XML Processing

Tree Processing (DOM)

Lecture "XML in Communication Systems"
Chapter 6

Dr.-Ing. Jesper Zedlitz
Research Group for Communication Systems
Dept. of Computer Science
Christian-Albrechts-University in Kiel

Recommended Reading

- Lauren Wood, Arnaud Le Hors, Vidur Apparao et al.:
  Document Object Model (DOM) Level 1 Specification (Second Edition)
  Version 1.0, W3C Working Draft 29 September, 2000
  http://www.w3.org/TR/2000/WD-DOM-Level-1-20000929/

Overview

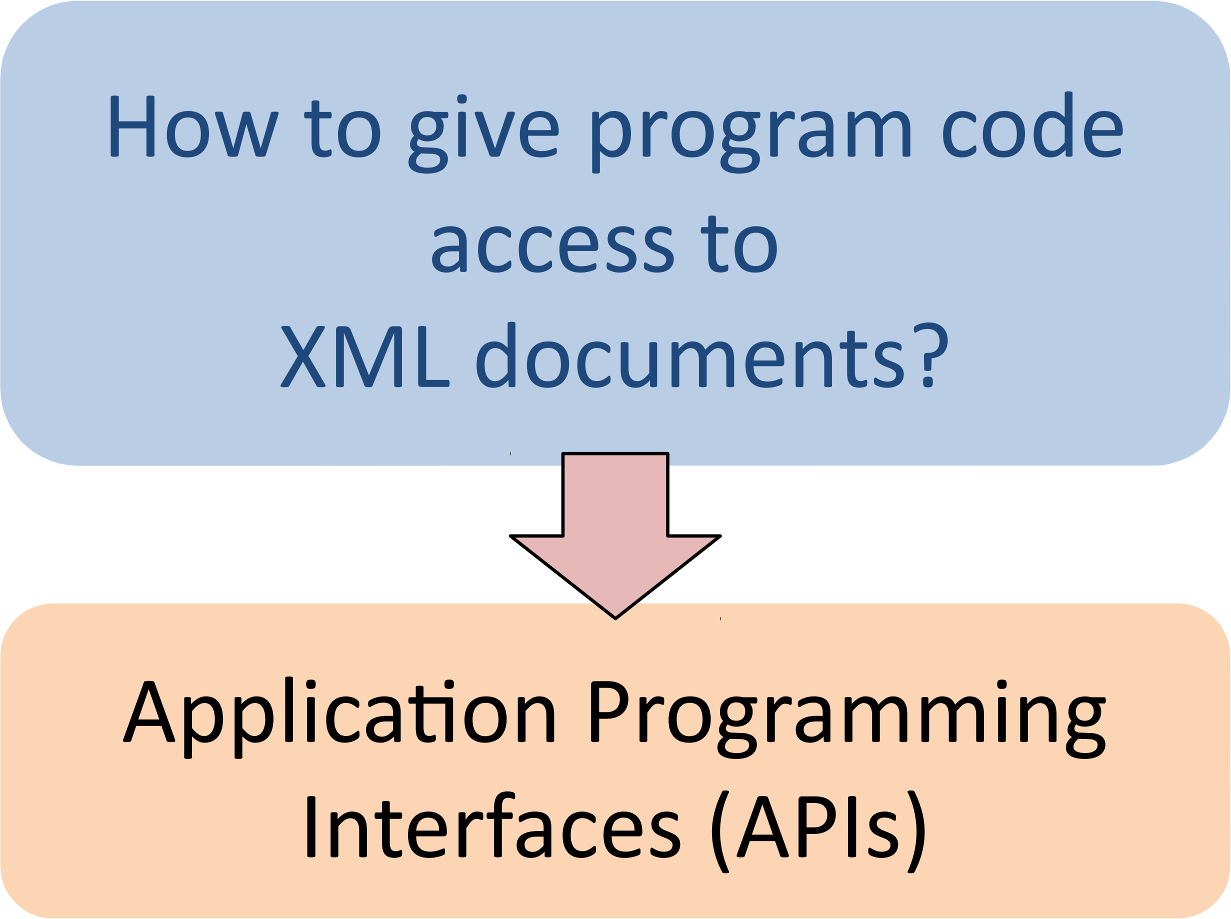
1. Introduction

2. Document Object Model (DOM)

Chapter 6.1

Introduction

Introduction

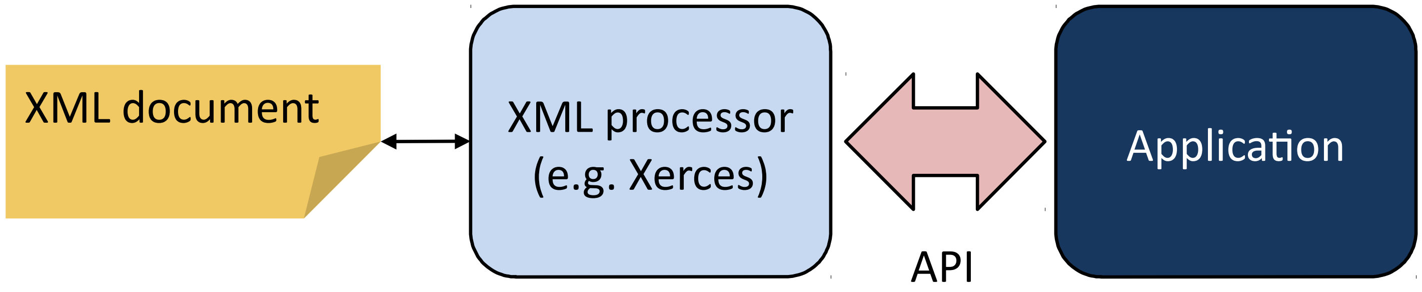


How to give program code access to XML documents?

Application Programming Interfaces (APIs)

