

## 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>

```

## writedom.py (Page 1 of 2)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # writedom.py
5: # Author: Bob Dondero
6: #-----
7:
8: import sys
9: import xml.dom.minidom
10:
11: #-----
12:
13: def handle_white_space(string):
14:
15:     if string is None:
16:         return 'None'
17:     if string.strip() == '':
18:         return 'WHITESPACE'
19:     return string
20:
21: #-----
22:
23: def translate_node_type(node_type):
24:
25:     node_type_map = {
26:         xml.dom.minidom.Node.ELEMENT_NODE:
27:             'Element',
28:         xml.dom.minidom.Node.ATTRIBUTE_NODE:
29:             'Attribute',
30:         xml.dom.minidom.Node.TEXT_NODE:
31:             'Text',
32:         xml.dom.minidom.Node.CDATA_SECTION_NODE:
33:             'Cdata Section',
34:         xml.dom.minidom.Node.ENTITY_REFERENCE_NODE:
35:             'Entity Reference',
36:         xml.dom.minidom.Node.ENTITY_NODE:
37:             'Entity',
38:         xml.dom.minidom.Node.PROCESSING_INSTRUCTION_NODE:
39:             'Processing Instruction',
40:         xml.dom.minidom.Node.COMMENT_NODE:
41:             'Comment',
42:         xml.dom.minidom.Node.DOCUMENT_NODE:
43:             'Document',
44:         xml.dom.minidom.Node.DOCUMENT_TYPE_NODE:
45:             'Document Type',
46:         xml.dom.minidom.Node.DOCUMENT_FRAGMENT_NODE:
47:             'Document Fragment',
48:         xml.dom.minidom.Node.NOTATION_NODE:
49:             'Notation'
50:     }
51:
52:     return node_type_map.get(node_type, 'Error')
53:
54: #-----
55:
56: def write_dom_tree(node, indent_level):
57:
58:     for _ in range(indent_level):
59:         print(' ', end='')
60:
61:     print(translate_node_type(node.nodeType), end='')
62:     print(':', end='')
63:     print(node.nodeName, end='')
64:     print('=', end='')
65:     print(handle_white_space(node.nodeValue), end='')

```

## writedom.py (Page 2 of 2)

```

66:
67:     attributes = node.attributes
68:     if attributes is not None:
69:         keys = list(attributes.keys())
70:         for key in keys:
71:             attribute = attributes.getNamedItem(key)
72:             print(' ', end='')
73:             print(translate_node_type(attribute.nodeType), end='')
74:             print(':', end='')
75:             print(key, end='')
76:             print('=', end='')
77:             print(handle_white_space(attribute.nodeValue), end='')
78:             print(')', end='')
79:
80:     print()
81:
82:     for child in node.childNodes:
83:         write_dom_tree(child, indent_level + 1)
84:
85: #-----
86:
87: def main():
88:
89:     if len(sys.argv) != 2:
90:         print('Usage: ' + sys.argv[0] + ' file', file=sys.stderr)
91:         sys.exit(1)
92:
93:     xml_file_name = sys.argv[1]
94:
95:     try:
96:         with open(xml_file_name, 'r', encoding='utf-8') as xml_file:
97:             xml_doc = xml_file.read()
98:
99:             xml_dom_tree = xml.dom.minidom.parseString(xml_doc)
100:
101:             write_dom_tree(xml_dom_tree, 0)
102:
103:     except Exception as ex:
104:         print(ex, file=sys.stderr)
105:         sys.exit(1)
106:
107: #-----
108:
109: if __name__ == '__main__':
110:     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, 'text/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: 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.

