JSON at Work: Schema, Search, and Transform

Tom Marrs

@TomMarrs

About Me …

livingsocial

O'REILLY

dzone
fresh links for developers
REFCARDZ

NFJS
No Fluff Just Stuff.com

GREAT INDIAN DEVELOPER SUMMIT

HTML 5

What's The Point?

Drive API Design with JSON Schema

What's The Point? #2

JSON Search and Transform

Simplify interaction with RESTful APIs

Agenda

Your Takeaway

Core JSON Schema + JSON Workflow
+ Really Cool Node Modules :-)

We’re Not Covering

REST

Deep JS

Other Languages

Required Downloads

Local Server
http://10.10.32.101

Click on link with my name - Tom Marrs, tmarrs, or whatever

Examples and Slides

https://github.com/tmarrs/presentations/tree/master/OSCON/2015/JSON-at-Work-Schema-Search-and-Transform

Where Are We?

JSON Schema Overview	1	JSON Search & Transform Overview	4
Core JSON Schema	2	JSON Search	5
API Design with JSON Schema	3	JSON Transform	6

What is JSON Schema?

Validate Structure + Format

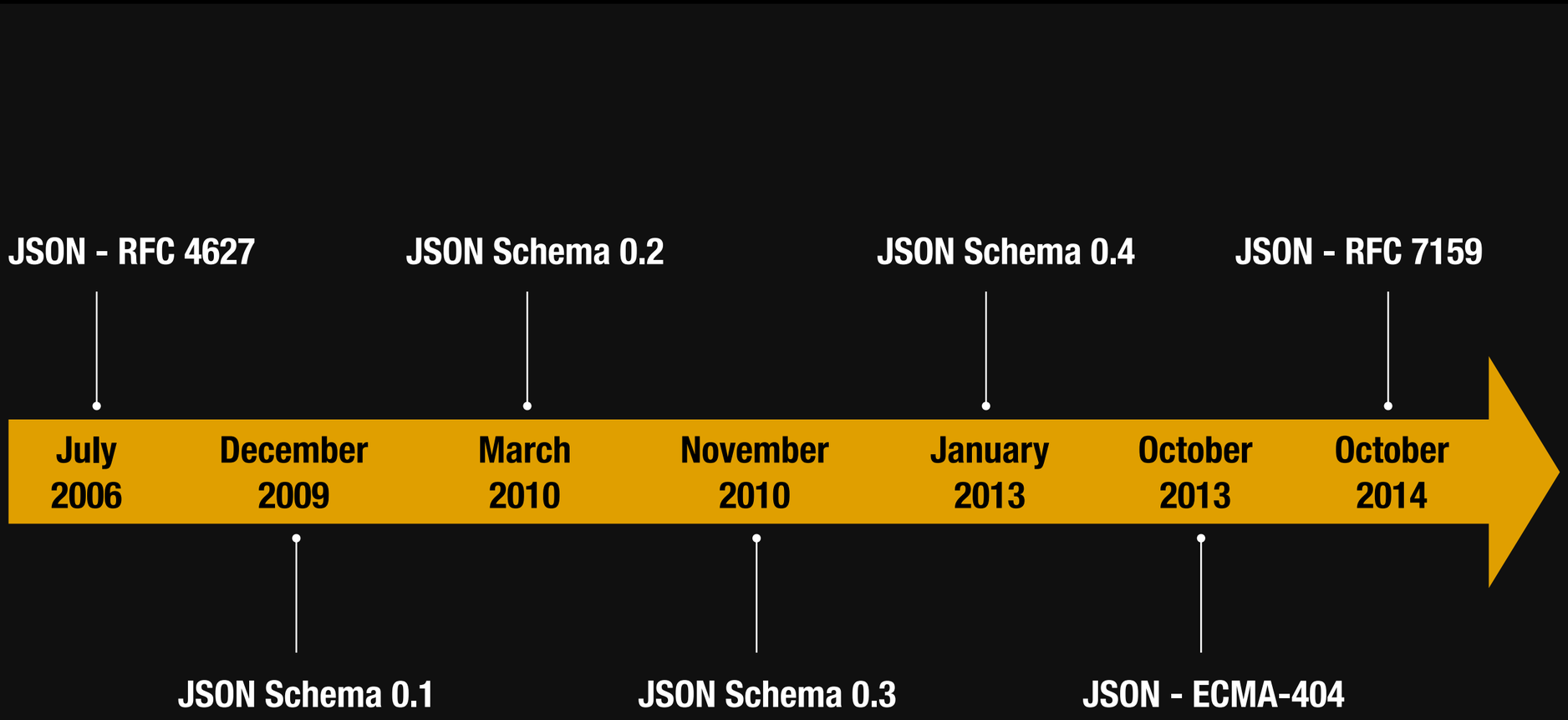
Basic JSON Schema

```
1
2  {
3    "$schema": "http://json-schema.org/draft-04/schema#",
4    "type": "object",
5    "properties": {
6      "email": {
7        "type": "string"
8      },
9      "firstName": {
10       "type": "string"
11     },
12     "lastName": {
13       "type": "string"
14     }
15   }
16 }
```

Basic JSON - Document

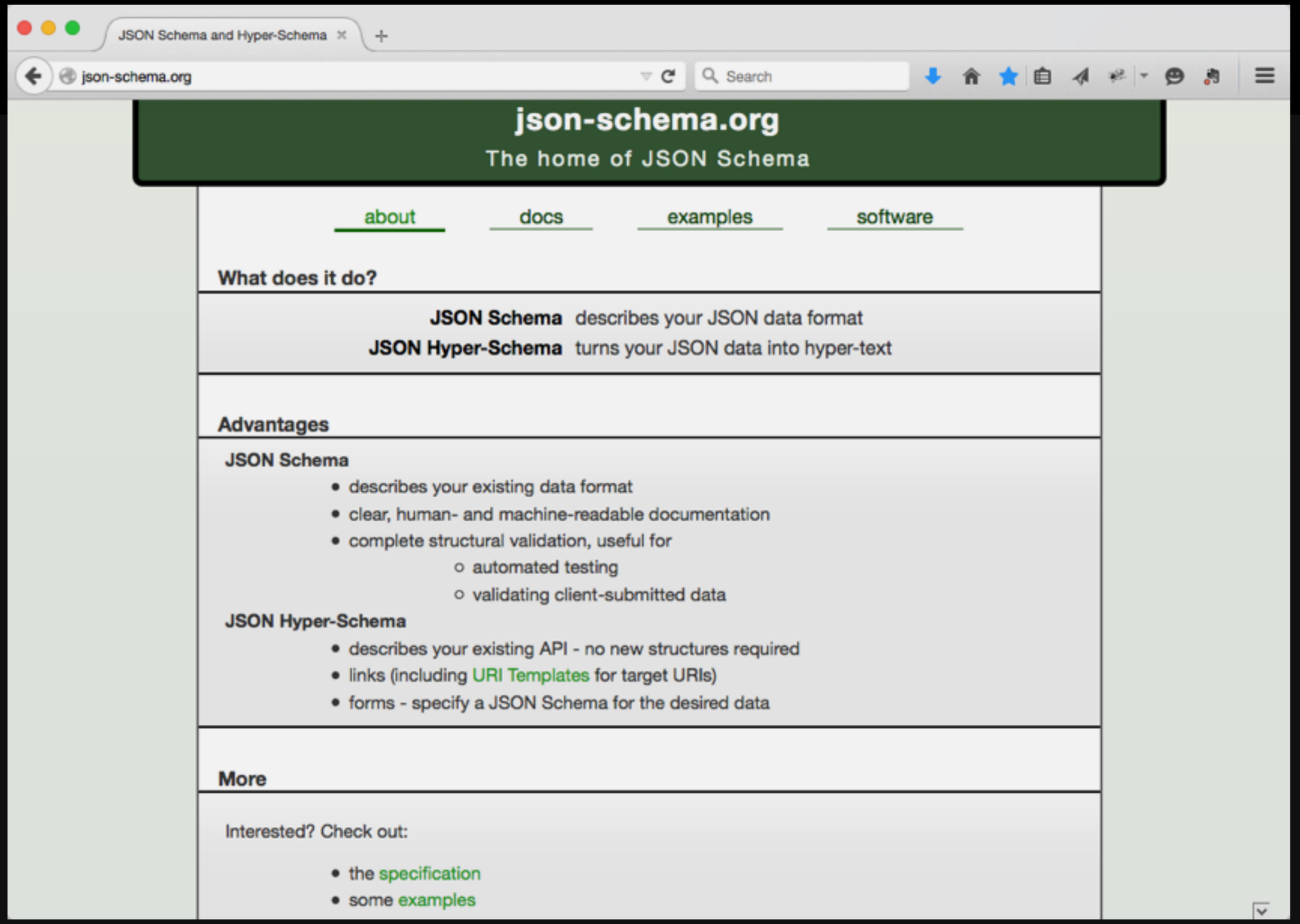
```
2  {
3     "email": "larsonrichard@ecratic.com",
4     "firstName": "Larson",
5     "lastName": " Richard"
6  }
```

JSON Schema Timeline - When?

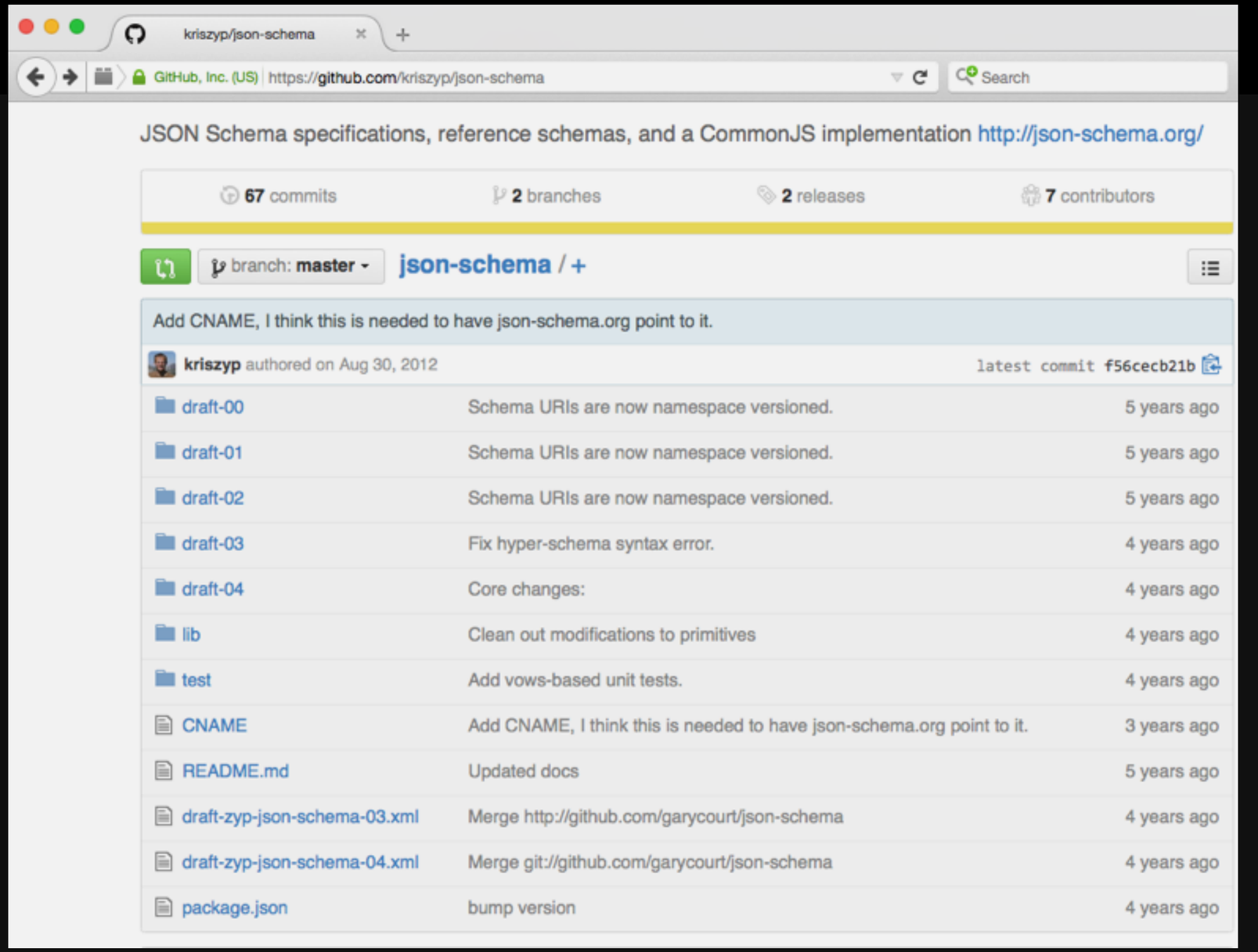


The Journey …

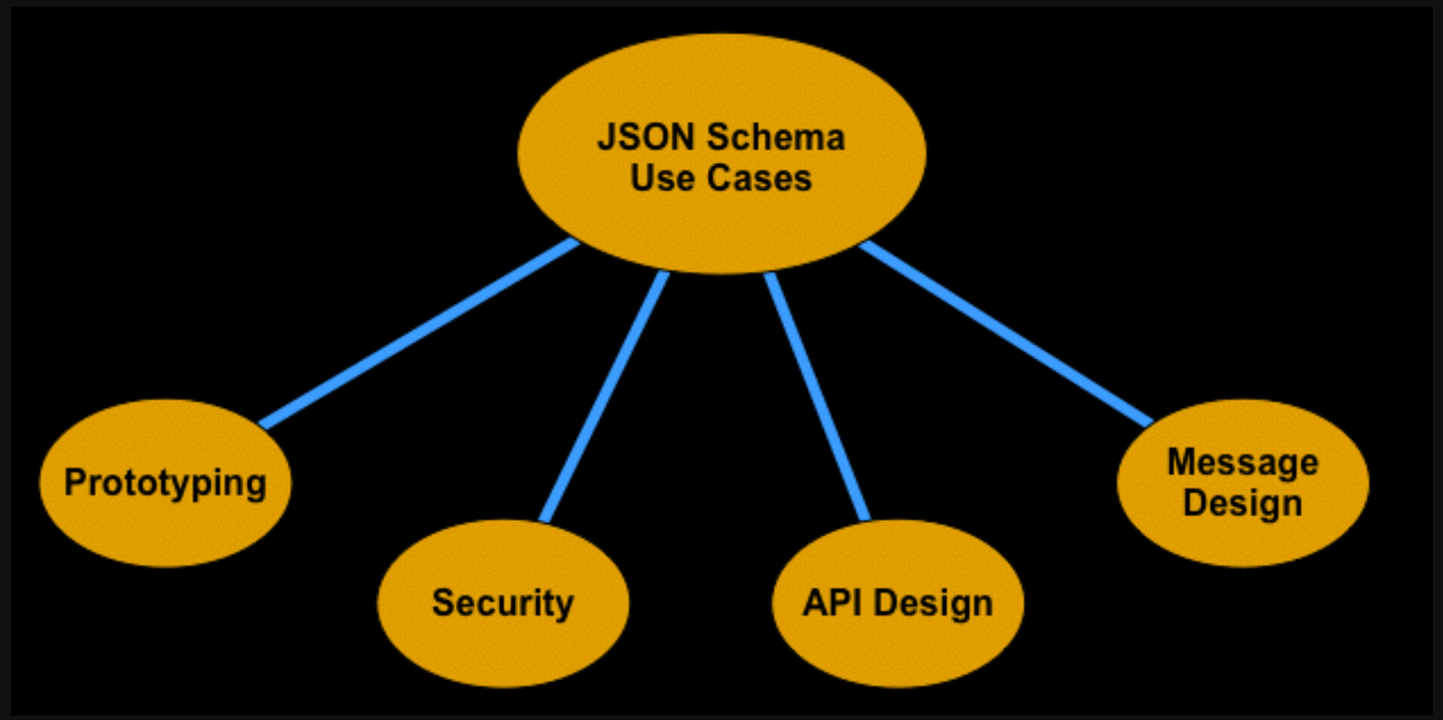
json-schema.org



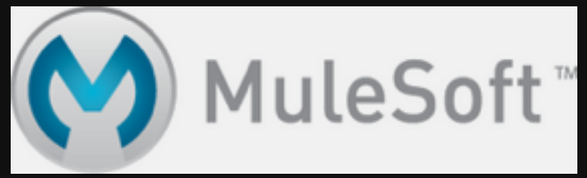
JSON Schema on GitHub



Where Does JSON Schema Fit?



Who uses JSON Schema?



Swagger

Google Developers

Why isn’t JSON Validation Enough?

Semantics	Syntax
Schema + Instance Document	Instance Document
Meaning	Well-formed - Valid JSON
Ex: Person, Order	{ }

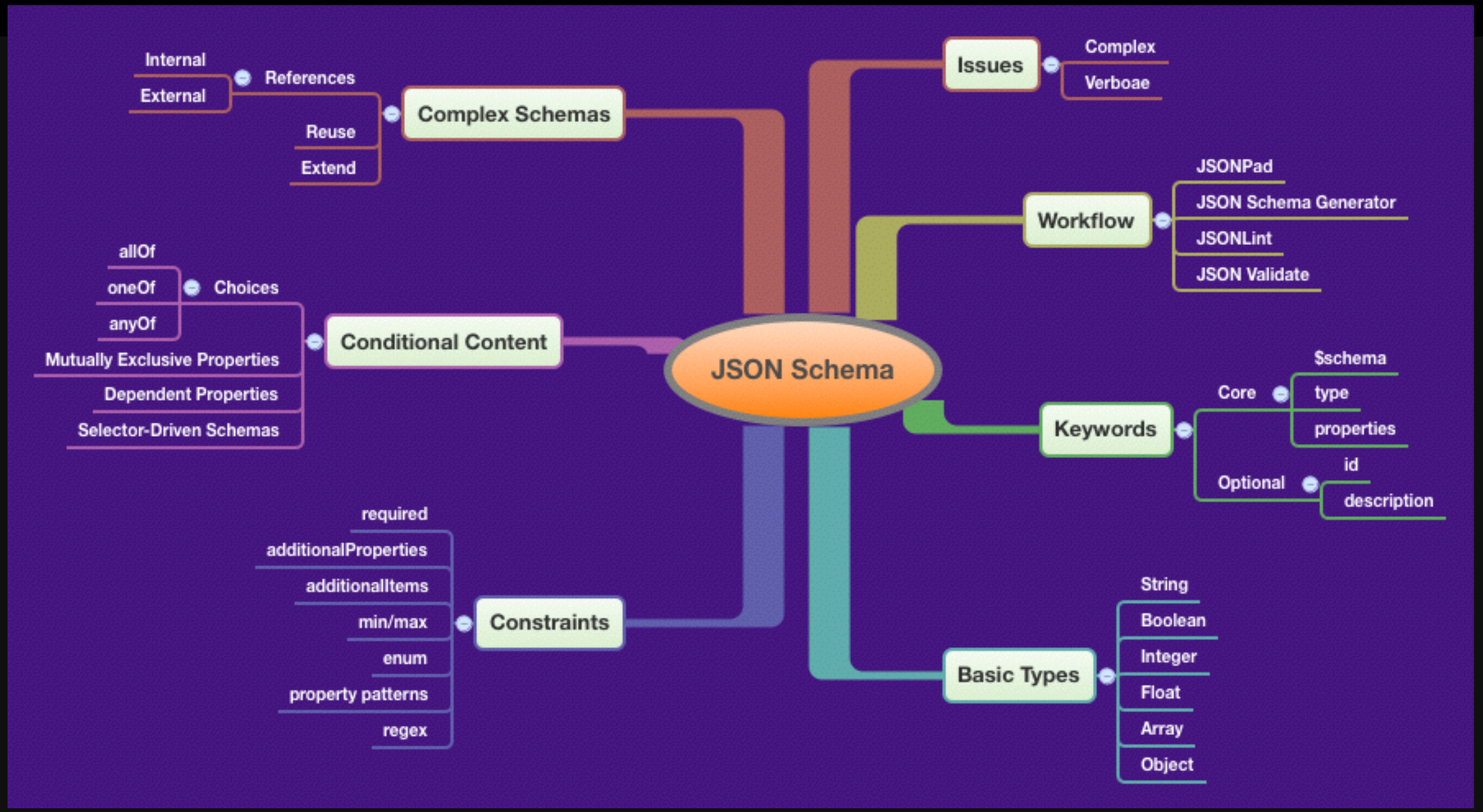
Haven’t We Seen This Before?

JSON Schema	XML Schema
No Reference	Instance Document references Schema
No Namespace - Yes!	Namespace Misery
.json	.xsd

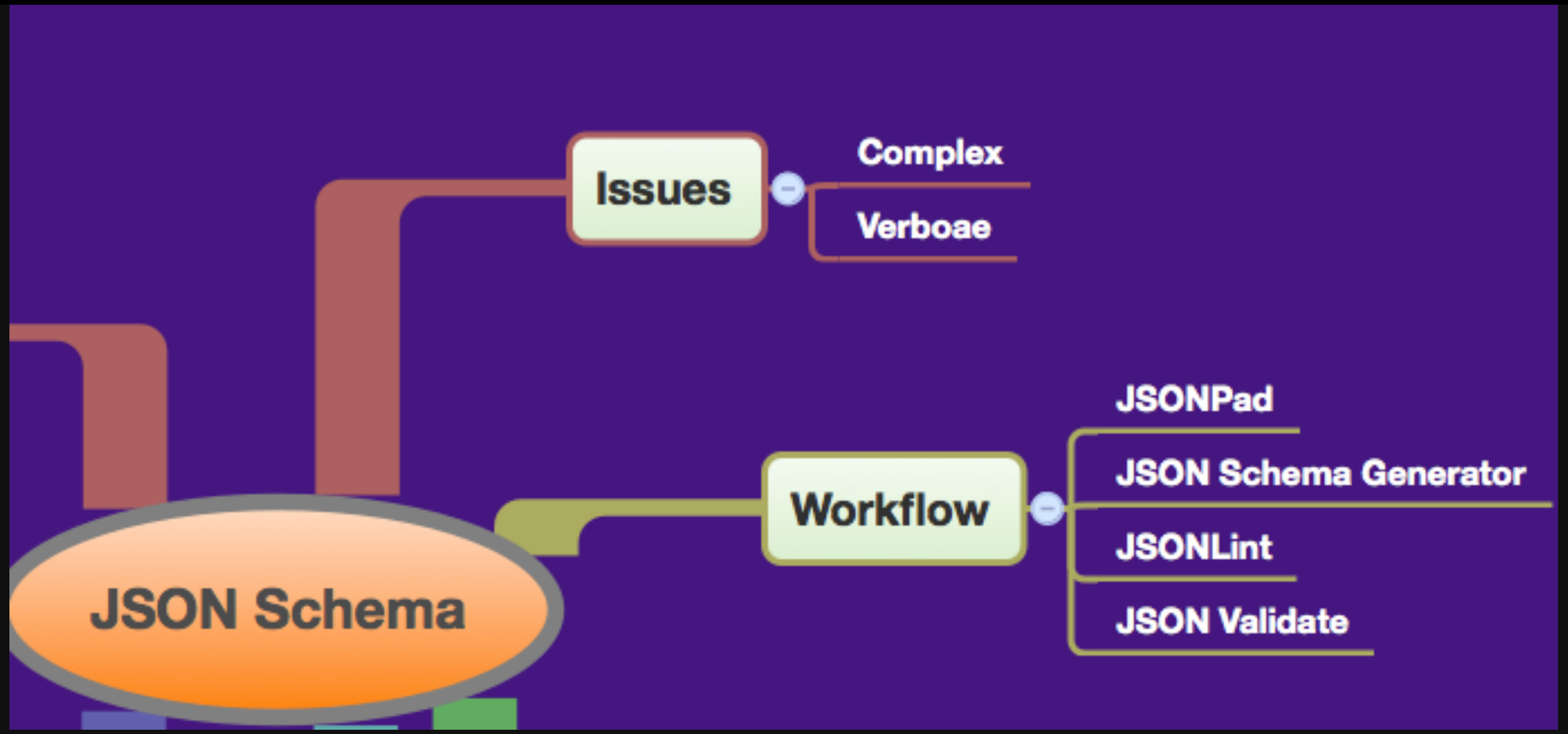
Where Are We?

