

Client-Side Web Programming: JavaScript (Part 2)

Copyright © 2024 by
Robert M. Dondero, Ph.D.
Princeton University

Objectives

- We will cover:
 - Baseline example
 - JavaScript client-side web programming
 - AJAX

Agenda

- **Baseline example**
- JavaScript client-side web programming
- AJAX
- AJAX via XMLHttpRequest
- AJAX via XMLHttpRequest enhancements
- AJAX wrap-up

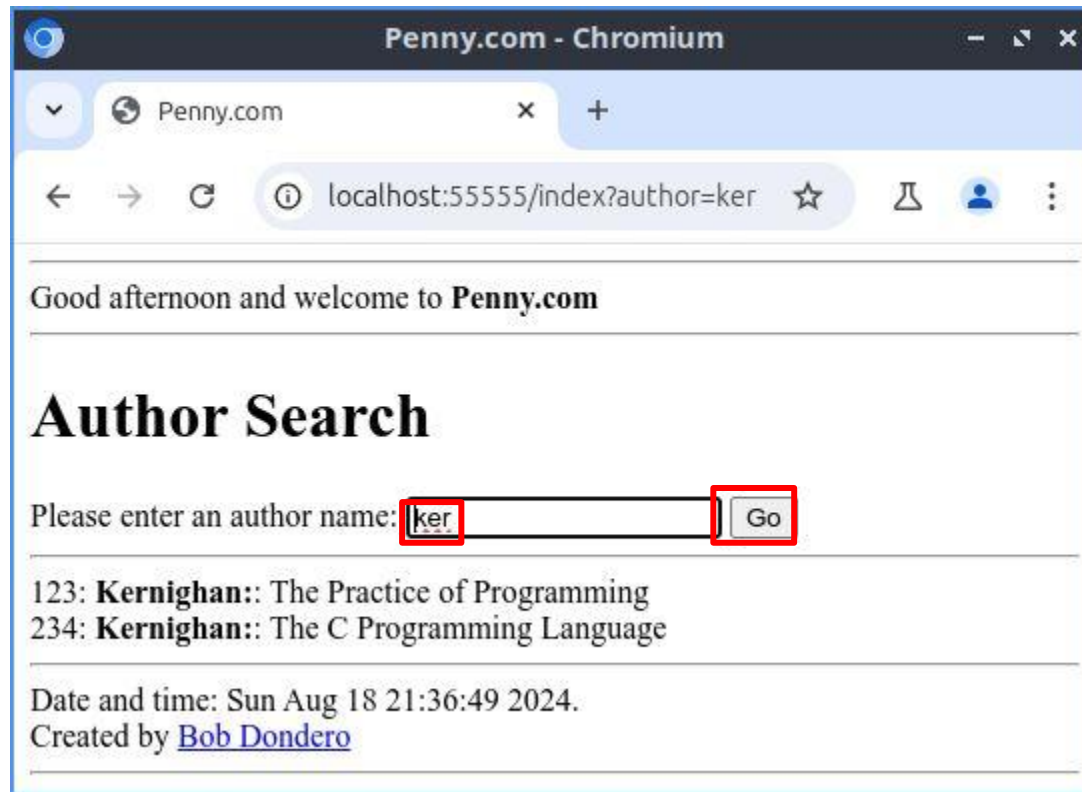
Baseline Example

- See **PennyOnePage** app



Baseline Example

- See PennyOnePage app (cont.)



Baseline Example

- See **PennyOnePage** app (cont.)
 - runserver.py
 - penny.sql, penny.sqlite
 - database.py
 - **penny.py**
 - **index.html**

PennyOnePage/penny.py (Page 1 of 1)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # penny.py
5: # Author: Bob Dondero
6: #-----
7:
8: import time
9: import flask
10: import database
11:
12: #-----
13:
14: app = flask.Flask(__name__, template_folder='.')
15:
16: #-----
17:
18: def get_ampm():
19:     if time.strftime('%p') == "AM":
20:         return 'morning'
21:     return 'afternoon'
22:
23: def get_current_time():
24:     return time.asctime(time.localtime())
25:
26: #-----
27:
28: @app.route('/', methods=['GET'])
29: @app.route('/index', methods=['GET'])
30: def index():
31:
32:     author = flask.request.args.get('author')
33:     if author is None:
34:         author = ''
35:     author = author.strip()
36:
37:     if author == '':
38:         books = []
39:     else:
40:         books = database.get_books(author) # Exception handling omitted
41:
42:     html = flask.render_template('index.html',
43:                                ampm=get_ampm(),
44:                                current_time=get_current_time(),
45:                                author=author,
46:                                books=books)
47:     response = flask.make_response(html)
48:     return response

```

PennyOnePage/index.html (Page 1 of 1)

