Spring 2024 CSE 380 (491-002)

1-11-2024

History of Databases

- Lots of stuff happened in the past
  - It's not important
  - I don't have any cool stories about "the good old days of computing"
- We (mostly) use relational databases now
  - That is what we'll focus on for the most part this semester
- NoSQL databases are growing in popularity
  - Out of scope for this course (take CSE 480 if you're interested)
  - Hot Take: You most likely do not need/want a NoSQL DB
  - Reasons (most) people use NoSQL
    - They're too lazy to structure their data
    - They're too broke and use free NoSQL DBs
    - They've drank the MongoDB Kool-Aid (don't listen to the propaganda!)
    - They've listened to someone trying to justify their bad decisions
      - Imagine you became a professor after I tried to convince you it's a great idea
  - NoSQL is like blockchain, very crucial for certain things, not applicable to most things

Data Models

- Data Models – Components
  - Mathematical representation of the data
    - Relational Models (CSV) = tables
    - Semi-structured Models (XML, JSON) = trees/graphs
  - Operations on data (what is allowed, i.e. comparisons, equality, math)
  - Constraints (what types of data are allowed, and where)

An Aside

- Difference Between a Database Schema and a Database Instance?
  - A database instance is the actual data that is in a database (a specific database)
  - A database schema is what sort of data *CAN* be in a database
    - Data types, where they can be, how many are allowed, etc.
- We'll Discuss Database Schemas a lot this Semester
  - But XML, and JSON can also have schemas

