

Web Data Models

Silviu Maniu

Introduction: XML, JSON, parsing, validation



Comprendre le monde,
construire l'avenir



Data Exchange Problem

- we are in an age of **Data**
- types of data: textual, medical, legal, GPS, ...
- computers need to communicate — data need to be readable by anyone
- how to keep long-term access to data?

Web Data Models

- **Objective of the course:** study data models that are interchangeable and are extensively used on the Web
- We will concentrate on XML, JSON and algorithms for evaluating / processing data
- **Not studying:** graph models, systems and data architectures

Web Data Models

- **When/Where:** Mondays, 9h00–12h15, PUIO (PSud)
- **Who:** Silviu Maniu, U. Paris-Sud silviu.maniu@lri.fr
- **Site:** <http://silviu.maniu.info/teaching/>
- **Evaluation:** first session 50% Project + 50% Exam; project deadline the week after the exam; second session (if grade <10): 100% Exam

Web Data Models

- Web Data Models: XML, JSON
- XML Validation: DTD, XML Schema, RelaxNG
- Processing XML: XPath, XQuery, XSLT

Data Exchange

- **Why?** To ensure programs read and use data **from other programs**

Data Exchange

- **How can we encode data?**

Data Exchange: Unstructured Data (Text/Binary)

Theory of Computation
Michael Sipser
Cengage Learning
2012
3

Artificial Intelligence
Stuart Russell
Peter Norvig
Pearson
2013
3

```
kXv00b0z000000I040afgf007对 XAN0
03A+000}k(0c000000p000$HJPV0.000B0000S000/%2E[M00-
0A0_0A00000i00=01+0p01.
"Hqh0v01IKF
09+c000-000kr000 00_0v)00mw0i0; r200}BE0000u>,E000^
000
0005L0I0:x000;000c 0J00S0
0600mX00000+0(0%
000010R00

osp0;Y-0000=0000Lu<M000Vj00X}00]Y0400$00.a0/0M0h0
H90N0P_000;000H0$J0Xe0-0000NJ0D000n!0000#00<Z
r>00w00y0x000e0;p000LJI~^t00;0%:01PLR&0
00)t0)00t0-
x0|0060h00g0040K0Y 0{aE0G00=0k010X00<0_00$xc000t0
0$9C0j*%>00u$0B\sa0E8t00t00T400>0vAj r0-I0Q50Ti
04p00hC00000aVu00B,C1E<0b0f=G00H0$BMA
000
0{0000yE`I0pt000000L00MMX00000?0GK#`qQ00^00600
0\P1 0{02t0ae0w)0*00]w0*.00R4>0000700=000X!0YK|00N
;`0:fV豈#t000g{7M0;0700000000s0l 0000h-0>09]000v00
e00000{0h>0 H
```

- What **is the format**? What are the fields?

text and binary, theory of Computation : name of the book, Machael Sisper : author, 3 : edition 2012 : year

- Need to **send the file and the parser.**

Data Exchange: Relational Database

Books

id	title	editor	year
1	Theory of Computation	Cengage	2012
2	Artificial Intelligence	Pearson	2013

Authors

id	first	last
1	Sipser	Michael
2	Russell	Stuart
3	Norvig	Peter

BookAuthor

author	book
1	1
2	2
3	2

- Fixed schema, strict typing

Data Exchange

- How can we encode data for **easy interchange**?

Two desiderata:

两个想法:

1. 易于阅读/解析 (交换)
2. 容易检测结构 (解析)

1. easy to read / parse (exchange)

2. easy to detect the structure (parse)

Semi-Structured Data (SSD)

- 尝试协调文档视图（文本，HTML）和DB中的严格结构
- 组织在语义实体中：类似实体分组在一起，同一组中的实体可能不具有相同的字段
- 定义为（可能嵌套的）集合的字段值
- 字段值的顺序不重要！

- an attempt to reconcile document view (text, HTML) and the strict structures in DB
- organized in **semantic entities**: similar entities grouped together, entities in the same group may not have same fields
- defined as **(possibly nested) set of field-value pairs**
- **order of fields not important!**

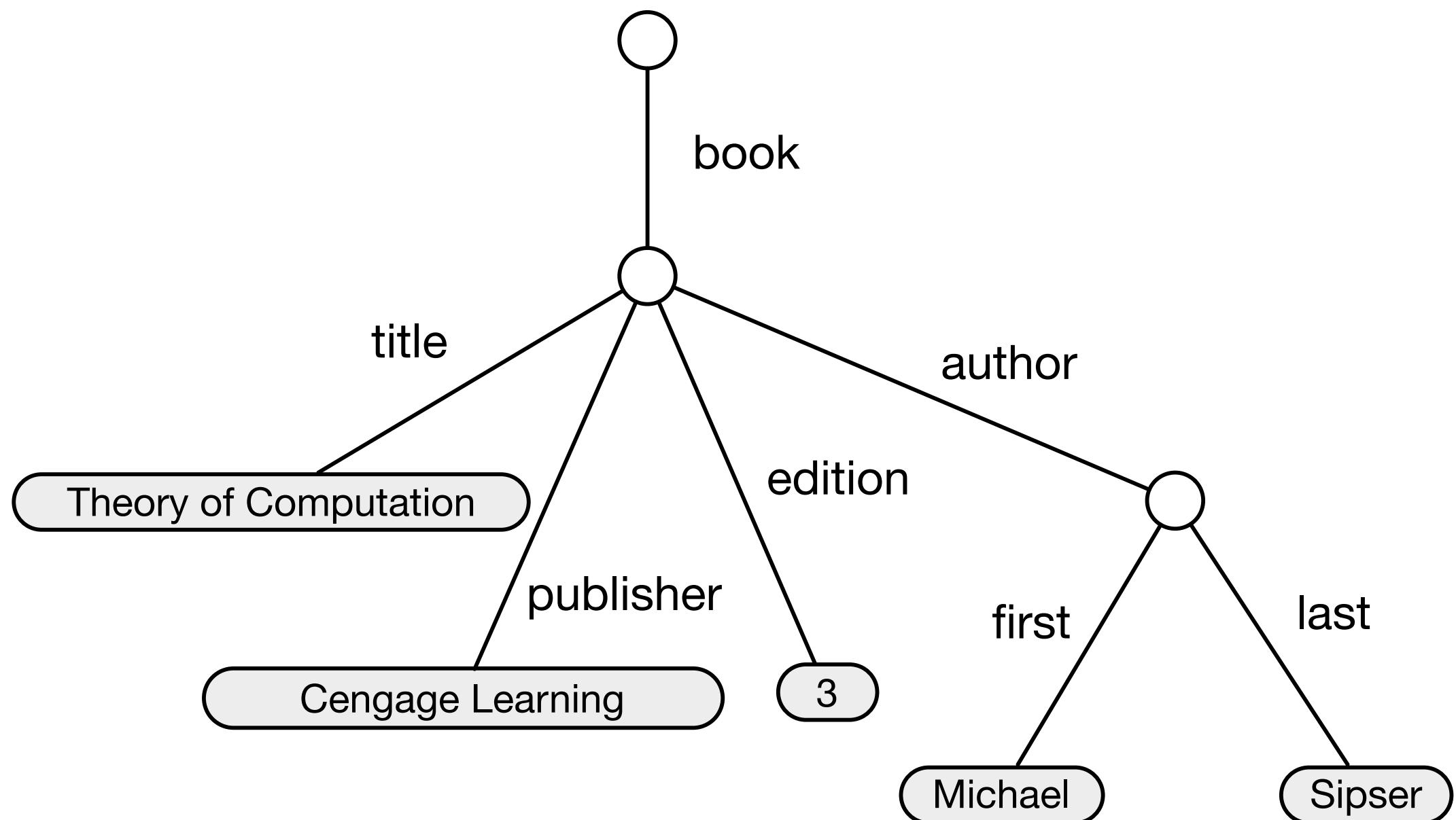
Semi-Structured Data (SSD)

```
{books:{  
  title: "Theory of Computation",  
  edition: 3  
  publisher: "Cengage Learning",  
  author: {first: "Michael", last: "Sipser"}  
},  
...}
```

set of field-value pairs

Semi-Structured Data (SSD)

- We need an **internal representation / data model** — **trees**



Semi-Structured Data (SSD)

- How to **store / represent**?
- Different formalisms to represent: Object Exchange Model, Lore, XML, JSON

