XML Programming

Robert M. Dondero, Ph.D.
Princeton University

Objectives

- We will cover:
  - XML
  - XML programming
  - AJAX revisited

Agenda

- **XML**
- XML programming
- AJAX revisited

XML

- Preliminary observation
  - Much of the world's information is:
    - Textual
    - Hierarchical

XML

- ***XML (Extensible Markup Language)***
  - A language for expressing **textual** documents in **hierarchical** (tree-structured) form
  - Worldwide Web Consortium (W3C) specification
  - Similar to HTML…

XML

- XML vs. HTML
  - ***Empty elements*** must be expressed using self-closing tags
    - Empty element: has start tag and no end tag
    - `<hr>`
      - Well formed in HTML
      - Malformed in XML
    - `<hr />`
      - Well formed in HTML
      - Well formed in XML

XML

- XML vs. HTML (cont.)
  - Element nesting must be proper
    - `<strong>...<em>...</strong>...</em>`
      - Well formed (or at least acceptable) in HTML
      - Malformed in XML
    - `<strong>...<em>...</em>...</strong>`
      - Well formed in HTML
      - Well formed in XML

XML

- XML vs. HTML (cont.)
  - Attribute values must be quoted
    - `<span id=myspan>Something</span>`
      - Well formed in HTML
      - Malformed in XML
    - `<span id="myspan">Something</span>`
      - Well formed in HTML
      - Well formed in XML

XML

- XML vs. HTML (cont.)
  - Allows processing instructions
    - `<? ProcessingInstruction ?>`
    - Provide information to XML processor about how to handle the XML document

XML

- XML vs. HTML (cont.)
  - **You define the tags!**

XML

- See **books.html**
  - Tag set is predefined

- See **books.xml**
  - Tag set is **not** predefined

books.html (Page 1 of 1)

```
 1: <!DOCTYPE html>
 2:
 3: <!-- ================================================================== -->
 4: <!-- books.html                                                         -->
 5: <!-- Author: Bob Dondero                                               -->
 6: <!-- ================================================================== -->
 7:
 8: <html>
 9:    <head>
10:       <title>Book List</title>
11:    </head>
12:    <body>
13:       <h1>Book List</h1>
14:       <table>
15:          <thead>
16:             <tr>
17:                <th>ISBN</th>
18:                <th>Author</th>
19:                <th>Title</th>
20:                <th>Price</th>
21:             </tr>
22:          </thead>
23:          <tbody>
24:             <tr>
25:                <td>123</td>
26:                <td>Kernighan</td>
27:                <td>The Practice of Programming<b/td>
28:                <td>40.74 dollars</td>
29:             </tr>
30:             <tr>
31:                <td>234</td>
32:                <td>Kernighan</td>
33:                <td>The C Programming Language</td>
34:                <td>24.99 dollars</td>
35:             </tr>
36:             <tr>
37:                <td>345</td>
38:                <td>Sedgewick</td>
39:                <td>Algorithms in C</td>
40:                <td>61.59 dollars</td>
41:             </tr>
42:          </tbody>
43:       </table>
44:    </body>
45: </html>
```

books.xml (Page 1 of 1)

```
 1: <?xml version="1.0"?>
 2:
 3: <!-- ================================================================== -->
 4: <!-- books.xml                                                          -->
 5: <!-- Author: Bob Dondero                                               -->
 6: <!-- ================================================================== -->
 7:
 8: <books>
 9:    <book>
10:       <isbn>123</isbn>
11:       <author>Kernighan</author>
12:       <title>The Practice of Programming</title>
13:       <price currency="dollars">40.74</price>
14:    </book>
15:    <book>
16:       <isbn>234</isbn>
17:       <author>Kernighan</author>
18:       <title>The C Programming Language</title>
19:       <price currency="dollars">24.99</price>
20:    </book>
21:    <book>
22:       <isbn>345</isbn>
23:       <author>Sedgewick</author>
24:       <title>Algorithms in C</title>
25:       <price currency="dollars">61.59</price>
26:    </book>
27: </books>
```

XML

- Some theory:
  - ***Content***
    - The document's raw data
  - ***Semantic structure***
    - The **organization** of the content
  - ***Presentation***
    - The **rendering** of the content

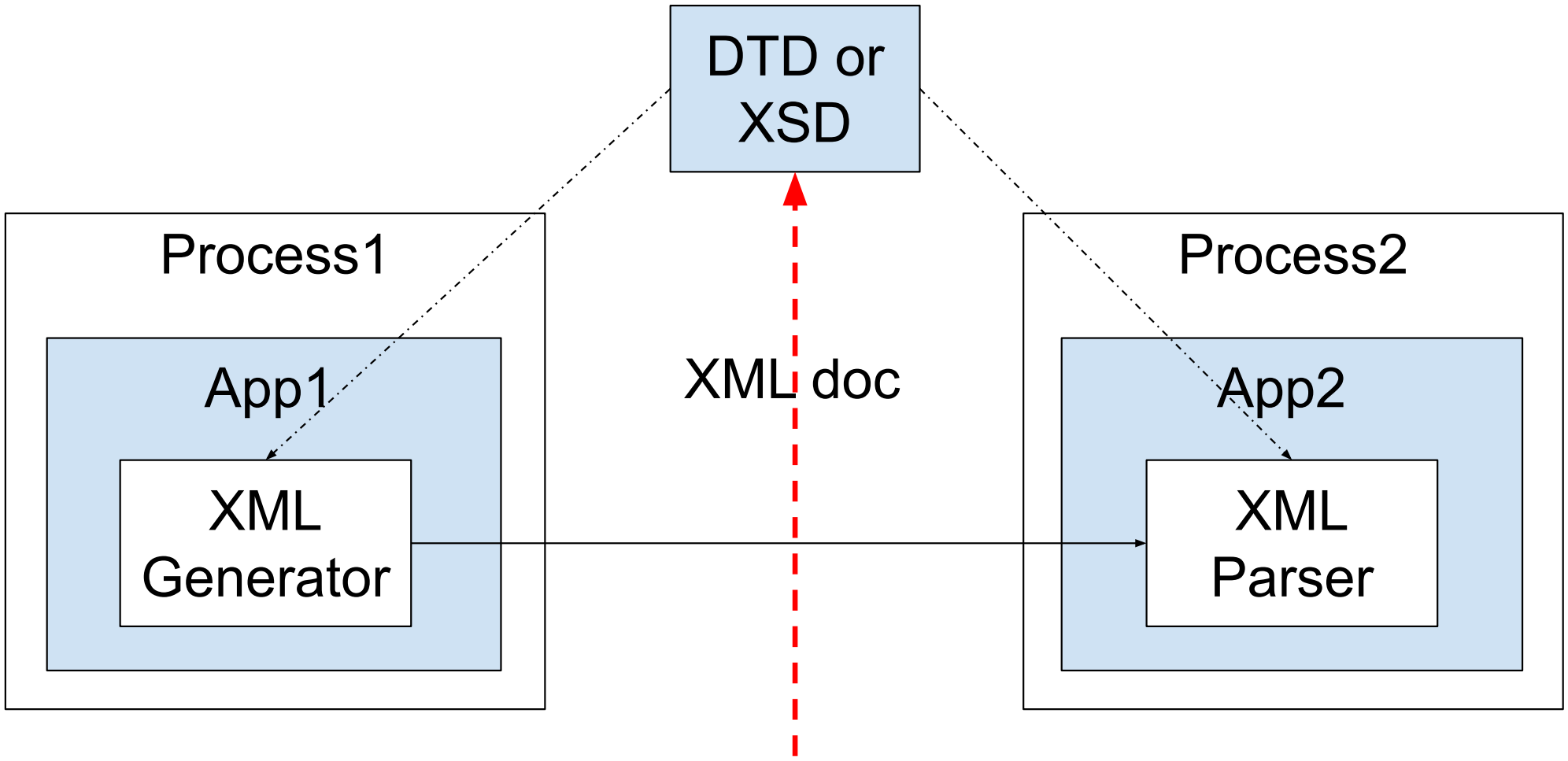
XML

- HTML
  - Tags define **presentation**
  - **Semantic structure** is absent
- XML
  - Tags define **semantic structure**
  - **Presentation** is absent

XML

- Applications of XML:
  - Publishing
    - See **Appendix 1**
  - Data communication
    - Covered now…

XML



Document Type Definition (DTD)
XML Schema Definition (XSD)

Agenda

- XML
- **XML programming**
- AJAX revisited

XML Programming

- Examples in this lecture:
  - In Python
    - Appropriate for XML programming on the **server-side** of a Web app
  - In JavaScript
    - Appropriate for XML programming on the **client-side** of a Web app (i.e., in a browser)

XML Programming

- To run the example **JavaScript** programs in this lecture:
  - `npm install xmldom`
  - `npm install xmlserializer`
  - `npm install sax-parser`

XML Programming

- **World Wide Web Consortium (W3C)** defines two standard APIs:
  - ***SAX***: The **S**imple **A**PI for **X**ML
    - See **Appendix 2**
  - ***DOM***: The **D**ocument **O**bject **M**odel
    - Covered now…

