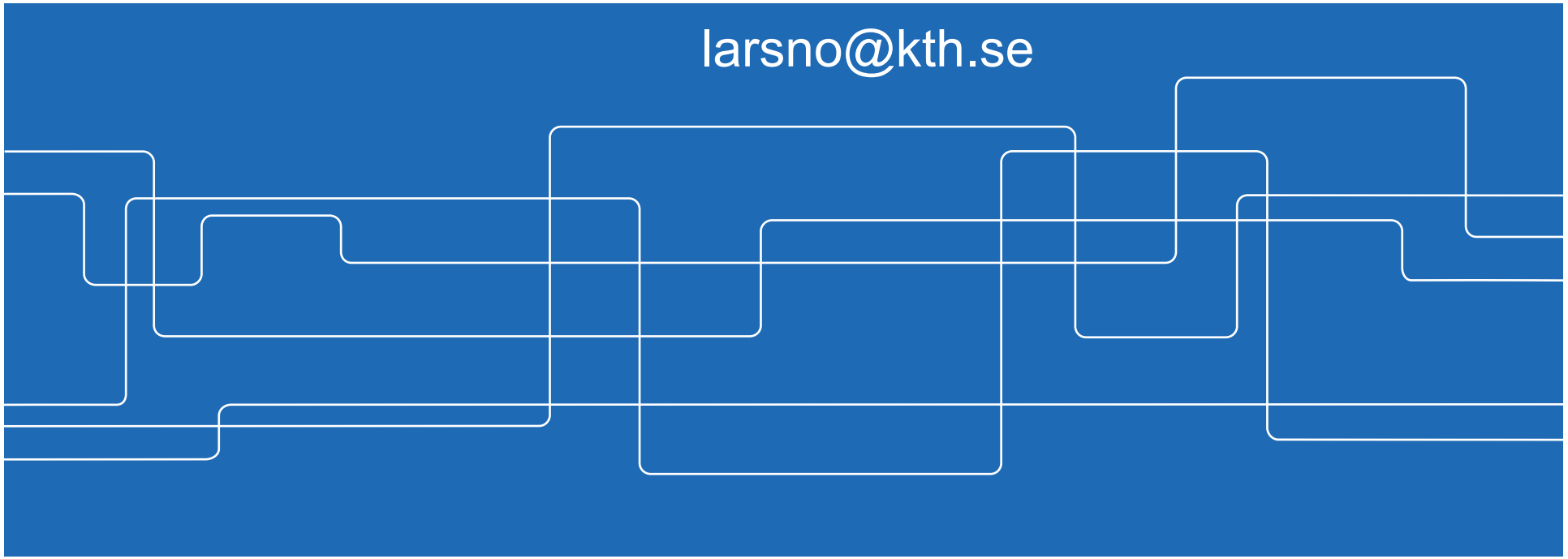
Java and XML parsing

EH2745 Lecture #6
Spring 2016

larsno@kth.se



Lecture Outline

- Quick Review
- Why are we doing this
- The XML ”language”
- Parsing Files

Quick Review

We have in the first set of Lectures covered the basics of the Java Language. Next is to make something useful with it.

Like learning a new human language requires practice, so does programming

I/O and Files in Java

My Java Program

JVM (Packages)

OS (Linux, OSX, Win..)

HW (CPU, RAM, HDD, I/O)

