

Semantic Web

From the World Wide Web to a *Web of Semantics*

Fabian Abel, Daniel Krause

IVS Semantic Web Group

16.10.2008

1 A Little Bit of History

- Internet
- World Wide Web
- World Wide Web Consortium (W3C)

2 World Wide Web

- HTML
- Links
- The World Wide Web ... how to proceed?

3 Semantic Web

- Semantic Web: What should be achieved?

4 XML - Extensible Markup Language

- Markup - languages, Meta - languages, and SGML
- Elements
- Attributes
- Well-formed XML
- Valid XML

5 XHTML

6 Document Type Definitions (DTD)

- Elements
- Attributes

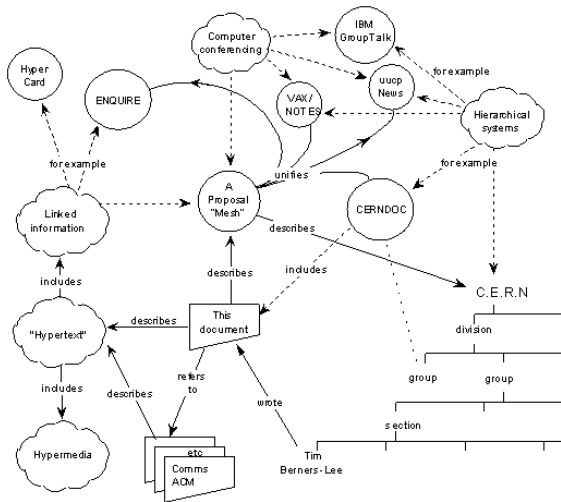
- begin of 60s: idea of a decentralized network (network of computers that can communicate with each other → data security, decentral storage, up-to-date data, ..)
- end of 60s: ARPA Net (ARPA: Advanced Research Projects Agency)
- 70s: academic use of ARPA Net, development of Usenet (electronic black-board)
- 80s: NSF (National Science Foundation): connection of important computing centers in the US (backbone) → "the net of nets" is called *Internet*
- Protocol: TCP/IP
- in Europe: investigations on ISO-normed protocols, later: also TCP/IP.

- Tim Berners-Lee, CERN March 1989, May 1990: Mesh Proposal
- 1990s: World Wide Web
Basis building blocks:
 - Hypertext
 - HTTP: Hypertext Transfer Protocol
 - URL: Uniform Resource Locator
 - HTML: Hypertext Markup Language
- break-through: with appropriate Browsers (Mosaic, Marc Andreessen, begin 90s), Netscape boom 1995/96, Internet Explorer delivered with MS operating systems, ...

Further readings:

*"Einführung in Internet und WWW", SELFHTML by Stefan Münz,
<http://de.selfhtml.org/intro/internet/index.htm>*

Mesh Proposal



<http://www.w3.org/History/1989/proposal.html>

- independent consortium for developing technical standards in the World Wide Web
- started 1994
- 1995: companies join W3C
- members of W3C: companies or organizations (not single persons)
 - join W3C in three-years contracts
 - pay membership fees
 - work on and develop web-standards
- structure of W3C:
 - main building block: so called *Activities*
 - activities: e.g. for HTML, XML, CSS, XSLT, RDF, OWL, ...
 - Each activity has: *Working Groups* (developing "standards") and *Interest Groups* (consultation of working group members)

Definition (Recommendations)

liable documents describing technical standards for the Web.

Review procedure:

- *Working Drafts*: technical notes, several working draft states possible, not liable
- after several reviews on working drafts within interest group and working group: *Candidate Recommendations*, still not liable
- after repeated, public review: *Proposed Recommendation*
- after final review: *Recommendation*

- Recommendations: versioned
 - e.g. HTML 4.01, CSS 2.0, XML 1.0, etc.
- New version of a recommendation requires a new review procedure.
- All recommendations of W3C available via <http://www.w3.org/TR/>

Other Standards: RFCs: Request for Comments

- internet standards, not specific to WWW
- focusing also on lower level layers of TCP/IP stack
- organized by Internet Engineering Task Force (IETF)
`www.ietf.org`

HTML - Markup Language of the WWW

- tags
- begin-tags `<tag_name>`, end-tags `</tag_name>`
- used for structuring content, formatting instructions
- Markup

```
<html>
```

```
  <head>
```

```
    <title> Title </title>
```

```
  </head>
```

```
  <body>
```

```
    <!-- This is a comment -->
```

```
    Content, content, content, ...
```

```
  </body>
```

```
</html>
```

- Headings `<h[1-6]> Text </h[1-6]>`, h1: BIG, h2: Big, h3: big, h4: SMALL, h5: Small, H6: small.
- grouped style `<p> block of text </p>`
- line-break `br`
- horizontal line `hr`
- lists: unnumbered (ul: **unnumbered list**)
``
`listitem`
`listitem`
``
- lists: numbered (ol: **ordered list**)
`<ol type="typ">`
types: a = alphabetical, i = roman, ...
- ` bold face` `<i> italics </i>`
`<u> underline </u>` `<strike> striked </strike>`
`^{super}` `_{sub}`
- plus many more...

