

## PennyViewport/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__)
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:     json_doc = json.dumps(books)
40:     response = flask.make_response(json_doc)
41:     response.headers['Content-Type'] = 'application/json'
42:     return response
```

## blank (Page 1 of 1)

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

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

```

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

```

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

```

66:         if (this.status !== 200) {
67:           alert('Error: Failed to fetch data from server');
68:           return;
69:         }
70:         let books = JSON.parse(this.response);
71:         let html = convertToHtml(books);
72:         let resultsDiv = document.getElementById('resultsDiv');
73:         resultsDiv.innerHTML = html;
74:       }
75:
76:       function handleError(request) {
77:         alert('Error: Failed to fetch data from server');
78:       }
79:
80:       let request = null;
81:
82:       function getResults() {
83:         let authorInput = document.getElementById('authorInput');
84:         let author = authorInput.value;
85:         let encodedAuthor = encodeURIComponent(author);
86:         let url = '/searchresults?author=' + encodedAuthor;
87:         if (request !== null)
88:           request.abort();
89:         request = new XMLHttpRequest();
90:         request.onload = handleResponse;
91:         request.onerror = handleError;
92:         request.open('GET', url);
93:         request.send();
94:       }
95:
96:       let timer = null;
97:
98:       function debouncedGetResults() {
99:         clearTimeout(timer);
100:        timer = setTimeout(getResults, 500);
101:      }
102:
103:      function setup() {
104:        getAmPm();
105:        window.setInterval(getAmPm, 1000);
106:        getDateTime();
107:        window.setInterval(getDateTime, 1000);
108:        let authorInput = document.getElementById('authorInput');
109:        authorInput.addEventListener('input', debouncedGetResults);
110:      }
111:
112:      document.addEventListener('DOMContentLoaded', setup);
113:
114:    </script>
115:  </body>
116: </html>

```

**PennyCss/static/penny.css (Page 1 of 1)**

```
1: /*-----*/
2: /* penny.css */
3: /* Author: Bob Dondero */
4: /*-----*/
5:
6: /* Element rules */
7:
8: body {font-family:Arial, Helvetica, sans-serif; font-size:16px;}
9:
10: input[type="text"] {font-size:16px;}
11:
12: a {color:#aaaaff;}
13:
14: table {border-spacing: 16px;}
15:
16: /*-----*/
17:
18: /* Class rules */
19:
20: .header {background-color:#295078; text-align:center; color:white;}
21:
22: .footer {background-color:#295078; text-align:center; color:white;}
23:
24: .grayborder {border-style:solid; border-color:gray; border-width:1px;}
```

**blank (Page 1 of 1)**

1: This page is intentionally blank.

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

```

1: <!DOCTYPE html>
2: <html>
3:
4:   <head>
5:     <title>Penny.com</title>
6:     <meta name="viewport"
7:       content="width=device-width, initial-scale=1">
8:     <link rel="stylesheet" href="static/penny.css">
9:   </head>
10:
11:   <body>
12:     <div class="header">
13:       <h2>Penny.com</h2>
14:       <h3>Good <span id="ampmSpan"></span> and welcome to
15:         Penny.com</h3>
16:     </div>
17:
18:     <br>
19:
20:     <table width="100%">
21:       <tbody>
22:         <tr>
23:           <td width="25%"><strong>Author name:</strong></td>
24:           <td width="75%">
25:             <input type="text" id="authorInput" autoFocus>
26:           </td>
27:         </tr>
28:         <tr>
29:           <td width="25%"><strong>Books:</strong></td>
30:           <td width="75%" class="grayborder" id="resultsTd"></td>
31:         </tr>
32:       </tbody>
33:     </table>
34:
35:     <br>
36:
37:     <div class="footer">
38:       Date and time: <span id="datetimeSpan"></span><br>
39:       Created by <a href="https://www.cs.princeton.edu/~rdondero">
40:         Bob Dondero</a>
41:     </div>
42:
43:     <script src=
44:       "https://cdn.jsdelivr.net/npm/mustache@4.2.0/mustache.min.js">
45:     </script>
46:
47:     <script>
48:       'use strict';
49:
50:       function getAmPm() {
51:         let dateTime = new Date();
52:         let hours = dateTime.getHours();
53:         let amPm = (hours < 12) ? 'morning' : 'afternoon';
54:         let ampMspan = document.getElementById('ampmSpan');
55:         ampMspan.innerHTML = amPm;
56:       }
57:
58:       function getDateime() {
59:         let dateTime = new Date();
60:         let datetimeSpan =
61:           document.getElementById('datetimeSpan');
62:         datetimeSpan.innerHTML = dateTime.toLocaleString();
63:       }
64:
65:       function convertToHtml(books) {

```

## PennyCss/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 = JSON.parse(this.response);
85:         let html = convertToHtml(books);
86:         let resultsTd = document.getElementById('resultsTd');
87:         resultsTd.innerHTML = html;
88:       }
89:
90:       function handleError(request) {
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:        getDateime();
121:        window.setInterval(getDateime, 1000);
122:        let authorInput = document.getElementById('authorInput');
123:        authorInput.addEventListener('input', debouncedGetResults);
124:      }
125:
126:      document.addEventListener('DOMContentLoaded', setup);
127:    </script>
128:  </body>
129: </html>