Accessing I/O is built on the `InputStream and FileStream`

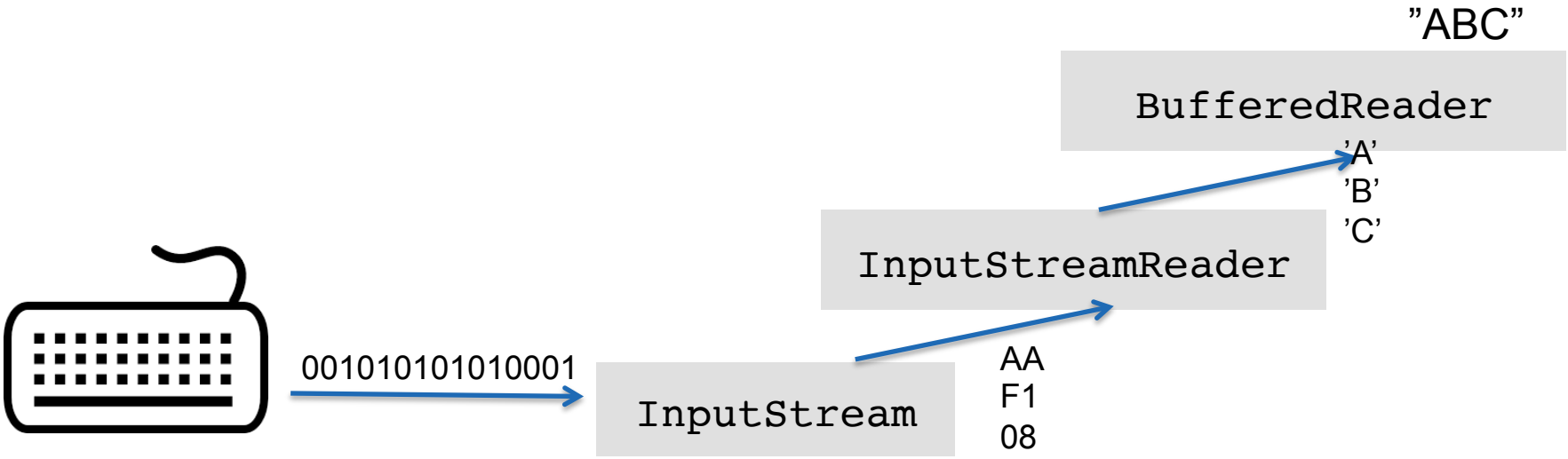
For keyboard input, this comes from **`System.`**`in` to which the `InputStream` is pointed.

`InputStream` is enhanced in `InputStreamreader` which maps the read Bytes to Characters

For file input, the `FileStream` is handed a filename

Enhanced reading

Both `InputStreamreader` and `FileReader` can be used by a `BufferedReader`, which reads sets of characters (e.g a line)




```
FileReader fr = new FileReader(“readme.txt”);
BufferedReader br = new BufferedReader(fr);
```

Parsing

Parsing involves analysing a string of characters (or words) and and breaking them down into components.

```
Likeifapersonreadsthislongparagraphoftextwhic
hactuallyhasseveralwordsinitbutsinceitiswritt
enwithoutanyseparatorsitisdifficulttoseethese
wordsfortunatelyapersonisreasoanblyintelligen
tandcanreadthisentenceanywayacomputercannotdo
sohoweversinceitisnotintelligentitneedssomewa
yoftellingitwhichcharactersshouldbegroupedint
ogroups(words)thatcanbestoredinitsmemoryandus
edintheprogram
```

Parsing - How to convert read lines to data the program can work with

5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa

Comma Separate Values (CSV) is a straifghtforward way to store the data in a file
The program(mer) needs to know the structure of the file.

```
String SplitBy = ",";
List<Flower> flowerList = new ArrayList<Flower>();
br = new BufferedReader(new FileReader(dataFile)
while ((line = br.readLine()) != null) {
        String[] flower = line.split(SplitBy);
        double[] param = {0.0,0.0,0.0,0.0};
        for (int j = 0; j < 4; j++) {
           param[j] = Double.parseDouble(flower[j]);
        }
flowerList.add(new Flower(param,flower[4]));
}
```

Is the file format important?

