# Practical Semantics

- @jasnell everywhere
- Technical Lead for Node.js @ IBM since 2015-01-01
- I've been doing open source and open standards for 15 years at least
- Helped write a few RFCs and open specs
    - PATCH, Prefer, Atompub, Atom Threading, Activity Streams, JSON Merge Patch, Variety of link rels.

let’s define some things.

Practical

“Practical” - *of or concerned with the actual doing or use of something rather than with theory and ideas.*

# Semantics

“Semantics” - “*the branch of linguistics and logic concerned with meaning…*”

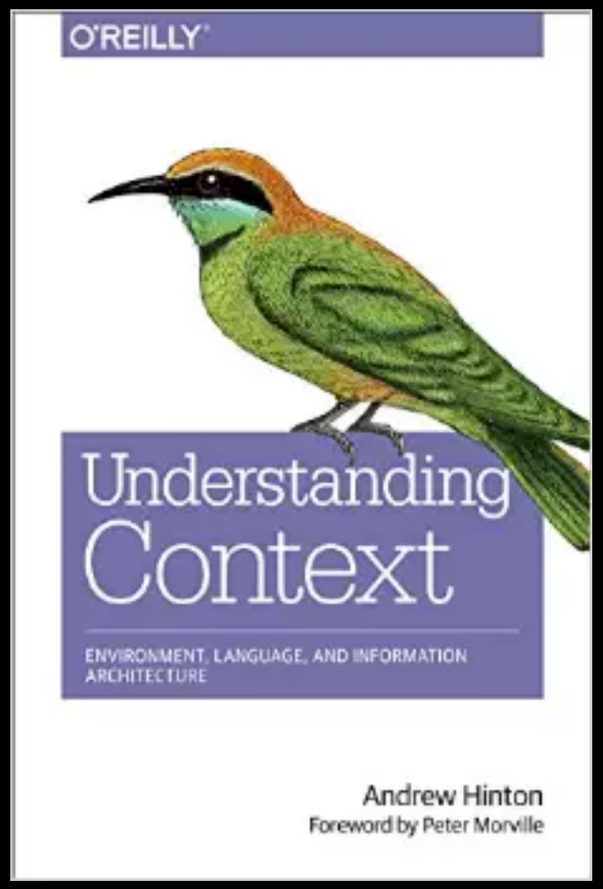
# Linked Data

“Linked Data” - essentially… hypermedia.

data that links to data.

Is the Semantic Web Practical?

Before we continue…




http://www.amazon.com/Understanding-Context-Environment-Information-Architecture

Let’s consider npm

- As of earlier today (2015-09-18T11:15:00-05:00)
  - 185,664 packages
  - 101,904,649 downloads in the *last day*
  - 2,507,459,755 downloads in the last month.

A “few” people are using npm.

lodash - 18 million+ downloads in the past month
(14,000+ dependent modules in npm)

express - 3 million+ downloads in the past month
(at *least* 6370+ dependent modules in npm,
not counting uses not tracked by npm)

chalk - 13 million+ downloads in the past month

(at least 5200+ dependent modules in npm)

(chalk colorizes and styles strings… that's all it does)

If the semantic web makes so much sense;
if it's so powerful and useful…

then it should show up strong on npm's stats…

right?

jsonld - downloaded 3,650 times in the last month.
(~47 dependent modules in npm)

n3 - downloaded 1,474 times in the last month.
(23 dependent modules in npm)

rdflib - downloaded 168 times in the last month
(6 dependent modules in npm)

Why are the numbers so low?

