Edinburgh Napier
UNIVERSITY

ADVANCED WEB TECHNOLOGIES
SET09103
TOPIC #06

DATA & DATASTORES

Dr Simon Wells
s.wells@napier.ac.uk
http://www.simonwells.org

Edinburgh Napier UNIVERSITY

TL/DR

Data can be at rest or in motion & exist in different forms at different times. Let's use this to our advantage by learning to store (Database), & serialise it (JSON).

Edinburgh Napier
UNIVERSITY

OVERVIEW

- Data Representation - Describing data structure.
- Data Transports - Telling recipients what to expect.
- Data Stores - Storing data for retrieval at scale.

Edinburgh Napier UNIVERSITY

DATA DRIVEN SITES

- As we start to build more feature rich sites - we generally have more data to structure, manipulate, store, & move around:
  - Data often sent to web-apps, and retrieved from web-apps.
  - Data often stored either in filesystem or in datastores.
  - Data often manipulated within a programming language.
- So considering how data is **stored** & **transported** is important.

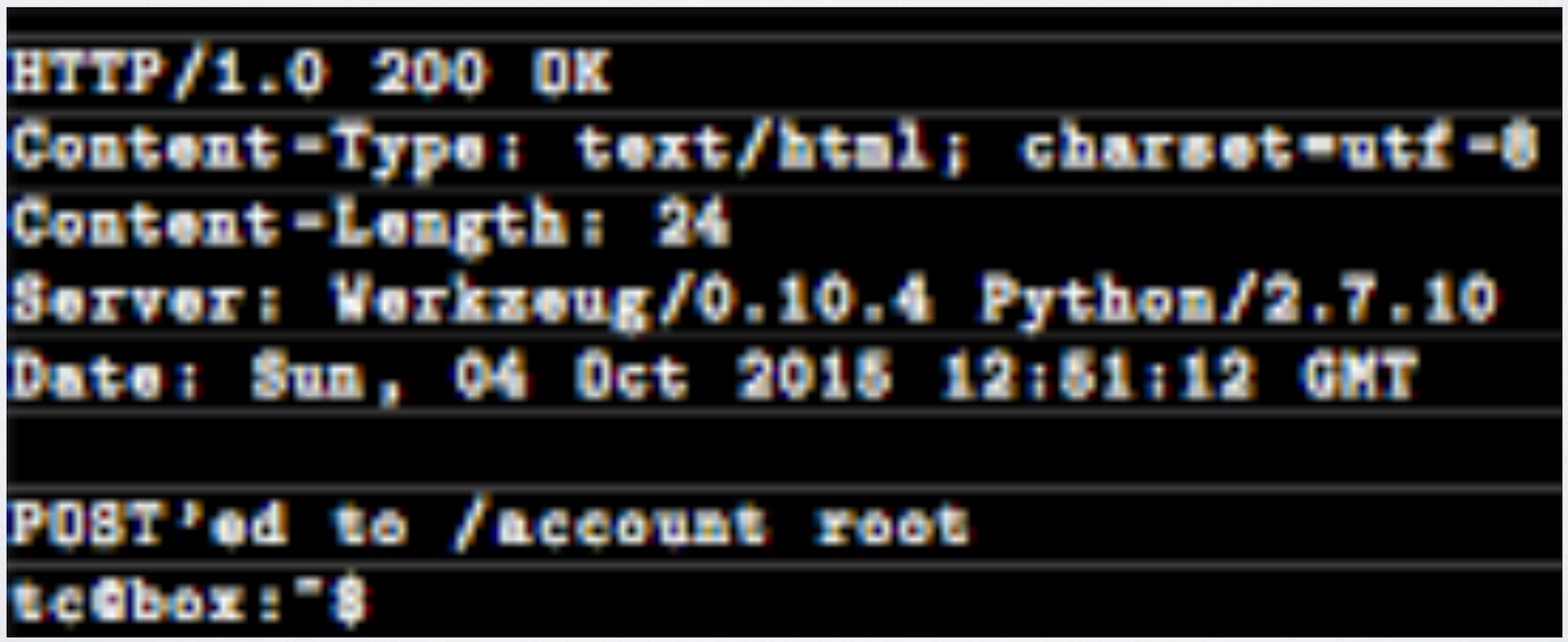
Edinburgh Napier
UNIVERSITY

DO WE TELL THE CLIENT HOW TO USE OUR DATA?

i.e. How does the browser recognise that the content is HTML, or plain text, or JSON, or XML, or …?

