

# Markup Languages

## What is markup?

SGML – The Origin of the <TAG> syntax

HTML – HyperText Markup Language

CSS – Cascading Style Sheets

XML – eXtensible Markup Language

2007-01-26

Jonas Kvarnström

2  
2007-01-26

## Markup 1: The Need for Markup

- **Plain text** (ASCII, Unicode) is often not enough
  - Need structure – headings, bullets, ...
  - Need style information – font, size, color, ...
  - Need to incorporate other objects – images, ...
- Solution: Use **markup**
  - Additional data or instructions included inside the text, intended to be interpreted by a program

## Markup 2: Procedural Markup

- In the beginning: **Procedural markup**
  - Tell the system what to do
    - **.SK 1**  
This added information, called "markup", serves two purposes:
      - **.TB 4**
      - **.OF 4**
      - **.SK 1**  
1.#Separating the logical elements of the document; and
      - **.OF 4**
      - **.SK 1**  
2.#Specifying the processing functions to be performed on those elements.
  - Easy to process – but difficult to change

Tab stop 4  
Offset 4  
Skip vertical space, 1 unit  
<explicitly numbered list>

## Markup 3: Structural Markup

4  
2007-01-26

- Eventually (late 60s): The idea of **structural markup**
  - Markup describes **what the text is**
    - This is a heading, this is a list, ...
    - `<h2>`Heading</h2>
    - `<ol>`
    - `<li>`Separating the logical elements of the document; and</li>
    - `<li>`Specifying the processing function to be performed on those elements.</li>
  - Does not specify how to display this
    - Instead: Specified by the document processor or a style sheet
    - "Level 2 headings should be in 18-pt Myriad Pro Black"
    - "Ordered lists should be indented by 10 pt and ..."
  - Don't mix **content** with **presentation**

Heading, level 2  
Ordered list  
List item start ...  
... and end  
List item start ...  
... and end  
End of list

## Markup 4: Uses

5  
2007-01-26

- Structural markup is not just for word processing
  - Also for what you might call **data structures / databases**
    - `<book>` `<title>`Thinking in Java</title>
    - `<author>`Bruce Eckel</author>
    - `<isbn>`...</isbn>
  - For **mathematical formulas** (MathML)...
    - `<apply>` `<power/>`
    - `<apply>` `<plus/>` `<ci>a</ci>` `<ci>b</ci>` `</apply>`
    - `<cn>2</cn>`
    - `</apply>`
  - For **music** (SMDL)...
    - `<ces id='ces4'>`
    - `<pitched exspec=exlist1>``<nompitch>``<gampitch>``<pitchnm>c</pitched>`
    - `<rest exspec=exlist1>``</rest>`
    - `</ces>`
  - ... for just about anything!

In some cases, this isn't really "markup" – the tags are the real information ...

## Markup 5: Common Structure?

6  
2007-01-26

- A **single markup language** cannot cover everything
- But a common structure has many benefits
  - Easier to learn similar languages
  - Common tools can be used (to some extent) – parsers, ...

## SGML 1: DTD, Tags

7  
2007-01-26

- **SGML** – Standard Generalized Markup Language
  - Meta-markup language from early 80's
    - A language in which you can define a markup language
    - Language definition is called DTD: Document Type Definition
    - Predecessor of XML – most of this applies to XML too!
  - Example DTD (extremely simple), defining four tags:
    - `<!ELEMENT note (from,to,text)>`
    - `<!ELEMENT from (#PCDATA)>`
    - `<!ELEMENT to (#PCDATA)>`
    - `<!ELEMENT text (#PCDATA)>`
  - Example document:
    - `<note>`
    - `<from>`Jonas</from>
    - `<to>`Anyone who is still listening</to>
    - `<text>`Don't fall asleep yet!</text>
    - `</note>`

Can also define complex patterns for tag nesting.  
Can define whether end tags are required, and so on.

