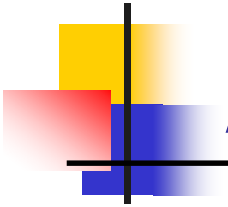
Digging into the Web with a Tiger: Lower-level Data Manipulation with J2SE 5.0 – Part 1

Neil Graham

Manager, XML Parser Development

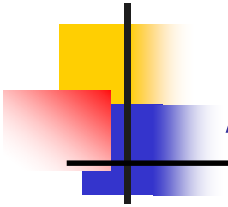
IBM Canada Ltd.

neilg@ca.ibm.com



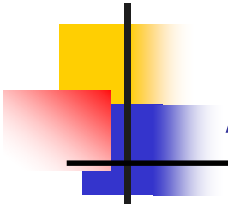
Agenda — Part 1

- Evolution of JAXP
- Overview of JAXP 1.3
- XML 1.1
- XInclude
- JAXP utilities



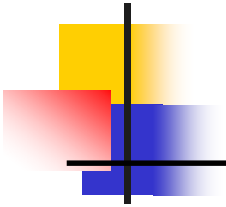
Agenda — Part 1 *(Continued)*

- Details on changes to javax.xml.parser and javax.xml.transform
- Overview of SAX, changes in SAX 2.0.1/2.0.2
- Overview of DOM
- DOM Level 3 — Core
- DOM Level 3 — Load and Save



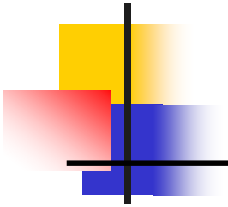
Agenda — Part 2

- Overview of XML Schema
- Examine changes to XML Schema, 2nd Edition
- XML Schema to Java datatype mapping
- JAXP 1.3 validation API
- JAXP 1.3 XPath API



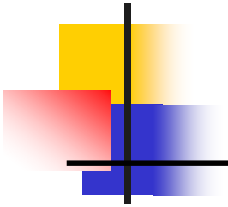
Evolution of JAXP

- JAXP: originally the Java API for XML Parsing
- JAXP 1.0 just contained interfaces/factories for invoking DOM and SAX parsers in an implementation-independent way
- JAXP now stands for the Java API for XML Processing; has been so since version 1.1 (Feb. 14, 2001)
  - JAXP 1.1 added API for invoking an XSLT transformer in an implementation-independent manner
  - Also provides means for treating DOM trees or SAX events as sources/results of transformations



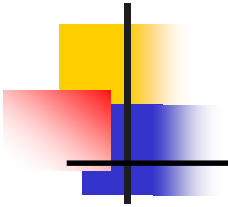
Evolution of JAXP *(Continued)*

- JAXP 1.2 (April 22, 2002): a maintenance release that added support for XML Schemas to parsing interfaces
- JAXP 1.3 (TBA — late Sept. 2004):
  - significant new revision
  - tracks developments in XML, DOM and SAX
  - new APIs for XPath processing and validation of SAX event streams and DOM trees



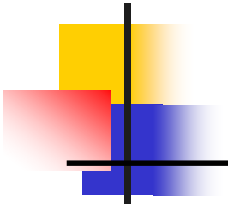
Overview of JAXP 1.3

- javax.xml: root, since 1.3 contains a class (XMLConstants) with useful constants
- javax.xml.parsers: since 1.0, contains DOM/SAX parser factories/interfaces
- javax.xml.transform: since 1.1, consists of interfaces/factories for XSLT transformations



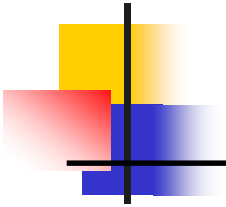
Overview of JAXP 1.3 *(Continued)*

- javax.xml.namespace: since 1.3, classes/interfaces for namespace manipulation (QName, NamespaceContext) (originally defined in JAXRPC 1.1)
- javax.xml.datatype: since 1.3, defines factories and interfaces providing Java types for XML Schema datatypes with no other Java mapping
- javax.xml.validation: since 1.3, interfaces for building in-memory representations of grammars (*e.g.*, XML Schemas) for use in validating documents represented as DOM trees or SAX event streams



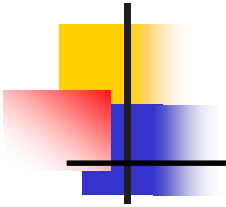
Overview of JAXP 1.3 *(Continued)*

- javax.xml.xpath: since 1.3, data model- and implementation-independent API for applying XPath expressions for documents
- org.xml.sax: Simple API for XML, endorsed API as defined in the opensource community (primarily on xml.org and sourceforge.net)
- org.w3c.dom: Document Object Model interfaces, endorsed by JAXP and defined by W3C



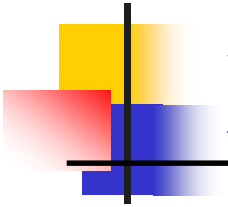
JAXP and XML

- XML 1.0 is in its 3rd edition
- Each edition has included further clarifications to enhance interoperability — no significant changes
- JAXP 1.3 requires parsers to conform to the 3rd edition
- XML 1.1, published Feb. 4, 2004
- JAXP 1.3 requires parsers to implement XML 1.1
- JAXP 1.3 also requires parsers to implement XML Namespaces 1.0 and 1.1