5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa

Is the file format important?

```
<Flower>
        <Sepal Length>5.0</Sepal Length>
        <Sepal Width>3.6</Sepal Width>
        <Petal Length>0.8</PetalLength>
        <Petal Width>0.2</Petal Width>
        <Species>Iris-Setosa</Species>
</Flower>
<Flower>
        <Sepal Length>5.0</Sepal Length>
        <Sepal Width>3.6</Sepal Width>
        <Petal Length>0.8</PetalLength>
        <Petal Width>0.2</Petal Width>
        <Species>Iris-Setosa</Species>
</Flower>
<Flower>
        <Sepal Length>5.0</Sepal Length>
        <Sepal Width>3.6</Sepal Width>
        <Petal Length>0.8</PetalLength>
        <Petal Width>0.2</Petal Width>
        <Species>Iris-Setosa</Species>
</Flower>
```

Which is the better choice?

- **Comma Separated Values**
  - Pros
    - Little extra data has to be sent (only the commas)
  - Cons:
    - Data has to arrive in the right order
- **Using Tags(XML)**
  - Pros:
    - Flexible format (data can arrive out of order)
    - Thanks to tags we can search for data
    - People can read it!!!
  - Cons:
    - Verbose – a lot of overhead!

XML – eXtensible Markup Language

```
<?xml version="1.0" encoding="UTF-8"?>
<note>
        <to>Tove</to>
        <from>Jani</from>
        <heading>Reminder</heading>
        <body>Don't forget me this weekend!</body>
</note>
```

XML is (M)a(ny) standard developed by W3C

- XML Core Working Group:
  - XML 1.0 (Feb 1998), 1.1 (candidate for recommendation)
  - XML Namespaces (Jan 1999)
  - XML Inclusion (candidate for recommendation)
- XSLT Working Group:
  - XSL Transformations 1.0 (Nov 1999), 2.0 planned
  - XPath 1.0 (Nov 1999), 2.0 planned
  - eXtensible Stylesheet Language XSL(-FO) 1.0 (Oct 2001)
- XML Linking Working Group:
  - XLink 1.0 (Jun 2001)
  - XPointer 1.0 (March 2003, 3 substandards)
- XQuery 1.0 (Nov 2002) plus many substandards
- XMLSchema 1.0 (May 2001)

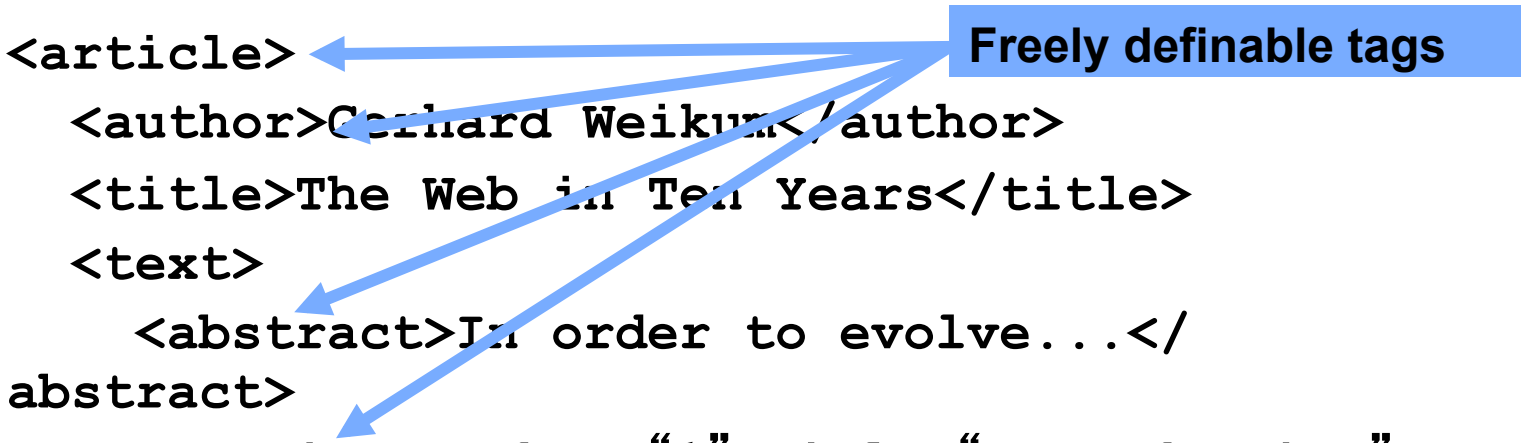
Source: XML for Beginners, Ralf Schenkel April 2003.