```

## PennyResponsive/static/penny.css (Page 1 of 2)

```

1: /*-----*/
2: /* penny.css */
3: /* Author: Bob Dondero */
4: /*-----*/
5:
6: /* Element rules */
7:
8: body {font-family:Arial, Helvetica, sans-serif; font-size:16px;}
9:
10: input[type="text"] {font-size:16px;}
11:
12: a {color:#aaaaff;}
13:
14: /*-----*/
15:
16: /* Class rules */
17:
18: .header {background-color:#295078; text-align:center; color:white;}
19:
20: .footer {background-color:#295078; text-align:center; color:white;}
21:
22: .grayborder {border-style:solid; border-color:gray; border-width:1px;}
23:
24: /*-----*/
25:
26: /* The following sections are derived from:
27:    https://www.w3schools.com/css/css_rwd_grid.asp */
28:
29: /*-----*/
30:
31: /* Make sure that the padding and border are included in the total
32:    width and height of all elements. */
33:
34: * {box-sizing: border-box;}
35:
36: /*-----*/
37:
38: /* Divide the window into 12 columns. An element with style "col-1"
39:    consumes 1 column, an element with style "col-2" consumes 2
40:    columns, and so forth. Each "col-" element should be placed within
41:    a "row" element. The "col-" elements within each row element
42:    should consume 12 columns. Each "col-" element floats to the left
43:    of its "row" element, and has a padding of 5 pixels. */
44:
45: .col-1 {width: 8.33%; float:left; padding:5px;}
46: .col-2 {width: 16.66%; float:left; padding:5px;}
47: .col-3 {width: 25%; float:left; padding:5px;}
48: .col-4 {width: 33.33%; float:left; padding:5px;}
49: .col-5 {width: 41.66%; float:left; padding:5px;}
50: .col-6 {width: 50%; float:left; padding:5px;}
51: .col-7 {width: 58.33%; float:left; padding:5px;}
52: .col-8 {width: 66.66%; float:left; padding:5px;}
53: .col-9 {width: 75%; float:left; padding:5px;}
54: .col-10 {width: 83.33%; float:left; padding:5px;}
55: .col-11 {width: 91.66%; float:left; padding:5px;}
56: .col-12 {width: 100%; float:left; padding:5px;}
57:
58: /*-----*/
59:
60: /* The "col-" elements inside a "row" element float to the left, and
61:    so normally would be "out of the flow of the page." Other
62:    elements would be placed as if the "col-" elements do not exist.
63:    Clear the flow to prevent that. */
64:
65: .row::after

```

## PennyResponsive/static/penny.css (Page 2 of 2)

```

66: {
67:     content: "";
68:     clear: both;
69:     display: table;
70: }
71:
72: /*-----*/
73:
74: /* On small windows (those whose maximum width is 600 pixels or less)
75:    suppress the display of "h2" elements. Also change the "col-"
76:    classes such that each "col-" element consumes all 12 columns
77:    within its "row" element. So multiple "col-" elements within a
78:    "row" element will be arranged vertically. */
79:
80: @media (max-width: 600px)
81: {
82:     h2 {display:none;}
83:
84:     /*
85:     .col-1 {width: 100%; float:left; padding:5px;}
86:     .col-2 {width: 100%; float:left; padding:5px;}
87:     .col-3 {width: 100%; float:left; padding:5px;}
88:     .col-4 {width: 100%; float:left; padding:5px;}
89:     .col-5 {width: 100%; float:left; padding:5px;}
90:     .col-6 {width: 100%; float:left; padding:5px;}
91:     .col-7 {width: 100%; float:left; padding:5px;}
92:     .col-8 {width: 100%; float:left; padding:5px;}
93:     .col-9 {width: 100%; float:left; padding:5px;}
94:     .col-10 {width: 100%; float:left; padding:5px;}
95:     .col-11 {width: 100%; float:left; padding:5px;}
96:     .col-12 {width: 100%; float:left; padding:5px;}
97:     */
98:
99:     /* Shortcut: */
100:    [class*="col-"] {width: 100%; float:left; padding:5px;}
101: }

