

Web Programming/Scripting: XML

Dr. Michele Weigle
Department of Computer Science
Old Dominion University
mweigle@cs.odu.edu

<http://www.cs.odu.edu/~mweigle/CS312-F11/>

1

XML

◆ What is XML?

◆ XML Syntax

◆ Viewing XML

◆ XML and JavaScript

◆ XML in Real Life

<?xml?>

<xml />

What is XML?

- ◆ eXtensible Markup Language
 - » markup language, like HTML
- ◆ Designed to transport and store data
 - » unlike HTML (designed to display data)
- ◆ XML tags are user-defined
- ◆ XML doesn't *do* anything
 - » provides a way to structure and store information

3

XML Example

```
<person>  
  <name>Richard Smith</name>  
  <address>2004 Lakewood Drive  
    Norfolk, VA 23556</address>  
</person>
```

- ◆ XML is self-descriptive
 - » example describes a person, their name and address
 - » just identifies the structure of the *data*, not how it might be displayed
- ◆ XML is plain-text
- ◆ XML has no pre-defined tags

4

XML and HTML

- ◆ XML separates data from HTML
 - » separates data from how it is displayed
- ◆ Example: Create a webpage to display data that's updated frequently
 - » with HTML only
 - ❖ edit HTML each time data changes
 - » with HTML and XML
 - ❖ create HTML layout once
 - ❖ only need to update XML with new data
 - » We'll use JavaScript later to demonstrate this

5

More Benefits of XML

- ◆ Simplifies data sharing
 - » XML is plain-text, so no complicated, proprietary format to parse (software and hardware independent)
 - » Allows for the creation of data that different applications can share
- ◆ Simplifies data transport
 - » XML can be used to exchange information between incompatible systems
- ◆ Simplifies platform changes
 - » XML is plain-text, so software or hardware changes won't affect the data

6

XML

◆ What is XML?

◆ XML Syntax

◆ Viewing XML

◆ XML and JavaScript

◆ XML in Real Life

<?xml?>

<xml />

Reference for today's material: <http://www.w3schools.com/xml/>

7

XML Document Structure

`<?xml version="1.0" encoding="ISO-8859-1"?>` *XML declaration*

`<person>` *root element*

`<name>Richard Smith</name>`

`<address>2004 Lakewood Drive` *child elements*

`Norfolk, VA 23556</address>`

`</person>` *end of root element*

◆ XML declaration

» version and encoding

◆ Root element

◆ Child elements

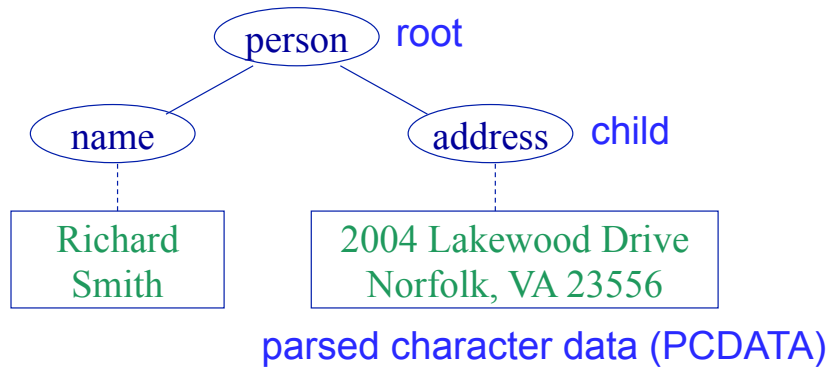
» child elements can have their own child elements

◆ End of root element

8

XML Tree

```
<person>
  <name>Richard Smith</name>
  <address>2004 Lakewood Drive
    Norfolk, VA 23556</address>
</person>
```



