XML

Smarter data for your applications

Why XML?

What is it?

Why do we need it?

What are smart data?

- Easy: data that are structured so that software can do useful work with them
- Examples:
  - relational database data
  - vector images
  - serialized Java objects
  - flat text databases

So, what are dumb data?

- Data which can only be used for one purpose, because of a lack of structure
- Example:
  - GIF image with text inside
  - Can be displayed, but requires OCR software to retrieve the text as text

Smart data on the web

- Server-side, lots:
  - RDBMSs, SGML, special formats…
- Client-side, hardly any:
  - HTML, PDF, plain text…
- This means that the web consists of millions of pages of information, all of which is very awkward to process automatically...

The trouble with this

- Your browser essentially becomes a stupid display client, using HTML instead of vt3270
- To work with the data in any way you must talk to software on the server, which performs the actual work
- The HTML in your browser is dumb data

The need for exchange

- Now that most businesses are online we see the need for exchanging information between heterogeneous systems
- However, the web and the internet themselves don’t support this in any way
- The alternatives, SGML, ASN.1 and EDIFACT, are large and complex...

A critique of HTML

- Extraordinarily flexible, but low on structure
- Fixed tag set (vocabulary)
- No automatic validation
- Unreliable use of the syntax
- Nobody uses the data model

How XML solves this

- Define your own tags (vocabulary)
- Validate against the definition
- Draconian error handling and strict definition of the syntax
- Smaller and simpler than SGML
- Standardized APIs for working with it
- A data model specification is coming

Dumb data

```
<p><font face="Arial"><strong>Upcoming Release
   1.0.5</strong></font></p>

[…]

<p><font face="Arial"><strong>March 20,
   1999</strong></font></p>

<p><font face="Arial">OpenXML 1.0.4 released with X3P
   Processor API, improved support for
XCatalog and NodeIterator, and minor bug fixes.</font></p>
```

Smart data

```
<product id="OpenXML">
  <name>OpenXML</name>
  <version>1.0.4</version>
  <last-release>19990320</last-release>
  <info-url>http://www.openxml.org/</info-url>
  <changes>
    This release introduces Processor API, new
XCatalog DTD, live NodeIterator and
    minor bug fixes in XML/HTML processing.
  </changes>
</product>
```