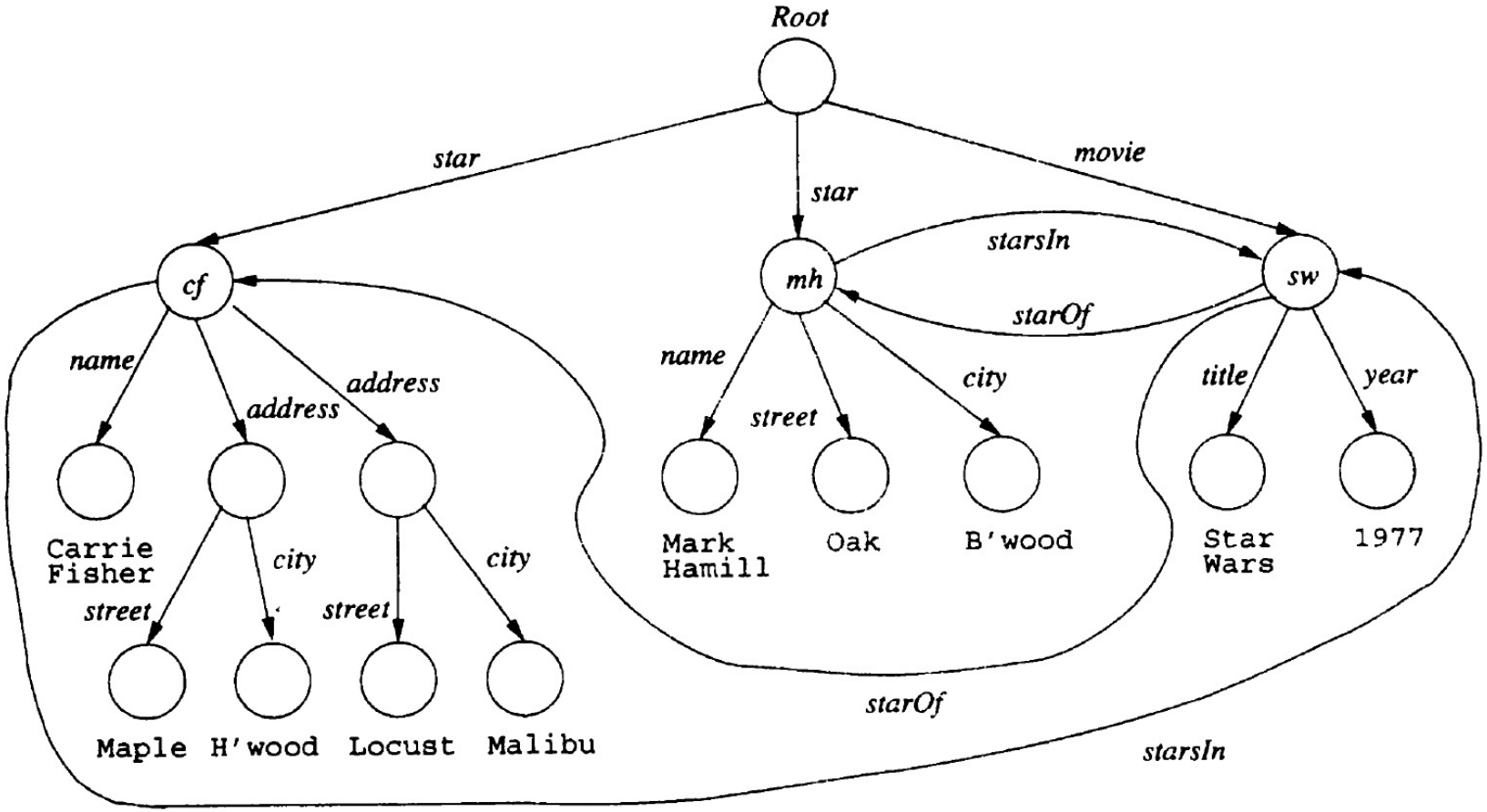
Relational vs Semi-structured

- Relational:
  - Rigid schema
  - Efficient implementation
- Semi-structured:
  - Motivated by flexibility
  - Data is “schemaless” (self-describing)
    - Although some types of schemas can exist

Semi-structured Data

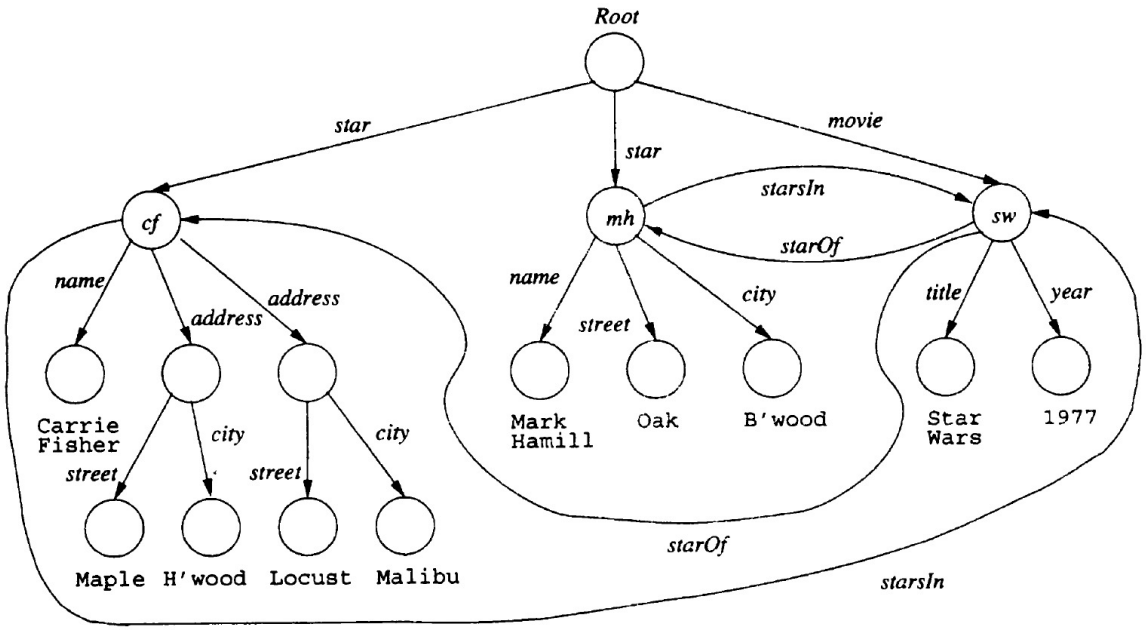
- Nodes = objects
- Arc labels (properties of objects)
- Atomic values at leaf nodes (nodes with no arcs out)
- Flexibility: no restriction on:
  - Labels out of a node
  - Number of successors with a given label

Example Semi-structured Data



Example Semi-structured Data

- Note the inconsistency between the two ”star” objects
  - Also note how they handle addresses differently
- Arcs represent attributes
  - See off cf we have an arc ”name” (the name of the star represented by the object)
- Arcs between separate nodes(starsIn, starOf) represent relationships between objects



What's the best way for me to make it clear which slides have practice questions?

