Client-Side Web Programming: JavaScript (Part 2)

Copyright © 2022 by
Robert M. Dondero, Ph.D.
Princeton University

Objectives

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

Agenda

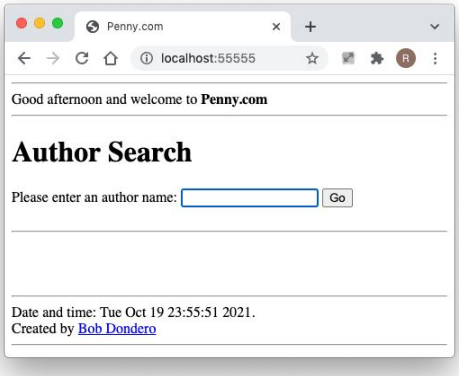
- **Baseline example**
- JavaScript client-side web pgmming
- AJAX
- AJAX via XMLHttpRequest
- (If time) AJAX via fetch
- AJAX wrap-up

Baseline Example

- The Penny web app can be expressed such that input & output areas are in same page...

Baseline Example

- See **PennyOnePage** app



5

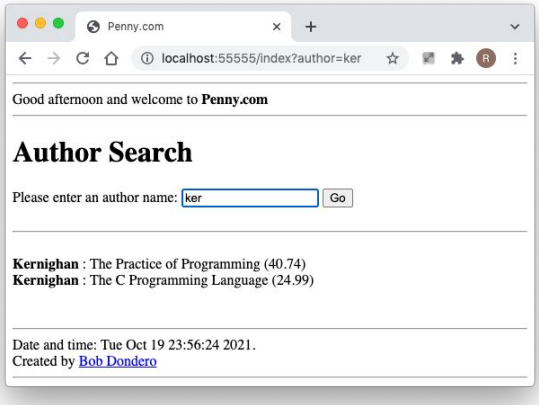
Baseline Example

[see slide}

Note: Immediately after page load, the Input element has keyboard focus.

Baseline Example

- See **PennyOnePage** app



Baseline Example

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

7

Baseline Example

[see slide]

Code notes: penny.py
Only one Flask route

Code notes: index.html
No reason to factor the header and footer into distinct templates

Baseline Example

- **PennyOnePage** vs. previous three-page versions:
  - (con) Doesn't illustrate state handling
  - (pro) Users prefer?
  - (pro) Better example for this lecture!

Agenda

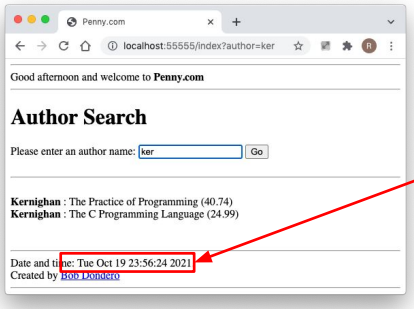
- Baseline example
- **JavaScript client-side web pgmming**
- AJAX
- AJAX via XMLHttpRequest
- (If time) AJAX via fetch
- AJAX wrap-up

JS Client-Side Web Pgmming

- What's wrong with PennyOnePage?

JS Client-Side Web Pgmming

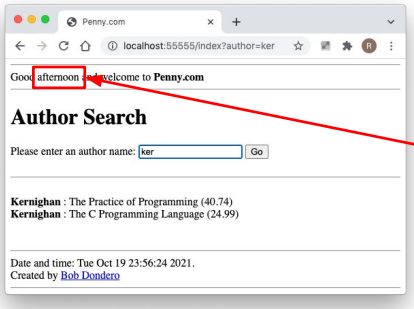
- **Problem**
  - Footer current date/time is computed once on server-side



Computed once by server!

JS Client-Side Web Pgmming

- **Problem**
  - Header “morning/afternoon” is computed once on server-side



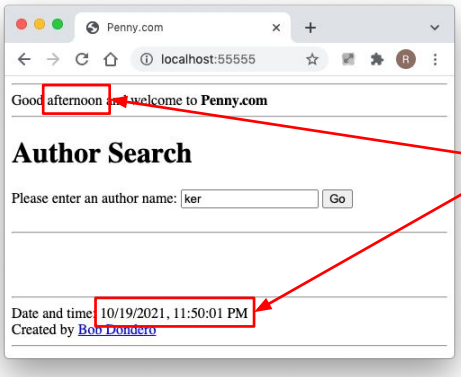
Computed once by server!

