



© 1997-1998 Adobe Systems Incorporated. All rights reserved.



Understanding & Implementing XML

*Everything you always wanted to know
about XML but were afraid to ask*

Leonard Rosenthol
Senior Software Engineer
Layout & Internet Technologies
Adobe Systems Incorporated



What you should leave with

- ◆ Understanding of what XML is, and what it is not! (ie. Avoid the hype)
- ◆ Ability to read and write XML
- ◆ Desire to purchase lots of Adobe Software



How we'll get there

- ◆ Just what is XML & how does it relate to HTML and SGML
- ◆ XLink, XPointer & other ways to associate and link data
- ◆ DTD's & XSchema - putting meaning on structure
- ◆ CSS2 & XSL - a little style goes a long way
- ◆ Applications for XML
 - ◆ MathML, SMIL, PGML/SVG, etc.



You're here because...

- ◆ You are interested in finding out what all the hype over XML is all about
- ◆ You're currently working the HTML and other web standards and want to see what the future holds
- ◆ You had nothing better to do after the keynote
- ◆ You a friend of mine, and you came to heckle



How I do things...

- ◆ You don't need to take copious notes to copy down the slides themselves.
 - ◆ A copy of this PDF file will be available from my website (<http://www.lazerware.com/>) by next week.
- ◆ Although I've left time for Q & A at the end, I am more than happy to take questions at any time during the presentation



All about the W3C

- ◆ World Wide Web Consortium
- ◆ International non-profit organization which is responsible for developing, tracking and managing new standards for the Web.
 - ◆ HTML, XML, CSS, and LOTS more
- ◆ See <http://www.w3.org>



What is XML?

& How does it relate to HTML and SGML?



SGML

- ◆ Standardized General Markup Language
- ◆ Developed in 1974 by Charles F. Goldfarb and a host of others as a means to create a single basis for any type of markup language
 - ◆ ISO Standard as of 1986
 - ◆ See <http://www.oasis-open.org/cover/sgml-xml.html>
- ◆ This is the foundation on which HTML and XML are based.



HTML

- ◆ HyperText Markup Language
- ◆ Developed by Tim Berners -Lee as part of a research project to enable sharing of data over the internet.
 - ◆ And look where's it's gotten us!
- ◆ Presentation NOT data oriented !
- ◆ Bastardization by Netscape and Microsoft not with standing (eg. <BLINK>), HTML has a fixed set of tags.



XML

- ◆ eXtensible Markup Language
- ◆ “SGML-lite”
- ◆ Developed by the W3C to address the biggest limitations of HTML
 - ◆ data vs. presentation
 - ◆ fixed set of tags
 - ◆ too much proprietary drech being employed to solve simple/general problems



The XML 1.0 specification

- ◆ XML 1.0 specification can be found at <http://www.w3.org/TR/1998/RECC-xml-19980210.html>.
- ◆ Though an even better version called the "Annotated XML Specification" by Tim Bray can be found at <http://www.xml.com/axml/testasml.htm>



XML is NOT a markup language!

- ◆ Although the name (extensible markup language) clearly gives you this impression, unlike HTML, XML itself has no tags to learn.
- ◆ XML is really a specification that allows for specific markup languages to be created for specific purposes all within the same compatible syntax (but not the same tag set!)
 - ◆ HTML, MathML, PGML/SVG, etc.



Design Principles of XML

- ◆ XML should be straightforwardly usable over the Internet
- ◆ XML shall support a wide variety of applications
- ◆ XML shall be compatible with SGML
- ◆ It shall be easy to write programs that process XML documents
- ◆ The number of optional elements in XML is to be kept to an absolute minimum - ideally zero.



Design Principles of XML (cont.)