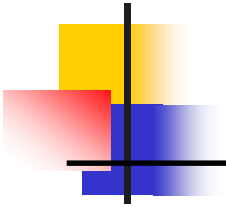
XML 1.1

- XML 1.0 was tethered to Unicode 2.0 for its definition of legal name characters.
- XML 1.1 specifies that parsers must accept any legal Unicode character plus tab, carriage return and line feed, in XML content
- XML 1.1 also allows virtually any Unicode character in names — including those that have yet to be assigned meanings in that standard
- Control characters are now permitted in character content (*e.g.*, &#7; is legal, though char 7 can't appear directly)



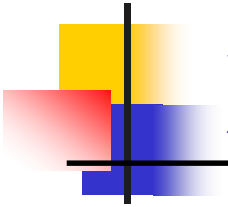
XML 1.1

- XML 1.1 also adds the Unicode line separator (0x2028) and the IBM mainframe newline (NEL, 0x85), to the set that participate in whitespace normalization
  - that is, are normalized to a linefeed, (0x0a) in the parser's initial scan of the document
- XML 1.0 permitted 0x7f-0x84, 0x86-0x9f to appear directly; these must be character references in XML 1.1



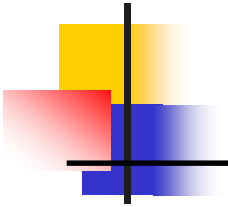
XML 1.1 *(Continued)*

- XML 1.1 states that processors should provide the ability for applications to determine whether a document is "fully normalized"
- Fully normalized:
  - Combining characters (*e.g.*, accent marks) may only occur with characters they do not combine with
    - (*e.g.*, a cedilla may occur after a 'b', but not after a 'c' since the 'c-cedilla' character must be used in such a situation)
  - No entity must begin or end with a combining character
  - Content, names, CDATA sections … must not begin or end with a combining character



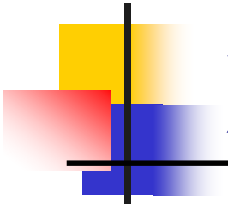
XML 1.1 *(Continued)*

- Full normalization is useful for assuring that semantically identical strings will be equal byte for byte
- JAXP 1.3 does not require parsers to support normalization checking



Introduction to XInclude

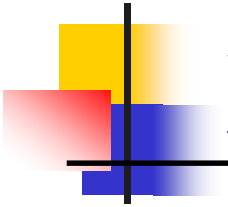
- XInclude (or XML Inclusions) describes a formalism for including content, either text or XML, that is located in one resource into another
- Including document can specify whether included resources are text or XML, and override autodetected encodings
- Location of resource can be relative to that of including document, or to some other location specified in the document using xml:base
- Portions of XML resources can be included through use of the XPointer framework



XInclude Example 1

- Including document:

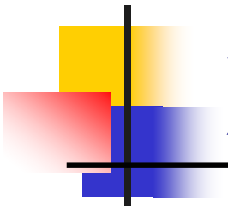
```
<x xmlns="http://www.tests.org/ex1"
   xmlns:xi="http://www.w3.org/2001/XInclude">
   <xi:include parse="xml"
               href="included/inc1.xml">
    <!-- this element+comment get replaced -->
    </xi:include>
</x>
```



XInclude Example 1 *(Continued)*

- Included document:

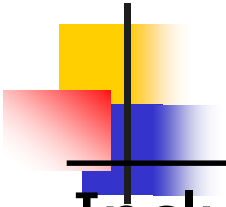
```
<?xml version="1.0"?>
<elem xmlns="http://www.tests.org/ex1">
    <p:content xmlns:p="http://www.fixup-
     test.org"/>
</elem>
```



XInclude Example 1 *(Continued)*

- Output:

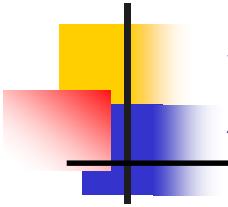
```
<?xml version="1.0" encoding="UTF-8"?>
<x xmlns="http://www.tests.org/ex1"
 xmlns:xi="http://www.w3.org/2001/XInclude">
   <elem xml:base="included/inc1.xml">
      <p:content xmlns:p="http://www.fixup-
       test.org"/>
   </elem>
</x>
```



XInclude Example 2

- Including document:

```
<x xmlns=http://www.tests.org/ex1 xml:base="included“
     xmlns:xi="http://www.w3.org/2001/XInclude”>
   <xi:include parse="xml" href="included/inc1.xml">
     <!-- that won't work... -->
     <xi:fallback>
        <xi:include parse="text“ href="inc2.txt"/>
     </xi:fallback>
   </xi:include>
</x>
```



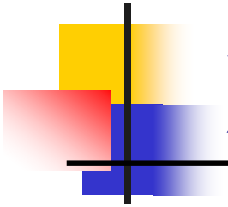
XInclude Example 2 *(Continued)*

- Included document:

```
Look at this – quite a bit different than last time!
And it'll certainly not get parsed as <xml/>.
```

- Output:

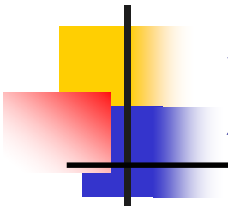
```
<?xml version="1.0" encoding="UTF-8"?>
<x xmlns="http://www.tests.org/ex1"
  xml:base="included/"
  xmlns:xi="http://www.w3.org/2001/XInclude">
        Look at this--quite a bit different than last
  time!  &#13; And it'll certainly not get parsed as
  &lt;xml/&gt;.&#13;
&#13;
</x>
```



XInclude Example 3

- Including document:

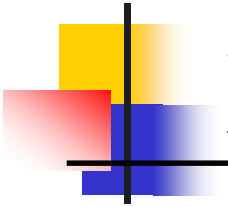
```
<x xmlns="http://www.tests.org/ex1"
   xmlns:xi="http://www.w3.org/2001/XInclude">
   <xi:include parse="xml"
               xpointer="element(/1/1)"
               href="inc3.xml"
               xml:base="included/">
    <!-- this element+comment get replaced -->
    </xi:include>
</x>
```



XInclude Example 3 *(Continued)*

- Included document:

```
<?xml version="1.0"?>
<elem xmlns="http://www.tests.org/ex1">
    <p:content xmlns:p="http://www.fixup-
     test.org">
       here is some new content
    </p:content>
</elem>
```

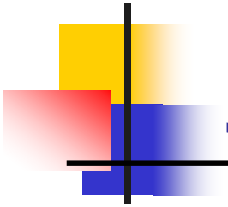


XInclude Example 3 *(Continued)*

- Output:

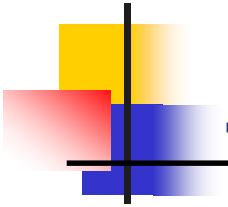
```
<?xml version="1.0" encoding="UTF-8"?>
<x xmlns="http://www.tests.org/ex1"

  xmlns:xi="http://www.w3.org/2001/XInclude">
   <p:content xmlns:p="http://www.fixup-
     test.org">
       here is some new content
    </p:content>
</x>
```



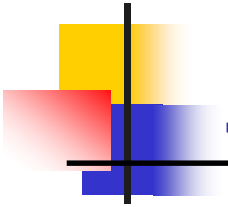
JAXP Utilities

- javax.xml.XMLConstants: defines constants useful in many XML contexts
  - Namespace of XML 1.0-defined attributes (*i.e.*, xml:space, xml:lang) as well as xmlns
  - namespace for XML Schema documents (*i.e.*, <xs:schema>), namespace of attributes defined to have meaning in XML Schema validation of instance documents (*e.g.*, schemaLocation), namespace of Relax NG documents
  - ID of feature for enabling "secure mode" in parsers (*e.g.*, causes parsers not to process internal entity declarations that could cause a denial-of-service attack)



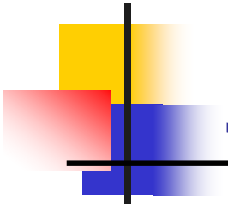
JAXP Utilities *(Continued)*

- javax.xml.namespace.NamespaceContext: stores prefix—namespace mapping for a given (document) context
- Provides methods to get the namespace URI for a given prefix, get a prefix for a given namespace URI, or get all prefixes bound to a namespace URI
- javax.xml.namespace.QName: representation for a name with a namespace



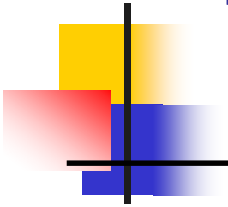
JAXP Utilities *(Continued)*

- *e.g.*, the name of the element <hr:employee xmlns:hr=http://www.management.com/> would be constructed as new QName(http://www.management.com, “employee”, “hr”)
- May also be constructed without a prefix (and, indeed, without a namespace URI)
- Contains getNamespaceURI(), getPrefix(), getLocalPart()



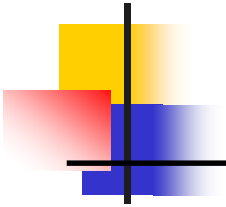
JAXP Utilities *(Continued)*

- The .equals and .hashCode() methods do not take the value of prefix into account
- The toString() method outputs contents according to a semi-standard formalism defined by James Clarke (the above would be "{http://www.management.com}employee").



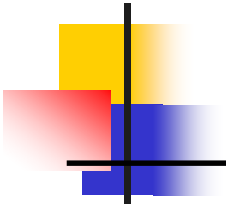
JAXP: Changes in Parsers Package

- Added reset() methods to DocumentBuilder and SAXParser
  - Will return these objects to the same state as upon creation; good for parser pools when properties/features need to be changed between invocations
- Added setSchema (see validation discussion in part 2) and setXIncludeAware methods to SAXParserFactory and DocumentBuilderFactory
- Added isXIncludeAware and getSchema to all abstract classes



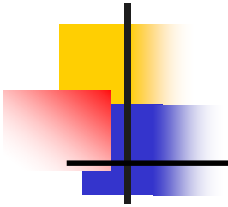
SAX Overview

- SAX 1.0: barebones API for event-based XML parsing; no namespaces, many things underspecified
  - Examples of interfaces: DocumentHandler, Parser, AttributeList; HandlerBase was the basic no-op implementation
  - Originated EntityResolver, ErrorHandler, DTDHandler, InputSource
- SAX 2.0: endorsed by JAXP 1.0 and above



