

JSON at Work: Schema, Search, and Transform

Tom Marrs



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

Examples and Slides

<https://github.com/tmarrs/presentations/tree/master/CS-OSUG/2015/JSON-at-Work-Schema-Search-and-Transform>

Where Are We?

**JSON Schema
Overview**

1

**JSON Search &
Transform Overview**

4

Core JSON Schema

2

JSON Search

5

**API Design with
JSON Schema**

3

JSON Transform

6

What is JSON Schema?

Validate Structure + Format

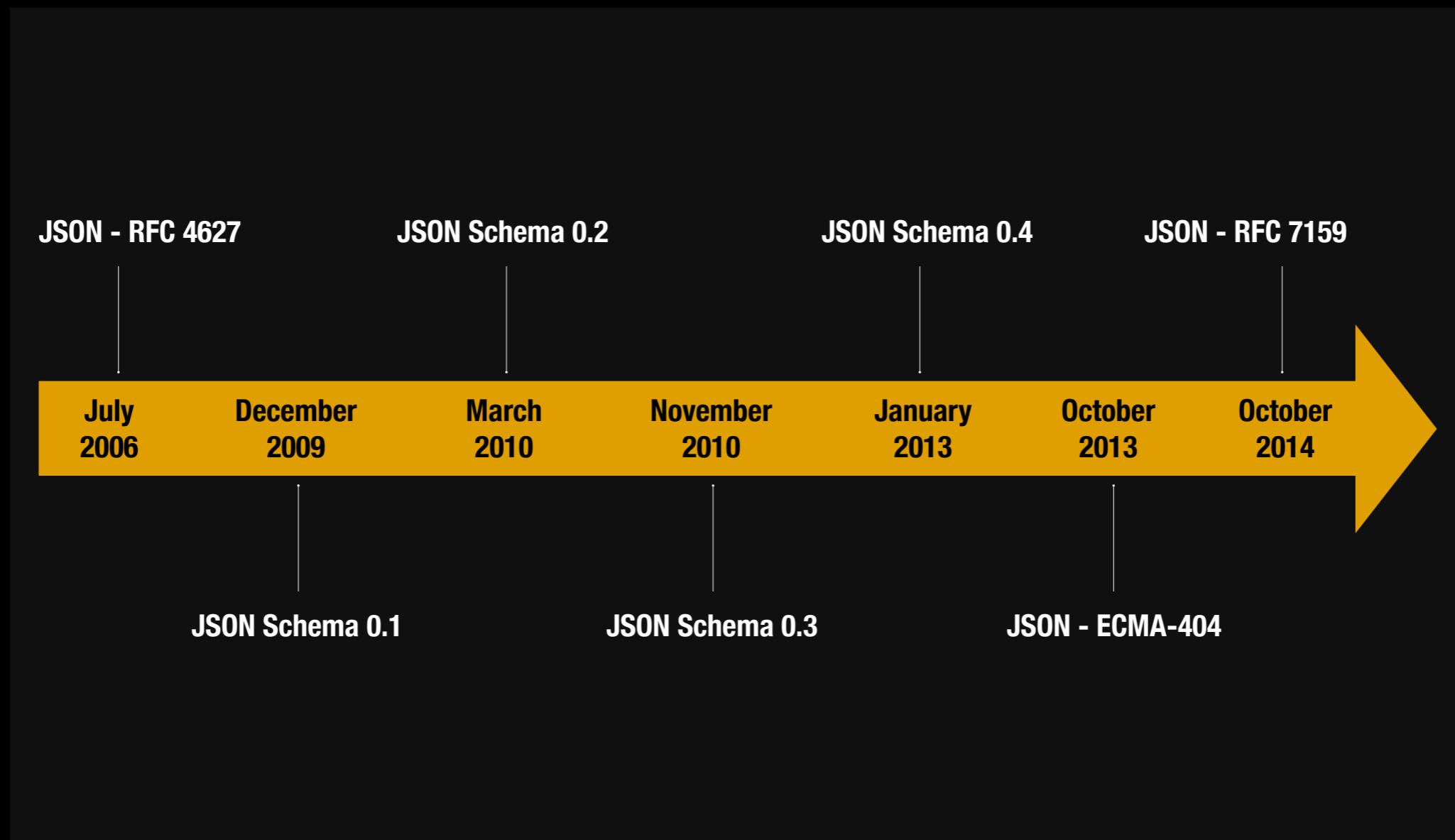
Basic JSON Schema

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

Basic JSON - Document

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

JSON Schema Timeline - When?



The Journey ...



json-schema.org

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 text "json-schema.org" and "The home of JSON Schema". Below the header is a navigation menu with four items: "about", "docs", "examples", and "software". The "about" item is underlined, indicating it is the current section. The main content area is divided into several horizontal sections. 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", lists the benefits of using JSON Schema and JSON Hyper-Schema. The third section, "JSON Schema", provides a bulleted list of its features. The fourth section, "JSON Hyper-Schema", also provides a bulleted list of its features. The fifth section, "More", includes a link to the specification and some examples.

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:

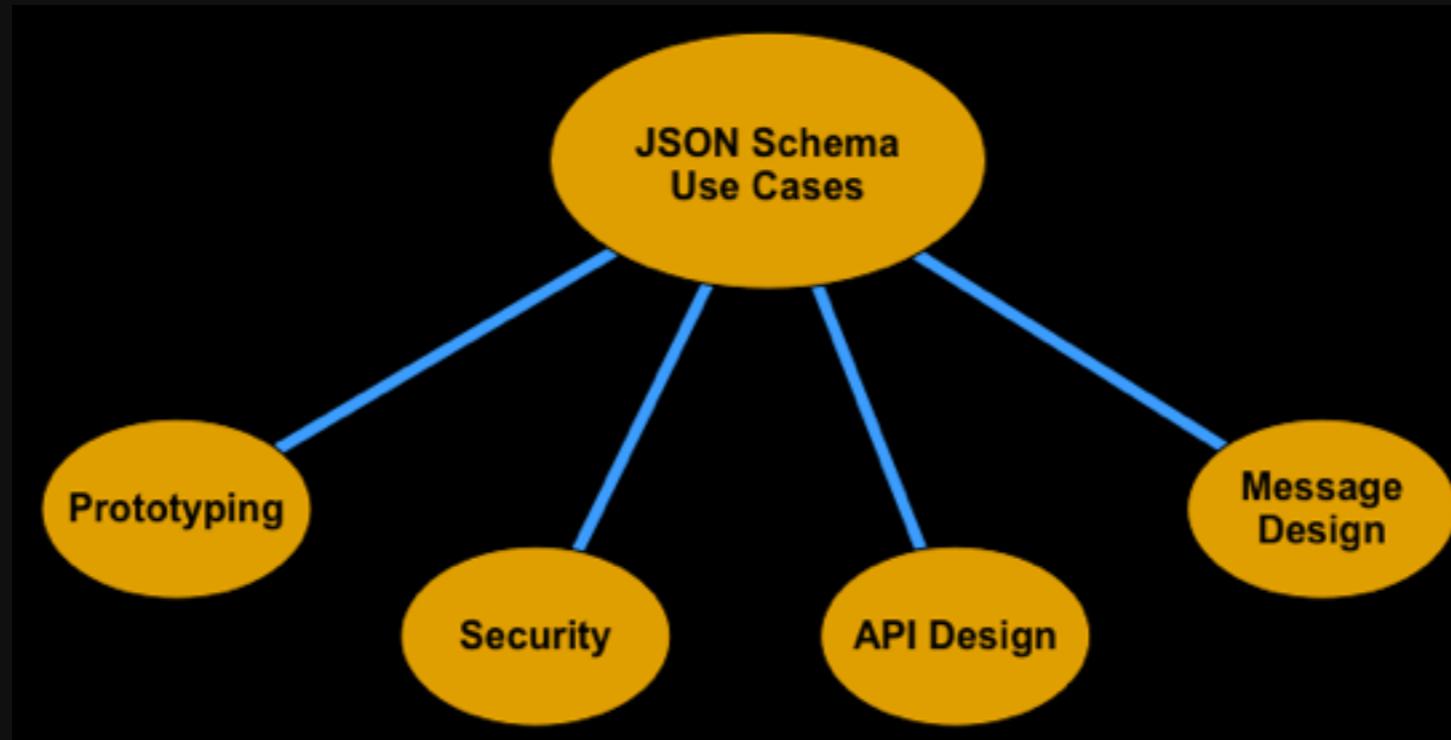
- the [specification](#)
- some [examples](#)

JSON Schema on GitHub

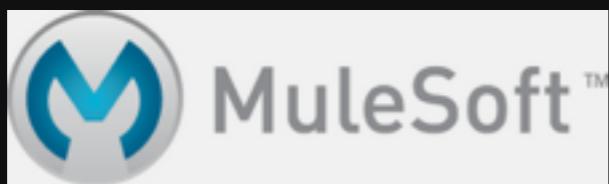
The screenshot shows a GitHub repository page for `kriszyp/json-schema`. The repository has 67 commits, 2 branches, 2 releases, and 7 contributors. The master branch is selected. The commit history lists various changes, including schema versioning, fixes, and updates to documentation and tests. A note in the commit history suggests adding a CNAME record to point to json-schema.org.

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

Where Does JSON Schema Fit?



Who uses JSON Schema?