```

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

```

1: <!DOCTYPE html>
2: <html>
3:
4:   <head>
5:     <title>Penny.com</title>
6:     <meta name="viewport"
7:       content="width=device-width, initial-scale=1">
8:     <link rel="stylesheet" href="static/penny.css">
9:   </head>
10:
11:   <body>
12:
13:     <div class="header">
14:       <h2>Penny.com</h2>
15:       <h3>Good <span id="ampmSpan"></span> and welcome to
16:         Penny.com</h3>
17:     </div>
18:
19:     <br>
20:
21:     <div class="row">
22:       <div class="col-3"><strong>Author name:</strong></div>
23:       <div class="col-9" grayborder>
24:         <input type="text" id="authorInput" autoFocus>
25:       </div>
26:     </div>
27:
28:     <br>
29:
30:     <div class="row">
31:       <div class="col-3"><strong>Books:</strong></div>
32:       <div class="col-9 grayborder" id="resultsDiv"></div>
33:     </div>
34:
35:     <br>
36:
37:     <div class="footer">
38:       Date and time: <span id="datetimeSpan"></span><br>
39:       Created by <a href="https://www.cs.princeton.edu/~rdondero">
40:         Bob Dondero</a>
41:     </div>
42:
43:     <script src=
44:       "https://cdn.jsdelivr.net/npm/mustache@4.2.0/mustache.min.js">
45:     </script>
46:
47:     <script>
48:       'use strict';
49:
50:       function getAmPm() {
51:         let dateTime = new Date();
52:         let hours = dateTime.getHours();
53:         let amPm = (hours < 12) ? 'morning' : 'afternoon';
54:         let ampMspan = document.getElementById('ampmSpan');
55:         ampMspan.innerHTML = amPm;
56:       }
57:
58:       function getDateime() {
59:         let dateTime = new Date();
60:         let datetimeSpan =
61:           document.getElementById('datetimeSpan');
62:         datetimeSpan.innerHTML = dateTime.toLocaleString();
63:       }
64:
65:       function convertToHtml(books) {

```

## PennyResponsive/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 = JSON.parse(this.response);
85:         let html = convertToHtml(books);
86:         let resultsDiv = document.getElementById('resultsDiv');
87:         resultsDiv.innerHTML = html;
88:       }
89:
90:       function handleError(request) {
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:        getDateime();
121:        window.setInterval(getDateime, 1000);
122:        let authorInput = document.getElementById('authorInput');
123:        authorInput.addEventListener('input', debouncedGetResults);
124:      }
125:
126:      document.addEventListener('DOMContentLoaded', setup);
127:    </script>
128:  </body>
129: </html>

```

## PennyBootstrap/index.html (Page 1 of 3)

```

1: <!DOCTYPE html>
2: <html>
3:   <head>
4:     <title>Penny.com</title>
5:
6:     <meta name="viewport"
7:       content="width=device-width, initial-scale=1">
8:
9:     <link rel="stylesheet" href=
10:       "https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstra
p.min.css">
11:
12:     <style>
13:       .header, .footer {background-color:#295078; color:white}
14:     </style>
15:
16:   </head>
17:
18:   <body>
19:     <div class="container-fluid, header">
20:       <center>
21:         <h2 class="d-none d-sm-block">Penny.com</h2>
22:         <h3>Good <span id="ampmSpan"></span> and welcome to
23:         Penny.com</h3>
24:       </center>
25:     </div>
26:
27:     <br>
28:
29:     <div class="container-fluid">
30:       <div class="row">
31:         <div class="col-sm-3"><h5>Author name:</h5></div>
32:         <div class="col-sm-9">
33:           <input type="text" class="form-control"
34:             id="authorInput" autoFocus>
35:         </div>
36:       </div>
37:       <br>
38:       <div class="row">
39:         <div class="col-sm-3"><h5>Books:</h5></div>
40:         <div class="col-sm-9">
41:           <div class="border" id="resultsDiv"></div>
42:         </div>
43:       </div>
44:     </div>
45:
46:     <br>
47:
48:     <div class="container-fluid, footer">
49:       <center>
50:         Date and time: <span id="datetimeSpan"></span><br>
51:         Created by <a
52:           href="http://www.cs.princeton.edu/~rdondero">
53:           Bob Dondero</a>
54:       </center>
55:     </div>
56:
57:     <script src=
58:       "https://cdn.jsdelivr.net/npm/mustache@4.2.0/mustache.min.js">
59:     </script>
60:
61:     <script>
62:
63:       'use strict';
64:

```

## PennyBootstrap/index.html (Page 2 of 3)

```

65:     function getAmPm() {
66:       let dateTime = new Date();
67:       let hours = dateTime.getHours();
68:       let amPm = (hours < 12) ? 'morning' : 'afternoon';
69:       let ampmSpan = document.getElementById('ampmSpan');
70:       ampmSpan.innerHTML = amPm;
71:     }
72:
73:     function getDateTime() {
74:       let dateTime = new Date();
75:       let datetimeSpan =
76:         document.getElementById('datetimeSpan');
77:       datetimeSpan.innerHTML = dateTime.toLocaleString();
78:     }
79:
80:     function convertToHtml(books) {
81:       let template = `
82:         {{#books}}
83:           {{isbn}}:
84:           <strong>{{author}}</strong>:
85:           {{title}}
86:           <br>
87:         {{/books}}
88:       `;
89:       let map = {books: books};
90:       let html = Mustache.render(template, map);
91:       return html;
92:     }
93:
94:     function handleResponse() {
95:       if (this.status !== 200) {
96:         alert('Error: Failed to fetch data from server');
97:         return;
98:       }
99:       let books = JSON.parse(this.response);
100:      let html = convertToHtml(books);
101:      let resultsDiv = document.getElementById('resultsDiv');
102:      resultsDiv.innerHTML = html;
103:    }
104:
105:    function handleError(request) {
106:      alert('Error: Failed to fetch data from server');
107:    }
108:
109:    let request = null;
110:
111:    function getResults() {
112:      let authorInput = document.getElementById('authorInput');
113:      let author = authorInput.value;
114:      let encodedAuthor = encodeURIComponent(author);
115:      let url = '/searchresults?author=' + encodedAuthor;
116:      if (request !== null)
117:        request.abort();
118:      request = new XMLHttpRequest();
119:      request.onload = handleResponse;
120:      request.onerror = handleError;
121:      request.open('GET', url);
122:      request.send();
123:    }
124:
125:    let timer = null;
126:
127:    function debouncedGetResults() {
128:      clearTimeout(timer);
129:      timer = setTimeout(getResults, 500);

```

## PennyBootstrap/index.html (Page 3 of 3)