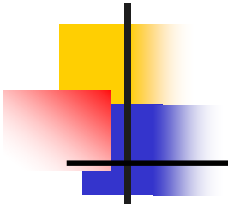
SAX Overview *(Continued)*

- SAX 2.0 details:
  - Introduced namespace-awareness
  - Introduced implementation-agnostic parser invocation mechanism (XMLReaderFactory)
  - Extensions framework to permit less-common application needs (*e.g.*, lexical information about entitiy references)



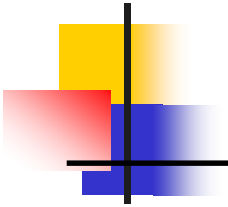
SAX Overview *(Continued)*

- SAX 2.0 details cont'd:
  - Helpers sub-package with new factory mechanism, DefaultHandler class (same as HandlerBase)
  - To avoid confusion with SAX 1.0, all additions defined in new interfaces-classes
  - Interfaces: ContentHandler, XMLReader, Attributes



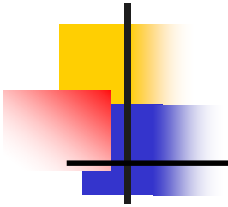
SAX Since 2.0

- SAX 2.0.1: January 29, 2002
  - Added default constructors to exception classes (broke signature-compatibility; that's why JAXP 1.2 could not endorse it)
  - Allowed EntityResolver.resolveEntity to throw IOExceptions as well as SAXExceptions (another signature compatibility breach)
  - Many documentation fixes/clarifications



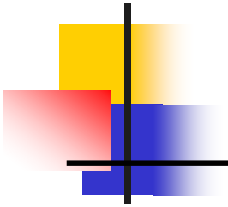
SAX since 2.0 *(Continued)*

- SAX 2.0.2: April 26, 2004
  - Finalized Extensions 1.1
    - Support for extended entity resolution (EntityResolver2 interface, resolveEntity(name, publicId, systemId, baseURI), getExterneralSubset(name, baseURI))
    - Added Attributes2, extends the Attributes interface by telling which attributes were specified/declared in the DTD
    - Locator2 extends Locator by providing access to XML document version information



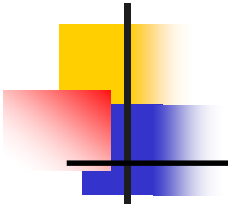
SAX 2.0.2 *(Continued)*

- Many more documentation fixes
- New features (all begin with "http://xml.org/sax/features/"):
  - Resolve-dtd-uris: whether systemId's passed to declaration events (*e.g.*, DTDHandler.notationDecl(), DTDHandler.unparsedEntityDecl()) will be fully resolved
  - String-interning: whether XML names and namespaces will be interned



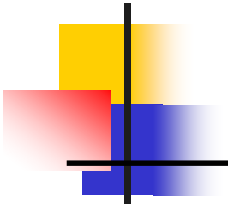
SAX 2.0.2 *(Continued)*

- New features (all begin with "http://xml.org/sax/features/" cont'd:
  - Unicode-normalization-checking: sets whether XML 1.1 normalization-checking will be performed
  - Xml-1.1: read-only, reflects whether the parser is able to process XML 1.1 documents



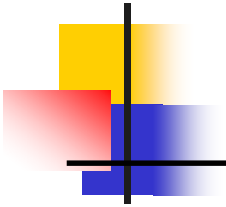
DOM: Evolution

- DOM level 1: 1 October, 1998; DOM level 1, 2nd edition: 29 September, 2000
  - Non-namespace-aware
  - Basic decision to make a language-independent API had been made
  - Basic structure apparent (Element, Attribute, Comment, … interfaces inheriting from Node)



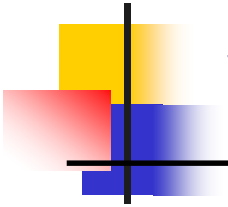
DOM: Evolution *(Continued)*

- DOM level 2 core: 13 November, 2000; same date for Events, Style, Traversal, Range and Views; DOM level 2 HTML: 9 January, 2003
  - Namespace-aware
  - Unlike SAX, reused existing interfaces for namespace-aware methods



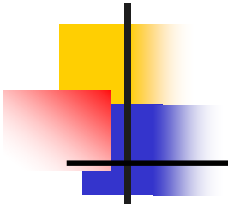
DOM: Evolution *(Continued)*

- DOM level 2 core cont'd
  - Because of limitations of languages without polymorphism, all namespace-aware methods end in "NS"
  - Added implementation-agnostic functionality for creating Documents DocumentTypes (to DOMImplementation)
  - Added above-mentioned modules so that different implementations could specialize in the functionalities they supported



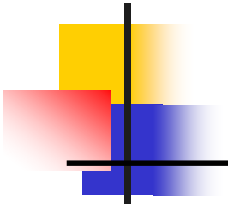
DOM Level 3 — What Isn't in JAXP

- New XPath module to permit querying of DOM's using xpath expressions
- New validation module permitting in-memory validation of DOM's
- Considerable modifications to events module to take advantage of changes in Core
- JAXP never supported DOM HTML module (either level 1 or level 2)



