05 – data formats

- XML
- JSON

- dret.net/lectures/xml/

Presentation of Web Content

- Recent trend:
  - from the document web to a web of data
- from structured representations of documents
  - to structured representations of data
  - in human readable and machine readable form
- Document Mark-up Languages
  - text plus metadata about the text
  - basic Idea: to separate structure (and format) from content of a text

Mark-Up Languages

- SGML (Standard Generalized Markup Language; 1986 -- approved as ISO international standard 8879)
  - Widely used: Defense, Aerospace, Semiconductor and Publishing industries
  - Very powerful and broad;
  - lack of stable tool support
  - 'Sounds Good Maybe Later'
- •HTML ('Killer--App' of the Web) –
  - Invented by Tim Berners Lee
  - HTML IETF in 1994, 1995 HTML 2.0 was published as IETF RFC 1866
  - Fixed vocabulary '(tag set')
- XML
  - development started in 1996 under auspices of W3C World Wide Web consortium
  - subset of SGML suitable for delivery of content over the web
- JSON / YAML
  - Data serialization language (not document centric)
  - Lessons learned from XM

HTML

- Core web technology, derived from SGML but much, much simpler
- Simple, fixed tag set
- Introduces anchor tag for hyperlinks
- Robust since tolerant
  - paragraph1 paragraph2
  - same as
  - paragraph1   paragraph2
- Based on 7--bit ascii
- Expresses structure and forma€ng information
  - <title> Structure Information </title>
  - Forma€ng information

HTML Limitation

- Limited, fixed tag set
  - How encode domain specific content (Chemistry, Math,...)?
- Mixes structure and format

XML Extensible Mark-up Language

- Subset of SGML for improved ease of implementation
- Meta-Language: allows defining mark-up languages
  - No pre-defined tag set
  - Purpose specific tags and document model is defined by a DTD or schema document
- Unicode character set
- W3C Recommendation (1998)

XML Suite of Standards

- XML Syntax (e.g. ‘closed tags: <para></para>)
  - 'well--formed' XML <=> syntactically correct
- XML Namespaces
  - – global semantic partitions of tag semantics (elements and attributes)
- XML Schema
  - Specifies allowed elements, their attributes, frequency, parent--child relationships etc
  - ‘valid’ XML <=> ‘semantically correct’ <=> conforms to a schema
- Xpath
  - Addressing specific information items in an XML document
  - Xpath 2.0 became a Recommendation on 23 January 2007.
- XSLT
  - language for transformation of XML documents
  - E.g. as a style--sheet: XML + XSLT → HTML for human consumption

XML Nested elements

Optional XML declaration (version of xml specification, encoding)

- <?xml version="1.0" encoding="UTF-8"?>
- <note>
- <recipients>
- <to>John1</to>
- <to>Jane</to>
- </recipients>
- <from>Jack</from>
- <heading>Reminder</heading>
- <body>Don't forget me this weekend!</body>
- </note>

root element

XML Elements

- Everything
  - from (including) the element's start tag
  - to (including) the element's end tag
- Contain a mix of:
  - other elements
  - Text
  - Attributes
-

XML Elements

- Attributes:
  - Name - value pairs that can be assigned to elements
  - provide additional information about elements
- Attribute specifications must be made within start tag of an element
- When to use elements, when a_ributes to represent information
  - Up to the designer;
  - consider:
    - An element can only have one a_ribute with the same name
    - An a_ribute cannot be further structured
    - A_ributes suitable for most iden6fiers and references, eg. id href
    - :
    - <img src="computer.gif">
    - <a href="demo.asp">

Well Formed" XML documents

- Well Formed XML
  - All XML Elements Must Have a Closing Tag
  - XML Tags are Case Sensitive
  - XML Elements Must be Properly Nested
  - XML Documents Must Have a Root Element
  - XML Attribute Values Must be Quoted
  - Entity References
    - Special characters (< >)

Namespaces

- Avoid name conflicts
- Precedes element names
- Conflict resolution
  - Use of ‘Qualified Names’ or Qnames
  - prefix_part : local_part
- root>
-
- <h:table xmlns:h="http://www.w3.org/TR/html4/">
- <h:tr>
- <h:td>Apples</h:td>
- <h:td>Bananas</h:td>
- </h:tr>
- </h:table>
-
- <f:table xmlns:f="http://w3schools.com/furniture">
- <f:name>
- African Coffee Table
- </f:name>
- <f:width>80</f:width>
- <f:length>120</f:length>
- </f:table>
-
- </root>

