

# **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**

**JSON Search & Transform Overview**

**4**

**Core JSON Schema**

**2**

**JSON Search**

**5**

**API Design with JSON Schema**

**3**

**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**

**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**

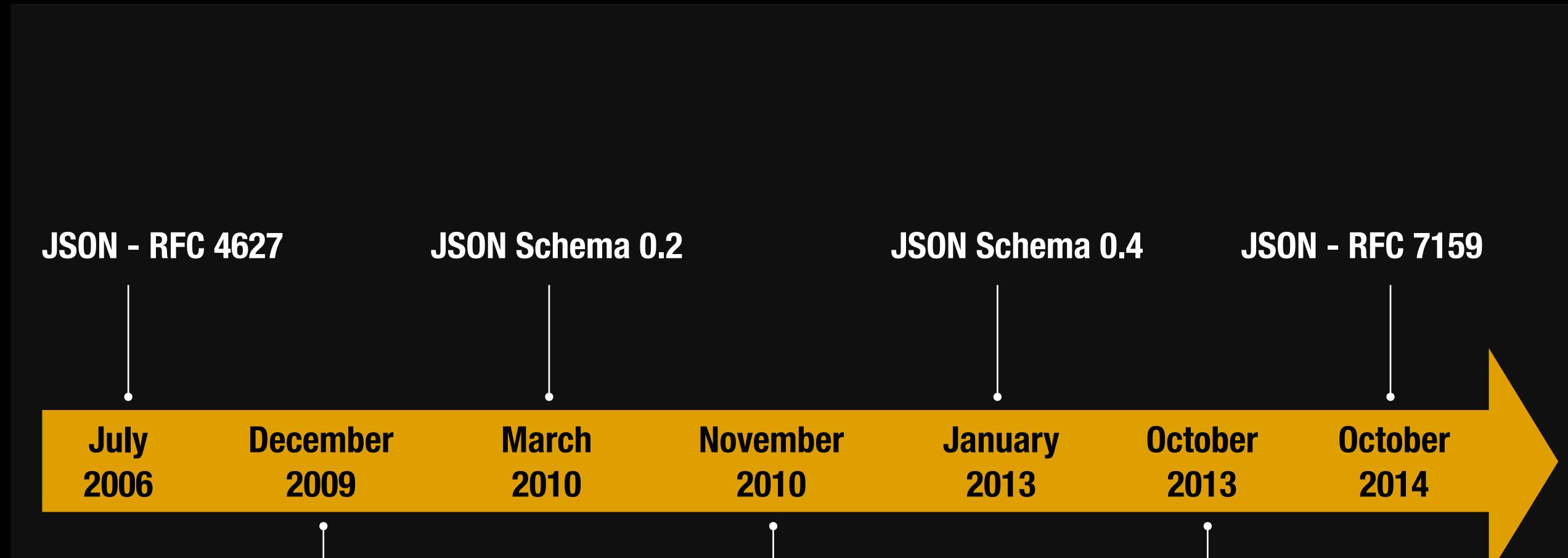
# Basic JSON Schema

```
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?



JSON Schema 0.1

JSON Schema 0.3

JSON - ECMA-404

# The Journey ...



# json-schema.org

The screenshot shows the homepage of json-schema.org as it would appear in a web browser. The page has a dark green header bar with the title "json-schema.org" and the subtitle "The home of JSON Schema". Below the header, there is a navigation menu with four items: "about" (underlined), "docs", "examples", and "software". The main content area is divided into several sections:

- 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 URIs)
    - forms - specify a JSON Schema for the desired data
- More**

Interested? Check out:

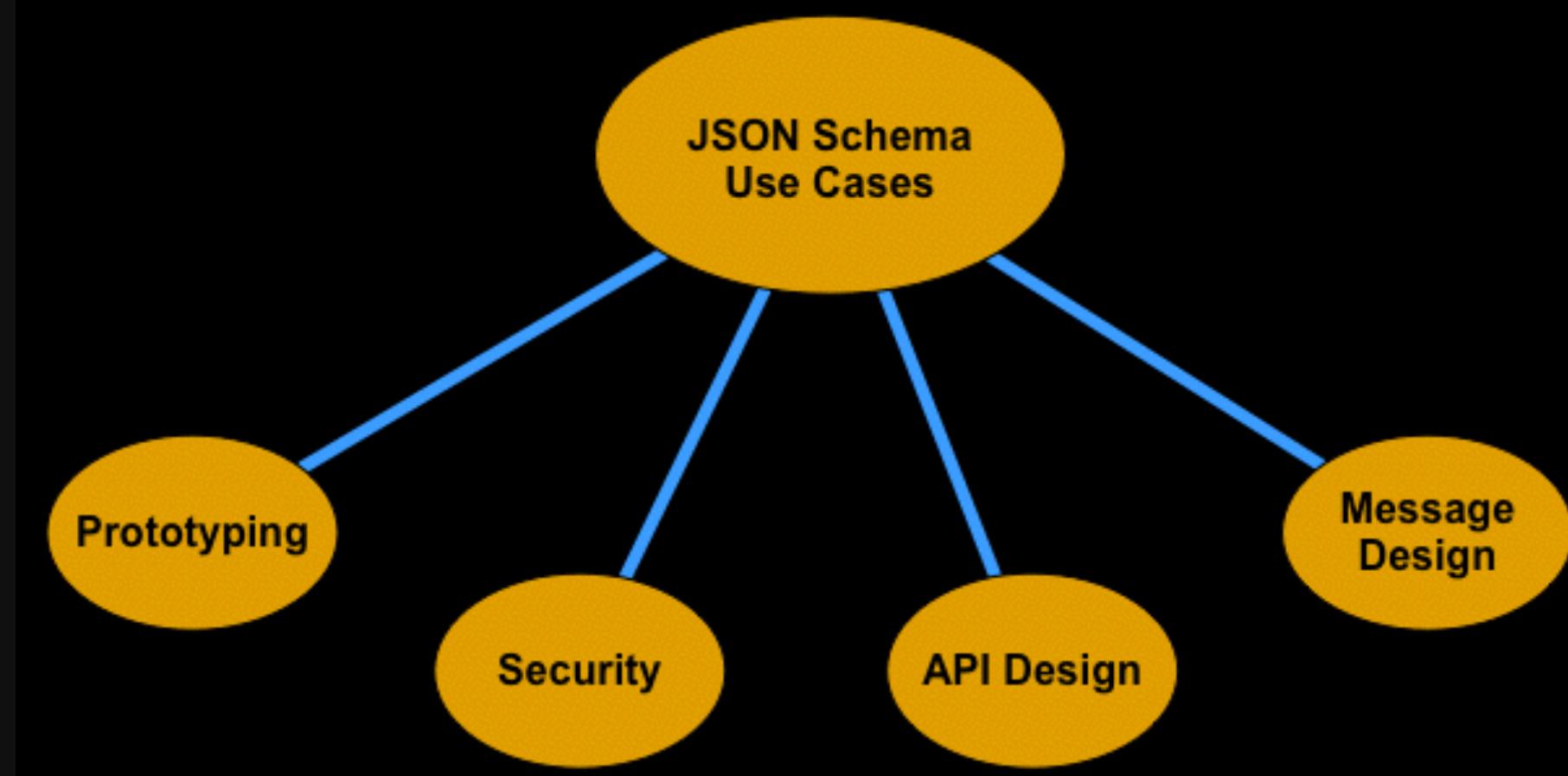
  - the [specification](#)
  - some [examples](#)

# JSON Schema on GitHub

The screenshot shows a GitHub repository page for `kriszyp/json-schema`. The page title is "JSON Schema specifications, reference schemas, and a CommonJS implementation <http://json-schema.org/>". Key statistics at the top include 67 commits, 2 branches, 2 releases, and 7 contributors. The current branch is master. A note says "Add CNAME, I think this is needed to have json-schema.org point to it." Below is a list of commits by user `kriszyp` from Aug 30, 2012, to the present.

Commit	Message	Date
<a href="#">draft-00</a>	Schema URIs are now namespace versioned.	5 years ago
<a href="#">draft-01</a>	Schema URIs are now namespace versioned.	5 years ago
<a href="#">draft-02</a>	Schema URIs are now namespace versioned.	5 years ago
<a href="#">draft-03</a>	Fix hyper-schema syntax error.	4 years ago
<a href="#">draft-04</a>	Core changes:	4 years ago
<a href="#">lib</a>	Clean out modifications to primitives	4 years ago
<a href="#">test</a>	Add vows-based unit tests.	4 years ago
<a href="#">CNAME</a>	Add CNAME, I think this is needed to have json-schema.org point to it.	3 years ago
<a href="#">README.md</a>	Updated docs	5 years ago
<a href="#">draft-zyp-json-schema-03.xml</a>	Merge <a href="http://github.com/garycourt/json-schema">http://github.com/garycourt/json-schema</a>	4 years ago
<a href="#">draft-zyp-json-schema-04.xml</a>	Merge <a href="git://github.com/garycourt/json-schema">git://github.com/garycourt/json-schema</a>	4 years ago
<a href="#">package.json</a>	bump version	4 years ago

# Where Does JSON Schema Fit?



# Who uses JSON Schema?



MuleSoft™



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

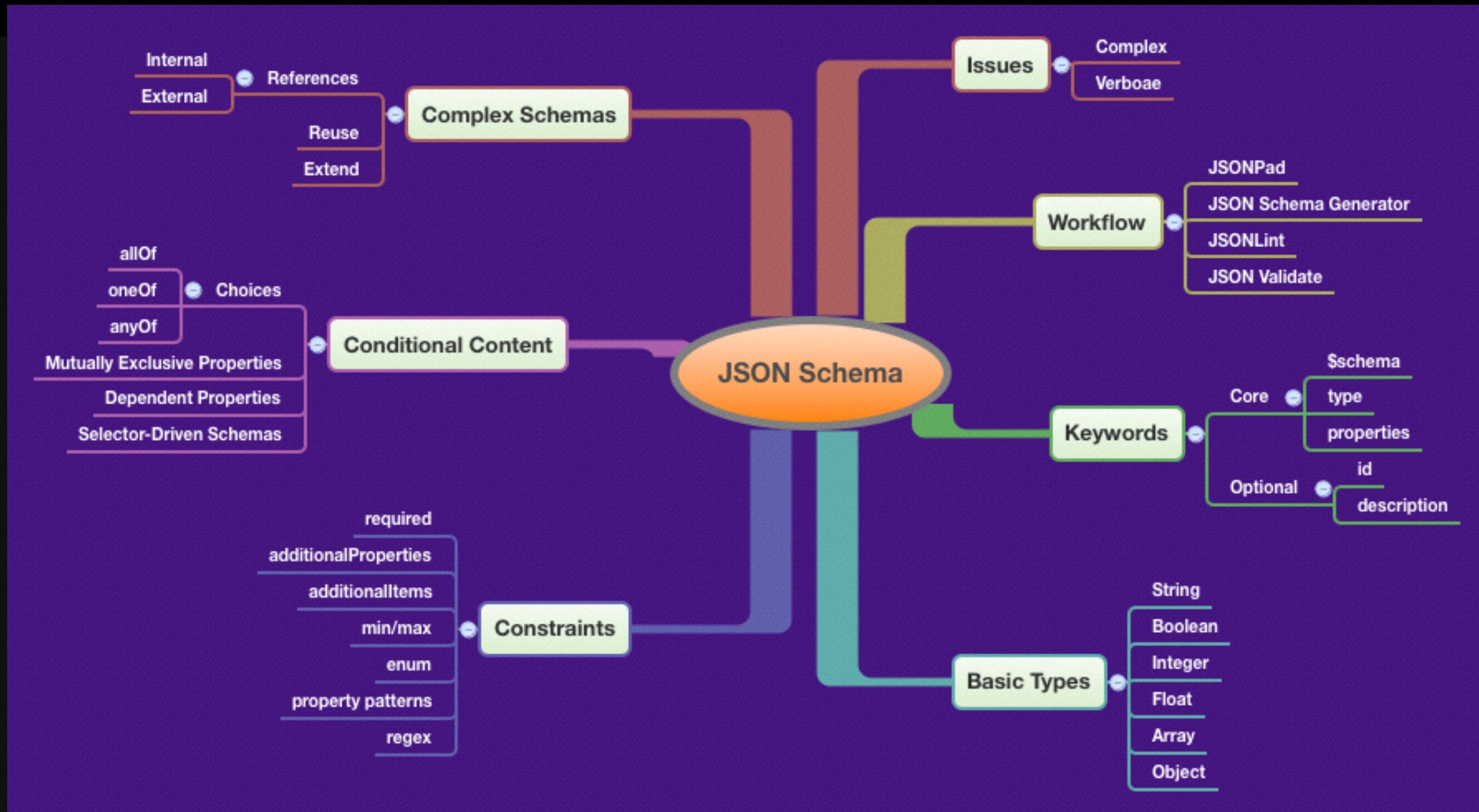
API Design with JSON Schema

3

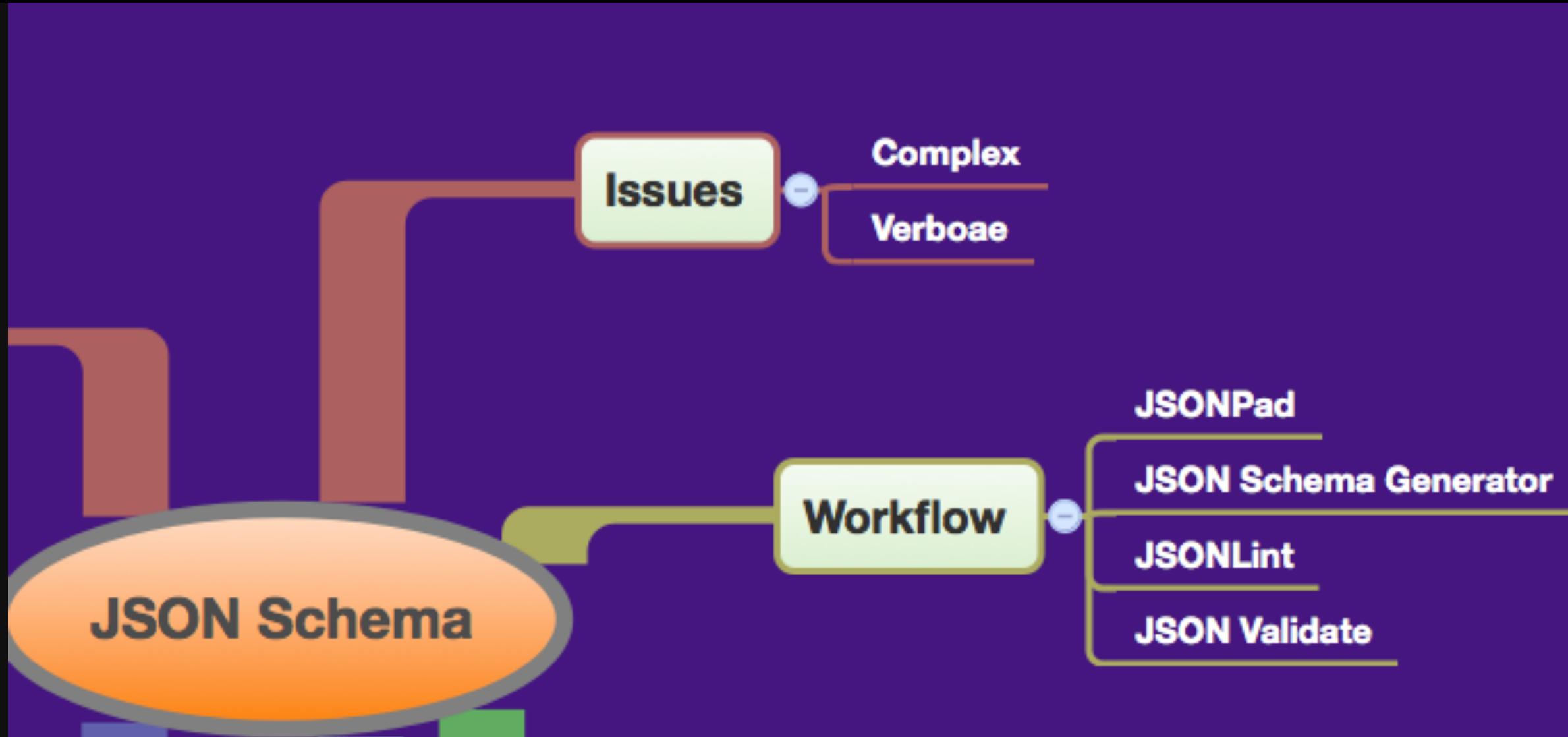
JSON Transform

6

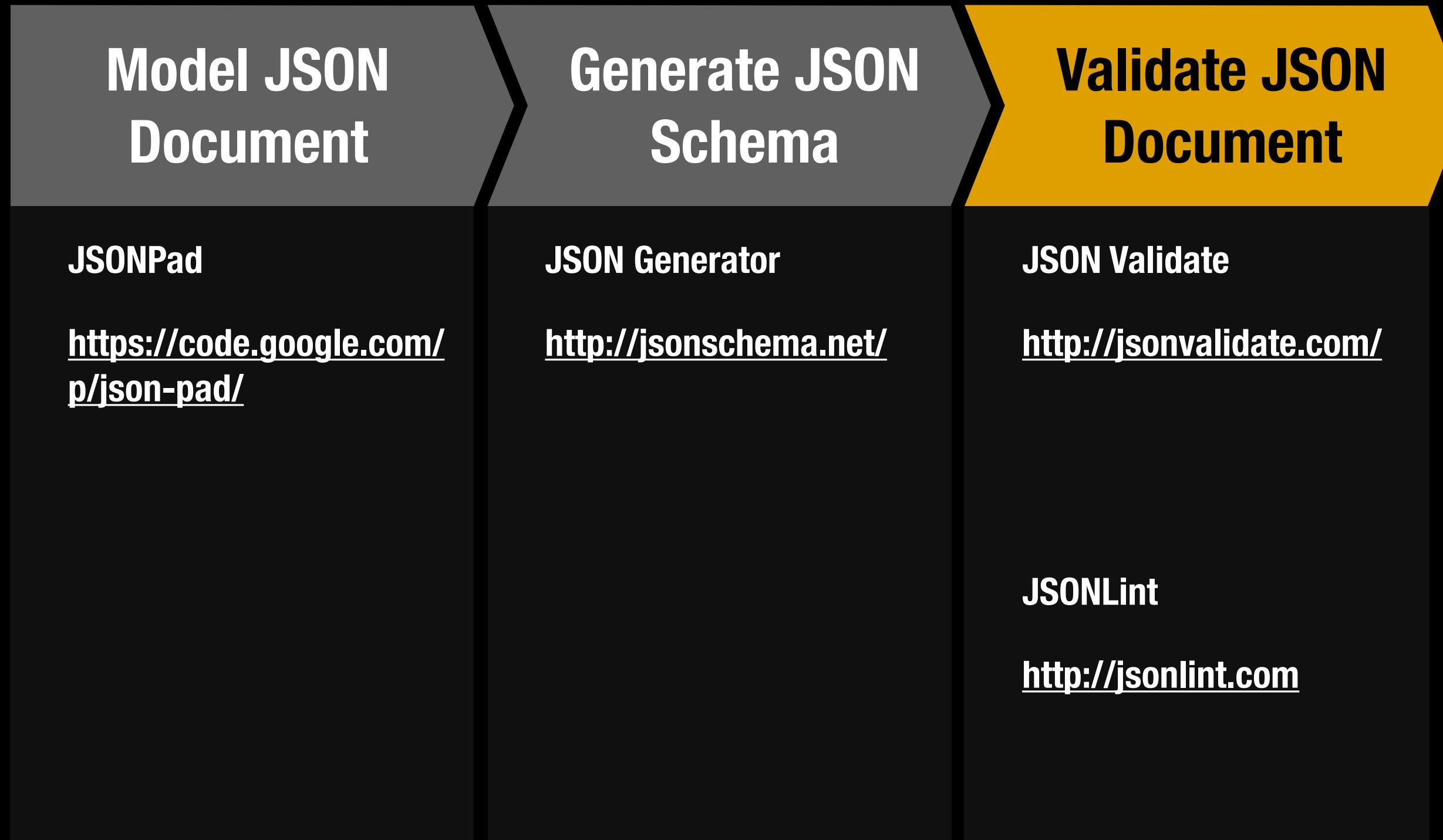
# Facets of JSON Schema



# JSON Schema - Issues and Workflow



# My JSON Schema Workflow



# JSONLint

The screenshot shows a web browser window for 'JSONLint - The JSON Validator'. The address bar displays 'jsonlint.com'. The page content includes a promotional message for 'JSONLint Pro', credits to Douglas Crockford and Zach Carter, and a code editor containing a JSON object. Below the code editor is a 'Validate' button. The 'Results' section at the bottom shows the text 'Valid JSON' in a green box.

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 a web browser. The title bar reads "JSON Validate" and the address bar shows "jsonvalidate.com". The main content area has two code editors: "JSON Schema" and "JSON Content". 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 }
```