#PCDATA: Parsed Character Data – can contain other tags

## SGML 2: Tag Attributes

8  
2007-01-26

- Elements (tags) can have **attributes**
  - #REQUIRED, #IMPLIED or #CURRENT (repeat previous value)
    - `<!ELEMENT image EMPTY>`
    - `<!ATTLIST image id CDATA #IMPLIED>`
    - `<!ATTLIST image height CDATA #REQUIRED>`
    - `<!ATTLIST image width CDATA #REQUIRED>`
    - `<image height="100" width="200">`
  - Enumerated (list of possible values)
    - `<!ELEMENT task (#PCDATA)>`
    - `<!ATTLIST task status (important|normal) #REQUIRED>`
    - `<!-- This is a comment -->`
    - `<task status="important">...</task>`

EMPTY: Nothing inside  
<image></image>

CDATA: Character Data (string data type)

Embed comments in  
<!-- ... --> (2 hyphens)

## SGML 3: Entities

9  
2007-01-26

- **Entities** can be used as SGML "macros"
  - Declared in the DTD
    - `<!ENTITY coursecode "TDDI48">`
    - `<!ENTITY teacher "Jonas Kvarnström">`
    - `<!ENTITY chapterTwo SYSTEM "chapter2.txt">`
  - Used in documents
    - Ampersand / entity name / semicolon
    - The course `&coursecode`; is taught by `&teacher`.
  - Can be used for single characters too
    - `<!ENTITY ouml "ö">`
    - `<!ENTITY eacute "é">`
    - `<!ENTITY lt "&#60;">`
  - (Also other kinds of entities)

## SGML 4: History

jonkv@ida  
10  
2007-01-26

- SGML **history**:
  - Was quickly adopted by government
    - US DoD
    - European defence, transport, energy, and space agencies
    - ...
  - Beyond this, not very widely spread...
    - Too few tools
    - Too flexible – too difficult to implement
    - Not much support from major software vendors
  - ... until HTML was developed.

See also <http://www.oasis-open.org/cover/gentle.html>

## HTML 1: Early History

jonkv@ida  
12  
2007-01-26

- **Hypertext** was an old concept even in 1989
  - Text where some words/phrases are links to other texts
  - Invented by Ted Nelson in 1968
- **HTML 1**: First developed by Tim Berners-Lee in 1989-91
  - HEP (High Energy Physics) / CERN (nuclear research)
  - Simple language defined in SGML
    - Few tags
    - Easy to use
  - First browser: "WorldWideWeb"
    - Used SGML tools
    - Wrote a simple DTD for HTML
    - Modified a text control for hyperlinks
    - Showed images in separate windows

Screenshot from 1993



## HTML 2: 1993–1994

jonkv@ida  
13  
2007-01-26

- **HTML 2**: Images and forms (1993–1994)
  - Mosaic: First browser with integrated text and graphics
    - Developed at NCSA (National Center for Supercomputing Applications)
  - Number of web servers explodes **Mosaic in 1994 (?)**
    - 50 in dec 1992
    - 623 in dec 1993
    - 10022 in dec 1994
    - 74700 in dec 1995
    - 603367 in dec 1996
    - 3689227 in dec 1998
    - 104944594 in dec 2006
  - 13 Oct 1994:
    - Mosaic Netscape 0.9
  - <http://www.zakon.org/robert/internet/timeline/#Growth>



## HTML 3: Chaos?

jonkv@ida  
14  
2007-01-26

- **1995-1997**: The beginning of chaos
  - Competition: Internet Explorer, Netscape Navigator, others
    - No time to think, no time to wait for standardization
    - Each browser adds its own tags, and only implements *parts* of HTML 3
    - Each browser interprets some tags slightly differently
    - Each browser treats malformed tags differently:
      - The web is full of documents that "look just fine on my computer!"  
<p><em>IMPORTANT</em></p></em>
  - Web page **design** is seen as important
    - HTML was not made for this – markup was supposed to be **structural**
  - Web designers use all kinds of tricks for pixel-"perfect" layout
    - <font size="8px">foo</font> -- but anything below 16 is unreadable
    - Best viewed in Internet Explorer 4.0 at 800x600, 16 bpp.