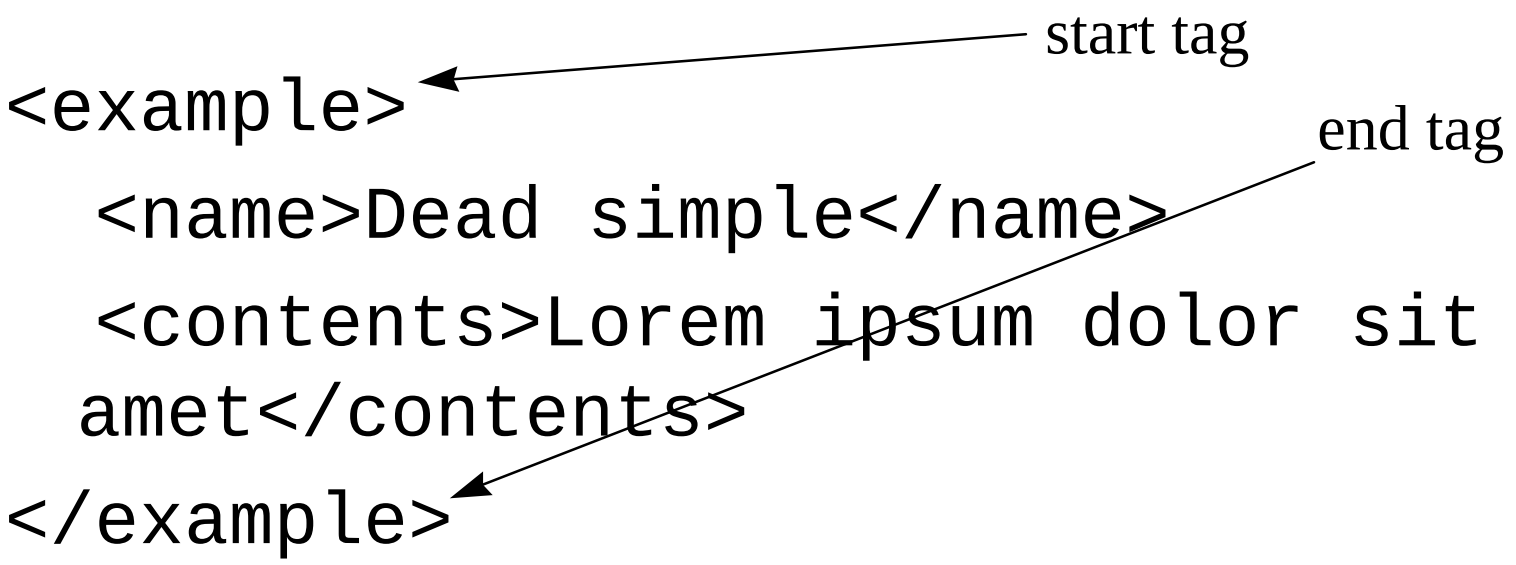
Quick technical intro

XML background

- An application profile (subset) of SGML
- Defined in a W3C recommendation
- Simplifies SGML by:
  - leaving out many syntactical options and variants
  - leaving out quite a few DTD features, some of which are useful
  - leaving out some troublesome features

Elements

- A simple and complete XML document:



Elements

- Attach semantics to a piece of the document
- Have an element type (‘example’,’para’,…)
- The element type definition (whether formal or informal) defines:
  - semantics
  - allowed contents (content model)
  - allowed attributes

Attributes

- Used to annotate the element tree with extra information
- Always attached to start tags:

```
<element-type attributename="value" …
```

- Elements can have any number of elements, but only one of each
- Attributes have little internal structure

Other stuff

- Processing instructions, used mainly for extension purposes (`<?target data?>`)
- Comments (`<!-- … -->`)
- Character references (`&#163;` = £)
- Entities:
  - named files or pieces of markup
  - can be referred to recursively, inserted at point of reference

Internationalization

- XML documents can use any character *encoding*
- The XML data model uses the Unicode character *set*
- This ensures interoperability, but causes trouble when Unicode is insufficient
- There are ways around this, though

Document types

- One of the ideas behind SGML is that documents, just like objects and normal values, have types
- A document type is a class of documents with similar structure and semantics
- Examples: slide presentations, journal articles, meeting agendas, method calls, encyclopedia articles etc

DTDs

- DTDs provide a standardized means for declaratively describing the structure of a document type
- This means:
  - which elements an element can contain
  - whether it can contain text or not
  - which attributes it can have
  - some typing and defaulting of attributes

A sample DTD

```
<!ELEMENT xsa (vendor, product+)>
<!ATTLIST xsa version CDATA #FIXED "1.0">
<!ELEMENT vendor   (name,email,url?)>
<!ELEMENT name     (#PCDATA)>
<!ELEMENT email    (#PCDATA)>
<!ELEMENT url      (#PCDATA)>

<!ELEMENT product  (name,version,
        last-release,info-url?,changes?)>
```

The XML standards

A quick guide

REC-xml 1.0

- Defines the basic markup language and syntactical rules
- Defines DTDs
- Divides processors into:
  - non-validating (only reads internal DTD)
  - validating

Linking

- XPointer:
  - defines addressing into XML documents
  - id(sect1-2).child(2,paragraph)
- XLink:
  - defines hyperlinking in XML
  - adds multi-way links with named roles
  - links outside documents

Style

- CSS2:
  - simple style language for defining display
  - no content reordering, little text generation
- XSL:
  - complex style language
  - defines transformation between XML DTDs
  - also defines a formatting DTD

Programming

- SAX:
  - event-based parser API defined outside W3C
  - simple and widely implemented
- DOM:
  - tree-based document API
  - partially implemented in browsers
  - many server-side implementations

Various

- Namespaces in XML:
  - A means of creating globally unique element type names
- Architectural forms:
  - SGML standard for element subtyping
  - Declarative transformation system
- Topic Maps:
  - ISO standard for navigational indexes

Various

- XML Infoset:
  - data model specification for XML
- XML Fragment Interchange:
  - interchange of document fragments
- XCatalog/SGML Open Catalogs:
  - resolution of symbolic DTD and entity names

Stronger validation

- XML schemas:
  - A language for defining XML vocabularies
  - Will provide primitive data typing and element type subtyping
  - Currently still in the W3C pipeline

Using XML

Actually making use
of all this

What is XML useful for?

