Advanced Web Services with JSON API

HOWDY!

I am Mateu

I am here because I am a decoupling nerd

You can find me at @e0ipso

You will learn about...

JSON API

What is it?

Why use it?

Drupal module

What's the status?

What are the limitations?

How does it relates to REST in core?

Outstanding problems

Still looking for solutions!

{json:api} paints your bike shed

```
// ...
{
  "type": "articles",
  "id": "1",
  "attributes": {
    "title": "Rails is Omakase"
  },
  "relationships": {
    "author": {
      "links": {
        "self": "/articles/1/relationships/author",
        "related": "/articles/1/author"
      },
      "data": { "type": "people", "id": "9" }
    }
  }
}
...
```

Defines:

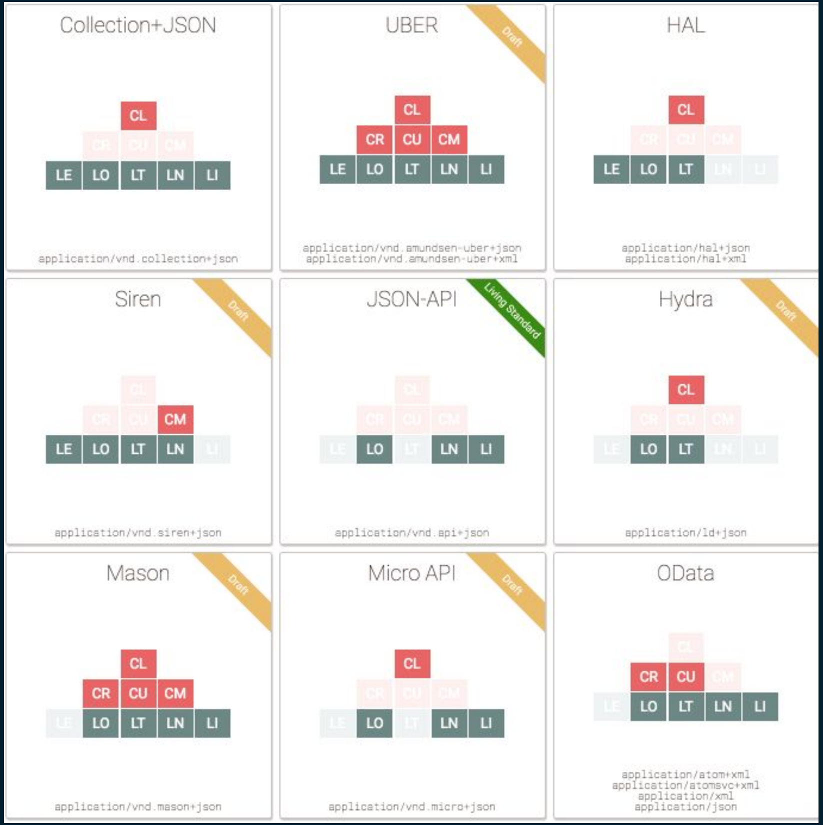
- Transport
- Interaction

```
GET /articles/1/relationships/comments HTTP/1.1
Accept: application/vnd.api+json
```

Creative Commons specification

Strongly driven by FE & UX experts

Steve Klabnik, Yehuda Katz, Dan Gebhardt, Tyler Kellen, Ethan Resnick



Why this one?

Since there are others, and a HAL implementation is already in core. And GraphQL in contrib.