## HTML 4: New Standards

jonkv@ida  
15  
2007-01-26

- Now, the situation is **improving**
  - HTML 4.01 and XHTML are well-defined standards
    - By the World Wide Web Consortium, <http://www.w3.org>
  - New style markup standards separate style from structure
    - CSS (Cascading Style Sheets)
  - Browsers are becoming more standards-compliant
    - Firefox, Opera, even Internet Explorer from version 7.0
  - Automated page validation services exist
    - <http://validator.w3.org/>

## HTML 5: A Basic HTML Page

jonkv@ida  
16  
2007-01-26

- A basic HTML page: An old TDDI48 lab
  - <!DOCTYPE html public "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
  - <html>
  - <head><title>TDDI48 Lab 1: Introduction to Java</title>
  - <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
  - <link rel="stylesheet" HREF="tddi48.css" TYPE="text/css">
  - </head>
  - <body>
  - <h1 class="labheader first">TDDI48/TDDB87 Lab 1: Introduction to the Lab Environment</h1>
  - <p class="author">Jonas Kvarnström...</body></html>



## HTML 6: The General Structure

jonkv@ida  
17  
2007-01-26

- The **DOCTYPE** declares that this is HTML
  - In our case, HTML 4.01
  - **Strict** version (only structural tags – use CSS for presentation!)
    - <!DOCTYPE html public "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
  - **Transitional** (use presentation tags sparingly!)
    - <!DOCTYPE html public "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
- All documents should have a DOCTYPE!
  - Without this, browsers don't know how to interpret your doc
    - Is it HTML 2, 3, 4, ...?
    - Does it rely on Netscape-specific bugs? IE-specific bugs?
  - They enter *quirks mode* -- *guess* what you probably mean
    - With a doctype, they should "follow instructions" to the letter

## HTML 7: The General Structure

jonkv@ida  
18  
2007-01-26

- The rest of the document is enclosed in an <html> tag
  - The top element of the HTML DTD
    - Split into <head> and <body>
  - <html>
    - <head>
    - ... global information about the document ...
    - </head>
    - <body>
    - ... the actual contents of the document ...
    - </body>
    - </html>

## HTML 8: The Head

jonkv@ida  
20  
2007-01-26

- In the **head section**, you declare:
  - A title (usually shown in the window title bar)
    - `<title>TDDI48 Lab 1: Introduction to Java</title>`
  - Possibly some meta-information
    - `<meta name="description" content="First lab for TDDI48">`
    - `<meta name="keywords" content="Java, IDEA, lab, tutorial">`
  - Some meta tags correspond to HTML protocol headers
    - `<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">`
    - `<meta http-equiv="Content-Language" content="en-GB">`

## HTML 9: Special Links

jonkv@ida  
20  
2007-01-26

- Possibly some "special" links (also in the head)
  - `<link rel="Prev" href="sec4.html" title="Section 4">`
  - `<link rel="Next" href="sec6.html" title="6. Using the LINK tag">`
  - `<link rel="Alternate" href="sec5-swedish.html" title="5. Att använda META-taggar">`
  - `<link rel="Alternate" media="print" type="application/postscript" href="sec5.ps" title="Printable version (postscript)">`
  - `<link rel="Section" href="sec1.html" title="Introduction">`
  - `<link rel="Section" href="sec2.html" title="The BODY tag">`
  - `<link rel="Start" href="contents.html">`
  - `<link rel="Index" href="myindex.html">`
  - `<link rel="Contents" href="contents.htm">`
  - `<link rel="Glossary" href="glossary.htm">`



## HTML 10: The Body

jonkv@ida  
21  
2007-01-26

- The body of the document contains the text
  - Headers: `<h1>`, `<h2>`, `<h3>`, ...
    - `<h1 class="labheader">TDDI48 Lab 1: Introduction to Java</h1>`
  - Paragraphs: `<p>` (next `<p>`, `<h1>`, ... ends the paragraph)
    - `<p>` This is the first paragraph.
    - `<p>` This is the second paragraph.
    - Use `<br>` for a line break within a paragraph
  - Unnumbered lists: `<ul>`, `<li>`
    - `<ul>`
    - `<li>` `<p>` This is the first bullet.
    - `<li>` `<p>` This is the second bullet.
    - `</ul>`
  - Numbered lists: `<ol>`, `<li>`

### TDDI48 Lab 1: Introducti

This is the first paragraph.  
This is the second paragraph.

