

Practical XML Parsing

or

Bringing SAXy Back



Paul Goracke
paul@corporationunknown.com

! (XML > JSON)

! (XML > YAML)

! (XML > Everything)

- XHTML, MathML, SVG
- AJAX, SMIL
- REST, SOAP, WSDL, XML-RPC
- RSS, Atom
- RDF, ebXML, XBRL
- Office Documents (OOXML)
- XUL, XAML

Parts of XML

XML Declaration

Tags

Elements

Attributes

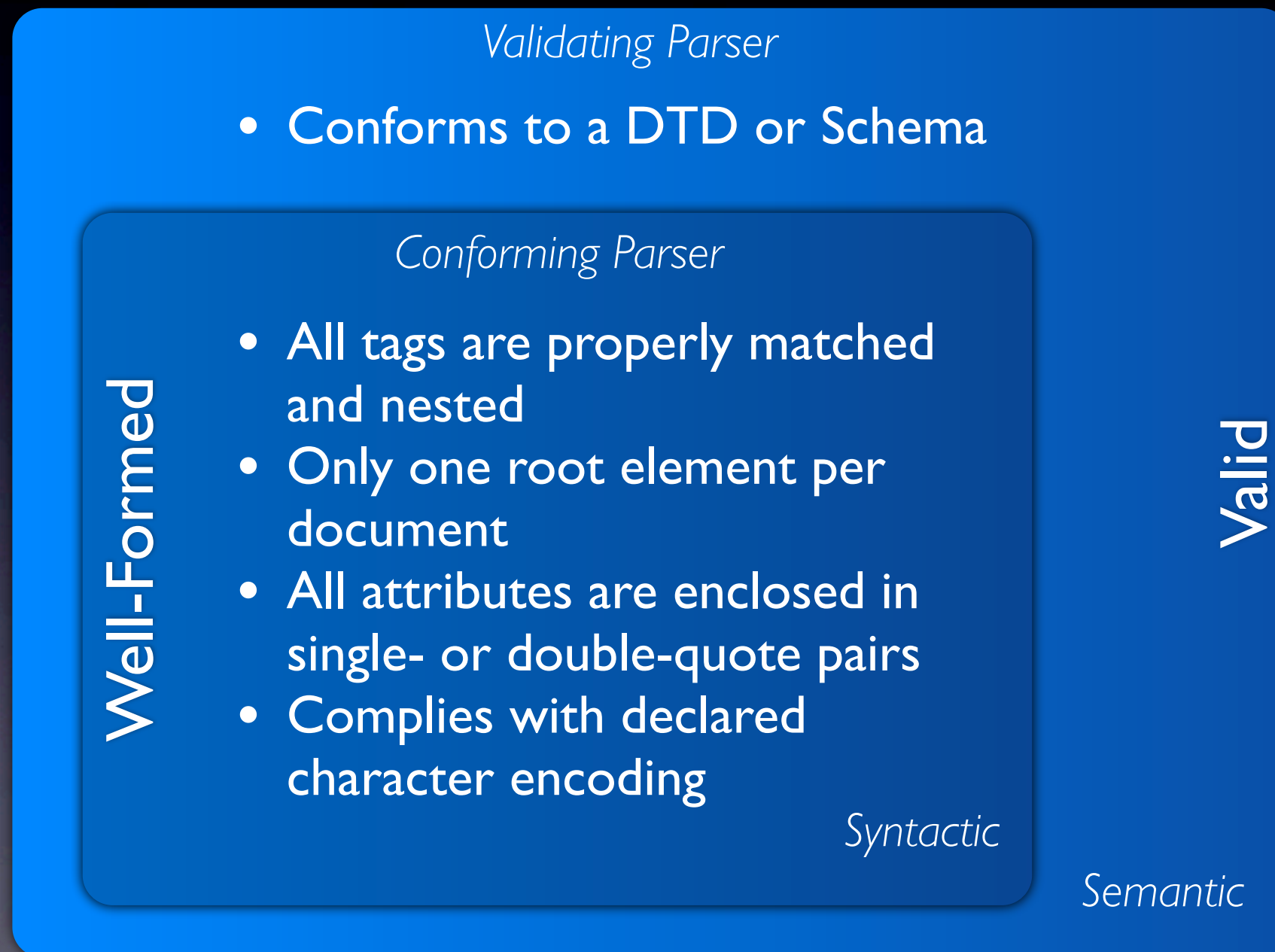
Text Nodes

Comments

Processing Instructions

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <?xml-stylesheet type="text/xml" href="fake.xsl"?>
3 <rss version="2.0" xmlns:dc="http://purl.org/dc/elements/1.1/"
4   xmlns:sy="http://purl.org/rss/1.0/modules/syndication/"
5   xmlns:content="http://purl.org/rss/1.0/modules/content/"
6   xmlns:atom="http://www.w3.org/2005/Atom"
7   xmlns:itms="http://phobos.apple.com/rss/1.0/modules/itms/">
8   <!-- This doc needs a comment! -->
9   <channel>
10     <title>iTunes 300 New Releases</title>
11     <link>http://itunes.apple.com/WebObjects/MZStore.woa/wa/viewNewReleases?pa
12     <atom:link
13       href="http://itunes.apple.com/WebObjects/MZStore.woa/wpa/MRSS/newrele
14       rel="self" type="application/rss+xml"/>
15     <description>iTunes Store: Today's 300 Newest Releases</description>
16     <language>en</language>
17     <copyright>Copyright 2008 Apple Inc. </copyright>
18     <lastBuildDate>Mon, 06 Apr 2009 07:46:53 -0700</lastBuildDate>
19     <generator>iTunes Music Store RSS v1.0.0</generator>
20     <webMaster>musicstore@apple.com (iTunes Music Store)</webMaster>
21     <ttl>240</ttl>
22