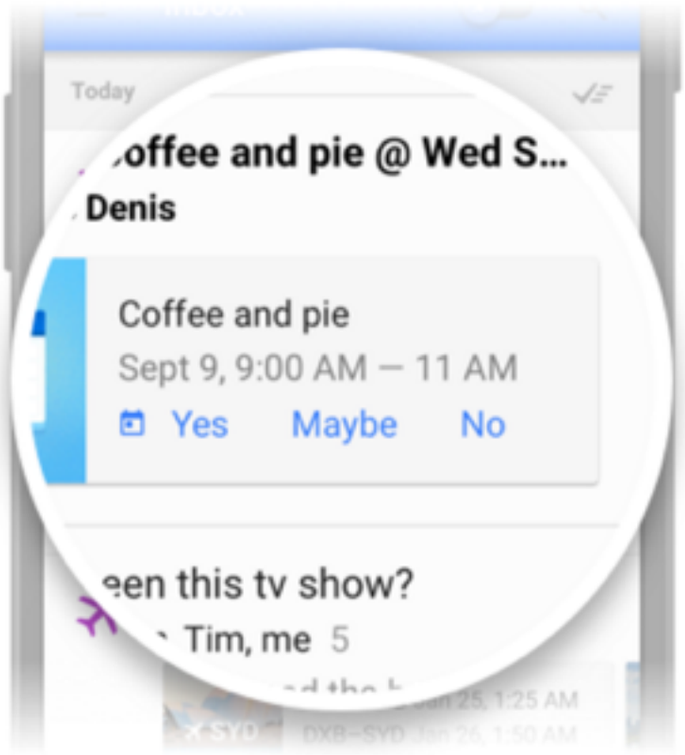
Is the Semantic Web Practical?

Let’s consider schema.org

- schema.org/Thing
  - “Between 100,000 and 250,000 domains”
- schema.org/CreativeWork
  - “Between 250,000 and 500,00 domains”
- schema.org/Organization
  - “Over 1,000,000 domains”
- schema.org/Product
  - “Over 1,000,000 domains”

# Rsvp Action

Use to declare an event with an RSVP action. An Event card will appear next to the email, including buttons for RSVP-i
attendance in the event.


RSVP actions in Gmail and in Inbox

# Event with RSVP Action

You may define RSVP buttons by declaring an Event with RsvpActions in the markup:

JSON-LD MICRODATA

```
<script type="application/ld+json">
{
  "@context": "http://schema.org",
  "@type": "Event",
  "name": "Taco Night",
  "startDate": "2015-04-18T15:30:00Z",
  "endDate": "2015-04-18T16:30:00Z",
  "location": {
    "@type": "Place",
    "address": {
      "@type": "PostalAddress",
      "name": "Google",
      "streetAddress": "24 Willie Mays Plaza",
      "addressLocality": "San Francisco",
      "addressRegion": "CA",
      "postalCode": "94107",
      "addressCountry": "USA"
    }
  },
  "potentialAction": [
    {
      "@type": "RsvpAction",
      "handler": {
        "@type": "HttpActionHandler",
        "url": "http://mysite.com/rsvp?eventId=123&value=yes"
      },
```

What’s going on here?

semantic data is easy to *produce*.

it’s not so easy to *consume*.

search engine's like google, yahoo, bing and yandex provide incentives for *producing* semantic data.

where are the incentives for consuming it?

where are the tools that make it consuming it *practical*?

The tools *do* exist.

But they are not being used.

Why?

affordance: “is often taken as a relation between an object or an environment and an organism, that affords the opportunity for that organism to perform an action.[1][2] For example, a knob affords twisting, and perhaps pushing, while a cord affords pulling” - wikipedia

When you look at something, do you know what it’s for? Do you understand how to use it?

When you consider the “Semantic Web”,
do you know what to do with it?

do you understand how to use it?

schema.org

schema.org

Search

About Schemas Documentation

# Welcome to Schema.org

Schema.org is a collaborative, community activity with a mission to create, maintain, and promote schemas for structured data on the Internet, on web pages, in email messages, and beyond.

Schema.org vocabulary can be used with many different encodings, including RDFa, Microdata and JSON-LD. These vocabularies cover entities, relationships between entities and actions, and can easily be extended through a well-documented extension model. Over 10 million sites use Schema.org to markup their web pages and email messages. Many applications from Google, Microsoft, Pinterest, Yandex and others already use these vocabularies to power rich, extensible experiences.

Schema.org is sponsored by Google, Microsoft, Yahoo and Yandex. The vocabularies are developed by an open community process, using the public-schemaorg@w3.org mailing list and through GitHub.

A shared vocabulary makes it easier for webmasters and developers to decide on a schema and get the maximum benefit for their efforts. It is in this spirit that the sponsors, together with the larger community have come together, to provide a shared collection of schemas.

We invite you to get started!

View our blog at blog.schema.org or see release history.

Terms and conditions

Ok… but what *do* you do with it?

Products > Google Apps > Gmail > Email Markup

# Email Markup

Empower your emails using schema.org markup.

