover** and **:focus**
  - **:hover** matches the element hovered by the pointer
  - **:active** matches the active element (b/w press and release)
  - **:focus** matches the element having the focus

Pseudo Elements

- :first-line
  - First line of the text of the element
- :first-letter
  - First letter of the text of the element
- :before and :after
  - Used to insert generated content before or after an element
  - H1:before {content: counter(chapno, upper−roman) ". "}
- May be combined:
  - P.special:before { content: "Special! "}
  - P.special:first−letter { color: gold }

Assigning Property Values, Cascading, and Inheritance

- The user agent (aka the browser) determines the style for every element
- Style = values for the properties that apply to the target media type (e.g., screen)
- The final value of a property is the result of a four-step calculation
  - **Specified** value: the value is determined through specification
  - **Computed** value: resolved into a value used for inheritance
  - **Used** value: converted into an absolute value if necessary
  - **Actual** value: transformed according to the limitations of the local environment

The Cascade

- Style sheets may have three different origins:
  - Author. The **author** specifies style sheets for a source document using CSS
  - User. The **user** may be able to specify style information for a particular document.
  - User agent: **default** style sheet.
- The CSS cascade assigns a weight to each style rule.
- When several rules apply, the one with the greatest weight takes precedence.
- By default, author > user > default
- For "!important" rules, user > author > default

Specified Value

- User agents must first assign a specified value to each property based on the following mechanisms (in order of precedence):
  1) If the cascade results in a value, use it.
  2) Otherwise, if the property is inherited and the element is not the root of the document tree, use the computed value of the parent element.
  3) Otherwise use the property's initial value. The initial value of each property is indicated in the property's definition.

Computed Values

- Specified values are resolved to computed values during the cascade; for example.
  - URIs are made absolute (if possible)
  - 'em' and 'ex' units are computed to pixel or absolute lengths.
  - Computing a value never requires the user agent to render the document.
- The computed value of a property is determined as specified by the Computed Value line in the definition of the property.
- If the specified value is 'inherit', the inheritance rules are used.
  - Values are inherited from the parent in the DOM tree.
  - Each property defines whether it is inherited or not.
  - When inheritance occurs, elements inherit computed values.

Cascading Order

1) Find all declarations that apply to the element and property in question
2) Sort according to importance (normal or important) and origin (author, user, or user agent)
3) Sort rules with the same importance and origin by specificity of selector
   - more specific selectors will override more general ones
   - pseudo-elements and pseudo-classes are counted as normal elements and classes, respectively
4) To break any remaining ties, sort by order specified:
   - the latter specified wins.
   - imported style sheets are always specified “before”

Specificity of a Selector

- A selector's specificity is calculated as follows:
  - count 1 if from is a 'style' attribute rather than a rule with a selector, 0 otherwise (= a)
  - count the number of ID attributes in the selector (= b)
  - count the number of other attributes and pseudo-classes in the selector (= c)
  - count the number of element names and pseudo-elements in the selector (= d)
- Concatenating the four numbers a-b-c-d (in a number system with a large base) gives the specificity.
  - `* {}` /* a=0 b=0 c=0 −> specificity = 0 */
  - `LI {}` /* a=0 b=0 c=1 −> specificity = 1 */
  - `UL LI {}` /* a=0 b=0 c=2 −> specificity = 2 */
  - `UL OL+LI {}` /* a=0 b=0 c=3 −> specificity = 3 */
  - `H1 + *[REL=up]{}` /* a=0 b=1 c=1 −> specificity = 1-1 */
  - `UL OL LI.red {}` /* a=0 b=1 c=3 −> specificity = 1-3 */
  - `LI.red[title] {}` /* a=0 b=2 c=1 −> specificity = 2-1 */
  - `#x34y {}` /* a=1 b=0 c=0 −> specificity = 1-0-0 */
- In HTML, values of an element's ‘style’ attribute are style sheet rules. These rules have no selectors, so a=1, b=0, c=0, and d=0.

Media

- Stylesheets allow to adapt the presentation to media
  - Media = device used to present the document
  - on the screen, on paper, with a speech synthesizer, with a Braille device, etc.
  - Some properties are specific to some media
- Style-Media Association
  - two ways to specify media dependencies for style sheets:
  - @import url("loudvoice.css") aural;
  - @media print { /* stylesheet for printing */ }
  - @media screen, print { BODY { line−height: 1.2 } }
  - <LINK rel="stylesheet" type="text/css" media="print, handheld" href="foo.css">

Media Types

Media Types	**Media Groups**			
	continuous/paged	visual/aural/tactile	grid/ bitmap	interactive/static
aural	continuous	aural	N/A	both
braille	continuous	tactile	grid	both
emboss	paged	tactile	grid	both
handheld	both	visual	both	both
print	paged	visual	bitmap	static
projection	paged	visual	bitmap	static
screen	continuous	visual	bitmap	both
tty	continuous	visual	grid	both
tv	both	visual, aural	bitmap	both