Swagger



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

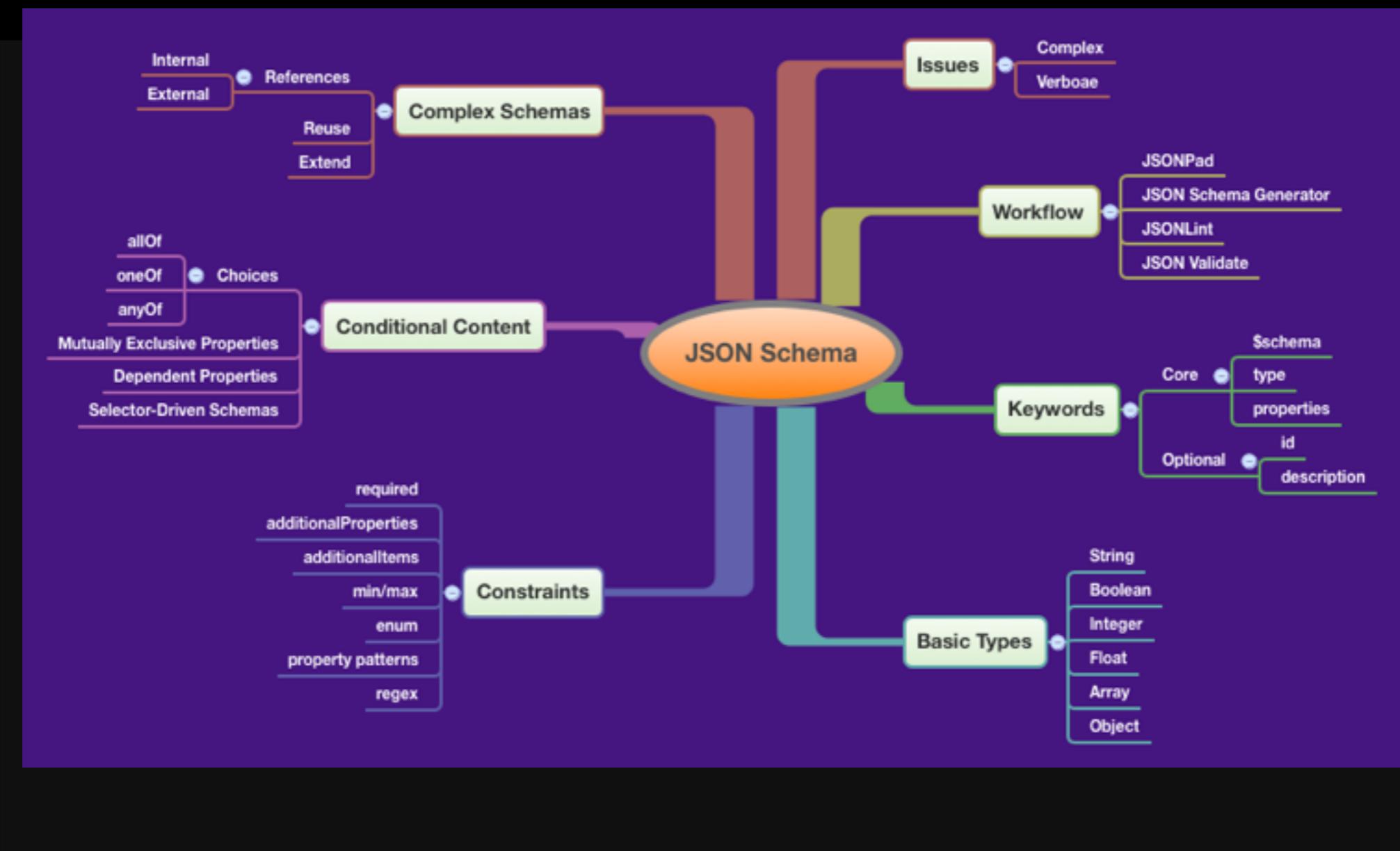
JSON Search

5

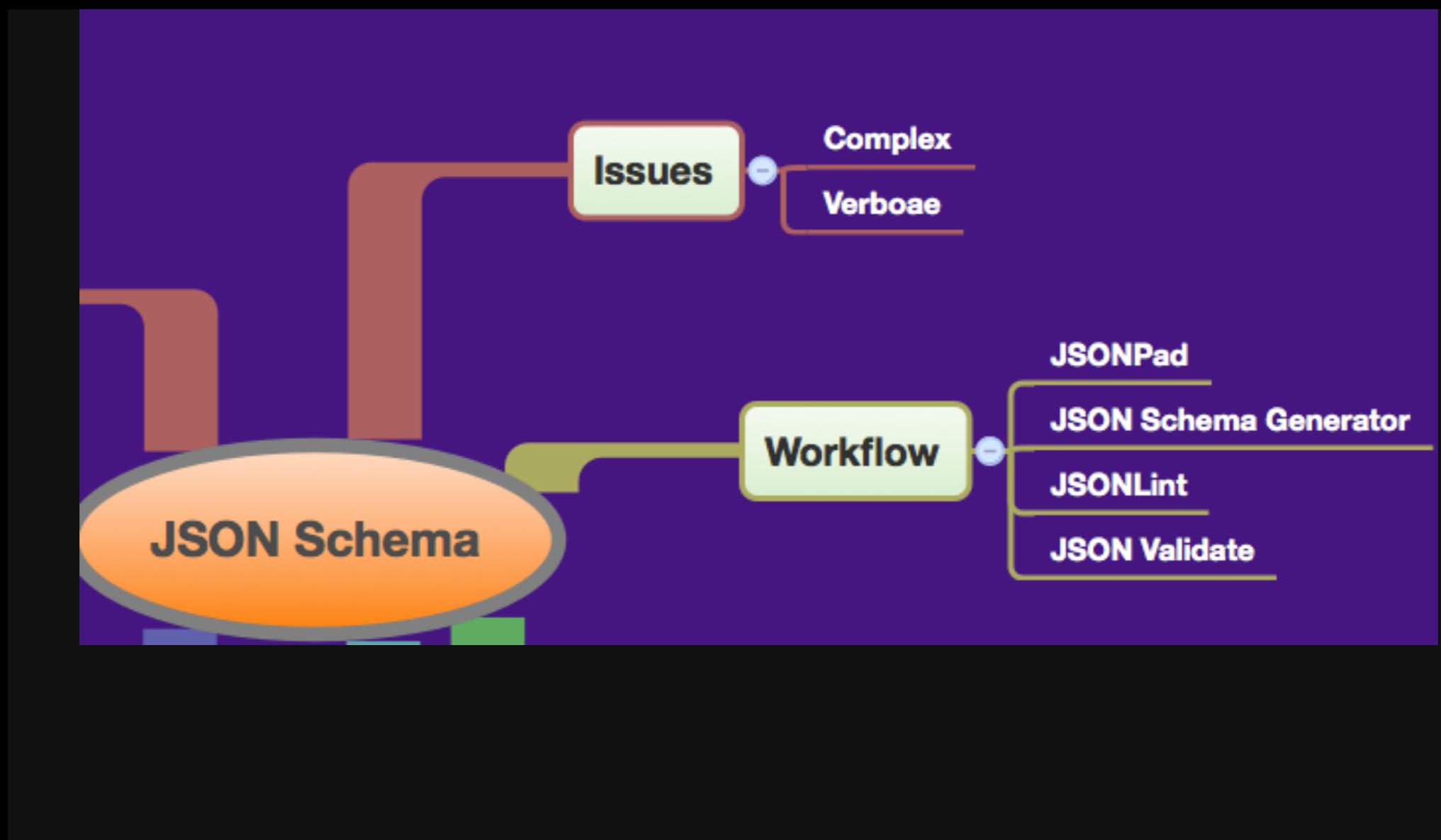
JSON Transform

6

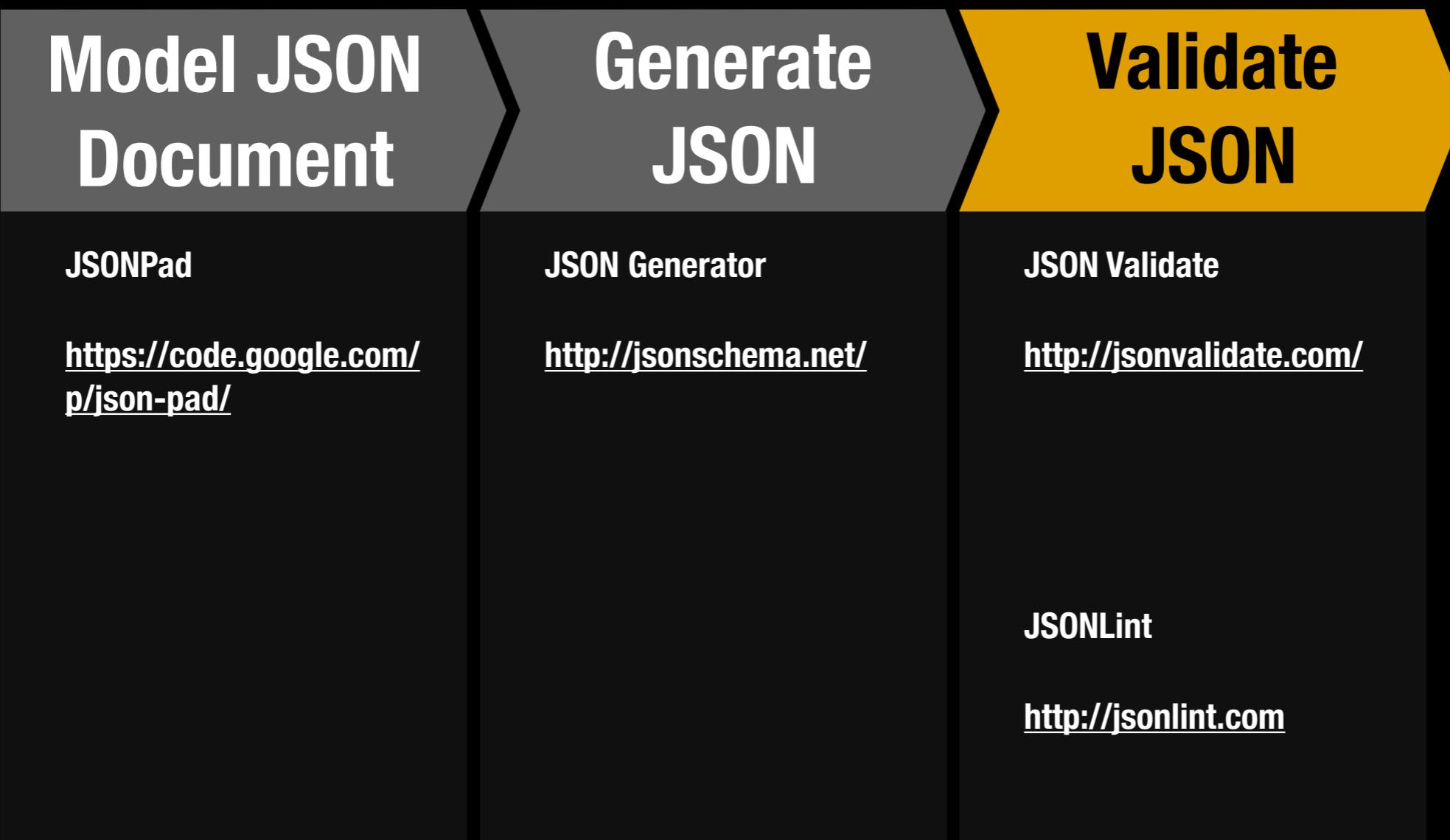
Facets of JSON Schema



JSON Schema - Issues and Workflow



My JSON Schema Workflow



JSONLint

The screenshot shows a web browser window for JSONLint. The address bar displays "jsonlint.com". The page content includes:

- A code editor area containing a JSON object with line numbers 1 through 21:

```
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
```
- A "Validate" button at the bottom left.
- A "Results" section at the bottom containing the message "Valid JSON" in a green box.
- Credit text: "Props to Douglas Crockford of JSON and JS Lint and Zach Carter, who provided the pure JS implementation of jsonlint."
- A "Kindling" logo with the text "JSON Lint is an idea from Arc90's Kindling".
- A "FAQ" link at the bottom right.

JSON Validate

The screenshot shows the JSON Validate interface on a web browser. The title bar reads "JSON Validate". The address bar shows "jsonvalidate.com". The main content area has a blue header bar with "JSON Validate" on the left and "Import About Help" on the right.

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

Buttons:

Validate Reset all

Links:

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>

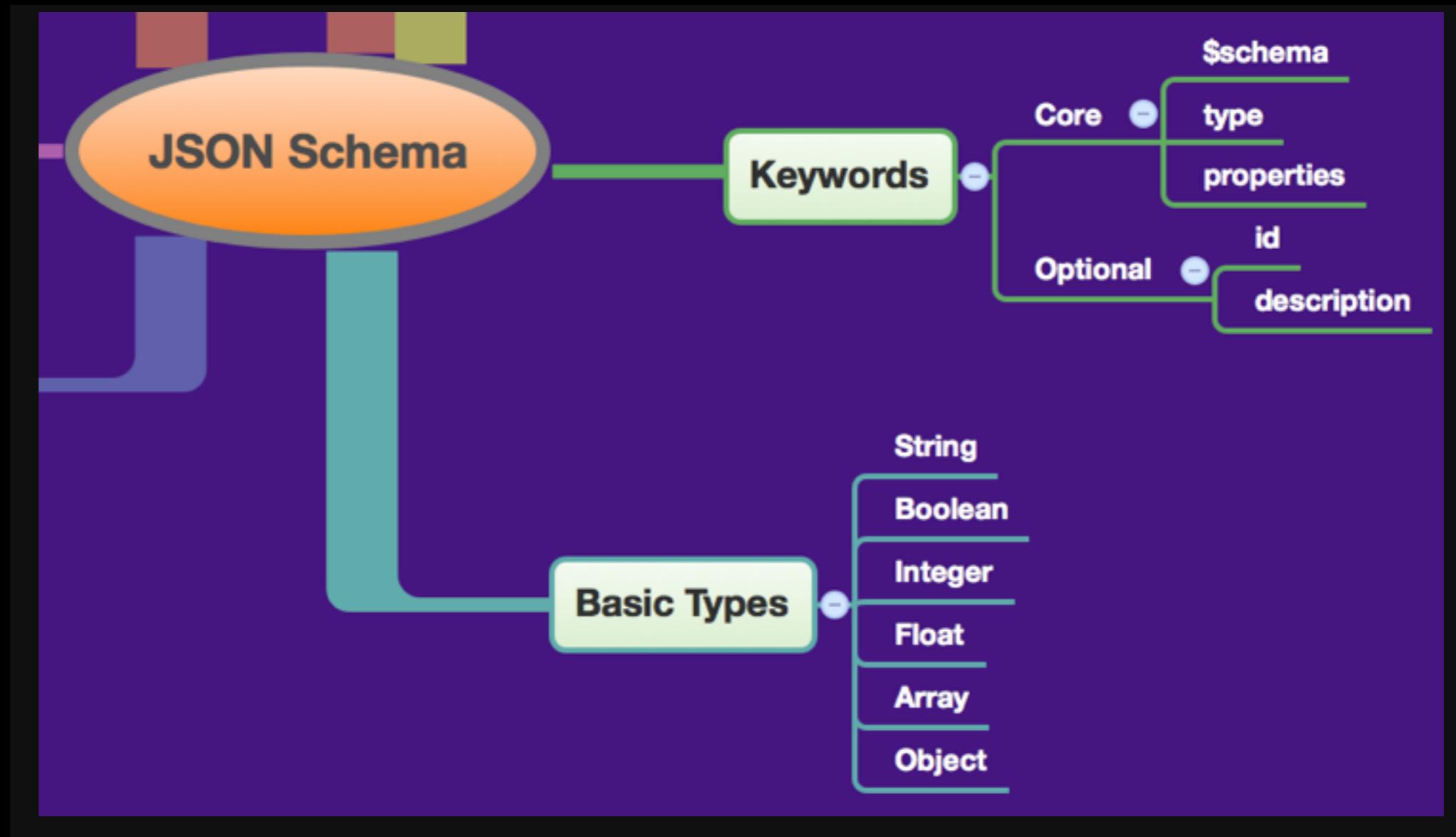
ujs-jsonvalidate

```
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	Specify JSON Schema version - “\$schema”: “ <u>http://</u> <u>json-schema.org/</u>
type	The data type - “type”: “string”
properties	The fields for an object

Optional Keywords

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

```
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 Document

```
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 Document

```
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 Document

```
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 Document

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

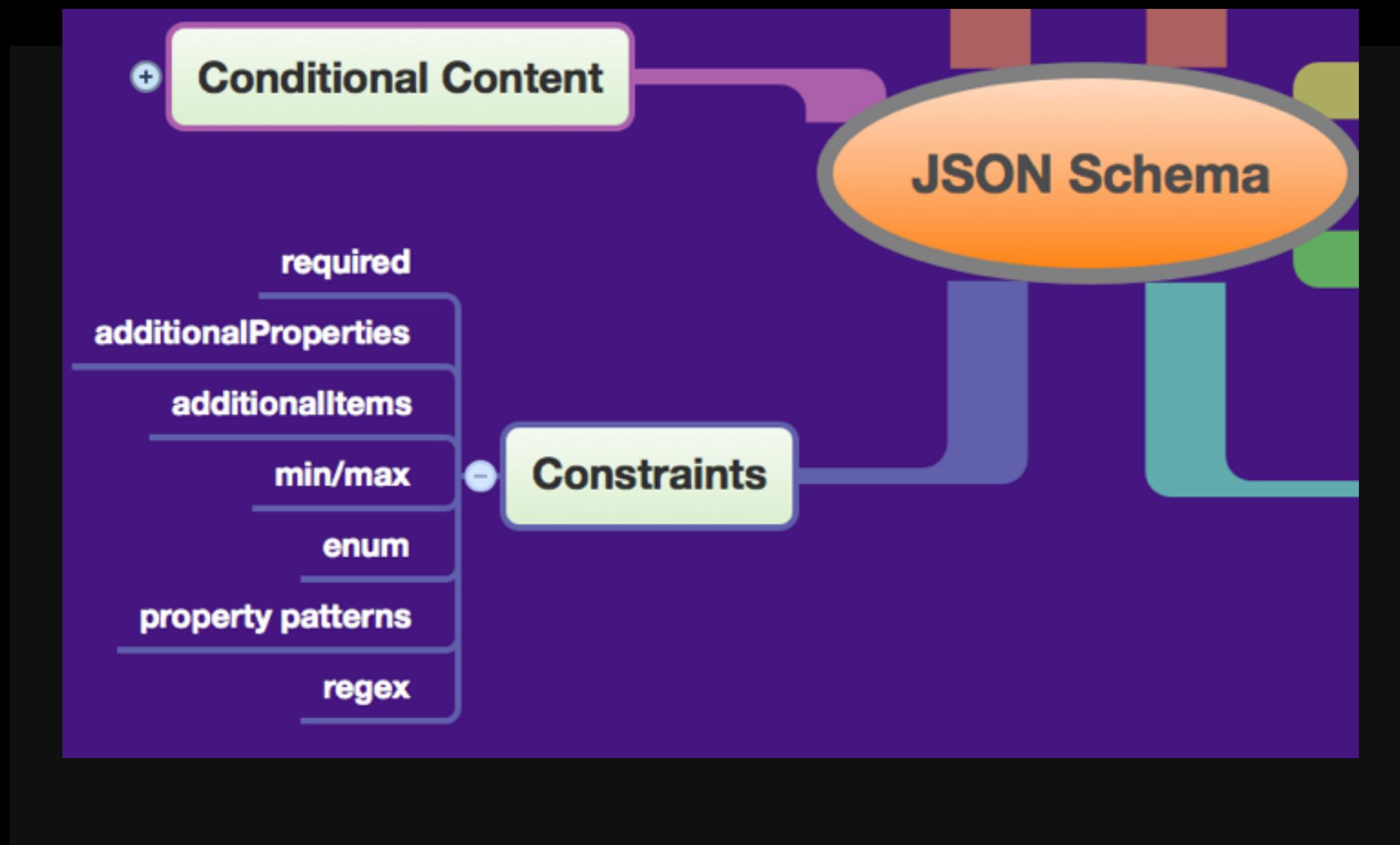
Named Object - JSON Schema

```
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 Document

```
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 }
```

