

According to <http://www.w3.org/>



The **Semantic Web** is about two things:

- It is about common formats for interchange of **data**, where on the original Web we only had interchange of **documents**.
- It is about languages for recording how data relates to real world objects.

That allows a person, or a machine, to start off in one database, and then move through an unending set of databases which are connected not by wires but by being about the same thing.

The Web was designed as an information space, with the goal that it should be useful not only for human-human communication,

but also that **machines** would be able to participate and help.

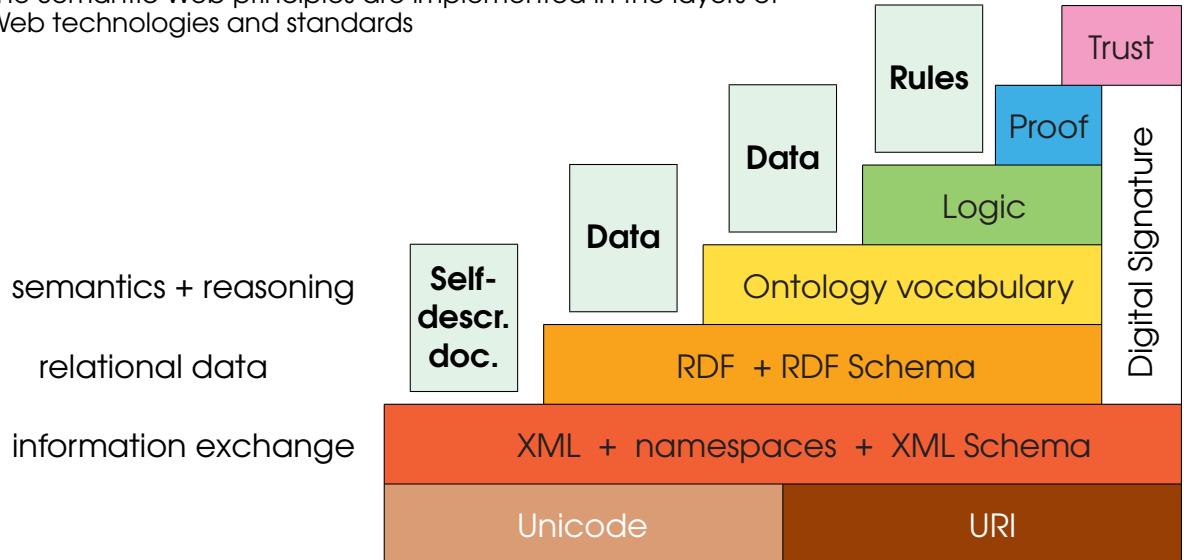
A major obstacle to this is the fact that most information on the Web is designed for **human** consumption, and even if it was derived from a database with well defined meanings, the structure of the data is not evident to a robot browsing the Web.

The Semantic Web approach develops languages for expressing information in a **machine processable form**

Layered approach to the Semantic Web

(by T. Berners-Lee)

The Semantic Web principles are implemented in the layers of Web technologies and standards



If HTML and the Web made all the online documents look like one huge **book**,
RDF, schema, and inference languages will make all the data in the world look like
one huge **database** — Weaving the Web, 1999

Unicode and URI

- **Unicode** is an industry standard designed to allow text and symbols from **all of the writing systems of the world** to be consistently represented and manipulated by computers. For details visit <http://www.unicode.org/>
- A **Uniform Resource Identifier (URI)** is a string of characters used to identify or name a resource (http://www.w3.org/Addressing/URL/URI_Overview.html). A URI can be classified as a locator or a name or both:

- A Uniform Resource Locator (URL) is a URI that, in addition to identifying a resource, provides means of obtaining a representation of the resource by describing its primary access mechanism or network 'location.' E.g., the URL <http://www.bbc.com/> is a URI that identifies a resource (BBC's home page) and implies that a representation of that resource is obtainable via HTTP from a network host named www.bbc.com
- A Uniform Resource Name (URN) is a URI that identifies a resource by name in a particular namespace. A URN can be used to talk about a resource without implying its location. E.g., the URN **urn:isbn:0-395-36341-1** is a URI that, like an International Standard Book Number (ISBN), allows one to talk about a book, but doesn't suggest where and how to obtain an actual copy of it

Structured Web documents

SGML — standard generalised markup language:

Historically, electronic manuscripts contained control codes or macros that caused the document to be formatted in a particular way ('specific coding'). In contrast, generic coding, which began in the late 1960s, uses **descriptive tags** (for example, 'heading,' rather than 'format-17'). Also in the late 1960s, New York book designer S. Rice proposed the idea of a universal catalog of parameterised 'editorial structure' tags.