对象交换模型

知识

XML

XML: e**X**tensible **M**arkup Language

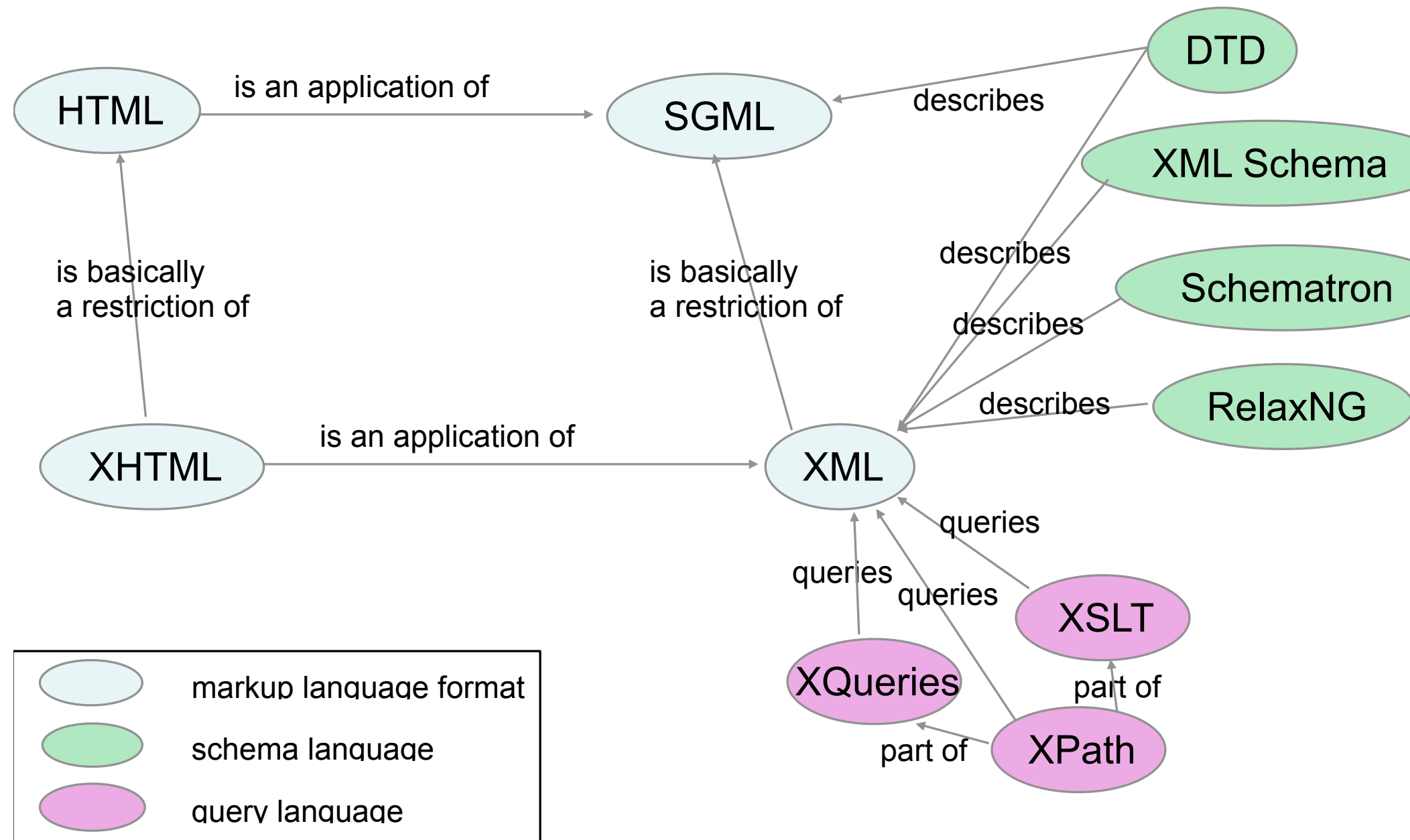
表示半结构化数据, 适合人和电脑使用
不是用于布局文档 (HTML, CSS, ...)
查询还不完善

- representation of semi-structured data, suitable for humans and computers
- is not for the layout of documents (HTML, CSS, ...)
- querying still not perfect

XML: Brief History

- **GML** (Generalized Markup Language), ~1960, IBM
- **SGML** (Standard Generalized Markup Language), 1985 — flexible, expressive
- **HTML** (Hypertext Markup Language), ~1990 — application of SGML for presentation of Web Documents
- **XML**, 1998 (first edition) — a W3C standard, subset of SGML

XML: The Landscape



Credit: B. Parsia, C. Hedeler, U. Sattler; Univ. of Manchester

XML: Example

Theory of Computation
Michael Sipser
Cengage Learning
2012
3

Artificial Intelligence
Stuart Russell
Peter Norvig
Pearson
2013
3

```
<?xml version="1.0" encoding="UTF-8"?>
<books>
  <book>
    <title>Theory of Computation</title>
    <author>Michael Sipser</author>
    <publisher>Cengage Learning</publisher>
    <year>2012</year>
    <edition>3</edition>
  </book>
  <book>
    <title>Artificial Intelligence</title>
    <author>Peter Norvig</author>
    <author>Stuart Russell</author>
    <year>2013</year>
    <edition>3</edition>
    <publisher>Pearson</publisher>
  </book>
</books>
```

XML Example: Wikipedia

University of Paris-Saclay

From Wikipedia, the free encyclopedia
(Redirected from *Université Paris-Saclay*)

Coordinates: 48°71′17.343″N 2°17′12.888″E﻿ / ﻿48.7117343°N 2.1712888°E﻿ / 48.7117343; 2.1712888

The **University of Paris-Saclay** (French: *Université Paris-Saclay*) is a French federal research university which is currently under development. The project is part of the *research-intensive* and *business cluster* Paris-Saclay, located near Paris in the Plateau de Saclay.^{[5][6]} The University of Paris-Saclay is expected to be the training and research center of the Paris-Saclay technology cluster, similar to Stanford University in the Silicon Valley and the Technion – Israel Institute of Technology in Israel.

The University's first academic year started in September 2015.^[7]



Contents [hide]
1 Mission
2 Ranking
3 Setup
4 Academic programmes
5 Research programmes
6 See also
7 References
8 External links

Mission [edit]

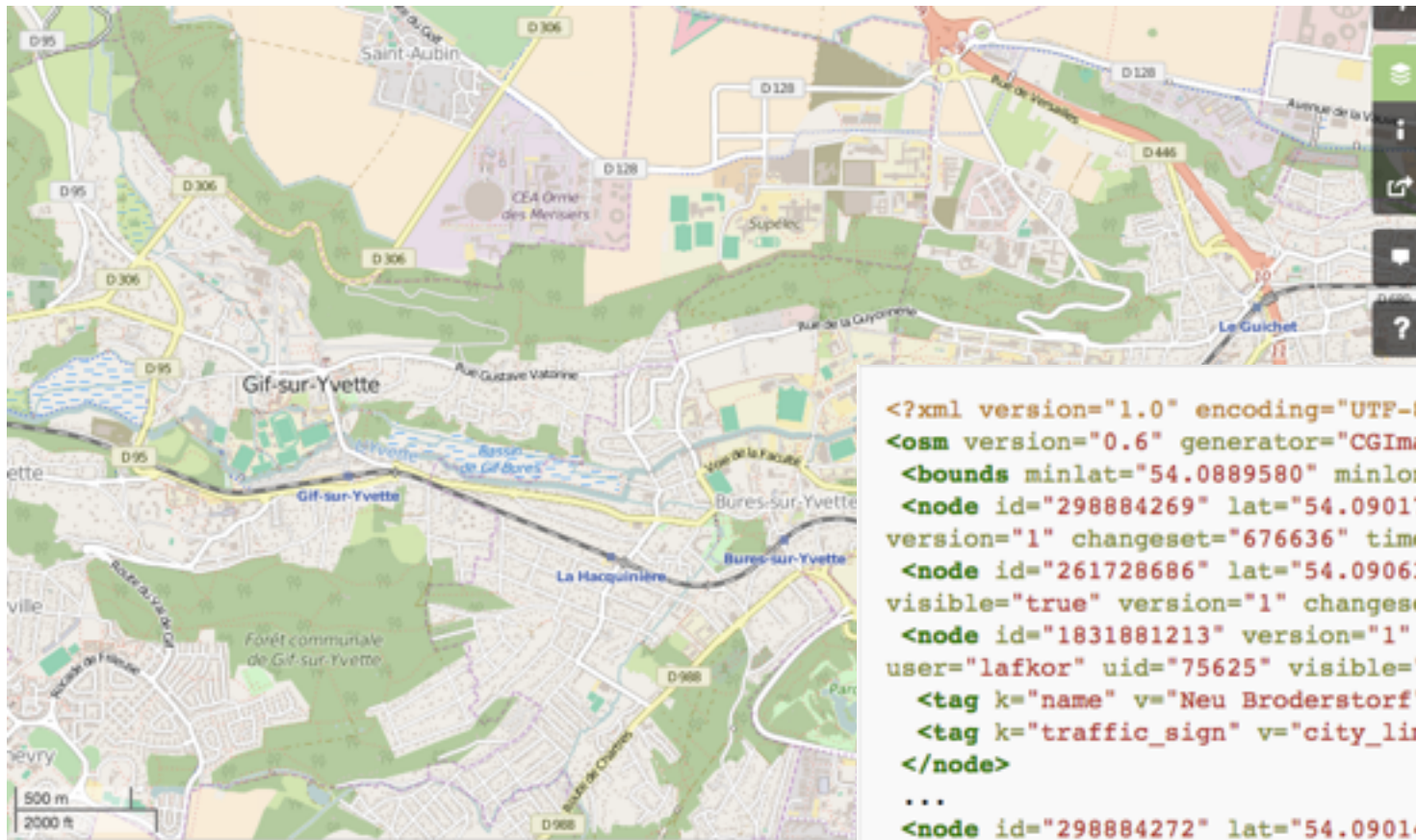
According to Dominique Vernay, chairman of the foundation developing Paris-Saclay in the Academic Ranking of World Universities compiled by Shanghai Jiao Tong University, "top university in continental Europe".^[8] The university wants to maximise the economic research, partly via *university* and *research spin-offs*, but also by working closely with