MEDIA TYPES

- An HTTP header to communicate to the recipient what to expect.
- Indicate the format of the data so the recipient can make a decision about how to handle it:
  - Header, e.g. **Context-Type: text/html;** + (optional) parameter: **charset=UTF-8**
- Standardised & publicised by the Internet Assigned Numbers Authority (IANA):
  - Many examples: text/html, text/javascript, text/plain, text/xml, application/json, application/pdf, audio/mpeg,, image,/png, image/jpeg, &c.
  - NB. Originated in **M**ultipurpose **I**nternet **M**ail **E**xtensions (**MIME**) types. WHATWG discourages use of mediatype and still uses mimetype. IANA & IETF use mediatype.
  - Annoyingly the HTTP header refers to "Content-Type" rather than "Media-Type".



MEDIA TYPES

- Composed of a type, subtype and optional parameters with following format:
  - **type “/” [tree “.”] subtype [“+” suffix] *[“;” parameter]**
  - e.g. for an HTML file:
    - **text/html; charset=UTF-8**
      - **type:** text
      - **subtype:** HTML
      - **parameter:** charset=UTF-8 - indicate character encoding
- Top level names can include: application, audio, example, image, message, model, multipart, text, video
- Tree Prefix: standard, vendor (vnd), personal (prs), unregistered (x)
- Suffix: Augmentation to media type to specify underlying structure of media type, e.g. +xml. +json,+zip (& others)

Edinburgh Napier UNIVERSITY

MEDIA TYPES

- We will build on this in the next topic when we bring everything together to design and build an API/Web Service.
  - HTTP Request + Verb
  - HTTP Response + Code + Mediatype + Content
    - So far: Return text/html + HTML content
    - Next topic: Return application/json + JSON content

- Interaction with a resource causes data manipulation.
- HTTP can return different types of data as a result of a request.
- HTTP can indicate the *type* of data being returned using the **mediatype** header.
- But…

WHAT DO WE MEAN BY DIFFERENT TYPES OF DATA?

Edinburgh Napier UNIVERSITY

TYPES OF DATA

- How we structure & store data has a great effect on how easily we can then use, or re-use it.

- Human readable: HTML

- Human & Machine readable (machine can parse the file and reconstruct or reuse the data) but otherwise written in plain text:

  - JSON, XML, YAML, RDF, ATOM, …

- We'll concentrate on JSON (but consider XML so you know it when you see it).

JSON

- **J**ava**S**cript **O**bject **N**otation
  - If you're familiar with JavaScript & OOP then JSON should make some sense
- An open standard format
- Text-based/Human-readable
- Used to transmit data objects made of attribute-value pairs
- Originally derived from Javascript as a way to serialise Javascript objects but became popular as a language independent data format, e.g. can generate and parse JSON in many languages, e.g. Javascript (trivial), Python, and libraries available if no default standard lib support
- Described in RFC7159 & ECMA 404 (minimal, syntax only)
- Official media type is **application/json** & files are named **.json**
- **http://www.json.org/**

Edinburgh Napier UNIVERSITY

```
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 25,
  "height_cm": 167.6,
  "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": [],
  "spouse": null
}
```