In 1969, C. Goldfarb was leading an IBM research project on integrated law office information systems. Together with E. Mosher and R. Lorie he invented the Generalized Markup Language (GML) as a means of allowing the text editing, formatting, and information retrieval subsystems to share documents. Instead of a simple tagging scheme GML introduced the concept of a

formally-defined document type with an explicit nested element structure.

The first working draft of the **SGML standard** was published in 1980.

For details consult, e.g., <http://www.w3.org/MarkUp/SGML/> and
<http://www.isgmlug.org/sgmlhelp/g-index.htm>

HTML

HTML (hypertext markup language) is an SGML application.

HTML is a **markup language** designed for the creation of web pages with hypertext and other information to be displayed in a web browser. HTML is used to structure information — denoting certain text as headings, paragraphs, lists and so on — and can be used to describe the **appearance** of a document.

It describes information as collections of documents connected by **hyperlinks**. Originally defined by Tim Berners-Lee, HTML is now an international standard. Later HTML specifications are maintained by the W3C; see <http://www.w3.org/MarkUp/>

```
<h2>A Semantic Web Primer</h2>
<i>by <b>G. Antoniou</b> and <b>F. van Harmelen</b> </i>
<br />The MIT Press
<br />ISBN 0-262-01210-3
```

- Do you understand the meaning of the piece above?
- What about machines?

HTML (cont.)

- Human reading:

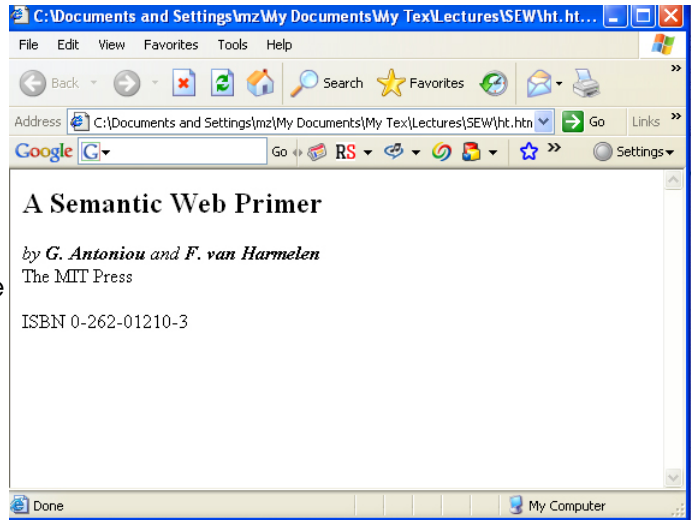
"A Semantic Web Primer" is a book written by G. Antoniou and F. van Harmelen and published by the MIT Press. Its ISBN is 0-262-01210-3.

- Machine 'reading':

Can the machine 'understand' that "A Semantic Web Primer" is the **title**?

Can the machine 'understand' that G. Antoniou and F. van Harmelen are the **authors** of this book?

How can we query such documents?



- HTML documents simply **display** information and **links** to other documents.
- HTML is based on a **fixed** set of tags.

XML (eXtensible Markup Language)

- XML is another **SGML** application (see <http://www.w3.org/XML/>
http://www.w3schools.com/xml/xml_what_is.asp)
 - XML is based on **tags**
 - tags may be nested
 - tags must be closed
- XML documents give more structural information about their pieces and relations between them through the **nesting structure**

`<booktitle>A Semantic Web Primer</booktitle>`
element

```
<book>  
  <title>A Semantic Web Primer</title>  
  <author>G. Antoniou</author>  
  <author>F. Van Harmelen</author>  
  <publisher>The MIT Press</publisher>  
  <ISBN>0-262-01210-3</ISBN>  
</book>
```

E.g., the *author* element 'refers' to the enclosing *book* element.

