books.html (Page 1 of 1)

```
<!DOCTYPE html>

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

<html>
   <head>
      <title>Book List</title>
   </head>
   <body>
      <h1>Book List</h1>
      <table>
         <thead>
            <tr>
               <th>ISBN</th>
               <th>Author</th>
               <th>Title</th>
               <th>Price</th>
            </tr>
         </thead>
         <tbody>
            <tr>
               <td>123</td>
               <td>Kernighan</td>
               <td>The Practice of Programming<b/td>
               <td>40.74 dollars</td>
            </tr>
            <tr>
               <td>234</td>
               <td>Kernighan</td>
               <td>The C Programming Language</td>
               <td>24.99 dollars</td>
            </tr>
            <tr>
               <td>345</td>
               <td>Sedgewick</td>
               <td>Algorithms in C</td>
               <td>61.59 dollars</td>
            </tr>
         </tbody>
      </table>
   </body>
</html>
```

books.xml (Page 1 of 1)

```
<!-- ================================================================= -->
<!-- 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>
```

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)

```
 1: //-------------------------------------------------------------------------
 2: // writedom.js
 3: // Author: Bob Dondero
 4: //-------------------------------------------------------------------------
 5:
 6: 'use strict';
 7:
 8: const fs = require('fs');
 9: const xmldom = require('xmldom');
10:
11: //-------------------------------------------------------------------------
12:
13: function handleWhiteSpace(s) {
14:     if (s === null)
15:         return 'None';
16:     if (s.trim() === '')
17:         return 'WHITESPACE';
18:     return s;
19: }
20:
21: //-------------------------------------------------------------------------
22:
23: function translateNodeType(nodeType) {
24:     if (nodeType ===  1) return 'Element';
25:     if (nodeType ===  2) return 'Attribute';
26:     if (nodeType ===  3) return 'Text';
27:     if (nodeType ===  4) return 'Cdata Section';
28:     if (nodeType ===  5) return 'Entity Reference';
29:     if (nodeType ===  6) return 'Entity';
30:     if (nodeType ===  7) return 'Processing Instruction';
31:     if (nodeType ===  8) return 'Comment';
32:     if (nodeType ===  9) return 'Document';
33:     if (nodeType === 10) return 'Document Type';
34:     if (nodeType === 11) return 'Document Fragment';
35:     if (nodeType === 12) return 'Notation';
36:     return 'Error';
37: }
38:
39: //-------------------------------------------------------------------------
40:
41: function writeDomTree(node, indentLevel) {
42:    for (let i = 0; i < indentLevel; i++)
43:       process.stdout.write('   ');
44:
45:    process.stdout.write(translateNodeType(node.nodeType));
46:    process.stdout.write(': ');
47:    process.stdout.write(String(node.nodeName));
48:    process.stdout.write('=');
49:    process.stdout.write(handleWhiteSpace(node.nodeValue));
50:
51:    let attributes = node.attributes;
52:    if (attributes)
53:       for (let i = 0; i < attributes.length; i++) {
54:          let attribute = attributes.item(i);
55:          process.stdout.write(' (');
56:          process.stdout.write(translateNodeType(attribute.nodeType));
57:          process.stdout.write(': ');
58:          process.stdout.write(attribute.nodeName);
59:          process.stdout.write('=');
60:          process.stdout.write(handleWhiteSpace(attribute.nodeValue));
61:          process.stdout.write(')');
62:       }
63:    process.stdout.write('\n');
64:    if (node.childNodes)
65:       for (let i = 0; i < node.childNodes.length; i++) {
```