Namespaces

-
- Prefix must be associated with an URI:
  - Defined in xmlns Attribute
  - xhtml: http://www.w3.org/1999/xhtml
- Scope of name space declaration:
  - Begins at element for which declared
  - Applies to en6re content of that element (except when overwriWen)
- Variants:
  - For the whole document (root element)
  - For a child node & its content (allowing to use several name spaces in one document)
  - Default namespace (no prefix applied)
    - All descendent elements assumed to be from this namespace unless specified otherwise locally for a child el

Rationale for Namespaces

- How the web works:
  - Individually created documents linked by ambiguous references
- Towards a global database of knowledge?
  - Key: allow for distributed knowledge creation and lazy integration
- Problems:
  - Collisions (of how things are named)
  - Joins (how to link related content)
- Namespaces:
  - Build on URI notion
  - Uniquely qualify intra-document name collisions
  - Provide technology for cooperation

XML Meta Documents

- Express constraints on an xml document:
  - What element names and atributes to use
  - How often an element may occur
  - How elements are nested (complex elements)
  - What values attributes may have
  - What content elements may have... etc.
- Examples:
  - DTD (Document Type Defini6on developed for SGML)
  - XML Schema

Document Type Definition (DTD)

- Heritage from SGML
- Problems:
  - Not extensible:
    - can import declarations but not refine or inherit declarations
- Document must be valid according to 1 DTD:
  - cannot build on elements from different DTDs
- Limited support for name spaces
- Poor data typing; mainly intended for text
- Not defined using XML syntax hence cannot use XML tools

```
<!DOCTYPE note
[
<!ELEMENT note
(to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading
(#PCDATA)>
<!ELEMENT body (#PCDATA)>
]>
```

XML Schema

- defines elements that can appear in a document
- defines attributes that can appear in a document
- defines which elements are child elements
- defines the order of child elements
- defines the number of child elements
- defines whether an element is empty or can include text
- defines data types for elements and attributes
- defines default and fixed values for elements and attributes

XML Schemas

- Support data types
  - allowable document content
  - Correctness / restrictions of data
- Use XML syntax
- Are Extensible
  - Reuse your Schema in other Schemas
  - Create your own data types derived from the standard types
  - Reference multiple schemas in the same document

XML Schemas

- <?xml version="1.0"?>
- <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
- targetNamespace="http://www.w3schools.com" xmlns="http://www.w3schools.com"
- elementFormDefault="qualified">
- <xs:element name="note">
- **<xs:complexType>**
- <xs:sequence>
- <xs:element name="to" type="xs:string"/>
- <xs:element name="from" type="xs:string"/>
- <xs:element name="heading" type="xs:string"/>
- <xs:element name="body" type="xs:string"/>
- </xs:sequence>
- </xs:complexType>
- </xs:element>
- </xs:schema>

Xpath

- Syntax for defining parts of an XML document
- Uses path expressions to
  - Select nodes in a XML document
  - navigate in XML documents
- XPath is a major element in XSLT
- XPath is a W3C recommendation
- Xpath expressions are built from
  - Location steps (one or more)
  - Predicates (one or more)

Xpath Terminology

- Nodes
  - element, attribute, text, namespace, processing-instruction, comment, and document nodes.
- Atomic values
  - nodes with no children or parent

- Parent
- Children
- Siblings

- Ancestors
- Descendants

Xpath + XQuery

- path expressions
  - select nodes
  - / to define hierarchy
- Predicates
  - Selct specific node
    - Value, position, attribute
  - [ ]
- Wildcards *
- Multiple paths |

- Xquery functions
  - Retrieve nodes
    - From paths
  - Retrieve node data
- Use Xpath syntax
- doc("books.xml")/bookstore/book/title
- doc("books.xml")/bookstore/book[price<30]

XSLT

- XSLT Is an XML oriented programming language
  - Uses XML as its syntax
  - Uses Xpath to select subsets of an XML document
- Weakly typed
- Not designed for large programming tasks
- Standard language for XML--to--XML transformations
- Is a func6onal programming language (hard to get used to for people trained in procedural languages)
- Functions are first-class citizens, supports passing
  - functions as arguments to other func6ons
  - Iteration usually done by recursion

