XML Data Management

3. Domain Object Model

Werner Nutt

Domain Object Model (DOM)

XML and HTML documents

- are (long) strings, physically
- represent trees

  ⇒ applications should manipulate documents abstractly as trees

  ⇒ define a tree API

W3C:

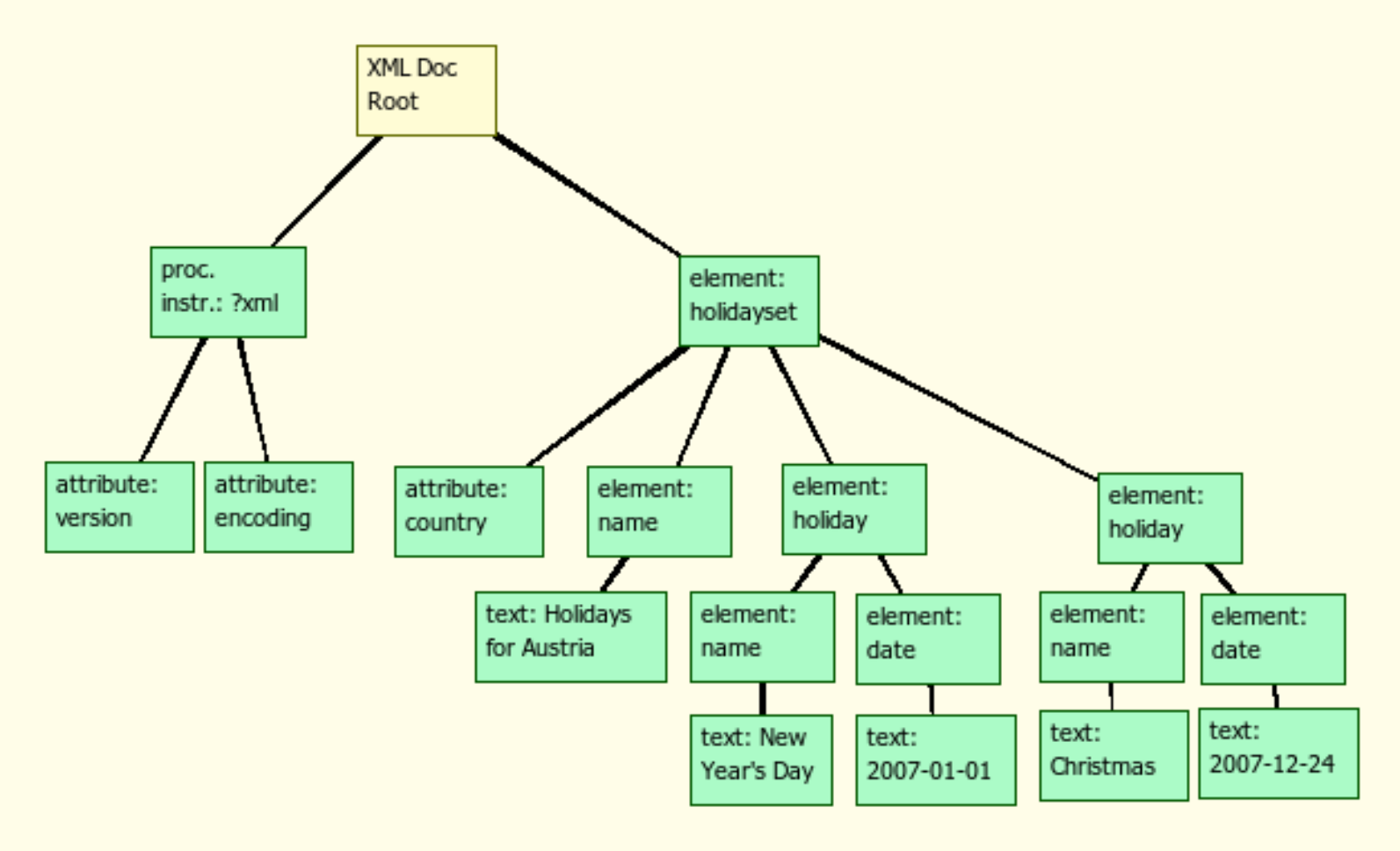
DOM is a platform and language-neutral interface
that will allow programs and scripts
to dynamically access and update
the content, structure and style of documents.

The document can
be further processed and the results
can be incorporated back into the presented page

Document

```
<?xml version='1.0' encoding='UTF-8'?>
<holidayset country="at">
  <name>Holidays for Austria</name>
  <holiday>
    <name>New Year's Day</name>
    <date>2007-01-01</date>
  </holiday>
  <holiday>
    <name>Christmas</name>
    <date>2007-12-24</date>
  </holiday>
</holidayset>
```

DOM Tree



Source: http://techbase.kde.org/Development/Tutorials/QtDOM_Tutorial

DOM Specifications

Level 0 (1996), Netscape et al.

