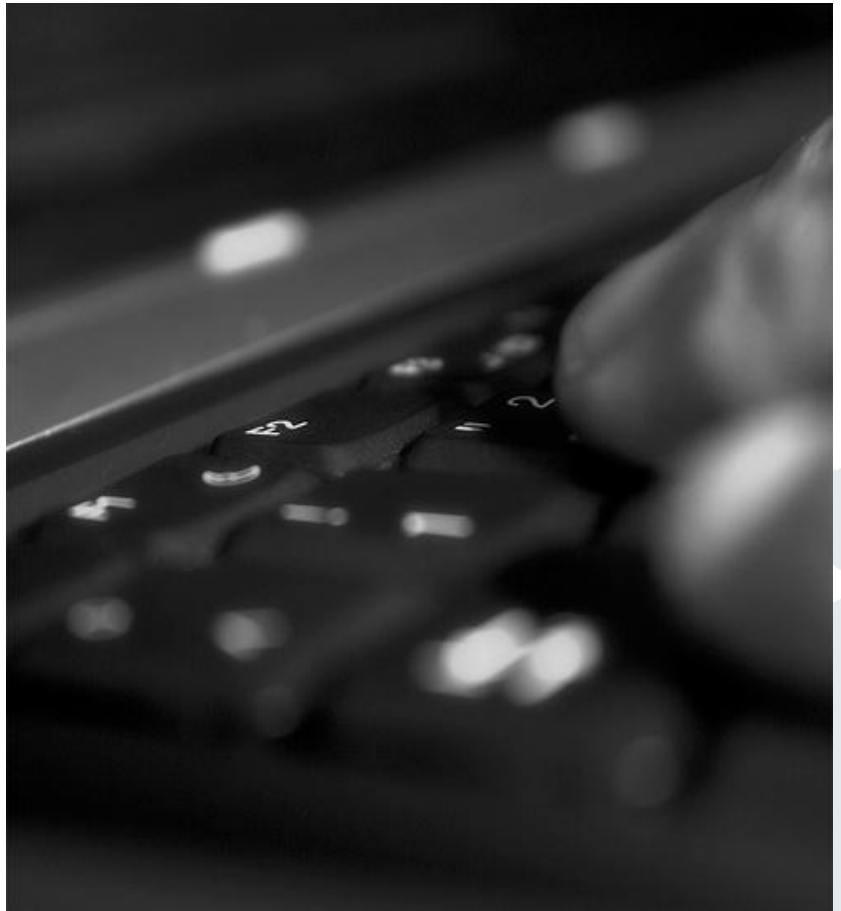


Lecture 10
Business Informatics 2 (PWIN)

IC Software Development III
Mark-up Languages

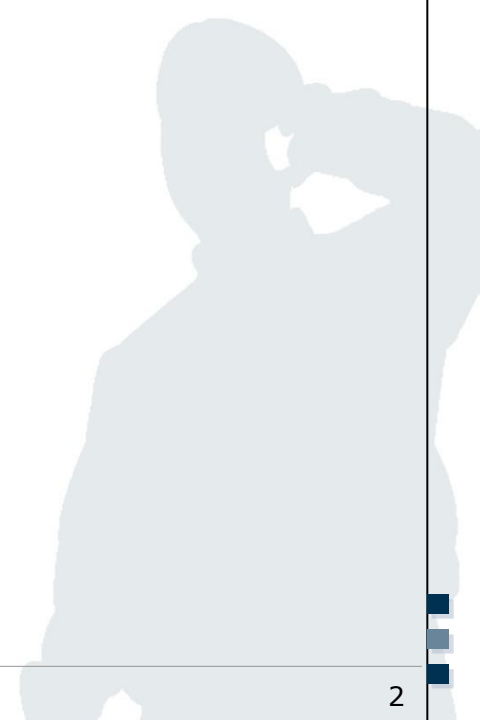
SS 2011

Dr. Andreas Albers
www.m-chair.net



Jenser (Flickr.com)

- From HTML to XML
- XML-Concept
- Processing of XML Documents
- XML Example Applications



- The Hypertext Mark-up Language (HTML) is a very simple description language for contents:
 - Hardly any semantic descriptions for content
 - Mainly structural and layout information such as sections, headlines, lists, etc. exist
- So, how can, for instance, a postal address in HTML be recognised and processed by a software system on a website?

```
<H1>Chair for Mobile Business & Multilateral Security</H1>
```

```
<H2>Grüneburgplatz 1</H2>
```

```
<H3>60623 Frankfurt am Main</H3>
```

