COMP20008 Elements of Data Processing

Semester 2 2018

Lecture 3: Data formats: structured, unstructured and semi-structured: continued

Announcements

- Anam Khan (**Head Tutor**)
  - Email: anamk@student.unimelb.edu.au
  - Contact if you have any issue with the tutorials!
- If you have a workshop scheduled Thursday 6:15pm, please check notice on the LMS regarding time change!
- Project due dates (reminder)
  - Phase 1: Python data wrangling warmup exercises (20%)
    - Due: 31st August (release 13th August)
  - Phase 2: Python data wrangling exercises (15%)
    - Due: 21st September (release 3rd September)
  - Phase 3: Data wrangling investigation on an open dataset (will be flexible in what to use) – Code and Oral (15%)
    - Due(code): 5th October (release 10th September)
    - Due (oral): will be held in your workshop class in Week 11 (8-14 October)

Today's lecture

- First finish slides from lecture 2 (XML)
  - namespaces
- Then look at another data format - JSON

XML

```
<?xml version=1.0" encoding="UTF-8"?>
<catalog>
 <book price = "55" currency = "USD">
    <title> Foundations of Databases </title>
    <author> Abiteboul </  author>
    <date>
      <year>1995</year>
      <month>January</month>
    </date>
 </book>
</catalog>
```

•book, catalog, title, author, date, year, month are elements

•price is an attribute (provides further information about an element, in this case the book element).

•currency is an attribute.

XML applications

- Mathematical Markup Language (MathML)
- ChemML (Chemical Markup Language)
- RSS, SOAP, SVG, …

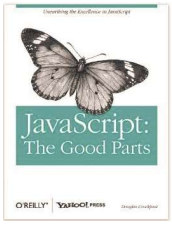
MathML Example: markup an equation in terms of presentation and semantics

In MathML, x^3+6x+6 is represented as

```
<mrow>
  <msup>
    <mi>x</mi> <mn>3</mn>
  </msup>
  <mo>+</mo>
  <mrow>
    <mn>6</mn> <mo>&InvisibleTimes;</mo> <mi>x</mi>
  </mrow>
  <mo>+</mo>
  <mn>6</mn>
</mrow>
```

JavaScript Object Notation (JSON)

- JSON (www.json.org)
- Douglas Crockford (pretty much alone)
  - c.f the development of XML by committee
- *"Javascript: the good parts"*
  - O' Reilly, Yahoo Press



JSON: JavaScript Object Notation

```
{
    "Catalog": [
        { "CD": {
            "title":  "Empire Burlesque",
            "artist": "Bon Dylan",
            "Country": "USA",
            "price": {
              "Currency": "USD",
              "value": 10.90
                  },
            "year": 1985
            }
        },
        { "CD": {
            "title": "Hide your heart",
            "artist": "Bonnie Taylor",
            "Country": "UK",
            "price": {
              "currency": "USD",
              "value": 9.90
                  },
            "year": 1988
            }
        }
    ]
}
```

```
<CATALOG>
  <CD>
  <TITLE>Empire Burlesque</TITLE>
  <ARTIST>Bob Dylan</ARTIST>
  <COUNTRY>USA</COUNTRY>
  <COMPANY>Columbia</COMPANY>
  <PRICE CURRENCY="USD"> 10.90</PRICE>
  <YEAR>1985</YEAR>
  </CD>

  <CD>
  <TITLE>Hide your heart</TITLE>
  <ARTIST>Bonnie Tyler</ARTIST>
  <COUNTRY>UK</COUNTRY>
  <COMPANY>CBS Records</COMPANY>
  <PRICE CURRENCY="USD">9.90</PRICE>
  <YEAR>1988</YEAR>
  </CD>
</CATALOG>
```

JSON compared to XML

- JSON is simpler and more compact/lightweight than XML. Easy to parse.
- Common JSON application – read and display data from a webserver using javascript.
  - https://www.w3schools.com/js/js_json.asp
- XML comes with a large family of other standards for querying and transforming (XQuery, XML Schema, XPATH, XSLT, namespaces, …)

JSON object Example (this slide and next slide)

```
{
  "firstName": "David",
  "lastName": "Lynn",
  "isAlive": true,
  "age": 25,
  "height_cm": 167.6,
  "address": {
      "streetAddress": "211 Fox Street",
      "city": "Greenville",
      "state": "NH",
      "postalCode": "80021"
 },
```