```

1: <!DOCTYPE html>
2: <html>
3:
4:     <head>
5:         <title>Penny.com</title>
6:     </head>
7:
8:     <body>
9:         <hr>
10:         Good {{ampm}} and welcome to <strong>Penny.com</strong>
11:         <hr>
12:
13:         <h1>Author Search</h1>
14:         <form action="/index" method="get">
15:             Please enter an author name:
16:             <input type="text" name="author" value="{{author}}" autoFocus>
17:             <input type="submit" value="Go">
18:         </form>
19:         <hr>
20:         {% for book in books: %}
21:             {{book['isbn']}}:
22:             <strong>{{book['author']}}:</strong>
23:             {{book['title']}}<br>
24:         {% endfor %}
25:
26:         <hr>
27:         Date and time: {{current_time}}.<br>
28:         Created by <a href="https://www.cs.princeton.edu/~rdondero">
29:             Bob Dondero</a>
30:         <hr>
31:     </body>
32:
33: </html>

```

Baseline Example

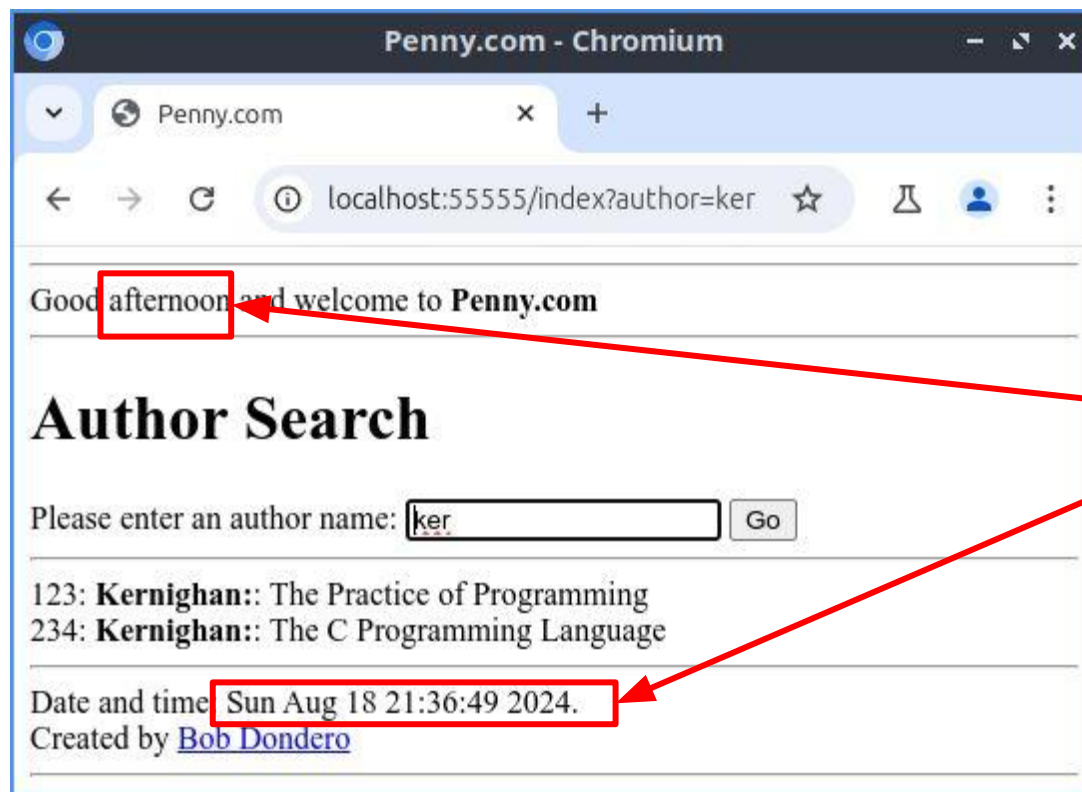
- PennyOnePage vs. PennyFlaskJinja:
 - (con) Doesn't illustrate multiple Flask routes (endpoints)
 - (con) Doesn't illustrate state handling
 - (pro) Users prefer?
 - (pro) Better example for this lecture!

Agenda

- Baseline example
- **JavaScript client-side web programming**
- AJAX
- AJAX via XMLHttpRequest
- AJAX via XMLHttpRequest enhancements
- AJAX wrap-up

JS Client-Side Web Pgmming

. Problem



Computed
once by
server!

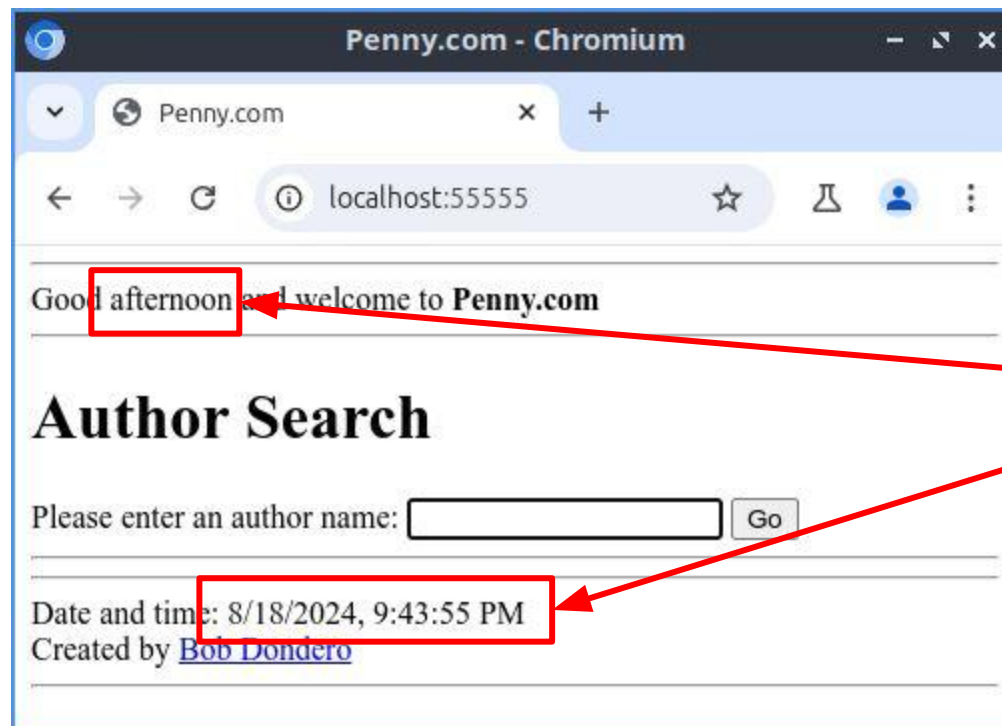
JS Client-Side Web Pgmming

- **Solution**

- Client-side web programming
- That is, program the browser...

JS Client-Side Web Pgmming

- See **PennyJavaScript** app



Computed
repeatedly
by client

JS Client-Side Web Pgmming

- See **PennyJavaScript** app (cont.)
 - runserver.py
 - penny.sql, penny.sqlite
 - database.py
 - **penny.py**
 - **index.html**

PennyJavaScript/penny.py (Page 1 of 1)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # penny.py
5: # Author: Bob Dondero
6: #-----
7:
8: import flask
9: import database
10:
11: #-----
12:
13: app = flask.Flask(__name__, template_folder='.')
14:
15: #-----
16:
17: @app.route('/', methods=['GET'])
18: @app.route('/index', methods=['GET'])
19: def index():
20:
21:     author = flask.request.args.get('author')
22:     if author is None:
23:         author = ''
24:     author = author.strip()
25:
26:     if author == '':
27:         books = []
28:     else:
29:         books = database.get_books(author) # Exception handling omitted
30:
31:     html_code = flask.render_template('index.html',
32:                                     author=author,
33:                                     books=books)
34:     response = flask.make_response(html_code)
35:     return response

