Programming XML

XML Excursion 2001

Introduction

Who am I?

- Director of Product Technology, SoftQuad Software Inc.
- Co-Designer XMetaL
- Chair, W3C DOM WG
- Chair, OASIS Entity Resolution TC

Topics Covered

- Basic concepts
- Definition of the DOM
- Fundamental Types
- XML Types
- HTML DOM
- CSS OM
- Events
- Traversal
- Range
- Current Work
- Demos and Scenarios

History

Form Validation

- Early use of DHTML
- Check the contents of elements
- Check the value of attributes
- Client-side processing finds syntax errors
- Lessens server overload
- Simple
- Effective

Sorting

- Intranet has phone book
- Lots of entries in some order
- Want to sort them by first name, or last name, or department, ...
- Show the sorted results - without going back to the server

Influences on the DOM Design

- DHTML
- SGML tools
- Object-oriented design
- Many programming languages and platforms
- New usages (e.g., XML for data interchange, e-commerce)

Basic Concepts

What is XML?

- Label information
- Almost self-describing data
- Internationalized
- Well-formed
- Strict error handling
- Can be documents or data streams

Dealing with XML

Tool doesn't understand XML

XML is stream of characters with angle brackets

Tool displays XML (simplistic)

Start tag turns on formatting, end tag turns off formatting

Tool does more with XML

Objects in XML document can be manipulated, accessed, changed

Find particular information in XML data stream

Tools that do something with XML all have their own "DOM" inside

What is an API?

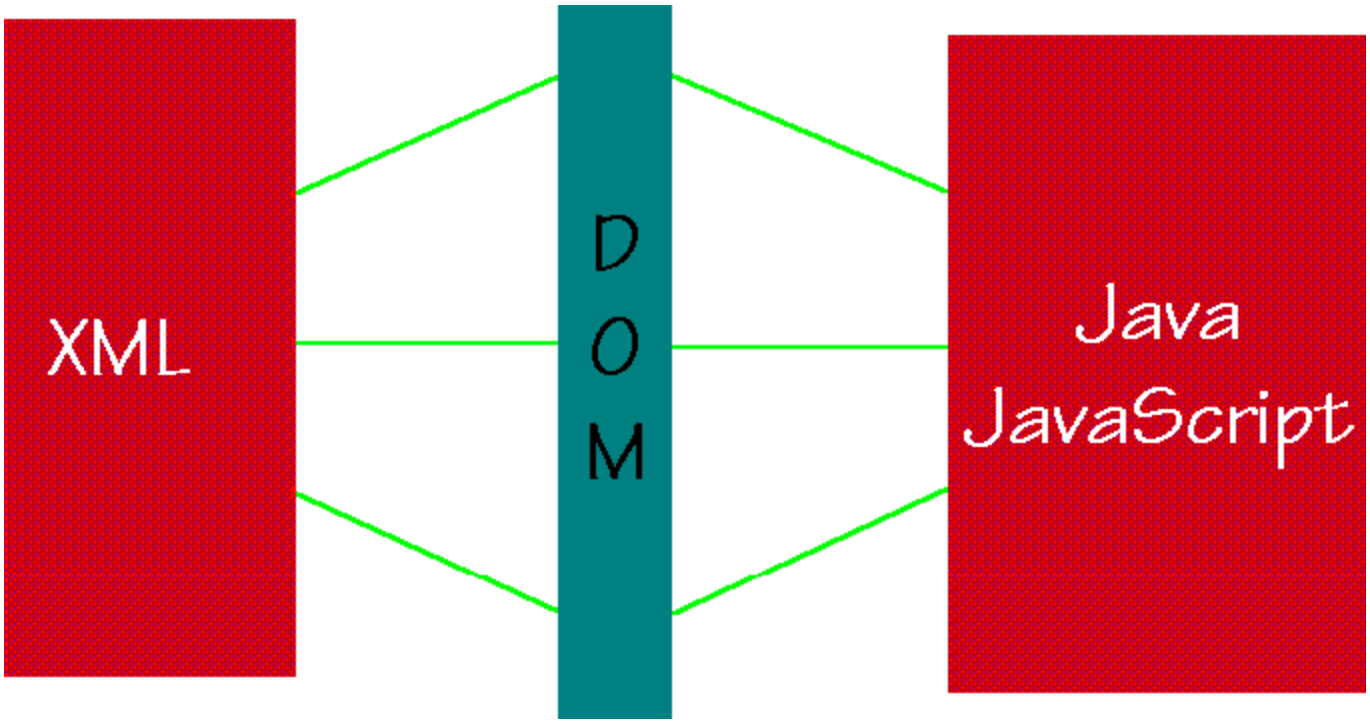
- Lets one program talk to another
- Lets part of one program talk to another part

