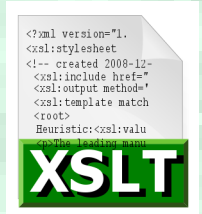




{JSON}



Lesson #10: XML/XSL/XSLT RSS Feeds JSON SEO/Monetization



M[💰]ONETIZE



Part 1: XML

- XML stands for EXtensible Markup Language.
- XML is a markup language much like HTML.
- XML was designed to carry data, not to display data.
- XML tags are not predefined. You must define your own tags.
- XML is not a replacement for HTML. HTML is about displaying information, while XML is about carrying information.

What is XML?

- XML documents don't do anything. It is just pure information wrapped in tags. Someone must write a piece of software to send, receive or display it.
- XML is nothing special. It is just plain text. Software that can handle plain text can also handle XML.
- XML separates data from HTML.

Why use XML?

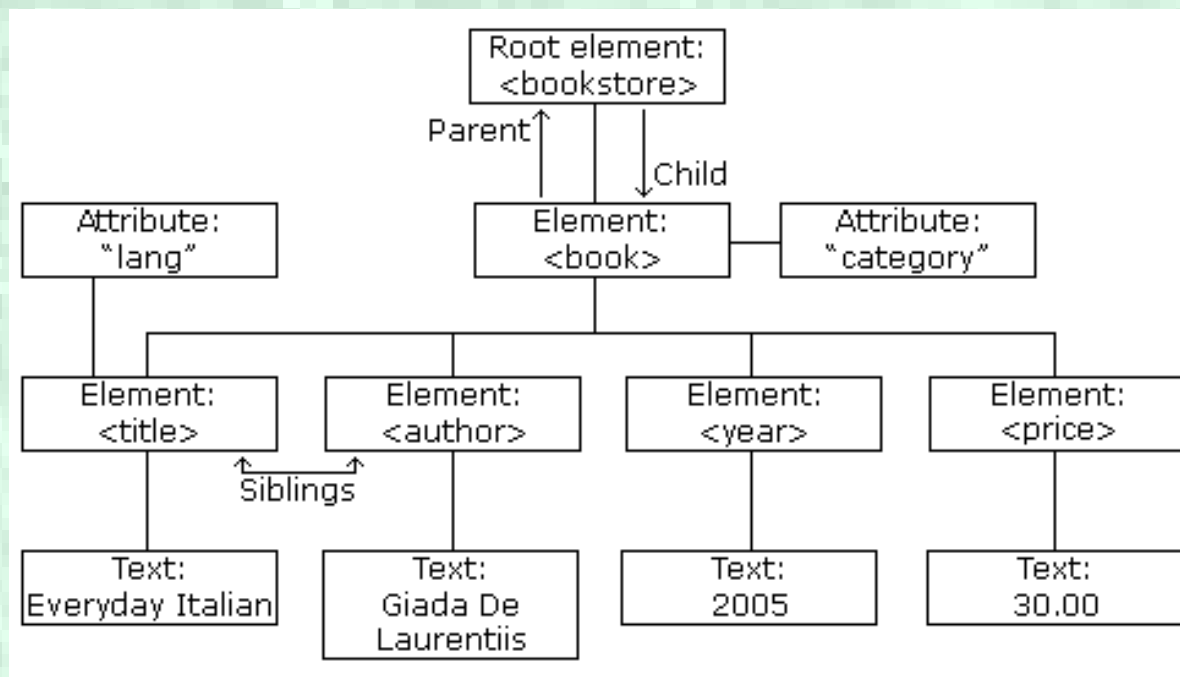
- With XML, data can be stored in separate XML files. This way you can concentrate on using HTML for layout and display, and be sure that changes in the underlying data will not require any changes to the HTML.
- With a few lines of JavaScript, you can read an external XML file and update the data content of your HTML.
- XML simplifies data sharing and transport because it is pure text format.

Why use XML?

- Since XML is independent of hardware, software and application, XML can make your data more available and useful.
- Different applications can access your data, not only in HTML pages, but also from XML data sources.
- With XML, your data can be available to all kinds of "reading machines" (Hand-held computers, voice machines, etc), and make it more accessible for people with disabilities.

XML structure

- XML documents form a tree structure that starts at "the root" and branches to "the leaves".



```
<bookstore>
  <book category="COOKING">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
  <book category="CHILDREN">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="WEB">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```

XML syntax

- All elements must have a closing tag.
- XML tags are case sensitive. With XML, the tag <Letter> is different from the tag <letter>. Opening and closing tags must be written with the same case.
- In XML, all elements must be properly nested within each other.
- XML documents must contain one element that is the parent of all other elements. This element is called the root element.

XML syntax

- In XML the attribute value must always be quoted.
- Some characters have a special meaning in XML. If you place a character like "<" inside an XML element, it will generate an error because the parser interprets it as the start of a new element. To avoid this error, replace the "<" character with an entity reference.

There are 5 predefined entity references in XML:

<	<	less than
>	>	greater than
&	&	ampersand
'	'	apostrophe
"	"	quotation mark

Note: Only the characters "<" and "&" are strictly illegal in XML. The greater than character is legal, but it is a good habit to replace it.

XML syntax

- The syntax for writing comments in XML is similar to that of HTML:
`<!-- This is a comment -->`
- Unlike HTML, the white-space in a document is not truncated in XML.
- XML elements must follow these naming rules:
Names can contain letters, numbers, and other characters; they cannot start with a number or punctuation character; they cannot start with the letters xml (or XML, or Xml, etc), they cannot contain spaces (avoid `.` `:` `-` in names).

XML attributes

- Attributes provide additional information about elements. Here are some examples:

```
<car colour="red">
```

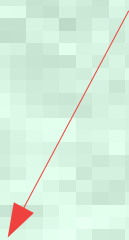
```
<car colour='yellow'>
```

```
<gangster name='Sammy "The Bull" Gravano'>
```

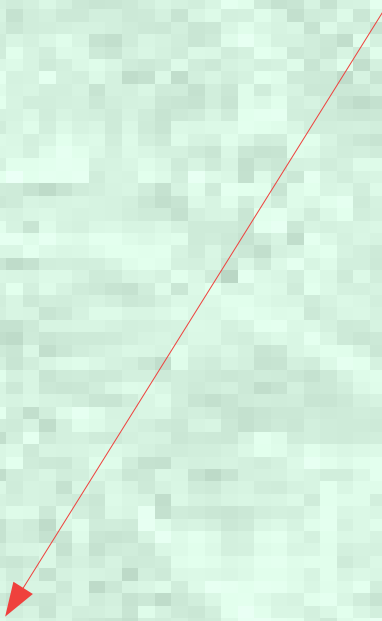
```
<gangster name="Sammy &quot;The Bull&quot; Gravano">
```

XML: attributes vs. elements

- You can use attributes or elements to describe information:



```
<car colour="red">  
  <make>Honda</make>  
  <make>Toyota</make>  
</car>
```



```
<car>  
  <colour>red</colour>  
  <make>Honda</make>  
  <make>Toyota</make>  
</car>
```

XML: attributes vs. elements

- You can use attributes or elements to describe information:

```
<note date="2008-01-10">  
  <to>Tove</to>  
  <from>Jani</from>  
</note>
```

```
<note>  
  <date>2008-01-10</date>  
  <to>Tove</to>  
  <from>Jani</from>  
</note>
```

```
<note>  
  <date>  
    <year>2008</year>  
    <month>01</month>  
    <day>10</day>  
  </date>  
  <to>Tove</to>  
  <from>Jani</from>  
</note>
```



XML: attributes vs. elements

Some things to consider when using attributes are:

- attributes cannot contain multiple values (elements can)
- attributes cannot contain tree structures (elements can)
- attributes are not easily expandable (for future changes)