```

## writebooks.py (Page 1 of 1)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # writebooks.py
5: # Author: Bob Dondero
6: #-----
7:
8: import sys
9: import xml.dom.minidom
10:
11: #-----
12:
13: def main():
14:
15:     try:
16:         with open('books.xml', 'r', encoding='utf-8') as xml_file:
17:             xml_doc = xml_file.read()
18:
19:             xml_dom_tree = xml.dom.minidom.parseString(xml_doc)
20:
21:             books_node = xml_dom_tree.childNodes[4]
22:
23:             i = 1
24:             while i < len(books_node.childNodes):
25:                 book_node = books_node.childNodes[i]
26:
27:                 isbn_node = book_node.childNodes[1]
28:                 author_node = book_node.childNodes[3]
29:                 title_node = book_node.childNodes[5]
30:                 price_node = book_node.childNodes[7]
31:
32:                 isbn = isbn_node.childNodes[0].nodeValue
33:                 author = author_node.childNodes[0].nodeValue
34:                 title = title_node.childNodes[0].nodeValue
35:                 price = price_node.childNodes[0].nodeValue
36:
37:                 currency = \
38:                     price_node.attributes.getNamedItem('currency').nodeValue
39:
40:                 print('ISBN:', isbn)
41:                 print('Author:', author)
42:                 print('Title:', title)
43:                 print('Price:', price, currency)
44:                 print()
45:                 i += 2
46:
47:     except Exception as ex:
48:         print(ex, file=sys.stderr)
49:         sys.exit(1)
50:
51: #-----
52:
53: if __name__ == '__main__':
54:     main()

```

## writebooks.js (Page 1 of 1)

```

1: //-----
2: // writebooks.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 parseFile(err, xmlDoc) {
14:     if (err) {
15:         process.stderr.write(err.message + '\n');
16:         process.exit(1);
17:     }
18:
19:     let parser = new xmldom.DOMParser();
20:     let xmlDomTree = parser.parseFromString(xmlDoc, 'text/xml');
21:
22:     let booksNode = xmlDomTree.childNodes[10];
23:
24:     let i = 1;
25:     while (i < booksNode.childNodes.length) {
26:         let bookNode = booksNode.childNodes[i];
27:
28:         let isbnNode = bookNode.childNodes[1];
29:         let authorNode = bookNode.childNodes[3];
30:         let titleNode = bookNode.childNodes[5];
31:         let priceNode = bookNode.childNodes[7];
32:
33:         let isbn = isbnNode.childNodes[0].nodeValue;
34:         let author = authorNode.childNodes[0].nodeValue;
35:         let title = titleNode.childNodes[0].nodeValue;
36:         let price = priceNode.childNodes[0].nodeValue;
37:
38:         let currency =
39:             priceNode.attributes.getNamedItem('currency').nodeValue;
40:
41:         process.stdout.write('ISBN: ' + isbn + '\n');
42:         process.stdout.write('Author: ' + author + '\n');
43:         process.stdout.write('Title: ' + title + '\n');
44:         process.stdout.write('Price: ' + price + ' ' +
45:             currency + '\n');
46:         process.stdout.write('\n');
47:         i += 2;
48:     }
49: }
50:
51: //-----
52:
53: function main() {
54:     fs.readFile('books.xml', 'utf-8', parseFile);
55: }
56:
57: if (require.main === module)
58:     main();

```

## writebooksshort.py (Page 1 of 1)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # writebooksshort.py
5: # Author: Bob Dondero
6: #-----
7:
8: import sys
9: import xml.dom.minidom
10:
11: #-----
12:
13: def main():
14:
15:     try:
16:         with open('books.xml', 'r', encoding='utf-8') as xml_file:
17:             xml_doc = xml_file.read()
18:
19:             xml_dom_tree = xml.dom.minidom.parseString(xml_doc)
20:
21:             book_nodes = xml_dom_tree.getElementsByTagName('book')
22:
23:             for book_node in book_nodes:
24:                 isbn_node = book_node.getElementsByTagName('isbn')[0]
25:                 author_node = book_node.getElementsByTagName('author')[0]
26:                 title_node = book_node.getElementsByTagName('title')[0]
27:                 price_node = book_node.getElementsByTagName('price')[0]
28:
29:                 isbn = isbn_node.childNodes[0].nodeValue
30:                 author = author_node.childNodes[0].nodeValue
31:                 title = title_node.childNodes[0].nodeValue
32:                 price = price_node.childNodes[0].nodeValue
33:                 currency = price_node.attributes.getNamedItem(
34:                     'currency').nodeValue
35:
36:                 print('ISBN:', isbn)
37:                 print('Author:', author)
38:                 print('Title:', title)
39:                 print('Price:', price, currency)
40:                 print()
41:
42:             except Exception as ex:
43:                 print(ex, file=sys.stderr)
44:                 sys.exit(1)
45:
46: #-----
47:
48: if __name__ == '__main__':
49:     main()