- Lets part of one program talk to another part
- Hides internal implementation details
- Different programs from different companies can use the same API

Object Models and Data Models

- The DOM is an object model
- Implicit, not explicit, data model
- Implementation details are hidden
- What matters are the interfaces

Where the DOM fits in



Reasons to use the W3C DOM

Obvious Reasons

- Ease of integration
- Interchangable parts
- Ease of learning

Other Reasons

- Consensus view
- Experienced designers
- Saves design time
- Works with other standards

Other DOM-like APIs

Other DOM-like APIs

- DOM is the basis of "competitors"
- Often language-specific
- Some change some detail
- Some use the DOM as a base

Compliance and Extensions

- DOM specification divided into modules
- Some modules are mandatory, some optional for DOM compliance
- DOM Compliance defined for each module
- Modules define basic functionality
- Implementations can define their own proprietary extensions and convenience methods
- Check documentation for what's standard, what's extensions
- Implementations can use any programming language

DOM Architecture

Terminology

Hosting Implementation

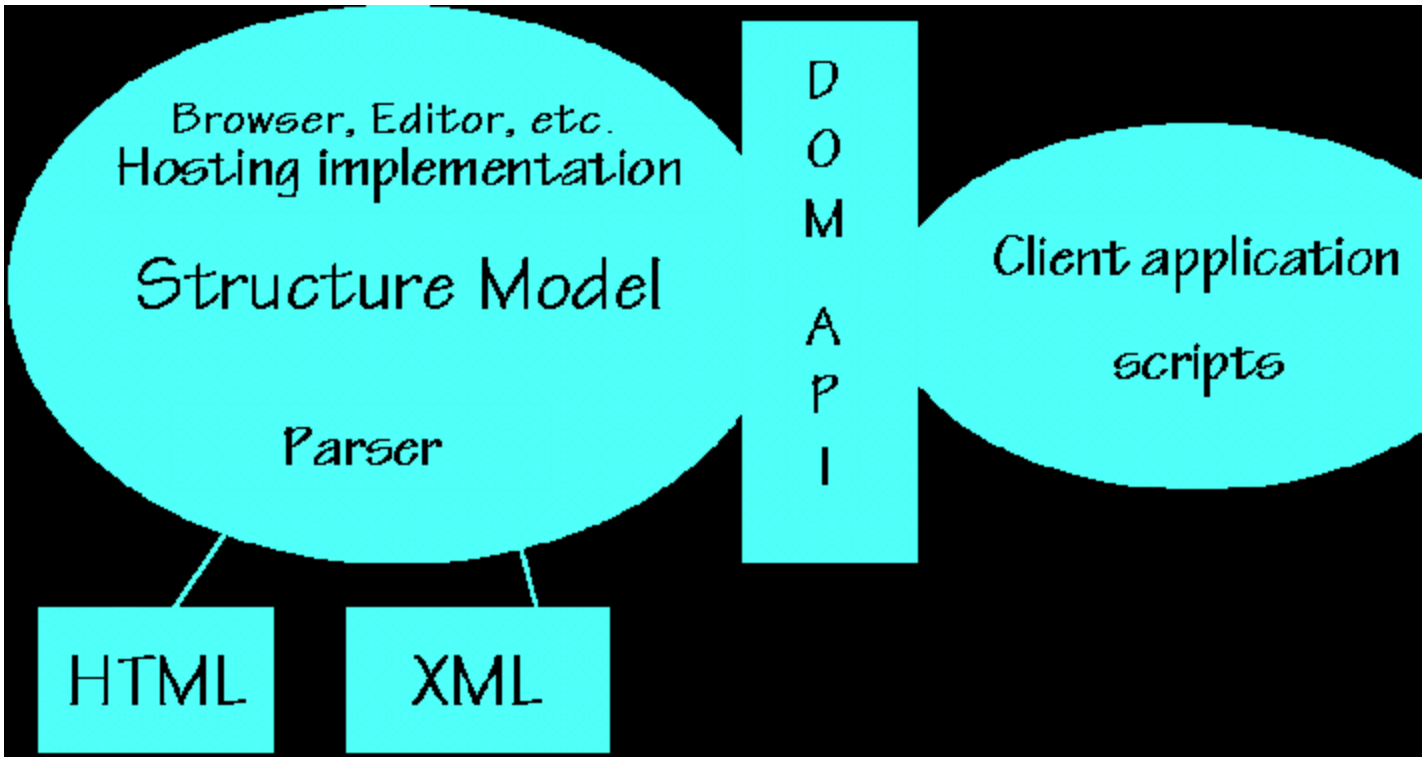
software that provides the interfaces, e.g., browser, XML parser/DOM