APPS SCRIPT QUICKSTART | END-TO-END EXAMPLE

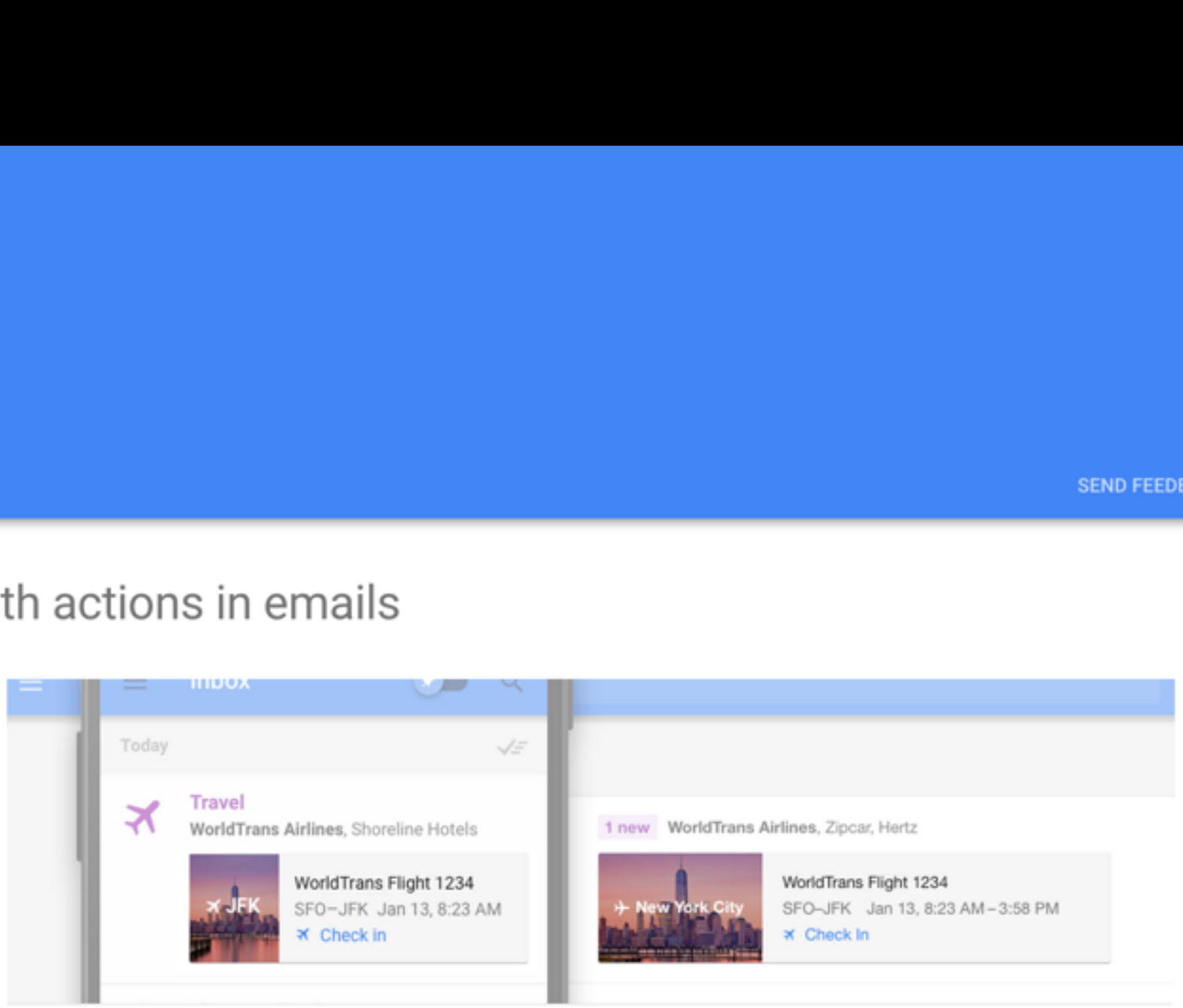
HOME | GUIDES | REFERENCE | SUPPORT | SEND FEEDBACK

## Increase user engagement with actions in emails

### Highlights in Inbox

Highlight your emails with important information and actions in Inbox by Gmail.