```

PennyJavaScript/index.html (Page 1 of 1)

```

1: <!DOCTYPE html>
2: <html>
3:   <head>
4:     <title>Penny.com</title>
5:   </head>
6:
7:   <body>
8:
9:     <hr>
10:    Good <span id="ampmSpan"></span> and welcome to
11:    <strong>Penny.com</strong>
12:    <hr>
13:
14:    <h1>Author Search</h1>
15:    <form action="/index" method="get">
16:      Please enter an author name:
17:      <input type="text" name="author" value="{{author}}" autoFocus>
18:      <input type="submit" value="Go">
19:    </form>
20:    <hr>
21:    {% for book in books: %}
22:      {{book['isbn']}}:
23:      <strong>
24:        {{book['author']}}:
25:      </strong>
26:      {{book['title']}}<br>
27:    {% endfor %}
28:
29:    <hr>
30:    Date and time: <span id="datetimeSpan"></span><br>
31:    Created by <a href="https://www.cs.princeton.edu/~rdondero">
32:    Bob Dondero</a>
33:    <hr>
34:
35:    <script>
36:
37:      'use strict';
38:
39:      function getAmPm() {
40:        let dateTime = new Date();
41:        let hours = dateTime.getHours();
42:        let amPm = (hours < 12) ? 'morning': 'afternoon';
43:        let ampmSpan = document.getElementById('ampmSpan');
44:        ampmSpan.innerHTML = amPm;
45:      }
46:
47:      function getDateTime() {
48:        let dateTime = new Date();
49:        let datetimeSpan = document.getElementById('datetimeSpan');
50:        datetimeSpan.innerHTML = dateTime.toLocaleString();
51:      }
52:
53:      function setup() {
54:        getAmPm();
55:        window.setInterval(getAmPm, 1000);
56:        getDateTime();
57:        window.setInterval(getDateTime, 1000);
58:      }
59:
60:      document.addEventListener('DOMContentLoaded', setup);
61:
62:    </script>
63:  </body>
64: </html>

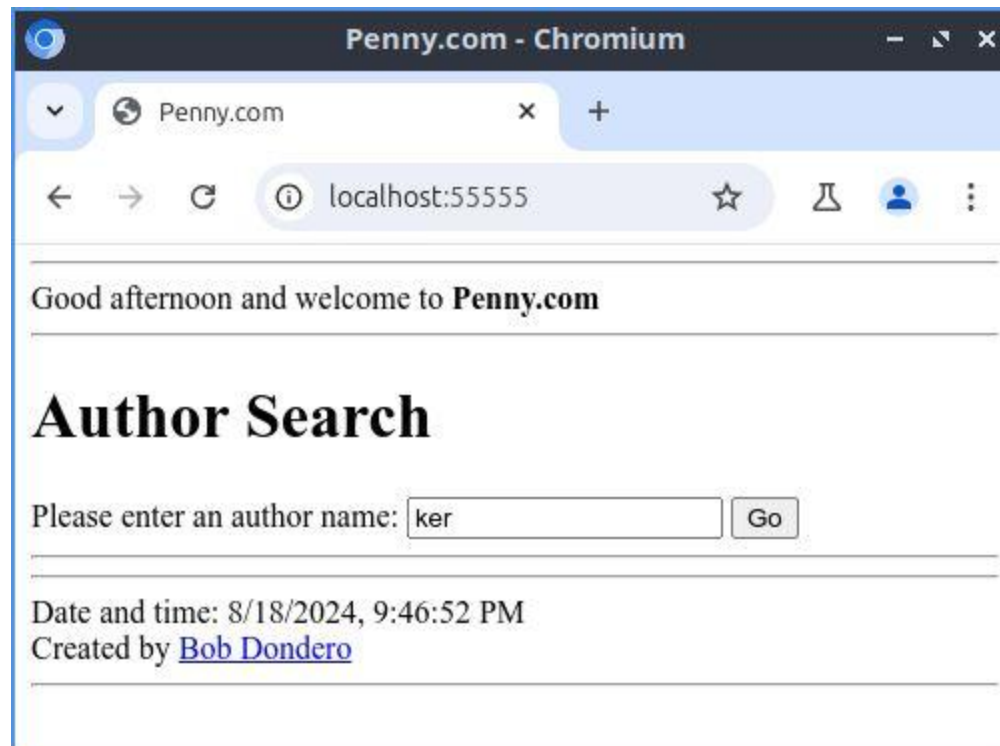
```

