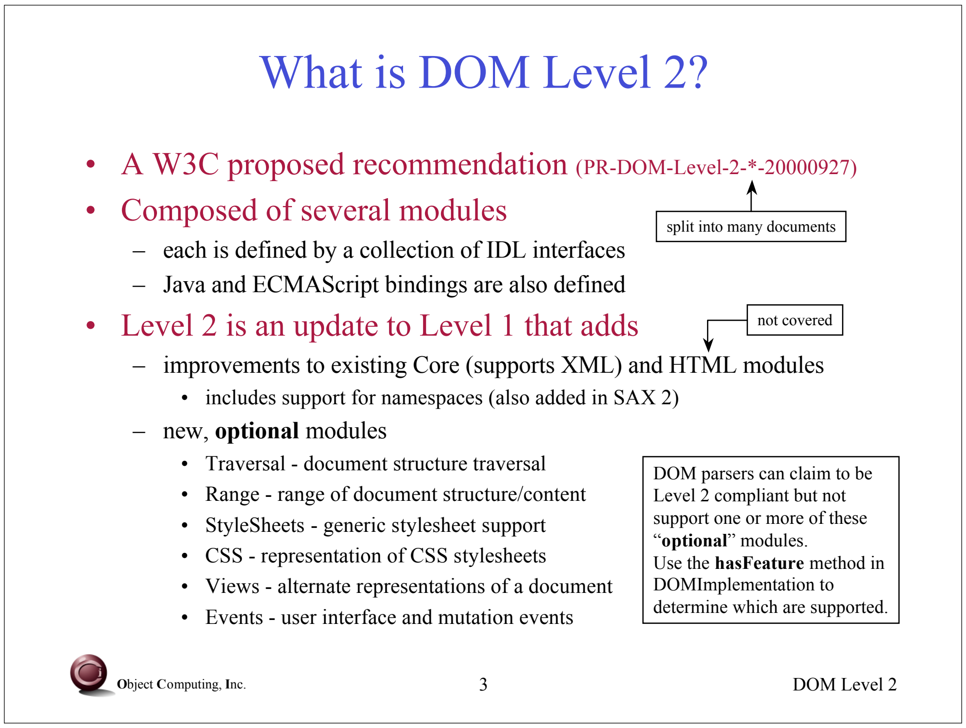
What's New in DOM Level 2?

Object Computing, Inc.

Mark Volkmann
Principal Software Engineer
http://ociweb.com
volkmann_m@ociweb.com

Contents



What is DOM Level 2?

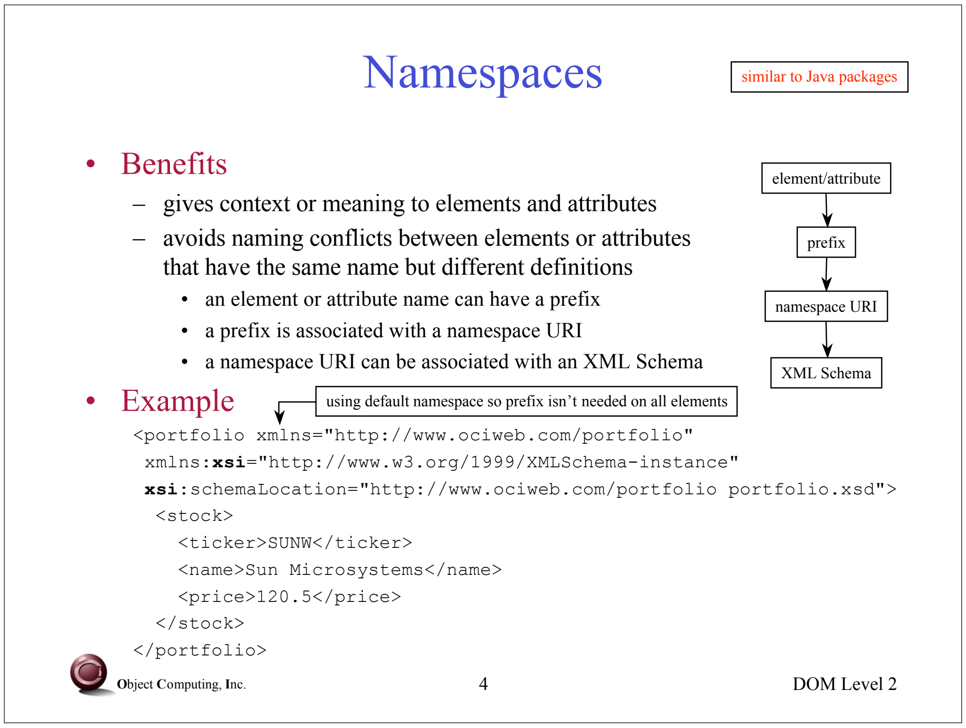
- A W3C proposed recommendation (PR-DOM-Level-2-*-20000927)
- Composed of several modules
  - each is defined by a collection of IDL interfaces
  - Java and ECMAScript bindings are also defined
- Level 2 is an update to Level 1 that adds
  - improvements to existing Core (supports XML) and HTML modules
    - includes support for namespaces (also added in SAX 2)
  - new, **optional** modules
    - Traversal - document structure traversal
    - Range - range of document structure/content
    - StyleSheets - generic stylesheet support
    - CSS - representation of CSS stylesheets
    - Views - alternate representations of a document
    - Events - user interface and mutation events

split into many documents

not covered

DOM parsers can claim to be Level 2 compliant but not support one or more of these "**optional**" modules.
Use the **hasFeature** method in DOMImplementation to determine which are supported.

Object Computing, Inc. 3 DOM Level 2



Namespaces

similar to Java packages

- Benefits
  - gives context or meaning to elements and attributes
  - avoids naming conflicts between elements or attributes that have the same name but different definitions
    - an element or attribute name can have a prefix
    - a prefix is associated with a namespace URI
    - a namespace URI can be associated with an XML Schema
- Example

using default namespace so prefix isn't needed on all elements

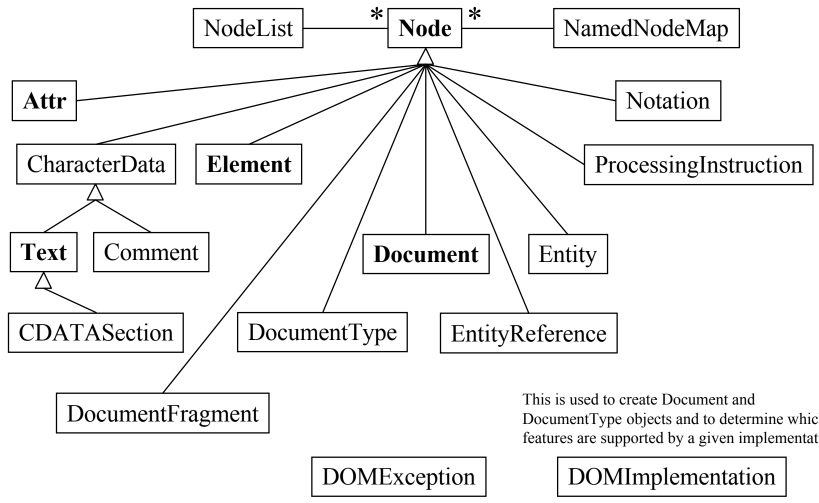
```
<portfolio xmlns="http://www.ociweb.com/portfolio"
 xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
 xsi:schemaLocation="http://www.ociweb.com/portfolio portfolio.xsd">
  <stock>
    <ticker>SUNW</ticker>
    <name>Sun Microsystems</name>
    <price>120.5</price>
  </stock>
</portfolio>
```

$$\begin{tikzcd}\boxed{\text{element/attribute}} \arrow[d] \\ \boxed{\text{prefix}} \arrow[d] \\ \boxed{\text{namespace URI}} \arrow[d] \\ \boxed{\text{XML Schema}}\end{tikzcd}$$

Object Computing, Inc. 4 DOM Level 2

Core Class Diagram

(mostly unchanged from DOM Level 1)



This is used to create Document and DocumentType objects and to determine which features are supported by a given implementation.

Additions To Core Interfaces

- `DOMImplementation` DOM Level 1 only provides **`hasFeature`** method
  - new methods **`createDocumentType`** and **`createDocument`** most are in the Document interface
  - instantiation is implementation specific (bootstrapping) but standard methods for creating all other DOM objects are defined
- `DocumentType` DOM Level 1 only provides access to root element name, entities and notations
  - new attributes `publicId`, `systemId` and `internalSubset`

    `<!DOCTYPE` *`rootName`* `SYSTEM "`*`systemId`*`" [`*`internalSubset`*`]>`

    `<!DOCTYPE` *`rootName`* `PUBLIC "`*`publicId`*`" "`*`systemId`*`" [`*`internalSubset`*`]>`

    `systemId` is a URI
- `Document`
  - new methods `createElement`**`NS`**, `createAttribute`**`NS`**, `getElementByTagName`**`NS`**, `getElementById`, `importNode`
    - **`importNode`** copies a node from another document and sets its `ownerDocument` attribute to this one
      - copy can be shallow or deep
      - still have to append it

    see Level 3 adoptNode method on page 71

Additions To Core Interfaces (Cont'd)

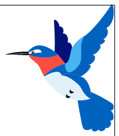
- Node
  - new methods
    - isSupported
      - like DOMImplementation hasFeature method but indicates whether a specific node type supports a given feature
    - normalize
      - eliminates empty child text nodes and combines adjacent child text nodes
  - new attributes namespaceURI, prefix and localName
- Element
  - new methods getAttribute**NS**, setAttribute**NS**, removeAttribute**NS**, getAttributeNode**NS**, setAttributeNode**NS**, getElementsByTagName**NS**, hasAttribute and hasAttribute**NS**
- Attr
  - new attribute ownerElement

Examples of features include validation, namespace processing, XML Schema support and expanding external entity references.

