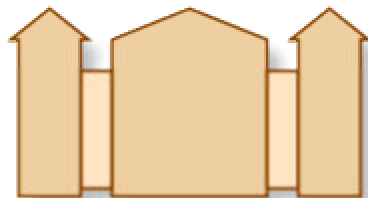


# Structured Documents on the Web

Jacco van Ossenbruggen  
CWI Amsterdam



# Talk overview

- Introduction and historical background
- Multiple delivery publishing (MDP)
- MDP on the Web: Style sheets
- Conclusion

# What is a “Document”?

Examples:

- Book, poem
- Article, paper, report
- Memo, e-mail, letter, etc

My definition:



A document is a self-contained unit of information, intended to be communicated to a human interpreter

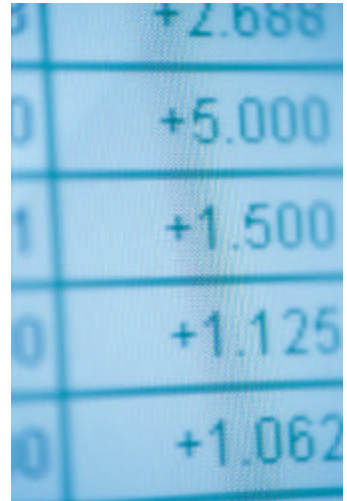
# What isn't a document?

All data that is:

- Fragmentary
- Intended solely for further machine processing

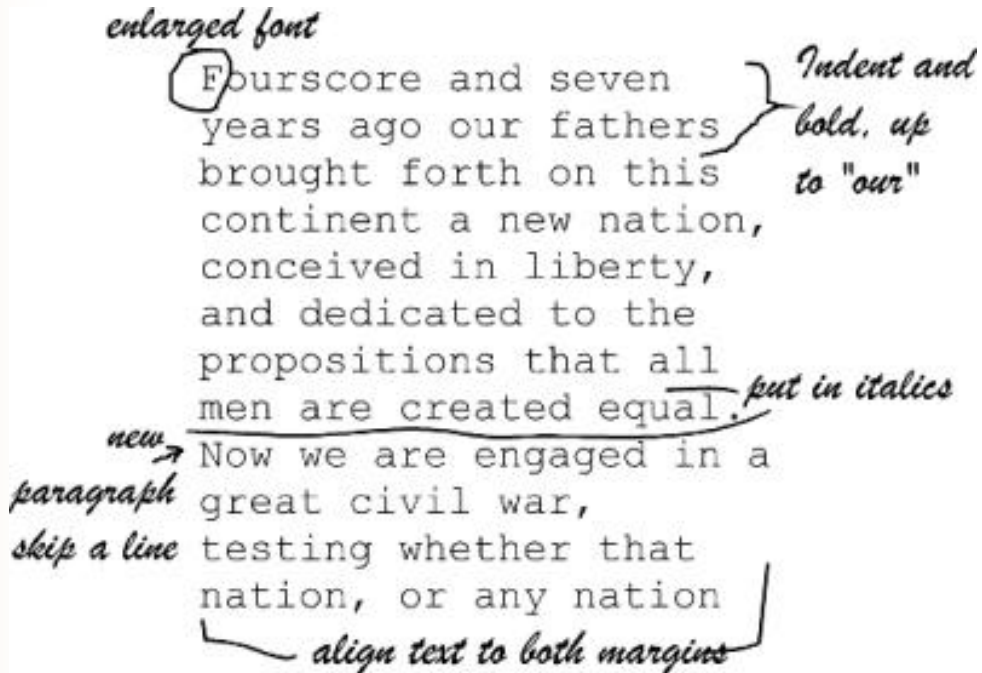
Examples:

- Database records
- HTTP requests
- Software source code
- ...



	+2.688
0	+5.000
1	+1.500
0	+1.125
0	+1.062

# What is Markup?



# Talk overview

- Introduction and historical background
- Multiple delivery publishing (MDP)
- MDP on the Web: Style sheets
- Conclusion

# Electronic Documents (then)

- Goal (authoring/production):
  - More efficient/effective production by using WYSIWYG authoring interfaces (WP, DTP)
- Goal (final-form):
  - Obtain same typographic quality as traditional print
- Production electronic, dissemination and final-form still on paper
- Authoring & storage format:
  - Mimics final-form presentation format



# Electronic Documents (now)

- Goal (authoring/production):
  - Efficient, industrial scale, full document life cycle
- Goal (final-form):
  - Improve communication by exploiting presentation potential of new media
    - Use of audio, video, animation, etc
    - Interactivity (hyperlinks, forms, etc.)
    - Dissemination over internet (WWW)
    - Use of document technology to access (legacy) information
- Both production & dissemination is electronic
- Authoring & storage format:
  - Differs radically from presentation format



