COMP20008 Elements of Data Processing

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

Announcements

- Project due dates (estimated)
  - Phase 1: Python exercises. (15%)
    - Due: 6th April
  - Phase 2: Concept formulation and initial investigation (12%)
    - Due: 25th April
  - Phase 3: Report (13%)
    - Due: 10th May
  - Phase 4: Oral (10%)
    - Due: will be held in your workshop class in Week 11 (15-19 May)
- Workshop this week
  - Available on LMS, have been some minor modifications, please download again before your class
- Today
  - Finish off last few slides on XML from last lecture
  - Look at JSON (another popular data exchange format)

XML: reminder

```
<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, …

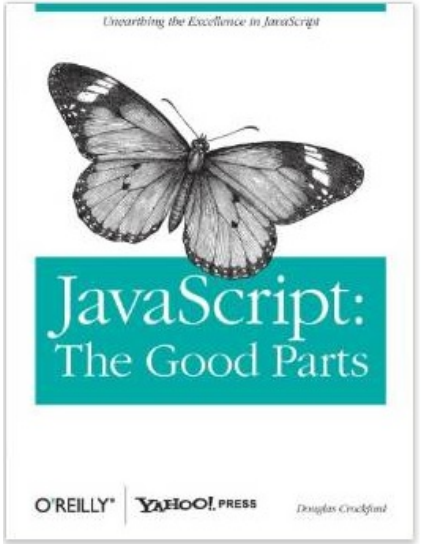
MathML Example: markup an equation in terms of presentation and semantics

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, …)

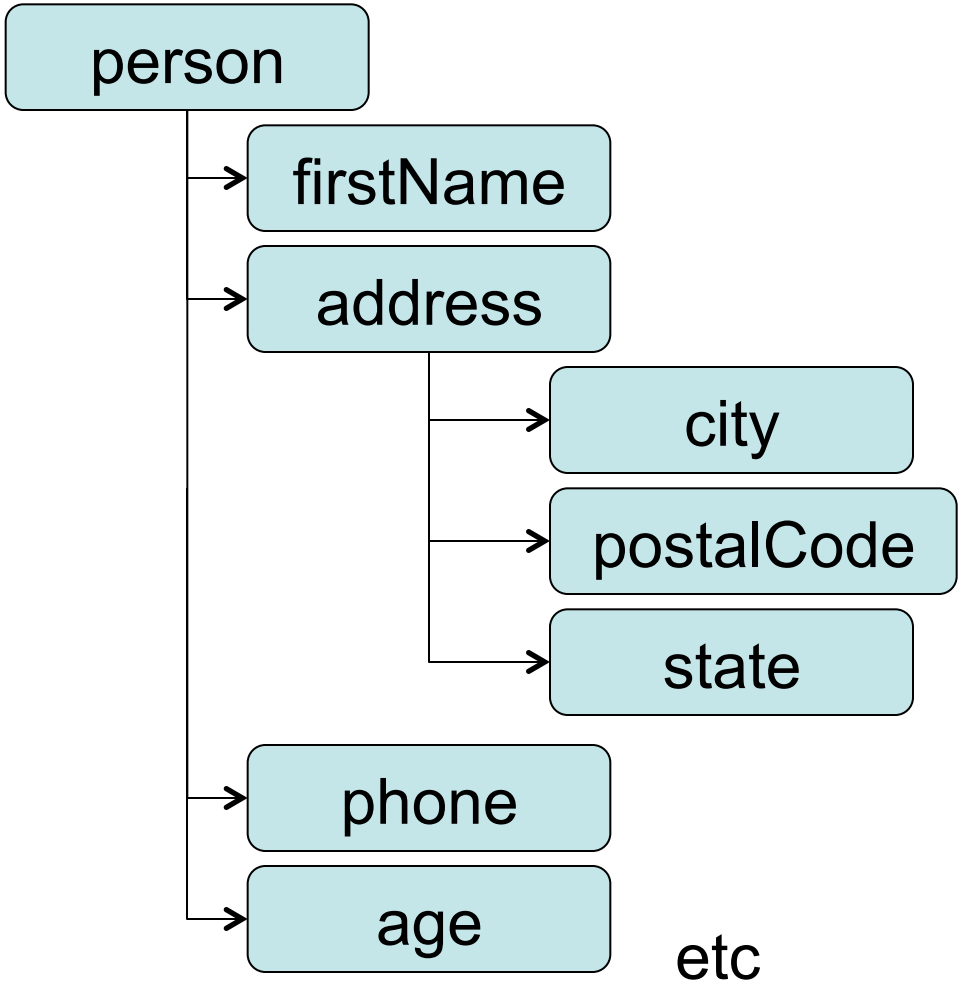
JSON Example (this slide and next slide)

```
{
  "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"
}
```

Diagram of a JSON object



XML Representation

```
<?xml version="1.0"?>
<customers>
        <customer>
                <name>David Lynn</name>
                <address>211 Fox Street Greenville, NH 80021</address>
                <phone>(315) 555-1812</phone>
                <email>dlynn@nhs.net</email>
        </customer>
</customers>
```

JSON syntax rules

- 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

- http://jsoneditoronline.org

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