Additions To Core Interfaces (Cont'd)

- NamedNodeMap
  - new methods getNamedItem**NS**, setNamedItem**NS** and removeNamedItem**NS**
- DOMException
  - new constants for code attribute
    - INVALID_ACCESS_ERR
      - method or parameter not supported by DOM implementation class
    - INVALID_MODIFICATION_ERR
      - object type modification not allowed
    - INVALID_STATE_ERR
      - object in an invalid state
    - NAMESPACE_ERR
      - object creation or modification results in invalid namespace usage
    - SYNTAX_ERR
      - invalid string

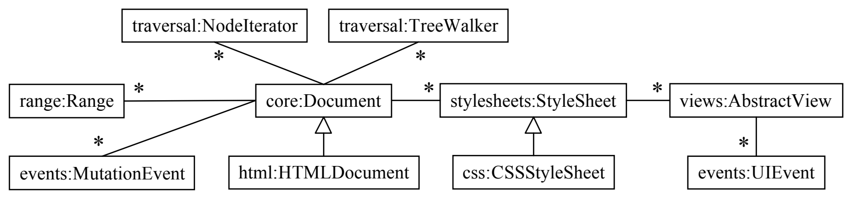
New Classes in Existing Modules

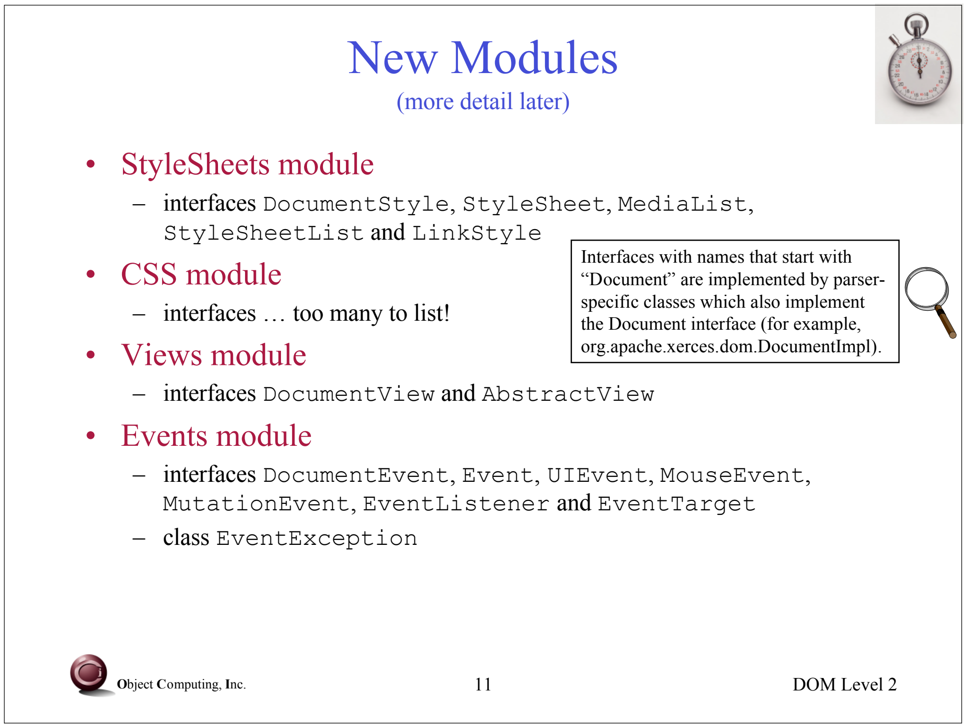


- DOMTimeStamp
  - added to the Core module
  - represents a time as a number of milliseconds
  - specific bindings can use a different type such as a long in Java or a Date object in ECMAScript
- HTMLDOMImplementation
  - added to the HTML module
  - extends `DOMImplementation`
  - adds `createHTMLDocument` method

Relationships Between Modules

- A **Range** identifies a subset of a Document
- Changes to the data and structure of a Document produce Mutation**Events**
- NodeIterators and TreeWalkers make **Traversal** of a Document structure easier
- An **HTML**Document is a specific kind of Document
- A Document can be associated with multiple **StyleSheets**
- **CSS** is a specific kind of StyleSheet ← Specific support for XSL stylesheets has not been defined yet but those can be represented using core objects.
- Applying a StyleSheet to a Document produces a **View**
- User interaction on a View produces UI**Events**



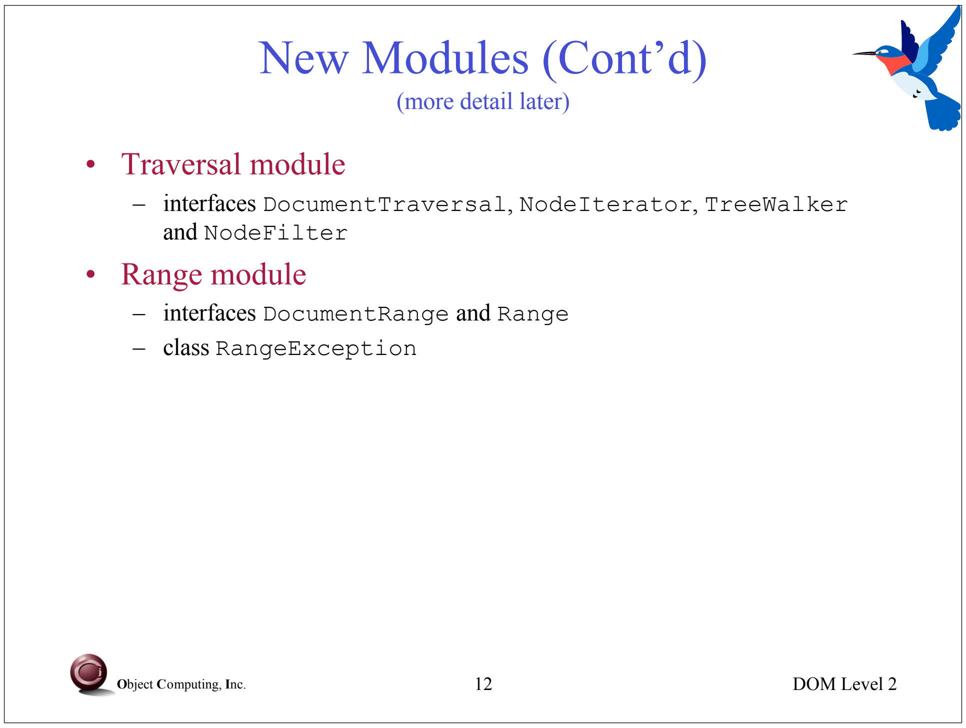


New Modules

(more detail later)

- StyleSheets module
  - interfaces `DocumentStyle`, `StyleSheet`, `MediaList`, `StyleSheetList` and `LinkStyle`
- CSS module
  - interfaces … too many to list!
- Views module
  - interfaces `DocumentView` and `AbstractView`
- Events module
  - interfaces `DocumentEvent`, `Event`, `UIEvent`, `MouseEvent`, `MutationEvent`, `EventListener` and `EventTarget`
  - class `EventException`

Interfaces with names that start with "Document" are implemented by parser-specific classes which also implement the Document interface (for example, org.apache.xerces.dom.DocumentImpl).



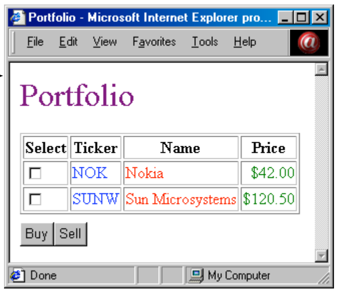
New Modules (Cont'd)

(more detail later)

- Traversal module
  - interfaces `DocumentTraversal`, `NodeIterator`, `TreeWalker` and `NodeFilter`
- Range module
  - interfaces `DocumentRange` and `Range`
  - class `RangeException`

Example Files

- The next several pages contain example files that will be used to demonstrate usage of DOM Level 2 features
- These files maximize the separation between
  - model
    - data as XML
  - view
    - transformation to HTML using XSLT
    - formatting (colors, fonts, sizes, …) using CSS
  - controller
    - processing user input using JavaScript and perhaps a Java Servlet


Portfolio - Microsoft Internet Explorer pro...
File Edit View Favorites Tools Help
Buy
Sell
Done
My Computer

Portfolio

Select	Ticker	Name	Price
☐	NOK	Nokia	$42.00
☐	SUNW	Sun Microsystems	$120.50

Example XML Document

(portfolio.xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl"
  href="portfolio.xsl"?>

<portfolio>
  <owner>R. Mark Volkmann</owner>

  <stock>
    <!-- home of Java -->
    <ticker>SUNW</ticker>
    <name>Sun Microsystems</name>
    <price>120.5</price>
    <quantity>176</quantity>
  </stock>

  <stock>
    <!-- has good WAP dev. env. -->
    <ticker>NOK</ticker>
    <name>Nokia</name>
    <price>42</price>
    <quantity>10</quantity>
  </stock>
```

```
  <mutualFund>
    <!-- great for college savings -->
    <symbol>JAMRX</symbol>
    <name>Janus Mercury Fund</name>
    <price>44.75</price>
    <quantity>671.001</quantity>
  </mutualFund>

  <mutualFund>
    <!-- for retirement -->
    <symbol>BARAX</symbol>
    <name>Baron Asset Fund</name>
    <price>62.86</price>
    <quantity>475.728</quantity>
  </mutualFund>

</portfolio>
```

Example XSL Stylesheet

(portfolio.xsl)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

  <xsl:template match="/">
    <html>
      <head>
        <title>Portfolio</title>
        <link rel="stylesheet" type="text/css" href="portfolio.css"/>
        <script language="JavaScript" src="portfolio.js"/>
      </head>
      <body>
        <xsl:apply-templates select="portfolio"/>
      </body>
    </html>
  </xsl:template>
```

reference to an external CSS stylesheet

reference to an external JavaScript file

Example XSL Stylesheet (Cont'd)

```
  <xsl:template match="portfolio">
    <div class="title">Portfolio</div>
    <form method="post" action="http://www.ociweb.com/PortfolioServlet">
      <table border="1">
        <tr>
          <th>Select</th> <th>Ticker</th> <th>Name</th> <th>Price</th>
        </tr>
        <xsl:apply-templates select="stock">
          <xsl:sort select="ticker"/>
        </xsl:apply-templates>
      </table>
      <div class="button">
        <input name="buyButton" type="submit" value="Buy"
         onclick="buy()"/>
        <input name="sellButton" type="submit" value="Sell"
         onclick="sell()"/>
      </div>
    </form>
  </xsl:template>
```

only processing stocks, not mutualFunds

Example XSL Stylesheet (Cont’d)

```
  <xsl:template match="stock">
    <tr>
      <td><input name="{ticker}" type="checkbox"/></td>
      <td class="ticker"><xsl:value-of select="ticker"/></td>
      <td class="name"><xsl:value-of select="name"/></td>
      <td class="price">
        $<xsl:value-of select="format-number(price, '#.00')"/>
      </td>
    </tr>
  </xsl:template>

</xsl:stylesheet>
```

Example CSS
(portfolio.css)

```
.button {
   margin-top:    1ex;
}
.name {
   color:         red;
}
.price {
   color:         green;
   text-align:    right;
}
.ticker {
   color:         blue;
}
.title {
   color:         purple;
   font-size:     24pt;
}
```

These CSS selectors are class names associated with HTML tags in the XSL. There are many other kinds of CSS selectors.