- HTML is not suitable for data storage because of its missing meta language features for describing the meaning of data



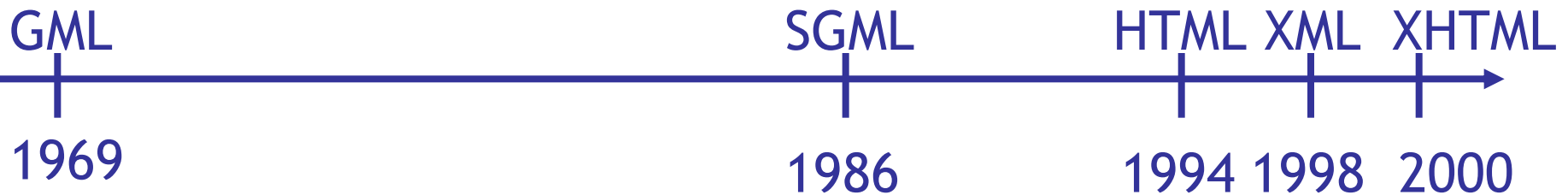
- High efforts for the generation of HTML pages from data stored in databases (scripts, formatting, presentation, etc.)
- Often redundant data storage, risk of data inconsistencies
- Careless HTML syntax is often tolerated. For instance, elements are not closed, but are still valid for many Internet Browsers

- `Erta Ale is a shield volcano, part of the East African Rift system.`



- Consequently, there is a need for a description of data, which eliminates these issues.

Development of meta languages for data description



GML: Generalized Meta Language by IBM

SGML: Standard Generalized Meta Language as Norm ISO 8879 for Data exchange and storage

HTML: Definition of version 2 as SGML-dialect

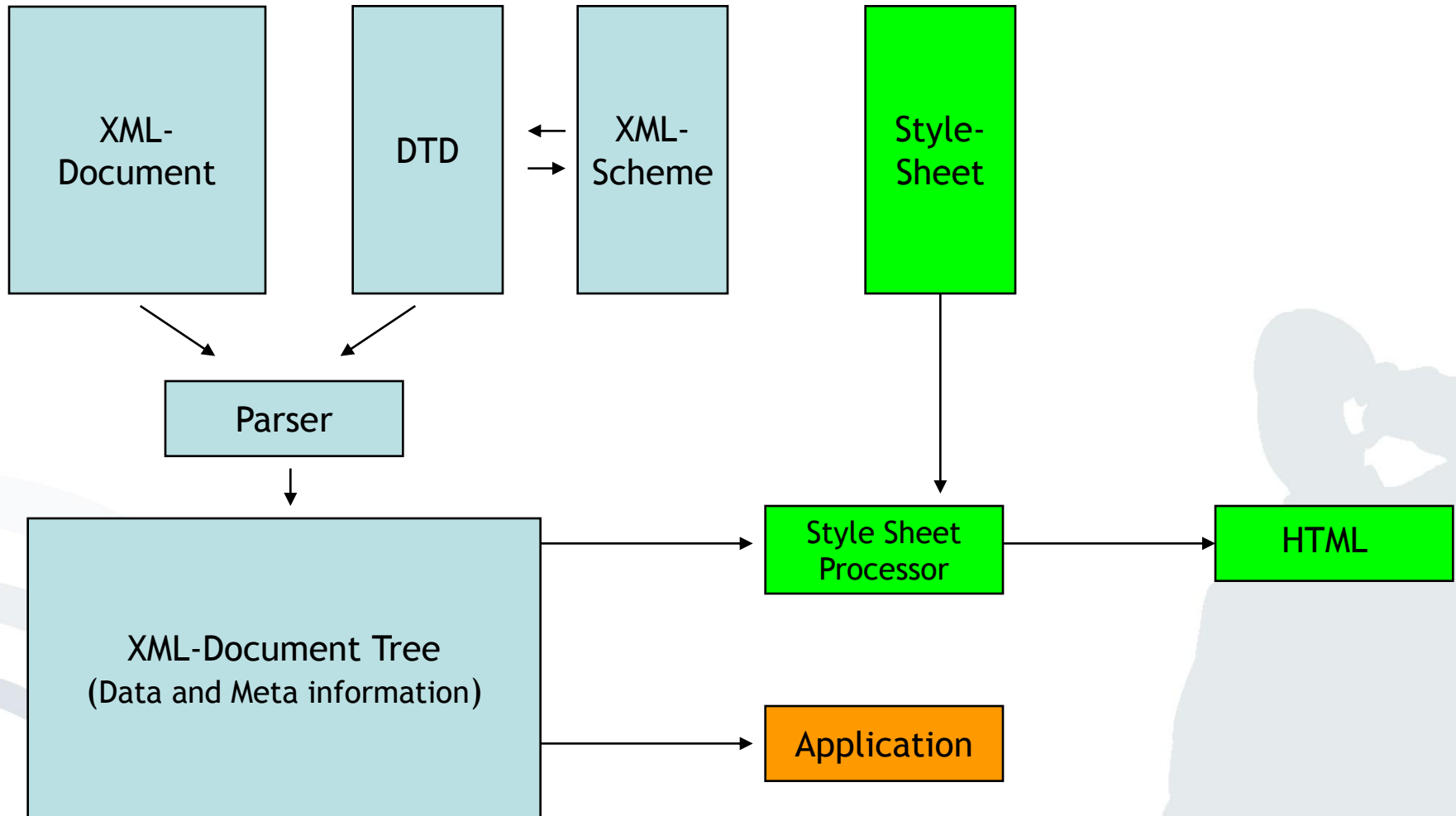
XML: Links HTML with the claim of SGML: Extensible Mark-up Language