JSON Schema Overview	1	JSON Search & Transform Overview	4
Core JSON Schema	**2**	JSON Search	5
API Design with JSON Schema	3	JSON Transform	6

Facets of JSON Schema



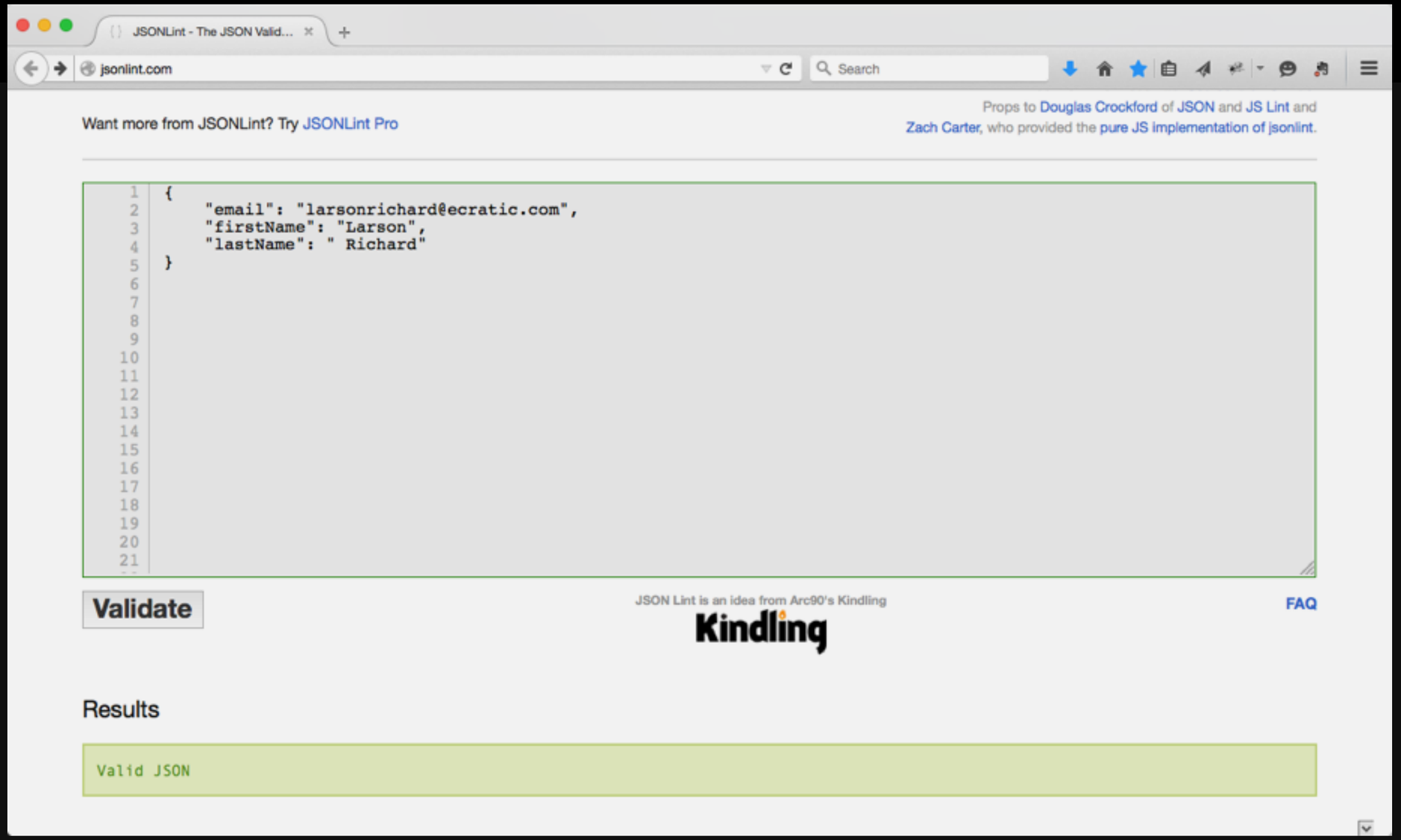
JSON Schema - Issues and Workflow



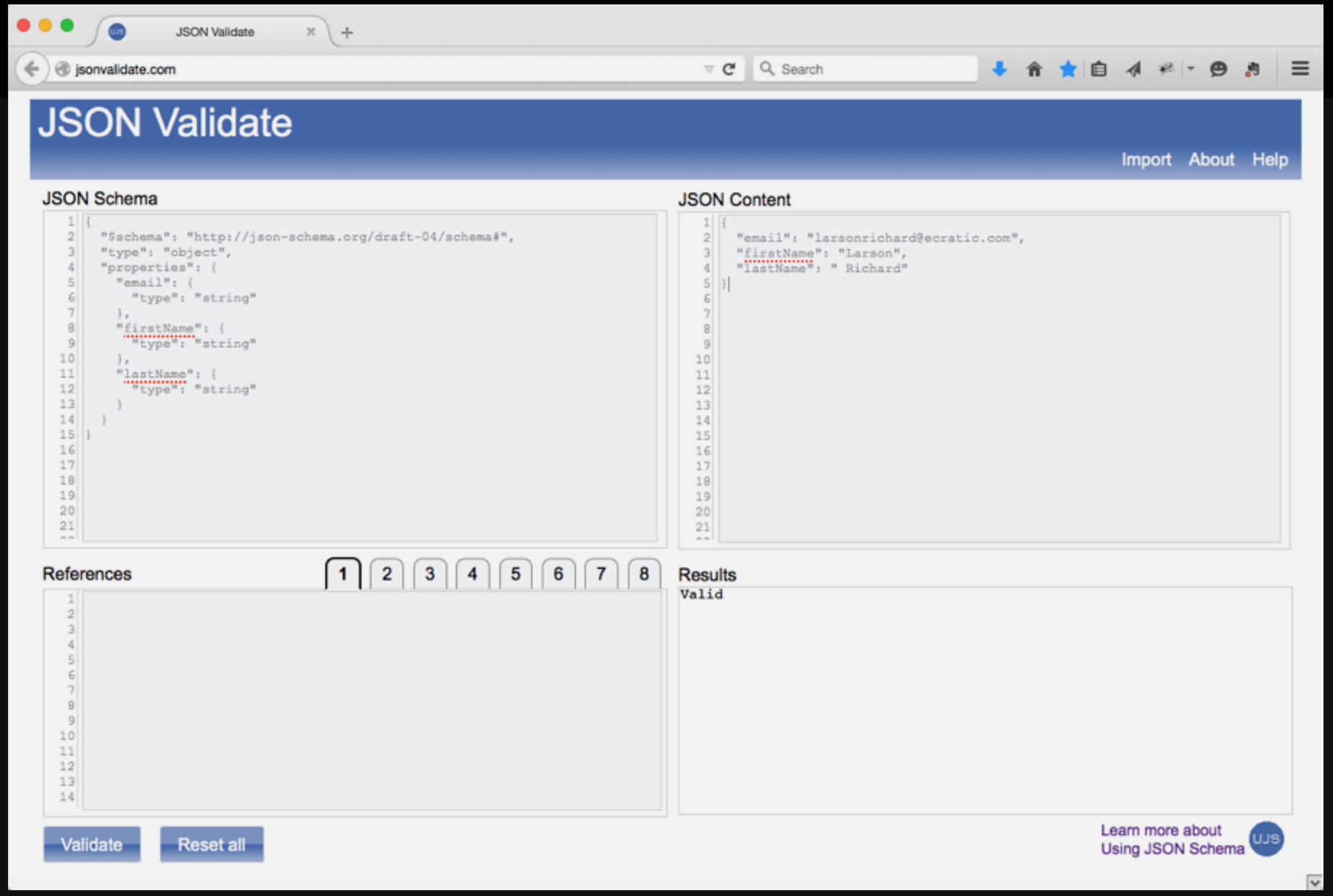
My JSON Schema Workflow

Model JSON Document	Generate JSON Schema	Validate JSON Document
JSONPad https://code.google.com/p/json-pad/	JSON Generator http://jsonschema.net/	JSON Validate http://jsonvalidate.com/ JSONLint http://jsonlint.com

JSONLint


JSONLint - The JSON Valid...
jsonlint.com
Search
Want more from JSONLint? Try JSONLint Pro
Props to Douglas Crockford of JSON and JS Lint and Zach Carter, who provided the pure JS implementation of jsonlint.
{
"email": "larsonrichard@ecratic.com",
"firstName": "Larson",
"lastName": " Richard"
}
Validate
JSON Lint is an idea from Arc90's Kindling
Kindling
FAQ
Results
Valid JSON

JSON Validate



Beware of Wonky WiFi - Hedge Your Bets!

PDD - Presentation-Driven Development

jsonlint

```
npm install -g jsonlint

jsonlint basic.json
```

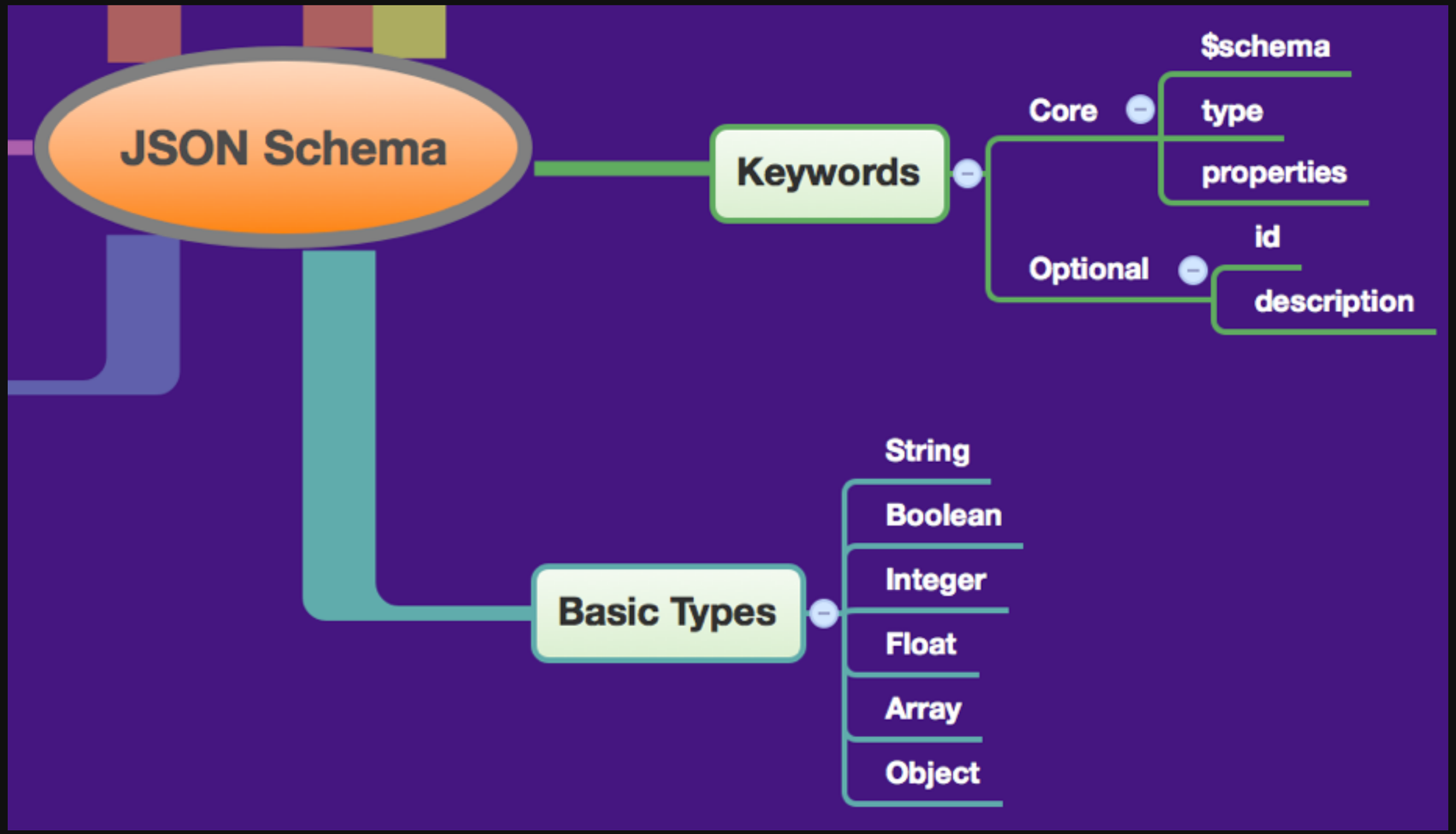
https://github.com/zaach/jsonlint

ujs-jsonvalidate

```
npm install -g ujs-jsonvalidate

validate ex-1-basic.json ex-1-basic-schema.json
```

https://github.com/usingjsonschema/ujs-jsonvalidate-nodejs

JSON Schema - Keywords and Basic Types



Basic Keywords

Keyword	Definition
`$schema`	Specify JSON Schema version - `"$schema": "http://json-schema.org/draft-04/schema#"`
`type`	The data type - `"type": "string"`
`properties`	The fields for an object

Optional Keywords

Keyword	Definition
`id`	(1): Path to field (2): URI to Schema
`description`	For documentation

Basic Types - JSON Schema

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "properties": {
    "email": {
      "type": "string"
    },
    "firstName": {
      "type": "string"
    },
    "lastName": {
      "type": "string"
    },
    "age": {
      "type": "integer"
    },
    "postedSlides": {
      "type": "boolean"
    },
    "rating": {
      "type": "number"
    }
  }
}
```

Basic Types - JSON Document

```json
{
  "email": "larsonrichard@ecratic.com",
  "firstName": "Larson",
  "lastName": " Richard",
  "age": 39,
  "postedSlides": true,
  "rating": 4.1
}
```

Where's the Validation?

Keyword	Definition
`additionalProperties`	enable/disable additional fields in an object
`required`	Which fields are required
`additionalItems`	enable/disable additional array elements

Basic Types Validation - JSON Schema

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "properties": {
    "email": {
      "type": "string"
    },
    "firstName": {
      "type": "string"
    },
    "lastName": {
      "type": "string"
    },
    "postedSlides": {
      "type": "boolean"
    },
    "rating": {
      "type": "number"
    }
  },
  "additionalProperties": false
}
```

Validation with Required - JSON Schema

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "properties": {
    "email": {
      "type": "string"
    },
    "firstName": {
      "type": "string"
    },
    "lastName": {
      "type": "string"
    },
    "postedSlides": {
      "type": "boolean"
    },
    "rating": {
      "type": "number"
    }
  },
  "additionalProperties": false,
  "required": ["email", "firstName", "lastName", "postedSlides", "rating"]
}
```

Validation with Required - JSON Doc

```
{
  "email": "larsonrichard@ecratic.com",
  "firstName": "Larson",
  "lastName": " Richard",
  "rating": 4.1
}
```

Number min/max - JSON Schema

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "properties": {
    "rating": { "type": "number", "minimum": 1.0, "maximum": 5.0 }
  },
  "additionalProperties": false,
  "required": ["rating"]
}
```

Number min/max - JSON Doc

```
2   {
3       "rating": "4.2"
4   }
```

Simple Array - JSON Schema

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "properties": {
    "tags": {
      "type": "array",
      "items": {
        "type": "string"
      },
      "additionalItems": false
    }
  },
  "additionalProperties": false,
  "required": ["tags"]
}
```

Simple Array - JSON Doc

```
2   {
3       "tags": ["fred"]
4   }
```

Array min/max - JSON Schema

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "properties": {
    "tags": {
      "type": "array",
      "minItems": 2,
      "maxItems": 4,
      "items": {
        "type": "string"
      },
      "additionalItems": false
    }
  },
  "additionalProperties": false,
  "required": ["tags"]
}
```

Array min/max - JSON Doc

```
2   {
3       "tags": ["fred", "a"]
4   }
```

Enum - JSON Schema

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "properties": {
    "tags": {
      "type": "array",
      "minItems": 2,
      "maxItems": 4,
      "items": {
        "type": "string",
        "enum": [
          "Open Source", "Java", "JavaScript", "JSON", "REST"
        ]
      },
      "additionalItems": false
    }
  },
  "additionalProperties": false,
  "required": ["tags"]
}
```

Enum - JSON Doc

```
2   {
3       "tags": ["Java", "REST"]
4   }
```

Named Object - JSON Schema

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "properties": {
    "speaker": {
      "type": "object",
      "properties": {
        "firstName": { "type": "string" },
        "lastName": { "type": "string" },
        "email": { "type": "string" },
        "postedSlides": { "type": "boolean" },
        "rating": { "type": "number" },
        "tags": {
          "type": "array",
          "items": { "type": "string" },
          "additionalItems": false
        }
      },
      "additionalProperties": false,
      "required": ["firstName", "lastName", "email",
                   "postedSlides", "rating", "tags"
      ]
    }
  },
  "additionalProperties": false,
  "required": ["speaker"]
}
```

Named Object - JSON Doc

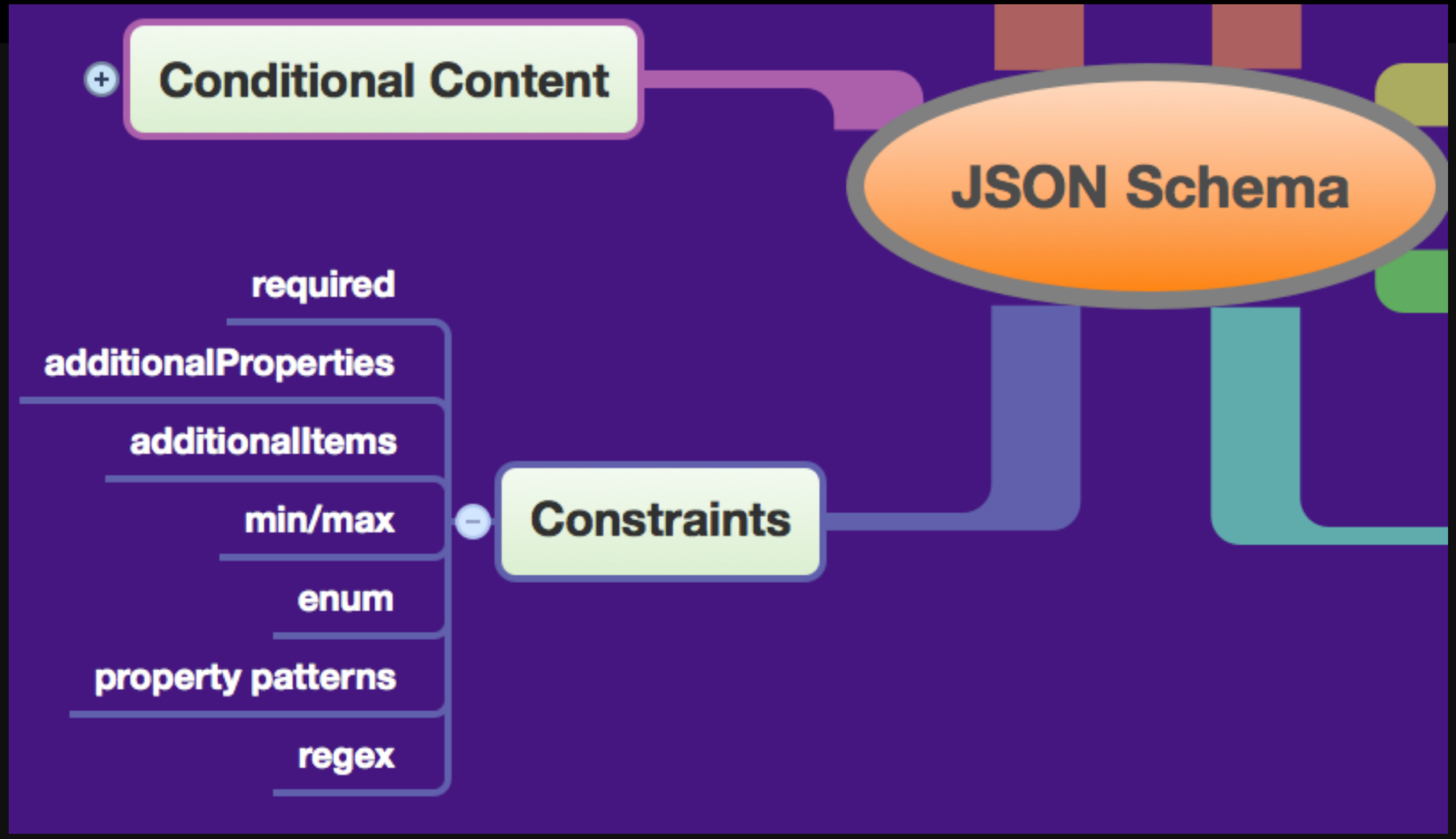
```
{
  "speaker": {
    "firstName": "Larson",
    "lastName": " Richard",
    "email": "larsonrichard@ecratic.com",
    "postedSlides": true,
    "rating": 4.1,
    "tags": [
      "JavaScript", "AngularJS", "Yeoman"
    ]
  }
}
```

Project 1 - Schema Modeling and Basic Types

projects/README.md#project-1---schema-modeling-and-basic-types

projects/EspressoCON.md

JSON Schema - Constraints



Property Patterns - JSON Schema

```
2  {
3    "$schema": "http://json-schema.org/draft-04/schema#",
4    "type": "object",
5    "properties": {
6      "city": { "type": "string" },
7      "state": { "type": "string" },
8      "zip": { "type": "string" },
9      "country": { "type": "string" }
10   },
11   "patternProperties": {
12     "^line[1-3]$": { "type": "string" }
13   },
14   "additionalProperties": false,
15   "required": ["city", "state", "zip", "country"]
16 }
```

Property Patterns - JSON Doc

```
2  {
3     "line1": "555 Main Street",
4     "line2": "#2",
5     "city": "Denver",
6     "state": "CO",
7     "zip": "80231",
8     "country": "USA"
9  }
```

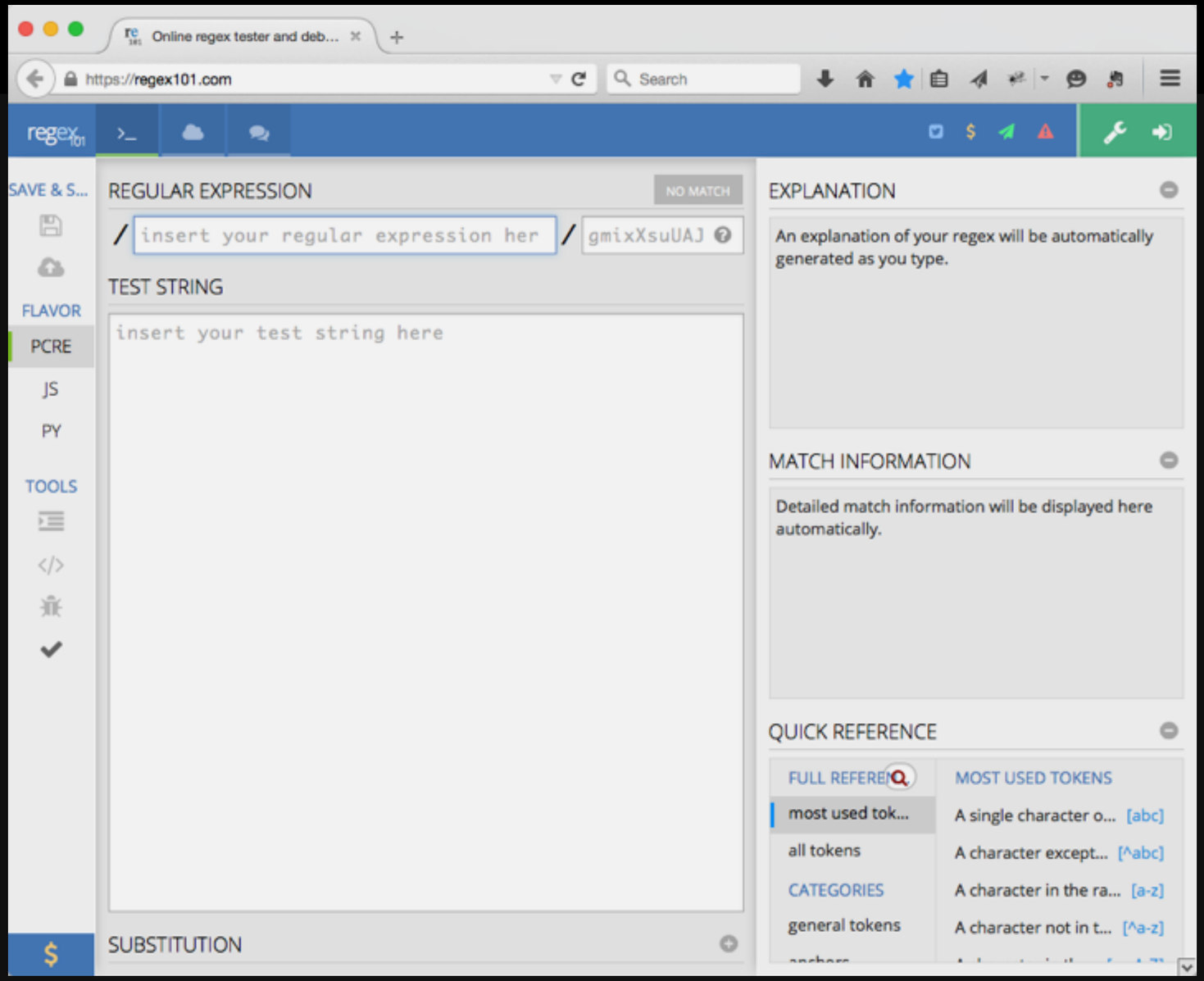
Regex - JSON Schema

```
3  {
4    "$schema": "http://json-schema.org/draft-04/schema#",
5    "type": "object",
6    "properties": {
7      "email": {
8        "type": "string",
9        "pattern": "^[\\w|-]+@[\\w|-]+\\.[A-Za-z]{2,4}$"
10     },
11     "firstName": {
12       "type": "string"
13     },
14     "lastName": {
15       "type": "string"
16     }
17   },
18   "additionalProperties": false,
19   "required": ["email", "firstName", "lastName"]
20 }
```