23     <dc:creator>iTunes Music Store</dc:creator>
24     <dc:date>2009-04-06T07:46:53-07:00</dc:date>
25
26     <sy:updatePeriod>hourly</sy:updatePeriod>
27     <sy:updateFrequency>1</sy:updateFrequency>
28     <sy:updateBase>2003-09-01T12:00+00:00</sy:updateBase>
29     <image>
30       <url>http://itunes.apple.com/images/rss/badge.gif</url>
31       <link>http://www.apple.com/itunes/</link>
32       <title>iTunes Music Store</title>
33       <height>31</height>
34       <width>88</width>
```

Well-Formed vs Valid



Use a Parser

Use a Parser

- Regexes will get ugly quick
 - Must be non-greedy
 - Must account for nesting
 - Empty `<node />`
 - Multiple lines
 - Attributes can begin with single- or double-quotes and must end same
 - Then you get to deal with entity encoding

Document Object Model

- Loads the XML document into memory as a tree of nodes to iterate (and manipulate)
- W3C Standard
- In my experience, one of the most commonly used XML models due to it “making sense” for most developers
- Main DOM class in Cocoa is NSXMLDocument

DOM: Create Document

```
- (NSXMLDocument *) newXMLDocumentFromFile:(NSString *)filePath {
    NSXMLDocument *xmlDoc;
    NSError *err = nil;

    NSURL *furl = [[NSURL alloc] initWithFileURLWithPath:filePath];
    if ( ! furl ) {
        NSLog( @"Can't create an URL from file %@.", filePath );
        return;
    }

    xmlDoc = [[NSXMLDocument alloc] initWithContentsOfURL:furl
                                                    options:NSXMLDocumentTidyHTML
                                                    error:&err];

