

{json:api}

@speaktochris

@NexmoDev

chris-guzman.com/jsonapi.pdf





Nexmo Developer

Everything you need to build connected applications with Nexmo

The screenshot shows the Nexmo Developer website with a blue header and a white content area. The header includes the Nexmo logo, a search bar with the URL https://developer.nexmo.com, and navigation links for Nexmo.com, Support, Sign In, and Try it free. Below the header are six service cards arranged in two rows of three:

- SMS**: Send and receive SMS. Includes Overview, Guides, Building Blocks, Tutorials, and API Reference.
- Voice**: Programmable Voice. Includes Overview, Guides, Building Blocks, Tutorials, and API Reference.
- Verify**: User Authentication Verify. Includes Overview, Guides, Building Blocks, Tutorials, and API Reference.
- Number insight**: User Identity Number insight. Includes Overview.
- Account**: Management Account. Includes Overview.

At the bottom left, there is a note: "Better APIs with JSON API - @speaktochris - chris-guzman.com/jsonapi.pdf".

Ruby & Android



{json:api}

A specification for building APIs in
JSON

{json:api}

A specification for building APIs in

JSON

(duh)

{ json:api }

{ json:api }

```
{ "json": "api" }
```



JSON:API
@jsonapi

Following



How would you like us to write our name?
(Please consider this to be case-insensitive)

17% JSONAPI

29% JSON API

47% JSON:API

7% Other (please reply)

349 votes • Final results

1:32 PM - 18 Sep 2017

12 Retweets 7 Likes



9



12



7



Better APIs with JSON:API - @speaktochris - chris-guzman.com/jsonapi.pdf

Why?



Better APIs with JSON:API - @speaktochris - chris-guzman.com/jsonapi.pdf



HOW STANDARDS PROLIFERATE:

(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC)

SITUATION:
THERE ARE
14 COMPETING
STANDARDS.

14?! RIDICULOUS!
WE NEED TO DEVELOP
ONE UNIVERSAL STANDARD
THAT COVERS EVERYONE'S
USE CASES.



Soon:

SITUATION:
THERE ARE
15 COMPETING
STANDARDS.

- Minimize requests & data between clients & servers
 - Consistency, popular among consultancies
 - Always backwards compatible
 - Stable 1.0, *not dead*

Is json-api development... dead? #1190

Issues 91 Pull requests 25 Projects 0 Insights

New issue

approxit commented on Jul 12 • edited

Those, who implemented JSON API in their Web API, know that this specification totally overkills all its competitors since end of may 2015. This concept fixes tons of use cases and has many implementations.

No, really

But... Now, 2 years after 1.0 release, nothing important happened. 1.1 release is still "upcoming". Vast usage catches areas uncovered by spec which issues are without much response (relationship pagination, resource actions, sparse fields on meta fields, O(1) includes). Proposals are hanging without any attention.

The only thing that is updated is... implementation list PRs.
That's very sad.

It's only me, or this project is really starving?

In my opinion JSON-API is the best thing that could happen to JSON based Web API format. Right now I feel that we already have 2 years loss of its development. How can we push more life to it?

5

remmeier commented on Jul 12

Contributor +

Unwatch 194 Unstar 4,728 Fork 572

Assignees
None yet

Labels
None yet

Projects
None yet

Milestone
No milestone

Notifications
Unsubscribe

You're receiving notifications because you're subscribed to this repository.

6 participants

History



{ json:api }

A SPECIFICATION FOR BUILDING APIs IN JSON.

Spec

- meta: a meta object that contains non-standard meta-information.
 - errors: an array of error objects
 - OR -
 - data: the document's “primary data”

Meta

```
{  
  "meta": {  
    "count": "42",  
    "copyright": "Copyright 2015 Example Corp.",  
    "authors": [  
      "Yehuda Katz",  
      "Steve Klabnik",  
      "Dan Gebhardt",  
      "Tyler Kellen"  
    ]  
  }  
}
```

Errors

```
{  
  "errors": [  
    {"id": "UUID",  
     "status": "418",  
     "code": "42",  
     "title": "You're a teapot",  
     "detail": "The object is not short nor stout",  
     "links": {  
       "about": "https://httpstatuses.com/418"  
     },  
     "meta": {  
       "email": "nobody@example.com"  
     }  
   ]  
}
```

Data

GET /articles

Accept: application/vnd.api+json

GET /articles/1

Accept: application/vnd.api+json

GET /articles/1/author

Accept: application/vnd.api+json

```
{  
  "data": {  
    "type": "articles",  
    "id": "1",  
    "attributes": {  
      "title": "Rails is Omakase"  
    },  
    "relationships": {},  
    "links": {  
      "self": "http://example.com/articles/1"  
    },  
  }  
}
```

Links

```
{"links": {  
    "self": "/articles/1",  
    "href": "http://example.com/articles/1"  
},  
}  
}
```