The "JSON Content" editor contains the following JSON:

```
1 {
2   "email": "larsonrichard@ecratic.com",
3   "firstName": "Larson",
4   "lastName": "Richard"
5 }
```

Below the code editors are sections for "References" and "Results". The "References" section shows numbered links from 1 to 8. The "Results" section displays the word "Valid". At the bottom are buttons for "Validate" and "Reset all". A footer link "Learn more about Using JSON Schema" is followed by the "UJS" logo.

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 }
```

JSON Content

```
1 {
2   "email": "larsonrichard@ecratic.com",
3   "firstName": "Larson",
4   "lastName": "Richard"
5 }
```

References

1 2 3 4 5 6 7 8

Results

Valid

Validate Reset all

Learn more about Using JSON Schema UJS

# Beware of Wonky WiFi - Hedge Your Bets!



PDD - Presentation-Driven Development

# jsonlint

```
npm install -g jsonlint
```

```
jsonlint basic.json
```

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

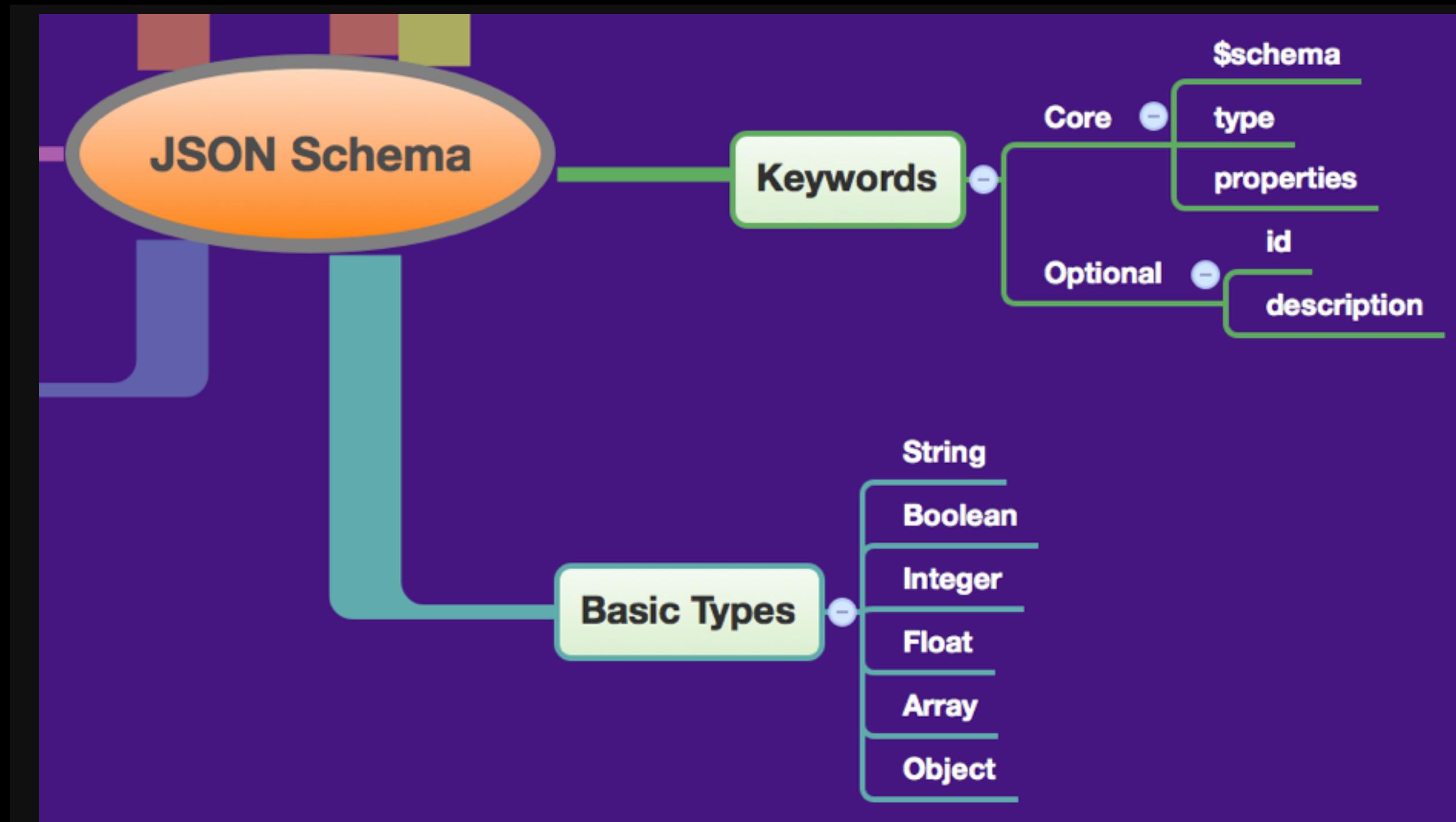
# ujs-jsonvalidate