- Facilities for modifying HTML documents on the client side in response to user-generated events

Level 1 (1998), W3C Rec

- Abstract tree model, capturing HTML and XML
- Functionality for navigating and manipulating documents

Level 2 (2000), W3C Recommendation

- Adds a style object to DOM, defines an event model, supports namespaces
- Level 2 Core: API to access and update content and structure of docs
- Level 2 HTML: API for HTML
- plus Level 2 Views, Style, Events, ...

Level 3 (2004)

- API for content models (DTD and Schemas) and XPath

DOM Objects and Interfaces

- Every component of a document is represented as an object
- Every object is an instance of a class
- The class interface defines the attributes and methods

  E.g., class Element has methods

  ```
  setAttribute(String name, String value)
  ```

- DOM interface is specified in IDL (= Interface Definition Language)

  http://www.w3.org/TR/DOM-Level-2-Core/idl/dom.idl

- Bindings for a variety of languages:

  Java, JavaScript, PHP, C++, VBScript, etc.

- The tree model underlying XPath is similar to DOM trees

Starting Point: The Node Interface (1)

```
interface Node {

  // Node Type
  const unsigned short   ELEMENT_NODE                = 1;
  const unsigned short   ATTRIBUTE_NODE              = 2;
  const unsigned short   TEXT_NODE                   = 3;
  const unsigned short   CDATA_SECTION_NODE          = 4;
  const unsigned short   ENTITY_REFERENCE_NODE       = 5;
  const unsigned short   ENTITY_NODE                 = 6;
  const unsigned short   PROCESSING_INSTRUCTION_NODE = 7;
  const unsigned short   COMMENT_NODE                = 8;
  // The top node of the tree
  const unsigned short   DOCUMENT_NODE               = 9;
  // Represents the DTD of the document
  const unsigned short   DOCUMENT_TYPE_NODE          = 10;
  const unsigned short   DOCUMENT_FRAGMENT_NODE      = 11;
  const unsigned short   NOTATION_NODE               = 12;
```

Starting Point: The Node Interface (2)

```
// Getting info about nodes
readonly attribute DOMString   nodeName;
         attribute DOMString   nodeValue;
                   // raises(DOMException) on setting
                   // raises(DOMException) on retrieval

// More info about nodes
readonly attribute unsigned short   nodeType;

// Node navigation
readonly attribute Node             parentNode;
readonly attribute NodeList         childNodes;
readonly attribute Node             firstChild;
readonly attribute Node             lastChild;
readonly attribute Node             previousSibling;
readonly attribute Node             nextSibling;

readonly attribute NamedNodeMap     attributes;
readonly attribute Document         ownerDocument;
```

Starting Point: The Node Interface (3)

```
// Methods to manipulate nodes
  Node          insertBefore(in Node newChild,
                             in Node refChild)
                                raises(DOMException);
  Node          replaceChild(in Node newChild,
                             in Node oldChild)
                                raises(DOMException);
  Node          removeChild(in Node oldChild)
                                raises(DOMException);
  Node          appendChild(in Node newChild)
                                raises(DOMException);
  boolean       hasChildNodes();
```

Starting Point: The Node Interface (4)