Relationships

```
{  
  "relationships": {  
    "author": {  
      "data": {  
        "type": "people",  
        "id": "9"  
      },  
      "links": {  
        "related": "/articles/1/author",  
        "self": "/articles/1/relationships/author"  
      }  
    }  
  }  
}
```

#Request comments with an article

GET /articles/1?include=comments

#Request authors & photos with an article

GET /articles/1?include=authors,photos

#Request comments as well as the author of each of those comments

GET /articles/1?include=comments.author

```
{ "data": [],  
  "included": [  
    {  
      "type": "people",  
      "id": "11",  
      "attributes": {  
        "name": "Oli Rumbelow",  
        "email": "oliver.rumbelow@example.com"  
      }  
    },  
    {  
      "type": "photos",  
      "id": "45",  
      "attributes": {  
        "url": "http://www.example.com/foobar",  
        "height": 1080,  
        "width": 1920  
      }  
    }]  
  ]}
```

#Request authors with id of 1 or 2

GET /authors/?filter[id]=1,2

#Request authors who are alive

GET /authors?filter[alive]=true

#Request comments published after a date

GET /comments?filter[published_after]=1990-01-01

Note: JSON:API is agnostic about how the filter query parameter can be used.

#Request only the title, & status fields

GET /articles?fields[articles]=title,status

#Request only the article's title & body fields

#plus the name of the author

GET /articles/1?&fields[articles]=title,body\
include=author&fields[people]=name

```
{"data": [  
  {  
    "type": "articles",  
    "id": "3",  
    "attributes": {  
      "title": "NodeJS Best Practices",  
      "status": "published"  
    }  
  }  
]}  
}]}
```

#Sort articles alphabetically by title

GET /articles/?sort=title

#Sort authors by name, then life

GET /authors?sort=name,alive

#Sort authors by reverse alphabetical order, then life

GET /authors?sort=-name,alive

Note: JSON:API is agnostic about how the sort query parameter can be used.

```
#Get the third of 13 pages, one article at a time
```

```
GET /articles?page[number]=3&page[size]=1
```

```
"links": {  
  "self": "http://example.com/articles?page[number]=3&page[size]=1",  
  "first": "http://example.com/articles?page[number]=1&page[size]=1",  
  "prev": "http://example.com/articles?page[number]=2&page[size]=1",  
  "next": "http://example.com/articles?page[number]=4&page[size]=1",  
  "last": "http://example.com/articles?page[number]=13&page[size]=1"  
}
```

Note: JSON:API is agnostic about how the page query parameter can be used.

```
GET /articles?include=comments,author  
&fields[people]=first_name,last_name  
&sort=-date
```

```
GET /articles?include=comments,author  
&fields[people]=first_name,last_name  
&sort=-date
```

- » fetch all articles with their associated comments and authors
- » Only be return the first and last names
- » Sorted by date, most recent first

POST /photos

```
{"data": {  
  "type": "photos",  
  "attributes": {  
    "title": "Ember Hamster",  
    "src": "http://example.com/images/productivity.png"  
  },  
  "relationships": {  
    "photographer": {  
      "data": { "type": "people", "id": "9" }  
    }  
  }  
}
```

PATCH /articles/1

```
{"data": {  
  "type": "articles",  
  "id": "1",  
  "attributes": {  
    "title": "To TDD or Not",  
    "text": "TLDR; It's complicated..."  
  }  
}
```

PATCH /articles/1/relationships/author

```
{  
  "data": {  
    "type": "people", "id": "12"  
  }  
}
```

PATCH /articles/1/relationships/tags

```
{  
  "data": [  
    { "type": "tags", "id": "2" },  
    { "type": "tags", "id": "3" }  
  ]  
}
```

```
PATCH /articles/1/relationships/author
```

```
{"data": null}
```

```
PATCH /articles/1/relationships/tags
```

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

```
DELETE /photos/1
```

```
#empty body
```

JSON:API

- » Can take advantage of caching
- » built on top of REST

vs GraphQL

- » Backed by heavier players
- » GraphiQL UI



Tom Dale

@tomdale

Follow



GraphQL will replace REST in the same way
MongoDB replaced PostgreSQL.

11:40 AM - 14 Oct 2016

125 Retweets 155 Likes



11

125



155



A photograph of a group of people at a Nasdaq event. In the foreground, a man in a blue plaid shirt is smiling and looking towards the camera. Behind him, several other people are clapping and smiling. Confetti is falling from above, creating a festive atmosphere. The background is a green wall with the word "Nasdaq" repeated in white.