Introduction

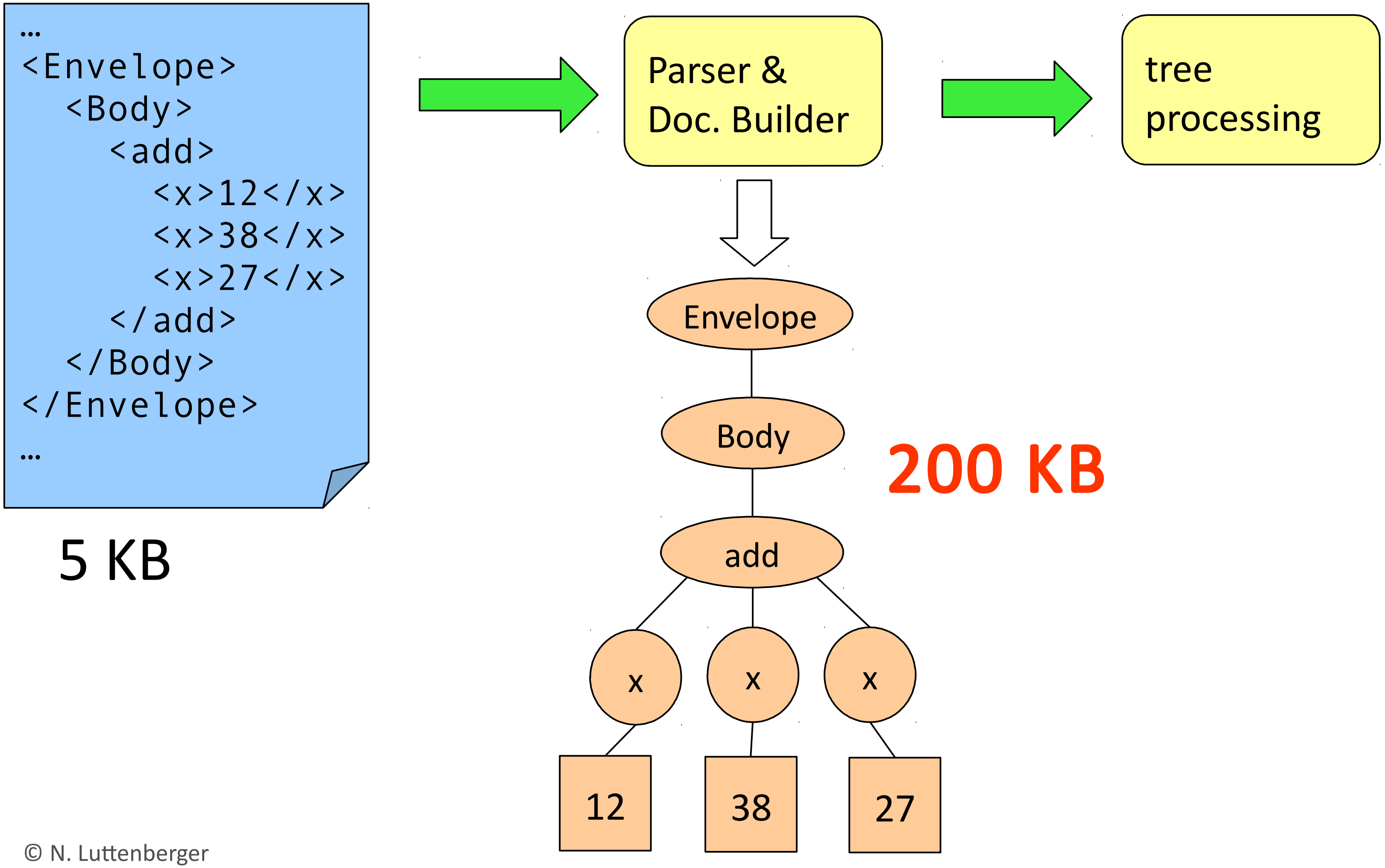
- XML processing



Introduction

- XML processors provide the structure and contents of XML documents to applications through APIs
  - Tree-based APIs
    - provide full parse tree to application
    - e.g., DOM W3C Recommendation (Document Object Model)
  - Stream-based APIs
    - notify application through parsing events
    - e.g., the SAX callback interfaces (Simple API for XML)

Tree-based APIs



```
…
<Envelope>
  <Body>
    <add>
      <x>12</x>
      <x>38</x>
      <x>27</x>
    </add>
  </Body>
</Envelope>
…
```

