

## Introduction to XML



## Agenda

- Overview of XML
- Building blocks
  - Elements
  - Attributes
  - Processing instructions
  - Comments
  - Character data
- XML namespace
- XML schema



## eXtensible Markup Language (XML)

### Definition

The eXtensible Markup Language (XML) is a W3C recommendation for creating special-purpose markup languages that enable the structuring, description and interchange of data.

- A simplified subset of **SGML** capable of describing many different kinds of data for any imaginable application domain.
- Languages based on XML are themselves described in a formal way, allowing programs to modify and validate documents in these languages **without prior knowledge of their form**.

## Quick exercise

- Which one below is **NOT** the reason of using XML for information exchange?
  - A. XML supports internationalization
  - B. XML is platform independent
  - C. XML supports efficient information processing
  - D. XML presents information in human-readable form.



## Quick exercise

- Is it easier to process **XML** than **XHTML**?
  - A. Yes
  - B. No
  - C. Sometimes
  - D. Cannot say



## XML Building Blocks

- Elements
  - The pairing of a **start tag** and an **end tag**.
- Attributes
  - A **name-value pair** that is part of a start tag of an Element.
- Processing instructions
  - Special **directives** to the application that will process the XML document.
- Comments
  - Messages helping a human reader understand the source code.
- Character Data
  - Characters (in a specific encoding)
  - Entities
  - Escapes



## XML elements

### Definition

The term **element** is a technical name for the pairing of a start tag and an end tag in an XML Document.

### Production Rule

$$\begin{aligned}\langle element \rangle &::= \langle EmptyElement \rangle \\ &\quad | \langle STag \rangle \langle content \rangle \langle ETag \rangle \\ \langle STag \rangle &::= '<' \langle Name \rangle \langle Attribute \rangle^* '>' \\ \langle ETag \rangle &::= '</' \text{ Name } '>' \\ \langle EmptyElement \rangle &::= '<' \text{ Name } \langle Attribute \rangle^* '/>'\end{aligned}$$

- XML elements must be **strictly nested**.
- Single root element
- Element names are **case sensitive**.
- Element names can include letters, digits, underscore, hyphen, period and colon; they **must** begin with a letter (or underscore).

## XML elements example

### Example

```
<!-- Example 1: Element with two tags -->  
<message> Welcome! </message>
```

```
<!-- Example 2: Empty Element (Single tag) -->  
<message/>
```

### Wrong Examples

```
<ATag><BTag> Nesting Problem </ATag></BTag>
```

```
<.wrong.element> some text </.wrong.element>
```



## Quick exercise

- Which of the following strings are correct XML element names?
  - A. `_myElement`
  - B. `xmlExtension`
  - C. `#myElement`
  - D. `1Element`
  - E. None of the above



## XML attributes

### Definition

The term **attribute**(s) refers to a theoretically arbitrary number of name-value pairs that can be included in the starting tag of an XML element.

### Production Rule

$\langle STag \rangle ::= \text{'<'} \langle TagName \rangle \langle Attribute \rangle^* \text{'>'}$   
 $\langle Attribute \rangle ::= \text{AttrName} \text{'=' Value}$

- The value part of the attribute has to be **quoted**.
- Attribute names starting with **xml:** are **reserved** by the XML specification.

### Example

```
<!-- Single attribute -->  
<yacht length="60f"/>
```

## Quick exercise

- Which of the following XML fragments are **well-formed**?
  - A. `<myElement myAttribute=someValue/>`
  - B. `<myElement myAttribute="someValue"/>`
  - C. `<myElement myAttribute='someValue'>`
  - D. `<myElement myAttribute="someValue'>`



## Quick exercise

- How can we make attributes having multiple values?
  - A. `<myElement myAttribute="value1 value2"/>`
  - B. `<myElement myAttribute="value1" myAttribute="value2"/>`
  - C. `<myElement myAttribute="value1, value2"/>`
  - D. attributes cannot have multiple values





## Processing instructions

### Definition

A special directive to the applications processing the XML documents.

### Production Rule

$\langle PI \rangle ::= \text{'<?' PITarget ... '?>'}$

```
<?php echo $a; ?>
```

### Example

```
<!-- Example: A popular one! -->  
<?xml version="1.0" encoding="UTF-8"?>
```

## Comments & character data

- **Comment** A **message** that helps the human reader understand the information contained in an XML document.

### Production Rule

$\langle \textit{Comment} \rangle ::= \text{'<!--' Char}^* \text{'-->'}$

- Character Data
  - **Encoding:** All characters in an XML document must comply with the document's encoding
    - those outside the encoding must be **escaped** and are called **character references**.
    - Alternatively **entities**
  - How to treat **whitespace**?