A Simple XML Document

```
<article>
  <author>Gerhard Weikum</author>
  <title>The Web in Ten Years</title>
  <text>
    <abstract>In order to evolve...</
abstract>
    <section number="1" title="Introduction">
      The <index>Web</index> provides the
universal...
    </section>
  </text>
</article>
```

Source: XML for Beginners, Ralf Schenkel April 2003.

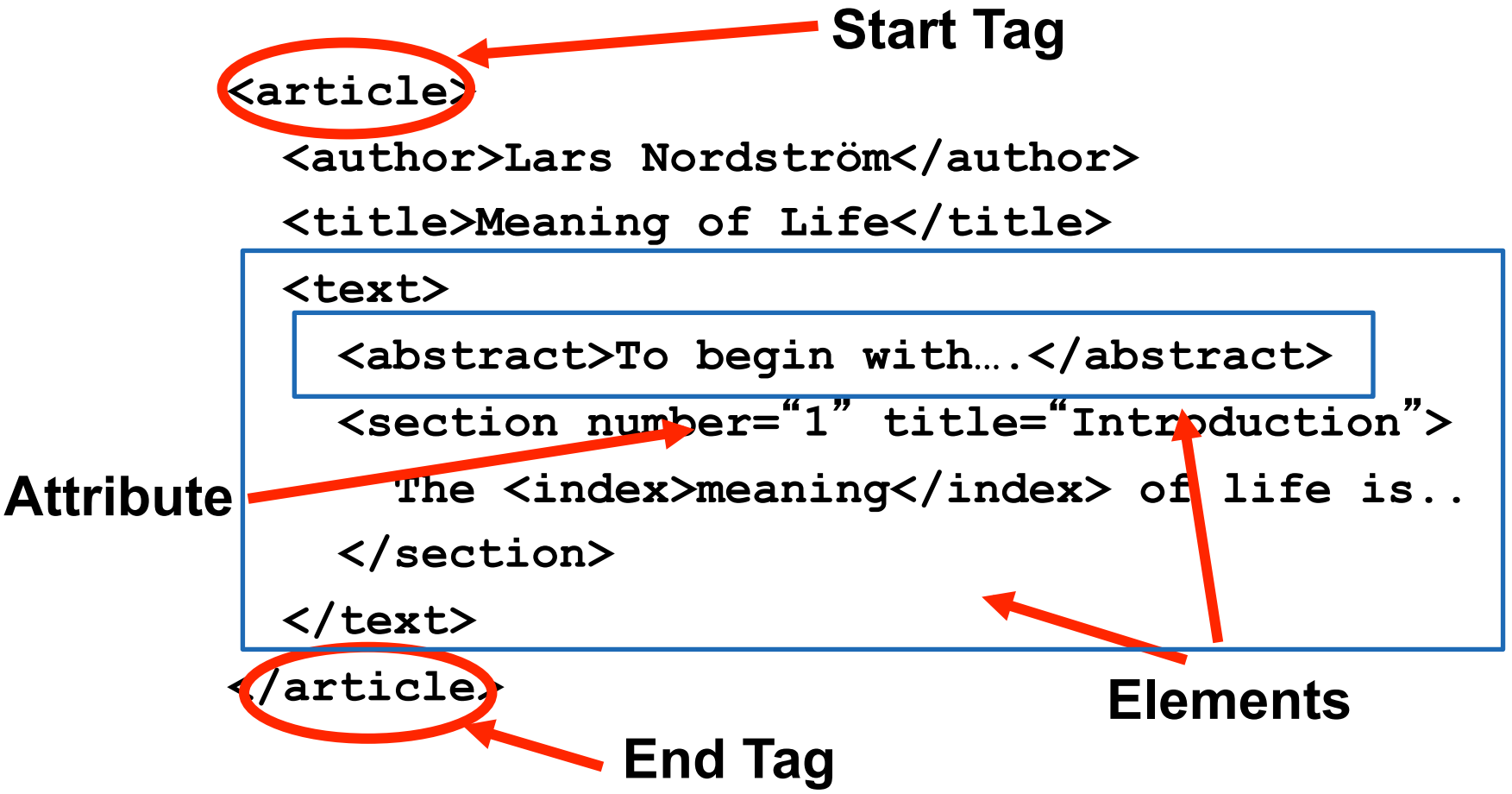
A Simple XML Document




```
<article>
  <author>Gerhard Weikum</author>
  <title>The Web in Ten Years</title>
  <text>
    <abstract>In order to evolve...</
abstract>
    <section number="1" title="Introduction">
      The <index>Web</index> provides the
universal...
    </section>
  </text>
</article>
```