141 repos

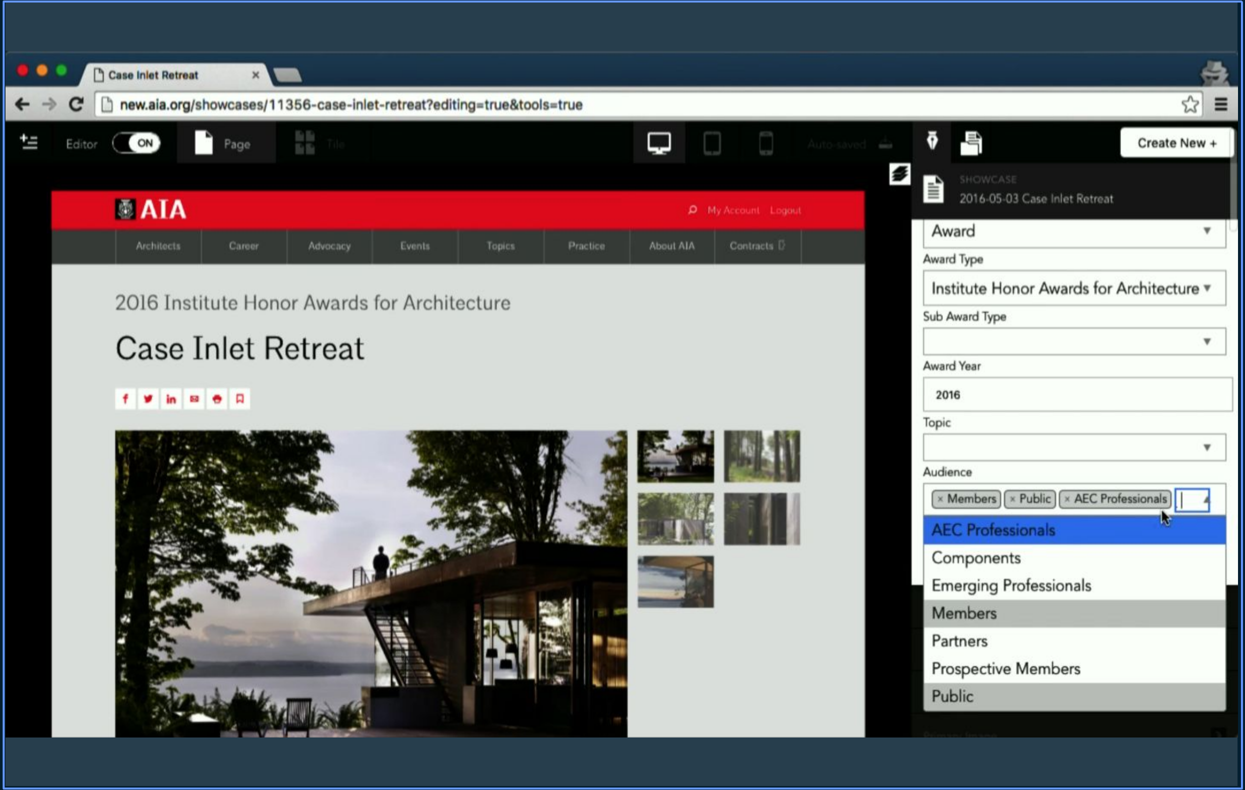
That’s a lot of traction

18 languages

And a lot of range

Client & Server

Total success!


Case Inlet Retreat

new.aia.org/showcases/11356-case-inlet-retreat?editing=true&tools=true

Editor ON Page

Create New +

SHOWCASE
2016-05-03 Case Inlet Retreat

AIA My Account Logout

Architects Career Advocacy Events Topics Practice About AIA Contracts

2016 Institute Honor Awards for Architecture

Case Inlet Retreat

Award

Award Type: Institute Honor Awards for Architecture

Sub Award Type:

Award Year: 2016

Topic:

Audience: × Members × Public × AEC Professionals

- AEC Professionals
- Components
- Emerging Professionals
- Members
- Partners
- Prospective Members
- Public

With a highlight on its flexibility

Stays neutral on implementation details and gives you space. Also provides extension system.

RTFM

HOW DID I GET HERE?