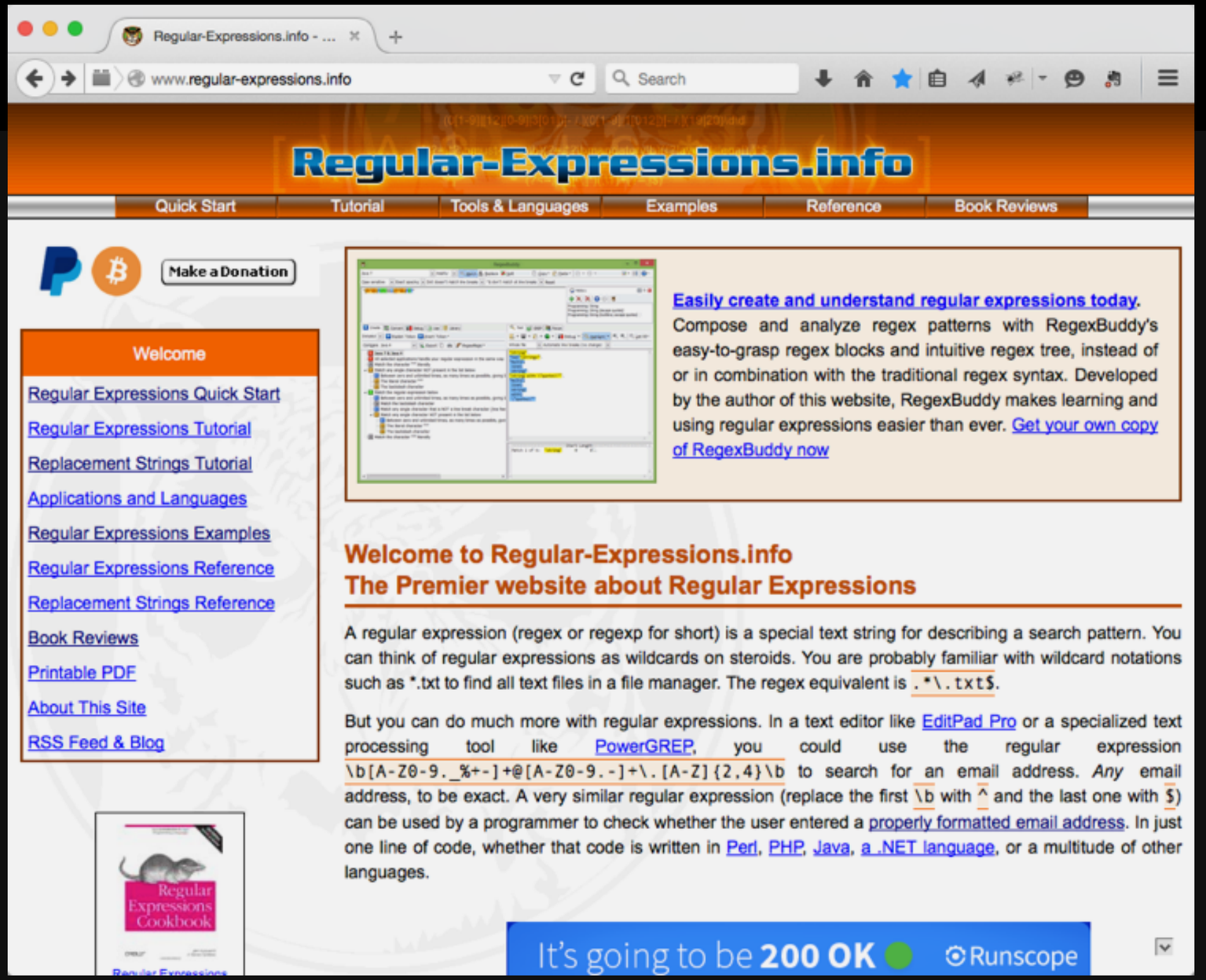
Regex - JSON Doc

```
2  {
3     "email": "larsonrichard@ecratic.com",
4     "firstName": "Larson",
5     "lastName": " Richard"
6  }
```

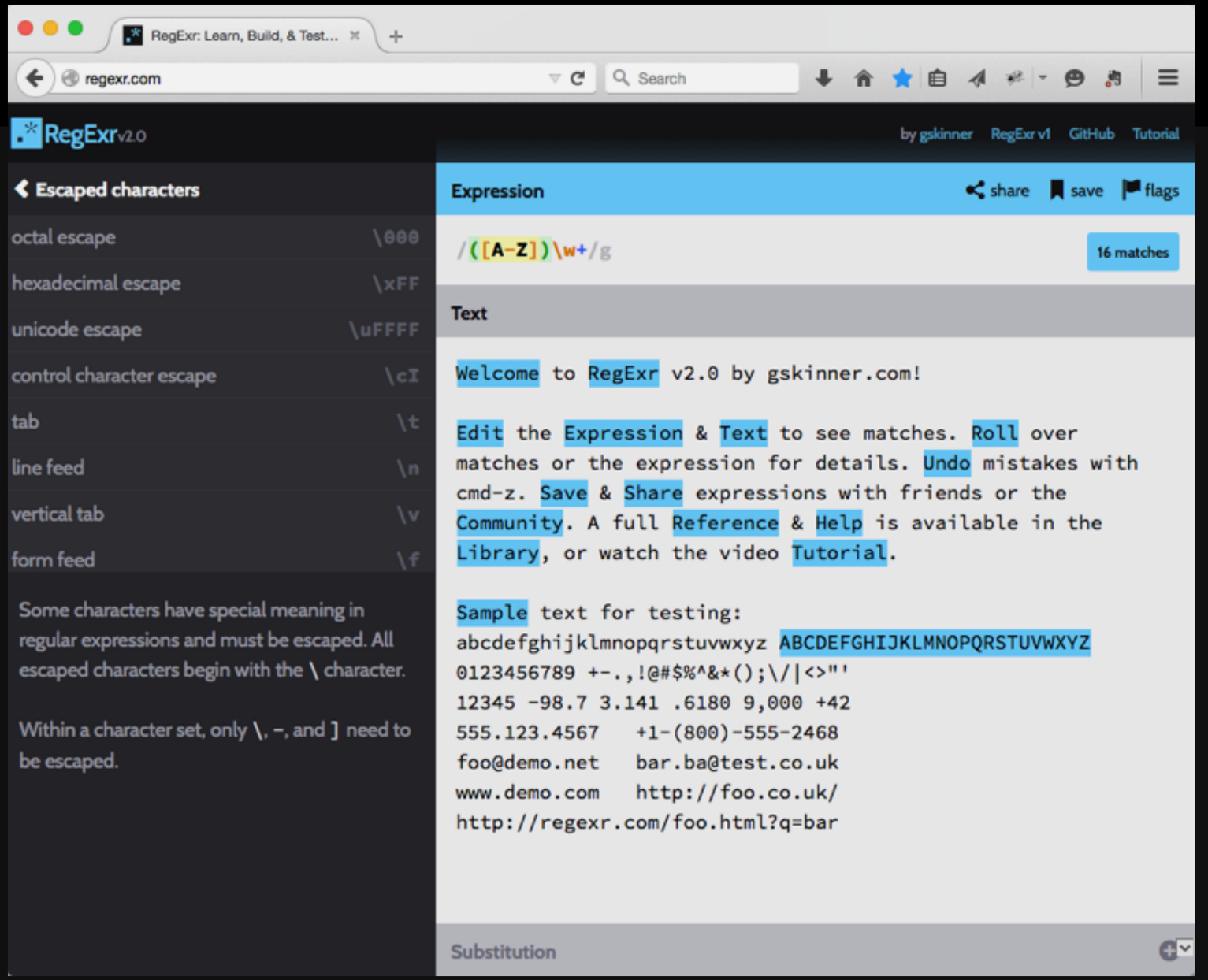
Help!! I’m Awful at Regexp! - Regex101



Regular Expressions Info



Regexr

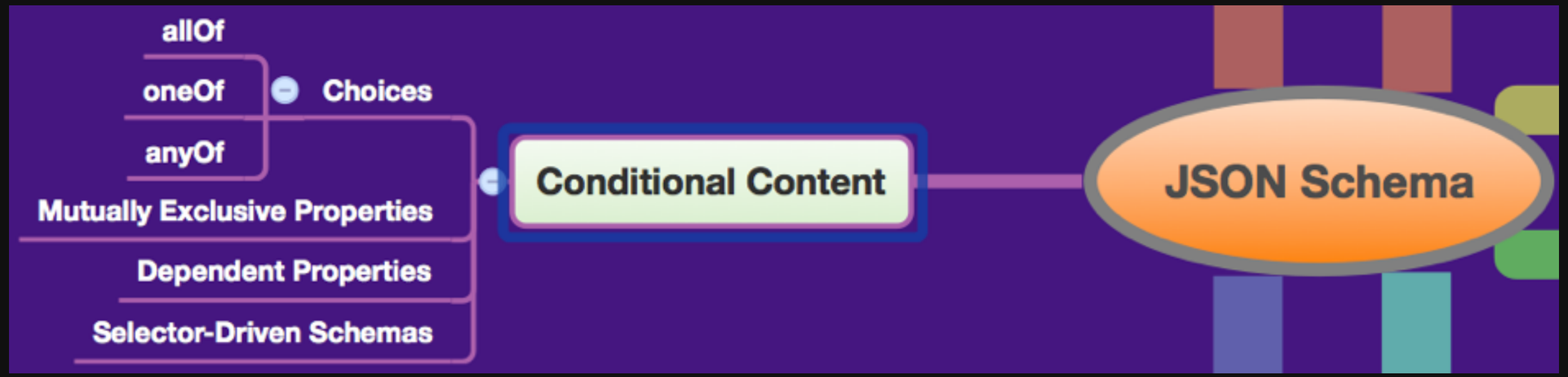

RegExr: Learn, Build, & Test...
regexr.com
Search
RegExr v2.0
by gskinner
RegExr v1
GitHub
Tutorial
Escaped characters
octal escape \000
hexadecimal escape \xFF
unicode escape \uFFFF
control character escape \cI
tab \t
line feed \n
vertical tab \v
form feed \f
Some characters have special meaning in regular expressions and must be escaped. All escaped characters begin with the \ character.
Within a character set, only \, -, and] need to be escaped.
Expression
share
save
flags
/([A-Z])\w+/g
16 matches
Text
Welcome to RegExr v2.0 by gskinner.com!
Edit the Expression & Text to see matches. Roll over matches or the expression for details. Undo mistakes with cmd-z. Save & Share expressions with friends or the Community. A full Reference & Help is available in the Library, or watch the video Tutorial.
Sample text for testing:
abcdefghijklmnopqrstuvwxyz ABCDEFGHIJKLMNOPQRSTUVWXYZ
0123456789 +-.,!@#$%^&*();\/|<>"'
12345 -98.7 3.141 .6180 9,000 +42
555.123.4567 +1-(800)-555-2468
foo@demo.net bar.ba@test.co.uk
www.demo.com http://foo.co.uk/
http://regexr.com/foo.html?q=bar
Substitution

Project 2 - Schema Constraints and Conditional Content

projects/README.md#project-2---schema-constraints-and-conditional-content

projects/EspressoCON.md

JSON Schema - Conditional Content



Dependent Properties - JSON Schema

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "properties": {
    "email": {
      "type": "string"
    },
    "firstName": {
      "type": "string"
    },
    "lastName": {
      "type": "string"
    },
    "tags": {
      "type": "array",
      "items": {
        "type": "string"
      },
      "additionalItems": false
    },
    "favoriteTopic": {
      "type": "string"
    }
  },
  "additionalProperties": false,
  "required": ["email", "firstName", "lastName"],
  "dependencies": {
    "tags": ["favoriteTopic"]
  }
}
```

Dependent Properties - JSON Doc

```
2  {
3    "email": "larsonrichard@ecratic.com",
4    "firstName": "Larson",
5    "lastName": " Richard",
6    "tags": [
7      "JavaScript", "AngularJS", "Yeoman"
8    ],
9    "favoriteTopic": "JavaScript"
10 }
```

allOf / anyOf / oneOf

`allOf`	All must match successfully
`anyOf`	One or more to match successfully
`oneOf`	One, and only one, to match successfully

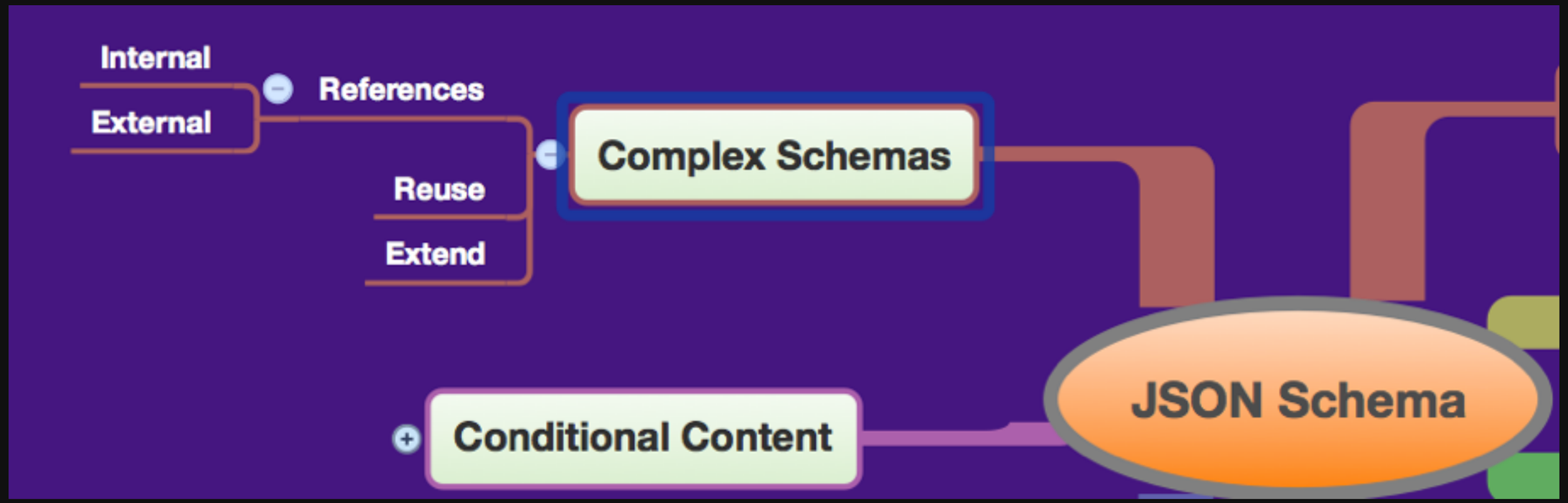
anyOf - JSON Schema

```
2  {
3    "$schema": "http://json-schema.org/draft-04/schema#",
4    "type": "object",
5    "properties": {
6      "email": { "type": "string" },
7      "firstName": { "type": "string" },
8      "lastName": { "type": "string" },
9      "postedSlides": {
10       "anyOf": [
11         { "type": "boolean" },
12         { "type": "string",
13           "enum": ["yes", "Yes", "no", "No"]
14         }
15       ]
16     },
17     "rating": { "type": "number" }
18   },
19   "additionalProperties": false,
20   "required": [ "email", "firstName", "lastName", "postedSlides", "rating"]
21 }
```

anyOf - JSON Doc

```
2  {
3    "email": "larsonrichard@ecratic.com",
4    "firstName": "Larson",
5    "lastName": " Richard",
6    "postedSlides": "yes",
7    "rating": 4.1
8  }
```

Complex Schemas



References (Internal) - JSON Schema

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "properties": {
    "email": {
      "$ref": "#/definitions/emailPattern"
    },
    "firstName": {
      "type": "string"
    },
    "lastName": {
      "type": "string"
    }
  },
  "additionalProperties": false,
  "required": ["email", "firstName", "lastName"],
  "definitions": {
    "emailPattern": {
      "type": "string",
      "pattern": "^[\\w|-]+@[\\w|-]+\\.[A-Za-z]{2,4}$"
    }
  }
}
```

References (Internal) - JSON Doc

```
2  {
3    "email": "larsonrichard@ecratic.com",
4    "firstName": "Larson",
5    "lastName": " Richard"
6  }
```

tinyserver

```
npm install -g tinyserver


tinyserver 8081
```

References (External) - JSON Schema

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "properties": {
    "email": {
      "$ref":
"http://localhost:8081/ex-14-my-common-schema.json#/definitions/emailPattern"
    },
    "firstName": {
      "type": "string"
    },
    "lastName": {
      "type": "string"
    }
  },
  "additionalProperties": false,
  "required": ["email", "firstName", "lastName"]
}
```

References (External) - JSON Schema

```
2  {
3    "$schema": "http://json-schema.org/draft-04/schema#",
4    "id": "http://localhost:8081/ex-14-my-common-schema.json",
5
6    "definitions": {
7      "emailPattern": {
8        "type": "string",
9        "pattern": "^[\\w|-]+@[\\w|-]+\\.[A-Za-z]{2,4}$"
10     }
11   }
12 }
```

References (External) - JSON Doc

```
{
  "email": "larsonrichard@ecratic.com",
  "firstName": "Larson",
  "lastName": " Richard"
}
```

Where Are We?

JSON Schema Overview	1	JSON Search & Transform Overview	4
Core JSON Schema	2	JSON Search	5
API Design with JSON Schema	3	JSON Transform	6

Real World Use Case

Design/Implement API and Consumer
in Parallel

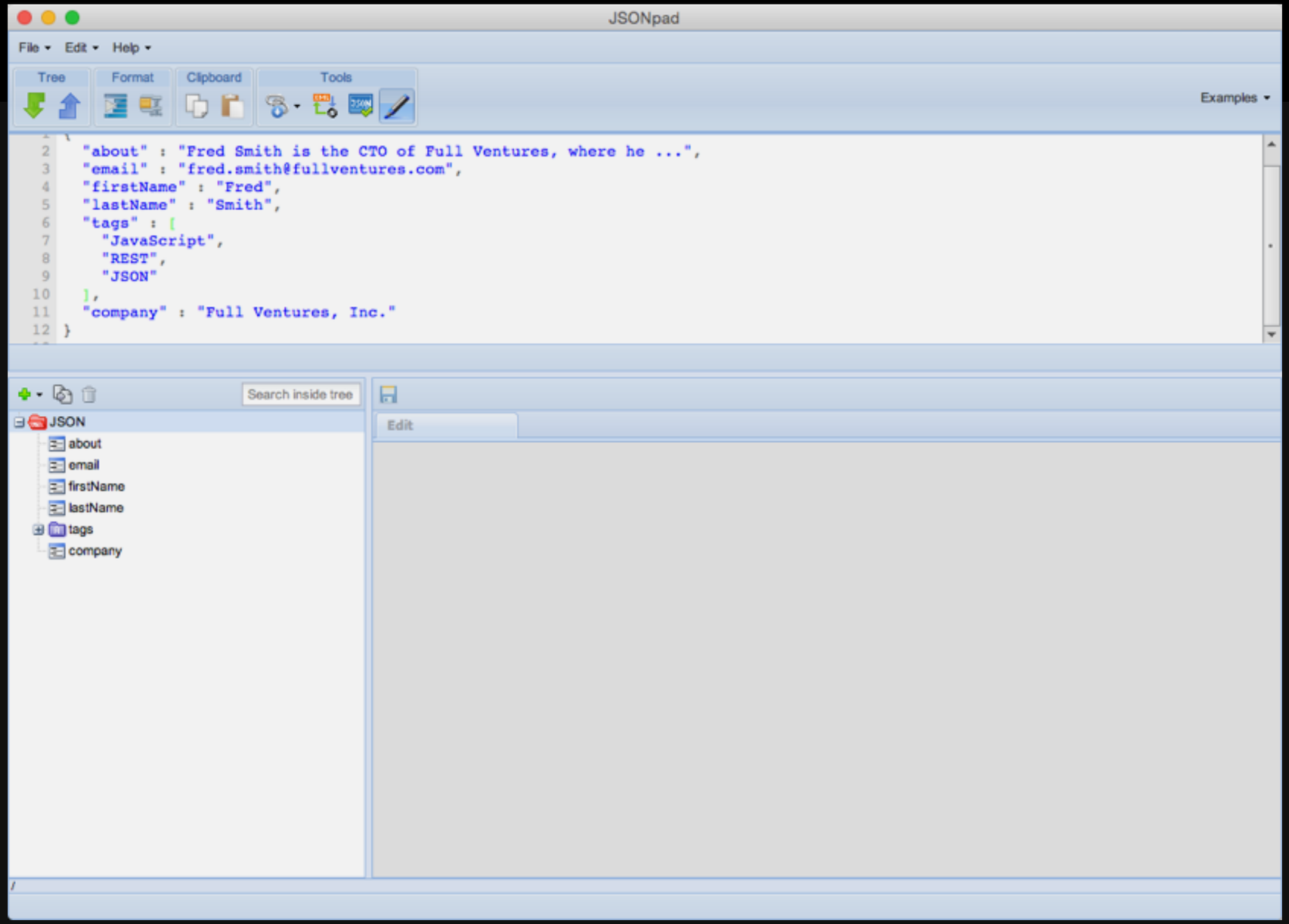
Our Scenario

Leverage JSON Schema to create a Stub REST API … without any code

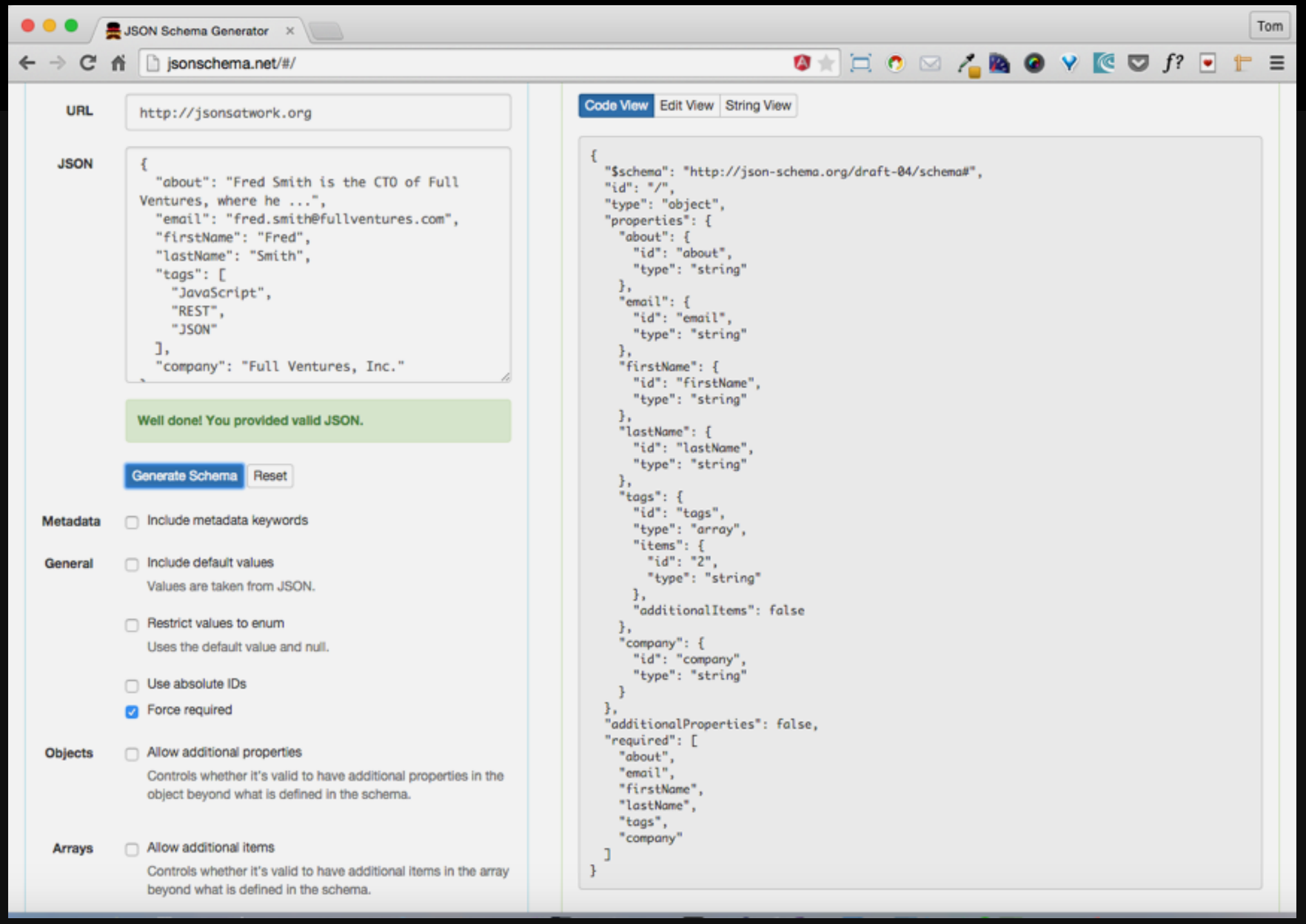
My JSON Schema Workflow for APIs

Model JSON Document	Generate JSON Schema	Validate JSON Document	Document JSON Schema (HTML)
JSONPad https://code.google.com/p/json-pad/	JSON Schema Generator http://jsonschema.net/	JSON Validate http://jsonvalidate.com/	Matic https://github.com/mattyod/matic/
		JSONLint http://jsonlint.com	Docson https://github.com/lbovet/docson

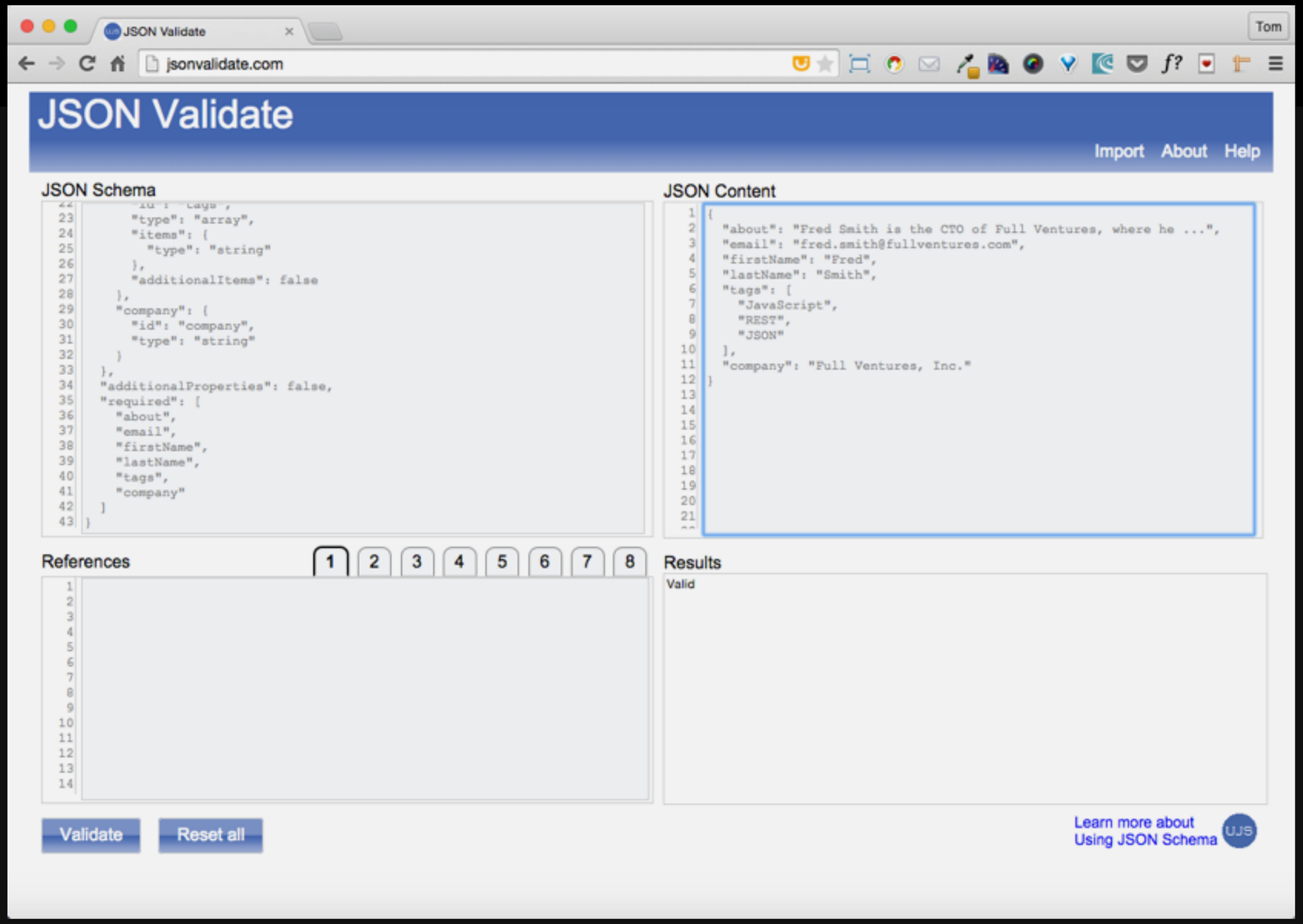
JSONPad Demo



JSON Schema Generator Demo



JSON Validate Demo



Matic