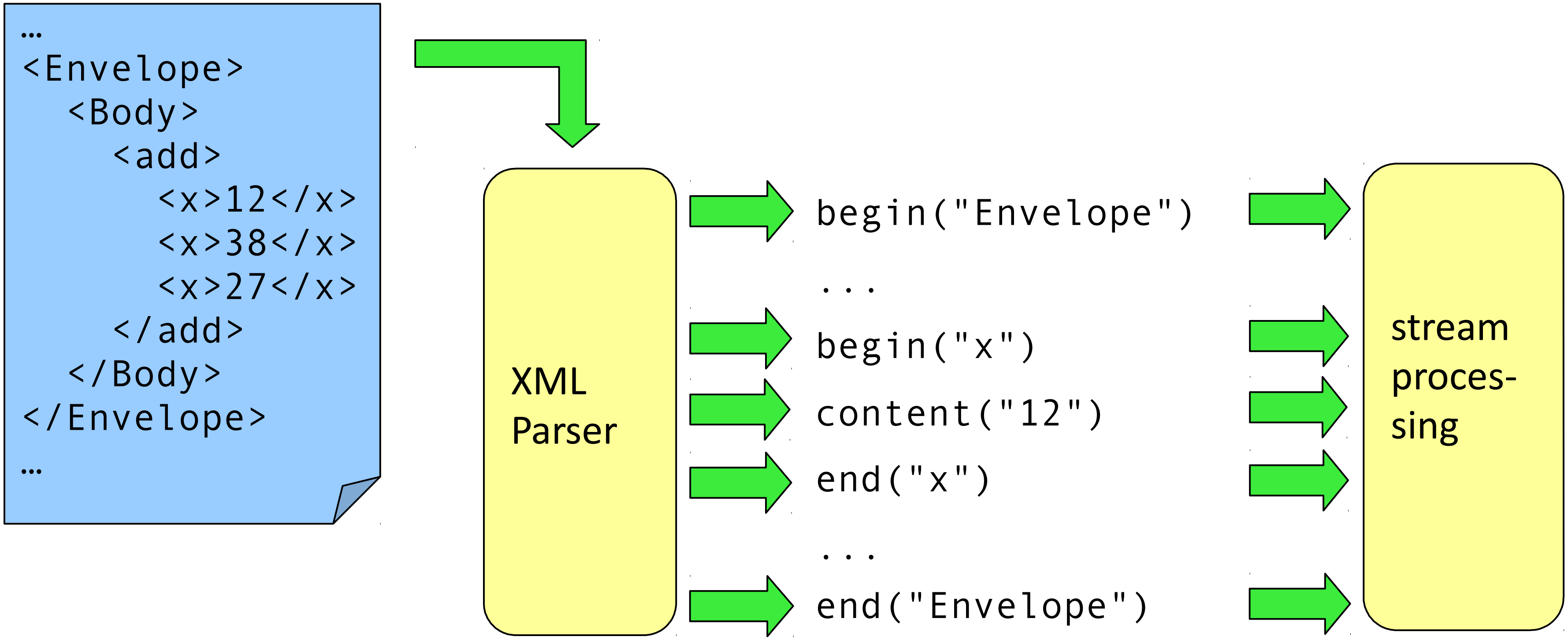
5 KB

Parser & Doc. Builder

tree processing

200 KB

Stream-based APIs



```
…
<Envelope>
  <Body>
    <add>
      <x>12</x>
      <x>38</x>
      <x>27</x>
    </add>
  </Body>
</Envelope>
…
```

XML Parser

```
begin("Envelope")
...
begin("x")
content("12")
end("x")
...
end("Envelope")
```

stream processing

Tree vs. Stream

- Pro and cons
  - tree processing
    - convenient tree navigation
    - straightforward impl. of modification operations

    but

    - consumes 10 to 100 times more memory than XML text
    - tree building must be completed before application processing starts
  - stream processing
    - low memory consumption
    - XML processing starts with first parsing event

    but

    - no internal document representation for navigation and processing
    - low overhead for validation of invalid documents

Chapter 6.2

Document Object Model (DOM)

Document Object Model (DOM)

- DOM—What is it?
  - With the DOM (Document Object Model), the W3C has defined a language- and platform-neutral view of XML documents much like the XML Information Set.