XML Programming

- A DOM parser:
  - Parses given XML doc in its entirety
  - Builds a DOM tree
    - An in-memory tree of objects/nodes

XML Programming

- **writedom** programs
  - Read a XML doc, write its entire DOM

XML Programming

- See **writedom.py**, **domfrompython.txt**

```
$ python writedom.py books.xml > domfrompython.txt
$ cat domfrompython.txt
Document: #document=None
   Comment: #comment=
============================================================
   Comment: #comment= books.xml
   Comment: #comment= Author: Bob Dondero
   Comment: #comment=
============================================================
…
```

writedom.py (Page 1 of 2)

```
#!/usr/bin/env python

#-------------------------------------------------------------------------
# writedom.py
# Author: Bob Dondero
#-------------------------------------------------------------------------

import sys
import xml.dom.minidom

#-------------------------------------------------------------------------

def handle_white_space(string):

    if string is None:
        return 'None'
    if string.strip() == '':
        return 'WHITESPACE'
    return string

#-------------------------------------------------------------------------

def translate_node_type(node_type):

    node_type_map = {
        xml.dom.minidom.Node.ELEMENT_NODE:
            'Element',
        xml.dom.minidom.Node.ATTRIBUTE_NODE:
            'Attribute',
        xml.dom.minidom.Node.TEXT_NODE:
            'Text',
        xml.dom.minidom.Node.CDATA_SECTION_NODE:
            'Cdata Section',
        xml.dom.minidom.Node.ENTITY_REFERENCE_NODE:
            'Entity Reference',
        xml.dom.minidom.Node.ENTITY_NODE:
            'Entity',
        xml.dom.minidom.Node.PROCESSING_INSTRUCTION_NODE:
            'Processing Instruction',
        xml.dom.minidom.Node.COMMENT_NODE:
            'Comment',
        xml.dom.minidom.Node.DOCUMENT_NODE:
            'Document',
        xml.dom.minidom.Node.DOCUMENT_TYPE_NODE:
            'Document Type',
        xml.dom.minidom.Node.DOCUMENT_FRAGMENT_NODE:
            'Document Fragment',
        xml.dom.minidom.Node.NOTATION_NODE:
            'Notation'
    }

    return node_type_map.get(node_type, 'Error')

#-------------------------------------------------------------------------

def write_dom_tree(node, indent_level):

    for _ in range(indent_level):
        print('   ', end='')

    print(translate_node_type(node.nodeType), end='')
    print(': ', end='')
    print(node.nodeName, end='')
    print('=', end='')
    print(handle_white_space(node.nodeValue), end='')
```

writedom.py (Page 2 of 2)

```

    attributes = node.attributes
    if attributes is not None:
        keys = list(attributes.keys())
        for key in keys:
            attribute = attributes.getNamedItem(key)
            print(' (', end='')
            print(translate_node_type(attribute.nodeType), end='')
            print(': ', end='')
            print(key, end='')
            print('=', end='')
            print(handle_white_space(attribute.nodeValue), end='')
            print(')', end='')

    print()

    for child in node.childNodes:
        write_dom_tree(child, indent_level + 1)

#-------------------------------------------------------------------------

def main():

    if len(sys.argv) != 2:
        print('Usage: ' + sys.argv[0] + ' file', file=sys.stderr)
        sys.exit(1)

    xml_file_name = sys.argv[1]

    try:
        with open(xml_file_name, 'r', encoding='utf-8') as xml_file:
            xml_doc = xml_file.read()

        xml_dom_tree = xml.dom.minidom.parseString(xml_doc)

        write_dom_tree(xml_dom_tree, 0)

    except Exception as ex:
        print(ex, file=sys.stderr)
        sys.exit(1)

#-------------------------------------------------------------------------

if __name__ == '__main__':
    main()
```

domfrompython.txt (Page 1 of 1)

```
    1: Document: #document=None
    2:    Comment: #comment= ================================================
=============
    3:    Comment: #comment= books.xml

    4:    Comment: #comment= Author: Bob Dondero

    5:    Comment: #comment= ================================================
=============
    6:    Element: books=None
    7:       Text: #text=WHITESPACE
    8:       Element: book=None
    9:          Text: #text=WHITESPACE
   10:          Element: isbn=None
   11:             Text: #text=123
   12:          Text: #text=WHITESPACE
   13:          Element: author=None
   14:             Text: #text=Kernighan
   15:          Text: #text=WHITESPACE
   16:          Element: title=None
   17:             Text: #text=The Practice of Programming
   18:          Text: #text=WHITESPACE
   19:          Element: price=None (Attribute: currency=dollars)
   20:             Text: #text=40.74
   21:          Text: #text=WHITESPACE
   22:       Text: #text=WHITESPACE
   23:       Element: book=None
   24:          Text: #text=WHITESPACE
   25:          Element: isbn=None
   26:             Text: #text=234
   27:          Text: #text=WHITESPACE
   28:          Element: author=None
   29:             Text: #text=Kernighan
   30:          Text: #text=WHITESPACE
   31:          Element: title=None
   32:             Text: #text=The C Programming Language
   33:          Text: #text=WHITESPACE
   34:          Element: price=None (Attribute: currency=dollars)
   35:             Text: #text=24.99
   36:          Text: #text=WHITESPACE
   37:       Text: #text=WHITESPACE
   38:       Element: book=None
   39:          Text: #text=WHITESPACE
   40:          Element: isbn=None
   41:             Text: #text=345
   42:          Text: #text=WHITESPACE
   43:          Element: author=None
   44:             Text: #text=Sedgewick
   45:          Text: #text=WHITESPACE
   46:          Element: title=None
   47:             Text: #text=Algorithms in C
   48:          Text: #text=WHITESPACE
   49:          Element: price=None (Attribute: currency=dollars)
   50:             Text: #text=61.59
   51:          Text: #text=WHITESPACE
   52:       Text: #text=WHITESPACE
```

blank (Page 1 of 1)

```
    1: This page is intentionally blank.
```

writedom.js (Page 1 of 2)

```
//-----------------------------------------------------------------------
// writedom.js
// Author: Bob Dondero
//-----------------------------------------------------------------------

'use strict';

const fs = require('fs');
const xmldom = require('xmldom');

//-----------------------------------------------------------------------

function handleWhiteSpace(s) {
    if (s === null)
        return 'None';
    if (s.trim() === '')
        return 'WHITESPACE';
    return s;
}

//-----------------------------------------------------------------------

function translateNodeType(nodeType) {
    if (nodeType ===  1) return 'Element';
    if (nodeType ===  2) return 'Attribute';
    if (nodeType ===  3) return 'Text';
    if (nodeType ===  4) return 'Cdata Section';
    if (nodeType ===  5) return 'Entity Reference';
    if (nodeType ===  6) return 'Entity';
    if (nodeType ===  7) return 'Processing Instruction';
    if (nodeType ===  8) return 'Comment';
    if (nodeType ===  9) return 'Document';
    if (nodeType === 10) return 'Document Type';
    if (nodeType === 11) return 'Document Fragment';
    if (nodeType === 12) return 'Notation';
    return 'Error';
}

//-----------------------------------------------------------------------

function writeDomTree(node, indentLevel) {
   for (let i = 0; i < indentLevel; i++)
      process.stdout.write('   ');

   process.stdout.write(translateNodeType(node.nodeType));
   process.stdout.write(': ');
   process.stdout.write(String(node.nodeName));
   process.stdout.write('=');
   process.stdout.write(handleWhiteSpace(node.nodeValue));

   let attributes = node.attributes;
   if (attributes)
      for (let i = 0; i < attributes.length; i++) {
         let attribute = attributes.item(i);
         process.stdout.write(' (');
         process.stdout.write(translateNodeType(attribute.nodeType));
         process.stdout.write(': ');
         process.stdout.write(attribute.nodeName);
         process.stdout.write('=');
         process.stdout.write(handleWhiteSpace(attribute.nodeValue));
         process.stdout.write(')');
      }
   process.stdout.write('\n');
   if (node.childNodes)
      for (let i = 0; i < node.childNodes.length; i++) {
```

writedom.js (Page 2 of 2)

```
         let child = node.childNodes[i];
         writeDomTree(child, indentLevel + 1);
      }
}

//-----------------------------------------------------------------------

function parseFile(err, xmlDoc) {
   if (err) {
      process.stderr.write(err.message + '\n');
      process.exit(1);
   }

   let parser = new xmldom.DOMParser();
   let xmlDomTree = parser.parseFromString(xmlDoc, 'text/xml');

   writeDomTree(xmlDomTree, 0);
}

//-----------------------------------------------------------------------

function main() {
   if (process.argv.length !== 3) {
      process.stderr.write('Usage: ' + process.argv[0] + ' '
         + process.argv[1] + ' file\n');
      process.exit(1);
   }

   let xmlFileName = process.argv[2];

   fs.readFile(xmlFileName, 'utf-8', parseFile);
}

//-----------------------------------------------------------------------

if (require.main === module)
   main();
```