```
npm install -g matic

npm install -g jade


matic
```

https://github.com/mattyod/matic

https://github.com/mattyod/matic-draft4-example

Matic - Speaker Schema

Speaker schema.

Verbose version of the Speaker schema.

Uses http://json-schema.org/draft-04/schema#

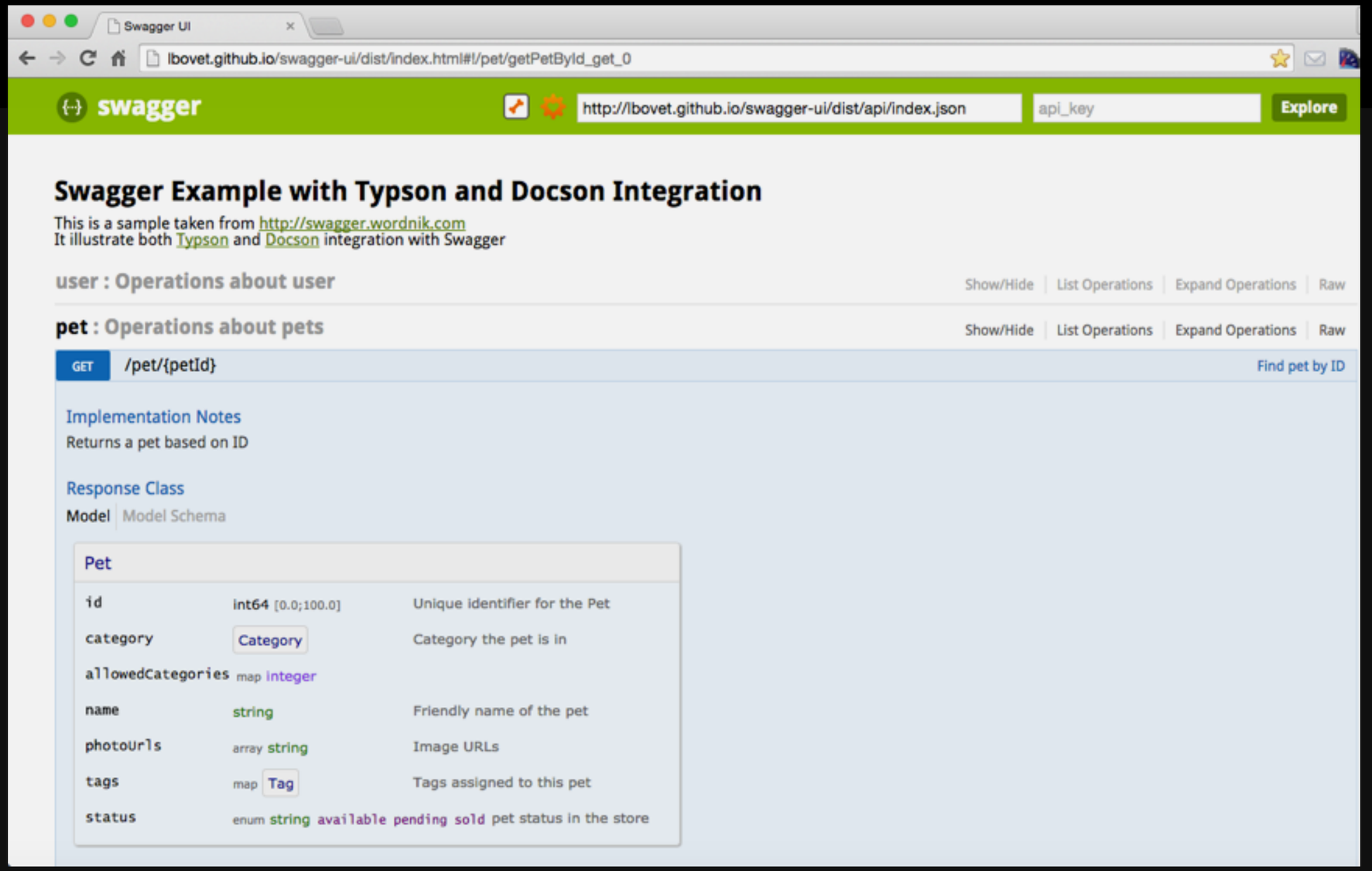
Return to index

```
{
"properties":
  *"about":
      "id":                 about
      "type":               string
      "title":              About schema.
      "description":        The speaker's bio.
      "name":               about
  *"email":
  *"firstName":
  *"lastName":
  *"tags":
  *"company":
"required":                 [ about, email, firstName, lastName, tags, company ]
}
```

Return to index

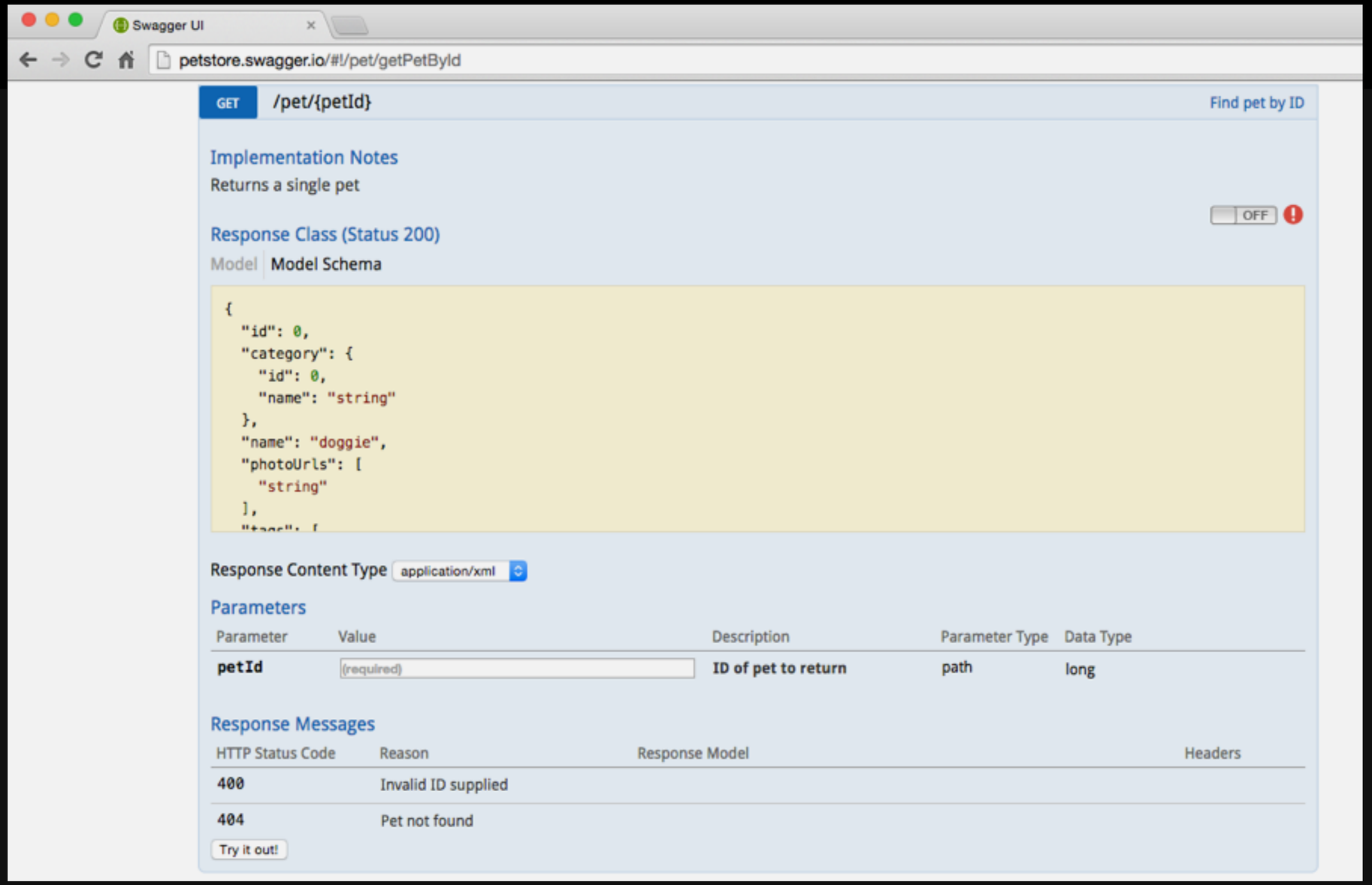
Built with Matic.js

Docson - Swagger Petstore



https://github.com/lbovet/docson

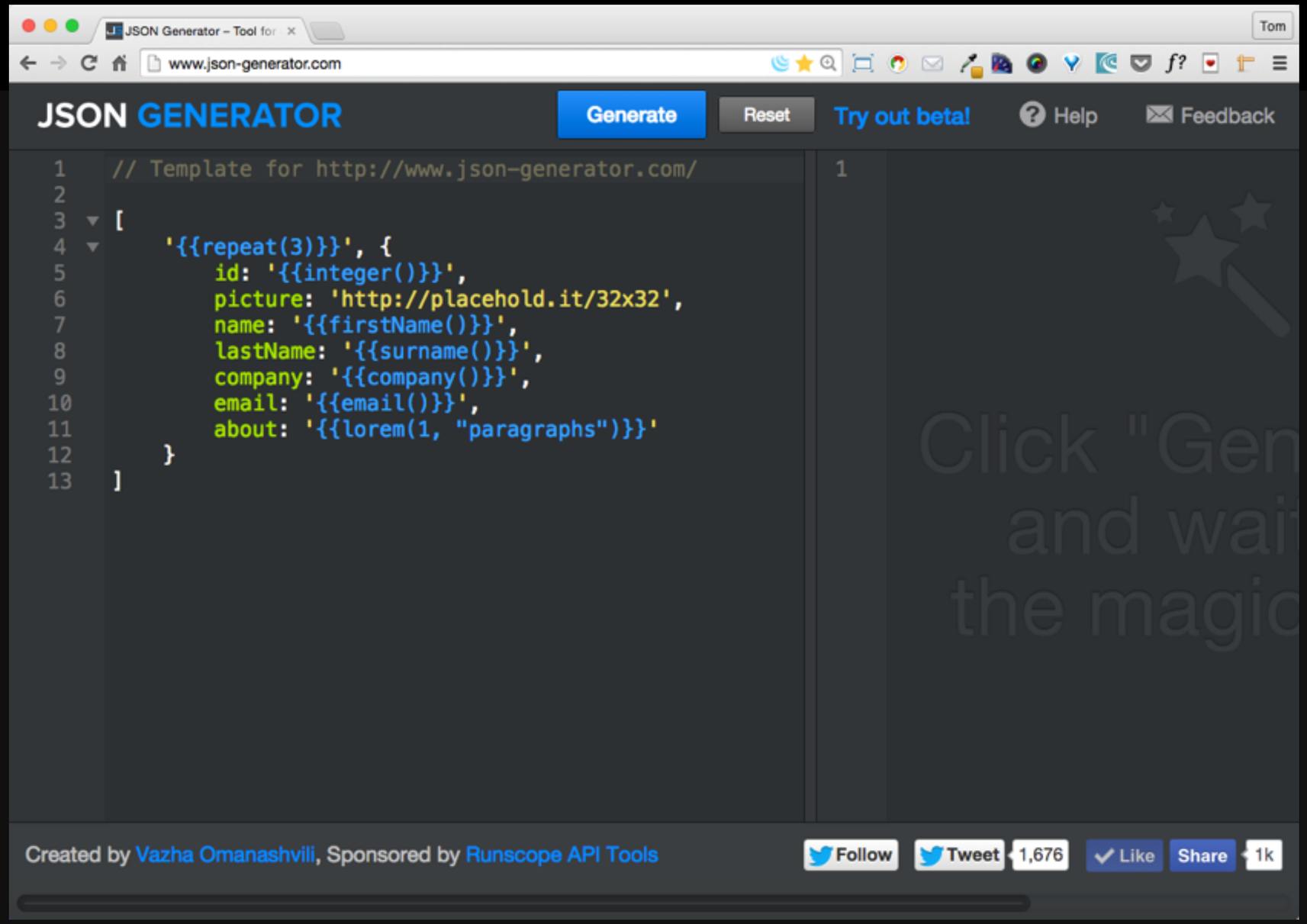
Swagger Petstore - Regular



My JSON Stub API Workflow

Model JSON Document	Generate JSON Document	Deploy Stub REST API
JSONPad	JSON Generator	JSON Server
https://code.google.com/p/json-pad/	http://www.json-generator.com/	https://github.com/typicode/json-server

JSON Generator Demo



JSON Server - Speakers