    [furl release];
    return xmlDoc;
}
```

DOM: Create Document

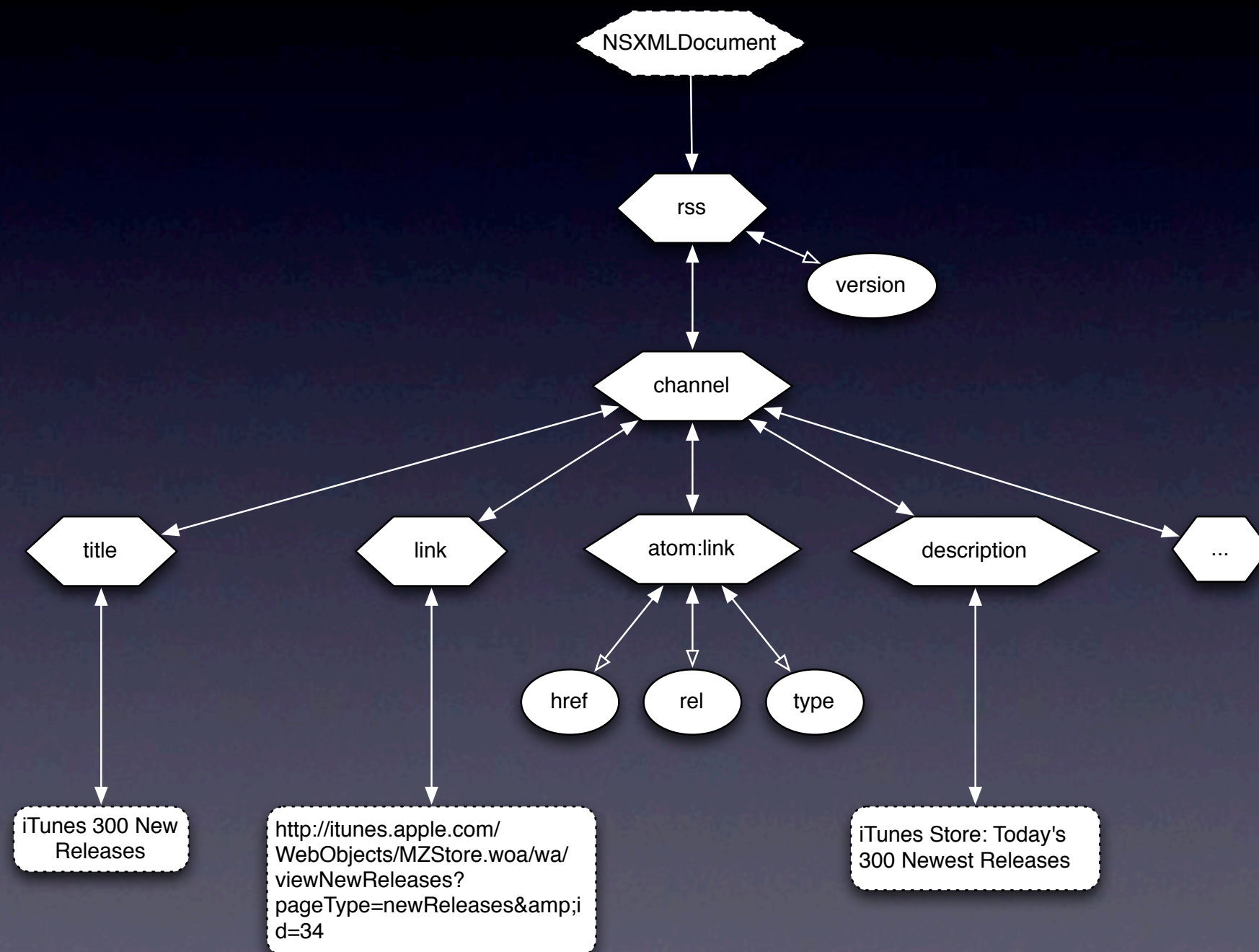
Additional NSXMLDocument constructors:

- (id) initWithData:(NSData *)data
options:(NSUInteger)mask
error:(NSError **)error
- (id) initWithXMLString:(NSString *)string
options:(NSUInteger)mask
error:(NSError **)error

DOM: The Tree

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <?xml-stylesheet type="text/xml" href="fake.xsl"?>
3 <rss version="2.0" xmlns:dc="http://purl.org/dc/elements/1.1/"
4   xmlns:sy="http://purl.org/rss/1.0/modules/syndication/"
5   xmlns:content="http://purl.org/rss/1.0/modules/content/"
6   xmlns:atom="http://www.w3.org/2005/Atom"
7   xmlns:itms="http://phobos.apple.com/rss/1.0/modules/itms/">
8   <!-- This doc needs a comment! -->
9   <channel>
10     <title>iTunes 300 New Releases</title>
11     <link>http://itunes.apple.com/WebObjects/MZStore.woa/wa/viewNewReleases?pr
12     <atom:link
13       href="http://itunes.apple.com/WebObjects/MZStore.woa/wpa/MRSS/newrele
14       rel="self" type="application/rss+xml"/>
15     <description>iTunes Store: Today's 300 Newest Releases</description>
16     <language>en</language>
17     <copyright>Copyright 2008 Apple Inc. </copyright>
18     <lastBuildDate>Mon, 06 Apr 2009 07:46:53 -0700</lastBuildDate>
19     <generator>iTunes Music Store RSS v1.0.0</generator>
20     <webMaster>musicstore@apple.com (iTunes Music Store)</webMaster>
21     <ttl>240</ttl>
22
23     <dc:creator>iTunes Music Store</dc:creator>
24     <dc:date>2009-04-06T07:46:53-07:00</dc:date>
25
26     <sy:updatePeriod>hourly</sy:updatePeriod>
27     <sy:updateFrequency>1</sy:updateFrequency>
28     <sy:updateBase>2003-09-01T12:00+00:00</sy:updateBase>
29     <image>
30       <url>http://itunes.apple.com/images/rss/badge.gif</url>
31       <link>http://www.apple.com/itunes/</link>
32       <title>iTunes Music Store</title>
33       <height>31</height>
34       <width>88</width>
```