Example JavaScript
(portfolio.js)

```
function buy() {
  alert("You pushed Buy.");
}

function sell() {
  alert("You pushed Sell.");
}
```

Code Examples

- Code examples are in Java using Xerces 1.2.0
  - implements the Traversal and Range modules
  - implements MutationEvents from the Events module
- Since there are currently no publicly available implementations for several of the optional DOM Level 2 modules, the code is a best guess as to how these modules might be used
  - fictional classes which implement DOM Level 2 interfaces are in *italics* and named *interfaceImpl* where *interface* is the name of the main interface that the class implements
  - fictional methods are also in *italics*

Traversal Module

- Provides interfaces that assist with traversing the node structure of XML documents
  - DOM Level 1 provides all that is needed but this module makes certain traversals easier
- `NodeIterator` interface
  - traverses nodes of a subtree in document order
  - methods: `nextNode()` and `previousNode()`
- `TreeWalker` interface
  - traverses nodes of a subtree based on the hierarchy
  - keeps track of current node; all traversals are relative to this
  - methods
    - `parentNode()`, `firstChild()`, `lastChild()`, `previousSibling()`, `nextSibling()`, `previousNode()` and `nextNode()`

This will not step outside the subtree.

These traverse in document order like `NodeIterator`.

Traversal Module (Cont'd)

- `NodeFilter` interface
  - can be associated with NodeIterators and TreeWalkers to restrict the nodes that are visited
  - must be specified when they are created
  - method
    - `short acceptNode(Node n)`
      - return value is one of these constants
        - » `FILTER_ACCEPT` includes the node
        - » `FILTER_SKIP` excludes the node but not necessarily its descendants
        - » `FILTER_REJECT` excludes the node AND its descendants

Traversal Module (Cont'd)

- `DocumentTraversal` interface

This is implemented by an implementation-specific class that also implements Document.

  - creates NodeIterators and TreeWalkers
  - allows NodeFilters to be associated with them
    - only way to do it
  - methods

```
• NodeIterator createNodeIterator
    (Node root,
     int whatToShow,
     NodeFilter filter,
     boolean expandEntities);
• TreeWalker createTreeWalker
    (Node root,
     int whatToShow,
     NodeFilter filter,
     boolean expandEntities);
```

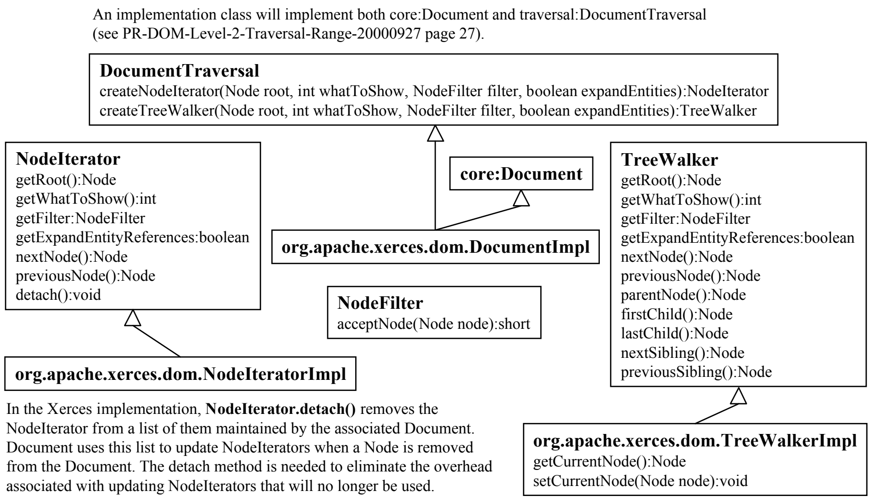
Pass null if not needed.

Set to true if entities contain additional nodes that should also be traversed.

`whatToShow` provides simple filtering based on node type. It must be set to one of these constants.

```
SHOW_ALL
SHOW_ELEMENT
SHOW_ATTRIBUTE
SHOW_TEXT
SHOW_CDATA_SECTION
SHOW_ENTITY_REFERENCE
SHOW_ENTITY
SHOW_PROCESSING_INSTRUCTION
SHOW_COMMENT
SHOW_DOCUMENT
SHOW_DOCUMENT_TYPE
SHOW_DOCUMENT_FRAGMENT
SHOW_NOTATION
```

Traversal Class Diagram



An implementation class will implement both core:Document and traversal:DocumentTraversal (see PR-DOM-Level-2-Traversal-Range-20000927 page 27).

DocumentTraversal
createNodeIterator(Node root, int whatToShow, NodeFilter filter, boolean expandEntities):NodeIterator
createTreeWalker(Node root, int whatToShow, NodeFilter filter, boolean expandEntities):TreeWalker
NodeIterator
getRoot():Node
getWhatToShow():int
getFilter:NodeFilter
getExpandEntityReferences:boolean
nextNode():Node
previousNode():Node
detach():void
core:Document
org.apache.xerces.dom.DocumentImpl
TreeWalker
getRoot():Node
getWhatToShow():int
getFilter:NodeFilter
getExpandEntityReferences:boolean
nextNode():Node
previousNode():Node
parentNode():Node
firstChild():Node
lastChild():Node
nextSibling():Node
previousSibling():Node
NodeFilter
acceptNode(Node node):short
org.apache.xerces.dom.NodeIteratorImpl
org.apache.xerces.dom.TreeWalkerImpl
getCurrentNode():Node
setCurrentNode(Node node):void

In the Xerces implementation, **NodeIterator.detach()** removes the NodeIterator from a list of them maintained by the associated Document. Document uses this list to update NodeIterators when a Node is removed from the Document. The detach method is needed to eliminate the overhead associated with updating NodeIterators that will no longer be used.

NodeIterator Example

```
import java.io.IOException;
import org.apache.xerces.parsers.*;
import org.w3c.dom.*;
import org.w3c.dom.traversal.*;
import org.xml.sax.SAXException;

public class NodeIteratorDemo implements NodeFilter {

  public static void main(String[] args) throws IOException, SAXException {
    new NodeIteratorDemo();
  }
```

To use multiple NodeFilter implementations in a single class, create separate classes that implement it.

NodeIterator Example (Cont'd)

```
private NodeIteratorDemo() throws IOException, SAXException {
  DOMParser parser = new DOMParser();
  parser.parse("portfolio.xml");
  Document doc = parser.getDocument();
  DocumentTraversal dt = (DocumentTraversal) doc;

  NodeIterator iter = dt.createNodeIterator
    (doc.getDocumentElement(), NodeFilter.SHOW_ELEMENT, this, true);

  // Only visits stock elements with a certain price.
  while (true) {
    Node stock = iter.nextNode();
    if (stock == null) break;
    System.out.println(getStockDescription(stock));
  }
}
```

The first parameter is the starting point of the traversal.
Pass null for the NodeFilter to filter only on showWhat.
The last parameter controls whether entity references are expanded.

the NodeFilter

Output

```
Sun Microsystems(SUNW) 176 shares at 120.5
```

NodeIterator Example (Cont'd)

```
/**
 * Accept only Stock elements with a price greater than $50.
 */
public short acceptNode(Node node) {
  if ("stock".equals(node.getNodeName())) {
    String priceText = getChildValue(node, "price");
    float price = new Float(priceText).floatValue();
    if (price > 50) return NodeFilter.FILTER_ACCEPT;
  }
  return NodeFilter.FILTER_REJECT;
}

/**
 * Gets a description of a stock element.
 */
private static String getStockDescription(Node stock) {
  NodeList children = stock.getChildNodes();
  return getChildValue(children, "name") +
         "(" + getChildValue(children, "ticker") + ") " +
         getChildValue(children, "quantity") + " shares " +
         "at " + getChildValue(children, "price");
}
```

not catching NumberFormatException

There's no need to verify that the node is an Element since **showWhat** already insures that only Elements will be passed to this method.

done for efficiency

NodeIterator Example (Cont'd)

These are **convenience methods** that are also used in subsequent code examples.

```
/**
 * Gets the first child element with a given name.
 */
private static Node getChild(Node parent, String name) {
  return getChild(parent.getChildNodes(), name);
}

/**
 * Gets the first node in a NodeList with a given name.
 */
private static Node getChild(NodeList children, String name) {
  for (int i = 0; i < children.getLength(); i++) {
    Node child = children.item(i);
    if (name.equals(child.getNodeName())) return child;
  }
  return null;
}
```

NodeIterator Example (Cont'd)

These are **convenience methods** that are also used in subsequent code examples.

```
/**
 * Gets the text inside the first child element with a given name.
 * This should only be used for child elements that are text-only.
 */
private static String getChildValue(Node parent, String childName) {
  return getChildValue(parent.getChildNodes(), childName);
}

/**
 * Gets the text inside the first node in a NodeList with a given name.
 * Only call this for child elements that are text-only.
 */
private static String getChildValue(NodeList children, String childName) {
  Node child = getChild(children, childName);
  if (child == null) return null;
  Node childText = child.getFirstChild();
  return childText == null ? null : childText.getNodeValue();
}
}
```

assumes that the first child is a text node

TreeWalker Example

```
import java.io.IOException;
import org.apache.xerces.parsers.*;
import org.w3c.dom.*;
import org.w3c.dom.traversal.*;
import org.xml.sax.SAXException;

public class TreeWalkerDemo implements NodeFilter {

  public static void main(String[] args) throws IOException, SAXException {
    new TreeWalkerDemo();
  }

  private TreeWalkerDemo() throws IOException, SAXException {
    DOMParser parser = new DOMParser();
    parser.parse("portfolio.xml");
    Document doc = parser.getDocument();
    DocumentTraversal dt = (DocumentTraversal) doc;

    TreeWalker walker = dt.createTreeWalker
      (doc.getDocumentElement(), NodeFilter.SHOW_ELEMENT, this, true);
```

The first parameter is the starting point of the traversal.
Pass null for the NodeFilter to filter only on showWhat.
The last parameter controls whether entity references are expanded.

To use multiple NodeFilter implementations in a single class, create separate classes that implement it.

the NodeFilter

TreeWalker Example (Cont'd)

```
    // Only visits stock and mutualFund elements with a certain price
    // and their children.
    for (Node security = walker.firstChild();
      security != null;
      security = walker.nextSibling()) {

      // Output child elements.
      for (Node child = walker.firstChild();
           child != null;
           child = walker.nextSibling()) {
        Node text = child.getFirstChild();
        System.out.println(child.getNodeName() + ": " + text.getNodeValue());
      }

      System.out.println(); // separates descriptions

      // Restore the current node of the TreeWalker.
      walker.parentNode();
    }
  }
```

Note that all TreeWalker navigation is relative to the "current" node.

Output

```
ticker: SUNW
name: Sun Microsystems
price: 120.5
quantity: 176

symbol: BARAX
name: Baron Asset Fund
price: 62.86
quantity: 475.728
```

TreeWalker Example (Cont'd)

```
  /**
   * Accept stock and mutualFund elements with price > $50 and their children.
   */
  public short acceptNode(Node node) {
    // If the node is a stock or mutual fund with a certain price then accept it.
    String nodeName = node.getNodeName();
    if ("stock".equals(nodeName) || "mutualFund".equals(nodeName)) {
      String priceText = getChildValue(node, "price");
      float price = new Float(priceText).floatValue();
      if (price > 50) return NodeFilter.FILTER_ACCEPT;
    }

    // If the node is a child of a stock or mutualFund then accept it.
    String parentNodeName = node.getParentNode().getNodeName();
    if ("stock".equals(parentNodeName) || "mutualFund".equals(parentNodeName)) {
      return NodeFilter.FILTER_ACCEPT;
    }

    return NodeFilter.FILTER_REJECT; // reject the node and all its descendants
  }
}
```

Children of stocks and mutual funds with a price less than or equal to $50 won't make it to this point.

