MIME and the Web

Larry Masinter

For W3C TAG meeting

Jan 4, 2012

MIME gives the **web persistent names** for **languages**

- *MIME has a lot of other features*
- What do these terms mean?
  - "persistent"
  - "name for"
  - "language"
- *How does MIME do it? Where does it break? What can we fix?*
- Some very general problems
- Focus on MIME-specific issues of general problems

What is a **language**?

- A language is a way of giving meaning to data
  "Given some data, what does it mean?"
- A "**File format**": a kind of language
  they're just binary languages
- Languages have **syntax** & **vocabulary**
- Languages usually use other languages
  - **protocol element** (a little language)
  - **abstract language** (defined in terms of structure)
  - **layer** (SVG on XML on Unicode)
- "URI" is a language, JavaScript, CSS are languages

What is a name?
How does MIME name languages?

- A name is protocol element
  - with some structure
  - used in other languages, protocols, apis, interfaces
  - Which has some meaning
- Meaning of MIME types
  - “which language should be used to interpret this data”

Persistent names

- languages change: how can names be **persistent?**
- With no evolution, updates, extensions to languages used in the web: no problems

CORE

- ***How do languages change?***
- ***What are problems with MIME during evolution?***

Languages and Implementations

- Languages (as with protocols, protocol elements, file formats, APIs) are used between systems to communicate
- Systems using a language should mean the "same" thing
- Need agreement between the systems that are communicating

Interoperability is a property of implementations, not specifications

Languages and Specifications

- **Specifications** are documents that describe a language and rules for implementations
  - How implementations should “understand” the language/API/protocol/protocol element’

Implementations to guide and validate single-user

- *Many* specifications used to define a single language
- What happens as those evolve?

Standards for Languages

- **Standards** represent agreements among implementations (in the form of a specification)

Persistent names for languages

- What is persistent about the name for a language?
- What is it that the name of a language identifies?
- How do languages evolve, grow, change over time?
- How can the name be persistent when the meaning changes?

Persistence and Evolution

- When a language evolves, it keeps its name
- A new language, even if it isn’t very different, would get a different name

Wait…

- How do languages evolve?
- What happens to systems that use those names with evolving meaning?

Talking about evolution, versioning

- These are really hard problems to model
- TAG has foundered in these waters before
- Let’s try to restrict the topic to “enough to solve MIME’s problems”

Everything should be as simple as possible, but no simpler…

- Focus on how languages evolve and names for languages track
- Giving version numbers: allow persistance and also new names

“language” is over-simplification

- Languages (file formats, protocols, protocol elements) are defined in terms of others
- Complex structure of interrelationships between components
- Each component can evolve independently

Implementations evolve

- The language is “as spoken”, not “as defined”
- Concrete and abstract languages
- References to other specification
- Syntax and parsing

specifications describe Languages

- References in specifications: how do rules apply when referenced specification is updated
- Editions, version numbers

More complexities

- Content negotiation
- Polyglot
- “multi-view”

Way of managing names needs to account for complexity

- With overlap, subsets, evolution of languages, multiple implementations
- Need to account for these for names to be persistent

Registry

- A way of naming something
  - Organization to manage registry
  - Key role of registry is to manage updates
    - When there are compatibility requirements
    - When there are requirements

MIME registry

- Key features to deal with permanence:
  - Compatibility rule
  - Change controller
  - Review process
  - Pointer to specification
- Email rules required backward compatibility (old valid content should not become invalid)
- Web needs additional compatibility rules
  - Sniffing, forward compatibility
- Does the web need the rules?
  - After all, lots of unregistered types work “fine”
  - Names are used for languages, not for specifications

Persistent name problems

- Forking (HTML)
- Versioning (javascript)
- References
- Compound languages (HTML + RDFa/lite + SVG + MathML)
- Layering

- Generalization: other “persistent names”:
  - Charset (addition of Euro)
  - Other web names (codes, URLs)

Content negotiation

  - *Which languages do you understand?*
  - *Which languages can you speak to me?*
- MIME types don’t help much
  - Wrong level of granularity
  - Ambition of reader implementers doesn’t match conservative requirements of senders

Persistent names and versions

- “version” parameter requires future proofing
- In-band version identifiers might be preferable
  - Except for “quirks mode” failure cases
- Users would like “version of language”
- Best a specification can give is “version of specification”
- Specifications and languages often don’t evolve in sync

Important use cases for this work

- HTML and versions
- JavaScript and versions
- Sniffing
- Charsets and “willful violation”
- Privilege upgrade
- CSS and vendor prefixes
- XML languages and versioning
- SVG and XML 1.1
- User-Agent (name for implementation)

TAG work on MIME and web

- Describe the “real” web
  - what’s really happening, not unattainable goal
- Not every case is the same
  - JavaScript vs. HTML vs. CSS
- Manage extensibility of languages by extending vocabulary, not syntax
  - Upgrades to processors other than end consumers are much easier
- MIME types without plugins in the web?

Success Criteria?

- Satisfy use cases for MIME types for
  - HTML, CSS, JavaScript, SVG, XHTML
- *In a way that we get community consensus, not just TAG agreement*
- Resolve differences between past findings and policies & current directions

Questions for TAG: scope too broad? How to narrow?

- Can we revisit versioning with more modest goals and make progress?
- Can we take on "persistent reference to language" as a focus for persistent names?
- Can we complete work on specification reference?