

# Client-Side Web Programming: JavaScript (Part 2)

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

# Objectives

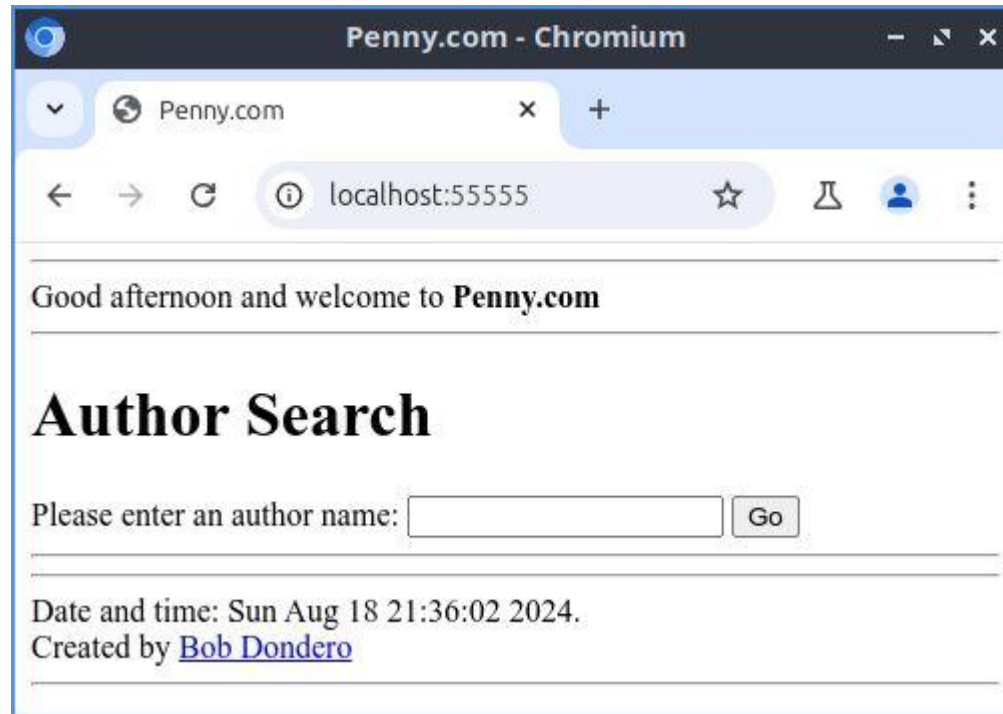
- We will cover:
  - Baseline example
  - JavaScript client-side web programming
  - AJAX