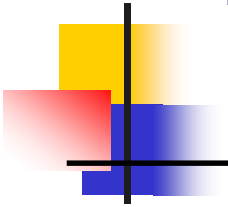
DOM Level 3 Core: Node Changes

- Added getBaseURI method: equivalent to querying what xml:base would be for the Node
- compareDocumentPosition(Node): short: determine whether the Node parameter is before, after or identical to this Node in document order (as defined by XPath 1.0)
- get/setTextContent: returns/replaces the complete textual content of the node (or NodeValue for Comments and ProcessingInstructions); null/unsettable for Documents



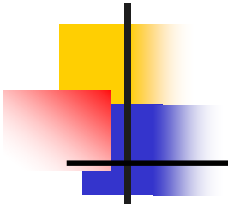
DOM Level 3 Core: Node Changes *(Continued)*

- lookupNamespaceURI: given a prefix, determine its namespace URI
- isSameNode(Node): boolean: determine whether two Nodes refer to the same object
- isEqualNode(Node): boolean: determine if two nodes are equal (same type, same name/namespace, same content) even if not the same object
- getFeature(String feature, String version): Object: return an implementation-specific Object (if one exists) supporting the feature/version pair



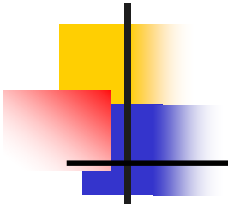
Changes *(Continued)*

- setUserData(String key, Object data, UserDataHandler handler): Object:
  - associates some user-specified data with a String key on the Node
  - Returns any previously-associated data
  - UserDataHandlers will be called when the Node's status changes — it is imported, adopted, renamed *etc*.
  - Key can also be used to retrieve the data from the node (getUserData(String): Object)



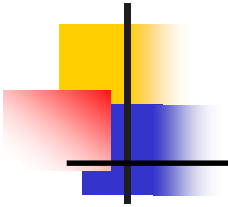
DOM Level 3 Core *(Continued)*

- Added a new TypeInfo interface
  - Allows querying of type name/namespace that validated an attribute or element
  - Contains method isDerivedFrom(typeName, typeNamespace, derivationMethod): boolean to help determine whether there is a derivation relationship between two types
- Attr: permits TypeInfo for the Attr to be queried; also provides a method to check whether the attribute is of type ID



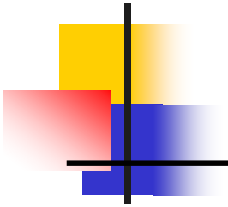
DOM Level 3 Core *(Continued)*

- Element: method for querying Element's TypeInfo; also complete set of methods for attaching an attribute of type ID
- Text:
  - Allows content to be queries to see if it's whitespace
  - Provides methods to collapse adjacent text nodes (getWholeText()) and replaceWholeText(String newText) to replace node's contents and those of adjacent Text nodes)



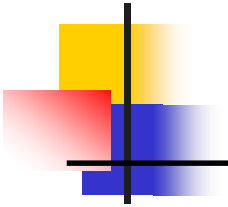
DOM Level 3 Document Changes

- Document: methods provided to get/set XML version, value of standalone pseudoattribute and declared (and to get actual) encoding
- Get/set document's base URI
- Set/get whether all possible error checks are performed on DOM operations
- Added method adoptNode(Node source): Node: imports source into this Document, simultaneously removing it from its original Document
- Convenience method provided to rename a node



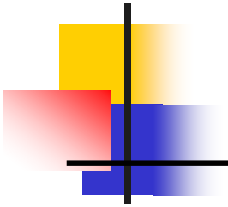
DOM Level 3 Document Changes *(Continued)*

- Method normalizeDocument(): Depending on the DOMConfiguration associated with the Document:
  - Updates tree of EntityReference Nodes
  - Performs Text node normalization
  - May perform namespace-fixup on the document
  - May revalidate the document
  - May even perform Unicode normalization on the Document



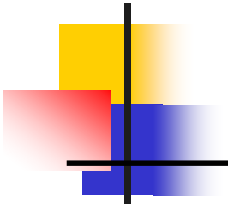
DOM Level 3 DOMConfiguration

- Allows name-value pairs to be attached to a Document
- Also can supply a list of parameter names recognized
- Permits LSResourceResolvers and DOMErrorHandlers to be associated with Document
- Also is means by which all features of normalizeDocument (well-formedness checking, validation, namespace-fixup *etc*.) are controlled



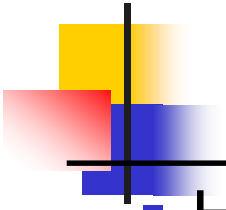
DOM Level 3 Core *(Continued)*

- Entity: may now query the actual and declared encodings as well as the XML version of the entity
- DOMImplementation: added getFeature(String feature, String version): Object. Allows objects implementing different modules to be returned without them all having to be defined on the same object
- *e.g.*, can query for xpath 3.0 or html 2.0