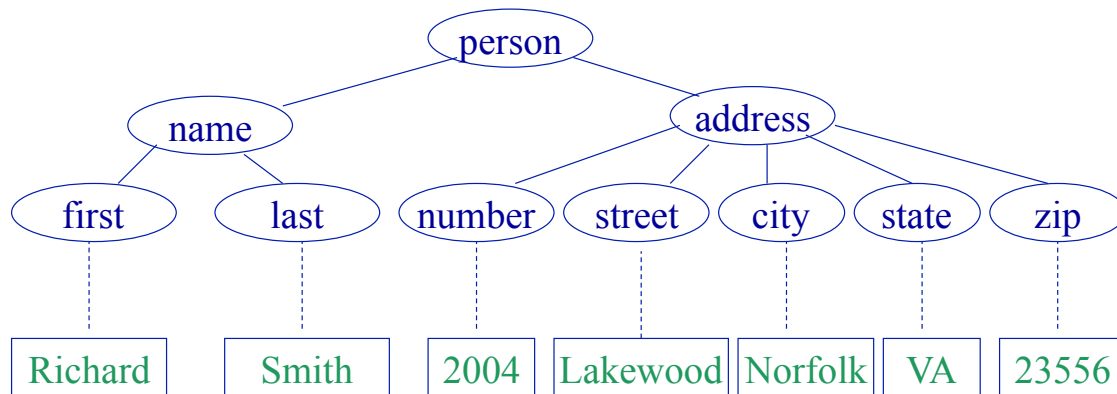
9

Another Representation

```
<person>
  <name> <first>Richard</first>
    <last>Smith</last>
  </name>
  <address> <number>2004</number>
    <street>Lakewood Drive</street>
    <city>Norfolk</city>
    <state>VA</state>
    <zip>23556</zip>
  </address>
</person>
```

10

Corresponding Tree Structure



11

XML Syntax Rules

- ◆ Similar to XHTML syntax (XHTML is a combination of HTML and XML)
 - » all elements must have a closing tag
 - » tags are case-sensitive
 - ❖ <name> is different than <Name>
 - » tags must be properly nested
 - ❖ <address><street>Lakewood</street></address>
 - » attribute values must be quoted
 - ❖ <address type="home"> or <address type='home'>
- ◆ All XML documents must have a root element
- ◆ Comments are the same as in HTML
- ◆ Whitespace is preserved (unlike HTML)

12

XML Elements

- ◆ Everything from the element's start tag to its end tag, inclusive
- ◆ Can contain other elements, simple text, or a mixture
- ◆ Can have attributes

<address> has element contents

<state> has text content

```
<person>
  <name> <first>Richard</first>
           <last>Smith</last>
  </name>
  <address> <number>2004</number>
            <street>Lakewood Drive</street>
            <city>Norfolk</city>
            <state>VA</state>
            <zip>23556</zip>
  </address>
</person>
```

13

XML Element Naming Rules

- ◆ Names can contain letters, numbers, or other characters
- ◆ Names must not start with a number or punctuation
- ◆ Names must not start with the letters 'xml'
- ◆ Names cannot contain spaces
- ◆ There are no reserved words
 - » any name can be used

14

Good Naming Practices


- ◆ Make names descriptive
- ◆ Names should be short and simple
- ◆ Avoid '-' characters
 - » some software may interpret as minus
- ◆ Avoid '.' characters
 - » some software may interpret as properties
- ◆ Avoid ':' characters
 - » colons are reserved in XML

15


Attributes vs. Elements

- ◆ Attributes
 - » best used for *meta-data* (data about data)
 - » ex: ID numbers
- ◆ Elements
 - » best used for the any part of the data itself

```
<phone type="mobile">  
757-123-4567  
</phone>
```



```
<phone>  
  <mobile>757-123-4567</mobile>  
</phone>
```



16

Checking XML Syntax

- ◆ http://www.w3schools.com/xml/xml_validator.asp
 - » has a tool where you can provide the URL of an XML file and it will check that the XML is "well-formed"
- ◆ You'll need to copy/paste your XML code into the form. URLs won't work.
 - » most servers/browsers won't allow scripts to access XML code that's not local to the script for security reasons

17

Specifying XML Structure

DTD
(document type definitions)

```
<!DOCTYPE person [  
  <!ELEMENT person (name, address)>  
  <!ELEMENT name (#PCDATA)>  
  <!ELEMENT address (#PCDATA)>  

```

```
<xs:element name="person">  
  
  <xs:complexType>  
    <xs:sequence>  
      <xs:element name="name" type="xs:string"/>  
      <xs:element name="address" type="xs:string"/>  
    </xs:sequence>  
  </xs:complexType>  
</xs:element>
```

XML Schema

18

XML

- ◆ What is XML?

<?xml?>

- ◆ XML Syntax

- ◆ Viewing XML

- ◆ XML and JavaScript

<xml />

- ◆ XML in Real Life

Reference for today's material: <http://www.w3schools.com/xml/>

19

Viewing XML Files

- ◆ Because XML does not describe how to format the data, browsers will display XML files without much formatting
- ◆ root and child elements will be color-coded
- ◆ '+' or '-' will be displayed to the left of each element
 - » click on the '+' to expand it
 - » click on the '-' to collapse it
- ◆ To see the raw XML source, use the browser's 'View Source' option.