Anchor:

```
<a name="anchor_name"> content </a>
```

Hyperlink:

```
<a href="URI"> link text</a>
```

URI: Uniform Resource Locator

- anchor in same file (e.g. #anchor_name)
- other file (e.g. myfile.html, in same directory, or /home/.../myfile.html)
- WWW address (http://...)
- FTP address (ftp://...)
- local address (file://...)

Email-link:

```
<a href="mailto:name@xyz.de">link text</a>
```

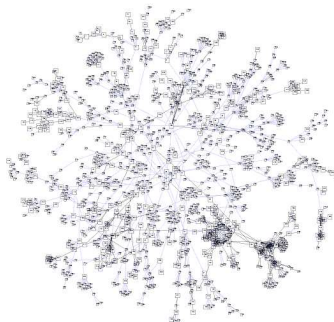
```
<a href="mailto:name@xyz.de?subject=topic">link text</a>
```

```
<a href="mailto:name@xyz.de?subject=topic&cc=name2@xyz.de&..">
```

- Structures (Cluster, Hubs, Authorities, Page ranks,...)
- Content (Retrieval, Text analysis, Data Mining,...)
- Usage (Access statistics, patterns, ...)

Techniques:

- Search engines (→ Structures, Content)
- Recommender Systems (→ Content, Usage)



Disadvantages:

- relationship between documents/elements are untyped
- how to discover & retrieve *relevant* information: progress, but still not ideal
- missing semantics – for machines!

Semantic Web - A definition by Tim Berners Lee

"The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation." (T. Berners-Lee, J. Hendler, O. Lassila: "The Semantic Web", Scientific American, May 17, 2001.

<http://www.sciam.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21>)

Example (Nice machine support)

Interaction scenario in Matt Ruff: Sewer, Gas & Electric. The Public Works Trilogy, 1997

Deutsche Übersetzung: G.A.S. Die Trilogie der Stadtwerke, dtv 2000, Seite 81-83

Example (Vision of a Semantic Web)

Scenario with Lucy and Pete, T. Berners-Lee, J. Hendler, O. Lassila: "The Semantic Web", Scientific American, May 17, 2001.

The entertainment system was belting out the Beatles' "We Can Work It Out" when the phone rang. When Pete answered, his phone turned the sound down by sending a message to all the other *local* devices that had a *volume control*. His sister, Lucy, was on the line from the doctor's office: "Mom needs to see a specialist and then has to have a series of physical therapy sessions. Biweekly or something. I'm going to have my agent set up the appointments." Pete immediately agreed to share the chauffeuring.

At the doctor's office, Lucy instructed her Semantic Web agent through her handheld Web browser. The agent promptly retrieved information about Mom's *prescribed treatment* from the doctor's agent, looked up several lists of *providers*, and checked for the ones *in-plan* for Mom's insurance within a *20-mile radius* of her *home* and with a *rating of excellent* or *very good* on trusted rating services. It then began trying to find a match between available *appointment times* (supplied by the agents of individual providers through their Web sites) and Pete's and Lucy's busy schedules. (The emphasized keywords indicate terms whose semantics, or meaning, were defined for the agent through the Semantic Web.)

In a few minutes the agent presented them with a plan. Pete didn't like it – University Hospital was all the way across town from Mom's place, and he'd be driving back in the middle of rush hour. He set his own agent to redo the search with stricter preferences about *location* and *time*. Lucy's agent, having *complete trust* in Pete's agent in the context of the present task, automatically assisted by supplying access certificates and shortcuts to the data it had already sorted through.

Almost instantly the new plan was presented: a much closer clinic and earlier times - but there were two warning notes. First, Pete would have to reschedule a couple of his *less important* appointments. He checked what they were - not a problem. The other was something about the insurance company's list failing to include this provider under *physical therapists*: "Service type and insurance plan status securely verified by other means," the agent reassured him.