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
  - book.py
  - database.py
  - **penny.py**
  - **index.html**

15

JavaScript Client-Side Web Programming

[see slide]

Code notes: penny.py
Need not handle date/time

Code notes: search.html
Note HTML code
Note JavaScript code

Agenda

- Baseline example
- JavaScript client-side web pgmming
- **AJAX**
- AJAX via XMLHttpRequest
- (If time) AJAX via fetch
- AJAX wrap-up

AJAX

- What's wrong with PennyJavaScript?

AJAX

- **Problem**:
  - Page state sometimes is inconsistent!!!
    - Example: User types “ker”, but doesn’t yet click Go
- **Solution**:
  - Refresh page with each keystroke

18

AJAX

Problem:

Page state sometimes is inconsistent!!!

Example: User types “ker” but doesn’t yet click Go

Generalizing:

When page contains both input and output elements, there is the chance of inconsistent page states

Solution:

Split into a multi-page app, or
Refresh page with each keystroke

AJAX

- **Problem**:
  - Horribly inefficient to refresh **entire** page
  - Horribly inefficient to do so in the **foreground**; GUI would be “laggy”
- **Solution**:
  - Refresh **part of** page with each keystroke
  - Do so in the **background**, while GUI remains responsive
  - But how???

19

JavaScript Client-Side Web Programming

Problem:

Horribly inefficient to refresh entire page
GUI would be laggy

Solution:

Refresh **part of** page with each keystroke
Do so while GUI remains responsive
But how???

AJAX: Motivation

- **Goal 1**: Consistent page state
  - Browser should refresh output element(s) with each user keystroke
- **Goal 2**: Responsive GUI
  - Browser should remain responsive to user keystrokes

AJAX: Motivation

Goal 1: Consistent page state
Browser should refresh book list with each user keystroke
User should not need to click “Go” button
Page state always should be consistent

Goal 2: Responsive GUI
Browser should remain responsive to user keystrokes
Browser should not block user typing while fetching data from server

AJAX: Definition

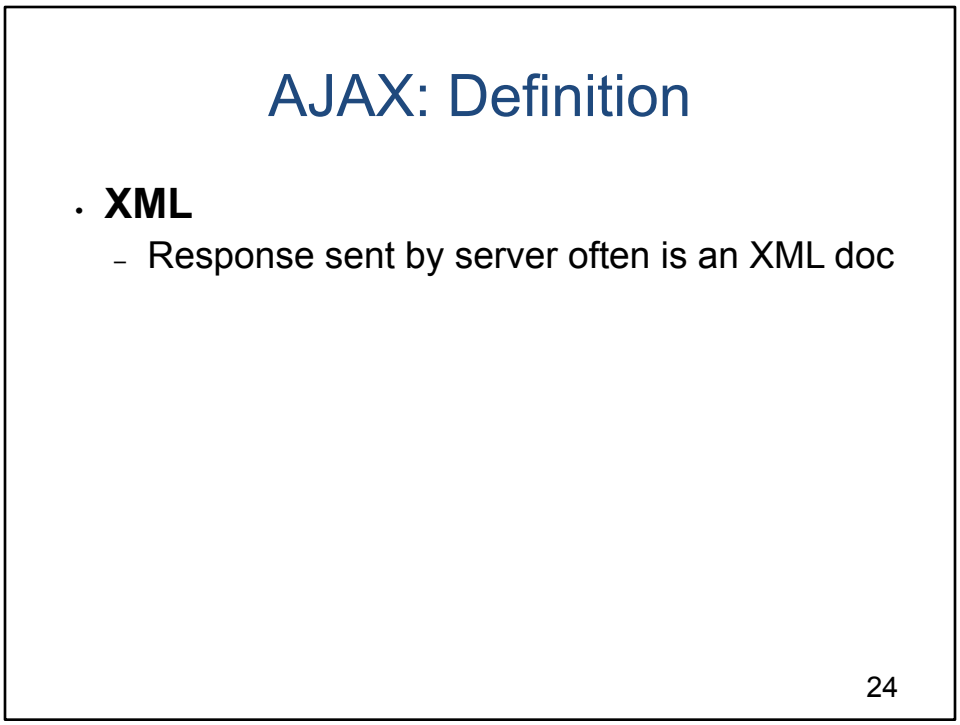
- ***AJAX: Asynchronous JavaScript and XML***
  - Let's consider each part of the name…
- **JavaScript**
  - AJAX is accomplished via function calls embedded in JavaScript code