  - DOM APIs exist for a wide variety of—predominantly object-oriented—programming languages (Java, C++, C, Perl, Python, …).

Document Object Model (DOM)

- DOM—What is it?
  - DOM allows *programs and scripts*
    - to build documents,
    - to navigate their structure,
    - to add, modify or delete elements and content
    - for implementations of querying, filtering, transformation, rendering etc. applications
  - One could read DOM as "Directly Obtainable in Memory"

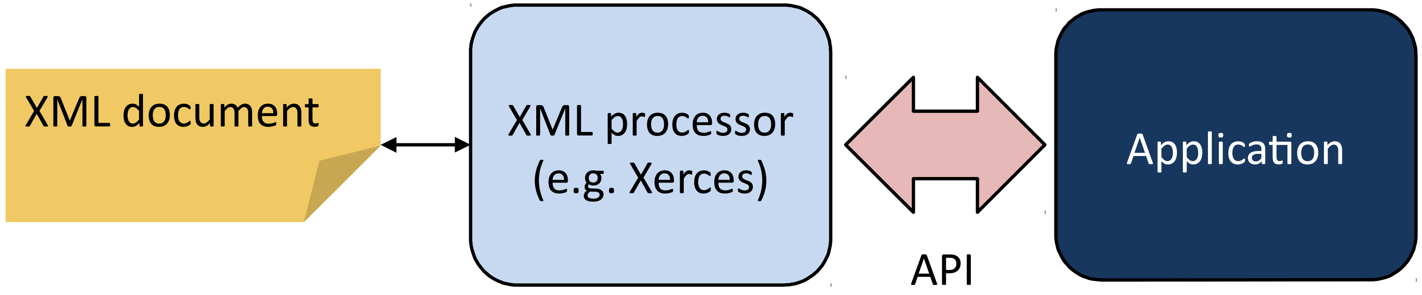
Document Object Model (DOM)

- From the Mozilla developer site

  "The Document Object Model (DOM) is a programming interface for HTML and XML documents. It provides a structured representation of the document and it defines a way that the structure can be accessed from programs so that they can change the document structure, style and content. The DOM provides a representation of the document as a structured group of nodes and objects that have properties and methods. Essentially, it connects web pages to scripts or programming languages."