domfromjavascript.txt (Page 1 of 1)

```
    1: Document: #document=None
    2:    Processing Instruction: undefined=version="1.0"
    3:    Text: #text=WHITESPACE
    4:    Comment: #comment= ================================================
=============
    5:    Text: #text=WHITESPACE
    6:    Comment: #comment= books.xml

    7:    Text: #text=WHITESPACE
    8:    Comment: #comment= Author: Bob Dondero

    9:    Text: #text=WHITESPACE
   10:    Comment: #comment= ================================================
=============
   11:    Text: #text=WHITESPACE
   12:    Element: books=None
   13:       Text: #text=WHITESPACE
   14:       Element: book=None
   15:          Text: #text=WHITESPACE
   16:          Element: author=None
   17:             Text: #text=Kernighan
   18:          Text: #text=WHITESPACE
   19:          Element: title=None
   20:             Text: #text=The Practice of Programming
   21:          Text: #text=WHITESPACE
   22:          Element: price=None (Attribute: currency=dollars)
   23:             Text: #text=40.74
   24:          Text: #text=WHITESPACE
   25:       Text: #text=WHITESPACE
   26:       Element: book=None
   27:          Text: #text=WHITESPACE
   28:          Element: author=None
   29:             Text: #text=Kernighan
   30:          Text: #text=WHITESPACE
   31:          Element: title=None
   32:             Text: #text=The C Programming Language
   33:          Text: #text=WHITESPACE
   34:          Element: price=None (Attribute: currency=dollars)
   35:             Text: #text=24.99
   36:          Text: #text=WHITESPACE
   37:       Text: #text=WHITESPACE
   38:       Element: book=None
   39:          Text: #text=WHITESPACE
   40:          Element: author=None
   41:             Text: #text=Sedgewick
   42:          Text: #text=WHITESPACE
   43:          Element: title=None
   44:             Text: #text=Algorithms in C
   45:          Text: #text=WHITESPACE
   46:          Element: price=None (Attribute: currency=dollars)
   47:             Text: #text=61.59
   48:          Text: #text=WHITESPACE
   49:       Text: #text=WHITESPACE
```

blank (Page 1 of 1)

```
    1: This page is intentionally blank.
```

XML Programming

- See **writedom.js**, **domfromjavascript.txt**

```
$ node writedom.js books.xml > domfromjavascript.txt
$ cat domfromjavascript.txt
Document: #document=None
   Processing Instruction: undefined=version="1.0"
   Text: #text=WHITESPACE
   Comment: #comment=
=============================================================
   Text: #text=WHITESPACE
   Comment: #comment= books.xml
   Text: #text=WHITESPACE
   Comment: #comment= Author: Bob Dondero
   Text: #text=WHITESPACE
   Comment: #comment=
=============================================================
…
```

XML Programming

- **writebooks** programs
  - Write all books in books.xml

XML Programming

- See **writebooks.py**

```
$ python writebooks.py
ISBN: 123
Author: Kernighan
Title: The Practice of Programming
Price: 40.74 dollars

ISBN: 234
Author: Kernighan
Title: The C Programming Language
Price: 24.99 dollars

ISBN: 345
Author: Sedgewick
Title: Algorithms in C
Price: 61.59 dollars

$
```

writebooks.py (Page 1 of 1)

```python
#!/usr/bin/env python

#-------------------------------------------------------------------------
# writebooks.py
# Author: Bob Dondero
#-------------------------------------------------------------------------

import sys
import xml.dom.minidom

#-------------------------------------------------------------------------

def main():

    try:
        with open('books.xml', 'r', encoding='utf-8') as xml_file:
            xml_doc = xml_file.read()

        xml_dom_tree = xml.dom.minidom.parseString(xml_doc)

        books_node = xml_dom_tree.childNodes[4]

        i = 1
        while i < len(books_node.childNodes):
            book_node = books_node.childNodes[i]

            isbn_node = book_node.childNodes[1]
            author_node = book_node.childNodes[3]
            title_node = book_node.childNodes[5]
            price_node = book_node.childNodes[7]

            isbn = isbn_node.childNodes[0].nodeValue
            author = author_node.childNodes[0].nodeValue
            title = title_node.childNodes[0].nodeValue
            price = price_node.childNodes[0].nodeValue

            currency = \
                price_node.attributes.getNamedItem('currency').nodeValue

            print('ISBN:', isbn)
            print('Author:', author)
            print('Title:', title)
            print('Price:', price, currency)
            print()
            i += 2

    except Exception as ex:
        print(ex, file=sys.stderr)
        sys.exit(1)

#-------------------------------------------------------------------------

if __name__ == '__main__':
    main()
```

writebooks.js (Page 1 of 1)

```javascript
//-------------------------------------------------------------------------
// writebooks.js
// Author: Bob Dondero
//-------------------------------------------------------------------------

'use strict';

const fs = require('fs');
const xmldom = require('xmldom');

//-------------------------------------------------------------------------

function parseFile(err, xmlDoc) {
   if (err) {
      process.stderr.write(err.message + '\n');
      process.exit(1);
   }

   let parser = new xmldom.DOMParser();
   let xmlDomTree = parser.parseFromString(xmlDoc, 'text/xml');

   let booksNode = xmlDomTree.childNodes[10];

   let i = 1;
   while (i < booksNode.childNodes.length) {
      let bookNode = booksNode.childNodes[i];

      let isbnNode = bookNode.childNodes[1];
      let authorNode = bookNode.childNodes[3];
      let titleNode = bookNode.childNodes[5];
      let priceNode = bookNode.childNodes[7];

      let isbn = isbnNode.childNodes[0].nodeValue;
      let author = authorNode.childNodes[0].nodeValue;
      let title = titleNode.childNodes[0].nodeValue;
      let price = priceNode.childNodes[0].nodeValue;

      let currency =
         priceNode.attributes.getNamedItem('currency').nodeValue;

      process.stdout.write('ISBN: ' + isbn + '\n');
      process.stdout.write('Author: ' + author + '\n');
      process.stdout.write('Title: ' + title + '\n');
      process.stdout.write('Price: ' +  price + ' ' +
         currency + '\n');
      process.stdout.write('\n');
      i += 2;
   }
}

//-------------------------------------------------------------------------

function main() {
   fs.readFile('books.xml', 'utf-8', parseFile);
}

if (require.main === module)
   main();
```

XML Programming

- See **writebooks.js**

```
$ node writebooks.js
ISBN: 123
Author: Kernighan
Title: The Practice of Programming
Price: 40.74 dollars

ISBN: 234
Author: Kernighan
Title: The C Programming Language
Price: 24.99 dollars

ISBN: 345
Author: Sedgewick
Title: Algorithms in C
Price: 61.59 dollars

$
```

XML Programming

- **writebooksshort** programs
  - Same as writebooks programs

XML Programming

- See **writebooksshort.py**

```
$ python writebooksshort.py
ISBN: 123
Author: Kernighan
Title: The Practice of Programming
Price: 40.74 dollars

ISBN: 234
Author: Kernighan
Title: The C Programming Language
Price: 24.99 dollars

ISBN: 345
Author: Sedgewick
Title: Algorithms in C
Price: 61.59 dollars

$
```

writebooksshort.py (Page 1 of 1)

```python
#!/usr/bin/env python

#-------------------------------------------------------------------------
# writebooksshort.py
# Author: Bob Dondero
#-------------------------------------------------------------------------

import sys
import xml.dom.minidom

#-------------------------------------------------------------------------

def main():

    try:
        with open('books.xml', 'r', encoding='utf-8') as xml_file:
            xml_doc = xml_file.read()

        xml_dom_tree = xml.dom.minidom.parseString(xml_doc)

        book_nodes = xml_dom_tree.getElementsByTagName('book')

        for book_node in book_nodes:
            isbn_node = book_node.getElementsByTagName('isbn')[0]
            author_node = book_node.getElementsByTagName('author')[0]
            title_node = book_node.getElementsByTagName('title')[0]
            price_node = book_node.getElementsByTagName('price')[0]

            isbn = isbn_node.childNodes[0].nodeValue
            author = author_node.childNodes[0].nodeValue
            title = title_node.childNodes[0].nodeValue
            price = price_node.childNodes[0].nodeValue
            currency = price_node.attributes.getNamedItem(
                'currency').nodeValue

            print('ISBN:', isbn)
            print('Author:', author)
            print('Title:', title)
            print('Price:', price, currency)
            print()

    except Exception as ex:
        print(ex, file=sys.stderr)
        sys.exit(1)

#-------------------------------------------------------------------------

if __name__ == '__main__':
    main()
```