- In a web context:
  - to simplify web publishing
  - to enable advanced client-side applications
  - information harvesting from web pages
  - better searching

Using XML outside the web

- In general applications:
  - as an exchange format
  - as a cross-language serialization format
  - for configuration/data files
  - for data files

Using XML outside the web

- In publishing, for structured documents
- EDI and e-commerce
- For database exchange and report generation (with XSL)

Some XML applications

- CSD: CORBA Software Description (OMG)
- CDF: Channel Definition Format (MS)
- MathML: Mathematical Markup Language (W3C)
- GedML: Genealogical exchange format (M. Kay)
- WIDL: Web IDL (webMethods)
- UXF: UML eXchange Format (J. Suzuki)
- XML-RPC: XML-based RPC (Userland)
- CML: Chemical Markup Language (OMF)

Some more XML applications

- VHG: Virtual HyperGlossary (VHG)
- ICE: Information and Content Exchange (Sun, Adobe et al)
- XSA: XML Software Autoupdate (me)
- WDDX: Web Distributed Data Exchange (Allaire)
- XHTML: Extensible HTML (W3C)

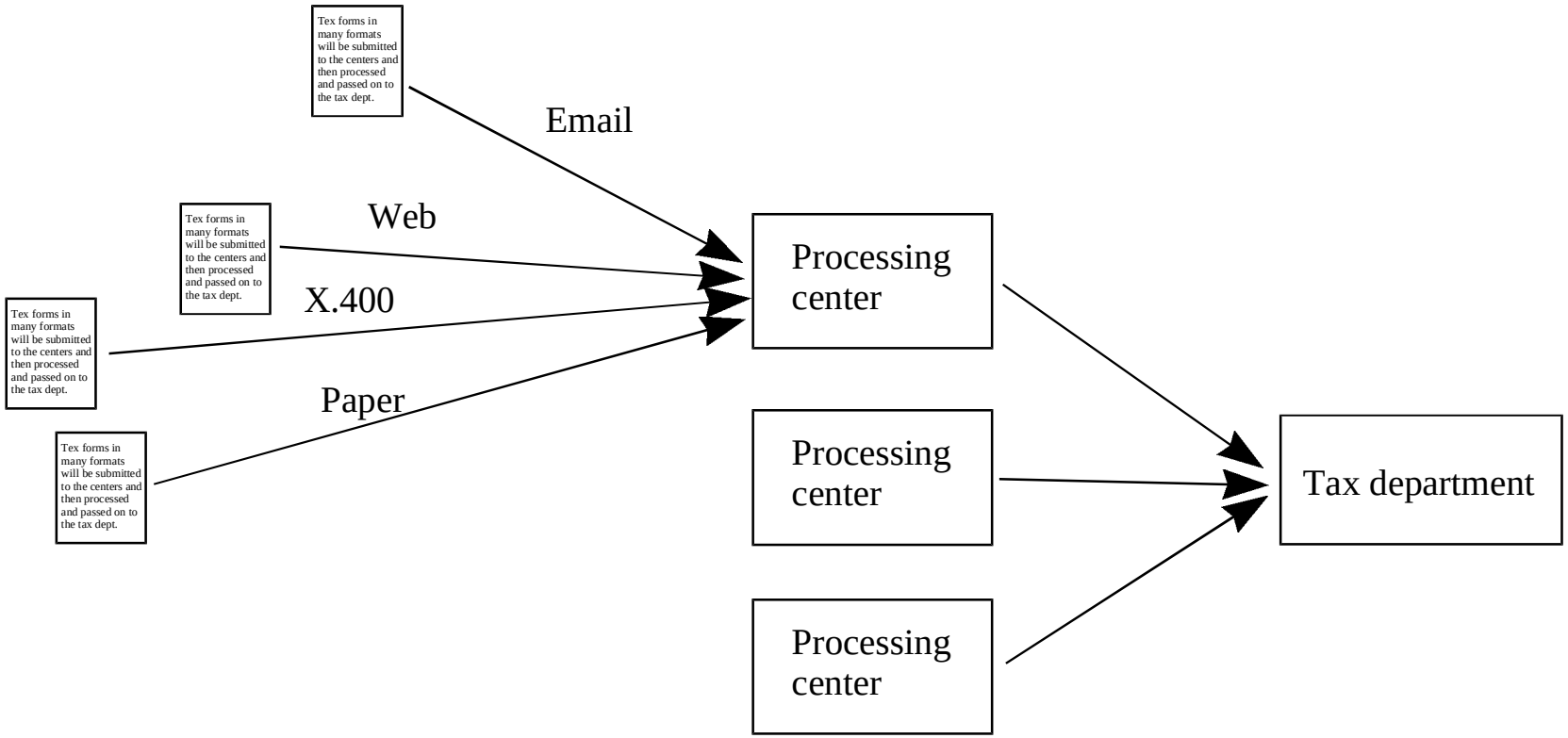
Tax forms in XML

Control information
in XML

Background

- Every year all adult Norwegians submit a tax form with financial information used to calculate the amount of tax to be paid
- Processing such amounts of paper requires enormous resources
- A project to allow these forms to be submitted electronically has been started