```
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	<b>Specify JSON Schema version -</b> “\$schema”: “ <u><a href="http://json-schema.org/draft-04/schema#">http://json-schema.org/draft-04/schema#</a></u> ”
type	<b>The data type -</b> “type”: “string”
properties	<b>The fields for an object</b>

# Optional Keywords

Keyword	Definition
id	<p>(1): Path to field (2): URI to Schema</p>
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

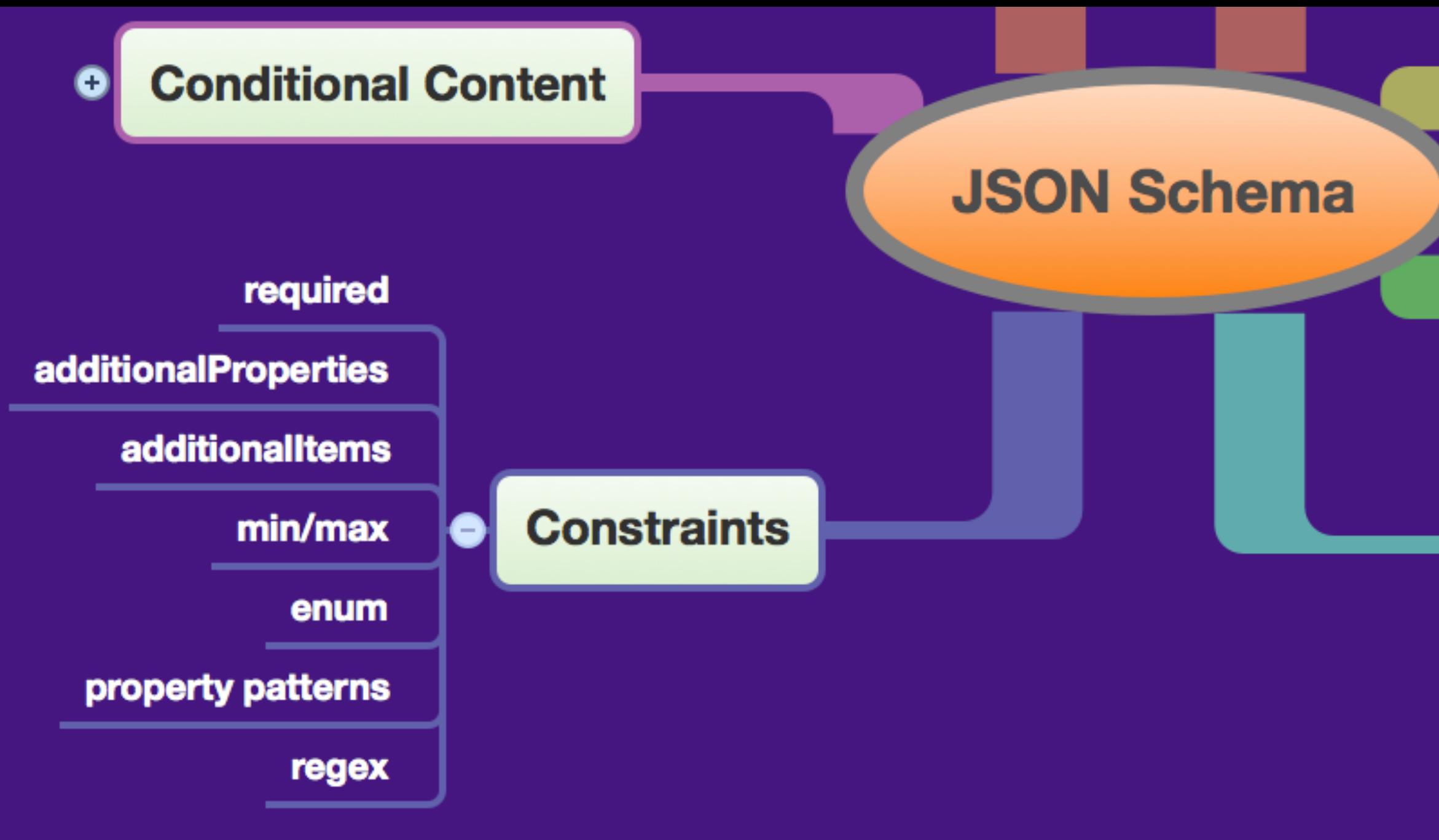
```
2  {
3    "speaker": {
4      "firstName": "Larson",
5      "lastName": "Richard",
6      "email": "larsonrichard@ecratic.com",
7      "postedSlides": true,
8      "rating": 4.1,
9      "tags": [
10        "JavaScript", "AngularJS", "Yeoman"
11      ]
12    }
13 }
```

# 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 web application interface. The main area is titled "REGULAR EXPRESSION" with a placeholder "insert your regular expression here". To the right, under "EXPLANATION", it says "An explanation of your regex will be automatically generated as you type." Below that is "MATCH INFORMATION" with the note "Detailed match information will be displayed here automatically." At the bottom is a "QUICK REFERENCE" section with tabs for "FULL REFERENCE" (selected) and "MOST USED TOKENS". The "most used tokens" tab lists several regex components with their descriptions:

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 sidebar, there are sections for "SAVE & S...", "FLAVOR" (set to PCRE), "TEST STRING" (placeholder "insert your test string here"), "TOOLS" (with icons for copy/paste, file operations, and search), and "SUBSTITUTION" (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. To the right of the sidebar is a large callout box containing text about RegexBuddy and a screenshot of the software interface. The main content area features a section titled "Welcome to Regular-Expressions.info" with a sub-section "The Premier website about Regular Expressions". It includes a paragraph about regular expressions and examples of how they can be used. At the bottom, there's a footer with a book cover thumbnail for "Regular Expressions Cookbook" and a Runscope monitoring badge.

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

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

# Regexpr

The screenshot shows the RegExr v2.0 web application interface. On the left, there is a sidebar titled "Escaped characters" with a list of escape sequences:

Character	Escape Sequence
octal escape	\000
hexadecimal escape	\xFF
unicode escape	\uFFFF
control character escape	\cI
tab	\t
line feed	\n
vertical tab	\v
form feed	\f

Below this, two paragraphs provide information about regular expression escaping:

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.

The main workspace is titled "Expression" and contains the regex pattern `/([A-Z])\w+/g`. A blue box indicates "16 matches". The "Text" area contains the string "Welcome to RegExr v2.0 by gskinner.com!". Below the text area is a descriptive message:

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.

The "Text" area also lists a sample of test strings:

```
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://regexpr.com/foo.html?q=bar
```

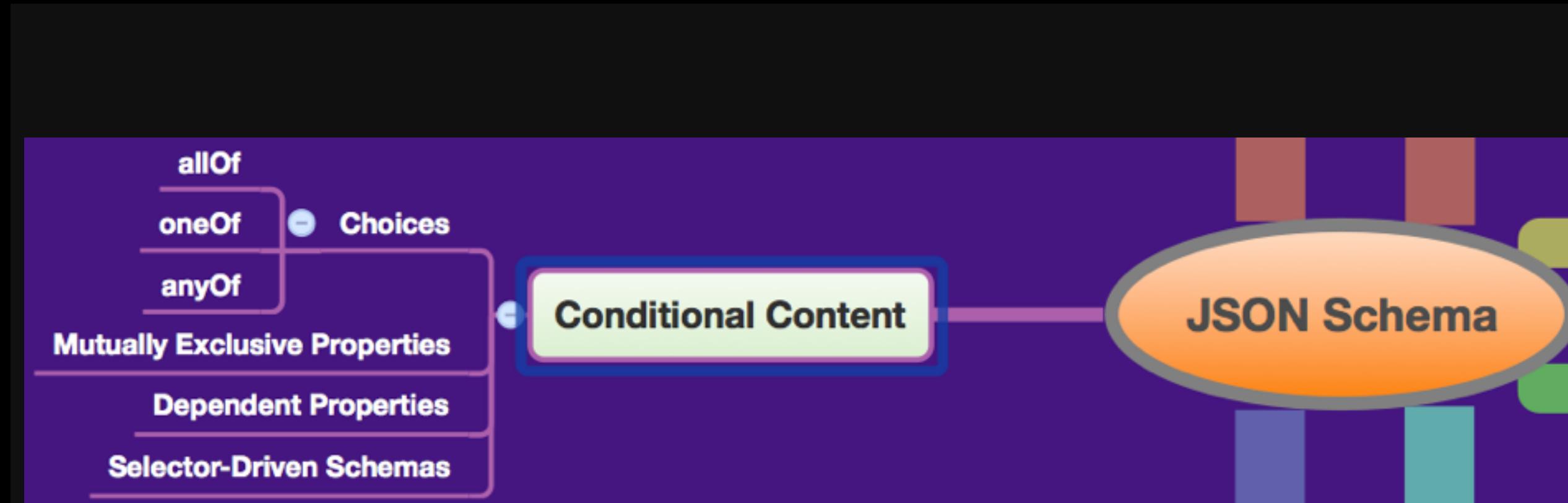
At the bottom, there is a "Substitution" section with a plus sign icon.