DOM: The Tree



DOM: Traverse the Tree

- Retrieve the document's root element

```
NSXMLElement * root = [xmlDoc rootElement];
```

- Traverse using NSXMLNode methods

- (NSXMLNode *) parent;
- (NSUInteger) childCount; // prefer over [[node children] count]
- (NSXMLNode *) childAtIndex:(NSUInteger)index;
- (NSArray *) children; // array of NSXMLNode *
- (NSXMLNode *) {next,previous}Node;
- (NSXMLNode *) {next,previous}Sibling;

- Inspect the current NSXMLNode

- (NSXMLNodeKind) kind;
- (NSUInteger) index; // 0-based
- (NSUInteger) level; // 1-based (=root)
- (NSString *) name;
- (NSString *) localName;
- (NSString *) URI;

Demo

DOM: Traverse the Tree

- Since different node types have different behaviors, much of the inspection code begins by checking the node kind and possibly casting

```
if ( [node kind] == NSXMLTextKind ) {  
    // store the text  
}  
else {  
    // inspect the name  
}
```

- So much for object-oriented programming

DOM: Explore the Tree

- Simplify a bit by fetching child elements with a given name

```
NSArray * titles = [elem elementsForName:@"title"];
```

- Except that only returns children, not descendants

XPath

- Syntax for specifying parts of an XML doc
- Paths, like Unix paths
 - /node
 - /node/@attr (attribute of node)
 - /node[attr='value'] (predicate)
 - /node/child[1] (first child, note 1-based)
- Axes (child::, following-sibling::, ...)

Demo

XSLT

- eXtensible Stylesheet Language (Transformations)
- Conditionals, loops, variables, parameters
- But more declarative than procedural
- Template match

XSLT

```
<?xml version="1.0">
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:itms="http://phobos.apple.com/rss/1.0/modules/itms/">
<xsl:template match="/">
  <html><body>
    <h2>iTunes 300 New Releases</h2>
    <table border="1">
      <tr><th>Title</th><th>Artist</th></tr>
      <xsl:for-each select="rss/channel/item">
        <tr>
          <td><xsl:value-of select="title" /></td>
          <td><xsl:value-of select="itms:artist" /></td>
        </tr>
      </xsl:for-each>
    </table>
  </body></html>
</xsl:template>
</xsl:stylesheet>
```