JSON Example (cont.)

```
"phoneNumbers": [
      {
       "type": "home",
       "number": "315 555-1812"
      },
     {
       "type": "office",
       "number": "646 555-4567"
     }
],
 "email": "dlynn@nhs.net"
}
```

XML Representation

```
<?xml version="1.0"?>
<customers>
        <customer>
                <firstname>David</firstname>
                <lastname>Lynn</lastname>
                …..
                <address>
                   <staddress>211 Fox Street</staddress>
                   <city>Greenville</city>
                …..
                </address>
                ….
                <email>dlynn@nhs.net</email>
        </customer>
</customers>
```

JSON syntax rules

- Object data is in name/value pairs
  `"firstName":"John"`
- JSON values
  - –A number (integer or floating point)
  - –A string (in double quotes)
  - –A Boolean (true or false)
  - –An array (in square brackets)
  - –An object (in curly braces)
  - –null

JSON syntax rules

- JSON Objects
  {"firstName":"John", "lastName":"Doe"}
- JSON Arrays
  "employees":[
  {"firstName":"John", "lastName":"Doe"},
  {"firstName":"Anna", "lastName":"Smith"},
  {"firstName":"Peter", "lastName":"Jones"}
  ]
- These objects repeat recursively down a hierarchy as needed.
- **In terms of syntax that's pretty much it!**

Example from Twitter

- https://developer.twitter.com/en/docs/tweets/data-dictionary/overview/intro-to-tweet-json

Using JSON (Python): Load JSON format to Python Dictionary and convert to JSON format from Python Dictionary

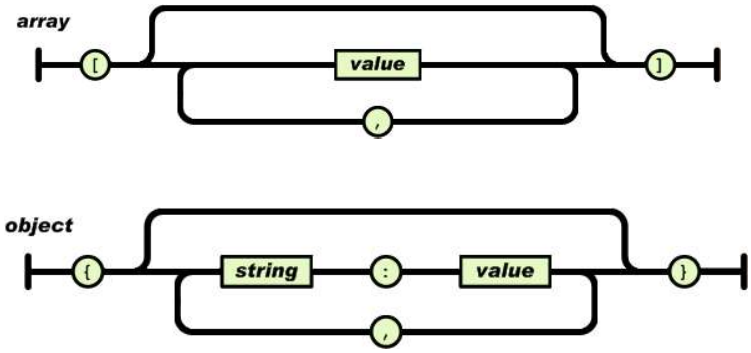
```
import json

json.loads(
    '["foo",
        {"bar":
            ["baz", null, 1.0, 2]
        }
    ]')

json.dumps(
    ['foo',
        {'bar':
            ('baz', None, 1.0, 2)
        }
    ]
)
```

Note: white space and indentation is for display purposes only!

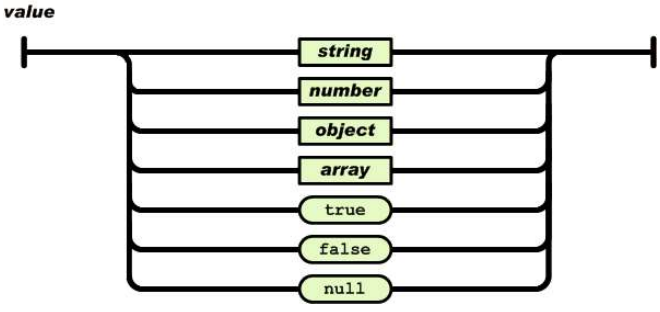
JSON vs XML cont.