```
<note day="10" month="01" year="2008"  
to="Tove" from="Jani" heading="Reminder"  
body="Don't forget me this weekend!">  
</note>
```



XML display

- By default, XML is displayed only in its raw form (source) in the browser.
- Unlike HTML, errors in XML code will be indicated.
- XML documents do not carry information about how to display the data. Since XML tags are "invented" by the author of the XML document, browsers do not know if a tag like <table> describes an HTML table or a dining table.

Ex: www.xmlfiles.com/examples/plant_catalog.xml

XML display with CSS

- With CSS (Cascading Style Sheets) you can add display information to an XML document.
- See this page for more information:
www.w3schools.com/xml/xml_display.asp
- Note: Formatting XML with CSS is not the most common method. W3C recommend using XSLT instead.
- We will revisit XML display again after seeing XSLT.

XML display with JavaScript

- With an XMLHttpRequest you can communicate with your server from inside a web page.
- You can update a web page with new data without reloading the page.
- You can request and receive new data from a server after the page has loaded.
- You can communicate with a server in the background.

XML display with JavaScript

- Creating an XMLHttpRequest object is done with one single line of JavaScript:
`var xmlhttp=new XMLHttpRequest()`
- Most browsers have a built-in XML parser to read and manipulate XML. The parser converts XML into a JavaScript accessible object (the XML DOM).
- See these pages:
w3schools.com/xml/xml_parser.asp
www.w3schools.com/xml/xml_dom.asp

Part 2: XSL and XSLT

- XSL stands for EXtensible Stylesheet Language, and is a style sheet language for XML documents.
- XSLT stands for XSL Transformations. XSLT is used to transform XML documents into other formats, like XHTML.
- The World Wide Web Consortium (W3C) started to develop XSL because there was a need for an XML-based Stylesheet Language.

CSS vs. XSL

- CSS = Style Sheets for HTML.
- HTML uses predefined tags, and the meaning of each tag is well understood.
- The <table> tag in HTML defines a table, and a browser knows how to display it.
- Adding styles to HTML elements are simple. Telling a browser to display an element in a special font or color, is easy too with CSS.

CSS vs. XSL

- XSL = Style Sheets for XML.
- XML does not use predefined tags (we can use any tag names we like), and therefore the meaning of each tag is not well understood.
- A <table> tag could mean an HTML table, a piece of furniture, or something else, and a browser does not know how to display it.
- XSL describes how the XML document should be displayed.

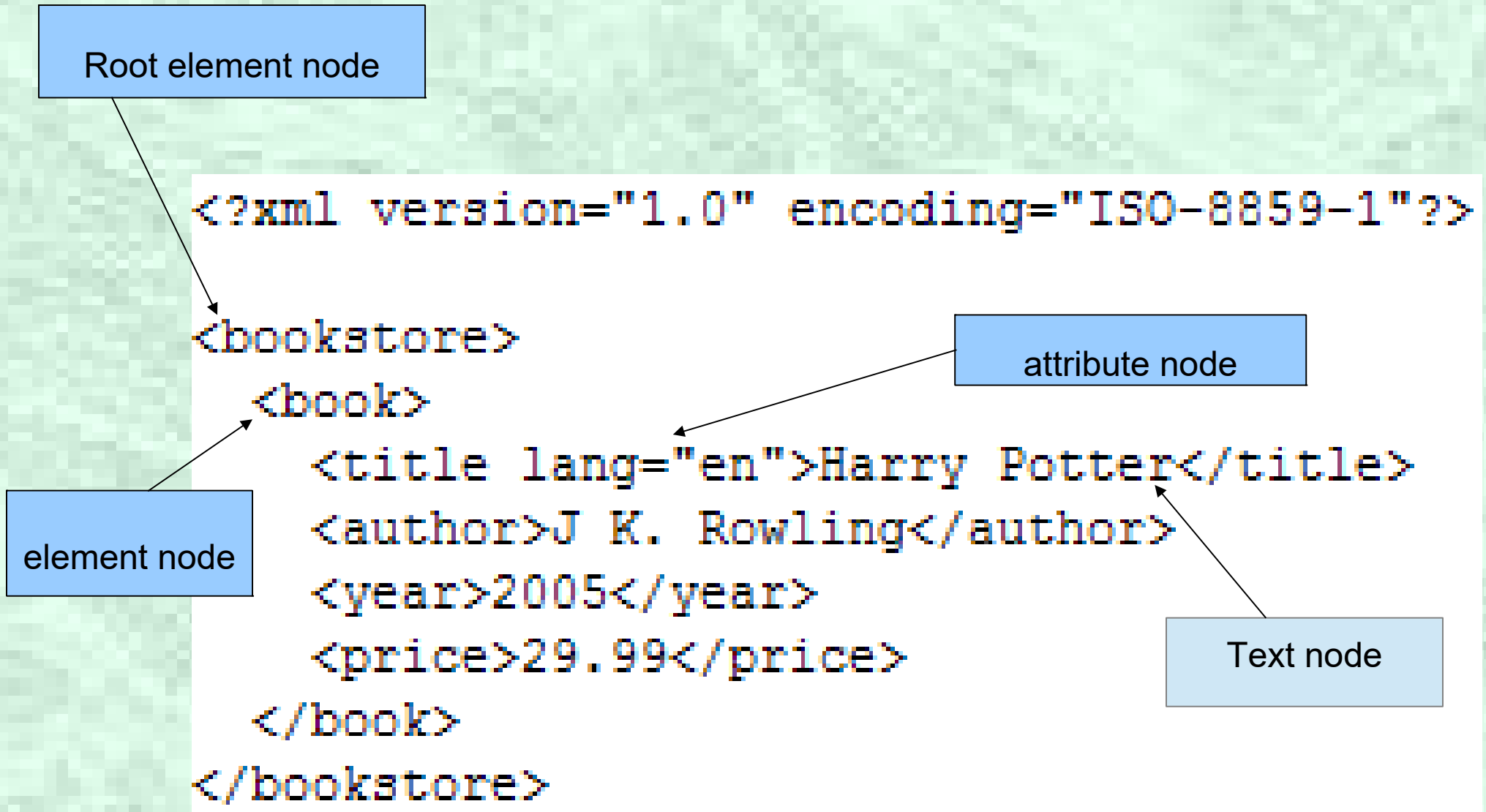
XSL consists of 3 parts

- XSLT: the language for transforming XML documents.
- XPath: the language for navigating in XML documents.
- XSL-FO: the language for formatting XML documents

XPath

- XPath is a syntax for defining parts of an XML document.
- It uses path expressions to navigate in XML documents.
- Contains a library of standard functions.
- Has 7 kinds of nodes: element, attribute, text, namespace, processing-instruction, comment, and document nodes.
- XML documents are treated as trees of nodes. The topmost element of the tree is called the root element.

XPath



XPath path expressions

- Nodes in XPath can be parents, children, siblings, ancestors and descendants.
- XPath uses path expressions to select nodes or node-sets in an XML document. The node is selected by following a path or steps.

Expression	Description
<i>nodename</i>	Selects all child nodes of the named node
/	Selects from the root node
//	Selects nodes in the document from the current node that match the selection no matter where they are
.	Selects the current node
..	Selects the parent of the current node
@	Selects attributes

XPath path expressions

- Some examples:

Path Expression	Result
bookstore	Selects all the child nodes of the bookstore element
/bookstore	Selects the root element bookstore Note: If the path starts with a slash (/) it always represents an absolute path to an element!
bookstore/book	Selects all book elements that are children of bookstore
//book	Selects all book elements no matter where they are in the document
bookstore//book	Selects all book elements that are descendant of the bookstore element, no matter where they are under the bookstore element
//@lang	Selects all attributes that are named lang

XPath predicates

- Predicates are used to find a specific node or a node that contains a specific value.
- Predicates are always embedded in square brackets.

Path Expression	Result
