Client-Side Web Programming: JavaScript (Part 2)

Robert M. Dondero, Ph.D.

Princeton University

Objectives

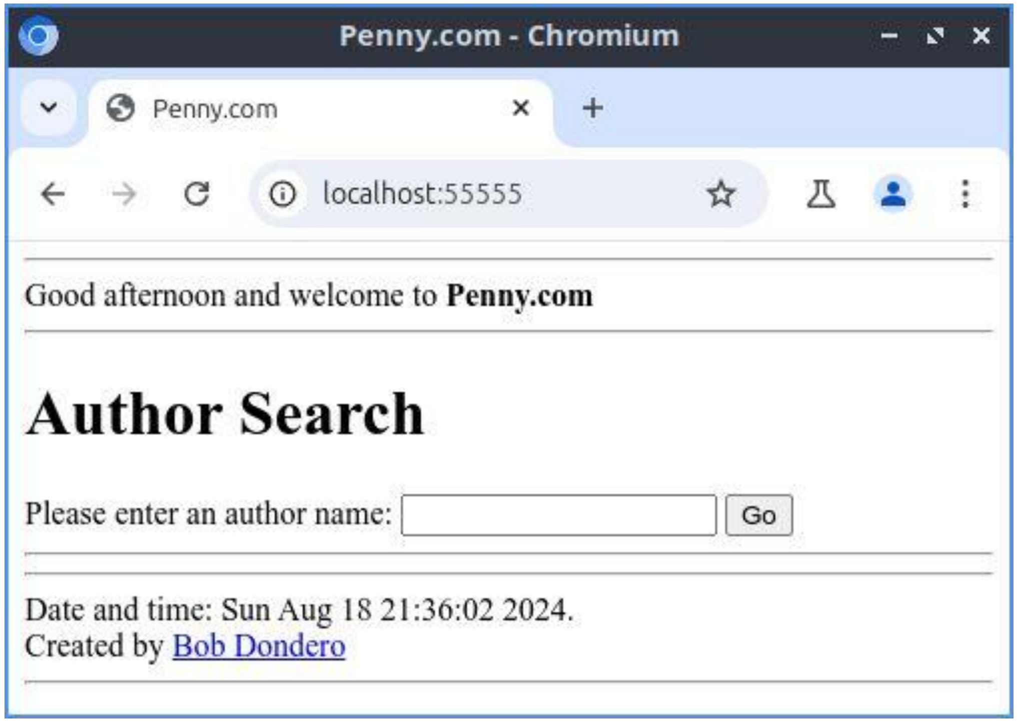
- We will cover:
  - Baseline example
  - JavaScript client-side web programming
  - AJAX

Agenda

- **Baseline example**
- JavaScript client-side web pgmming
- AJAX
- AJAX via XMLHttpRequest
- AJAX enhancement 1
- AJAX enhancement 2
- AJAX enhancement 3
- AJAX wrap-up