Agenda

- Baseline example
- JavaScript client-side web programming
- **AJAX**
- AJAX via XMLHttpRequest
- AJAX via XMLHttpRequest enhancements
- AJAX wrap-up

AJAX

- **Problem:**
 - Page state sometimes is inconsistent
 - Example: User types “ker”, but doesn’t yet click Go



AJAX

- **Solution:**
 - Revert to multi-page app, or
 - Stick with one-page app, and update the page with each keystroke...

AJAX

- **Problem:**
 - Inefficient to fetch an **entire** new page with each keystroke
- **Solution:**
 - Update **part of** the current page – the output element – with each keystroke

AJAX

- **Problem:**
 - Shouldn't update part of page **synchronously**; GUI would be “laggy”
- **Solution:**
 - Should update part of page **asynchronously**, while GUI remains responsive
- But how???

AJAX

- ***AJAX: Asynchronous JavaScript and XML***
 - **JavaScript**
 - AJAX is accomplished via function calls embedded in JavaScript code
 - **Asynchronous**
 - With AJAX, the browser communicates with the server asynchronously, and so remains responsive
 - **XML**
 - With AJAX, the response sent by the server is often (but not necessarily) a XML document

Agenda

- Baseline example
- JavaScript client-side web programming
- AJAX
- **AJAX via XMLHttpRequest**
- AJAX via XMLHttpRequest enhancements
- AJAX wrap-up

Aside: JSON in JavaScript

To convert a JSON doc to a JavaScript data structure:

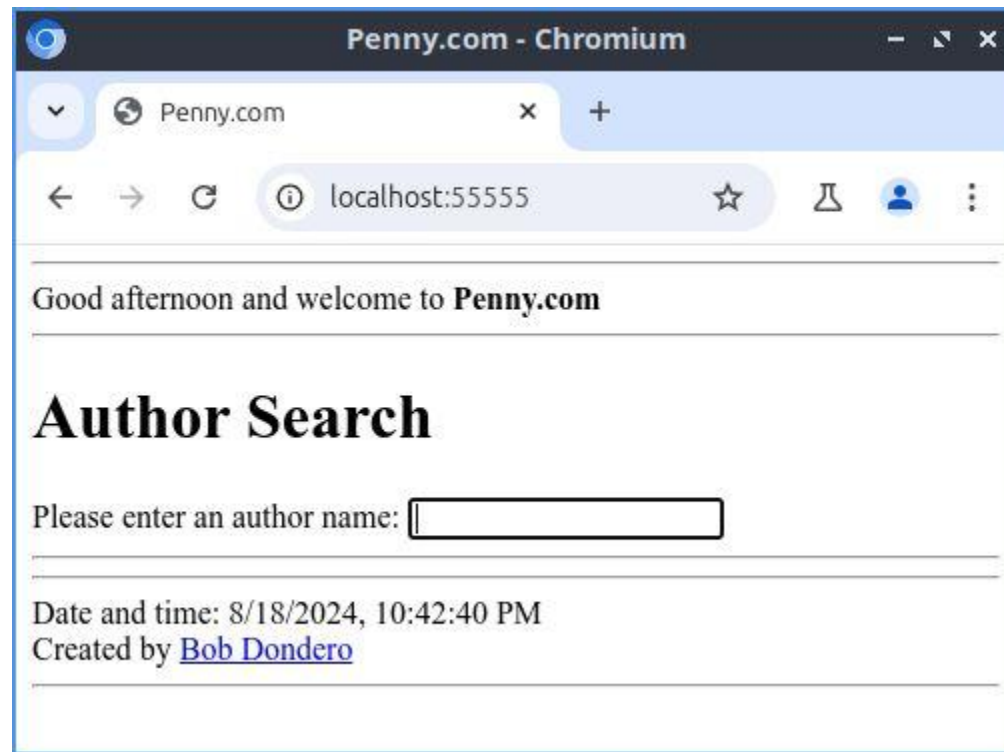
```
ds = JSON.parse(json_doc) ;
```

To convert a JavaScript data structure to a JSON doc:

```
json_doc = JSON.stringify(ds) ;
```