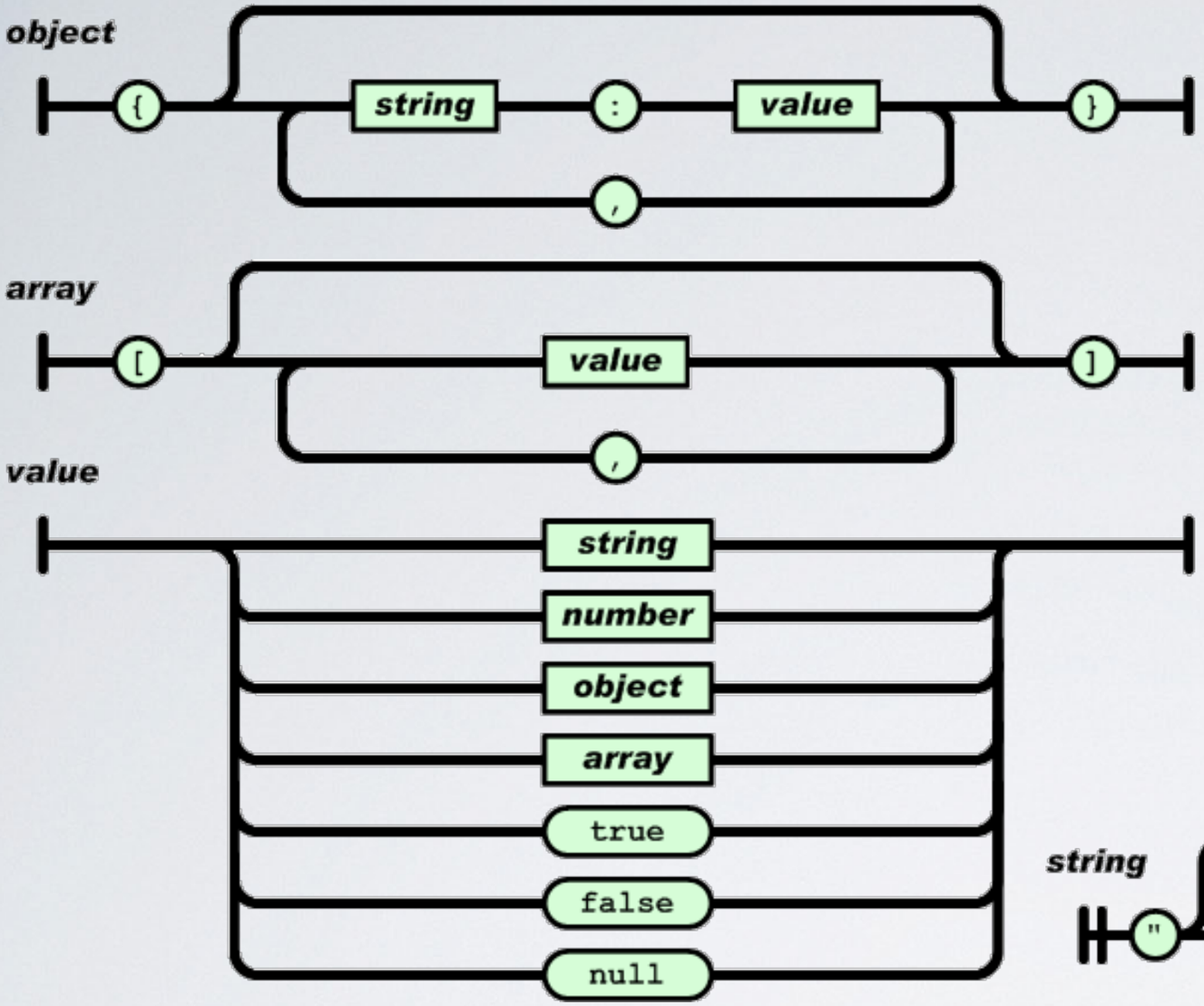
JSON LANGUAGE

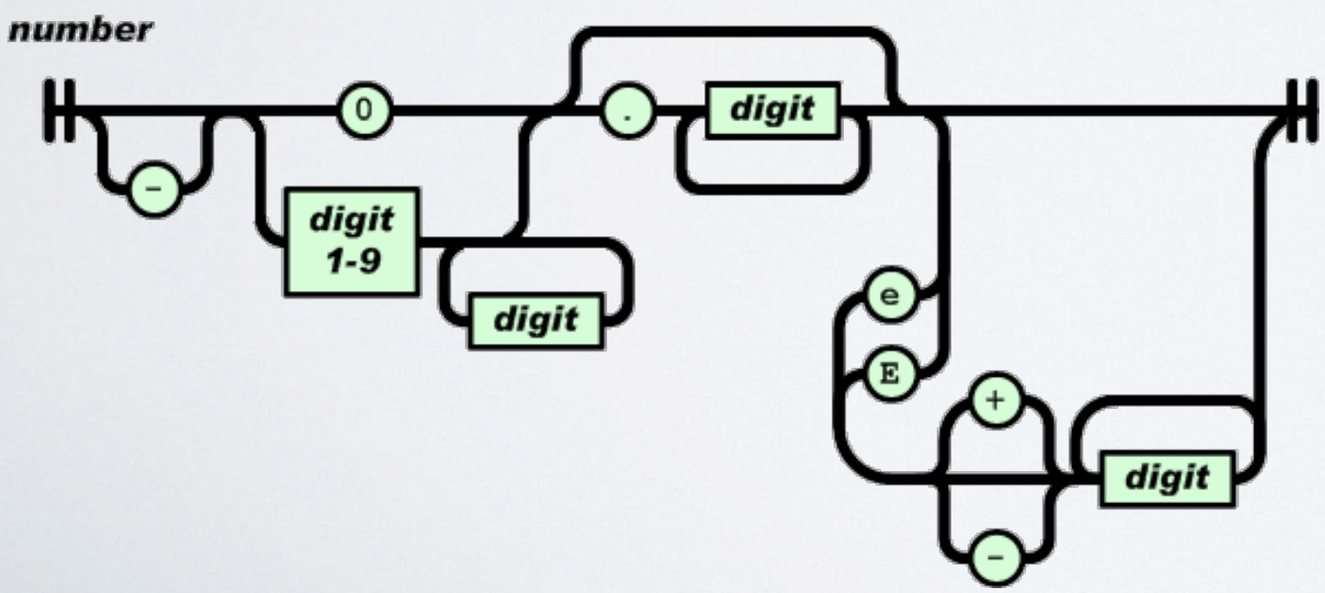
- Plain text format - can create & edit using your text editor
- Is iteratively constructed starting with either an object {} or an array []
- Array can store a comma separated list of values
- Object can store a comma separated list of key:value pairs
- Values are objects, arrays, strings, numbers, booleans, null
- An easier way to visualise this is with railroad diagrams