HIGHLIGHT KEY INFORMATION




Oh… now it starts to make more sense.

an anecdotal tale…

http://vocab.example.org/Thing

urn:lsid:example.org:vocab:Thing

I know what to do with this.

http://vocab.example.org/Thing

But what in the world is this?

urn:lsid:example.org:vocab:Thing

RDF, Turtle, JSON or HTML?

# Turtle:

```
@prefix xyz: <http://vocabs.example.org/>
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>
<http://example.org/Foo> a xyz:Thing ;
  xyz:hasPurpose xyz:Educating ;
  xyz:created “2015-08-08T12:34:56Z”^^xsd:dateTime
```

Where are the tools for *easily* producing and consuming this?

# JSON-LD:

```
{
  “@context”: “http://vocabs.example.org”,
  “@type”: “Thing”,
  “hasPurpose”: “Educating”,
  “created”: “2015-08-08T12:34:56Z”
}
```

Why would a developer choose Turtle over JSON?

- JSON-LD is a semantic data model serialized as JSON.
- JSON-LD is a fantastic tool for normalizing data and allowing for extensibility.
- It makes certain assumptions.

# What's the right way to process this?

```
{
  "@context": "http://www.w3.org/ns/activitystreams",
  "@type": "Create",
  "actor": {
    "@type": "Person",
    "@id": "http://sally.example.org",
    "displayName": "Sally"
  },
  "object": {
    "@type": "Note",
    "content": "This is a simple note"
  },
  "published": "2015-01-25T12:34:56Z"
}
```

JSON.parse(input) ?

```
[
  {
    "@type": ["http://www.w3.org/ns/activitystreams#Create"],
    "http://www.w3.org/ns/activitystreams#actor": [
      {
        "@id": "http://sally.example.org",
        "@type": ["http://www.w3.org/ns/activitystreams#Person"],
        "http://www.w3.org/ns/activitystreams#displayName": [
          {"@value": "Sally"}
        ]
      }
    ],
    "http://www.w3.org/ns/activitystreams#object": [
      {
        "@type": ["http://www.w3.org/ns/activitystreams#Note"],
        "http://www.w3.org/ns/activitystreams#content": [
          {"@value": "This is a simple note"}
        ]
      }
    ],
    "http://www.w3.org/ns/activitystreams#published": [
      {
        "@type": "http://www.w3.org/2001/XMLSchema#dateTime",
        "@value": "2015-01-25T12:34:56Z"
      }
    ]
  }
]
```


```
{
  "@context": "http://www.w3.org/ns/activitystreams",
  "@graph": [
    {
      "@id": "_:b0",
      "@type": "Create",
      "actor": "http://sally.example.org",
      "object": "_:b1",
      "published": "2015-01-25T12:34:56Z"
    },
    {
      "@id": "_:b1",
      "@type": "Note",
      "content": "This is a simple note"
    },
    {
      "@id": "acct:sally@example.org",
      "@type": "Person",
      "displayName": "Sally"
    }
  ]
}
```

But, wait, it get’s worse.

```
{
  "@context": {
    "@vocab": "http://www.w3.org/ns/activitystreams/",
    "xsd": "http://www.w3.org/2001/XMLSchema#",
    "Foo": "Create", "Bar": "Person", "Baz": "Note",
    "doer": "actor", "label": "displayName",
    "words": "content",
    "when": {"@id": "published", "@type": "xsd:dateTime"}
  },
  "@type": "Foo",
  "doer": {
    "@type": "Bar",
    "@id": "http://sally.example.org",
    "label": "Sally"
  },
  "object": {
    "@type": "Baz",
    "words": "This is a simple note"
  },
  "when": "2015-01-25T12:34:56Z"
}
```

```
<http://sally.example.org>
  <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
  <http://www.w3.org/ns/activitystreams#Person> .

<http://sally.example.org>
  <http://www.w3.org/ns/activitystreams#displayName>
  "Sally" .

_:c14n0
  <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
  <http://www.w3.org/ns/activitystreams#Create> .

_:c14n0
  <http://www.w3.org/ns/activitystreams#actor>
  <http://sally.example.org> .
```

btw, this format is called “turtle”

```
_:c14n0
  <http://www.w3.org/ns/activitystreams#object>
  _:c14n1 .

_:c14n0
  <http://www.w3.org/ns/activitystreams#published>
  "2015-01-25T12:34:56Z"^^<http://www.w3.org/2001/XMLSchema#dateTime> .

_:c14n1
  <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
  <http://www.w3.org/ns/activitystreams#Note> .

_:c14n1
  <http://www.w3.org/ns/activitystreams#content>
  "This is a simple note" .
```

http://json-ld.org/playground/

JSON.parse and JSON.stringify

= format only (semantics are secondary)

jsonld.expand and jsonld.compact

= *semantic* model (format is secondary)

“Given the additional complexity JSON-LD *requires*, why would I use it?”

