User Interaction: XML and JSON

Assoc. Professor Donald J. Patterson
INF 133 Fall 2012
<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>
<!DOCTYPE bookstore [ 

<!ELEMENT bookstore (book+)>
<!ELEMENT book (title, author, year, (price)*)>
<!ELEMENT title (CDATA)>  
<!ELEMENT author (CDATA)>  
<!ELEMENT year (CDATA)>  
<!ELEMENT price (CDATA)>  

<!ATTLIST book category CDATA #REQUIRED>
<!ATTLIST title lang CDATA #IMPLIED> 
]>
• Details
  • All XML Elements Must Have a Closing Tag
  • HTML
    • <p>This is a paragraph
    • <p>This is another paragraph
  • XML
    • <p>This is a paragraph</p>
    • <p>This is another paragraph</p>
• Details
  • XML Tags are Case Sensitive
    • <Message>This is incorrect</message>
    • <message>This is correct</message>
    • <Message>This is correct</Message>
• Details
  • XML Elements Must be Properly Nested
    • HTML might have this
      • <b><i>This text is bold and italic</i></b>
    • Valid XML requires this:
      • <b><i>This text is bold and italic</i></b>
• Details
  • XML Documents Must Have a Root Element
    • This is the top-level tag
      • <root>
      •  <child>
      •   <subchild>.....</subchild>
      •   </child>
      • </root>
Details
- XML Nodes may have attributes
- Which describe the tag
- XML Attribute Values Must be Quoted
  - Invalid:

```xml
<note date=12/11/2007>
  <to>Tove</to>
  <from>Jani</from>
</note>
```

- Valid:

```xml
<note date="12/11/2007">
  <to>Tove</to>
  <from>Jani</from>
</note>
```
Details

Special characters:

- If you put a "<" in your data it will mess up XML parsing
  - `<message>`if salary < 1000 then`</message>`
- So 5 characters are special
  - `<`, `>`, `&`, `'`, "`
  - `&lt;`, `&gt;`, `&amp;`, `apos;`, `&quot;`
  - `<message>`if salary `&lt;` 1000 then`</message>`
• Details
  • Comments in XML
    • <!-- This is a comment -->
  • White-space is preserved
    • <message>There is a lot of space</message>
Attributes and Elements are pretty interchangeable

```xml
<person sex="female">
  <firstname>Anna</firstname>
  <lastname>Smith</lastname>
</person>

<person>
  <sex>female</sex>
  <firstname>Anna</firstname>
  <lastname>Smith</lastname>
</person>
```
<note date="10/01/2008">
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>

<note>
  <date>10/01/2008</date>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>

<note>
  <date>
    <day>10</day>
    <month>01</month>
    <year>2008</year>
  </date>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
• On beyond XML
  • XML validation
  • Schemas like XML - DTD
  • Namespaces
  • XSLT
    • transforms XML to HTML for viewing
• Look at Chrome debugging tools to see the “DOM”
• JSON
  • also structured text
  • also with a syntax applied
  • it can also represent a huge variety of information
  • It also enables data transport
    • Across systems, languages, and networks
• So what does JSON look like?
{  "place": [  {  "suggestion": "at home",  "meta": {  "id": "null",  "index": 0  },  "size": "20.0"  }  ],  "activity": [  {  "suggestion": "working",  "meta": {  "id": "null",  "index": 2  },  "size": "10.55833333333334"  },  {  "suggestion": "sleeping",  "meta": {  "id": "null",  "index": 3  },  "size": "10.0"  }  ],  "other": [  {  "suggestion": "(do not disturb)",  "meta": {  "id": "null",  "index": 1  },  "size": "10.0"  }  ],  "error": [  "false"  ]  }
What is JSON?