XHTML: HTML based on XML

- Basic idea of all Standard Generalized Mark-up Languages (SGML):
 - Create processable documents by adding meta information about structure and content
 - Establish a system und manufacturer independent standard
 - Separation of structure, content and presentation of a document
- SGML dialects:
 - LaTeX
 - Postscript

- Simple and easy to read for humans (not binary)
- Light subset of SGML, carrying only the most relevant language features
- Standardised
- Self-describing due to included meta descriptions
- Extendable with new elements -> creation of application specific meta models
- Suitable for data storage

- **DTD**
Document Type Definition - describes the structure of an XML document and defines its *grammar*.
- **XML Scheme**
Alternative approach to DTD with additional features
- **Parser**
Translates an XML document in a document tree while making its elements accessible for applications
- **Style Sheet**
Layout information for rendering the XML documents contents
- **Style-Sheet-Processor**
Implements the style information and generates the result pages

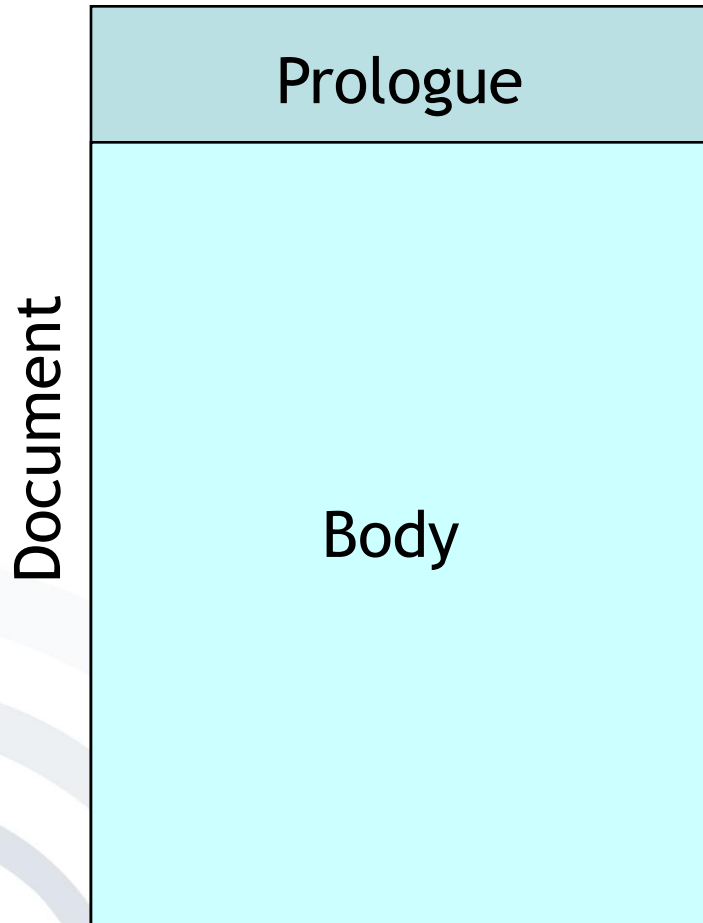


Some general XML Applications

- Sharing of data between different components of an application (e.g. Microsoft Excel / Access)
- Storage of application data in plain, non-binary text files (e.g. Microsoft Word Format)
- Advancing Electronic Data Exchange (EDI):
 - Transactions between banks
 - Producers and suppliers sharing product data
- User generated Content (e.g. Google Maps Layers)
- Access to Services and Applications via the Internet (e.g. Web Service APIs)

- From HTML to XML
- XML-Concept
- Processing of XML Documents
- XML Example Applications

XML-Document Structure



Prologue contains the XML-version and information about the used character encoding.