20

Example XML Files

- ◆ <http://www.w3schools.com/xml/note.xml>
 - » sample XML file
- ◆ http://www.w3schools.com/xml/note_error.xml
 - » XML file with an error
- ◆ http://www.w3schools.com/xml/cd_catalog.xml
 - » CD catalog
- ◆ <http://www.w3schools.com/xml/simple.xml>
 - » restaurant menu

21

Styling XML with CSS

- ◆ Include a line in the XML file to load the CSS

```
<?xml-stylesheet type="text/css" href="filename.css"?>
```
- ◆ The CSS tags should be the same as the XML element names
 - » http://www.w3schools.com/xml/cd_catalog.txt
- ◆ Example XML formatted with the CSS
 - » http://www.w3schools.com/xml/cd_catalog_with_css.xml
- ◆ The standards folks (W3C) recommend using XSLT instead of CSS for XML styling

22

Styling XML with XSLT

- ◆ eXtensible Stylesheet Language Transformations
 - » more sophisticated than CSS
 - » transforms XML into HTML
- ◆ Include line in the XML to load the XSLT
 - `<?xml-stylesheet type="text/xsl" href="filename.xsl"?>`

23

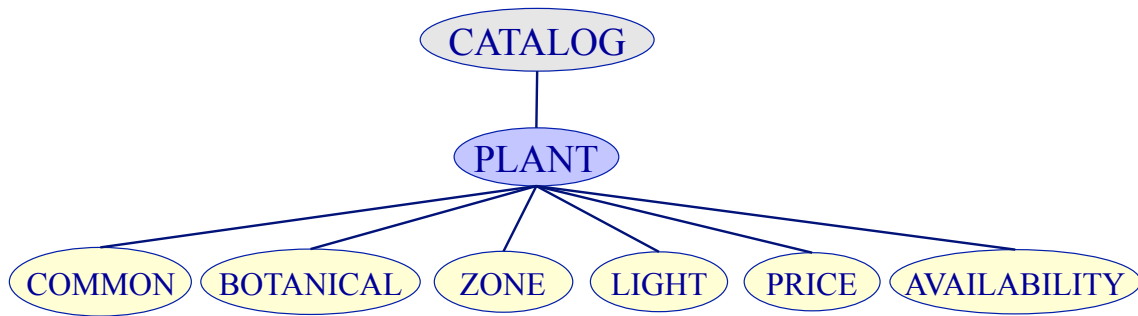
XSLT

- ◆ Special tags in HTML that reference the XML elements
- ◆ Begins with version information
 - `<html xsl:version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns="http://www.w3.org/1999/xhtml">`
- ◆ Select each element with
 - » `<xsl:for-each select="root/element"> ... </xsl:for-each>`
- ◆ Inside the for-each loop, display sub-elements with
 - » `<xsl:value-of select="element"/>`

24

Example: Plant Catalog

- ◆ http://www.cs.odu.edu/~mweigle/cs312/xml/plant_catalog.xml
- ◆ <http://www.cs.odu.edu/~mweigle/cs312/xml/plants.xml>



25

XML

- ◆ What is XML?

<?xml?>

- ◆ XML Syntax

- ◆ Viewing XML

- ◆ XML and JavaScript

<xml />

- ◆ XML in Real Life

Reference for today's material: <http://www.w3schools.com/xml/>

26

XMLHttpRequest (XHR) Object

- ◆ Provides a way to communicate with a server after a web page has loaded
- ◆ Allows for
 - » updating a web page with new data without reloading the page
 - » requesting data from a server after the page has loaded
 - » receiving data from a server after the page has loaded
 - » sending data to a server in the background
- ◆ xmlHelperFns.js
 - » sendXHR(url)
 - » stateChange()

27

Setting Up XMLHttpRequest (XHR)

sendXHR(url)

```
function sendXHR(url)
{
    if (xmlhttp == null) {
        xmlhttp = new XMLHttpRequest();
    }
    if (xmlhttp == null) {
        alert("Your browser does not support XMLHttpRequest.");
        return;
    }
    xmlhttp.onreadystatechange = stateChange;
    xmlhttp.open("GET", url, true);
    xmlhttp.send(null);
}
```

requests the document found at the given URL

xmlhttp is the variable that allows us to make the request

stateChange function is called when the state of the XMLHttpRequest object changes

open() sets up an HTTP request for the URL

send() sends the request to the server

28

Setting Up XMLHttpRequest

stateChange()

```
function stateChange()
{
  if (xmlhttp.readyState == 4) {
    if (xmlhttp.status == 200) {
      // HTTP response code (200 is OK)
      doStuff(); // you must write this function
    } else {
      alert ("Problem retrieving data: " + xmlhttp.statusText);
    }
  }
}
```

readyState

0 - request is not initialized
1 - request has been set up
2 - request has been sent
3 - request is in process
4 - *request is complete*

<http://www.cs.odu.edu/~mweigle/cs312/xml/xmlHelperFns.js>

29

XMLHttpRequest

Example 1

- ◆ Update data on page without reload
 - » use functions in xmlHelperFns.js
 - » write doStuff() function

```
<script src=http://www.cs.odu.edu/~mweigle/cs312/xml/xmlHelperFns.js type="text/
javascript"></script>
<script type="text/javascript">
<!--
var xmlhttp;    // this MUST be declared as global variable

function doStuff()
{
  document.getElementById('T1').innerHTML = xmlhttp.responseText;
}
// -->
</script>
```