The "owner" child element of "portfolio" is rejected.

getChild and getChildValue convenience methods belong here too.

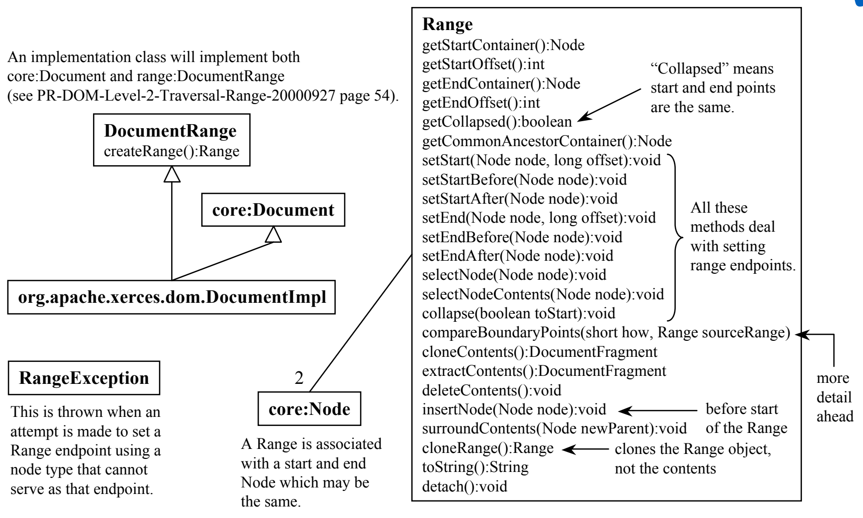
Range Module

- Identifies a range of content between two points
  - in a Document, DocumentFragment or Attr
  - each end point is defined by a node and an offset into the data of that node
    - if the node is subclass of CharacterData (Text, CDATASection or Comment) then the offset is a number of characters into the text
    - otherwise the offset is a number of child nodes
  - ranges do not have to represent well-formed XML
  - could represent a user mouse selection
- Provides operations that
  - act on the nodes in the range
    - cloneContents, extractContents and deleteContents
  - add new nodes relative to the range
    - insertNode inserts a single node before the start
    - surroundContents surrounds all the nodes in the range with a new node

The child nodes of an **Attr** are Text and EntityReference nodes.

Usually it is desirable to operate on ranges that are well-formed but operations on them have well-defined behaviors even when the range is not well-formed. See example ahead.

Range Class Diagram



An implementation class will implement both core:Document and range:DocumentRange (see PR-DOM-Level-2-Traversal-Range-20000927 page 54).
DocumentRange
createRange():Range
core:Document
org.apache.xerces.dom.DocumentImpl
RangeException
This is thrown when an attempt is made to set a Range endpoint using a node type that cannot serve as that endpoint.
2
core:Node
A Range is associated with a start and end Node which may be the same.
Range
getStartContainer():Node
getStartOffset():int
getEndContainer():Node
getEndOffset():int
getCollapsed():boolean
getCommonAncestorContainer():Node
setStart(Node node, long offset):void
setStartBefore(Node node):void
setStartAfter(Node node):void
setEnd(Node node, long offset):void
setEndBefore(Node node):void
setEndAfter(Node node):void
selectNode(Node node):void
selectNodeContents(Node node):void
collapse(boolean toStart):void
compareBoundaryPoints(short how, Range sourceRange)
cloneContents():DocumentFragment
extractContents():DocumentFragment
deleteContents():void
insertNode(Node node):void
surroundContents(Node newParent):void
cloneRange():Range
toString():String
detach():void
"Collapsed" means start and end points are the same.
All these methods deal with setting range endpoints.
more detail ahead
before start of the Range
clones the Range object, not the contents

Creating Ranges

- To create a Range
  - create an uninitialized Range object
    - a class that implements the Document interface will also implement the DocumentRange interface
    - `Range range = ((DocumentRange) document).createRange();`
  - set the range start point
    - `range.setStartBefore(node);` - when an Element, starts before start tag
    - `range.setStartAfter(node);` - when an Element, starts after start tag
    - `range.setStart(node, offset);` - `offset` is a long
  - set the range end point
    - `range.setEndBefore(node);` - when an Element, stops before end tag
    - `range.setEndAfter(node);` - when an Element, stops after end tag
    - `range.setEnd(node, offset);` - `offset` is a long
  - can set both endpoints based on a single node
    - `range.selectNode(node);` - when an Element, selects start and end tags
    - `range.selectNodeContents(node);` - when an Element, omits start and end tags

Range Operations

- `DocumentFragment fragment = range.cloneContents();`
  - copies the range content
- `range.deleteContents();`
  - deletes the range content; the range is collapsed after this
- `DocumentFragment fragment = range.extractContents();`
  - also deletes but can be reinserted since a handle to the content is returned
- `range.insertNode(newNode);`
  - inserts a new node before the start of the range; the range does not change
- `range.surroundContents(newNode);`
  - extracts content of range, inserts new node where range content was, adds range content to new node, and selects newNode (expanding the range)

Operations on non-well-formed Ranges have a well-formed result. For example:

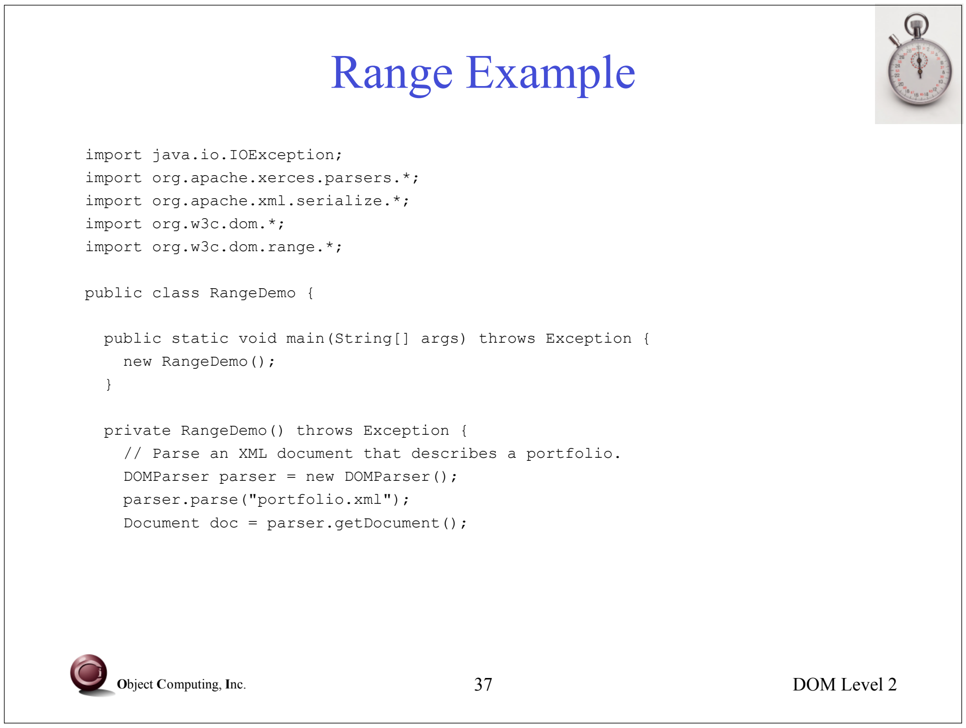
Source document with range in bold	**Result after calling deleteContents on the range**

`<pets>`
  `<a>Win`**`nie</a>`**
  `<b>Chester</b>`
  `<c>Pab``lo</c>`
`</pets>`
	`<pets>`
	  `<a>Win</a>`
	  `<c>lo</c>`
	`</pets>`

Comparing Range Endpoints

- Can compare ranges
  - constants specify the type of comparison to be performed
    - `START_TO_START`
    - `START_TO_END`
    - `END_TO_START`
    - `END_TO_END`
  - return value indicates
    - before (-1)
    - equal (0)
    - after (1)
  - example

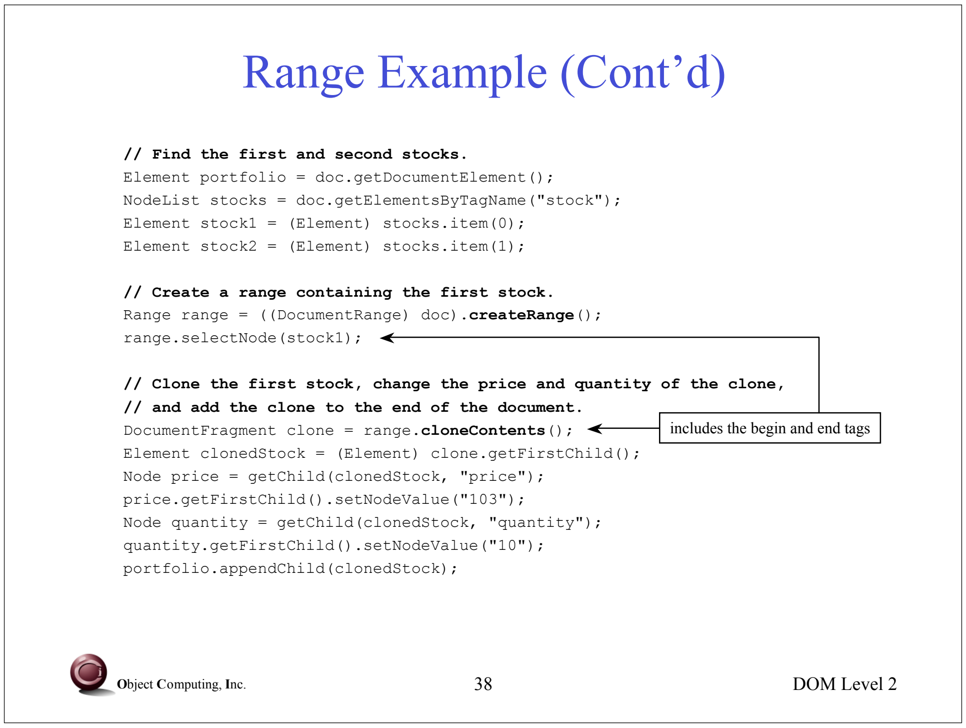
```
int comparison = range1.compareBoundaryPoints
    (Range.START_TO_START, range2);
```