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 Document

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

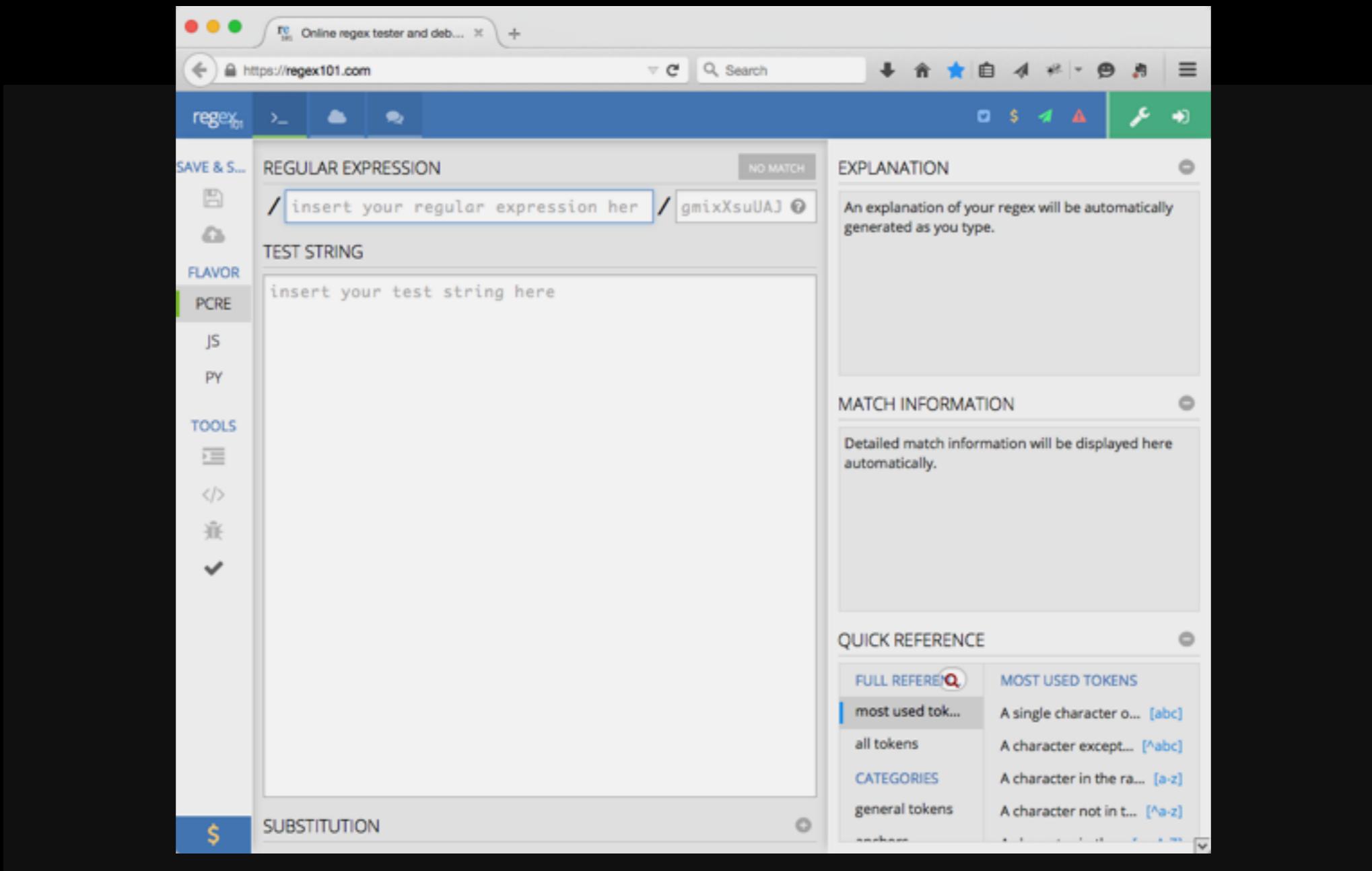
Regex - JSON Schema

```
3  {
4    "$schema": "http://json-schema.org/draft-04/schema#",
5    "type": "object",
6    "properties": {
7      "email": {
8        "type": "string",
9        "pattern": "^\w+@\w+\.\w{2,4}$"
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 Document

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

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



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 a "Welcome" section containing links to Regular Expressions Quick Start, Tutorial, Replacement Strings Tutorial, Applications and Languages, Examples, Reference, and more. There are also links for Book Reviews, Printable PDF, About This Site, and RSS Feed & Blog. A "Make a Donation" button is visible next to payment icons for PayPal and Bitcoin. The main content area features a callout box for RegexBuddy, a screenshot of the software interface, and text about easily creating and understanding regular expressions. Below this, a section titled "Welcome to Regular-Expressions.info" discusses what regular expressions are and how they can be used. At the bottom, there's a footer with a book cover thumbnail for "Regular Expressions Cookbook" and a status bar from Runscope indicating "It's going to be 200 OK".

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

Regexr

The screenshot shows the RegExr v2.0 web application. The interface includes a sidebar on the left with information about escaped characters and a main workspace on the right where users can enter regular expressions and test them against sample text.

Sidebar (Escaped characters):

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

Main Workspace:

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

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

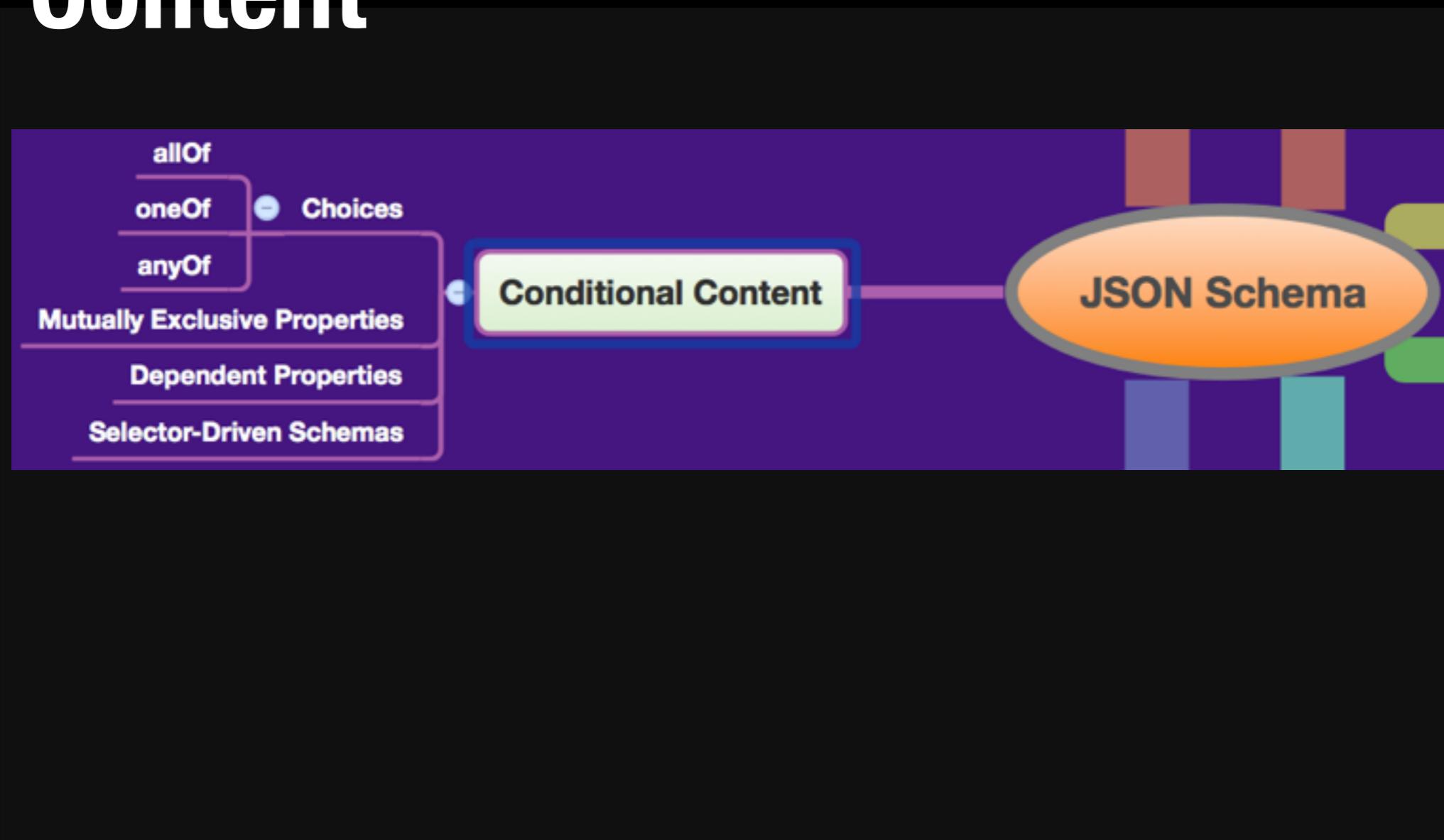
Instructions: Edit the Expression & Text to see matches. Roll over matches or the expression for details. Undo mistakes with cmd-z. Save & Share expressions with friends or the Community. A full Reference & Help is available in the Library, or watch the video Tutorial.

Sample text for testing:

```
abcdefghijklmnoprstuvwxyz 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:

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 Document

```
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

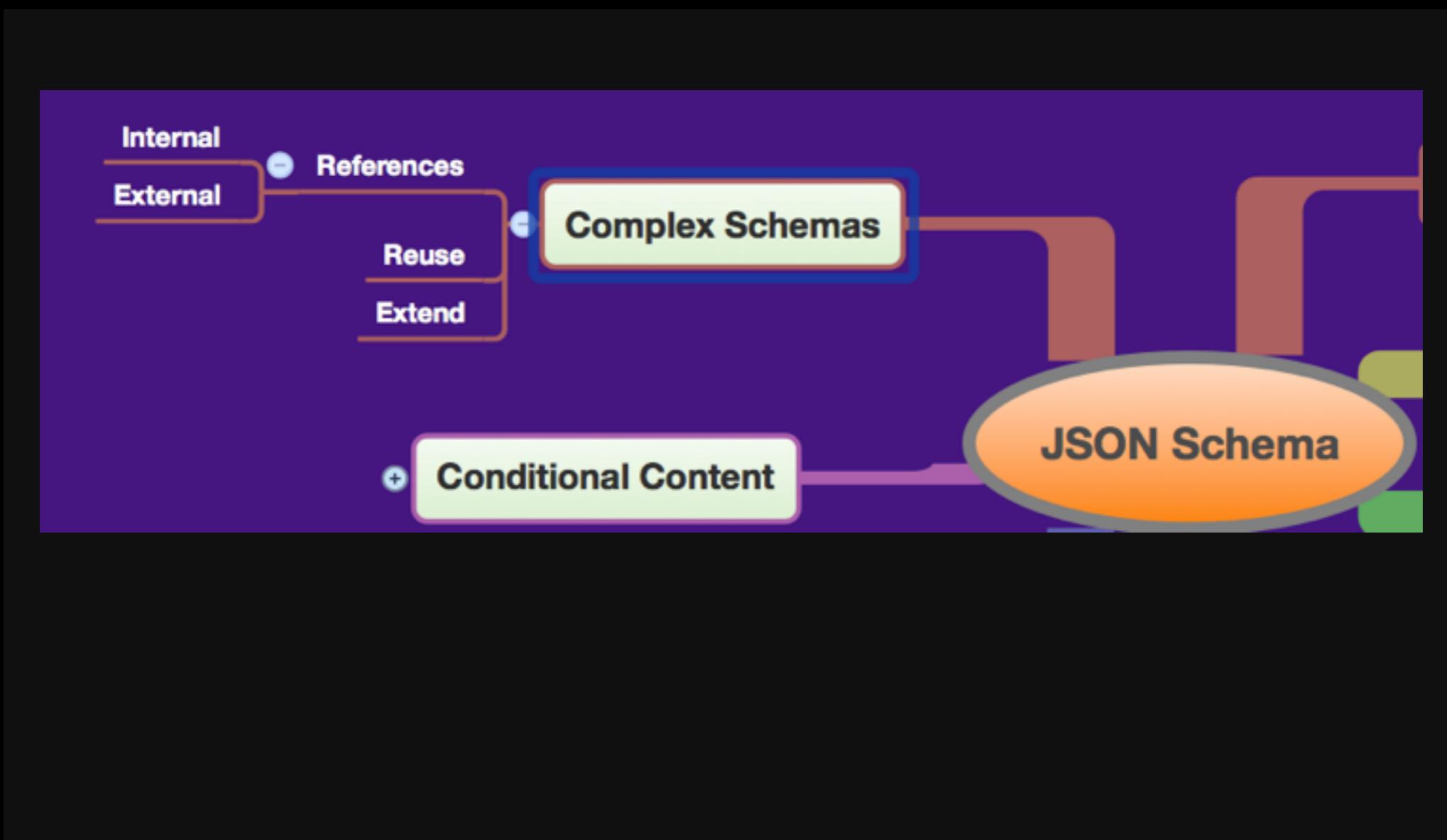
anyOf - JSON Schema

```
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 Document

```
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 Document

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

http-server