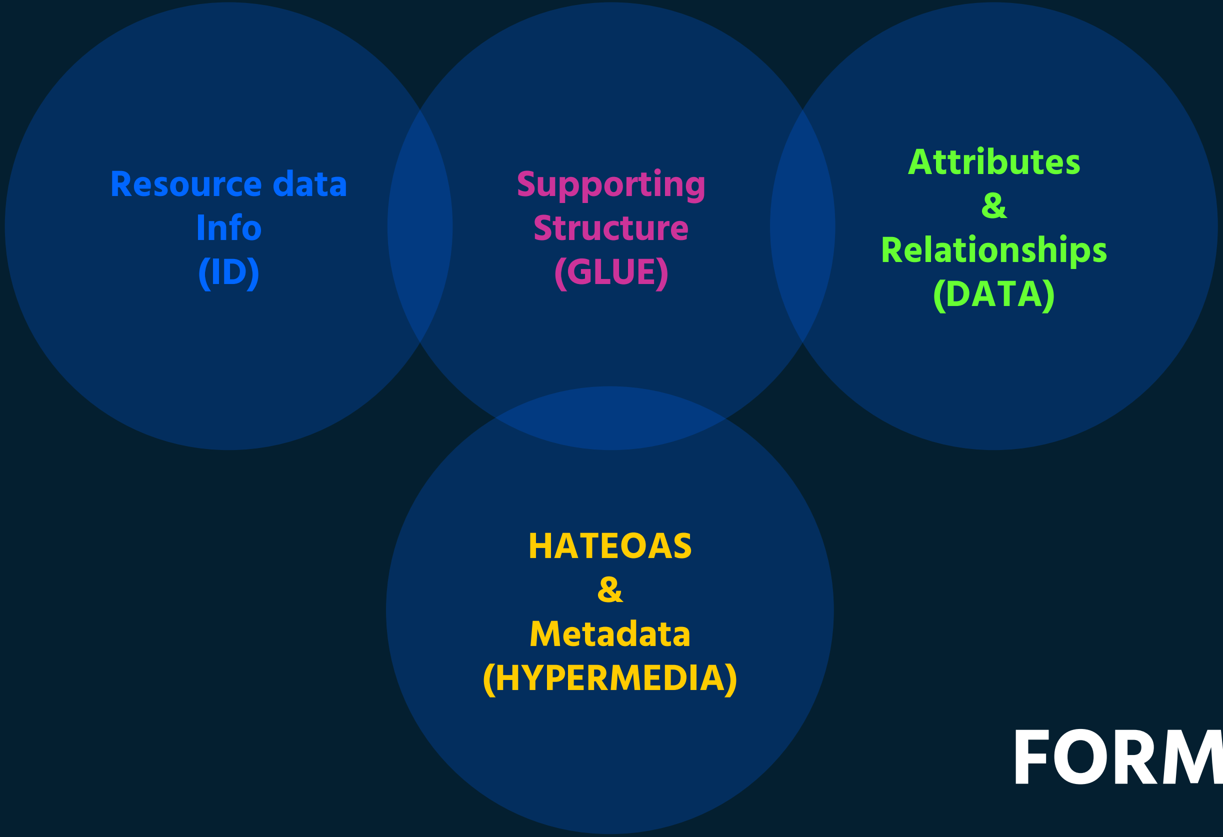
Response to the typical problems

- Multiple round trip requests
- Bloated responses
- Content discovery

They all have known solid solutions!

1.
TRANSPORT FORMAT

The shape of the JSON object


Resource data Info (ID)
Supporting Structure (GLUE)
Attributes & Relationships (DATA)
HATEOAS & Metadata (HYPERMEDIA)

FORMAT

```
{
  "data": {
    "type": "articles",
    "id": "1",
    "attributes": {…},
    "relationships": {…},
  },
  "links": {…},
  "meta": {…}
}
```

FORMAT

```
{
  …
    "attributes": {
      "title": "Drupal 8!",
      "body": "Lorem ipsum"
      …
    },
  …
}
```

FORMAT

```
…
  "relationships": {
      "links": {
          "self": "articles/1/relationships"
      },
      "tags": {
          "data": [{
              "type": "tags",
              "id": "2"
          }]
      }
  …
```

FORMAT

2.
RESOURCE INTERACTION

How do we *get* and *update* data

Uses REST

GET, POST, PUT, PATCH, DELETE, …

Typical request

```
GET /articles HTTP/1.1
Accept: application/vnd.api+json
```

RESPONSE

/api/node/article?_format=api_json

The typical solutions

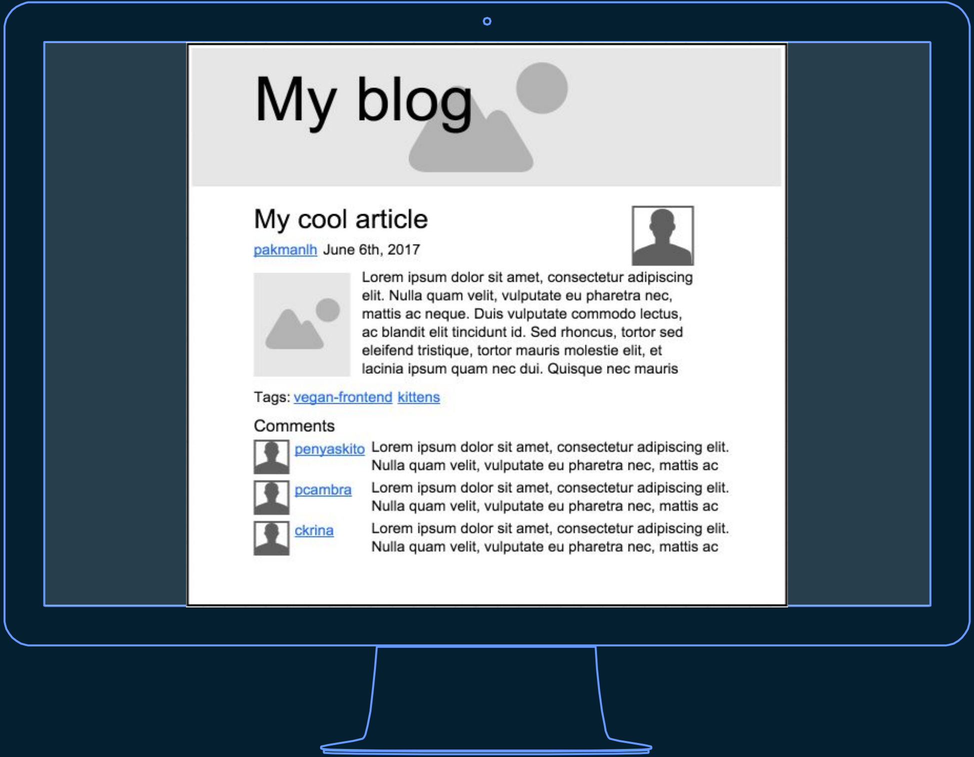
- ⚠ Multiple round trip requests
- ✓ Resource embedding