AJAX via XMLHttpRequest

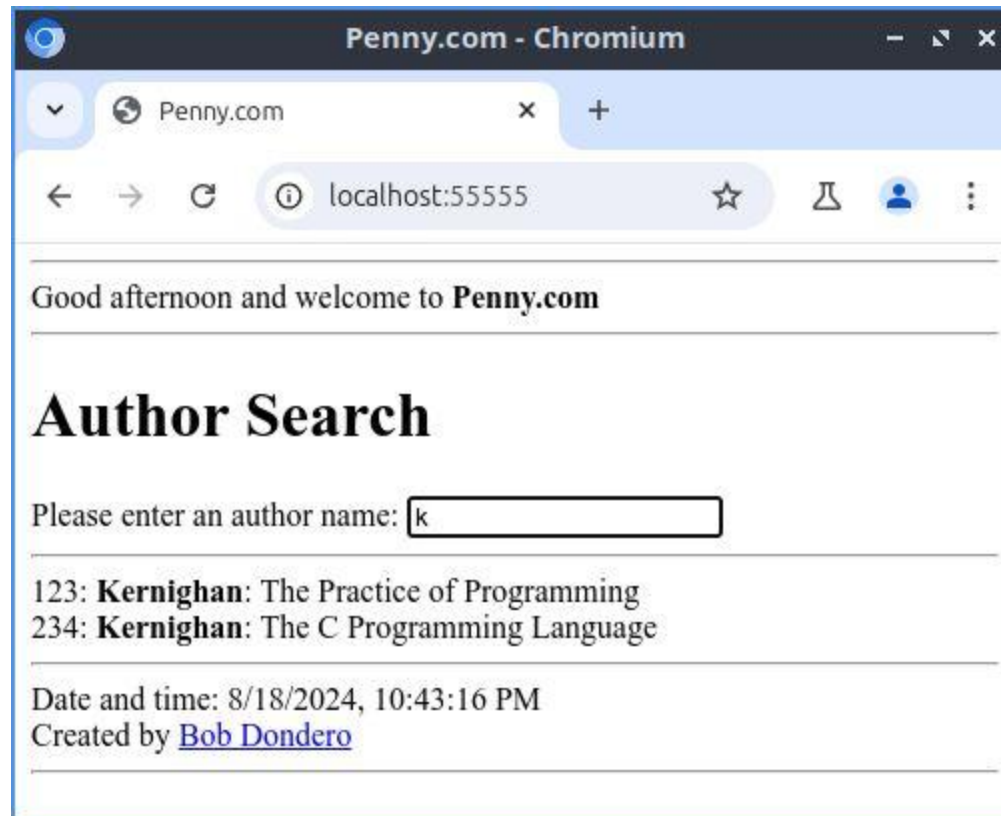
- See PennyAjax1 app



No
"Go"
button

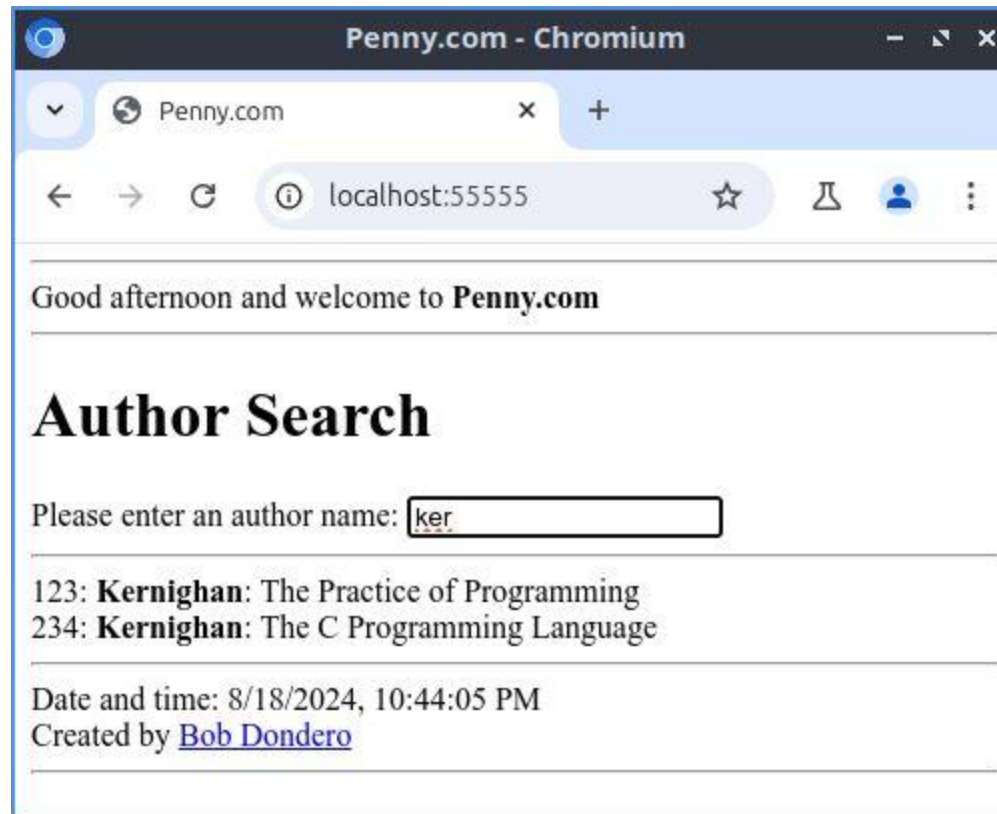
AJAX via XMLHttpRequest

- See **PennyAjax1** app (cont.)