Range Example

```
import java.io.IOException;
import org.apache.xerces.parsers.*;
import org.apache.xml.serialize.*;
import org.w3c.dom.*;
import org.w3c.dom.range.*;

public class RangeDemo {

  public static void main(String[] args) throws Exception {
    new RangeDemo();
  }

  private RangeDemo() throws Exception {
    // Parse an XML document that describes a portfolio.
    DOMParser parser = new DOMParser();
    parser.parse("portfolio.xml");
    Document doc = parser.getDocument();
```

Object Computing, Inc. 37 DOM Level 2

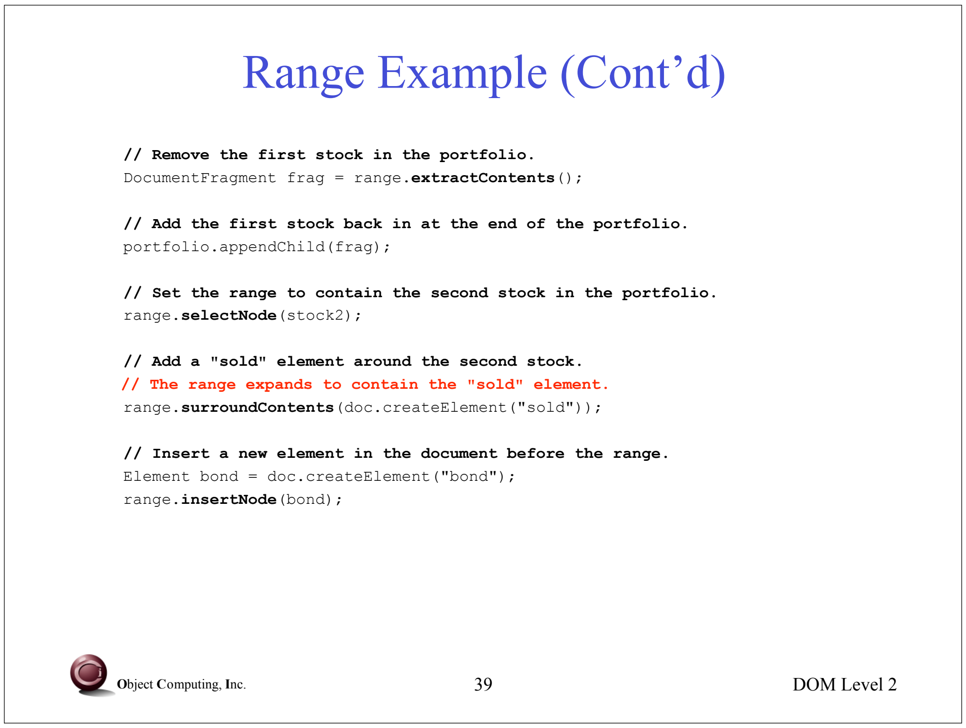


Range Example (Cont'd)

```
    // Find the first and second stocks.
    Element portfolio = doc.getDocumentElement();
    NodeList stocks = doc.getElementsByTagName("stock");
    Element stock1 = (Element) stocks.item(0);
    Element stock2 = (Element) stocks.item(1);

    // Create a range containing the first stock.
    Range range = ((DocumentRange) doc).createRange();
    range.selectNode(stock1);

    // Clone the first stock, change the price and quantity of the clone,
    // and add the clone to the end of the document.
    DocumentFragment clone = range.cloneContents();
    Element clonedStock = (Element) clone.getFirstChild();
    Node price = getChild(clonedStock, "price");
    price.getFirstChild().setNodeValue("103");
    Node quantity = getChild(clonedStock, "quantity");
    quantity.getFirstChild().setNodeValue("10");
    portfolio.appendChild(clonedStock);
```

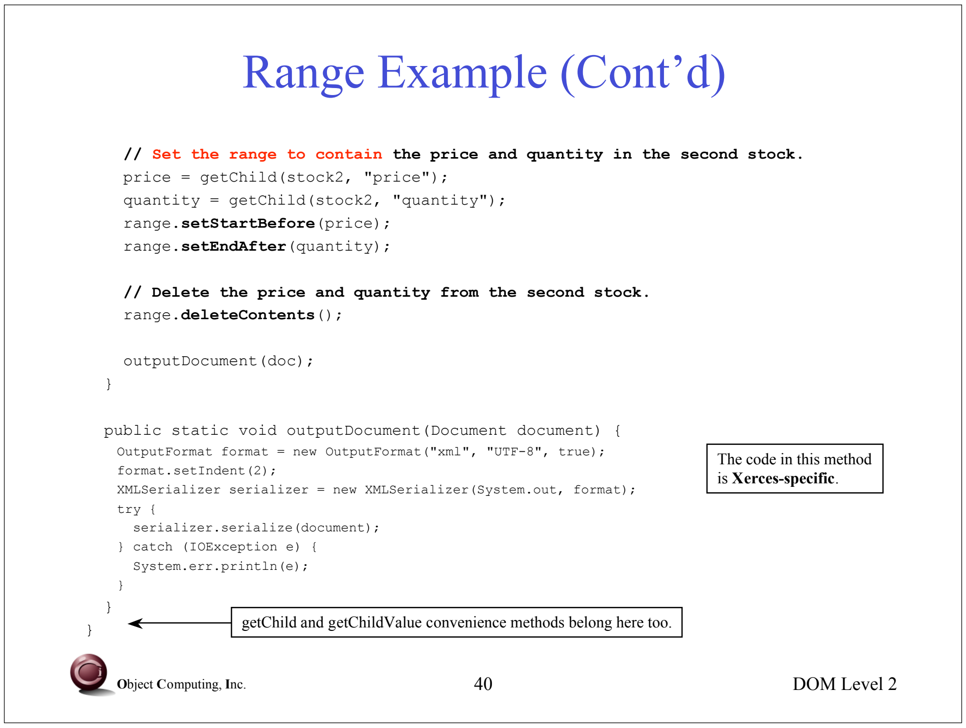
includes the begin and end tags

Object Computing, Inc. 38 DOM Level 2



Range Example (Cont'd)

```
// Remove the first stock in the portfolio.
DocumentFragment frag = range.extractContents();

// Add the first stock back in at the end of the portfolio.
portfolio.appendChild(frag);

// Set the range to contain the second stock in the portfolio.
range.selectNode(stock2);

// Add a "sold" element around the second stock.
// The range expands to contain the "sold" element.
range.surroundContents(doc.createElement("sold"));

// Insert a new element in the document before the range.
Element bond = doc.createElement("bond");
range.insertNode(bond);
```



Range Example (Cont'd)

```
    // Set the range to contain the price and quantity in the second stock.
    price = getChild(stock2, "price");
    quantity = getChild(stock2, "quantity");
    range.setStartBefore(price);
    range.setEndAfter(quantity);

    // Delete the price and quantity from the second stock.
    range.deleteContents();

    outputDocument(doc);
  }

  public static void outputDocument(Document document) {
    OutputFormat format = new OutputFormat("xml", "UTF-8", true);
    format.setIndent(2);
    XMLSerializer serializer = new XMLSerializer(System.out, format);
    try {
      serializer.serialize(document);
    } catch (IOException e) {
      System.err.println(e);
    }
  }
}
```

The code in this method is **Xerces-specific**.

getChild and getChildValue convenience methods belong here too.

Range Example Output

```
<?xml version="1.0" encoding="UTF-8"?>
<portfolio>
  <owner>R. Mark Volkmann</owner>
  <bond/>
  <sold>
    <stock>
      <!-- has good WAP dev. env. -->
      <ticker>NOK</ticker>
      <name>Nokia</name>
    </stock>
  </sold>
  <mutualFund>
    <!-- great for college savings -->
    <symbol>JAMRX</symbol>
    <name>Janus Mercury Fund</name>
    <price>44.75</price>
    <quantity>671.001</quantity>
  </mutualFund>
  <mutualFund>
    <!-- for retirement -->
    <symbol>BARAX</symbol>
    <name>Baron Asset Fund</name>
    <price>62.86</price>
    <quantity>475.728</quantity>
  </mutualFund>
  <stock>
    <!-- home of Java -->
    <ticker>SUNW</ticker>
    <name>Sun Microsystems</name>
    <price>103</price>
    <quantity>10</quantity>
  </stock>
  <stock>
    <!-- home of Java -->
    <ticker>SUNW</ticker>
    <name>Sun Microsystems</name>
    <price>120.5</price>
    <quantity>176</quantity>
  </stock>
</portfolio>
```

added before 2nd stock

surrounded 2nd stock

removed price and quantity

copied 1st stock, modified price and quantity, and appended it here

moved 1st stock here

StyleSheets Module

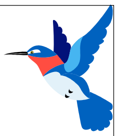
- For representing any kind of stylesheet
  - such as
    - CSS - supported now
    - XSL - not specifically supported yet but can be represented with core objects
  - specific stylesheet languages can be supported by additional interfaces that derive from those in this module
- Supports
  - associating stylesheets with documents and disassociating them
    - using the LinkStyle interface
      - for CSS this is done by adding a “style” (internal) or “link” (external) tag to an HTML document or removing one
        - » both implementation classes will also implement LinkStyle ← see p. 44-45
      - for XSL this is done by adding a “xml-stylesheet” processing instruction to an XML document or removing one
        - » the implementation class will also implement LinkStyle
  - stylesheets that include other stylesheets
    - see parentStyleSheet attribute (see p. 44)

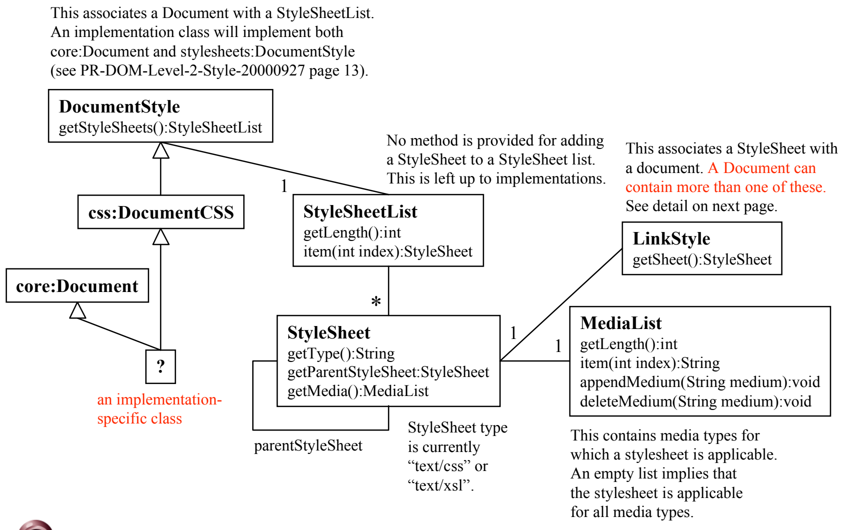
StyleSheets Module (Cont’d)

- Does not support
  - creation and modification of stylesheets
    - provided by other modules that are based on this one
    - the CSS module provides this for CSS stylesheets
  - applying stylesheets to documents to create a View
    - this is currently implementation specific
    - not a definite requirement of DOM Level 3 either
      - from the Level 3 requirements document (WD-DOM-Requirements-20000412) “The API **may** permit creation of new views of the document.”