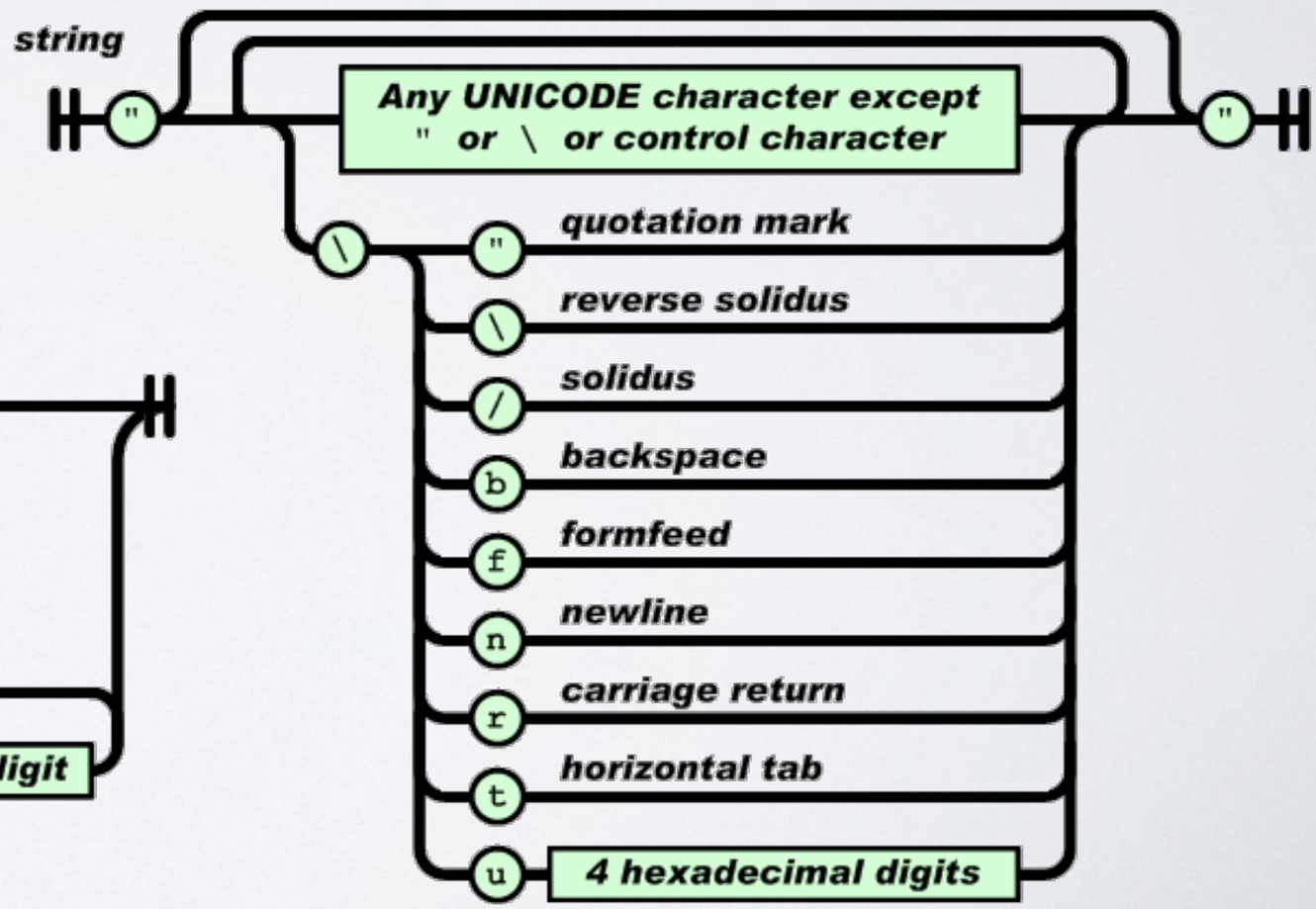
```
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 25,
  "height_cm": 167.6,
  "address": {
    "streetAddress": "21 2nd
Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    }
  ],
  "children": [],
  "spouse": null
}
```