- This is the first bullet.
- This is the second bullet.

- Using `<ol>`- instead.
- Using `<ol>`- instead.

## HTML 11: Character styles

jonkv@ida  
22  
2007-01-26

- Explicit styling – obsolete, **use CSS instead!**
  - Italics: `<i>...</i>`
  - Bold: `<b>...</b>`
  - Underline: `<u>...</u>`
  - Fonts: `<font face="Arial,Helvetica">...</font>`
- Implicit styling (may be OK sometimes; **CSS is better**)
  - Emphasized (usually italics): `<em>`

## HTML 12: Tables

jonkv@ida  
23  
2007-01-26

- Tables:**
  - `<table cellpadding="4">`
    - `<thead>`
      - `<tr><th>Item</th><th>Price</th></tr>`
      - `</thead>`
      - `<tbody>`
        - `<tr><td>Foo</td><td>$100.00</td></tr>`
        - `<tr><td>Bar</td><td>$200.00</td></tr>`
        - `<tr><td>Gazonk</td><td>$300.00</td></tr>`
      - `</tbody>`
    - `</table>`
  - cellpadding = additional space within each cell

Item	Price
Foo	\$100.00
Bar	\$200.00
Gazonk	\$300.00

## HTML 13: Embedded images

jonkv@ida  
24  
2007-01-26

- Images:**
  - Use the `img` tag
    - ``
    - Attribute **longdesc** points to a long description: `longdesc="desc.html"`
    - Attribute **title** usually shown as tooltip: `title="Click here to go to ..."`
    - Specify **height** and **width** for faster layout: `height="100" width="200"`
    - align**="left" to let text flow on the right hand side of the image
    - align**="right"...
  - Or the more generic object tag
    - `<OBJECT data="image.png" type="image/png" ...>`
    - Shown if the object type is not supported
    - `</OBJECT>`

## HTML 14: Hyperlinks

jonkv@ida  
25  
2007-01-26

- Hypertext: **References** to other locations!
  - Inserting a link to another document:
    - There is an `<a href="index.html">index</a>`.
    - Read more at `<a href="http://java.sun.com">the Java web site</a>`.
  - Links within documents
    - First, place an anchor:  
`<a name="top"><h1>Contents</h1><a>`
    - Then, point to that anchor using "#" fragment syntax  
`<a href="#top">Go to the top</a>`
  - Links can contain images
    - `<a href="/"></a>`

Uses the URI fragment identifier syntax

## HTML 15: Character Entities, Comments

jonkv@ida  
26  
2007-01-26

- Standard set of **character entities** in HTML
  - Borrowed from SGML
  - Always on the form `&entity;` (don't forget the semicolon!)
  - These three should always be written as entities:
    - `&lt;` -- less than, "<"
    - `&gt;` -- greater than, ">"
    - `&amp;` -- ampersand, "&"
  - Many others that might not exist in your character encoding
    - `&acute;` -- é
    - `&ouml;` -- ö (but no need to encode this if you use iso-8859-1)
    - `&nbsp;` -- non-breaking space
- Comments** can be embedded into HTML
  - Use standard SGML comments: `<!-- comment -->`

## HTML 16: Standards

jonkv@ida  
27  
2007-01-26

- Please don't write **non-standard HTML!**
  - Use the standard
    - HTML 4.01: <http://www.w3.org/TR/html4>
  - Include a DOCTYPE
    - Strict version (no presentation – use CSS!):  
`<!DOCTYPE html public "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">`
    - Transitional (use CSS with possibly a few presentation tags):  
`<!DOCTYPE html public "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">`
  - Validate your documents
    - HTML validator: <http://validator.w3.org/>

