

# XML-powered Exhibit: A Case Study of JSON & XML Coexistence

- Chimezie Ogbuji (chee-meh)

[ogbujic@ccf.org](mailto:ogbujic@ccf.org) / [chimezie@gmail.com](mailto:chimezie@gmail.com)

(Cleveland Clinic Foundation)

[http://metacognition.info/Publications/xml\\_exhibit](http://metacognition.info/Publications/xml_exhibit)

# Background and Motivation

- Longitudinal patient data in a Content Management System (**CMS**)
- **XForms** 1.1 (formsPlayer) met our dynamic data entry needs
- Needed a light framework for read-only browsing of patient data
- Built a prototype in 30 minutes

# XML / JSON: A Percieved Divide

- **XML**
  - Syntactic expressiveness
  - Datatyping (strong typing)
  - Formal querying language
- Javascript Object Notation (**JSON**)
  - Simple (less verbose)
  - Pervasive support
  - Low-cost native implementation

# XML on the Web: The Vision

- Rich Web Application Backplane (**RWAB**)
- **XML** data model / binding mechanism
- Comprehensive submission mechanism
- Event-based interaction
- Independent set of controls / widgets
- Declarative approach (**XForms**)
- Imperative approach (**AJAX**)

# Developing an Exhibit Page

- Describe your **JSON** 'schema'
- Select your widgets / lenses
  - Tabular lens
  - Tile-based lens
  - Timeline lens
  - Map lens
- Embed the widgets within vanilla **HTML**
- Point the document at a data source

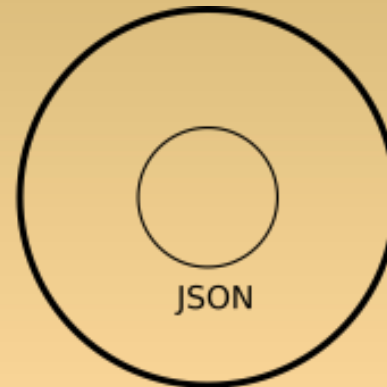
# Key (Architectural) Divergence

- **RWAB** is a *revolutionary* approach to enabling *R/W* web applications
- Exhibit is an *evolutionary* approach to enabling *read-only* web applications
- A roadmap for a future **HTML** specification *must* consider the former
- The latter is a quick, well-engineered hack for the interim

# Is there an overlap?



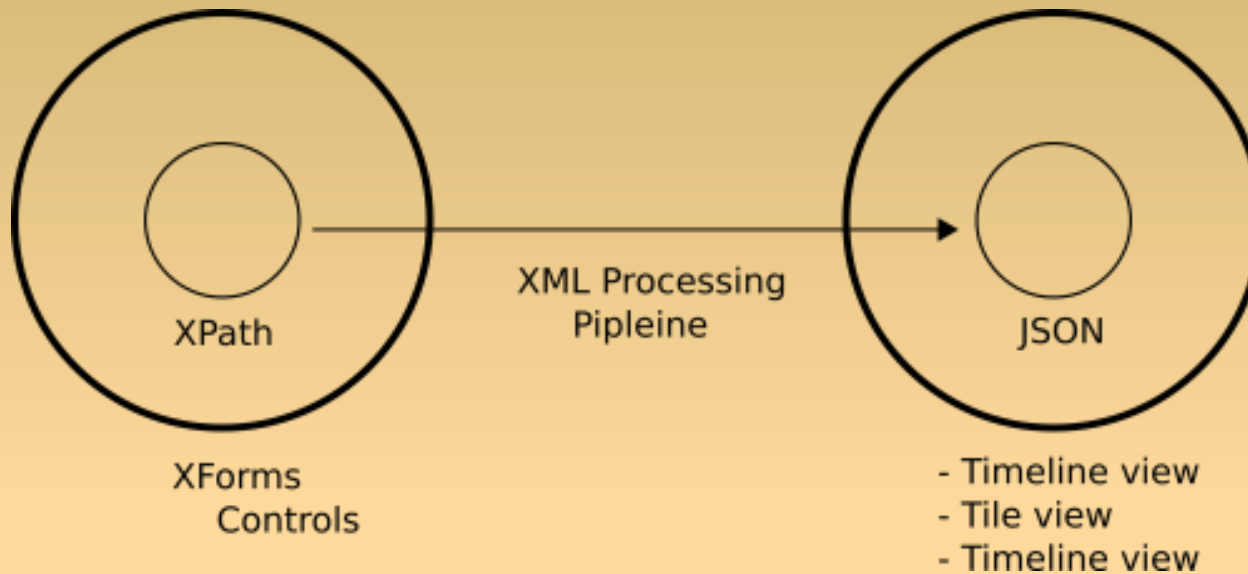
XForms  
Controls



- Timeline view  
- Tile view  
- Timeline view

- Model View Control approach to UI

# Using the Overlap to Our Advantage



- With **XML** at the start: *built-in* future-proofing
- (Functional) **XML** pipelines can be built once for all instances of a schema

# Atom as a Usecase

- Well-designed
- Rich set of metadata (facets)
- Simple
- Supports a popular niche: content syndication
- Has an associated publishing protocol: Atom Publishing Protocol (**APP**)
- Promise of massive deployment (consider **GData**)