- ⚠ Bloated responses
- ✓ Sparse fieldsets

- ⚠ Content discovery
- ✓ Collections and filters

EXTREMELY SIMPLE

Your project will have way more stuff than this!



My blog

My cool article

pakmanlh June 6th, 2017

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla quam velit, vulputate eu pharetra nec, mattis ac neque. Duis vulputate commodo lectus, ac blandit elit tincidunt id. Sed rhoncus, tortor sed eleifend tristique, tortor mauris molestie elit, et lacinia ipsum quam nec dui. Quisque nec mauris

Tags: vegan-frontend kittens

Comments

penyaskito Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla quam velit, vulputate eu pharetra nec, mattis ac

pcambra Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla quam velit, vulputate eu pharetra nec, mattis ac

ckrina Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla quam velit, vulputate eu pharetra nec, mattis ac

- **1**: GET articles/12
  - **2**: GET articles/12 => tags/34
  - **3**: GET articles/12 => tags/88
  - **4**: GET articles/12 => users/88
    - **5**: GET articles/12 => users/88 => images/5
  - **6**: GET articles/12/comments
  - **7**: GET articles/12 => comment/2
    - **8**: GET articles/12 => comment/2 => user/8
      - **9**: GET articles/12 => comment/2 => user/8 => image/9
  - **10**: GET articles/12 => comment/7 [...]
    - **11**: GET articles/12 => comment/7 [...]
      - **12**: GET articles/12 => comment/7 [...]
- **MORE!**

```
GET /articles/12?
include=
   author,author.pic,
   tags,
   comment,comment.author,
   comment.author.pic
```

Resource embedding

```
GET /articles/12?
fields[articles]=
   title,
   created
```

Sparse fieldsets

```
…
"attributes": {
  "title": "My article",
  ~~"uuid": "12345-1234-34",~~
  "created": "10-05-2012",
  ~~"status": "1",~~
  ~~"body": {…},~~
  ~~"langcode": "en"~~
}
…
```

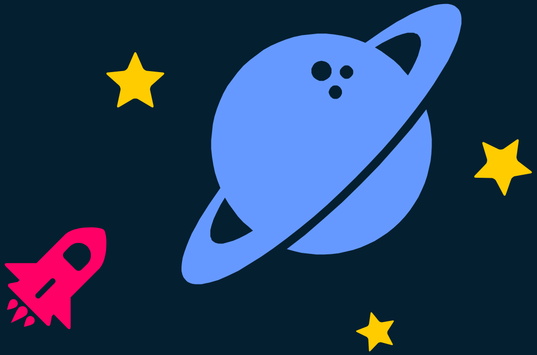
“Give me the cover image and the publication year of all the albums of all the bands having one of the members under 35 currently living in Murcia.

Oh! And while you're at it, output the name of the band and that member as well.”