Source: XML for Beginners, Ralf Schenkel April 2003.

A Simple XML Document



A Simple XML Document

```
<article>
  <author>Gerhard Weikum</author>
  <title>The Web in Ten Years</title>
  <text>
    <abstract>In order to evolve...</abstract>
    <section number="1" title="Introduction">
      The <index>Web</index> provides the universal...
    </section>
  </text>
</article>
```

Attributes with **name** and **value**

Elements in XML Documents

(Freely definable) **tags**: `article`, `title`, `author`

- with start tag: `<article>` etc.
- and end tag: `</article>` etc.

Elements: `<article> ... </article>`

Elements have a **name** (`article`) and a **content** (. . .)

Elements may be nested.

Elements may be empty: `<this_is_empty/>`

Element content is typically parsed character data (PCDATA), i.e., strings with special characters, and/or nested elements (*mixed content* if both).

Each XML document has exactly one root element and forms a tree.

Elements with a common parent are ordered.

Elements vs. Attributes

Elements may have **attributes** (in the start tag) that have a **name** and

a **value**, e.g. `<section number="1">`.

What is the difference between elements and attributes?

Only one attribute with a given name per element (but an arbitrary number of subelements)

Attributes have no structure, simply strings (while elements can have subelements)

As a *rule of thumb*:

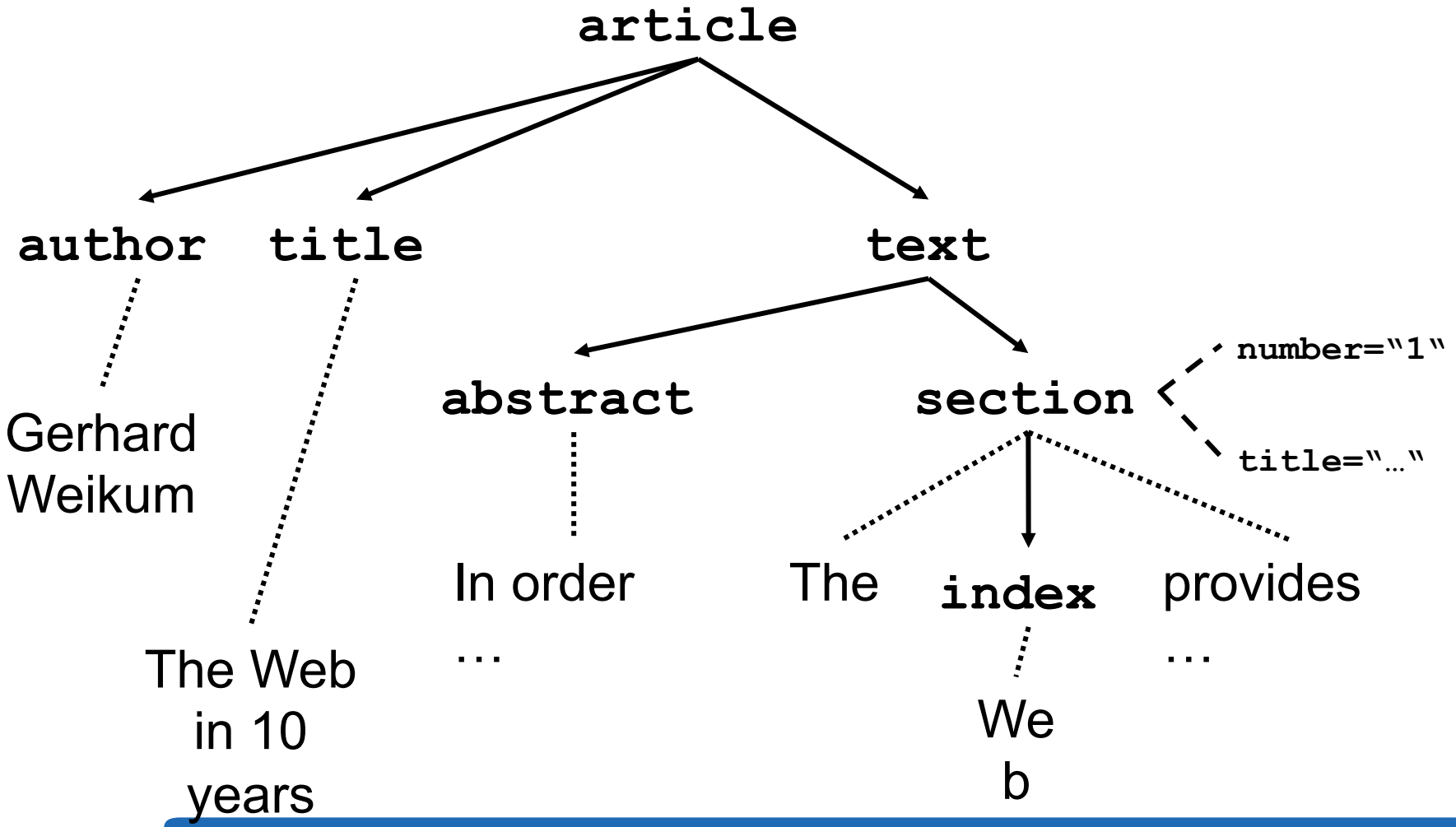
Content into elements

Metadata into attributes

Example:

```
<person born="1912-06-23" died="1954-06-07">
Alan Turing</person> proved that…
```

XML Documents as Ordered Trees



More on XML Syntax

Some special characters must be escaped using **entities:**

< → <

& → &

(will be converted back when reading the XML doc)

Some other characters may be escaped, too:

> → >

" → "

' → '

Well-Formed XML Documents

A **well-formed** document must adher to, among others, the following rules:

Every start tag has a matching end tag.

Elements may nest, but must not overlap.

There must be exactly one root element.

Attribute values must be quoted.

An element may not have two attributes with the same name.

Comments and processing instructions may not appear inside tags.

No unescaped **<** or **&** signs may occur inside character data.

Well-Formed XML Documents

A **well-formed** document must adher to, among others, the following rules:

Every start tag has a matching end tag.

Elements may nest, but must not overlap

ame.

No unescaped < or & signs may occur inside character data.

Only well-formed documents can be processed by XML parsers.

