JSON Programming

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

Objectives

- We will cover:
  - JSON
  - JSON programming
  - XML/JSON and AJAX

Agenda

- **JSON**
- JSON programming
- XML/JSON and AJAX

JSON

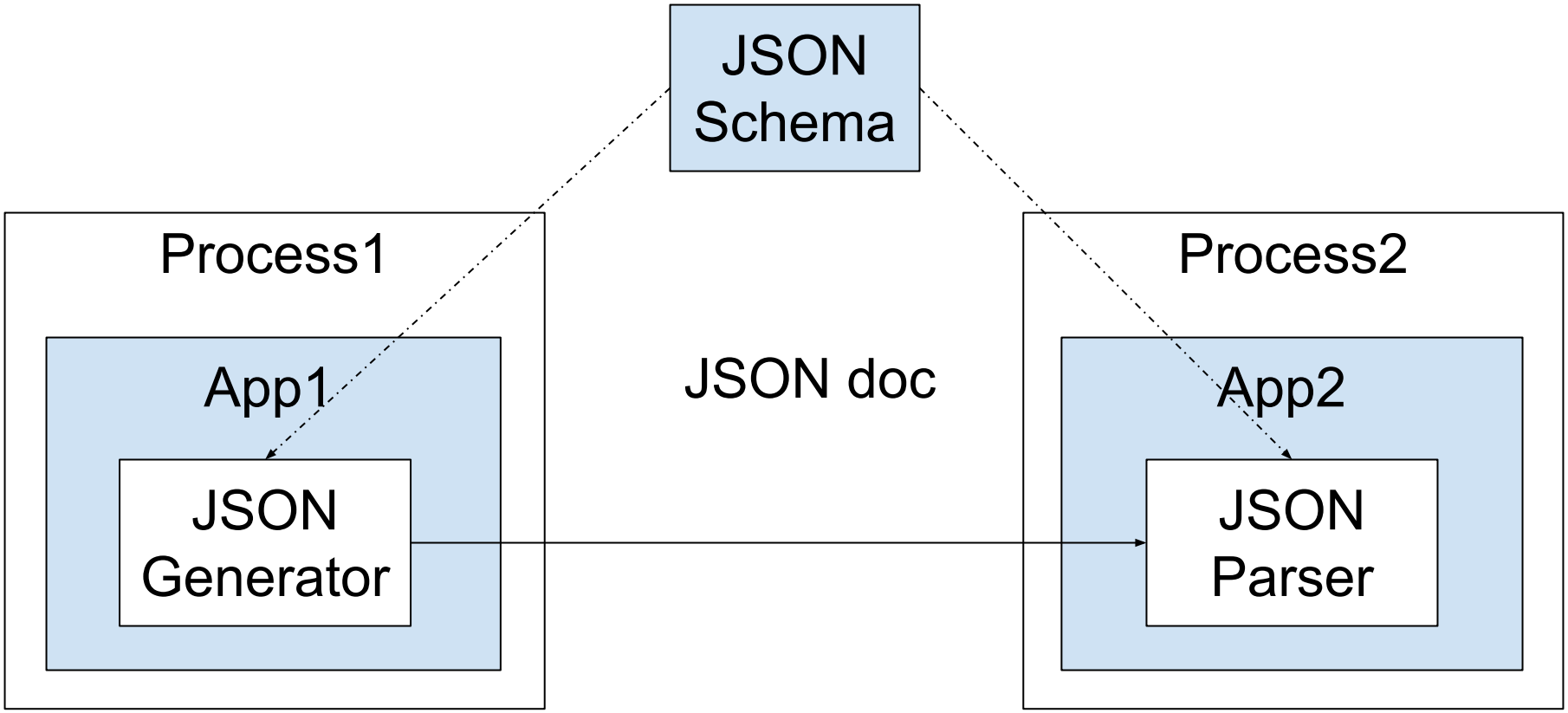
- ***JSON (JavaScript Object Notation)***
  - Like XML:
    - Textual; human readable
    - Hierarchical
  - Unlike XML:
    - Derived from JavaScript array and object notation

JSON

- ***JSON document***
  - A source code expression of a JavaScript data structure
    - JavaScript data structure can consist of:
      - Strings, Numbers, Booleans, or null
      - Objects or arrays having properties that are Strings, Numbers, Booleans, null, objects, or arrays
- See **books.json**

JSON

JSON for data comm



JSON

- Examples in this lecture:
  - In Python
    - Appropriate for JSON programming on the **server-side** of a Web app
  - In JavaScript
    - Appropriate for JSON programming on the **client-side** of a Web app (i.e., in a browser)