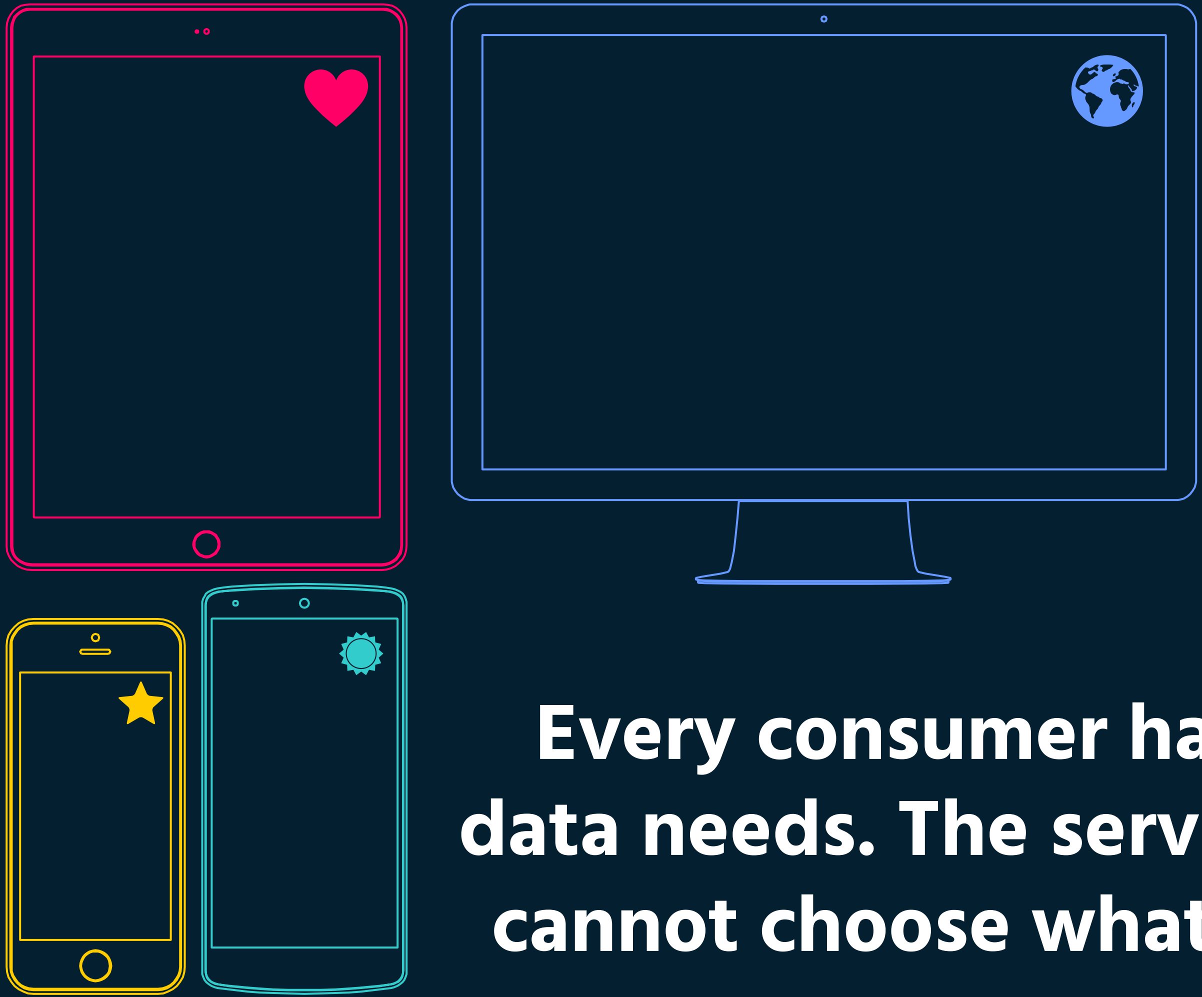
```
GET /bands?
filter[members.city][value]=Murcia&
filter[members.age][value]=35&
filter[members.age][operator]="<="&
include=albums,albums.cover,members&
fields[bands]=name,albums,members&
fields[members]=name&
fields[albums]=publication&
fields[images]=uri
```

Collections and filters

WRITE URL QUERIES



Every **API consumer** requests the resource data it needs. It can be different every time.



Every consumer has different data needs. The server (Drupal) cannot choose what those are.

Every resource 4 *“endpoints”*

1. `/bands/1234`
   - GET, PUT, PATCH, DELETE
2. `/bands`
   - GET, POST
3. `/bands/1234/albums`
   - GET
4. `/bands/1234/relationships/albums`
   - GET, PATCH

3.
PERFORMANCE

How fast is the Drupal module?