Document Object Model (DOM)

- Two major DOM concepts



1. An XML Processor offering a DOM interface parses the XML input document, and constructs the complete XML document tree in memory.
2. The XML application then issues DOM library calls to explore and manipulate the XML document, or generate new XML documents.

Document Object Model (DOM)

- Example: JavaScript
  - All of the properties, methods, and events available for manipulating and creating web pages are organized into objects, e.g.,
    - the `document` object represents the document itself,
    - the `table` object implements the special `HTMLTableElement` DOM interface for accessing HTML tables, and so forth

Document Object Model (DOM)

- Example: JavaScript

```
<html>
  <head>
    <script>
      // run this function when the document is loaded
      window.onload = function() {
        // create a couple of elements
        // in an otherwise empty HTML page
        heading = document.createElement("h1");
        heading_text = document.createTextNode("Big Head!");
        heading.appendChild(heading_text);
        document.body.appendChild(heading);
        }
    </script>
  </head>
  <body></body>
</html>
```

Document Object Model (DOM)

- DOM structure model based on OO concepts
  - *methods* (to access or change object's state)
  - *interfaces* (declaration of a set of methods)
  - *objects* (encapsulation of data and methods)
- Roughly similar to the XPath/XSLT data model

  (to be discussed later)
  - DOM "product" ≈ a parse tree
- Language-independence
  - DOM interfaces defined in Interface Definition Language (IDL) from Corba Specification (OMG)

Document Object Model (DOM)

- W3C DOM Specification
  - second in the "XML family" of recommendations
    - Level 1, W3C Rec, Sept. 2000 (2nd ed.)
    - Level 2, W3C Rec, Nov. 2000
    - Level 3, W3C Rec, April 2004

Document Object Model (DOM)

- DOM Level 1

  Basic representation and manipulation of document structure and content

  - DOM Core Interfaces
    - Fundamental interfaces: basic interfaces to structured documents
    - Extended interfaces (XML-specific): CDATASection, DocumentType, Notation, Entity, EntityReference, ProcessingInstruction
  - DOM HTML Interfaces
    - more convenient access to HTML documents
  - No access to contents of DTD

Document Object Model (DOM)

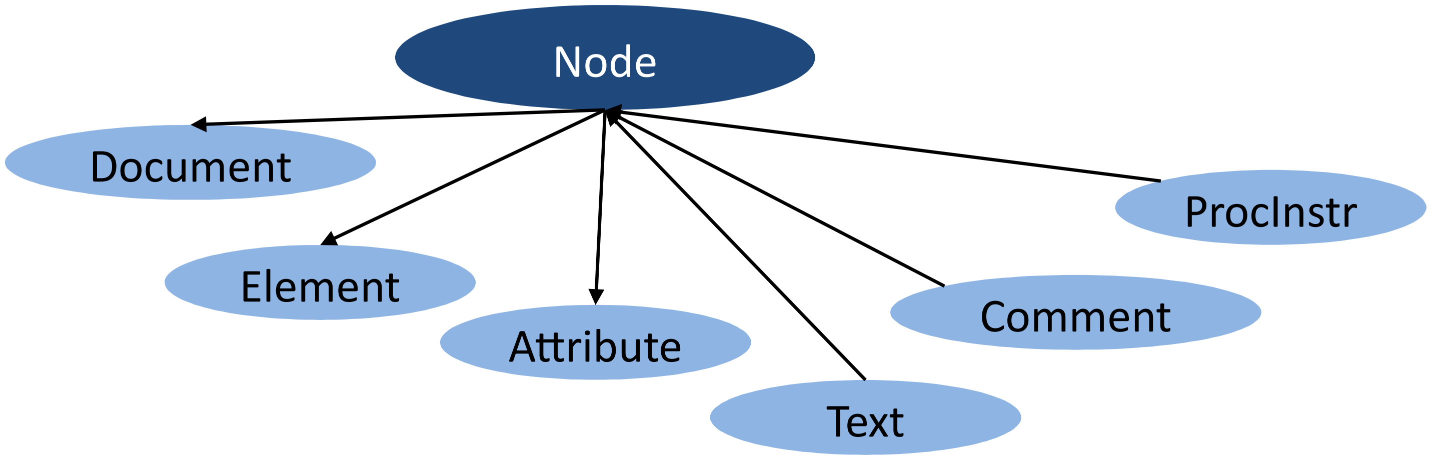
- DOM Level 2 adds
    - support for namespaces
    - accessing elements by ID attribute values
    - optional features
        - interfaces to document views and style sheets
        - an event model (for user actions on elements)
        - methods for traversing the document tree and manipulating regions of document (e.g., selected by the user of an editor)