```
130:     }
131:
132:     function setup() {
133:         getAmPm();
134:         window.setInterval(getAmPm, 1000);
135:         getDateTIme();
136:         window.setInterval(getDateTIme, 1000);
137:         let authorInput = document.getElementById('authorInput');
138:         authorInput.addEventListener('input', debouncedGetResults);
139:     }
140:
141:     document.addEventListener('DOMContentLoaded', setup);
142:
143:     </script>
144: </body>
145: </html>
```

## blank (Page 1 of 1)

1: This page is intentionally blank.



## PennyModal/indexRaw.html (Page 1 of 3)

```

1: <!DOCTYPE html>
2: <html>
3:   <head>
4:     <title>Penny.com</title>
5:
6:     <meta name="viewport"
7:       content="width=device-width, initial-scale=1">
8:
9:     <script src=
10:      "https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap
.bundle.min.js">
11:     </script>
12:
13:     <link rel="stylesheet" href=
14:      "https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstra
p.min.css">
15:
16:     <style>
17:       .header, .footer {background-color:#295078; color:white}
18:     </style>
19:
20:   </head>
21:
22:   <body>
23:
24:     <div class="modal" id="booksModal">
25:       <div class="modal-dialog">
26:         <div class="modal-content">
27:           <div class="modal-header">
28:             <h4 class="modal-title">Books</h4>
29:             <button class="btn-close" data-bs-dismiss="modal">
30:               </button>
31:           </div>
32:           <div class="modal-body" id="booksModalBody">
33:             Modal body; to be replaced.
34:           </div>
35:           <div class="modal-footer">
36:             <button class="btn btn-danger"
37:               data-bs-dismiss="modal">
38:               Close
39:             </button>
40:           </div>
41:         </div>
42:       </div>
43:     </div>
44:
45:     <div class="container-fluid, header">
46:       <center>
47:         <h2 class="d-none d-sm-block">Penny.com</h2>
48:         <h3>Good <span id="ampmSpan"></span> and welcome to
49:         Penny.com</h3>
50:       </center>
51:     </div>
52:
53:     <br>
54:
55:     <div class="container-fluid">
56:       <div class="row">
57:         <div class="col-sm-2"><h5>Author name:</h5></div>
58:         <div class="col-sm-8">
59:           <input type="text" class="form-control"
60:             id="authorInput" autoFocus>
61:         </div>
62:         <div class="col-sm-2">
63:           <button id="submitbutton">Submit</button>

```

## PennyModal/indexRaw.html (Page 2 of 3)

```

64:           </div>
65:         </div>
66:       </div>
67:
68:       <br>
69:
70:       <div class="container-fluid, footer">
71:         <center>
72:           Date and time: <span id="datetimeSpan"></span><br>
73:           Created by <a
74:             href="http://www.cs.princeton.edu/~rdondero">
75:             Bob Dondero</a>
76:         </center>
77:       </div>
78:
79:       <script src=
80:        "https://cdn.jsdelivr.net/npm/mustache@4.2.0/mustache.min.js">
81:       </script>
82:
83:       <script>
84:
85:         'use strict';
86:
87:         function getAmPm() {
88:           let dateTime = new Date();
89:           let hours = dateTime.getHours();
90:           let amPm = (hours < 12) ? 'morning' : 'afternoon';
91:           let ampmSpan = document.getElementById('ampmSpan');
92:           ampmSpan.innerHTML = amPm;
93:         }
94:
95:         function getDateTime() {
96:           let dateTime = new Date();
97:           let datetimeSpan =
98:             document.getElementById('datetimeSpan');
99:           datetimeSpan.innerHTML = dateTime.toLocaleString();
100:         }
101:
102:         function convertToHtml(books) {
103:           let template = `
104:             {{#books}}
105:               {{isbn}}:
106:               <strong>{{author}}</strong>:
107:               {{title}}
108:               <br>
109:             {{/books}}
110:           `;
111:           let map = {books: books};
112:           let html = Mustache.render(template, map);
113:           return html;
114:         }
115:
116:         function handleResponse() {
117:           if (this.status !== 200) {
118:             alert('Error: Failed to fetch data from server');
119:             return;
120:           }
121:           let books = JSON.parse(this.response);
122:           let html = convertToHtml(books);
123:           let booksModalBodyNode =
124:             document.getElementById('booksModalBody');
125:           booksModalBodyNode.innerHTML = html;
126:           let booksModalNode =
127:             document.getElementById('booksModal');
128:           let booksModal = new bootstrap.Modal(booksModalNode);

```

## PennyModal/indexRaw.html (Page 3 of 3)