writebooksshort.js (Page 1 of 1)

```javascript
//-------------------------------------------------------------------------
// writebooksshort.js
// Author: Bob Dondero
//-------------------------------------------------------------------------

'use strict';

const fs = require('fs');
const xmldom = require('xmldom');

//-------------------------------------------------------------------------

function parseFile(err, xmlDoc) {
   if (err) {
      process.stderr.write(err.message + '\n');
      process.exit(1);
   }

   let parser = new xmldom.DOMParser();
   let xmlDomTree = parser.parseFromString(xmlDoc, 'text/xml');

   let bookNodes = xmlDomTree.getElementsByTagName('book');

   for (let i = 0; i < bookNodes.length; i++) {
      let bookNode = bookNodes[i];

      let isbnNode = bookNode.getElementsByTagName('isbn')[0];
      let authorNode = bookNode.getElementsByTagName('author')[0];
      let titleNode = bookNode.getElementsByTagName('title')[0];
      let priceNode = bookNode.getElementsByTagName('price')[0];

      let isbn = isbnNode.childNodes[0].nodeValue;
      let author = authorNode.childNodes[0].nodeValue;
      let title = titleNode.childNodes[0].nodeValue;
      let price = priceNode.childNodes[0].nodeValue;
      let currency =
         priceNode.attributes.getNamedItem('currency').nodeValue;

      process.stdout.write('ISBN: ' + isbn + '\n');
      process.stdout.write('Author: ' + author + '\n');
      process.stdout.write('Title: ' + title + '\n');
      process.stdout.write('Price: ' +  price + ' ' + currency + '\n');
      process.stdout.write('\n');
   }
}

//-------------------------------------------------------------------------

function main() {
   fs.readFile('books.xml', 'utf-8', parseFile);
}

if (require.main === module)
   main();
```

XML Programming

- See **writebooksshort.js**

```
$ node writebooksshort.js
ISBN: 123
Author: Kernighan
Title: The Practice of Programming
Price: 40.74 dollars

ISBN: 234
Author: Kernighan
Title: The C Programming Language
Price: 24.99 dollars

ISBN: 345
Author: Sedgewick
Title: Algorithms in C
Price: 61.59 dollars

$
```

XML Programming

- **roundtripxml** programs
  - **Parse**: XML doc → DOM tree
    - Common
  - **Generate**: DOM tree → XML doc
    - Less common

XML Programming

- See **roundtripxml.py**

```
$ python roundtripxml.py books.xml
<?xml version="1.0" ?><!--
============================================================ --><!--
books.xml                                                    --><!--
Author: Bob Dondero                                          --><!--
============================================================ --><books>
   <book>
      <isbn>123</isbn>
      <author>Kernighan</author>
      <title>The Practice of Programming</title>
      <price currency="dollars">40.74</price>
   </book>
   <book>
      <isbn>234</isbn>
      <author>Kernighan</author>
      <title>The C Programming Language</title>
      <price currency="dollars">24.99</price>
   </book>
   <book>
      <isbn>345</isbn>
      <author>Sedgewick</author>
      <title>Algorithms in C</title>
      <price currency="dollars">61.59</price>
   </book>
</books>
$
```

roundtripxml.py (Page 1 of 1)

```python
#!/usr/bin/env python

#-------------------------------------------------------------------------
# roundtripxml.py
# Author: Bob Dondero
#-------------------------------------------------------------------------

import sys
import xml.dom.minidom

#-------------------------------------------------------------------------

def main():

    if len(sys.argv) != 2:
        print('Usage: ' + sys.argv[0] + ' file', file=sys.stderr)
        sys.exit(1)

    xml_file_name = sys.argv[1]

    try:

        with open(xml_file_name, 'r', encoding='utf-8') as xml_file:
            xml_doc = xml_file.read()

        # XML to DOM tree:
        xml_dom_tree = xml.dom.minidom.parseString(xml_doc)

        # DOM tree to XML:
        xml_doc = xml_dom_tree.toxml()

        # Show that it worked:
        print(xml_doc)

    except Exception as ex:
        print(ex, file=sys.stderr)
        sys.exit(1)

#-------------------------------------------------------------------------

if __name__ == '__main__':
    main()
```

roundtripxml.js (Page 1 of 1)

```javascript
//-------------------------------------------------------------------------
// roundtripxml.js
// Author: Bob Dondero
//-------------------------------------------------------------------------

// WARNING:  DOESN'T HANDLE PROCESSING INSTRUCTION!!!

'use strict';

const fs = require('fs');
const xmldom = require('xmldom');
const xmlserializer = require('xmlserializer');

//-------------------------------------------------------------------------

function parseFile(err, xmlDoc) {
   if (err) {
      process.stderr.write(err.message + '\n');
      process.exit(1);
   }

   // XML to DOM tree:
   let parser = new xmldom.DOMParser();
   let xmlDomTree = parser.parseFromString(xmlDoc, 'text/xml');

   // DOM tree to XML:
   // WARNING:  DOESN'T HANDLE PROCESSING INSTRUCTION!!!
   xmlDoc = xmlserializer.serializeToString(xmlDomTree);

   // Show that it worked:
   process.stdout.write(xmlDoc);
}

//-------------------------------------------------------------------------

function main() {
   if (process.argv.length !== 3) {
      process.stderr.write('Usage: ' + process.argv[0] + ' '
         + process.argv[1] + ' file\n');
      process.exit(1);
   }

   let xmlFileName = process.argv[2];

   fs.readFile(xmlFileName, 'utf-8', parseFile);
}

if (require.main === module)
   main();
```

XML Programming

- See **roundtripxml.js**

```
$ node roundtrip.xml.js books.xml

ERROR
```

XML Programming

- See **roundtripxml.js** (cont.)

```
$ node roundtripxml.js books.xml
<!-- ============================================================ -->
<!-- books.xml                                                    -->
<!-- Author: Bob Dondero                                          -->
<!-- ============================================================ -->

<books xmlns="undefined">
   <book>
      <isbn>123</isbn>
      <author>Kernighan</author>
      <title>The Practice of Programming</title>
      <price currency="dollars">40.74</price>
   </book>
   <book>
      <isbn>234</isbn>
      <author>Kernighan</author>
      <title>The C Programming Language</title>
      <price currency="dollars">24.99</price>
   </book>
   <book>
      <isbn>345</isbn>
      <author>Sedgewick</author>
      <title>Algorithms in C</title>
      <price currency="dollars">61.59</price>
   </book>
</books>
$
```

After removing the processing instruction from books.xml

Agenda

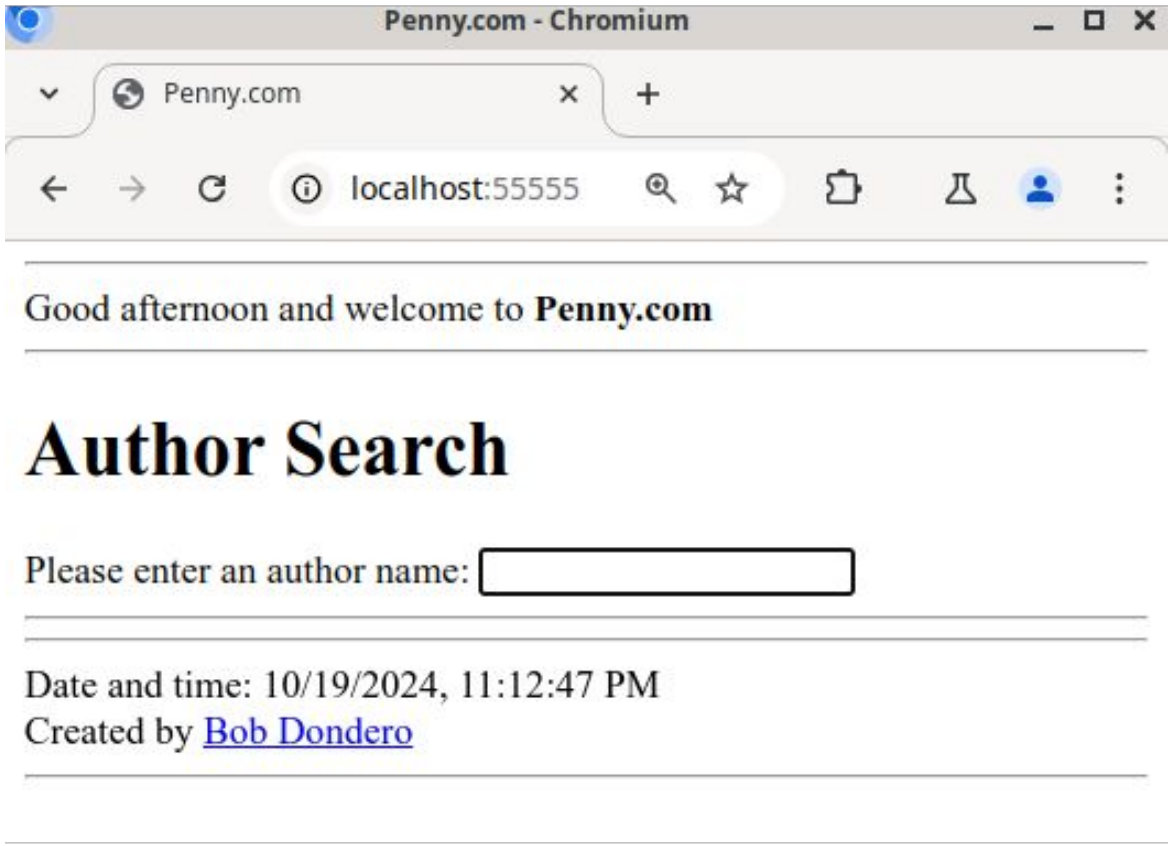
- XML
- XML programming
- **AJAX revisited**

