

JSON at Work: Schema, Search, and Transform

Tom Marrs

@TomMarrs



About Me ...



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

JSON Schema Overview 1

Core JSON Schema 2

API Design with JSON Schema 3

JSON Search & Transform Overview 4

JSON Search 5

JSON Transform 6

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

Where Are We?

JSON Schema Overview 1

Core JSON Schema 2

API Design with JSON Schema 3

JSON Search & Transform Overview 4

JSON Search 5

JSON Transform 6

What is JSON Schema?

Validate Structure + Format

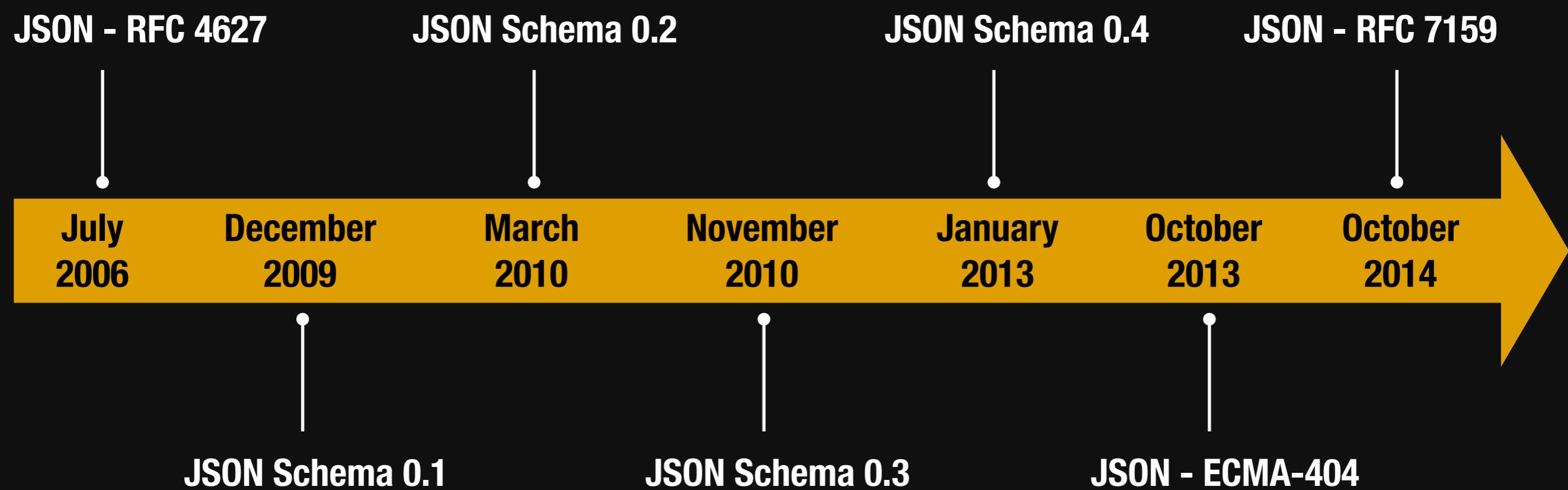
Basic JSON Schema

```
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 ...



The screenshot shows the homepage of json-schema.org as it would appear in a desktop web browser. The page has a dark green header bar with the site's name and a sub-header 'The home of JSON Schema'. Below the header is a navigation menu with four items: 'about' (underlined), 'docs', 'examples', and 'software'. The main content area is divided into several horizontal sections by thin black lines. The first section, 'What does it do?', contains two bullet points: 'JSON Schema describes your JSON data format' and 'JSON Hyper-Schema turns your JSON data into hyper-text'. The second section, 'Advantages', is currently empty. The third section, 'JSON Schema', lists five bullet points detailing its benefits. The fourth section, 'JSON Hyper-Schema', lists three bullet points detailing its benefits. The fifth section, 'More', contains a heading 'Interested? Check out:' followed by two bullet points linking to the 'specification' and 'examples'.

JSON Schema and Hyper-Schema × +

json-schema.org

Search

about docs examples software

What does it do?

JSON Schema describes your JSON data format
JSON Hyper-Schema turns your JSON data into hyper-text

Advantages

JSON Schema

- describes your existing data format
- clear, human- and machine-readable documentation
- complete structural validation, useful for
 - automated testing
 - validating client-submitted data

JSON Hyper-Schema

- describes your existing API - no new structures required
- links (including [URI Templates](#) for target URLs)
- forms - specify a JSON Schema for the desired data

More

Interested? Check out:

- the [specification](#)
- some [examples](#)

JSON Schema on GitHub

JSON Schema specifications, reference schemas, and a CommonJS implementation <http://json-schema.org/>

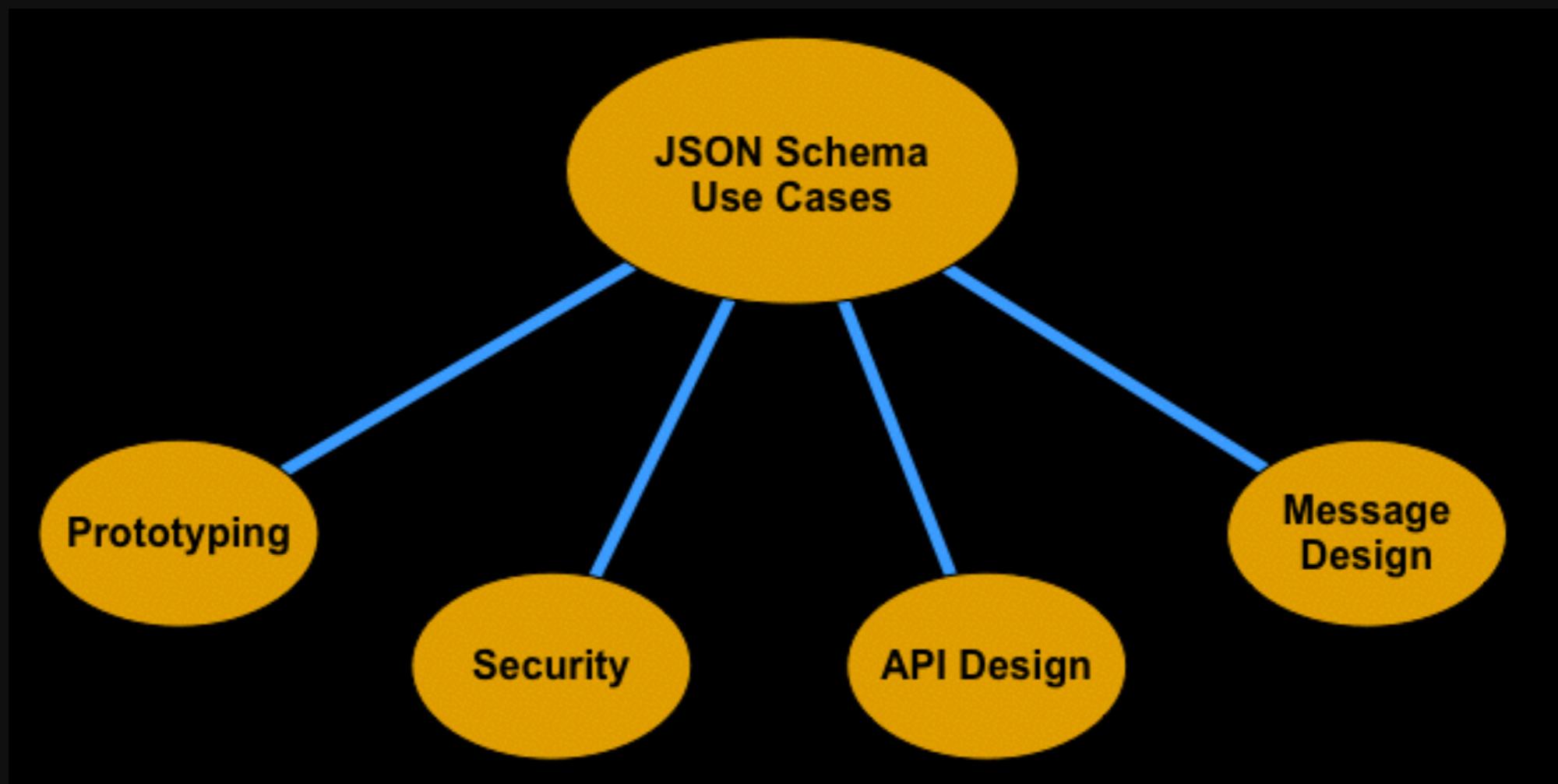
67 commits 2 branches 2 releases 7 contributors

branch: master json-schema / +

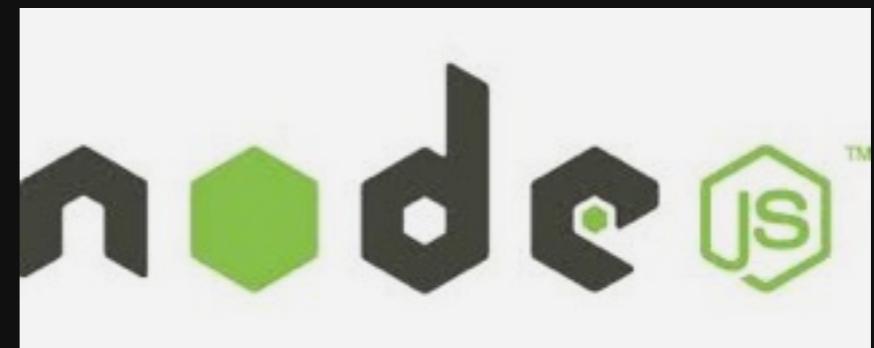
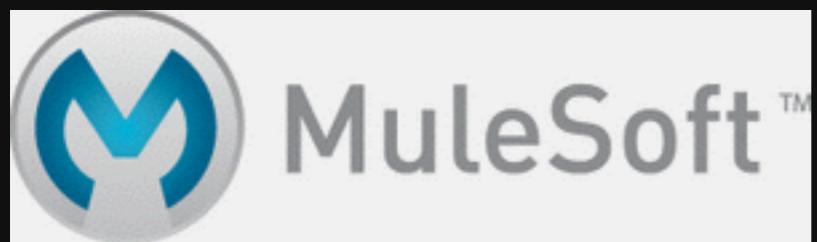
Add CNAME, I think this is needed to have json-schema.org point to it.

Author	Date	Commit Message	Time Ago
kriszyp	authored on Aug 30, 2012	latest commit f56cecb21b	5 years ago
draft-00	Schema URIs are now namespace versioned.	5 years ago	
draft-01	Schema URIs are now namespace versioned.	5 years ago	
draft-02	Schema URIs are now namespace versioned.	5 years ago	
draft-03	Fix hyper-schema syntax error.	4 years ago	
draft-04	Core changes:	4 years ago	
lib	Clean out modifications to primitives	4 years ago	
test	Add vows-based unit tests.	4 years ago	
CNAME	Add CNAME, I think this is needed to have json-schema.org point to it.	3 years ago	
README.md	Updated docs	5 years ago	
draft-zyp-json-schema-03.xml	Merge http://github.com/garycourt/json-schema	4 years ago	
draft-zyp-json-schema-04.xml	Merge git://github.com/garycourt/json-schema	4 years ago	
package.json	bump version	4 years ago	

Where Does JSON Schema Fit?



Who uses JSON Schema?



Swagger



Google
Developers

Why isn't JSON Validation Enough?

Semantics

**Schema + Instance
Document**

Meaning

Ex: Person, Order

Syntax

Instance Document

Well-formed - Valid JSON

{ }

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

Core JSON Schema 2

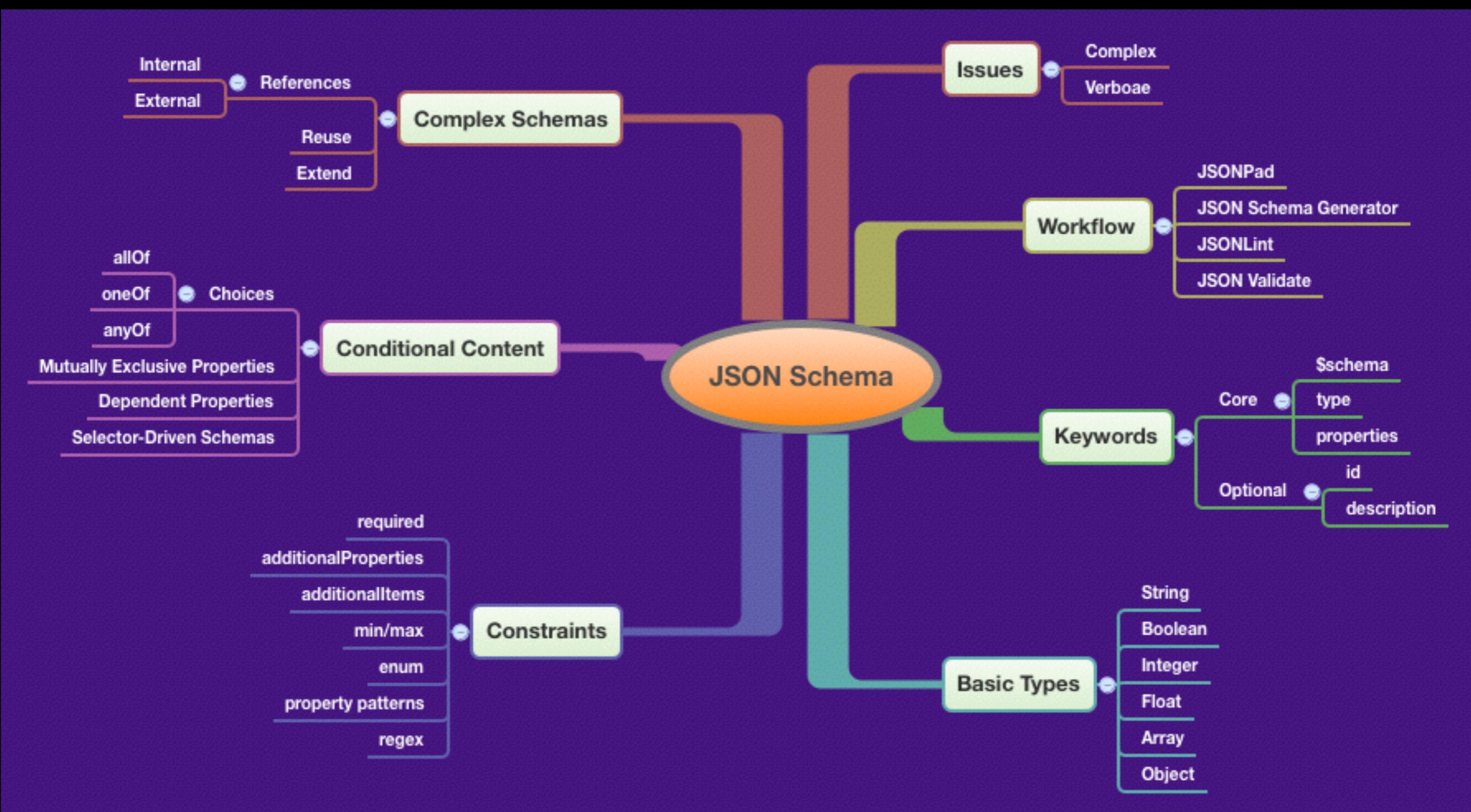
API Design with JSON Schema 3

JSON Search & Transform Overview 4

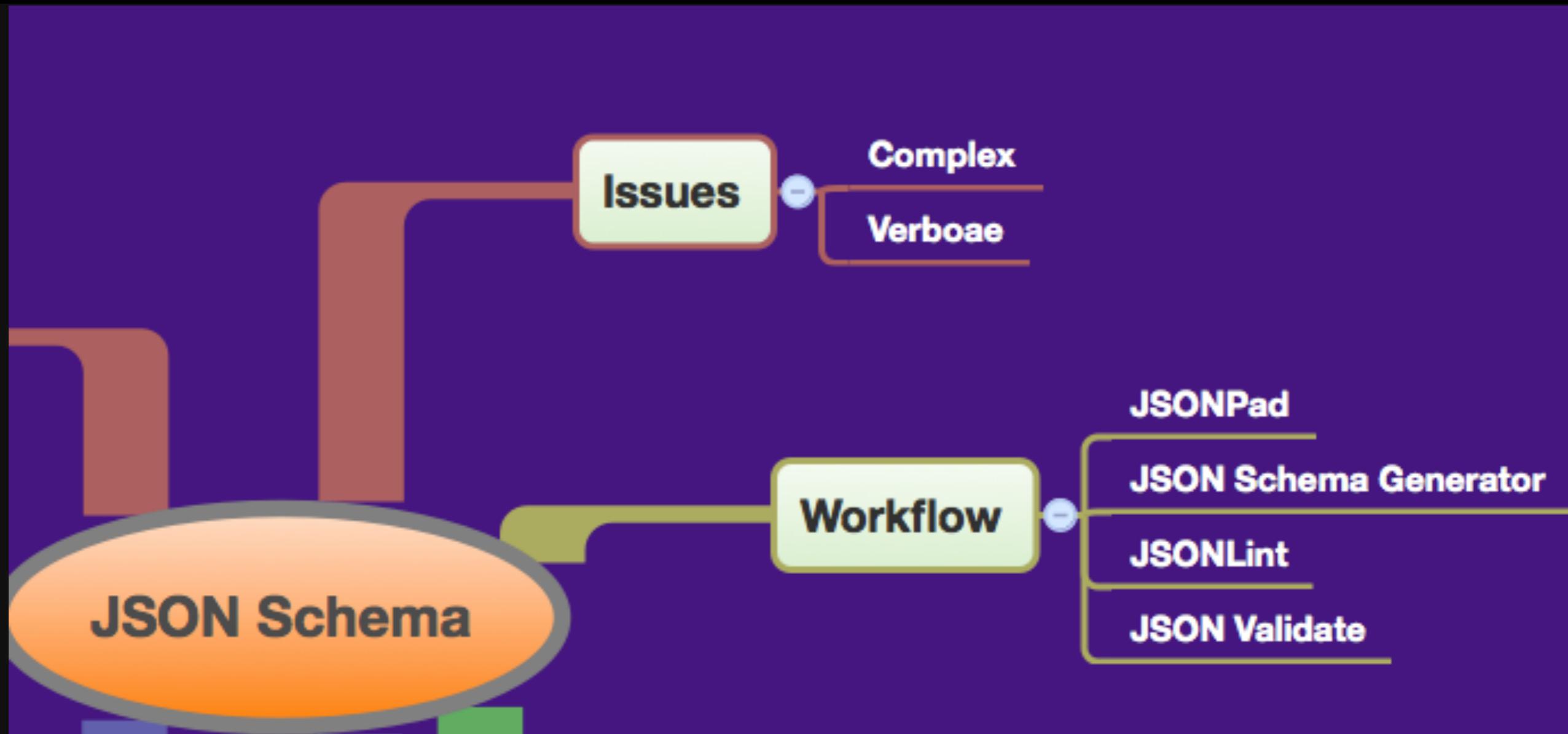
JSON Search 5

JSON Transform 6

Facets of JSON Schema



JSON Schema - Issues and Workflow



My JSON Schema Workflow

Model JSON Document

JSONPad

[https://
code.google.com/p/
json-pad/](https://code.google.com/p/json-pad/)

Generate JSON Schema

JSON Generator

<http://jsonschema.net/>

Validate JSON Document

JSON Validate

<http://jsonvalidate.com/>

JSONLint

<http://jsonlint.com>

JSONLint

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.