XSLT



The screenshot shows a web browser window titled "tunes_xsl.html". The address bar displays the file path "file:///Users/paul/Dropbox/Presentation/tunes_xsl.html". The page content is titled "iTunes 300 New Releases" and features a table with two columns: "Title" and "Artist". The table lists 30 music releases, including various artists, Piers Faccini, Unkle Bob, Taylor Deupree, Blu Mar Ten, PSAPP, The Veils, Venetian Snares, Siouxsie and the Banshees, Distance, Liverpool Collective, Uneaq, Markus Schulz, Sonny Landreth, Krazy Baldhead, Vernon & Dacosta, Carlos Gardel, Trigger the Bloodshed, and Dominik Eulberg.

Title	Artist
XS Dein 30+ Club - Various Artists	Various Artists
Two Grains of Sand - Piers Faccini	Piers Faccini
Satellite / Bad Dream - Unkle Bob	Unkle Bob
Weather & Worn - EP - Taylor Deupree	Taylor Deupree
Close - EP - Blu Mar Ten	Blu Mar Ten
Early Cats and Tracks, Vol. 2 - PSAPP	PSAPP
Sun Gangs - The Veils	The Veils
Filth - Venetian Snares	Venetian Snares
Bassline & Garage EP 1 - Various Artists	Various Artists
Hyaena (Bonus Track Version) [Remastered] - Siouxsie and the Banshees	Siouxsie and the Banshees
Night Vision (Skream's "So Nasty" Version) - Single - Distance	Distance
Fields of Anfield Road - EP - Liverpool Collective	Liverpool Collective
50 Trance Hits, Vol. 1 - Various Artists	Various Artists
Shake It Up - Single - Uneaq	Uneaq
Toronto '09: The Full Versions, Vol. 1 - Markus Schulz	Markus Schulz
Voices of Americana: Sonny Landreth - Sonny Landreth	Sonny Landreth
The B Suite - Krazy Baldhead	Krazy Baldhead
A Part of Mine (feat. Ingrid Hakanson) - EP - Vernon & Dacosta	Vernon & Dacosta
Latin Pearls, Vol. 1 - Carlos Gardel	Carlos Gardel
The Great Depression - Trigger the Bloodshed	Trigger the Bloodshed
Tinderbox (Remastered & Expanded) - Siouxsie and the Banshees	Siouxsie and the Banshees
Sensorika - EP - Dominik Eulberg	Dominik Eulberg

XSLT

Relevant NSXMLDocument methods:

- (id) objectByApplyingXSLT:(NSData *)xslt
arguments:(NSDictionary *)arguments
error:(NSError **)error
- (id) objectByApplyingXSLTatURL:(NSURL *)xsltURL
arguments:(NSDictionary *)arguments
error:(NSError **)error
- (id) objectByApplyingXSLTString:(NSString *)xslt
arguments:(NSDictionary *)arguments
error:(NSError **)error

XQuery

- XQuery : XML :: SQL : RDBMS Table
- Designed to query XML data

```
for $x in doc("books.xml")/bookstore/book
where $x/price > 30
order by $x/title
return $x/title
```


Document Object Model

Pros	Cons
<ul style="list-style-type: none">• Low coding cost of entry• Allows almost arbitrary manipulation of tree, and thus the document	<ul style="list-style-type: none">• Not available on iPhone• Must parse the entire document before you can do anything with it• Memory needs increase with document size

Simple API for XML

- Event-driven: You receive callback notifications (delegate messages)
- Stream-capable, may begin parsing before entire document received*

* not available in NSXMLParser

- Lower, more predictable memory requirement

SAX: Begin Parsing

- Create an NSXMLParser object
- Configure how it parses
- Set your delegate
- Start the parser a-parsing

```
- (void) parseXMLFile:(NSURL *)fileURL {  
    NSXMLParser* parser =  
        [[NSXMLParser alloc] initWithContentsOfURL:fileURL];  
    parser.delegate = self;  
    [parser parse];  
    // if necessary, do something with parsed data  
    [parser release];  
}
```