```

## writebooksshort.js (Page 1 of 1)

```

1: //-----
2: // writebooksshort.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 parseFile(err, xmlDoc) {
14:     if (err) {
15:         process.stderr.write(err.message + '\n');
16:         process.exit(1);
17:     }
18:
19:     let parser = new xmldom.DOMParser();
20:     let xmlDomTree = parser.parseFromString(xmlDoc, 'text/xml');
21:
22:     let bookNodes = xmlDomTree.getElementsByTagName('book');
23:
24:     for (let i = 0; i < bookNodes.length; i++) {
25:         let bookNode = bookNodes[i];
26:
27:         let isbnNode = bookNode.getElementsByTagName('isbn')[0];
28:         let authorNode = bookNode.getElementsByTagName('author')[0];
29:         let titleNode = bookNode.getElementsByTagName('title')[0];
30:         let priceNode = bookNode.getElementsByTagName('price')[0];
31:
32:         let isbn = isbnNode.childNodes[0].nodeValue;
33:         let author = authorNode.childNodes[0].nodeValue;
34:         let title = titleNode.childNodes[0].nodeValue;
35:         let price = priceNode.childNodes[0].nodeValue;
36:         let currency =
37:             priceNode.attributes.getNamedItem('currency').nodeValue;
38:
39:         process.stdout.write('ISBN: ' + isbn + '\n');
40:         process.stdout.write('Author: ' + author + '\n');
41:         process.stdout.write('Title: ' + title + '\n');
42:         process.stdout.write('Price: ' + price + ' ' + currency + '\n');
43:         process.stdout.write('\n');
44:     }
45: }
46:
47: //-----
48:
49: function main() {
50:     fs.readFile('books.xml', 'utf-8', parseFile);
51: }
52:
53: if (require.main === module)
54:     main();

```

## roundtripxml.py (Page 1 of 1)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # roundtripxml.py
5: # Author: Bob Dondero
6: #-----
7:
8: import sys
9: import xml.dom.minidom
10:
11: #-----
12:
13: def main():
14:
15:     if len(sys.argv) != 2:
16:         print('Usage: ' + sys.argv[0] + ' file', file=sys.stderr)
17:         sys.exit(1)
18:
19:     xml_file_name = sys.argv[1]
20:
21:     try:
22:
23:         with open(xml_file_name, 'r', encoding='utf-8') as xml_file:
24:             xml_doc = xml_file.read()
25:
26:             # XML to DOM tree:
27:             xml_dom_tree = xml.dom.minidom.parseString(xml_doc)
28:
29:             # DOM tree to XML:
30:             xml_doc = xml_dom_tree.toxml()
31:
32:             # Show that it worked:
33:             print(xml_doc)
34:
35:     except Exception as ex:
36:         print(ex, file=sys.stderr)
37:         sys.exit(1)
38:
39: #-----
40:
41: if __name__ == '__main__':
42:     main()

```

## roundtripxml.js (Page 1 of 1)

