books.json (Page 1 of 1)

```
[
   {  "author": "Kernighan",
      "title": "The Practice of Programming",
      "price": 40.74,
      "currency": "dollars"
   },
   {  "author": "Kernighan",
      "title": "The C Programming Language",
      "price": 24.99,
      "currency": "dollars"
   },
   {  "author": "Sedgewick",
      "title": "Algorithms in C",
      "price": 61.59,
      "currency": "dollars"
   }
]
```

blank (Page 1 of 1)

This page is intentionally blank.

writebooksjson.py (Page 1 of 1)

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

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

import sys
import json

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

def main():

    try:
        with open('books.json', 'r', encoding='UTF-8') as json_file:
            json_doc = json_file.read()

        books = json.loads(json_doc)

        for book in books:
            print('Author:', book['author'])
            print('Title:', book['title'])
            print('Price:', book['price'], book['currency'])
            print()

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

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

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

writebooksjson.js (Page 1 of 1)

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

'use strict';

let fs = require('fs');

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

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

   // JSON to JavaScript:
   let books = JSON.parse(jsonDoc);

   for (let book of books) {
      process.stdout.write('Author: ' + book.author + '\n');
      process.stdout.write('Title: ' + book.title + '\n');
      process.stdout.write('Price: ' + book.price + ' ' +
         book.currency + '\n');
      process.stdout.write('\n');
   }
}

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

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

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

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

roundtripjson.py (Page 1 of 1)

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

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

import sys
import json

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

def main():

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

    json_file_name = sys.argv[1]

    try:
        with open(json_file_name, 'r', encoding='UTF-8') as json_file:
            json_doc = json_file.read()

        # JSON to Python object tree:
        object_tree = json.loads(json_doc)

        # Python object tree to JSON:
        json_doc = json.dumps(object_tree)

        # Show that it worked:
        print(json_doc)

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

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

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

roundtripjson.js (Page 1 of 1)

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

'use strict';

const fs = require('fs');

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

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

   // JSON to JavaScript object tree:
   let object_tree = JSON.parse(jsonDoc);

   // JavaScript object tree to JSON:
   jsonDoc = JSON.stringify(object_tree);

   // Show that it worked:
   process.stderr.write(jsonDoc + '\n');
}

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

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

   let jsonFileName = process.argv[2];

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

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

PennyJQuery3/book.py (Page 1 of 1)

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

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

class Book:

    def __init__(self, author, title, price):
        self._author = author
        self._title = title
        self._price = price

    def get_author(self):
        return self._author

    def get_title(self):
        return self._title

    def get_price(self):
        return self._price

    def to_tuple(self):
        return (self._author, self._title, self._price)

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

def _test():
    book = Book('Kernighan', 'The Practice of Programming', 40.74)
    print(book.get_author())
    print(book.get_title())
    print(book.get_price())
    print(book.to_tuple())

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

blank (Page 1 of 1)

```
This page is intentionally blank.
```

PennyJQuery3/penny.py (Page 1 of 1)

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

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

import flask
import database

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

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

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

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

    html_code = flask.render_template('index.html')
    response = flask.make_response(html_code)
    return response

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

@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

    html_code = flask.render_template('books.html', books=books)
    response = flask.make_response(html_code)
    return response
```

PennyJQuery3/books.html (Page 1 of 1)

```
{% for book in books: %}
    <strong>{{book.get_author()}}</strong>:
    {{book.get_title()}}
    (${{'%.2f' % book.get_price()}})
    <br>
{% endfor %}
```

PennyJQuery3/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/~rdondero">
22:       Bob Dondero</a>
23:       <hr>
24:
25:       <script src=
26: "https://ajax.googleapis.com/ajax/libs/jquery/3.7.1/jquery.min.js">
27:       </script>
28:
29:       <!-- <script src="static/jquery-3.7.1.min.js"></script> -->
30:
31:       <script>
32:
33:          'use strict';
34:
35:          function getAmPm() {
36:             let dateTime = new Date();
37:             let hours = dateTime.getHours();
38:             let amPm = 'morning';
39:             if (hours >= 12)
40:                amPm = 'afternoon';
41:             $('#ampmSpan').html(amPm);
42:          }
43:
44:          function getDateTime() {
45:             let dateTime = new Date();
46:             $('#datetimeSpan').html(dateTime.toLocaleString());
47:          }
48:
49:          function handleResponse(data) {
50:             $('#resultsDiv').html(data);
51:          }
52:
53:          function handleError(request) {
54:             if (request.statusText !== 'abort')
55:                alert('Error: Failed to fetch data from server');
56:          }
57:
58:          let request = null;
59:
60:          function getResults() {
61:             let author = $('#authorInput').val();
62:             let encodedAuthor = encodeURIComponent(author);
63:             let url = '/searchresults?author=' + encodedAuthor;
64:             if (request != null)
65:                request.abort();
```