# 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**

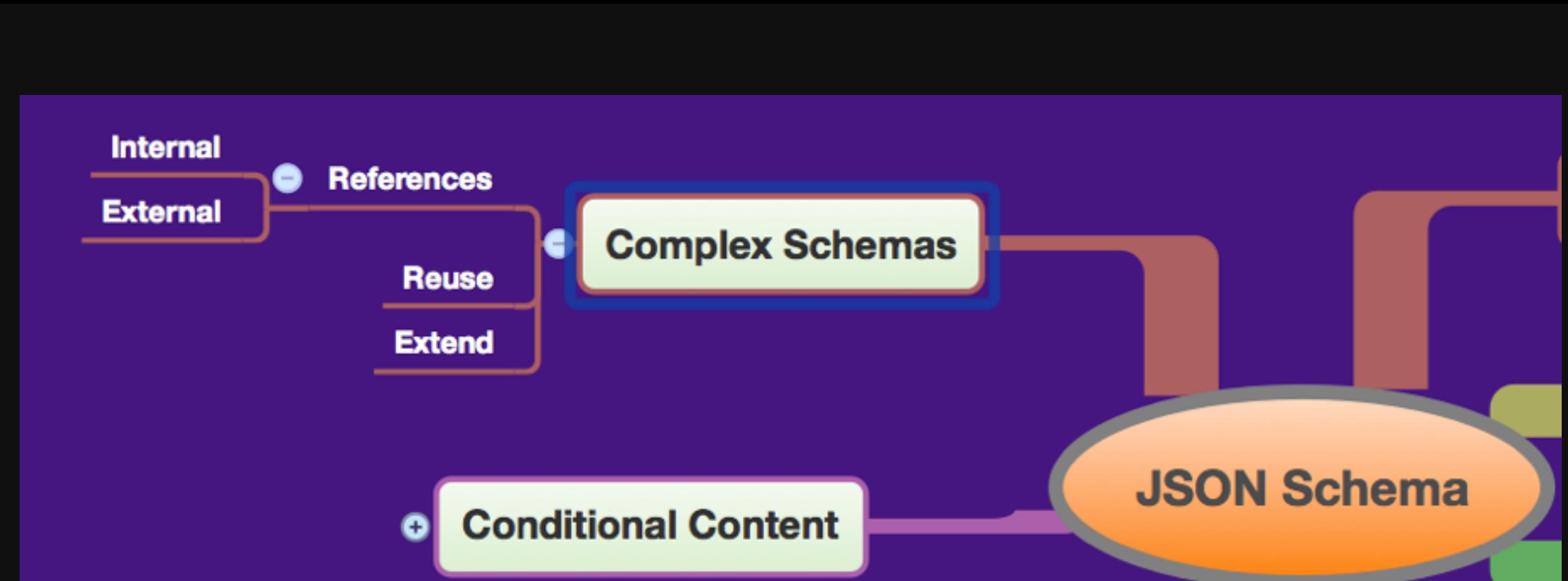
# anyOf - JSON Schema

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

**JSON Search & Transform Overview**

**4**

**Core JSON Schema**

**2**

**JSON Search**

**5**

**API Design with JSON Schema**

**3**

**JSON Transform**

**6**

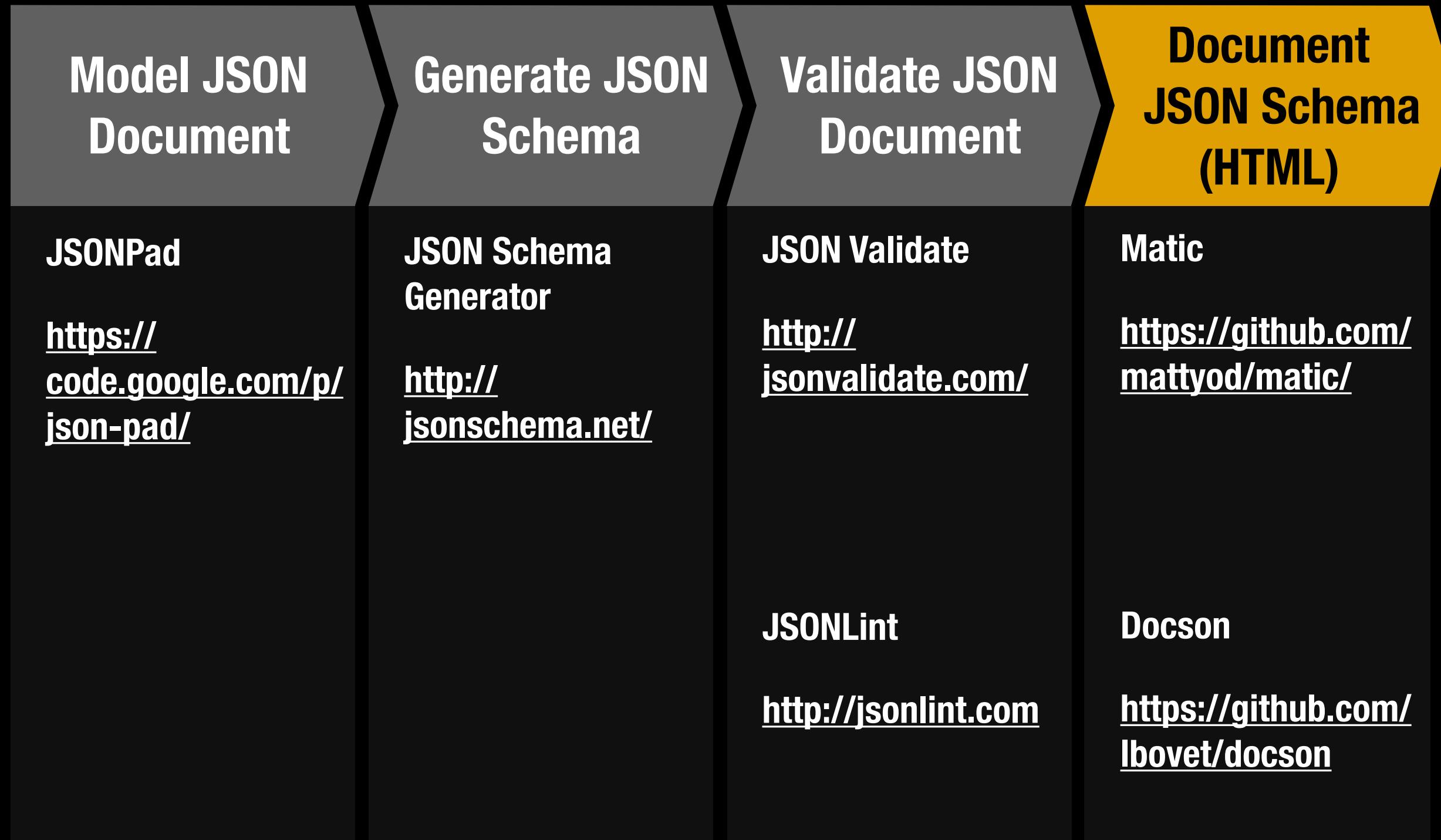
# Real World Use Case

**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



# JSONPad Demo

The screenshot shows the JSONPad application interface. The main window title is "JSONpad". The menu bar includes "File", "Edit", "Help", "Tree", "Format", "Clipboard", "Tools", and "Examples". The "Tools" menu is currently active, indicated by a yellow outline. The toolbar contains icons for download (green arrow), upload (blue arrow), copy (ctrl+c), paste (ctrl+v), refresh (refresh), and a pencil for edit.

The code editor displays the following JSON code:

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

The tree view on the left shows the JSON structure with nodes: about, email, firstName, lastName, tags, and company. The "tags" node is expanded, showing its three child items: "JavaScript", "REST", and "JSON".

The right panel is titled "Edit" and is currently empty.

# JSON Schema Generator Demo

The screenshot shows a web-based JSON Schema Generator tool. On the left, a URL input field contains `http://jsonsatwork.org`. Below it, a JSON code editor displays a sample JSON object:

```
{  
    "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." Below this, there are three buttons: "Generate Schema" (highlighted in blue), "Reset", and "Metadata". Under "Metadata", there is an unchecked checkbox for "Include metadata keywords".

On the right, the "Code View" tab is selected, showing the generated JSON Schema:

```
{
  "$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 browser's address bar shows `jsonschema.net/#/`. The top right corner of the window has a "Tom" icon.

# JSON Validate Demo

The screenshot shows a web-based JSON validation tool with the following sections:

- JSON Schema:** A code editor containing a JSON schema definition. The schema includes properties like "about", "email", "firstName", "lastName", "tags", and "company".

```
22    "about": "Fred Smith is the CTO of Full Ventures, where he ...",
23    "email": "fred.smith@fullventures.com",
24    "firstName": "Fred",
25    "lastName": "Smith",
26    "tags": [
27      "JavaScript",
28      "REST",
29      "JSON"
30  ],
31  "company": {
32    "id": "company",
33    "type": "string"
34  },
35  "additionalProperties": false,
36  "required": [
37    "about",
38    "email",
39    "firstName",
40    "lastName",
41    "tags",
42    "company"
43  ]
```
- JSON Content:** A code editor containing a JSON object to be validated.

```
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 list of numbered items from 1 to 14.
- Results:** A panel displaying the validation status: "Valid".
- Buttons:** "Validate" and "Reset all".
- Links:** "Learn more about Using JSON Schema" with a "UJS" icon.

# 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 "swagger" logo and a "http://lbovet.github.io/swagger-ui/dist/api/index.json" link. There are also "api\_key" and "Explore" buttons.

## Swagger Example with Tyson and Docson Integration

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

**user : Operations about user** Show/Hide | List Operations | Expand Operations | Raw

**pet : Operations about pets** Show/Hide | List Operations | Expand Operations | Raw

**GET /pet/{petId}** Find pet by ID

**Implementation Notes**  
Returns a pet based on ID

**Response Class**

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

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

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

# Swagger Petstore - Regular

The screenshot shows the Swagger UI interface for the Petstore API. The URL in the browser is `petstore.swagger.io/#/pet/getPetById`. The main section displays the `GET /pet/{petId}` endpoint. The response schema is shown as a JSON object:

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

The response content type is set to `application/xml`. A table lists the parameters for this endpoint:

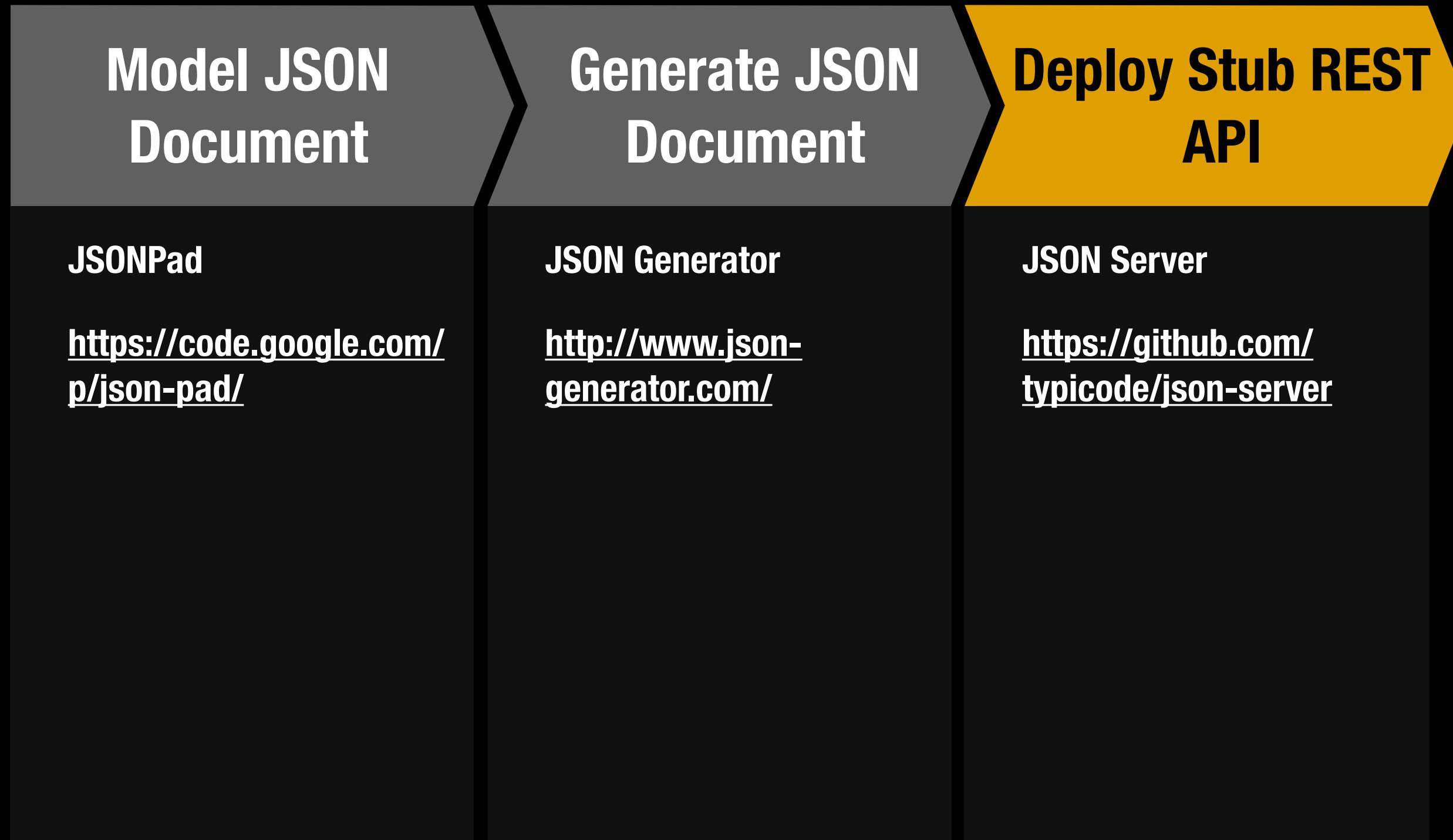
Parameter	Value	Description	Parameter Type	Data Type
<b>petId</b>	(required)	ID of pet to return	path	long

Below the table, the **Response Messages** section shows two entries:

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

A **Try it out!** button is located at the bottom left of the interface.

# My JSON Stub API Workflow



# JSON Generator Demo

The screenshot shows a web browser window for the "JSON Generator - Tool for API Testing" at [www.json-generator.com](http://www.json-generator.com). The main content area is titled "JSON GENERATOR". A code editor displays the following JSON template:

```
1 // Template for http://www.json-generator.com/
2
3 [
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 ]
```

A large watermark with the text "Click 'Generate' and wait for the magic" is overlaid on the right side of the page. At the bottom, there is footer text: "Created by Vazha Omanashvili, Sponsored by Runscope API Tools" and social sharing links for Twitter, Facebook, and LinkedIn.

# 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

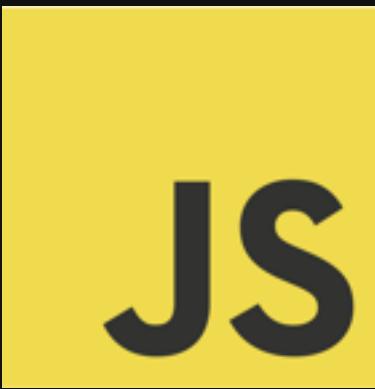
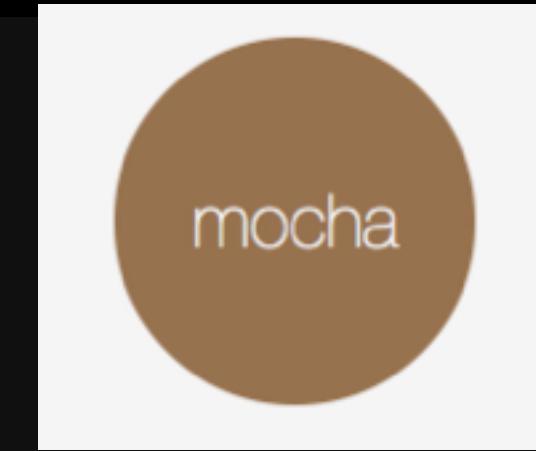
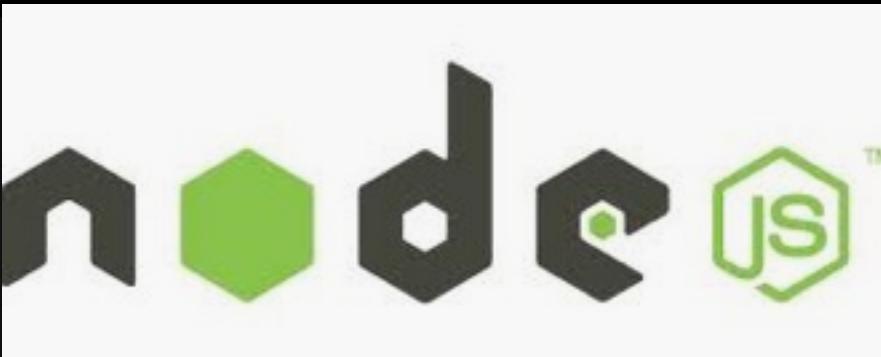
API Design with JSON Schema

3

JSON Transform

6

# Our Client Stack



# Firebase Open Data Set

The screenshot shows a web browser displaying the Firebase Open Data Sets documentation at <https://www.firebaseio.com/docs/open-data/>. The page features a dark sidebar on the left with navigation links for various platforms and security rules. The main content area has a blue header with the text "FIREBASE" and "Open Data Sets". It describes Firebase Open Data Sets as live-updated Firebases from various sources. 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.

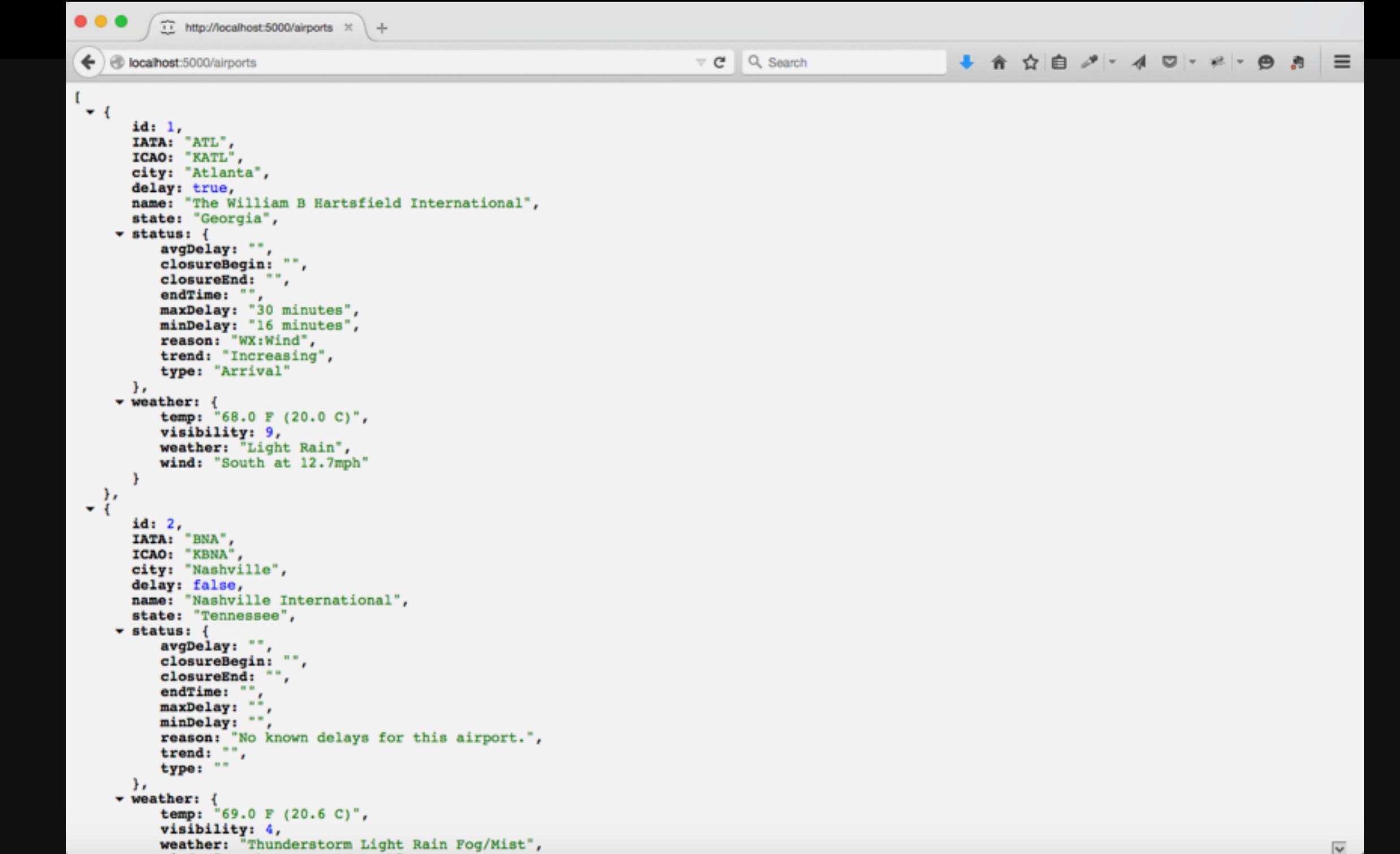
Data Set	Description
<a href="#">Airport Delays</a>	Get the latest airport delay and status updates in realtime.
<a href="#">Cryptocurrencies</a>	Get the latest USD/BTC and USD/LTC exchange rates in realtime.
<a href="#">Earthquakes</a>	Information on earthquakes anywhere on Earth in realtime.
<a href="#">Parking</a>	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>

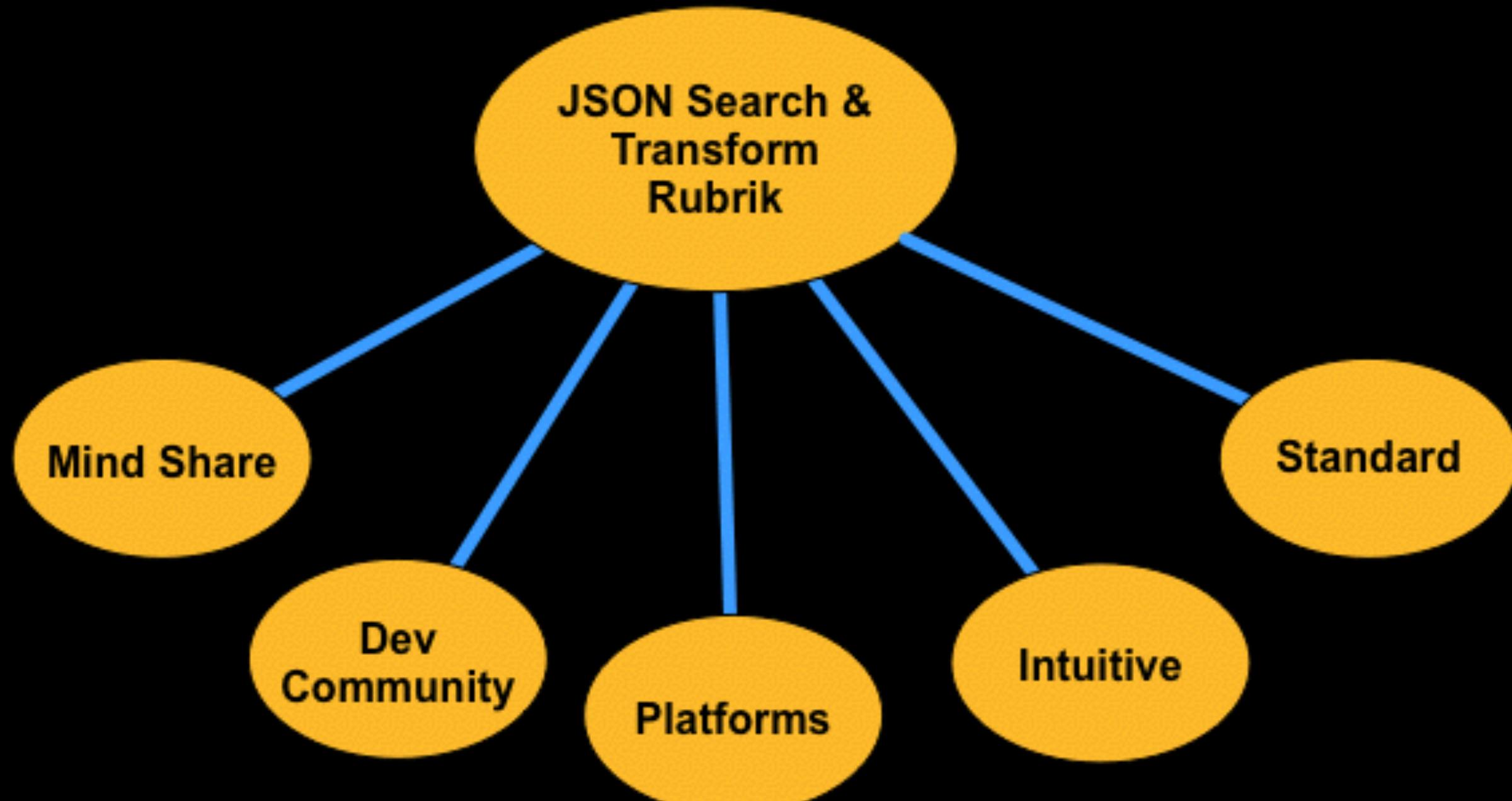
# Airports Stub Service



The screenshot shows a web browser window displaying a JSON response from the URL `http://localhost:5000/airports`. The JSON data lists two airports, each with their details and current status.