SAX: Delegate Methods

- Manage “global” state for this document

```
- (void) parserDidStartDocument:(NSXMLParser*)parser {  
    // create storage for parsed objects  
    self.items = [NSMutableArray array];  
    // create storage for text nodes  
    self.currentString = [NSMutableString string];  
    // maybe begin a database transaction  
}  
  
- (void) parserDidEndDocument:(NSXMLParser*)parser {  
    // successfully parsed, so commit transaction  
    // clear storage used  
    self.currentString = nil;  
    self.items = nil;  
}
```

SAX: Delegate Methods

- Handle beginning of elements

```
- (void)parser:(NSXMLParser *)parser
  didStartElement:(NSString *)elementName
    namespaceURI:(NSString *)namespaceURI
  qualifiedName:(NSString *) qualifiedName
    attributes:(NSDictionary *)attributeDict
{
    if ([elementName isEqualToString:kName_Item]) {
        self.currentSong = [[[Song alloc] init] autorelease];
    } else if ([elementName isEqualToString:kName_Title] || [elementName
isEqualToString:kName_Category] || [elementName isEqualToString:kName_Artist] ||
[elementName isEqualToString:kName_Album] || [elementName
isEqualToString:kName_ReleaseDate]) {
        [currentString setString:@""];
        storingCharacters = YES;
    }
}
```


SAX: Delegate Methods

- Handle ending of elements

```
- (void)parser:(NSXMLParser *)parser
didEndElement:(NSString *)elementName
namespaceURI:(NSString *)namespaceURI
qualifiedName:(NSString *)qName
{
    if ([elementName isEqualToString:kName_Item]) {
        [self finishedCurrentSong];
    } else if ([elementName isEqualToString:kName_Title]) {
        currentSong.title = currentString;
    } else if ([elementName isEqualToString:kName_Category]) {
        currentSong.category = currentString;
    } else if ([elementName isEqualToString:kName_Artist]) {
        currentSong.artist = currentString;
    } else if ([elementName isEqualToString:kName_Album]) {
        currentSong.album = currentString;
    } else if ([elementName isEqualToString:kName_ReleaseDate]) {
        currentSong.releaseDate = [parseFormatter dateFromString:currentString];
    }
    storingCharacters = NO;
}
```


SAX: Delegate Methods

- Handle characters
 - Need to assume you will receive randomly sized chunks of text
 - Append to your mutable string
 - `didStartElement` handles creating empty strings
 - `didEndElement` handles assigning current string
 - CDATA and entities are already handled for you

```
- (void)parser:(NSXMLParser *)parser  
foundCharacters:(NSString *)string  
{  
    if (storingCharacters) [currentString appendString:string];  
}
```

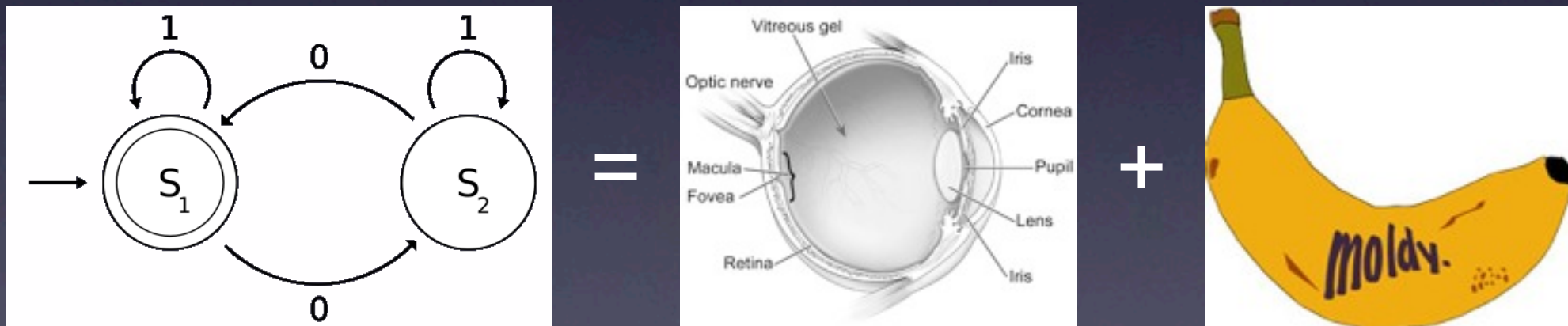
SAX: Delegate Methods

- Handle errors

```
- (void)parser:(NSXMLParser *)parser  
parseErrorOccurred:(NSError *)parseError  
{  
    // rollback database transaction?  
    // clear storage used  
    self.currentString = nil;  
    self.items = nil;  
  
    // do something useful with parseError  
}
```

