

## 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/~rdonero">
25:    Bob Donero</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:
44:      function getDateTime() {
45:        let dateTime = new Date();
46:        let datetimeSpan =
47:          document.getElementById('datetimeSpan');
48:        datetimeSpan.innerHTML = dateTime.toLocaleString();
49:      }
50:
51:      function convertToHtml(books) {
52:        let template = `
53:          {{#books}}
54:            {{isbn}}:
55:            <strong>{{author}}</strong>:
56:            {{title}}
57:            <br>
58:          {{/books}}
59:        `;
60:        let map = {books: books};
61:        let html = Mustache.render(template, map);
62:        return html;
63:      }
64:
65:      function handleResponse() {

```

## 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() {
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 setupHeader() {
104:        getAmPm();
105:        let apInterval = window.setInterval(getAmPm, 1000);
106:        window.addEventListener('beforeunload',
107:          function(e) {window.clearInterval(apInterval);})
108:      }
109:
110:      function setupFooter() {
111:        getDateTime();
112:        let dtInterval = window.setInterval(getDateTime, 1000);
113:        window.addEventListener('beforeunload',
114:          function(e) {window.clearInterval(dtInterval);})
115:      }
116:
117:      function setup() {
118:        setupHeader();
119:        setupFooter();
120:        let authorInput = document.getElementById('authorInput');
121:        authorInput.addEventListener('input', debouncedGetResults);
122:      }
123:
124:      document.addEventListener('DOMContentLoaded', setup);
125:
126:    </script>
127:  </body>
128: </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 3)

```

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/~rdonero">
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 getDateTime() {
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 3)

```

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

```

**PennyCss/index.html (Page 3 of 3)**

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

**blank (Page 1 of 1)**

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

## 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 3)

```

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/~rdonero">
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 getDateTime() {
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 3)

```

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

```

**PennyResponsive/index.html (Page 3 of 3)**

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

**blank (Page 1 of 1)**

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



## 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:       'use strict';
63:
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 ampSpan = document.getElementById('ampmSpan');
70:       ampSpan.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() {
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 setupHeader() {
133:         getAmPm();
134:         let apInterval = window.setInterval(getAmPm, 1000);
135:         window.addEventListener('beforeunload',
136:             function(e) {window.clearInterval(apInterval);})
137:     }
138:
139:     function setupFooter() {
140:         getDateTime();
141:         let dtInterval = window.setInterval(getDateTime, 1000);
142:         window.addEventListener('beforeunload',
143:             function(e) {window.clearInterval(dtInterval);})
144:     }
145:
146:     function setup() {
147:         setupHeader();
148:         setupFooter();
149:         let authorInput = document.getElementById('authorInput');
150:         authorInput.addEventListener('input', debouncedGetResults);
151:     }
152:
153:     document.addEventListener('DOMContentLoaded', setup);
154:
155:     </script>
156: </body>
157: </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() {
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 setupHeader() {
153:         getAmPm();
154:         let apInterval = window.setInterval(getAmPm, 1000);
155:         window.addEventListener('beforeunload',
156:             function(e) {window.clearInterval(apInterval);})
157:     }
158:
159:     function setupFooter() {
160:         getDateTime();
161:         let dtInterval = window.setInterval(getDateTime, 1000);
162:         window.addEventListener('beforeunload',
163:             function(e) {window.clearInterval(dtInterval);})
164:     }
165:
166:     function setup() {
167:         setupHeader();
168:         setupFooter();
169:         let submitButton = document.getElementById('submitbutton');
170:         submitButton.addEventListener('click', getResults);
171:     }
172:
173:     document.addEventListener('DOMContentLoaded', setup);
174:
175: </script>
176: </body>
177: </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.toLocaleString());
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 setupHeader() {
148:         getAmPm();
149:         let apInterval = window.setInterval(getAmPm, 1000);
150:         $(window).on('beforeunload',
151:             function(e) {window.clearInterval(apInterval);})
152:     }
153:
154:     function setupFooter() {
155:         getDateTime();
156:         let dtInterval = window.setInterval(getDateTime, 1000);
157:         $(window).on('beforeunload',
158:             function(e) {window.clearInterval(dtInterval);})
159:     }
160:
161:     function setup() {
162:         setupHeader();
163:         setupFooter();
164:         $('#submitbutton').on('click', getResults);
165:     }
166:
167:     $('document').ready(setup);
168:
169: </script>
170: </body>
171: </html>
```

## blank (Page 1 of 1)

1: This page is intentionally blank.

## PennyModal/indexReact.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:     <div id="root"></div>
45:
46:     <script type="text/babel">
47:
48:       'use strict';
49:
50:       //-----
51:
52:       function PennyHeader() {
53:         const [datetime, setDatetime] = React.useState(new Date());
54:
55:         function updateHeader() {
56:           let apInterval = window.setInterval(
57:             () => {setDatetime(new Date());},
58:

```

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

```

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

```

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

```

124:
125:     function fetchBooks(author) {
126:         function handleResponse() {
127:             if (this.status !== 200) {
128:                 alert('Error: Failed to fetch data from server');
129:                 return;
130:             }
131:             let books = JSON.parse(this.response);
132:             setBooks(books);
133:             setShow(true);
134:         }
135:
136:         function handleError() {
137:             alert('Error: Failed to fetch data from server');
138:         }
139:
140:         let encodedAuthor = encodeURIComponent(author);
141:         let url = '/searchresults?author=' + encodedAuthor;
142:         let request = new XMLHttpRequest();
143:         request.onload = handleResponse;
144:         request.onerror = handleError;
145:         request.open('GET', url);
146:         request.send();
147:         return () => {request.abort();}
148:     }
149:
150:     return (
151:         <div>
152:             <BooksModal s={show} scb={setShow} b={books} />
153:             <ReactBootstrap.Container fluid>
154:                 <ReactBootstrap.Row>
155:                     <ReactBootstrap.Col sm={2}>
156:                         <h5>Author name:</h5>
157:                     </ReactBootstrap.Col>
158:                     <ReactBootstrap.Col sm={8}>
159:                         <ReactBootstrap.Form.Control
160:                             type="text" ref={inputRef} autoFocus
161:                         />
162:                     </ReactBootstrap.Col>
163:                     <ReactBootstrap.Col sm={2}>
164:                         <button onClick={ (event) => {
165:                             fetchBooks(inputRef.current.value);
166:                         }} >
167:                             Submit
168:                         </button>
169:                     </ReactBootstrap.Col>
170:                 </ReactBootstrap.Row>
171:             </ReactBootstrap.Container>
172:         </div>
173:     );
174: }
175:
176: //-----
177:
178: function PennyFooter() {
179:     const [datetime, setDatetime] = React.useState(new Date());
180:
181:     function updateFooter() {
182:         let dtInterval = window.setInterval(
183:             () => {setDatetime(new Date());},
184:             1000
185:         );
186:         return () => {window.clearInterval(dtInterval);};
187:     }
188:     React.useEffect(updateFooter, []);

```

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

```

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

```