Baseline Example

- See **PennyOnePage** app



Good afternoon and welcome to **Penny.com**

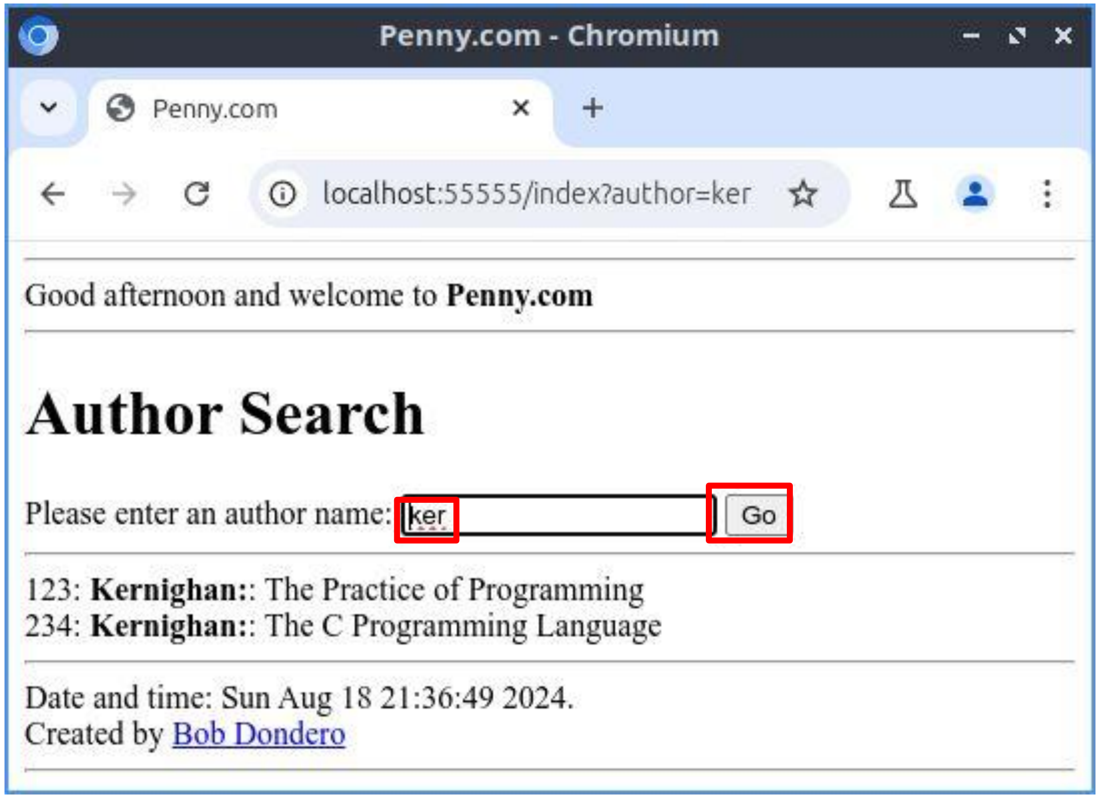
Author Search

Please enter an author name: [ ] Go

Date and time: Sun Aug 18 21:36:02 2024.
Created by Bob Dondero

Baseline Example

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



Good afternoon and welcome to **Penny.com**

Author Search

Please enter an author name: ker Go

123: **Kernighan:**: The Practice of Programming
234: **Kernighan:**: The C Programming Language

Date and time: Sun Aug 18 21:36:49 2024.
Created by Bob Dondero

Baseline Example

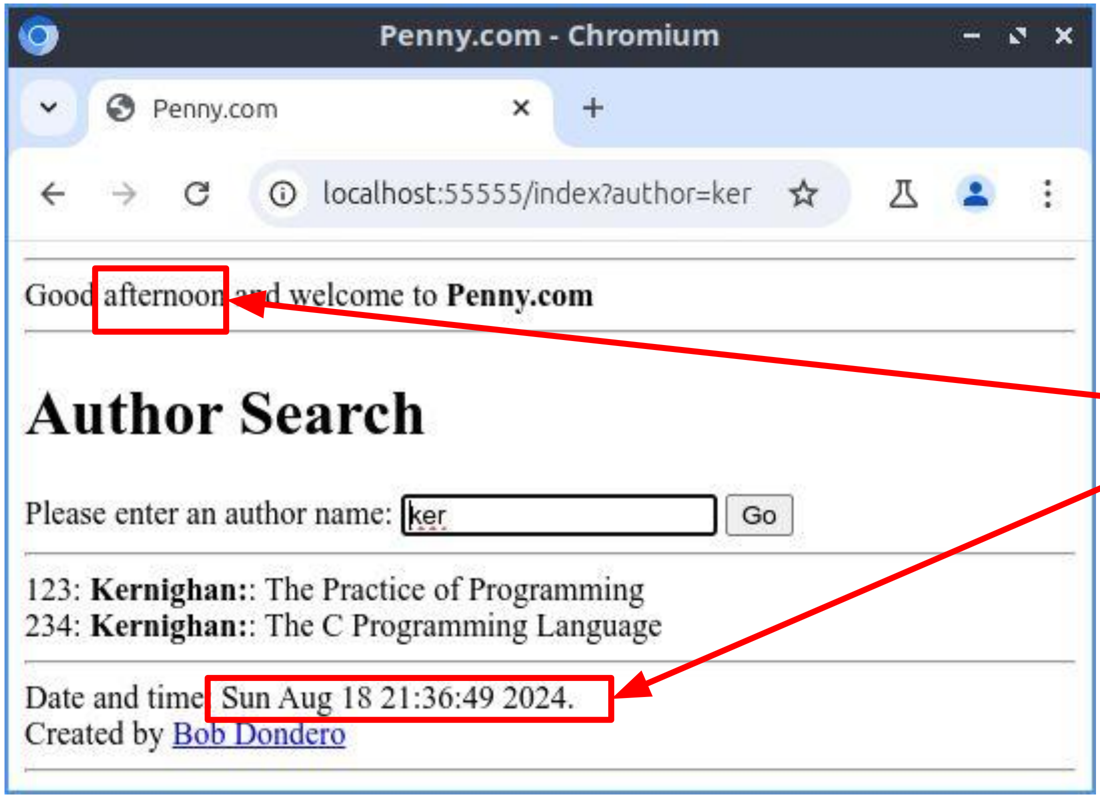
- See **PennyOnePage** app (cont.)
  - runserver.py
  - penny.sql, penny.sqlite
  - database.py
  - **penny.py**
  - **index.html**