Edinburgh Napier UNIVERSITY

JSON RAILROAD DIAGRAMS







JSON EXAMPLE

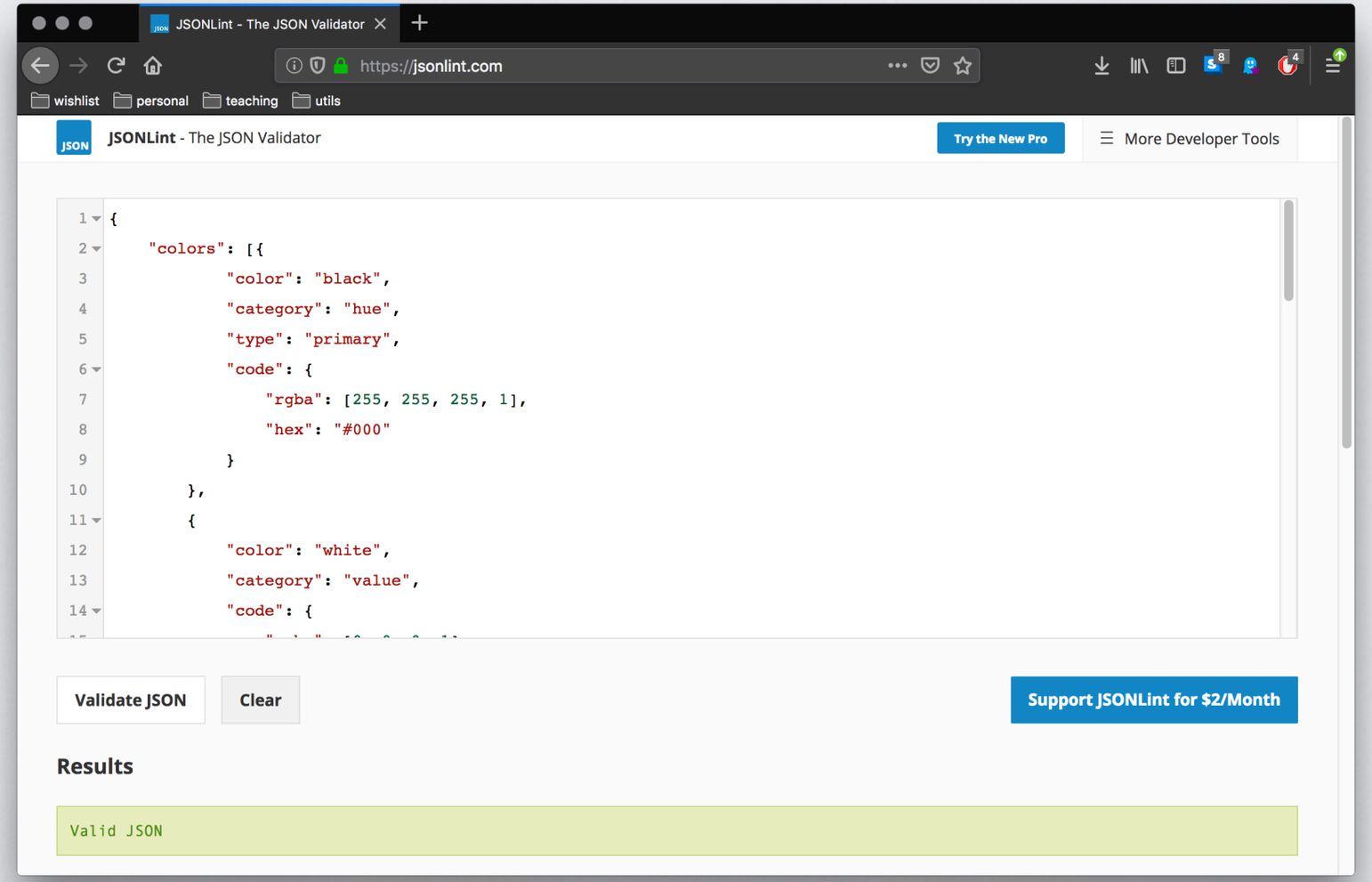
- Datatypes:
  - Numbers
  - Strings
  - Booleans
  - Arrays []
  - Objects {}
  - null
- http://json.org/
- http://jsonlint.com/

```
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 25,
  "height_cm": 167.6,
  "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": [],
  "spouse": null
}
```