Client Application

software that uses the interfaces, e.g., script

Basic Architecture

Basic Architecture



DOM and SAX

- DOM builds a document structure model and gives the API to that
- SAX gives API to event-based XML parsing
- DOM is easier to program
- SAX takes less memory
- DOM useful for tree-based processing
- SAX useful for stream processing
- Many systems implement both

... could use either, or both, depending on need

Independence

- DOM is platform-neutral
- DOM is language-neutral
- DOM is independent of types of implementation
- DOM can be used anywhere XML is found

IDL

- Interfaces defined in OMG IDL
- Not a programming language
- Can create bindings to other languages from IDL
- DOM specifies Java and ECMAScript bindings

- DOM specifies Java and ECMAScript bindings
- Other languages can be used

ECMA Script and J(ava)Script

- JavaScript and JScript have an object model
- J(ava)Script includes collections such as images, forms, applets, links
- J(ava)Script includes ways of accessing and manipulating a document
- ECMA Script is different - pure programming language
- ECMA Script doesn't include any collections
- ECMA Script doesn't know what a document is

Fundamental Interfaces

Core - the Starting Point

- Navigation and manipulation of documents
- Fundamental interfaces (HTML and XML)
- Extended interfaces (XML only)

The Programmer's View

HTML and XML share some constructs:

- Objects are labelled
- Elements contain content
- Attributes add more information

Requirements

Structure Navigation:

- Navigate from any element to any other element
- Access all data

Document Manipulation

- add, remove, change elements
- add, remove, change attributes, and
- add, remove, change text content

Different tools, different methods

- Can't define everything everyone needs
- Different tools will use different languages
- Some tools may use COM, some may use CORBA

- Some tools may use COM, some may use CORBA
- Different types of tools need different things

The Parser and the DOM

- Hosting implementation has to have a parser
- Could be HTML parser, or XML
- The parser converts the document into some internal structure
- The DOM API acts on this structure
- The DOM API does not act on the document directly
- The parser also writes out the document

Functional Relationships

- Need to define how to represent things
- Elements are types of *Nodes*
- Can then use standard methods for navigating and manipulating Nodes
- Elements are in the document tree
- Attributes aren't in the tree
- Attributes are associated with an element
- Entity references are in the tree

Examples from DHTML

```
document.write("There are " + document.images.length + "
images");
```

- Don't know which elements are images in XML
- Need general methods

```
document.write("There are " +
document.getElementsByTagName("IMG").length + "images");
```

Using Core DOM

- Defines fundamental methods
- Generalizes HTML methods for general tag sets
- Adds special interfaces for XML constructs, e.g. CDATA Sections and Entities

Using the HTML DOM

- Convenience methods for known tag set
- Looks like Dynamic HTML or JavaScript
- Useful if you have HTML documents
- Not applicable to XHTML