# Electronic Documents: Issues

Problem: many document formats cannot cope with changing environment (c.f. issues in software engineering)

- Hardware dependencies (use of printer/typesetter specific control sequences)
- Software dependencies (use of proprietary formats)
- Presentation dependencies (layout and style)

Related issues:

- Longevity (many documents need to last >30 years)
- Maintenance & reuse
- Flexibility & tailorability

# “Solution”

(Semi-automatically) convert all documents to new format or new layout

- Expensive
- Time consuming
- Error prone (& pretty boring too!)
- Loss of (implicit) information

# Real solution

Multiple delivery publishing model

# Multiple delivery publishing (MDP)

- MDP distinguishes two formats
  - One for authoring and long term storage
  - Another one for final-form presentation
- Mappings from source to target format
- Source format can now abstract from all details that are likely to change in the target
- Sounds pretty straightforward eh?
- But it actually meant...

# Revolution!

## Software developers

No longer control their application's own file format

## Document authors

No longer control style and layout of their documents

## Tools

No longer used the "sacred" WYSIWYG paradigm

Multiple delivery publishing was not obvious at all!



# MDP: Nothing new ...

- This approach was already advocated by Goldfarb et al. in the 70's!
- Source documents encoded using IBM's Generic Markup Language (GML)
- GML was standardized by ISO in 1986 as SGML

# MDP & SGML

- MDP and SGML remained highly controversial
  - People do not like to give up control or change the way they work
  - MDP could not always match the output quality of traditional tools
  - MDP is no silver bullet!
  - Primarily suited for content-driven applications
  - Not for layout-driven applications
- SGML standard is extremely complex
  - Still not fully implemented
  - Huge and inflexible
  - Mainly used in academic and large organizations

# “SGML” revival due to the Web

- HTML already is an application of SGML  
(*eh... sort of*)
- XML is a stream-lined and simplified subset of SGML (*it really is, this time*)
- Published in 1998, XML already had more applications that year than SGML ever had!



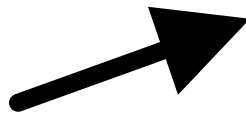
# Talk overview

- Introduction and historical background
- Multiple delivery publishing (MDP)
- MDP on the Web: Style sheets
- Conclusion

# MDP: easy reuse of source document

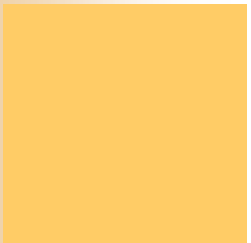


source document

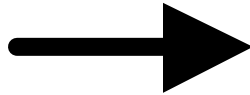


target presentations

# MDP: easy reuse of style specification



source document



target presentations

# MDP: Document design dimensions:

- Content versus markup
  - what is in the tags, what is between the tags?
- Embedded versus external markup
  - What is encoded in the same file, what is stored elsewhere?
- Declarative versus procedural
  - Specify what or specify how
- Domain independent versus domain specific
  - <title> or <product-shelf-number>?
- Layout-driven versus content-driven applications
  - magazine cover or technical manual?
- Visual markup versus structured markup
  - <i> or <emph>?

# Source vs. presentation format

- Source format:
  - Structured, declarative markup
  - Can be domain independent but...
  - ...is usually tailored to a specific domain
  - Provide sufficiently rich structure for style sheets and other processing
- Presentation format:
  - Visual, often procedural markup
  - Can be platform/medium independent but...
  - ... is usually tailored to a specific output medium/device
  - Provide sufficient information to obtain high quality output
- How do you classify your favorite document format?

# Domain independent vs. domain specific

## Domain independent:

- Examples: HTML, Docbook, (LaTeX)
- Wide deployment: easy to learn, many (cots) tools available
- Poor semantics for automatic processing other than presentation
- Tools only need to deal with predefined markup semantics

## Domain specific:

- Examples: product specific documents standards (e.g. automobile and aircraft industry)
- Users need training, tailor-made tools might need to be developed
- Rich (domain-specific) semantics for further processing (retrieval, screen scraping etc.)
- Need tools tailored to domain-specific document formats or ...