Lucy registered her assent at about the same moment Pete was muttering, "Spare me the details," and it was all set. (Of course, Pete couldn't resist the details and later that night had his agent explain how it had found that provider even though it wasn't on the proper list.)

SGML: Standard Generalized Markup Language

- Meta - language for marking the structure of data.
- ISO 8879 standard: "Information processing - Text and Office Systems - Standard Generalized Markup Language (SGML)", 1986
- allows the definition of markup languages with so-called DTD (Document Type Definitions)
- DTDs define which elements the corresponding markup language has, which attributes these tags have, and how and whether elements can be nested.
(elements: the tags that can be used for markup)
- SGML is very advanced - too sophisticated to use?

- HTML (Hypertext Markup Language) - a markup language defined with SGML
- XML (Extensible Markup Language) - a metalanguage for creating XML-based markup languages
 - subset of SGML
 - does not have a fixed set of tags
 - allows users to define tags on their own
 - allows the definition of markup-languages with DTDs (then: tags are defined)

An XML document consists of a prolog, some elements, and an optional epilog.

Prolog

- XML declaration `<?xml version="1.0"?>`
- optional: character encoding
`<?xml version="1.0" encoding="ISO-8859-1"?>`
possible encodings: UTF-8, UTF-16 (UTF: Unicode Transformation Format), UTF-16 is the ISO 10646 standard UCS, ISO-8859-1
- optional: whether the document is self-contained or refers to external structuring documents
`<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>`
- such a reference looks e.g. like
`<!DOCTYPE book SYSTEM "book.dtd">`
 - structure is defined in the document type definition-file "book.dtd",
 - SYSTEM : only locally recognized name / file
 - otherwise: PUBLIC

Elements

- main building blocks - the "objects of discourse" of the XML document
- elements consist of begin-tag, content, end-tag
e.g. `<author>J. W. Goethe</author>`

- naming tags: restrictions:

- first character: letter | underscore | colon, e.g.
`myname`, `_myname`, `:myname`
- no element may begin with `[x|X][m|M][l|L]`
- XML is case sensitive

- content of an element: `[text | other elements]+ | nothing`

```
<author>J. W. Goethe
  <date-of-birth> 28. August 1749 </date-of-birth>
  <place-of-birth> Frankfurt am Main </place-of-birth>
</author>
```

- empty elements `<author></author>` can be abbreviated as `<author/>`

Attributes

Express properties of elements.

An attribute is a name-value-pair inside the opening tag of an element:

```
<author name="J. W. Goethe" date-of-birth="28. August 1749" />
```

Attributes vs. elements:

- attributes: properties of an element
- elements can be nested, attributes not
- order of elements is important, order of attributes not

Comments

Like in HTML: `<!-- this is a comment -->`

Processing instruction

Pass information on how to handle elements `<? .. ?>`

Example (A reference to a stylesheet)

```
<? stylesheet type="text/css" href="./styles/mystyle.css" ?>
```

Definition (Well formed)

A XML document is well-formed if it is syntactically correct.

Syntactic Rules:

- 1 there is only one root element
- 2 each element consists of an opening and a corresponding closing tag, abbreviation possible
- 3 tags may not overlap, correct nesting:

```
<author>J. W. Goethe
  <date-of-birth> 28. August 1749 </date-of-birth>
</author>
```

not correct:

```
<author>J. W. Goethe
  <date-of-birth> 28. August 1749 </author>
</date-of-birth>
```

- 4 attributes of an element have a unique name
- 5 naming restrictions for elements

- So far: element names have been freely chosen.
- Question: How can two applications exchange XML documents?
- → **Structuring information**
- Two solutions in XML: DTD (Document Type Definition) and XML Schema

Definition (Valid XML)

An XML document is valid, if

- 1 it is well-formed
- 2 uses and fulfills structuring information

Observe:

- No further revisions of HTML
- XHTML now a W3C recommendation

Why XHTML?

- always: need for new markup
- extensions → easy in XML!
- HTML: success-story in the Web.
- goal: defining XML-based HTML, as close to the well-known and widely used HTML as possible.

Basic Idea

- describe HTML in an XML conform way: XHTML
 - XHTML can be edited and validated
 - XHTML can use applications (scripts and applets) that rely upon HTML or XML.

Advantages

- *Extensibility*: Under HTML, the addition of a new group of elements requires alteration of the entire DTD. In an XML-based DTD, all that is required is that the new set of elements be internally consistent and well-formed to be added to an existing DTD. This greatly eases the development and integration of new collections of elements
- *Portability*: XML documents are required to be well-formed (elements nest properly). Browsers currently compensate ill-formed HTML, what about non-desktop devices?

