Lecture 8 – JSON

Open Data Management & the Cloud

(Data Science & Scientific Computing / UniTS – DMG)

JSON

- JSON (JavaScript Object Notation) is based on a subset of the JavaScript Programming Language Standard ECMA-262 3rd Edition
- JSON is a way of storing and communicating data with specific rules (like XML, YAML, etc.)
- JSON files has extension .json
- JSON uses key-value pairs
- JSON was designed to be human and machine readable
- JSON is easy to read and write
- Language independent even if it comes from JavaScript

https://www.json.org

JSON Example

```json
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 27,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    }
  ],
  "children": [
      "Catherine",
      "Thomas",
      "Trevor"
  ],
  "spouse": null
}
```

Source: https://en.wikipedia.org/wiki/JSON#Syntax

JSON Data Types (1)

- There are only 6 data types in JSON
  - String `"Hello World"`
  - Number `10  1.5  30  1.2e10`
  - Boolean `true  false`
  - Null `null`
  - Array `[ 1, 2, 3]  [ "Hello", "World" ]`
  - Object `{ "name" : "John" }  { "age" : 21 }`
- A *value* can be a string, a number, a boolean, a null, an array or an object
- Array and Objects are also called structures
- In a .json file there can be one (and only one) value

JSON Data Types (2)

- String `"Hello World"`
  - It is a sequence of zero or more Unicode characters
  - It is wrapped in double quotes, using backslash escapes
    - `\"` `\\` `\/`
    - `\b` (backspace), `\f` (formfeed), `\n` (linefeed), `\r` (return), `\t` (tab), `\u` (4 hex digits UNICODE)
  - A character is represented as a single character string
- Number `10` `1.5` `30` `1.2e10`
  - Any kind of number integer, float, exponential
  - The octal and hexadecimal formats are not supported
- Boolean
  - Values can be `true` `false`
- Null
  - Value can be `null`

JSON Data Types (3)

- Array `[ 1, 2, 3]` `[ "Hello", "World" ]`
  - It is an ordered list of values
  - It begins with a left square bracket `[` and ends with a right square bracket `]`
  - Values are separated by comma `,`
  - It is not mandatory all values have the same data type `[ 1, "Bye", true]`
- Object
  - It is an unordered set of name-values pairs
  - It begins with a left curly brace `{` and ends with a right curly brace `}`
  - Each name is followed by colon `:`
  - The name-value pairs are separated by comma `,`

```
{ "name" : "John" , "age" : 21 }
```

JSON Example… again

```json
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 27,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    }
  ],
  "children": [
      "Catherine",
      "Thomas",
      "Trevor"
  ],
  "spouse": null
}
```

Source: https://en.wikipedia.org/wiki/JSON#Syntax

JSON Web Resources

- https://www.json.org
- https://json-schema.org/
- https://www.jsonschemavalidator.net
- https://www.w3schools.com/python/python_json.asp

XML vs JSON (1)

- Both can store data (and metadata) to transfer them
- Both are language-independent
- Both have hierarchical structure
- JSON is simpler to learn, read and write with respect to XML
- JSON files are more human-readable than XML files
- JSON supports few data types with respect to XML
- XML supports include and import of external XML or binary files
- XML supports references (entities, etc.)
- XML supports attributes
- XML supports namespaces
- XML has a robust schema document (XSD) format for validation process
  - JSON validation schema is more a “well formed” check than a proper validation

XML vs JSON (2)

XML fragment:

```xml
<pixel_reading>
	<device>Camera</device>
	<patch>cyan</patch>
	<RGB resolution="8">
		<red>0</red>
		<green>255</green>
		<blue>255</blue>
	</RGB>
</pixel_reading>
```

JSON fragment 1:

```json
{
  "pixel_reading": {
    "device": "Camera",
    "patch": "cyan",
    "RGB": {
      "red": 0,
      "green": 255,
      "blue": 255,
    }
  }
}
```

The resolution attribute is missing!

JSON fragment 2:

```json
{
  "pixel_reading": {
    "device": "Camera",
    "patch": "cyan",
    "RGB": {
      "resolution": 8,
      "reading": {
        "red": 0,
        "green": 255,
        "blue": 255,
      }
    }
  }
}
```