Agenda

- JSON
- **JSON programming**
- XML/JSON and AJAX

JSON Programming

- **writebooksjson** programs
  - Write all books in books.json

JSON Programming

- See **writebooksjson.py**

```
$ python writebooksjson.py
Author: Kernighan
Title: The Practice of Programming
Price: 40.74 dollars

Author: Kernighan
Title: The C Programming Language
Price: 24.99 dollars

Author: Sedgewick
Title: Algorithms in C
Price: 61.59 dollars

$
```

JSON	Python
array	list
object	dict
Number	int, float
String	str
Boolean	bool
null	None

JSON Programming

- See **writebooksjson.js**

```
$ node writebooksjson.js
Author: Kernighan
Title: The Practice of Programming
Price: 40.74 dollars

Author: Kernighan
Title: The C Programming Language
Price: 24.99 dollars

Author: Sedgewick
Title: Algorithms in C
Price: 61.59 dollars

$
```

JSON Programming

- **roundtrip** programs
  - Translate from JSON to data structure
  - Translate from data structure to JSON

JSON Programming

- See **roundtripjson.py**

```
$ python roundtripjson.py books.json
[{"author": "Kernighan", "title": "The
Practice of Programming", "price": 40.74,
"currency": "dollars"}, {"author":
"Kernighan", "title": "The C Programming
Language", "price": 24.99, "currency":
"dollars"}, {"author": "Sedgewick",
"title": "Algorithms in C", "price":
61.59, "currency": "dollars"}]
$
```

JSON Programming

- See **roundtripjson.js**

```
$ node roundtripjson.js
[{"author":"Kernighan","title":"The Practice of
Programming","price":40.74,"currency":"dollars"},
{"author":"Kernighan","title":"The C Programming
Language","price":24.99,"currency":"dollars"},{"a
uthor":"Sedgewick","title":"Algorithms in
C","price":61.59,"currency":"dollars"}]

$
```

JSON Programming

- XML pros
  - Appropriate for data comm and publishing
- JSON pros
  - Compact
  - Easy to handle in JavaScript
  - Easy to handle in Python, …
  - OK to handle in Java, …

Aside: Data Comm Formats

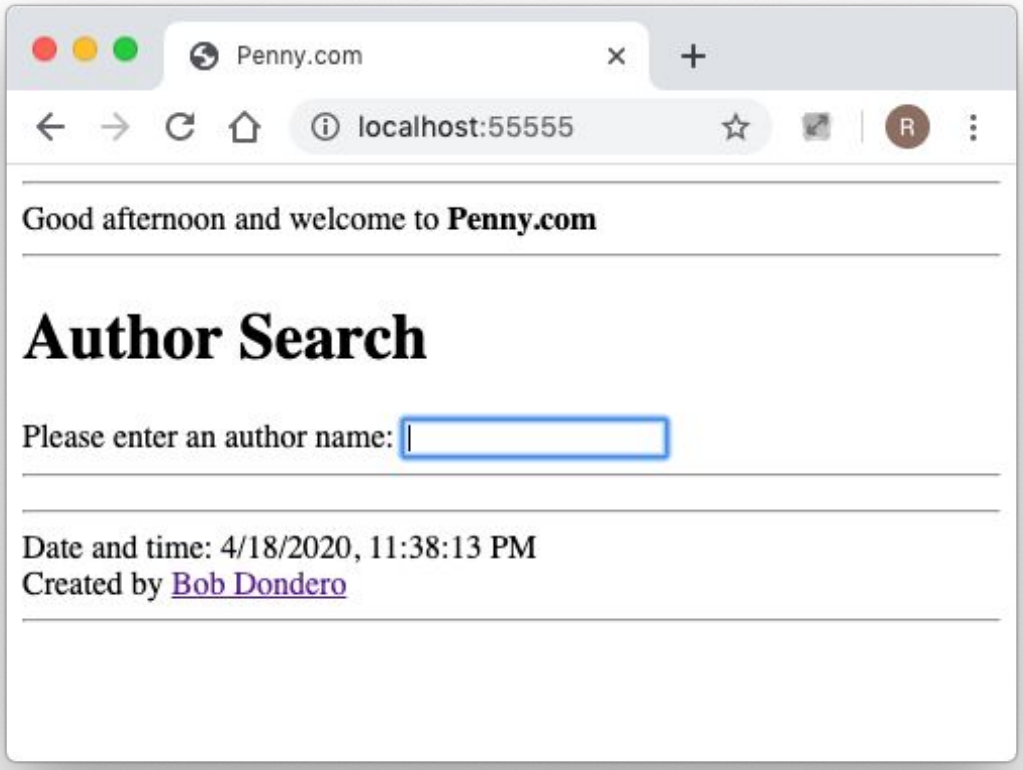
- Data comm formats
  - Binary:
    - Pickled Python objects, …
  - Human readable:
    - plain text, HTML, XML, JSON, …
- See:
  - https://en.wikipedia.org/wiki/Comparison_of_data_serialization_formats