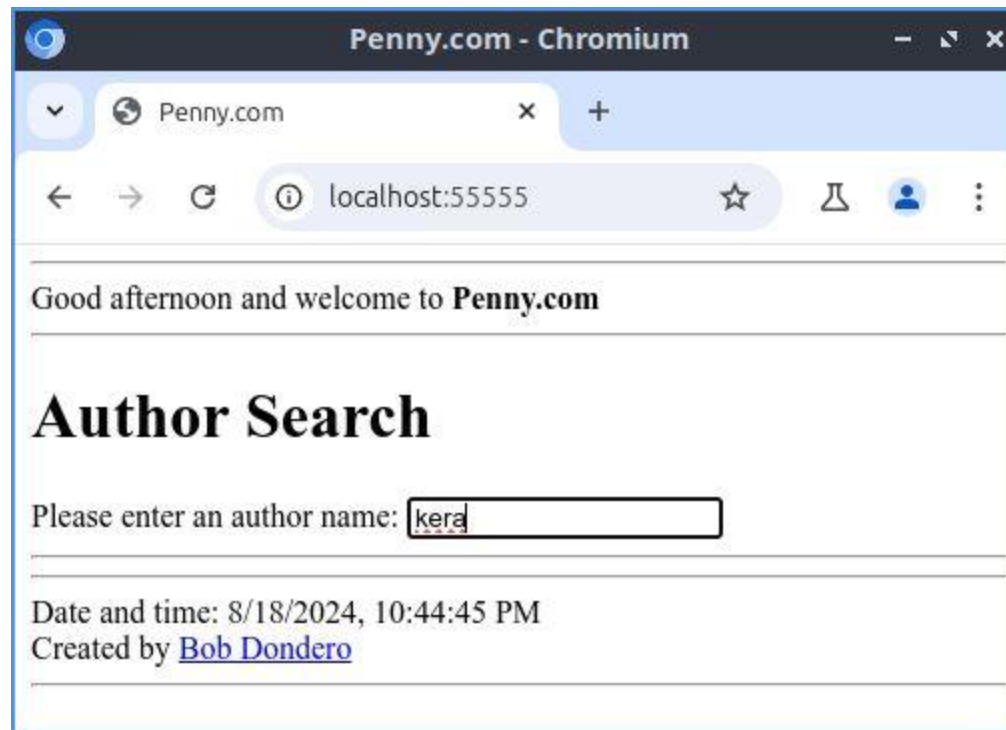
AJAX via XMLHttpRequest

- See **PennyAjax1** app (cont.)



AJAX via XMLHttpRequest

- See **PennyAjax1** app (cont.)



AJAX via XMLHttpRequest

- See **PennyAjax1** app (cont.)
 - runserver.py
 - penny.sql, penny.sqlite
 - database.py
 - **penny.py**
 - **index.html**

PennyAjax1/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.

PennyAjax1/index.html (Page 1 of 2)

```

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

```

PennyAjax1/index.html (Page 2 of 2)

```

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

```

AJAX via XMLHttpRequest

- See **PennyAjax1** app (cont.)
 - Note:
 - Could design `search_results()` to return a **HTML fragment** instead of a JSON doc
 - That would be more convenient if the client is a browser
 - That would be less convenient if the client is:
 - A desktop app
 - An Android app
 - An iOS app

Agenda

- Baseline example
- JavaScript client-side web programming
- AJAX
- AJAX via XMLHttpRequest
- **AJAX via XMLHttpRequest enhancements**
- AJAX wrap-up

AJAX Enhancements

- **Problem:**
 - Code to convert JavaScript data structure to HTML doc is ugly, inefficient
- **Solution:**
 - Use a **template engine**

AJAX Enhancements

- Python
 - Mustache, CheetahTemplate, Django, Genshi, **Jinja2**, Kid, Topsite, ...
- JavaScript
 - **Mustache**, Squirrelly, Handlebars, ...
- Java
 - Mustache, FreeMarker, Hamlets, Tiles, Thymeleaf, WebMacro, WebObjects, Velocity, ...

https://en.wikipedia.org/wiki/Web_template_system

AJAX Enhancements

- See **PennyAjax2** app
 - runserver.py
 - penny.sql, penny.sqlite
 - database.py
 - penny.py
 - **index.html**

PennyAjax2/index.html (Page 1 of 2)

```

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

```

PennyAjax2/index.html (Page 2 of 2)

```

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

```

AJAX Enhancements

- **How to fetch the Mustache library...**
- **Option 1**
 - Command browser to fetch Mustache library from the **cdn** website
- **Option 2**
 - Command browser to fetch Mustache library from *your website*

Aside: Mustache

- Template (informally)
 - HTML string with placeholders
 - Each placeholder is identified by a Mustache variable

```
Hello <strong>{ {username} }</strong>  
and welcome
```

Aside: Mustache

- To instantiate a template:

```
let map = {somevar: someval, ...};  
let html = Mustache.render(sometemplate, map);
```

- For each placeholder identified by `somevar` in `sometemplate`, replaces the placeholder with `someval`
- Automatically escapes `someval`
- Returns the resulting string

Aside: Mustache

- Template can contain:
 - Variables

```
... {{author}} ...
```

Aside: Mustache

- Template can contain:
 - Iteration constructs

```
{ { #books } }  
    <strong>{ { author } }</strong>  
    ...  
{ { /books } }
```

Note:

- Unusual implicit specification of iteration object
- If books is falsy, then block is not rendered

Aside: Mustache

- Template can contain:
 - Selection constructs

```
{ {#books} }  
...  
{ {/books} }  
{ {^books} }  
...  
{ {/books} }
```

If books is truthy, then first block is rendered

If books is falsy, then the second block is rendered

Aside: Mustache

- Template can contain:
 - Includes of other templates

```
...  
{ {>header} }  
...  
...  
{ {>footer} }  
...
```

Aside: Mustache

- There is more to Mustache
- For more information:
 - <https://github.com/janl/mustache.js>

AJAX Enhancements

- **Problem:**
 - Server will respond to requests in arbitrary order
- **Solution:**
 - Abort previous request

AJAX Enhancements

- See **PennyAjax3** app
 - runserver.py
 - penny.sql, penny.sqlite
 - database.py
 - penny.py
 - **index.html**