# CSS

## Cascading Style Sheets

2007-01-26

Jonas Kvarnström

## CSS motivation 1

jonkv@ida  
29  
2007-01-26

- How should we define **the look of a web page**?
  - Old-style HTML: Embedded within the document
    - `<h2><font face="Arial, Helvetica" size="32"><b>Chapter 1</b></font></h2>`
    - `<!-- spacing image -->`
    - `<p><font face="Arial, Helvetica" size="12">Intro</font></p>`
    - `<p><font face="Times New Roman" size="11">First paragraph of the main text</font></p>`
    - `<p><font face="Times New Roman" size="11">Second paragraph</font><font face="Courier New, monospaced" size="10" color="green">some code</font></code> and more text</font></p>`
  - Disadvantages:
    - Cluttered with repetitive style information – hard to read, large
    - Consistency is hard to achieve
    - Want to change chapter style? Search and replace throughout the document – or throughout the website!

## CSS motivation 2

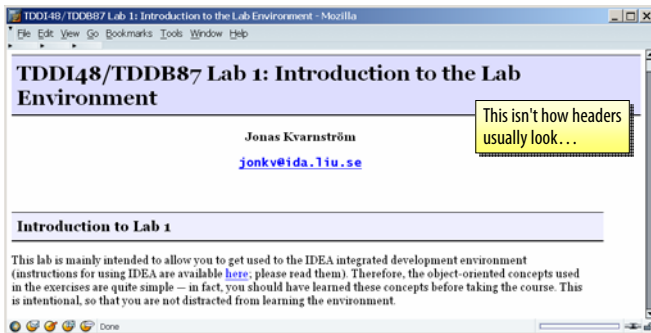
jonkv@ida  
30  
2007-01-26

- How should styles be defined?
  - Modern HTML: Use CSS – **Cascading Style Sheets**
    - `<h2>Chapter 1</h2>`
    - `<p>Intro</p>`
    - `<p class="intro">First paragraph of the main text</p>`
    - `<p>Second paragraph</code>some code</code> and more text</p>`
    - `H2 { font-family: Arial, helvetica; font-size: 32px; font-weight: bold; }`
    - `H2 { margin-bottom: 28px; }`
    - `P { font-family: Times New Roman; font-size: 11px; }`
    - `P.intro { font-family: Arial, Helvetica; font-size: 12px; }`
    - `CODE { font-family: Courier New, monospaced; font-size: 10px; color: green; }`
  - Advantages:
    - Document is cleaner, easier to read
    - Semantic** markup: "this is an intro paragraph", not "this has font x"
    - Easy to change styles in a central location
    - Document is smaller; also, CSS style file downloaded/cached separately

## CSS 1: Example

jonkv@ida  
31  
2007-01-26

- `<h1 class="labheader first">TDDI48/TDDB87 Lab 1: Intro...</h1>`
- `<P class="author">Jonas Kvarnström<BR><BR><a href="mailto:jonkv@ida.liu.se">a ...</a></P>`
- `<h2 class="labheader">Introduction to Lab 1</h2>`