```
1  {
2      "email": "larsonrichard@eclaric.com",
3      "firstName": "Larson",
4      "lastName": "Richard"
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
```

Validate

JSON Lint is an idea from Arc90's Kindling

Kindling

Results

Valid JSON

JSON Validate

The screenshot shows the JSON Validate interface on jsonvalidate.com. The top navigation bar includes tabs for 'Import', 'About', and 'Help'. The main area is divided into two code editors: 'JSON Schema' on the left and 'JSON Content' on the right. The 'JSON Schema' editor contains the following JSON:

```
1 {
2   "$schema": "http://json-schema.org/draft-04/schema#",
3   "type": "object",
4   "properties": {
5     "email": {
6       "type": "string"
7     },
8     "firstName": {
9       "type": "string"
10    },
11    "lastName": {
12      "type": "string"
13    }
14  }
15
16
17
18
19
20
21
^~
```

The 'JSON Content' editor contains the following JSON, with the 'lastName' field underlined in red, indicating it is missing or invalid:

```
1 {
2   "email": "larsonrichard@ecratic.com",
3   "firstName": "Larson",
4   "lastName": "Richard"
5 }
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
^~
```

Below the code editors are sections for 'References' (with numbered buttons 1 through 8) and 'Results' (showing 'Valid'). At the bottom are buttons for 'Validate' and 'Reset all', along with a link to learn more about using JSON Schema.

JSON Schema

```
1 {
2   "$schema": "http://json-schema.org/draft-04/schema#",
3   "type": "object",
4   "properties": {
5     "email": {
6       "type": "string"
7     },
8     "firstName": {
9       "type": "string"
10    },
11    "lastName": {
12      "type": "string"
13    }
14  }
15
16
17
18
19
20
21
^~
```

JSON Content

```
1 {
2   "email": "larsonrichard@ecratic.com",
3   "firstName": "Larson",
4   "lastName": "Richard"
5 }
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
^~
```

References

1 2 3 4 5 6 7 8

Results

Valid

Validate **Reset all**

Learn more about Using JSON Schema 

Beware of Wonky WiFi - Hedge Your Bets!



PDD - Presentation-Driven Development

jsonlint