```
npm install -g json-server
```

```
json-server -p 5000./speakers.json
```

```
http://localhost:5000/speakers
```

https://github.com/typicode/json-server

Project 3 - API Modeling with Schema

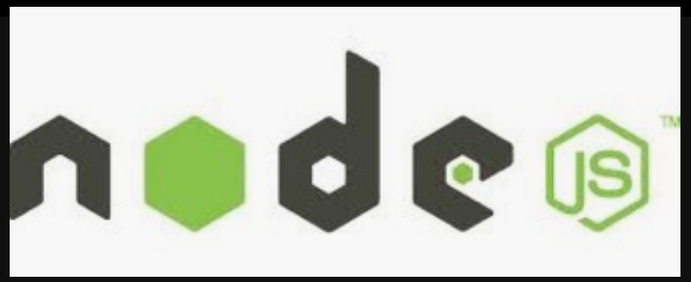
projects/README.md#project-3---api-modeling-with-schema

projects/EspressoCON.md

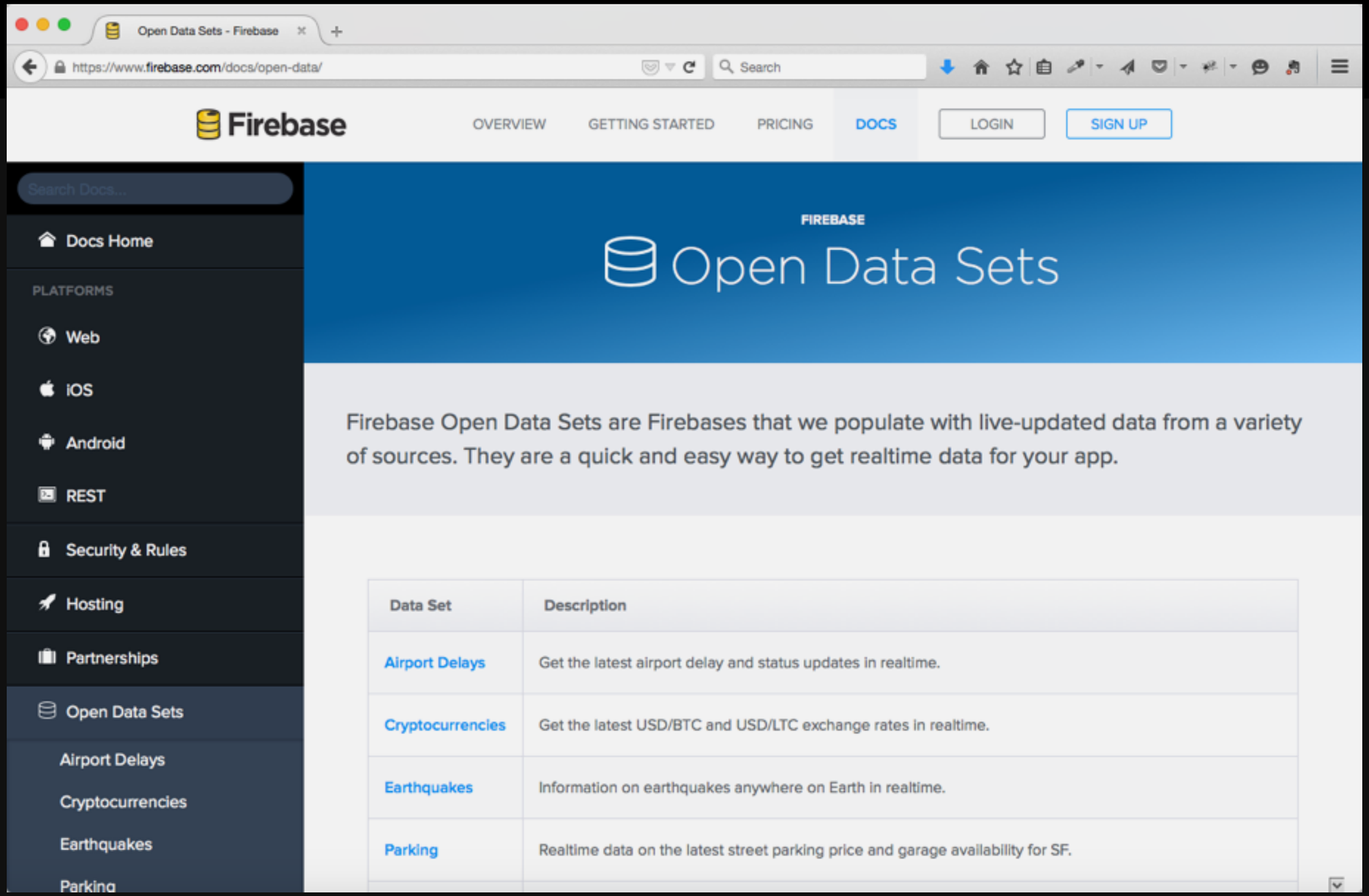
Where Are We?

JSON Schema Overview	1	**JSON Search & Transform Overview**	**4**
Core JSON Schema	2	JSON Search	5
API Design with JSON Schema	3	JSON Transform	6

Our Client Stack



Firebase Open Data Set

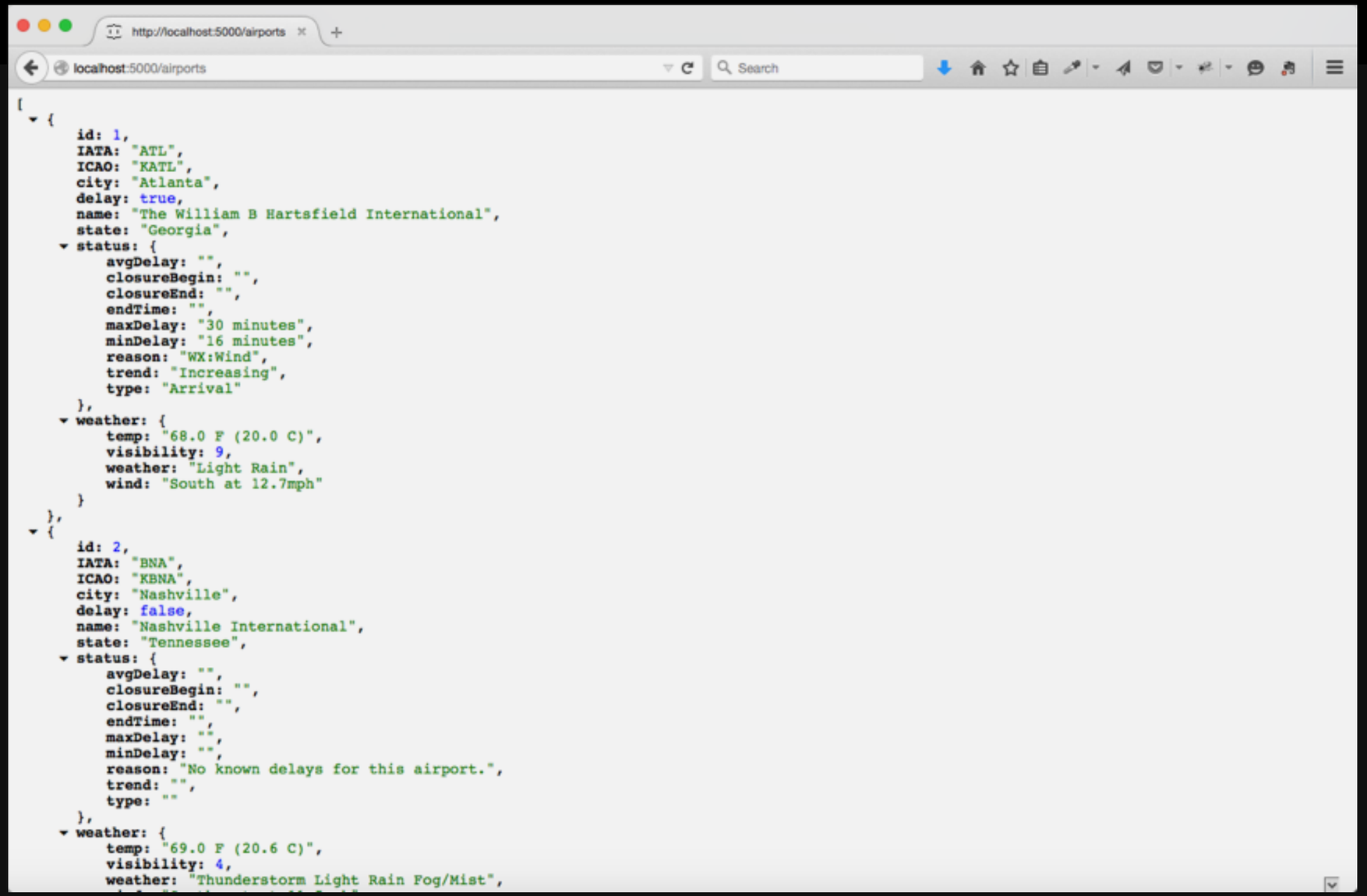


JSON Server - Airports

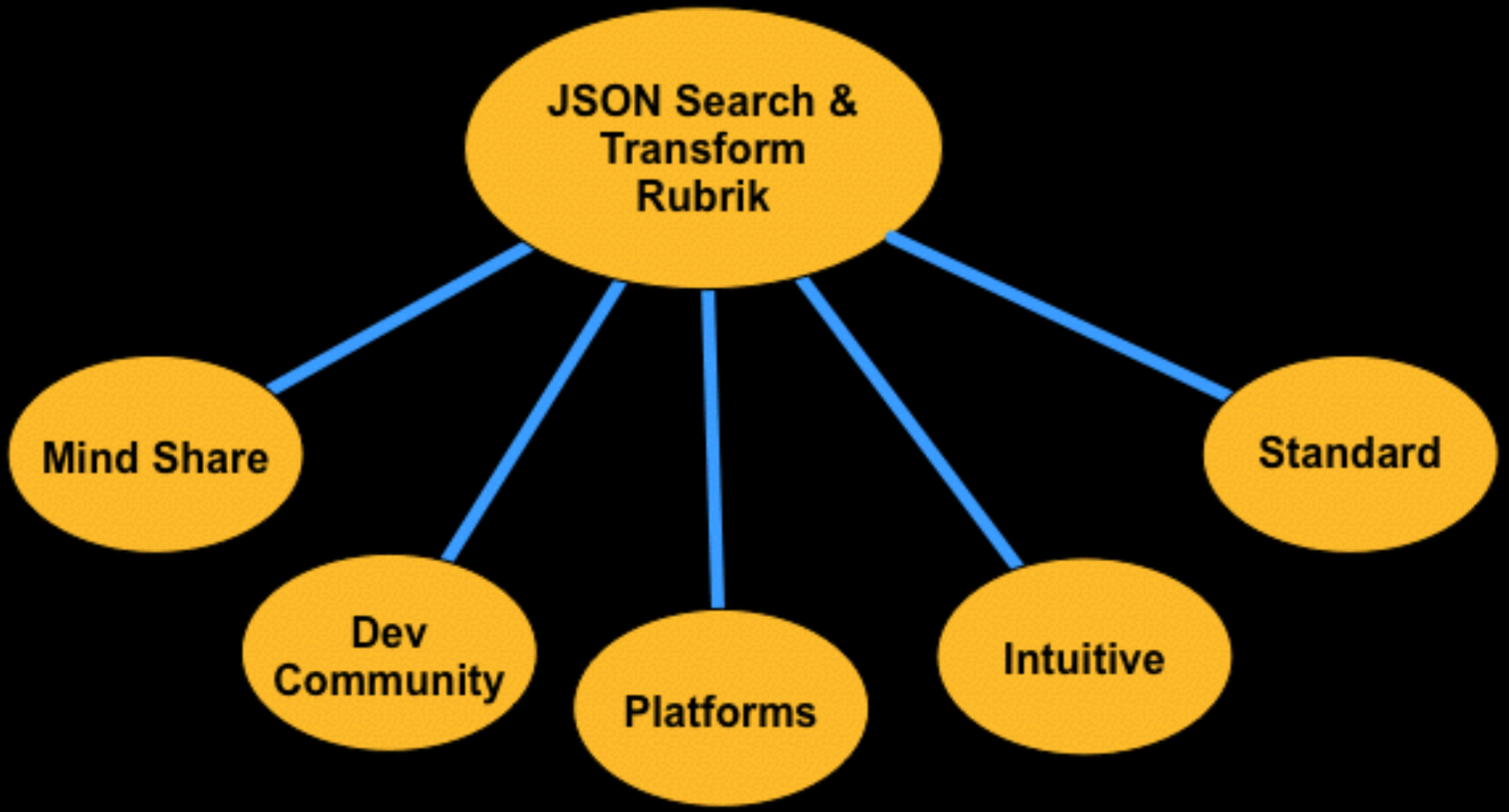
```
json-server -p 5000./airports.json
```

http://localhost:5000/airports

Airports Stub Service



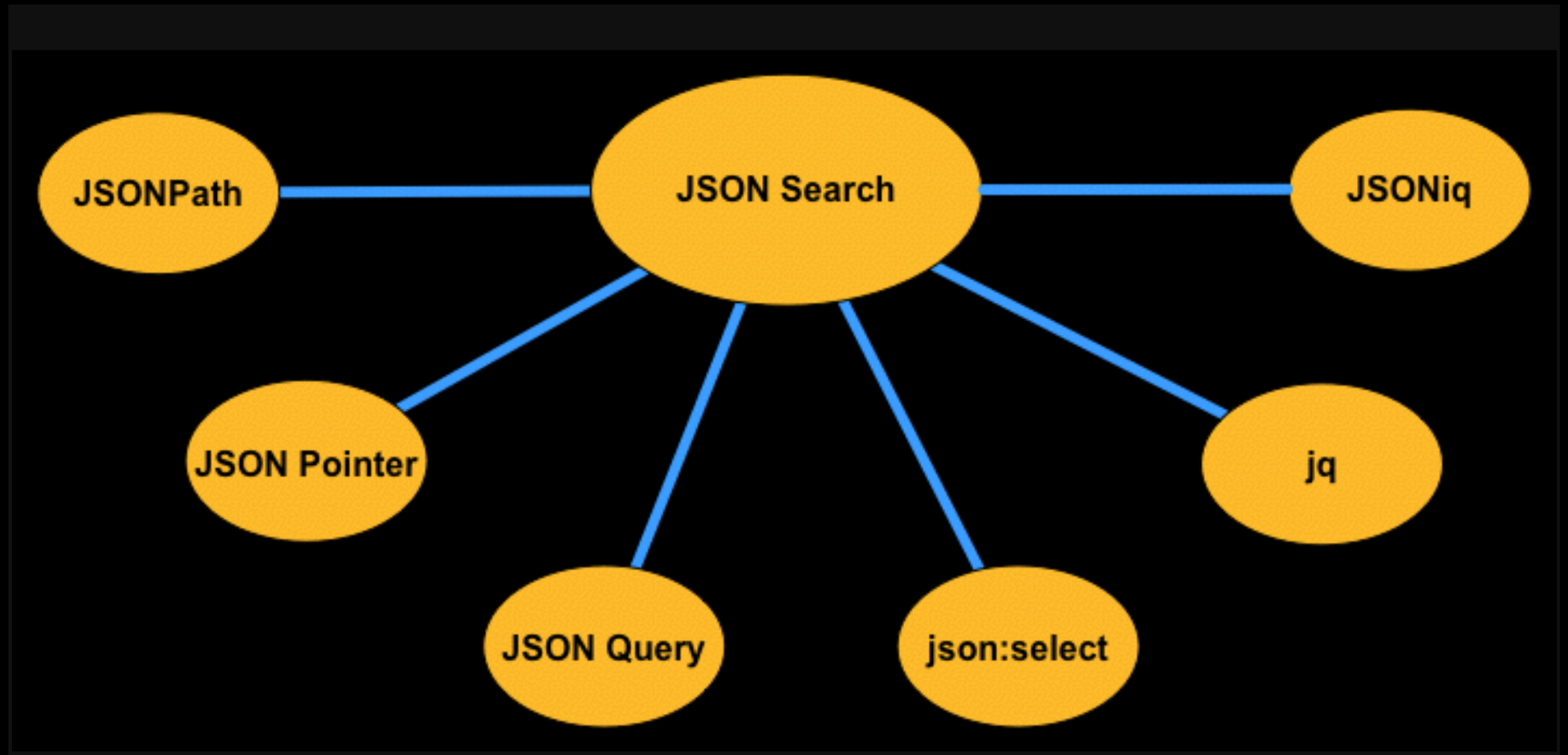
JSON Search & Transform Rubrik



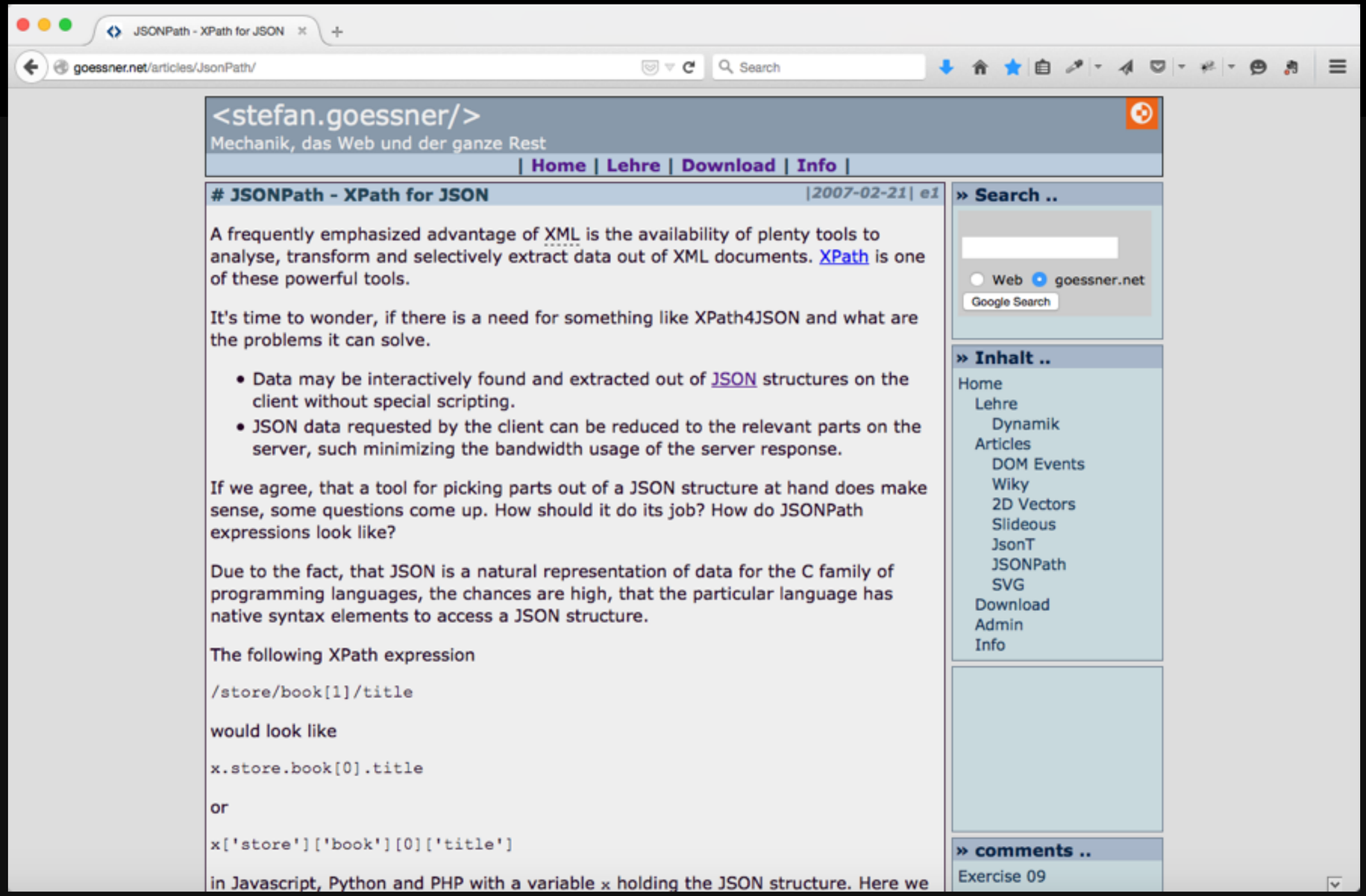
Where Are We?

JSON Schema Overview 1	JSON Search & Transform Overview 4
Core JSON Schema 2	**JSON Search 5**
API Design with JSON Schema 3	JSON Transform 6

JSON Search Tools



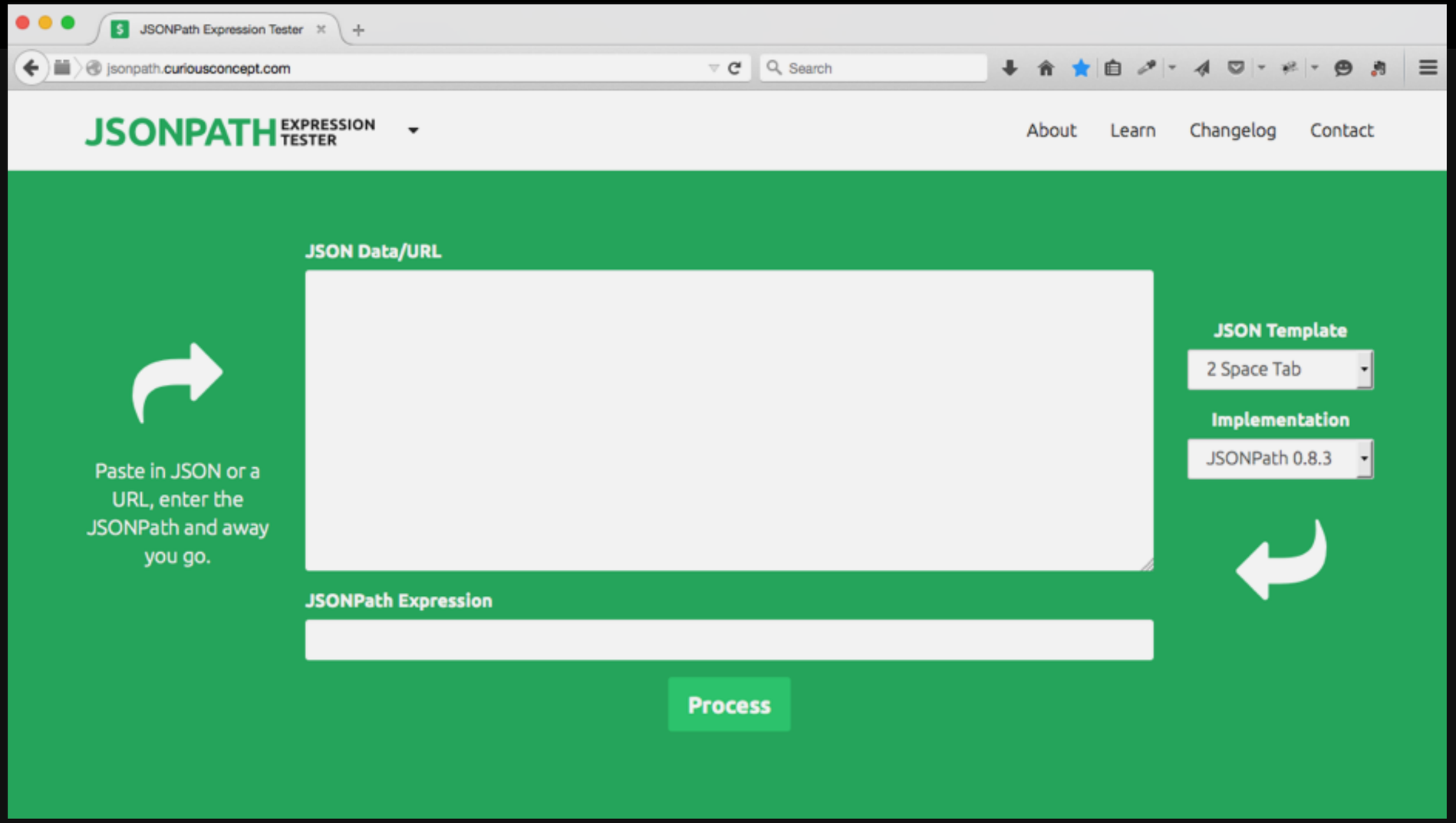
JSONPath


JSONPath - XPath for JSON
goessner.net/articles/JsonPath/
Search
<stefan.goessner/>
Mechanik, das Web und der ganze Rest
| Home | Lehre | Download | Info |
JSONPath - XPath for JSON
|2007-02-21| e1
A frequently emphasized advantage of XML is the availability of plenty tools to analyse, transform and selectively extract data out of XML documents. XPath is one of these powerful tools.
It's time to wonder, if there is a need for something like XPath4JSON and what are the problems it can solve.
Data may be interactively found and extracted out of JSON structures on the client without special scripting.
JSON data requested by the client can be reduced to the relevant parts on the server, such minimizing the bandwidth usage of the server response.
If we agree, that a tool for picking parts out of a JSON structure at hand does make sense, some questions come up. How should it do its job? How do JSONPath expressions look like?
Due to the fact, that JSON is a natural representation of data for the C family of programming languages, the chances are high, that the particular language has native syntax elements to access a JSON structure.
The following XPath expression
/store/book[1]/title
would look like
x.store.book[0].title
or
x['store']['book'][0]['title']
in Javascript, Python and PHP with a variable x holding the JSON structure. Here we
» Search ..
Web
goessner.net
Google Search
» Inhalt ..
Home
Lehre
Dynamik
Articles
DOM Events
Wiky
2D Vectors
Slideous
JsonT
JSONPath
SVG
Download
Admin
Info
» comments ..
Exercise 09

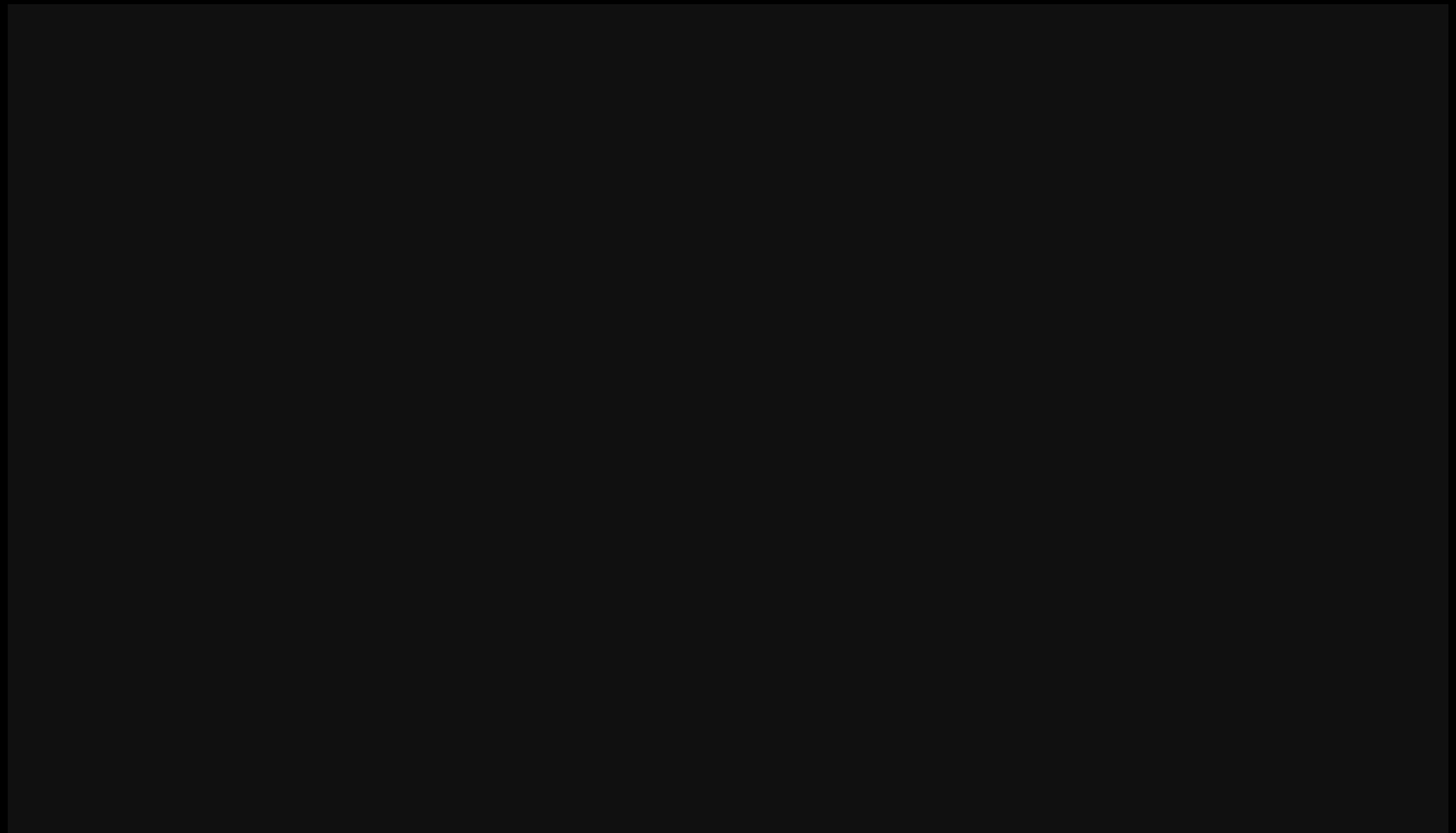
JSONPath Syntax

XPath	JSONPath	Result
`/store/book/author`	`$.store.book[*].author`	the authors of all books in the store
`//author`	`$..author`	all authors
`/store/*`	`$.store.*`	all things in store, which are some books and a red bicycle.
`/store//price`	`$.store..price`	the price of everything in the store.
`//book[3]`	`$..book[2]`	the third book
`//book[last()]`	`$..book[(@.length-1)]` `$..book[-1:]`	the last book in order.
`//book[position()<3]`	`$..book[0,1]` `$..book[:2]`	the first two books
`//book[isbn]`	`$..book[?(@.isbn)]`	filter all books with isbn number
`//book[price<10]`	`$..book[?(@.price<10)]`	filter all books cheapier than 10
`//*`	`$..*`	all Elements in XML document. All members of JSON structure.

JSONPath Expression Tester



JSONPath Demo



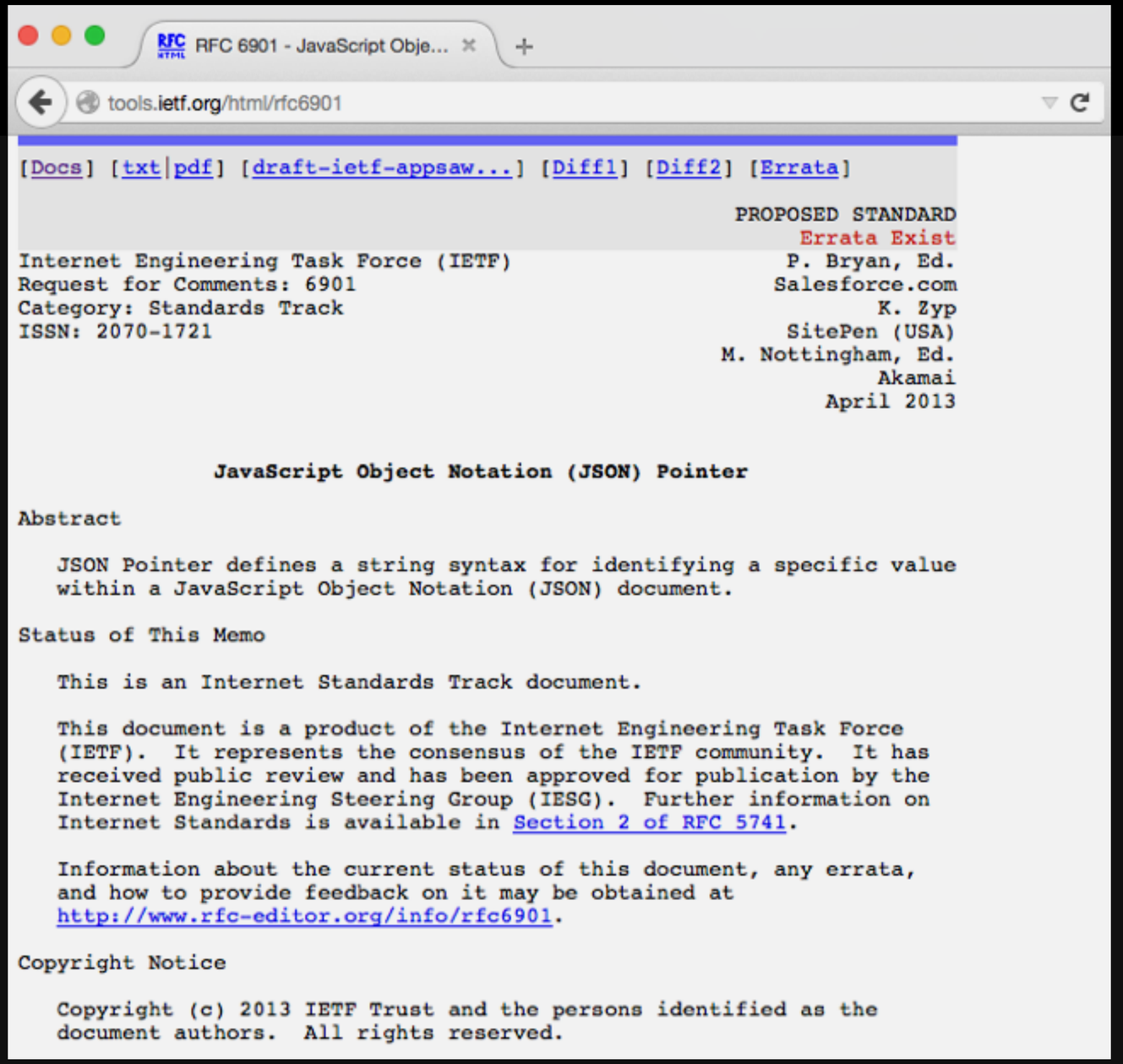
JSONPath Test

```
2  var expect = require('chai').expect;
3  var request = require('request');
4  var jp = require('jsonpath');
5
6  describe('jsonpath', function() {
7    describe('api', function() {
8      it('should return 200', function(done) {
9        var options = {
10         url: 'http://localhost:5000/airports',
11         headers: {
12           'Content-Type': 'application/json'
13         }
14       };
15       request.get(options, function(err, res, body) {
16         expect(res.statusCode).to.equal(200);
17         console.log('\n\n\n\nJSONPath Test');
18         var obj = JSON.parse(res.body);
19         console.log('\n\n1st & 3rd Object weather: ');
20         console.log(jp.query(obj, '$[0,2].weather'));
21         console.log('\n\nAll Airport Codes: ');
22         console.log(jp.query(obj, '$..IATA'));
23         done();
24       });
25     });
26   });
27 });
```

JSONPath Scorecard

Mindshare	Y
Dev Community	Y
Platforms	JS, Node.js, Java, RoR
Intuitive	Y
Standard	N

JSON Pointer



[Docs] [txt|pdf] [draft-ietf-appsaw...] [Diff1] [Diff2] [Errata]

PROPOSED STANDARD
Errata Exist

Internet Engineering Task Force (IETF)
Request for Comments: 6901
Category: Standards Track
ISSN: 2070-1721

P. Bryan, Ed.
Salesforce.com
K. Zyp
SitePen (USA)
M. Nottingham, Ed.
Akamai
April 2013

JavaScript Object Notation (JSON) Pointer

Abstract

JSON Pointer defines a string syntax for identifying a specific value within a JavaScript Object Notation (JSON) document.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc6901.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

JSON Pointer Syntax

For example, given the JSON document

```
{
   "foo": ["bar", "baz"],
   "": 0,
   "a/b": 1,
   "c%d": 2,
   "e^f": 3,
   "g|h": 4,
   "i\\j": 5,
   "k\"l": 6,
   " ": 7,
   "m~n": 8
}
```

The following JSON strings evaluate to the accompanying values:

```
 ""           // the whole document
 "/foo"       ["bar", "baz"]
 "/foo/0"     "bar"
 "/"          0
 "/a~1b"      1
 "/c%d"       2
 "/e^f"       3
 "/g|h"       4
 "/i\\j"      5
 "/k\"l"      6
 "/ "         7
 "/m~0n"      8
```

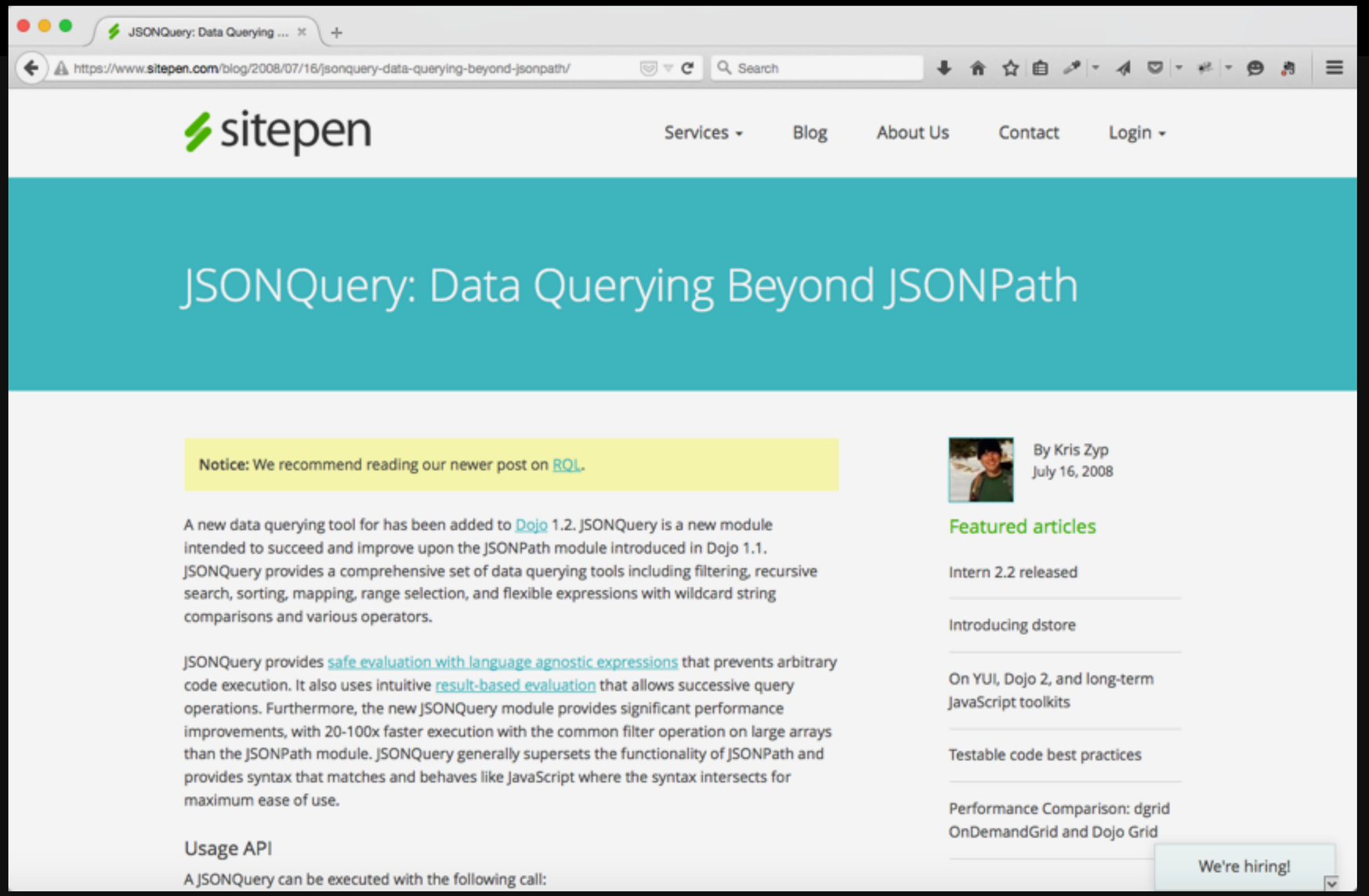
JSON Pointer Test

```
2  var expect = require('chai').expect;
3  var request = require('request');
4  var pointer = require('json-pointer');
5
6  describe('json-pointer', function() {
7    describe('api', function() {
8      it('should return 200', function(done) {
9        var options = {
10         url: 'http://localhost:5000/airports',
11         headers: {
12           'Content-Type': 'application/json'
13         }
14       };
15       request.get(options, function(err, res, body) {
16         expect(res.statusCode).to.equal(200);
17         var obj = JSON.parse(res.body);
18         console.log('\n\n\n\nJSON Pointer Test');
19         console.log('\n\n1st Object: ');
20         console.log(pointer.get(obj, '/0'));
21         console.log('\nIATA on 2nd Object: ');
22         console.log(pointer.get(obj, '/1/IATA'));
23         done();
24       });
25     });
26   });
27 });
```

JSON Pointer Scorecard

Mindshare	Y
Dev Community	Y
Platforms	JS, Node.js, Java, RoR
Intuitive	Y
Standard	RFC 6901 - Woot!

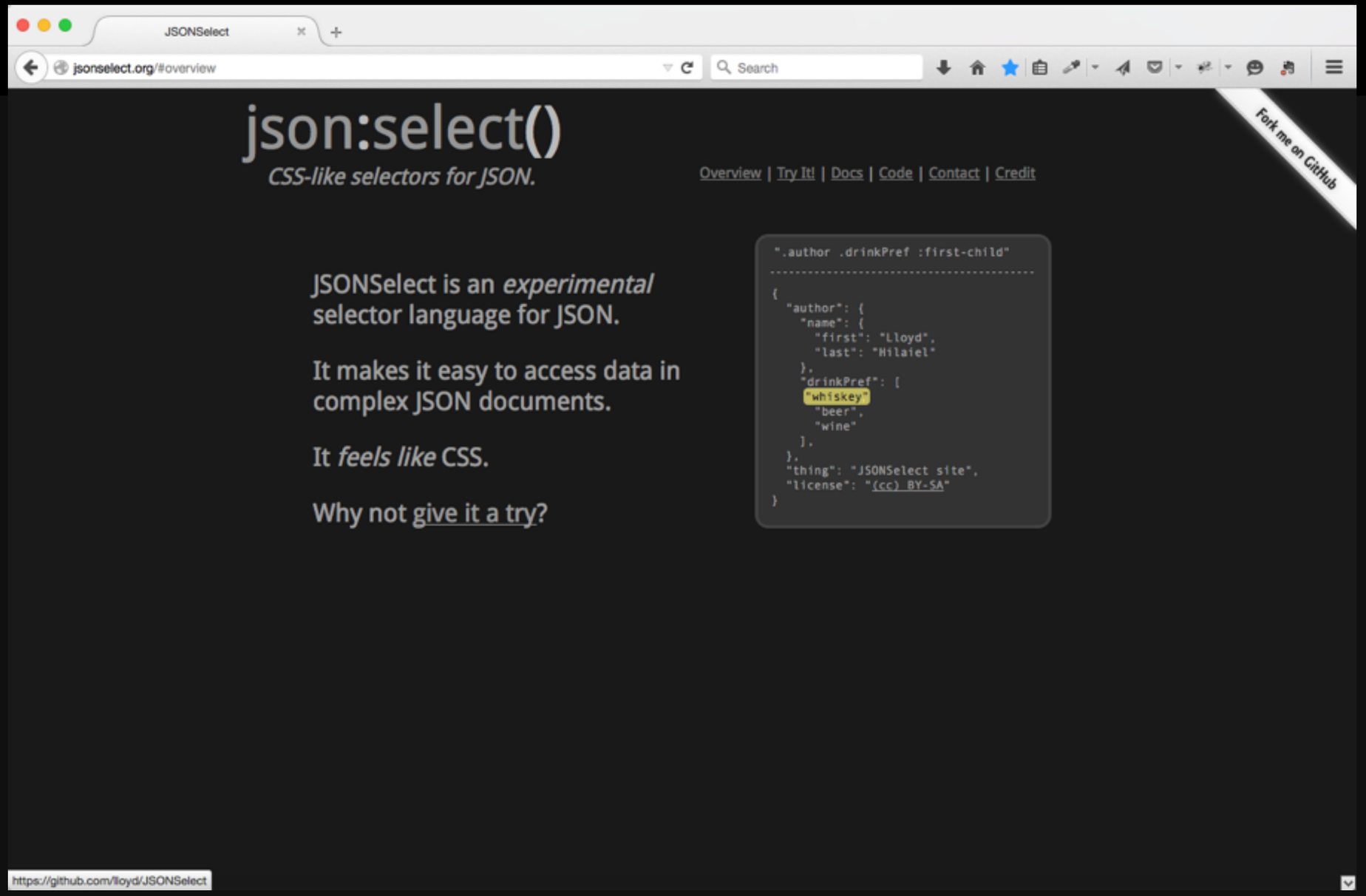
JSON Query


JSONQuery: Data Querying ...
https://www.sitepen.com/blog/2008/07/16/jsonquery-data-querying-beyond-jsonpath/
Search
sitepen
Services
Blog
About Us
Contact
Login
JSONQuery: Data Querying Beyond JSONPath
Notice: We recommend reading our newer post on RQL.
A new data querying tool for has been added to Dojo 1.2. JSONQuery is a new module intended to succeed and improve upon the JSONPath module introduced in Dojo 1.1. JSONQuery provides a comprehensive set of data querying tools including filtering, recursive search, sorting, mapping, range selection, and flexible expressions with wildcard string comparisons and various operators.
JSONQuery provides safe evaluation with language agnostic expressions that prevents arbitrary code execution. It also uses intuitive result-based evaluation that allows successive query operations. Furthermore, the new JSONQuery module provides significant performance improvements, with 20-100x faster execution with the common filter operation on large arrays than the JSONPath module. JSONQuery generally supersets the functionality of JSONPath and provides syntax that matches and behaves like JavaScript where the syntax intersects for maximum ease of use.
Usage API
A JSONQuery can be executed with the following call:
By Kris Zyp
July 16, 2008
Featured articles
Intern 2.2 released
Introducing dstore
On YUI, Dojo 2, and long-term JavaScript toolkits
Testable code best practices
Performance Comparison: dgrid OnDemandGrid and Dojo Grid
We're hiring!

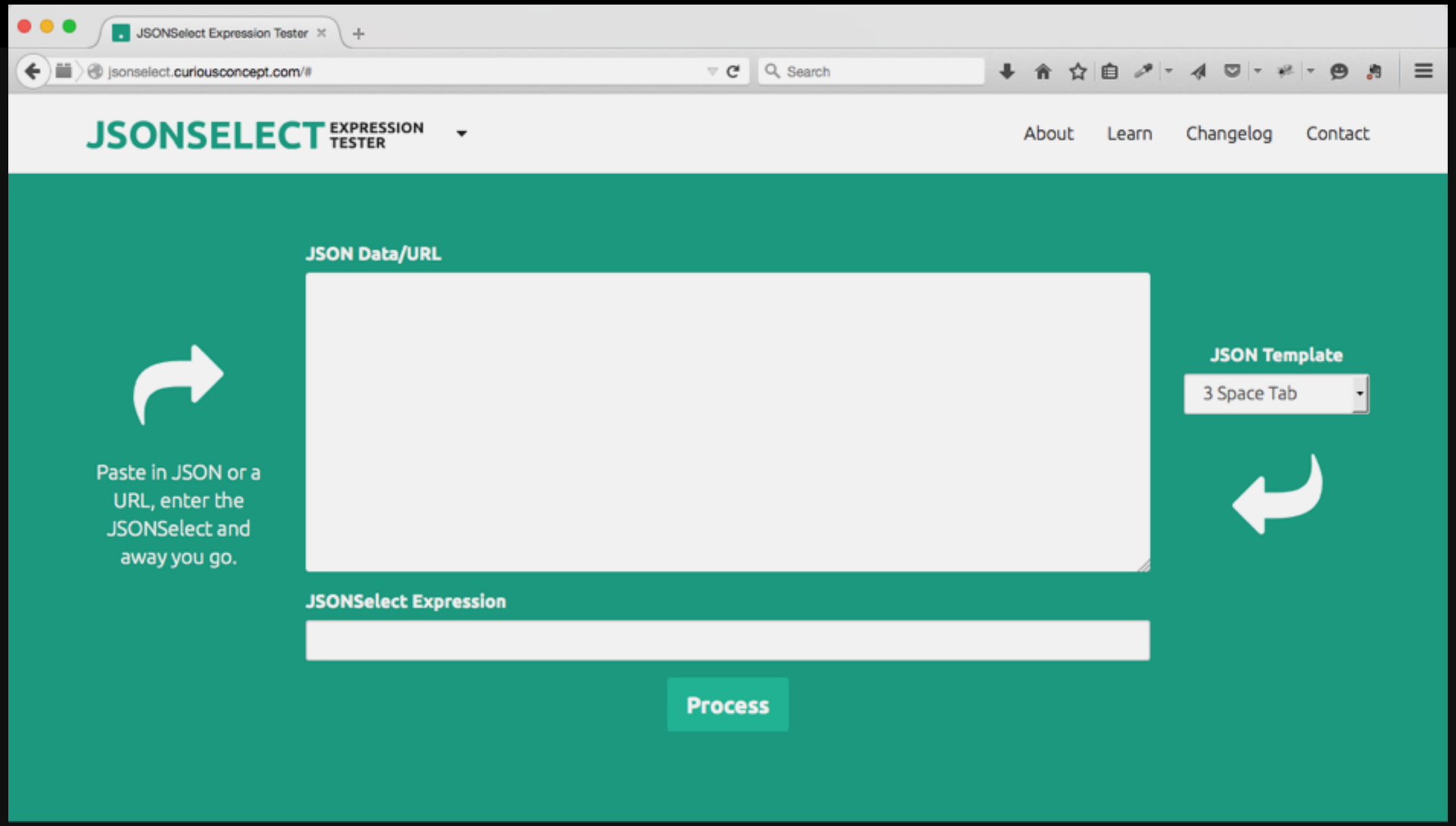
JSON Query Scorecard

Mindshare	N?
Dev Community	N? Possibly Dead
Platforms	JS, Node.js
Intuitive	Y
Standard	N

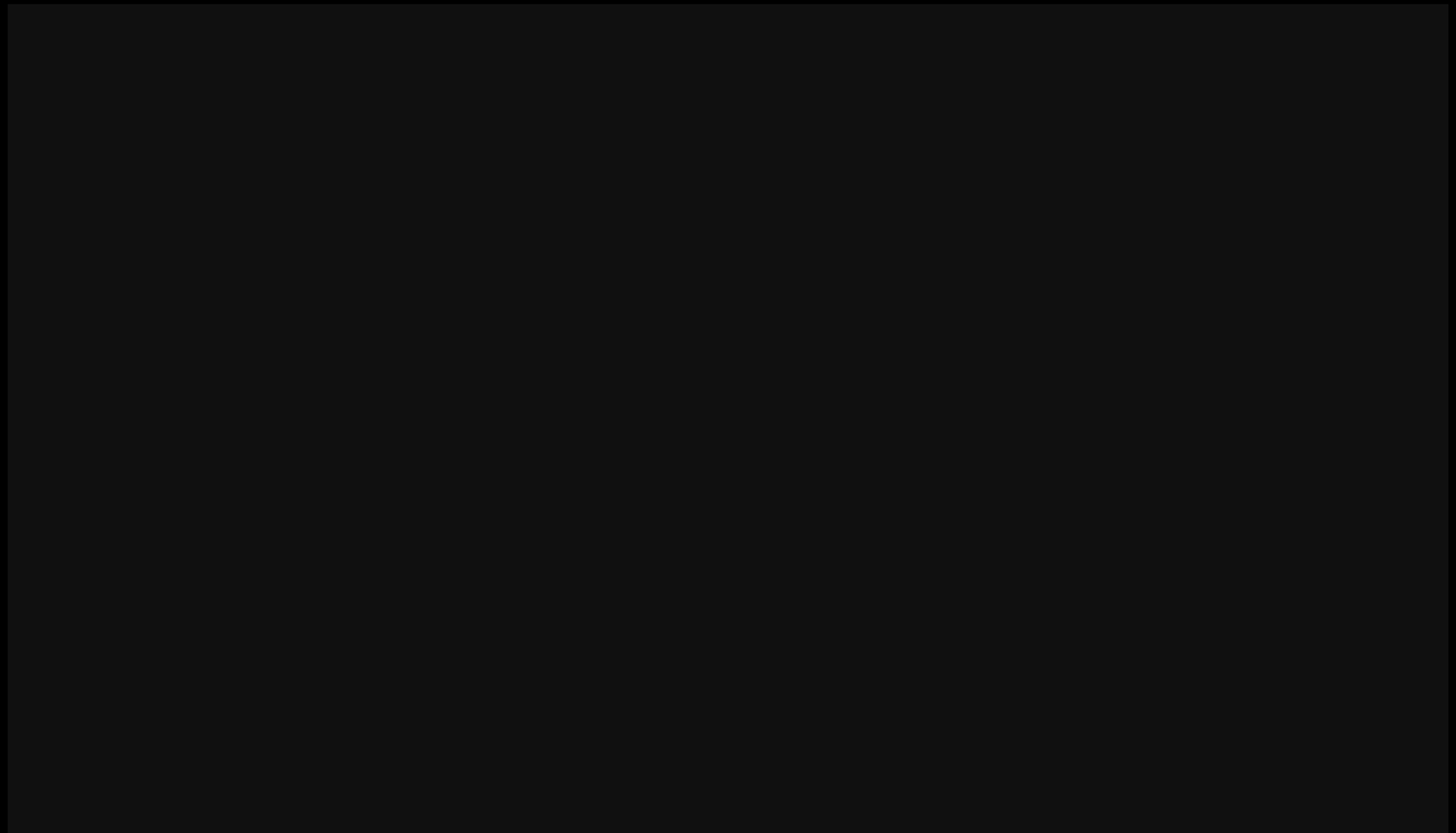
json:select


JSONSelect
jsonselect.org/#overview
Search
json:select()
CSS-like selectors for JSON.
Overview | Try It! | Docs | Code | Contact | Credit
Fork me on GitHub
JSONSelect is an experimental selector language for JSON.
It makes it easy to access data in complex JSON documents.
It feels like CSS.
Why not give it a try?
".author .drinkPref :first-child"
{
"author": {
"name": {
"first": "Lloyd",
"last": "Hilaiel"
},
"drinkPref": [
"whiskey"
"beer",
"wine"
],
},
"thing": "JSONSelect site",
"license": "(cc) BY-SA"
}
https://github.com/lloyd/JSONSelect

json:select Expression Tester



json:select Demo



json:select Test

```
var expect = require('chai').expect;
var request = require('request');
var jsonSelect = require('JSONSelect');

describe('JSONSelect', function() {
  describe('api', function() {
    it('should return 200', function(done) {
      var options = {
        url: 'http://localhost:5000/airports',
        headers: {
          'Content-Type': 'application/json'
        }
      };
      request.get(options, function(err, res, body) {
        expect(res.statusCode).to.equal(200);
        console.log('\n\n\n\nJSONSelect Test');
        var obj = JSON.parse(res.body);
        console.log('\n\n1st & 2nd Object weather: ');
        console.log(jsonSelect.match('.status ~ .weather', obj).slice(0,2));
        console.log('\n\nAll Airport Codes: ');
        console.log(jsonSelect.match('.IATA', obj));
        done();
      });
    });
  });
});
```

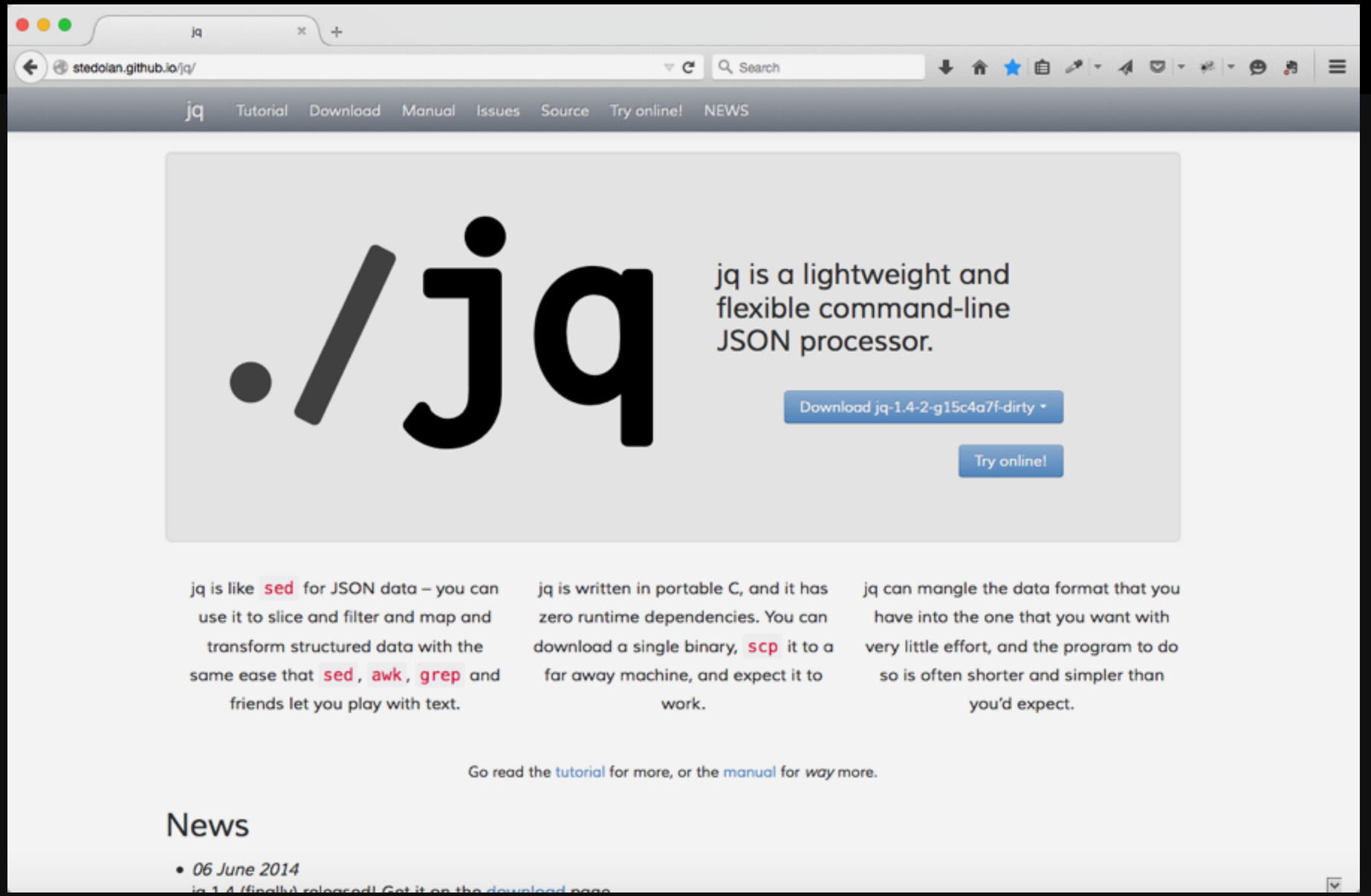
json:select Scorecard

Mindshare	Y?
Dev Community	Y
Platforms	JS, Node.js, RoR
Intuitive	Y - CSS
Standard	N

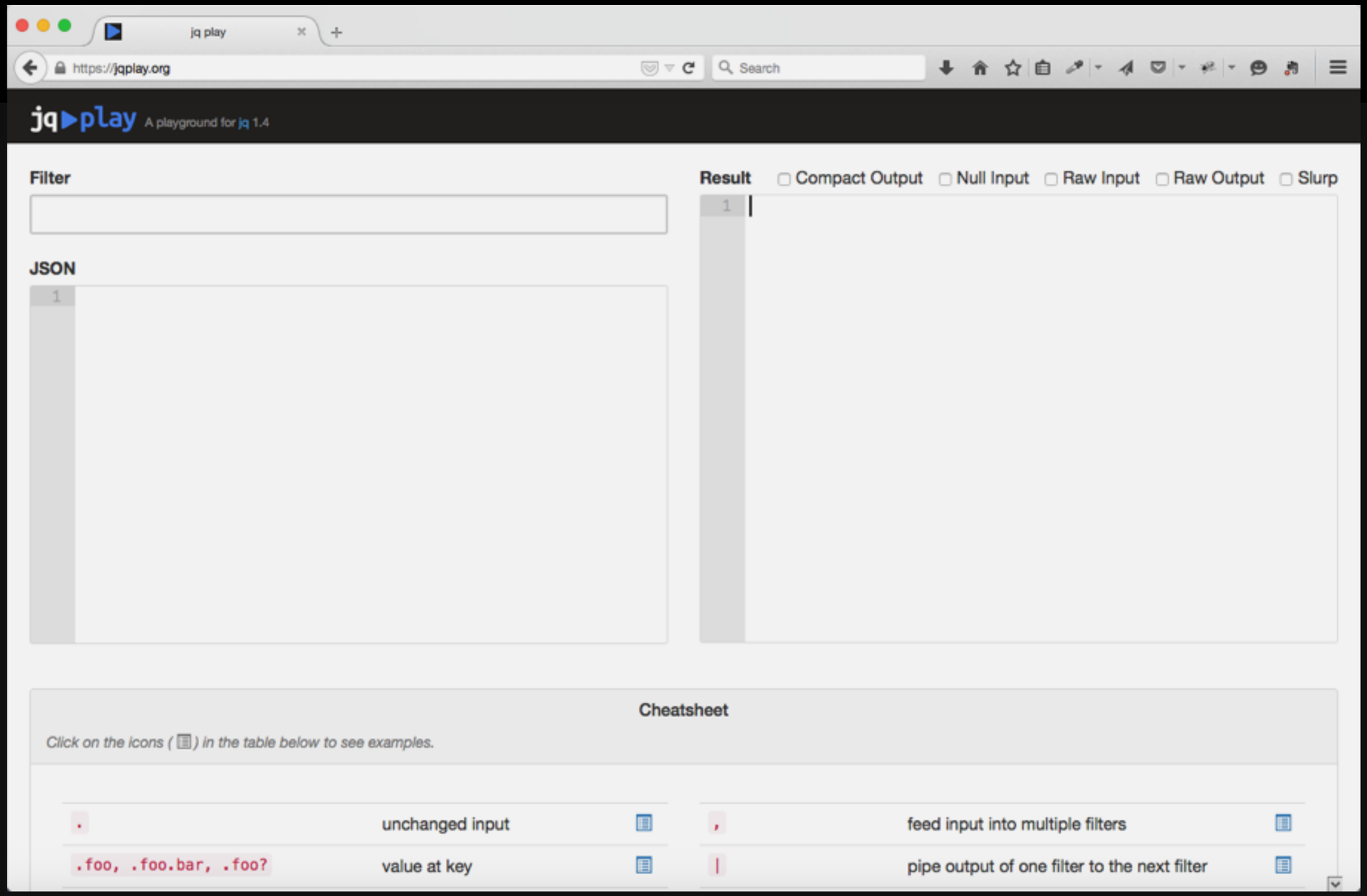
My JSON Search Choices

API	Rank
JSON Pointer	1
JSONPath	2
json:select	3
JSON Query	4

jq


jq
stedolan.github.io/jq/
Search
jq Tutorial Download Manual Issues Source Try online! NEWS
./jq
jq is a lightweight and flexible command-line JSON processor.
Download jq-1.4-2-g15c4a7f-dirty
Try online!
jq is like sed for JSON data – you can use it to slice and filter and map and transform structured data with the same ease that sed, awk, grep and friends let you play with text.
jq is written in portable C, and it has zero runtime dependencies. You can download a single binary, scp it to a far away machine, and expect it to work.
jq can mangle the data format that you have into the one that you want with very little effort, and the program to do so is often shorter and simpler than you'd expect.
Go read the tutorial for more, or the manual for way more.
News
06 June 2014

jq play


jq play
https://jqplay.org
Search
jq play A playground for jq 1.4
Filter
JSON
Result
Compact Output
Null Input
Raw Input
Raw Output
Slurp
Cheatsheet
Click on the icons ( ) in the table below to see examples.
. unchanged input
.foo, .foo.bar, .foo? value at key
, feed input into multiple filters
| pipe output of one filter to the next filter

jq Examples