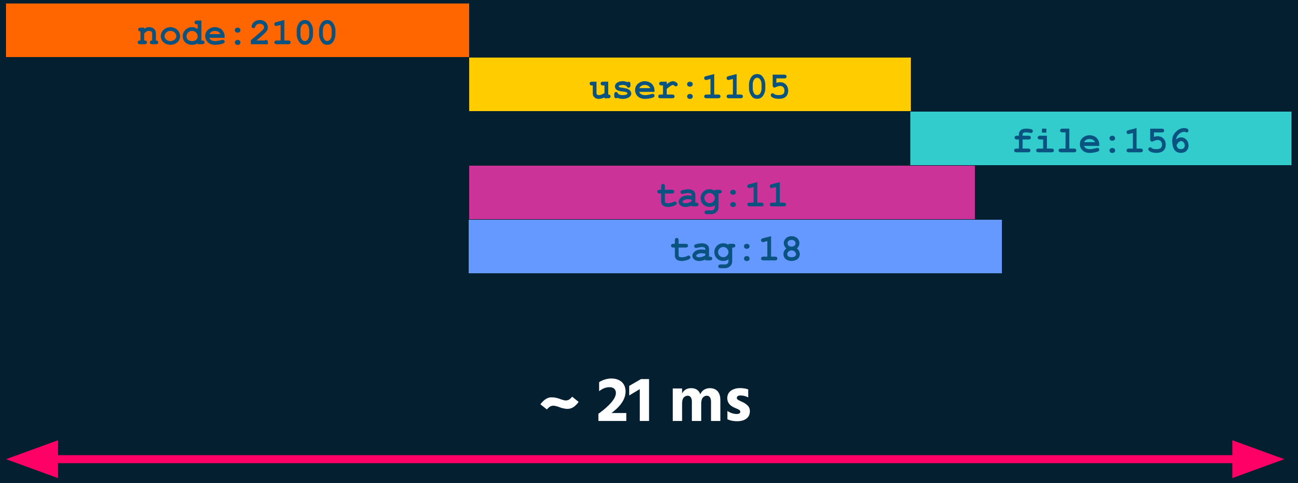
Benchmarking JSON API

- `ab -v4 -k -c8 -n10 -A u:p`
- `node:2100`
- `include`
  - Author
    - Author image
  - Tags (2 tags)

Benchmarking core HAL JSON

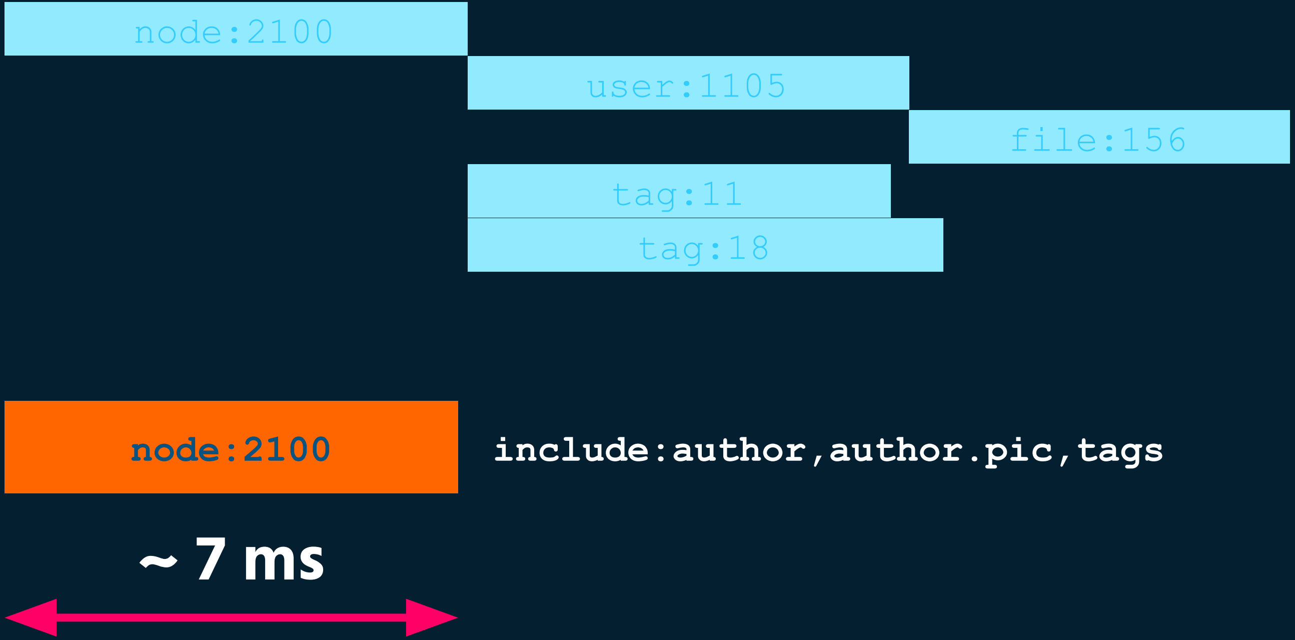
- `ab -v4 -k -c8 -n10 -A u:p`
- `node:2100`
  - `user:1105`
    - `file:156` (slow path)
  - ~~`tag:11`~~
  - ~~`tag:18`~~

Results (core): anonymous



Using Keep Alive

Results (jsonapi): anonymous

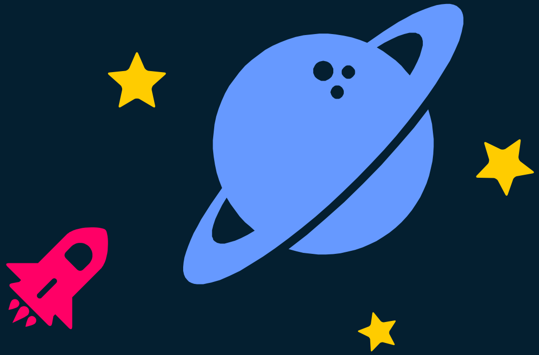


include:author,author.pic,tags

	Core (ms)	{json:api} (ms)
Anonymous	21	7
Auth	320	115
Uncached	392	182

https://gist.github.com/e0ipso/4b1b346b296fbf0c918450fef5b0b3d7

AVOID BOOTSTRAPS



And unnecessary HTTP round trips.