Agenda

- Baseline example
- **JavaScript client-side web pgmming**
- AJAX
- AJAX via XMLHttpRequest
- AJAX enhancement 1
- AJAX enhancement 2
- AJAX enhancement 3
- AJAX wrap-up

JS Client-Side Web Pgmming

- **Problem**



Good afternoon and welcome to **Penny.com**

Author Search

Please enter an author name: ker

123: **Kernighan:**: The Practice of Programming
234: **Kernighan:**: The C Programming Language

Date and time: Sun Aug 18 21:36:49 2024.
Created by Bob Dondero

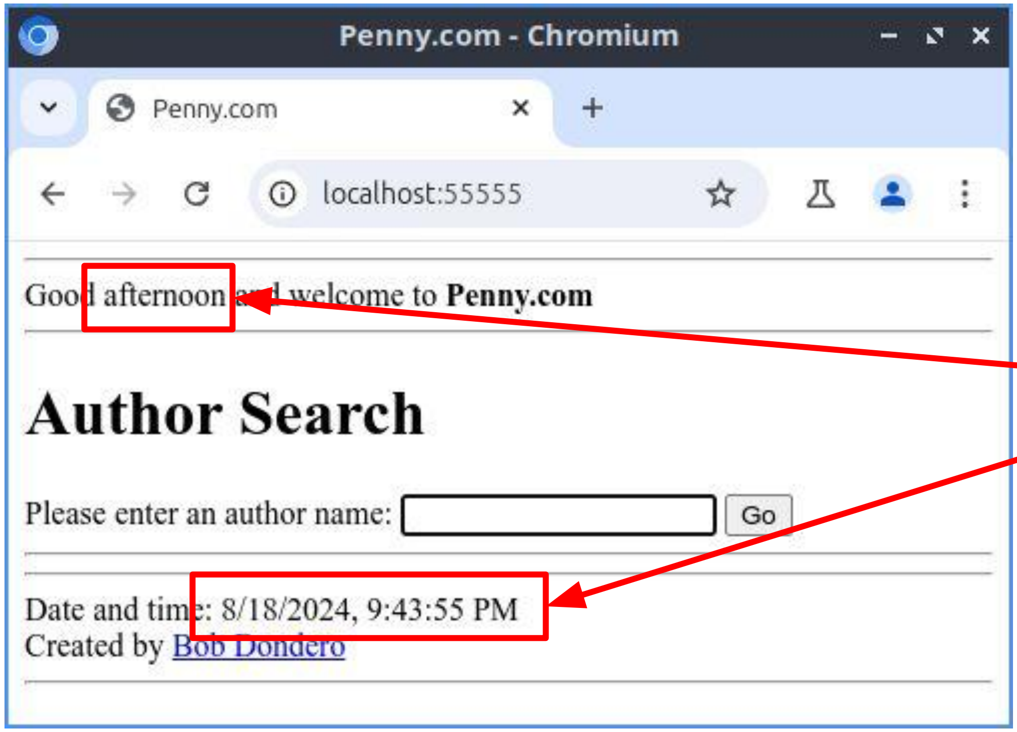
Computed once by server!