```
import json

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

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

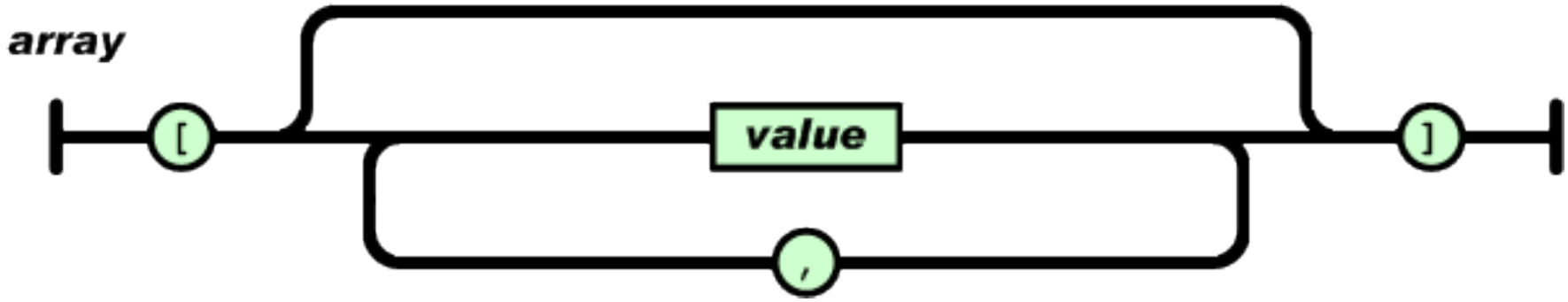
Note: white space and indentation is for display purposes only!

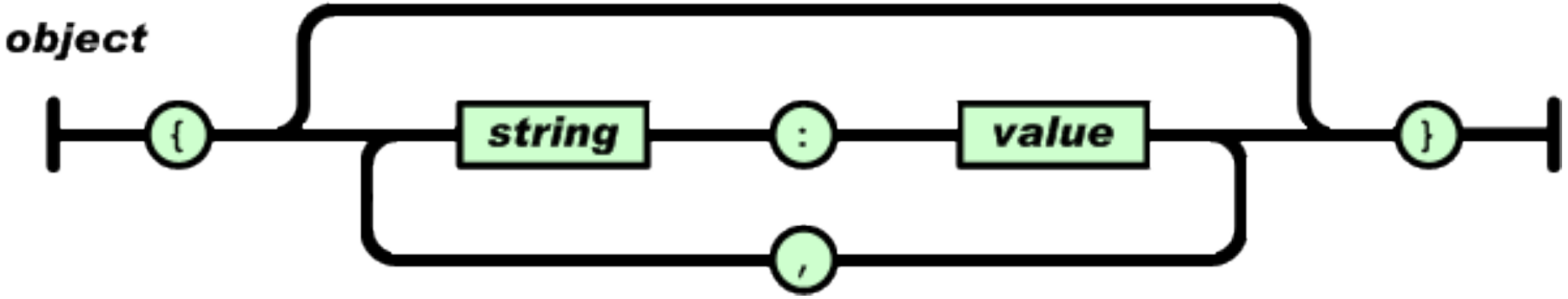
JSON vs XML cont.

- XML allows complex schema definitions (via regular expressions)
  - allows formal validation
  - makes you consider the data design more closely