# Agenda

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

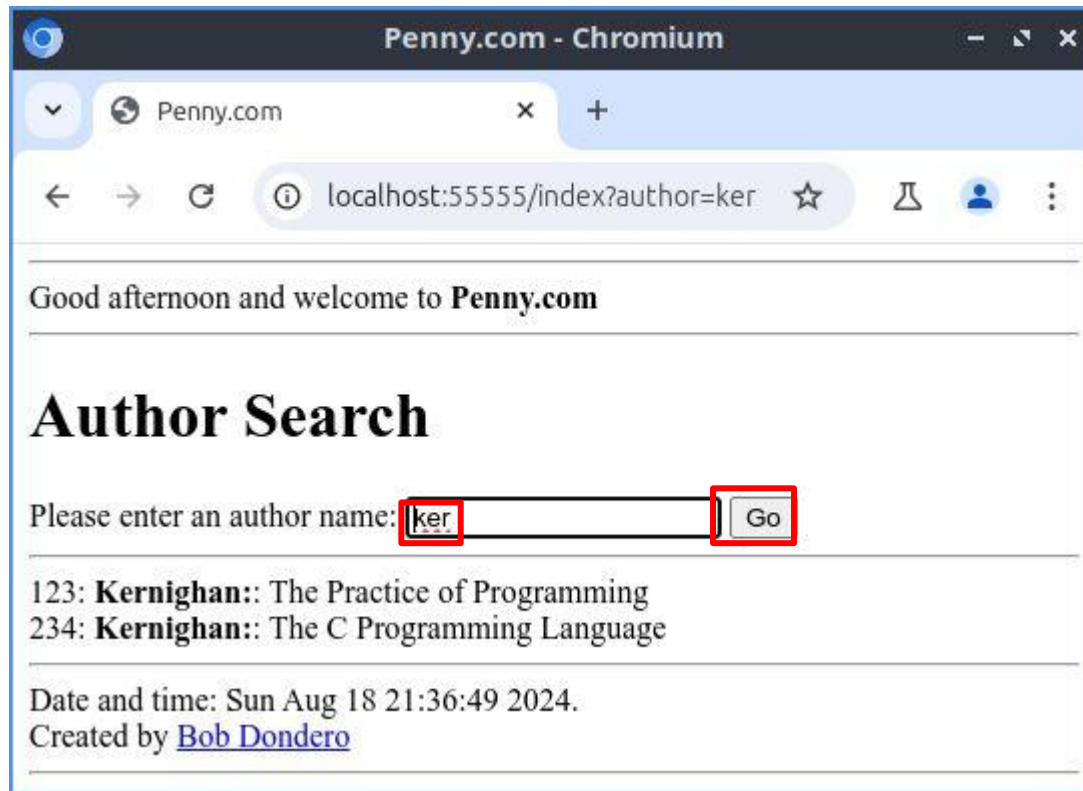
# Baseline Example

- 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**

# 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!

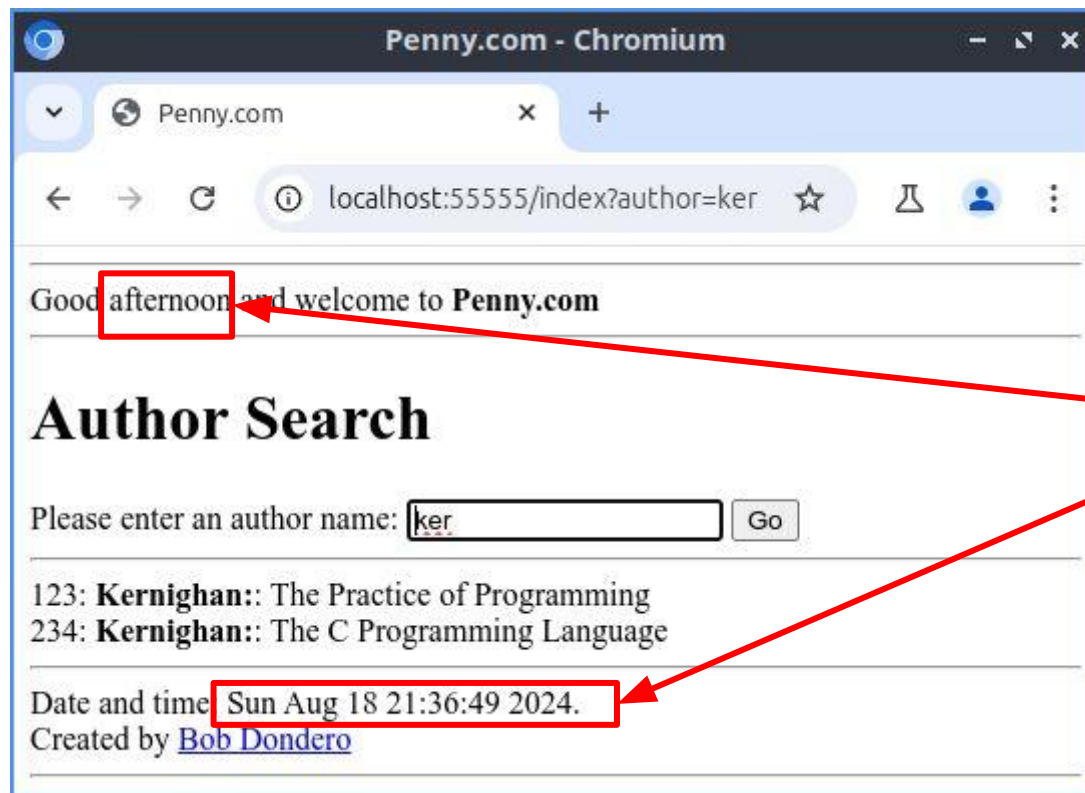
# Agenda

- 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!