SAX: Delegate Methods

- For typical usage, that's pretty much it
- Congratulations, you've just made a finite state machine



SAX: Miscellaneous

- You may choose to terminate early

```
- (void)parser:(NSXMLParser *)parser
  didEndElement:(NSString *)elementName
  namespaceURI:(NSString *)namespaceURI
  qualifiedName:(NSString *)qName
{
    if ([elementName isEqualToString:kName_Artist]) {
        if ( [currentString isEqualToString:@"Celine Dion"] ) {
            // no sense looking at the rest of this list
            [parse abortParsing];
        }
        else {
            self.currentSong = currentString;
        }

        storingCharacters = NO;
    }
}
```

SAX: Miscellaneous

- You may change delegate mid-stream

```
- (void)parser:(NSXMLParser *)parser
  didStartElement:(NSString *)elementName
    namespaceURI:(NSString *)namespaceURI
  qualifiedName:(NSString *) qualifiedName
    attributes:(NSDictionary *)attributeDict
{
    if ([elementName isEqualToString:kName_Item]) {
        parser.delegate = self.songHandler;
        // songHandler is now responsible for <item> descendants
        // and needs to return control back to us at </item>
    }
    ...
}
```

SAX: Miscellaneous

- You may fire off more downloads while still parsing

```
- (void)parser:(NSXMLParser *)parser
  didEndElement:(NSString *)elementName
  namespaceURI:(NSString *)namespaceURI
  qualifiedName:(NSString *)qName
{
    if ([elementName isEqualToString:@"itms:coverArt"]) {
        // fire async download of url in currentString
    }
    ...
}
```


libxml2

- MIT-licensed XML library (in C)
<http://xmlsoft.org/>
- Available on iPhone
- High performance
- Can process in chunks for streams

Namespaces

- Namespaces look like attributes but are their own thing

`xmlns:itms="http://phobos.apple.com/rss/1.0/modules/itms/"`

- The "value" defines a unique namespace
- Defines a prefix ("itms") which is used for elements and attributes belonging to the namespace within its scope

Namespaces

- By convention, a namespace usually has a universal prefix everyone uses
- But it's not guaranteed
- To ensure compatibility, you should do a two-step comparison of name and URI

Namespaces

```
...
[parser setShouldProcessNamespaces:YES];
parser.delegate = self;
[parser parse];
...

- (void)parser:(NSXMLParser *)parser
  didStartElement:(NSString *)elementName
    namespaceURI:(NSString *)namespaceURI
  qualifiedName:(NSString *) qualifiedName
    attributes:(NSDictionary *)attributeDict
{
    // elementName = @"itms:artist"
    // namespaceURI = @"http://phobos.apple.com/..."
    // qualifiedName = @"artist"

    if ( [namespaceURI isEqualToString:kNamespace_itms]
        && [qualifiedName isEqualToString:@"artist"] ) {
        ...
    }
}
```

Namespaces

- Most code examples ignore namespaces
- Most code works fine with the assumption that a prefix is universal
- It's up to you to decide if the effort is worth it for your situation

References

- Apple Documentation
 - Reference Library > Guides > Cocoa > Data Management
 - Tree-Based XML Programming Guide for Cocoa (DOM, NSXML *)
 - Event-Driven XML Programming Guide for Cocoa (SAX, NSXMLParser *)
- Harvard Extension CSI E-259:
 - <http://deimos3.apple.com/WebObjects/Core.woa/Browse/extension.harvard.edu.1278515723.01278515730>