*- Any REST Developer Anywhere*

Go read this:

http://manu.sporny.org/2014/json-ld-origins-2/

(warning: some NSFW language)

And watch these:

https://www.youtube.com/watch?v=vioCbTo3C-4

https://www.youtube.com/watch?v=4x_xzT5eF5Q

Practical Semantics Rule #1 :

*You aren’t going to need most of it.*

(Btw, notice that I haven't said anything about specific vocabularies yet)

# Case Study : Activity Streams 2.0

- Activity Streams v2.0 is a vocabulary for describing past, ongoing or future activity.
- It uses JSON-LD as a basis.

# “Sally created a note”

```
{
  "@context": "http://www.w3.org/ns/activitystreams",
  "@type": "Create",
  "actor": {
    "@type": "Person",
    "@id": "http://sally.example.org",
    "displayName": "Sally"
  },
  "object": {
    "@type": "Note",
    "content": "This is a simple note"
  },
  "published": "2015-01-25T12:34:56Z"
}
```

# “James accepted Mike’s Invitation to Restfest”

```
{
  "@context": "http://www.w3.org/ns/activitystreams",
  "@type": "Accept",
  "actor": {
    "@type": "Person",
    "@id": “http://jasnell.me",
    "displayName": "James"
  },
  "object": {
    "@type": "Invite",
    "actor": "http://amundsen.com",
    "target": "http://restfest.org"
  },
  "published": "2015-01-25T12:34:56Z"
}
```

- We use JSON-LD, *but* …
  - We require a normative @context
  - We require a specific compacted form of JSON
  - We do not require RDF processing

```
{
  "@context": "http://www.w3.org/ns/activitystreams",
  "@type": "Create",
  "actor": {
    "@type": "Person",
    "@id": "http://sally.example.org",
    "displayName": "Sally"
  },
  "object": {
    "@type": "Note",
    "content": "This is a simple
  },
  "published": "2015-01-25T12:34:56Z
}
```

This is valid Activity Streams 2.0 JSON-LD

```
[
  {
    "@type": ["http://www.w3.org/ns/activitystreams#Create"],
    "http://www.w3.org/ns/activitystreams#actor": [
      {
        "@id": “http://sally.example.org",
        "@type": ["http://www.w3.org/ns/activitystreams#Person"],
        "http://www.w3.org/ns/activitystreams#displayName": [
          {"@value": "Sally"}
        ]
      }
    ],
    "http://www.w3.org/ns/activitystreams#object": [
      {
        "@type": ["http://www.w3.org/ns/activitystreams#Note"],
        "http://www.w3.org/ns/activitystreams#content": [
          {"@value": "This is a simple note"}
        ]
      }
    ],
    "http://www.w3.org/ns/activitystreams#published": [
      {
        "@type": "http://www.w3.org/2001/XMLSchema#dateTime",
        "@value": "2015-01-25T12:34:56Z"
      }
    ]
  }
]
```

This is valid JSON-LD, but…

It’s not valid Activity Streams 2.0

```
{
  "@context": "http://www.w3.org/ns/activitystreams",
  "@graph": [
    {
      "@id": "_:b0",
      "@type": "Create",
      "actor": “http://sally.example.org",
      "object": "_:b1",
      "published": "2015-01-25T12:34:56Z"
    },
    {
      "@id": "_:b1",
      "@type": "Note",
      "content": "This is a simple note"
    },
    {
      "@id": "acct:sally@example.org",
      "@type": "Person",
      "displayName": "Sally"
    }
  ]
}
```