<http://www.cs.odu.edu/~mweigle/cs312/xml/text-noreload.html>

30

XMLHttpRequest

Example 1

- ◆ Write HTML to call sendXHR() for one file when the page loads
- ◆ Create div for text to be written to
- ◆ Setup button to call sendXHR() for a new file when pressed

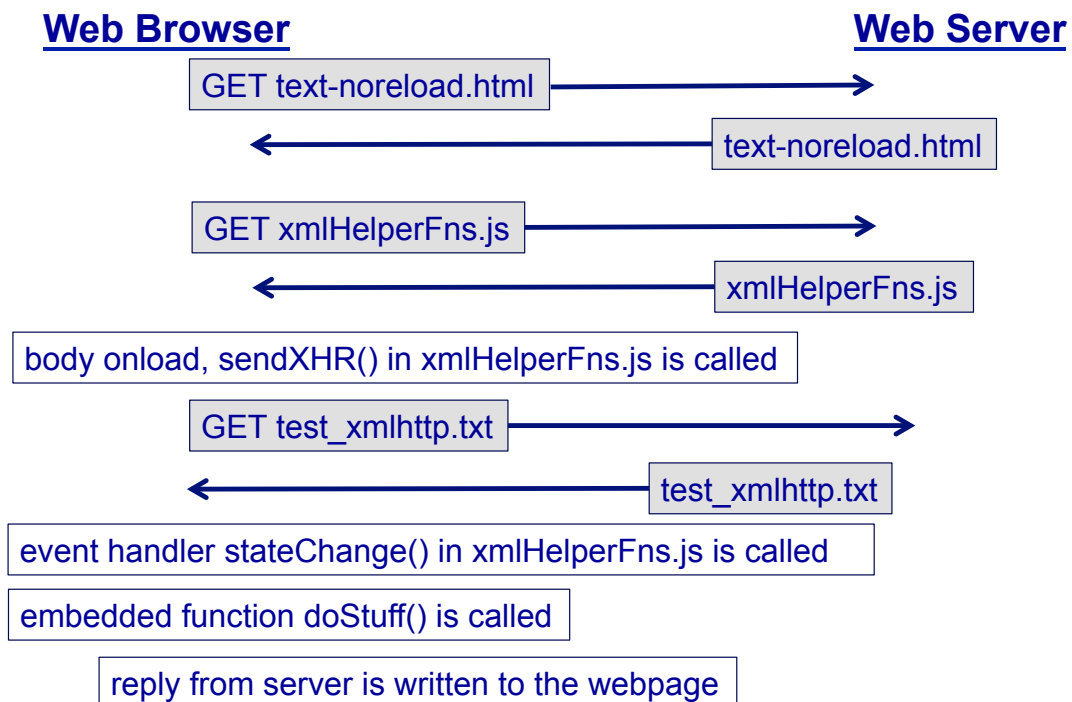
```
<body onload="sendXHR('test_xmlhttp.txt')">  
  
<div id="T1" style="border:1px solid black; height:50; width:300; padding:10"></div>  
  
<p><button onclick="sendXHR ('test_xmlhttp2.txt')">Click</button> </p>  
</body>
```