AJAX Revisited

- **AJAX**
  - Asynchronous JavaScript and **XML**

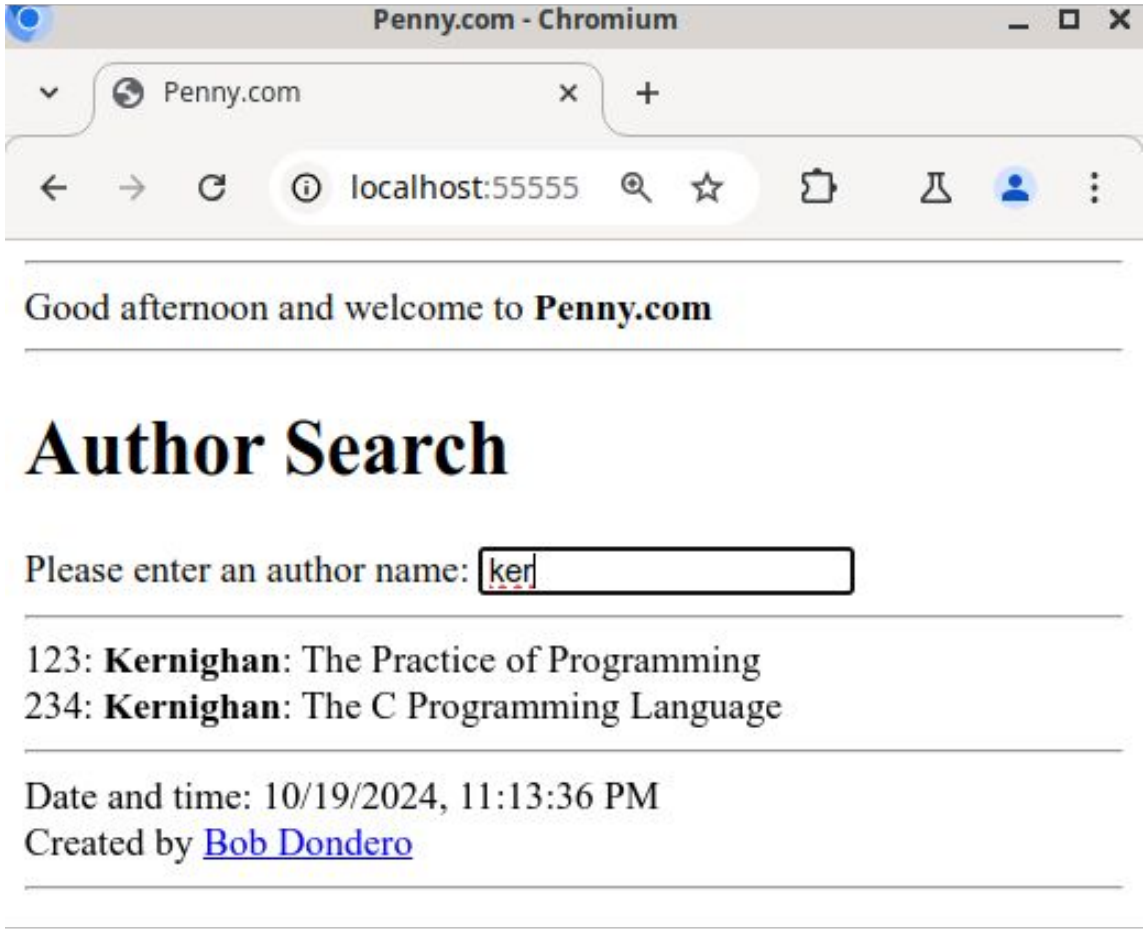
AJAX Revisited

- See **PennyXml** app



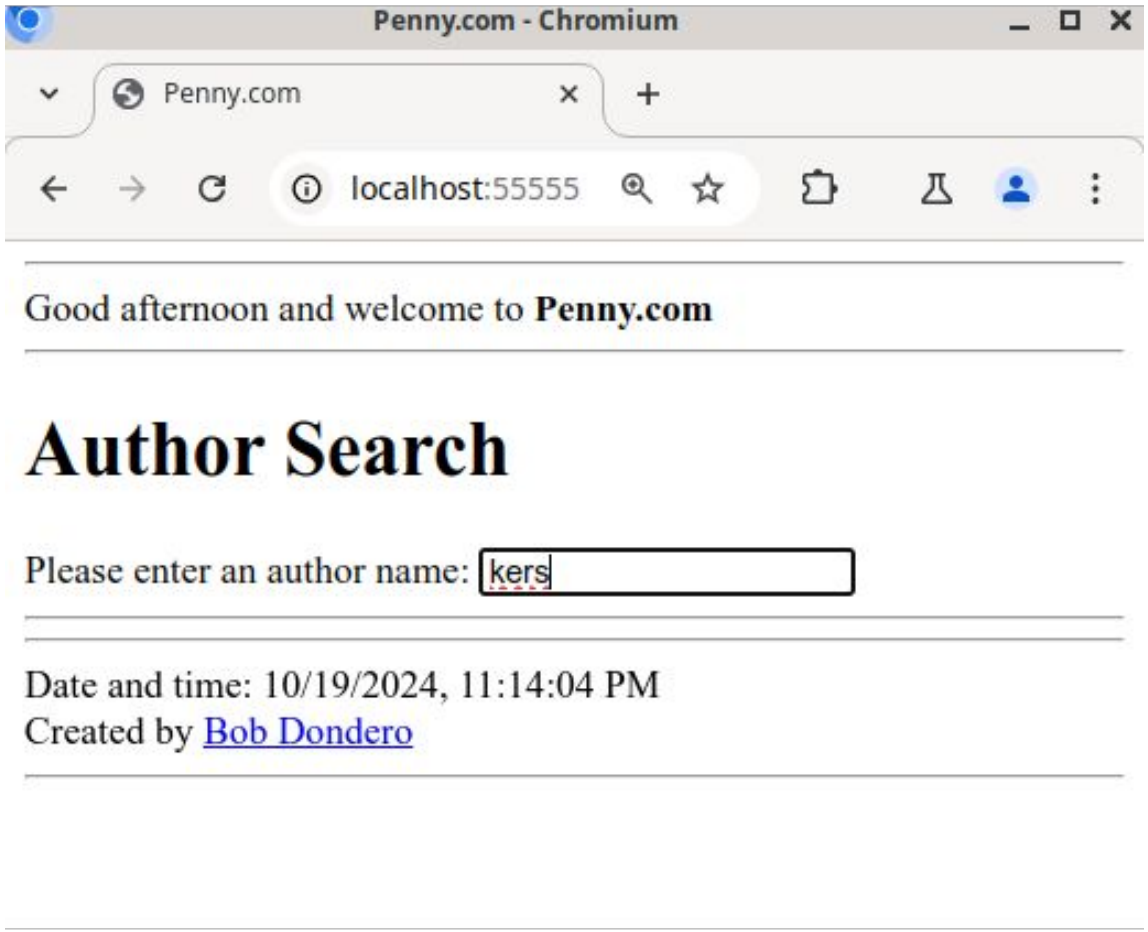
AJAX Revisited

- See **PennyXml** app (cont.)



AJAX Revisited

- See **PennyXml** app (cont.)



AJAX Revisited

- See **PennyXml** app (cont.)
  - runserver.py
  - penny.sql
  - penny.sqlite
  - database.py
  - **penny.py**
  - **index.html**
    - AJAX automatically parses XML doc

PennyXml/penny.py (Page 1 of 1)

```
#!/usr/bin/env python

#-----------------------------------------------------------------------
# penny.py
# Author: Bob Dondero
#-----------------------------------------------------------------------

import json
import flask
import database

#-----------------------------------------------------------------------

app = flask.Flask(__name__, template_folder='.')

#-----------------------------------------------------------------------

@app.route('/', methods=['GET'])
@app.route('/index', methods=['GET'])
def index():

    return flask.send_file('index.html')

#-----------------------------------------------------------------------

@app.route('/searchresults', methods=['GET'])
def search_results():

    author = flask.request.args.get('author')
    if author is None:
        author = ''
    author = author.strip()

    if author == '':
        books = []
    else:
        books = database.get_books(author) # Exception handling omitted

    xml_doc = '<?xml version="1.0"?>'
    xml_doc += '<books>'
    for book in books:
        xml_doc += '<book>'
        xml_doc += '<isbn><![CDATA[' + book['isbn'] + ']]></isbn>'
        xml_doc += '<author><![CDATA[' + book['author'] + ']]></author>'
        xml_doc += '<title><![CDATA[' + book['title'] + ']]></title>'
        xml_doc += '</book>'
    xml_doc += '</books>'

    response = flask.make_response(xml_doc)
    response.headers['Content-Type'] = 'application/xml'
    return response
```

blank (Page 1 of 1)

This page is intentionally blank.