PennyAjax3/index.html (Page 1 of 2)

```

1: <!DOCTYPE html>
2: <html>
3:   <head>
4:     <title>Penny.com</title>
5:   </head>
6:   <body>
7:
8:
9:     <hr>
10:    Good <span id="ampmSpan"></span> and welcome to
11:    <strong>Penny.com</strong>
12:    <hr>
13:
14:    <h1>Author Search</h1>
15:    Please enter an author name:
16:    <input type="text" id="authorInput" autoFocus>
17:    <hr>
18:    <div id="resultsDiv"></div>
19:
20:    <hr>
21:    Date and time: <span id="datetimeSpan"></span><br>
22:    Created by <a href="https://www.cs.princeton.edu/~rdondero">
23:    Bob Dondero</a>
24:    <hr>
25:
26:    <script src=
27:    "https://cdn.jsdelivr.net/npm/mustache@4.2.0/mustache.min.js">
28:    </script>
29:
30:    <script>
31:
32:      'use strict';
33:
34:      function getAmPm() {
35:        let dateTime = new Date();
36:        let hours = dateTime.getHours();
37:        let amPm = (hours < 12) ? 'morning' : 'afternoon';
38:        let ampmSpan = document.getElementById('ampmSpan');
39:        ampmSpan.innerHTML = amPm;
40:      }
41:
42:      function getDateTime() {
43:        let dateTime = new Date();
44:        let datetimeSpan =
45:        document.getElementById('datetimeSpan');
46:        datetimeSpan.innerHTML = dateTime.toLocaleString();
47:      }
48:
49:      function convertToHtml(books) {
50:        let template = `
51:          {{#books}}
52:            {{isbn}}:
53:            <strong>{{author}}</strong>:
54:            {{title}}
55:            <br>
56:          {{/books}}
57:        `;
58:        let map = {books: books};
59:        let html = Mustache.render(template, map);
60:        return html;
61:      }
62:
63:      function handleResponse() {
64:        if (this.status !== 200) {
65:          alert('Error: Failed to fetch data from server');

```

PennyAjax3/index.html (Page 2 of 2)

```

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

```

AJAX Enhancements

- **Problem:**
 - Server could be overwhelmed with requests
- **Solution:**
 - *Debounce* the requests

AJAX Enhancements

- See **PennyAjax4** app
 - runserver.py
 - penny.sql, penny.sqlite
 - database.py
 - penny.py
 - **index.html**

PennyAjax4/index.html (Page 1 of 2)

```

1: <!DOCTYPE html>
2: <html>
3:   <head>
4:     <title>Penny.com</title>
5:   </head>
6:   <body>
7:
8:
9:     <hr>
10:    Good <span id="ampmSpan"></span> and welcome to
11:    <strong>Penny.com</strong>
12:    <hr>
13:
14:    <h1>Author Search</h1>
15:    Please enter an author name:
16:    <input type="text" id="authorInput" autoFocus>
17:    <hr>
18:    <div id="resultsDiv"></div>
19:
20:    <hr>
21:    Date and time: <span id="datetimeSpan"></span><br>
22:    Created by <a href="https://www.cs.princeton.edu/~rdondero">
23:    Bob Dondero</a>
24:    <hr>
25:
26:    <script src=
27:    "https://cdn.jsdelivr.net/npm/mustache@4.2.0/mustache.min.js">
28:    </script>
29:
30:    <script>
31:
32:      'use strict';
33:
34:      function getAmPm() {
35:        let dateTime = new Date();
36:        let hours = dateTime.getHours();
37:        let amPm = (hours < 12) ? 'morning' : 'afternoon';
38:        let ampmSpan = document.getElementById('ampmSpan');
39:        ampmSpan.innerHTML = amPm;
40:      }
41:
42:      function getDateTime() {
43:        let dateTime = new Date();
44:        let datetimeSpan =
45:        document.getElementById('datetimeSpan');
46:        datetimeSpan.innerHTML = dateTime.toLocaleString();
47:      }
48:
49:      function convertToHtml(books) {
50:        let template = `
51:          {{#books}}
52:            {{isbn}}:
53:            <strong>{{author}}</strong>:
54:            {{title}}
55:            <br>
56:          {{/books}}
57:        `;
58:        let map = {books: books};
59:        let html = Mustache.render(template, map);
60:        return html;
61:      }
62:
63:      function handleResponse() {
64:        if (this.status !== 200) {
65:          alert('Error: Failed to fetch data from server');

```

PennyAjax4/index.html (Page 2 of 2)

```

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

```

AJAX Enhancements

- **Bonus:**
 - Debouncing reduces (but does not eliminate) the need to abort requests!

Question (13webjavascript2)

- Does debouncing eliminate the need to abort previous AJAX requests? Answer “yes” or “no”.
 - Browse to <https://cos333attend.cs.princeton.edu> to answer

[2 points]

Agenda

- Baseline example
- JavaScript client-side web programming
- AJAX
- AJAX via XMLHttpRequest
- AJAX via XMLHttpRequest enhancements
- **AJAX wrap-up**

AJAX Wrap-Up

| AJAX Implementation | Browser Built-In or Library? | COS 333 Coverage? |
|------------------------------------|------------------------------|-------------------|
| XMLHttpRequest | Built-in | This lecture |
| <i>fetch & AbortController</i> | Built-in | Appendix |
| <i>Axios</i> | Library | None |
| <i>jQuery</i> | Library | Next lecture |