<http://www.cs.odu.edu/~mweigle/cs312/xml/text-noreload.html>

31

XMLHttpRequest

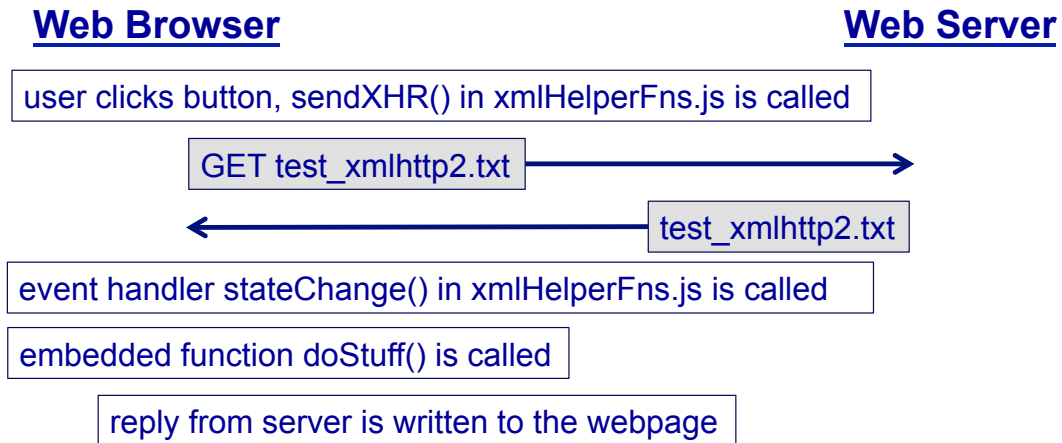
Example 1 Data Flow



32

XMLHttpRequest

Example 1 Data Flow



33

XMLHttpRequest

Example 2

- ◆ Display HTTP response headers
 - » only need to change the doStuff() function
 - » doesn't matter what file is requested

```
document.getElementById('p1').innerHTML = xmlhttp.getAllResponseHeaders();
```

```
<p>getAllResponseHeaders() returns the HTTP response header upon loading a
URL.</p>
<p id="p1" style="white-space: pre"></p> white-space: pre doesn't work in IE

<button onclick="sendXHR ('test_xmlhttp.txt')">Get Headers</button>
</body>
```

<http://www.cs.odu.edu/~mweigle/cs312/xml/get-header.html>

34

Response Headers

- ◆ To get a particular response header, use `getResponseHeader('OptionName')`
- ◆ Examples:
 - » `xmlhttp.getResponseHeader('Last-Modified');`
 - ❖ Wed, 05 Nov 2008 18:49:18 GMT
 - » `xmlhttp.getResponseHeader('Content-Length');`
 - ❖ 93

35

Parsing XML with XHR and JavaScript

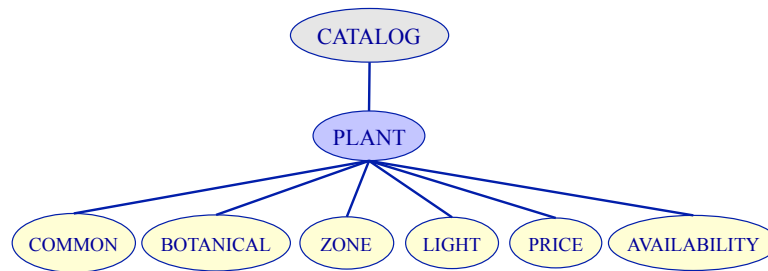
- ◆ Most modern browsers have a built-in XML parser
 - » reads XML into memory
 - ❖ using XMLHttpRequest
 - » converts it into an XML DOM (document object model) object that can be accessed with JavaScript
- ◆ The XML and the HTML/JavaScript must be on the same web server.

36

Parsing XML

XML DOM

- ◆ XML Document Object Model
 - » defines a standard way for accessing and manipulating XML documents
- ◆ Elements are accessed through a tree-structure

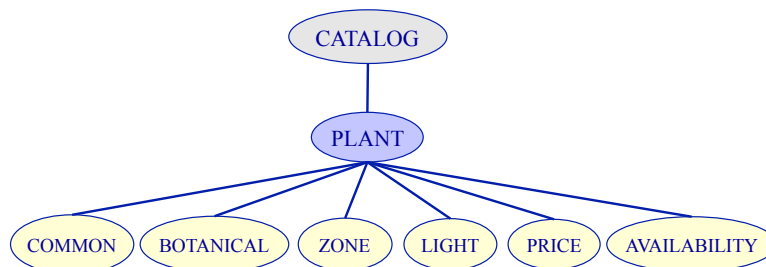


37

Parsing XML

XML DOM Properties and Methods

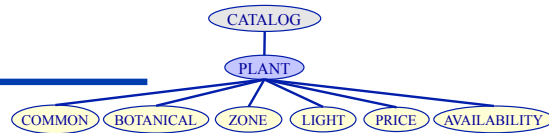
- ◆ `x.nodeName` - the name of `x`
- ◆ `x.nodeValue` - the value of `x`
- ◆ `x.parentNode` - the parent node of `x`
- ◆ `x.childNodes` - the child nodes of `x` (is an array)
- ◆ `x.getElementsByTagName(name)` - get all elements of `x` with a specified tag name



38

Parsing XML

XML DOM Example



```
xmlDoc.getElementsByTagName("COMMON")[0].childNodes[0].nodeValue
```

xmlDoc - XML document created by the parser

getElementsByTagName("COMMON")[0] – first <COMMON> element

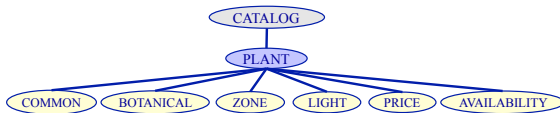
childNodes[0] – first child of the <COMMON> element (the text node)

nodeValue – value of the node (the text itself)

