Welcome to INFO216:
Knowledge Graphs

Andreas L. Opdahl
<Andreas.Opdahl@uib.no>

Session S11-S12: Web APIs

- Themes:
  - *what are web APIs (web services)?*
    - service-oriented architecture (SOA)
    - JavaScript Object Notation (JSON)
  - *semantic web APIs (web services)*
    - JSON-LD
    - programming JSON-LD in Python

Readings

- Sources:
  - json.org
  - json-ld.org
    - in particular the videos on top of the page
  - seksjon 2 i JSON-LD 1.0 Processing Algorithms and API (https://www.w3.org/TR/json-ld-api/#features)

Web APIs
(RESTful web services)

Web APIs

- *When we make programs, what if we could call other programs over the web (almost) as easy as we can call methods (or sub-routines, procedures, functions) we have written ourselves?*
- Web APIs let ut do this!
- A Web API offers several operations or functions that
  - take inputs in a well-defined format
  - perform a well-defined operation on the inputs
  - return outputs in a well-defined format
- Examples of functions:
  - registering student12345 to course info216 at uib
  - listing all students in the course
  - unregistering the student from the course

Example calls

- We call a Web API by requesting an IRL
- Examples:
  - registering student12345 to course info216 at uib:
    GET http//onewebapi.example.no/register?
    university=uib&course=info216&student=student12345
  - listing all info216 students:
    GET http//onewebapi.example.no/listStudents?
    university=uib&course=info216
  - unregistering from the course:
    GET http//onewebapi.example.no/unregister?
    university=uib&course=info216&student=student12345
- Of course, our program can change the values uib, student12345 and info216 between each call

Example calls (different style)

- We can design Web APIs with different IRL style
- Examples:
  - registering student12345 to course info216 at uib:
    POST http//onewebapi.example.no
    /registrations/uib/info216/student12345
  - listing all info216 students:
    GET http//onewebapi.example.no
    /registrations/uib/info216
  - unregistering from the course:
    DELETE http//onewebapi.example.no
    /registrations/uib/info216/student12345
- The HTTP operations (PUT, GET, POST, DELETE, and also PATCH) are used for creation, retrieval, update and deletion

Service-oriented architecture (SOA)

- Building applications by combining operations from different Web APIs
  - company internal over intranets
  - public over the internet
  - “service orchestration” and “choreography”
  - the applications can offer new operations in turn
  - operations are ideally small and stateless
- Everything is based on established proven standards:
  - HTTP, IRL, JSON (...TCP/IP, Unicode...)

Providing Web APIs

- A bit more complicated
  - you need a web server (Apache, Nginx...)
  - the web server must allow executing programs
  - translate resource requests into program calls
- The program itself is simple:
  - takes input parameters on command line (GET)
  - may also take longer input in JSON (POST)
  - outputs results in JSON
    - which the web server returns to the client

Providing Web APIs

- How to translate resource requests into program calls?
- Example:
  - we have written this program:
    /home/programs/student-list <univ-code> <course-code>
  - we receive this request:
    GET /listStudents?university=uib&course=info216
    Host: onewebapi.example.no
  - we need a rewrite rule like this (example from Apache, based on *regular expressions*):
    RewriteRule
    ^/listStudents?university=([a-z]+)&course=([a-z0-9]+)$
    /home/programs/student-list $1 $2

Providing Web APIs

- How to translate resource requests into program calls?
- Example:
  - we have written this program:
    /home/programs/student-list <univ-code> <course-code>
  - we receive this request:
    GET /listStudents?university=uib&course=info216
    Host: onewebapi.example.no
  - we need a rewrite rule like this (example from Apache, based on *regular expressions*):
    RewriteRule
    ^/listStudents?university=([a-z]+)&course=([a-z0-9]+)$
    /home/programs/student-list $1 $2