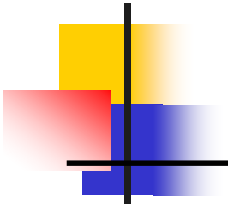
DOM Level 3 Core: Error Reporting

- DOMErrorHandler: implemented by application and attached to DOMConfiguration; allows normalizeDocument() *et al* to provide application with feedback without throwing an exception
- DOMError: object returned to DOMErrorHandler that associates error code with DOMLocator, message text and any related Exception
- DOMLocator: analogous to SAX Locator
  - may provide any or all of line/column or byte or UTF-16 offset
  - Also may provide Node at which error occurred



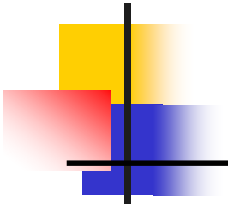
DOM Level 3 Core: Bootstrapping

- How does an application get a DOMImplementation to create a Document, for instance?
- org.w3c.dom.bootstrap.DOMImplementationRegistry: looks over the system, in a manner analogous to JAXP's factory mechanisms, to find all DOMImplementationSources
- Application can then query the registry for an appropriate DOMImplementation supporting some features, or ask it to provide all DOMImplementations it knows about that support those features



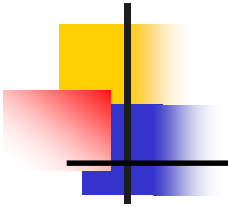
DOM Level 3 Core: Bootstrapping *(Continued)*

- An application can also register its own DOMImplementationSource on the singleton DOMImplementationRegistry instance
- DOMImplementationSource: contains a set of DOMImplementations, associated with features (application-specific and provided by DOM) and versions



DOM Level 3 Core: Bootstrapping *(Continued)*

- Application can get a specific implementation supporting features, or a complete list of such implementations
- Lists of DOMImplementations come in DOMImpleentationLists — a very simple iterator



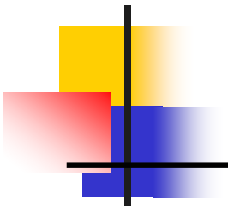
DOM Level 3 Core: Example

- The following should be embedded in a try block for safety

```
// this is bogus; JDK doesn't seem to contain
// default DOMImplementationSources
System.setProperty(
   DOMImplementationRegistry.PROPERTY,
  "com.sun.org.apache.xerces.internal.dom.DOMXS
    ImplementationSourceImpl");
// get DOM Implementation using DOM Registry
DOMImplementationRegistry registry =
    DOMImplementationRegistry.newInstance();
```

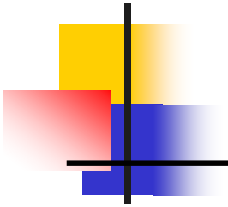
DOM Level 3 Core: Example *(Continued)*

```
DOMImplementation impl =
    registry.getDOMImplementation("LS");
Document doc =
  impl.createDocument("http://test.org",
  "root", null);
DOMConfiguration config =
  doc.getDomConfig();
// this class implements DOMErrorHandler
DOMErrorHandler errorHandler = new
  DOMBoot();
// set error handler
    config.setParameter("error-handler",
      errorHandler);
```



DOM Level 3 Core: Example *(Continued)*

```
// set validation feature
config.setParameter("validate",true);
// set schema language
config.setParameter("schema-type",
  XMLConstants.W3C_XML_SCHEMA_NS_URI);
// set schema location
config.setParameter(
   "schema-location","ex4.xsd");
```

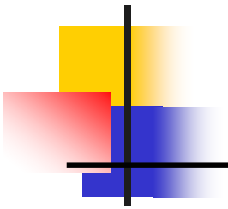


DOM Level 3 Core: Example *(Continued)*

```
// fill document
Element root = doc.getDocumentElement();
root.setAttributeNS("", "rootAttr", "42");
Element child =
  doc.createElementNS("http://test.org",
  "child");
Text text = doc.createTextNode(
   "some text ");
child.appendChild(text);
```

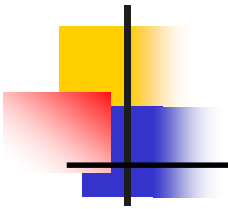
DOM Level 3 Core: Example *(Continued)*

```
text = doc.createTextNode(
  "and more text ");
child.appendChild(text);
root.appendChild(child);
System.out.println(
  "there are 2 text nodes.");
doc.normalizeDocument();
System.out.println("there are " +
  child.getChildNodes().getLength()
  + " text nodes.");
// make the doc invalid
child.setAttributeNS("", "badAttr",
  "no attrs on elem");
doc.normalizeDocument();
```



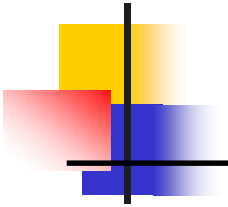
DOM Level 3 Core: DOMErrorHandler Implementation

```
public boolean handleError(DOMError error){
 short severity = error.getSeverity();    if (severity ==
  error.SEVERITY_ERROR) {
   System.out.println("[dom3-error]:"
    +error.getMessage());
  }
  else if (severity ==
    error.SEVERITY_WARNING) {
   System.out.println("[dom3-warning]:"
    +error.getMessage());
  }
  else {
        System.out.println(
  "[dom3-fatalError]:"
  +error.getMessage());
        return false;
    }
    return true;
}
```