39

Parsing XML

Accessing Elements



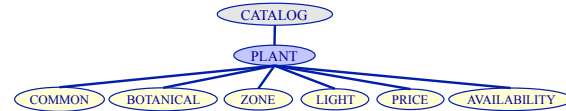
```
xmlDoc.getElementsByTagName("COMMON")[0].childNodes[0].nodeValue
```

```
<CATALOG>
  <PLANT>
    <COMMON>Bloodroot</COMMON>
    <BOTANICAL>Sanguinaria canadensis</BOTANICAL>
    <ZONE>4</ZONE>
    <LIGHT>Mostly Shady</LIGHT>
    <PRICE>$2.44</PRICE>
    <AVAILABILITY>031599</AVAILABILITY>
  </PLANT>
  <PLANT>
    <COMMON>Columbine</COMMON>
    <BOTANICAL>Aquilegia canadensis</BOTANICAL>
    <ZONE>3</ZONE>
    <LIGHT>Mostly Shady</LIGHT>
    <PRICE>$9.37</PRICE>
    <AVAILABILITY>030699</AVAILABILITY>
  </PLANT>
  ...
</CATALOG>
```

40

Parsing XML

Questions



```
xmlDoc.getElementsByTagName("COMMON")[0].childNodes[0].nodeValue
```

```
<CATALOG>
  <PLANT>
    <COMMON>Bloodroot</COMMON>
    <BOTANICAL>Sanguinaria canadensis</BOTANICAL>
    <ZONE>4</ZONE>
    <LIGHT>Mostly Shady</LIGHT>
    <PRICE>$2.44</PRICE>
    <AVAILABILITY>031599</AVAILABILITY>
  </PLANT>
  <PLANT>
    <COMMON>Columbine</COMMON>
    <BOTANICAL>Aquilegia canadensis</BOTANICAL>
    <ZONE>3</ZONE>
    <LIGHT>Mostly Shady</LIGHT>
    <PRICE>$9.37</PRICE>
    <AVAILABILITY>030699</AVAILABILITY>
  </PLANT>
  ...
</CATALOG>
```

Bloodroot

What code will access the following?

- 1) \$2.44
- 2) Columbine

41

Parsing XML

Example 1 - Display First Element

- ◆ Parse the plant_catalog.xml file and print the common name, botanical name, and price of the first <PLANT> element.
- ◆ Components:
 - » HTML code
 - ❖ call parseXML ('plant_catalog.xml') when loading body
 - » JavaScript function parseXML(xmlFilename)
 - ❖ loads XML file
 - ❖ inserts information from the first element into the HTML

<http://www.cs.odu.edu/~mweigle/cs312/xml/plant-parser.html>

42

Example 1

parseXML JavaScript function

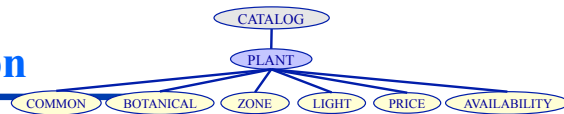
- ◆ Include xmlHelperFns.js
 - » `<script src=http://www.cs.odu.edu/~mweigle/cs312/xml/xmlHelperFns.js type="text/javascript"></script>`
- ◆ Call openXML() from xmlHelperFns.js
 - » turns off asynchronous loading so parser doesn't start until entire document loaded
 - » loads the document using XHR
 - » returns XML document object (xmlDoc)
- ◆ If xmlDoc is null, alert the user and return

<http://www.cs.odu.edu/~mweigle/cs312/xml/plant-parser.html>

43

Example 1

parseXML JavaScript function



- ◆ Insert information from the first element into the HTML

```
document.getElementById("common").innerHTML =  
    xmlDoc.getElementsByTagName("COMMON")  
    [0].childNodes[0].nodeValue;
```
- ```
document.getElementById("botanical").innerHTML =
 xmlDoc.getElementsByTagName("BOTANICAL")
 [0].childNodes[0].nodeValue;
```
- ```
document.getElementById("price").innerHTML =  
    xmlDoc.getElementsByTagName("PRICE")[0].childNodes  
    [0].nodeValue;
```