Ranking [edit]

University Paris-Saclay is still under its integrative process, therefore not yet appearing in rankings. However, an independent simulation ranked the university in the 18th position

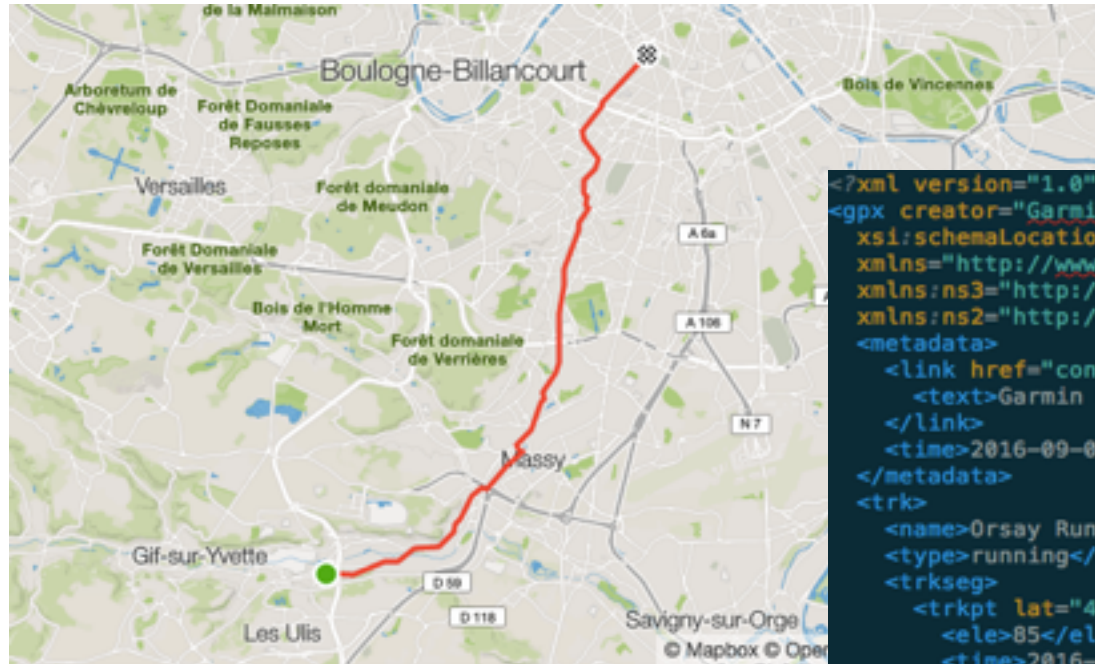
```
<?xml version="1.0"?>
<api batchcomplete="">
  <query>
    <normalized>
      <n from="University_of_Paris-Saclay" to="University of Paris-Saclay" />
    </normalized>
    <pages>
      <page _idx="42196987" pageid="42196987" ns="0" title="University of Paris-Saclay">
        <revisions>
          <rev contentformat="text/x-wiki" contentmodel="wikitext" xml:space="preserve">{{Infobox University
|name          = University of Paris-Saclay
|native_name   = Université Paris-Saclay
|image         = [[File:University of Paris-Saclay.png|250px]]
|Photomontage
|photola       = Polytech ParisSud.JPG
|photolb       = Danone Saclay.JPG
|photo2a       = CEA Saclay.JPG
|photo2b       = Institut d'optique Graduate School.JPG
|photo4a       = LAL Paris-Saclay.JPG
|photo4b       = Siege GSK Paris-Saclay.JPG
|size          = 250
|space         = 2
|}}
|city          = [[Paris]], [[Versailles]], [[Saint-Quentin-en-Yvelines]], [[Saint-Aubin, Essonne|Saint-Aubin]]
|country       = [[France]]
|coord        = {{coord|48.7117343|2.1712888|type:edu|display=inline,title}}
|campus        = [[Paris-Saclay]], [[Université de Versailles-Saint-Quentin-en-Yvelines|Versailles Saint-Quentin-en-Yvelines]],
[[Université Paris-Sud]]
|president     = Gilles Bloch<ref name = "UpsPresident">&gt;{{Cite web |title = Gilles Bloch elected President of
Université Paris-Saclay |url = http://www.universite-paris-saclay.fr/en/news/gilles-bloch-has-been-elected-president-of-
universite-paris-saclay |work = universite-paris-saclay.fr |accessdate = 29 May 2016}}&lt;/ref&gt;
|area km2      = 77 &lt;ref name="area">&gt;{{Cite web |title = About Paris-Saclay |url =
http://orientation.blog.lemonde.fr/2013/05/19/paris-saclay-bientot-20-de-la-recherche-francaise-sur-un-immense-campus/ |work
= Lemonde.fr}}&lt;/ref&gt;
|established   = December 29, 2014&lt;ref name = "UpsEstablished">&gt;{{Cite web |title = Establishment of Université
Paris-Saclay «Décret n° 2014-1674 du 29 décembre 2014» |url =
https://www.legifrance.gouv.fr/eli/decret/2014/12/29/MENS1425099D/jo |work = legifrance.gouv.fr |accessdate = 29 May
2016}}&lt;/ref&gt;
|students     = {{formatnum:60000}}&lt;ref name="EPPS">&gt;{{Cite web |title = A World Class University |url =
http://www.epps.fr/en/a-global-cluster/a-world-class-university/ |work = epps.fr}}&lt;/ref&gt;
```


XML Example: Open Maps



```
<?xml version="1.0" encoding="UTF-8"?>
<osm version="0.6" generator="CGImap 0.0.2">
  <bounds minlat="54.0889580" minlon="12.2487570" maxlat="54.0913900" maxlon="12.2524800"/>
  <node id="298884269" lat="54.0901746" lon="12.2482632" user="SvenHRO" uid="46882" visible="true"
version="1" changeset="676636" timestamp="2008-09-21T21:37:45Z"/>
  <node id="261728686" lat="54.0906309" lon="12.2441924" user="PikoWinter" uid="36744"
visible="true" version="1" changeset="323878" timestamp="2008-05-03T13:39:23Z"/>
  <node id="1831881213" version="1" changeset="12370172" lat="54.0900666" lon="12.2539381"
user="lafkor" uid="75625" visible="true" timestamp="2012-07-20T09:43:19Z">
    <tag k="name" v="Neu Broderstorf"/>
    <tag k="traffic_sign" v="city_limit"/>
  </node>
  ...
  <node id="298884272" lat="54.0901447" lon="12.2516513" user="SvenHRO" uid="46882" visible="true"
version="1" changeset="676636" timestamp="2008-09-21T21:37:45Z"/>
  <way id="26659127" user="Masch" uid="55988" visible="true" version="5" changeset="4142606"
timestamp="2010-03-16T11:47:08Z">
    <nd ref="292403538"/>
    <nd ref="298884289"/>
    ...
    <nd ref="261728686"/>
    <tag k="highway" v="unclassified"/>
    <tag k="name" v="Pastower Straße"/>
  </way>
  <relation id="56688" user="kmvar" uid="56190" visible="true" version="28" changeset="6947637"
timestamp="2011-01-12T14:23:49Z">
    <member type="node" ref="294942404" role=""/>
    ...
    <member type="node" ref="364933006" role=""/>
    <member type="way" ref="4579143" role=""/>
    ...
  </relation>
</osm>
```