AJAX: Definition

- **Asynchronous**
  - User types key; browser accepts keystroke
  - JavaScript engine tells browser to send request to HTTP server; JavaScript engine provides callback function to browser
  - JavaScript engine proceeds

AJAX: Definition

- **Asynchronous** (cont.)
  - Sometime later at an unknown time (**asynchronously**)…
  - HTTP server sends response to browser
  - Browser places call of callback function on JavaScript event queue
  - When callback function reaches front of JavaScript event queue…
  - JavaScript engine calls callback function
  - Callback function affects DOM



AJAX: Definition

XML

Response sent by server often is (but need not be) an XML doc

Response could be a JSON doc, HTML fragment, plain text, …

(AJA“X” is a misnomer)

We’ll cover XML and JSON soon

Agenda

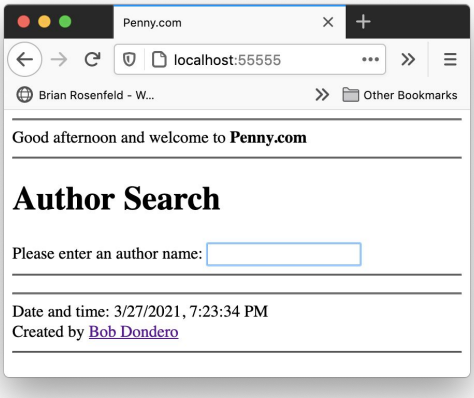
- Baseline example
- JavaScript client-side web pgmming
- AJAX
- **AJAX via XMLHttpRequest**
- (If time) AJAX via fetch
- AJAX wrap-up

AJAX via XMLHttpRequest

- See **PennyAjax** app
  - What it does...

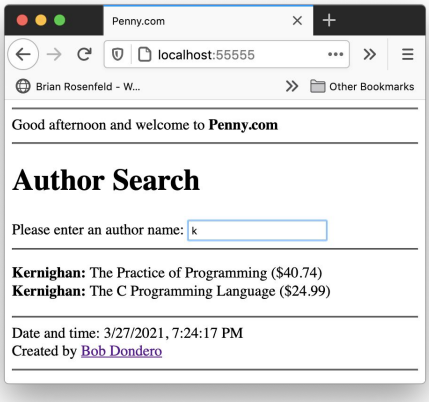
AJAX via XMLHttpRequest

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



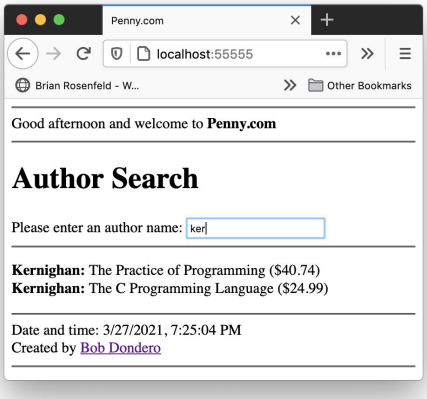
AJAX via XMLHttpRequest

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



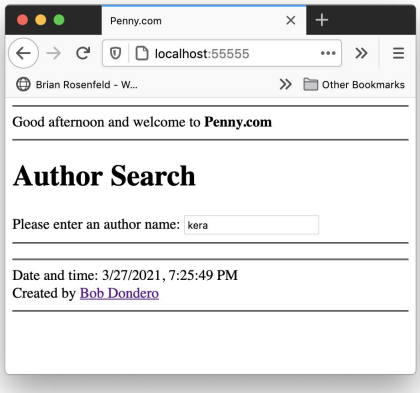
AJAX via XMLHttpRequest

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



AJAX via XMLHttpRequest

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



AJAX via XMLHttpRequest

- See **PennyAjax** app (cont.)
  - How it works…
  - Uses AJAX via ***XMLHttpRequest***