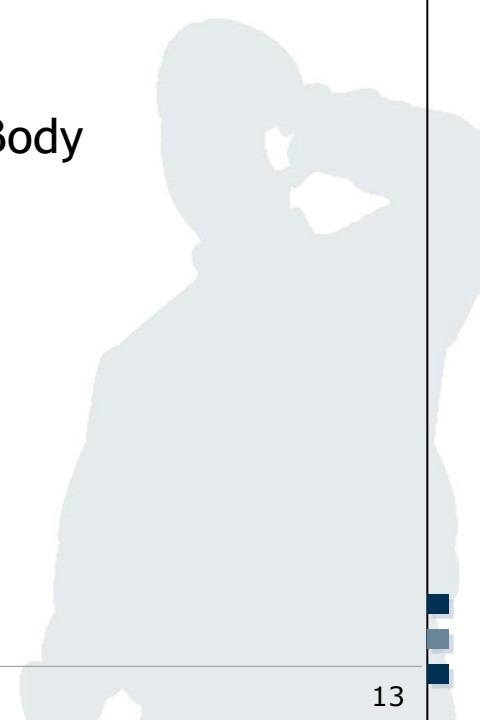
Body contains data

```
<?xml version=„1.0“ encoding=„ISO-8859-1“ ?>
```

Prologue

```
<flirt>  
  <name>Daisy</name>  
  <mobile>+436508469249</mobile>  
  <email>daisy@m-chair.net</email>  
  <city>Innsbruck</city>  
  <first date>2010-12-23</first date>  
  <last date>2011-02-01</last date>  
  <birthday>1983-11-13</birthday>  
  <vegetarian>no</vegetarian>  
  <status>single</status>  
</flirt>
```

Body



- XML expects closed elements!
 - `<name>` is a tag
 - Syntax: `<StartTag>content</EndTag>`
 - Start tags must correspond to end tags, and vice versa
 - `<name>Daisy</name>`
- Attributes are included in the start tag:
 - `<city residence=„first“>Innsbruck</city>`

- An **element**: Everything between two tags; for instance
 - `<title>Complete Guide to DB2</title>`
- Elements may be **nested**; for instance
 - `<book>`
 - `<title>Complete Guide to DB2</title>`
 - `<author>Chamberlin</author>`
 - `</book>`
- Empty element
 - `<red></red>` abbreviated `<red/>`
- An XML document has a unique **root element**.

- An XML document is **well-formed**, if
 - It only contains properly encoded legal Unicode characters.
 - None of the special syntax characters such as "<" and "&" appears “un-escaped” in the data.
 - The begin, end, and empty-element tags, which delimit the elements, are correctly nested, whereas none is missing or overlapping.
 - The element tags are case-sensitive; the beginning and end tags must match exactly.
 - There is a single "root" element which contains all the other elements.

- This type of nesting is not allowed:

```
<telephone>  
  <mobile>+4916008154712  
  <home>+4972138488551  
  </mobile></home>  
</telephone>
```