StyleSheets Class Diagram

(some methods are not shown)





This associates a Document with a StyleSheetList. An implementation class will implement both core:Document and stylesheets:DocumentStyle (see PR-DOM-Level-2-Style-20000927 page 13).

No method is provided for adding a StyleSheet to a StyleSheet list. This is left up to implementations.

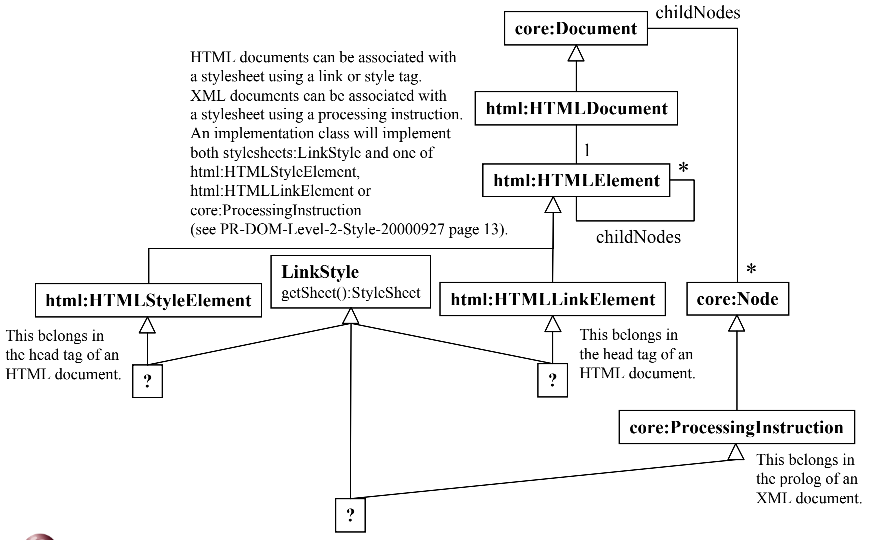
This associates a StyleSheet with a document. A Document can contain more than one of these. See detail on next page.

StyleSheet type is currently “text/css” or “text/xsl”.

This contains media types for which a stylesheet is applicable. An empty list implies that the stylesheet is applicable for all media types.

StyleSheets Class Diagram (Cont'd)

(most methods are not shown)



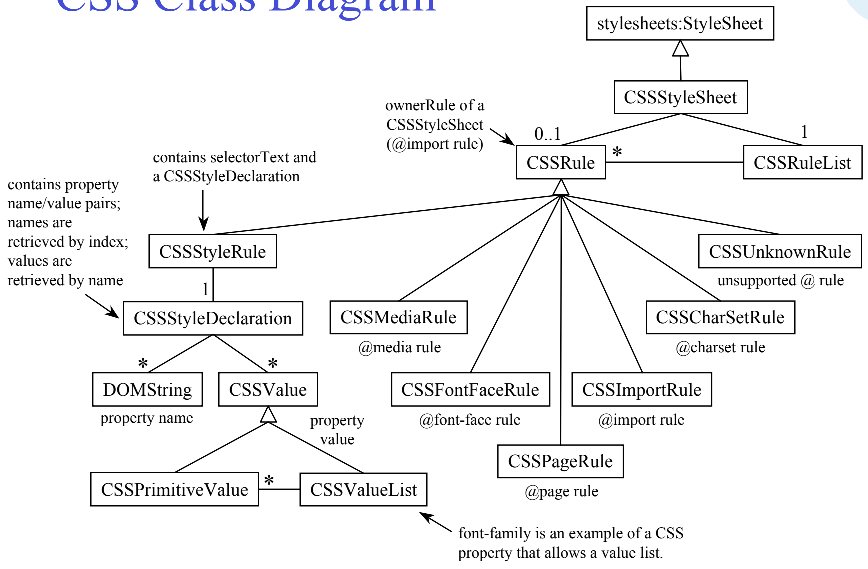
core:Document
childNodes
HTML documents can be associated with a stylesheet using a link or style tag. XML documents can be associated with a stylesheet using a processing instruction. An implementation class will implement both stylesheets:LinkStyle and one of html:HTMLStyleElement, html:HTMLLinkElement or core:ProcessingInstruction (see PR-DOM-Level-2-Style-20000927 page 13).
html:HTMLDocument
1
html:HTMLElement
*
childNodes
LinkStyle
getSheet():StyleSheet
html:HTMLStyleElement
html:HTMLLinkElement
core:Node
*
This belongs in the head tag of an HTML document.
?
?
This belongs in the head tag of an HTML document.
core:ProcessingInstruction
This belongs in the prolog of an XML document.
?

CSS Module

- Builds on the StyleSheets module
- Provides ability to
  - determine which cascading stylesheets are associated with a document
    - see DocumentCSS which implements stylesheets:DocumentStyle on p. 49
  - create a DOM representation of an existing cascading stylesheet
  - modify existing cascading stylesheets
  - create new cascading stylesheets
    - see DOMImplementationCSS

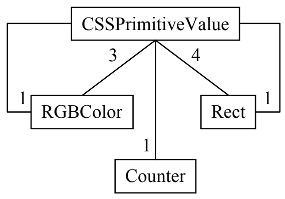
  } see CSSStyleSheet, which implements stylesheet:StyleSheet, and its descendants on the next page
- Does not provide ability to
  - apply a CSS stylesheet to XML or HTML documents
- The StyleSheets module provides ability to associate CSS stylesheets with a document
  - see stylesheets:LinkStyle which is implemented by html:HTMLStyleElement and html:HTMLLinkElement on the previous page

CSS Class Diagram

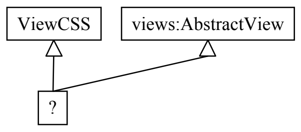


CSS Class Diagram (Cont'd)

A CSSPrimitiveValue can contain a reference to a RGBColor, Rect or Counter. RGBColor and Rect contain multiple references to CSSPrimitiveValues that specify amounts of red, green and blue or top/right/bottom/left coordinates.



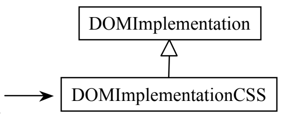
An implementation class will implement both css:ViewCSS and views:AbstractView (see PR-DOM-Level-2-Style-20000927 page 38).



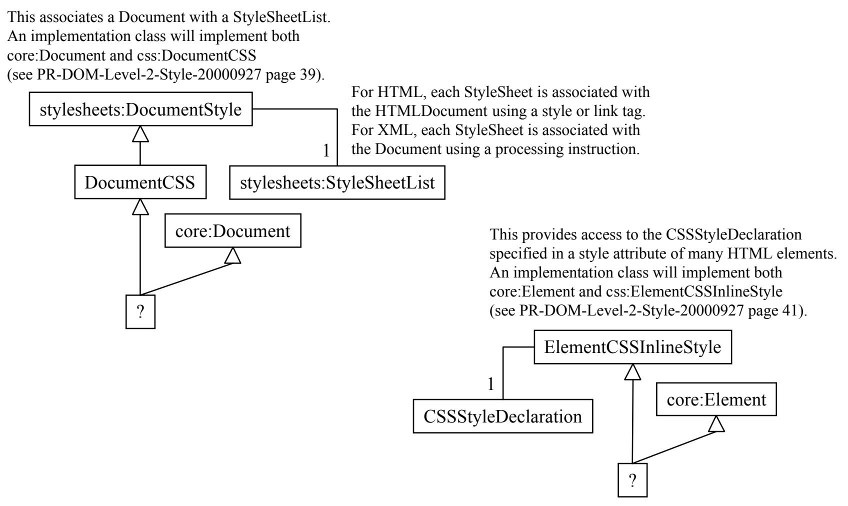
CSS2Properties

This provides a convenient way to get and set all the properties in a CSSStyleDeclaration.

This provides a method to create a CSSStyleSheet but not a method for associating it with a Document.



CSS Class Diagram (Cont'd)


This associates a Document with a StyleSheetList.
An implementation class will implement both
core:Document and css:DocumentCSS
(see PR-DOM-Level-2-Style-20000927 page 39).
stylesheets:DocumentStyle
1
DocumentCSS
stylesheets:StyleSheetList
For HTML, each StyleSheet is associated with
the HTMLDocument using a style or link tag.
For XML, each StyleSheet is associated with
the Document using a processing instruction.
core:Document
?
This provides access to the CSSStyleDeclaration
specified in a style attribute of many HTML elements.
An implementation class will implement both
core:Element and css:ElementCSSInlineStyle
(see PR-DOM-Level-2-Style-20000927 page 41).
ElementCSSInlineStyle
1
CSSStyleDeclaration
core:Element
?

Creating a CSS StyleSheet

```
// Create a new, empty CSS stylesheet.
DOMImplementationCSS domImpl = new DOMImplementationCSSImpl();
String media = "screen";
CSSStyleSheet styleSheet =
  domImpl.createCSSStyleSheet("stylesheet title", media);

// Add a rule in a way that requires string parsing.
styleSheet.insertRule
  (0, "td.price {color: red; text-align: right}");
```

This is the appropriate media type for web browsers on a standard PC. See REC-CSS2-19980512 section 7.3 for a list of recognized media types.

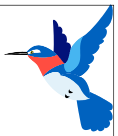
This is just a short description of the stylesheet, not currently used for anything.

New rules can't just be appended at the end. Their position in the list must be specified.

It seems that an approach that doesn't require string parsing should be supported too. If it was, it might look like this. Things in italics are not defined by DOM Level 2.

```
CSSStyleDeclaration styleDecl = new CSSStyleDeclarationImpl();
String priority = null; // can set to "important"
styleDecl.setProperty("color", "red", priority);
styleDecl.setProperty("text-align", "right", priority);
CSSStyleRule rule = new CSSStyleRuleImpl("td#price", styleDecl);
CSSRuleList ruleList = styleSheet.getRuleList();
ruleList.add(rule);
```

Associating a CSS StyleSheet With An XML Document



```
// Create an empty XML document. Could also parse an existing XML document.
DOMImplementation domImpl = new DOMImplementationImpl();
String namespaceURI = "http://www.ociweb.com/portfolio";
String rootElementName = "portfolio";
DocumentType docType = null;
Document doc =
  domImpl.createDocument(namespaceURI, rootElementName, docType);

// Create a processing instruction which associates a CSS text file
// with the XML document.
ProcessingInstruction pi = doc.createProcessingInstruction
  ("xml-stylesheet", "type=\"text/css\" href=\"portfolio.css\"");
doc.appendChild(pi);
```

A DTD is not used in this example. Also set this to null when using XML Schema instead of DTD.