## CSS 2: Selectors

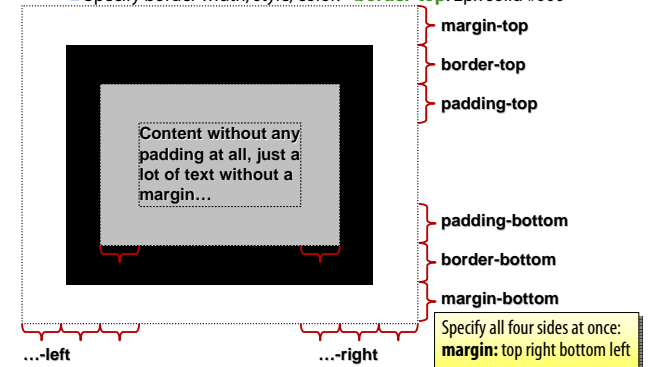
jonkv@ida  
32  
2007-01-26

- CSS **Selectors**: What items do a style apply to?
  - Specify a style for all tags of a certain kind
    - CSS: `H1 { ... }`
    - HTML: `<H1>My heading</H1>`
  - Specify a style for specific classes of tags
    - CSS: `P.idea { ... }`
    - HTML: `<P class="idea">IDEA-specific instructions here</P>`
  - Specify a style for a specific element, given a unique ID
    - CSS: `DIV#exercise4`
    - HTML: `<DIV id="exercise4">...</DIV>`

## CSS 3: Layout

jonkv@ida  
33  
2007-01-26

- Layout**: Margins, borders and padding
  - Specify border width, style, color: `"border-top: 2px solid #000"`



## CSS 4: Measurements

jonkv@ida  
34  
2007-01-26

- Units of measurement** for layout:
  - 1 px = pixel (can be rescaled for high-resolution devices)
  - 1 pt = point (1/72 inch)
  - 1 mm = millimeter
  - 1 cm = centimeter
  - 1 in = inch
  - 1 ex
  - 1 em  $\left\{ \begin{matrix} M \\ X \end{matrix} \right\}$  1 ex
- Also: % (percent of some base value)
- Also used for **width** and **height**

## CSS 5: Colors

jonkv@ida  
35  
2007-01-26

- Color** specifications:
  - Named colors
    - aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, white, yellow
  - RGB colors
    - `#f00`
    - `#ff0000`
    - `rgb(255,0,0)`
    - `rgb(100%,0%,0%)`

## CSS 6: Heading Example Revisited

jonkv@ida  
36  
2007-01-26

- Back to the example...**
  - `<h1 class="labheader">TDDI48/TDDB87...</h1>`
    - `H1.labheader, H2.labheader { margin-top: 3em; /* Except for "first" items */ border-top: 1px solid #000; border-bottom: 2px solid #000; padding: 8px; text-align: left; font-weight: bold; }`
    - `H1.labheader { font-size: 180%; background-color: #ddddff; color: #000 }`
    - `H2.labheader { font-size: 120%; background-color: #eeeeff; color: #000; width: 42em }`
    - `H1.labheader.first { margin-top: 0em } /* More specific! */`

TDDI48/TDDB87 Lab 1: Introduction to the Lab Environment

## CSS 7: Another Example

jonkv@ida  
37  
2007-01-26

- **Alter the behaviour** of existing tags
  - Make the CODE tag blue
  - Make sure there are never line breaks in CODE
    - `<p>`  
Implement a concrete `<code>draw()</code>` method that prints "A circle is drawn".
    - `CODE` {  
font-family: monospace;  
white-space: nowrap;  
color: #4669ad  
}

Implement a concrete `draw()` method that prints "A circle is drawn".

## CSS 8: Readability

jonkv@ida  
38  
2007-01-26

- Change styles for **better readability**
  - Example: Course web page
  - Paragraphs within the "main" table: Max width 45 em
    - `TABLE#main P { max-width: 45em; }`
    - Not supported by IE 6 (but present in IE 7)
  - Line spacing: Slightly more (130%)
    - `TABLE#main P { line-spacing: 130%; }`

## CSS 8: SPAN and DIV

jonkv@ida  
39  
2007-01-26

- SPAN and DIV lets you (almost) **generate new tags**
  - DIV creates a new block – can contain paragraphs
  - SPAN is used within paragraphs
    - First create the package.  
Right-click the source path in the project pane (if it is not visible, select `<span class="menu">Window | Project: Alt-1</span>`). Select `<span class="menu">New | Package</span>` and enter "se.liu.ida.<em>youremail</em>".
    - `SPAN.menu` { color: #46ad69; font-family: monospace; }  
`SPAN.menu:before` { content: "[[" }  
`SPAN.menu:after` { content: "]]" }

First create the package. Right-click the source path in the project pane (if it is not visible, select `[[window | Project: Alt-1]]`). Select `[[New | Package]]` and enter "se.liu.ida.youremail".