XSLT

- A transforma6on in the XSLT language is expressed in the form of an XSL stylesheet
  - root element: <xsl:stylesheet>
  - an xml document using the XSLT namespace, i.e. tags are in the namespace http://www.w3.org/1999/XSL/Transform
- The body is a set of templates or rules
  - The ‘match’ attribute specifies an Xpath of elements in source tree
  - Body of template defines contribution of source elements to result Tree
- A template rule associates a pattern with a sequence constructor
  - The pattern matches nodes in the source document “for nodes satisfying pattern X do this”
  - The resulting nodes and atomic values can be used to produce parts of a result tree

- <xsl:template match=“/”>
- <html>
- ...
- <xsl:for--each select=“actor/name[@gender=“female”]>
- <h3>Name:</h3>
- <xsl:value--of select=“.”>
- <xsl:for--each>
- </html>
- <xsl:template>

JSON

- “JavaScript Object Notation (JSON) is a lightweight, text-based, language-independent data interchange format. […] JSON defines a small set of formatting rules for the portable representation of structured data.”
  - [RFC 4627 http://tools.iey.org/html/rfc4627]
- Popularization and formal specification defined by Douglas Crockford (senior JavaScript architect at PayPal
- JSON Homepage: http://www.json.org/

JSON

- Open standard for data interchange
  - Text-based & human readable
  - Machine readable and easy to parse
  - Not a mark-up language but 'self-descriptive'
- Not extensible (different from XML)

```
{"employees":
  [
  {"firstName":"John",
   "lastName":"Doe"},
  {"firstName":"Anna",
   "lastName":"Smith"},
  {"firstName":"Peter",
   "lastName":"Jones"}
  ]
}
```

JSON

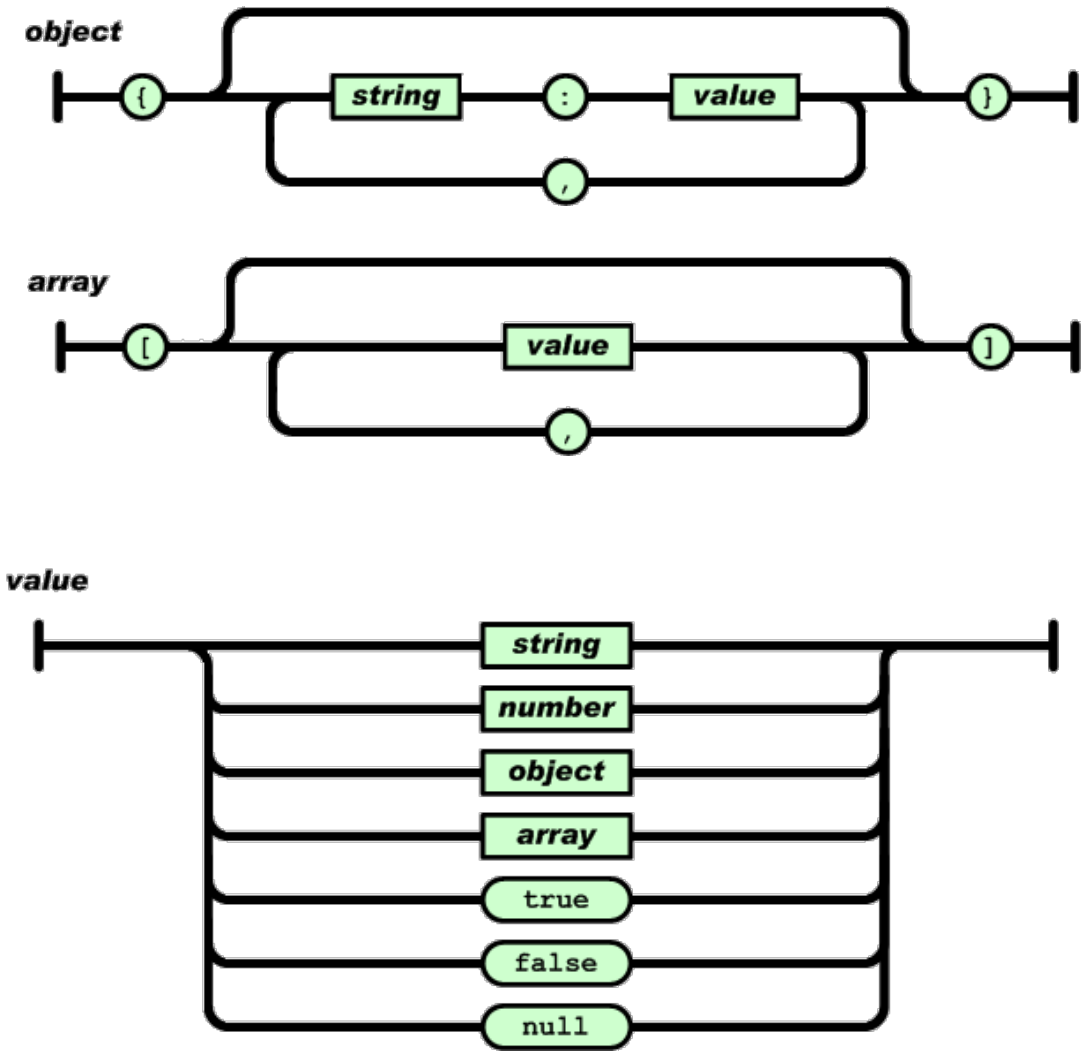
- Programming language independent
- Maps to two universal structures:
  - 1) An unordered collec6on of name value pairs:
    - `{"firstName":"John", "lastName":"Doe"}`
    - In other languages:
      - Object, record, struct, dic6onary, hash table, keyed list, or associa6ve array
  - 2) An ordered list of values:
    - `["Monday", "Tuesday", "Wednesday"]`
    - in other languages:
      - array, vector, list, or sequence