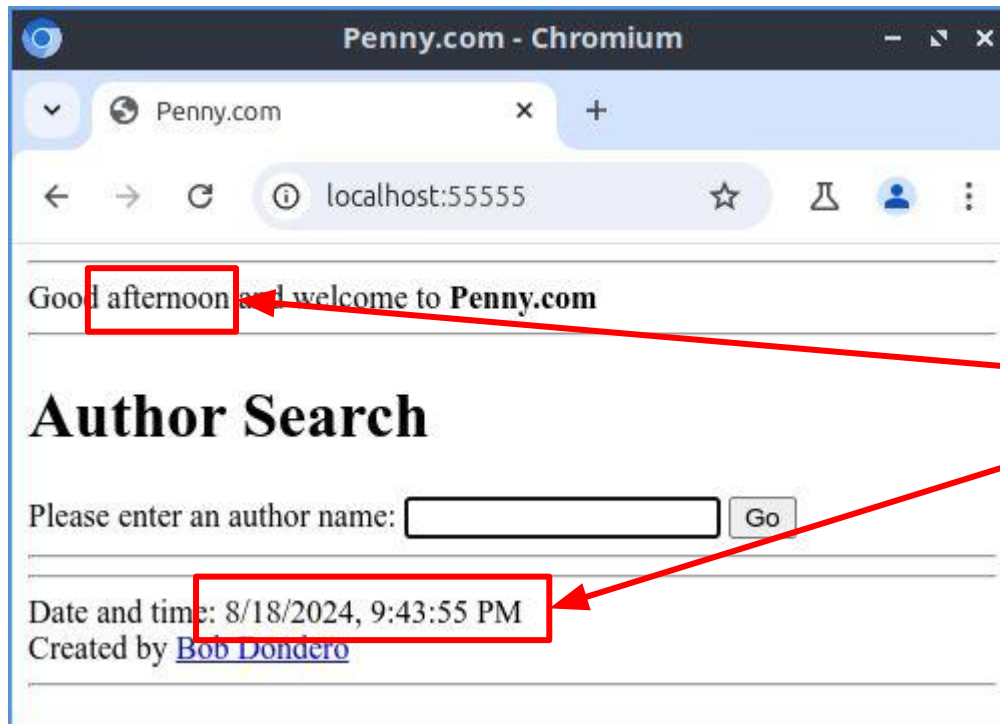
# JS Client-Side Web Prgmming

- **Solution**

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

# JS Client-Side Web Prgmming

- See **PennyJavaScript** app



Computed repeatedly by client

# JS Client-Side Web Pgmming

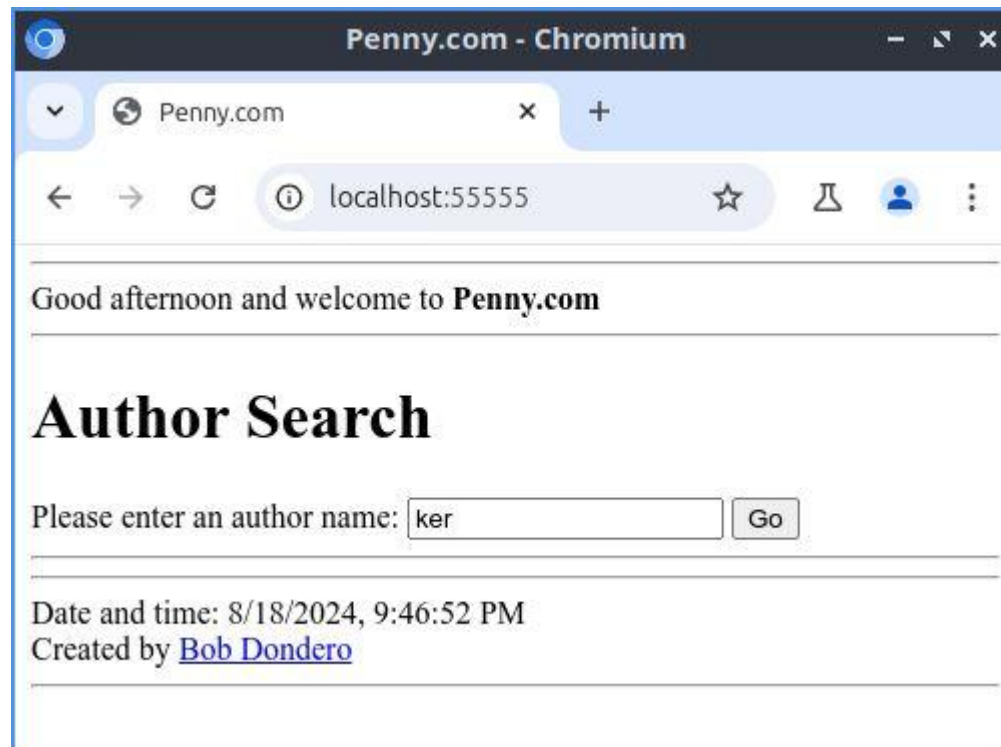
- See **PennyJavaScript** app (cont.)
  - runserver.py
  - penny.sql, penny.sqlite
  - database.py
  - **penny.py**
  - **index.html**

