EXTENSIBLE MARKUP LANGUAGE

XML

A BIT OF HISTORY

- W3C recommendation since 1998
  - Evolution of SGML (ancestor of HTML)
  - Language for the description of structured information (meta-language)
    - machine-readable
    - human-readable

XML

SYNTAX AND VOCABULARY

Vocabulary	Syntax
Start tag	`<myElement>`
End tag	`</myElement>`
Empty-element tag	`<myElement/>`
Element	`<myElement/>` `<myElement>…</myElement>`
Text content	`<monElement>voici du texte</monEleme`
Comments	`<!--Somecomments -->`
Processing Instructions	`<?SomeInstruction?>`
Attribute	`<myElement myAttribute="data" />`

XML

THE NOTION OF DOCUMENT

- Specific syntax
  - Declaration of character set and encoding
    - mostly Unicode and UTF-8
  - Use of tags, attributes and text content with constraints on:
    - Well-formedness
      - Nested tags with specific constraints
      - <a> is forbidden!
    - Validity (in some cases)
- Examples of XML languages
  - XHTML, SVG, RSS, ...

XML

BENEFITS

- Simple Syntax
  - Easy to learn, to generate, to read
- Generic syntax
  - Learn once, apply many times for different vocabulary
  - No semantic associated
- Extensible syntax
  - Easy to add new elements, new attributes, new values
  - XML language evolve in a backward/forward compatible way
- Wide-spread adoption
  - Lots of existing tools to create, parse, analyse, validate ...