```

1: //-----
2: // roundtripxml.js
3: // Author: Bob Dondero
4: //-----
5:
6: // WARNING: DOESN'T HANDLE PROCESSING INSTRUCTION!!!
7:
8: 'use strict';
9:
10: const fs = require('fs');
11: const xmldom = require('xmldom');
12: const xmlserializer = require('xmlserializer');
13:
14: //-----
15:
16: function parseFile(err, xmlDoc) {
17:     if (err) {
18:         process.stderr.write(err.message + '\n');
19:         process.exit(1);
20:     }
21:
22:     // XML to DOM tree:
23:     let parser = new xmldom.DOMParser();
24:     let xmlDocTree = parser.parseFromString(xmlDoc, 'text/xml');
25:
26:     // DOM tree to XML:
27:     // WARNING: DOESN'T HANDLE PROCESSING INSTRUCTION!!!
28:     xmlDoc = xmlserializer.serializeToString(xmlDocTree);
29:
30:     // Show that it worked:
31:     process.stdout.write(xmlDoc);
32: }
33:
34: //-----
35:
36: function main() {
37:     if (process.argv.length !== 3) {
38:         process.stderr.write('Usage: ' + process.argv[0] + ' '
39:             + process.argv[1] + ' file\n');
40:         process.exit(1);
41:     }
42:
43:     let xmlFileName = process.argv[2];
44:
45:     fs.readFile(xmlFileName, 'utf-8', parseFile);
46: }
47:
48: if (require.main === module)
49:     main();

```



## PennyXml/penny.py (Page 1 of 1)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # penny.py
5: # Author: Bob Dondero
6: #-----
7:
8: import json
9: import flask
10: import database
11:
12: #-----
13:
14: app = flask.Flask(__name__, template_folder='.')
15:
16: #-----
17:
18: @app.route('/', methods=['GET'])
19: @app.route('/index', methods=['GET'])
20: def index():
21:
22:     return flask.send_file('index.html')
23:
24: #-----
25:
26: @app.route('/searchresults', methods=['GET'])
27: def search_results():
28:
29:     author = flask.request.args.get('author')
30:     if author is None:
31:         author = ''
32:     author = author.strip()
33:
34:     if author == '':
35:         books = []
36:     else:
37:         books = database.get_books(author) # Exception handling omitted
38:
39:     xml_doc = '<?xml version="1.0"?>'
40:     xml_doc += '<books>'
41:     for book in books:
42:         xml_doc += '<book>'
43:         xml_doc += '<isbn><![CDATA[' + book['isbn'] + ']]></isbn>'
44:         xml_doc += '<author><![CDATA[' + book['author'] + ']]></author>'
45:         xml_doc += '<title><![CDATA[' + book['title'] + ']]></title>'
46:         xml_doc += '</book>'
47:     xml_doc += '</books>'
48:
49:     response = flask.make_response(xml_doc)
50:     response.headers['Content-Type'] = 'application/xml'
51:     return response

```

## blank (Page 1 of 1)

```

1: This page is intentionally blank.

```

## PennyXml/index.html (Page 1 of 2)

```

1: <!DOCTYPE html>
2: <html>
3:   <head>
4:     <title>Penny.com</title>
5:   </head>
6:
7:   <body>
8:     <hr>
9:     Good <span id="ampmSpan"></span> and welcome to
10:    <strong>Penny.com</strong>
11:    <hr>
12:
13:    <h1>Author Search</h1>
14:    Please enter an author name:
15:    <input type="text" id="authorInput" autoFocus>
16:    <hr>
17:    <div id="resultsDiv"></div>
18:
19:    <hr>
20:    Date and time: <span id="datetimeSpan"></span><br>
21:    Created by <a href="https://www.cs.princeton.edu/~rdonero">
22:    Bob Donero</a>
23:    <hr>
24:
25:    <script src=
26:    "https://cdn.jsdelivr.net/npm/mustache@4.2.0/mustache.min.js">
27:    </script>
28:
29:    <script>
30:
31:      'use strict';
32:
33:      function getAmPm() {
34:        let dateTime = new Date();
35:        let hours = dateTime.getHours();
36:        let amPm = (hours < 12) ? 'morning' : 'afternoon';
37:        let ampMspan = document.getElementById('ampmSpan');
38:        ampMspan.innerHTML = amPm;
39:      }
40:
41:      function getDateTime() {
42:        let dateTime = new Date();
43:        let datetimeSpan =
44:        document.getElementById('datetimeSpan');
45:        datetimeSpan.innerHTML = dateTime.toLocaleString();
46:      }
47:
48:      function convertToBooks(xmlDomTree) {
49:        let books = [];
50:        let bookNodes = xmlDomTree.getElementsByTagName('book');
51:        for (let bookNode of bookNodes) {
52:          let isbn = bookNode.getElementsByTagName('isbn')[0]
53:            .childNodes[0].nodeValue;
54:          let author = bookNode.getElementsByTagName('author')[0]
55:            .childNodes[0].nodeValue;
56:          let title = bookNode.getElementsByTagName('title')[0]
57:            .childNodes[0].nodeValue;
58:          let book =
59:            {'isbn': isbn, 'author': author, 'title': title};
60:          books.push(book);
61:        }
62:        return books;
63:      }
64:
65:      function convertToHtml(books) {