- ◆ XML documents should be human readable and reasonably clear
- ◆ The XML design should be prepared quickly
- ◆ The design of XML shall be formal and concise
- ◆ XML documents shall be easy to create
- ◆ Terseness in XML markup is of minimal importance



Show me some XML!

```
<?XML version="1.0" encoding="UTF-8"?>
<!DOCTYPE customerDB SYSTEM "customerDB.dtd">
<!-- Customer DataBase for some unnamed company -->
<DOCUMENT>
<CUSTOMER>
  <NAME>
    <LASTNAME>Edwards</LASTNAME>
    <FIRSTNAME>Britta</FIRSTNAME>
  </NAME>
  <DATE>April 17, 1998</DATE>
  <ORDERS>
    <ITEM>
      <PRODUCT>Cucumber</PRODUCT>
      <NUMBER>5</NUMBER>
      <PRICE>1.25</PRICE>
    </ITEM>
    <ITEM>
      <PRODUCT>Lettuce</PRODUCT>
      <NUMBER>2</NUMBER>
      <PRICE>.98</PRICE>
    </ITEM>
  </ORDERS>
</CUSTOMER>
</DOCUMENT>
```



Another example

```
<?XML version="1.0" encoding="UTF-8"?>
<!-- Minneapolis Airline Schedule - January 3rd, 1998 -->
<SCHEDULE>
  <AIRLINE>NorthWest</AIRLINE>
    <FLIGHT>
      <NUMBER>449</NUMBER>
      <STATUS>Cancelled</STATUS>
    </FLIGHT>
    <FLIGHT>
      <NUMBER>640</NUMBER>
      <STATUS depart="0100">Delayed</STATUS>
    </FLIGHT>
  <AIRLINE>TWA</AIRLINE>
    <FLIGHT>
      <NUMBER>1010</NUMBER>
      <STATUS gate="17 Gold">On Time</STATUS>
    </FLIGHT>
</SCHEDULE>
```



HTML vs. XML

- ◆ No fixed tag set
- ◆ ALL tags have both a start and end
 - ◆ Can't just do `<P>` - have to do `<P></P>`
 - ◆ ``
- ◆ Tags must be perfectly nested
 - ◆ `<I>foo</I>` - NOT!
- ◆ Capitalization of tags is significant
 - ◆ `<BOLD> text </bold>` - NOT!
- ◆ Whitespace is significant
 - ◆ Spaces, tabs, etc. are maintained!



Valid vs. Well-Formed

- ◆ A valid XML document is a “stand-alone” document containing all the information necessary to not only display the document but also do intelligent processing
 - ◆ In other words - it has a DTD!
- ◆ A well-formed XML document is a perfectly legal XML document, but does not contain information about structure to allow for intelligent processing
 - ◆ In other words - no DTD!



Looking closer at XML

```
<?XML version="1.0"?>
<!-- this is a portion of a database of B movies; Thanks John Simpson -->
<flixinfo>
    <title role="main">Shoot the Piano Player</title>
    <title role="alt" xml:lang="FR">Tirez sur le Pianiste </title>
    <crew>
        <director>Fran ois Truffaut</director>
    </crew>
    <plotsummary xml:space="default">Jaded piano player changes his
name and takes up with gangsters, much to his girlfriend's
chagrin</plotsummary>
    <distributor>&MUL;</distributor>
    <dialog><!CDATA[This film doesn't contain the line, "Play it,
Sam" -- but it MIGHT have.]]></dialog>
</flixinfo>
```



XML Declaration

```
<?XML version="1.0" encoding="UTF-8"?>  
<flixinfo>  
    <removed_to_save_space />  
</flixinfo>
```

The XML declaration is a processing instruction (PI) that identifies a document as being in XML. It also includes the version of XML used, and optionally other information such as the encoding used. UTF-8, Unicode in 8 bits, is the default for XML files, so the above is actually redundant.



Comment

```
<?XML version="1.0"?>  
  
<!-- this is a portion of a database of B movies; Thanks John Simpson -->  
  
<flixinfo>  
    <removed_to_save_space />  
</flixinfo>
```

A comment is text that's there for human readers - you want the parser to completely ignore it.

Unlike comments in HTML, putting computer-centric data (like JavaScript) into an XML comment violates the XML specification and your document would be considered invalid & incorrect XML.



Root Element

```
<?XML version="1.0"?>  
<flixinfo>  
    <removed_to_save_space />  
</flixinfo>
```

Elements are the basic building blocks of markup, and consist of the tags, the attributes and the content. The root element is the element which contains all other elements. An XML document **MUST** contain a root element - and only **ONE** root element.

The text inside of the element's tags is referred to as the element's name. ([a-zA-Z0-9_.-])



Attribute/value pair

```
<?XML version="1.0"?>
<flixinfo>
    <title role="main">Shoot the Piano Player</title>
    <title role="alt" xml:lang="FR">Tirez sur le Pianiste </title>
</flixinfo>
```

An attribute is additional data that is associated with an element, rather than being it's own element.