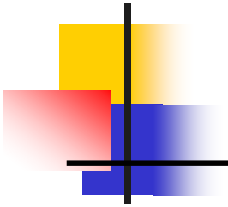
DOM Level 3 Core: DOMErrorHandler Implementation *(Continued)*

```
  else {
   System.out.println("[dom3-fatalError]:"
    +error.getMessage());
   return false;
  }
  return true;
 }
```



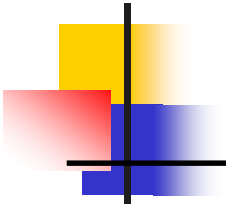
DOM Level 3 Core: Example Output

```
there are 2 text nodes.
there are 1 text nodes.
[dom3-error]: cvc-type.3.1.1: Element
  'child' is a simple type, so it cannot have
  attributes, excepting
those whose namespace name is identical to
  'http://www.w3.org/2001/XMLSchema-instance'
  and whose
[local name] is one of 'type', 'nil',
  'schemaLocation'
or 'noNamespaceSchemaLocation'. However, the
  attribute, 'badAttr' was found.
```



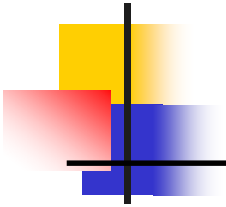
DOM Level 3 Load/Save

- Intended to provide an implementation-independent means to parse XML files into DOM trees and serialize DOM trees into XML files
- To bootstrap: must get a DOMImplementationLS object from the DOMImplementationRegistry
- There is no defined inheritance relation between DOMImplementation and DOMImplementationLS
- DOMImplementationLS can create LSParser, LSSerializer, LSInput and LSOutput objects



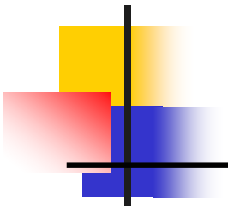
Load/Save: LSInput

- Much the same function as a SAX InputSource (may contain an InputReader, InputStream, systemId, declare an encoding, *etc*.)
- Unlike InputSource, it's an interface
- Also has a field for a String
- May contain baseURI information
- It may also be asserted that the XML is fully normalized



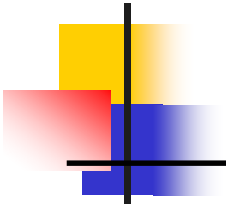
Load/Save: LSOutput

- Also an interface
- Contains methods for getting and setting both Writers and OutputStreams
- Also contains methods for getting/setting systemId and encoding



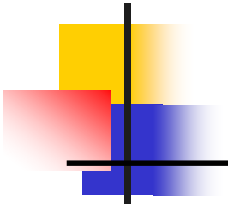
Load/Save: LSResourceResolver

- Much like SAX's EntityResolver2
- LSResourceResolver.resolveResource(…) does not contain a name parameter
- It does indicate the type of the entity being requested (*i.e.*, XML entity or XML Schema)
- Also indicates the namespace of the resource, if pertinent



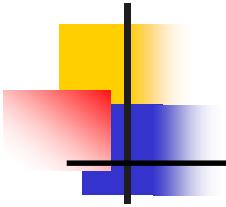
Load/Save: LSParser

- Allows a Document to be parsed from a URI or an LSInputSource
- Allows its DOMConfiguration to be queried (so parameters can be set)
- Application can query whether parser is busy
- LSParserFilters can also be set/queried



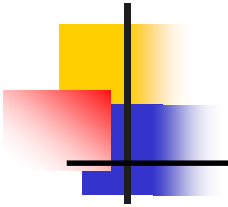
LSParser: Unimplemented Features

- LSParsers may operate asynchronously or synchronously
- Only synchronous operation is supported in Tiger
- parseWithContext(LSInput, Node, short): Node: allows a document fragment to be parsed into an existing Document, either augmenting or replacing children of Node



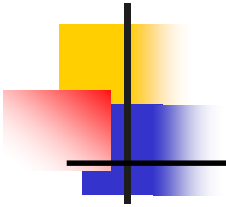
LSSerializer

- Permits writing a Node to a String, a URI, or an LSOutput
- Allows its DOMConfiguration to be queried (so parameters can be set)
- Application may set a string to be used when a newline is needed
- Allows setting/querying of a LSSerializerFilter



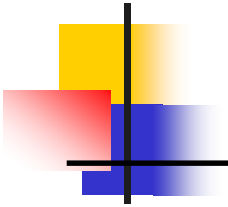
LSParserFilter

- Application should first call getWhatToShow(): int, which tells the LSParser what Node types to inform it of
- startElement(Element): short: presented to application after startTag sarsed; permits efficient skipping of subtree
- acceptNode(Node): short: called by LSParser after Node has been processed; application may accept, reject, or even modify it



LSSerializererFilter

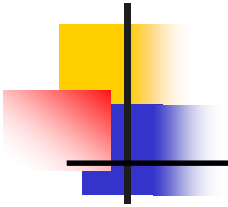
- Extends NodeFilter from traversals module
- getWhatToShow(…) and acceptNode(…) work analogously to LSParserFilter
- Note that the int returned by LSParserFilter#getWhatToShow() is defined in NodeFilter, though there's no formal dependence