JS Client-Side Web Pgmming

- **Solution**
  - Client-side web programming
  - That is, program the browser…

JS Client-Side Web Pgmming

- See **PennyJavaScript** app



Good afternoon and welcome to **Penny.com**

Author Search

Please enter an author name:

Date and time: 8/18/2024, 9:43:55 PM
Created by Bob Dondero

Computed repeatedly by client

JS Client-Side Web Pgmming

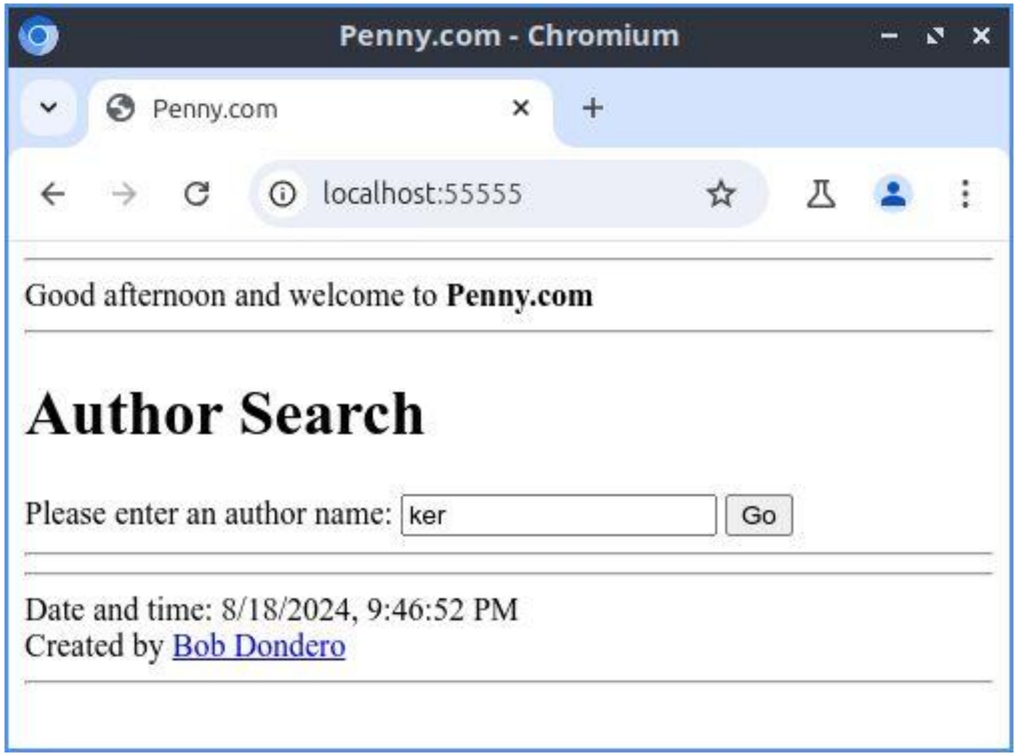
- See **PennyJavaScript** app (cont.)
  - runserver.py
  - penny.sql, penny.sqlite
  - database.py
  - **penny.py**
  - **index.html**

Agenda

- Baseline example
- JavaScript client-side web pgmming
- **AJAX**
- AJAX via XMLHttpRequest
- AJAX enhancement 1
- AJAX enhancement 2
- AJAX enhancement 3
- AJAX wrap-up