Attributes names and their values are both constrained by the document's DTD. If a document is well-formed but not valid (ie. No DTD), then there are no constraints.

Attribute names are limited to ([a-zA-Z0-9_.-])



Text content of element

```
<?XML version="1.0"?>  
<flixinfo>  
    <title role="main">Shoot the Piano Player</title>  
</flixinfo>
```

Nothing fancy here - it's the actual text data for the element in the specified encoding and/or language.

All content must be text - no binary “chunks” allowed. (there are exceptions and workarounds but it's best to avoid them to make your parser happy)



xml:lang attribute

```
<?XML version="1.0"?>
<flixinfo>
    <title role="alt" xml:lang="FR">Tirez sur le Pianiste </title>
</flixinfo>
```

xml:lang is a special “built-in” attribute that allows you to specify the natural language for the content of an element. This is most useful, as seen in the example, when you want to have multiple “translations” of a element.

The xml: portion of the name is part of a new addition to XML called namespaces - this helps avoid collisions when multiple DTD’s are in play.



XML Namespaces

- ◆ XML Namespaces is a proposed recommendation (enhancement) to the XML 1.0 specification called
 - ◆ See <http://www.w3.org/TR/WD-xml-names>



Character entities

```
<?XML version="1.0"?>
<flixinfo>
    <crew>
        <director>Fran&#231;ois Truffaut</director>
    </crew>
</flixinfo>
```

An entity is a “macro” or “boilerplate text” that you can reference in your XML document to avoid duplication of text or ease changes. They take the form of `&xxxx;` and can either be “built-in” or specified via the DTD.

In this example, we’re using a built-in entity referring to ASCII character 231 (c).



xml:space attribute

```
<?XML version="1.0"?>
<flixinfo>
    <plotsummary xml:space="default">Jaded piano player changes his
name and takes up with gangsters, much to his girlfriend's
chagrin</plotsummary>
</flixinfo>
```

xml:space is another special “built-in” attribute. It allows you to specify what a browser is to do with whitespace in the content of the element. “default” is (as you might guess) the default, but if you were trying to include a poem and wanted to maintain the look of the text (via whitespace) you would probably use xml:space=“preserve”.



"Built-in" entity

```
<?XML version="1.0"?>
<flixinfo>
    <plotsummary xml:space="default">Jaded piano player changes his
name and takes up with gangsters, much to his girlfriend&apos;s
chagrin</plotsummary>
</flixinfo>
```

In this example, we're using a built-in entity which represents the apostrophe character (').

The other built-in entities include & (&), > (>), < (<), and " (").



General entity

```
<?XML version="1.0"?>  
<flixinfo>  
    <distributor>&MUL;</distributor>  
</flixinfo>
```

Although built-in entities are useful, XML allows for the quite powerful ability to specify (via the DTD) your own entities (called general entities). These can be anything you want and can be as short or long as you'd like with any valid name.

In this example, &MUL; has been defined to be “Movies Unlimited” a distributor in Philadelphia.



Marked (CDATA) section

```
<?XML version="1.0"?>
<flixinfo>
    <dialog><!CDATA[This film doesn't contain the line, "Play it,
Sam" -- but it MIGHT have.]]></dialog>
</flixinfo>
```

Marked sections (CDATA) are used when you want to keep the parser off your data, and you want it in its “original” form without the need to convert certain characters to entities, deal with whitespace issues, etc.



XLink & XPointer

taking linking to the next level



Let's review HTML linking

- ◆ `My Home Page`
 - ◆ Requires a special tag `<a>`
 - ◆ Uses an attribute (`href`) to specify the URL
 - ◆ URL specifies the service type (`http`), system name (`server`) and document path - of which the first two are optional.



HTML Linking (cont.)

- ◆ `Table of Contents`
 - ◆ Can also provide a fragment identifier to go to a named section of a document
 - ◆ `Table of Contents`
- ◆ CGI's can also do fancy linking tricks by using the '?' notation in a URL.



But why would I need more?