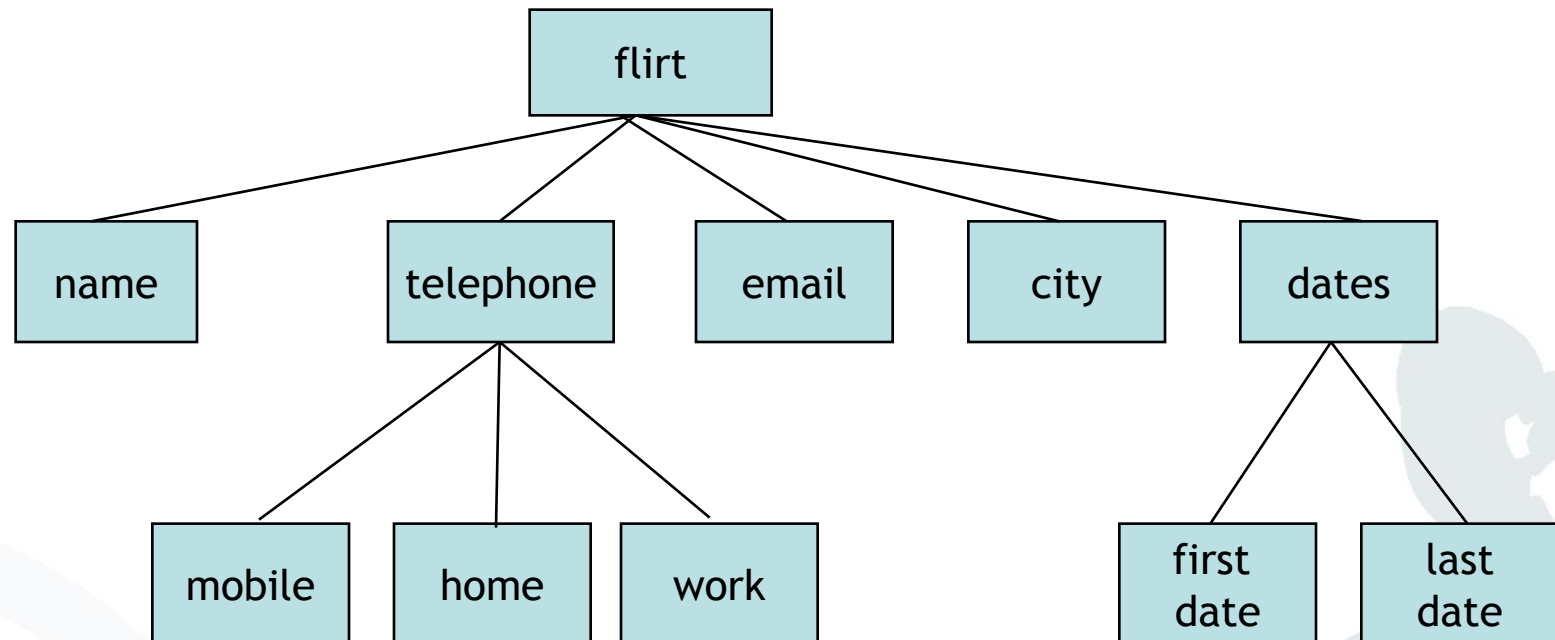
- It cannot be determined if a number belongs to <mobile> or <home>.
- The document is not well-formed. This can be automatically detected by a parser software.

- As in HTML, some characters are used for the syntax:

Character notation

<	<
>	>
&	&
,	'
”	"

- XML document tags can also be considered as objects in an object-oriented database or a tree (document tree):



- Because of the distinct, tree-like structure and similarity to object-oriented systems, computers are able to unambiguously recognise the data structure when reading an XML document.

Document Type Definition (DTD)

- Document Type Definition (DTD) describes the structure of a document and defines a *grammar* for the XML document.
- Comparable to a type or variable declaration programming language.
- It defines which elements and references may appear in the document based the DTD.
- The DTD also declares entities which are allowed to be used in the XML document.

- Element-Content:

EMPTY	Empty element
ANY	Any content
	Selection list
,	Sequence
()	Grouping
(#PCDATA)	<i>Parsed Character Data</i> (mixed data)

- Cardinalities:

	empty: exactly one value is necessary
+	At least one value
?	None or one value
*	None or multiple values

- Rule declaration for the elements in a DTD:

<!ELEMENT flirt	(name,telephone,email,city,contacts, birthday, status)>	
<!ELEMENT name	(&#PCDATA)>	← Text
<!ELEMENT telephone(mobile home work)+>		
<!ELEMENT mobile	(&#PCDATA)>	
<!ELEMENT home	(&#PCDATA)>	
<!ELEMENT email	(&#PCDATA)>	
<!ELEMENT city	(&#PCDATA)>	
<!ELEMENT contacts	(&#PCDATA)>	
<!ELEMENT birthday	(&#PCDATA)>	
<!ELEMENT status	(&#PCDATA)>	

Selection list

- An XML document, which complies with a DTD is called „valid“.
- The validity of an XML document can be automatically determined by a parser software.
- This concept allows consumers of XML documents (e.g. a software application) to verify that the XML documents contents comply with their expected document format
 - Specified document structure
 - Allowed elements and data
 - ...

- “XML Scheme” is an alternative to the DTD.
- XML Scheme eliminates some of the DTD weaknesses:
 - Better content modelling for syntax check
 - Order and nesting is configurable
 - Configurable value margins
 - Checking of element data types
 - Better definition of the cardinalities with Min. and Max.
 - More detailed data type selection analogue to programming languages and databases (e.g. boolean, number, float, date time, ...)

- XML documents are especially beneficial if data is shared across applications, between users or even across independent enterprises.
- How can tag mix-ups be prevented, if data from different sources with identical tag names is merged?