writedom.js (Page 2 of 2)

```
66:          let child = node.childNodes[i];
67:          writeDomTree(child, indentLevel + 1);
68:       }
69: }
70:
71: //-------------------------------------------------------------------------
72:
73: function parseFile(err, xmlDoc) {
74:    if (err) {
75:       process.stderr.write(err.message + '\n');
76:       process.exit(1);
77:    }
78:
79:    let parser = new xmldom.DOMParser();
80:    let xmlDomTree = parser.parseFromString(xmlDoc, 'application/xml');
81:
82:    writeDomTree(xmlDomTree, 0);
83: }
84:
85: //-------------------------------------------------------------------------
86:
87: function main() {
88:    if (process.argv.length !== 3) {
89:       process.stderr.write('Usage: ' + process.argv[0] + ' '
90:          + process.argv[1] + ' file\n');
91:       process.exit(1);
92:    }
93:
94:    let xmlFileName = process.argv[2];
95:
96:    fs.readFile(xmlFileName, 'utf-8', parseFile);
97: }
98:
99: //-------------------------------------------------------------------------
100:
101: if (require.main === module)
102:    main();
```

domfromjavascript.txt (Page 1 of 1)

```
    1: Document: #document=None
    2:    Comment: #comment= ================================================
=============
    3:    Text: #text=WHITESPACE
    4:    Comment: #comment= books.xml

    5:    Text: #text=WHITESPACE
    6:    Comment: #comment= Author: Bob Dondero

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

blank (Page 1 of 1)

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

writebooks.py (Page 1 of 1)

```
#!/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)

```
//---------------------------------------------------------------------
// 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[8];

   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();
```

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();
```

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);
```

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 = '<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
```

PennyXml/index.html (Page 1 of 3)

```
<!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 3)

```
66:             let template = `
67:                {{#books}}
68:                   {{isbn}}:
69:                   <strong>{{author}}</strong>:
70:                   {{title}}
71:                   <br>
72:                {{/books}}
73:             `;
74:             let map = {books: books};
75:             let html = Mustache.render(template, map);
76:             return html;
77:          }
78:
79:          function handleResponse() {
80:             if (this.status !== 200) {
81:                alert('Error: Failed to fetch data from server');
82:                return;
83:             }
84:             let books = convertToBooks(this.responseXML);
85:             let html = convertToHtml(books);
86:             let resultsDiv = document.getElementById('resultsDiv');
87:             resultsDiv.innerHTML = html;
88:          }
89:
90:          function handleError() {
91:             alert('Error: Failed to fetch data from server');
92:          }
93:
94:          let request = null;
95:
96:          function getResults() {
97:             let authorInput = document.getElementById('authorInput');
98:             let author = authorInput.value;
99:             let encodedAuthor = encodeURIComponent(author);
100:            let url = '/searchresults?author=' + encodedAuthor;
101:            if (request !== null)
102:               request.abort();
103:            request = new XMLHttpRequest();
104:            request.onload = handleResponse;
105:            request.onerror = handleError;
106:            request.open('GET', url);
107:            request.send();
108:         }
109:
110:         let timer = null;
111:
112:         function debouncedGetResults() {
113:            clearTimeout(timer);
114:            timer = setTimeout(getResults, 500);
115:         }
116:
117:         function setupHeader() {
118:            getAmPm();
119:            let apInterval = window.setInterval(getAmPm, 1000);
120:            window.addEventListener('beforeunload',
121:               function(e) {window.clearInterval(apInterval);})
122:         }
123:
124:         function setupFooter() {
125:            getDateTime();
126:            let dtInterval = window.setInterval(getDateTime, 1000);
127:            window.addEventListener('beforeunload',
128:               function(e) {window.clearInterval(dtInterval);})
129:         }
130:
```

PennyXml/index.html (Page 3 of 3)

```
131:         function setup() {
132:            setupHeader();
133:            setupFooter();
134:            let authorInput = document.getElementById('authorInput');
135:            authorInput.addEventListener('input', debouncedGetResults);
136:         }
137:
138:         document.addEventListener('DOMContentLoaded', setup);
139:
140:      </script>
141:   </body>
142: </html>
```

roundtripxml.py (Page 1 of 1)

```
#!/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)

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

'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, 'application/xml');

   // DOM tree to XML:
   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();
```

booksmalformed.xml (Page 1 of 1)

```
<!-- ============================================================= -->
<!-- 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()
```

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)

```
<!-- ============================================================= -->
<!-- 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)

```
<!-- ============================================================= -->
<!-- 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)

```

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

booksusingxsdinvalid.xml (Page 1 of 1)

```
<!-- ============================================================= -->
<!-- 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)

```