Edinburgh Napier UNIVERSITY

JSONLINT.COM

- A useful online tool for quickly editing & *validating* JSON files
- Edit then copy/paste to text file and save



Edinburgh Napier
UNIVERSITY

JSON RESOURCES

- **References**:
  - JSON Website: http://json.org/
- **Standards**:
  - RFC7159: http://tools.ietf.org/html/rfc7159
  - ECMA-404: http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf
  - JSON Schema: http://json-schema.org
- **Tools:**
  - JSONLint: http://jsonlint.com/

Edinburgh Napier UNIVERSITY

XML

- XML was meant to be the *"solution to everything"*
- A free, open standard, e**X**tensible **M**arkup **L**anguage
  - Defines rules for encoding documents in a format that is human and machine readable
- Defined by W3C XML 1.0 specification (& several ancillary specifications, e.g. XML Namespaces, XPath, XSLT, XQuery)
- Design goals were for simplicity, generality, and usability
- Became widely used for representation of arbitrary data structures (like JSON has since) - and an important technology in the development of web services
- Official media type is **application/xml text/xml** & files are named **.xml**

Edinburgh Napier UNIVERSITY

XML APPLICATIONS

- Many document formats use XML syntax:
  - RSS, Atom, SOAP, XHTML, MS Office/Office Open XML, Open/Libre Office Open Document Format, iWork, XMPP, .NET config files, OS X config files
- Internet data exchange (RFC 7303) - rules for constructing Internet Media Types (e.g. if building a RESTful HATEOAS based system)

Edinburgh Napier UNIVERSITY

INTRODUCTION TO KEY PARTS OF XML

- Characters - An XML document is a string of **unicode** characters

- Processor & Application - Processor (often known as a **parser**) analyses markup and passes structured information to an application (NB. Specification sets out requirements for what an XML processor must be do/not do)

- Markup & Content - characters of document divided in markup and content. Generally markup strings start and end with **<** and **>** or **&** and **;**. If characters are not markup then they are content

- Start, End, & Empty Element Tags: e.g. <section> </section> <line break />

- Elements: A logical component of a document that begins with a start-tag and ends with a *matching* end-tag, e.g. <greeting>Hello Napier</greeting>

- Attributes: Within a tag, e.g. between < and >, a a tag may have key=''value'' pairs, e.g. <img src='' merchiston-campus.png'', alt=''View of the lovely buildings at Napier's Merchiston campus''>

- XML declaration: <?xml version=''1.0'' encoding=''UTF-8''?>

Edinburgh Napier
UNIVERSITY

WELL FORMED XML

- An XML document is a tree structure - a single "root" element contains all other "child" elements

- Contents are only properly encoded Unicode characters

- Special syntax characters (e.g. <, >, &) only used when performing markup roles unless otherwise escaped

- Start, end and empty element tags must be correctly nested with none missing or overlapping

- Element tags are case sensitive and pairs of start/end tags must match exactly

- Tag names cannot contain !"#$%&'()*+,/;<=>?@[\]^`{|}~ or space chara or start with '-' or '.' or a numeric digit

Edinburgh Napier UNIVERSITY

XML SCHEMAS

- Document Type Definition (DTD)
  - Oldest, inherited from SGML, ubiquitous because in XML 1.0 standard, can be embedded within XML documents
- XML Schema (XSD)
  - Successor to DTDs, more powerful, rich datatyping, detailed constraints on XML logical structure, Based on XML so can be processed by ordinary XML tools

```xml
<person>
  <firstName>John</firstName>
  <lastName>Smith</lastName>
  <age>25</age>
  <address>
    <streetAddress>21 2nd Street</streetAddress>
    <city>New York</city>
    <state>NY</state>
    <postalCode>10021</postalCode>
  </address>
  <phoneNumbers>
    <phoneNumber>
      <type>home</type>
      <number>212 555-1234</number>
    </phoneNumber>
    <phoneNumber>
      <type>fax</type>
      <number>646 555-4567</number>
    </phoneNumber>
  </phoneNumbers>
  <gender>
    <type>male</type>
  </gender>
</person>
```