Providing Web APIs

- How to translate resource requests into program calls?
- Example:
  - we have written this program:
    /home/programs/student-list <univ-code> <course-code>
  - we receive this request:
    GET /listStudents?university=uib&course=info216
    Host: onewebapi.example.no
  - we need a rewrite rule like this (example from Apache, based on *regular expressions*):
    RewriteRule
    ^/listStudents?university=([a-z]+)&course=([a-z0-9]+)$
    /home/programs/student-list uib $2

Providing Web APIs

- How to translate resource requests into program calls?
- Example:
  - we have written this program:
    /home/programs/student-list <univ-code> <course-code>
  - we receive this request:
    GET /listStudents?university=uib&course=info216
    Host: onewebapi.example.no
  - we need a rewrite rule like this (example from Apache, based on *regular expressions*):
    RewriteRule
    ^/listStudents?university=([a-z]+)&course=([a-z0-9]+)$
    /home/programs/student-list uib $2

Providing Web APIs

- How to translate resource requests into program calls?
- Example:
  - we have written this program:
    /home/programs/student-list <univ-code> <course-code>
  - we receive this request:
    GET /listStudents?university=uib&course=info216
    Host: onewebapi.example.no
  - we need a rewrite rule like this (example from Apache, based on *regular expressions*):
    RewriteRule
    ^/listStudents?university=([a-z]+)&course=([a-z0-9]+)$
    /home/programs/student-list uib info216

JSON

JSON

```
{
  "homepage": "http://me.markus-lanthaler.com",
  "name": "Markus Lanthaler",
  "workplaceHomepage": "http://www.tugraz.at/"
}
```

Name-Value pair

Object

Name

Value (String, Number, Boolean, null or Array)

JavaScript Object Notation (JSON)
www.json.org

JSON

```
{
  "homepage": "http://me.markus-lanthaler.com",
  "name": "Markus Lanthaler",
  "workplaceHomepage": "http://www.tugraz.at/"
}
```

Name-Value pair

Object

Name

Value (String, Number, Boolean, null or Array)

```
{
  "homepage": "http://me.markus-lanthaler.com",
  "name":
      {
        “firstname” : "Markus",
        “lastname” : “Lanthaler”
      },
  "workplaceHomepages": [ "http://www.tugraz.at/", “http://www.uib.no” ]
}
```

Array

Element (String, Number, Boolean, null or Object)

JSON

```
{
  "homepage": "http://me.markus-lanthaler.com",
  "name": "Markus Lanthaler",
  "workplaceHomepage": "http://www.tugraz.at/"
}
```

Name-Value pair

Object

Name

Value (String, Number, Boolean, null or Array)

```
{
  "homepage": "http://me.markus-lanthaler.com",
  "name":
      {
        “firstname” : "Markus",
        “lastname” : “Lanthaler”
      },
  "workplaceHomepages": [ "http://www.tugraz.at/", “http://www.uib.no” ]
}
```

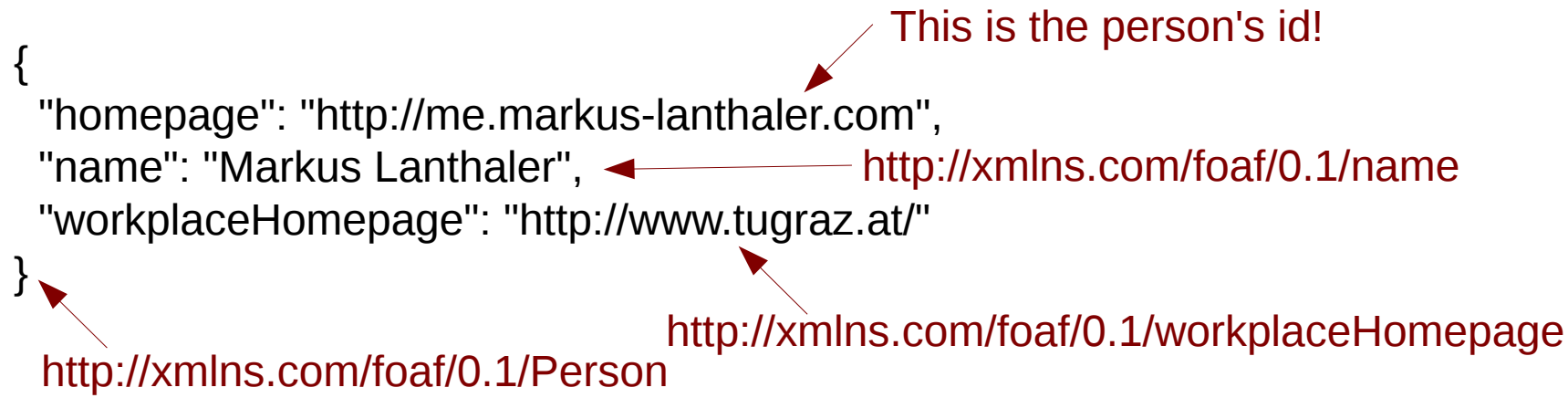
Almost identical to Python dictionaries and lists!