```
npm install -g http-server
```

```
http-server -p 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 Document

```
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

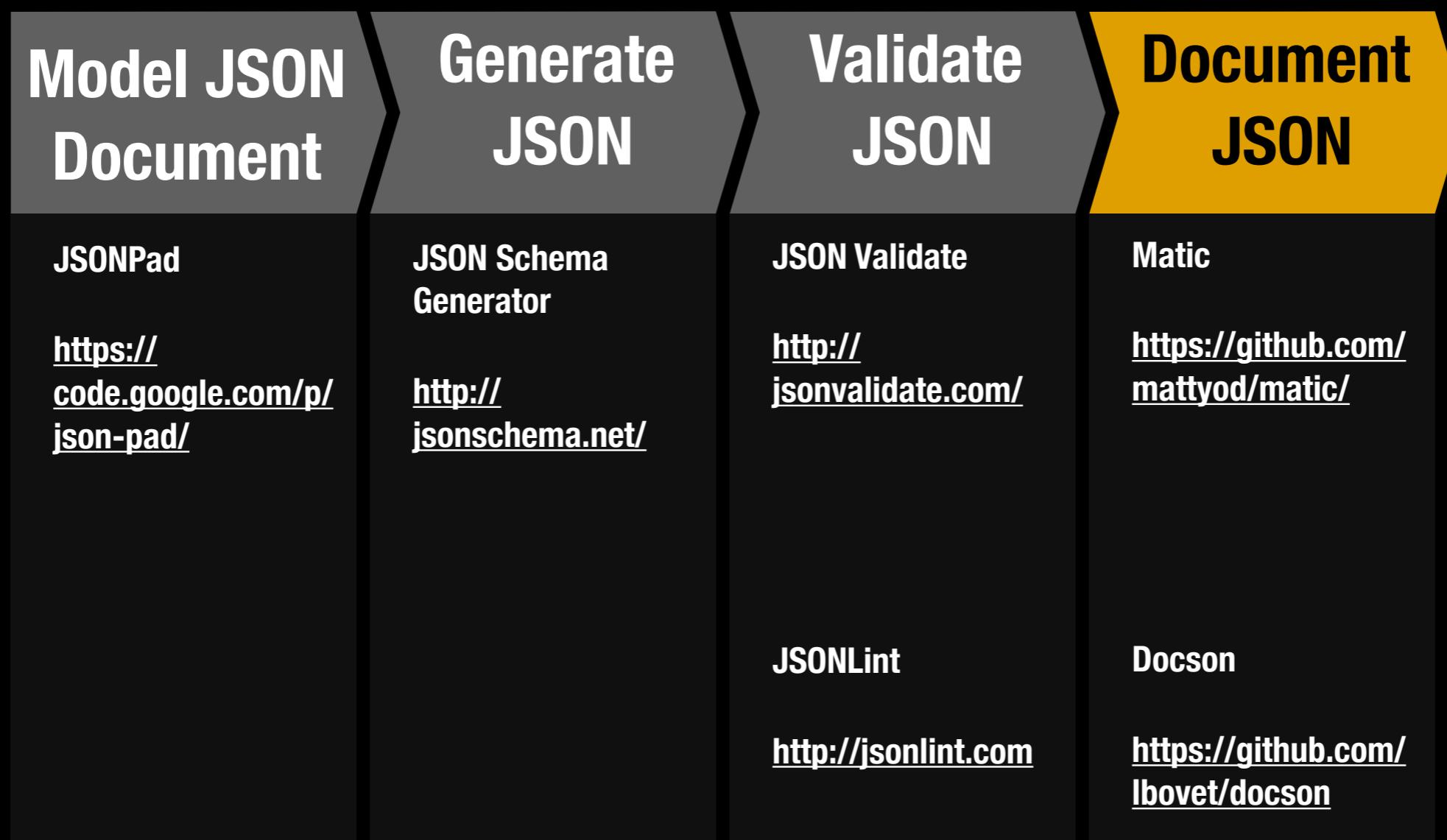
Real World Use Case

**Design/Implement API and Consumer
in Parallel**

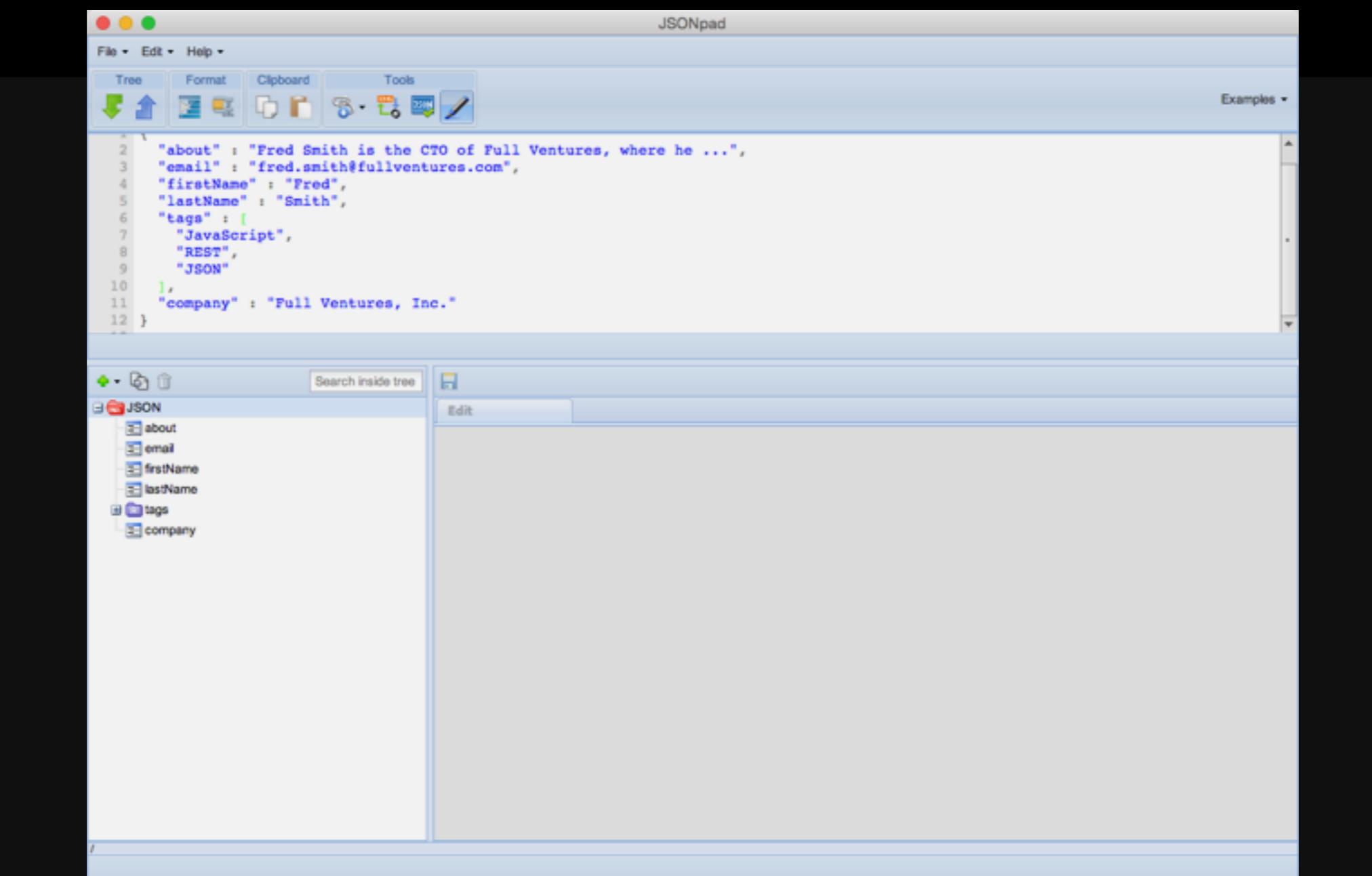
Our Scenario

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

My JSON Schema Workflow for APIs



JSONPad Demo



JSON Schema Generator Demo

The screenshot shows a web-based JSON Schema Generator interface. On the left, a URL input field contains `http://jsonsatwork.org`. Below it, a JSON preview area 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 below the JSON preview says "Well done! You provided valid JSON." Below this, there are two buttons: "Generate Schema" (highlighted in blue) and "Reset".

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 interface includes several configuration sections with checkboxes:

- 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.

JSON Validate Demo

The screenshot shows the JSON Validate demo interface. The top navigation bar includes tabs for "Import", "About", and "Help". The main area is divided into two sections: "JSON Schema" and "JSON Content".

JSON Schema:

```
24     "id": "tags",
25     "type": "array",
26     "items": {
27       "type": "string"
28     },
29     "additionalItems": false
30   },
31   "company": {
32     "id": "company",
33     "type": "string"
34   }
35   },
36   "additionalProperties": false,
37   "required": [
38     "about",
39     "email",
40     "firstName",
41     "lastName",
42     "tags",
43     "company"
44   ]
45 }
```

JSON Content:

```
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  }
13
14
15
16
17
18
19
20
21
--
```

References:

1 2 3 4 5 6 7 8

Results:

Valid

Buttons:

Validate Reset all

Links:

Learn more about Using JSON Schema UJS

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 indicates the URL is http://lbovet.github.io/swagger-ui/dist/index.html#/pet/getPetById_get_0. The main content area displays the **Swagger Example with Typson and Docson Integration**. It includes sections for **user** and **pet** operations. The **pet** section is currently active, showing a **GET /pet/{petId}** operation. Below this, there is an **Implementation Notes** section stating "Returns a pet based on ID". A **Response Class** section is shown, with the **Model** tab selected, displaying a detailed schema for a **Pet** object:

Name	Type	Description
id	<code>int64 [0.0;100.0]</code>	Unique identifier for the Pet
category	<code>Category</code>	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 available pending sold</code>	pet status in the store

At the bottom of the page, the URL <https://github.com/lbovet/docson> is displayed.

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 a **GET** request for the endpoint `/pet/{petId}`. The response class for status 200 is shown, containing a JSON schema for a pet object. The parameters section shows a required parameter `petId` of type long. The response messages section lists two status codes: 400 (Invalid ID supplied) and 404 (Pet not found). A "Try it out!" button is visible at the bottom left.

Implementation Notes
Returns a single pet

Response Class (Status 200)

Model Model Schema

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

Response Content Type application/xml

Parameters

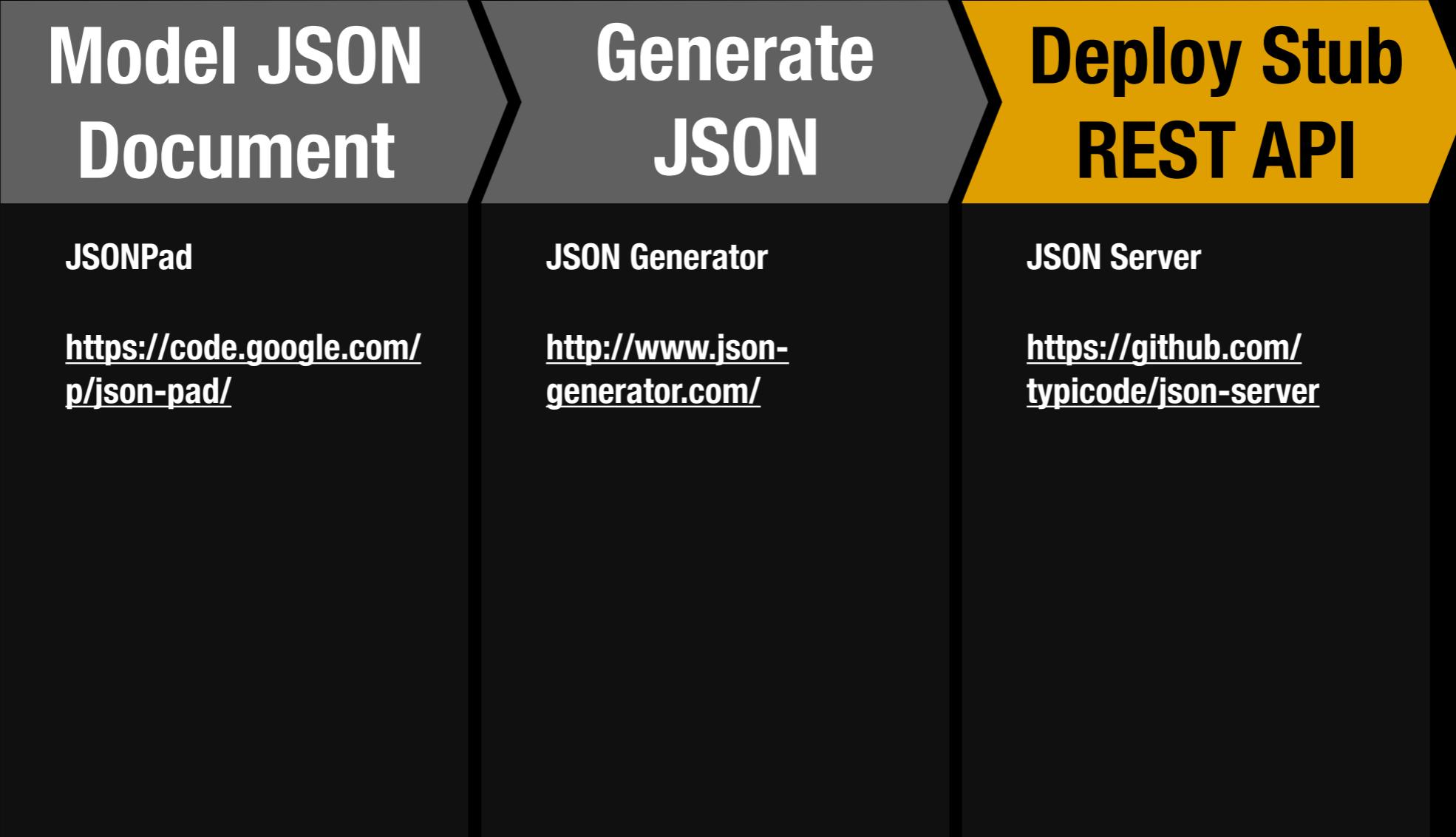
Parameter	Value	Description	Parameter Type	Data Type
petId	[required]	ID of pet to return	path	long

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



JSON Generator Demo

The screenshot shows a web browser window titled "JSON Generator - Tool for..." with the URL "www.json-generator.com". The main content area is titled "JSON GENERATOR". On the left, there is a code editor containing a JSON template:

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

On the right side of the interface, there is a large, semi-transparent watermark with the text "Click 'Generate' and wait for the magic!" and a wand icon.

At the bottom of the page, there is a footer bar with the text "Created by Vazha Ormanashvili, 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>

Where Are We?

**JSON Schema
Overview**

1

Core JSON Schema

2

**API Design with
JSON Schema**

3

**JSON Search &
Transform
Overview**

4

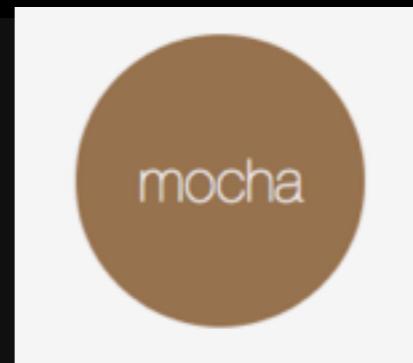
JSON Search

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 features a dark 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 also present. The main content area has a large blue header with the text "FIREBASE" and "Open Data Sets" next to a database icon. Below this, a paragraph explains what Firebase Open Data Sets are. A table lists five available datasets with their descriptions.

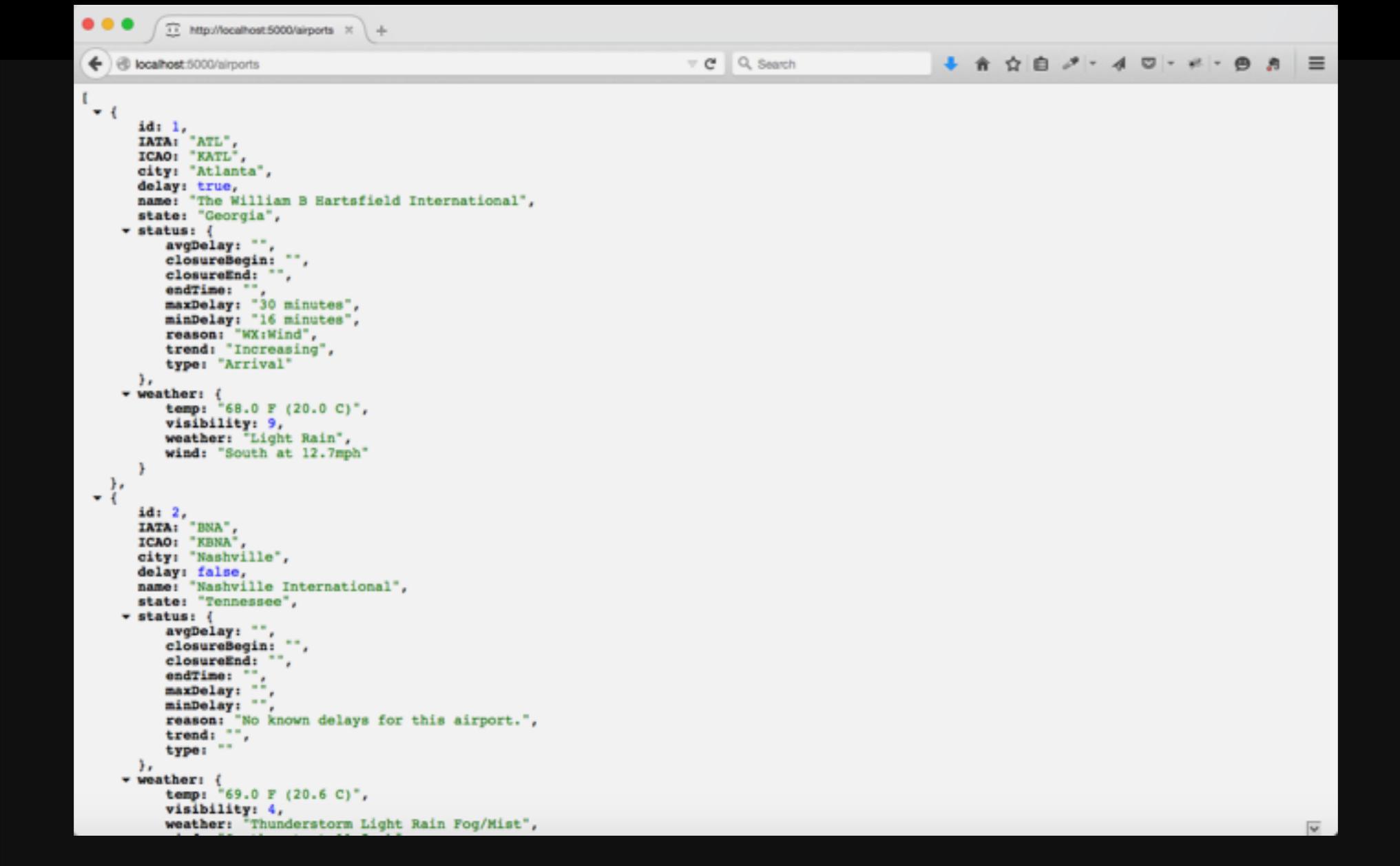
Data Set	Description
Airport Delays	Get the latest airport delay and status updates in realtime.
Cryptocurrencies	Get the latest USD/BTC and USD/LTC exchange rates in realtime.
Earthquakes	Information on earthquakes anywhere on Earth in realtime.
Parking	Realtime data on the latest street parking price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

JSON Server - Airports

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

<http://localhost:5000/airports>

Airports Stub Service



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