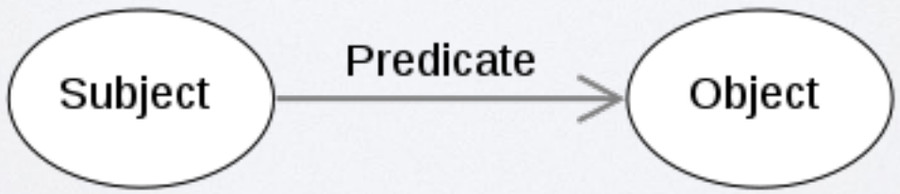
```xml
<person firstName="John" lastName="Smith" age="25">
  <address streetAddress="21 2nd Street" city="New York"
state="NY" postalCode="10021" />
  <phoneNumbers>
     <phoneNumber type="home" number="212 555-1234"/>
     <phoneNumber type="fax"  number="646 555-4567"/>
  </phoneNumbers>
  <gender type="male"/>
</person>
```

```json
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 25,
  "height_cm": 167.6,
  "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": [],
  "spouse": null
}
```

Edinburgh Napier UNIVERSITY

RDF

- Resource Description Framework
- Started as a data model for metadata
- Has become a *de facto* information modelling language - simplest way to define a binary relationship
- Doesn't have a single syntax: RDF/XML, Turtle, Notation3, Triples, JSON-LD, RDF/JSON
- Basic idea is to state some knowledge, put that knowledge at a URI, then to process that knowledge automatically (semantic web) e.g. There is a person whose name is Simon
  - Subject: the resource we're referring to
  - Object: The value (resource or literal) associated with the subject
  - Predicate: a term that describes or modifies the subject, ie. denotes a relationship between subject & object



Edinburgh Napier UNIVERSITY

YAML

- Originally "Yet Another Markup Language" but now "YAML Ain't Markup Language"
- Human readable language for data serialisation
- Often used for configuration files (.yaml files)
- https://yaml.org
- Specification (1.2): https://yaml.org/spec/1.2/spec.html

```
--- !clarkevans.com/^invoice
invoice: 34843
date   : 2001-01-23
bill-to: &id001
    given  : Chris
    family : Dumars
    address:
        lines: |
            458 Walkman Dr.
            Suite #292
        city    : Royal Oak
        state   : MI
        postal  : 48046
ship-to: *id001
product:
    - sku         : BL394D
      quantity    : 4
      description : Basketball
      price       : 450.00
    - sku         : BL4438H
      quantity    : 1
      description : Super Hoop
      price       : 2392.00
tax  : 251.42
total: 4443.52
comments: >
    Late afternoon is best.
    Backup contact is Nancy
    Billsmer @ 338-4338.
```

Edinburgh Napier UNIVERSITY

ATOM & RSS

- XML languages used for "web feeds" & data syndication

- Share recent entries (blogging), site updates, full text, multimedia files with consumers (feed reader)

- Share with other sites (syndication) - pull feed of data and incorporate it into your site

Edinburgh Napier
UNIVERSITY

```
<?xml version="1.0" encoding="utf-8"?>

<feed xmlns="http://www.w3.org/2005/Atom">

	<title>Dr Simon Wells</title>
	<subtitle>sesquipedalia verba</subtitle>
	<link href="http://simonwells.org/feed/" rel="self" />
	<link href="http://simonwells.org/" />
	<id>urn:uuid:60a76c80-d399-11d9-b91C-0003939e0af6</id>
	<updated>2019-10-13T18:30:02Z</updated>


	<entry>
		<title>Publications</title>
		<link href="http://www.simonwells.org/publications/" />
		<link rel="alternate" type="text/html" href="http://www.simonwells.org/publications.html"/>
		<id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344efa6a</id>
		<updated>2018-04-09T09:30:02Z</updated>
		<summary>Some text.</summary>
		<content type="xhtml">
			<div xmlns="http://www.w3.org/1999/xhtml">
				<p>S. Wells (2019) "A Paroxysm of Dissent: Partisan Political Advertising During the Brexit Campaign"
Abstract for paper presented at the 3rd European Conference on Argumentation: Reason to Dissent (ECA19), Groningen, Nethlands</
p>
			</div>
		</content>
		<author>
			<name>Simon Wells</name>
			<email>mail@simonwells.org</email>
		</author>
	</entry>

</feed>
```