AJAX

- **Consider**:
  - User types “ker”, but doesn’t yet click Go



Good afternoon and welcome to **Penny.com**

Author Search

Please enter an author name: ker Go

Date and time: 8/18/2024, 9:46:52 PM
Created by Bob Dondero

AJAX

- **Problem:**
  - Page state sometimes is inconsistent
- **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 pgmming
- AJAX
- **AJAX via XMLHttpRequest**
- AJAX enhancement 1
- AJAX enhancement 2
- AJAX enhancement 3
- AJAX wrap-up

Aside: JSON in Python

Recall:
To convert a JSON doc to a **Python** data structure:

```
ds = json.loads(json_doc)
```

To convert a **Python** data structure to a JSON doc:

```
json_doc = json.dumps(ds)
```

Aside: JSON in JavaScript

To convert a JSON doc to a **JavaScript** data structure:

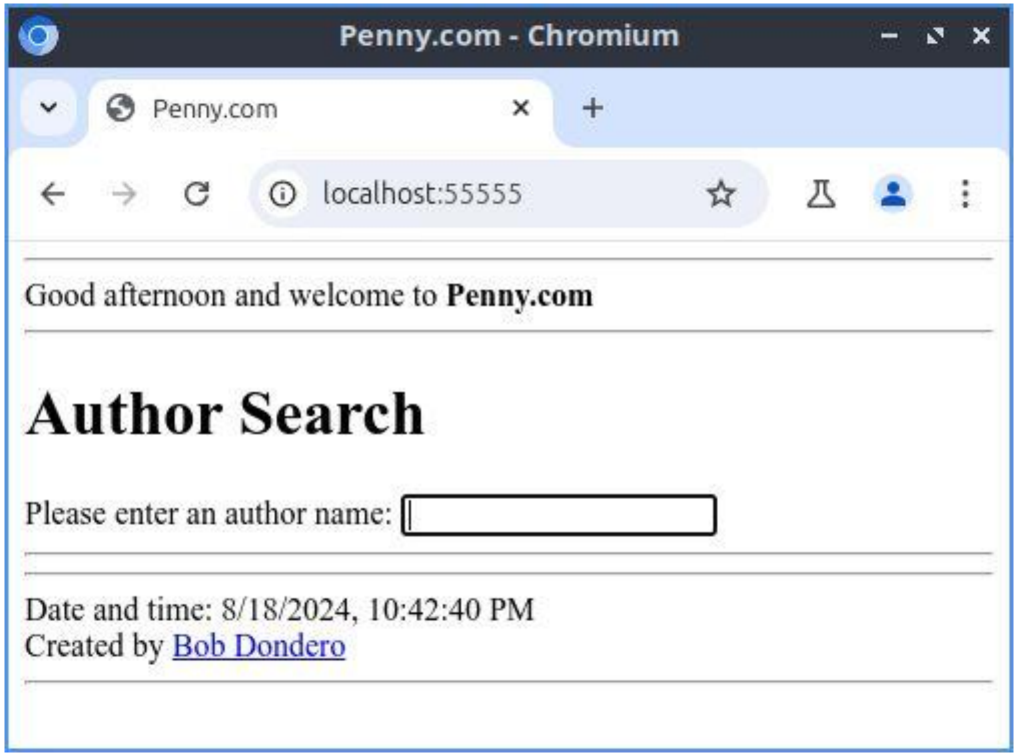
```
ds = JSON.parse(jsonDoc);
```

To convert a **JavaScript** data structure to a JSON doc:

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

AJAX via XMLHttpRequest

- See **PennyAjax1** app



Good afternoon and welcome to **Penny.com**

Author Search

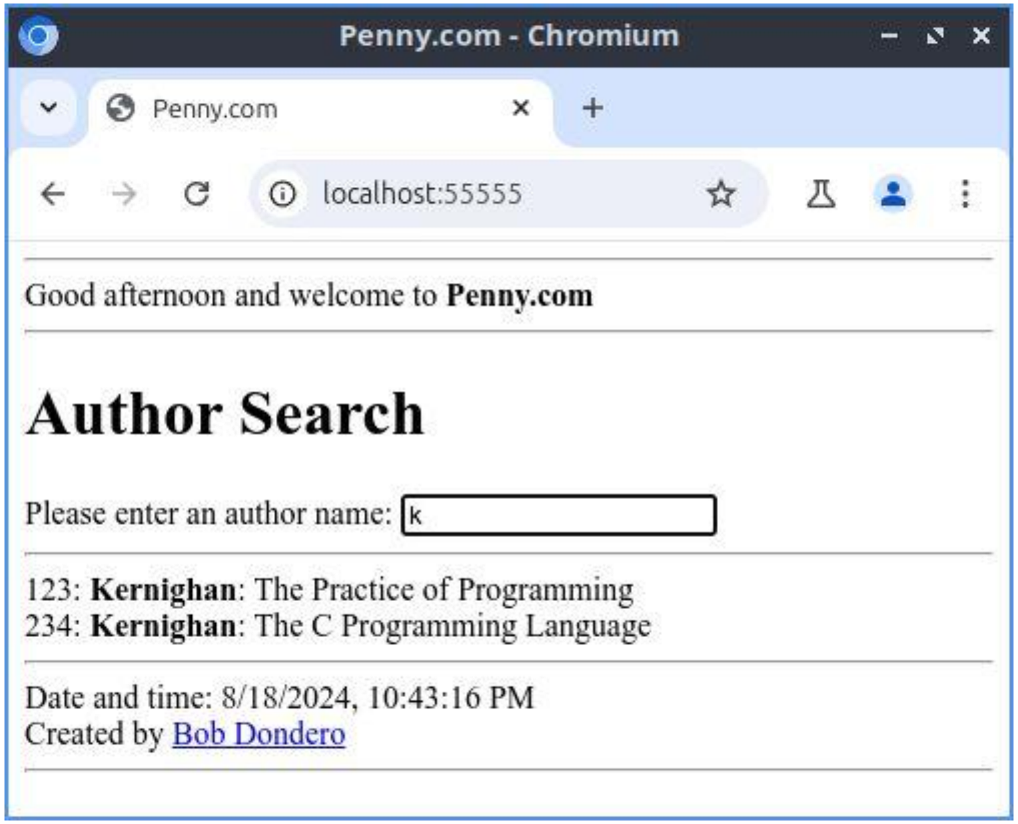
Please enter an author name:

Date and time: 8/18/2024, 10:42:40 PM
Created by Bob Dondero

No "Go" button

AJAX via XMLHttpRequest

- See **<u>PennyAjax1</u>** app (cont.)