## Escape

- XML provides escape facilities for including characters which are problematic to include directly.
  - The characters "<" and "&" are key syntax markers and may never appear in content
  - Some character encodings support only a subset of Unicode.
  - It might not be possible to type the character on the author's machine.
  - Some characters have glyphs that cannot be visually distinguished from other characters
- All permitted Unicode characters may be represented with a numeric character reference.
  - 中 : can be represented as  
    &#20013; or &#x4e2d;



## Entities

- Some special characters are frequently referenced in an XML document.
- The XML specification defines five "predefined entities" representing special characters.



| Name | Character | Unicode code point (decimal) | Standard |
|------|-----------|------------------------------|----------|
| quot | "         | U+0022 (34)                  | XML 1.0  |
| amp  | &         | U+0026 (38)                  | XML 1.0  |
| apos | '         | U+0027 (39)                  | XML 1.0  |
| lt   | <         | U+003C (60)                  | XML 1.0  |
| gt   | >         | U+003E (62)                  | XML 1.0  |

- e.g. **&amp;** and **&lt;**;



## An XML document - putting it all together!

```
<?xml version="1.0" encoding="UTF-8"?>
<message from="yiannis" to="family">
  <text> Hey, I'm in Wellington! </text>
  <!-- Attachment is optional -->
  <attachment>
    <desc> Photo from Wellington </desc>
    <item>
      <binaryData>
        111000101010111...
      </binaryData>
    </item>
  </attachment>
</message>
```

- An XML Document consists of:
  - A root element
  - Child elements
  - Comments
  - Processing Instructions

## Some problems

- How to determine the **correctness** of an XML document:
  - **Physical structure** of the document  
**Well-formed** (Parsers)
  - **Logical structure** of the document  
**Validity** (Schemas)
  - **Element name clashes** between Documents  
Namespaces



## XML namespaces

- XML namespaces uses **Uniform Resource Identifiers** (URI) for uniquely qualifying local names.

Qualified Name = Namespace Identifier + Local Name

- A **namespace identifier** is associated with a **prefix**, a name that contains only legal XML element name characters with the exception of the colon (:)
- Qualified names are obtained as a combination of the prefix, the **colon** character, and the local element name



## Namespaces in an XML document

```
<msg:message from="yiannis" to="family"
  xmlns:msg="http://www.w2c.com/ns/email"
  xmlns:po="http://www.w2c.com/ns/purchase">
  <msg:text>
    <msg:desc>A Purchase Order</msg:desc>
    <msg:item>
      <po:order>
        <po:item>
          <po:desc>Laptop Computer</po:desc>
          <po:price>1300 GBP</po:price>
        </po:item>
      </po:order>
    </msg:item>
  </msg:text>
</msg:message>
```



## XML namespaces - A couple more last things

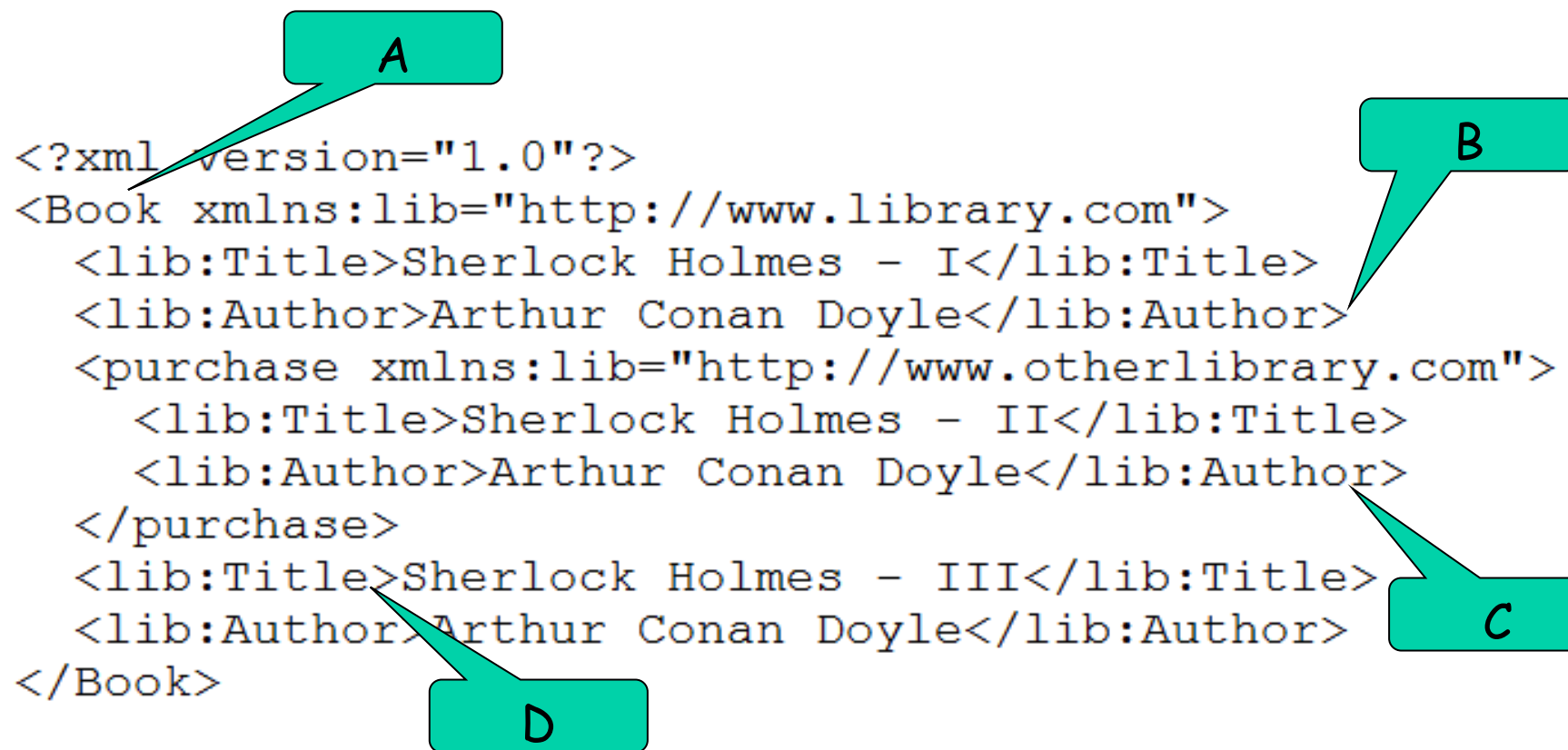
- Default namespaces
  - Elements belonging to the default namespace don't require prefixes.

```
xmlns="http://www.w3.org/1999/xhtmll"  
or  
xmlns=""
```