It seems that an approach that adds a stylesheet directly to the list of stylesheets maintained by the document should be supported too. If it was, it might look like this. The `add` method is not defined by DOM Level 2.

```
StyleSheetList sheetList = doc.getStyleSheetList();
sheetList.add(styleSheet); // adds stylesheet created on previous page
```

Views Module

- A view is an alternate representation of a document
  - for example, the result of applying an XSL stylesheet to an XML document might be an HTML document which would be a view of the XML document
- A `Document` subclass can implement
  - `DocumentView` to associate a source Document with a default view Document
  - `AbstractView` to associate a view Document with a source Document
    - it seems likely that a DOM implementation will contain a class which implements both Document and AbstractView
      - DOM Level 2 doesn’t define this

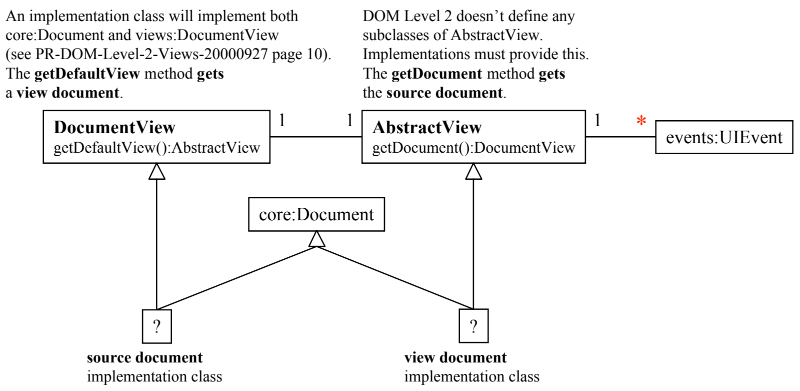
A Work In Progress

- From PR-DOM-Level-2-Views-20000927
  - "The only semantics of the AbstractView defined here create an association between a view and its target document."
  - "There are no subinterfaces of AbstractView defined in the DOM Level 2."
- From Joe Kesselman, IBM Research, DOM WG member
  - "DOM Level 2 Views is a placeholder for the concept of views, included only because we needed to design a "hook" for a view into one of the other modules to allow this future expansion."
  - "DOM Level 3 is expected to describe Views in greater detail."

UIEvents and Views

- A `UIEvent` occurs on a view
  - it is associated with an AbstractView which is associated with a source document
- Types of `UIEvents` include
  - receiving focus (`DOMFocusIn`)
  - losing focus (`DOMFocusOut`)
  - mouse clicks or key presses (`DOMActivate`)
- When a `UIEvent` occurs
  - the view on which it occurred can be obtained from the `UIEvent`
  - the source document can be obtained from the view
  - in the future, might be able to apply a different stylesheet to the source document to generate and display a different view in response to a UIEvent

Views Class Diagram


An implementation class will implement both core:Document and views:DocumentView (see PR-DOM-Level-2-Views-20000927 page 10). The **getDefaultView** method **gets** a **view document**.
DOM Level 2 doesn't define any subclasses of AbstractView. Implementations must provide this. The **getDocument** method **gets** the **source document**.
DocumentView
getDefaultView():AbstractView
1
1
AbstractView
getDocument():DocumentView
1
*
events:UIEvent
core:Document
?
?
source document
implementation class
view document
implementation class

Events Module

- Useful in applications that support
  - user interaction on rendered documents
    - could occur in a web browser
  - modification of documents
    - could occur in an XML editor
- Goals
  - define event handling that includes
    - event descriptions
    - listener registration
    - event delivery through the document tree structure
  - support user interface control events and document mutation events
  - provide some of the event functionality present in current browser scripting languages such as JavaScript

Event Types

- These types of events are supported
  - UI device events
    - caused by user interaction via mouse, keyboard, etc.
    - key events: down, up and press
      - not defined in DOM Level 2 but is in DOM Level 3
      - see PR-DOM-Level-2-20000927 section 1.6.3 and WD-DOM-Level-3-Events-20000901
    - mouse events: down, up, click, move, over and out
      - stores mouse location, state of mouse buttons and ctrl, alt, shift and meta keys
  - UI logical events
    - not specific to a particular kind of input device
    - events: focus in, focus out and activate ← These can happen as a result of a key press (for example, the tab key) or mouse click.
  - mutation events
    - caused by document modifications
    - events indicate node inserted, node removed, attribute modified, character data modified and subtree modified (encompasses all the previous mutation events)

"subtree modified" events are delivered AFTER more specific events.

Event Processing

- Steps
  - EventListeners register with EventTargets ← Implementations of `Node` subclasses implement `EventTarget`.
  - events are delivered to EventTargets
  - EventTargets forward events to their registered EventListeners
  - EventListeners receive events and act on them
- Types of event delivery
  - capturing
    - event is delivered to the Document node first and then propagated downward to containing nodes of the **target node** until the target node is reached
    - all events can be captured

      This is the node on which the event actually occurred.
  - bubbling
    - after event is delivered to the target, it is propagated upward to containing nodes of the target node until the Document node is reached
      - opposite of capturing
    - some types of events do not bubble

Event Delivery Phases

- There are four phases in event delivery
  - phase 1 - Capture
    - the event is delivered to all ancestors of the event target starting from the root of the tree (the Document) and proceeding to the event target
      - if an ancestor has listeners registered to **use capture** for the type of the current event then they are notified
  - phase 2 - At Target
    - the event is delivered to the event target
      - if the target has listeners registered for the type of the current event then they are notified
  - phase 3 - Bubbling
    - for events which bubble (not all do), the event is delivered to all ancestors of the event target again starting from the parent of the event target and proceeding to the root of the tree (the Document)
      - if an ancestor has listeners registered to **not use capture** (bubbling) for the type of the current event then they are notified
  - phase 4 - Default Action
    - more on this later

Event Listeners

- When adding event listeners,
  a specific event type must be specified
  - for mutation events, the event type DOMSubtreeModified can be used to request notification of all document changes at or below a given node
- Must specify phase of event delivery (capture or bubbling) when registering to listen to events
  - can register for both capture AND bubbling
    - some event types do not support bubbling
- Use this method in the EventTarget interface

```
void addEventListener(String eventType,
                      EventListener listener,
                      boolean useCapture);
```

Event Listeners (Cont'd)

- Default actions
  - events can have a default action
    - for example, traversing a hyperlink when it is clicked
  - some default actions are cancelable
    - none of the currently defined MutationEvents are cancelable
  - during any phase of event delivery, any listener can prevent the default action of a cancelable event from occurring

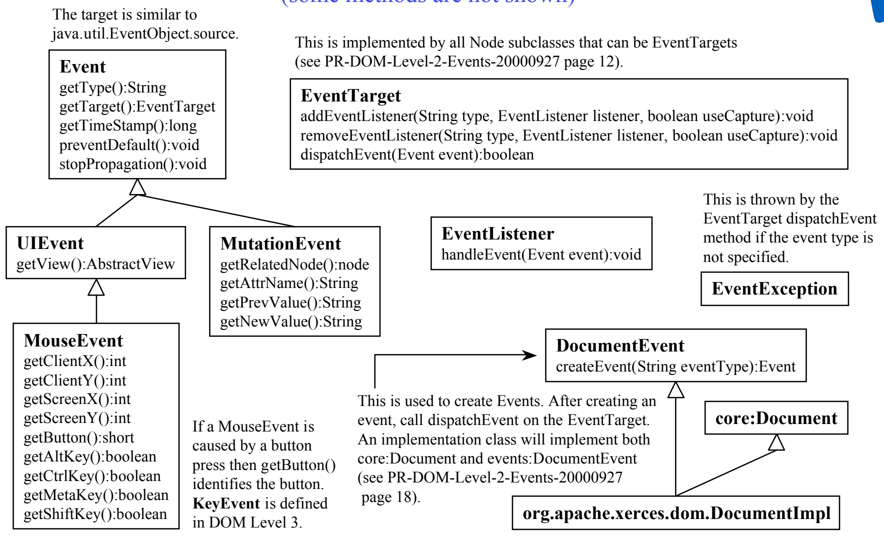
    ```
    event.preventDefault();
    ```
  - this does not prevent further event propagation
- Event propagation
  - during any phase of event delivery, any listener can stop further event propagation from occurring

    ```
    event.stopPropagation();
    ```
  - this does not prevent the default action from occurring

may want to do both

Events Class Diagram

(some methods are not shown)



The target is similar to java.util.EventObject.source.

This is implemented by all Node subclasses that can be EventTargets (see PR-DOM-Level-2-Events-20000927 page 12).

This is thrown by the EventTarget dispatchEvent method if the event type is not specified.

If a MouseEvent is caused by a button press then getButton() identifies the button. **KeyEvent** is defined in DOM Level 3.

This is used to create Events. After creating an event, call dispatchEvent on the EventTarget. An implementation class will implement both core:Document and events:DocumentEvent (see PR-DOM-Level-2-Events-20000927 page 18).

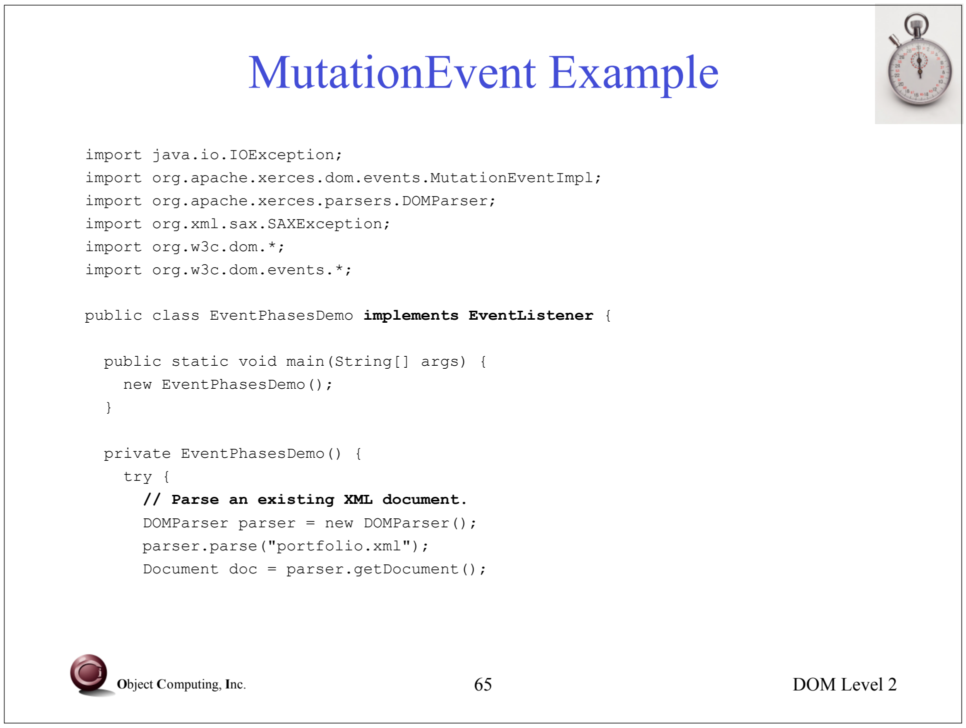
UIEvent Example

```
public class UIEventListener implements EventListener {
  private HTMLButtonElement buyButton;

  public UIEventListener(HTMLViewImpl htmlView) {
    // Find a particular HTML button in the HTML view.
    // Assumuptions:
    // The HTML button has an ID attribute.
    // HTMLViewImpl is an implementation class that
    // implements both AbstractView and HTMLDocument .
    buyButton =
      (HTMLButtonElement) htmlView.getElementById("buyButton");

    // If the button was found, register to be notified
    // when it is pressed.
    if (buyButton != null) {
      buyButton.addEventListener("DOMActivate", this, false);
    }
  }
```