```
[{"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": ""}}]
```

JSON Search & Transform Rubric



Where Are We?

JSON Schema
Overview

1

JSON Search &
Transform Overview

4

Core JSON Schema

2

JSON Search

5

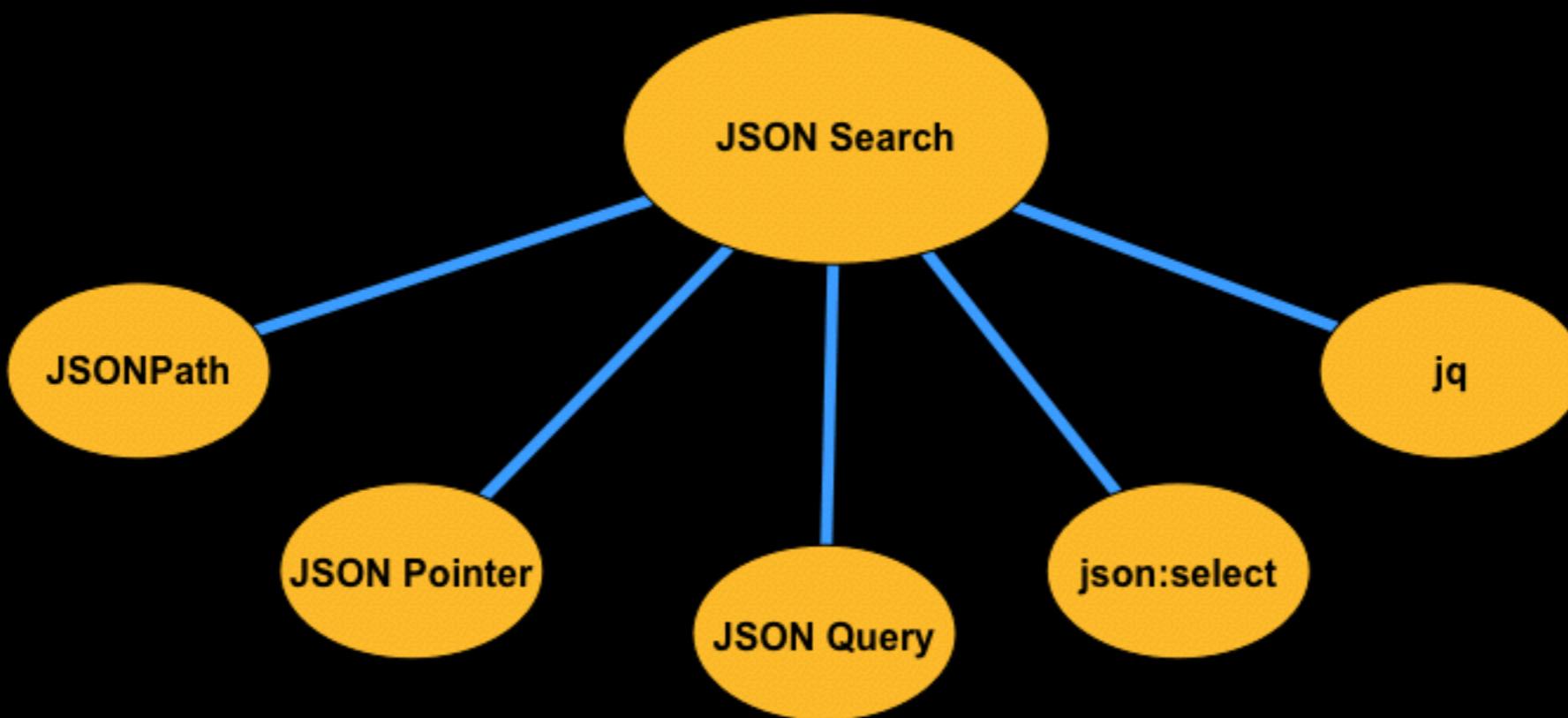
API Design with
JSON Schema

3

JSON Transform

6

JSON Search Tools



JSONPath

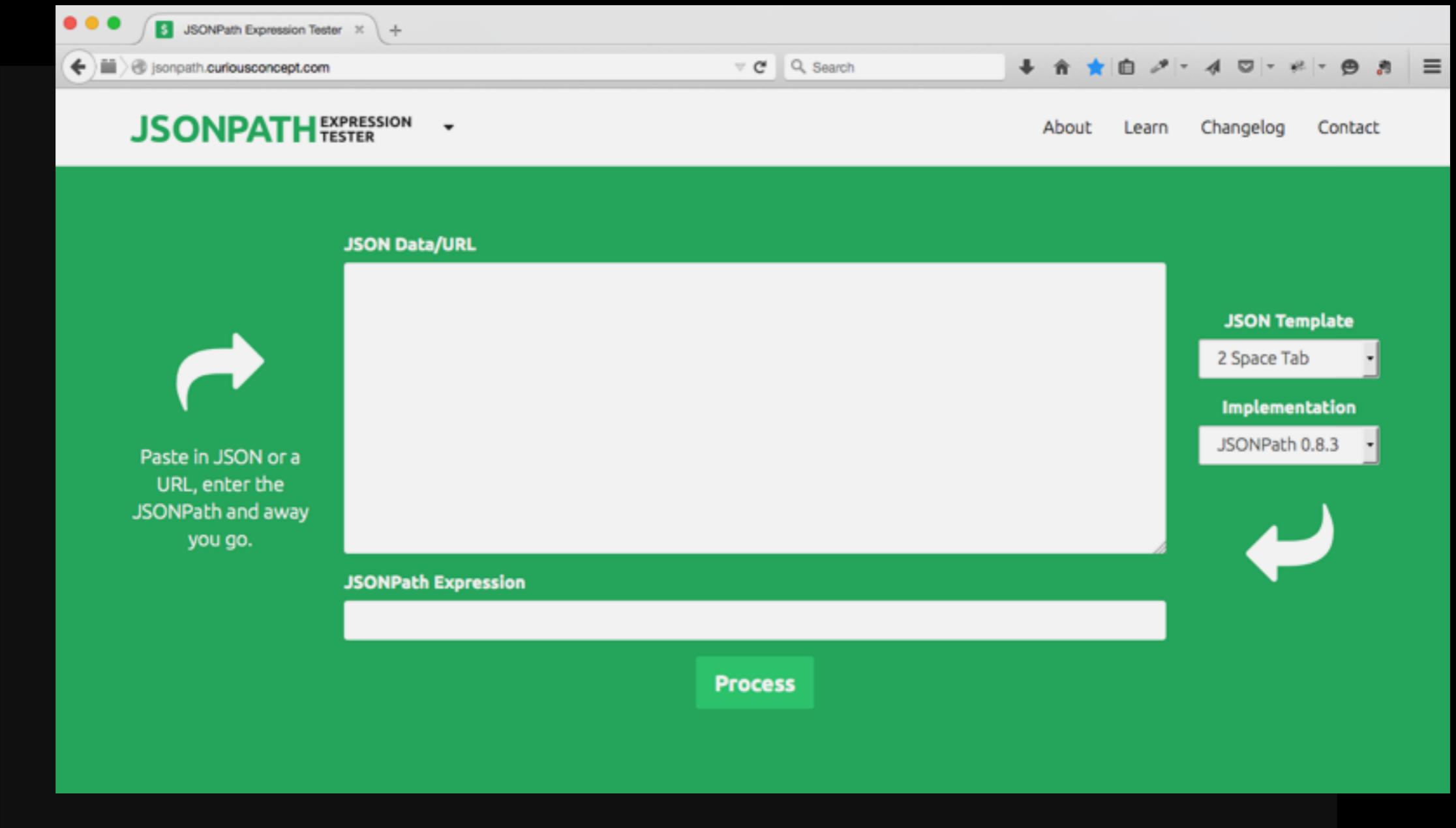
The screenshot shows a web browser window with the following details:

- Title Bar:** JSONPath - XPath for JSON
- Address Bar:** goessner.net/articles/JsonPath/
- Content Area:**
 - 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



JSONPath Online Evaluator

The screenshot shows a web browser window for the "JSONPath Online Evaluator" at ashphy.com/JSONPathOnlineEvaluator/. The page title is "JSONPath Online Evaluator". Below the title, it says "Author: Kazuki Hamasaki [ashphy@ashphy.com]" and "This evaluator uses [JSONPath - XPath for JSON](#)".

Inputs

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

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

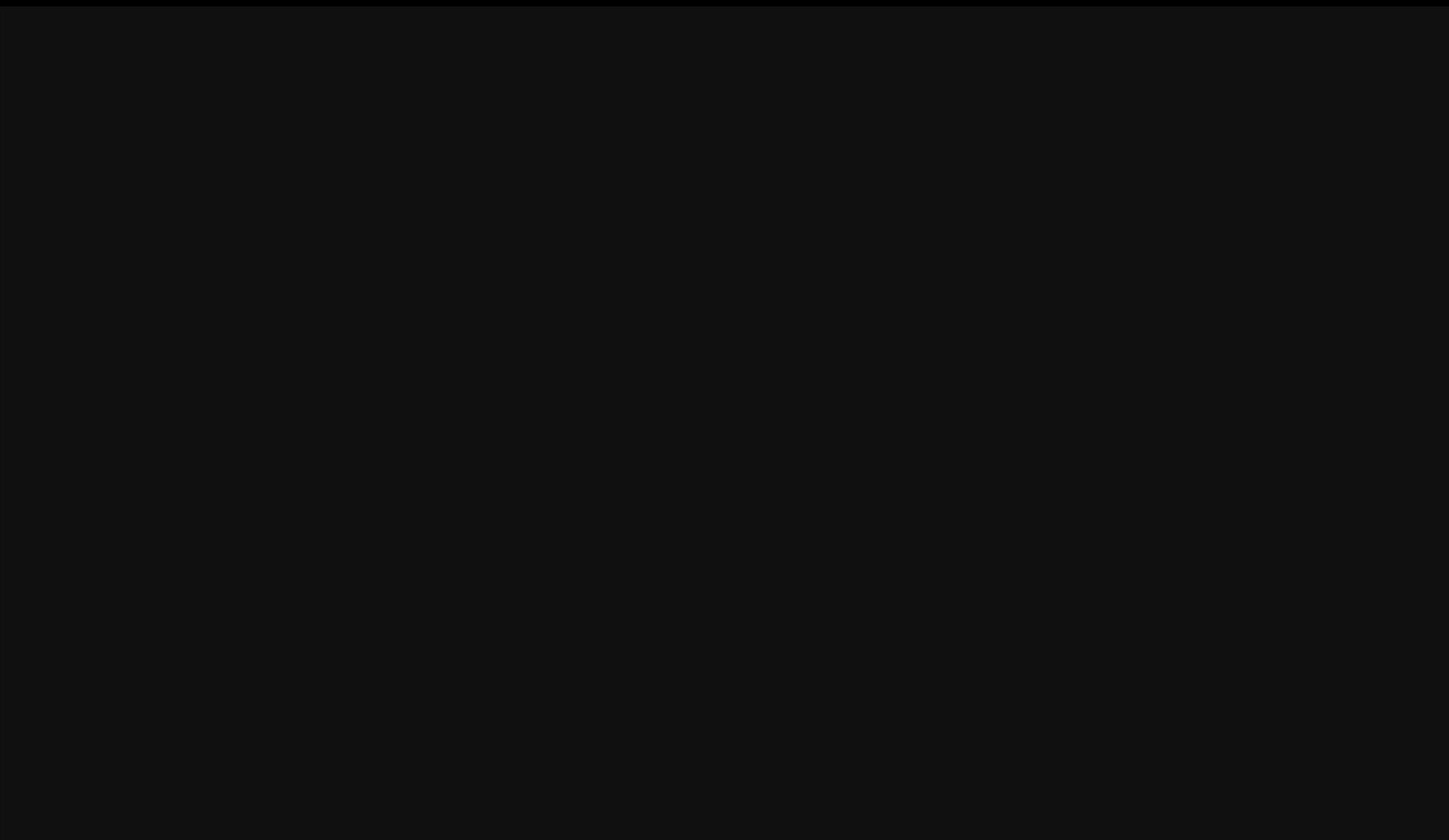
JSON

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

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 displaying the IETF 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:

- Navigation links: [Docs] [txt|pdf] [draft-ietf-appsaw...] [Diff1] [Diff2] [Errata]
- Document status: PROPOSED STANDARD
- Editor: P. Bryan, Ed.
- Contributor: Salesforce.com
- Reviewer: K. Zyp
- SitePen (USA)
- Reviewers: M. Nottingham, Ed.
- Reviewers: Akamai
- Date: April 2013
- Section: 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.
- Text: 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](#).
- Text: 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~n"	8

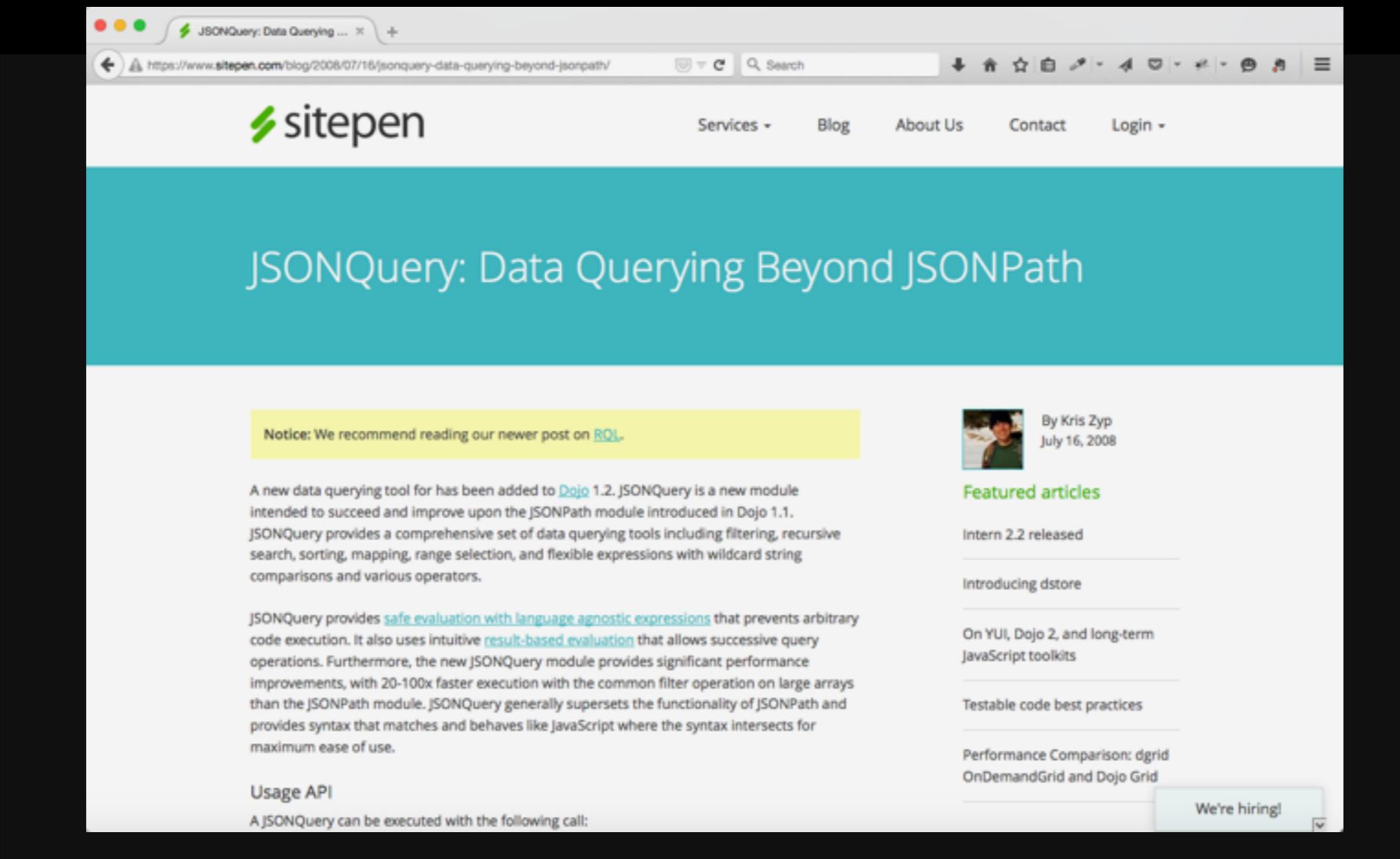
JSON Pointer Test