XML Example: GPS data



```
<?xml version="1.0" encoding="UTF-8"?>
<gpx creator="Garmin Connect" version="1.1"
  xsi:schemaLocation="http://www.topografix.com/GPX/1/1 http://www.topografix.com/GPX/1/1.xsd"
  xmlns="http://www.topografix.com/GPX/1/1"
  xmlns:ns3="http://www.garmin.com/xmlschemas/TrackPointExtension/v1"
  xmlns:ns2="http://www.garmin.com/xmlschemas/GpxExtensions/v3" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <metadata>
    <link href="connect.garmin.com">
      <text>Garmin Connect</text>
    </link>
    <time>2016-09-01T09:42:42.000Z</time>
  </metadata>
  <trk>
    <name>Orsay Running</name>
    <type>running</type>
    <trkseg>
      <trkpt lat="48.6968187801539897918701171875" lon="2.18710030429065227508544921875">
        <ele>85</ele>
        <time>2016-09-01T09:42:42.000Z</time>
        <extensions>
          <ns3:TrackPointExtension>
            <ns3:hr>94</ns3:hr>
            <ns3:cad>61</ns3:cad>
          </ns3:TrackPointExtension>
        </extensions>
      </trkpt>
      <trkpt lat="48.6968181096017360687255859375" lon="2.1871216781437397003173828125">
        <ele>85</ele>
        <time>2016-09-01T09:42:43.000Z</time>
        <extensions>
          <ns3:TrackPointExtension>
            <ns3:hr>94</ns3:hr>
            <ns3:cad>61</ns3:cad>
          </ns3:TrackPointExtension>
        </extensions>
      </trkpt>
      <trkpt lat="48.69682037271559238433837890625" lon="2.18720809556543827056884765625">
        <ele>84.8000030517578125</ele>
        <time>2016-09-01T09:42:46.000Z</time>
        <extensions>
          <ns3:TrackPointExtension>
            <ns3:hr>97</ns3:hr>
            <ns3:cad>61</ns3:cad>
          </ns3:TrackPointExtension>
        </extensions>
      </trkpt>
      <trkpt lat="48.6968382261693477630615234375" lon="2.18727774918079376220703125">
        <ele>84.59999847412109375</ele>
        <time>2016-09-01T09:42:48.000Z</time>
      </trkpt>
    </trkseg>
  </trk>
</gpx>
```

XML: Basics

- XML is a specialization of SGML
- W3C standard since 1998 (<http://www.w3.org/XML>)
- designed to be simple, generic and extensible
- piece of text that describes structure and data

XML: Basics

- associated with a tree
- divided into smaller pieces of data called elements (associated with nodes in tree):
 - XML documents contains elements
 - elements can contain elements
 - non-ambiguous hierarchical structure
- XML document: a root element containing all other elements

XML: Basics

- XML is a **meta-language**: allows to **create markup-languages**
- **no need for a parser**, one can fix one's own “language” — describe all **admissible structures** (allowed element names, how they can be put together, data types, ...)
- done using **XML type definition languages** (DTD, Relax NG, XML Schema)

XML: Components

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE book SYSTEM "book.dtd">
<books>
  <book>
    <title>Theory of Computation</title>
    <author>Michael Sipser</author>
    <publisher>Cengage Learning</publisher>
    <year>2012</year>
    <edition>3</edition>
  </book>
  <book>
    <title>Artificial Intelligence</title>
    <author>Peter Norvig</author>
    <author>Stuart Russell</author>
    <year>2013</year>
    <edition>3</edition>
    <publisher>Pearson</publisher>
  </book>
</books>
```

Declarations

Root

XML: Declarations

- XML Declaration (optional)

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

- Document type declaration (optional) — contains the structure, content and tags of a document

```
<!DOCTYPE book SYSTEM "book.dtd">
```

XML: Elements

- elements are delimited by tags
- tags are enclosed in angle brackets, e.g., <author>, </book>
- tags are case-sensitive: <BOOK> is not the same as <book>
- we can have: start tags: <...>, e.g., <book>, end tags: </...>, e.g., </book>
- a pair of matching start- and end tags defines an element
- attributes specify properties of an element, <book title="Theory of Computation">

XML: Elements

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

```
<!DOCTYPE book SYSTEM "book.dtd">
```

```
<books>
```

```
  <book>
```

```
    <title>Theory of Computation</title>
```

```
    <author>Michael Sipser</author>
```

```
    <publisher>Cengage Learning</publisher>
```

```
    <year>2012</year>
```

```
    <edition>3</edition>
```

```
  </book>
```

```
  <book>
```

```
    <title>Artificial Intelligence</title>
```

```
    <author>Peter Norvig</author>
```

```
    <author>Stuart Russell</author>
```

```
    <year>2013</year>
```

```
    <edition>3</edition>
```

```
    <publisher>Pearson</publisher>
```

```
  </book>
```

```
</books>
```

Start tag

End tag

Element

XML: Elements

`<element-name attr-dec1 ... attr-decn>`

`content`

`</element-name>`

任意的

- arbitrary number of attributes allowed
- each attribute name occurs only once in the same element
- the element `content` can be: empty, text and/or one or more elements
- a pair of matching start- and end tags defines an element
- an empty element can be abbreviated as, `<book ... />`

XML: Prologue

`<?xml param1 param2 ... ?>`

- each param is of the form `parameter-name="parameter-value"`

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

- XML version
- character encoding
- an XML document should have an XML declaration, but does not need to

XML: Prologue

<!DOCTYPE book SYSTEM "book.dtd">

XML类型定义语言

- XML type definition language; here, **DTD** (**XML Document Type Definition**)
- each element is associated with its **content model** (regular expression)

books \rightarrow (book)* *: 大于等于0个书本

book \rightarrow title, author, publisher, edition?
? 匹配前面的子表达式零次或一次

XML: Others

- **comments:** `<!-- comment here -->`
- **processing instructions:** `<?php sql("SELECT * FROM books") ... ?>`
- **entity references:** `<p> Text: &text </p>`

XML: Full Grammar

[1]	document	::=	<u>prolog</u> <u>element</u> <u>Misc</u> *
[2]	Char	::=	<i>a Unicode character</i>
[3]	S	::=	(<u>' '</u> <u>'\t'</u> <u>'\n'</u> <u>'\r'</u>)+
[4]	NameChar	::=	(Letter Digit <u>'.'</u> <u>'-'</u> <u>':'</u>
[5]	Name	::=	(<u>Letter</u> <u>'_'</u> <u>':'</u>) (<u>NameChar</u>)*
[22]	prolog	::=	<u>XMLDecl</u> ? <u>Misc</u> * (<u>doctypeDecl</u> <u>Misc</u> *)?
[23]	XMLDecl	::=	<u>'<?xml'</u> <u>VersionInfo</u> <u>EncodingDecl</u> ? <u>SDDecl</u> ? <u>s</u> ? <u>'?></u>
[24]	VersionInfo	::=	<u>s</u> 'version' <u>Eq</u> ("" <u>VersionNum</u> "" "" <u>VersionNum</u> ""')
[25]	Eq	::=	<u>s</u> ? <u>'='</u> <u>s</u> ?
[26]	VersionNum	::=	<u>'1.0'</u>
[39]	element	::=	<u>EmptyElemTag</u> <u>STag</u> <u>content</u> <u>Etag</u>
[40]	STag	::=	<u>'<'</u> <u>Name</u> (<u>S</u> <u>Attribute</u>)* <u>s</u> ? <u>'>'</u>
[41]	Attribute	::=	<u>Name</u> <u>Eq</u> <u>AttValue</u>
[42]	Etag	::=	<u>'</'</u> <u>Name</u> <u>s</u> ? <u>'>'</u>
[43]	content	::=	(<u>element</u> <u>Reference</u> <u>CharData</u> ?)*
[44]	EmptyElemTag	::=	<u>'<'</u> <u>Name</u> (<u>S</u> <u>Attribute</u>)* <u>s</u> ? <u>'/>'</u>
[67]	Reference	::=	<u>EntityRef</u> <u>CharRef</u>
[68]	EntityRef	::=	<u>'&'</u> <u>Name</u> <u>';'</u>
[84]	Letter	::=	[a-zA-Z]
[88]	Digit	::=	[0-9]

XML: Parsing Algorithm

XML grammar = EBNF, formed of **lhs ::= rhs**

Algorithm

1. 从document开始，根据规则扩张符号，从而建立解析树
2. 树叶是即将要展开的字符
3. xml能正确解析需要做左往右的遍历

1. starting with **document**, **expand symbols using the rules**, thus constructing a **parse tree**
2. **leaves** are the characters which have no further expansion
3. XML parsed if it perfectly **matches the parses tree left-to-right traversal**

XML: Dialects 方言

- **XHTML** (W3C) - XML version of HTML
- **SVG** (W3C) - Vector Graphics
- **MathML** (W3C) - mathematical formulas
- **X3D** (Web3D) - 3D graphics
- **SOAP** (RPC using HTTP), **WSDL** (W3C) - Web services
- **RDF** (W3C), **OWL** (W3C) - metadata/knowledge in Semantic Web
- ...

XML: Correctness

Two levels of correctness:

两级正确性:

1. 格式良好的XML (最小, 弱要求): 数据以树状结构组织
2. 有效的XML (可选, 严格要求): 需要符合特定方言 (DTD, ...)

1. **well-formed XML** (minimal, weak requirement):
data is organized in a tree-like structure

2. **valid XML** (optional, strict requirement): needs
to conform to a specific dialect (DTD, ...)

XML: Well-Formed

- the set of tags/elements is not fixed, one can define anything (unlike HTML: `<h1>`, `<p>`, ...)
- elements can be nested, and of arbitrary depth:
`<book> <book> <book> ... </book> </book>
</book>`
- the same element can occur multiple times in the document

XML: Well-Formed

XML is **well-formed** if:

1. there is **exactly one root element**
2. **tags inside <> are correct** (escape characters)
3. **tags are properly nested**
4. **attributes are unique and are quoted**^{引证}
5. **no comments inside tags**

Very **weak property** = we can parse a document into a tree

XML: Well-Formed

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE book SYSTEM "book.dtd">
<books>
  <book>
    <title>Theory of Computation</title>
    <author>Michael Sipser</author>
    <publisher>Cengage Learning</publisher>
    <year>2012</year>
    <edition>3</edition>
  </book>
  <book>
    <title>Artificial Intelligence</title>
    <author>Peter Norvig</author>
    <author>Stuart Russell</author>
    <year>2013</year>
    <edition>3</edition>
    <publisher>Pearson</publisher>
  </book>
</books>
```