Good afternoon and welcome to **Penny.com**

Author Search

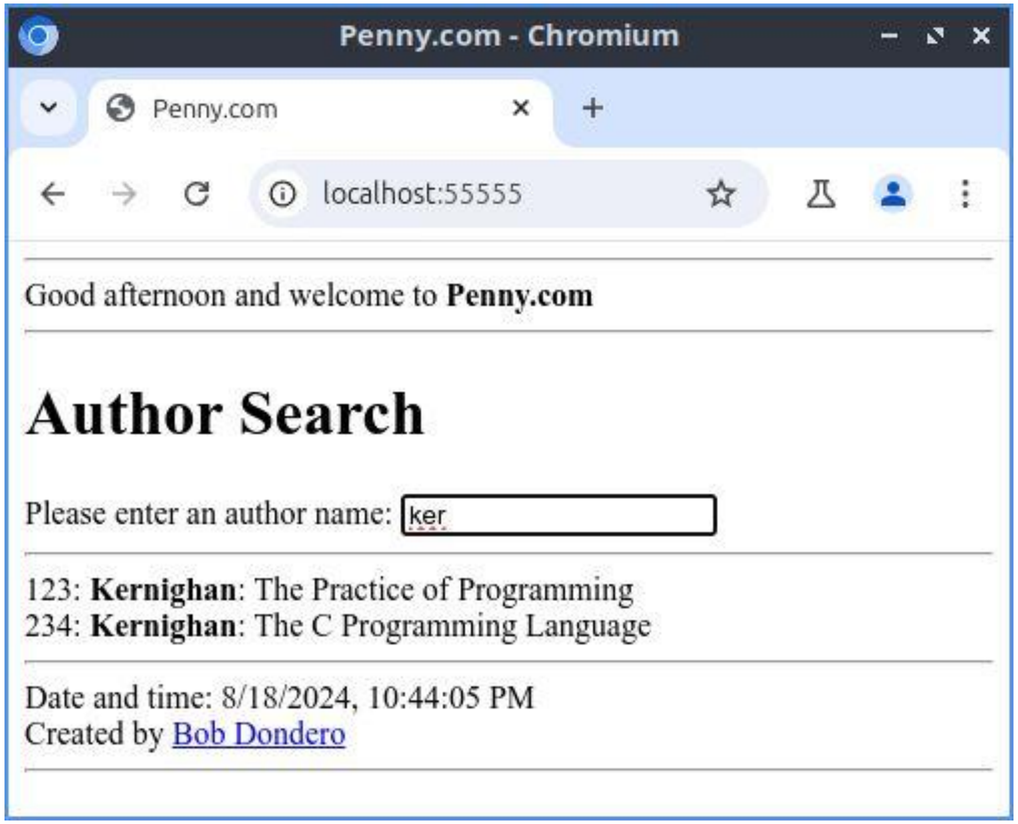
Please enter an author name: k

123: **Kernighan**: The Practice of Programming
234: **Kernighan**: The C Programming Language

Date and time: 8/18/2024, 10:43:16 PM
Created by Bob Dondero

AJAX via XMLHttpRequest

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



Good afternoon and welcome to **Penny.com**

Author Search

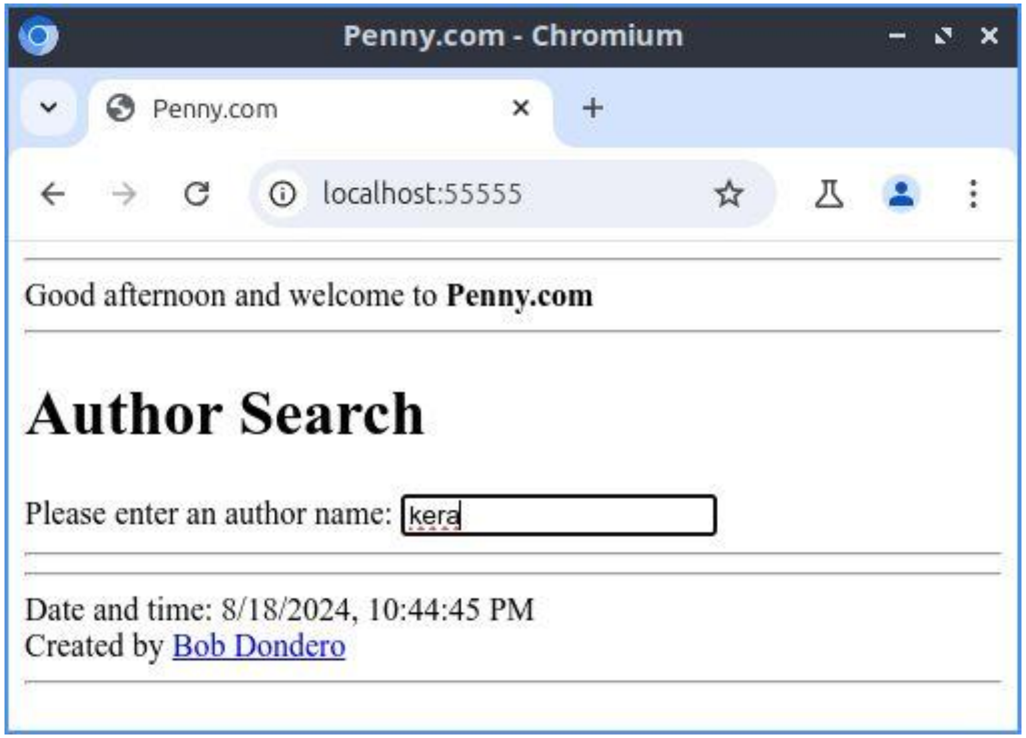
Please enter an author name: ker

123: **Kernighan**: The Practice of Programming
234: **Kernighan**: The C Programming Language