## CSS 9: SPAN and DIV are useful!

jonkv@ida  
40  
2007-01-26

- **Use** SPAN and DIV!
  - Do not use inline formatting
  - Instead, create new classes for all relevant items
    - `SPAN.label { ... }`
    - `SPAN.author { ... }`
    - `SPAN.subject { ... }`
    - `SPAN.email { ... }`
    - `DIV.message { ... }`
  - Don't say "this should be bold"
    - Use semantic markup with indirection
    - Say "This is a subject" [HTML]
    - Say "Subjects should be bold" [CSS]

## CSS 10: Setting Page Colors and Attributes

jonkv@ida  
41  
2007-01-26

- Altering the page body and link colors:
  - `BODY` {  
background: url("http://www.ida.liu.se/~jonkv/background.png")  
/\* background: white; \*/  
color: black; /\* for text \*/  
}
  - `A:link` { color: #0000ff }  
`A:visited` { color: #a669ad }  
`A:active` { ... }
  - `A:external:link` { color: #0000ff }  
`<A class="external" href="http://...">Link to external site</a>`

## CSS 11: Different Media

jonkv@ida  
42  
2007-01-26

- Different rules can be applied to different **media**
  - aural, braille, embossed, handheld, print, projection, screen, tty, tv
    - @media print {  
`BODY` {  
font-size: 10.5pt;  
font-family: book antiqua, serif;  
}  
`PRE` { font-size: 80%; }  
`P, LI, DD` {  
orphans: 2;  
widows: 2;  
max-width: 13cm;  
}

## CSS 12: External or Inline

jonkv@ida  
43  
2007-01-26

- Style sheets can be **external** or **inline**
  - Internal – easier for small pages, if you're lazy
    - `<head>`  
`<STYLE type="text/css">`  
`A.maplink { text-decoration: none }`  
`</STYLE>`  
`</head>`
  - External
    - `<head>`  
`<link rel="stylesheet"`  
`HREF="standard.css" TYPE="text/css" title="Standard">`  
`</head>`
    - Can be reused between different documents
    - Can be cached separately by the browser
    - Often better authoring support ("CSS mode", not "CSS in HTML-mode")

## CSS 13: Alternate Style Sheets

jonkv@ida  
44  
2007-01-26

- **Alternate style sheets** can be specified
  - User agent should let you choose using menus
    - `<link rel="stylesheet"`  
`HREF="standard.css" TYPE="text/css" title="Standard">`
    - `<link rel="alternate stylesheet"`  
`HREF="large.css" TYPE="text/css" title="Large fonts">`
    - `<link rel="alternate stylesheet"`  
`HREF="bw.css" TYPE="text/css" title="Black and white">`



## CSS 14: Standards

jonkv@ida  
45  
2007-01-26

- **Use CSS** in your documents!
  - Write reusable, well-commented CSS
  - Use the standard
    - CSS 2: <http://www.w3.org/TR/REC-CSS2/>
  - Validate your documents
    - <http://jigsaw.w3.org/css-validator/>
  - Test your documents in different browsers
    - Browsers are buggy (especially CSS in IE 6)
    - Browsers only support part of CSS
  - Test your documents in a text-only browser
    - Use "links" or "lynx", for example – important for accessibility
  - Test your documents in low, medium and high resolution
    - No hardcoded tiny / huge font sizes...
  - Test at different zoom levels (Firefox: Ctrl-plus / Ctrl-minus)

# XML: eXtensible Markup Language

## A brief introduction

2007-01-26

Jonas Kvarnström

jonkv@ida  
47  
2007-01-26

## XML 1: eXtensible Markup Language

- **SGML is complex**
  - Difficult to implement in its entirety
  - Takes a long time to understand all features
  - Many features that are (almost) never used
- **XML: SGML light** (almost)
  - Many options removed, fixed to one setting
  - Started 1996; standard 1998

## XML 2: Applications

jonkv@ida  
48  
2007-01-26

- Like SGML, many applications exist
  - **CML**: Chemical Markup Language
    - `<chem> <molecule>`
      - `<atom n="2" charge="+1"> H </atom>`
      - `<atom charge="-2"> O </atom>`
    - `</molecule> </chem>`
  - **BSML**: Bioinformatic Sequence Markup Language
  - **XHTML**: A stricter version of HTML
    - Subset of HTML 4, with stricter syntax (end tags required, ...)
  - Plain **data storage**
    - Configuration files
    - Simple databases
    - Data exchange over the net
    - ...