PennyXml/index.html (Page 1 of 2)

```
<!DOCTYPE html>
<html>
   <head>
      <title>Penny.com</title>
   </head>

   <body>
      <hr>
      Good <span id="ampmSpan"></span> and welcome to
      <strong>Penny.com</strong>
      <hr>

      <h1>Author Search</h1>
      Please enter an author name:
      <input type="text" id="authorInput" autoFocus>
      <hr>
      <div id="resultsDiv"></div>

      <hr>
      Date and time: <span id="datetimeSpan"></span><br>
      Created by <a href="https://www.cs.princeton.edu/~rdondero">
      Bob Dondero</a>
      <hr>

      <script src=
         "https://cdn.jsdelivr.net/npm/mustache@4.2.0/mustache.min.js">
      </script>

      <script>

         'use strict';

         function getAmPm() {
            let dateTime = new Date();
            let hours = dateTime.getHours();
            let amPm = (hours < 12) ? 'morning' : 'afternoon';
            let ampmSpan = document.getElementById('ampmSpan');
            ampmSpan.innerHTML = amPm;
         }

         function getDateTime() {
            let dateTime = new Date();
            let datetimeSpan =
               document.getElementById('datetimeSpan');
            datetimeSpan.innerHTML = dateTime.toLocaleString();
         }

         function convertToBooks(xmlDomTree) {
            let books = [];
            let bookNodes = xmlDomTree.getElementsByTagName('book');
            for (let bookNode of bookNodes) {
               let isbn = bookNode.getElementsByTagName('isbn')[0]
                  .childNodes[0].nodeValue;
               let author = bookNode.getElementsByTagName('author')[0]
                  .childNodes[0].nodeValue;
               let title = bookNode.getElementsByTagName('title')[0]
                  .childNodes[0].nodeValue;
               let book =
                   {'isbn': isbn, 'author': author, 'title': title};
               books.push(book);
            }
            return books;
         }

         function convertToHtml(books) {
```

PennyXml/index.html (Page 2 of 2)

```
            let template = `
               {{#books}}
                  {{isbn}}:
                  <strong>{{author}}</strong>:
                  {{title}}
                  <br>
               {{/books}}
            `;
            let map = {books: books};
            let html = Mustache.render(template, map);
            return html;
         }

         function handleResponse() {
            if (this.status !== 200) {
               alert('Error: Failed to fetch data from server');
               return;
            }
            let books = convertToBooks(this.responseXML);
            let html = convertToHtml(books);
            let resultsDiv = document.getElementById('resultsDiv');
            resultsDiv.innerHTML = html;
         }

         function handleError() {
            alert('Error: Failed to fetch data from server');
         }

         let request = null;

         function getResults() {
            let authorInput = document.getElementById('authorInput');
            let author = authorInput.value;
            let encodedAuthor = encodeURIComponent(author);
            let url = '/searchresults?author=' + encodedAuthor;
            if (request !== null)
               request.abort();
            request = new XMLHttpRequest();
            request.onload = handleResponse;
            request.onerror = handleError;
            request.open('GET', url);
            request.send();
         }

         let timer = null;

         function debouncedGetResults() {
            clearTimeout(timer);
            timer = setTimeout(getResults, 500);
         }

         function setup() {
            getAmPm();
            window.setInterval(getAmPm, 1000);
            getDateTime();
            window.setInterval(getDateTime, 1000);
            let authorInput = document.getElementById('authorInput');
            authorInput.addEventListener('input', debouncedGetResults);
         }

         document.addEventListener('DOMContentLoaded', setup);

      </script>
   </body>
</html>
```

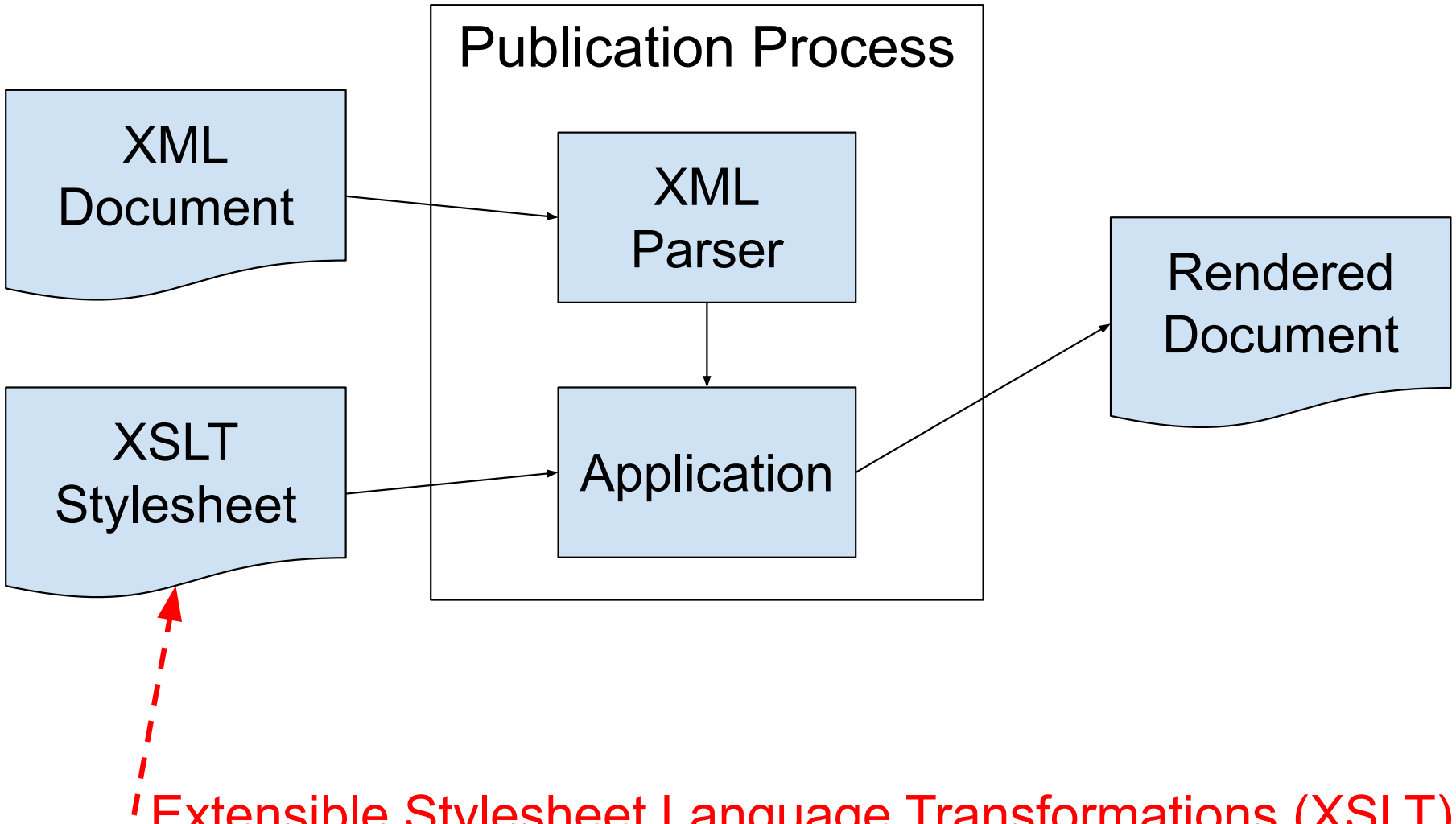
Summary

- We have covered:
  - XML
  - XML programming
  - AJAX revisited

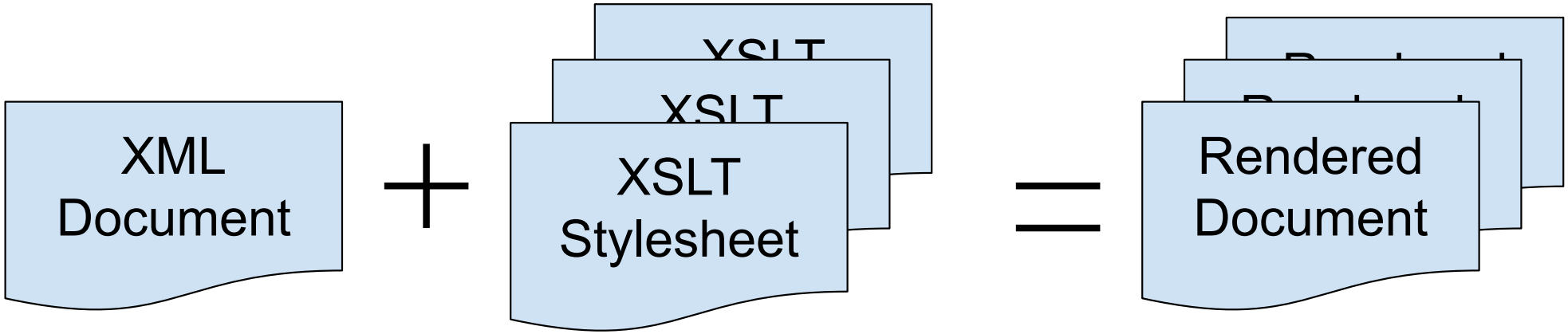
- See also:
  - **Appendix 1**: XML for Publishing
  - **Appendix 2**: XML Programming: SAX
  - **Appendix 3**: XML Checking