Document Object Model (DOM)

- DOM Level 3 adds
  - Loading and writing of docs

Document Object Model (DOM)

- Excerpt from the inheritance hierarchy of node objects



NodeList – interface to handle ordered lists of Nodes, such as the elements returned by the Element.getElementsByTagName method

NamedNodeMap – interface to handle unordered sets of nodes referenced by their name attribute, such as the attributes of an Element.

Document Object Model (DOM)

- DOM establishes two basic types of operations

  1. Navigation:
     the ability to traverse the node hierarchy, and

  2. Reference:
     the ability to access a collection of nodes by name.

Document Object Model (DOM)

- Navigation
  - The structure of the document determines the inheritance of element attributes. Thus, it is important to be able to navigate among the node objects representing parent and child elements.
  - Given a node, you can find out where it is located in the document structure model and you can refer to the parent, child as well as siblings of this node.
  - This might be done using the `NodeList` object, which represents an ordered collection of nodes.

Document Object Model (DOM)

- Reference
  - Example: a gallery page is filled with individual images. A unique name or ID can be assigned to each image using the NAME or ID attribute.
  - It is possible to create an index of image names or ID's by iterating over a list of nodes.
  - To reference an image the `NamedNodeMap` object can be used, which represents an (unordered) collection of nodes that can be accessed by name.

Document Object Model (DOM)

- Node methods

```
getNodeType
hasAttributes
getParentNode
hasChildNodes
getLastChild
insertBefore(newChild,refChild)
replaceChild(newChild,oldChild)
getPreviousSibling
getOwnerDocument
getNodeValue
getAttributes

getChildNodes
getFirstChild
```

Document Object Model (DOM)

- Document methods

```
getDocumentElement
createAttribute(name)
createElement(tagName)
createTextNode(data)
getDocType()
getElementById(IdVal)
```

Document Object Model (DOM)

- Element methods

```
getTagName
getAttributeNode(name)
setAttributeNode(attr)
removeAttribute(name)
getElementsByTagName(name)
hasAttribute(name)
```

Document Object Model (DOM)

- Object creation in DOM
  - Each DOM object X lives in the context of a Document: `X.getOwnerDocument()`
  - Objects implementing interface X are created by factory methods `D.createX(…)`, where D is a `Document` object.
  - Examples:

    ```
    createElement("A"), createAttribute("href"),
    createTextNode("Hello!")
    ```

  - Creation and persistent saving of `Documents` left to be specified by implementations.

DOM: Implementations

- Java-based parsers
  e.g. IBM XML4J, Apache Xerces, Apache Crimson
- MS IE5 browser
  COM programming interfaces for C/C++ and MS Visual Basic,
  ActiveX object programming interfaces for script languages
- XML::DOM (Perl implementation of DOM Level 1)

Document Object Model (DOM)

- Concluding remarks
  - Inherent memory hunger of the DOM may lead
    - to heavy swapping activity or even "out-of-memory" failures.
  - To remedy:
    - Try to preprocess the input XML document to reduce its overall size.
    - Use an XPath/XSLT processor to preselect interesting document regions,
    - Or: Use a completely different approach to XML processing (→ SAX).