Example Questions (for you to solve)

- Add the director George Lucas and producer Gary Kurtz to this graph representation
- Add information about Empire Strikes Back and Return of the Jedi, including the facts the both Carrie Fisher and Mark Hamill appeared in these movies
- Add information about the movie studios (Fox) for the films, and the address of the studio (Hollywood)

Common (Semi-structured) File Formats Useful for Databases

- Data used in databases come in many forms
  - Need to read data into the database
  - Save data in a format for another application
  - In general good to know
- Two main data types we'll look at
  - eXtensible Markup Language (XML)
  - JavaScript Object Notation (JSON)

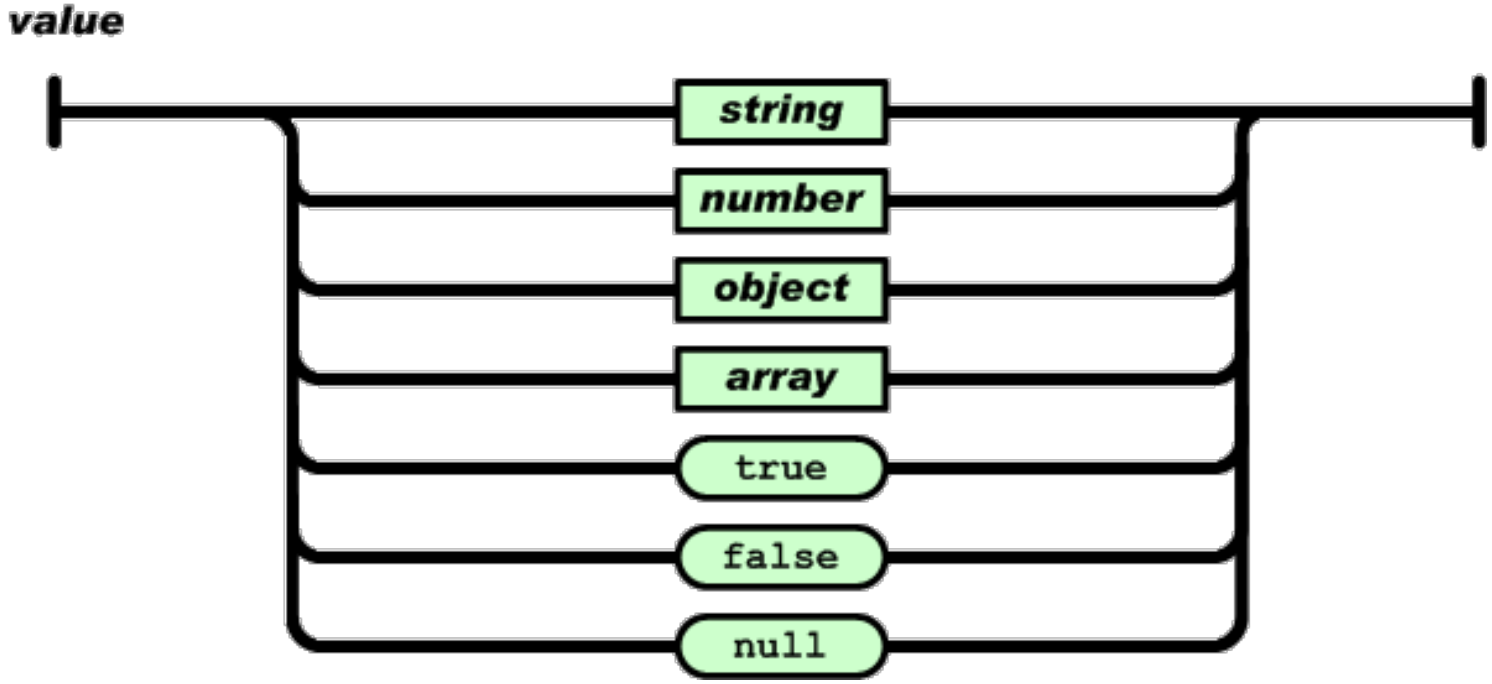
Well-formed vs Valid

- Well-formed:
  - Complies with all constraints of the file format (XML, JSON)
- Valid:
  - Is well-formed, and also follows a ”schema”