Appendix 1:
XML for Publishing

XML for Publishing

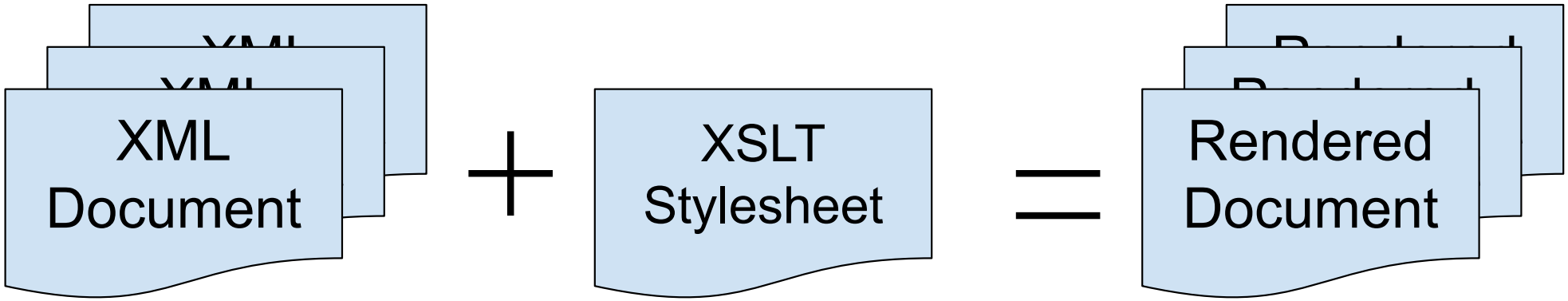


XML for Publishing



Same XML doc can be presented in multiple ways

XML for Publishing



Multiple XML docs can be presented in the same way

Appendix 2:
XML Programming: SAX

SAX Programming

- **Problem**
  - DOM trees can be very large
  - Sometimes you need only a small part
- **Solution**
  - Use SAX API instead of DOM API

SAX Programming

- A SAX parser:
  - Traverses given XML document
  - Calls methods (which you define) as it encounters each start tag, end tag, text, etc.

SAX Programming

- **writeauthorssax** programs
  - Write all authors in books.xml

SAX Programming

- See **writeauthorssax.py**

```
$ python writeauthorssax.py
Kernighan
Kernighan
Sedgewick
$
```

writeauthorssax.py (Page 1 of 1)

```python
#!/usr/bin/env python

#------------------------------------------------------------------------
# writeauthorssax.py
# Author: Bob Dondero
#------------------------------------------------------------------------

import sys
import xml.sax

#------------------------------------------------------------------------

class AuthorsHandler(xml.sax.ContentHandler):

    def __init__(self):
        xml.sax.ContentHandler.__init__(self)
        self._in_author_element = False
        self._author_name = ''

    def startElement(self, name, attrs):
        if name == 'author':
            self._in_author_element = True

    def characters(self, content):
        if self._in_author_element:
            self._author_name += content

    def endElement(self, name):
        if name == 'author':
            print(self._author_name)
            self._author_name = ''
            self._in_author_element = False

#------------------------------------------------------------------------

def main():

    try:
        with open('books.xml', 'r', encoding='utf-8') as xml_file:
            xml_doc = xml_file.read()

        xml.sax.parseString(xml_doc, AuthorsHandler())

    except Exception as ex:
        print(ex, file=sys.stderr)
        sys.exit(1)

#------------------------------------------------------------------------

if __name__ == '__main__':
    main()
```

writeauthorssax.js (Page 1 of 1)

```javascript
//-----------------------------------------------------------------------
// writeauthorssax.js
// Author: Bob Dondero
//-----------------------------------------------------------------------

'use strict';

const fs = require('fs');
const saxparser = require('sax-parser');

//-----------------------------------------------------------------------

function parse(cb) {
   let authorName = '';
   let inAuthorElement = false;

   cb.onStartElementNS(function(elem, attrs, prefix, uri, namespace) {
      if (elem === 'author')
         inAuthorElement = true;
   });

   cb.onCharacters(function(chars) {
      if (inAuthorElement)
         authorName += chars;
   });

   cb.onEndElementNS(function(elem, prefix, uri) {
      if (elem === 'author') {
         process.stdout.write(authorName + '\n');
         authorName = '';
         inAuthorElement = false;
      }
   });
}

//-----------------------------------------------------------------------

function parseFile(err, xmlDoc) {
   if (err) {
      process.stderr.write(err.message + '\n');
      process.exit(1);
   }

   let parser = new saxparser.SaxParser(parse);
   parser.parseString(xmlDoc);
}

//-----------------------------------------------------------------------

function main() {
   fs.readFile('books.xml', 'utf-8', parseFile);
}

if (require.main === module)
   main(process.argv);
```

SAX Programming

- See **writeauthorssax.js**

```
$ node writeauthorssax.js
Kernighan
Kernighan
Sedgewick
$
```

Appendix 3:
XML Checking

XML Checking

- To run the example Python programs in this appendix:
  - `python -m pip install lxml`

XML Checking: Well-Formedness

- Computer science jargon…
  - An XML doc is ***well-formed*** iff it conforms to the XML specification

XML Checking: Well-Formedness

- See books.xml (revisited)
- See **booksmalformed.xml**
- See **checkxml.py**

booksmalformed.xml (Page 1 of 1)

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

<!-- ================================================================ -->
<!-- books.xml                                                        -->
<!-- Author: Bob Dondero                                              -->
<!-- ================================================================ -->

<books>
   <book>
      <isbn>123</isbn>
      <author>Kernighan</author>
      <title>The Practice of Programming</title>
      <price currency="dollars">40.74</price>
   </book>
   <book>
      <isbn>234</isbn>
      <author>Kernighan</author>
      <title>The C Programming Language</title>
      <price currency="dollars">24.99</price>
   </book>
   <book>
      <isbn>345</isbn>
      <author>Sedgewick</author>
      <title>Algorithms in C</title>
      <price currency="dollars">61.59</prices>
   </book>
</books>
```

checkxml.py (Page 1 of 1)

```
#!/usr/bin/env python

#-------------------------------------------------------------------------
# checkxml.py
# Author: Bob Dondero
#-------------------------------------------------------------------------

import sys
import xml.dom.minidom

#-------------------------------------------------------------------------

def main():

    if len(sys.argv) != 2:
        print('Usage: ' + sys.argv[0] + ' file', file=sys.stderr)
        sys.exit(1)

    xml_file_name = sys.argv[1]

    try:
        with open(xml_file_name, 'r', encoding='utf-8') as xml_file:
            xml_doc = xml_file.read()

        xml.dom.minidom.parseString(xml_doc)
        print('The document is well-formed.')

    except Exception as ex:
        print(ex, file=sys.stderr)
        sys.exit(1)

#-------------------------------------------------------------------------

if __name__ == '__main__':
    main()
```

XML Checking: Well-Formedness

```
$ python checkxml.py books.xml
The document is well-formed.
$ python checkxml.py booksmalformed.xml
mismatched tag: line 25, column 39
```

XML Checking: Validity

- ***DTD (Document Type Definition)***
  - An ISO standard
  - A DTD defines whether an XML doc is **valid**
- ***XSD (XML Schema Definition)***
  - A W3C standard
  - A XSD defines whether an XML doc is **valid**
- Others: ***RELAX NG***, ***Schematron***, ***DSDL***, …
  - See Wikipedia “XML” page
- Can use to define a comm protocol

XML Checking: Validity

- Computer science jargon…
  - A DTD or XSD defines a ***grammar***
  - The grammar defines a ***language***
    - A language is a set of documents
  - An XML document is ***valid*** with respect to a DTD or XSD iff it is an element of the language defined by that DTD or XSD

XML Checking: Validity

- See **books.dtd**
- See **booksusingdtd.xml**
- See **booksusingdtdinvalid.xml**
- See **checkxmlusingdtd.py**

books.dtd (Page 1 of 1)

```
<!-- ============================================================ -->
<!-- books.dtd                                                    -->
<!-- Author: Bob Dondero                                          -->
<!-- ============================================================ -->