Is this XML **well-formed**?

yes, because we can parse this XML document into a tree

XML: Valid

文件类型：数据生产者和数据消费者之间的合同

验证允许检查合同

错误检测留给解析器

更简单的应用程序，更高速的XML吞吐量（如果输入验证，不需要所有运行时检查）

- document type: contract between data producer and data consumer
- validation allows checking of contract
- error detection is left to the parser
- simpler applications, higher-speed XML throughput (if input validated, no need for all runtime checks)

Parsing and Processing XML

Processing Model

验证比写代码/解析器更好

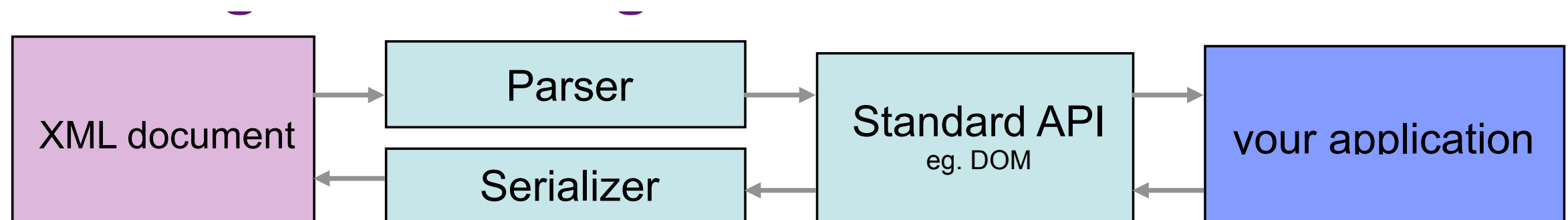
XML应用程序在由XML解析器提供的逻辑树视图上操作

- validation better than writing code/parser
- XML application operate on the logical tree view — provided by an XML parser
- how does an XML parser communicate the tree structure to the application?

Parsing and Serializing XML

解析器读和分析xml文件，可以生成能反映此文件的解析树
寄存器根据数据结构（树，连接的对象）并生成xml文件

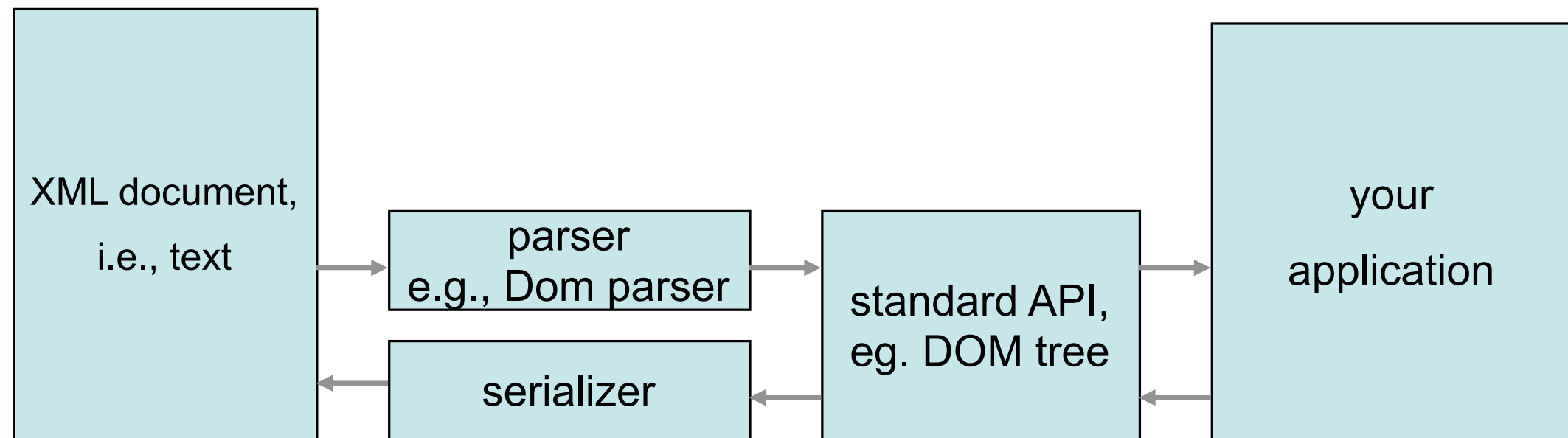
- **parser**: reads and analyzes XML document, may generate a parse tree that reflects document
- **serializer**: takes a data structure (trees, linked objects) and generates an XML document



Credit: B. Parsia, C. Hedeler, U. Sattler; Univ. of Manchester

DOM (Document Object Model)

- DOM tree: internal representation of XML, accessing XML document in the form of a tree