<http://www.cs.odu.edu/~mweigle/cs312/xml/plant-parser.html>

44

Parsing XML

Example 2 - Plant Table

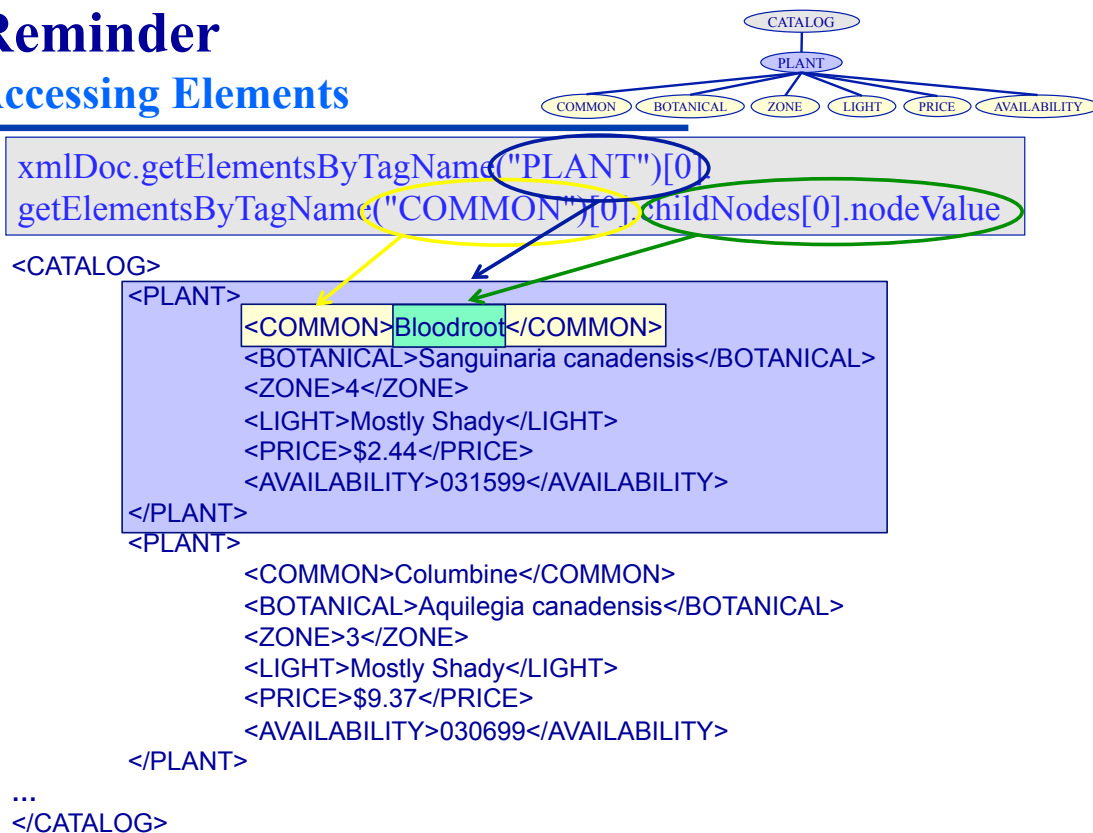
- ◆ Display a table of all plants' common name and price data
- ◆ Use JavaScript to loop through the elements
- ◆ Steps:
 - » load XML
 - ❖ same as in previous example
 - » use JavaScript to create an HTML table
 - ❖ use `getElementsByTagName()` to get all XML PLANT nodes
 - ❖ for each PLANT node, display data from COMMON and PRICE as table data

<http://www.cs.odu.edu/~mweigle/cs312/xml/plant-table.html>

45

Reminder

Accessing Elements



46

Example 2

Create Table of All Plants

```
var x = xmlDoc.getElementsByTagName("PLANT");  
  
document.write("<table border='1'>");  
for (var i=0; i<x.length; i++) {  
    document.write("<tr>");  
    document.write("<td>");  
    document.write(x[i].getElementsByTagName("COMMON")[0].childNodes[0].nodeValue);  
    document.write("</td>");  
  
    document.write("<td>");  
    document.write(x[i].getElementsByTagName("PRICE")[0].childNodes[0].nodeValue);  
    document.write("</td>");  
    document.write("</tr>");  
}  
document.write("</table>");
```

x points to all of the PLANT elements.

x.length is the number of PLANT elements.

x[i] is a plant element

<http://www.cs.odu.edu/~mweigle/cs312/xml/plant-table.html>

47

Example XML Application

- ◆ Display table of plants
 - » like Example 2

- ◆ When the user clicks on table entry, display the full information about that plant
 - » write show() JavaScript function that displays the full information
 - » call show() when a row in the table is clicked

<http://www.cs.odu.edu/~mweigle/cs312/xml/plant-app.html>

48

Example XML Application

show() JavaScript

- ◆ Same code as before to load the XML
- ◆ But, return the array of PLANT elements instead of the whole doc
 - » `xmlDoc.getElementsByTagName("PLANT");`
- ◆ Name the function `getPlantsFromXML()`