```
npm install -g jsonlint
```

```
jsonlint basic.json
```

<https://github.com/zaach/jsonlint>

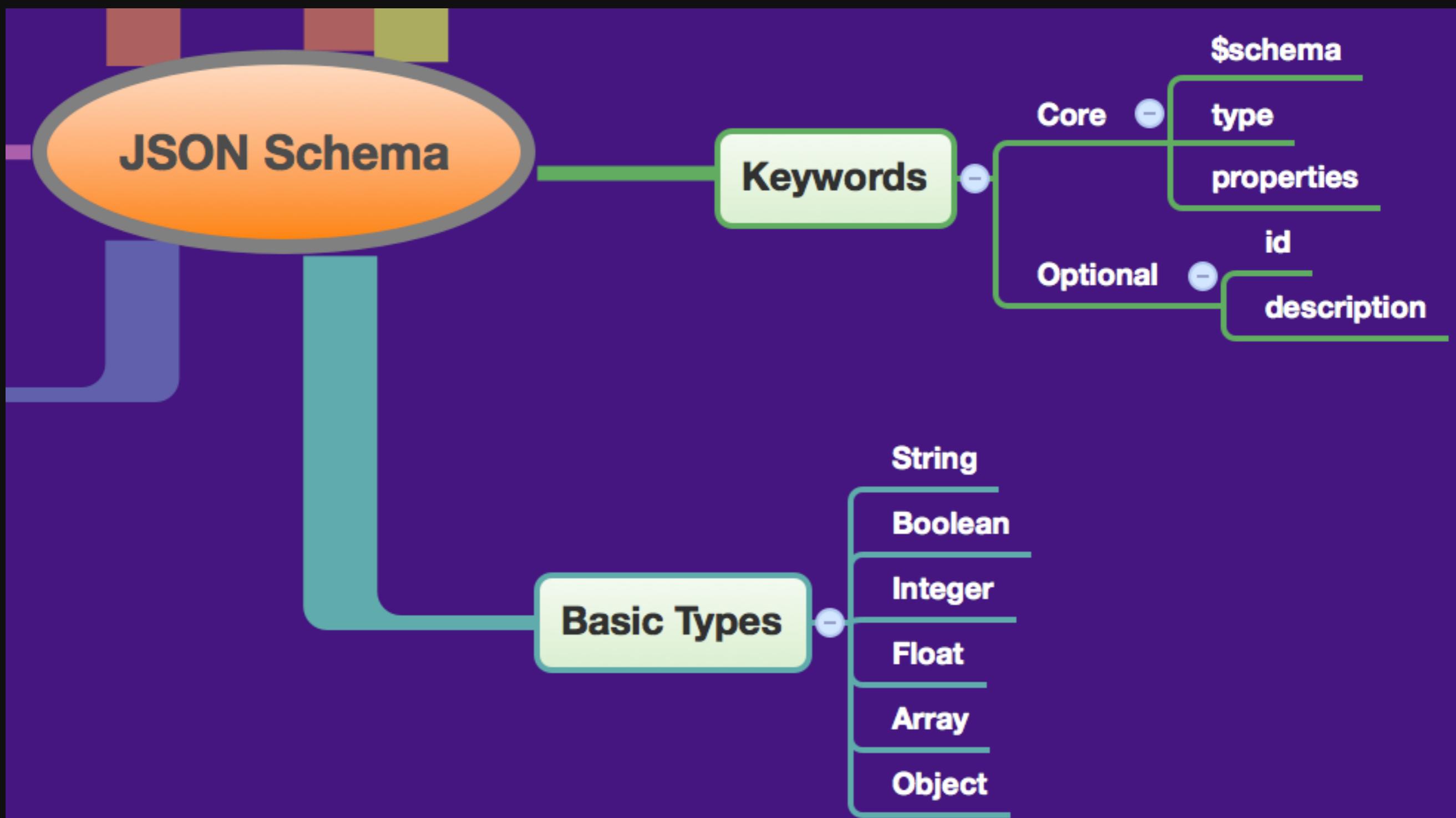
ujs-jsonvalidate

```
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	<p>Specify JSON Schema version -</p> <pre data-bbox="912 774 2535 970">"schema": "<u>http://json-schema.org/draft-04/schema#</u>"</pre>
type	<p>The data type -</p> <pre data-bbox="912 1204 1841 1298">"type": "string"</pre>
properties	The fields for an object

Optional Keywords

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

Basic Types - JSON Schema

```
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         "age": {
16             "type": "integer"
17         },
18         "postedSlides": {
19             "type": "boolean"
20         },
21         "rating": {
22             "type": "number"
23         }
24     }
25 }
```

Basic Types - JSON Document

```
1  {
2      "email": "larsonrichard@ecratic.com",
3      "firstName": "Larson",
4      "lastName": "Richard",
5      "age": 39,
6      "postedSlides": true,
7      "rating": 4.1
8  }
```

Where's the Validation?

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

Basic Types Validation - JSON Schema

```
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         "postedSlides": {
16             "type": "boolean"
17         },
18         "rating": {
19             "type": "number"
20         }
21     },
22     "additionalProperties": false
23 }
```

Validation with Required - JSON Schema

```
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         "postedSlides": {
16             "type": "boolean"
17         },
18         "rating": {
19             "type": "number"
20         }
21     },
22     "additionalProperties": false,
23     "required": ["email", "firstName", "lastName", "postedSlides", "rating"]
24 }
```

Validation with Required - JSON Doc

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

Number min/max - JSON Schema

```
2  {
3      "$schema": "http://json-schema.org/draft-04/schema#",
4      "type": "object",
5      "properties": {
6          "rating": { "type": "number", "minimum": 1.0, "maximum": 5.0 }
7      },
8      "additionalProperties": false,
9      "required": ["rating"]
10 }
```

Number min/max - JSON Doc

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

Simple Array - JSON Schema

```
1
2  {
3      "$schema": "http://json-schema.org/draft-04/schema#",
4      "type": "object",
5      "properties": {
6          "tags": {
7              "type": "array",
8              "items": {
9                  "type": "string"
10             },
11             "additionalItems": false
12         }
13     },
14     "additionalProperties": false,
15     "required": ["tags"]
16 }
```

Simple Array - JSON Doc

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

Array min/max - JSON Schema

```
2  {
3      "$schema": "http://json-schema.org/draft-04/schema#",
4      "type": "object",
5      "properties": {
6          "tags": {
7              "type": "array",
8              "minItems": 2,
9              "maxItems": 4,
10             "items": {
11                 "type": "string"
12             },
13             "additionalItems": false
14         }
15     },
16     "additionalProperties": false,
17     "required": ["tags"]
18 }
```

Array min/max - JSON Doc

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

Enum - JSON Schema

```
2  {
3      "$schema": "http://json-schema.org/draft-04/schema#",
4      "type": "object",
5      "properties": {
6          "tags": {
7              "type": "array",
8              "minItems": 2,
9              "maxItems": 4,
10             "items": {
11                 "type": "string",
12                 "enum": [
13                     "Open Source", "Java", "JavaScript", "JSON", "REST"
14                 ]
15             },
16             "additionalItems": false
17         }
18     },
19     "additionalProperties": false,
20     "required": ["tags"]
21 }
```

Enum - JSON Doc

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

Named Object - JSON Schema

```
2  {
3      "$schema": "http://json-schema.org/draft-04/schema#",
4      "type": "object",
5      "properties": {
6          "speaker": {
7              "type": "object",
8              "properties": {
9                  "firstName": { "type": "string" },
10                 "lastName": { "type": "string" },
11                 "email": { "type": "string" },
12                 "postedSlides": { "type": "boolean" },
13                 "rating": { "type": "number" },
14                 "tags": {
15                     "type": "array",
16                     "items": { "type": "string" },
17                     "additionalItems": false
18                 }
19             },
20             "additionalProperties": false,
21             "required": ["firstName", "lastName", "email",
22                         "postedSlides", "rating", "tags"
23                     ]
24     }
25 },
26     "additionalProperties": false,
27     "required": ["speaker"]
28 }
```

Named Object - JSON Doc

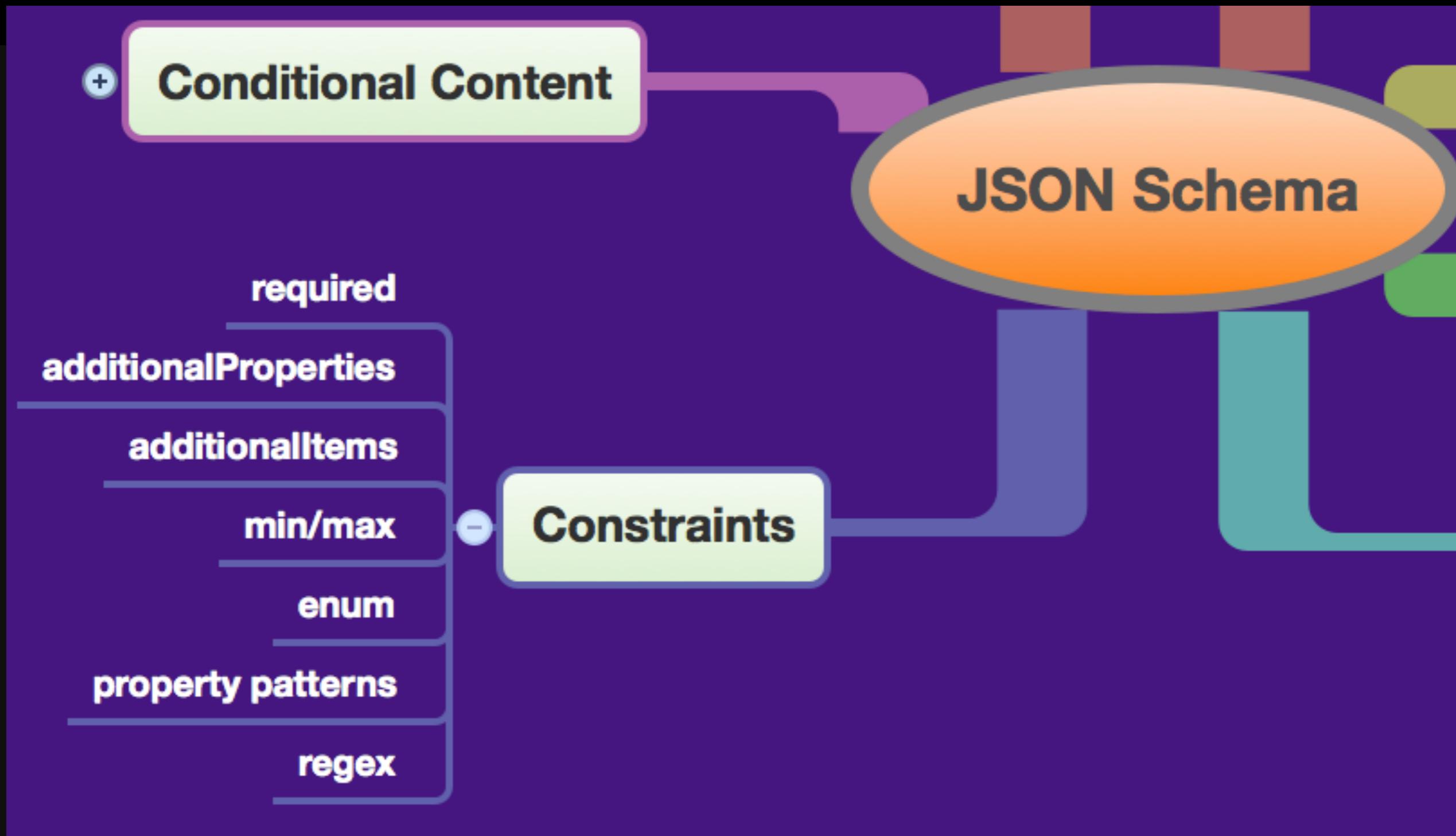
```
2  {
3      "speaker": {
4          "firstName": "Larson",
5          "lastName": "Richard",
6          "email": "larsonrichard@ecratic.com",
7          "postedSlides": true,
8          "rating": 4.1,
9          "tags": [
10              "JavaScript", "AngularJS", "Yeoman"
11          ]
12      }
13  }
```

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 Regex! - Regex101

The screenshot shows the Regex101 online regex tester and debugger interface. The main window has several sections:

- REGULAR EXPRESSION:** A text input field containing the placeholder text "insert your regular expression here". To its right is a search icon and a "NO MATCH" button.
- TEST STRING:** A text input field containing the placeholder text "insert your test string here".
- EXPLANATION:** A panel stating, "An explanation of your regex will be automatically generated as you type."
- MATCH INFORMATION:** A panel stating, "Detailed match information will be displayed here automatically."
- QUICK REFERENCE:** A table with two columns:

FULL REFERENCE	MOST USED TOKENS
most used tokens	A single character or... [abc]
all tokens	A character except... [^abc]
CATEGORIES	A character in the ra... [a-z]
general tokens	A character not in t... [^a-z]
anchors	

On the left side, there is a vertical sidebar with the following sections and icons:

- SAVE & S...** (file icon)
- FLAVOR:** A dropdown menu currently set to "PCRE". Other options include "JS" and "PY".
- TOOLS:** A dropdown menu with icons for clipboard, copy, paste, and other tools.
- SUBSTITUTION:** A section with a dollar sign icon.