```
[  
  {  
    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",  
      wind: "North at 10.5mph"  
    }  
  }]
```

# JSON Search & Transform Rubrik



# Where Are We?

JSON Schema Overview

1

JSON Search & Transform Overview

4

Core JSON Schema

2

JSON Search

5

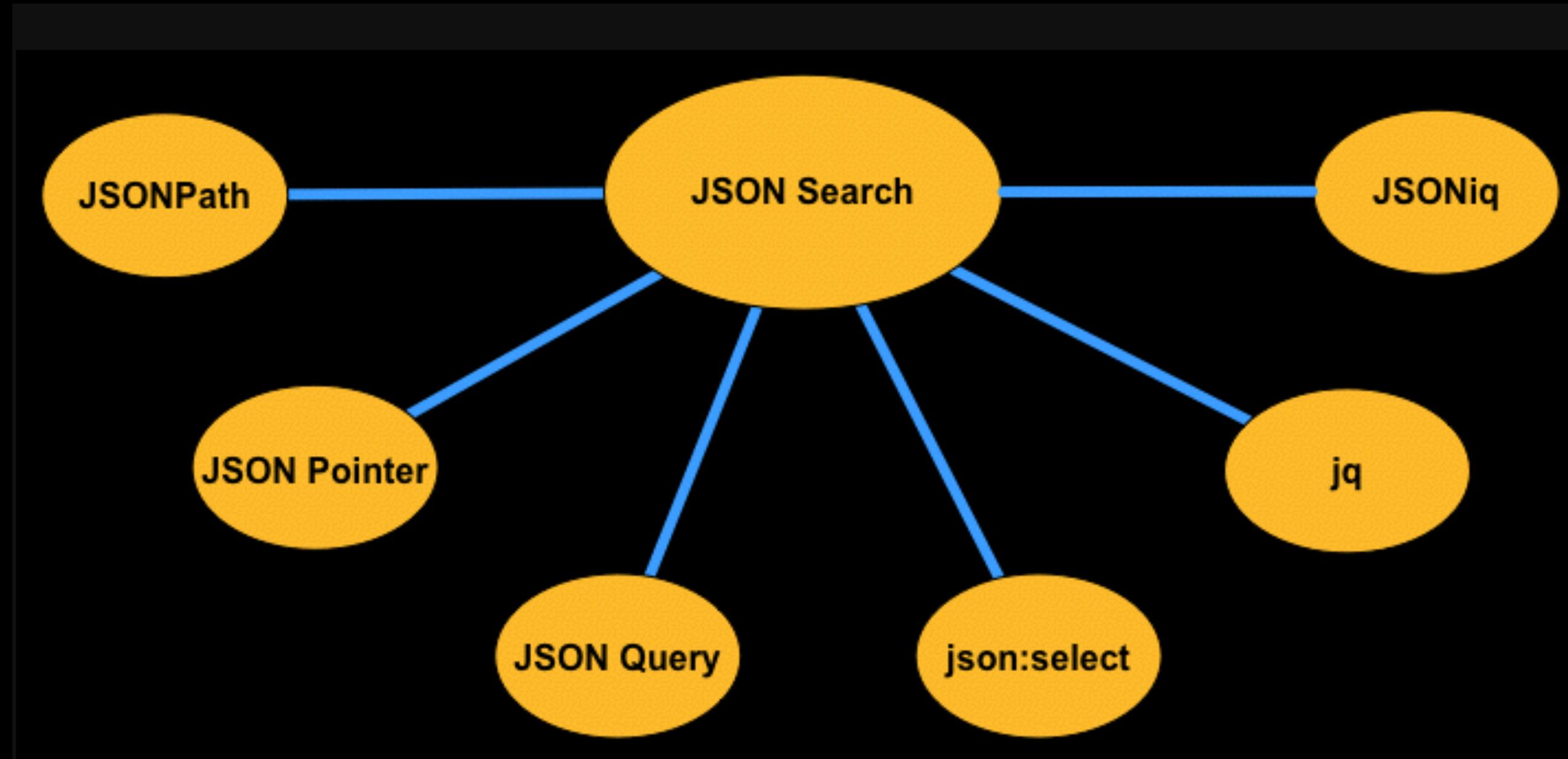
API Design with JSON Schema

3

JSON Transform

6

# JSON Search Tools



# JSONPath

The screenshot shows a web browser window with the title "JSONPath - XPath for JSON". The URL in the address bar is "goessner.net/articles/JsonPath/". The page content is an article titled "# JSONPath - XPath for JSON" posted on "2007-02-21 | e1". The article discusses the advantages of XML tools like XPath and introduces JSONPath as a similar tool for JSON. It lists benefits such as interactive data extraction and server-side bandwidth reduction. The sidebar includes search, content navigation, and comments sections.

<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

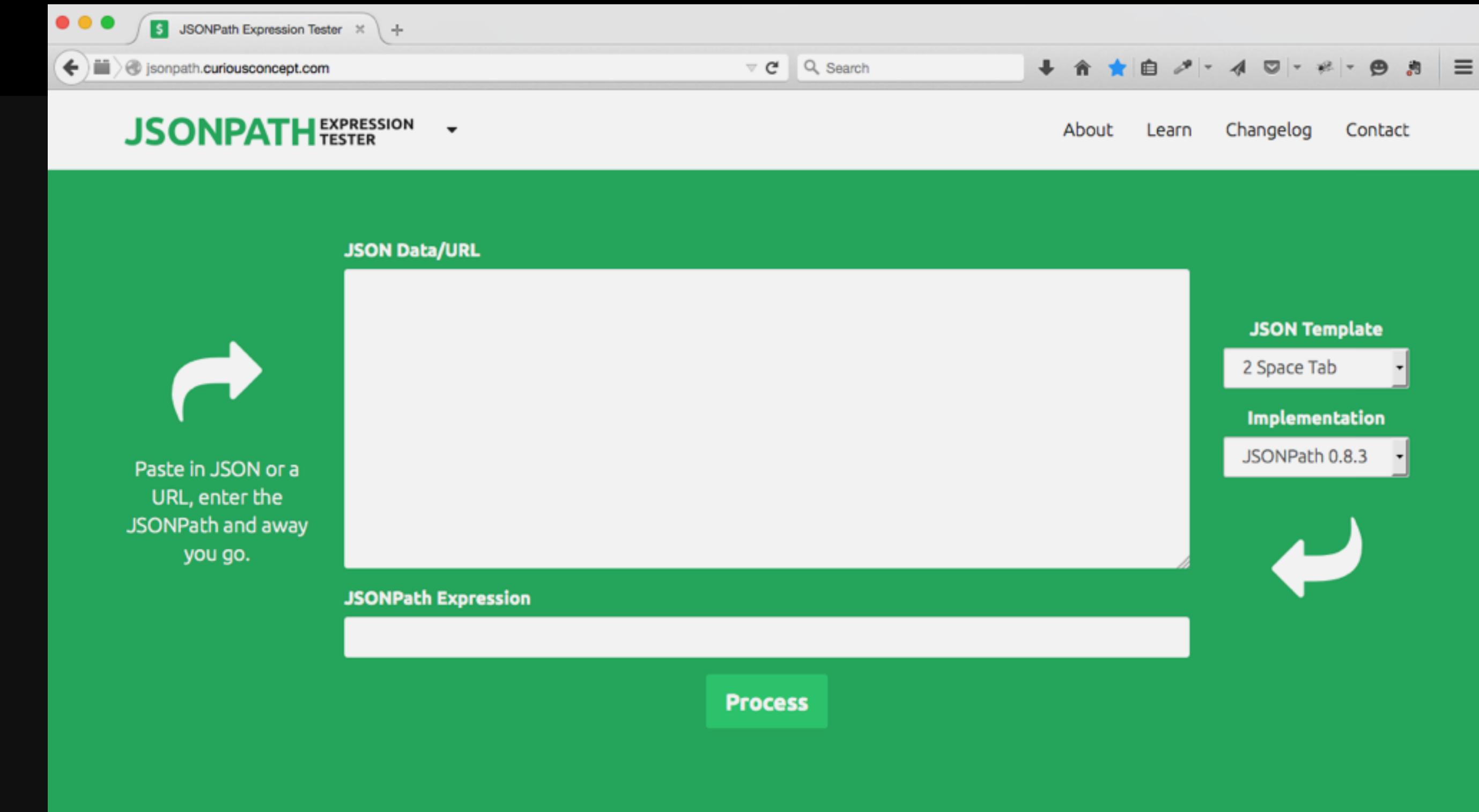
» Inhalt ..  
Home  
Lehre  
Dynamik  
Articles  
DOM Events  
Wiky  
2D Vectors  
Slideous  
JsonT  
JSONPath  
SVG  
Download  
Admin  
Info

» comments ..  
Exercise 09

# JSONPath Syntax

<b>XPath</b>	<b>JSONPath</b>	<b>Result</b>
/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 the homepage of the JSONPath Expression Tester. The page has a dark green header bar with the title "JSONPath Expression Tester" and a navigation menu with links for "About", "Learn", "Changelog", and "Contact". Below the header is a large green main area containing several input fields and controls. 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 field labeled "JSON Data/URL". At the bottom, there's a "JSONPath Expression" input field and a green "Process" button. To the right, there are two dropdown menus: "JSON Template" set to "2 Space Tab" and "Implementation" set to "JSONPath 0.8.3". A large white arrow points left from the implementation section.

JSONPATH EXPRESSION TESTER

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 Online Evaluator

The screenshot shows a web browser window titled "JSONPath Online Evaluator" with the URL "ashphy.com/JSONPathOnlineEvaluator/". The page content includes the title "JSONPath Online Evaluator", author information ("Author: Kazuki Hamasaki [ashphy@ashphy.com]"), and a note about the evaluator ("This evaluator uses JSONPath - XPath for JSON"). The interface is divided into two main sections: "Inputs" and "Evaluation Results".

**Inputs**

JSONPath Syntax  
\$.phoneNumbers[:1].type

Example '\$.phoneNumbers[\*].type' See also [JSONPath expressions](#)

JSON

```
{  
  "firstName": "John",  
  "lastName" : "doe",  
  "age"    : 26,  
  "address" :  
  {  
    "streetAddress": "naist street",  
    "city"        : "Nara",  
    "postalCode"  : "630-0192"  
  },  
  "phoneNumbers":  
  [  
    {  
      "type"  : "iPhone",  
      "number": "0123-4567-8888"  
    },  
    {  
      "type"  : "Android",  
      "number": "0123-4567-8889"  
    }  
  ]}
```

**Evaluation Results**