Agenda

- JSON
- JSON programming
- **XML/JSON and AJAX**

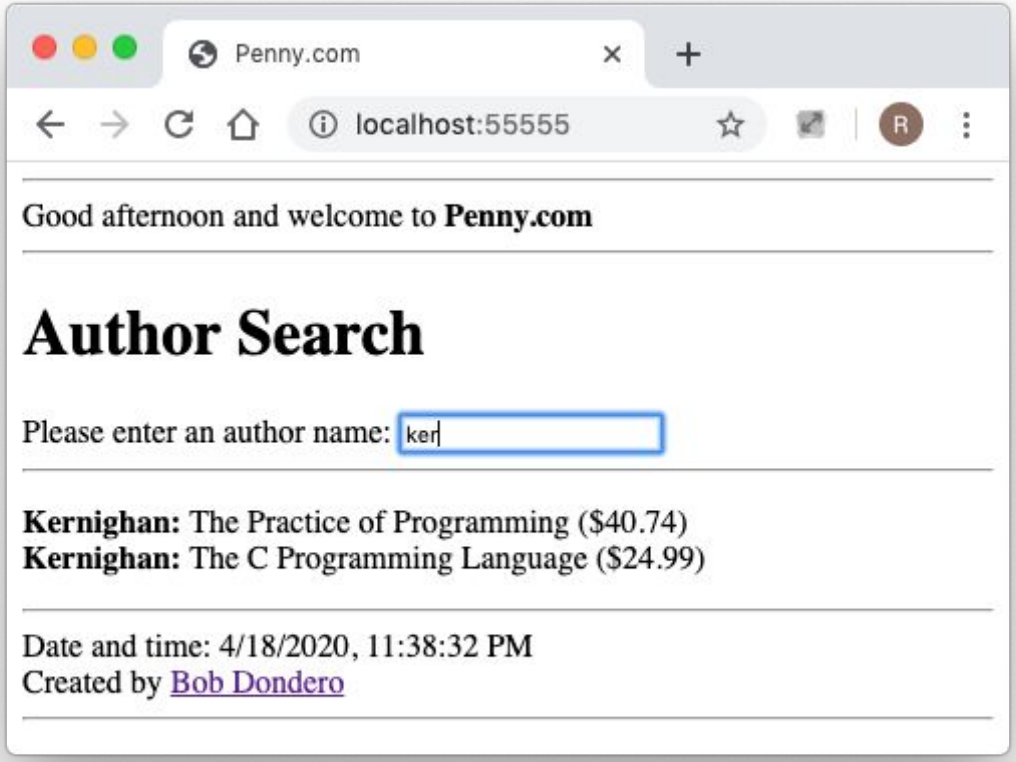
XML/JSON and AJAX

- Recall **PennyJQuery3** app



XML/JSON and AJAX

- Recall **PennyJQuery3** app



XML/JSON and AJAX

- Recall **PennyJQuery3** app
  - runserver.py
  - penny.sql, penny.sqlite
  - **book.py**, database.py
  - **books.html**
  - **penny.py**
  - **index.html**

XML/JSON and AJAX

- Observation
  - Server sends AJAX response as HTML
  - Server *could* send AJAX response as XML or JSON

XML/JSON and AJAX

- See **PennyJQueryXml** app
  - runserver.py
  - penny.sql, penny.sqlite
  - **book.py**, database.py
  - ~~books.html~~
  - **penny.py**
  - **index.html**