Processing model



One way of using XML

- The most obvious way to use XML in this project: store the form data in XML
- This allows using a single submission format that is easy to parse, validate and generate
- The same format can then be used for delivery to the tax department

Another way

- Tax laws change almost annually, and the forms must follow them
- This requires modifications to:
  - the validation software
  - the form input software for all platforms and submission methods
- Modifying source code is expensive...

Solution

- Store the form fields and their relationships as XML:
  - field 1 is ‘Name’, field 2 is...
  - field 15 is the sum of fields 11-14
  - the value of field 17 should be 0 or larger than 15’000,-
  - if field 18 is ‘yes’, fields 19-22 must be filled in
  - ...

Improvements

- This allows:
  - validation software to be generalized, so that it uses the XML document to validate form contents
  - generalization of input software, to display forms as described by the document and also to do validation before submission

What’s new here?

- The idea of using XML documents to define custom validation requirements
- The idea of letting the same XML document control GUI generation
- In other words: to let markup customize or control applications

Why not SGML?

- Extra SGML features not needed
- SGML parsers are large, and not available in all languages
- Using full SGML in an applet is not feasible
- In general: SGML software is hard to integrate

Recipe Markup Language

An advanced web site

Recipe Markup Language

- Fictional markup language for recipes with:
  - metadata (complexity, time to prepare, country of origin, kind of dish …)
  - ingredients (amounts and alternatives)
  - steps to prepare (with alternatives for ingredients and tools)

The website

- Lets users find and view recipes
- Recipes can be searched for, using metadata:
  - give me an Italian soup that's easy to make in no more than 30 minutes...
- With nutritional and price databases:
  - ...and costs less than 5 GBP and has less than 500 calories

Personalized service

- With registered users:
  - menu generation based on preferences (price, complexity, tastes, nutritional value…)
  - shopping list generation based on the menus and preferences (shopping frequency, refrigerator size)

Customized display

- A Java applet could display the recipe
- The user could fill in
  - ingredient alternatives
  - equipment alternatives
  - skill level
- The applet could update nutritional values, preparation times and recipe text accordingly

Cooking assistance

- During cooking the applet could
  - leave out irrelevant information
  - show only the current step
  - warn about things that need to be done, such as
    - pre-heating
    - steps that need to be started soon
    - things that have been heated long enough

Science fiction

- In the future, most kitchen appliances are likely to have CPUs and IP-addresses
- The Java applet in your web browser could then actually monitor your cooking
- You could also search for recipes that could be made with the ingredients in your refrigerator without typing them in

Making money

- Selling access to
  - the personalized service
  - the display applet
- Selling the recipes with software
  - as a book
  - on CD-ROM
- Selling targeted advertisements

Why is this interesting?

- A general example of the services that become possible with structured information
- Similar things can likely be done with other kinds of structured information
- Not possible with a plain RDBMS solution

Conclusion

Will XML replace HTML?

- Not in the short term:
  - uncounted millions of documents in HTML
  - thousands of HTML-aware applications
  - the simple-and-sloppy general approach to HTML will probably survive
- HTML is being reformulated in XML
- This, however, requires a strict syntax

Will XML succeed?

- It has already :-)
- Netscape and Microsoft support it
- XML parsers exist for most programming languages
- Oracle, IBM, Microsoft, Adobe and Sun have released XML software
- 143 free XML tools have been released

Why is XML good?

- It’s simple
- It’s widely supported
- It provides an easy means for structuring web data and defining exchange formats
- It has a large number of accompanying standards that provide additional functionality

More reasons why XML is good

- It’s well-designed (informed by a decade of experience with SGML)
- It’s extremely extensible (partly thanks to the layered standards approach)

When should I use XML?

- When you have structured information that:
  - needs to cross system boundaries
  - is hard to structure by other means
  - needs to be presented in a web browser
  - can leverage the accompanying standards
  - can reuse the concepts in XML

When should I not use XML?

- For things that are better handled by other means, such as:
  - programming language syntax
  - normal object serialization

The next generation will do things with SGML that we can't even imagine yet - it is that versatile.

William W. Davis,
chairman ANSI SGML
Task Group, April '95