XHTML 1.0 vs. HTML 4.0

- XHTML 1.0 "rebuilds" HTML 4.0
- some differences [due to the use of XML]

Difference: Documents must be well-formed

Correct nesting of elements!

Difference: Case sensitive

HTML is not case sensitive, XML and therefor XHTML are case sensitive.
All element names and attribute names in XHTML in lower case letters

Difference: For non-empty Elements: End-Tag required

Example in HTML 4.0:

```
<p>here is the paragraph</p>  
<p>and here is another paragraph</p>
```

In HTML 4.0, a couple of elements have optional end-tags, eg. `<td>`, `<dd>`,
`<dt>`, `<option>`

Require in XHTML the end-tag!

Difference: Attribute values must always be quoted

```
<td rowspan="3">
```

Difference: Attribute minimization

In XHTML, attribute-value-pairs must be written in full. E.g Attribute names like *compact* or *checked* cannot occur without their value being specified

```
<ul compact="compact">
```

Difference: Empty Elements

Example in HTML 4.0:

```
<p>Some text with a <br>linebreak</p>  
<p><IMG SRC="picture.gif" ALT="some picture"></p>
```

Equivalent in XHTML 1.0:

```
<p>Some text with a <br/>linebreak</p>  
<p></p>
```

Difference: Mime-Type

- HTML: mime-type *text/html*
- XHTML: mime-types *text/html*, *text/xml*, *application/xml*

Difference: XHTML needs ... the XML Declaration!

```
<?xml version="1.0" encoding="UTF-8"?>
```

Difference: XHTML refers to ... a different DTD

HTML 4.01:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Strict//EN"  
    "http://www.w3.org/TR/html4/strict.dtd">
```

XHTML 1.0:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"  
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"  
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"  
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```

- Strict: no use of deprecated HTML tags which are now substituted by CSS or others
- Transition: use of deprecated HTML tags (mostly formatting tags)
- Frameset: if frames are used

Difference: Root Element

The root element in XHTML 1.0 has an attribute
xmlns="http://www.w3.org/1999/xhtml"

```
<html xmlns="http://www.w3.org/1999/xhtml">
```

Difference: Basic Structure: XHTML requires root, head, and body

```
<html xmlns="http://www.w3.org/1999/xhtml">  
  <head>  
    <title>Text</title>  
    <!-- gegebenenfalls andere Elemente im Kopfbereich -->  
  </head>  
  <body>  
    <h1>Text</h1>  
  </body>  
</html>
```

Difference: Anchor

In HTML 4.0: *name* or *id* as attribute in anchors

```
<a href="#myanchor"> My anchor</a>
```

somewhere else:

```
<a name="myanchor"> Here we go</a>
```

- In XHTML: Anchor must be defined using the XML attribute *id* of type ID
- valid XHTML: *id* must be used.
- the *name* attribute is deprecated

```
<a href="#myanchor"> My anchor</a>
```

somewhere else:

```
<a id="myanchor" name="myanchor"> Here we go</a>
```

Other differences

embedding script languages, etc. In HTML: some nesting rules: E.g. that `<a>`-Elements might not be further nested.

Difference: File-names

- *.html* → Browser uses HTML parser
- *.xml* → Browser uses XML parser
- HTML parser: relaxed against errors in the HTML
- XML parser: document must be syntactically correct

How to name an XHTML-File?

- an XHTML-File is XML, so .xml?
Browsers normally display the tree structure of the file
The tree will only be displayed if the file is syntactically correct
- an XHTML-File as HTML?
Saving the XHTML file with extension *html* or *htm*: Browsers display the file as we know it from HTML files. *Try on your browser to see how this is done!*

- Situation: HTML / XHTML consist of approximately 80 elements.
- Browsers on PC can easily handle the rules for displaying HTML
- What about other devices, e.g. mobile phones or smart cards?

Idea

Modulize XHTML, so that a device can implement some modules of XHTML (a subset of the language)

- Modulization: also enhancements
- XHTML 1.1: Module-based XHTML
- only "strict" (no transitional, no framesets)

SELFHTML von Stefan Münz: Die Module von XHTML 1.1:

<http://de.selfhtml.org/html/xhtml1/modularisierung.htm>

Where can Document Type Definitions be specified:

- external DTD: a DTD in a separate file
- internal DTD: within the XML document itself

Rule for the practice

to use DTDs over several documents: external DTDs are preferable (avoid duplication of code!)

Consider the element