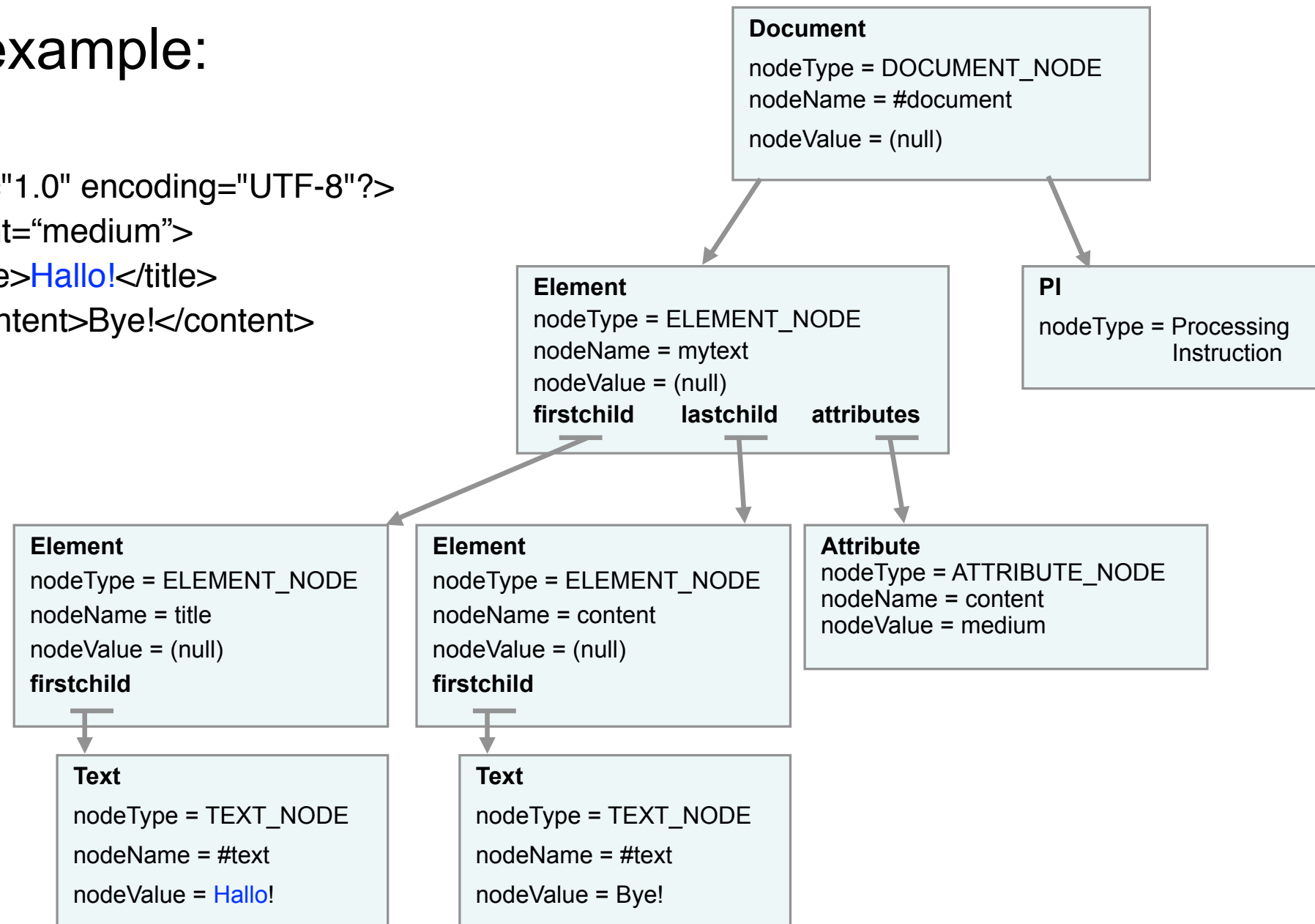


Credit: B. Parsia, C. Hedeler, U. Sattler; Univ. of Manchester

DOM tree

A simple example:

```
<?xml version="1.0" encoding="UTF-8"?>
<mytext content="medium">
  <title>Hallo!</title>
  <content>Bye!</content>
</mytext>
```



Credit: B. Parsia, C. Hedeler, U. Sattler; Univ. of Manchester

DOM tree

Correspondences between DOM and XML document:

- XML document D = tree $t(D)$
- element e in D = node $t(e)$ in $t(D)$
- empty element = leaf node

DOM details

DOM是一种语言和xml文件的中间视图平台
dom的api有很多种编程语言

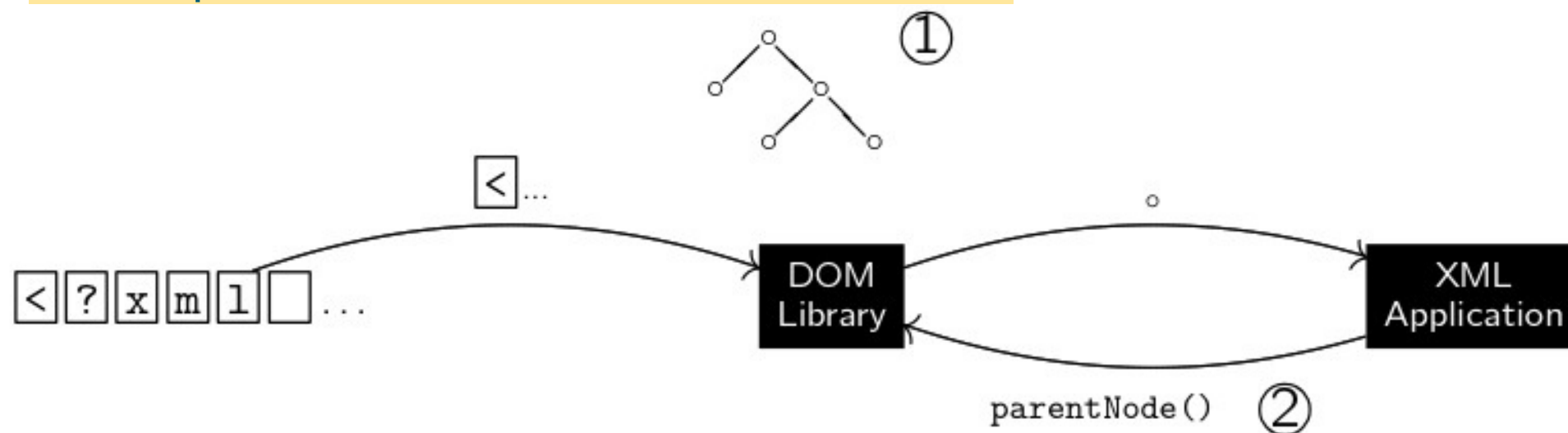
- **DOM (W3C)**: a language- and platform-neutral view of XML documents
- **APIs for DOM** exist for a wide variety of programming languages (Java, C++, Python, ...)

DOM details

DOM design based on two concepts:

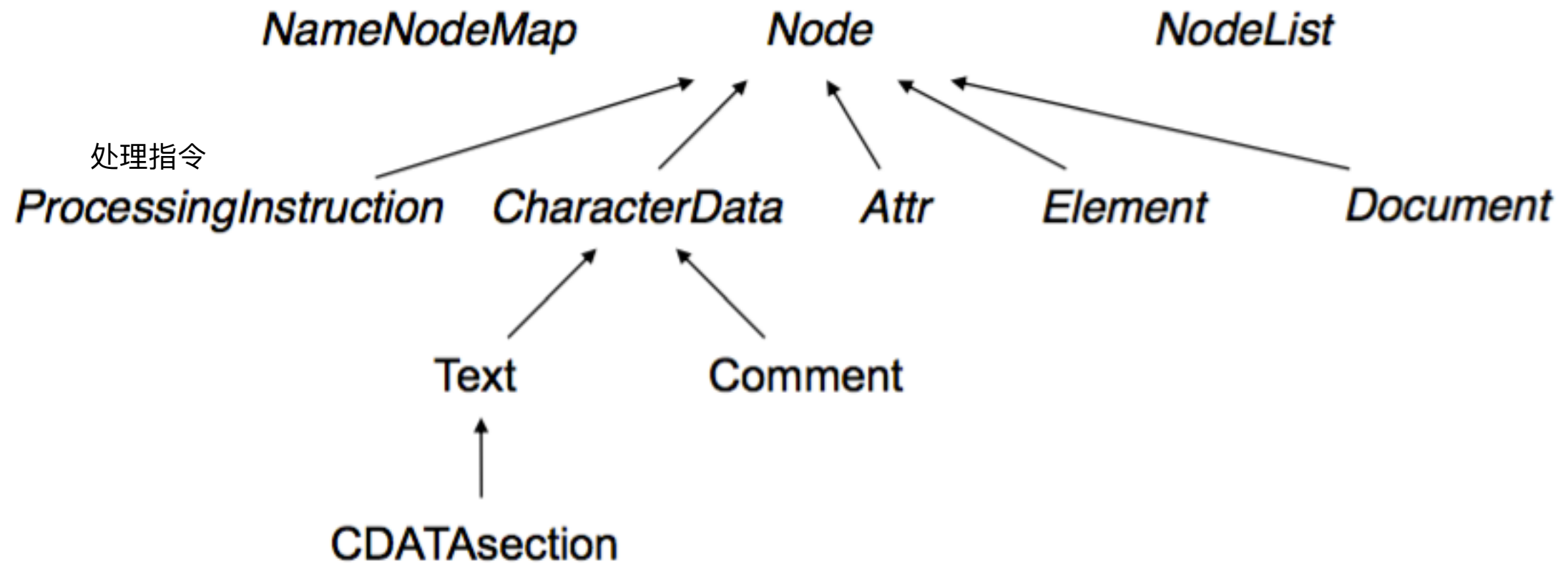
1 xml处理器解xml文件并保持内存中的DOM树
2 xml程序通过api调用来探索和操作xml文件