# Presentation of domain specific document formats

- *Generic* tools that can process *user*-defined markup
  - Software adapts to document structure
- No predefined (presentation) semantics
  - Also need to be user-defined

# Beyond presentation semantics

- Document-oriented semantics
  - static: style and layout  
(e.g. style sheets, focus second half of this talk)
  - dynamic: scheduling & animation (*see seminar 10/12/2002*)
  - interaction: linking & forms
- Other semantics:
  - do not describe the document,  
but the *domain* of the document's *content*
  - can still be related to document
    - annotations & meta data
  - RDF(S), OWL, DAML+OIL, etc. (*see seminar 28/1/2003*)



# Talk overview

- Introduction and historical background
- Multiple delivery publishing (MDP)
- MDP on the Web: Style sheets
- Conclusion

# Multiple delivery publishing on the Web

Bloodtype Function	W3C/HTML		
Markup	HTML		
Style	CSS		
Linking	<a href=		
Addressing	<a name		

# Multiple delivery publishing on the Web

Bloodtype Function	W3C/HTML		ISO/SGML
Markup	HTML		SGML
Style	CSS		DSSSL
Linking	<a href=		HyTime, TEI
Addressing	<a name		HyTime, TEI

# Multiple delivery publishing on the Web

Bloodtype Function	W3C/HTML	W3C/XML	ISO/SGML
Markup	HTML	XML	SGML
Style	CSS	CSS, XSLT, XSL FO	DSSSL
Linking	<a href=	XLink, (HLink)	HyTime, TEI
Addressing	<a name	XPath, XPointer	HyTime, TEI

# Style sheets: HTML & CSS

HTML with embedded visual markup:

```
<h3 align="center">  
  <font color="black">  
    The Need for Style Sheets  
  </font>  
</h3>
```

versus HTML with separate CSS style sheet:

HTML:

```
<h3>The Need for Style Sheets</h3>
```

CSS (optional!):

```
h3 { text-align: center; color: black }
```

# Style sheets: XML & CSS

- Example fragment using MyOwnML (XML):

```
<product>
  <type>X112332</type>
  <color>dark blue</color>
  ...
</product>
```

- With XML, your style sheet needs to specify more than just the style (CSS2):

```
product { display: list-item; ...}
type    { display: none; ...}
color   { display: block; ...}
```

# Style sheets: XML & CSS

- With XML, style sheets are no longer optional
- Information presented with CSS remains in the same order
- Source tree and target tree have similar structure (allows cascading)
- Style properties are inherited via the source tree (!)

# Transformations: XML and XSLT

- What if the desired target tree differs radically from the source tree?
  - assigning CSS properties will not suffice
  - need a language to describe XML (tree) transformations:
- XSL Transformations (XSLT)
  - more on XSL later!
  - XSLT transforms from XML to:
    - XML (including XHTML)
    - HTML (for legacy browsers, outputs “old” SGML syntax)
    - plain text (can be used to generate other text formats such as RTF, BibTeX, ...)



# Transformations: XML and XSLT

- XSLT itself also uses XML syntax (unlike CSS ...)
  - so you can transform XSLT using XSLT...
  - ... but it doesn't look really human friendly!
- The structure of the target tree and source tree can differ (unlike CSS):
  - XSLT style sheets can be chained, not cascaded

# XSLT template rules

- Transformations are described as a set of one or more template rules
- Each template rule consists of two parts:
  - A pattern that is matched against the source tree: the selector
  - A template to be filled in and added to the result tree
- XSLT selectors are based on XPath, e.g:

<code>-product</code>	<code>/product</code>
<code>-color type</code>	<code>product/color</code>
<code>-catalog//product</code>	<code>text()</code>
<code>-id("w11")</code>	<code>product[1]</code>
<code>-@class</code>	<code>/ * @*</code>

# XSLT: Example (I)

A single template rule may be sufficient...

```
<xsl:template match="/">
  <html>
    <head>
      <title>Product Report Summary</title>
    </head><body>
      <p>...<table>
        <tr><td>
          <xsl:value-of select="product/type"/>
        </td><td>
          <xsl:value-of select="product/color"/>
        <td> ... </tr></table>...</body></html>
    </xsl:template>
```

# XSLT: Example (II)

... or a style sheet can contain many (smaller) template rules

```
<xsl:template match="/">
  <table>
    <xsl:apply-templates/>
  </table>
</xsl:template>

<xsl:template match="product">
  <tr>
    <xsl:apply-templates/>
  </tr>
</xsl:template>

<xsl:template match="color|type">
  <td>
    <xsl:apply-templates/>
  </td>
</xsl:template>

...
```

