# JSON Schema

## Core Concepts, Common Pitfalls, and Debugging


👱 Ben Hutton

💼 JSON Schema core


🙌 opencollective.com/json-schema

🎤 ASC 2019 - API Specification Conference 2019

# Validation saves lives

# Validation (probably?) saves lives

# Validation changes lives

~~Validation (probably?) saves lives~~

# Story

# Matchmaker Exchange API

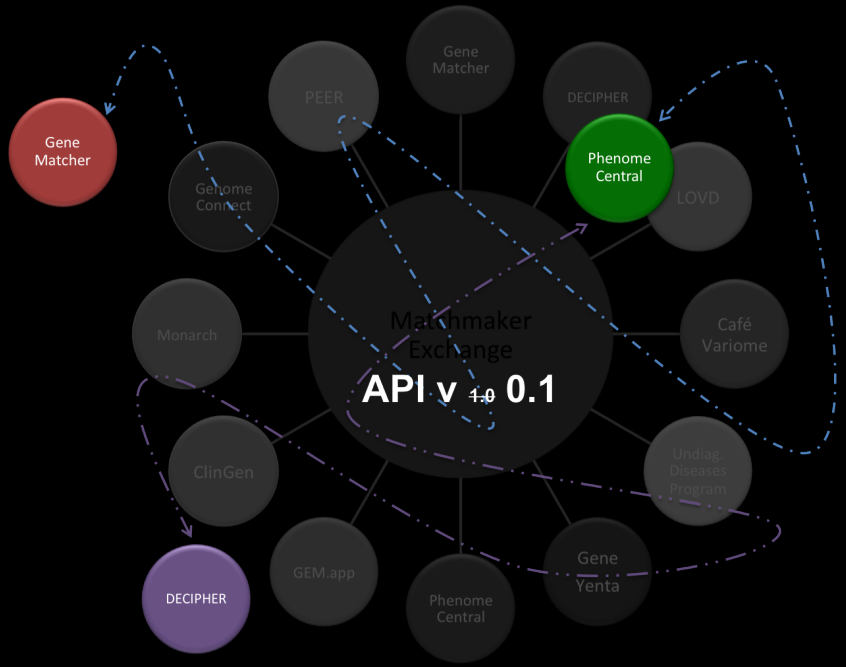
- Siloed databases of patient data
  - (including DNA variants)
- Rare and undiagnosed cases
  - (> 10 per country rare)
- Complex logal and ethical situation
- Discover and exchange pateint data

# Matchmaker Exchange API

- Each database holds slightly different data
- Representation of a patient needs to be uniform
- Write specification using words and examples
- Release v1.0!

## Sounds easy, right?

# Nope


Gene Matcher
PEER
Gene Matcher
DECIPHER
Phenome Central
Genome Connect
LOVD
Monarch
Matchmaker Exchange
API v ~~1.0~~ 0.1
Café Variome
ClinGen
Undiag. Diseases Program
DECIPHER
GEM.app
Phenome Central
Gene Yenta

# Building a schema

# Requirements for queries

- Root object must have a `patient`
- `patient` must have either properties `genomicFeatures` or `phenotypicFeatures` or both

```
{
  "$schema": "http://json-schema.org/draft-07/schema",
  "title": "MatchMakerExchange format for queries",
  "patient": {
    "$comment": "`patient`: `genomicFeatures` or `phenotypicFeatures` or both",
    "oneOf": [
      "genomicFeatures",
      "phenotypicFeatures"
    ]
  }
}
```

This is an unknown keyword. Unknown keywords are ignored.