Regular Expressions Info

The screenshot shows the homepage of Regular-Expressions.info. At the top, there's a navigation bar with links for Quick Start, Tutorial, Tools & Languages, Examples, Reference, and Book Reviews. Below the navigation is a sidebar with links for Welcome, Regular Expressions Quick Start, Regular Expressions Tutorial, Replacement Strings Tutorial, Applications and Languages, Regular Expressions Examples, Regular Expressions Reference, Replacement Strings Reference, Book Reviews, Printable PDF, About This Site, and RSS Feed & Blog. A "Make a Donation" button is also present. The main content area features a section about RegexBuddy with a screenshot of the software interface showing a tree view of regex components. Below this is a welcome message and a detailed explanation of what regular expressions are and how they can be used. At the bottom, there's a book cover for "Regular Expressions Cookbook" and a footer with a Runscope status message.

Regular-Expressions.info

Quick Start Tutorial Tools & Languages Examples Reference Book Reviews

Welcome

Regular Expressions Quick Start
Regular Expressions Tutorial
Replacement Strings Tutorial
Applications and Languages
Regular Expressions Examples
Regular Expressions Reference
Replacement Strings Reference
Book Reviews
Printable PDF
About This Site
RSS Feed & Blog

Make a Donation

Easily create and understand regular expressions today. Compose and analyze regex patterns with RegexBuddy's easy-to-grasp regex blocks and intuitive regex tree, instead of or in combination with the traditional regex syntax. Developed by the author of this website, RegexBuddy makes learning and using regular expressions easier than ever. [Get your own copy of RegexBuddy now](#)

Welcome to Regular-Expressions.info
The Premier website about Regular Expressions

A regular expression (regex or regexp for short) is a special text string for describing a search pattern. You can think of regular expressions as wildcards on steroids. You are probably familiar with wildcard notations such as *.txt to find all text files in a file manager. The regex equivalent is .*\..txt\$.

But you can do much more with regular expressions. In a text editor like [EditPad Pro](#) or a specialized text processing tool like [PowerGREP](#), you could use the regular expression \b[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,4}\b to search for an email address. Any email address, to be exact. A very similar regular expression (replace the first \b with ^ and the last one with \$) can be used by a programmer to check whether the user entered a [properly formatted email address](#). In just one line of code, whether that code is written in [Perl](#), [PHP](#), [Java](#), [a .NET language](#), or a multitude of other languages.

It's going to be **200 OK** ● ©Runscope

RegExr

The screenshot shows the RegExr v2.0 interface. On the left, there's a sidebar titled "Escaped characters" with a list of various escape sequences and their corresponding regular expression patterns. Below this, two sections of explanatory text are provided.

Expression: `/([A-Z])\w+/g` 16 matches

Text: Welcome to RegExr v2.0 by gskinner.com!

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.

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
```

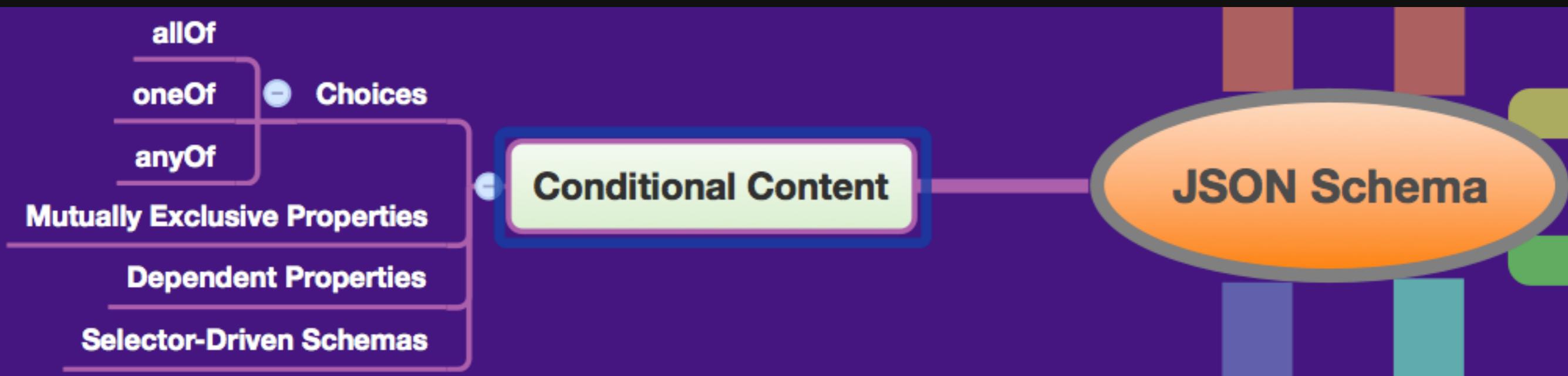
Substitution + ↴

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

```
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         "tags": {
16             "type": "array",
17             "items": {
18                 "type": "string"
19             },
20             "additionalItems": false
21         },
22         "favoriteTopic": {
23             "type": "string"
24         }
25     },
26     "additionalProperties": false,
27     "required": ["email", "firstName", "lastName"],
28     "dependencies": {
29         "tags": ["favoriteTopic"]
30     }
31 }
```

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

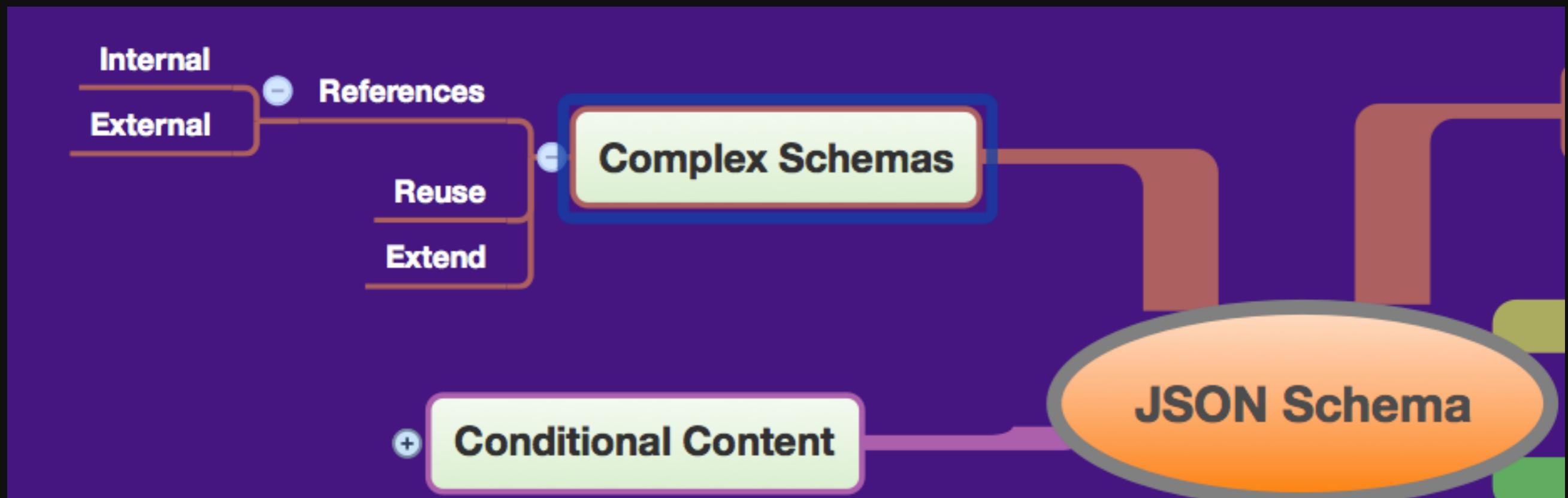
anyOf - JSON Schema

```
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

```
2  {
3      "$schema": "http://json-schema.org/draft-04/schema#",
4      "type": "object",
5      "properties": {
6          "email": {
7              "$ref": "#/definitions/emailPattern"
8          },
9          "firstName": {
10             "type": "string"
11         },
12         "lastName": {
13             "type": "string"
14         }
15     },
16     "additionalProperties": false,
17     "required": ["email", "firstName", "lastName"],
18     "definitions": {
19         "emailPattern": {
20             "type": "string",
21             "pattern": "^[\\w\\-]+@[\\w\\-]+\\. [A-Za-z]{2,4}$"
22         }
23     }
24 }
```

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

```
2  {
3      "$schema": "http://json-schema.org/draft-04/schema#",
4      "type": "object",
5      "properties": {
6          "email": {
7              "$ref":
8                  "http://localhost:8081/ex-14-my-common-schema.json#/definitions/emailPattern"
9          },
10         "firstName": {
11             "type": "string"
12         },
13         "lastName": {
14             "type": "string"
15         }
16     },
17     "additionalProperties": false,
18     "required": ["email", "firstName", "lastName"]
19 }
```

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

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

Where Are We?

JSON Schema Overview 1

Core JSON Schema 2

API Design with JSON Schema 3

JSON Search & Transform Overview 4

JSON Search 5

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

JSONPad

<https://code.google.com/p/json-pad/>

Generate JSON Schema

JSON Schema Generator

<http://jsonschema.net/>

Validate JSON Document

JSON Validate

<http://jsonvalidate.com/>

JSONLint

<http://jsonlint.com>

Document JSON Schema (HTML)

Matic

<https://github.com/mattyod/matic/>

Docson

<https://github.com/lbovet/docson>

JSONPad Demo

The screenshot shows the JSONPad application interface. The main window title is "JSONpad". The menu bar includes "File", "Edit", and "Help". The toolbar contains icons for "Tree" (selected), "Format", "Clipboard", and "Tools" (with XML and JSON options). A "Examples" dropdown is also present.

The code editor area displays the following JSON code:

```
2 "about" : "Fred Smith is the CTO of Full Ventures, where he ...",
3 "email" : "fred.smith@fullventures.com",
4 "firstName" : "Fred",
5 "lastName" : "Smith",
6 "tags" : [
7     "JavaScript",
8     "REST",
9     "JSON"
10 ],
11 "company" : "Full Ventures, Inc."
12 }
```

The bottom left pane is a tree view titled "JSON" showing the structure of the JSON object:

- about
- email
- firstName
- lastName
- tags
- company

The right side of the interface is a large, empty "Edit" pane.

JSON Schema Generator Demo

The screenshot shows a web-based JSON Schema Generator tool. On the left, a JSON document is pasted into a text area:

```
{  
  "about": "Fred Smith is the CTO of Full Ventures, where he ...",  
  "email": "fred.smith@fullventures.com",  
  "firstName": "Fred",  
  "lastName": "Smith",  
  "tags": [  
    "JavaScript",  
    "REST",  
    "JSON"  
,  
  "company": "Full Ventures, Inc."  
}
```

A green message box at the bottom left says "Well done! You provided valid JSON."