1. An XML processor parses the XML document, and keeps a DOM tree in-memory
2. XML application then issues API calls to explore and manipulate the XML document



Credit: P. Genevès, CNRS

DOM Core

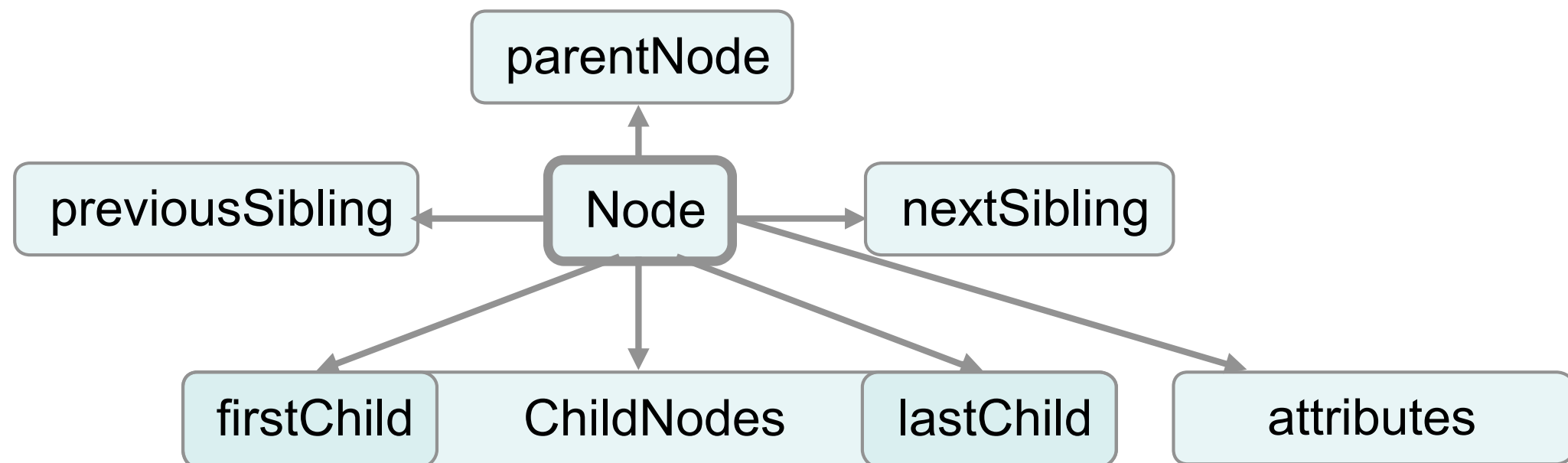


DOM Core

DOM type	Method	Comment
Node	nodeName	redefined in subclasses
	nodeValue	
	parentNode : Node	
	firstChild : Node	leftmost child
	nextSibling : Node	returns NULL for root elem or last child or attributes
	childNodes : NodeList	
	attributes : NamedNodeMap	
	ownerDocument: Document	
	replaceChild : Node	
Document	createElement : Element	creates element with given tag name
	createComment : Comment	
	getElementsByTagName: NodeList	list of all Elem nodes in document order

DOM interface

Can use the **DOM Node interface**, which has the the following attributes:



- other methods: `appendChild`, `hasAttributes`, `insertBefore`, ...

DOM performance

两步方法能够有随机访问对象
随机访问对象是指所有的xml都在主内存中
1 通过大量的交换活动
2 如果应用未正确编程，则内存不足

- The two step approach enables to have an **object that is random-access**
- **Random-access:** all the XML is in main memory:
 1. issues with **heavy swapping activity**
交换
 2. **out-of-memory** if application not properly programmed
内存不足 适当的正确的

DOM performance

我们只需要一小部分xml文件，但所有文件总存在内存中，有时会比原始的xml文件大几倍，从而占内存

- DOM = memory hungry!
- Even if we only need a small part of the XML, everything is kept in memory! — size can be several times that of the original XML
- **Solutions?**

DOM performance

1 预处理xml文件，只将相关部分存在内存里
2 用不同方式处理xml (SAX)

- **Solution 1:** preprocess the XML to only have the relevant parts in memory
- **Solution 2:** use a different way to process XML (SAX)

Simple API for XML (简称SAX) 是个循序存取XML的解析器API。SAX提供一个机制从XML文件读取资料。它是除了文档对象模型 (DOM) 的另外一种流行选择。事件在遇到任一XML特性时触发, 以及遇到他们结尾时再次触发。XML属性也作为传给元素事件资料的一部分。SAX运行时是单向的; 解析过的资料无法在不重新开始的情况下再次读取。

SAX (Simple API for XML)

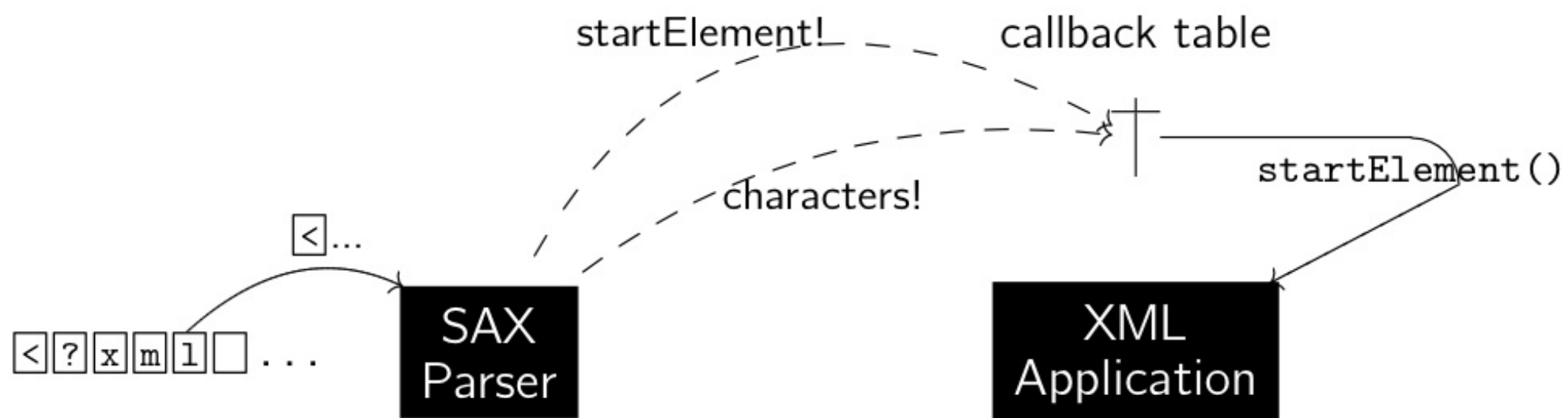
SAX不是W3C标准, 但维护良好
恒定空间中的SAX进程:
没有中间数据结构
SAX向应用程序发送每个XML (流)

- SAX is not an W3C standard, but well maintained
- SAX processes in constant space:
 - ◆ no intermediate data structure
 - ◆ SAX sends events to the application for each piece of XML (streaming)

顺序读取并且只读一次
不记录发送的内容，只发送事件
应用可以和解析程序同时进行

SAX Operation

- reads input sequentially and only once
- no memory of what has been sent is retained — only events are sent
- application can act in parallel with the parsing process



Credit: P. Genevès, CNRS

SAX Operation

SAX reports the following events:

Event	... reported when seen	Parameters sent
<i>startDocument</i>	$\langle ?xml \dots ? \rangle^8$	
<i>endDocument</i>	$\langle EOF \rangle$	
<i>startElement</i>	$\langle t \ a_1=v_1 \dots a_n=v_n \rangle$	$t, (a_1, v_1), \dots, (a_n, v_n)$
<i>endElement</i>	$\langle /t \rangle$	t
<i>characters</i>	<i>text content</i>	Unicode buffer ptr, length
<i>comment</i>	$\langle !--c-- \rangle$	c
<i>processingInstruction</i>	$\langle ?t \ pi? \rangle$	t, pi
	\vdots	