- Namespace prefixed attributes
  - Attributes can also have namespaces associated with them.
  - Attributes are never subject to the default namespace.
  - An attribute without an explicit namespace prefix is considered not to be in any namespace.



## Quick exercise



The diagram shows an XML snippet with four callouts labeled A, B, C, and D. Callout A points to the XML declaration. Callout B points to the first `<lib:Title>` element. Callout C points to the second `<lib:Title>` element. Callout D points to the `<lib:Author>` element of the third `<lib:Title>` element.

```
<?xml version="1.0"?>
<Book xmlns:lib="http://www.library.com">
  <lib:Title>Sherlock Holmes - I</lib:Title>
  <lib:Author>Arthur Conan Doyle</lib:Author>
  <purchase xmlns:lib="http://www.otherlibrary.com">
    <lib:Title>Sherlock Holmes - II</lib:Title>
    <lib:Author>Arthur Conan Doyle</lib:Author>
  </purchase>
  <lib:Title>Sherlock Holmes - III</lib:Title>
  <lib:Author>Arthur Conan Doyle</lib:Author>
</Book>
```

`http://www.otherlibrary.com`

## XML schema



- An **XML schema** describes the logic structure of an XML document.
- An **XML schema** enables the following:
  - defines elements that can appear in a document
  - defines attributes that can appear in a document
  - defines which elements are child elements
  - defines the order of child elements
  - defines the number of child elements
  - defines whether an element is empty or can include text
  - defines default and fixed values for elements and attributes
- Standard
  - Document Type Definition (DTD)
  - XML Schema Definition (XSD)

## Quick exercise

- What are the differences between XSD and DTD?
  - A. XSD document is an XML document but DTD document is not.
  - B. XSD allows you to create data types. DTD cannot.
  - C. XSD can specify an upper limit for the number of occurrences of an element. DTD cannot.
  - D. None of the above.





# JSON

- JavaScript Object Notation
- Subset of JavaScript
- Similarity with XML
  - Both JSON and XML is **plain text**
  - Both JSON and XML is "**self-describing**" (human readable)
  - Both JSON and XML is **hierarchical** (values within values)
  - Both JSON and XML can be fetched with an HttpRequest



## JSON example

```
{ "employees": [  
  { "firstName": "John", "lastName": "Doe" },  
  { "firstName": "Anna", "lastName": "Smith" },  
  { "firstName": "Peter", "lastName": "Jones" }  
]}
```

## Corresponding XML document

```
<employees>
  <employee>
    <firstName>John</firstName> <lastName>Doe</lastName>
  </employee>
  <employee>
    <firstName>Anna</firstName> <lastName>Smith</lastName>
  </employee>
  <employee>
    <firstName>Peter</firstName> <lastName>Jones</lastName>
  </employee>
</employees>
```

## JSON exercise

- Which of the following data type is NOT supported by JSON?
  - A. Number
  - B. Boolean
  - C. null
  - D. date

## Summary

- What is XML?
- How to build a simple XML document?
- How to avoid ambiguities?
  - XML namespace
- How to determine the correctness of a XML document?
  - XML schema
- What's difference with JSON?

