MicroXML: Who, What, When, Where, Why

Who, When, Where

- Two blog posts by James Clark in December 2010
  - Defining MicroXML as a concept
  - Specifying what's in, what's out, and a grammar
- About ten other blog posts by various people, including me
- There is now a W3C Community Group, with Uche Ogbuji as chair and James and I as co-editors
- *http://www.w3.org/community/microxml*

Why: tentative design goals

- The syntax of MicroXML shall be a subset of XML 1.0
- MicroXML shall define a data model as well as a syntax
- MicroXML shall be dramatically simpler than XML as regards its specification, syntax and data model
  - This applies even though XML tools already exist
  - SGML tools exist, but XML was not pointless
  - “To boldly go where no XML has gone before”

Why: tentative design goals

- MicroXML shall be designed to complement rather than replace XML, JSON and HTML
- MicroXML shall support the needs of documents, in particular mixed content
- MicroXML shall support Unicode
- MicroXML shall support the use of text editors for authoring
- MicroXML shall be able to straightforwardly represent HTML

Two example documents

```
<doc><w>Hello</w> <w>world</w>!</doc>

<sp who="Faust" desc='leise'
  xml:lang="de">
   <l>Habe nun, ach! Philosophie,</l>
   <l>Juristerei, und Medizin</l>
   <l>und leider auch Theologie</l>
   <l>durchaus studiert mit heißem
  Bemüh'n.</l>
</sp>
```

What's in like Flynn

- Elements and attributes
- ASCII name characters
- Unicode character content and attribute values
- A single normative data model
- Hex character references
- `< > & &quot; &apos;`

What's out beyond doubt

- Mandatory draconian error handling
  - Tools can be either draconian or permissive
- The whole DTD ball of mud
- Encodings other than UTF-8
- XML declaration
- Attribute value normalization
- CDATA sections

No verdict yet

- Unicode names for elements/attributes
  - Old XML, or XML 1.0 Fifth Edition / XML 1.1?
- empty tags
- comments
- bare DOCTYPE declarations
  - `<!DOCTYPE html>`
- `>` in attribute values (for Canonical XML compatibility)
- decimal character references

Data model

- Like the XML Infoset or the XPath Data Model
  - But as simple as possible
- Just a tree of element objects
  - No document, attribute, entity, etc. etc. info items
- MicroXML processors:
  - MUST expose the whole data model
  - MAY do so by constructing a tree, pushing events, letting events be pulled, or otherwise

Element objects

- An element object has three parts:
  - Name (a string)
  - Attributes (a map from strings to strings)
  - Children (a sequence of elements and/or strings)
- Element objects form a tree mirroring the doc
- Consecutive string children are consolidated
- Namespaces don't affect the data model
  - Namespace declarations are just attributes

Processing instructions

- No PIs?
- PIs in the syntax but not the data model?
- PIs in the prolog but not the epilog?
- PIs that look like start-tags only?

Prefixes: three possibilities

- No prefixes except `xml:`
  - Elements can be namespaced by convention, using `xmlns` attributes
- Prefixes on attributes only
  - Attributes can be namespaced too, using `xmlns` attributes
- Prefixes on both attributes and elements

Processing instructions

- The `xml-style` and `xml-model`PIs in the document prolog are standardized by W3C
- The `php` PI is not, but is very common
- Adding PIs to the data model would greatly complicate it:
  - Without PIs: just element objects
  - With Pis: document, element, and PI objects

xml:*

- Four attributes with the `xml:` prefix:
  - `xml:space` for application whitespace handling
  - `xml:lang` for language identification
  - `xml:base` for base URI
  - `xml:id` for element identification
- Semantics are the same as for their full XML equivalents

MicroXML and JSON

- MicroXML is both richer and poorer than JSON as a data format
- Interoperability is a Good Thing
- Convert any JSON to MicroXML and back
- Convert any MicroXML to JSON and back
- Convert MicroXML to clean JSON and back
  - There's more than one way to do it
  - No consensus in the JSON and XML communities

MicroLark

- The first parser just for MicroXML
  - The name alludes to Tim Bray's original 1998 XML parser Lark, but no code is shared
- Written in Java by John Cowan
  - Open source (Apache 2.0 license)
- Provides pull parser, push parser, and element tree constructor in about 1/3 the LOCs of Lark
  - Applications can provide their own subclasses of `Element` to customize the tree

MicroLark Iterators

- “Poor man's XPath” based on chained method calls
- For “/html//p[3]” write:

```
new ElementIterator(document)
      .root("html")
      .descendants("p")
      .positionFilter(3)
```

- Then iterate on the result to fetch the elements

MicroLark Validators (not yet)

- ID/IDREF/IDREFS validator
  - Specify namespace/element/attribute triples and namespace/attribute pairs via Java API
  - Retrieve elements by ID
  - Validate IDs for uniqueness
  - Validate IDREF and IDREFS attributes against IDs
- Examplotron validator
  - Validate a document against a parsed Examplotron schema

Examplotron

- Defined for XML by Eric van der Vlist in 2003
- Examplotron schemas are just sample documents
- Element content models are deduced from content
- Elements in the schema are required by default
- Attributes in the schema are permitted but not required in the document

Examplotron

- `eg:occurs` indicates repetition of an element
  - Can be “*”, “+”, “?”, “.” (once), “-” (zero times)
- `eg:content` indicates specific content model
  - ordered, unordered, or mixed-content child elements
  - a few simple types: string, integer, decimal, double, date, dateTime, boolean, base64Binary
- More at *http://examplotron.org*

When: Now!

- Join the community group
  - As an individual
  - As a representative of your organization (which does not need to belong to W3C)
- Post to the mailing list
- Add pages to the wiki
- Talk to your friends
- Make MicroXML a reality