XML (cont.)

- XML allows the representation of information that is also **machine-accessible**
- XML **separates** content from formatting
(XML was designed to carry data, not to display data)
- XML is a **metalanguage** for markup:
it doesn't have a fixed set of tags but allows users to define tags of their own
XML applications (extensions) for various domains:
MathML (mathematics), BSML (bioinformatics), NewsML etc.
- XML is not a single markup language that can be extended for other uses, but rather it is a **common notation** that markup languages can build upon.
(You can define your own markup languages,
e.g., for describing recipes, football players, etc.)
- XML was designed to transport and store data
- XML can serve as a **uniform data exchange format**



The XML syntax

```
<?xml version="1.0" encoding="UTF-16"?> ← prolog
<email>
  <head> ← opening tag
  { <from address="michael@dcs.bbk.ac.uk">Michael Zakharyashev</from>
    <to address="mark@dcs.bbk.ac.uk">Mark Levene</to>
    <subject>REF impact</subject>
  }
  </head> ← closing tag
  <body>
    <!-- the actual content is here --> ← comment
  </body>
</email>
```

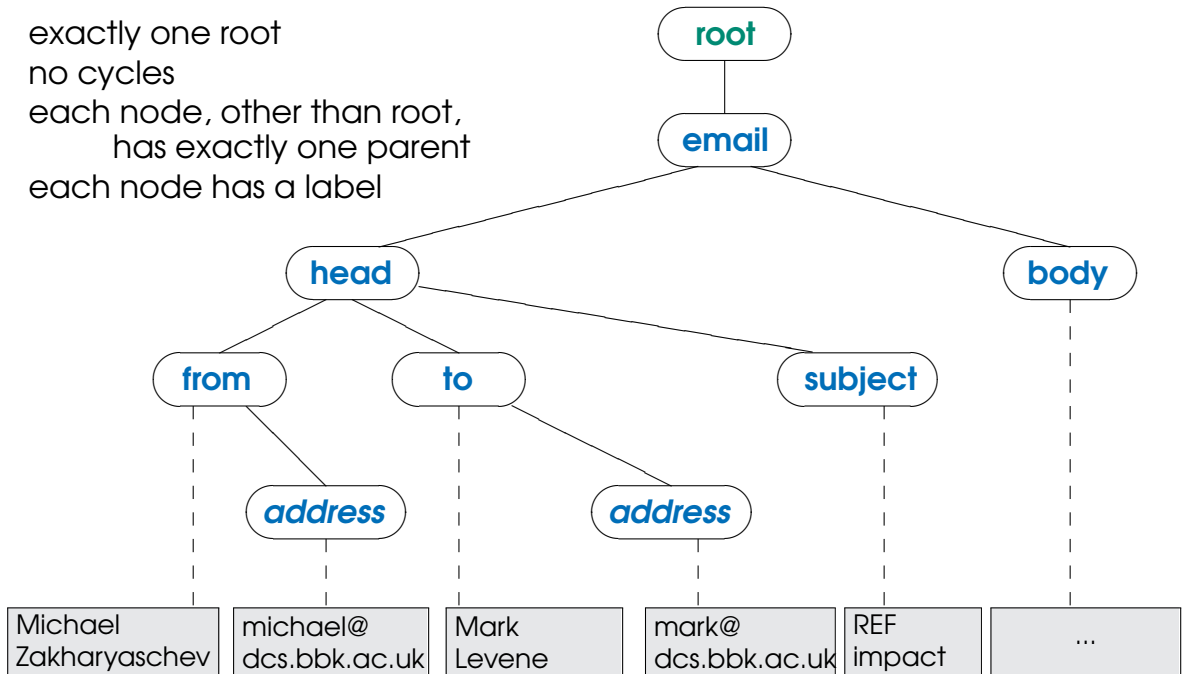
attribute

Syntactic correctness: **well-formed XML documents**

NB. XML syntax is not part of this module, but you should be able to read XML documents

The tree model of XML documents

- exactly one root
- no cycles
- each node, other than root, has exactly one parent
- each node has a label



the **order** of elements is **important**

Structuring XML documents

Imagine two applications that try to **communicate**, and that they wish to use the same **vocabulary**.