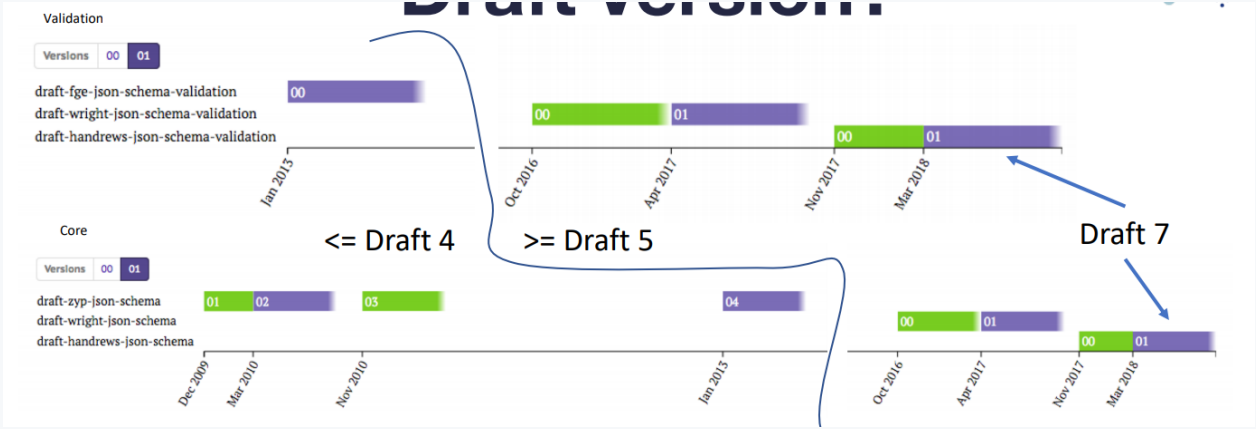
# What's a "schema" anyway?

# JSON Schema

- A vocabulary that allows you to annotate and validate JSON documents.
- An IETF "personal draft" document specification
- JSON!
- Must be a boolean or an object

# JSON Schema

- This talk covers draft-7. Not draft-4, not draft-8



# Key Concepts

**Schema “keywords”** : Object properties that are applied to the instance

**Instance** : The JSON document being validated or associated with a Schema

**Root Schema** : Schema that is the whole JSON document
**Subschema** : A schema as a value of an object or array

Keywords fall under one or both of two categories (mostly):

**Assertions** : produce a boolean result when applied to an instance
**Annotations** : attach information to an instance for application use

# "properties" keyword

`properties` is an object

The values of the object are applied to the instance according to matching objects keys

```
{
  "$schema": "http://json-schema.org/draft-07/schema",
  "title": "MatchMakerExchange format for queries",
  "properties": {
    "patient": {
      "$comment": "`patient`: `genomicFeatures` or `phenotypicFeatures` or both",
      "oneOf": [
        {
          "properties": {
            "genomicFeatures": true
          }
        },
        {
          "properties": {
            "phenotypicFeatures": true
          }
        }
      ]
    }
  }
}
```

Remember, the values of a `properties` object must be Schemas. A Boolean is a valid Schema.

# A Boolean can be a schema?

# What's a Schema again? (pt 2)

# An Object or a Boolean

## Expressing no constraints

```
{}
```

An empty object

```
true
```

a `true` boolean value

## Result in false assertion for validation

```
{ "not": {} }
```

`not` keyword: inverts the assertion result

```
false
```

a `false` boolean value

# Does it work?

```
{
  "$schema": "http://json-schema.org/draft-07/schema",
  "title": "MatchMakerExchange format for queries",
  "properties": {
    "patient": {
      "oneOf": [
        {
          "properties": {
            "genomicFeatures": {
              "type": [
                "array"
              ]
            }
          }
        },
        {
          "properties": {
            "phenotypicFeatures": {
              "type": [
                "array"
              ]
            }
          }
        }
```

```
{
  "patient": {
    "phenotypicFeaturs": "long nose"
  }
}
```

😕 But didn't I define that should be an array?

# oneOf what?

# (More) Key Concepts

**Applicator keywords**: Determin how subschemas are applied to an instance location.

Include, but not limited to...

`oneOf`, `allOf`, and `anyOf`

The result of applying a schema to an instance is includes an assertion of validity.

The value of `*Of` keywords must be an array of Schemas.

The resulting assertion is a combination of the assertion results.

For example, `oneOf` requires that ONLY one of the Schemas in the array is valid.