# 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:

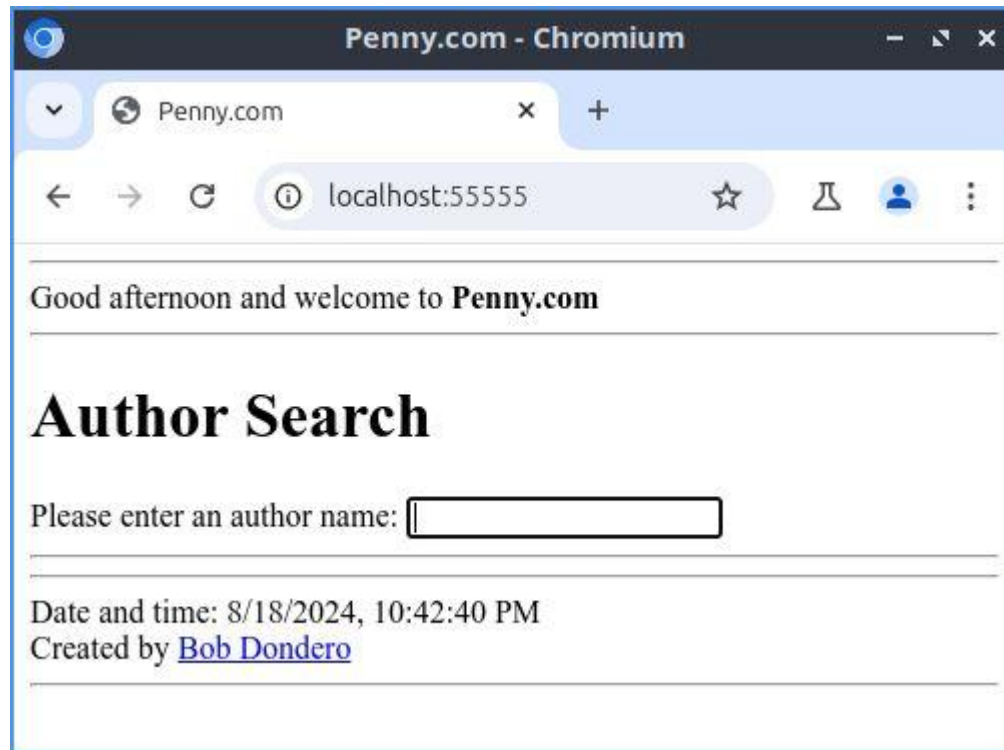
```
ds = JSON.parse(json_doc);
```

To convert a JavaScript data structure to a JSON doc:

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

# AJAX via XMLHttpRequest

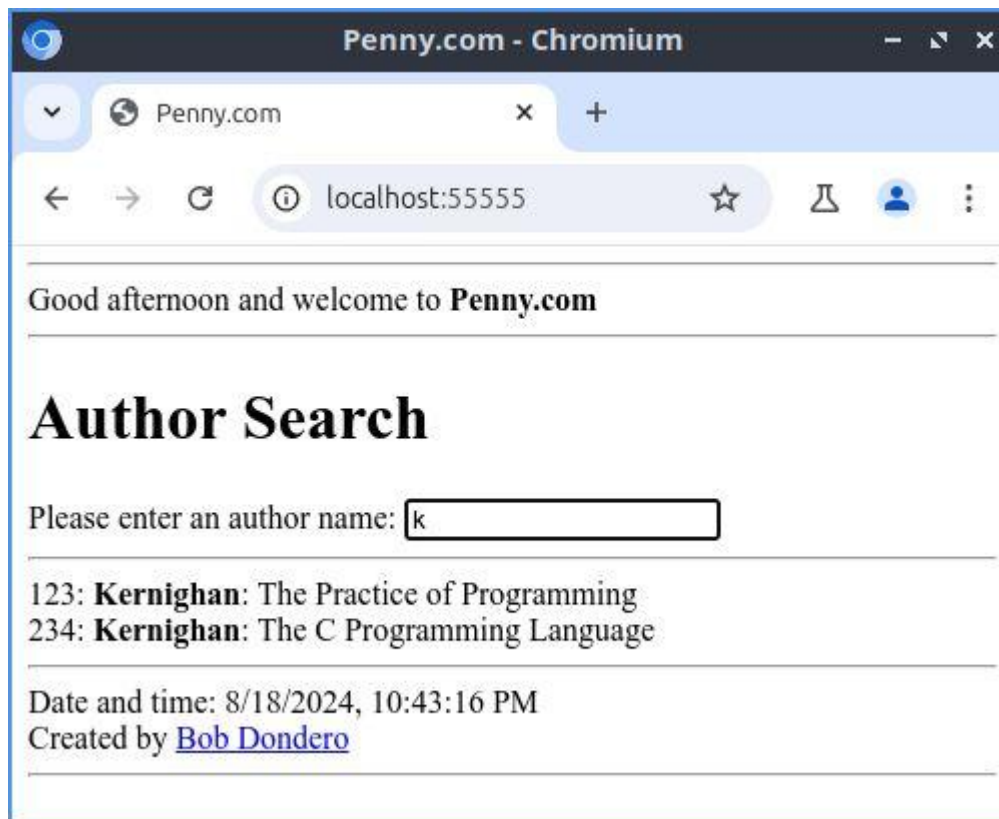
- See **PennyAjax1** app



No  
"Go"  
button

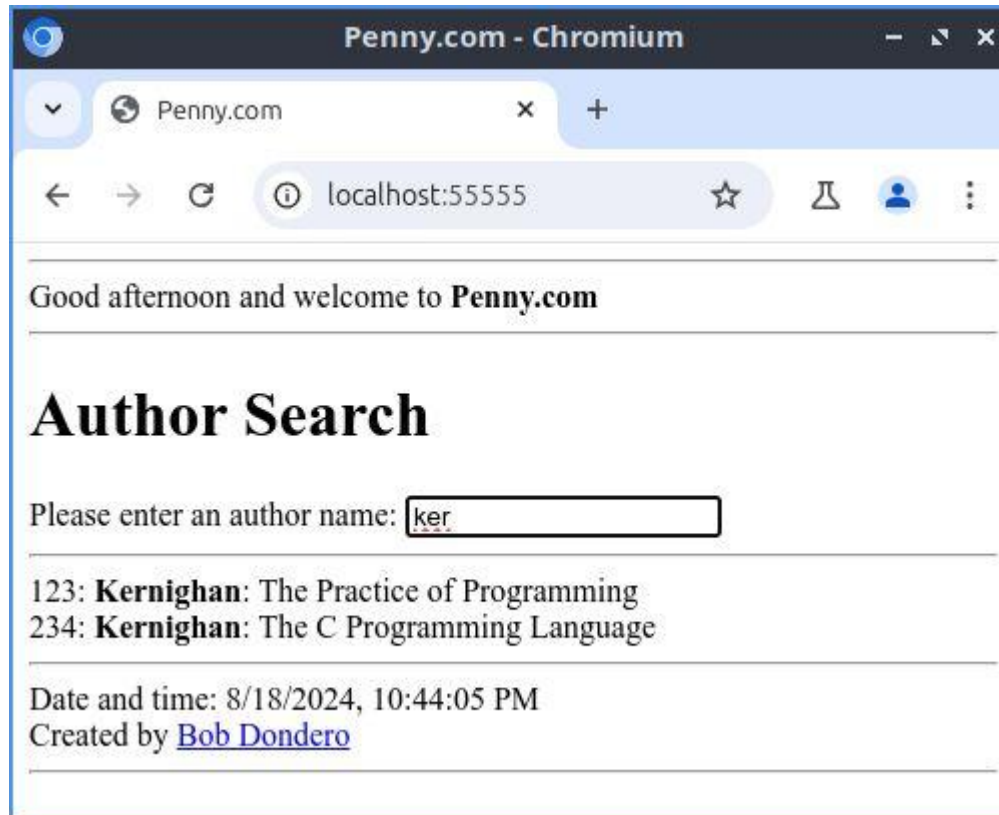
# AJAX via XMLHttpRequest

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



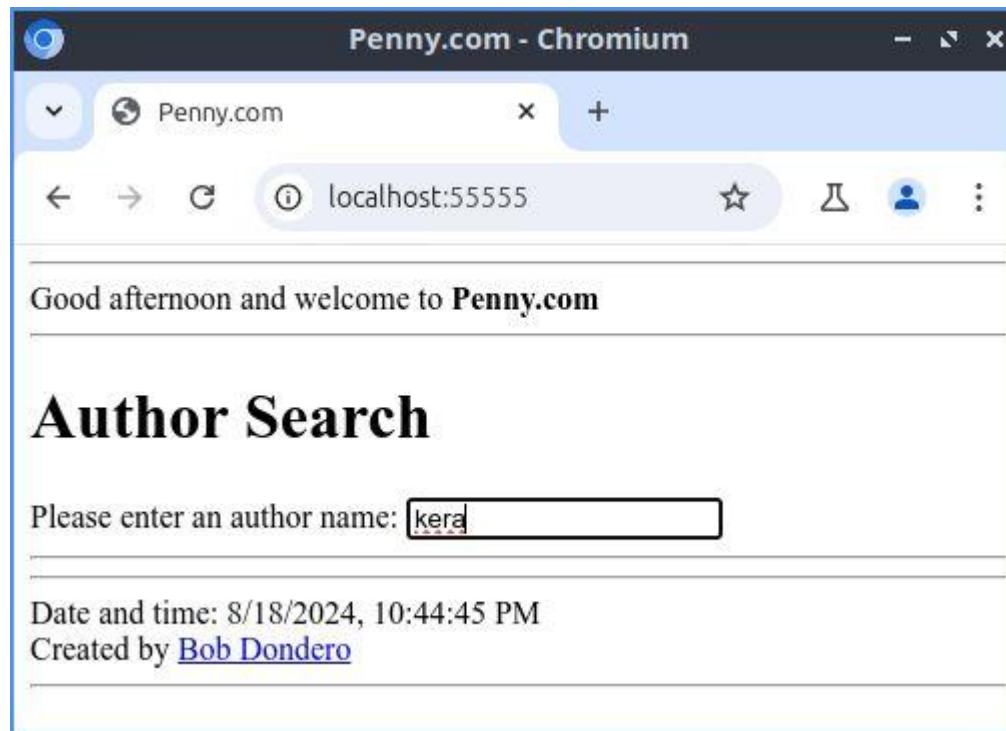
# AJAX via XMLHttpRequest

- 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**

# 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](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**

# 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**



# 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**

# AJAX Enhancements

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

# 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?
<b>XMLHttpRequest</b>	Built-in	This lecture
<i>fetch &amp; AbortController</i>	Built-in	Appendix
<i>Axios</i>	Library	None
<i>jQuery</i>	Library	Next lecture

# AJAX Wrap-Up

AJAX Implementation	Firefox	Chrome
<b>XMLHttpRequest</b>	12+ (2012)	31+ (2013)
<b>fetch</b>	39+ (2015)	42+ (2015)
<b>AbortController</b>	57+ (2017)	66+ (2018)
<b>Axios</b>	12+ (2012)	31+ (2013)
<b>jQuery</b>	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**

# 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**