SAX example

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE book SYSTEM "book.dtd">
<books>
  <book>
    <title>Theory of Computation</title>
    <author>Michael Sipser</author>
    <publisher>Cengage Learning</publisher>
    <year>2012</year>
    <edition>3</edition>
  </book>
  <book>
    <title>Artificial Intelligence</title>
    <author>Peter Norvig</author>
    <author>Stuart Russell</author>
    <year>2013</year>
    <edition>3</edition>
    <publisher>Pearson</publisher>
  </book>
</books>
```

What is the order of operations? It corresponds to what?

SAX vs DOM

优点：SAX解析器的内存使用量一般远低于DOM解析器使用量，DOM解析器在任何处理开始之前，必须把整棵树放在内存，所以DOM解析器的内存使用量完全根据输入资料的大小。相对来说，SAX解析器的内存内容，是只基于XML档案的最大深度（XML树的最大深度）和单一XML项目上XML属性储存的最大资料。这两个总是比整颗解析树本身还小。

缺点：某些种类的XML验证需要存取整份文件。例如，一个DTD IDREF属性需要文件内有项目使用指定字串当成DTD ID属性。要在SAX解析器内验证，必须追踪每个之前遇过的ID和IDREF属性，检查是否有任何相符。更甚者，一个IDREF找不到对应的ID，使用者只会在整份文件都解析完后才发现，若这种连结对于建立有效输出是重要的，那用在处理整份文件的时间只是浪费。

另外，某些XML处理仅要求存取整份文件。举例来说，XSLT及XPath需要能够任何时间存取任何解析过的XML树。当SAX以用来建构此树时，DOM解析器在设计上已经是如此了。

sax事件的顺序是由文件树的前序遍历定义的

定义的很好，但是不一定允许在xml的所有查询，如果需要，可以重建xml树

- Order of SAX events is determined by a **preorder traversal** of the document tree
- Well-defined, but **may not allow all possible queries on the XML**
- If really needed, enough to **rebuild the whole XML tree. How?**

JSON

JSON

(JavaScript Object Notation)

- another tree data structure formalism — simpler than XML
- coming from JavaScript
- details at <http://www.json.org/xml.html>

Twitter API (JSON)

Example Request

GET

`https://api.twitter.com/1.1/statuses/show.json?
id=210462857140252672`

Example Result

```
{  
  "coordinates": null,  
  "favorited": false,  
  "truncated": false,  
  "created_at": "Wed Jun 06 20:07:10 +0000 2012",  
  "id_str": "210462857140252672",  
  "entities": {  
    "urls": [  
      {  
        "expanded_url": "https://dev.twitter.com/terms/display-guidelines",  
        "url": "https://t.co/Ed4omjYs",  
        "indices": [  
          76,  
          97  
        ],  
        "display_url": "dev.twitter.com/terms/display-\u2026"  
      }  
    ],  
    "hashtags": [  
      {  
        "text": "Twitterbird",  
        "indices": [  
          19,  
          31  
        ]  
      }  
    ]  
  }  
}
```

<https://dev.twitter.com/rest/tools/console>

JSON Principles

JSON - [fragment of JavaScript](#) from set of literals called items:

- **atomic**: numbers, booleans, strings
- **arrays** (composite): `[1, 2, "three", "four"]`
- **objects** (composite) — sets, unordered lists, associative arrays: `{"first": "Michael", "second": "Sipser"}`
- **can be nested**: `[{"one": 1, "two": 2}, {"name": {"first": "Michael", "second": "Sipser"}}]`

JSON - XML

```
{"book": {  
  
  "title": "Artificial Intelligence",  
  
  "publisher": "Pearson",  
  
  "authors": {  
  
    "author": [  
  
      {"first": "Peter", "last":  
        "Norvig"},  
  
      {"first": "Stuart", "last":  
        "Russell"}  
  
    ]  
  
  }  
  
}}
```

```
<book title="Artificial  
Intelligence" publisher="Pearson">  
  
  <authors>  
  
    <author first="Peter"  
      last="Norvig" />  
  
    <author first="Stuart"  
      last="Russell" />  
  
  </authors>  
  
</book>
```

JSON - XML

<pre>{"book": { "title": "Artificial Intelligence", "publisher": "Pearson", "authors": { "author": [{"first": "Peter", "last": "Norvig"}, {"first": "Stuart", "last": "Russell"}] } }}</pre>	<pre><book title="Artificial Intelligence" publisher="Pearson"> <authors> <author first="Peter" last="Norvig" /> <author first="Stuart" last="Russell" /> </authors> </book></pre>
	因为child element在JSON里是存在一个表格里，所以在表格里顺序就是matter的

Order of children matters!

XML to JSON

- Elements are mapped to objects —
ElementName : contents
 - 元素映射到对象 - ElementName: 内容
 - 内容是一个列表
 - 属性由对象{...}分隔
 - 子元素在一个数组[...]
 - 空元素需要一个明确的空列表
 - 没有属性需要明确的空对象
- **contents** is a list
- **attributes** are delimited by an object {...}
- **child elements** are in an array [...]
- **empty elements** require an explicitly empty list
- **no attributes** requires an explicitly empty object

JSON Advantages

更简单的理解：较少冗长，更人性化的阅读

更简单的处理：XML需要DOM / SAX，JSON在字典/列表上运行
对于配置文件（名称 - 值对）很好

冗长

- much simpler to understand: less verbose, more humanly readable
人性化地
- simpler processing: XML needs DOM/SAX, JSON operates on dictionaries / lists
- good for config files (name-value pairs)

Validating XML: DTD

DTD (Document Type Definition)

- XML need a **schema**: a way to tell a parser that the document is valid in regards to what is expected
有关
- different (more strict) concept to well-formedness
- many different schema languages
- we will see first DTD (**Document Type Definition**)

DTD

- defines the allowed tags/elements in a document
- the contents of each elements by way of regular expressions
- the attributes of an element, and their respective types

DTD: Syntax

- **<!ELEMENT elem_name elem_regexp>** — an element named **elem_name** contains elements described by the regular expression **elem_regexp**
- **<!ATTLIST elem_name att_name att_type att_values>** — the element **elem_name** has an attribute named **att_name** of type **att_type** and having possible values described by **att_values**

DTD: Syntax

- **regular expressions** are formed of `*`, `+`, `?`, sequence `[,]`, `EMPTY`, `ANY`, `#PCDATA` (text)
- **attribute types** are `ID` (primary key), `IDREF` (foreign key), `CDATA` (text), `v1 | v2 | , ..., vn` (fixed value list)
- **attribute values** are `v` (default value), `#REQUIRED` (mandatory attribute), `#IMPLIED` (optional attribute), `#FIXED v` (constant value `v`)

DTD Example

Books.xml

book.dtd

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE book SYSTEM "book.dtd">
<books>
  <book>
    <title>Theory of Computation</title>
    <author>Michael Sipser</author>
    <publisher>Cengage Learning</publisher>
    <year>2012</year>
    <edition>3</edition>
  </book>
  <book>
    <title>Artificial Intelligence</title>
    <author>Peter Norvig</author>
    <author>Stuart Russell</author>
    <year>2013</year>
    <edition>3</edition>
    <publisher>Pearson</publisher>
  </book>
</books>
```

```
<!ELEMENT books (book*)>
<!ELEMENT book (publisher,edition, authors)>
<!ATTLIST book title CDATA #REQUIRED>
<!ELEMENT publisher #PCDATA>
<!ELEMENT edition #PCDATA>
<!ELEMENT authors (author+)>
<!ELEMENT author (first,last)>
<!ELEMENT first #PCDATA>
<!ELEMENT last #PCDATA>
```

Is this XML **valid**? What is the DTD for validation?

DTD Example

Books.xml

book.dtd

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE book SYSTEM "book.dtd">
<books>
  <book>
    <title>Theory of Computation</title>
    <author>Michael Sipser</author>
    <publisher>Cengage Learning</publisher>
    <year>2012</year>
    <edition>3</edition>
  </book>
  <book>
    <title>Artificial Intelligence</title>
    <author>Peter Norvig</author>
    <author>Stuart Russell</author>
    <year>2013</year>
    <edition>3</edition>
    <publisher>Pearson</publisher>
  </book>
</books>
```

```
<!ELEMENT books (book*)>
<!ELEMENT book (publisher,edition, authors)>
<!ATTLIST book title CDATA #REQUIRED>
<!ELEMENT publisher #PCDATA>
<!ELEMENT edition #PCDATA>
<!ELEMENT authors (author+)>
<!ELEMENT author (first,last)>
<!ELEMENT first #PCDATA>
<!ELEMENT last #PCDATA>
```

How can we [modify the DTD](#) to validate?

Further Reading

- Marc H. Scholl, XML and Databases <http://www.cse.unsw.edu.au/~cs4317/09s1/XMLDB.pdf>
- W3C, XML Tutorial <http://www.w3schools.com/xml/>
- W3C, XML Standard <https://www.w3.org/XML/>
- W3C, DOM Standard <https://www.w3.org/DOM/>
- JSON Standard <http://www.json.org/>