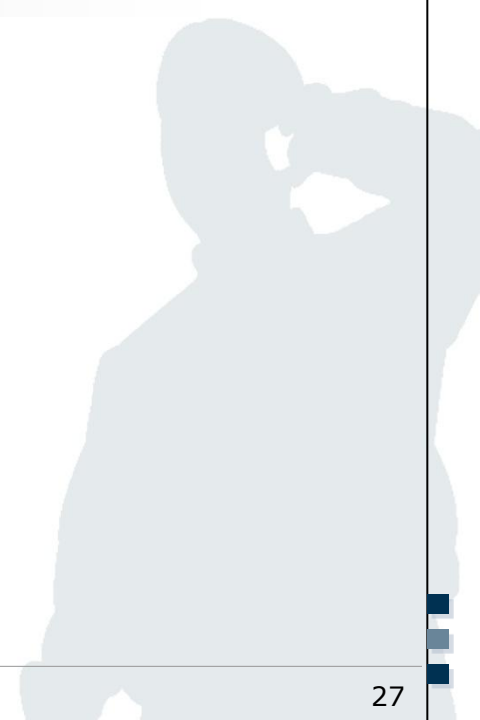
```
<Book>  
  <Title>Computer Networks</Title>  
  ...  
</Book>  
  
<Author>  
  <Title>Professor</Title>  
  ...  
</Author>
```

Idea

- A Universal Resource Identifier (URI), which allows the introduction of a namespace defined by a globally unique path. For this, a prefix for an element is created:

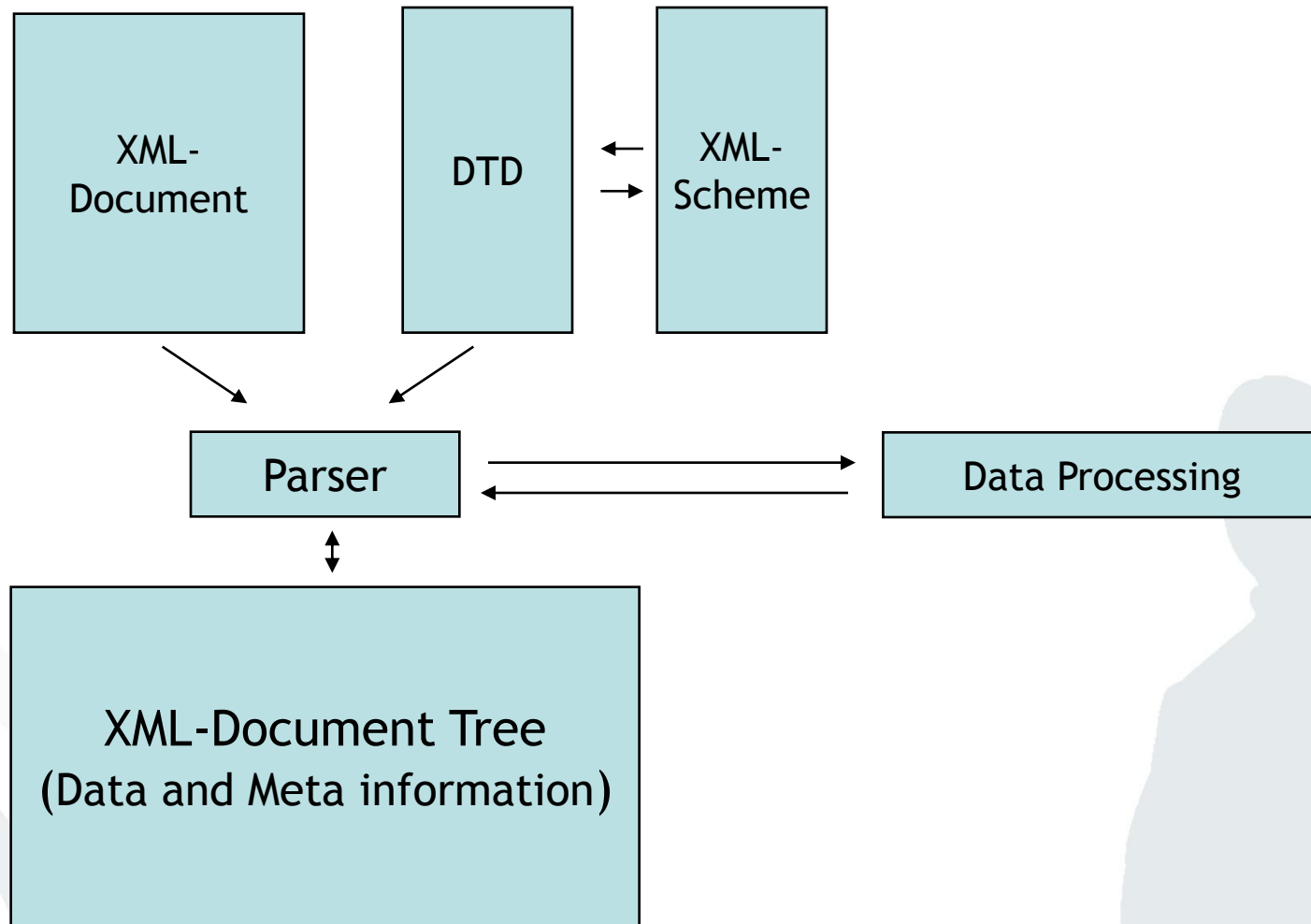
```
<book
  xmlns:book="http://www.amazonen.de/namespaces/books"
  xmlns:aut="http://www.amazonen.de/namespaces/authors"
>
  <book:Title>Networks</book:Title>
</book>
```