/bookstore/book[1]	Selects the first book element that is the child of the bookstore element. Note: IE5 and later has implemented that [0] should be the first node, but according to the W3C standard it should have been [1]!!
/bookstore/book[last()]	Selects the last book element that is the child of the bookstore element
/bookstore/book[last()-1]	Selects the last but one book element that is the child of the bookstore element
/bookstore/book[position()<3]	Selects the first two book elements that are children of the bookstore element
//title[@lang]	Selects all the title elements that have an attribute named lang
//title[@lang='eng']	Selects all the title elements that have an attribute named lang with a value of 'eng'
/bookstore/book[price>35.00]	Selects all the book elements of the bookstore element that have a price element with a value greater than 35.00
/bookstore/book[price>35.00]/title	Selects all the title elements of the book elements of the bookstore element that have a price element with a value greater than 35.00

XPath wildcards

- Wildcards are used to select unknown XML elements.

Wildcard	Description
*	Matches any element node
@*	Matches any attribute node
node()	Matches any node of any kind

Path Expression	Result
/bookstore/*	Selects all the child nodes of the bookstore element
//*	Selects all elements in the document
//title[@*]	Selects all title elements which have any attribute

XPath (several paths)

- By using the | operator in an XPath expression you can select several paths.

Path Expression	Result
<code>//book/title //book/price</code>	Selects all the title AND price elements of all book elements
<code>//title //price</code>	Selects all the title AND price elements in the document
<code>/bookstore/book/title //price</code>	Selects all the title elements of the book element of the bookstore element AND all the price elements in the document

XPath axes

- An axis defines a node-set relative to the current node.
- A location path can be absolute or relative.
- An absolute location path starts with a slash (/) and a relative location path does not. In both cases the location path consists of one or more steps, each separated by a slash.
- A step consists of an axis, a node-test and zero or more predicates.
`axisname::nodetest[predicate]`

XPath axes examples

- A few examples.

Example	Result
<code>child::book</code>	Selects all book nodes that are children of the current node
<code>attribute::lang</code>	Selects the lang attribute of the current node
<code>child::*</code>	Selects all children of the current node
<code>attribute::*</code>	Selects all attributes of the current node
<code>child::text()</code>	Selects all text child nodes of the current node
<code>child::node()</code>	Selects all child nodes of the current node
<code>descendant::book</code>	Selects all book descendants of the current node
<code>ancestor::book</code>	Selects all book ancestors of the current node
<code>ancestor-or-self::book</code>	Selects all book ancestors of the current node - and the current as well if it is a book node
<code>child::*/*child::price</code>	Selects all price grandchildren of the current node

XPath operators

- An XPath expression returns either a node-set, a string, a Boolean, or a number.

Operator	Description	Example	Return value
	Computes two node-sets	//book //cd	Returns a node-set with all book and cd elements
+	Addition	6 + 4	10
-	Subtraction	6 - 4	2
*	Multiplication	6 * 4	24
div	Division	8 div 4	2
=	Equal	price=9.80	true if price is 9.80 false if price is 9.90
!=	Not equal	price!=9.80	true if price is 9.90 false if price is 9.80
<	Less than	price<9.80	true if price is 9.00 false if price is 9.80
<=	Less than or equal to	price<=9.80	true if price is 9.00 false if price is 9.90
>	Greater than	price>9.80	true if price is 9.90 false if price is 9.80
>=	Greater than or equal to	price>=9.80	true if price is 9.90 false if price is 9.70
or	or	price=9.80 or price=9.70	true if price is 9.80 false if price is 9.50
and	and	price>9.00 and price<9.90	true if price is 9.80 false if price is 8.50
mod	Modulus (division remainder)	5 mod 2	1

What is XSLT?

- XSL Transformations.
- The most important part of XSL.
- Used to transform an XML document into another XML document, or another type of document that is recognized by a browser, like HTML and XHTML. Normally XSLT does this by transforming each XML element into an (X)HTML element.
- All major browsers support XSLT.

Starting XSLT

- How to transform XML into XHTML using XSLT?
- The root element that declares the document to be an XSL style sheet is `<xsl:stylesheet>` or `<xsl:transform>`