- JSON stands for "JavaScript Object Notation"
- JSON was designed to pass data around between browsers and servers
- JSON has no tags, only data
- JSON has no meta-data
• JSON also does not DO Anything
• It is a data format
• A program must be written to manipulate the data
  • To search the data
  • To display the data
  • To change the data
• JSON was developed by people who thought that the meta-data in XML was
  • unnecessary
  • too big
  • too hard to maintain
  • not that valuable
• It also happens to be the native data storage format in Javascript / browsers
Details

- Two basic structures
  - object:
    - name/value pairs
    - think Map
  - array
    - list of values
    - think List
• Details
  • The basic type is a value which can be
    • a string
    • a number
    • an object
    • an array
    • “true”
    • “false”
    • “null”
• Details
  • Object
    • Delimited by curly braces
    • name/values are separated by colons
    • elements are separated by commas
      • names are always strings
      • values are always values

http://json.org/
- Details
  - Array
    - Delimited by square braces
    - elements are separated by commas
    - elements are always values
Details

- String
  - is UNICODE, recommended is "utf-8"
  - is always in double quotes
  - uses \ escape sequences

JSON

http://json.org/
• Details
  • Number
- Details
  - White space outside of quotes is ignored
{
    "place": [
        {
            "suggestion": "at home",
            "meta": {
                "id": "null",
                "index": 0
            },
            "size": "20.0"
        }
    ],
    "activity": [
        {
            "suggestion": "working",
            "meta": {
                "id": "null",
                "index": 2
            },
            "size": "10.55833333333334"
        },
        {
            "suggestion": "sleeping",
            "meta": {
                "id": "null",
                "index": 3
            },
            "size": "10.0"
        }
    ],
    "other": [
        {
            "suggestion": "(do not disturb)",
            "meta": {
                "id": "null",
                "index": 1
            },
            "size": "10.0"
        }
    ],
    "error": [
        "false"
    ]
}
• Supported languages
On beyond JSON

- JSON validation tools are easy to find
  - For example, [jsonlint.com](http://jsonlint.com)
- No defined schema language
- No built-in namespaces (no meta-data!)
- No built-in transformation languages
XML vs JSON

- XML is like a Ferrari
  - A Ferrari will get you to Las Vegas faster
- JSON is like a good bicycle
  - A bicycle can go off-road

- XML is beautiful and powerful
- XML is well-engineered and well-researched
- JSON is much lighter weight
- JSON is easier to just get going fast
XML vs JSON

- XML is like a Ferrari
  - A Ferrari will get you to Las Vegas faster
- JSON is like a good bicycle
  - A bicycle can go off-road

- XML is beautiful and powerful
- XML is well-engineered and well-researched
- JSON is much lighter weight
- JSON is easier to just get going fast
XML vs JSON

• XML is like a Ferrari
  • A Ferrari will get you to Las Vegas faster
• JSON is like a good bicycle
  • A bicycle can go off-road

• XML is beautiful and powerful
• XML is well-engineered and well-researched
• JSON is much lighter weight
• JSON is easier to just get going fast
• JSON is like XML
  • They are both human-readable text
  • They are both hierarchical/ tree-structured
  • Both can be parsed and used in many languages
  • Both can be passed in AJAX requests
    • (despite the X in AJAX)
• JSON is different than XML
  • JSON does not have tags
  • JSON is less verbose
    • quicker to write
    • quicker to read
    • quicker to transport
  • JSON can be parsed trivially using the eval() procedure in Javascript
• JSON has arrays, XML does not
• XML is extensible JSON usually isn’t
Using either looks like:
- get the JSON/XML string
- convert it to a data structure
  - JSON -> eval()
  - XML -> some parse function (lib dependent)
- Use the data

Do not process either type of data by “hand”.
- input: Use a library to parse the data
- output:
  - Create the data in native data structures
  - Use a program or method to output the data structure in JSON/XML
• Represent this as
  • XML
  • JSON
• There is not an absolutely correct answer to how to interpret this tree in the respective languages.
• There are multiple ways to interpret what this tree means.
<xml version="1.0">
<class>
  <INF_221>
    <program>
      search engine
    </program>
  </INF_221>
  <INF_133>
    <quiz>
      crossword puzzle
    </quiz>
    <Assignment>
      <XML/>
    </Assignment>
  </INF_133>
</class>

{
  "class": {
    "INF 221": {
      "program": "search engine"
    },
    "INF 133": {
      "quiz": "Crossword puzzle",
      "Assignment": "XML"
    }
  }
}