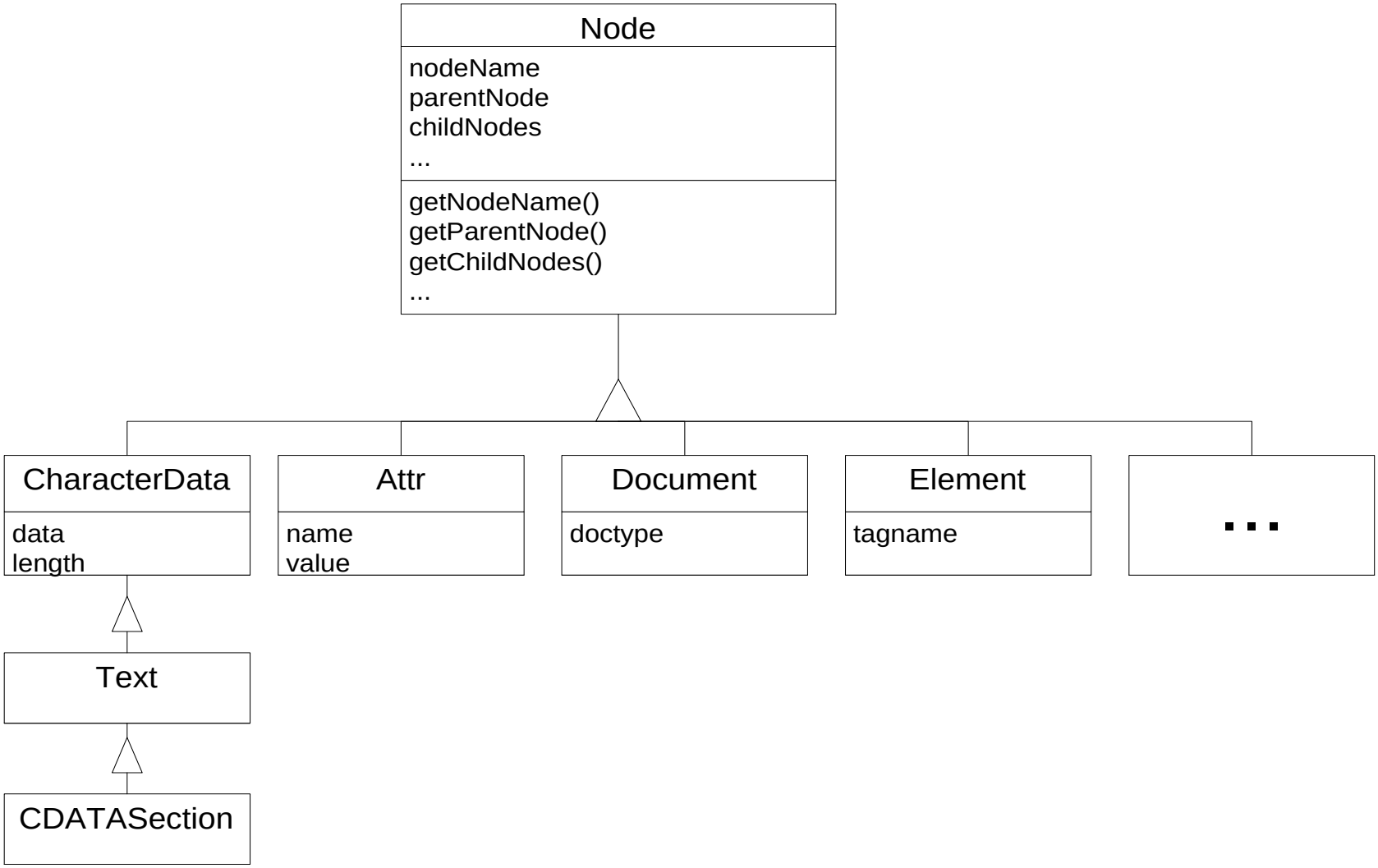
```
  Node        cloneNode(in boolean deep);

  // Merge adjacent text nodes etc.
  void        normalize();

  // tests DOM implementation for feature
  // e.g., isSupported('XHTML', '2.0')
  boolean     isSupported(in DOMString feature,
                          in DOMString version);
  // Namespace of the element
  readonly attribute DOMString      namespaceURI;

  // Introduced in DOM Level 2:
  attribute DOMString               prefix;
  readonly attribute DOMString      localName;
  boolean      hasAttributes();
};
```

DOM Node Interfaces: Overview



Node Interface: Comments

- Not all methods are applicable to all nodes
  - E.g., **`appendChild`** for a text node leads to an exception

  $\Rightarrow$ when in doubt, check the node type

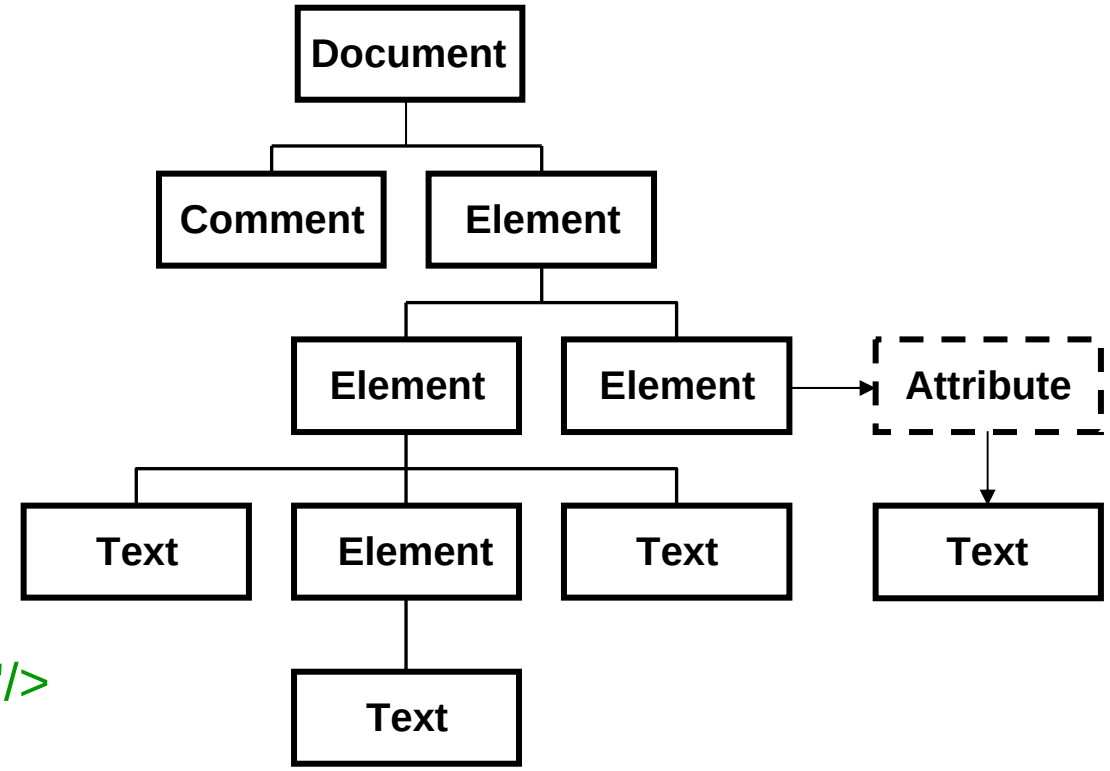
- Not all node attributes have a value
  - E.g., a comment does not have a name, the value is null