DOM 3 LS: Example

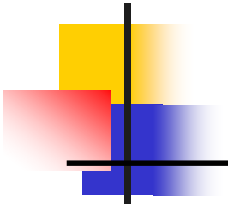
- The following parses a document, preventing all PI's or elements or attribute with localName "silly" from becoming part of the DOM, then serializes it

```
// this is bogus; JDK doesn't seem to contain
// default DOMImplementationSources
System.setProperty(
  DOMImplementationRegistry.PROPERTY,
  "com.sun.org.apache.xerces.internal.dom.
  DOMXSImplementationSourceImpl");
// get DOM Implementation using DOM Registry
DOMImplementationRegistry registry =
      DOMImplementationRegistry.newInstance();
```



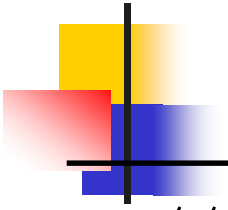
DOM 3 LS: Example *(Continued)*

```
// note the explicit need to cast
DOMImplementationLS impl =
(DOMImplementationLS)
 registry.getDOMImplementation("LS");
// create DOMBuilder
// not validating so schemaType is null
builder = impl.createLSParser(
  DOMImplementationLS.MODE_SYNCHRONOUS,
  null);
```



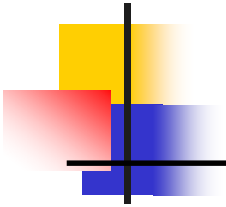
DOM 3 LS: Example *(Continued)*

```
DOMConfiguration config =
  builder.getDomConfig();
// create filter
LSParserFilter filter = new DOMLSTest();
builder.setFilter(filter);
// set validation feature
config.setParameter("validate", false);
config.setParameter("namespaces", true);
// parse document
System.out.println("Parsing "+argv[0]
    +", removing silly elements and PI's...");
Document doc = builder.parseURI(argv[0]);
```



DOM 3 LS: Example *(Continued)*

```
// create LSSerializer
LSSerializer domWriter =
  impl.createLSSerializer();
System.out.println("Serializing document");
config = domWriter.getDomConfig();
config.setParameter("xml-declaration",
  true);
// serialize document to argv[1]
LSOutput dOut = impl.createLSOutput();
FileOutputStream fos = new
  FileOutputStream(new File(argv[1]));
dOut.setByteStream(fos);
domWriter.write(doc,dOut);
```

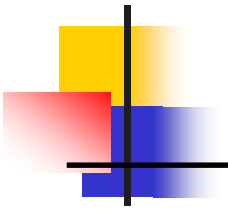


DOM 3 LS Example: acceptNode Implementation

```
public short acceptNode(Node node) {
  if(node.getNodeType() ==
      Node.ELEMENT_NODE) {
   // loop through attrs removing silly ones
       Element elt = (Element)node;
       NamedNodeMap attrMap =
      node.getAttributes();
   for(int i=0; i<attrMap.getLength(); i++ )
  {
    if(attrMap.item(i).getLocalName().
    equalsIgnoreCase("silly")){
        elt.removeAttributeNode((Attr)
          attrMap.item(i));
```

DOM 3 LS Example: acceptNode *(Continued)*

```
            }
             }
        }
        else if(node.getNodeType() ==
            Node.PROCESSING_INSTRUCTION_NODE) {
          return NodeFilter.FILTER_REJECT;
        }
        return NodeFilter.FILTER_ACCEPT;
    }
```

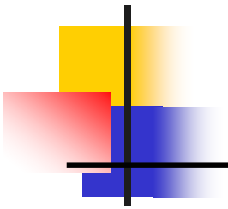


DOM 3 LS Example: getWhatToShow

```
public int getWhatToShow() {
 return NodeFilter.SHOW_PROCESSING_INSTRUCTION
        | NodeFilter.SHOW_ELEMENT;
}
```

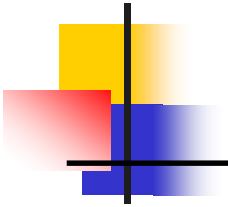
DOM 3 LS Example: startElement

```
public short startElement(Element elt) {
  if(elt.getLocalName().equalsIgnoreCase(
          "silly")) {
    return NodeFilter.FILTER_REJECT;
    }
   return NodeFilter.FILTER_ACCEPT;
}
```



DOM 3 LS Example: Example Instance

```
<root xmlns="http://www.tests.org/ex5"
    xmlns:ns="http://silly.com">
    <ns:serious>A serious element</ns:serious>
  <!-- and a comment -->
  <silly>
        <ns:serious>another serious
            element</ns:serious>
  </silly>
  <?processing instruction?>
  <serious silly="a silly attribute"
       attr="another attribute"/>
</root>
```



DOM 3 LS Example: Example Output

```
<?xml version="1.0" encoding="UTF-8"?>
<root xmlns="http://www.tests.org/ex5"
        xmlns:ns="http://silly.com">
    <ns:serious>A serious
     element</ns:serious>
    <!-- and a comment -->
    <serious attr="another attribute"/>
</root>
```