```
In jq-play
----------
.airports

.airports[10]

.airports[10] | { id, IATA, weather }

.airports[10:15] | .[] | { id, IATA, weather }


In curl
-------
curl 'http://localhost:5000/airports'

curl 'http://localhost:5000/airports' | jq .[10]

curl 'http://localhost:5000/airports' | jq '.[10] | { id, IATA, weather }'

curl 'http://localhost:5000/airports' | jq '.[10:15] | .[] | { id, IATA, weather }'
```

JSONiq


JSONiq - The JSON Query Lang...
www.jsoniq.org
Search
The JSONiq Language
JSONiq Book
JSONiq Extension to XQuery
Implementations
Live Sandbox
Mailing-List
{ "codename" : ["JSONiq"] }
The JSON Query Language
Hello
my name is
JSONiq
Decades of Lessons Learnt
JSONiq is a query and processing language specifically designed for the popular JSON data model. The main ideas behind JSONiq are based on lessons learnt in more than 30 years of relational query systems and more than 15 years of experience with designing and implementing query languages for semi-structured data.
Complex Processing
A JSONiq program is an expression; the result of the program is the result of the evaluation of the expression. Expressions have fundamental role in the language: every language construct is an expression, and expressions are fully composable. Project, Filter, Join, Group... Like SQL, JSONiq can do all that.
The SQL of NoSQL
JSONiq is an expressive and highly optimizable language to query and update NoSQL stores. It enables developers to leverage the same productive high-level language across a variety of NoSQL products.

JSONiq Examples

```
2  42 instance of integer  # true
3  42 instance of decimal  # true
4  42.6 instance of decimal  # true
5  42.6e10 instance of double  # true
6  "fred" instance of string  # true
7  true instance of boolean  # true
8  null instance of null  # true
9
10
11 "Douglas" || " " || "Crockford",  # Douglas Crockford
12 "Douglas" || () || "Crockford"    # DouglasCrockford
13
14 [ "question", "answer" ][][1]  #  "question"
15
16 {
17   questions: [
18     "What JSON Search tool should I use?",
19     { "faq" : "We're still figuring it out." }
20   ]
21 }.questions[][2].faq          # "We're still figuring it out."
```

jq Scorecard

Mindshare	Y
Dev Community	Y
Platforms	CLI - Linux / Mac OS X / Windows
Intuitive	Y
Standard	N

JSONiq Scorecard

Mindshare	N
Dev Community	Y
Platforms	Zorba.io, IBM Data Power
Intuitive	Y
Standard	N

Project 4 - JSON Search

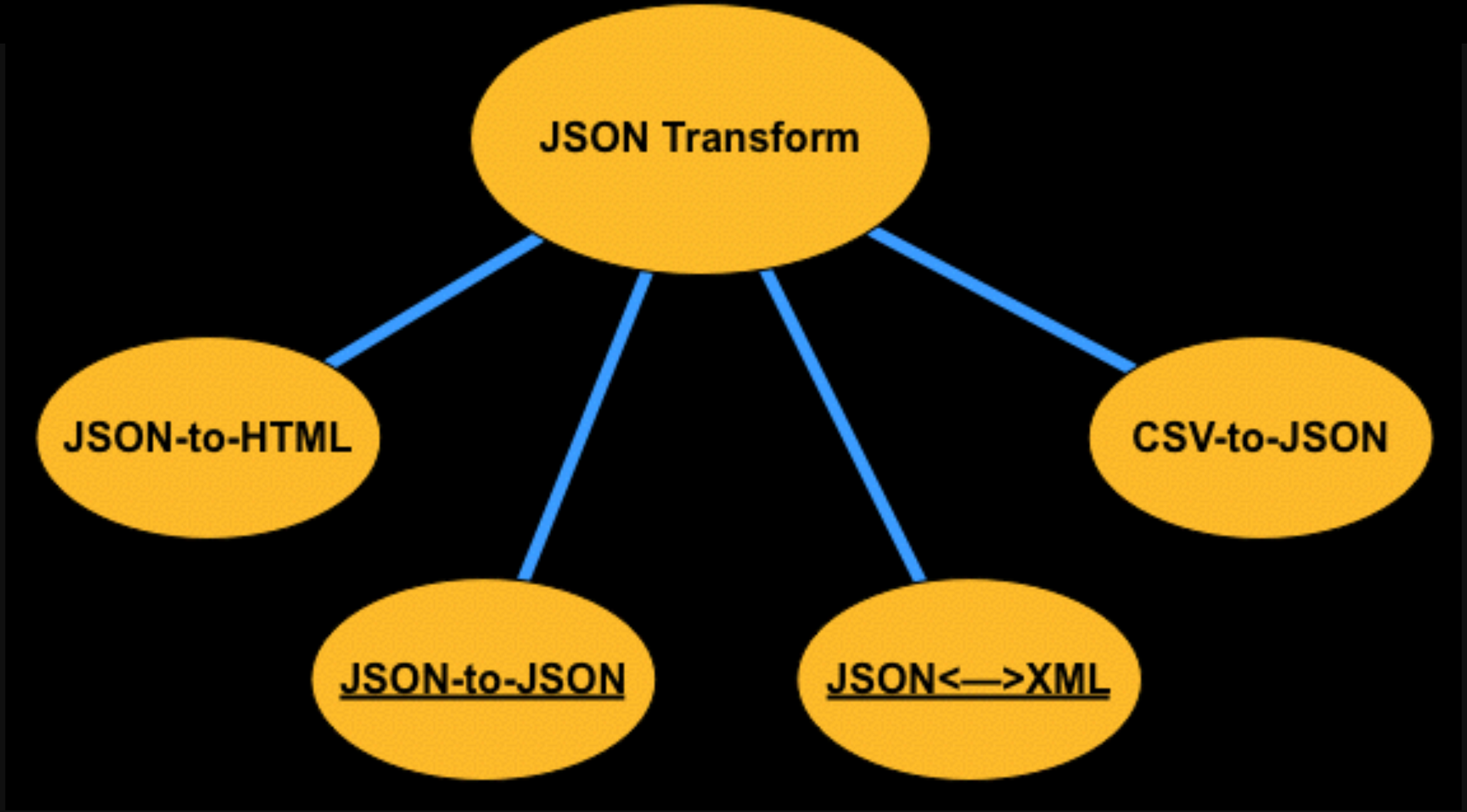
projects/README.md#project-4---json-search

projects/EspressoCON.md

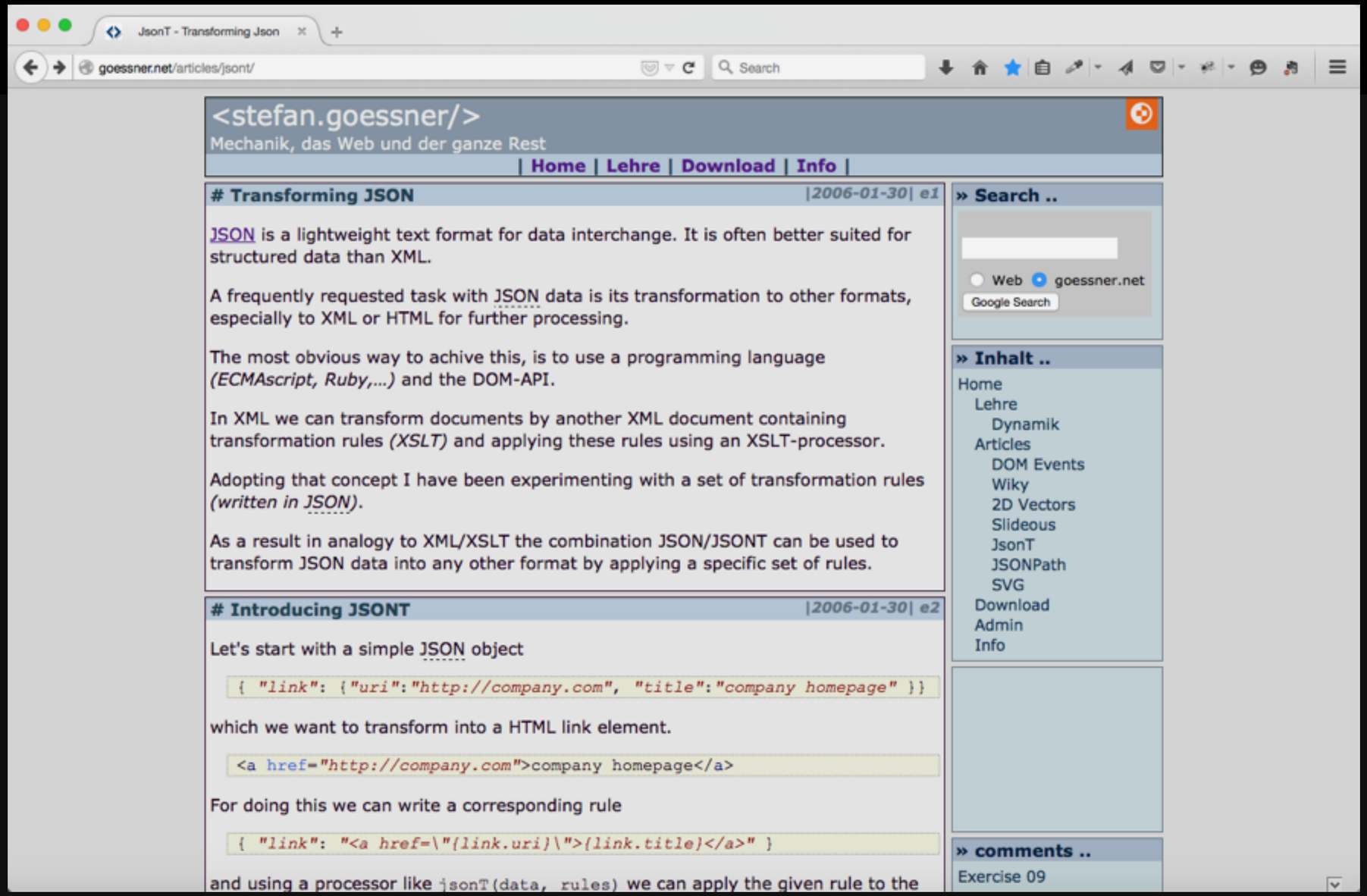
Where Are We?

JSON Schema Overview	1	JSON Search & Transform Overview	4
Core JSON Schema	2	JSON Search	5
API Design with JSON Schema	3	**JSON Transform**	**6**

JSON Transform



JSON-T


JsonT - Transforming Json
goessner.net/articles/jsont/
Search
<stefan.goessner/>
Mechanik, das Web und der ganze Rest
| Home | Lehre | Download | Info |
Transforming JSON |2006-01-30| e1
JSON is a lightweight text format for data interchange. It is often better suited for structured data than XML.
A frequently requested task with JSON data is its transformation to other formats, especially to XML or HTML for further processing.
The most obvious way to achive this, is to use a programming language (ECMAscript, Ruby,...) and the DOM-API.
In XML we can transform documents by another XML document containing transformation rules (XSLT) and applying these rules using an XSLT-processor.
Adopting that concept I have been experimenting with a set of transformation rules (written in JSON).
As a result in analogy to XML/XSLT the combination JSON/JSONT can be used to transform JSON data into any other format by applying a specific set of rules.
Introducing JSONT |2006-01-30| e2
Let's start with a simple JSON object
{ "link": {"uri":"http://company.com", "title":"company homepage" }}
which we want to transform into a HTML link element.
company homepage
For doing this we can write a corresponding rule
{ "link": "{link.title}" }
and using a processor like jsonT(data, rules) we can apply the given rule to the
» Search ..
Web goessner.net
Google Search
» Inhalt ..
Home
Lehre
Dynamik
Articles
DOM Events
Wiky
2D Vectors
Slideous
JsonT
JSONPath
SVG
Download
Admin
Info
» comments ..
Exercise 09

JSON-T Syntax

simple array

```
["red", "green", "blue"]
```

+

```
["self": "<ul>\n{$}</ul>",
 "self[*]": "  <li>{$}</li>\n"]
```

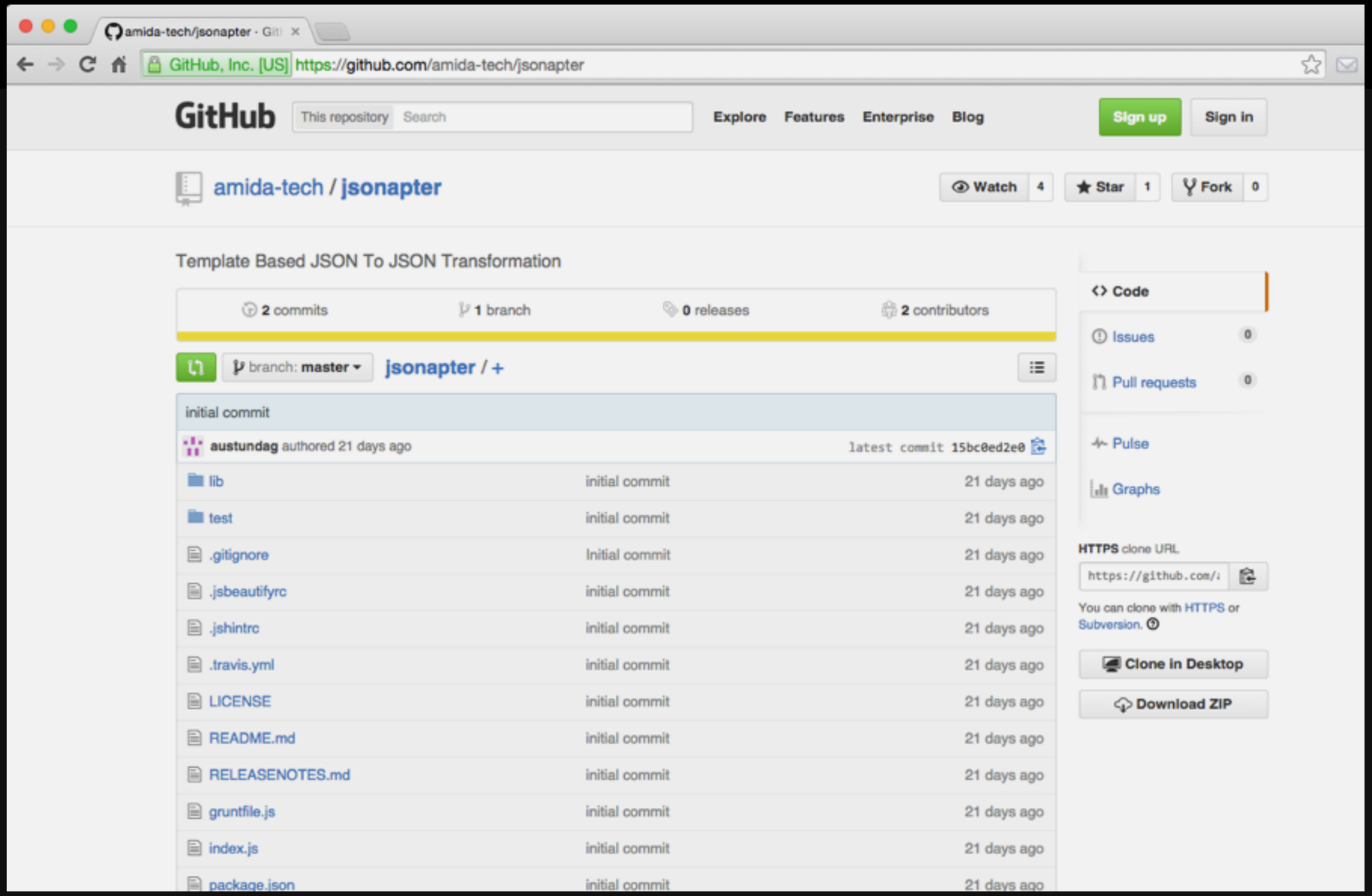
=

```
<ul>
  <li>red</li>
  <li>green</li>
  <li>blue</li>
</ul>
```

JSON-T Scorecard

Mindshare	N
Dev Community	N
Platforms	JS
Intuitive	Y
Standard	N

jsonapter


amida-tech/jsonapter · Gith
GitHub, Inc. [US] https://github.com/amida-tech/jsonapter
GitHub
This repository Search
Explore Features Enterprise Blog
Sign up Sign in
amida-tech / jsonapter
Watch 4 Star 1 Fork 0
Template Based JSON To JSON Transformation
2 commits 1 branch 0 releases 2 contributors
branch: master jsonapter / +
initial commit
austundag authored 21 days ago latest commit 15bc0ed2e0
lib initial commit 21 days ago
test initial commit 21 days ago
.gitignore Initial commit 21 days ago
.jsbeautifyrc initial commit 21 days ago
.jshintrc initial commit 21 days ago
.travis.yml initial commit 21 days ago
LICENSE initial commit 21 days ago
README.md initial commit 21 days ago
RELEASENOTES.md initial commit 21 days ago
gruntfile.js initial commit 21 days ago
index.js initial commit 21 days ago
package.json initial commit 21 days ago
Code
Issues 0
Pull requests 0
Pulse
Graphs
HTTPS clone URL
https://github.com/a
You can clone with HTTPS or Subversion.
Clone in Desktop
Download ZIP

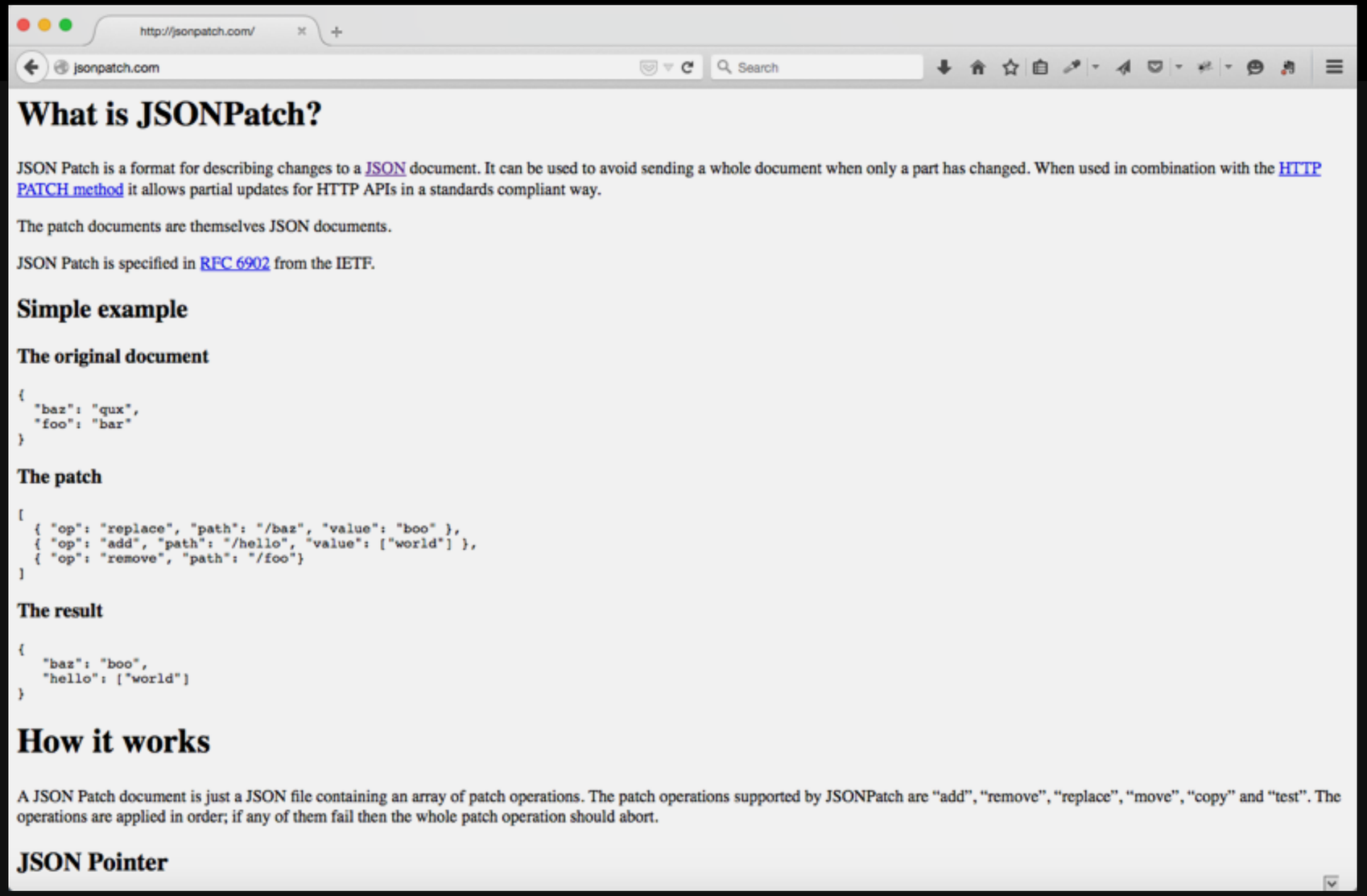
jsonapter Test