- XML allows complex schema definitions (via regular expressions)
  - allows formal validation
  - makes you consider the data design more closely
- JSON is more **streamlined, lightweight and compressed**
  - Which appeals to programmers looking for speed and efficiency
  - Widely used for storing data in noSQL databases

Jason format (from json.org)



JASON format (json.org)



Exercise

- *Represent the following information in JSON*

```
<Person>
  <FirstName>Homer</FirstName>
  <LastName>Simpson</LastName>
  <Relatives>
      <Relative>Grandpa</Relative>
     <Relative>Marge</Relative>
     <Relative>Lisa</Relative>
      <Relative>Bart</Relative>
  </Relatives>
  <FavouriteBeer>Duff</FavouriteBeer>
</Person>
```

Check it is well formed
-http://jsonlint.com

JSON: Summary

- JavaScript Object Notation
- Lightweight, streamlined, standard method of data exchange
- Originally designed to speed up client/server interactions:
  - By running in the client browser
- Native Javascript, so can be executed as code
- Can be used to represent any kind of semi structured data
- Lacks context and schema definitions

JSON Schema

- Written in JSON itself
- Describes the structure of other data
- Easy to validate a JSON document against its schema using a schema validator
  - E.g. http://jsonschemalint.com/draft4/

JSON: JavaScript Object Notation

```
{
    "Catalog": [
        { "CD": {
            "title":  "Empire Burlesque",
            "artist": "Bon Dylan",
            "Country": "USA",
            "price": {
              "Currency": "USD",
              "value": 10.90
                  },
            "year": 1985
             }
        },
        { "CD":  {
            "title": "Hide your heart",
            "artist": "Bonnie Taylor",
            "Country": "UK",
            "price": {
              "currency": "USD",
              "value": 9.90
                  },
            "year": 1988
             }
        }
    ]
}
```

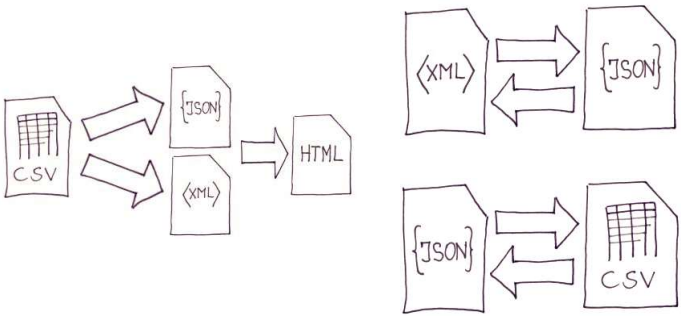
```
<CATALOG>
  <CD>
  <TITLE>Empire Burlesque</TITLE>
  <ARTIST>Bob Dylan</ARTIST>
  <COUNTRY>USA</COUNTRY>
  <COMPANY>Columbia</COMPANY>
  <PRICE CURRENCY="USD"> 10.90</PRICE>
  <YEAR>1985</YEAR>
  </CD>

  <CD>
  <TITLE>Hide your heart</TITLE>
  <ARTIST>Bonnie Tyler</ARTIST>
  <COUNTRY>UK</COUNTRY>
  <COMPANY>CBS Records</COMPANY>
  <PRICE CURRENCY="USD">9.90</PRICE>
  <YEAR>1988</YEAR>
  </CD>
</CATALOG>
```

JSON Schema Example [http://json-schema.org/examples.html]

```
{
    "type" : "object",
    "properties" : {
        "Catalog" : {
            "type" : "array",
            "items" :{
                "type" : "object",
                "properties" : {
                    "title": { "type" : "number" },
                    "artist": { "type" : "string" },
                    "Country": { "type" : "string" },
                     "price": {"type": "object",
                          "properties":
                                {"currency": {type: "number"},
                                "value": {type:"number"}
                                }
                        }
                        }
                }
            }
        }
}
```

Python libraries for JSON and XML

- json
- lxml



XML, JSON, CSV and HTML conversion tools

What you should know about data formats

- Why do we have different data formats and why do we wish to transform between different formats?
- Motivation for using relational databases to manage information
- What is a csv, what is a spreadsheet, what is the difference?
- Be able to read and write regular expressions in python format (operators .^$*+|[]())
- Difference between HTML and XML and when to use each
- Motivation behind using XML and XML namespaces
- Be able to read and write data in XML (elements, attributes, namespaces)
- Be able to read and write data in JSON
- Difference between XML and JSON. Applications where each can be used.
- The purpose of using schemas for XML and JSON data.

Further reading

- Further reading
  - Relational databases
    - Pages 403-409 of http://i.stanford.edu/~ullman/focs/ch08.pdf
  - XML: a gentle introduction
    - http://www.tei-c.org/release/doc/tei-p5-doc/en/html/SG.html
  - JSON
    - http://json.org