On the right, the generated JSON Schema is displayed in a "Code View" tab:

```
{  
  "$schema": "http://json-schema.org/draft-04/schema#",  
  "id": "/",  
  "type": "object",  
  "properties": {  
    "about": {  
      "id": "about",  
      "type": "string"  
    },  
    "email": {  
      "id": "email",  
      "type": "string"  
    },  
    "firstName": {  
      "id": "firstName",  
      "type": "string"  
    },  
    "lastName": {  
      "id": "lastName",  
      "type": "string"  
    },  
    "tags": {  
      "id": "tags",  
      "type": "array",  
      "items": {  
        "id": "2",  
        "type": "string"  
      },  
      "additionalItems": false  
    },  
    "company": {  
      "id": "company",  
      "type": "string"  
    }  
  "additionalProperties": false,  
  "required": [  
    "about",  
    "email",  
    "firstName",  
    "lastName",  
    "tags",  
    "company"  
  ]  
}
```

The interface includes several configuration sections on the left:

- URL:** http://jsonsatwork.org
- JSON:** The pasted JSON document.
- Metadata:**
 - Include metadata keywords
- General:**
 - Include default values

Values are taken from JSON.
 - Restrict values to enum

Uses the default value and null.
 - Use absolute IDs
 - Force required
- Objects:**
 - Allow additional properties

Controls whether it's valid to have additional properties in the object beyond what is defined in the schema.
- Arrays:**
 - Allow additional items

Controls whether it's valid to have additional items in the array beyond what is defined in the schema.

At the top right, there are browser-style buttons and a user name "Tom".

JSON Validate Demo

The screenshot shows the JSON Validate demo interface. At the top, the title "JSON Validate" is displayed in a large blue header. Below it, the URL "jsonvalidate.com" is shown in the address bar. The main area is divided into two main sections: "JSON Schema" and "JSON Content".

JSON Schema: A code editor containing a JSON schema definition. The schema includes properties like "tags", "company", and "required" fields.

```
22     "tags": [
23       "array",
24       "items": {
25         "type": "string"
26       },
27       "additionalItems": false
28     },
29     "company": {
30       "id": "company",
31       "type": "string"
32     }
33   },
34   "additionalProperties": false,
35   "required": [
36     "about",
37     "email",
38     "firstName",
39     "lastName",
40     "tags",
41     "company"
42   ]
43 ]
```

JSON Content: A code editor containing a JSON object. The object contains fields such as "about", "email", "firstName", "lastName", "tags", and "company".

```
1 {
2   "about": "Fred Smith is the CTO of Full Ventures, where he ...",
3   "email": "fred.smith@fullventures.com",
4   "firstName": "Fred",
5   "lastName": "Smith",
6   "tags": [
7     "JavaScript",
8     "REST",
9     "JSON"
10 ],
11   "company": "Full Ventures, Inc."
12 }
```

References: A sidebar on the left showing numbered references from 1 to 14.

Results: A panel on the right displaying the validation status "Valid".

Buttons: At the bottom left are "Validate" and "Reset all" buttons. At the bottom right is a link "Learn more about Using JSON Schema" with a "UJS" logo.

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

The screenshot shows the Swagger UI interface for the Petstore API. The title bar reads "Swagger UI" and the address bar shows "lbovet.github.io/swagger-ui/dist/index.html#/pet/getPetById_get_0". The main header has a green bar with the "swagger" logo, a gear icon, and the URL "http://lbovet.github.io/swagger-ui/dist/api/index.json". There are also "api_key" and "Explore" buttons.

Swagger Example with Typson and Docson Integration

This is a sample taken from <http://swagger.wordnik.com>
It illustrate both [Typson](#) and [Docson](#) integration with Swagger

user : Operations about user

pet : Operations about pets

GET /pet/{petId} [Find pet by ID](#)

Implementation Notes
Returns a pet based on ID

Response Class

[Model](#) [Model Schema](#)

Pet

id	<code>int64 [0.0;100.0]</code>	Unique identifier for the Pet
category	Category	Category the pet is in
allowedCategories	<code>map integer</code>	
name	<code>string</code>	Friendly name of the pet
photoUrls	<code>array string</code>	Image URLs
tags	<code>map Tag</code>	Tags assigned to this pet
status	<code>enum string</code>	available pending sold pet status in the store

<https://github.com/lbovet/docson>

Swagger Petstore - Regular

Swagger UI

petstore.swagger.io#!/pet/getPetById

GET /pet/{petId} Find pet by ID

Implementation Notes
Returns a single pet

Response Class (Status 200) OFF !

Model Model Schema

```
{
  "id": 0,
  "category": {
    "id": 0,
    "name": "string"
  },
  "name": "doggie",
  "photoUrls": [
    "string"
  ],
  "tags": [
    "string"
  ]
}
```

Response Content Type application/xml

Parameters

Parameter	Value	Description	Parameter Type	Data Type
petId	(required)	ID of pet to return	path	long

Response Messages

HTTP Status Code	Reason	Response Model	Headers
400	Invalid ID supplied		
404	Pet not found		

Try it out!

My JSON Stub API Workflow

Model JSON Document

JSONPad

[https://
code.google.com/p/
json-pad/](https://code.google.com/p/json-pad/)

Generate JSON Document

JSON Generator

[http://www.json-
generator.com/](http://www.json-generator.com/)

Deploy Stub REST API

JSON Server

[https://github.com/
typicode/json-server](https://github.com/typicode/json-server)

JSON Generator Demo

The screenshot shows a web application titled "JSON GENERATOR" at www.json-generator.com. The interface includes a toolbar with "Generate", "Reset", "Try out beta!", "Help", and "Feedback" buttons. A large text area on the left contains a JSON template with code comments and placeholder values:

```
1 // Template for http://www.json-generator.com/
2
3 [           1
4   {{repeat(3)}}, {
5     id: '{{integer()}}',
6     picture: 'http://placehold.it/32x32',
7     name: '{{firstName()}}',
8     lastName: '{{surname()}}',
9     company: '{{company()}}',
10    email: '{{email()}}',
11    about: '{{lorem(1, "paragraphs")}}'
12  }
13 ]
```

The right side of the screen features a dark background with a starry theme and the text "Click 'Generate' and wait for the magic". At the bottom, there are social sharing links: "Follow" (Twitter), "Tweet" (Twitter), "1,676" (Facebook), "Like" (Facebook), "Share" (Facebook), and "1k" (Facebook).

Created by [Vazha Omanashvili](#), Sponsored by [Runscope API Tools](#)

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

Core JSON Schema 2

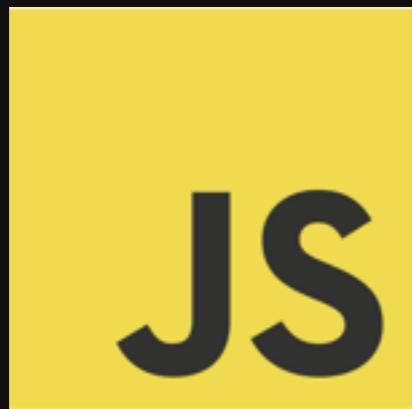
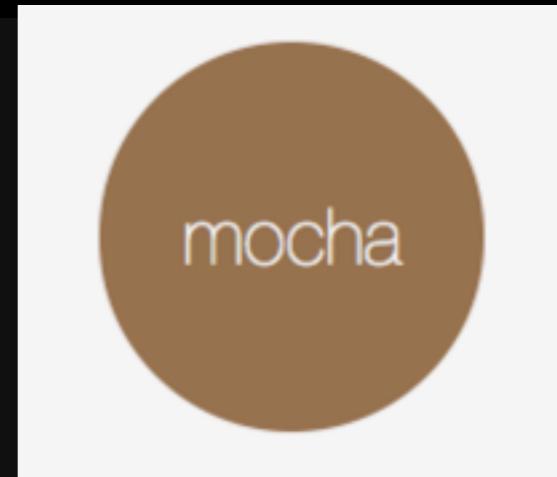
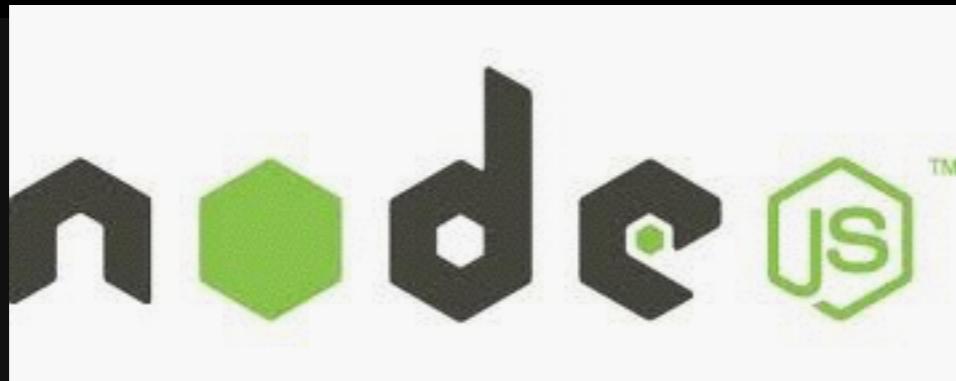
API Design with JSON Schema 3