- ◆ HTML links go *from* one single point *to* one single point
- ◆ HTML links only go in one direction
- ◆ HTML links retrieve the entire document to which it links
- ◆ Only one thing can happen when you click on a bit of linked text
- ◆ Linked documents can't be easily reassembled (next page/prev page)
- ◆ Using fragment identifiers requires changes to the linked resource
- ◆ HTML links require specific knowledge of the target's content



Not yet ready for prime time

- ◆ Unlike the XML 1.0 specification, XLink and XPointer are very much still in the development stages. Although there are a few working implementations of the current specs, it may be some time before something is approved by the W3C.



Links vs. Pointers

- ◆ XLink is the “replacement” for the hyperlinking that we do today with HTML. It extends it in new and exotic ways.
- ◆ XPointer for addressing specific portions of an XML document - “structure-aware” linking, if you will.
- ◆ You can use XLink without XPointer, but not the other way around.



Where to find more info...

- ◆ Current XLink draft is at
<<http://www.w3.org/TR/1998/WD-xlink-19980303>>
- ◆ Current XPointer draft is at
<<http://www.w3.org/TR/1998/WD-xptr-19980303>>
- ◆ Both XLink and XPointer are due to be promoted to “proposed recommendations” by June of 1999.



XLink Terminology - Resource

- ◆ A thing that can be involved in a link - including the thing that's doing the pointing
 - ◆ $X \rightarrow Z$ (X links to Z, but both X and Z are resources on the link)
 - ◆ X is the local resource, Z is the remote resource
- ◆ Possible resources include (but are not limited to) XML & HTML documents, portions of a document, images, multimedia, and files for downloading.



XLink Terminology - Locator

- ◆ The specific piece of the link's definition that tells you where to find the resource to which you're linked.
 - ◆ Usually this is the URL, but it could be an XPointer or something else.
 - ◆ Equivalent to the "href" in HTML



XLink Terminology - Links or Linking Elements

- ◆ “the explicit relationship between two or more data objects or portions of data objects”
- ◆ The XML element that defines the relationship
- ◆ They can be either inline or out-of-line
 - ◆ Inline links are like HTML links - the element itself is a resource.
 - ◆ Out-of-line links don't exist at a specific point in the document, and may even occur in some other document.



XLink Terminology - Links (cont.)

- ◆ They can either be simple or extended
 - ◆ Simple links are unidirectional (like in HTML) and can only be expressed inline.
 - ◆ Extended links can be either unidirectional or multi-directional. They can also be expressed using either inline or out-of-line conventions.



XLink Terminology - Traversal

- ◆ Traversing a link is to “make it happen”.
- ◆ In HTML, that means the user has clicked on the link - however in XLink, there is also the possibility of automatic traversal of links.



Anatomy of a Link

```
<reviewlink xml:link="simple"  
    href="http://www.flixml.com/detour.xml"  
    inline="true"  
    role="ReviewLink"  
    title="Link to review"  
    content-role="completerevuew"  
    content-title="FULL Review"  
    show="new"  
    actuate="user"  
    behavior="default">
```



Anatomy of a Link - xml:link

```
<reviewlink xml:link="simple"  
    href="http://www.flixml.com/detour.xml"  
    inline="true"  
    role="ReviewLink"  
    title="Link to review"  
    content-role="completerevuew"  
    content-title="FULL Review"  
    show="new"  
    actuate="user"  
    behavior="default">
```

- ◆ Xml:link is what marks an element as a linking element
- ◆ Values include "simple", "extended", "locator", "group" or "document"



Anatomy of a Link - href

```
<reviewlink xml:link="simple"  
    href="http://www.flixml.com/detour.xml"  
>
```

- ◆ Serves the same purpose as in HTML
- ◆ Can include fragment identifier (#) or new connector symbol (|).
 - ◆ Connector provides for server defined behavior
- ◆ Use of the ? in URL's is still being addressed by the working group.



Anatomy of a Link - inline

```
<reviewlink xml:link="simple"  
    href="http://www.flixml.com/detour.xml"  
    inline="true"  
>
```

- ◆ Used to specify if the link is inline or out-of-line
 - ◆ Default is inline="true"
- ◆ An inline link element is basically the same as an `` tag in HTML



Anatomy of a Link - role

```
<reviewlink xml:link="simple"  
    href="http://www.flixml.com/detour.xml"  
    role="ReviewLink"  
>
```

- ◆ This is a special purpose link that is really only useful given an intelligent browser/processor which would adjust it's behavior/grouping/display based on the role of the link in a particular setting.