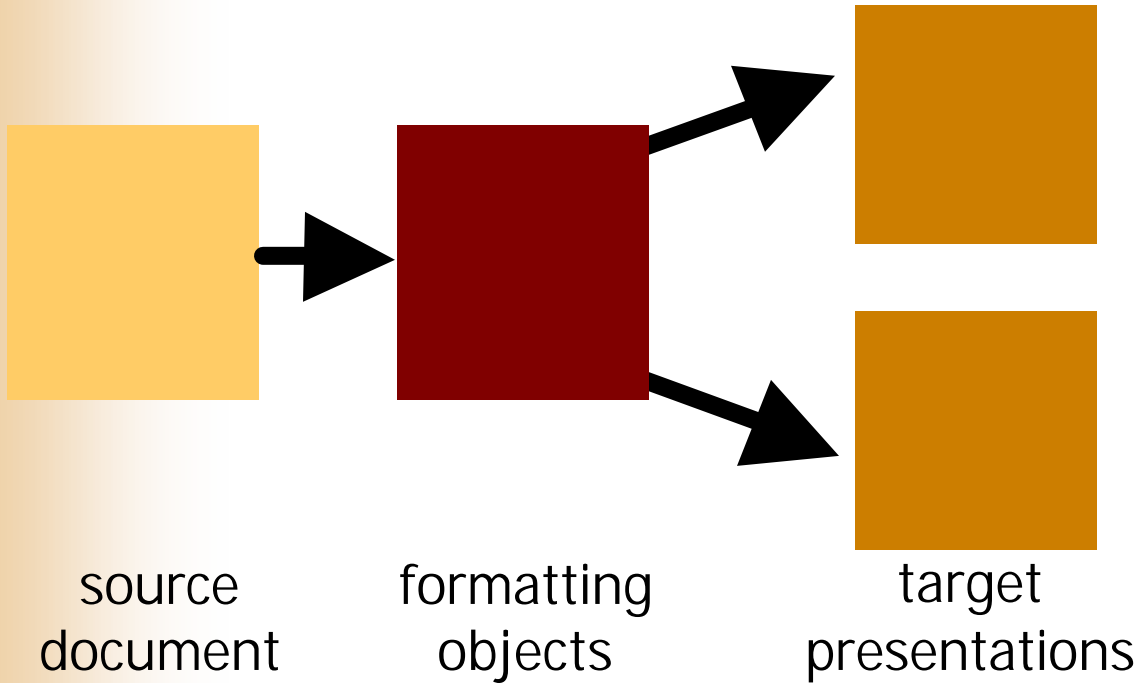
# Style sheets: Formatting objects (I)

- All these style sheet examples actually do two things:
  - specify how an XML document should be presented
  - specify how that presentation should be encoded in HTML
- Drawbacks:
  - need to start all over again for target formats other than HTML
  - limited by the presentation capabilities of HTML & CSS

# Style sheets: Formatting objects (II)

- Solution:
  - design new target language (argh!)
  - a language that is designed to describe formatting semantics
  - such a language is called a formatting vocabulary
  - elements in the language are called formatting objects (FO)
- Example: the formatting vocabulary defined by XSL
  - fo:block, fo:flow, fo:footnote, fo:external-graphic, fo:page-sequence
- XSL well suited for on-line and paper-based formatting beyond HTML

# Style sheets: Formatting objects (III)



# Style sheets: Formatting objects (IV)

- Advantages:

- Style sheets can be independent from final-form presentation format
- Formatting objects have more advanced formatting semantics than HTML/CSS

- Disadvantages

- Yet another layer of abstraction
- Relative little tool support (XSL became a W3C Recommendation on 15 October 2001)
- XSL FOs are not suited for all output media (SMIL, SVG etc.)



# MDP wrap up: pros & cons

- Advantages:
  - Longevity
  - Reusability
  - Flexibility & Tailorability
- Disadvantages:
  - Complexity
  - High dependency on tools (?!)
  - Training
  - High Initial investment
- Works best for content-driven material
  - becomes cheaper due to massive use on the Web
  - free tool support
    - XML parsers/browsers, XSLT engines, XSL FO formatters, etc.
  - many “off-the-shelf” source & target formats to choose from
    - XHTML, SVG, SMIL, MathML, Docbook, PDF, ...

# Further reading

- Overview pages at [www.w3.org](http://www.w3.org):
  - <http://www.w3.org/XML/>
  - <http://www.w3.org/Style/XSL/>
  - <http://www.w3.org/Style/CSS/>
- Recommendations (+ drafts) at [www.w3.org/TR/](http://www.w3.org/TR/):
  - <http://www.w3.org/TR/xsl>
  - <http://www.w3.org/TR/xslt>
  - <http://www.w3.org/TR/REC-xml>
  - <http://www.w3.org/TR/REC-CSS2>
- Tutorials and more
  - <http://www.xml.com>
  - <http://www.mulberrytech.com/>
  - <http://www.mulberrytech.com/quickref/>  
(personal favorite)