Edinburgh Napier UNIVERSITY

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<rss version="2.0" xmlns:atom="http://www.w3.org/2005/Atom">
  <channel>
    <title>Dr Simon Wells</title>
    <link>http://www.simonwells.org/</link>
    <description>Recent content on Dr Simon Wells</description>
    <generator>Hugo -- gohugo.io</generator>
    <language>en-gb</language>
    <copyright>Copyright &amp;amp;copy; 2001-2018 - Simon Wells</copyright>
    <lastBuildDate>Mon, 09 Apr 2018 00:00:00 +0000</lastBuildDate>

        <atom:link href="http://www.simonwells.org/index.xml" rel="self" type="application/rss+xml"/>

    <item>
      <title>Publications</title>
      <link>http://www.simonwells.org/publications/</link>
      <pubDate>Mon, 09 Apr 2018 00:00:00 +0000</pubDate>

      <guid>http://www.simonwells.org/publications/</guid>
      <description>S. Wells (2019) "A Paroxysm of Dissent: Partisan Political Advertising During the Brexit Campaign"
Abstract for paper presented at the 3rd European Conference on Argumentation: Reason to Dissent (ECA19),
Groningen, Nethlands</description>
    </item>

  </channel>
</rss>
```

Edinburgh Napier UNIVERSITY

PUTTING IT ALL TOGETHER

Send an Event to Account route

PUT https://cool-web-app.example/accounts/<userid>

Example Request:

```
PUT /accounts/i-04cee0b5ed03c3-5c9b17c3 HTTP/1.1
Content-Type: application/json

{
    "auth": "XLQLrvxlc8IM6FGOv28aH5g0t35",
    "uid": "user@our.example",
    "contact": {"type": "mobile_number",
                "number": "00447982321123"}
}
```

- Interaction with a resource causes data manipulation.
- HTTP can return different types of data as a result of a request.
- HTTP can indicate the *type* of data being returned using the **mediatype** header.
- Data can be structured in various ways.
- But…

WHAT DO WE MEAN BY STORING DATA?

Edinburgh Napier
UNIVERSITY

SERVING PAGES

- Multiple options (with different advantages/disadvantages). Increasingly dynamic:
  - Static server + HTML in files (traditional static site).
  - Return HTML as string from Flask.
  - Return HTML as file from Flask (static serving).
  - Return HTML from template using Flask. Flask assembles & returns HTML by completing the template (dynamic page generation).
- Where is the data that is used to complete the template?

Edinburgh Napier UNIVERSITY

TOWARDS DATA DRIVEN SITES

- Data can be *live* data, i.e. stored in variables in the Python app. Generated at runtime,
- Data can be retrieved from other locations, i.e. third-party APIs. Often JSON.
- Data can be retrieved from file on disk.
- Data can be retrieved from a database.
- When should we choose a database instead of disk storage?

```
from flask import Flask, g
import sqlite3

app = Flask(__name__)
db_location = 'var/test.db'

def get_db():
    db = getattr(g, 'db', None)
    if db is None:
        db = sqlite3.connect(db_location)
        g.db = db
    return db

@app.teardown_appcontext
def close_db_connection(exception):
    db = getattr(g, 'db', None)
    if db is not None:
        db.close()

def init_db():
    with app.app_context():
        db = get_db()
        with app.open_resource('schema.sql', mode='r') as f:
            db.cursor().executescript(f.read())
        db.commit()

@app.route("/")
def root():
    db = get_db()
    db.cursor().execute('insert into albums values
        (“American Beauty", "Grateful Dead", "CD")')
    db.commit()

    page = []
    page.append('<html><ul>')
    sql = "SELECT rowid, * FROM albums ORDER BY artist"
    for row in db.cursor().execute(sql):
        page.append('<li>')
        page.append(str(row))
        page.append('</li>')

    page.append('</ul></html>')
    return ''.join(page)
```

Edinburgh Napier
UNIVERSITY

Edinburgh Napier UNIVERSITY

SUMMARY

- This topic:
  - We've considered media-types, data transports, and data storage.
- Next:
  - API & Web Service Design & Development

Edinburgh Napier UNIVERSITY

WRAPPING UP

- We've taken a look at:
- Data representation (JSON + others)
- Data transport (mediatypes)
- Data storage (SQLite)