Array

Element (String, Number, Boolean, null or Object)

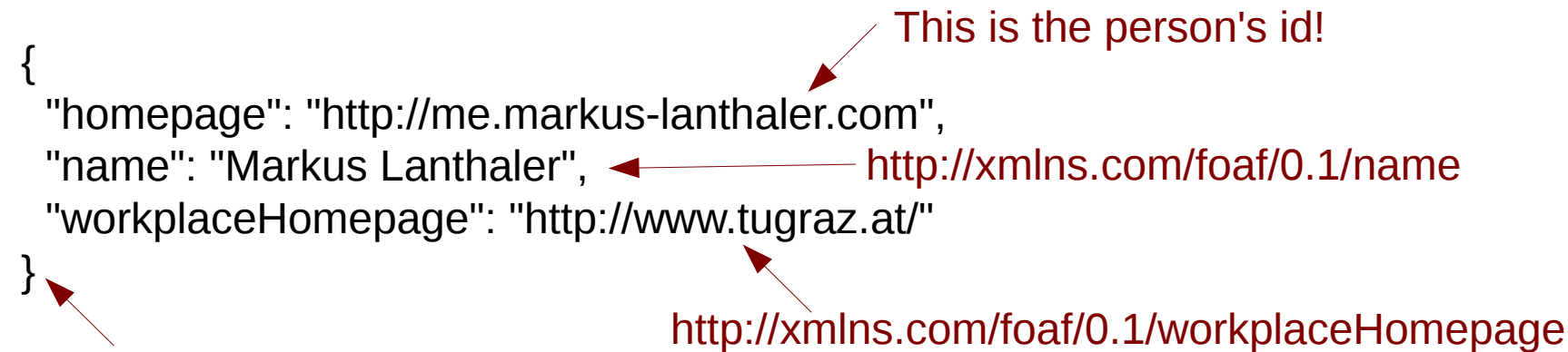
JSON-LD

JSON



How to represent semantic data in JSON?

JSON-LD




```
{
  "@id": "http://me.markus-lanthaler.com",
  “@type” : “http://xmlns.com/foaf/0.1/Person”,
  "http://xmlns.com/foaf/0.1/name": "Markus Lanthaler",
  "http://xmlns.com/foaf/0.1/workplaceHomepage":
        { “@id” : "http://www.tugraz.at/" }
}
```

JSON Linked Data (JSON-LD)
json-ld.org

JSON-LD to Turtle

```
{
  "@id": "http://me.markus-lanthaler.com",
  “@type” : “http://xmlns.com/foaf/0.1/Person”,
  "http://xmlns.com/foaf/0.1/name": "Markus Lanthaler",
  "http://xmlns.com/foaf/0.1/workplaceHomepage":
        { “@id” : "http://www.tugraz.at/" }
}
```