```
<xsl:transform version="1.0"  
xmlns:xsl="http://www.w3.org/1999/XSL/Tra  
nsform">
```

```
<xsl:template match="/">
```

Transformation Example

- We want to transform the following XML document ("cdcatalog.xml") into HTML.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<catalog>
  <cd>
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>
    <country>USA</country>
    <company>Columbia</company>
    <price>10.90</price>
    <year>1985</year>
  </cd>
  .
  .
</catalog>
```

See transformation example here: www.w3schools.com/xml/xsl_transformation.asp

Transformation Example

- Then you create an XSL Style Sheet ("cdcatalog.xsl") with a transformation template.
- Add the following line in your XML file:
`<?xml-stylesheet type="text/xsl" href="cdcatalog.xsl"?>`
- The `<xsl:template match="/">` element defines a template. The `match="/"` attribute associates the template with the root of the XML source document.

No data copied yet!

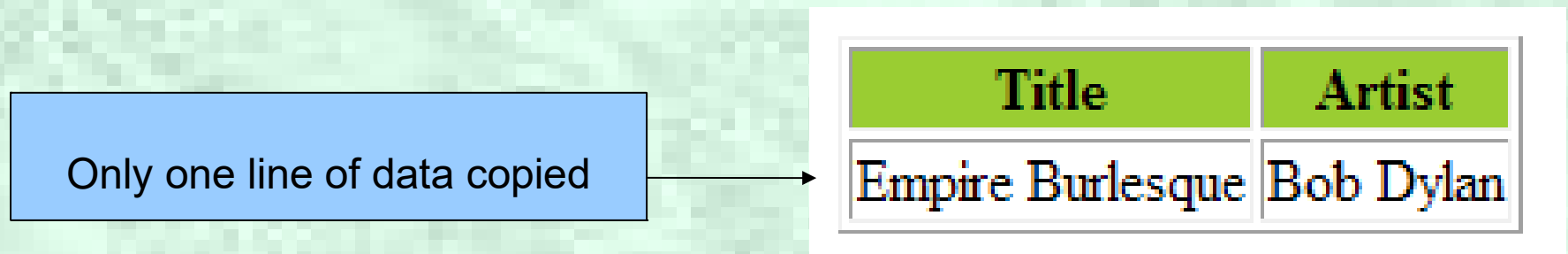
Title	Artist
.	.

Transformation Example

- The `<xsl:value-of>` element can be used to extract the value of an XML element and add it to the output stream of the transformation. The value of the `select` attribute is an XPath expression.

`<xsl:value-of select="catalog/cd/title"/>`

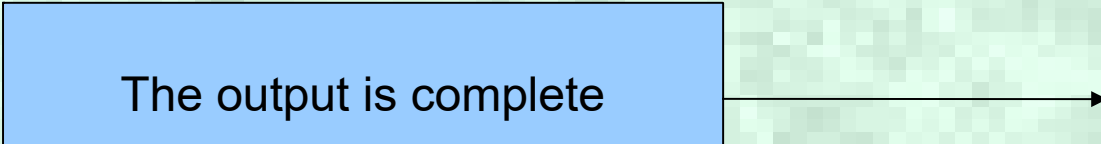
`<xsl:value-of select="catalog/cd/artist"/>`



Transformation Example

- The XSL `<xsl:for-each>` element can be used to select every XML element of a specified node-set.
- `<xsl:for-each select="catalog/cd">`
`<tr><td>`
 `<xsl:value-of select="title"/></td><td>`
 `<xsl:value-of select="artist"/>`
`</td></tr></xsl:for-each>`

The output is complete



A blue rectangular box containing the text "The output is complete" has a black arrow pointing from its right side to the top-left corner of the table below.

Title	Artist
Empire Burlesque	Bob Dylan
Hide your heart	Bonnie Tyler
Greatest Hits	Dolly Parton
Still got the blues	Gary Moore
Eros	Eros Ramazzotti
One night only	Bee Gees
Sylvias Mother	Dr.Hook
Maggie May	Rod Stewart
Romanza	Andrea Bocelli

Transformation Example

- We can also filter the output from the XML file by adding a criterion to the select attribute in the <xsl:for-each> element.

```
<xsl:for-each select="catalog/cd[artist='Bob Dylan']">
```

- To sort the output, simply add an <xsl:sort> element inside the <xsl:for-each> element in the XSL file.

```
<xsl:sort select="artist"/>
```

Transformation Example

- To put a conditional if test against the content of the XML file, add an `<xsl:if>` element to the XSL document.

`<xsl:if test="price > 10">`

- To insert a multiple conditional test against the XML file, add the `<xsl:choose>`, `<xsl:when>`, and `<xsl:otherwise>` elements to the XSL file.