```
'0' => "iPhone"
```

# 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". Below the address bar, there are several blue links: "[Docs]", "[txt|pdf]", "[draft-ietf-appsaw...]", "[Diff1]", "[Diff2]", and "[Errata]". The main content area starts with the title "PROPOSED STANDARD" and the subtitle "Errata Exist". It lists the following information:

Internet Engineering Task Force (IETF)	P. Bryan, Ed.
Request for Comments: 6901	Salesforce.com
Category: Standards Track	K. Zyp
ISSN: 2070-1721	SitePen (USA)
	M. Nottingham, Ed.
	Akamai
	April 2013

Below this, the section "JavaScript Object Notation (JSON) Pointer" is titled "Abstract". The abstract states: "JSON Pointer defines a string syntax for identifying a specific value within a JavaScript Object Notation (JSON) document." The section "Status of This Memo" is titled "This is an Internet Standards Track document." It explains that the document is a product of the IETF, represents the consensus of the community, has received public review, and been approved by the IESG. It also provides a link to "Section 2 of RFC 5741". The section "Copyright Notice" contains the 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~n"	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

The screenshot shows a web browser window with the title bar "JSONQuery: Data Querying ...". The address bar contains the URL "https://www.sitepen.com/blog/2008/07/16/jsonquery-data-querying-beyond-jsonpath/". The page itself has a teal header with the sitepen logo and navigation links for Services, Blog, About Us, Contact, and Login. The main content area features the title "JSONQuery: Data Querying Beyond JSONPath". Below the title is a yellow box containing a notice about reading a newer post on RQL. The main text discusses the addition of JSONQuery to Dojo 1.2, its intended improvements over JSONPath, and its comprehensive set of querying tools. It also mentions safe evaluation and result-based evaluation. A bio for Kris Zyp follows, along with a sidebar for featured articles and a hiring notice.

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 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 titled "JSONSelect" displaying the homepage of jsonselect.org. The page features a large title "json:select()" and a subtitle "CSS-like selectors for JSON.". It includes a brief description of the tool, a selector example, and a call-to-action button.

**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"`

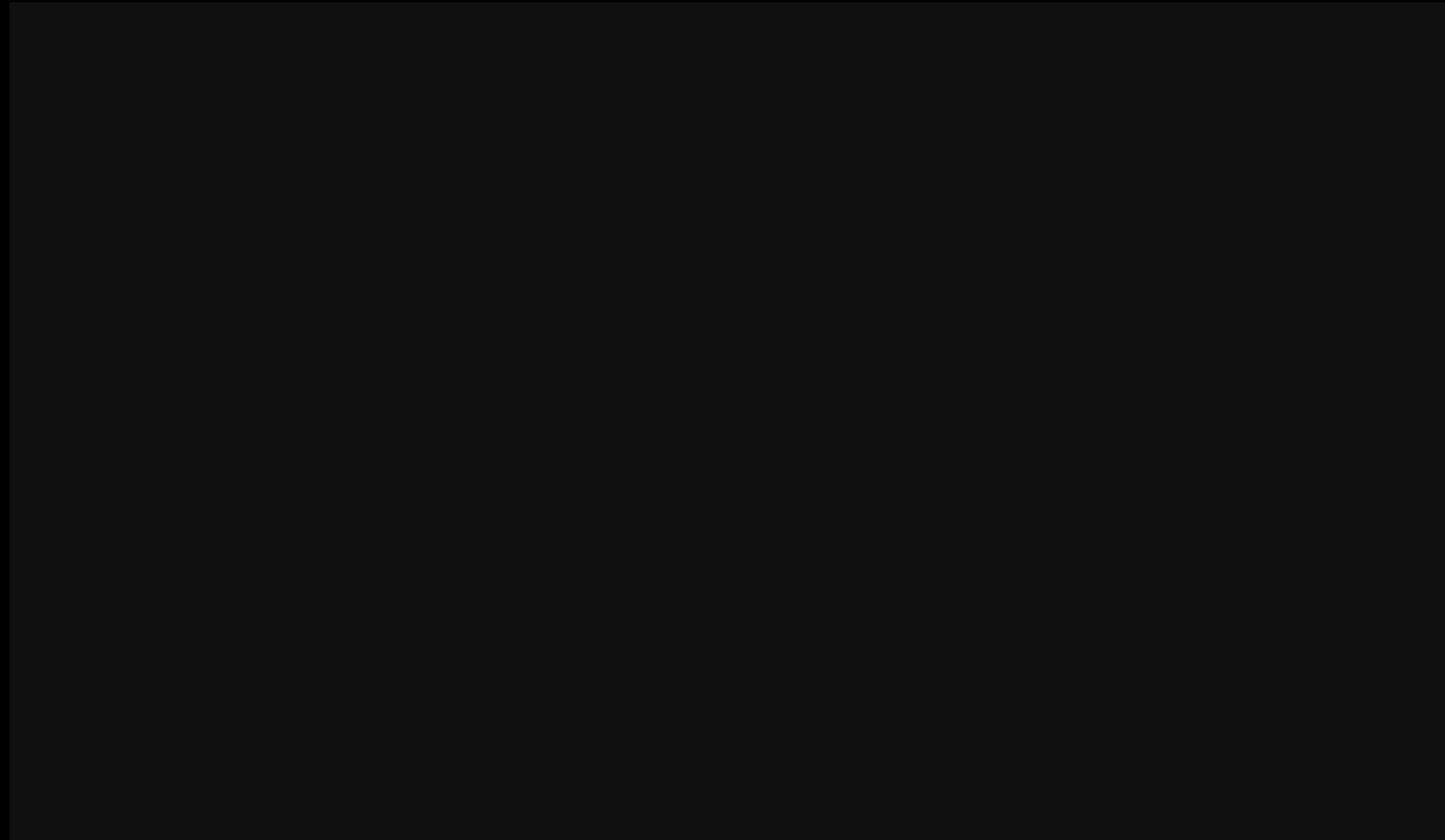
```
{
  "author": {
    "name": {
      "first": "Lloyd",
      "last": "Hilaiel"
    },
    "drinkPref": [
      "whiskey",
      "beer",
      "wine"
    ],
    "thing": "JSONSelect site",
    "license": "(cc) BY-SA"
  }
}
```

<https://github.com/lloyd/JSONSelect>

# json:select Expression Tester

The screenshot shows a web browser window for the "JSONSelect Expression Tester" application. The URL in the address bar is `jsonselect.curiousconcept.com/#`. The page has a teal header with the title "JSONSELECT EXPRESSION TESTER". In the top right corner, there are links for "About", "Learn", "Changelog", and "Contact". Below the header, there's a large teal section containing two main input fields: "JSON Data/URL" and "JSONTemplate". The "JSON Data/URL" field is a large white area with a white arrow icon pointing towards it, and the placeholder text "Paste in JSON or a URL, enter the JSONSelect and away you go.". The "JSONTemplate" field is a dropdown menu set to "3 Space Tab". Below these fields is a "JSONSelect Expression" input field, which is currently empty. At the bottom center of the teal section is a teal button labeled "Process".

# json:select Demo



# json:select Test

```
2  var expect = require('chai').expect;
3  var request = require('request');
4  var jsonSelect = require('JSONSelect');
5
6  describe('JSONSelect', 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\nJSONSelect Test');
18          var obj = JSON.parse(res.body);
19          console.log('\n\n1st & 2nd Object weather: ');
20          console.log(jsonSelect.match('.status ~ .weather', obj).slice(0,2));
21          console.log('\n\nAll Airport Codes: ');
22          console.log(jsonSelect.match('.IATA', obj));
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

# jq

The screenshot shows a web browser window displaying the official jq website at [stedolan.github.io/jq/](https://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, the text 'jq is a lightweight and flexible command-line JSON processor.' is displayed. Below this are two buttons: 'Download jq-1.4-2-g15c4a7f-dirty' and 'Try online!'. The main content area is divided into three columns. The first column contains text about jq being like `sed` for JSON. The second column discusses its implementation in C and zero dependencies. The third column highlights its ability to transform data. At the bottom, there's a link to the tutorial and manual, and a 'News' section with a single entry from June 2014.

jq

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

# jq play

The screenshot shows the jq play playground interface. At the top, there's a browser header with tabs, a search bar, and various icons. Below it is the jq play logo and a subtitle "A playground for jq 1.4". The main area has two large code editors: one for "Filter" and one for "Result". Both editors show the number "1" at the top left. To the right of the Result editor are several checkboxes for output options: "Compact Output", "Null Input", "Raw Input", "Raw Output", and "Slurp". Below the editors is a "JSON" section containing a single digit "1". At the bottom is a "Cheatsheet" table with examples of jq filters:

Cheatsheet			
Click on the icons (☰) in the table below to see examples.			
•	unchanged input	☰	feed input into multiple filters
.foo, .foo.bar, .foo?	value at key	☰	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

The screenshot shows the official website for JSONiq, a query language. The header includes a navigation bar with links for "The JSONiq Language", "JSONiq Book", "JSONiq Extension to XQuery", "Implementations", "Live Sandbox", and "Mailing-List". The main title "The JSON Query Language" is displayed prominently. Below the title are three main sections: "Decades of Lessons Learnt" (with a "Hello my name is JSONiq" graphic), "Complex Processing" (with two interlocking gears), and "The SQL of NoSQL" (with a cloud icon containing a database cylinder). Each section contains descriptive text.

# { "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

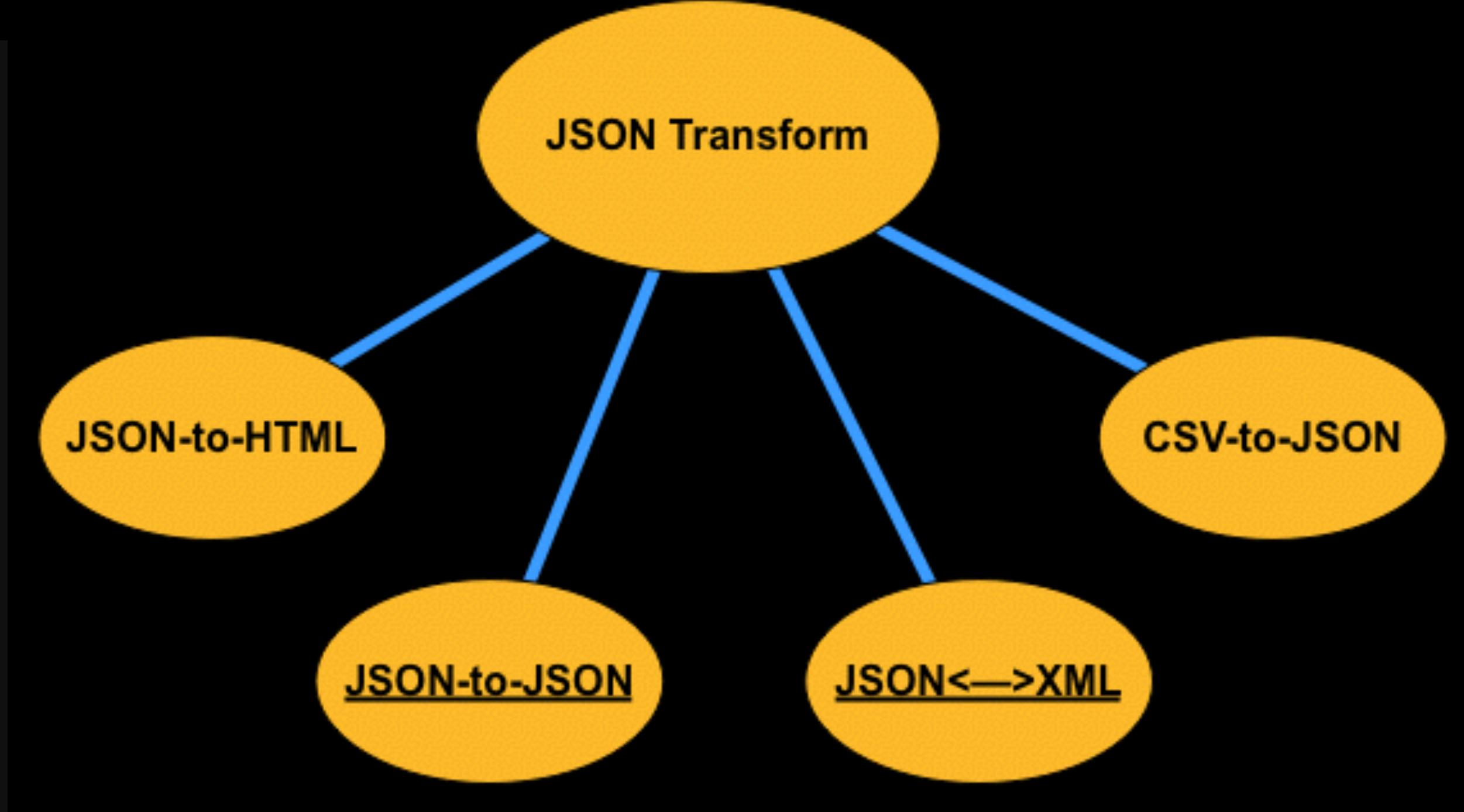
API Design with JSON Schema

3

JSON Transform

6

# JSON Transform



# JSON-T

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 articles:

**# 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

The screenshot shows the GitHub repository page for 'amida-tech/jsonapter'. The repository name is 'jsonapter' and it is described as 'Template Based JSON To JSON Transformation'. It has 2 commits, 1 branch, 0 releases, and 2 contributors. The master branch is selected. The repository contains files like lib, test, .gitignore, .jsbeautifyrc, .jshintrc, .travis.yml, LICENSE, README.md, RELEASENOTES.md, gruntfile.js, index.js, and package.json. The repository has 4 watches, 1 star, and 0 forks. The right sidebar includes links for Code, Issues (0), Pull requests (0), Pulse, Graphs, and clone options via HTTPS or Subversion.

Template Based JSON To JSON Transformation

2 commits · 1 branch · 0 releases · 2 contributors

branch: master / +

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/amida-tech/jsonapter>

You can clone with [HTTPS](#) or [Subversion](#).