AJAX via XMLHttpRequest

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

32

AJAX: Example

Code notes: penny.py
index() simply returns index.html
search_results() returns HTML fragment, not HTML page

Code notes: index.html
Use of XMLHttpRequest

[See next slides for dynamic trace]

AJAX via XMLHttpRequest

- See **PennyAjax** app (cont.)
  - Dynamic trace:
    - **Step 1**: Browser sends HTTP request to app for / or /index
    - **Step 2**: App receives HTTP request, calls index(), and delivers page (`index.html`) to browser

AJAX via XMLHttpRequest

- See **PennyAjax** app (cont.)
  - Dynamic trace (cont.):
    - **Step 3**: Browser receives HTTP response, renders page, interprets JavaScript code, calls `setup()`, and installs `getResults()` as event handler for input events on `input` element
    - **Step 4**: User types 'k' in input element

AJAX via XMLHttpRequest

- See **PennyAjax** app (cont.)
  - Dynamic trace (cont.):
    - **Step 5**: Browser calls `getResults()`, sends HTTP request to app for /searchresults?author='k', and remains responsive to user input
    - **Step 6**: App receives HTTP request, calls `search_results()`, and delivers HTML fragment (book list) to browser

AJAX via XMLHttpRequest

- See **PennyAjax** app (cont.)
  - Dynamic trace (cont.):
    - **Step 7**: Browser receives HTML fragment, queues call of `request.processReadyStateChange()`

AJAX via XMLHttpRequest

- See **PennyAjax** app (cont.)
  - Dynamic trace (cont.):
    - **Step 8**: Eventually, browser calls `request.processReadyStateChange()` – validates HTTP response, assigns `request.responseText` to `resultsParagraph.innerHTML`
    - **Step 9**: Browser displays book list

AJAX via XMLHttpRequest

- **Question**:
  - Why abort previous request?
- **Answer**:
  - We're interested in the response for only the most recent request
  - Before sending a new request, abort the old one
    - Doesn't hurt to abort the old request if it has been completely processed

Agenda

- Baseline example
- JavaScript client-side web pgmming
- AJAX
- AJAX via XMLHttpRequest
- **(If time) AJAX via fetch**
- AJAX wrap-up

AJAX via fetch

- See **PennyAjaxFetch** app
  - What it does…
    - Same as PennyAjax
  - How it works…
    - Uses AJAX via ***fetch***

AJAX via fetch

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

AJAX via fetch

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

fetch(url, {signal: this._controller.signal})
   .then(usingResponseGetText)
   .then(usingTextUpdateResultsParagraph)
   .catch(function(err) {console.log(err);});
```

- Fetch a response from `url`
- After that's finished, call `usingResponseGetText`
- After that's finished, call `usingTextUpdateResultsParagraph`
- If an exception occurs, log the exception to the console

Agenda

- Baseline example
- JavaScript client-side web pgmming
- AJAX
- AJAX via XMLHttpRequest
- (If time) AJAX via fetch
- **AJAX wrap-up**

AJAX Wrap-Up

- **How to implement AJAX:**
  - XMLHttpRequest
    - The oldest mechanism
  - Fetch API + AbortController
    - Fetch API uses promises
  - ***jQuery***
    - jQuery library provides a function
    - Covered next lecture

AJAX Wrap-Up

	Firefox	Chrome
XMLHttpRequest	12+ (2012)	31+ (2013)
fetch	39+ (2015)	42+ (2015)
AbortController	57+ (2017)	66+ (2018)

Implication: For the next few years use XMLHttpRequest, or polyfill to XMLHttpRequest, or use jQuery

Aside: Single Page Apps

- PennyAjax app is a…
- ***Single page app (SPA)***
  - Server delivers one HTML page to the browser
  - Browser interacts with server to update parts of the page
- SPAs are very popular
- SPAs are enabled by AJAX

Summary

- We have covered:
  - Baseline example
  - JavaScript client-side web programming
  - AJAX