Title	Artist
Empire Burlesque	Bob Dylan
Hide your heart	Bonnie Tyler
Greatest Hits	Dolly Parton
Still got the blues	Gary Moore
Eros	Eros Ramazzotti
One night only	Bee Gees
Sylvias Mother	Dr. Hook
Maggie May	Rod Stewart
Romanza	Andrea Bocelli

```
<tr>
  <td><xsl:value-of select="title"/></td>
  <xsl:choose>
    <xsl:when test="price > 10">
      <td bgcolor="#ff00ff">
        <xsl:value-of select="artist"/></td>
      </xsl:when>
    <xsl:otherwise>
      <td><xsl:value-of select="artist"/></td>
    </xsl:otherwise>
  </xsl:choose>
</tr>
```

Transformation Example

- The `<xsl:apply-templates>` element applies a template to the current element or to the current element's child nodes.

```
<xsl:template match="/">
  <html>
  <body>
  <h2>My CD Collection</h2>
  <xsl:apply-templates/>
  </body>
</html>
</xsl:template>

<xsl:template match="cd">
  <p>
    <xsl:apply-templates select="title"/>
    <xsl:apply-templates select="artist"/>
  </p>
</xsl:template>

<xsl:template match="title">
  Title: <span style="color:#ff0000">
    <xsl:value-of select="."/></span>
  <br />
</xsl:template>

<xsl:template match="artist">
  Artist: <span style="color:#00ff00">
    <xsl:value-of select="."/></span>
  <br />
</xsl:template>
```

Title: Empire Burlesque
Artist: Bob Dylan

Title: Hide your heart
Artist: Bonnie Tyler

Title: Greatest Hits
Artist: Dolly Parton

Part 3: JSON

- JSON: JavaScript Object Notation.
- JSON is a syntax for storing and exchanging data.
- JSON is an easier-to-use alternative to XML.

JSON vs. XML

JSON



```
{ "employees": [  
  
  { "firstName": "John",  
    "lastName": "Doe"},  
  
  { "firstName": "Anna",  
    "lastName": "Smith"},  
  
  { "firstName": "Peter",  
    "lastName": "Jones"}  
] }
```

XML



```
<employees>  
  <employee>  
    <firstName>John</firstName>  
    <lastName>Doe</lastName>  
  </employee>  
  <employee>  
    <firstName>Anna</firstName>  
    <lastName>Smith</lastName>  
  </employee>  
  <employee>  
    <firstName>Peter</firstName>  
    <lastName>Jones</lastName>  
  </employee>  
</employees>
```

What Is JSON?

- The JSON format is syntactically identical to the code for creating JavaScript objects.
- Because of this similarity, instead of using a parser (like XML does), a JavaScript program can use standard JavaScript functions to convert JSON data into native JavaScript objects.

JSON Is...

Like XML Because:

- Both JSON and XML are "self describing" (human readable).
- Both JSON and XML are hierarchical (values within values).
- Both JSON and XML can be parsed and used by lots of programming languages.
- Both JSON and XML can be fetched with an XMLHttpRequest.

Unlike XML Because:

- JSON doesn't use end tags.
- JSON is shorter.
- JSON is quicker to read and write.
- JSON can use arrays.

JSON Tutorials

JSON Syntax

[w3schools.com/js/js_json_syntax.asp](https://www.w3schools.com/js/js_json_syntax.asp)

Create web page from JSON (Part 1)

[w3schools.com/js/js_json_eval.asp](https://www.w3schools.com/js/js_json_eval.asp)

Create web page from JSON (Part 2)

[w3schools.com/js/js_json_http.asp](https://www.w3schools.com/js/js_json_http.asp)

Part 4: RSS Feeds

- RSS is a family of Web feed formats used to publish frequently updated works such as blog entries, news headlines, audio, and video, in a standardized format. An RSS document (which is called a feed or channel) includes full or summarized text, plus metadata such as publishing dates and authorship.
- RSS feeds benefit publishers by letting them syndicate content automatically.

What are RSS Feeds?

- RSS feeds can be read using software called an RSS reader or aggregator, which can be web-based or desktop-based. A standardized XML file format allows the information to be published once and viewed by many different programs.
- The user subscribes to a feed by entering the feed's URI into the reader or by clicking an RSS icon in a browser that initiates the subscription process. The RSS reader checks the user's subscribed feeds regularly for new work, downloads any updates that it finds, and provides a user interface to monitor and read the feeds.

Usage of RSS Feeds

- Everyday more and more websites, news services and blogs are adding RSS content. RSS is a method of syndicating content.
- RSS feed creators provide content without forcing it on consumers.
- RSS feeds contain what are referred to as "items". The items are usually connected in some way and contain a common theme or other similarity.
- Making RSS feeds is extremely easy.

Making an RSS Feed

- First, you create an item:

```
<item>
```

```
<title>CPS530 Final Exam</title>
```

```
<description>The CPS530 final exam may  
consist of 100 multiple choice  
questions.</description>
```

```
<link>http://cps530.scs.ryerson.ca</link>
```

```
</item>
```

Making an RSS Feed

- Of course, most of the time, you have multiple items:

```
<item>
```

```
<title>CPS530 Final Exam</title>
```

```
<description>The CPS530 final exam may consist of 100  
multiple choice questions.</description>
```

```
<link>http://cps530.scs.ryerson.ca</link>
```

```
</item>
```

```
<item>
```

```
<title>Lesson on RSS Feeds</title>
```

```
<description>Lesson #9 has for topic RSS  
feeds.</description>
```

```
<link>http://cps530.scs.ryerson.ca</link>
```

```
</item>
```