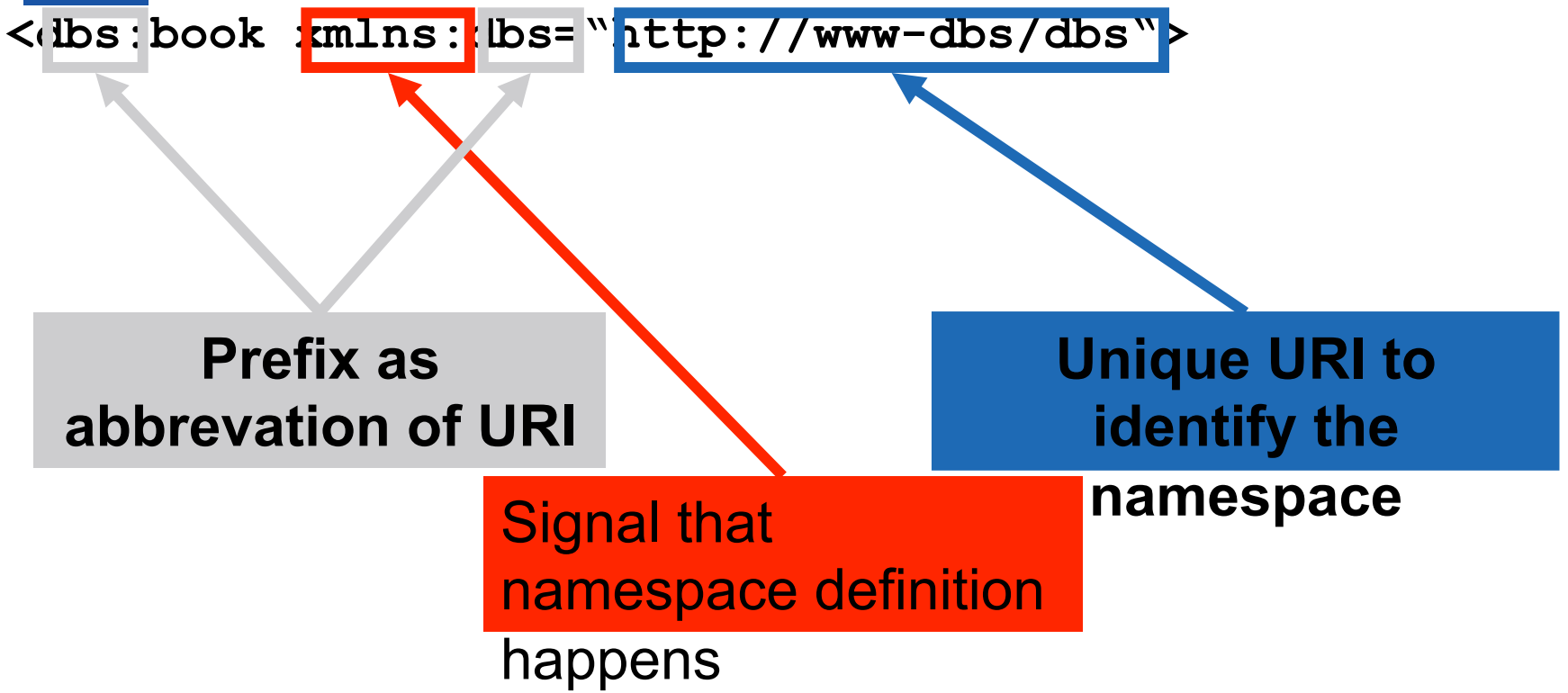
2.3 Namespaces

```
<library>
  <description>Library of the CS Department</description>
  <book bid="HandMS2000">
    <title>Principles of Data Mining</title>
    <description>
      Short introduction to <em>data mining</em>, useful
      for the IRDM course
    </description>
  </book>
</library>
```

Semantics of the `description` element is ambigous
Content may be defined differently
Renaming may be impossible (standards!)

⇒ Disambiguation of separate XML applications using unique prefixes

Namespace Syntax



Namespace Example

```
<dbs:book xmlns:dbs="http://www-dbs/dbs">
  <dbs:description> ... </dbs:description>
  <dbs:text>
    <dbs:formula>
      <mathml:math xmlns:mathml="http://www.w3.org/1998/
Math/MathML">
        ...
      </mathml:math>
    </dbs:formula>
  </dbs:text>
</dbs:book>
```

Default Namespace

Default namespace may be set for an element and its content (but *not* its attributes):

```
<book xmlns="http://www-dbs/dbs">
  <description>...</description>
<book>
```

Can be overridden in the elements by specifying the namespace there (using prefix or default namespace)