Working with JSON-LD Best practices and improvements

Kyle Den Hartog

@id – the term that makes JSON a graph

- This term is used to denote the "identifier" of a node within a graph (view in "visualized" mode) - https://tinyurl.com/yz8hmgje
- The id term in the JSON-LD document is different then the id property in a context. Careful here, they mean similar things but the effects when using the data is different.

@type – the "descriptor" term

- In graph view this is like a "label" to a node of the graph
- I like to think of this like an interface in typescript
- It describes which terms are valid in JSON-LD, otherwise they're probably going to get "dropped" when signing
  - This is a "feature" of JSON-LD signatures
  - Intended for safety, but often times leads to headaches
  - Some libraries have moved away from this method
    - https://github.com/timothee-haudebourg/json-ld/issues/13

@vocab property – the catch all term

- Treat this property as a guardrail
- It's useful when learning but you probably don't want to be using this in production
- If you are using it in production to prevent dropping terms be explicit
- Best practice:
  - Use "https://w3id.org/undefinedTerm"

JSON-LD Frames – where things get real

- We recognize these are difficult to work with
- I haven't quite figured out what the best practices for this are yet
- Probably need to be careful with taking whatever is handed to you by the requestor and make sure to validate it

Contexts – the tricky magic that makes it work

- Think of this like a map function that maps alias properties to a fully qualified URI
- Good practices around context definitions are important
- Be careful to make sure the locally cached version matches the remote document
  - Wallets probably need to lean on remote versions so good context hygiene is important
- The remote document should be append-only and versioned if you're signing documents that depend on them
- Availability of the remote context can impact things
- Schema.org is a shaky ontology to build on be careful of semantic drift

@id in contexts

- "@id": "https://schema.org/image"
  "image": "https://schema.org/image"
- Both are ways to set the expanded IRI of a term
- changing this after signing documents that depend on that term will break signatures (If they use URDNA2015)
- Required to be a IRI (URI with international character set)
- Best practice is to have this as a URL to a human readable definition of the term
- Usage of hashlinks in IRIs is an improvement I've been working on
  - demo later
- "id": "@id" <- This is magic that makes JSON without "@" work properly

@type in contexts

- In general, this has only been used to define datetime in VCs
- There's a lot of power that can be leveraged here that's untapped
- "type": "@type" <- magic that maps type to @type reserved term
- I don't have enough implementor experience with this yet

@protected – the guard term

- You probably have run into this term and been confused why it's causing problems
- This is a new V1.1 term which is used to protect from term redefinition
- Order of context URLs in the list matters here
- Latest jsonld.js DB libraries should tell you which terms fail now

Context improvements

- Idea for new "@definition" term – Meant to incorporate the human readable definition (semantics) into the machine-readable documents
- Combined with Hashlinks (a draft spec not fully stable yet) on the "@id" terms and this gets immutable terms that intentionally "break" when the semantics change
- Demo
  - https://github.com/kdenhartog/context-integrity