4.
DRUPAL MODULE

Our implementation of the standard.

drupal.org/project/jsonapi

That was expected, wasn’t it?

Drupal Integration

- Integrates with Authentication Providers
  - OAuth 2 Bearer Token (via simple_oauth)
- Full cacheability metadata support

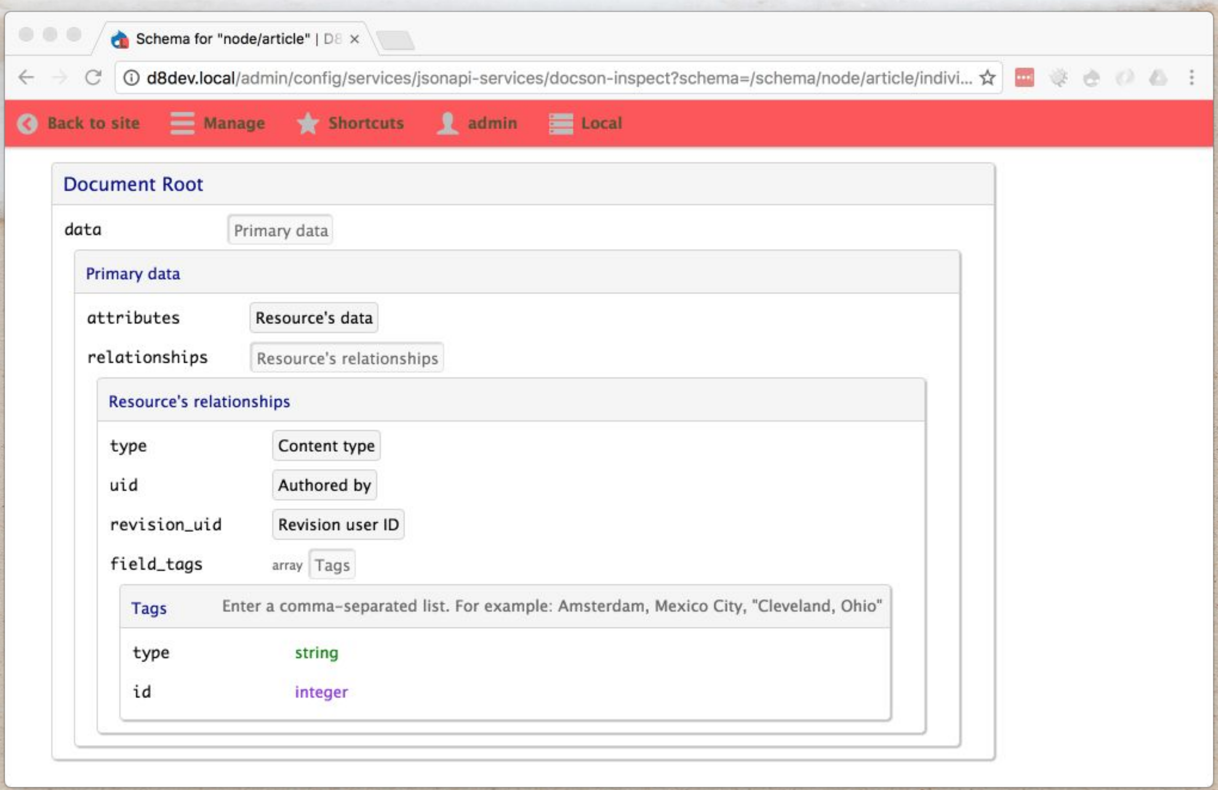
Oriented to entity bundles

- Each resource is a bundle
- `/api/node/page`
- Automatically enabled (can be disabled)
- You can do **any** entity query as filter
- Works with **config** entities!