```
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



A screenshot of a web browser displaying a blog post titled "JSONQuery: Data Querying Beyond JSONPath" on the sitepen.com website. The post discusses the addition of a new data querying tool to Dojo 1.2, which is intended to succeed and improve upon the JSONPath module. It highlights features like filtering, recursive search, sorting, mapping, range selection, and flexible expressions with wildcard string comparisons and various operators. The post also mentions safe evaluation and result-based evaluation. A sidebar on the right features a photo of Kris Zyp, the author, and links to other featured articles.

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

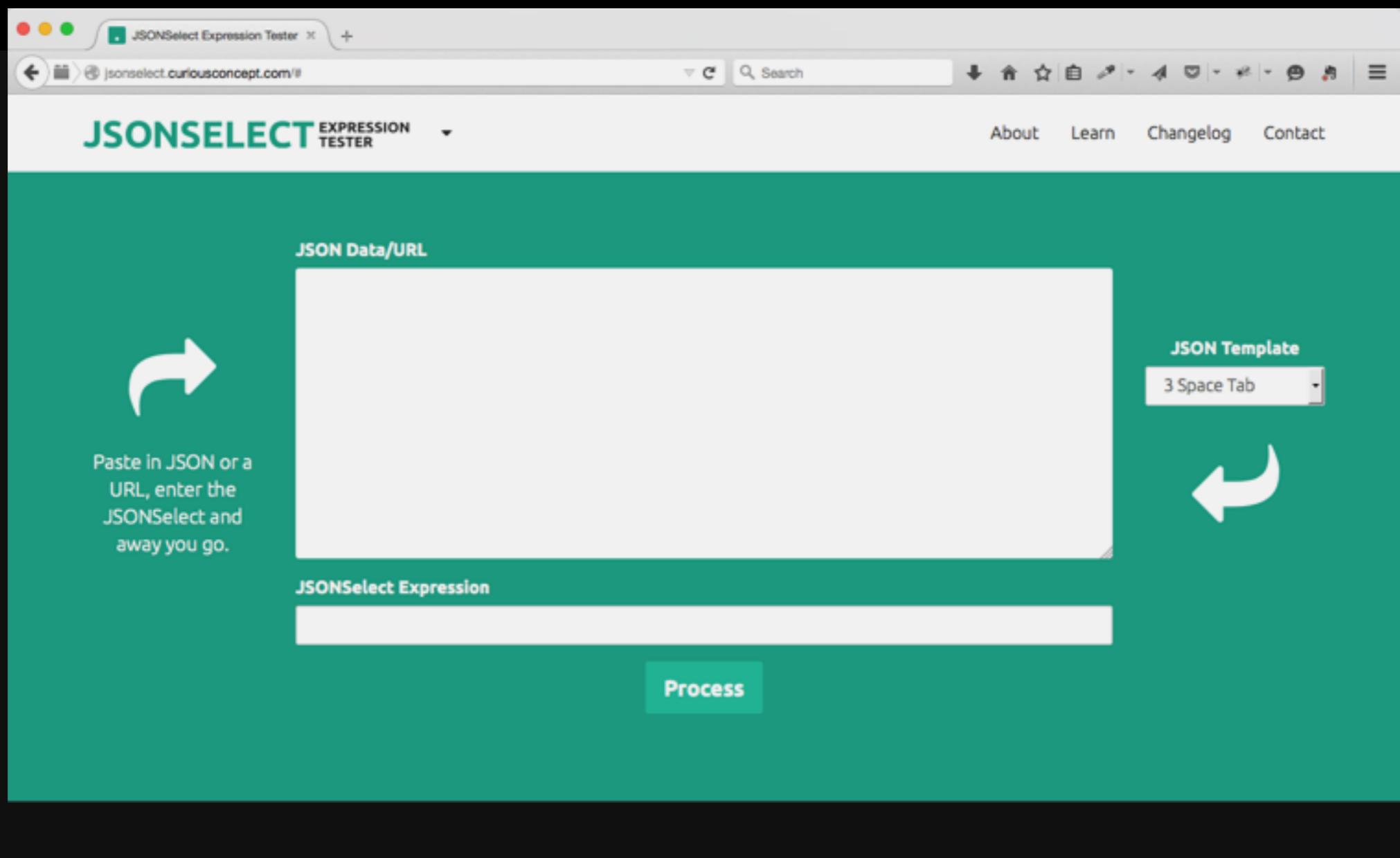
Mindshare	N?
Dev Community	N? Dead Replaced by RQL
Platforms	JS, Node.js
Intuitive	Y
Standard	N

json:select

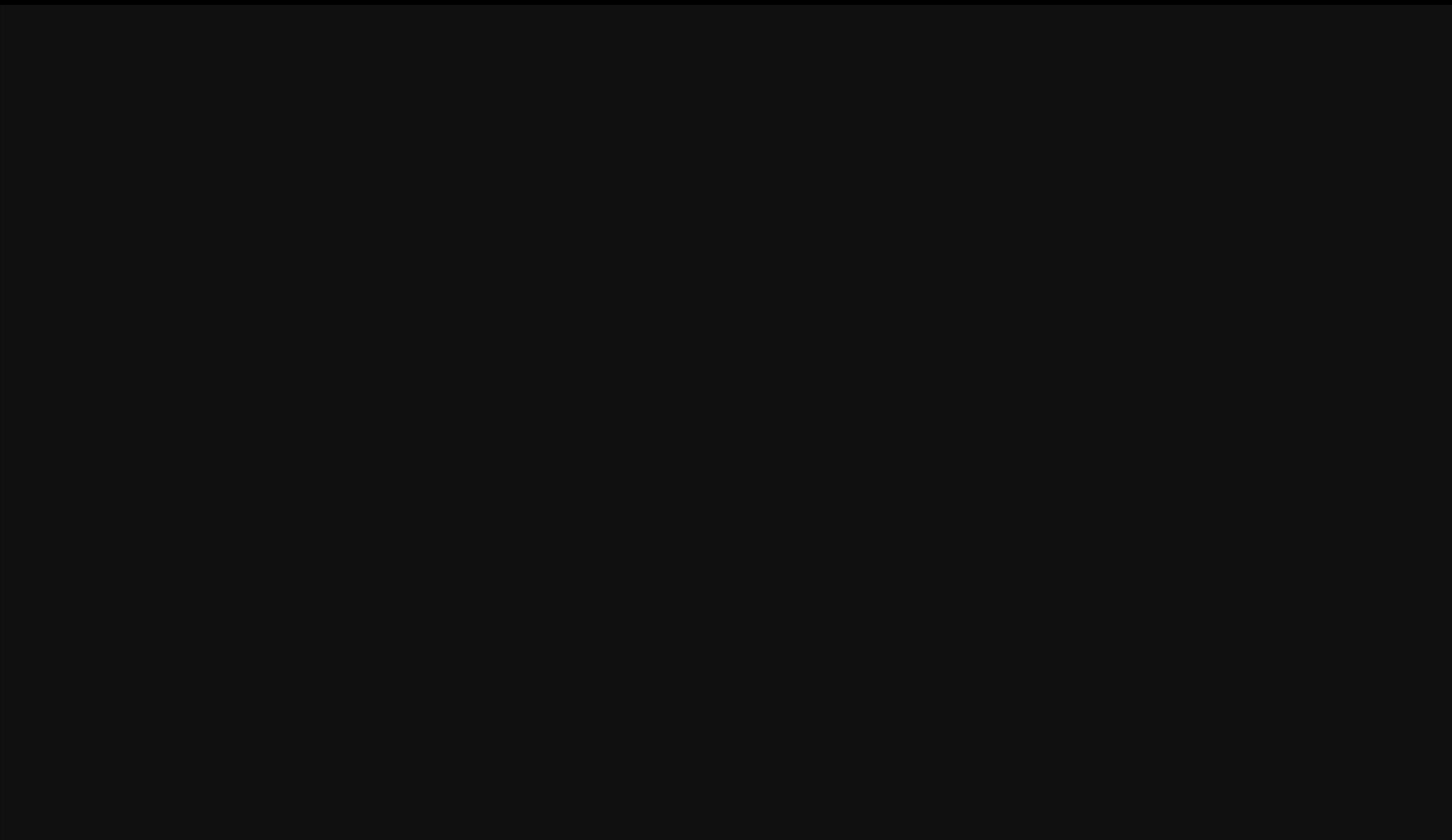
The screenshot shows a dark-themed web browser window displaying the [JSONSelect](https://jsonselect.org/) homepage. The title bar reads "JSONSelect". The address bar shows the URL "https://jsonselect.org/". The page features a large title "json:select()" and a subtitle "CSS-like selectors for JSON.". Below this, there is a section with the text "JSONSelect is an *experimental* selector language for JSON.", followed by "It makes it easy to access data in complex JSON documents.", "It *feels like CSS.*", and "Why not give it a try?". To the right, there is a code block showing a JSON document and a selector query. A "Fork me on GitHub" button is visible in the top right corner of the page area. At the bottom left, there is a link to the GitHub repository: "https://github.com/lloyd/JSONSelect".