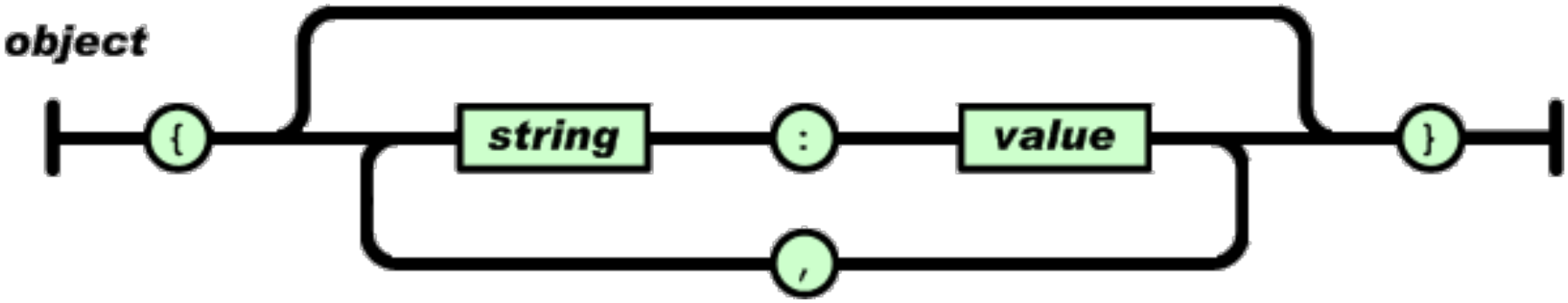
JSON Files

- Plain text file format
  - Note, not a flat file structure like CSV
- Originally derived from JavaScript, but is now a language-independent data format
- Whitespace (outside of strings) is meaningless
- One (root) JSON object allowed per file, but there are ways to store many different things in one JSON object