PennyJQuery3/index.html (Page 2 of 2)

```
66:             let requestData = {
67:                type: 'GET',
68:                url: url,
69:                success: handleResponse,
70:                error: handleError
71:             };
72:             request = $.ajax(requestData);
73:          }
74:
75:          let timer = null;
76:
77:          function debouncedGetResults() {
78:             clearTimeout(timer);
79:             timer = setTimeout(getResults, 500);
80:          }
81:
82:          function setup() {
83:             getAmPm();
84:             window.setInterval(getAmPm, 1000);
85:             getDateTime();
86:             window.setInterval(getDateTime, 1000);
87:             $('#authorInput').on('input', debouncedGetResults);
88:          }
89:
90:          $('document').ready(setup);
91:
92:       </script>
93:    </body>
94: </html>
```

PennyJQueryXml/book.py (Page 1 of 1)

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

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

class Book:

    def __init__(self, author, title, price):
        self._author = author
        self._title = title
        self._price = price

    def get_author(self):
        return self._author

    def get_title(self):
        return self._title

    def get_price(self):
        return self._price

    def to_xml(self):
        pattern = '<book>'
        pattern += '<author><![CDATA[%s]]></author>'
        pattern += '<title><![CDATA[%s]]></title>'
        pattern += '<price><![CDATA[%f]]></price>'
        pattern += '</book>'
        return pattern % (self._author, self._title, self._price)

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

def _test():
    book = Book('Kernighan', 'The Practice of Programming', 40.74)
    print(book.to_xml())

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

PennyJQueryXml/penny.py (Page 1 of 1)

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

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

import flask
import database

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

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

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

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

    html_code = flask.render_template('index.html')
    response = flask.make_response(html_code)
    return response

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

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

    xml_doc = '<?xml version="1.0"?>'

    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.to_xml()
    xml_doc += '</books>'

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

PennyJQueryXml/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://ajax.googleapis.com/ajax/libs/jquery/3.7.1/jquery.min.js">
      </script>

      <!-- <script src="static/jquery-3.7.1.min.js"></script> -->

      <script>

         'use strict';

         function getAmPm() {
            let dateTime = new Date();
            let hours = dateTime.getHours();
            let amPm = 'morning';
            if (hours >= 12)
               amPm = 'afternoon';
            $('#ampmSpan').html(amPm);
         }

         function getDateTime() {
            let dateTime = new Date();
            $('#datetimeSpan').html(dateTime.toLocaleString());
         }

         function handleResponse(xmlDomTree) {
            try {
               let resultsNode = $('#resultsDiv');
               resultsNode.text('');

               let books = xmlDomTree.getElementsByTagName('book');
               for (let book of books) {
                  let authorNode = book.childNodes[0];
                  let author = authorNode.childNodes[0].nodeValue

                  let titleNode = book.childNodes[1];
                  let title = titleNode.childNodes[0].nodeValue

                  let priceNode = book.childNodes[2];
                  let price = priceNode.childNodes[0].nodeValue;
                  price = parseFloat(price).toFixed(2);
```

PennyJQueryXml/index.html (Page 2 of 2)

```
                  let strongNode = $('<strong>', {text: author});
                  let textNode = document.createTextNode(': ' + title
                     + ' ($' + price + ')');
                  let brNode = $('<br>');

                  resultsNode.append(strongNode);
                  resultsNode.append(textNode);
                  resultsNode.append(brNode);
               }
            }
            catch (e) {
               $('#resultsDiv').text(e);
            }
         }

         function handleError(request) {
            if (request.statusText !== 'abort')
               alert('Error: Failed to fetch data from server');
         }

         let request = null;

         function getResults() {
            let author = $('#authorInput').val();
            let encodedAuthor = encodeURIComponent(author);
            let url = '/searchresults?author=' + encodedAuthor;
            if (request != null)
               request.abort();
            let requestData = {
               type: 'GET',
               url: url,
               success: handleResponse,
               error: handleError
            };
            request = $.ajax(requestData);
         }

         let timer = null;

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

         function setup() {
            getAmPm();
            window.setInterval(getAmPm, 1000);
            getDateTime();
            window.setInterval(getDateTime, 1000);
            $('#authorInput').on('input', debouncedGetResults);
         }

         $('document').ready(setup);

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

PennyJQueryJson/book.py (Page 1 of 1)

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

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

class Book:

    def __init__(self, author, title, price):
        self._author = author
        self._title = title
        self._price = price

    def get_author(self):
        return self._author

    def get_title(self):
        return self._title

    def get_price(self):
        return self._price

    def to_dict(self):
        return {'author': self._author, 'title': self._title,
            'price': self._price}

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

def _test():
    book = Book('Kernighan', 'The Practice of Programming', 40.74)
    print(book.to_dict())

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

PennyJQueryJson/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():

    html_code = flask.render_template('index.html')
    response = flask.make_response(html_code)
    return response

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

@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

    dicts = []
    for book in books:
        dicts.append(book.to_dict())
    json_doc = json.dumps(dicts)

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

PennyJQueryJson/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://ajax.googleapis.com/ajax/libs/jquery/3.7.1/jquery.min.js">
      </script>