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

json:select Expression Tester



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  });
```

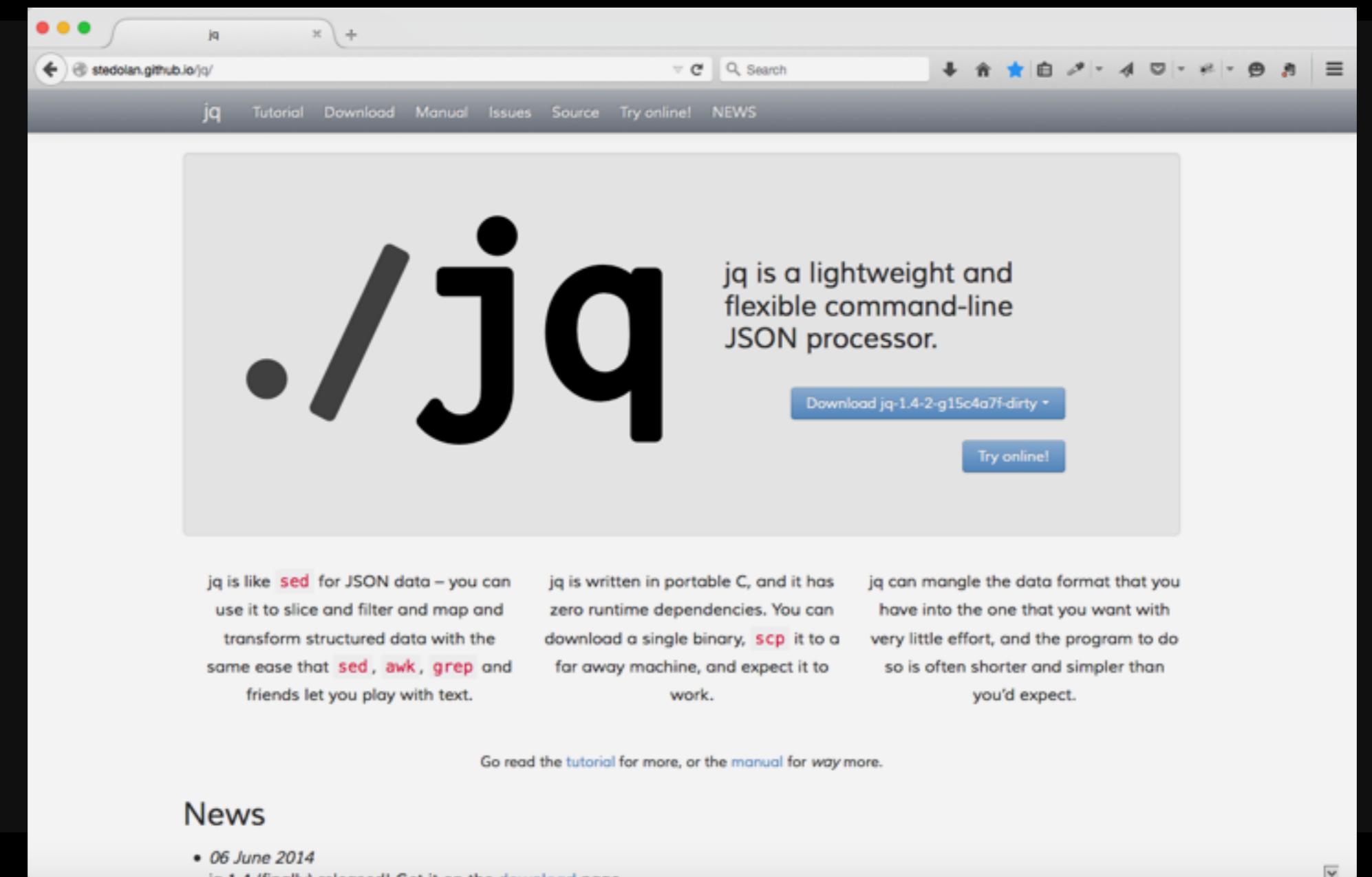
json:select Scorecard

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



A screenshot of a web browser 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, a text box contains the tagline: 'jq is a lightweight and flexible command-line JSON processor.' Below the tagline 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 explains that jq is like `sed` for JSON data. The second column states that jq is written in portable C and has zero runtime dependencies. The third column notes that jq can mangle data formats. At the bottom, there's a link to the tutorial and manual, and a 'News' section with a single item about the release of version 1.4.

jq

stedolan.github.io/jq/

jq Tutorial Download Manual Issues Source Try online! NEWS

./jq

jq is a lightweight and flexible command-line JSON processor.

Download jq-1.4-2-g15c4a7f-dirty

Try online!

jq is like `sed` for JSON data – you can use it to slice and filter and map and transform structured data with the same ease that `sed`, `awk`, `grep` and friends let you play with text.

jq is written in portable C, and it has zero runtime dependencies. You can download a single binary, `scp` it to a far away machine, and expect it to work.

jq can mangle the data format that you have into the one that you want with very little effort, and the program to do so is often shorter and simpler than you'd expect.

Go read the [tutorial](#) for more, or the [manual](#) for way more.

News

- 06 June 2014
jq 1.4 (finally) released! Get it on the [download](#) page.

jq play

The screenshot shows the jqplay.org web interface. At the top, there's a browser header with tabs, a search bar, and various icons. Below it is the jqplay logo and a subtext "A playground for jq 1.4".

The main area has two large text input boxes: "Filter" on the left and "Result" on the right. Above the Result box are several checkboxes: "Compact Output", "Null Input", "Raw Input", "Raw Output", and "Slurp".

On the left, under "JSON", there's a code editor window containing the number "1".

At the bottom, there's a "Cheatsheet" section with a note: "Click on the icons (≡) in the table below to see examples." It contains four rows of examples:

Icon	Description	Icon	Description
·	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 }'
```

jq Scorecard

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

Where Are We?

**JSON Schema
Overview**

1

**JSON Search &
Transform Overview**

4

Core JSON Schema

2

JSON Search

5

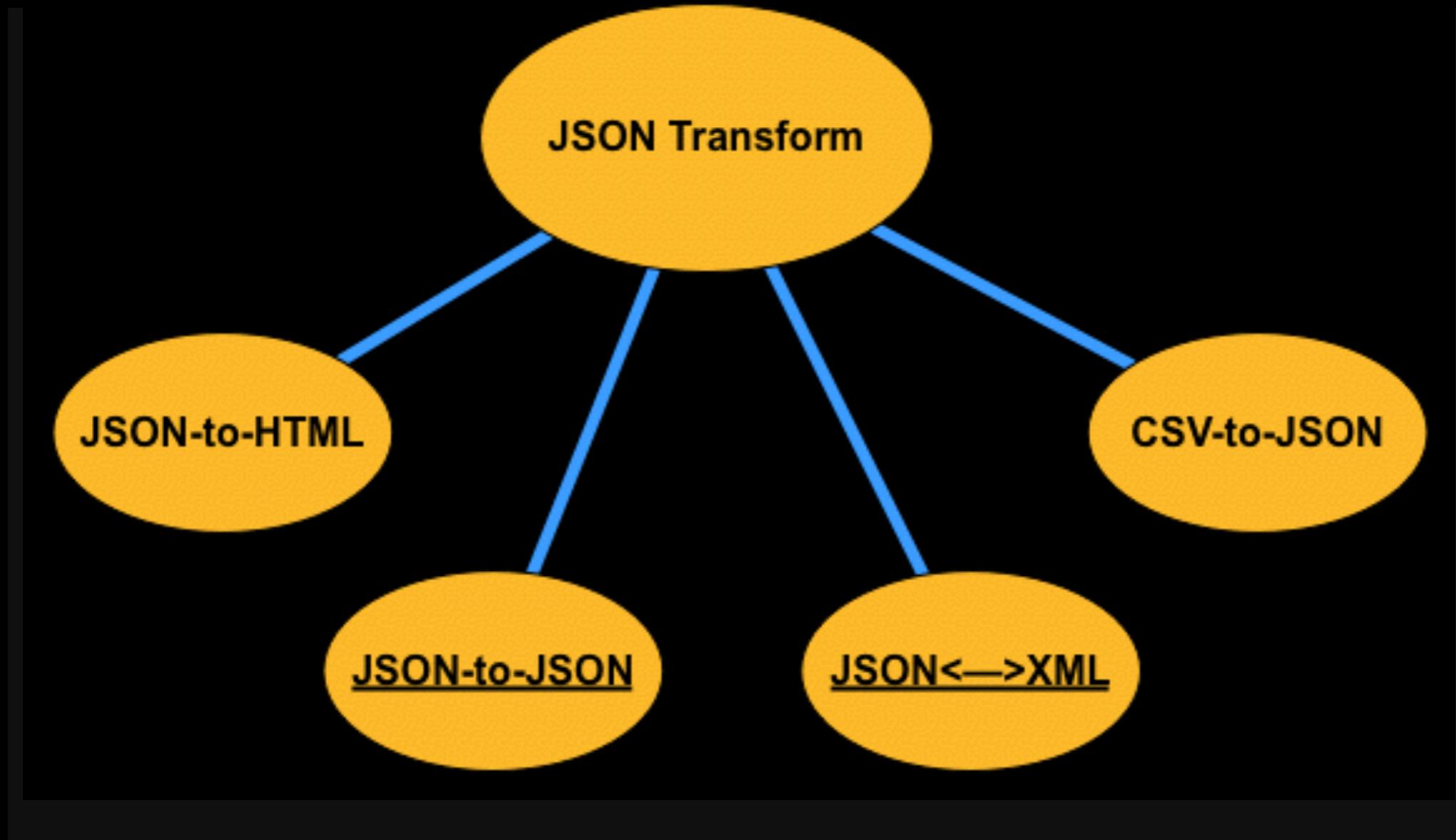
**API Design with
JSON Schema**

3

JSON Transform

6

JSON Transform



JSON-T

The screenshot shows a web browser window with the title "JsonT - Transforming Json". The URL in the address bar is "goessner.net/articles/jsont/". The page content is an article titled "<stefan.goessner/>" under the heading "Mechanik, das Web und der ganze Rest". The main content area contains two sections: "# Transforming JSON" and "# Introducing JSONT".

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 commit history shows an initial commit by 'austundag' 21 days ago, which included 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.

Template Based JSON To JSON Transformation

2 commits · 1 branch · 0 releases · 2 contributors

branch: master · jsonapter / +

initial commit

austundag authored 21 days ago

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

Watch 4 · Star 1 · Fork 0

Code Issues Pull requests

Pulse Graphs

HTTPS clone URL: <https://github.com/amida-tech/jsonapter>

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

Clone in Desktop · Download ZIP

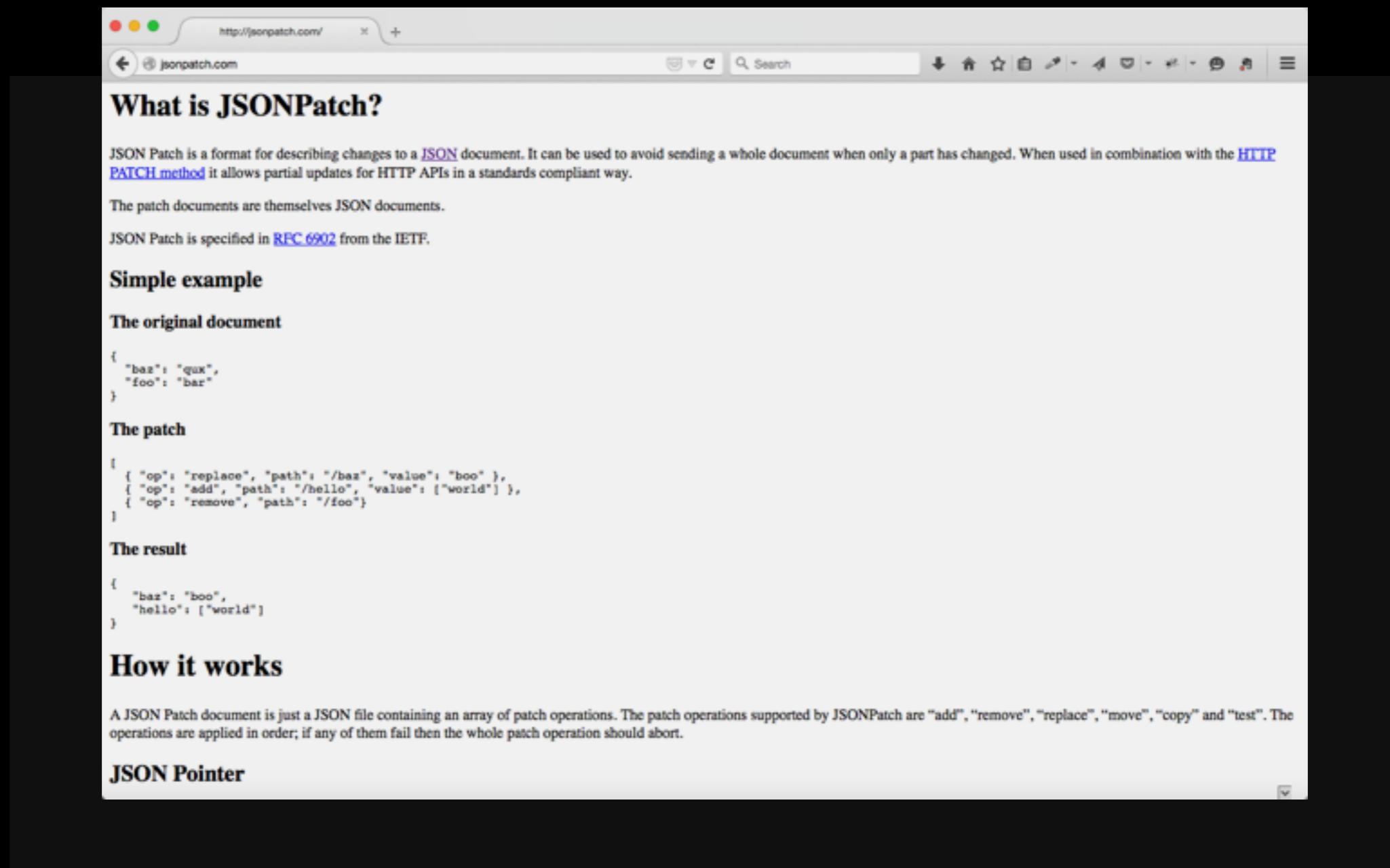
jsonapter Test

```
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\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 // Output:
37 // {
38 //   "name": "John Doe",
39 //   "email": "john.doe@example.com",
40 //   "about": "John Doe is a software engineer at Google."}
```

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

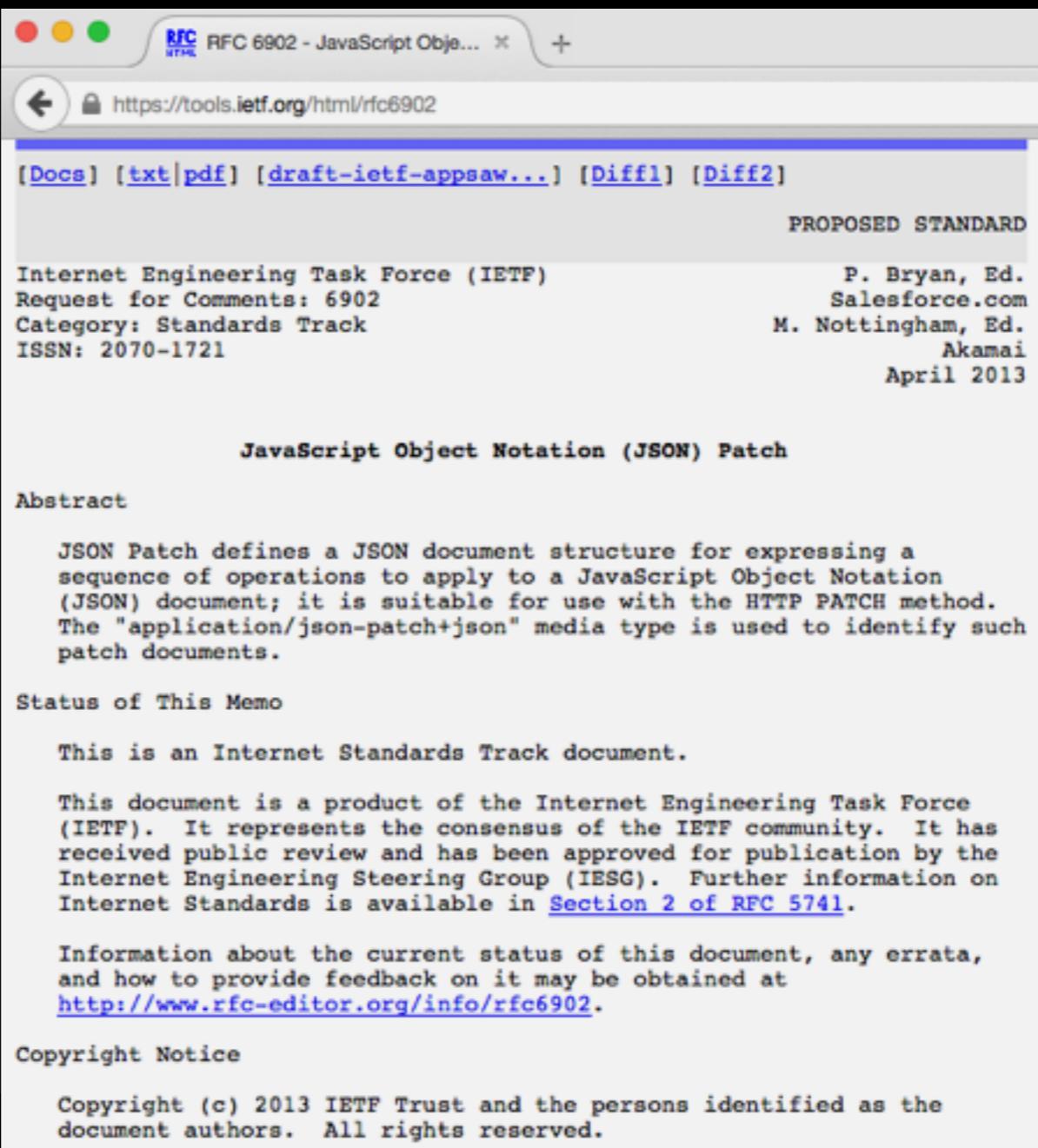
```
{ "baz": "boo", "hello": ["world"] }
```

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 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". The page content includes the RFC header with authors P. Bryan and M. Nottingham, and the date April 2013. It also includes the title "JavaScript Object Notation (JSON) Patch", an abstract, status information, copyright notice, and a link to the IESG status section.