<http://www.cs.odu.edu/~mweigle/cs312/xml/plant-app.html>

49

Example XML Application

show() JavaScript

- ◆ Add `show()` function

```
function show (i)
{
    common = (plants[i].getElementsByTagName("COMMON")[0].childNodes
[0].nodeValue);
    botanical = (plants[i].getElementsByTagName("BOTANICAL")[0].childNodes
[0].nodeValue);
    zone = (plants[i].getElementsByTagName("ZONE")[0].childNodes[0].nodeValue);
    light = (plants[i].getElementsByTagName("LIGHT")[0].childNodes[0].nodeValue);
    price = (plants[i].getElementsByTagName("PRICE")[0].childNodes[0].nodeValue);

    txt = "Name: " + common + " (<em>" + botanical + "</em>)<br />Zone: " + zone +
        "<br />Light: " + light + "<br />Price: " + price;
    document.getElementById("show").innerHTML = txt;
}
```

plants is a global variable
plants[i] is a plant element

write info to 'show' element

<http://www.cs.odu.edu/~mweigle/cs312/xml/plant-app.html>

50

Example XML Application

HTML with "show" Element

- ◆ Create 'show' div section for the information to be placed in

```
<body>
```

```
<div id="show">
```

Click on one of the table rows to display the full plant information.

```
</div>
```

<http://www.cs.odu.edu/~mweigle/cs312/xml/plant-app.html>

51

Example XML Application

JavaScript to Create the Table

```
var plants = getPlantsfromXML('plant_catalog.xml');
```

```
if (plants != null) {  
  document.write("<table border='1'>");  
  for (var i=0; i<x.length; i++) {  
    document.write("<tr onclick='show(\" + i + \")'>");  
    document.write("<td>");  
    document.write(plants[i].getElementsByTagName("COMMON")[0].childNodes  
[0].nodeValue);  
    document.write("</td>");  
  
    document.write("<td>");  
    document.write(plants[i].getElementsByTagName("PRICE")[0].childNodes  
[0].nodeValue);  
    document.write("</td>");  
    document.write("</tr>");  
  }  
  document.write("</table>");  
}
```

Same as code in Example 2
except for onclick and plants

<http://www.cs.odu.edu/~mweigle/cs312/xml/plant-app.html>

52

XML

- ◆ What is XML?

<?xml?>

- ◆ XML Syntax

- ◆ Viewing XML

- ◆ XML and JavaScript

<xml />

- ◆ XML in Real Life

Reference for today's material: <http://www.w3schools.com/xml/>

53

XML Encodings

- ◆ Remember this?

`<?xml version="1.0" encoding="ISO-8859-1"?>`

- ◆ ISO-8859-1 specifies a particular type of character encoding

- » Latin alphabet (covers most Western languages)
- » Each character is encoded with a single byte

- ◆ Other single-byte encodings

- » Windows-1252, UTF-8

- ◆ Double-byte encodings

- » UTF-16

54

XML Encodings

- ◆ Windows Notepad saves files as single-byte ANSI (ASCII) by default
 - » You can use 'Save as...' to save as double-byte Unicode (UTF-16)
- ◆ Make sure that you
 - » always use the encoding attribute when you create XML files
 - » use an editor that supports encoding
 - » know what encoding your editor supports

55

Real-Life XML Examples

- ◆ XMLNews
 - » specification for exchanging news and other information
 - » News Industry Text Format (NITF) – XML document type definition (DTD)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<nitf>
  <head>
    <title>Colombia Earthquake</title>
  </head>
  <body>
    <headline>
      <h1>143 Dead in Colombia Earthquake</h1>
    </headline>
    <byline>
      <bytag>By Jared Kotler, Associated Press Writer</bytag>
    </byline>
    <dateline>
      <location>Bogota, Colombia</location>
      <date>Monday January 25 1999 7:28 ET</date>
    </dateline>
  </body>
</nitf>
```

56

Real-Life XML Examples

◆ XML Weather Service

» Used by NOAA

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<current_observation>
  <credit>NOAA's National Weather Service</credit>
  <credit_URL>http://weather.gov/</credit_URL>
  <location>New York/John F. Kennedy Intl Airport, NY</location>
  <station_id>KJFK</station_id>
  <latitude>40.66</latitude>
  <longitude>-73.78</longitude>
  <observation_time_rfc822>Mon, 11 Feb 2008 06:51:00 -0500 EST
  </observation_time_rfc822>
  <weather>A Few Clouds</weather>
  <temp_f>11</temp_f>
  <temp_c>-12</temp_c>
  <relative_humidity>36</relative_humidity>
  <wind_dir>West</wind_dir>
  <wind_degrees>280</wind_degrees>
  <wind_mph>18.4</wind_mph>
  <wind_gust_mph>29</wind_gust_mph>
  <pressure_mb>1023.6</pressure_mb>
  <pressure_in>30.23</pressure_in>
  <dewpoint_f>-11</dewpoint_f>
  <dewpoint_c>-24</dewpoint_c>
  <windchill_f>-7</windchill_f>
  <windchill_c>-22</windchill_c>
  <visibility_mi>10.00</visibility_mi>
</current_observation>
```

57

XML

◆ What is XML?

<?xml?>

◆ XML Syntax

◆ Viewing XML

◆ XML and JavaScript

<xml />

◆ XML in Real Life