      <!-- <script src="static/jquery-3.7.1.min.js"></script> -->

      <script>

         'use strict';

         function getAmPm() {
            let dateTime = new Date();
            let hours = dateTime.getHours();
            let amPm = 'morning';
            if (hours >= 12)
               amPm = 'afternoon';
            $('#ampmSpan').html(amPm);
         }

         function getDateTime() {
            let dateTime = new Date();
            $('#datetimeSpan').html(dateTime.toLocaleString());
         }

         function handleResponse(books) {
            try {
               let resultsNode = $('#resultsDiv');
               resultsNode.text('');

               for (let book of books) {
                  let author = book['author'];
                  let title = book['title'];
                  let price = book['price'].toFixed(2);

                  let strongNode = $('<strong>', {text: author});
                  let textNode = document.createTextNode(': ' + title
                     + ' ($' + price + ')');
                  let brNode = $('<br>');

                  resultsNode.append(strongNode);
                  resultsNode.append(textNode);
```

PennyJQueryJson/index.html (Page 2 of 2)

```
                  resultsNode.append(brNode);
               }
            }
            catch (e) {
               $('#resultsDiv').text(e);
            }
         }

         function handleError(request) {
            if (request.statusText !== 'abort')
               alert('Error: Failed to fetch data from server');
         }

         let request = null;

         function getResults() {
            let author = $('#authorInput').val();
            let encodedAuthor = encodeURIComponent(author);
            let url = '/searchresults?author=' + encodedAuthor;
            if (request != null)
               request.abort();
            let requestData = {
               type: 'GET',
               url: url,
               success: handleResponse,
               error: handleError
            };
            request = $.ajax(requestData);
         }

         let timer = null;

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

         function setup() {
            getAmPm();
            window.setInterval(getAmPm, 1000);
            getDateTime();
            window.setInterval(getDateTime, 1000);
            $('#authorInput').on('input', debouncedGetResults);
         }

         $('document').ready(setup);

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

booksbyauthorjson.py (Page 1 of 1)

```
 1: #!/usr/bin/env python
 2:
 3: #-----------------------------------------------------------------------
 4: # booksbyauthorjson.py
 5: # Author: Bob Dondero
 6: #-----------------------------------------------------------------------
 7:
 8: import sys
 9: import urllib.request
10: import json
11:
12: def main():
13:
14:     if len(sys.argv) != 4:
15:         print('Usage: python %s host port author' % sys.argv[0])
16:         sys.exit(1)
17:
18:     try:
19:         host = sys.argv[1]
20:         port = sys.argv[2]
21:         author = sys.argv[3]
22:
23:         url = 'http://' + host + ':' + port
24:         url += '/searchresults?author=' + author
25:         with urllib.request.urlopen(url) as in_flo:
26:             json_bytes = in_flo.read()
27:
28:         json_doc = json_bytes.decode('utf-8')
29:         books = json.loads(json_doc)
30:         for book in books:
31:             print(book['author'])
32:             print(book['title'])
33:             print(book['price'])
34:             print()
35:
36:     except Exception as ex:
37:         print(ex, file=sys.stderr)
38:         sys.exit(1)
39:
40: if __name__ == '__main__':
41:     main()
```

blank (Page 1 of 1)

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

booksmalformed.json (Page 1 of 1)

```
[
    {  "author": "Kernighan",
       "title": "The Practice of Programming",
       "price": 40.74,
       "currency": "dollars"
    },
    {  "author": "Kernighan",
       "title": "The C Programming Language",
       "price": 24.99,
       "currency": "dollars"
    },
    {  "author": "Sedgewick",
       "title": "Algorithms in C",
       "price": 61.59
       "currency": "dollars"
    }
]
```

checkjson.py (Page 1 of 1)

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

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

import sys
import json

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

def main():

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

    json_file_name = sys.argv[1]

    try:
        with open(json_file_name, 'r', encoding='UTF-8') as json_file:
            json_doc = json_file.read()

        json.loads(json_doc)

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

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

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

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

books.schema.json (Page 1 of 1)

```
{
   "type": "array",
   "items": { "$ref": "#/definitions/book" },

   "definitions":
   {
      "book":
      {
         "type": "object",
         "required": [ "author", "title", "price", "currency" ],
         "properties":
         {
            "author": { "type": "string"},
            "title": { "type": "string" },
            "price": { "type": "number" },
            "currency": { "type": "string" }
         }
      }
   }
}
```

booksinvalid.json (Page 1 of 1)

```
[
   {  "author": "Kernighan",
      "title": "The Practice of Programming",
      "price": 40.74,
      "currency": "dollars"
   },
   {  "author": "Kernighan",
      "title": "The C Programming Language",
      "price": 24.99,
      "currency": "dollars"
   },
   {  "author": "Sedgewick",
      "title": "Algorithms in C",
      "price": 61.59
   }
]
```

checkjsonusingschema.py (Page 1 of 1)

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

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

import sys
import json
import jsonschema

def main():

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

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

    try:
        with open(json_file_name, mode='r', encoding='utf-8') \
            as json_file:
            json_doc = json_file.read()

        object_tree = json.loads(json_doc)
        print('The document is well-formed.')

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

        schema = json.loads(schema_doc)

        jsonschema.validate(instance=object_tree, schema=schema)
        print('The document is valid.')

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

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