Other work

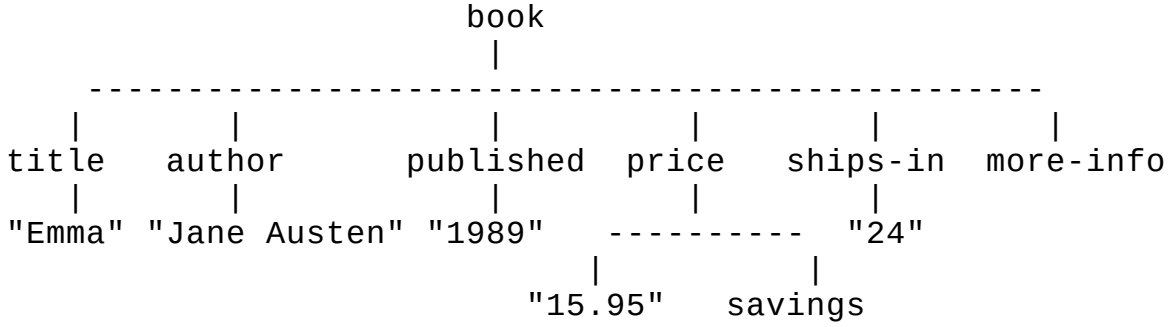
- Can define a DOM for other schemas
- SVG
- MathML
- Both based on DOM Level 2
- Demo of SVG

Core Example

Example: Book

```
<book>
<title>Emma</title>
<author>Jane Austen</author>
<published>1989</published>
<price>15.95
<savings category="A"></savings>
</price>
<ships-in units="hours">24</ships-in>
<more-info ISBN="089966" />
</book>
```

Tree View



Is it a Tree?

- Commonly called a tree (logical structure)
- Uses tree-walking paradigms
- Doesn't need to be implemented as a tree (physical structure)
- "Document tree" contains elements
- "Structure model" contains everything

Important Relationships

- Parent
- Child
- Sibling
- Ancestor
- Descendant

Example Task

- Each book is in a category
- The category determines the savings
- This week, Category "A"=30% savings
- Insert the savings value and final price

Manipulating Information

```
var doc_elt = document.documentElement;
var price = doc_elt.getElementsByTagName("price").item(0).first
//value is 15.95
var savings_elt = doc_elt.getElementsByTagName("savings").item
var att = savings_elt.getAttribute("category");
if (att == "A") {
   save = price * 0.30;
// value is 4.78
   savings_elt.appendChild(document.createTextNode(save));
}
total_elt = document.createElement("total");
text = document.createTextNode(price - save);
// value is 11.17
total_elt.appendChild(text);
savings_elt.appendChild(total_elt);
```

Final Result

```
<book>
<title>Emma</title>
<author>Jane Austen</author>
<published>1989</published>
<price>15.95
<savings category="A">4.78<total>11.17</total></savings>
</price>
<ships-in units="hours">24</ships-in>
<more-info ISBN="089966">
</book>
```

Demo! Perl and XMetaL.