Making an RSS Feed

- The next step is to place all your items in a channel:

```
<channel><title>Dr. Hamelin's Channel</title>
<description>Welcome to my RSS Feeds</description>
<link>http://www.cs.ryerson.ca/dhamelin</link>
<item>
<title>CPS530 Final Exam</title>
<description>The CPS530 final exam may consist of 100
  multiple choice questions.</description>
<link>http://cps530.scs.ryerson.ca</link>
</item>
<item>
<title>Lesson on RSS Feeds</title>
<description>Lesson #9 has for topic RSS
  feeds.</description>
<link>http://cps530.scs.ryerson.ca</link>
</item></channel>
```

Making an RSS Feed

- Finally you will need to designate the file by indicating it is an XML file by inserting xml and rss defining tags at the beginning and </rss> at the very end.

```
<?xml version="1.0"?>
```

```
<rss version="2.0">
```

```
<channel><title>Dr. Hamelin's Channel</title>
```

```
<description>Welcome to my RSS Feeds</description>
```

```
<link>http://www.cs.ryerson.ca/dhamelin</link>
```

```
<item>
```

```
  <title>CPS 530 Final Exam</title>
```

```
  <description>The CPS530 final exam may consist of 100 multiple choice  
  questions.</description>
```

```
  <link>http://cps530.scs.ryerson.ca</link></item>
```

```
<item>
```

```
  <title>Lesson on RSS Feeds</title>
```

```
  <description>Lesson #9 has for topic RSS feeds.</description>
```

```
  <link>http://cps530.scs.ryerson.ca</link>
```

```
</item>
```

```
</channel>
```

```
</rss>
```

Viewing the RSS Feed

- Just save your feed with a .xml extension and upload it to your server. Appearance can vary depending on the aggregator. Validate your feed at feedvalidator.org. See course website for link to RSS 2.0 specification.
- See it in action:
cps530.scs.ryerson.ca/rssfeed01.xml

Dr. Hamelin's Channel

Welcome to my RSS Feeds
[CPS 530 Final Exam](#)
The CPS530 final exam may consist of 100 multiple choice questions.
[Lesson on RSS Feeds](#)
Lesson #11 has for topic RSS feeds.

Dr. Hamelin's Channel

DR. HAMELIN'S CHANNEL now
CPS 530 Final Exam
The CPS530 final exam may consist of 100 multiple choice questions.

Dr. Hamelin's Channel

DR. HAMELIN'S CHANNEL now
Lesson on RSS Feeds
Lesson #11 has for topic RSS feeds.

See RSS documentation: www.w3schools.com/xml/xml_rss.asp

Part 5: Search Engine Optimization

- SEO is the practice of improving and promoting a website in order to increase the number of visitors the site receives from search engines.
- The majority of web traffic is driven by the major search engines, Google, Bing, and Yahoo!. Although social media and other types of traffic can generate more visits to your website, search engines are still the primary method of navigation for most Internet users.

Search Engine Optimization

- SEO techniques can be classified into two broad categories: techniques that search engines recommend as part of good design (white hat SEO) and those techniques of which search engines do not approve (black hat SEO).
- White hats tend to produce results that last a long time (real content), whereas black hats anticipate that their sites may eventually be banned either temporarily or permanently once the search engines discover what they are doing (fake or deceptive content).

How to Improve Search Engine Rankings

Note: Not necessarily in order of importance.

1. Add Google Analytics to each page

Google Analytics is a free analytics software package that can provide you with a range of critical data about your site and how it is performing in the search engines.

Sign up also for Google Webmaster Tools, which also have a ton of information that you can use to learn more about your keywords and web pages.

Knowledge is power!

How to Improve Search Engine Rankings

2. Make each page unique

Google ranks the relevance of each website according to the content it contains, and is always seeking relevant content not contained anywhere else on the Internet. This means that the content of every page needs to be completely different not just from any other site on the Web, but also any other page on the same site.

How to Improve Search Engine Rankings

3. Use meta description tags

Meta description tags are used to differentiate web pages (although not as much as title tags) so you also need to be careful to describe each page differently to avoid any duplicate content issues.

Make sure to limit the <meta> description tags to 160 characters in length, including spaces (Should be at least 70 characters though).

How to Improve Search Engine Rankings

4. Remove repetitive wording from the website

Unique content is vital to the success of any site's SEO. Avoid including information such as copyright text, contact details and company slogans on every page of the site. If there is not enough unique content on every page then you run the risk of your site being penalised for duplicate content. That's why it's important to remove such repetitive wording from the website layout so that the true informational content of the site is not diluted in any way.

How to Improve Search Engine Rankings

5. Improve your content structure

Use <h1> to <h6> tags to define a structure to your website.

6. Avoid low-quality link sites

The quality of the pages that link to you is more important than the number. Never buy traffic from shady sources!

How to Improve Search Engine Rankings

7. Create effective <title> tags

Make sure your title is clear and contains your most important keywords. Be sure that each page has a unique title. Should be between 10 and 70 characters.

8. Use alt attributes for all images

It is important for validation and most important for SEO. Never forget the alt attribute.

How to Improve Search Engine Rankings

9. Optimize your text/html ratio to at least 15%

A ratio between 25% and 70% is ideal. When it goes beyond that, the page might run the risk of being considered spam. As long as the content is relevant and gives essential information, it is a plus to have more of it.