JSON Search & Transform Overview 4

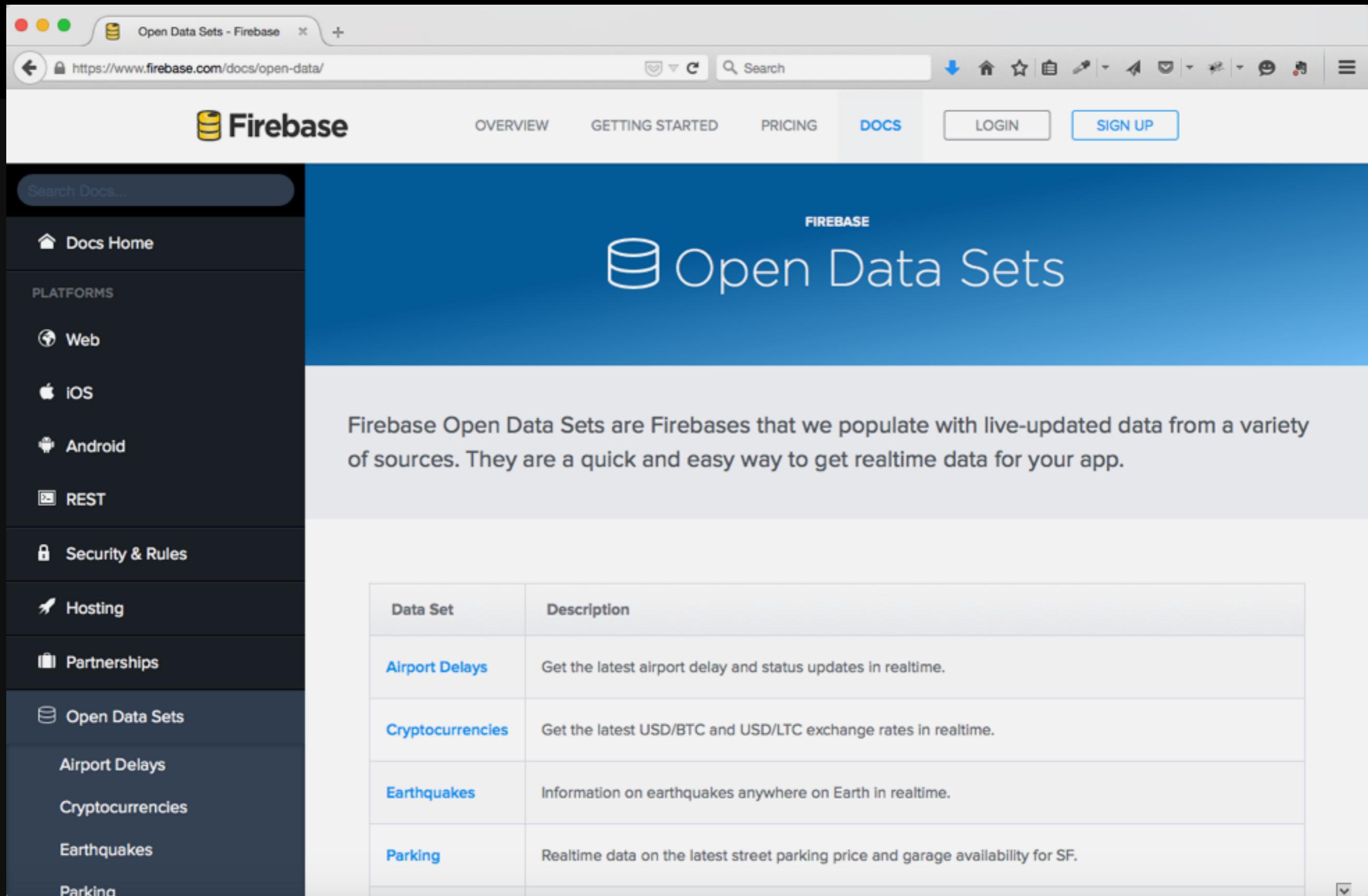
JSON Search 5

JSON Transform 6

Our Client Stack



Firebase Open Data Set



The screenshot shows a web browser displaying the Firebase Open Data Sets documentation at <https://www.firebaseio.com/docs/open-data/>. The page has a blue header with the Firebase logo and navigation links for Overview, Getting Started, Pricing, Docs (which is highlighted), Login, and Sign Up. A search bar is at the top right. On the left is a dark sidebar with links for Docs Home, Platforms (Web, iOS, Android, REST), Security & Rules, Hosting, Partnerships, and Open Data Sets (which is also highlighted). The main content area features a large database icon and the title "Open Data Sets". Below this, a paragraph explains what Firebase Open Data Sets are. A table lists five available datasets: Airport Delays, Cryptocurrencies, Earthquakes, and Parking.

FIREBASE

Open Data Sets

Firebase Open Data Sets are Firebases that we populate with live-updated data from a variety of sources. They are a quick and easy way to get realtime data for your app.

Data Set	Description
Airport Delays	Get the latest airport delay and status updates in realtime.
Cryptocurrencies	Get the latest USD/BTC and USD/LTC exchange rates in realtime.
Earthquakes	Information on earthquakes anywhere on Earth in realtime.
Parking	Realtime data on the latest street parking price and garage availability for SF.

JSON Server - Airports

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

<http://localhost:5000/airports>

Airports Stub Service

```
[{"id": 1, "IATA": "ATL", "ICAO": "KATL", "city": "Atlanta", "delay": true, "name": "The William B Hartsfield International", "state": "Georgia", "status": {"avgDelay": "", "closureBegin": "", "closureEnd": "", "endTime": "", "maxDelay": "30 minutes", "minDelay": "16 minutes", "reason": "WX:Wind", "trend": "Increasing", "type": "Arrival"}, "weather": {"temp": "68.0 F (20.0 C)", "visibility": 9, "weather": "Light Rain", "wind": "South at 12.7mph"}}, {"id": 2, "IATA": "BNA", "ICAO": "KBNA", "city": "Nashville", "delay": false, "name": "Nashville International", "state": "Tennessee", "status": {"avgDelay": "", "closureBegin": "", "closureEnd": "", "endTime": "", "maxDelay": "", "minDelay": "", "reason": "No known delays for this airport.", "trend": "", "type": ""}, "weather": {"temp": "69.0 F (20.6 C)", "visibility": 4, "weather": "Thunderstorm Light Rain Fog/Mist"}]}
```

JSON Search & Transform Rubrik



Where Are We?

JSON Schema Overview 1

Core JSON Schema 2

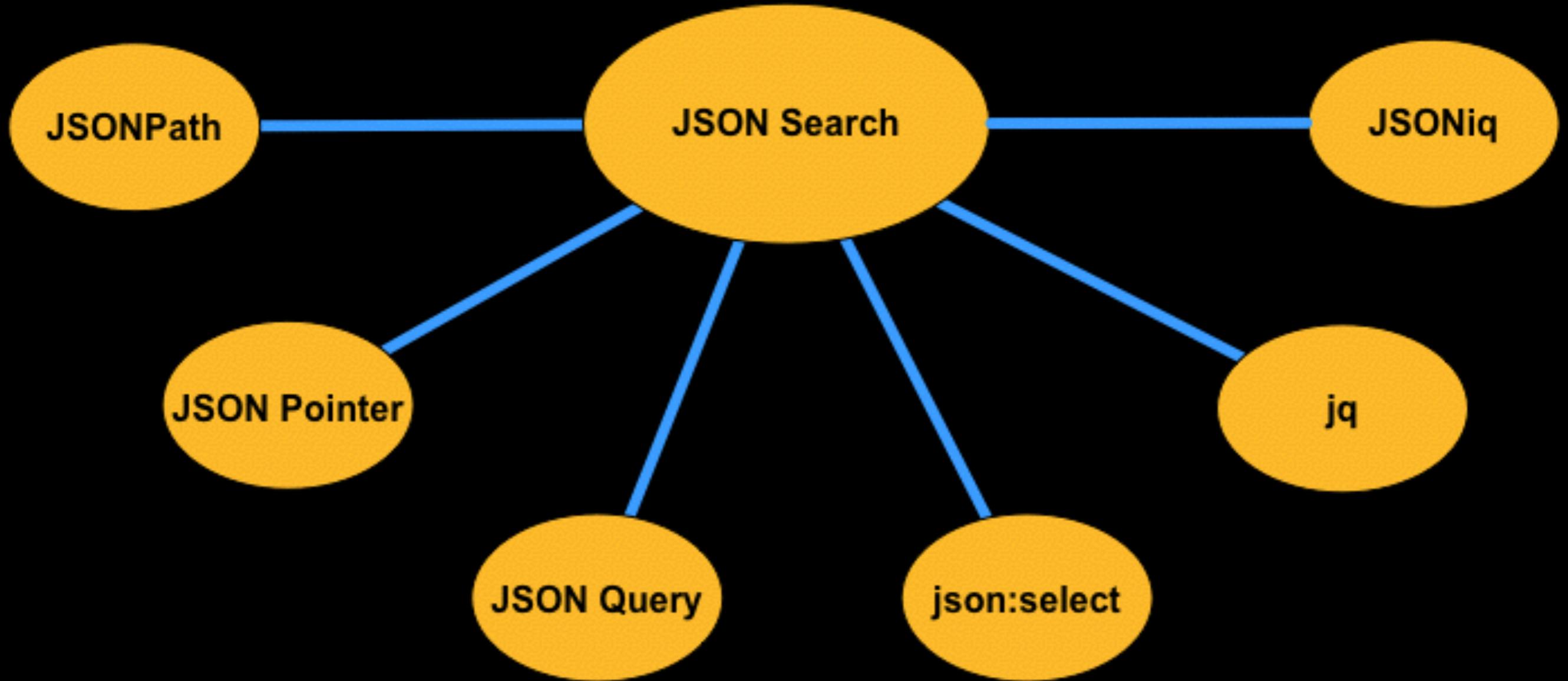
API Design with JSON Schema 3

JSON Search & Transform Overview 4

JSON Search 5

JSON Transform 6

JSON Search Tools



JSONPath

The screenshot shows a web browser window with the following details:

- Title Bar:** JSONPath - XPath for JSON
- Address Bar:** goessner.net/articles/JsonPath/
- Page Content:**
 - Header:** <stefan.goessner/>
Mechanik, das Web und der ganze Rest
 - Navigation:** Home | Lehre | Download | Info
 - Section:** # JSONPath - XPath for JSON | 2007-02-21 | e1
 - Text:** 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.
 - Text:** It's time to wonder, if there is a need for something like XPath4JSON and what are the problems it can solve.
 - List:**
 - 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.
 - Text:** 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?
 - Text:** 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.
 - Text:** The following XPath expression
 - Text:** /store/book[1]/title
 - Text:** would look like
 - Text:** x.store.book[0].title
 - Text:** or
 - Text:** x['store']['book'][0]['title']
 - Text:** in Javascript, Python and PHP with a variable x holding the JSON structure. Here we
- Right Sidebar:**
 - Search:** >> Search ..
Web goessner.net
Google Search
 - Inhalt:** >> Inhalt ..
Home
Lehre
Dynamik
Articles
DOM Events
Wiky
2D Vectors
Slideous
JsonT
JSONPath
SVG
Download
Admin
Info
 - Comments:** >> 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 cheaper than 10
//*	\$..*	all Elements in XML document. All members of JSON structure.

JSONPath Expression Tester

The screenshot shows a web browser window for the "JSONPath Expression Tester" application. The URL in the address bar is "jsonpath.curiousconcept.com". The page has a green header with the title "JSONPATH EXPRESSION TESTER". On the left, there's a large white arrow pointing right with the text "Paste in JSON or a URL, enter the JSONPath and away you go.". In the center, there's a large white input area labeled "JSON Data/URL". Below it is a smaller input area labeled "JSONPath Expression". At the bottom center is a green button labeled "Process". To the right, there are two dropdown menus: "JSON Template" set to "2 Space Tab" and "Implementation" set to "JSONPath 0.8.3". The overall design is clean and modern.

JSONPath Expression Tester

jsonpath.curiousconcept.com

Search

About Learn Changelog Contact

JSON Data/URL

Paste in JSON or a URL, enter the JSONPath and away you go.

JSONPath Expression

Process

JSON Template

2 Space Tab

Implementation

JSONPath 0.8.3

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

The screenshot shows a web browser window displaying the RFC 6901 document. The title bar reads "RFC 6901 - JavaScript Obj...". The address bar shows the URL "tools.ietf.org/html/rfc6901". The page content includes the RFC header with authors (P. Bryan, Ed. from Salesforce.com, K. Zyp from SitePen (USA), M. Nottingham, Ed. from Akamai), the title "JavaScript Object Notation (JSON) Pointer", an "Abstract" section, a "Status of This Memo" section, a "Copyright Notice" section, and a "Page-Footer" section at the bottom.

RFC 6901 - JavaScript Object Notation (JSON) Pointer

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

RFC 6901

JSON Pointer

April 2013

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 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 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 supersedes 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

The screenshot shows a web browser window with the title bar "JSONSelect". The address bar contains "jsonselect.org/#overview". The page content is as follows:

json:select()

CSS-like selectors for JSON.

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?