```
129:     booksModal.show();
130:   }
131:
132:   function handleError(request) {
133:     alert('Error: Failed to fetch data from server');
134:   }
135:
136:   let request = null;
137:
138:   function getResults() {
139:     let authorInput = document.getElementById('authorInput');
140:     let author = authorInput.value;
141:     let encodedAuthor = encodeURIComponent(author);
142:     let url = '/searchresults?author=' + encodedAuthor;
143:     if (request !== null)
144:       request.abort();
145:     request = new XMLHttpRequest();
146:     request.onload = handleResponse;
147:     request.onerror = handleError;
148:     request.open('GET', url);
149:     request.send();
150:   }
151:
152:   function setup() {
153:     getAmPm();
154:     window.setInterval(getAmPm, 1000);
155:     getDateTIme();
156:     window.setInterval(getDateTIme, 1000);
157:     let submitButton = document.getElementById('submitButton');
158:     submitButton.addEventListener('click', getResults);
159:   }
160:
161:   document.addEventListener('DOMContentLoaded', setup);
162:
163: </script>
164: </body>
165: </html>
```

## blank (Page 1 of 1)

1: This page is intentionally blank.

## PennyModal/indexjQuery.html (Page 1 of 3)

```

1: <!DOCTYPE html>
2: <html>
3:   <head>
4:     <title>Penny.com</title>
5:
6:     <meta name="viewport"
7:       content="width=device-width, initial-scale=1">
8:
9:     <script src=
10:      "https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap
.bundle.min.js">
11:   </script>
12:
13:   <link rel="stylesheet" href=
14:     "https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstra
p.min.css">
15:
16:   <style>
17:     .header, .footer {background-color:#295078; color:white}
18:   </style>
19:
20: </head>
21:
22: <body>
23:
24:   <div class="modal" id="booksModal">
25:     <div class="modal-dialog">
26:       <div class="modal-content">
27:         <div class="modal-header">
28:           <h4 class="modal-title">Books</h4>
29:           <button class="btn-close" data-bs-dismiss="modal">
30:             </button>
31:         </div>
32:         <div class="modal-body" id="booksModalBody">
33:           Modal body; to be replaced.
34:         </div>
35:         <div class="modal-footer">
36:           <button class="btn btn-danger"
37:             data-bs-dismiss="modal">
38:             Close
39:           </button>
40:         </div>
41:       </div>
42:     </div>
43:   </div>
44:
45:   <div class="container-fluid, header">
46:     <center>
47:       <h2 class="d-none d-sm-block">Penny.com</h2>
48:       <h3>Good <span id="ampmSpan"></span> and welcome to
49:       Penny.com</h3>
50:     </center>
51:   </div>
52:
53:   <br>
54:
55:   <div class="container-fluid">
56:     <div class="row">
57:       <div class="col-sm-2"><h5>Author name:</h5></div>
58:       <div class="col-sm-8">
59:         <input type="text" class="form-control"
60:           id="authorInput" autoFocus>
61:       </div>
62:       <div class="col-sm-2">
63:         <button id="submitbutton">Submit</button>

```

## PennyModal/indexjQuery.html (Page 2 of 3)

```

64:         </div>
65:       </div>
66:     </div>
67:
68:     <br>
69:
70:     <div class="container-fluid, footer">
71:       <center>
72:         Date and time: <span id="datetimeSpan"></span><br>
73:         Created by <a
74:           href="http://www.cs.princeton.edu/~rdondero">
75:           Bob Dondero</a>
76:       </center>
77:     </div>
78:
79:     <script src=
80:       "https://cdn.jsdelivr.net/npm/jquery@3.7.1/dist/jquery.min.js">
81:   </script>
82:
83:   <script src=
84:     "https://cdn.jsdelivr.net/npm/mustache@4.2.0/mustache.min.js">
85:   </script>
86:
87:   <script>
88:
89:     'use strict';
90:
91:     function getAmPm() {
92:       let dateTime = new Date();
93:       let hours = dateTime.getHours();
94:       let amPm = 'morning';
95:       if (hours >= 12)
96:         amPm = 'afternoon';
97:       $('#ampmSpan').html(amPm);
98:     }
99:
100:    function getDateTime() {
101:      let dateTime = new Date();
102:      $('#datetimeSpan').html(dateTime.toLocaleDateString());
103:    }
104:
105:    function convertToHtml(books) {
106:      let template = `
107:        {{#books}}
108:          {{isbn}}:
109:          <strong>{{author}}</strong>:
110:          {{title}}
111:          <br>
112:        {{/books}}
113:      `;
114:      let map = {books: books};
115:      let html = Mustache.render(template, map);
116:      return html;
117:    }
118:
119:    function handleResponse(books) {
120:      let html = convertToHtml(books);
121:      $('#booksModalBody').html(html);
122:      $('#booksModal').modal('show');
123:    }
124:
125:    function handleError(request) {
126:      if (request.statusText !== 'abort')
127:        alert('Error: Failed to fetch data from server');
128:    }

```

## PennyModal/indexjQuery.html (Page 3 of 3)