Objects and Interfaces

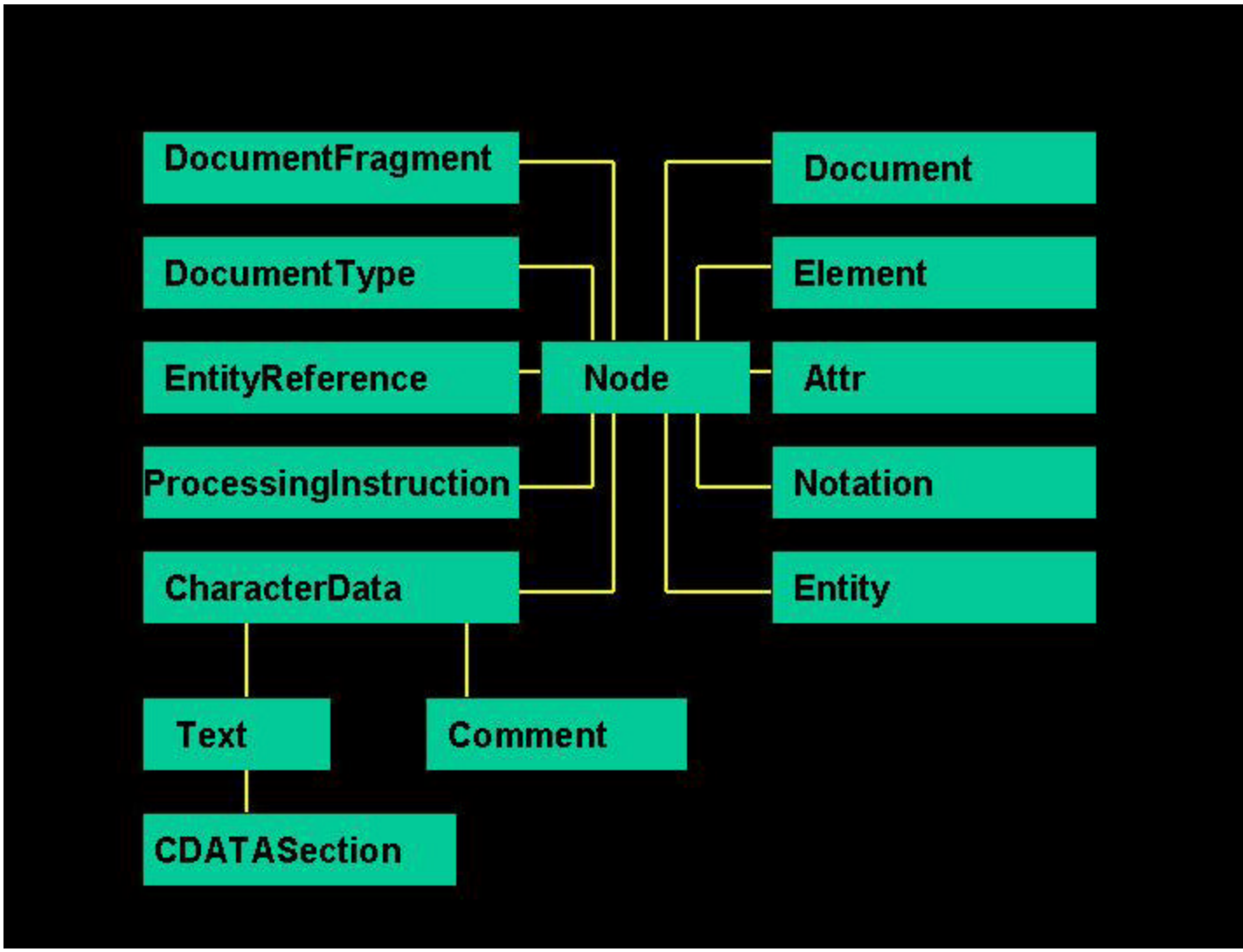
Implementation Independence

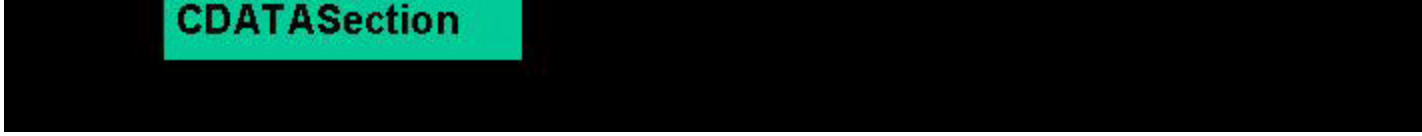
- Need a way to define functions and methods
- Shouldn't constrain implementations
- Allow many programming languages
- Allow inheritance

Interfaces are the Answer

- Interfaces don't define implementation
- Classes implement interfaces
- Interfaces can inherit
- Classes can implement more than one interface, e.g., Level 1 and Level 2
- Classes can implement proprietary interfaces as well

Core Node Inheritance





Getting from A to B

- Two ways to walk the document
- Tree-walking (`getNextSibling, getPreviousSibling`)
- NodeList (`getChildNodes`)
- NodeList is like a JavaScript collection
- NodeLists are live

DOMImplementation

```
interface DOMImplementation {
boolean            hasFeature(in DOMString feature,
                              in DOMString version);
// Introduced in DOM Level 2:
DocumentType       createDocumentType(in DOMString qualifiedNar
                                      in DOMString publicId,
                                      in DOMString systemId)
                                      raises(DOMException);
Document           createDocument(in DOMString namespaceURI,
                                  in DOMString qualifiedName,
                                  in DocumentType doctype)
                                  raises(DOMException);
};
```