10. Get more indexed pages

A site with many pages has a better chance of being visible and known.

How to Improve Search Engine Rankings

11. Do not have too many links on your page

Limit the number of links to 200 per page. Use **nofollow** to optimize the juice that you want to pass to each link. Of course, avoid broken links at all cost!

12. Improve your number of backlinks

Backlinks are links that point to your website from other websites. They are like letters of recommendation for your site. Since this factor is crucial to SEO, you should have a strategy to improve the quantity and quality of backlinks.

How to Improve Search Engine Rankings

13. Make sure with and without www redirects to the same page

Redirecting requests from a non-preferred domain is important because search engines consider URLs with and without "www" as two different websites.

14. Have a robots.txt file

A robots.txt file allows you to restrict the access of search engine robots that crawl the web and it can prevent these robots from accessing specific directories and pages. It also specifies where the XML sitemap file is located.

How to Improve Search Engine Rankings

15. Have a sitemap.xml file

A sitemap lists URLs that are available for crawling and can include additional information like your site's latest updates, frequency of changes and importance of the URLs. This allows search engines to crawl the site more intelligently.

Generate an XML sitemap for your website and submit it to both Google Webmaster Tools and Bing Webmaster Tools. It is also good practice to specify your sitemap's location in your robots.txt file.

How to Improve Search Engine Rankings

16. Have user-friendly urls

Have URLs that do not contain query strings if possible. Clean URLs are not only SEO-friendly but are also important for usability. Avoid underscores as well in the URL (hyphens are better than underscores for URLs).

17. Have an old establish domain name.

The older the domain, the higher the rank. Also a later expiry date will encourage a higher ranking.

How to Improve Search Engine Rankings

18. Start a blog!

While publishing your content on other sites might be a good strategy, publishing it on your own site garners more benefits. Starting a blog is a great way to boost your SEO and attract visitors. A comment widget is also a good idea (think Facebook or Disqus).

19. Think of mobile devices!

The number of people using the Mobile Web is huge; over 75 percent of consumers have access to smartphones. Your website should look nice on the most popular mobile devices and render fast too!

How to Improve Search Engine Rankings

20. Have a favicon and an og image

Make sure your favicon is consistent with your brand. The og image is a banner that advertises you on social network.

21. Have a 404 error page

Provide visitors with a beautiful and helpful 404 Error Page to increase user retention.

How to Improve Search Engine Rankings

22. Place the most important content above the fold line

23. Keep your web page below 300kb (the average on the web is 320kb)

Site speed is becoming an important factor for ranking high in Google search results and enriching the user experience.

How to Improve Search Engine Rankings

24. Make your website known on social networks

Place share buttons for the major social networks. Have Facebook, Twitter, and Google+ pages. AddThis is a good free service.

26. Get local

Advertise outside the web at large, and be known locally.

27. Enable GZIP compression on your web server

GZIP compression is widely supported and reduces the load time of a web page.

How to Improve Search Engine Rankings

28. Update your content constantly!

Keep your content fresh.

29. Check your site's progress

Use tools like woorank, iwebchk or nibbler.

More info here:

<https://www.link-assistant.com/news/html-tags-for-seo.html>

Website Monetization

- Website monetization is the process of converting existing traffic being sent to a particular website into revenue. The most popular ways of monetizing a website are by implementing Pay per click (PPC), Cost per impression (CPI/CPM), and CPA (cost per action) advertising. Various ad networks facilitate a webmaster in placing advertisements on pages of the website to benefit from the traffic the site is experiencing.

Website Monetization

- Pay per click (also called Cost per click) is a marketing strategy put in place by search engines and various Advertising networks, where an advert, usually targeted by keywords or general topic, is placed on a relevant website. The advertiser then pays for every click that is made on the advert.
- Cost per impression (also called Cost per mille) is a strategy where an advert is placed on a relevant website, usually targeted to the content sector of that site. The advertiser then pays for every time the advert is displayed to a user.

Website Monetization

- Affiliate programs are another popular way of monetizing existing website traffic. By joining a business' affiliate program, any searches for products within that business' catalog may earn affiliates a commission on each sale that was originally referred through their website.
- Banner advertising consists of placing a graphical banner advertisement on a webpage. The role of this banner is to catch the eye of incoming traffic to the page, enticing readers to click the advertisement.

Website Monetization

- Some recommended ways:
- Google AdSense: The king of all monetization programs. Highest payouts but stringent rules for admission and continuation.
- Commission Junction: Probably the best source for affiliate programs.
- Amazon affiliate program: Good reliable program.
- And the AdSense alternatives...

Monetizing with Adsense Alternatives

- Adsense alternatives are easier to get approved for but get lower revenue.
- Adsense is strictly CPC but other programs offer CPM and CPA (cost per action: sales, leads).
- Adsense offer only banners but other programs also offer pop-unders, interstitials, sliders, push notifications, and video pre-rolls.

Some programs have low minimum payments, an advantage for small sites.

End of Lesson