```
129:
130:     let request = null;
131:
132:     function getResults() {
133:         let author = $('#authorInput').val();
134:         let encodedAuthor = encodeURIComponent(author);
135:         let url = '/searchresults?author=' + encodedAuthor;
136:         if (request != null)
137:             request.abort();
138:         let requestData = {
139:             type: 'GET',
140:             url: url,
141:             success: handleResponse,
142:             error: handleError
143:         };
144:         request = $.ajax(requestData);
145:     }
146:
147:     function setup() {
148:         getAmPm();
149:         window.setInterval(getAmPm, 1000);
150:         getDateTime();
151:         window.setInterval(getDateTime, 1000);
152:         $('#submitbutton').on('click', getResults);
153:     }
154:
155:     $('document').ready(setup);
156:
157: </script>
158: </body>
159: </html>
```

## blank (Page 1 of 1)

1: This page is intentionally blank.

## PennyModal/indexReact1.html (Page 1 of 4)

```

1: <!DOCTYPE html>
2: <html>
3:   <head>
4:     <meta name="viewport"
5:       content="width=device-width, initial-scale=1">
6:
7:     <!-- For development: -->
8:     <!--
9:     <script src=
10:       "https://cdn.jsdelivr.net/npm/react@18.3.1/umd/react.developmen
t.js">
11:     </script>
12:     <script src=
13:       "https://cdn.jsdelivr.net/npm/react-dom@18.3.1/umd/react-dom.de
velopment.js">
14:     </script>
15:     -->
16:
17:     <!-- For production: -->
18:     <script src=
19:       "https://cdn.jsdelivr.net/npm/react@18.3.1/umd/react.production
.min.js">
20:     </script>
21:     <script src=
22:       "https://cdn.jsdelivr.net/npm/react-dom@18.3.1/umd/react-dom.pr
oduction.min.js">
23:     </script>
24:
25:     <script src=
26:       "https://cdn.jsdelivr.net/npm/babel-standalone@6.26.0/babel.min
.js">
27:     </script>
28:
29:     <script src=
30:       "https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap
.bundle.min.js">
31:     </script>
32:
33:     <link rel="stylesheet" href=
34:       "https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstra
p.min.css">
35:
36:     <style>
37:       .header, .footer {background-color:#295078; color:white}
38:     </style>
39:
40:     <title>Penny.com</title>
41:   </head>
42:
43:   <body>
44:
45:     <div id="root"></div>
46:
47:     <script type="text/babel">
48:       'use strict';
49:
50:       //-----
51:
52:       function PennyHeader() {
53:         const [datetime, setDatetime] = React.useState(new Date());
54:
55:         function updateHeader() {
56:           window.setInterval(
57:             () => {setDatetime(new Date());},
58:             1000

```

## PennyModal/indexReact1.html (Page 2 of 4)

```

59:         );
60:       }
61:
62:       React.useEffect(updateHeader, []);
63:
64:       let hours = datetime.getHours();
65:       let amp = (hours < 12) ? 'morning' : 'afternoon';
66:       return (
67:         <div className="container-fluid header">
68:           <center>
69:             <h2 className="d-none d-sm-block">Penny.com</h2>
70:             <h3>Good {amp} and welcome to Penny.com</h3>
71:           </center>
72:         </div>
73:       );
74:     }
75:
76:     //-----
77:
78:     function BooksModalBody(props) {
79:       let books = props.b;
80:       return (
81:         <div>
82:           {books.map((book) => (
83:             <div key={book.isbn}>
84:               {book.isbn}&nbsp;&nbsp;&nbsp;
85:               <strong>{book.author}</strong>&nbsp;&nbsp;&nbsp;
86:               ({book.title})<br />
87:             </div>
88:           ))}
89:         </div>
90:       );
91:     }
92:
93:     //-----
94:
95:     function PennySearch() {
96:       const [author, setAuthor] = React.useState([0, null]);
97:       const [books, setBooks] = React.useState(null);
98:
99:       const inputRef = React.useRef();
100:      const modalRef = React.useRef();
101:
102:      function fetchBooks() {
103:        function handleResponse() {
104:          if (this.status !== 200) {
105:            alert('Error: Failed to fetch data from server');
106:            return;
107:          }
108:          let books = JSON.parse(this.response);
109:          console.log("setting books:");
110:          console.log(books);
111:          setBooks(books);
112:        }
113:
114:        function handleError(request) {
115:          if (request.statusText !== 'abort')
116:            alert('Error: Failed to fetch data from server');
117:        }
118:
119:        if (author[1] === null)
120:          return;
121:
122:        let encodedAuthor = encodeURIComponent(author[1]);
123:        let url = '/searchresults?author=' + encodedAuthor;

```

## PennyModal/indexReact1.html (Page 3 of 4)