```

## PennyXml/index.html (Page 2 of 2)

```

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 setup() {
118:        getAmPm();
119:        window.setInterval(getAmPm, 1000);
120:        getDateTime();
121:        window.setInterval(getDateTime, 1000);
122:        let authorInput = document.getElementById('authorInput');
123:        authorInput.addEventListener('input', debouncedGetResults);
124:      }
125:
126:      document.addEventListener('DOMContentLoaded', setup);
127:
128:    </script>
129:  </body>
130: </html>

```

## writeauthorssax.py (Page 1 of 1)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # writeauthorssax.py
5: # Author: Bob Dondero
6: #-----
7:
8: import sys
9: import xml.sax
10:
11: #-----
12:
13: class AuthorsHandler(xml.sax.ContentHandler):
14:
15:     def __init__(self):
16:         xml.sax.ContentHandler.__init__(self)
17:         self._in_author_element = False
18:         self._author_name = ''
19:
20:     def startElement(self, name, attrs):
21:         if name == 'author':
22:             self._in_author_element = True
23:
24:     def characters(self, content):
25:         if self._in_author_element:
26:             self._author_name += content
27:
28:     def endElement(self, name):
29:         if name == 'author':
30:             print(self._author_name)
31:             self._author_name = ''
32:             self._in_author_element = False
33:
34: #-----
35:
36: def main():
37:
38:     try:
39:         with open('books.xml', 'r', encoding='utf-8') as xml_file:
40:             xml_doc = xml_file.read()
41:
42:             xml.sax.parseString(xml_doc, AuthorsHandler())
43:
44:     except Exception as ex:
45:         print(ex, file=sys.stderr)
46:         sys.exit(1)
47:
48: #-----
49:
50: if __name__ == '__main__':
51:     main()

```

## writeauthorssax.js (Page 1 of 1)

```

1: //-----
2: // writeauthorssax.js
3: // Author: Bob Dondero
4: //-----
5:
6: 'use strict';
7:
8: const fs = require('fs');
9: const saxparser = require('sax-parser');
10:
11: //-----
12:
13: function parse(cb) {
14:     let authorName = '';
15:     let inAuthorElement = false;
16:
17:     cb.onStartElementNS(function(elem, attrs, prefix, uri, namespace) {
18:         if (elem === 'author')
19:             inAuthorElement = true;
20:     });
21:
22:     cb.onCharacters(function(chars) {
23:         if (inAuthorElement)
24:             authorName += chars;
25:     });
26:
27:     cb.onEndElementNS(function(elem, prefix, uri) {
28:         if (elem === 'author') {
29:             process.stdout.write(authorName + '\n');
30:             authorName = '';
31:             inAuthorElement = false;
32:         }
33:     });
34: }
35:
36: //-----
37:
38: function parseFile(err, xmlDoc) {
39:     if (err) {
40:         process.stderr.write(err.message + '\n');
41:         process.exit(1);
42:     }
43:
44:     let parser = new saxparser.SaxParser(parse);
45:     parser.parseString(xmlDoc);
46: }
47:
48: //-----
49:
50: function main() {
51:     fs.readFile('books.xml', 'utf-8', parseFile);
52: }
53:
54: if (require.main === module)
55:     main(process.argv);

```

## booksmalformed.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</prices>
26:   </book>
27: </books>

```

## checkxml.py (Page 1 of 1)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # checkxml.py
5: # Author: Bob Dondero
6: #-----
7:
8: import sys
9: import xml.dom.minidom
10:
11: #-----
12:
13: def main():
14:
15:     if len(sys.argv) != 2:
16:         print('Usage: ' + sys.argv[0] + ' file', file=sys.stderr)
17:         sys.exit(1)
18:
19:     xml_file_name = sys.argv[1]
20:
21:     try:
22:         with open(xml_file_name, 'r', encoding='utf-8') as xml_file:
23:             xml_doc = xml_file.read()
24:
25:             xml.dom.minidom.parseString(xml_doc)
26:             print('The document is well-formed.')
27:
28:     except Exception as ex:
29:         print(ex, file=sys.stderr)
30:         sys.exit(1)
31:
32: #-----
33:
34: if __name__ == '__main__':
35:     main()

```

## books.dtd (Page 1 of 1)