using bubbling, not capture

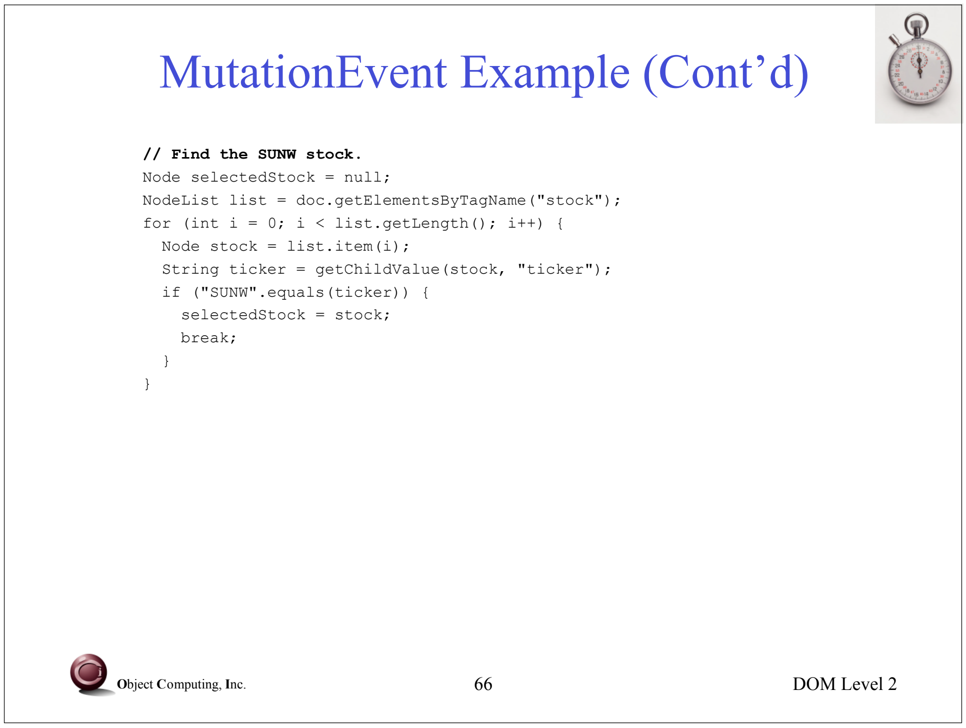
UIEvent Example (Cont'd)

```
  // This is called when an EventTarget this object registered with
  // receives an event.
  public void handleEvent(Event event) {
    // Get the node that generated the event.
    EventTarget target = event.getTarget();

    if (target == buyButton &&
        event.getType().equals("DOMActivate")) {
      // Buy the selected stocks.
    }
  }
}
```



MutationEvent Example

```
import java.io.IOException;
import org.apache.xerces.dom.events.MutationEventImpl;
import org.apache.xerces.parsers.DOMParser;
import org.xml.sax.SAXException;
import org.w3c.dom.*;
import org.w3c.dom.events.*;

public class EventPhasesDemo implements EventListener {

  public static void main(String[] args) {
    new EventPhasesDemo();
  }

  private EventPhasesDemo() {
    try {
      // Parse an existing XML document.
      DOMParser parser = new DOMParser();
      parser.parse("portfolio.xml");
      Document doc = parser.getDocument();
```



MutationEvent Example (Cont'd)

```
      // Find the SUNW stock.
      Node selectedStock = null;
      NodeList list = doc.getElementsByTagName("stock");
      for (int i = 0; i < list.getLength(); i++) {
        Node stock = list.item(i);
        String ticker = getChildValue(stock, "ticker");
        if ("SUNW".equals(ticker)) {
          selectedStock = stock;
          break;
        }
      }
```

MutationEvent Example (Cont’d)

```
    if (selectedStock != null) {
      // Find the text node within the "price" child of the stock.
      Node priceText = getChild(selectedStock, "price").getFirstChild();

      // Add event listeners for capture and bubbling at several levels.
      String eventType = MutationEventImpl.DOM_CHARACTER_DATA_MODIFIED;
      ((EventTarget) priceText).addEventListener(eventType, this, false);
      ((EventTarget) selectedStock).addEventListener(eventType, this, true);
      ((EventTarget) selectedStock).addEventListener(eventType, this, false);
      ((EventTarget) doc).addEventListener(eventType, this, true);
      ((EventTarget) doc).addEventListener(eventType, this, false);

      // Change the price value to test the event listeners.
      priceText.setNodeValue("150");
    }
  } catch (IOException e) {
    System.err.println(e);
  } catch (SAXException e) {
    System.err.println(e);
  }
}
```

MutationEventImpl is a Xerces class.

Pass false (bubbling) for notification in “At Target” phase.

MutationEvent Example (Cont’d)

```
  public void handleEvent(Event e) {
    System.out.println
      ("received " + e.getType() + " event " +
       "on " + e.getCurrentNode().getNodeName());
    System.out.println
      ("during phase " + e.getEventPhase() + " for target " + e.getTarget());
    System.out.println();

    //e.stopPropagation();
  }
}
```

could stop the event from being propagated to other listeners

getChild and getChildValue convenience methods belong here too.

MutationEvent Example Output

phase 1 is capture
phase 2 is at target
phase 3 is bubbling

```
received DOMCharacterDataModified event on #document
during phase 1 for target [#text: 150]

received DOMCharacterDataModified event on stock
during phase 1 for target [#text: 150]

received DOMCharacterDataModified event on #text
during phase 2 for target [#text: 150]

received DOMCharacterDataModified event on stock
during phase 3 for target [#text: 150]

received DOMCharacterDataModified event on #document
during phase 3 for target [#text: 150]
```

Level 3 Already?

- Even though there are few Level 2 implementations and no publicly available complete implementations, DOM Level 3 work is underway
- Level 3 requirements have been documented
  - WD-DOM-Requirements-20000412
  - still a working draft because the requirements have not been finalized
- Working drafts for some Level 3 modules are available
  - WD-DOM-Level-3-Core-20000901
  - WD-DOM-Level-3-Events-20000901
  - WD-DOM-Level-3-Content-Models-and-Load-Save-20001101
  - these are summarized on the following pages

Level-3-Core

- Adds new members to some of the Level 2 Core module interfaces
  - Entity
    - adds attributes and accessors for encoding (declared), actualEncoding and version (of an external parsed entity)
  - Document
    - adds attributes and accessors for encoding (declared), actualEncoding, version (of the document), standAlone and strictErrorChecking
    - adds these new methods
      - `Node **adoptNode**(Node node)` changes the ownerDocument of a Node, its attributes and all its descendants
      - `NodeList **getElementsByAttributeValue**`
        ```
        (String attrNamespaceURI,
         String attrLocalName,
         String attrValue)
        ```
        finds all Elements in a Document, regardless of name, with a given attribute value

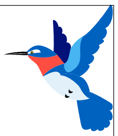
When error checking is not strict an implementation can choose not to test for certain types of errors.

`adoptNode` differs from `importNode` in that it does not make a copy of the node.

alternative to getElementsByTagName

continued on next page

Level-3-Core (Cont’d)



- Node
  - adds methods to
    - determine the relationship between two Nodes in terms of document order and tree hierarchy
    - retrieve the content of a Node and its descendants in one string
    - determine whether two Node references refer to the same node
      - » same functionality as the Java == operator
    - make a Node and its descendants “namespace well-formed”
      - » means that namespaces are properly declared and prefixes are correctly applied
    - attach user data to a Node and retrieve it
    - get the namespace URI of a Node
- Text
  - adds one new method
    - `boolean getIsWhitespaceInElementContent();` determines whether text content includes consecutive whitespace characters (often referred to as “ignorable” whitespace)

Level-3-Events

- Adds new interfaces to the Level 2 Event module
- Defines KeyEvent
  - a sub-interface of UIEvent
  - counterpart to MouseEvent defined in Level 2
  - similar to java.awt.event.KeyEvent
- Adds ability to group EventListeners
  - can register individual EventListeners or groups of EventListeners with an EventTarget
  - event propagation can be stopped for a specific group instead of stopping it from reaching all registered listeners

Level-3-Content-Models-and-Load-Save

- Content-Models (CMs)
  - DTDs and XML Schemas are examples of CMs
  - provides a data structure representation for CMs
  - can associate a CM with a document
  - can modify CMs
  - multiple documents that are associated with the same CM can be processed without having to load the CM multiple times
  - can create new CMs and save them to files
  - can validate a CM
  - can validate a document or a portion of a document against a CM
- Load-Save
  - provides a standard API for loading XML from and saving XML to streams, files and URIs
  - when loading XML, the associated CM can optionally be loaded or a CM that has already been loaded can be reused
  - supports implementation of catalogs for lookup of documents, including CMs, by public ID

same thing specified in two sections

From WD-DOM-Level-3-Content-Models-and-Load-Save-20000901 , "… it is anticipated that **lowest common denominator** general APIs generated in this chapter can support **both DTDs and XML Schemas**, and other XML content models down the road." Specific support for XML Schema will be added in DOM Level 3 or later.

Thank You For Attending!

- I'd love to meet with you throughout the week to discuss
  - **DOM**
    - really I enjoy talking about anything related to XML or Java!
  - **Consulting** services available from OCI
    - we specialize in XML, OOAD, Java, C++, CORBA and EJB
  - **Training** available from OCI
    - we have a wide variety of courses on all of our consulting specialties
    - includes two courses on XML
      - **eXtensible Markup Language**
        - » **topics** include XML syntax, CSS, XSLT, DTD, Namespaces, XML Schema, XHTML, SVG, WML and more
      - **XML Programming Using Java**
        - » **topics** include SAX, SAX2, DOM, DOM Level 2, JAXP, JDOM, XML and databases, XML and servlets, and SOAP
  - **Career** opportunities with OCI
- Check out our web site for more info. - ociweb.com