```

124:         let request = new XMLHttpRequest();
125:         request.onload = handleResponse;
126:         request.onerror = handleError;
127:         request.open('GET', url);
128:         request.send();
129:         return () => {request.abort();}
130:     }
131:
132:     React.useEffect(fetchBooks, [author]);
133:
134:     function showModal() {
135:         if (books === null)
136:             return;
137:         let modalNode = modalRef.current;
138:         let booksModal = new bootstrap.Modal(modalNode);
139:         booksModal.show()
140:     }
141:
142:     React.useEffect(showModal, [books]);
143:
144:     return (
145:         <div className="container-fluid">
146:             <div className="row">
147:                 <div className="col-sm-2">
148:                     <h5>Author name:</h5>
149:                 </div>
150:                 <div className="col-sm-8">
151:                     <input
152:                         type="text"
153:                         className="form-control"
154:                         ref={inputRef}
155:                         autoFocus
156:                     />
157:                 </div>
158:                 <div className="col-sm-2">
159:                     <button onClick={ (event) => {
160:                         setAuthor([Math.random(),
161:                             inputRef.current.value]);
162:                     }} >
163:                         Submit
164:                     </button>
165:                 </div>
166:             </div>
167:
168:             {books &&
169:                 <div className="modal" ref={modalRef} >
170:                     <div className="modal-dialog">
171:                         <div className="modal-content">
172:                             <div className="modal-header">
173:                                 <h4 className="modal-title">
174:                                     Books
175:                                 </h4>
176:                                 <button type="button"
177:                                     className="btn-close"
178:                                     data-bs-dismiss="modal">
179:                                 </button>
180:                             </div>
181:                             <div className="modal-body">
182:                                 <BooksModalBody b={books} />
183:                             </div>
184:                             <div className="modal-footer">
185:                                 <button type="button"
186:                                     className="btn btn-danger"
187:                                     data-bs-dismiss="modal">
188:                                     Close

```

## PennyModal/indexReact1.html (Page 4 of 4)

```

189:                                     </button>
190:                                 </div>
191:                             </div>
192:                         </div>
193:                     </div>
194:                 </div>
195:             </div>
196:         );
197:     }
198:
199:     //-----
200:
201:     function PennyFooter() {
202:         const [datetime, setDatetime] = React.useState(new Date());
203:
204:         function updateFooter() {
205:             window.setInterval(
206:                 () => {setDatetime(new Date());},
207:                 1000
208:             );
209:         }
210:
211:         React.useEffect(updateFooter, []);
212:
213:         return (
214:             <div className="container-fluid, footer">
215:                 <center>
216:                     Date and time: {datetime.toLocaleString()}
217:                     <br />
218:                     Created by&nbsp;
219:                     <a href="https://www.cs.princeton.edu/~rdontero">
220:                         Bob Dontero</a>
221:                 </center>
222:             </div>
223:         );
224:     }
225:
226:     //-----
227:
228:     function App() {
229:         return (
230:             <div>
231:                 <PennyHeader />
232:                 <br />
233:                 <PennySearch />
234:                 <br />
235:                 <PennyFooter />
236:             </div>
237:         );
238:     }
239:
240:     //-----
241:
242:     let domRoot = document.getElementById('root');
243:     let reactRoot = ReactDOM.createRoot(domRoot);
244:     reactRoot.render(
245:         <React.StrictMode>
246:             <App />
247:         </React.StrictMode>
248:     );
249: </script>
250: </body>
251: </html>
252:

```

## PennyModal/indexReact2.html (Page 1 of 4)

```

1: <!DOCTYPE html>
2: <html>
3:   <head>
4:     <meta name="viewport"
5:       content="width=device-width, initial-scale=1">
6:
7:     <!-- For development: -->
8:     <!--
9:     <script src=
10:       "https://cdn.jsdelivr.net/npm/react@18.3.1/umd/react.developmen
t.js">
11:     </script>
12:     <script src=
13:       "https://cdn.jsdelivr.net/npm/react-dom@18.3.1/umd/react-dom.de
velopment.js">
14:     </script>
15:     -->
16:
17:     <!-- For production: -->
18:     <script src=
19:       "https://cdn.jsdelivr.net/npm/react@18.3.1/umd/react.production
.min.js">
20:     </script>
21:     <script src=
22:       "https://cdn.jsdelivr.net/npm/react-dom@18.3.1/umd/react-dom.pr
oduction.min.js">
23:     </script>
24:
25:     <script src=
26:       "https://cdn.jsdelivr.net/npm/babel-standalone@6.26.0/babel.min
.js">
27:     </script>
28:
29:     <script src=
30:       "https://cdn.jsdelivr.net/npm/react-bootstrap@2.10.0/dist/react
-bootstrap.min.js">
31:     </script>
32:
33:     <link rel="stylesheet" href=
34:       "https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstra
p.min.css">
35:
36:     <style>
37:       .header, .footer {background-color:#295078; color:white}
38:     </style>
39:
40:     <title>Penny.com</title>
41:   </head>
42:
43:   <body>
44:
45:     <div id="root"></div>
46:
47:     <script type="text/babel">
48:
49:       'use strict';
50:
51:       //-----
52:
53:       function PennyHeader() {
54:         const [datetime, setDatetime] = React.useState(new Date());
55:
56:         function updateHeader() {
57:           window.setInterval(
58:             () => {setDatetime(new Date());},

```

## PennyModal/indexReact2.html (Page 2 of 4)