Ok, maybe not the
best analogy

cerebris/jsonapi-resources: A JSON API — Implementations

Chris

← → ⌂ jsonapi.org/implementations/#client-libraries-javascript

Implementations

Implementations

Client libraries

- JavaScript
- TypeScript
- iOS
- Ruby
- PHP
- Dart
- Perl
- Java
- Android
- R
- Elm
- .NET
- Python

Server libraries

- Swift
- PHP
- Node.js
- Ruby
- Python
- Go
- .NET
- Java
- Scala
- Elixir
- Haskell
- Perl
- Vala
- Rust

Examples

Related Tools

- Playground
- Ruby
- Node.js

Note: This specification marked 1.0 on May 29th, 2015. The implementations below have not been verified for compliance, but a test suite is now being assembled to vet them.

Client libraries

JavaScript

- [ember-data](#) is one of the original exemplar implementations. There is now an [official adapter](#) to support json-api.
- [backbone-jsonapi](#) is a Backbone adapter for JSON API. Supports fetching Models & Collections from a JSON API source.
- [backbone-relational-jsonapi](#) is a parsing layer for Backbone.Relational. Entities specified in JSON API are automatically parsed to be injected into Backbone.Relational relations.
- [orbit.js](#) is a standalone library for coordinating access to data sources and keeping their contents synchronized. Orbit's Common Library includes [JSONAPISource](#) for accessing JSON API servers. Orbit can be used independently or with Ember.js through the [ember-orbit](#) integration library.
- [YAYSON](#) is an isomorphic library for serializing and reading JSON API data. Extend it to fit your models or just use it with plain objects.
- [Ember JSON API Resources](#) is an [Ember CLI](#) Addon for a lightweight solution for data persistence in an [Ember.js](#) application.
- [hapi-json-api](#) Plugin for the hapi framework; enforces Accept/Content-type rules and rewrites Boom errors to be spec compliant.
- [jsonapi-datastore](#) is a lightweight standalone library for reading, serializing, and synchronizing relational JSON API data.
- [json-api-store](#) A lightweight JavaScript library for using JSON API in the browser.
- [superagent-jsonapi](#) A really lightweight (50 lines) JSON-API client addon for superagent, the

jsonapi.org/implementations/#client-libraries-javascript

holidayextras/jsonapi-server: A config driven NodeJS framework implementing json:api and GraphQL

Code Issues 35 Pull requests 15 Projects 0 Wiki Insights

578 commits 9 branches 53 releases 21 contributors MIT

Branch: master New pull request Create new file Upload files Find file Clone or download

pmcnr-hx Merge pull request #326 from holidayextras/greenkeeper/initial ... Latest commit 8a50805 2 days ago

documentation	Correct spelling of linkage	4 months ago
example	Expose some more Swagger 2.0 configuration	3 months ago
lib	Fix error message for .many	a month ago
test	Cleanup scripts and use nyc + coveralls for coverage reporting.	3 days ago
.editorconfig	v0.6.0	2 years ago
.eslintrc	Disallow use of `var` in favor of `let` or `const`	7 months ago
.gitignore	Cleanup scripts and use nyc + coveralls for coverage reporting.	3 days ago
.jshintrc	jshintrc file for complexity analysis	2 years ago
.npmignore	Update ignore files for correct exclusions.	a month ago
.travis.yml	Greenkeeper lockfile maintenance.	2 days ago
CHANGELOG.md	Update CHANGELOG and bump package version for release.	3 days ago
CONTRIBUTING.md	Update documentation on where to find the coverage reports.	8 months ago
LICENSE	v0.6.0	2 years ago

Chris

[j] JSON API—Latest Specification [j] JSON API—Implementations [j] doga/jsonapi_for_rails: A Rails plugin for providing JSON API compliant APIs with very little coding.

GitHub, Inc. [US] | https://github.com/doga/jsonapi_for_rails

This repository Search Pull requests Issues Marketplace Explore Watch 4 Star 19 Fork 3

doga / jsonapi_for_rails

Code Issues 1 Pull requests 1 Projects 0 Insights

A Rails plugin for providing JSON API compliant APIs with very little coding. <http://jsonapi.org/format/> <http://rubyonrails.org/>

71 commits 1 branch 8 releases 2 contributors MIT

Branch: master New pull request Create new file Upload files Find file Clone or download

doga Merge pull request #4 from pschutz93/patch-1 ... Latest commit d98d46f 6 days ago

File	Description	Date
bin	add tests	2 years ago
certs	add digital signature, gem now requires rails >= 4	2 years ago
lib	fix bug when updating a to-many relationship	2 years ago
test	fix bug when updating a to-many relationship	2 years ago
.travis.yml	add travis file	2 years ago
Gemfile	add travis file	2 years ago
Gemfile.lock	Gemfile.lock update	2 years ago
MIT-LICENSE	pristine Rails plugin	2 years ago
README.md	Update README.md	7 days ago
Rakefile	HTTP response contains an errors document if record creation/update f...	2 years ago
jsonapi_for_rails.gemspec	update README	2 years ago
README.md		

{json:api}

@speaktochris

@NexmoDev

chris-guzman.com/jsonapi.pdf