Anatomy of a Link - title

```
<reviewlink xml:link="simple"  
    href="http://www.flixml.com/detour.xml"  
    title="Link to review"  
>
```

- ◆ “a displayable caption that explains the part the resource plays in the link”
- ◆ Useful for things such as tooltips, sight-impaired browsers, etc.



Anatomy of a Link - content-role

```
<reviewlink xml:link="simple"  
    href="http://www.flixml.com/detour.xml"  
    content-role="completereview"  
>
```

- ◆ Enhances the role attribute by providing some more specific information about the content - the specific resource being pointed to.
- ◆ Like the role attribute, this is only useful when you have an intelligent browser/processor.



Anatomy of a Link - content-title

```
<reviewlink xml:link="simple"  
    href="http://www.flixml.com/detour.xml"  
    content-title="FULL Review"  
>
```

- ◆ The current spec “does not require that application software make any particular use of title information”
- ◆ However, we can guess that possible uses for this would be to serve in the same way with respect to content-role that title does to role.



Anatomy of a Link - show

```
<reviewlink xml:link="simple"  
    href="http://www.flixml.com/detour.xml"  
    show="new"  
>
```

- ◆ Show tells the browser how to display the retrieved resource - not in terms of styling but in terms of relation to the linking element itself
 - ◆ Replace - like HTML does today
 - ◆ New - open up a new window
 - ◆ Embed - insert into the source document at the link's location



Anatomy of a Link - actuate

```
<reviewlink xml:link="simple"  
    href="http://www.flixml.com/detour.xml"  
    actuate="user"  
>
```

- ◆ Tells the browser what the “trigger” is to activate the link.
 - ◆ User - user has to take an explicit action (ie. Clicking)
 - ◆ Auto - no user action is required - it just happens
 - ◆ Consider using this with `show="embed"`



Anatomy of a Link - behavior

```
<reviewlink xml:link="simple"  
    href="http://www.flixml.com/detour.xml"  
    behavior="default"  
>
```

- ◆ This is another application-specific attribute whose value can be anything you want, since the action will be determined by the application and the browser.



Out-of-line Links

```
<referencelink xml:link="simple"  
    href="http://www.miscellanea.org"  
    inline=false  
    role="crossreference"  
    content-role="Trivia"  
/>
```

- ◆ A description of a target resource without providing an explicit way of getting there
- ◆ Consider a series of out-of-line links in an article which an intelligent browser could use to display a bibliography



Extended links

```
<localresource [local resource attributes] [link attributes] >
    <remotesource [remote resource1 attributes] />
    <remotesource [remote resource2 attributes] />
    <remotesource [remote resource3 attributes] />
</localresource>
```

- ◆ Allows for links that go from one resource to many resources, as well as links that “go both ways”
- ◆ Works by separating out the info on local resources from that of remote resources.



Extended Example

```
<distribexlink xml:link="extended"
  inline="true"
  title="Buy It Online!"
  <distriblink xml:link="locator"
    href="http://movie.reel.com/12345.html"
    title="... at Reel.com" />
  <distriblink xml:link="locator"
    href="http://www.moviesunlimited.com/"
    title="... at Movies Unlimited" />
  <distriblink xml:link="locator"
    href="http://www.netvideo.com/mm9876.html"
    title="... at MediaMart" />
  Buy a Copy
</distribexlink>
```



Extended Link Groups

```
<genrelinkgroup xml:link="group"
  steps="1"
  title="Noir Films"
  <genreflixinfo xml:link="document"
    href="shootpianoplayer.xml"
    title="Shoot the Piano Player" />
  <genreflixinfo xml:link="document"
    href="detour.xml"
    title="Detour" />
</ genrelinkgroup >
```

- ◆ Allows for separate files containing the extended links
- ◆ Allows multiple files to share a common set of extended links



XPointers

- ◆ XPointers are the next generation for URL's
- ◆ Addresses many of the shortcomings of URL's as we've discussed before



DTD's & XML Schema

putting meaning on structure



DTD's

- ◆ A way to describe the structure of an XML document such that a parser can do validation and an intelligent parser could do much more interesting things.
- ◆ Uses a syntax completely different from standard XML - and it's pretty ugly.
 - ◆ This is one reason why you'll find most XML documents on the web today do NOT include DTD's