JSON Types

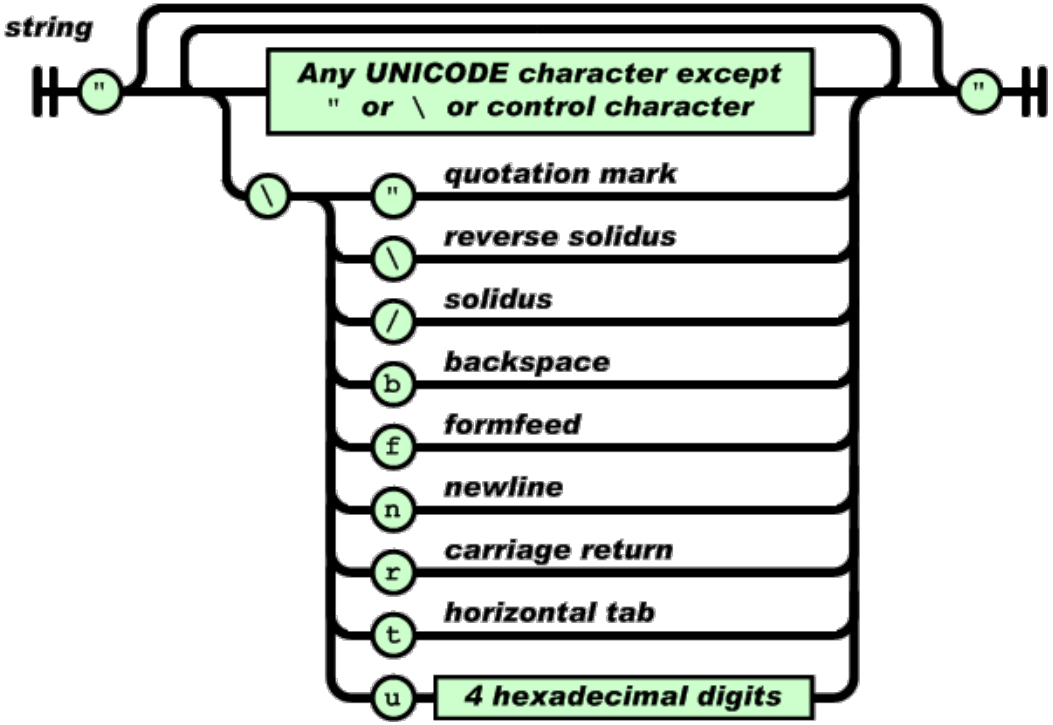


JSON Objects



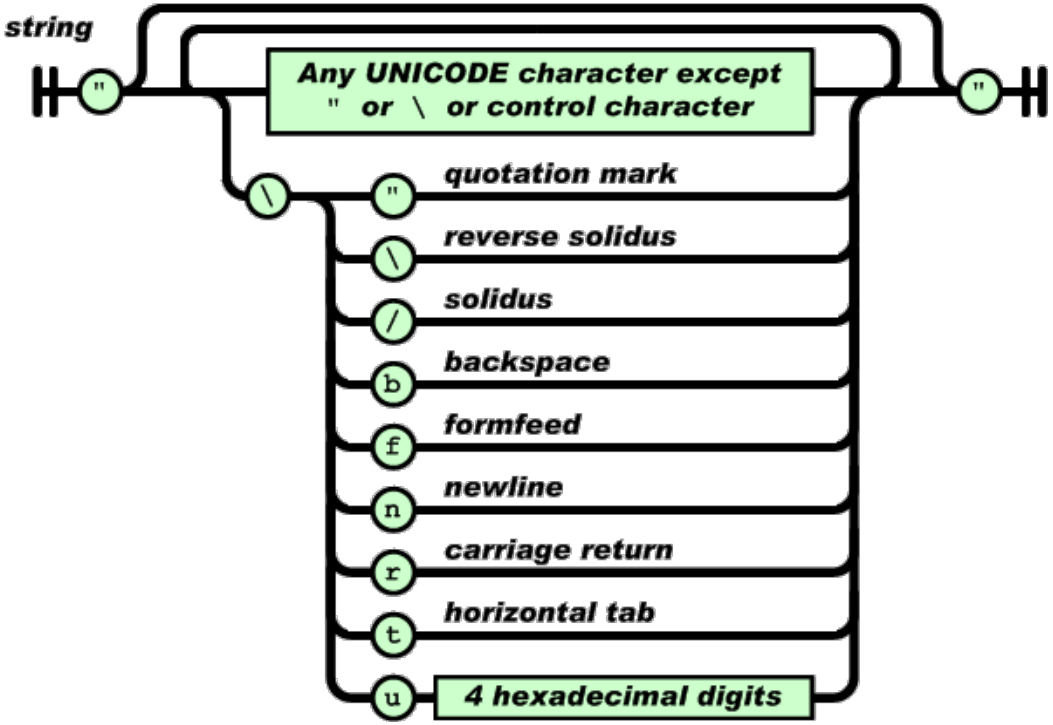
- Object is an unordered collection of name/value pairs, where the names (aka keys) are strings
  - { "name": "James", "age": "don't' worry about it", "classes": ["CSE380", "CSE498"]}

JSON Strings



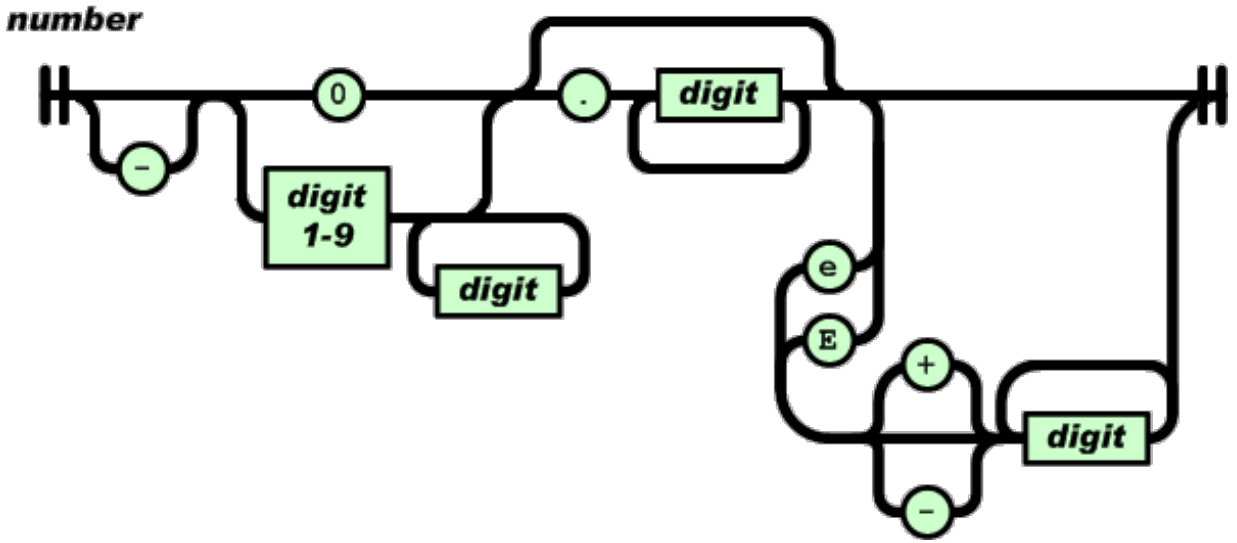
- A sequence of zero or more "characters"
- Strings are deliminated with double quotations marks and support backslash escaping
  - "Hi\nEveryone"

