XML, JSON, HTTP, and REST
Consuming the web - Crash Course

Rasmus Lystrøm
External Associate Professor
ITU

What is XML?

eXtensible Markup Language

Markup language like HTML

Designed to carry data, not to display data

Tags are not predefined – You must define your own tags

Designed to be self-descriptive

XML Does Not DO Anything

XML was created to structure, store, and transport information

```xml
<?xml version="1.0" encoding="UTF-8"?>
<note id="1">
  <to>Tom</to>
  <from>Kathleen</from>
  <heading>Reminder</heading>
  <body>Don't forget small change!</body>
</note>
```

How Can XML be Used?

Separates data from HTML

Simplifies data sharing

Simplifies data transport

Simplifies platform changes

Used to Create New Internet Languages

- XHTML
- WSDL for describing web services
- RSS and ATOM for news feeds

What is JSON?

JavaScript Object Notation

Lightweight text-data interchange format

Language independent (uses JavaScript syntax)

“Self-describing” and easy to understand

What is JSON?

Like XML

Plain text

“Self-describing" (human readable)

Hierarchical (values within values)

Can be parsed by JavaScript

Can be transported using AJAX

Unlike XML

No end tag

Shorter

Quicker to read and write

Can be parsed using built-in JavaScript eval()

Uses arrays

No reserved words

JSON Syntax

(subset of the JavaScript object notation syntax)

Data is in name/value pairs
Data is separated by commas
Curly braces hold objects
Square brackets hold arrays

JSON Name/Value Pairs

A name/value pair consists of a field name (in double quotes), followed by a colon, followed by a value:

```
"firstName" : "John"
```

This is simple to understand, and equals to the JavaScript statement:

```
firstName = "John"
```

JSON Data Types

Number (integer or floating point)

String (in double quotes)

Boolean (true or false)

Array (in square brackets)

Object (in curly brackets)

null

Examples

Objects

```
{ "firstName":"John" , "lastName":"Doe" }
```

Array

```
{
  "employees": [
    { "firstName":"John" , "lastName":"Doe" },
    { "firstName":"Jane" , "lastName":"Doe" },
    { "firstName":"John" , "lastName":"Smith" }
  ]
}
```

REST

REpresentational State Transfer

REST

Maps your CRUD actions to HTTP verbs

Action	Verb
Create	POST
Retrieve	GET
Update	PUT
Delete	DELETE

HTTP status codes

Code	Meaning
200	OK
201	Created
202	Accepted
204	No content
301	Moved permanently
302	Moved temporarily
400	Bad request
401	Unauthorized
403	Forbidden
404	Not found
500	Internal server error
501	Not implemented
503	Service unavailable

HTTP headers

Header Field	**Description**	**Example**
Accept	Content-Types that are acceptable for the response	text/plain, application/json, application/xml
Content-Type	The MIME type of the body of the request (POST and PUT)	application/x-www-form-urlencoded
Authorization	Authentication credentials for HTTP authentication	**OAuth** realm="http://sp.example.test/", oauth_consumer_key="0685bd9184jfhq22", oauth_token="ad180jjd733klru7", oauth_signature_method="HMAC-SHA1", oauth_signature="wOJIO9A2W5mFwDgiDvZb…
WWW-Authenticate	Authentication scheme	WWW-Authenticate: OAuth realm="http://sp.example.test/"

Why REST?

Simple, both conceptually and programmatically

Simpler and cleaner than SOAP

REST is the new black