AJAX Wrap-Up

| AJAX Implementation | Firefox | Chrome |
|------------------------|------------|------------|
| XMLHttpRequest | 12+ (2012) | 31+ (2013) |
| fetch | 39+ (2015) | 42+ (2015) |
| AbortController | 57+ (2017) | 66+ (2018) |
| Axios | 12+ (2012) | 31+ (2013) |
| jQuery | 12+ (2012) | 31+ (2013) |

AJAX Wrap-Up

- PennyAjax app is a *single page app (SPA)*
- SPAs are enabled by AJAX

Summary

- We have covered:
 - Baseline example
 - JavaScript client-side web programming
 - AJAX
- See also:
 - **Appendix 1: AJAX via fetch**

Appendix 1: AJAX via fetch

AJAX via fetch

- **Option 1:**
 - **fetch()** function
 - Uses **promises**

AJAX via fetch

- See **PennyAjaxFetch1** app
 - runserver.py
 - penny.sql, penny.sqlite
 - database.py
 - penny.py
 - **index.html**

PennyAjaxFetch1/index.html (Page 1 of 2)

```

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

```

PennyAjaxFetch1/index.html (Page 2 of 2)

```

64:        return html;
65:      }
66:
67:      function usingTextUpdateResultsDiv(text) {
68:        let books = JSON.parse(text);
69:        let html = convertToHtml(books);
70:        let resultsDiv = document.getElementById('resultsDiv');
71:        resultsDiv.innerHTML = html;
72:      }
73:
74:      function handleError(err) {
75:        if (err.name !== 'AbortError')
76:          alert('Error: Failed to fetch data from server');
77:      }
78:
79:      let controller = null;
80:
81:      function getResults() {
82:        let authorInput = document.getElementById('authorInput');
83:        let author = authorInput.value;
84:        let encodedAuthor = encodeURIComponent(author);
85:        let url = '/searchresults?author=' + encodedAuthor;
86:        if (controller !== null)
87:          controller.abort();
88:        controller = new AbortController();
89:        fetch(url, {signal: controller.signal})
90:          .then(usingResponseGetText)
91:          .then(usingTextUpdateResultsDiv)
92:          .catch(handleError);
93:      }
94:
95:      let timer = null;
96:
97:      function debouncedGetResults() {
98:        clearTimeout(timer);
99:        timer = setTimeout(getResults, 500);
100:      }
101:
102:      function setup() {
103:        getAmPm();
104:        window.setInterval(getAmPm, 1000);
105:        getDateTime();
106:        window.setInterval(getDateTime, 1000);
107:        let authorInput = document.getElementById('authorInput');
108:        authorInput.addEventListener('input', debouncedGetResults);
109:      }
110:
111:      document.addEventListener('DOMContentLoaded', setup);
112:
113:    </script>
114:  </body>
115: </html>

```

AJAX via fetch

```
fetch(url)
  .then(usingResponseGetText)
  .then(usingTextUpdateResultsDiv)
  .catch(handleError) ;
```

- Fetch a response from `url`
- After that's finished, call `usingResponseGetText`, passing it the value returned by `fetch`
- After that's finished, call `usingTextUpdateResultsDiv`, passing it the value returned by `usingResponseGetText`
- If an exception occurs, call `handleError`, passing it the Error object

AJAX via fetch

```
if (this._controller !== null)
  this._controller.abort();
this._controller = new AbortController();

fetch(url, {signal: this._controller.signal})
  .then(usingResponseGetText)
  .then(usingTextUpdateResultsDiv)
  .catch(handleError);
```

Use of `AbortController` allows abort of request

AJAX via fetch

- **Option 2:**
 - **fetch()** function
 - Uses **promises**
 - **Async and await**

AJAX via fetch

- See **PennyAjaxFetch2** app
 - runserver.py
 - penny.sql, penny.sqlite
 - database.py
 - penny.py
 - **index.html**

PennyAjaxFetch2/index.html (Page 1 of 2)

```

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

```

PennyAjaxFetch2/index.html (Page 2 of 2)

```

62:          await fetch(url, {signal: controller.signal});
63:          if (! response.ok)
64:            throw new Error();
65:          let text = await response.text();
66:          let books = JSON.parse(text);
67:          let html = convertToHtml(books);
68:          let resultsDiv = document.getElementById('resultsDiv');
69:          resultsDiv.innerHTML = html;
70:        }
71:      } catch (err) {
72:        if (err.name !== 'AbortError')
73:          alert('Error: Failed to fetch data from server');
74:      }
75:    }
76:
77:    let controller = null;
78:
79:    function getResults() {
80:      let authorInput = document.getElementById('authorInput');
81:      let author = authorInput.value;
82:      let encodedAuthor = encodeURIComponent(author);
83:      let url = '/searchresults?author=' + encodedAuthor;
84:      if (controller !== null)
85:        controller.abort();
86:      controller = new AbortController();
87:      fetchBooks(url);
88:    }
89:
90:    let timer = null;
91:
92:    function debouncedGetResults() {
93:      clearTimeout(timer);
94:      timer = setTimeout(getResults, 500);
95:    }
96:
97:    function setup() {
98:      getAmPm();
99:      window.setInterval(getAmPm, 1000);
100:      getDateTime();
101:      window.setInterval(getDateTime, 1000);
102:      let authorInput = document.getElementById('authorInput');
103:      authorInput.addEventListener('input', debouncedGetResults);
104:    }
105:
106:    document.addEventListener('DOMContentLoaded', setup);
107:
108:  </script>
109: </body>
110: </html>

```