JSON Strings



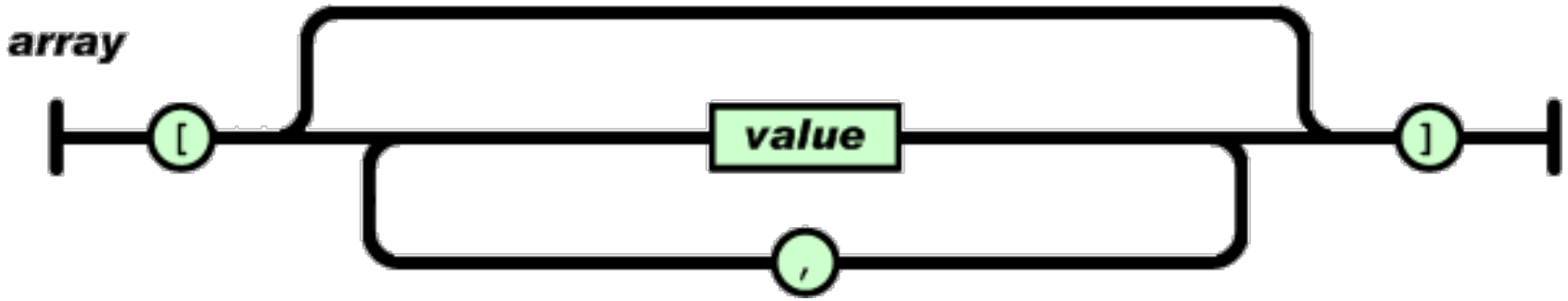
- Which are valid JSON Strings?
  - James
  - "James\n"
  - "James_Project_1"
  - ""

JSON Numbers



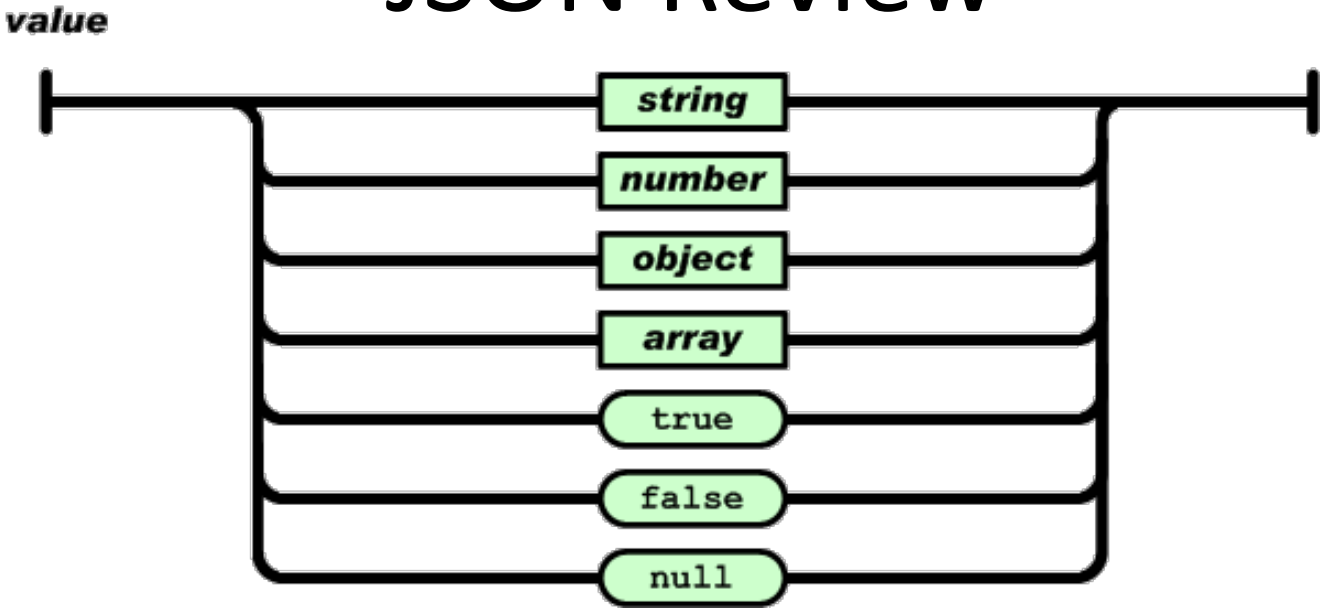
- A signed decimal number that may contain a fractional part and may use exponential E notation
  - 4
  - 12.5
  - 5.0e10
  - -5.6