XML/JSON and AJAX

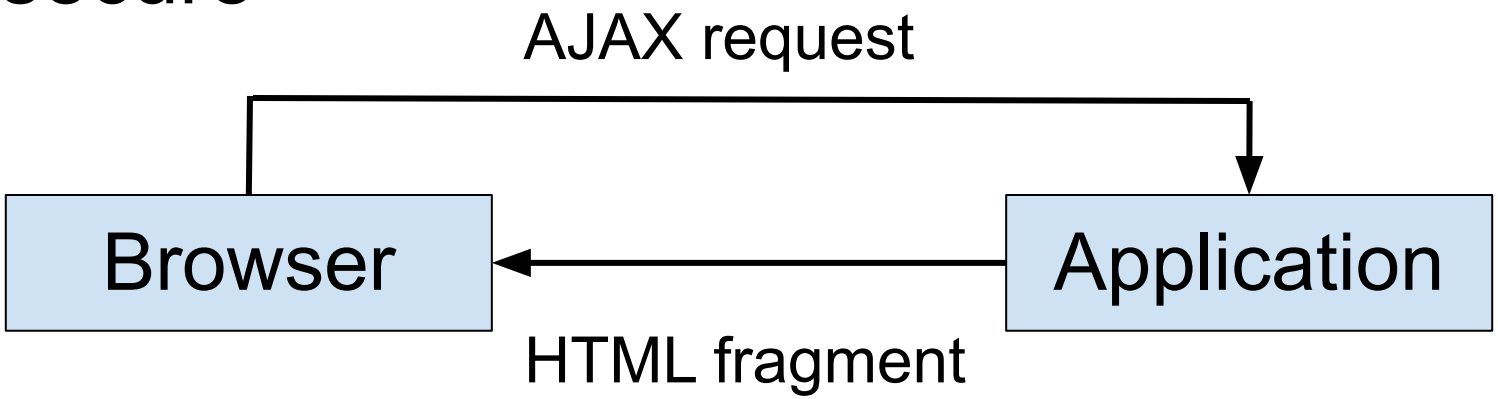
- See **PennyJQueryJson** app
  - runserver.py
  - penny.sql, penny.sqlite
  - **book.py**, database.py
  - ~~books.html~~
  - **penny.py**
  - **index.html**

Question (lecture19)

- So the server ***could*** send XML/JSON fragments instead of HTML fragments. Why ***should*** the server send XML/JSON fragments instead of HTML fragments? (No fair looking at the following slides.)
  - Browse to https://cos333attend.cs.princeton.edu to answer

XML/JSON and AJAX

- **Answer 1:** Sending XML/JSON is more secure



Browser inserts HTML fragment into DOM
Dangerous!

XML/JSON and AJAX

- **Answer 2:** Sending XML or JSON defines a convenient API
  - Example…

XML/JSON and AJAX

- See **booksbyauthorjson.py**

```
$ python booksbyauthorjson.py localhost 55555 ker
Kernighan
The Practice of Programming
40.74

Kernighan
The C Programming Language
24.99

$
```

Summary

- We have covered:
  - JSON
  - JSON programming
  - XML/JSON and AJAX
- See also:
  - **Appendix**: JSON Checking

Appendix:
JSON Checking

JSON Checking

- To run the example Python programs in this appendix:
  - `python –m pip install jsonschema`

JSON Checking: Well-Formedness

- Computer science jargon…
  - A JSON document is ***well-formed*** iff it conforms to the JSON specification

JSON Checking: Well-Formedness

- See books.json (revisited)
- See **booksmalformed.json**
- See **checkjson.py**

```
$ python checkjson.py books.json
The document is well-formed.
$ python checkjson.py booksmalformed.json
Expecting ',' delimiter: line 15 column 7 (char 353)
$
```

JSON Checking: Validity

- ***JSON Schemas***
  - Internet draft
    - Published by Internet Engineering Task Force
    - 12th draft released 2022 June 10
    - See http://json-schema.org/
    - See Wikipedia JSON page
  - A JSON schema defines whether a JSON doc is ***valid***

JSON Checking: Validity

- See **books.schema.json**
- See books.json (revisited)
- See **booksinvalid.json**
- See **checkjsonusingschema.py**

JSON Checking: Validity

```
$ python checkjsonusingschema.py books.json books.schema.json
The document is well-formed.
The document is valid.
$ python checkjsonusingschema.py booksinvalid.json books.schema.json
The document is well-formed.
'currency' is a required property

Failed validating 'required' in schema['items']:
    {'properties': {'author': {'type': 'string'},
                    'currency': {'type': 'string'},
                    'price': {'type': 'number'},
                    'title': {'type': 'string'}},
     'required': ['author', 'title', 'price', 'currency'],
     'type': 'object'}

On instance[2]:
    {'author': 'Sedgewick', 'price': 61.59, 'title': 'Algorithms in
C'}
$
```