Document

```
readonly attribute DocumentType       doctype;
readonly attribute Element          documentElement;
Element            createElement(in DOMString tagName)
                                    raises(DOMException);
DocumentFragment   createDocumentFragment();
Text               createTextNode(in DOMString data);
Comment            createComment(in DOMString data);
CDATASection       createCDATASection(in DOMString data)
                                    raises(DOMException);
ProcessingInstruction createProcessingInstruction(in DOMString
                                                   in DOMStri
                                    raises(DOMException);
Attr               createAttribute(in DOMString name)
                                    raises(DOMException);
EntityReference    createEntityReference(in DOMString name)
                                    raises(DOMException);
NodeList           getElementsByTagName(in DOMString tagname);
// Introduced in DOM Level 2:
Node               importNode(in Node importedNode,
                              in boolean deep)
                                    raises(DOMException);
Element            createElementNS(in DOMString namespaceURI,
```

```
Element             createElementNS(in DOMString namespaceURI,
                                    in DOMString qualifiedName)
                                       raises(DOMException);
Attr                createAttributeNS(in DOMString namespaceURI,
                                      in DOMString qualifiedName
                                       raises(DOMException);
NodeList            getElementsByTagNameNS(in DOMString namespac
                                           in DOMString localNam
Element             getElementById(in DOMString elementId);
```

Nodes

Part of the Node interface

```
readonly attribute DOMString      nodeName;
         attribute DOMString      nodeValue;
readonly attribute unsigned short nodeType;
readonly attribute Node           parentNode;
readonly attribute NodeList       childNodes;
readonly attribute Node           firstChild;
readonly attribute Node           lastChild;
readonly attribute Node           previousSibling;
readonly attribute Node           nextSibling;
Node                insertBefore(in Node newChild,
                                 in Node refChild)
                                 raises(DOMException);
Node                appendChild(in Node newChild)
                                raises(DOMException);
boolean             hasChildNodes();
// Modified or Introduced in DOM Level 2:
void                normalize();
boolean             isSupported(in DOMString feature,
                                in DOMString version);
readonly attribute DOMString        namespaceURI;
         attribute DOMString        prefix;
                    // raises(DOMException) on setting
readonly attribute DOMString        localName;
boolean             hasAttributes();
```

Elements

Part of the Element interface

```
readonly attribute DOMString       tagName;
DOMString           getAttribute(in DOMString name);
void                setAttribute(in DOMString name,
                                 in DOMString value)
                                 raises(DOMException);
void                removeAttribute(in DOMString name)
                                    raises(DOMException);
Attr                getAttributeNode(in DOMString name);
Attr                setAttributeNode(in Attr newAttr)
```

```
Attr                setAttributeNode(in Attr newAttr)
                                raises(DOMException);
Attr                removeAttributeNode(in Attr oldAttr)
                               raises(DOMException);
NodeList            getElementsByTagName(in DOMString name);
// Introduced in DOM Level 2:
DOMString           getAttributeNS(in DOMString namespaceURI,
                                   in DOMString localName);
void                setAttributeNS(in DOMString namespaceURI,
                                   in DOMString qualifiedName,
                                   in DOMString value)
                                    raises(DOMException);
void                removeAttributeNS(in DOMString namespaceURI,
                                      in DOMString localName)
                                    raises(DOMException);
Attr                getAttributeNodeNS(in DOMString namespaceURI
                                    in DOMString localName);
Attr                setAttributeNodeNS(in Attr newAttr)
                                    raises(DOMException);
NodeList            getElementsByTagNameNS(in DOMString namespac
                                          in DOMString localNam
boolean             hasAttribute(in DOMString name);
boolean             hasAttributeNS(in DOMString namespaceURI,
                                   in DOMString localName);
```

Text, Comments, etc.

Basis is `CharacterData Node`

```
      attribute  DOMString data;
void              insertData(in unsigned long offset,
                             in DOMString arg);
void              deleteData(in unsigned long offset,
                             in unsigned long count)
                             raises(DOMException);
```

Text

```
interface Text : CharacterData {
 Text  splitText(in unsigned long offset)
                 raises(DOMException);
};
```

- Text nodes don't merge automatically
- Normalized form is merged
- May need to normalize between operations