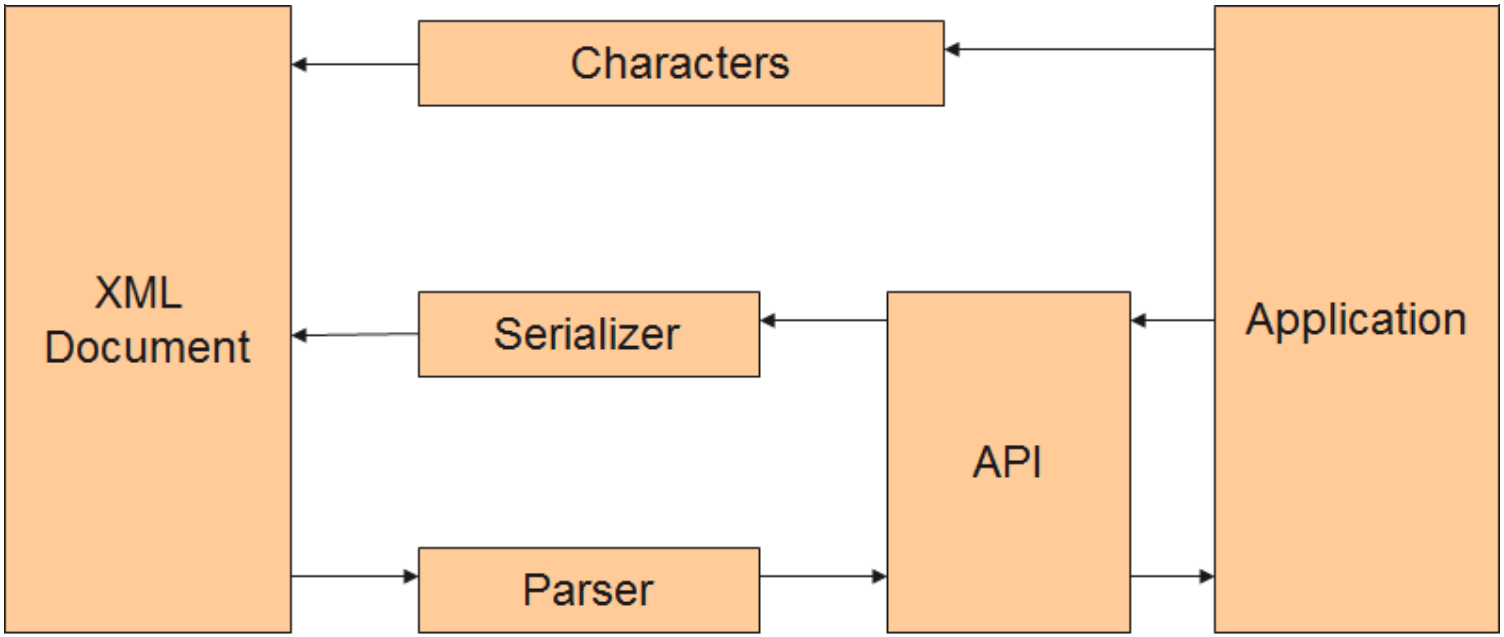
XML

DRAWBACKS

- Verbosity
  - Length of tags/attribute names
  - Redundancy of start/end tags
  - XML files become quickly big
    - Need for compression (zip, gzip, EXI)
- Performance
  - XML is character/string-based
  - String processing is slow
    - Need for more compact binary representation
- Validation
  - Need to have the full document

XML

PROCESSING OF DOCUMENTS



XML

NAMESPACES

- Goals
  - Clearly associate elements and attributes with a namespace for:
    - identification
      - Ability to mix elements and attributes from different XML vocabulary
    - validation
  - Principles
    - Definition of a prefix string and association with a unique identifier (URN, URL, ...à

```
xmlns:prefix="MyUniqueIdentifier"
```

    - Use of a prefix in front of elements and attributes
      - Qualified name=prefix + ':' + local name (ex: xlink:href)

```
prefix:attribute="value"
```

XML

NAMESPACES EXAMPLES

```
<bk:book xmlns:bk='urn:loc.gov:books' xmlns:isbn='urn:ISBN:0-395-36341-6'>
  <bk:title>Cheaper by the Dozen</bk:title>
  <isbn:number>1568491379</isbn:number>
</bk:book>
```

```
<html:html xmlns:html='http://www.w3.org/1999/xhtml'>
 <html:head>
  <html:title>Hello</html:title>
 </html:head>
 <html:body><html:p>Hello World!</html:p></html:body>
</html:html>
```

XML

VALIDATION OF DOCUMENTS

- Purpose
  - Checking constraints in the vocabulary according to a grammar or schema
    - Additional step after well-formedness
- Methods
  - DTD: Document Type Definition (old, limited, not XML)
  - XML Schema: XML grammar (some limitations)
  - RelaxNG: alternative XML grammar (trendy)
  - Schematron: assertions to be tested on an XML doc