For this purpose it is necessary to define

- all the element and attribute names that may be used;
- the structure:
 - what values an attribute may take;
 - which elements may or must occur within other elements;
 - etc.

An XML document is **valid** if it is well-formed, uses structuring information and **respects** that **structuring information**.

There are two ways of defining the structure of XML documents:

- Document Type Definition (DTD), used already for SGML documents
- XML Schema, a significantly richer language
 - based on **XML** (DTD uses a separate syntax);
 - provides a sophisticated **set of data types** (DTD is limited to strings only);
 - allows one to define **new types** by extending or restricting existing ones.

XML Schema: example

XML-schema is a document describing the valid format of XML data-sets: what elements are (not) allowed at any point, their attributes, the number of occurrences, etc.

```
<schema xmlns="http://www.w3.org/2001/XMLSchema" version="1.0">
  <element name="email" type="emailType" />
  <complexType name="emailType">
    <sequence>
      <element name="head" type="headType" />
      <element name="body" type="bodyType" />
    </sequence>
  </complexType>
  <complexType name="headType">
    <sequence>
      <element name="from" type="addressType" />
      <element name="to" type="addressType" minOccurs="1"
        maxOccurs="unbounded" />
      <element name="subject" type="string" />
    </sequence>
  </complexType>
  ...
</schema>
```

XML Schema

- elements `<element name="..." ... />`
 - `type="..."`
 - `minOccurs="x"`, where **x** may be any natural number (1 by default)
 - `maxOccurs="x"`, where **x** may be any natural number or **unbounded** (1 by default)

- attributes `<attribute name="..." ... />`
 - `type="..."`
 - `use="x"`, where **x** may be **optional** or **required**
 - `default="..."`
 - the attribute is to appear unconditionally with the supplied value used whenever the attribute is not actually present
 - `fixed="..."`
 - the attribute value if present must equal the supplied constraint value, and if absent receives the supplied value as for default

XML Schema: data types

- built-in data types:
 - numerical: **integer**, **short**, **byte**, **long**, **float**, **decimal**
 - string: **string**, **ID**, **IDREF**, **language**
 - date and time: **time**, **date**, **dateTime**, ...

User-defined data types:

- simple data types (see slide 16) **(cannot have attributes)**
- complex data types
 - **sequence** — a sequence of existing data type elements,
the appearance of which in a predefined order is important
 - **all** — a collection of elements that must appear,
but the order of which is not important
 - **choice** — a collection of elements, of which one will be chosen.

XML Schema: data type extension

```
<complexType name="lecturerType">
  <sequence>
    <element name="firstname" type="string"
      minOccurs="0" maxOccurs="unbounded" />
    <element name="lastname" type="string" />
    <element name="email" type="string" minOccurs="0" maxOccurs="1" />
  </sequence>
  <attribute name="title" type="string" use="optional" />
  <attribute name="rank" type="string" />
</complexType>

+

<complexType name="extendedLecturerType">
  <extension base="lecturerType">
    <sequence>
      <element name="email" type="string" minOccurs="0" maxOccurs="1" />
    </sequence>
    <attribute name="rank" type="string" />
  </extension>
</complexType>
```

Instances of the extended type are also instances of the original type

XML Schema: data type restriction

```
<complexType name="restrictedLecturerType">
  <restriction base="lecturerType">
    <sequence>
      <element name="firstname" type="string" minOccurs="1" maxOccurs="2" />
    </sequence>
    <attribute name="title" type="string" use="required" />
  </restriction>
</complexType>
```

Instances of the restricted type are also instances of the original type

```
<simpleType name="dayOfMonth">
  <restriction base="integer">
    <minInclusive value="1" />
    <maxInclusive value="31" />
  </restriction>
</simpleType>
```

```
<simpleType name="dayOfWeek">
  <restriction base="string">
    <enumeration value="Sun" />
    <enumeration value="Mon" />
    ...
    <enumeration value="Sat" />
  </restriction>
</simpleType>
```