- From HTML to XML
- XML-Concept
- Processing of XML Documents
- XML Example Applications



- Processing an XML document requires a parser
- A parser is a software that reads DTDs, schemas and XML documents and enables an application to access all of the XML document elements.
- General parsing process
 1. An application (e.g. Microsoft Word) opens an XML document.
 2. The parser reads the XML document and the corresponding DTDs, schemes.
 3. The parser checks if the XML document is well-formed and valid.
 4. Parser offers an application interface with functions like „ListElements()“.
 5. The application accesses the elements of the XML document using the available interfaces, and processes the received data.
 6. The application saves the modified/updated XML document.

Processing of XML-Documents



- There are two types of parsers:
 - Document Object Model (DOM)
 - Simple API for XML (SAX)
- **DOM parsers** load all elements in the memory and create a tree data structure, which can be then processed.
- **SAX-Parsers** navigate through a document offering only parts of its contents without loading it completely into memory.

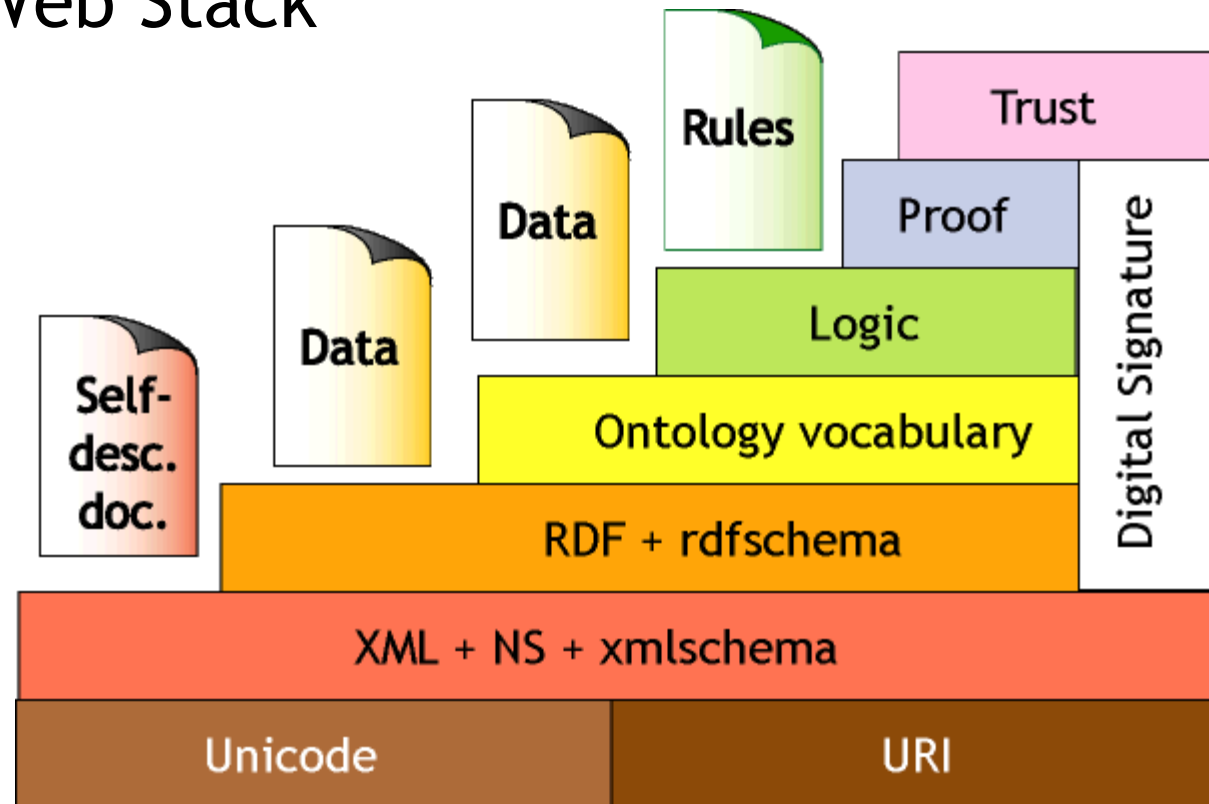
- Comparison of DOM and SAX-Parser
 - SAX is able to parse files of any size.
 - SAX is efficient, if only parts of the file are relevant.
 - SAX is easy to use.
- DOM allows free access and changes to a document.
- DOM creates a full image of the document in memory.