A code block on the right illustrates the selector ".author .drinkPref :first-child" applied to a JSON object. The "whiskey" value under "drinkPref" is highlighted with a yellow box.

```
".author .drinkPref :first-child"
-----
{
  "author": {
    "name": {
      "first": "Lloyd",
      "last": "Hilaiel"
    },
    "drinkPref": [
      "whiskey"
      "beer",
      "wine"
    ],
    "thing": "JSONSelect site",
    "license": "(cc) BY-SA"
  }
}
```

At the bottom left is the URL "https://github.com/lloyd/JSONSelect". A "Fork me on GitHub" button is located at the top right of the page.

json:select Expression Tester

The screenshot shows a web browser window for the "JSONSelect Expression Tester" application. The title bar indicates the site is "JSONSelect Expression Tester" at "jsonselect.curiousconcept.com/#". The main interface has a teal header with the "JSONSELECT EXPRESSION TESTER" logo and navigation links for "About", "Learn", "Changelog", and "Contact". Below the header, there's a large white input area labeled "JSON Data/URL" with a placeholder for JSON or a URL. To the left of this area is a large white arrow pointing right, and below it is a descriptive text: "Paste in JSON or a URL, enter the JSONSelect and away you go.". To the right of the input area is another large white arrow pointing left, and above it is a dropdown menu labeled "JSON Template" with the option "3 Space Tab" selected. At the bottom of the page, there's a teal footer bar with the text "JSONSelect Expression" and a teal button labeled "Process".

JSONSelect EXPRESSION TESTER

About Learn Changelog Contact

JSON Data/URL

Paste in JSON or a URL, enter the JSONSelect and away you go.

JSON Template

3 Space Tab

JSONSelect Expression

Process

json:select Demo

json:select 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  });
```

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

The screenshot shows a web browser window displaying the official jq website at stedolan.github.io/jq/. The page features a large, stylized logo with a dot and a slash followed by the letters 'jq'. To the right of the logo is a text block stating 'jq is a lightweight and flexible command-line JSON processor.' Below this are two buttons: 'Download jq-1.4-2-g15c4a7f-dirty' and 'Try online!'. The main content area contains three columns of text comparing jq to other tools like sed, awk, and grep. At the bottom, there's a 'News' section with a single item about the release of jq 1.4.

jq

stedolan.github.io/jq/

jq Tutorial Download Manual Issues Source Try online! NEWS

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 1.4 (finally) released! Get it on the [download](#) page.

jq play

The screenshot shows the jq play web application interface. At the top, there is a browser header with tabs, a search bar, and various icons. The main title is "jq play" followed by the subtitle "A playground for jq 1.4". Below the title, there are two large input fields: "Filter" and "JSON". The "Filter" field contains the expression ".foo, .foo.bar, .foo?". The "Result" field displays the output "1 |". To the right of the Result field are several checkboxes: "Compact Output", "Null Input", "Raw Input", "Raw Output", and "Slurp". Below these fields, there is a "Cheatsheet" section with a note: "Click on the icons (ⓘ) in the table below to see examples." The cheatsheet table has four rows:

Cheatsheet	
<code>.</code>	unchanged input
<code>.foo, .foo.bar, .foo?</code>	value at key
<code>,</code>	feed input into multiple filters
<code> </code>	pipe output of one filter to the next filter

jq Examples

```
2 In jq-play
3 -----
4 .airports
5
6 .airports[10]
7
8 .airports[10] | { id, IATA, weather }
9
10 .airports[10:15] | .[] | { id, IATA, weather }
11
12
13 In curl
14 -----
15 curl 'http://localhost:5000/airports'
16
17 curl 'http://localhost:5000/airports' | jq .[10]
18
19 curl 'http://localhost:5000/airports' | jq '.[10] | { id, IATA, weather }'
20
21 curl 'http://localhost:5000/airports' | jq '.[10:15] | .[] | { id, IATA, weather }'
```

JSONiq

JSONiq - The JSON Query Lang... +

www.jsoniq.org

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

Core JSON Schema 2

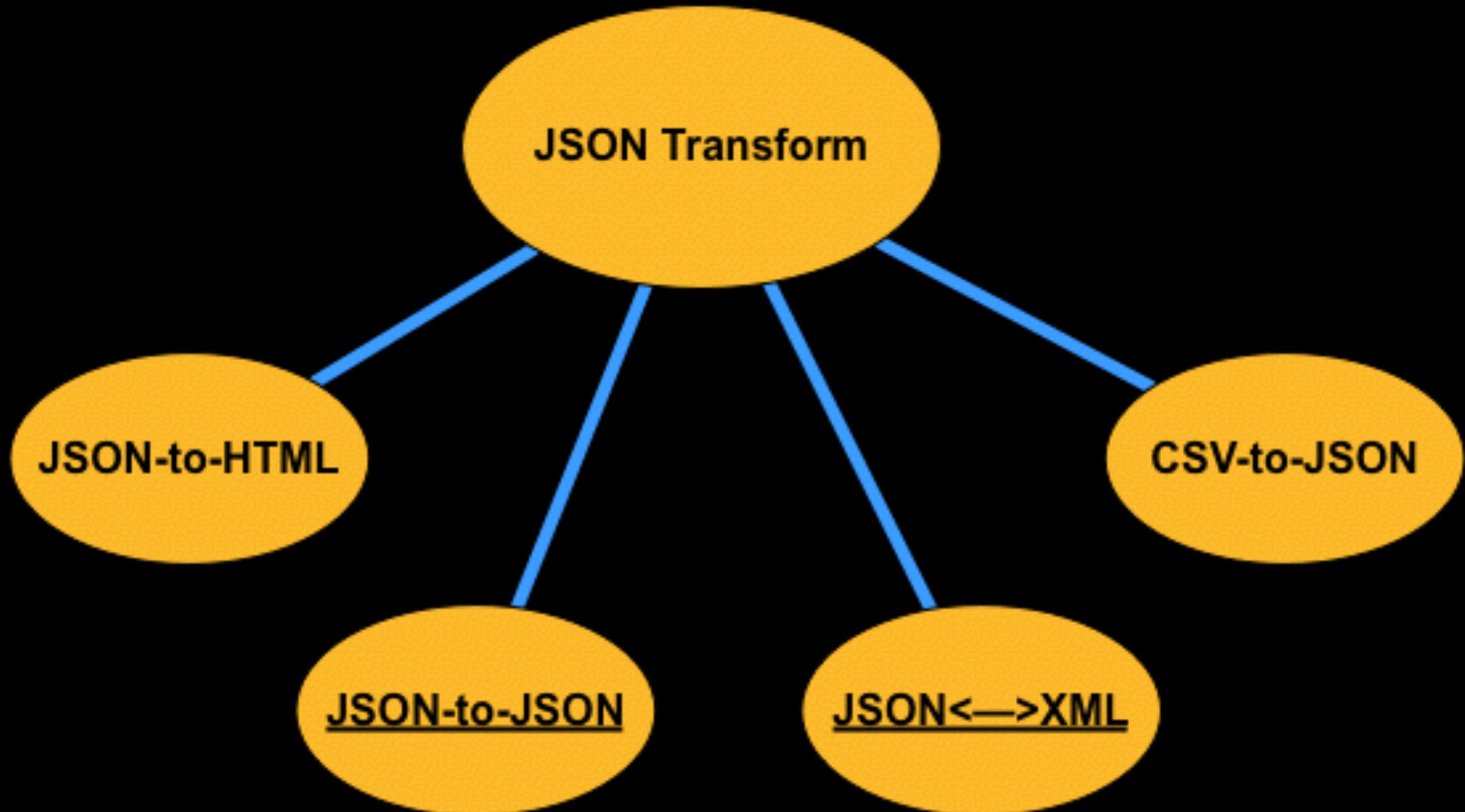
API Design with JSON Schema 3

JSON Search & Transform Overview 4

JSON Search 5

JSON Transform 6

JSON Transform



JSON-T

The screenshot shows a web browser window with the title "JsonT - Transforming Json". The address bar displays "goessner.net/articles/jsont/". The main content area contains two sections: "# Transforming JSON" and "# Introducing JSONT". The "# Transforming JSON" section discusses the use of XSLT-like rules to transform JSON data into other formats like XML or HTML. It includes code snippets for both JSON and HTML. The "# Introducing JSONT" section shows how a simple JSON object can be transformed into an HTML link element. A sidebar on the right provides search and navigation links.

<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 achieve 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.

```
<a href="http://company.com">company homepage</a>
```

For doing this we can write a corresponding rule

```
{ "link": "<a href=\"{link.uri}\">\">{link.title}</a>" }
```

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 · GitHub

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/: Clone in Desktop Download ZIP

You can clone with HTTPS or Subversion.

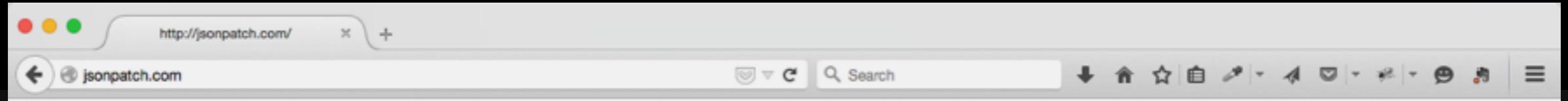
jsonapter Test

```
2  var expect = require('chai').expect;
3  var jsonfile = require('jsonfile');
4  var j2j = require('jsonapter').instance();
5
6  var template = {
7    content: {
8      name: {
9        dataTransform: function(input) {
10          return input.firstName + ' ' + input.lastName;
11        }
12      },
13      email: { dataKey: 'email' },
14      about: { dataKey: 'about' },
15    }
16  };
17
18 describe('jsonapter', function() {
19   describe('run', function() {
20     it('should transform JSON', function(done) {
21       var jsonFileName = './data/speaker.json';
22
23       jsonfile.readFile(jsonFileName, function(err, jsonObj) {
24         if (!err) {
25           console.log(jsonObj);
26           console.log('\n\n\njsonapter Test');
27           var output = j2j.run(template, jsonObj);
28           console.log('\n\n\nTransformed JSON');
29           console.log(JSON.stringify(output));
30         }
31         done();
32       });
33     });
34   });
35 });
36
37 // Output:
38 // {
39 //   "name": "John Doe",
40 //   "email": "john.doe@example.com",
41 //   "about": "I am a speaker at Node.js conferences."
42 // }
```

jsonapter Scorecard

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

JSON Patch



The screenshot shows a web browser window with the URL <http://jsonpatch.com/> in the address bar. The page title is "What is JSONPatch?". Below the title, there is a paragraph of text and several code snippets. The browser interface includes a back button, forward button, search bar, and various toolbar icons.

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

The screenshot shows a web browser window displaying the IETF RFC 6902 document. The title bar reads "RFC 6902 - JavaScript Obj...". The address bar shows the URL "https://tools.ietf.org/html/rfc6902". Below the address bar, there are links for "[Docs]", "[txt|pdf]", "[draft-ietf-appsaw...]", "[Diff1]", and "[Diff2]". A blue horizontal bar highlights the "[Docs]" link. The main content area has a header "PROPOSED STANDARD". On the left, it lists the document's metadata: "Internet Engineering Task Force (IETF)", "Request for Comments: 6902", "Category: Standards Track", and "ISSN: 2070-1721". On the right, it lists the editors: "P. Bryan, Ed. Salesforce.com", "M. Nottingham, Ed. Akamai", and the date "April 2013". The title of the document is "JavaScript Object Notation (JSON) Patch". The "Abstract" section describes JSON Patch as defining a structure for expressing operations to apply to a JSON document using the HTTP PATCH method. The "Status of This Memo" section states it is an Internet Standards Track document. It also notes that the document is a product of the IETF, represents community consensus, and has been approved by the IESG. A link to "Section 2 of RFC 5741" is provided. The "Copyright Notice" section at the bottom states the copyright belongs to the IETF Trust and document authors.

RFC 6902 - JavaScript Obj...

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

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

PROPOSED STANDARD

Internet Engineering Task Force (IETF)
Request for Comments: 6902
Category: Standards Track
ISSN: 2070-1721

P. Bryan, Ed.
Salesforce.com
M. Nottingham, Ed.
Akamai
April 2013

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>.

Copyright Notice

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

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