```
<author>
  <name> J. W. Goethe </name>
  <date-of-birth> 28. August 1749 </date-of-birth>
</author>
```

Example (DTD for an XML document)

A DTD for this element type

```
<!ELEMENT author (name, date-of-birth)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT date-of-birth (#PCDATA)>
```

Meaning of the DTD:

- element types author, name, date-of-birth can be used in the document
- a author element contains a name and a date-of-birth element, in that order
- a name or a date-of-birth element may have any content
- #PCDATA is the only atomic type for elements

Example (DTD for an XML document)

An author element can contain either a name or a date-of-birth

```
<!ELEMENT author (name | date-of-birth)>  
<!ELEMENT name (#PCDATA)>  
<!ELEMENT date-of-birth (#PCDATA)>
```

Example (DTD for an XML document)

An author element contains a name and a date-of-birth (in any order)

```
<!ELEMENT author ((name, date-of-birth)|(date-of-birth, name))>  
<!ELEMENT name (#PCDATA)>  
<!ELEMENT date-of-birth (#PCDATA)>
```

- ?: zero times or once
- *: zero or more times
- +: once or more times

default / no cardinality operator: exactly one.

Example

```
<!ELEMENT author (phone_no*)>
```

Attribute Types

- CDATA: string (sequence of characters)
- ID: identifier, unique name
- IDREF: reference to another element with a respective ID attribute
- IDREFs: series of IDREFs
- (v1 | .. |vn): enumeration of possible values

- #REQUIRED: the attribute **must** occur in the element type
- #IMPLIED: occurrence is optional
- #FIXED "value": Every element **must** have this attribute, which has always the value given after #FIXED in the DTD, cannot be overwritten in the XML document itself
- "value": the default value of an element, can be overwritten in the XML document

Example

Consider the element

```
<order orderNo="23456" customer="John Smith" date="October 15, 2002">  
  <item itemNo="a528" quantity="1"/>  
  <item itemNo="c817" quantity="3"/>  
</order>
```

Example (DTD for an XML document)

```
<!ELEMENT order (item+)>  
<!ATTLIST order  
  orderNo    ID        #REQUIRED  
  customer   CDATA     #REQUIRED  
  date       CDATA     #REQUIRED >  
<!ELEMENT item EMPTY>  
<!ATTLIST item  
  itemNo     ID        #REQUIRED  
  quantity   CDATA     #REQUIRED  
  comments   CDATA     #IMPLIED >
```

Example from "A Semantic Web Primer", G. Antoniou, F. v. Harmelen

Example (Using IDREF and IDREFS)

```
<!ELEMENT family (person+)>
<!ELEMENT person (name)>
<!ELEMENT name (#PCDATA)>
<!ATTLIST person
  id      ID      #REQUIRED
  mother  IDREF  #IMPLIED
  father  IDREF  #IMPLIED
  children IDREFS #IMPLIED>
```

An XML element that respects this DTD:

```
<family>
  <person id="bob" mother="mary" father="peter">
    <name> Bob Marley </name>
  </person>
  <person id="bridget" mother="mary">
    <name> Bridget Jones </name>
  </person>
  <person id="mary" children="bob bridget">
    <name> Mary Poppins </name>
  </person>
  <person id="peter" children="bob" >
    <name> Peter Marley </name>
  </person>
</family>
```

Example from "A Semantic Web Primer", G. Antoniou, F. v. Harmelen

Example (EMAIL)

DTD for Email

```
<!ELEMENT email (head,body)>
<!ELEMENT head (from,to+,cc*,subject)>
<!ELEMENT from EMPTY>
<!ATTLIST from name CDATA #IMPLIED
              address CDATA #REQUIRED>
<!ELEMENT to EMPTY>
<!ATTLIST to name CDATA #IMPLIED
            address CDATA #REQUIRED>
<!ELEMENT cc EMPTY>
<!ATTLIST cc name CDATA #IMPLIED
            address CDATA #REQUIRED>
<!ELEMENT subject (#PCDATA)>
<!ELEMENT body (text,attachment)>
<!ELEMENT text (#PCDATA)>
<!ELEMENT attachment EMPTY>
<!ATTLIST attachment encoding (mime|binhex) "mime"
                    file CDATA #REQUIRED>
```

Example from "A Semantic Web Primer", G. Antoniou, F. v. Harmelen

- syntax not expressible in XML
- limited datatypes
- limited possibilities to express the cardinality of elements
- namespaces are not supported
- no inheritance

→ XML Schema (next lecture)