XML SCHEMA

EXAMPLE

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="personne">
<xs:complexType>
<xs:sequence>
<xs:element name="nom" type="xs:string" />
<xs:element name="prenom" type="xs:string" />
<xs:element name="date_naissance" type="xs:date" />
<xs:element name="etablissement" type="xs:string" />
<xs:element name="num_tel" type="xs:string" />
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<personne>
<nom>MBODJ</nom>
<prenom>Babacar</prenom>
<date_naissance>1996-10-06</date_naissance>
<etablissement>NIIT</etablissement>
<num_tel>764704140</num_tel>
</personne>
```

RELAXNG

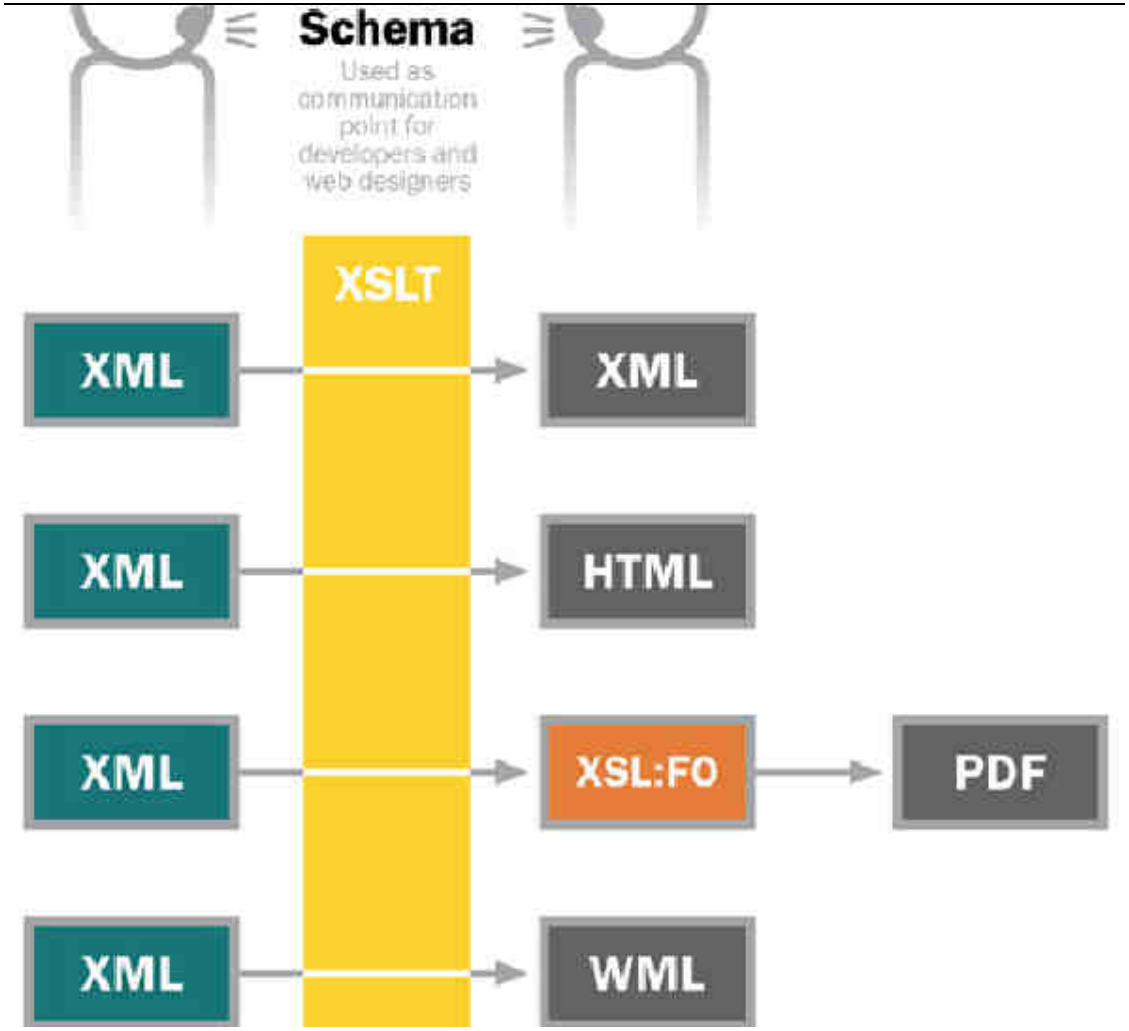
EXAMPLE

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0">
 <start>
   <ref name="personne"/>
 </start>
 <define name="personne">
   <element name="personne">
     <interleave>
       <ref name="nom"/>
       <ref name="prenom"/>
       <optional>
         <ref name="date_naissance"/>
       </optional>
     </interleave>
   </element>
 </define>
 <define name="nom">
   <element name="nom">
     <text/>
   </element>
 </define>
 <define name="prenom">
   <element name="prenom">
     <text/>
   </element>
 </define>
 <define name="date_naissance">
   <element name="date_naissance">
     <data type="date"/>
   </element>
 </define>
</grammar>
```

EXTENSIBLE STYLESHEET LANGUAGE TRANSFORMATIONS (XSLT)

- Tentative to push CSS further
  - Use XML to describe the rules
    - <template> (vs CSS {})
  - Applicable to some elements
    - Xpath (vs CSS selector)
  - Turing-complete Programming language

XSLT POSSIBILITIES



XSLT EXAMPLE

```
<?xml version="1.0" ?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0">
 <xsl:output method="xml" indent="yes"/>
 <xsl:template match="person">
  <name username="{@username}">
   <xsl:value-of select="name" />
  </name>
 </xsl:template>
</xsl:stylesheet>
```

XPATH

- Specific syntax to address an element in an XML document
- file/folder-like syntax
  - Each step in the par is delimited by '/'
  - Each step uses:
    - An axis
      - direction to follow to the next element (parent, ancestor, descendant, child)
    - Node test
    - Predicates to validate
- XPath paths can be
  - Absolute: from the document root
  - Relative: starting from a given element

XPATH EXAMPLES

- Root of the document

```
/
```

- Current node in the document

```
.
```

- Parent of the document

```
..
```

- The attribute `anAttribute` of the parent node

```
../@anAttribute
```

- The list of all elements called ElementB which are children of an element called ElementA, child of the root

```
/ElementA/ElementB
```