<!ELEMENT books (book*)>
<!ELEMENT book (isbn, author, title, price)>
<!ELEMENT isbn (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT price (#PCDATA)>
<!ATTLIST price currency CDATA #REQUIRED>
```

booksusingdtd.xml (Page 1 of 1)

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

<!-- ============================================================ -->
<!-- booksusingdtd.xml                                            -->
<!-- Author: Bob Dondero                                          -->
<!-- ============================================================ -->

<!DOCTYPE books SYSTEM "books.dtd">

<books>
   <book>
      <isbn>123</isbn>
      <author>Kernighan</author>
      <title>The Practice of Programming</title>
      <price currency="dollars">40.74</price>
   </book>
   <book>
      <isbn>234</isbn>
      <author>Kernighan</author>
      <title>The C Programming Language</title>
      <price currency="dollars">24.99</price>
   </book>
   <book>
      <isbn>345</isbn>
      <author>Sedgewick</author>
      <title>Algorithms in C</title>
      <price currency="dollars">61.59</price>
   </book>
</books>
```

booksusingdtdinvalid.xml (Page 1 of 1)

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

<!-- ================================================================= -->
<!-- booksusingdtdbad.xml                                              -->
<!-- Author: Bob Dondero                                               -->
<!-- ================================================================= -->

<!DOCTYPE books SYSTEM "books.dtd">

<books>
   <book>
      <isbn>123</isbn>
      <author>Kernighan</author>
      <title>The Practice of Programming</title>
      <price currency="dollars">40.74</price>
   </book>
   <book>
      <isbn>234</isbn>
      <author>Kernighan</author>
      <title>The C Programming Language</title>
      <price currency="dollars">24.99</price>
   </book>
   <book>
      <isbn>345</isbn>
      <author>Sedgewick</author>
      <price currency="dollars">61.59</price>
      <title>Algorithms in C</title>
   </book>
</books>
```

checkxmlusingdtd.py (Page 1 of 1)

```
#!/usr/bin/env python

#-----------------------------------------------------------------------
# checkxmlusingdtd.py
# Author: Bob Dondero
#-----------------------------------------------------------------------

import sys
from lxml import etree

#-----------------------------------------------------------------------

def main():

    if len(sys.argv) != 2:
        print('Usage: ' + sys.argv[0] + ' file', file=sys.stderr)
        sys.exit(1)

    xml_file_name = sys.argv[1]

    try:
        with open(xml_file_name, 'r', encoding='utf-8') as xml_file:
            xml_doc = xml_file.read()

        parser = etree.XMLParser(dtd_validation=True)
        etree.fromstring(xml_doc, parser)
        print('The document is well-formed and valid.')

    except Exception as ex:
        print(ex, file=sys.stderr)
        sys.exit(1)

#-----------------------------------------------------------------------

if __name__ == '__main__':
    main()

#-----------------------------------------------------------------------

# Example executions:

# $ python checkxmlusingdtd.py booksusingdtd.xml
# The document is well-formed and valid.

# $ python checkxmlusingdtd.py booksusingdtdinvalid.xml
# Element book content does not follow the DTD, expecting
#   (author , title , price), got (author price title ),
#   line 25, column 11 (booksusingdtdinvalid.xml, line 25)

```

XML Checking: Validity

```
$ python checkxmlusingdtd.py booksusingdtd.xml
The document is well-formed and valid.
$ python checkxmlusingdtd.py booksusingdtdinvalid.xml
Element book content does not follow the DTD,
expecting (isbn , author , title , price), got (isbn
author price title ), line 28, column 11 (<string>,
line 28)
$
```

XML Checking: Validity

- See **books.xsd**
- See **booksusingxsd.xml**
- See **booksusingxsdinvalid.xml**
- See **checkxmlusingxsd.py**

books.xsd (Page 1 of 1)

```
 1: <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 2:
 3: <!-- ============================================================= -->
 4: <!-- books.xsd                                                     -->
 5: <!-- Author: Bob Dondero                                           -->
 6: <!-- ============================================================= -->
 7:
 8: <xsd:element name="books" type="Books" />
 9:
10: <xsd:complexType name="Books">
11:    <xsd:sequence>
12:       <xsd:element name="book" type="Book"
13:          minOccurs="0" maxOccurs="unbounded" />
14:    </xsd:sequence>
15: </xsd:complexType>
16:
17: <xsd:complexType name="Book">
18:    <xsd:sequence>
19:       <xsd:element name="isbn" type="xsd:string" />
20:       <xsd:element name="author" type="xsd:string" />
21:       <xsd:element name="title" type="xsd:string" />
22:       <xsd:element name="price" type="Price" />
23:    </xsd:sequence>
24: </xsd:complexType>
25:
26: <xsd:complexType name="Price">
27:    <xsd:simpleContent>
28:       <xsd:extension base="xsd:string">
29:          <xsd:attribute name="currency" type="xsd:string"
30:             use="required" />
31:       </xsd:extension>
32:    </xsd:simpleContent>
33: </xsd:complexType>
34:
35: </xsd:schema>
```

booksusingxsd.xml (Page 1 of 1)

```
 1: <?xml version="1.0"?>
 2:
 3: <!-- ============================================================= -->
 4: <!-- booksusingxsd.xml                                             -->
 5: <!-- Author: Bob Dondero                                           -->
 6: <!-- ============================================================= -->
 7:
 8: <books xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 9:    xsi:noNamespaceSchemaLocation="books.xsd">
10:    <book>
11:       <isbn>123</isbn>
12:       <author>Kernighan</author>
13:       <title>The Practice of Programming</title>
14:       <price currency="dollars">40.74</price>
15:    </book>
16:    <book>
17:       <isbn>234</isbn>
18:       <author>Kernighan</author>
19:       <title>The C Programming Language</title>
20:       <price currency="dollars">24.99</price>
21:    </book>
22:    <book>
23:       <isbn>345</isbn>
24:       <author>Sedgewick</author>
25:       <title>Algorithms in C</title>
26:       <price currency="dollars">61.59</price>
27:    </book>
28: </books>
```

booksusingxsdinvalid.xml (Page 1 of 1)

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

<!-- ================================================================= -->
<!-- booksusingschemabad.xml                                           -->
<!-- Author: Bob Dondero                                               -->
<!-- ================================================================= -->

<books xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:noNamespaceSchemaLocation="books.xsd">
   <book>
      <isbn>123</isbn>
      <author>Kernighan</author>
      <title>The Practice of Programming</title>
      <price currency="dollars">40.74</price>
   </book>
   <book>
      <isbn>234</isbn>
      <author>Kernighan</author>
      <title>The C Programming Language</title>
      <price currency="dollars">24.99</price>
   </book>
   <book>
      <isbn>234</isbn>
      <author>Sedgewick</author>
      <price currency="dollars">61.59</price>
      <title>Algorithms in C</title>
   </book>
</books>
```

checkxmlusingxsd.py (Page 1 of 1)

```
#!/usr/bin/env python

#-------------------------------------------------------------------------
# checkxmlusingxsd.py
# Author: Bob Dondero
#-------------------------------------------------------------------------

import sys
from lxml import etree

#-------------------------------------------------------------------------

def main():

    if len(sys.argv) != 3:
        print('Usage: ' + sys.argv[0] + ' xmlfile schemafile',
            file=sys.stderr)
        sys.exit(1)

    xml_file_name = sys.argv[1]
    schema_file_name = sys.argv[2]

    try:
        with open(schema_file_name, 'r', encoding='utf-8') \
            as schema_file:
            schema_doc = schema_file.read()

        with open(xml_file_name, mode='r', encoding='utf-8') \
            as xml_file:
            xml_doc = xml_file.read()

        schema = etree.XMLSchema(etree.XML(schema_doc))
        parser = etree.XMLParser(schema=schema)
        etree.fromstring(xml_doc, parser)

        print('The document is well-formed and valid.')

    except Exception as ex:
        print(ex, file=sys.stderr)
        sys.exit(1)

#-------------------------------------------------------------------------

if __name__ == '__main__':
    main()

#-------------------------------------------------------------------------

# Example executions:

# $ python checkxmlusingxsd.py booksusingxsd.xml books.xsd
# The document is well-formed and valid.

# $ python checkxmlusingxsd.py booksusingxsdinvalid.xml books.xsd
# Element 'price': This element is not expected.
#    Expected is ( title ). (<string>, line 0)

```

XML Checking: Validity

```
$ python checkxmlusingxsd.py booksusingxsd.xml
books.xsd
The document is well-formed and valid.
$ python checkxmlusingxsd.py booksusingxsdinvalid.xml
books.xsd
Element 'price': This element is not expected.
Expected is ( title ). (<string>, line 0)
$
```

XML Checking: Validity

- Advantages of validation via **DTDs**:
  - DTDs can be defined inline
  - DTDs are compact and readable
  - DTDs are widely supported

XML Checking: Validity

- Advantages of validation via **XSDs**:
  - XSDs support strong typing
    - Can specify that an element contains an integral number, real number, date, ...
  - XSDs provide natural way to map XML doc to typed objects
  - XSDs are written in XML; so can create/parse using XML tools
    - Can define a XSD, which defines a XSD, which ...