Attr

- Two ways to get at attributes
- This one more powerful, allows for entities

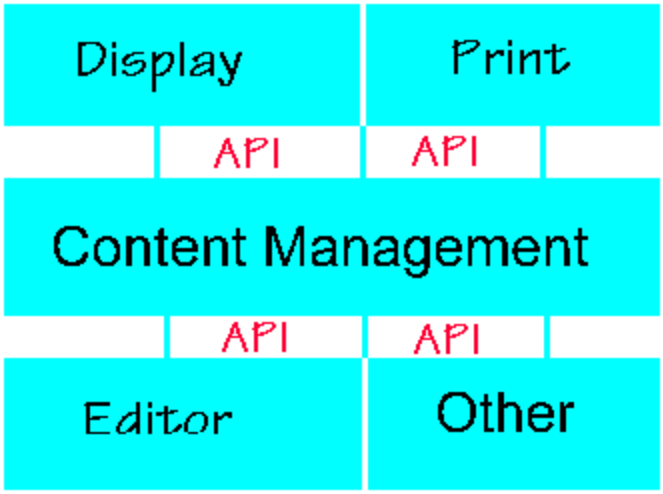
- This one more powerful, allows for children

```
interface Attr : Node {
  readonly attribute DOMString      name;
  readonly attribute boolean        specified;
           attribute DOMString      value;
            // raises(DOMException) on setting
  // Introduced in DOM Level 2:
  readonly attribute Element        ownerElement;
};
```

DocumentFragment

- Same allowed contents as XML parsed entity
- Used as "clipboard"
- Not part of document tree
- `insertBefore` etc insert contents, not the DocumentFragment node

Application Linkage Scenario



What the DOM adds

- Ease of integration
- One model for all pieces
- Write scripts once, tweak for other tools
- Demo using Tamino and XMetaL

Stylesheets

Stylesheet Features

Stylesheet Features

- Abstract support for stylesheets
- Style-specific API builds on this
- Get linked stylesheets
- Associate a stylesheet with a document

CSS

CSS Features

- Programmatically access and modify CSS
- Dynamically control the inclusion and exclusion of style sheets
- Manipulate CSS rules and properties.
- Interface for CSS Stylesheets
- Interfaces for each specific type of rule
  - style declarations
  - @import...
- Generic rule interface
- Optional CSS2Properties shortcuts

Events

Events Features

- Generic event system
- Registration of event handlers
- Event flow through a document structure
- Contextual information for each event

Basic Event Flow

- Event sent towards EventTarget
- EventTarget specified in the Event's target attribute
- Event listeners triggered when Event reaches target
- If multiple listeners, order is unspecified
- Some events are cancelable

Event Capturing

- Operates from top of tree (document element)
- Triggers capturing event listeners on the way down
- Stops when event's target is reached

- Stops when event's target is reached
- Can stop earlier by using `stopPropagation` method
- Listeners at same level still get event

Event Bubbling

- Event dispatched to target EventTarget
- Bubbling events then trigger additional event listeners on the way up the tree
- Goes up to Document node
- Bubbling can be stopped by using `stopPropagation`

Event Sets

- Contain context information
- UI event set (e.g., focusIn and focusOut)
- Mutation event set (e.g., NodeRemoved)
  parent, prevValue, etc.
- Mouse event set (e.g., click)
  screenX, screenY, ctrl-click, etc.

Traversal

NodeIterators

- NodeIterators simplify tree traversal
- "Ordered list" view of document
- Methods to move forward and backward
- Reference node is last node returned
- Iterator moves relative to reference node - even if document modified

Flags and Filters

- Flags say what to show (elements, entity references, ...)
- Flags are like built-in filters
- Filters are user-written functions
- Filters say whether NodeIterator or TreeWalker should "see" node
- Filters return
  - FILTER_SKIP
  - FILTER_ACCEPT
  - FILTER_REJECT

TreeWalker

- Looks like document tree

- Looks like document tree
- Methods to move up and down as well as back and forth
- Maintain location relative to a node if tree modified
- Remain attached to that node if it is moved

Range

Range Features

- Like a selection, but without GUI
- Boundary points anywhere in the document
- Can perform operations within the range
  - delete
  - clone
  - surround
- Document is fixed up if modified
- Useful for editing operations

Level 3 DOM

Modules

- Content models, validation
- Load, save, serialize
- Embedded DOM
- Events (key events, event grouping)
- Core

Current Status

- Level 1 is a W3C Recommendation; widely implemented
- Level 2 is a W3C Recommendation; increasingly widely implemented
- First Working drafts out for Level 3

Further Information

- DOM information page at

  http://www.w3.org/DOM

- Combined Level 1 and 2 specification at