This is also valid JSON-LD, but…


It’s also not valid Activity Streams 2.0

```
{
  "@context": {
    "@vocab": "http://www.w3.org/ns/activitystreams/",
    "xsd": "http://www.w3.org/2001/XMLSchema#",
    "Foo": "Create", "Bar": "Person", "Baz": "Note",
    "doer": "actor", "label": "displayName",
    "words": "content",
    "when": {"@id": "published", "@type": "xsd:dateTime"}
  },
  "@type": "Foo",
  "doer": {
    "@type": "Bar",
    "@id": "http://sally.example.org",
    "label": "Sally"
  },
  "object": {
    "@type": "Baz",
    "words": "This is a simple note"
  },
  "when": "2015-01-25T12:34:56Z"
}
```

This is valid JSON-LD also…

But, you guessed it… it’s not valid Activity Streams 2.0

Practical Semantics Rule #2:

*Just because you can do something, doesn't mean you should.*

# Relative URLs

In HTML, using Relative URLs makes sense because the browser maintains the context…

```
<html>
  <body>
    <a href=“about.html”>About Me</a>
  </body>
</html>
```

In JSON-LD, you *can* use Relative URLs if you'd like, but you really shouldn't

```
{
  “@context”: “http://www.w3.org/ns/activitystreams”,
  “@type”: “Link”,
  “href”: “about.html”
}
```

- JSON has no “URL” primitive. It does not differentiate between:

  “`plain text`” and “`http://a.url/of/some?kind`”

- JSON Parsers do not preserve Base URL context so the relative URLs you use in your JSON documents can become easily separated from the Base URL.

```
{
  “@context”: “http://www.w3.org/ns/activitystreams”,
  “@type”: “Link”,
  “href”: “about.html”
}
```

```
{
  “@context”: “http://www.w3.org/ns/activitystreams”,
  “@type”: “Link”,
  “href”: “http://example.org/about.html”
}
```

But what about @base ?

Just Don’t.

```
{
  “@context”: [
    {“@base”: “http://foo.example.com”},
    {“@base”: “http://bar.example.org”}
  ],
  “href”: “about.html”
}
```

```
{
  “@context”: [
    {“@base”: “http://bar.example.org”},
    {“@base”: “http://foo.example.com”}
  ],
  “href”: “about.html”
}
```

Absolute URLs makes life simple.

JSON-LD is good, but over-reliance on JSON-LD mechanisms add unnecessary complexity.

# Dates and Times

```
{
  “@context”: “http://vocab.example.org”,
  “somedate”: 1440866771
}
```

```
{
  “@context”: “http://vocab.example.org”,
  “somedate”: “2015-08-29T12:34:56Z”
}
```

ISO-8601

Anything Else

Remember: Practical Semantics is all about Affordance.

When I first look at something,
I should already know what to do with it.

- ✓ HTTP URL's
- ✓ Absolute URL's
- ✓ Standard Formats
- ✓ Normative JSON-LD Context
- ✓ Use common vocabularies
- ✓ Avoid the use of RDF-isms

Using a normative JSON-LD context and requiring compact serialization reduces the WTF Factor for developers.

Whenever possible, reuse existing commonly accepted terms (vocabularies!)

When it comes to Semantic Data, there are two choices for a RESTful app developer:

*Producing* … so that others can do more interesting things your data.

*Consuming* … so that you can do more interesting things with other people's data.

Producing Semantic Markup is an Obvious Win

1. Search Engines reward you with better SEO.
2. Let’s be honest, your users are likely far more creative than you.

# Actions in the Inbox

# Mobile Application Deep Linking

Knowledge Graph

Machine Learning

- Producing Semantic Markup does not require you to “buy in” to the full semantic web philosophy.
- You can simply add a bit more markup to your Dust templates for instance…

Consuming Semantic Markup is harder

1. It can get complicated. Limiting options is a good thing.

2. The benefits are non-obvious until you actually start doing it.

3. Don’t try to get fancy. Keeping it simple is always the best choice.

It’s not all bad.

# OSLC

“Open Services for Lifecycle Collaboration”

# W3C Web Annotations

schema.org

- If you have buy-in from the community, adopting Semantic Web and Linked Data technologies is simple. The *incentives* for participating become clear.

- If you don’t… well, good luck.

A Challenge to You.

Build applications using common vocabularies and linked data.

Make them simple.

Don’t get fancy.

Make the benefits obvious.

Most importantly : help improve the tools so that the benefits are obvious to *others*

Thank you.