JSON Linked Data (JSON-LD)
json-ld.org

JSON-LD to Turtle

<http://me.markus-lanthaler.com>

```
{
  "@id": "http://me.markus-lanthaler.com",
  “@type” : “http://xmlns.com/foaf/0.1/Person”,
  "http://xmlns.com/foaf/0.1/name": "Markus Lanthaler",
  "http://xmlns.com/foaf/0.1/workplaceHomepage":
        { “@id” : "http://www.tugraz.at/" }
}
```

JSON Linked Data (JSON-LD)
json-ld.org

JSON-LD to Turtle

```
<http://me.markus-lanthaler.com>
        a       http://xmlns.com/foaf/0.1/Person";
```

```
{
  "@id": "http://me.markus-lanthaler.com",
  "@type" : "http://xmlns.com/foaf/0.1/Person",
  "http://xmlns.com/foaf/0.1/name": "Markus Lanthaler",
  "http://xmlns.com/foaf/0.1/workplaceHomepage":
        { "@id" : "http://www.tugraz.at/" }
}
```

JSON Linked Data (JSON-LD)
json-ld.org

JSON-LD to Turtle

```
<http://me.markus-lanthaler.com>
        a       http://xmlns.com/foaf/0.1/Person";
        <http://xmlns.com/foaf/0.1/name>
                "Markus Lanthaler";
```

```
{
  "@id": "http://me.markus-lanthaler.com",
  “@type” : “http://xmlns.com/foaf/0.1/Person”,
  "http://xmlns.com/foaf/0.1/name": "Markus Lanthaler",
  "http://xmlns.com/foaf/0.1/workplaceHomepage":
        { “@id” : "http://www.tugraz.at/" }
}
```

JSON Linked Data (JSON-LD)
json-ld.org

JSON-LD to Turtle

```
<http://me.markus-lanthaler.com>
        a       http://xmlns.com/foaf/0.1/Person";
        <http://xmlns.com/foaf/0.1/name>
                "Markus Lanthaler";
        <http://xmlns.com/foaf/0.1/workplaceHomepage>
                <http://www.tugraz.at/"> .
```

```
{
  "@id": "http://me.markus-lanthaler.com",
  "@type" : "http://xmlns.com/foaf/0.1/Person",
  "http://xmlns.com/foaf/0.1/name": "Markus Lanthaler",
  "http://xmlns.com/foaf/0.1/workplaceHomepage":
        { "@id" : "http://www.tugraz.at/" }
}
```

JSON Linked Data (JSON-LD)
json-ld.org

Some reserved keys in JSON-LD

- @id: signifies that the JSON object with the @id key is identified by a particular IRI
- @type: signifies that the JSON object with the @type key has a particular RDF type (or several types)
- @value: signifies that a value is a literal
- @context: signifies a JSON object that contains the context (or semantic mapping) for the other objects in the same JSON array
- @base, @graph, @language, @vocab, ...

JSON-LD context

```
{
  "@context": {
    "name": "http://xmlns.com/foaf/0.1/name",
    "homepage": {
      "@id": "http://xmlns.com/foaf/0.1/homepage",
      "@type": "@id"
    }
  },
  "@id": "http://me.markus-lanthaler.com/",
  "name": "Markus Lanthaler",
  "homepage": "http://www.markus-lanthaler.com/"
}
```

Another JSON-LD context

```
{
  "@context": {
    "website": "http://xmlns.com/foaf/0.1/homepage"
  },
  "@id": "http://me.markus-lanthaler.com/",
  "http://xmlns.com/foaf/0.1/name": "Markus Lanthaler",
  "website": { "@id": "http://www.markus-lanthaler.com/" }
}
```

JSON-LD forms

- The same graph can be expressed in several different ways:
  - expanded form: *expansion* removes context by pushing semantics out into the objects
    - also does regularisation
  - compact form: *compaction* simplifies the objects by pulling semantics back into the context
  - *flattened* form: normalised form for easier parsing by computer
- Regularised and normalised forms are easier to program than “free” JSON-LD because they have a more consistent structure