```

1: <!-- ===== -->
2: <!-- books.dtd -->
3: <!-- Author: Bob Dondero -->
4: <!-- ===== -->
5:
6: <!ELEMENT books (book*)>
7: <!ELEMENT book (isbn, author, title, price)>
8: <!ELEMENT isbn (#PCDATA)>
9: <!ELEMENT author (#PCDATA)>
10: <!ELEMENT title (#PCDATA)>
11: <!ELEMENT price (#PCDATA)>
12: <!ATTLIST price currency CDATA #REQUIRED>

```

## booksusingdtd.xml (Page 1 of 1)

```

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

```

## booksusingdtdinvalid.xml (Page 1 of 1)

```

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

```

## checkxmlusingdtd.py (Page 1 of 1)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # checkxmlusingdtd.py
5: # Author: Bob Dondero
6: #-----
7:
8: import sys
9: from lxml import etree
10:
11: #-----
12:
13: def main():
14:
15:     if len(sys.argv) != 2:
16:         print('Usage: ' + sys.argv[0] + ' file', file=sys.stderr)
17:         sys.exit(1)
18:
19:     xml_file_name = sys.argv[1]
20:
21:     try:
22:         with open(xml_file_name, 'r', encoding='utf-8') as xml_file:
23:             xml_doc = xml_file.read()
24:
25:             parser = etree.XMLParser(dtd_validation=True)
26:             etree.fromstring(xml_doc, parser)
27:             print('The document is well-formed and valid.')
28:
29:     except Exception as ex:
30:         print(ex, file=sys.stderr)
31:         sys.exit(1)
32:
33: #-----
34:
35: if __name__ == '__main__':
36:     main()
37:
38: #-----
39:
40: # Example executions:
41:
42: # $ python checkxmlusingdtd.py booksusingdtd.xml
43: # The document is well-formed and valid.
44:
45: # $ python checkxmlusingdtd.py booksusingdtdinvalid.xml
46: # Element book content does not follow the DTD, expecting
47: # (author , title , price), got (author price title ),
48: # line 25, column 11 (booksusingdtdinvalid.xml, line 25)
49:

```

## 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)

```

1: <?xml version="1.0"?>
2:
3: <!-- ===== -->
4: <!-- booksusingschemabad.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>234</isbn>
24:     <author>Sedgewick</author>
25:     <price currency="dollars">61.59</price>
26:     <title>Algorithms in C</title>
27:   </book>
28: </books>

```

## checkxmlusingxsd.py (Page 1 of 1)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # checkxmlusingxsd.py
5: # Author: Bob Dondero
6: #-----
7:
8: import sys
9: from lxml import etree
10:
11: #-----
12:
13: def main():
14:
15:     if len(sys.argv) != 3:
16:         print('Usage: ' + sys.argv[0] + ' xmlfile schemafile',
17:               file=sys.stderr)
18:         sys.exit(1)
19:
20:     xml_file_name = sys.argv[1]
21:     schema_file_name = sys.argv[2]
22:
23:     try:
24:         with open(schema_file_name, 'r', encoding='utf-8') \
25:             as schema_file:
26:             schema_doc = schema_file.read()
27:
28:         with open(xml_file_name, mode='r', encoding='utf-8') \
29:             as xml_file:
30:             xml_doc = xml_file.read()
31:
32:         schema = etree.XMLSchema(etree.XML(schema_doc))
33:         parser = etree.XMLParser(schema=schema)
34:         etree.fromstring(xml_doc, parser)
35:
36:         print('The document is well-formed and valid.')
37:
38:     except Exception as ex:
39:         print(ex, file=sys.stderr)
40:         sys.exit(1)
41:
42: #-----
43:
44: if __name__ == '__main__':
45:     main()
46:
47: #-----
48:
49: # Example executions:
50:
51: # $ python checkxmlusingxsd.py booksusingxsd.xml books.xsd
52: # The document is well-formed and valid.
53:
54: # $ python checkxmlusingxsd.py booksusingxsdinvalid.xml books.xsd
55: # Element 'price': This element is not expected.
56: # Expected is ( title ). (<string>, line 0)
57:

```