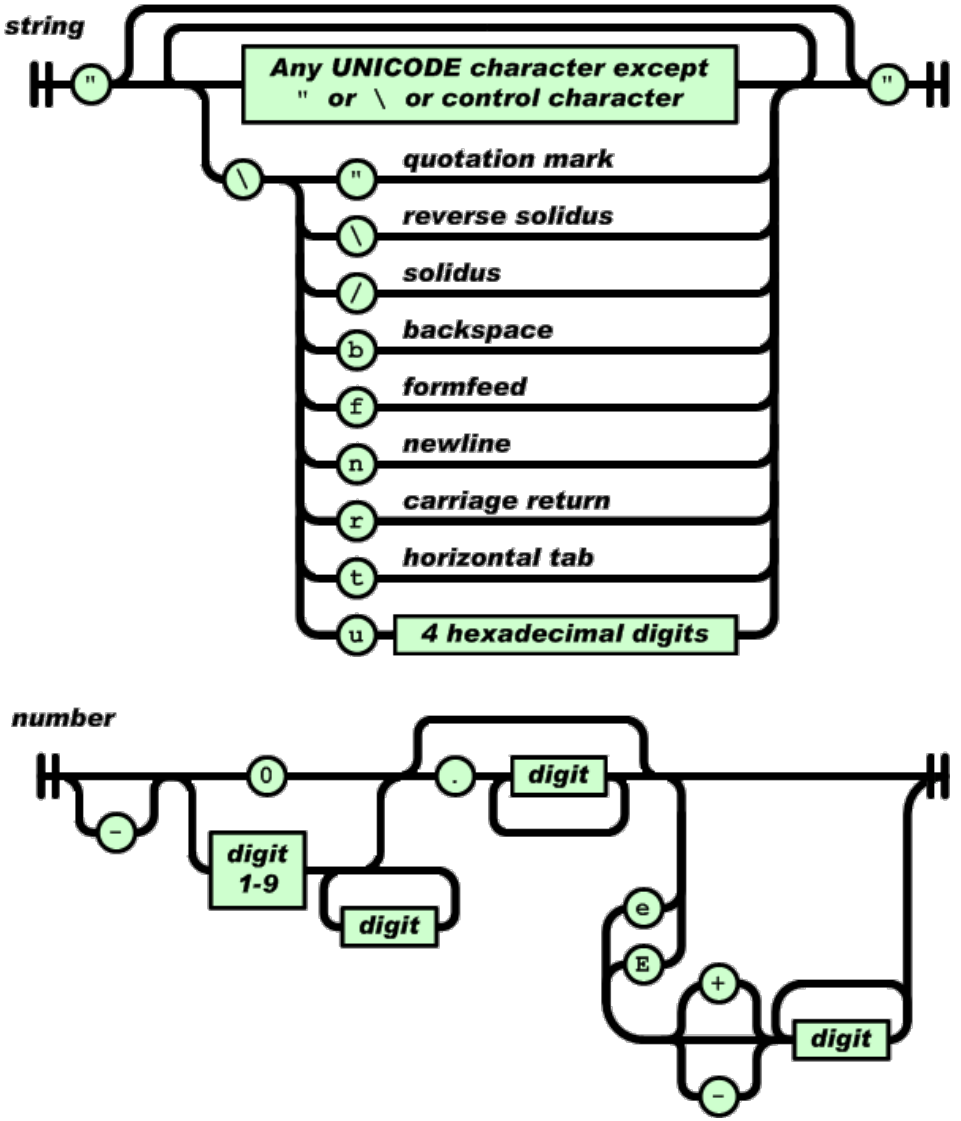
JSON Syntax

- Primitive types:
  - Strings
  - Numbers
  - Booleans (true/false)
  - Null
- Structured Types
  - Objects
  - Array