XML Schema: more on data types

Consider the following XML element

```
<internationalPrice currency="GBP">423.46</internationalPrice>
```

How can one define an XML Schema description for this element?

```
<element name="internationalPrice">  
  <complexType > ← anonymous complex type  
    <simpleContent > ← extending simple type  
      <extension base="decimal">  
        <attribute name="currency" type="string" />  
      </extension>  
    </simpleContent > ← indicates that the type may contain  
  </complexType > ← only character data and no elements  
</element >
```

Namespaces

Problem: name clashes

Solution: different prefix for each schema

prefix:name

```
<?xml version="1.0" encoding="UTF-16"?>
<vu:instructors xmlns:uky="http://www.uky.edu/schema"
                xmlns:gu="http://www.gu.au/schema"
                xmlns:vu="http://www.vu.com/schema">
  <uky:faculty uky:title="lecturer" uky:name="John Smiths" />
  <gu:academicStaff gu:title="lecturer" gu:name="Mate Jones" />
</vu:instructors>
```

- namespace declaration `xmlns:prefix="location"`
- location used by default `xmlns="location"`

A **namespace** is a context in which a group of one or more identifiers might exist.
An identifier defined in a namespace is associated with that namespace.

What can XML do?

So far:

XML Does not DO Anything and was not designed to DO anything!
XML document is just pure information wrapped in XML tags.

XSL (EXtensible Stylesheet Language) is a family of recommendations for defining XML document transformation and presentation.

XSL is an XML application. It consists of three parts:

- XSLT — a language for transforming XML documents
- XPath — a language for navigating in XML documents
- XSL-FO — a language for formatting XML documents

For more information visit:

http://www.w3schools.com/xsl/xsl_languages.asp

<http://www.w3.org/Style/XSL/>

Querying XML documents: XPath

<http://www.w3.org/TR/xpath>, <http://www.w3schools.com/xpath/default.asp>

XPath expressions

- `/library/author`
- `//author`
(anywhere in the document)
- `/library/@location`
(attribute node)
- `//book[@title="Bestiario"]`
(attribute node)
- `//book[@title="Bestiario"]`
(filter expression;
all books with the title)
- `//author[1]`
(the first node)
- `//author[1]/book[last()]`
(the last node)
- `//book[not @title]`
(all books without a title)

```
<?xml version="1.0" encoding="UTF-16"?>
<library xmlns="http://www.lib.org/schema"
        location="Bremen">
  <author name="Jorge Luis Borges">
    <book title="Labyrinths" />
    <book title="Doctor Brodie's Report" />
    <book title="The Garden of Forking Paths" />
  </author>
  <author name="Adolfo Bioy Casares">
    <book title="The Invention of Morel" />
  </author>
  <author name="Julio Cortázar">
    <book title="Bestiario" />
    <book title="Un tal Lucas" />
  </author>
  ...
</library>
```

see next page for explanations

XPath syntax

1. `/library/author` addresses all **author** elements that are children of the **library** element, which resides immediately below the root
2. `//author` `//` means that we should consider **all** elements in the document and check whether they are of type **author**
3. `/library/@location` symbol `@` is used to denote attribute nodes
4. `//book/@title="Bestiario"` addresses all **title** attribute nodes within **book** elements anywhere in the document with value `"Bestiario"`
5. `//book[@title="Bestiario"]` addresses all books with title `"Bestiario"`
6. `//author[1]` addresses the first **author** element node in the document
7. `//author[1]/book[last()]` addresses the last **book** element within the first **author** element node in the document
8. `//book[not @title]` addresses all **book** element nodes without a **title** attribute

XML: summary

- XML is a metalanguage that allows users to define markup for their documents using **tags**, a standard syntax for **meta data**.
- Nesting of tags introduces **structure**. The structure of documents can be enforced using **schemas** or DTDs.
- XML **separates** content and structure from formatting.
- XML is the **de facto standard** for the representation of structured information on the Web and supports machine processing of information.
- XML is supported by **query languages**.
- XML creates application-independent documents and data.
- XML has become the universal syntax for exchanging data between organisations.

The impact of XML has been so extensive that it can be considered as a **data revolution**.

- But is XML powerful enough for the purposes of the Semantic Web?