  $\Rightarrow$ see table on next slide for **`nodeNames`** and **`nodeValues`**

Interface	nodeName	nodeValue
Attr	name of attribute	value of attribute
CDATASection	#cdata-section	content of the CDATA Section
Comment	#comment	content of the comment
Document	#document	null
DocumentFragment	#document-fragment	null
DocumentType	document type name	null
Element	tag name	null
Entity	entity name	null
EntityReference	name of entity referenced	null
Notation	notation name	null
Processing Instruction	target	entire content excluding the target
Text	#text	content of the text node

DOM Tree: Example

```
<?xml version="1.0"?>
<!--DOM Demo-->
<xdoc>
  <welcome>
        Hello,
        <polite>Ladies</polite>
        and Gentlemen
  </welcome>
  <applause kind="sustained"/>
</xdoc>
```



Element, Text

Elements

- can have children (Elements and Text)
- can have attributes

Text

- inherits from Character Data (substringData, insertData)
- contains as data a DomString
  - sequence of 16-bit units
  - coded in UTF-16 *(= Unicode Transformation Format)*

Attribute Nodes

Different from Elements

- attributes are not children of a node (rather properties)
  - no sibling and parent relationships
    $\Rightarrow$ accessible by element navigation
  - no ordering among attributes
    $\Rightarrow$ access by name
- attributes can have values (but need not)
  - assigned values
  - default values (from DTD)

Document: The Mother of all Nodes

The root of the DOM tree is a node of type Document

- access point for the data in the tree:
  “owner document” of the nodes contained
- provides factory methods for the generation
  of other nodes
- distinct from root element of the document
- children:
  - Element
  - possibly DocumentType, Comment,
    ProcessingInstruction, etc.

Document (cntd)

```
interface Document : Node {
  readonly attribute DocumentType     doctype;
  readonly attribute DOMImplementation  implementation;
  readonly attribute Element          documentElement;
  Element             createElement(in DOMString tagName)
                                        raises(DOMException);
  DocumentFragment    createDocumentFragment();

  Text                createTextNode(in DOMString data);
  Comment             createComment(in DOMString data);
  CDATASection        createCDATASection(in DOMString data)
                                        raises(DOMException);
  ProcessingInstruction
              createProcessingInstruction(in DOMString target,
                                        in DOMString data)
                                        raises(DOMException);
  Attr                createAttribute(in DOMString name)
                                        raises(DOMException);
```

Document (cntd)

```
EntityReference   createEntityReference(in DOMString name)
                                        raises(DOMException);

NodeList    getElementsByTagName(in DOMString tagname);

Node        importNode(in Node importedNode,
                       in boolean deep)
                                raises(DOMException);

Element     createElementNS(in DOMString namespaceURI,
                            in DOMString qualifiedName)
                                raises(DOMException);

Attr        createAttributeNS(in DOMString namespaceURI,
                              in DOMString qualifiedName)
                                raises(DOMException);

NodeList    getElementsByTagNameNS(in DOMString namespaceURI,
                                   in DOMString localName);

Element     getElementById(in DOMString elementId);
};
```

Java API for XML Processing (JAXP)

Implements DOM (and SAX/StAX parsing interfaces + XSLT)

Packages

- org.w3c.dom
  - contains Java version of DOM interfaces
  - Java binding for DOM
- javax.xml.parsers contains classes
  - DocumentBuilder: creates instances of Document
  - DocumentBuilderFactory:
    creates instances of DocumentBuilder