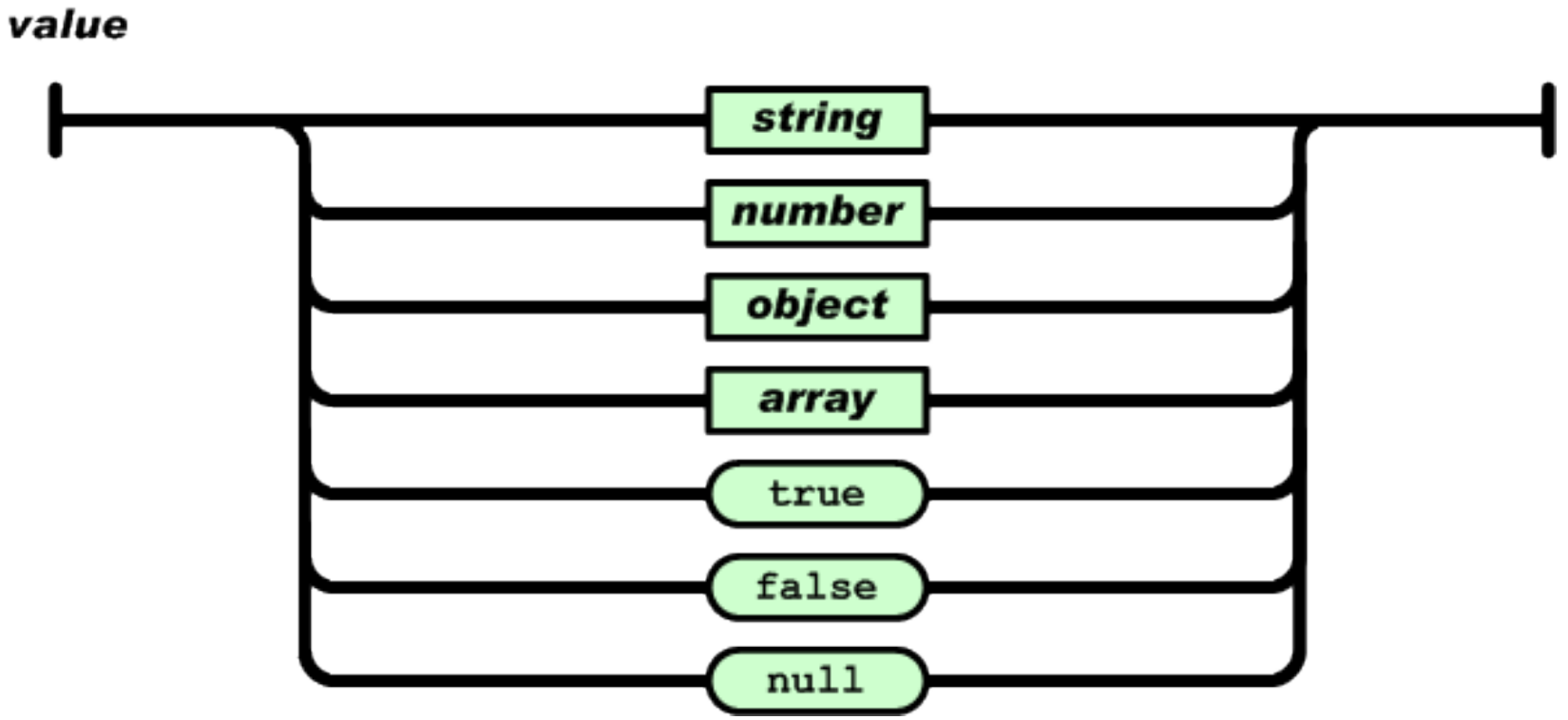
- JSON is more **streamlined, lightweight and compressed**
  - Which appeals to programmers looking for speed and efficiency
  - Widely used for storing data in noSQL databases (we will come back to this later, in the distributed/cloud topic)

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>
```

JSON: Summary

- JavaScript Object Notation
- Lightweight, streamlined, standard method of data exchange
- Designed to speed up client/server interactions:
  - By running in the client browser
- Native Javascript, so can be executed as code
- Lacks context and schema definitions
- Integral to the Big Data paradigm (NoSQL)

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

- json
- ElementTree
- html.parser

JSON Application: Linked Data

- We need to connect data together --- form links.
  - A key part of the *Semantic Web*
  - Also important for the *Internet of Things*
    - (26 billion things by 2020, each continuously producing data)

1. Principles of links from Tim Berners-Lee
    1. All kinds of conceptual things, they have names now that start with HTTP.
    2. If I take one of these HTTP names and I look it up, I will get back some data in a standard format which is kind of useful data that somebody might like to know about that thing, about that event.
    3. When I get back that information it's not just got somebody's height and weight and when they were born, it's got relationships. And when it has relationships, whenever it expresses a relationship then the other thing that it's related to is given one of those names that starts with HTTP.

Standards for Linked Data

- Widely used standards (W3C Recommendations)
  - JSON-LD (JSON Linked Data)
  - RDF (Resource Description Framework)

Linked Data Examples

- Websites
  - https://test-5791.myshopify.com/products/example-t-shirt

- Google Knowledge Graph
  - https://developers.google.com/search/docs/guides/intro-structured-data
  - https://developers.google.com/apis-explorer/#p/kgsearch/v1/kgsearch.entities.search

JSON-LD (example from json-ld.org)

- Provide mechanisms for specifying unambiguous meaning in JSON data
- Provides extra keys with “@” sign
  - “@context” (used to define meanings of terms, map to identifiers)
  - “@type”
  - “@id”

JSON-LD Example (from https://en.wikipedia.org/wiki/JSON-LD)

```
{"@context": {
    "name": "http://xmlns.com/foaf/0.1/name",
    "homepage": {
        "@id": "http://xmlns.com/foaf/0.1/workplaceHomepage",
        "@type": "@id"
    },
    "Person": "http://xmlns.com/foaf/0.1/Person"
  },
  "@id": "http://me.example.com",
  "@type": "Person",
  "name": "John Smith",
  "homepage": "http://www.example.com/"
}
```

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.
- -The motivation behind Linked Data and the purpose of using JSON-LD to represent it.

Further reading

- Further reading
  - Relational databases
    - Pages 403-409 of http://i.stanford.edu/~ullman/focs/ch08.pdf
  - XML
    - http://www.tei-c.org/release/doc/tei-p5-doc/en/html/SG.html
  - JSON and JSON-LD
    - http://json.org
    - https://cloudant.com/blog/webizing-your-database-with-linked-data-in-json-ld/#.Vtp_UMfB_Gw
    - http://searchengineland.com/demystifying-knowledge-graph-201976