Expansion

- Goals:
  - removing the contextual information object
  - pushing it out into the other objects
  - ensuring all values are in a regular form
- Steps:
  - expanding all properties to absolute IRIs
  - expressing all values in arrays in expanded form
- The most verbose and regular way of writing JSON-LD:
  - all contextual information is stored locally with each value

Expansion (before)

```
{
  "@context": {
    "name": "http://xmlns.com/foaf/0.1/name",
    "homepage": {
      "@id": "http://xmlns.com/foaf/0.1/homepage",
      "@type": "@id"
    }
  },
  "@id": "http://me.markus-lanthaler.com/",
  "name": "Markus Lanthaler",
  "homepage": "http://www.markus-lanthaler.com/"
}
```

Expansion (after)

```
[
  {
    "@id": "http://me.markus-lanthaler.com/",
    "http://xmlns.com/foaf/0.1/name": [
      { "@value": "Markus Lanthaler" }
    ],
    "http://xmlns.com/foaf/0.1/homepage": [
      { "@id": "http://www.markus-lanthaler.com/" }
    ]
  }
]
```

Compaction

- The opposite of extraction
  - based on a user-supplied context object
- Goals:
  - condensing contextual information into an object
  - pulling it out of the other objects
  - expressing data in application specific terms
  - making data easier to read for humans
  - make data easy to program against
- Steps:
  - condenses properties with IRIs
  - adds the @context to the array

Compaction (before)

```
[{
    "@id": "http://me.markus-lanthaler.com/",
    "http://xmlns.com/foaf/0.1/name": [
      { "@value": "Markus Lanthaler" }
    ],
    "http://xmlns.com/foaf/0.1/homepage": [
      { "@id": "http://www.markus-lanthaler.com/" }
]}]
```

+

```
{  "@context": {
    "name": "http://xmlns.com/foaf/0.1/name",
    "homepage": {
      "@id": "http://xmlns.com/foaf/0.1/homepage",
      "@type": "@id"
}}}
```

Compaction (after)

```
{
  "@context": {
    "name": "http://xmlns.com/foaf/0.1/name",
    "homepage": {
      "@id": "http://xmlns.com/foaf/0.1/homepage",
      "@type": "@id"
    }
  },
  "@id": "http://me.markus-lanthaler.com/",
  "name": "Markus Lanthaler",
  "homepage": "http://www.markus-lanthaler.com/"
}
```

Flattening

- Expansion:
  - ensures that a document is in a uniform structure
- Flattening:
  - also ensures that the shape of the data is deterministic
- Steps:
  - all properties of a node are collected in a single JSON object
  - all blank nodes are labeled with a blank node identifier
- Flattening makes it easier to process JSON-LD data

Flattening (before)

```
{
  "@context": {
    "name": "http://xmlns.com/foaf/0.1/name",
    "knows": "http://xmlns.com/foaf/0.1/knows"
  },
  "@id": "http://me.markus-lanthaler.com/",
  "name": "Markus Lanthaler",
  "knows": [
    {
      "name": "Dave Longley"
    }
  ]
}
```

Flattening (after – expanded form)

```
[ {
    "@id": "_:t0",
    "http://xmlns.com/foaf/0.1/name": [
      { "@value": "Dave Longley" }
    ]
  },
  {
    "@id": "http://me.markus-lanthaler.com/",
    "http://xmlns.com/foaf/0.1/name": [
      { "@value": "Markus Lanthaler" }
    ],
    "http://xmlns.com/foaf/0.1/knows": [
      { "@id": "_:t0" }
] } ]
```

Flattening (after – compacted form)

```
{
  "@context": {
    "name": "http://xmlns.com/foaf/0.1/name",
    "knows": "http://xmlns.com/foaf/0.1/knows"
  },
  "@graph": [
    {
      "@id": "_:t0",
      "name": "Dave Longley"
    },
    {
      "@id": "http://me.markus-lanthaler.com/",
      "name": "Markus Lanthaler",
      "knows": { "@id": "_:t0" }
} ] }
```