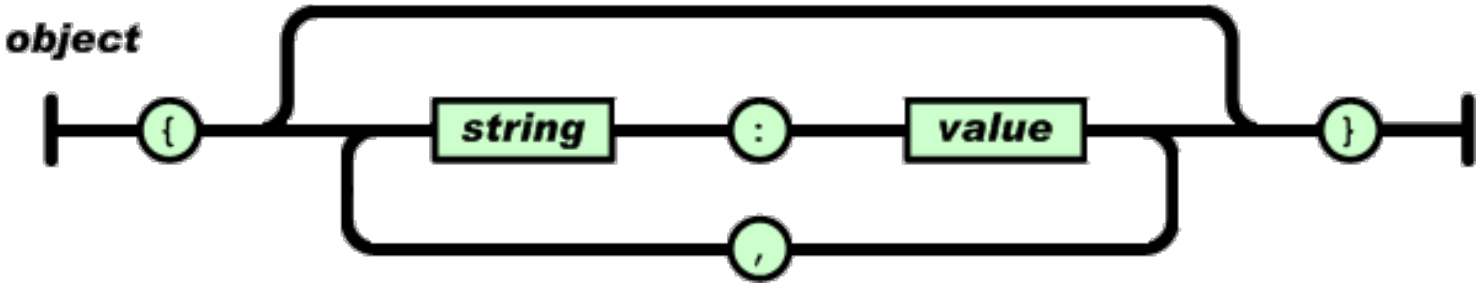
JSON Array



- An ordered list of zero or more values, each of which may be of any type
  - []
  - [4, "James", true]
  - [[] , null]
  - [[1,2,3,4], {"team": "USA"}]

JSON Review


value
string
number
object
array
true
false
null


object
{
string
:
value
}
,

Example JSON File

```
{
  "name": "James",
  "classes": [ {"Name": "CSE380", "Students": 48, "Topic": "Databases and Cloud"},
               {"Name": "CSE498", "Students": 158, "Topic": "Capstone"} ],
  "Employed": true
}
```

JSON Schema

- A schema is a definition of how the data should be formatted
- Schema can be used for validation
- JSON Schema is widely used, but still evolving
- Widely used standard: http://json-schema.org

Example JSON Schema

Example JSON File

```
{
        "id": 1,
        "name": "frozen pizza",
        "price": 10.50,
        "tags": [ "frozen", "pizza"]
}
```

- Schema can help us define:
  - What is the data?
  - What is id?
  - Is name required?
  - Can price be 0?
  - Are all tags strings?

Example JSON Schema

JSON Schema File

```
{
        "title": "Product",
        "description": "Product from supermarket",
        "type": "object
}
```

Example JSON File

```
{
        "id": 1,
        "name": "frozen pizza",
        "price": 10.50,
        "tags": [ "frozen", "pizza"]
}
```

Example JSON Schema

JSON Schema File

```
{
        "title": "Product",
        "description": "Product from supermarket",
        "type": "object,
        "properties": {
                "id": {
                        "description": "Unique id",
                        "type": "integer" } },
        "required": ["id"]
}
```

Example JSON File

```
{
        "id": 1,
        "name": "frozen pizza",
        "price": 10.50,
        "tags": [ "frozen", "pizza"]
}
```

Example JSON Schema

Example JSON File

```
{
        "id": 1,
        "name": "frozen pizza",
        "price": 10.50,
        "tags": [ "frozen", "pizza"]
}
```

JSON Schema File

```
{
        "title": "Product",
        "description": "Product from supermarket",
        "type": "object,
        "properties": {
                "id": {
                        "description": "Unique id",
                        "type": "integer" } ,
                "name": {
                        "description": "item name",
                        "type": "string"} },
        "required": ["id", "name"]

}
```

Example JSON Schema

Example JSON File

```
{
        "id": 1,
        "name": "frozen pizza",
        "price": 10.50,
        "tags": [ "frozen", "pizza"]
}
```

JSON Schema File

```
{
        "title": "Product",
        "description": "Product from supermarket",
        "type": "object,
        "properties": {
                "id": {
                        "description": "Unique id",
                        "type": "integer" } ,
                "name": {
                        "description": "item name",
                        "type": "string"} ,
                "price": {
                        "type": "number",
                        "minimum": 0 } },
        "required": ["id", "name", "price"]

}
```

Example JSON Schema

Are JSON Schema required to interpret JSON data?

NO!