Date and time: 8/18/2024, 10:44:05 PM
Created by Bob Dondero

AJAX via XMLHttpRequest

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



Good afternoon and welcome to **Penny.com**

Author Search

Please enter an author name: kera

Date and time: 8/18/2024, 10:44:45 PM
Created by Bob Dondero

AJAX via XMLHttpRequest

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

Agenda

- Baseline example
- JavaScript client-side web pgmming
- AJAX
- AJAX via XMLHttpRequest
- **AJAX enhancement 1**
- AJAX enhancement 2
- AJAX enhancement 3
- AJAX wrap-up

AJAX Enhancement 1

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

AJAX Enhancement 1

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

Agenda

- Baseline example
- JavaScript client-side web pgmming
- AJAX
- AJAX via XMLHttpRequest
- AJAX enhancement 1
- **AJAX enhancement 2**
- AJAX enhancement 3
- AJAX wrap-up

AJAX Enhancement 2

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

AJAX Enhancement 2

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

Agenda

- Baseline example
- JavaScript client-side web pgmming
- AJAX
- AJAX via XMLHttpRequest
- AJAX enhancement 1
- AJAX enhancement 2
- **AJAX enhancement 3**
- AJAX wrap-up

AJAX Enhancement 3

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

AJAX Enhancement 3

- 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 Enhancement 3

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

AJAX Enhancement 3

- **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`, Mustache replaces the placeholder with `someval`
- Automatically sanitizes/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}}
```

```
{{#somearray}}
   <strong>{{.}}</strong>
{{/somearray}}
```

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

Agenda

- Baseline example
- JavaScript client-side web pgmming
- AJAX
- AJAX via XMLHttpRequest
- AJAX enhancement 1
- AJAX enhancement 2
- AJAX enhancement 3
- **AJAX wrap-up**

AJAX Wrap-Up

- PennyAjax app is a ***single page app (SPA)***
- SPAs are common
  - Google docs, Gmail, Slack, Ed, Netflix, …
- SPAs are implemented using AJAX

AJAX Wrap-Up

AJAX Implementation	Browser Built-In or Library?	Async Mechanism	COS 333 Coverage
XMLHttpRequest	Built-in	Callbacks	This lecture
fetch & AbortController	Built-in	Promises	This lecture appendix
Axios	Library	Promises	None
jQuery	Library	Callbacks (or promises)	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)

Lecture Summary

- In this lecture we 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(getText)
    .then(updateResultsDiv)
    .catch(handleError);
```

- Fetch a response from `url` and wait for it to finish
- Call `getText()`, passing it the value returned by `fetch`, and wait for it to finish
- Call `updateResultsDiv()`, passing it the value returned by `getText()`
- 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(getText)
   .then(updateResultsDiv)
   .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**

AJAX via fetch

```
try {
   let response = await fetch(url);
   if (! response.ok) throw new Error();
   let text = await response.text();
   …
}
catch (err) {
   …
}
```

- Fetch a response from `url`, wait for the operation to finish, and then assign the result to `response`
- Send a `text()` message to `response`, wait for the operation to finish, and then assign the result to `text`
- If an exception occurs, catch the Error object

AJAX via fetch

```
if (controller !== null)
   controller.abort();
controller = new AbortController()
try {
   let response = await fetch(url,
      {signal:controller.signal});
   if (! response.ok) throw new Error();
   let text = await response.text();
   …
}
catch (err) {
   …
}
```

Use of `AbortController` allows abort of request