## XML 3: DTD and Schema Examples

jonkv@ida  
49  
2007-01-26

- Document type specified using **DTD or XML schema**
  - Not required -- useful for data validation
  - Example XML document:
    - `<?xml version="1.0"?"`
    - `<text timestamp="08:45:00.000">`
    - The deadline of `<name>`homework 1`</name>` is `<emph>`March 9th 2137`</emph>`.
    - `</text>`
  - Example DTD:
    - `<!ELEMENT text (#PCDATA | emph | name)*>`
    - `<!ATTLIST text`
    - `timestamp NMTOKEN #REQUIRED>`
    - Elements, subelements and attributes; ordering constraints
    - Strings and tokens – no other data types
  - Example XML schema: Next slide

## XML 4: DTD and Schema Examples

jonkv@ida  
50  
2007-01-26

- Example XML document:
  - `<?xml version="1.0"?"`
  - `<text timestamp="08:45:00.000">`
  - The deadline of `<name>`homework 1`</name>` is `<emph>`March 9th 2137`</emph>`.
  - `</text>`
- Corresponding XML Schema:
  - Stronger constraints and type system (integers, dates, user-defined, ...)
  - `<xsd:element name="text">`
  - `<xsd:complexType mixed="true">`
  - `<xsd:sequence>`
  - `<xsd:element ref="name"/>` `<xsd:element ref="emph"/>`
  - `</xsd:sequence>`
  - `<xsd:attribute name="timestamp" type="xsd:date"`
  - `use="required"/>`
  - `</xsd:complexType>`
  - `</xsd:element>`

## XML 5: Namespaces

jonkv@ida  
51  
2007-01-26

- **Namespaces** for short / unique identification
  - Like packages in Java
    - `<?xml version="1.0"?"`
    - `<!-- all elements here are explicitly in the HTML namespace -->`
    - `<html:html xmlns:html="http://www.w3.org/TR/REC-html40">`
    - `<html:head>`
    - `<html:title>`Frobnostration`</html:title>`
    - `</html:head>`
    - `<html:body>`
    - `<html:p>`Moved to
    - `<html:a href="http://frob.com">`here.`</html:p>`
    - `</html:body>`
    - `</html:html>`

Defines "html:" namespace prefix for this tag and its contents

URI is only used as an identifier – does not have to point to a retrievable resource!

## XML 6: Namespaces

jonkv@ida  
52  
2007-01-26

- You can mix multiple namespaces
  - `<?xml version="1.0"?"`
  - `<!-- both namespace prefixes are available throughout -->`
  - `<bk:book xmlns:bk="urn:loc.gov:books"`
  - `xmlns:isbn="urn:ISBN:0-395-36341-6">`
  - `<bk:title>`Cheaper by the Dozen`</bk:title>`
  - `<isbn:number>`1568491379`</isbn:number>`
  - `</bk:book>`
- You can define your own namespace...
  - `<ida:course xmlns:ida="http://www.ida.liu.se/courseXML">`
  - `<ida:teacher>`Jonas Kvarnström`</ida:teacher>`
  - `<ida:name>`...`</ida:name>`
  - `<othertag>`...`</othertag>`
  - `</ida:course>`

## XML 7: SAX and the DOM

jonkv@ida  
53  
2007-01-26

- Two standard APIs for **parsing XML**
  - (And HTML, to some extent)
    - **SAX**: Simple API for XML Parsing
    - **DOM**: Document Object Model
  - More about this in another lecture!