```
var expect = require('chai').expect;
var jsonfile = require('jsonfile');
var j2j = require('jsonapter').instance();

var template = {
  content: {
    name: {
      dataTransform: function(input) {
        return input.firstName + ' ' + input.lastName;
      }
    },
    email: { dataKey: 'email' },
    about: { dataKey: 'about' },
  }
};

describe('jsonapter', function() {
  describe('run', function() {
    it('should transform JSON', function(done) {
      var jsonFileName = './data/speaker.json';

      jsonfile.readFile(jsonFileName, function(err, jsonObj) {
        if (!err) {
          console.log(jsonObj);
          console.log('\n\n\n\jsonapter Test');
          var output = j2j.run(template, jsonObj);
          console.log('\n\n\n\Transformed JSON');
          console.log(JSON.stringify(output));
        }
        done();
```

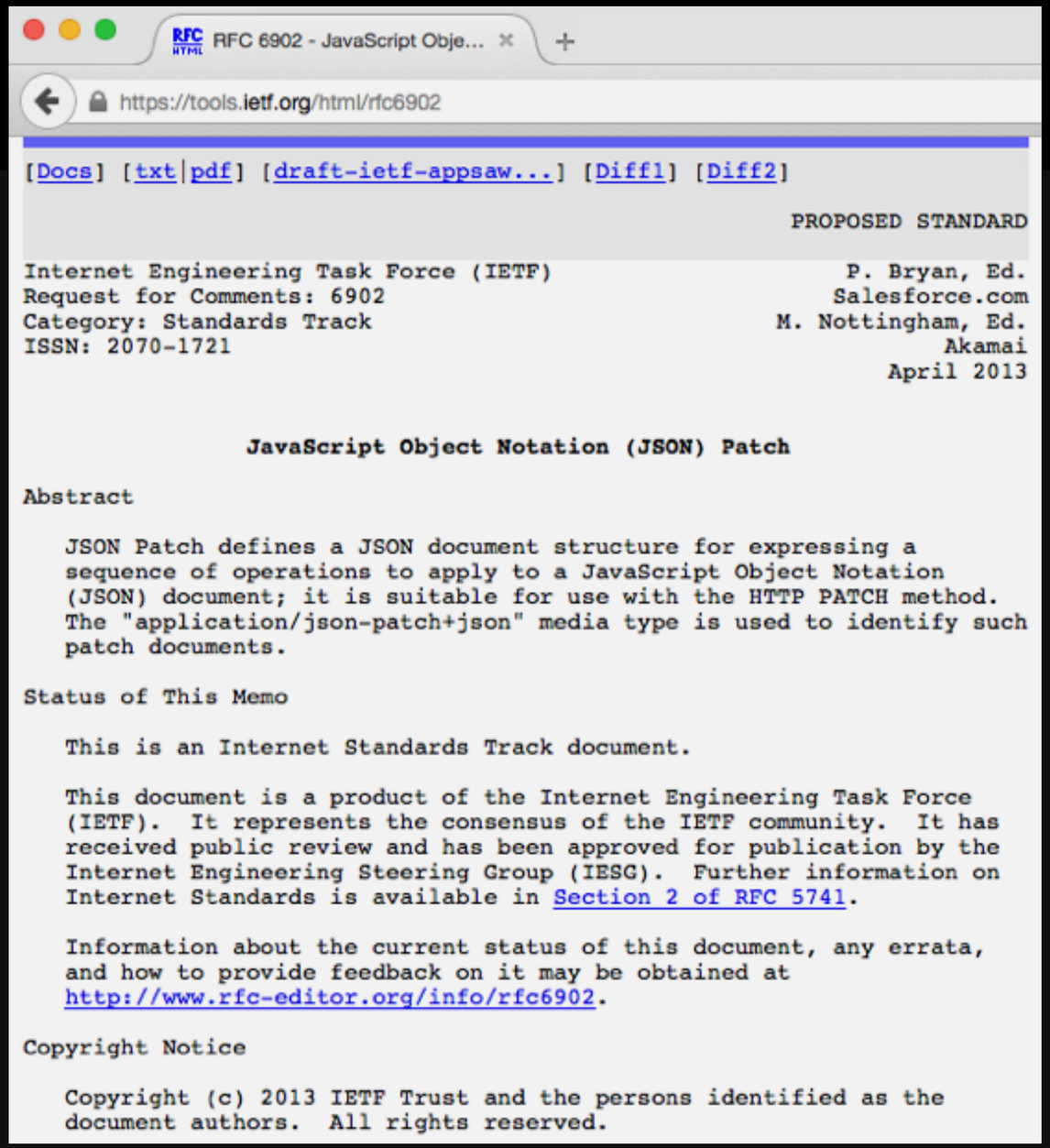
jsonapter Scorecard

Mindshare	N
Dev Community	Y
Platforms	JS, Node.js
Intuitive	Y
Standard	N

JSON Patch


http://jsonpatch.com/
jsonpatch.com
Search
What is JSONPatch?
JSON Patch is a format for describing changes to a JSON document. It can be used to avoid sending a whole document when only a part has changed. When used in combination with the HTTP PATCH method it allows partial updates for HTTP APIs in a standards compliant way.
The patch documents are themselves JSON documents.
JSON Patch is specified in RFC 6902 from the IETF.
Simple example
The original document
{
"baz": "qux",
"foo": "bar"
}
The patch
[
{ "op": "replace", "path": "/baz", "value": "boo" },
{ "op": "add", "path": "/hello", "value": ["world"] },
{ "op": "remove", "path": "/foo"}
]
The result
{
"baz": "boo",
"hello": ["world"]
}
How it works
A JSON Patch document is just a JSON file containing an array of patch operations. The patch operations supported by JSONPatch are “add”, “remove”, “replace”, “move”, “copy” and “test”. The operations are applied in order; if any of them fail then the whole patch operation should abort.
JSON Pointer

JSON Patch Standard



[Docs] [txt|pdf] [draft-ietf-appsaw...] [Diff1] [Diff2]

PROPOSED STANDARD

Internet Engineering Task Force (IETF) — P. Bryan, Ed., Salesforce.com
Request for Comments: 6902 — M. Nottingham, Ed., Akamai
Category: Standards Track — April 2013
ISSN: 2070-1721

JavaScript Object Notation (JSON) Patch

Abstract

JSON Patch defines a JSON document structure for expressing a sequence of operations to apply to a JavaScript Object Notation (JSON) document; it is suitable for use with the HTTP PATCH method. The "application/json-patch+json" media type is used to identify such patch documents.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc6902.

JSON Patch Meaning

HTTP PATCH - Partial Updates

Content-Type: application/json-patch+json

JSON Patch Syntax

Simple example

The original document

```
{
  "baz": "qux",
  "foo": "bar"
}
```

The patch

```
[
  { "op": "replace", "path": "/baz", "value": "boo" },
  { "op": "add", "path": "/hello", "value": ["world"] },
  { "op": "remove", "path": "/foo"}
]
```

The result

```
{
   "baz": "boo",
   "hello": ["world"]
}
```

JSON Patch Test

```
var expect = require('chai').expect;
var jsonfile = require('jsonfile');
var jsonpatch = require('json-patch');

var template = [
  { op: 'add', path: '/submittedSlides', value: true },
  { op: 'remove', path: '/tags' },
  { op: 'remove', path: '/company' }
];

describe('json-patch', function() {
  describe('apply', function() {
    it('should patch JSON', function(done) {
      var jsonFileName = './data/speaker.json';

      jsonfile.readFile(jsonFileName, function(err, jsonObj) {
        if (!err) {
          console.log(jsonObj);
          console.log('\n\n\n\JSONPatch Test');
          var output = jsonpatch.apply(jsonObj, template);
          console.log('\n\n\n\Patch JSON');
          console.log(JSON.stringify(output));
        }
        done();
      });
    });
  });
});
```

JSON Patch Scorecard

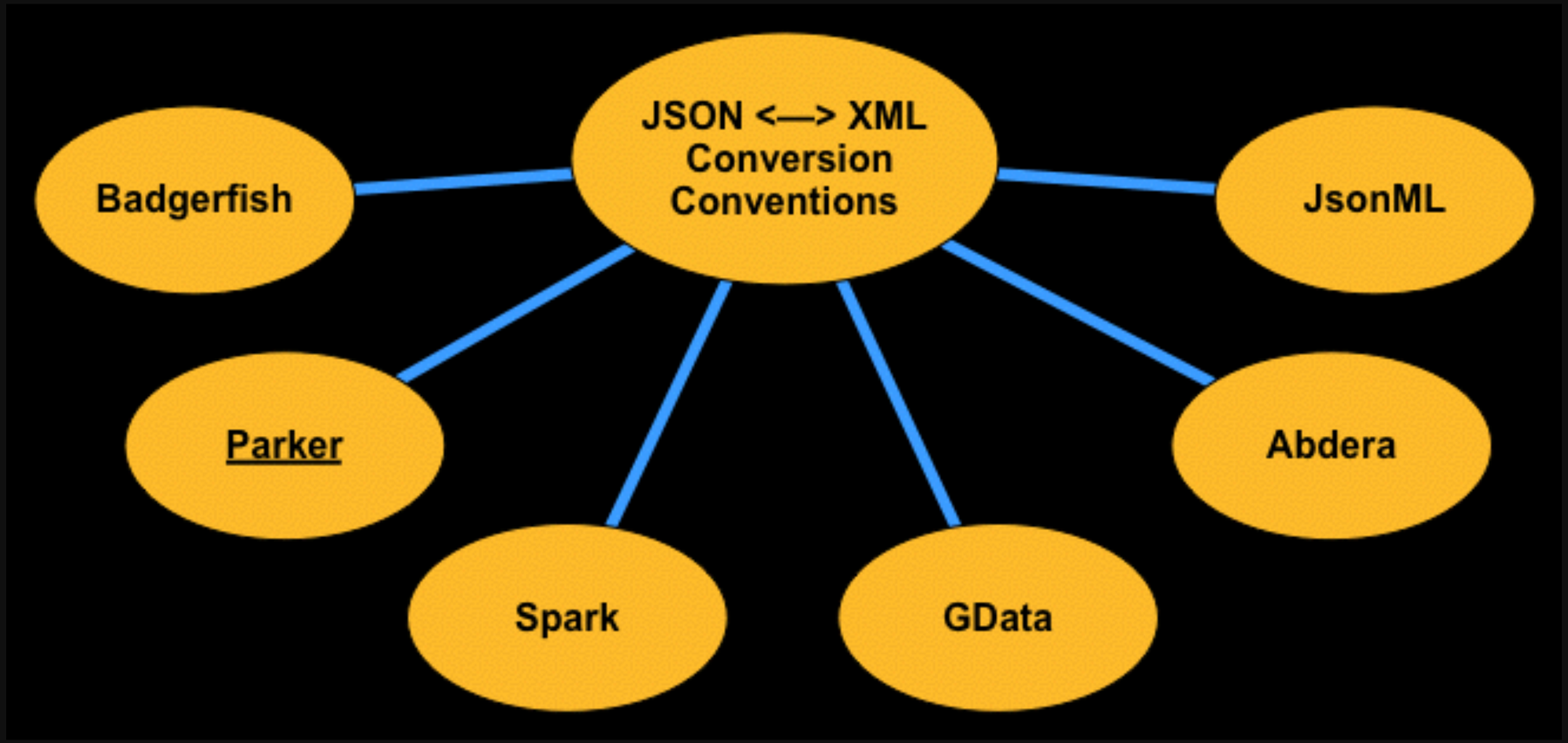
Mindshare	Y
Dev Community	Y
Platforms	JS, Node.js, Java, RoR, etc.
Intuitive	Y
Standard	RFC 6902 - Yes!

Project 5 - JSON Transform

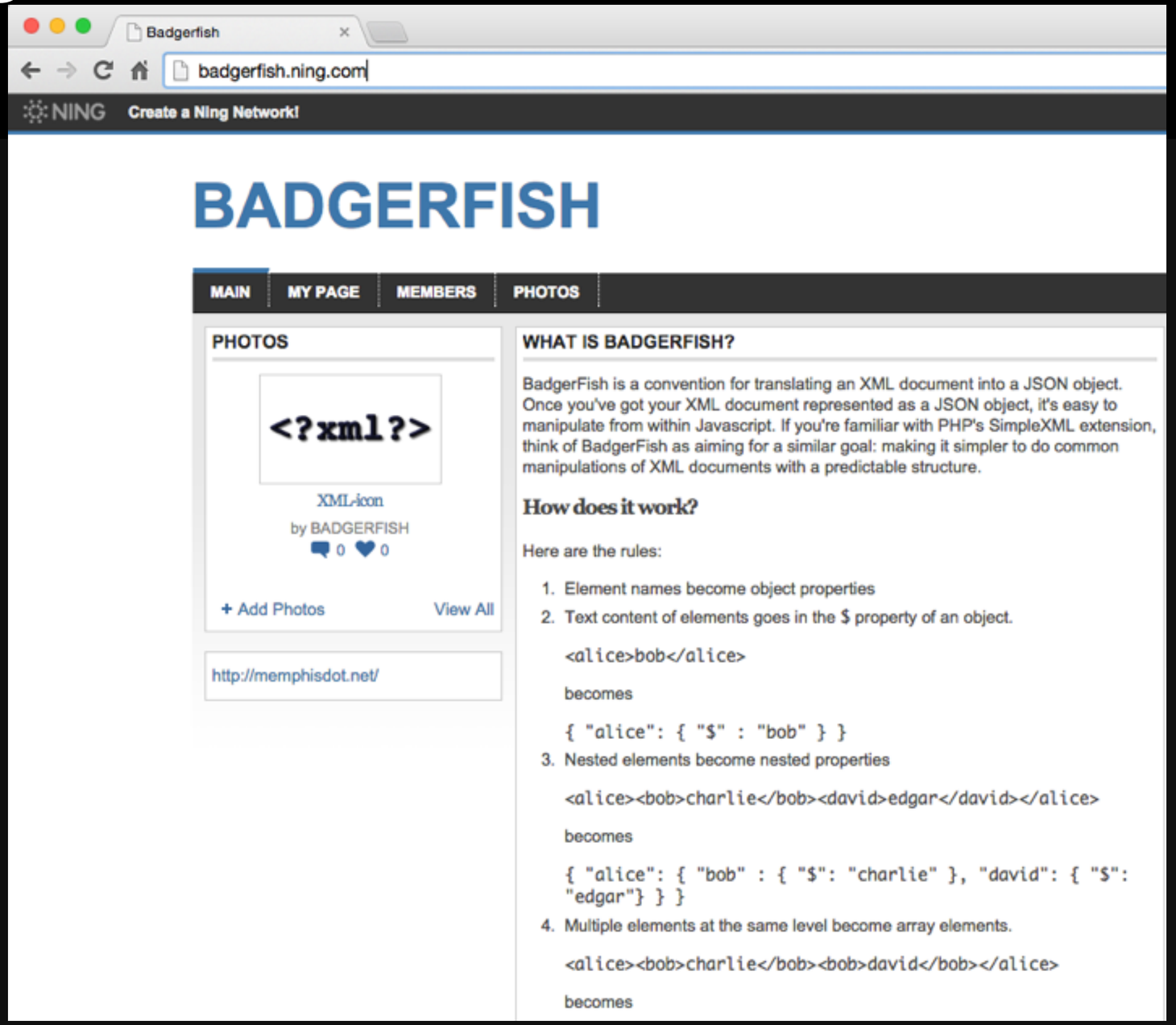
projects/README.md#project-5---json-transform

projects/EspressoCON.md

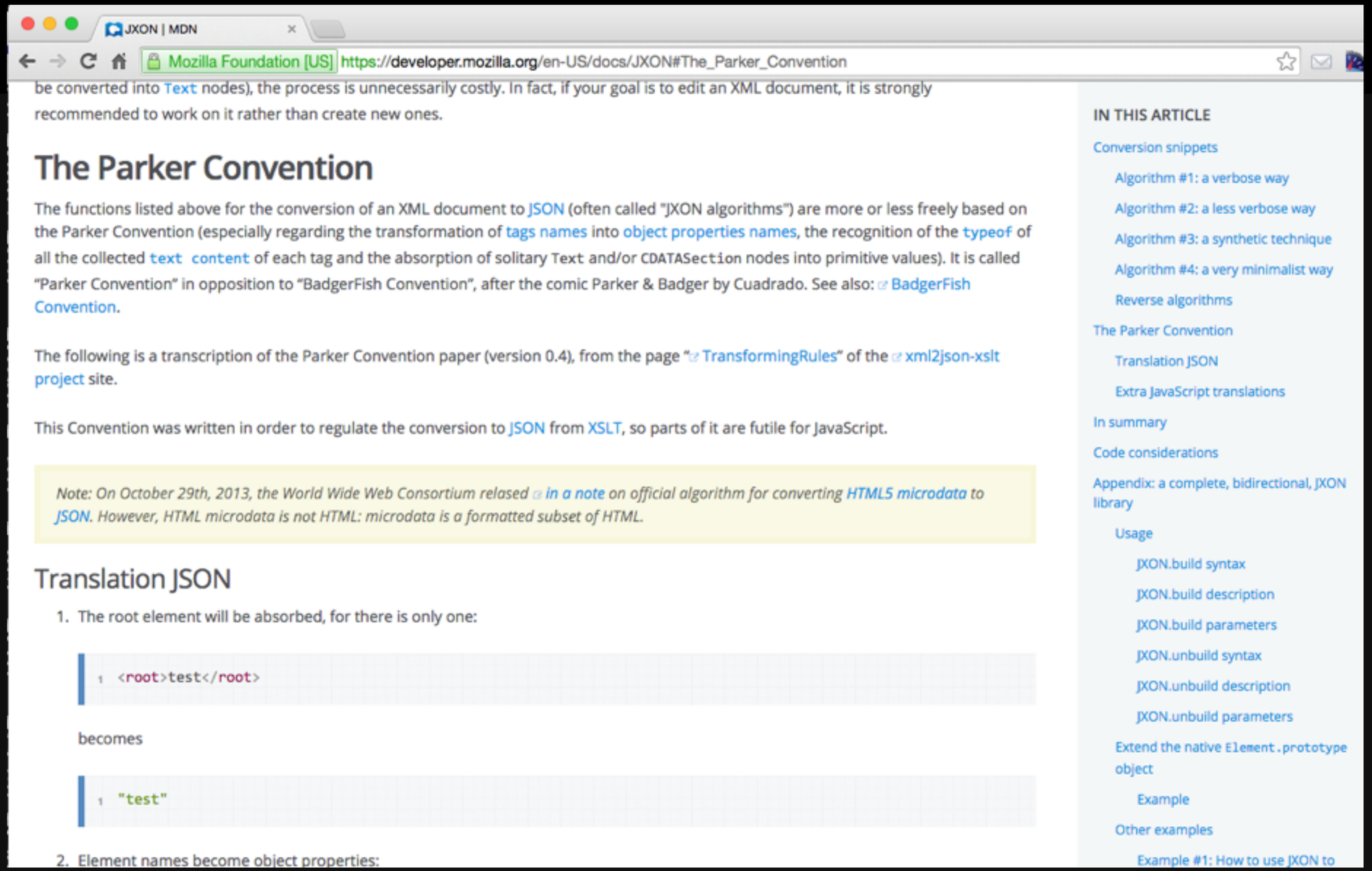
JSON - XML Conversion Conventions



Badgerfish Convention



Parker Convention



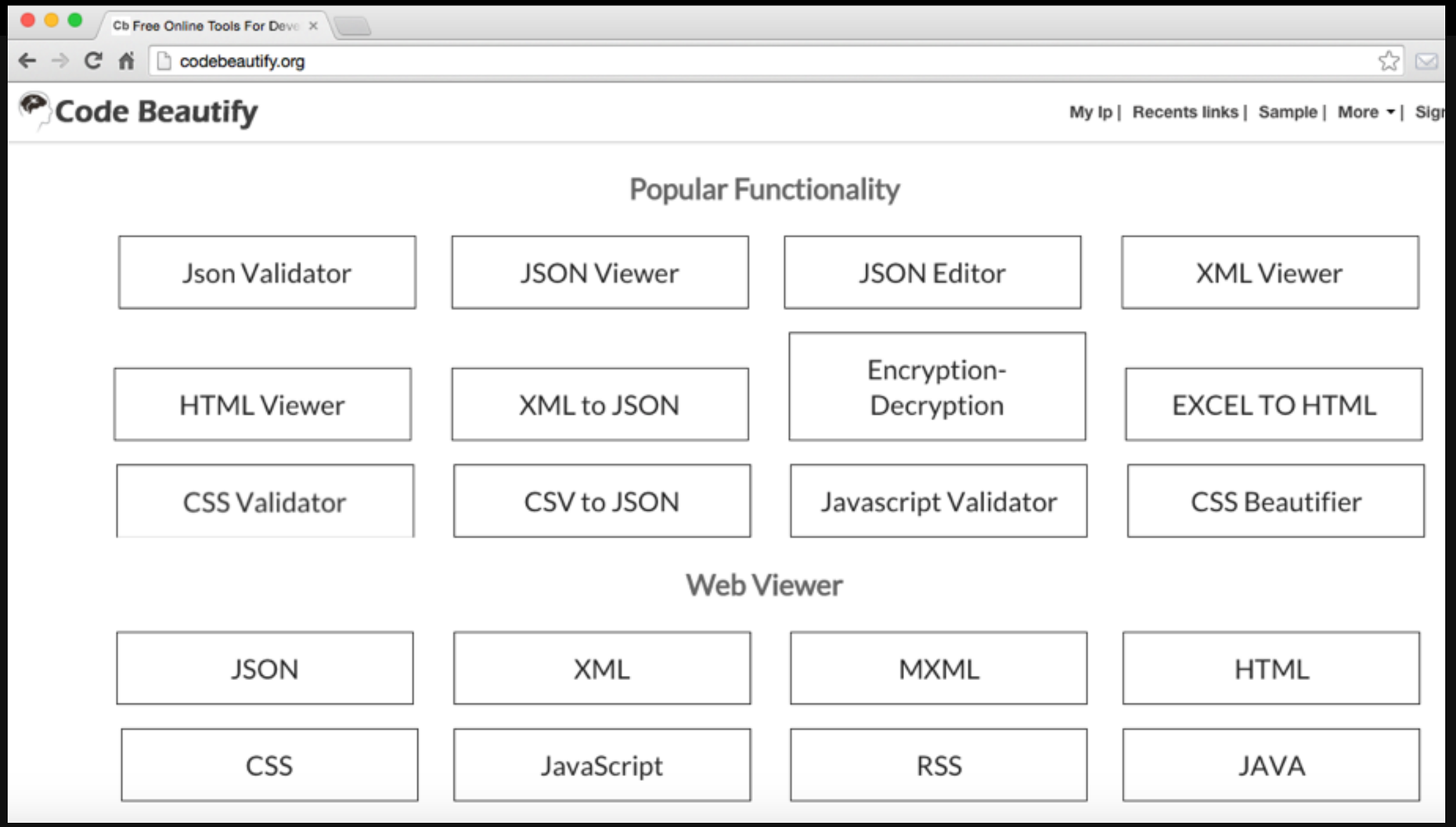
JXON Test - XML -> JSON

```
2  var expect = require('chai').expect;
3  var fs = require('fs');
4  var jxon = require('jxon');
5  var jsonfile = require('jsonfile');
6
7  describe('jxon', function() {
8    describe('stringToJs', function() {
9      it('should transform XML to JSON', function(done) {
10       var xmlFileName = './data/speaker.xml';
11
12       fs.readFile(xmlFileName, 'utf8', function (err, xmlData) {
13         if (!err) {
14           console.log('\n\n\n\jxon Test - XML ==> JSON');
15           console.log('\n\n\n\XML');
16           console.log(xmlData);
17           var output = jxon.stringToJs(xmlData);
18           console.log('\n\n\n\Transformed JSON');
19           console.log(JSON.stringify(output));
20         }
21         done();
22       });
23     });
24   });
```

JXON Test - JSON -> XML

```
26    describe('jsToString', function() {
27      it('should transform JSON to XML', function(done) {
28        var jsonFileName = './data/speaker.json';
29
30        jsonfile.readFile(jsonFileName, function(err, jsonObj) {
31          if (!err) {
32            console.log('\n\n\n\jxon Test - JSON ==> XML');
33            console.log('\n\n\n\JSON');
34            console.log(jsonObj);
35            var xml = jxon.jsToString(jsonObj);
36            console.log('\n\n\n\Transformed XML');
37            console.log(xml);
38          }
39          done();
40        });
41      });
42    });
43  });
```

If All Else Fails … codebeautify.org



Agenda

What's The Point?

Drive API Design with JSON Schema

What's The Point? #2

JSON Search and Transform

Simplify interaction with RESTful APIs

Questions?

Tom Marrs

@TomMarrs

thomasamarrs@comcast.net

JSON Resources

JSON Spec - http://tools.ietf.org/html/rfc7159

ECMA 404 - http://www.ecma-international.org/publications/standards/Ecma-404.htm

JSON.org - http://www.json.org

JSONLint - http://www.jsonlint.com

JSON Resources

JSON Generator - http://www.json-generator.com/

JSONPad - https://code.google.com/p/json-pad/

JSON Editor Online - http://jsoneditoronline.org/

JSON Schema Resources

json-schema.org - http://json-schema.org/

JSON Schema Spec - http://json-schema.org/latest/json-schema-core.html

JSON Schema Validation Spec - http://json-schema.org/latest/json-schema-validation.html

JSON Hyper-Schema Spec - http://json-schema.org/latest/json-schema-hypermedia.html

JSON Schema Resources

Using JSON Schema - http://usingjsonschema.com/

JSON Validate - http://jsonvalidate.com/

Understanding JSON Schema - http://spacetelescope.github.io/understanding-json-schema/

jsonschema.net - http://jsonschema.net/

IETF - https://tools.ietf.org/html/draft-zyp-json-schema-04

JSON Schema Resources

JSON Schema GitHub Repo - https://github.com/kriszyp/json-schema

JSON Schema Google Group - https://groups.google.com/forum/#!forum/json-schema

Put Some JSON Schema in your life - https://kreuzwerker.de/en/blog/posts/put-some-json-schema-in-your-life

JSON Schema Resources

Docson GitHub Repo - https://github.com/lbovet/docson

Swagger Petstore - http://petstore.swagger.io/#!/pet/getPetById

Docson/Typson Swagger Petstore - http://lbovet.github.io/swagger-ui/dist/index.html#!/pet/getPetById_get_0

JSONPath Resources

http://goessner.net/articles/JsonPath/

https://github.com/jayway/JsonPath

https://rubygems.org/gems/jsonpath/versions/0.5.6

https://www.npmjs.com/package/json-path

https://www.npmjs.com/package/jsonpath

JSON Pointer Resources

https://tools.ietf.org/html/rfc6901

https://www.npmjs.com/package/json-pointer

https://rubygems.org/gems/json-pointer

https://github.com/fge/jackson-coreutils

http://susanpotter.net/blogs/software/2011/07/why-json-pointer-falls-short/

https://zato.io/blog/posts/json-pointer-rfc-6901.html

JSON Query Resources

https://github.com/jcrosby/jsonquery

https://github.com/mmckegg/json-query

https://www.npmjs.com/package/json-query

json:select Resources

http://jsonselect.org/#overview

https://github.com/lloyd/JSONSelect

https://github.com/lloyd/JSONSelect/blob/master/JSONSelect.md

https://www.npmjs.com/package/JSONSelect

https://github.com/fd/json_select

JPath Resources

http://bluelinecity.com/software/jpath/

https://www.npmjs.com/package/jpath

https://www.npmjs.com/package/node-jpath

https://github.com/merimond/jpath

jq Resources

http://stedolan.github.io/jq/

http://stedolan.github.io/jq/tutorial/

https://github.com/stedolan/jq

https://robots.thoughtbot.com/jq-is-sed-for-json

https://zerokspot.com/weblog/2013/07/18/processing-json-with-jq/

https://jqplay.org/

JSON Transform Resources

http://goessner.net/articles/jsont/

https://github.com/amida-tech/jsonapter

http://codebeautify.org/

JSON Patch Resources

http://jsonpatch.com/

https://tools.ietf.org/html/rfc6902

http://jsonpatchjs.com/

https://rubygems.org/gems/json_patch

https://github.com/flipkart-incubator/zjsonpatch

https://www.npmjs.com/package/json-patch

https://www.npmjs.com/package/jsonpatch

JSON - XML Conversion Resources

http://wiki.open311.org/JSON_and_XML_Conversion/

http://badgerfish.ning.com/

https://developer.mozilla.org/en-US/docs/JXON#The_Parker_Convention

http://www.thomasfrank.se/xml_to_json.html

http://www.utilities-online.info/xmltojson/

JSON Groups

Google - http://groups.google.com/group/json-schema

Yahoo! - http://tech.groups.yahoo.com/group/json/