# Sample Atom Entry

- Excerpt from Norm Walsh's Atom feed

```
<atom:entry>
  <atom:title>Modern laptops</atom:title>
  <atom:link
href="http://norman.walsh.name/2007/04/03/modernLaptops"
type="text/html"
    rel="alternate"/>
  <atom:id>http://norman.walsh.name/2007/04/03/modernLaptops
</atom:id>
  <atom:published>2007-04-03T11:19:14Z</atom:published>
  <atom:updated>2007-04-03T20:15:56Z</atom:updated>
  <dc:subject>SelfReference</dc:subject>
```

# Mapping from Atom to JSON

- Capture most if not all of the Atom metadata
- **XSLT** is perfect for this
- First, we need a convention
- Turtle has syntactic *sugar* similar to **JSON**
- Start with an existing Atom-to-Turtle pipeline as the basis

# A JSON Convention

- Non-repeating properties are flat key/value pairs
  - *atom:updated*, *atom:published*, *atom:title*, etc..
- Properties that repeat are modelled as homogeneous lists
  - *atom:category* and *dc:subject*

# Atom in, JSON out

- The same Atom entry as **JSON**

```
{
  items: [
    { type: "Entry",
      label: "Modern laptops",
      id: "http://norman.walsh.name/2007/04/03/modernLaptops",
      uri: "http://norman.walsh.name/2007/04/03/modernLaptops",
      published: " 2007-04-03T11:19:14Z",
      updated: "2007-04-03T20:15:56Z",
      source: "http://norman.walsh.name/atom/whatsnew-fulltext.xml",
      author: "Norman Walsh",
      subject: ["SelfReference"],
      category: []
    },
    ... other entries ..
  ]
}
```

# Exhibit Schema for Atom / JSON

- Exhibit allows authors to supply a *schema* for the data source
- Gives Exhibit cues on how to render certain properties (timeline view)
- Similar to how **XForms** allows **XSD** datatypes to determine an appropriate widget

# An Exhibit Schema

```
{
  types: {
    "Entry": {
      pluralLabel: "Entries",
    },
    properties: {
      "subject": {
        valueType: item ,
        reverseLabel: 'subject of',
        pluralLabel: 'subjects'
      },
      "category": {
        valueType: item ,
        reverseLabel: 'category of',
        pluralLabel: 'categories',
      },
      "updated": { valueType: date },
      "published": { valueType: date },
    },
    ...
  }
}
```

# The Transform

- `xsl:stylesheet/xsl:output/@method='text'`
- `xsl:stylesheet/xsl:output/@media-type='application/json'`
- *A hard-coded JSON* schema
  - Inlined with the data source
- An **atom:entry** template

# The Exhibit UI Template

- /html/head/link[@href = '.. JSON source ..' and @rel='exhibit/data']
- exhibit-browse-panel (faceted browsing)
- exhibit-control-panel
- exhibit-view-panel
  - \*[@ex:role='exhibit-view' and ..]
  - table[@ex:role='exhibit-lens']

# Screenshot

**Browsing Atoms**

4 Entry filtered from 41 originally ([reset](#))

sorted by: [published](#), [author](#), and [source](#); then by... •  grouped as sorted •  show duplicates

- Modern laptops**  
Norman Walsh, [file:///home/chimezi... 2F03%2FmodernLaptops](#).  
Published: [2007-04-03T11:19:14Z](#)  
Updated: [2007-04-03T20:15:56Z](#)  
Source: [http://norman.walsh... hatsnew-fulltext.xml](#)
- Every street in Amherst (second attempt)**  
Norman Walsh, [file:///home/chimezi... F04%2F02%2FamherstMA](#).  
Published: [2007-04-02T17:35:34Z](#)  
Updated: [2007-04-02T17:47:44Z](#)  
Source: [http://norman.walsh... hatsnew-fulltext.xml](#)
- Feeling Feisty**  
Norman Walsh, [file:///home/chimezi... 7%2F04%2F01%2Ffeisty](#).  
Published: [2007-04-01T16:32:35Z](#)  
Updated: [2007-04-02T17:00:17Z](#)  
Category(ies): [linux](#) and [ubuntu](#)  
Source: [http://norman.walsh... hatsnew-fulltext.xml](#)
- Take it easy, take it slow**  
Norman Walsh, [file:///home/chimezi... 007%2F03%2F26%2Feasy](#).  
Published: [2007-03-26T20:37:07Z](#)  
Updated: [2007-03-27T12:20:07Z](#)  
Category(ies): [thinkpad](#)  
Source: [http://norman.walsh... hatsnew-fulltext.xml](#)

# Putting it Together: Planet Atom

- <http://planetatom.net/exhibit>
- <http://svn.platom.python-hosted.com>
  - `/trunk/media/templates/atom2json.xsl`
- <http://planetatom.net/output/index.js>
- .. in addition ..
- <http://planetatom.net/output/index.rdf> (AtomOWL)
- <http://planetatom.net/output/index.xhtml>  
(XHTML+RDFa+GRDDL)

# Final Thoughts

- Very simple to put together
- Clean MVC separation
- Pluggable on XML systems via pipelining
- Cross-platform support
- Technological camp warfare does a disservice to innovation: **XHTML** vs **HTML**, **XML** vs **RDF**, **XML** vs **JSON**, etc..