```

59:             1000
60:           );
61:         }
62:         React.useEffect(updateHeader, []);
63:
64:         let hours = datetime.getHours();
65:         let amp = (hours < 12) ? 'morning' : 'afternoon';
66:         return (
67:           <ReactBootstrap.Container fluid className="header">
68:             <center>
69:               <h2>Penny.com</h2>
70:               <h3>Good {amp} and welcome to Penny.com</h3>
71:             </center>
72:           </ReactBootstrap.Container>
73:         );
74:       }
75:
76:       //-----
77:
78:       function BooksModal(props) {
79:         let books = props.b;
80:         let show = props.s;
81:         let showCallback = props.scb;
82:
83:         function handleClose() {
84:           showCallback(false);
85:         }
86:
87:         return (
88:           <ReactBootstrap.Modal show={show}
89:             onHide={handleClose}>
90:             <ReactBootstrap.Modal.Header closeButton>
91:               <ReactBootstrap.Modal.Title>
92:                 Books
93:               </ReactBootstrap.Modal.Title>
94:             </ReactBootstrap.Modal.Header>
95:             <ReactBootstrap.Modal.Body>
96:               {books.map((book) => (
97:                 <div key={book.isbn}>
98:                   {book.isbn}:&nbsp;
99:                   <strong>{book.author}</strong>:&nbsp;
100:                   {book.title}<br />
101:                 </div>
102:               ))}
103:             </ReactBootstrap.Modal.Body>
104:             <ReactBootstrap.Modal.Footer>
105:               <ReactBootstrap.Button variant="danger"
106:                 onClick={handleClose}>
107:                 Close
108:               </ReactBootstrap.Button>
109:             </ReactBootstrap.Modal.Footer>
110:           </ReactBootstrap.Modal>
111:         );
112:       }
113:
114:       //-----
115:
116:       function PennySearch() {
117:         const [author, setAuthor] = React.useState([0, null]);
118:         const [books, setBooks] = React.useState(null);
119:         const [show, setShow] = React.useState(false);
120:
121:         const inputRef = React.useRef();
122:
123:         function fetchBooks() {

```

## PennyModal/indexReact2.html (Page 3 of 4)

```

124:     function handleResponse() {
125:         if (this.status !== 200) {
126:             alert('Error: Failed to fetch data from server');
127:             return;
128:         }
129:         let books = JSON.parse(this.response);
130:         setBooks(books);
131:     }
132:
133:     function handleError(request) {
134:         if (request.statusText !== 'abort')
135:             alert('Error: Failed to fetch data from server');
136:     }
137:
138:     if (author[1] === null)
139:         return;
140:
141:     let encodedAuthor = encodeURIComponent(author[1]);
142:     let url = '/searchresults?author=' + encodedAuthor;
143:     let request = new XMLHttpRequest();
144:     request.onload = handleResponse;
145:     request.onerror = handleError;
146:     request.open('GET', url);
147:     request.send();
148:     return () => {request.abort();}
149: }
150: React.useEffect(fetchBooks, [author]);
151:
152: function handleShow() {
153:     setShow(true);
154: }
155: React.useEffect(handleShow, [books]);
156:
157: return (
158:     <div>
159:         {books &&
160:         <BooksModal s={show} scb={setShow} b={books} />
161:         }
162:         <ReactBootstrap.Container fluid>
163:             <ReactBootstrap.Row>
164:                 <ReactBootstrap.Col sm={2}>
165:                     <h5>Author name:</h5>
166:                 </ReactBootstrap.Col>
167:                 <ReactBootstrap.Col sm={8}>
168:                     <ReactBootstrap.Form.Control
169:                         type="text" ref={inputRef} autoFocus
170:                     />
171:                 </ReactBootstrap.Col>
172:                 <ReactBootstrap.Col sm={2}>
173:                     <button onClick={ (event) => {
174:                         setAuthor ([Math.random(),
175:                             inputRef.current.value]);
176:                     }} >
177:                         Submit
178:                     </button>
179:                 </ReactBootstrap.Col>
180:             </ReactBootstrap.Row>
181:         </ReactBootstrap.Container>
182:     </div>
183: );
184: }
185:
186: //-----
187:
188: function PennyFooter() {

```

## PennyModal/indexReact2.html (Page 4 of 4)

```

189:     const [datetime, setDatetime] = React.useState(new Date());
190:
191:     function updateFooter() {
192:         window.setInterval(
193:             () => {setDatetime(new Date());},
194:             1000
195:         );
196:     }
197:     React.useEffect(updateFooter, []);
198:
199:     return (
200:         <ReactBootstrap.Container fluid className="footer">
201:             <center>
202:                 Date and time: {datetime.toLocaleString()}
203:                 <br />
204:                 Created by&nbsp;
205:                 <a href="https://www.cs.princeton.edu/~rdondero">
206:                     Bob Dondero</a>
207:             </center>
208:         </ReactBootstrap.Container>
209:     );
210: }
211:
212: //-----
213:
214: function App() {
215:     return (
216:         <div>
217:             <PennyHeader />
218:             <br />
219:             <PennySearch />
220:             <br />
221:             <PennyFooter />
222:         </div>
223:     );
224: }
225:
226: //-----
227:
228: let domRoot = document.getElementById('root');
229: let reactRoot = ReactDOM.createRoot(domRoot);
230: reactRoot.render(
231:     <React.StrictMode>
232:         <App />
233:     </React.StrictMode>
234: );
235:
236: </script>
237: </body>
238: </html>
239:

```