An example

```
<!ELEMENT PRICE ( #PCDATA ) >
<!ELEMENT NUMBER ( #PCDATA ) >
<!ELEMENT PRODUCT ( #PCDATA ) >
<!ELEMENT ITEM ( PRICE | NUMBER | PRODUCT )* >
<!ELEMENT ORDERS ( ITEM )* >
<!ELEMENT DATE ( #PCDATA ) >
<!ELEMENT FIRSTNAME ( #PCDATA ) >
<!ELEMENT LASTNAME ( #PCDATA ) >
<!ELEMENT NAME ( LASTNAME | FIRSTNAME )* >
<!ELEMENT CUSTOMER ( NAME | DATE | ORDERS )* >
<!ELEMENT DOCUMENT ( CUSTOMERS )* >
```



But let's not linger on DTD's...

- ◆ Because they are not long for this world (YEAH!!)
- ◆ A number of companies have made proposals to replace DTD's, and as such the W3C has tasked a subgroup of the XML Coordination Group to decide what to do.
 - ◆ The subgroup is called the XML Schema Working Group, so the replacement is referred to either as XML Schema or just XSchema.



CSS2 & XSL

a little styling goes a long way



Separation of Design & Content

- ◆ XML serves to address the content, but we then need to find something to use to specify design (styling/layout)
 - ◆ Cascading Style Sheets (CSS level 2)
 - ◆ eXtended Style Sheets (XSL)
- ◆ Design should address not just the “web” and “computers” but many devices and forms
 - ◆ Including dynamic styling & style transformations



CSS

- ◆ Started in 1996, by the W3C, as the solution for going beyond the limited tag set of HTML for advanced styling features
 - ◆ Separate style info *physically* from content
 - ◆ Separate style info *syntactically* from content
- ◆ Current draft for CSS level 2 can be found at <http://www.w3.org/TR/REC-CSS2>



Example of CSS & XML

```
...  
<title role="main">Ms. 45</title>  
<title role="alt">Angel of Vengeance</title>  
<cast>  
    <leadingfemale>Zoe Tamerlis</leadingfemale>  
    <supportingmale>Steve Singer</supportingmale>  
</cast>  
  
...  
title[role="main"] { font-size: x-large }  
title[role="alt"] { font-size: x-large }  
leadingfemale { font-family: sans-serif; font-size: x-large }  
supportingmale {font-family: sans-serif; font-size: x-large }
```

Ms. 45

Angel of Vengeance

Zoe Tamerlis

Steve Singer



XSL

- ◆ A style language that looks like XML, so you feel right at home
- ◆ Derivative of DSSSL (SGML's styling language)
- ◆ Current draft for XSL can be found at <http://www.w3.org/TR/NOTE-XSL.html>



Example of XSL

```
<rule>
    <target-element type="plotsummary" />
    <DIV font-size="12pt" font-family="sans-serif" font-
style="italic">
        <children />
    </DIV>
</rule>
```



Cool features of XSL

- ◆ Element matching
 - ◆ By context
 - ◆ by wildcard
 - ◆ by attributes
- ◆ Actions
 - ◆ Somewhere between CGI and SSI
- ◆ Macros (named actions)
- ◆ Scripting



Applications for XML



MathML

- ◆ Markup language for describing a mathematical equation/formula
- ◆ See <http://www.w3.org/MarkUp/Math/>



CML

- ◆ Chemical Markup Language
 - ◆ Used to describe chemical compositions, molecular bonding issues, etc.



RDF

- ◆ Resource Description Framework
- ◆ Metadata (data about data) support for XML
- ◆ See <<http://www.w3.org/RDF/>>



SMIL

- ◆ Synchronized Multimedia Integration Language
- ◆ Standardized streaming multimedia format
- ◆ See
<<http://www.w3.org/AudioVideo/#SMIL>>



SVG

- ◆ Scalable Vector Graphics
- ◆ Vector graphics format that everyone can agree on
 - ◆ robust enough language and implementation which is compatible functionally with existing vector technology
 - ◆ You should be able to replace Postscript with SVG
- ◆ See <http://www.w3.org/Graphics/Activity>



PGML/SVG DEMO



Q & A

This presentation can be found online at

<http://www.lazerware.com/>