- JSON mime type:
  - applicaOon/json
- formally defined in
  - RFC 4627

JSON Syntax


object
{
string
:
value
}
,
array
[
value
]
,
value
string
number
object
array
true
false
null


string
"
Any UNICODE character except
" or \ or control character
"
\
"
quotation mark
\
reverse solidus
/
solidus
b
backspace
f
formfeed
n
newline
r
carriage return
t
horizontal tab
u
4 hexadecimal digits
number
-
0
digit
1-9
digit
.
digit
e
E
+
-
digit

XML vs JSON (json.org/xml.html)

- Simplicity
  - XML is simpler than SGML,
  - but JSON is much simpler than XML.
  - JSON has a much smaller grammar and maps more directly onto the data structures used in modern programming languages.
- Extensibility
  - JSON is not extensible because it does not need to be.
  - JSON is not a document markup language, so it is not necessary to define new tags or attributes to represent data in it.
- Interoperability
  - JSON has the same interoperability potential as XML.
- Openness
  - JSON is at least as open as XML, perhaps more so because it is not in the center of corporate/political standardization struggles.

XML vs JSON (json.org/xml.html)

- XML is human readable
  - JSON is much easier for human to read than XML. It is easier to write, too. It is also easier for machines to read and write.
- XML can be used as an exchange format to enable users to move their data between similar applications
  - The same is true for JSON.
- XML provides a structure to data so that it is richer in information
  - The same is true for JSON.
- XML is easily processed because the structure of the data is simple and standard
  - JSON is processed more easily because its structure is simpler.

XML vs JSON (json.org/xml.html)

- There is a wide range of reusable software available to programmers to handle XML so they don't have to re-invent code
  - JSON, being a simpler notation, needs much less specialized software. In the languages JavaScript and Python, the JSON notation is built into the programming language
- XML separates the presentation of data from the structure of that data.
  - JSON structures are based on arrays and records. That is what data is made of. XML structures are based on elements (which can be nested), attributes (which cannot), raw content text, entities, DTDs, and other meta structures.
- Exchange format
  - JSON is a better data exchange format.
  - XML is a better document exchange format. Use the right tool for the right job.
- Many views of the one data
  - JSON does not provide any display capabilities because it is not a document markup language.

XML vs JSON (json.org/xml.html)

- Self-Describing Data
  - XML and JSON have this in common.
- Complete integration of all traditional databases and formats
  - (Statements about XML are sometimes given to a bit of hyperbole.)
  - XML documents can contain any imaginable data type - from classical data like text and numbers, or multimedia objects such as sounds, to active formats like Java applets or ActiveX components.
- JSON does not have a <[CDATA[]]> feature,
  - so it is not well suited to act as a carrier of sounds or images or other large binary payloads. JSON is optimized for data. Besides, delivering executable programs in a data-interchange system could introduce dangerous security problems.
- Internationalization
  - XML and JSON both use Unicode.
- Open and extensible
  - XML's one-of-a-kind open structure allows you to add other state-of-the-art elements when needed. This means that you can always adapt your system to embrace industry-specific vocabulary.
  - Those vocabularies can be automatically converted to JSON, making migration from XML to JSON very straightforward.

XML vs JSON (json.org/xml.html)

- XML is easily readable by both humans and machines
  - JSON is easier to read for both humans and machines.
- XML is object-oriented
  - Actually, XML is document-oriented.
  - JSON is data-oriented. JSON can be mapped more easily to object-oriented systems.

JSON and Python

- 18.2. json — JSON encoder and decoder
  - exposes an API familiar to users of the standard library marshal and pickle modules.
  - Dict, list, string number, boolean, null
- import json
- json.dumps/json.dump
  - serialize
  - Generate/print string representations of object
- json.loads/json.load
  - deserialize
  - Generate python object from string/file

JSON and AJAX

```
var my_JSON_object;
var http_request = new XMLHttpRequest();
http_request.open("GET", url, true);
http_request.onreadystatechange = function () {
    var done = 4, ok = 200;
    if (http_request.readyState === done &&
            http_request.status === ok) {
      my_JSON_object =
            JSON.parse(http_request.responseText);
    }
};
http_request.send(null);
```