```
2  var expect = require('chai').expect;
3  var jsonfile = require('jsonfile');
4  var jsonpatch = require('json-patch');
5
6  var template = [
7    { op: 'add', path: '/submittedSlides', value: true },
8    { op: 'remove', path: '/tags' },
9    { op: 'remove', path: '/company' }
10];
11
12 describe('json-patch', function() {
13   describe('apply', function() {
14     it('should patch JSON', function(done) {
15       var jsonFileName = './data/speaker.json';
16
17       jsonfile.readFile(jsonFileName, function(err, jsonObj) {
18         if (!err) {
19           console.log(jsonObj);
20           console.log('\n\n\nJSONPatch Test');
21           var output = jsonpatch.apply(jsonObj, template);
22           console.log('\n\n\nPatch JSON');
23           console.log(JSON.stringify(output));
24         }
25         done();
26       });
27     });
28   });
29 });
```

JSON Patch Scorecard

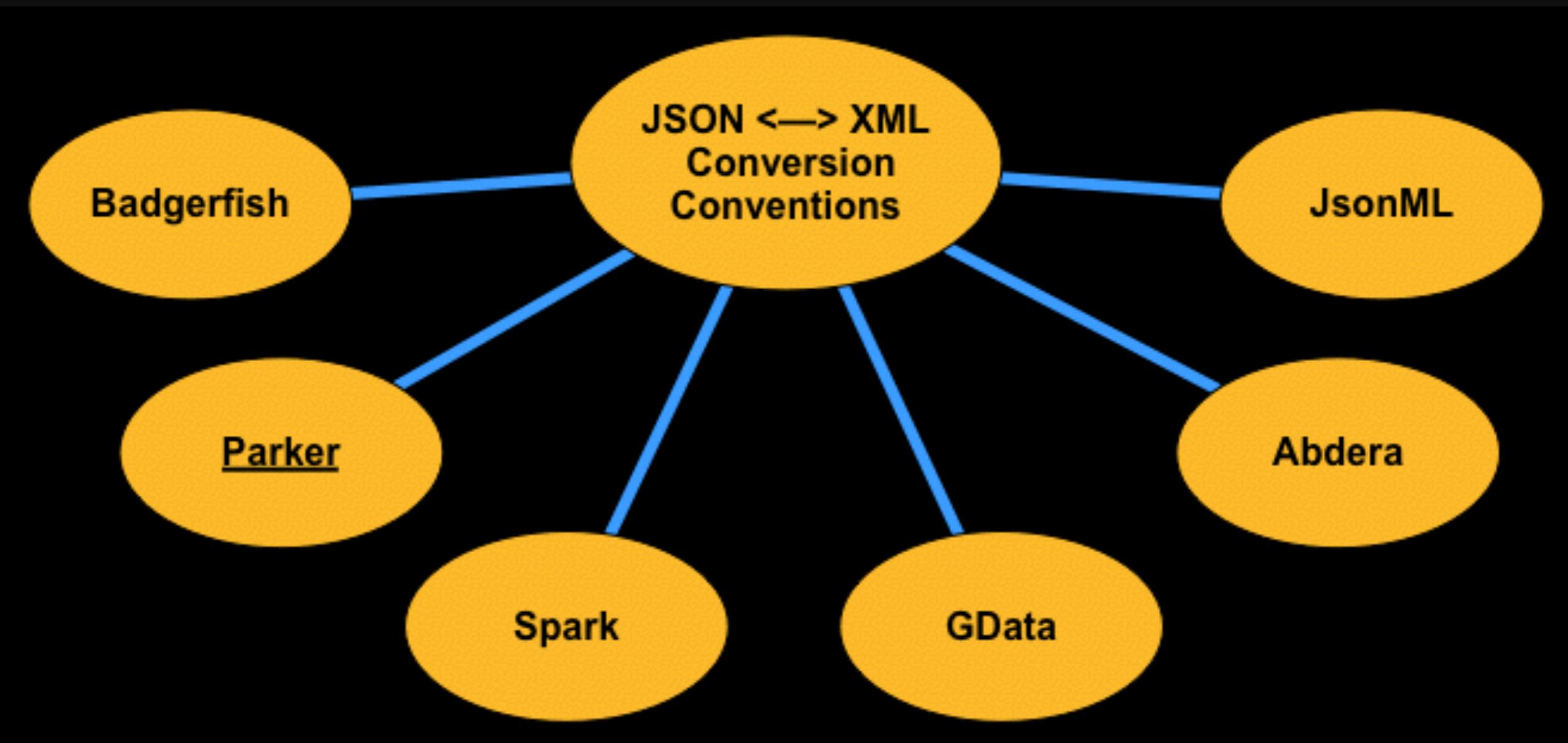
Mindshare	Y
Dev Community	Y
Platforms	JS, Node.js, Java, RoR, etc.
Intuitive	Y
Standard	RFC 6902 - Yes!

Project 5 - JSON Transform

[projects/README.md#project-5---json-transform](#)

[projects/EspressoCON.md](#)

JSON - XML Conversion Conventions



Badgerfish Convention

The screenshot shows a Ning network interface with the title "BADGERFISH". The main navigation bar includes "MAIN", "MY PAGE", "MEMBERS", and "PHOTOS". The "PHOTOS" section displays a single photo titled "<?xml ?>" by "BADGERFISH", which is described as an XML icon. Below the photo are buttons for "+ Add Photos" and "View All". A link to "http://memphisdot.net/" is also present. To the right, a "WHAT IS BADGERFISH?" section defines it as a convention for translating XML documents into JSON objects, mentioning its ease of manipulation from JavaScript and its similarity to PHP's SimpleXML extension.

WHAT IS BADGERFISH?

BadgerFish is a convention for translating an XML document into a JSON object. Once you've got your XML document represented as a JSON object, it's easy to manipulate from within Javascript. If you're familiar with PHP's SimpleXML extension, think of BadgerFish as aiming for a similar goal: making it simpler to do common manipulations of XML documents with a predictable structure.

How does it work?

Here are the rules:

1. Element names become object properties
2. Text content of elements goes in the \$ property of an object.

```
<alice>bob</alice>
```

becomes

```
{ "alice": { "$" : "bob" } }
```
3. Nested elements become nested properties

```
<alice><bob>charlie</bob><david>edgar</david></alice>
```

becomes

```
{ "alice": { "bob" : { "$": "charlie" }, "david": { "$": "edgar" } } }
```
4. Multiple elements at the same level become array elements.

```
<alice><bob>charlie</bob><bob>david</bob></alice>
```

becomes

Parker Convention

The screenshot shows a Mozilla Firefox browser window with the title bar "JXON | MDN". The address bar displays "Mozilla Foundation [US] https://developer.mozilla.org/en-US/docs/JXON/The_Parker_Convention". The main content area contains text about the Parker Convention, followed by a section titled "The Parker Convention" with a detailed explanation. A note at the bottom of the page states that the convention was written for XSLT and is mostly futile for JavaScript. The right side of the page features a sidebar titled "IN THIS ARTICLE" with a list of links related to the Parker Convention.

The Parker Convention

The functions listed above for the conversion of an XML document to [JSON](#) (often called "JXON algorithms") are more or less freely based on the Parker Convention (especially regarding the transformation of [tags names](#) into [object properties names](#), the recognition of the [typeof](#) of all the collected [text content](#) of each tag and the absorption of solitary Text and/or CDATASection nodes into primitive values). It is called "Parker Convention" in opposition to "BadgerFish Convention", after the comic Parker & Badger by Cuadrado. See also: [BadgerFish Convention](#).

The following is a transcription of the Parker Convention paper (version 0.4), from the page "[TransformingRules](#)" of the [xml2json-xslt project](#) site.

This Convention was written in order to regulate the conversion to [JSON](#) from [XSLT](#), so parts of it are futile for JavaScript.

Note: On October 29th, 2013, the World Wide Web Consortium released [in a note](#) on official algorithm for converting [HTML5 microdata](#) to [JSON](#). However, [HTML microdata](#) is not [HTML](#): [microdata](#) is a formatted subset of [HTML](#).

Translation JSON

1. The root element will be absorbed, for there is only one:

```
1 <root>test</root>
```

becomes
 - 1 "test"
2. Element names become object properties:

IN THIS ARTICLE

- [Conversion snippets](#)
- [Algorithm #1: a verbose way](#)
- [Algorithm #2: a less verbose way](#)
- [Algorithm #3: a synthetic technique](#)
- [Algorithm #4: a very minimalist way](#)
- [Reverse algorithms](#)
- [The Parker Convention](#)
- [Translation JSON](#)
- [Extra JavaScript translations](#)
- [In summary](#)
- [Code considerations](#)
- [Appendix: a complete, bidirectional, JXON library](#)
- [Usage](#)
 - [JXON.build syntax](#)
 - [JXON.build description](#)
 - [JXON.build parameters](#)
 - [JXON.unbuild syntax](#)
 - [JXON.unbuild description](#)
 - [JXON.unbuild parameters](#)
 - [Extend the native `Element.prototype` object](#)
 - [Example](#)
 - [Other examples](#)
 - [Example #1: How to use JXON to](#)

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\njxon Test - XML ==> JSON');
15            console.log('\n\n\nXML');
16            console.log(xmlData);
17            var output = jxon.stringToJs(xmlData);
18            console.log('\n\n\nTransformed JSON');
19            console.log(JSON.stringify(output));
20          }
21          done();
22        });
23      });
24    });
25  });
26});
```

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\njxon Test - JSON ==> XML');
33         console.log('\n\n\nJSON');
34         console.log(jsonObj);
35         var xml = jxon.toString(jsonObj);
36         console.log('\n\n\nTransformed XML');
37         console.log(xml);
38       }
39       done();
40     });
41   });
42 });
43 });
```

If All Else Fails ... codebeautify.org

The screenshot shows the homepage of codebeautify.org. At the top, there's a navigation bar with links for 'My Ip', 'Recent links', 'Sample', 'More', and 'Sign In'. Below the navigation, the main title 'Code Beautify' is displayed next to a profile icon. A banner at the top says 'Cb Free Online Tools For Developers'.

Popular Functionality

- Json Validator
- JSON Viewer
- JSON Editor
- XML Viewer
- HTML Viewer
- XML to JSON
- Encryption-Decryption
- EXCEL TO HTML
- CSS Validator
- CSV to JSON
- Javascript Validator
- CSS Beautifier

Web Viewer

- JSON
- XML
- MXML
- HTML
- CSS
- JavaScript
- RSS
- JAVA

Agenda

JSON Schema Overview 1

Core JSON Schema 2

API Design with JSON Schema 3

JSON Search & Transform Overview 4

JSON Search 5

JSON Transform 6

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/mckegg/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://github.com/lloyd/JSONSelect/blob/master/JSONSelect.md)

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

[https://github.com/fd/json select](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/json/>

<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://wiki.open311.org/JSON_and_XML_Conversion/)

<http://badgerfish.ning.com/>

[https://developer.mozilla.org/en-US/docs/JXON#The Parker Convention](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/>