[\[Docs\]](#) [\[txt|pdf\]](#) [\[draft-ietf-appsaw...\]](#) [\[Diff1\]](#) [\[Diff2\]](#)

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" ]  
}
```

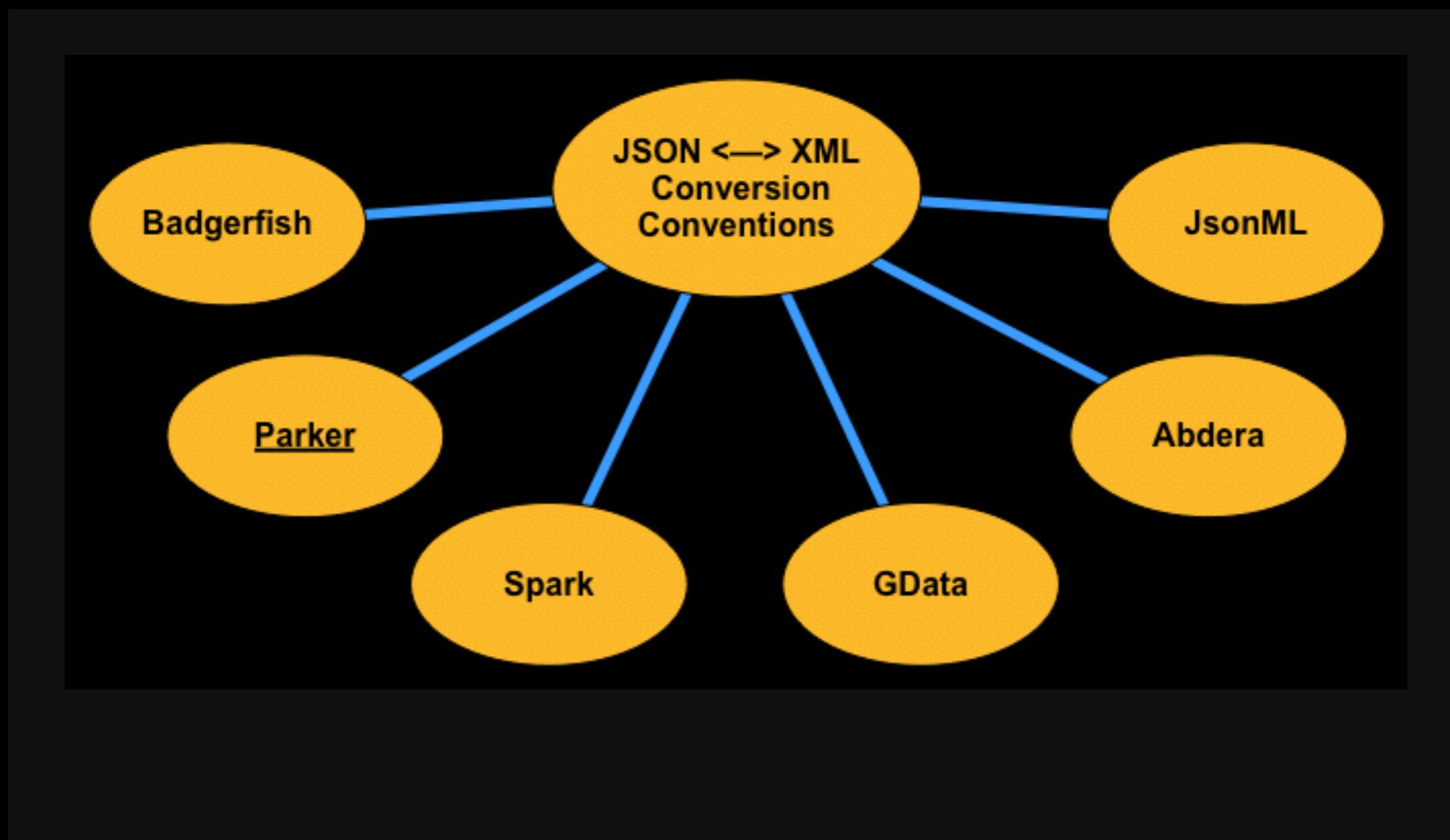
JSON Patch Test

```
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 });
```

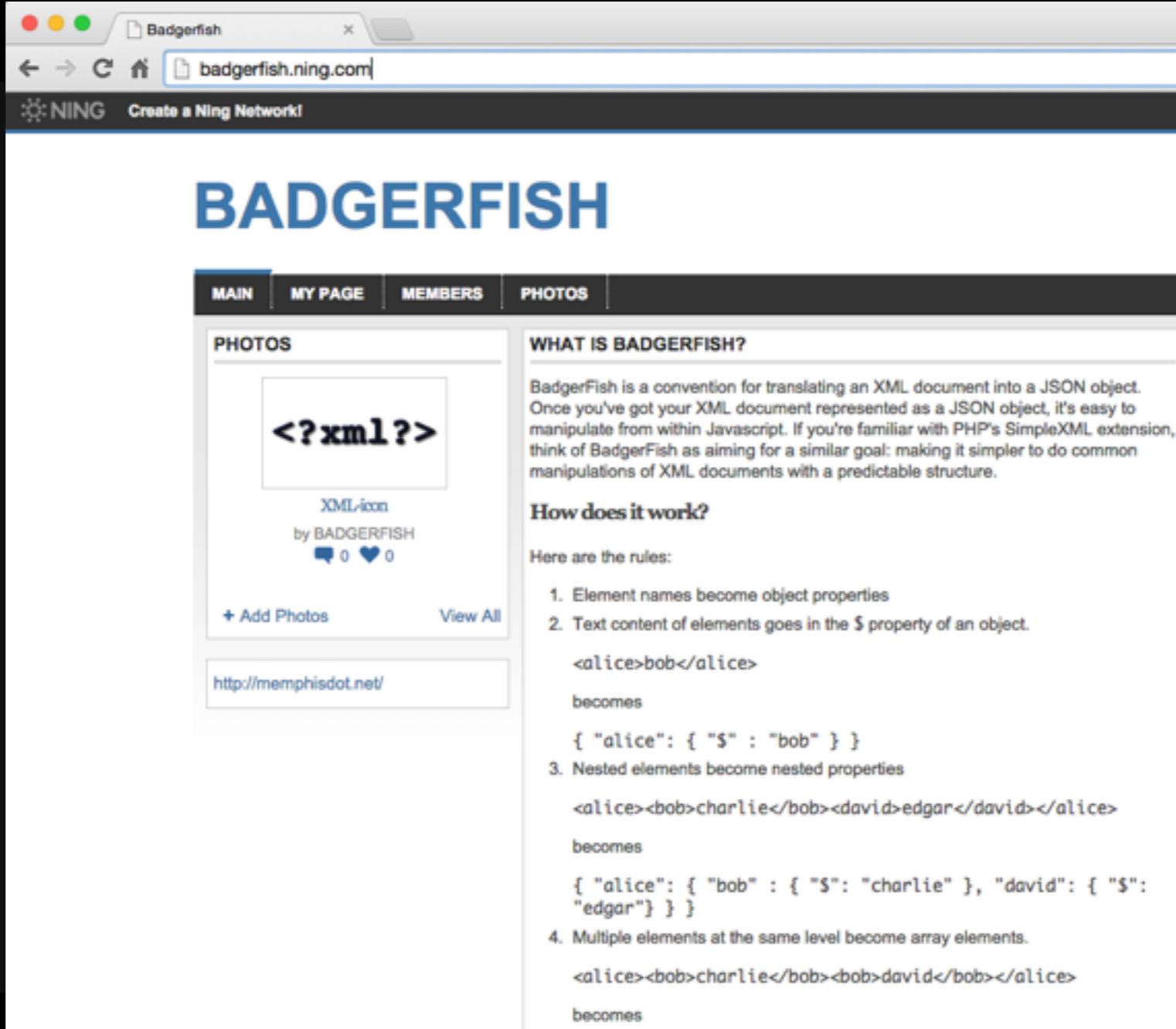
JSON Patch Scorecard

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

JSON - XML Conversion



Badgerfish Convention



The screenshot shows the homepage of the "BADGERFISH" Ning network. The page has a dark header with the Ning logo and a "Create a Ning Network!" button. Below the header, the word "BADGERFISH" is prominently displayed in large blue letters. A navigation bar with tabs for "MAIN", "MY PAGE", "MEMBERS", and "PHOTOS" is visible. The "PHOTOS" tab is active, showing a single photo thumbnail for a file named "<?xml?>". The photo details show it was uploaded by BADGERFISH, has 0 comments, and 0 likes. Below the photo, there's a link to "http://memphisdot.net/". To the right of the photo, a section titled "WHAT IS BADGERFISH?" provides a brief explanation of the convention. It states that BadgerFish is a convention for translating XML documents into JSON objects, making them easier to manipulate from JavaScript. It compares this to PHP's SimpleXML extension. The "How does it work?" section lists four 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 web browser window displaying the MDN page for JXON's Parker Convention. The URL is https://developer.mozilla.org/en-US/docs/JXON/The_Parker_Convention. The page content discusses the Parker Convention, its history, and its purpose in regulating JSON conversion from XML. It includes a note about the official algorithm for HTML5 microdata conversion and examples of how XML elements are converted into JSON objects. A sidebar on the right lists various articles related to JXON and its usage.

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

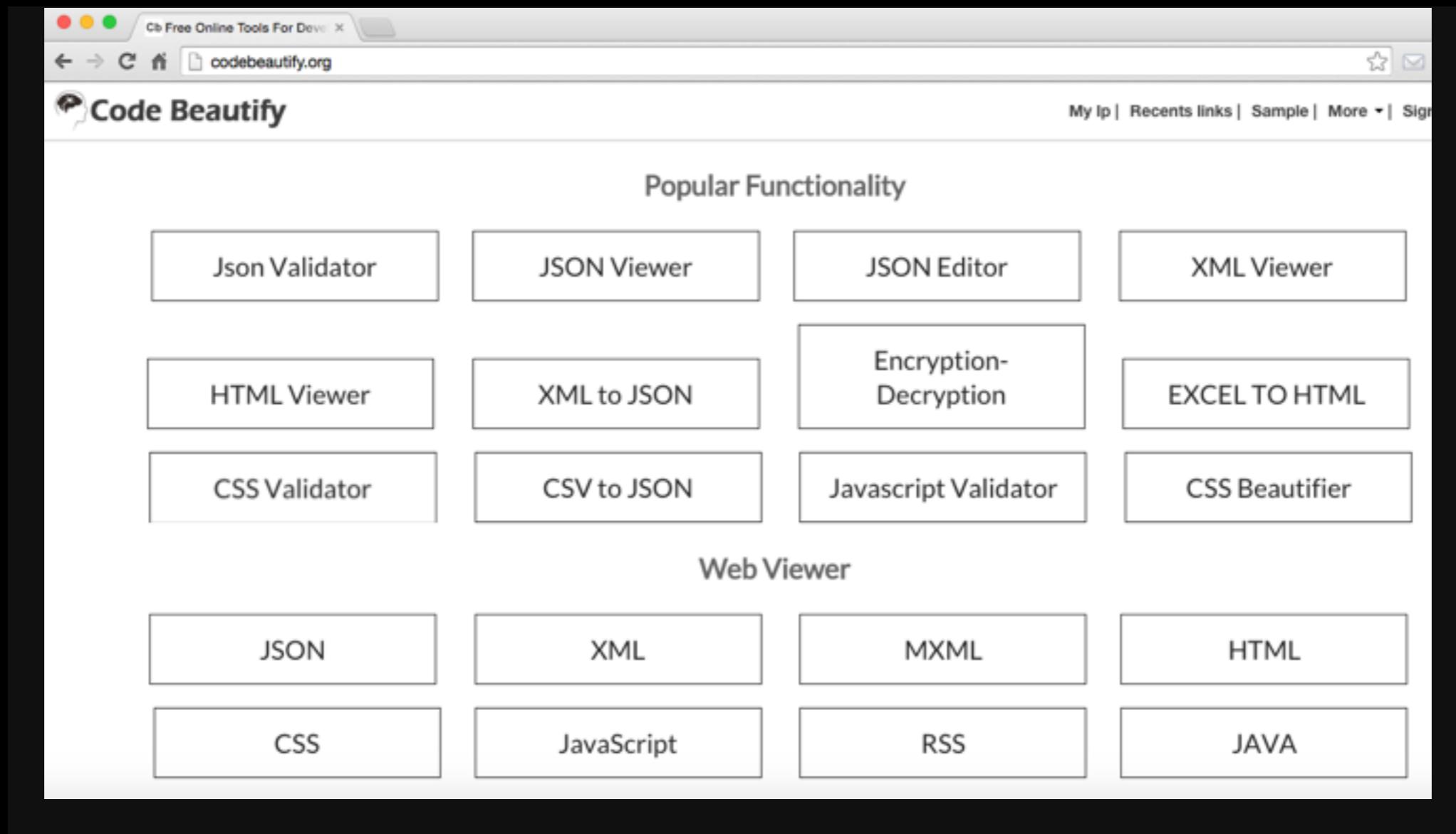
```
2 var expect = require('chai').expect;
3 var fs = require('fs');
4 var jxon = require('jxon');
5 var jsonfile = require('jsonfile');
6
7 describe('jxon', function() {
8     describe('stringToJs', function() {
9         it('should transform XML to JSON', function(done) {
10             var xmlFileName = './data/speaker.xml';
11
12             fs.readFile(xmlFileName, 'utf8', function (err, xmlData) {
13                 if (!err) {
14                     console.log('\n\n\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.jsToString(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

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/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

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/

<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

[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/>