Clone In Desktop · Download ZIP

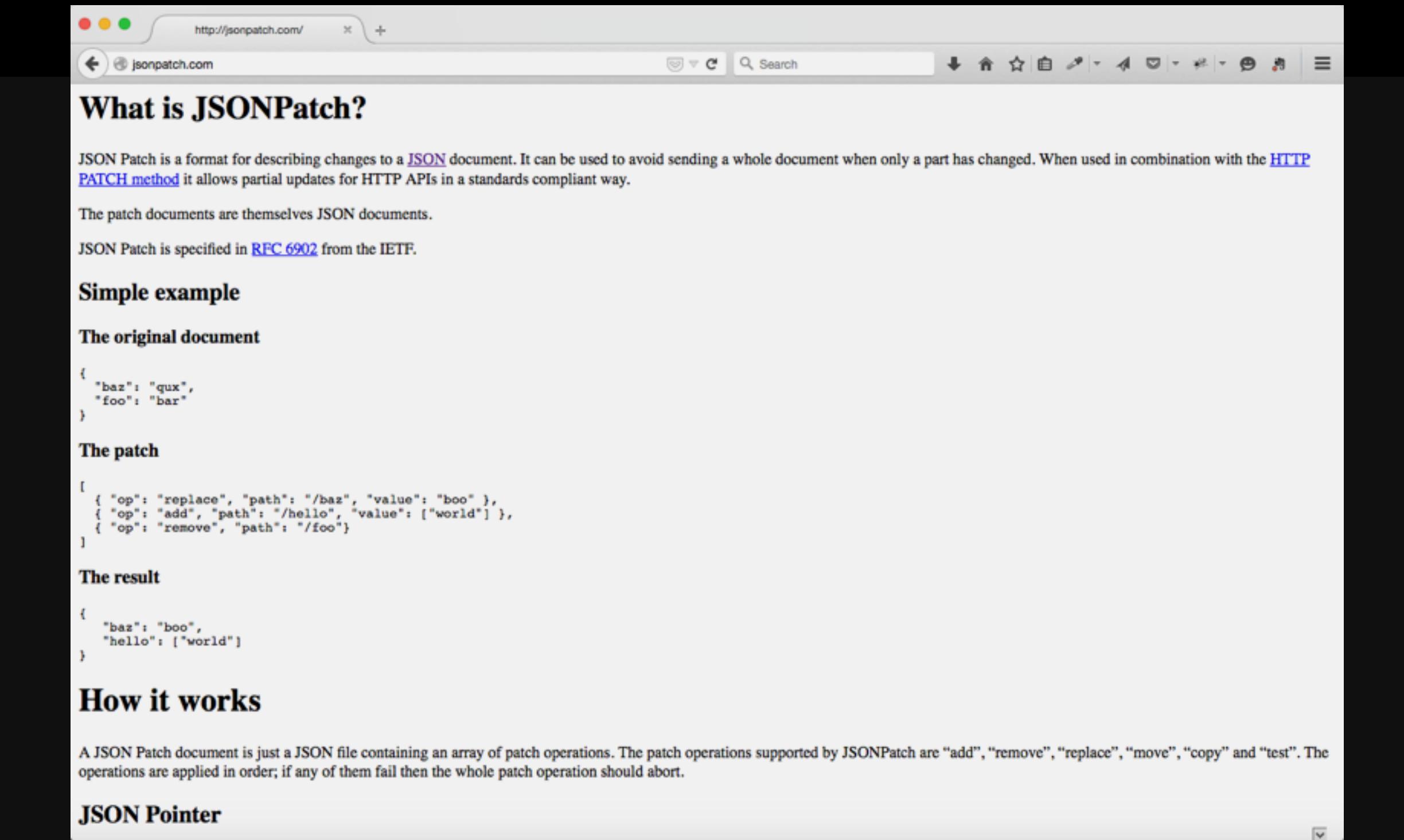
# jsonapter Test

```
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\n\njsonapter Test');
27           var output = j2j.run(template, jsonObj);
28           console.log('\n\n\n\nTransformed JSON');
29           console.log(JSON.stringify(output));
30         }
31         done();
32       });
33     });
34   });
35 });
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
788
789
789
790
791
792
793
794
795
796
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
888
889
889
890
891
892
893
894
895
896
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
979
979
980
981
982
983
984
985
986
987
988
989
989
990
991
992
993
994
995
996
997
998
999
999
1000
```

# 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 content is as follows:

## 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]". The main content area is titled "PROPOSED STANDARD". It contains information about the Internet Engineering Task Force (IETF), Request for Comments 6902, Category: Standards Track, ISSN: 2070-1721, and the editors: P. Bryan (Ed. at Salesforce.com) and M. Nottingham (Ed. at Akamai). The date is April 2013. The document title is "JavaScript Object Notation (JSON) Patch". The "Abstract" section defines JSON Patch as a structure for expressing operations to apply to a JSON document, suitable for HTTP PATCH. The "Status of This Memo" section states it is an Internet Standards Track document. The "Copyright Notice" section includes the IETF Trust copyright notice.

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

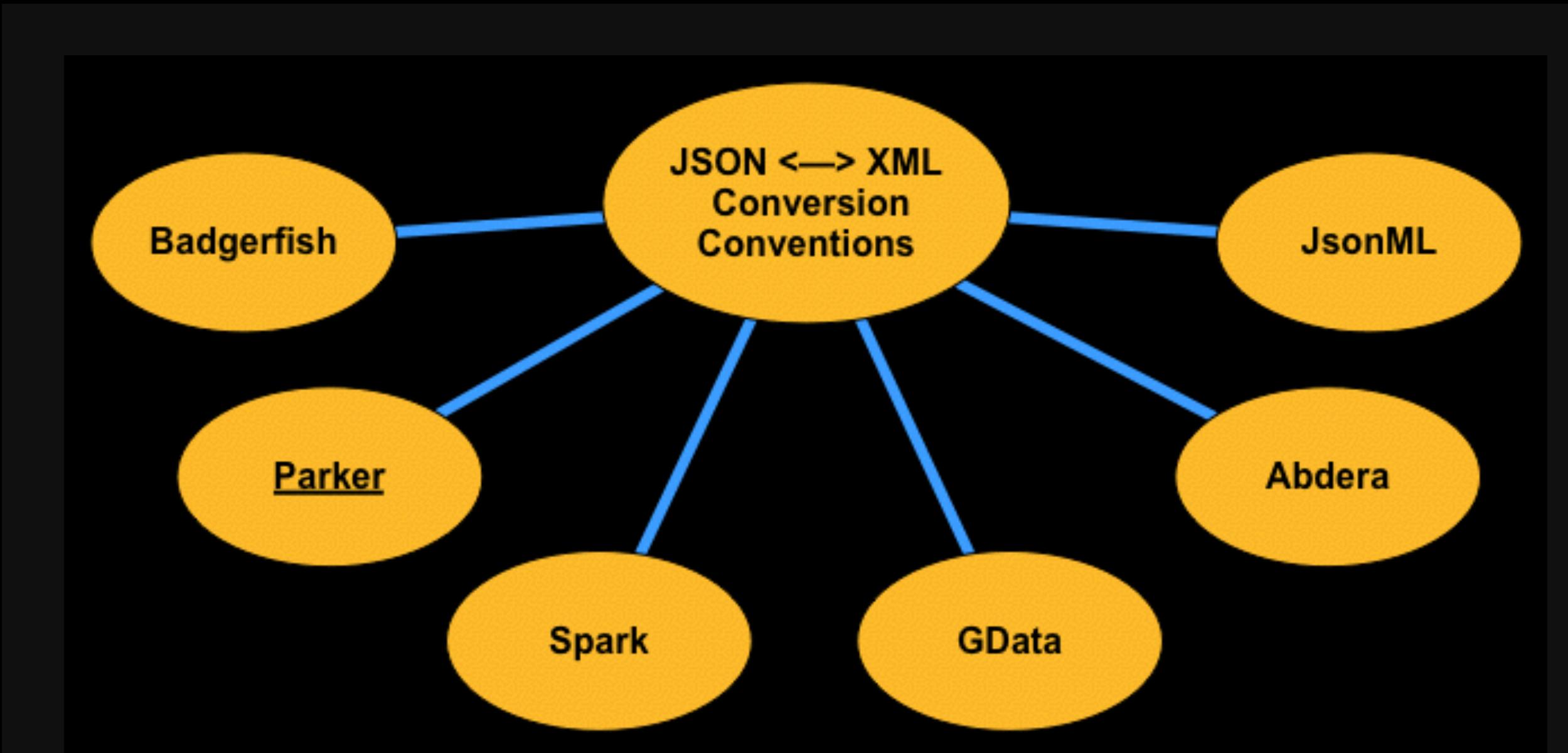
Mindshare	Y
Dev Community	Y
Platforms	JS, Node.js, Java, RoR, etc.
Intuitive	Y
Standard	RFC 6902 - Yes!

# Project 5 - JSON Transform

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

[\*\*projects/EspressoCON.md\*\*](#)

# JSON - XML Conversion Conventions



# Badgerfish Convention

The screenshot shows the homepage of the 'Badgerfish' Ning network. The title 'BADGERFISH' is prominently displayed at the top. A navigation bar below it includes links for MAIN, MY PAGE, MEMBERS, and PHOTOS. The main content area features a section titled 'WHAT IS BADGERFISH?' which defines it as a convention for translating XML documents into JSON objects. It explains that element names become object properties and text content becomes the \$ property. Below this, there's an example showing the transformation of an XML snippet into a JSON object. The sidebar on the left contains a 'PHOTOS' section with a single photo thumbnail labeled '<?xml?>' and a link to 'http://memphisdot.net/'.

**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, its history, and a transcription from the XML2JSON-XSLT project. A note about HTML microdata is also present. On the right side, there is a sidebar titled "IN THIS ARTICLE" with a list of related topics.

be converted into [Text](#) nodes), the process is unnecessarily costly. In fact, if your goal is to edit an XML document, it is strongly recommended to work on it rather than create new ones.

## 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

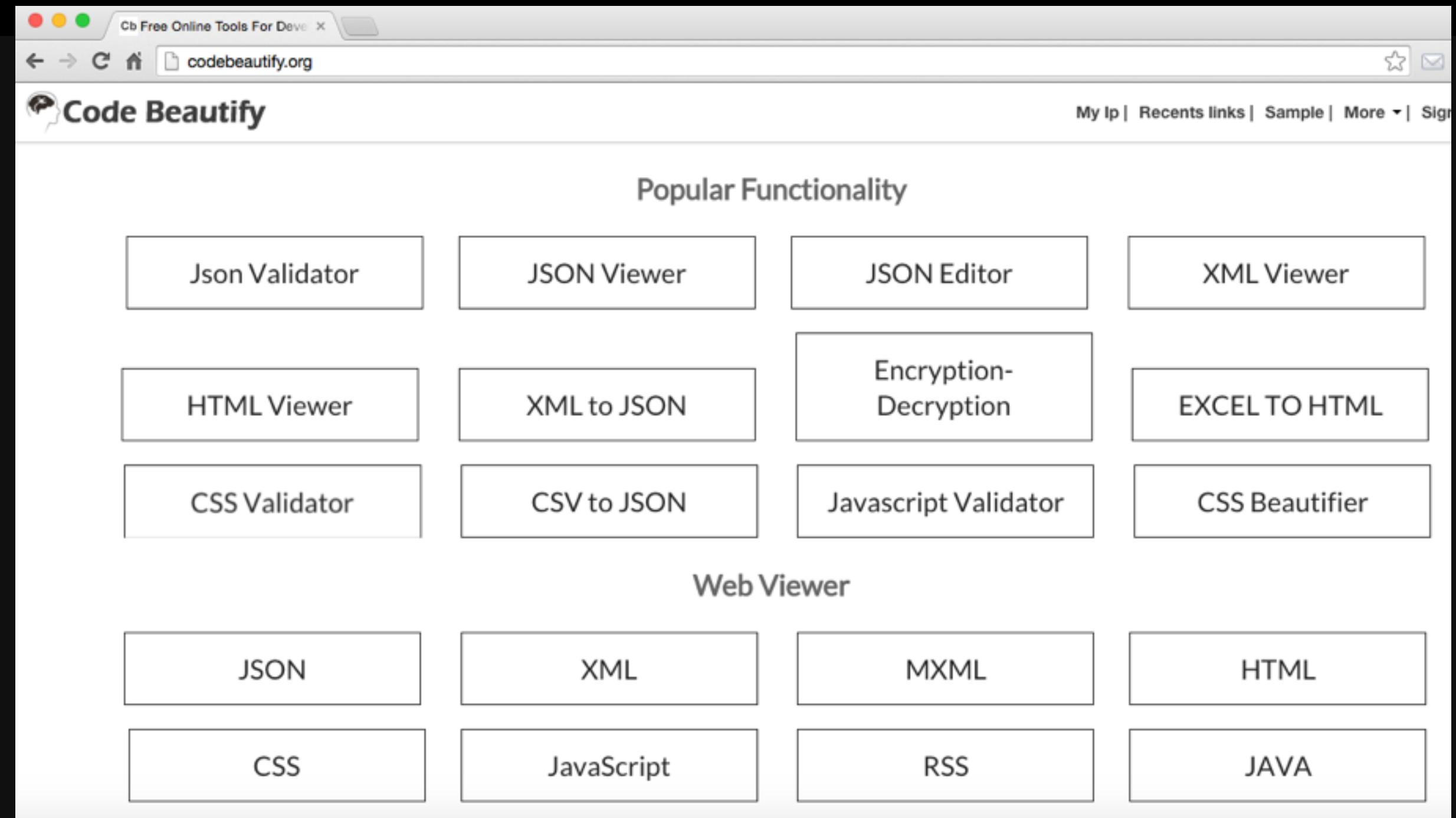
```
2 var expect = require('chai').expect;
3 var fs = require('fs');
4 var jxon = require('jxon');
5 var jsonfile = require('jsonfile');

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   });
});
```

# 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



# Agenda

**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'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](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/Tyson Swagger Petstore - [http://lbovet.github.io/swagger-ui/dist/index.html#!/pet/getPetById\\_get\\_0](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://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/mmclegg/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\\_tutorial/](http://goessner.net/articles/json_tutorial/)

<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://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.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/>