Typical application of DOM und SAX parsers:

- **DOM parsers** are useful when editing entire documents at once. For instance, for editing a structured text in a word processor.
- **SAX parsers** are useful for quick retrieval of records, e.g. for accessing addresses in an XML-based customer database.

- From HTML to XML
- XML-Concept
- Processing of XML Documents
- XML Example Applications

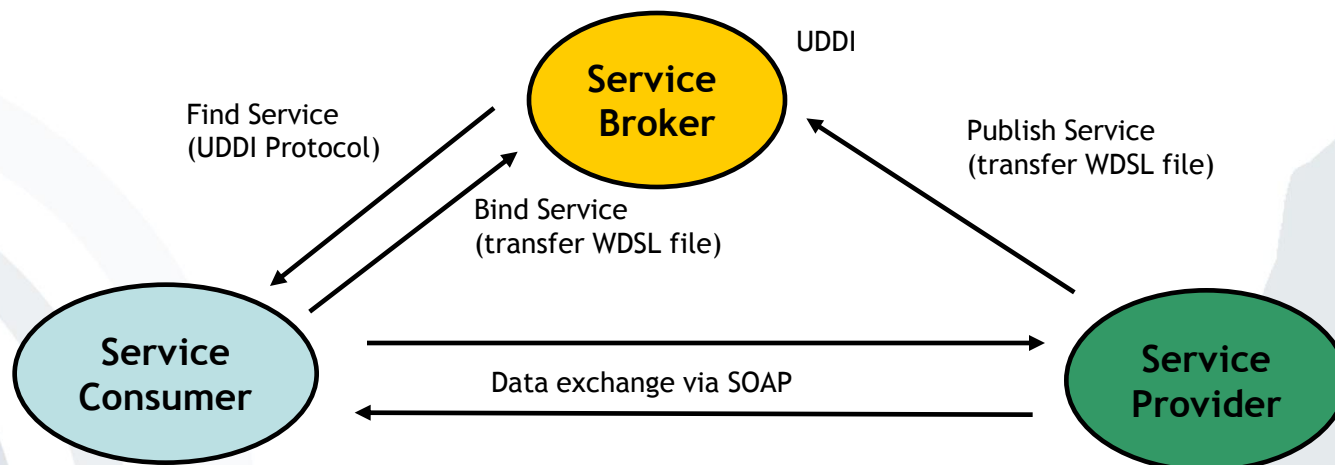
- Semantic Web Stack



Source: Tim Berners-Lee (W3C), <http://www.w3.org/2000/Talks/1206-xml2k-tbl/slide10-0.html>

- The term *Web Service* describes a standardised way of integrating Web-based applications using the XML, SOAP, WSDL and UDDI over the Internet protocol.
 - XML is used to tag the data,
 - SOAP is used to transfer the data,
 - WSDL is used for describing the services available and
 - UDDI is used for listing what services are available.

Source: www.webopedia.com/TERM/W/Web_services.html



- OFX, OFE: Open Financial Exchange for Finance Information (www.ifxforum.org)
- MathML: Mathematical formula description language (www.w3.org/Math)
- SAML: Security Assertion Mark-up Language for exchanging authentication and authorisation information (www.oasis-open.org)
- EPAL: Enterprise Privacy Authorisation Language is a formal language to specify fine-grained enterprise privacy policies (<http://www.zurich.ibm.com/security/enterprise-privacy/epal/>)
- ...



- Tim Berners-Lee (2000), W3C Talk,
Internet: www.w3.org/2000/Talks/1206-xml2k-tbl/slide10-0.html, last visited 2011-04-01.
- Webopedia,
Internet: www.webopedia.com/TERM/W/Web_services.html,
last visited 2011-04-01.