However, with a schema, we might have well-formed JSON that is not valid!

JSON Schema File

```
{
    "title": "Product",
    "description": "Product from supermarket",
    "type": "object,
    "properties": {
        "id": {
            "description": "Unique id",
            "type": "integer" } ,
        "name": {
            "description": "item name",
            "type": "string"} ,
        "price": {
            "type": "number",
            "minimum": 0 } ,
        "tags" : {
            "type": "array",
            "items": {
                "type": "string"},
            "minItems": 1 } },
    "required": ["id", "name", "price"]
}
```

XML Files

- XML is also (similar to JSON) meant to convey semi-structured data
- HTML is a subset of XML
- In general used heaving in web services and inter-database communications
- Often used for text document formatting
  - Microsoft office, OpenOffice, etc.
- Often used in configuration files

XML Tags and Elements

- Tags come in 3 types:
  - Start-tags (e.g. <section>)
  - End-tags (e.g. </section>)
  - Empty-element tags (e.g. <section />
- An element consists of a start-tag, optional content, and a matching end-tag. Or, and element is just an empty-element tag

- Example Elements
  - <head> </head>
  - <h> Hello <p /> </h>
  - <h />

XML Attributes

- XML attributes are name/value pairs within a start-tag or empty-element tag
- There can be only one value in each pair, so multiple values are sometimes combined (often as a space-delimited string)

- Example Attributes
  - <person name="James"> </person>
  - <location office = "3145EB" />

Well-formed XML

- The document contains only legal Unicode characters (properly escaped)
- Begin, end, and empty-element tags are correctly nested
- Element tags are case sensitive (must be exact match)
- Tag names cannot contain special characters
- There's a single "root" element that contains all other elements

XML Schemas

- Schemas describe structure and limitations of XML documents
- Schemas are written in XML, and will often have schemas of their own
- Provide basic set of types
  - Integer, byte, string, floating-point numbers
  - Can be combined into complex types to define elements
- http://www.w3schools.com/xml/xml_schema.asp

XML Schema Example

Valid XML

```
<bookstore>
   <name> James' Store </name>
   <topic>
      <name> XML </name>
      <book isbn="123-456-789">
         <title> James' Book </title>
      </book>
      <author> James </author>
   </topic>
</bookstore>
```

XML Schema

```
<xsd:schema
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:annotation>
  <xsd:documentation xlm:lang="en">
    XML Schema for a Bookstore as an example.
  </xsd:documentation>
</xsd:annotation>
```

XML Schema Example

Valid XML

```
<bookstore>
  <name> James' Store </name>
  <topic>
    <name> XML </name>
    <book isbn="123-456-789">
      <title> James' Book </title>
    </book>
    <author> James </author>
  </topic>
</bookstore>
```

XML Schema

```
…
<xsd:element name="bookstore"
        type="bookstoreType"/>
<xsd:complexType name="bookstoreType">
 <xsd:sequence>
  <xsd:element name="name"
        type="xsd:string"/>
  <xsd:element name="topic"
        type="topicType"
        minOccurs="1"/>
 </xsd:sequence>
</xsd:complexType>
```

XML Schema Example

Valid XML

```
<bookstore>
   <name> James' Store </name>
   <topic>
      <name> XML </name>
      <book isbn="123-456-789">
         <title> James' Book </title>
      </book>
      <author> James </author>
   </topic>
</bookstore>
```

XML Schema

```
…
…
<xsd:complexType name="topicType">
  <xsd:element name="name"
               type="xsd:string"/>
  <xsd:element name="book"
               type="bookType"
               minOccurs="0"/>
</xsd:complexType>
```

XML Schema Example

Valid XML

```
<bookstore>
   <name> James' Store </name>
   <topic>
      <name> XML </name>
      <book isbn="123-456-789">
         <title> James' Book </title>
      </book>
      <author> James </author>
   </topic>
</bookstore>
```

XML Schema

```
…
…
…
<xsd:complexType name="bookType">
  <xsd:element name="title"
               type="xsd:string"/>
  <xsd:element name="author"
               type="xsd:string"/>
  <xsd:attribute name="isbn"
                 type="isbnType"/>
</xsd:complexType>
<xsd:simpleType name="isbnType">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="[0-9]{3}[-][0-9]{3}[-][0-9]{3}"/>
  </xsd:restriction>
</xsd:simpleType>
```

Example Questions

- Create an XML and JSON representation of the semi-structured data graph from slide 9

Any Questions?