Automatic schema generation

- Uses type data to generate the schema
- `/schema/node/page`
- Automatically enabled (can be disabled)

Schema usages?
GENERATE DOCS



Schema for "node/article" | D8 ×
d8dev.local/admin/config/services/jsonapi-services/docson-inspect?schema=/schema/node/article/indivi...
Back to site
Manage
Shortcuts
admin
Local

Document Root

data — Primary data

Primary data

attributes — Resource's data

relationships — Resource's relationships

Resource's relationships

type — Content type

uid — Authored by

revision_uid — Revision user ID

field_tags — array Tags

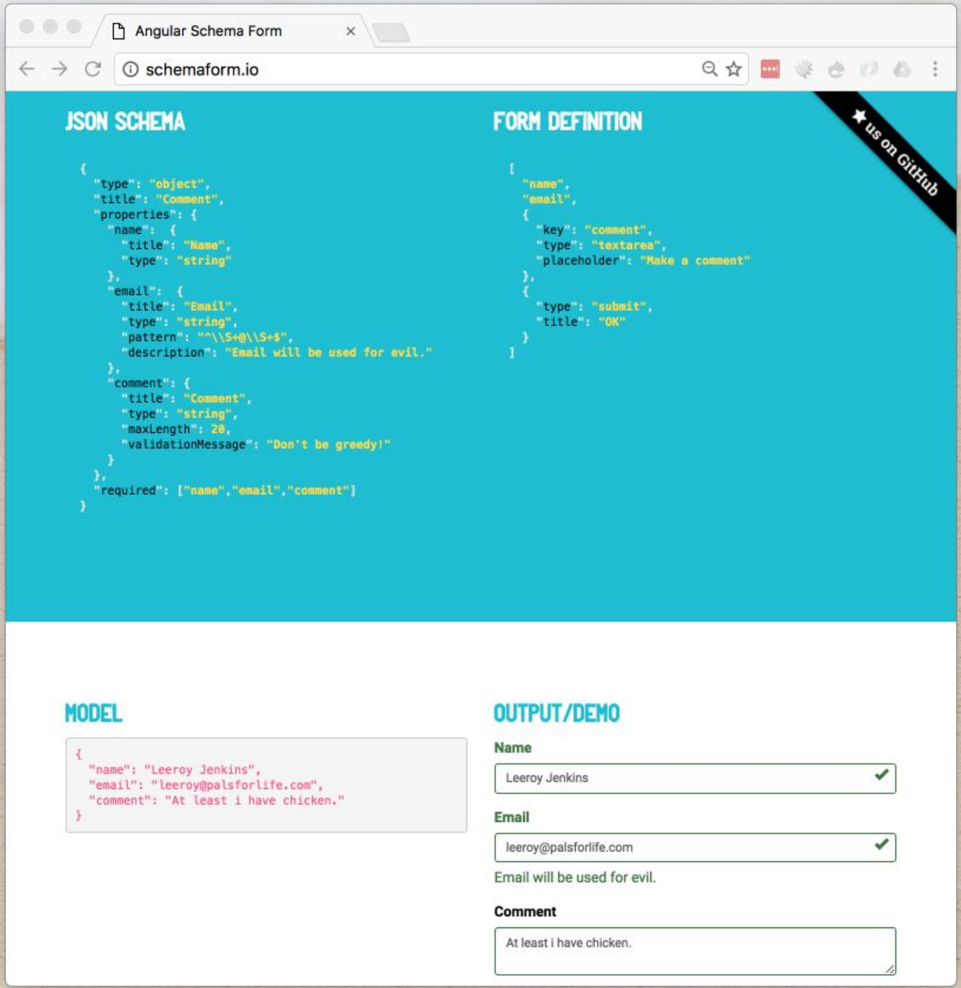
Tags — Enter a comma-separated list. For example: Amsterdam, Mexico City, "Cleveland, Ohio"

type — string

id — integer

Schema usages?

GENERATE FORMS



JSON SCHEMA

```
{
  "type": "object",
  "title": "Comment",
  "properties": {
    "name":  {
      "title": "Name",
      "type": "string"
    },
    "email":  {
      "title": "Email",
      "type": "string",
      "pattern": "^\\S+@\\S+$",
      "description": "Email will be used for evil."
    },
    "comment": {
      "title": "Comment",
      "type": "string",
      "maxLength": 20,
      "validationMessage": "Don't be greedy!"
    }
  },
  "required": ["name","email","comment"]
}
```

FORM DEFINITION

```
[
  "name",
  "email",
  {
    "key": "comment",
    "type": "textarea",
    "placeholder": "Make a comment"
  },
  {
    "type": "submit",
    "title": "OK"
  }
]
```

MODEL

```
{
  "name": "Leeroy Jenkins",
  "email": "leeroy@palsforlife.com",
  "comment": "At least i have chicken."
}
```

OUTPUT/DEMO

Name: Leeroy Jenkins ✓

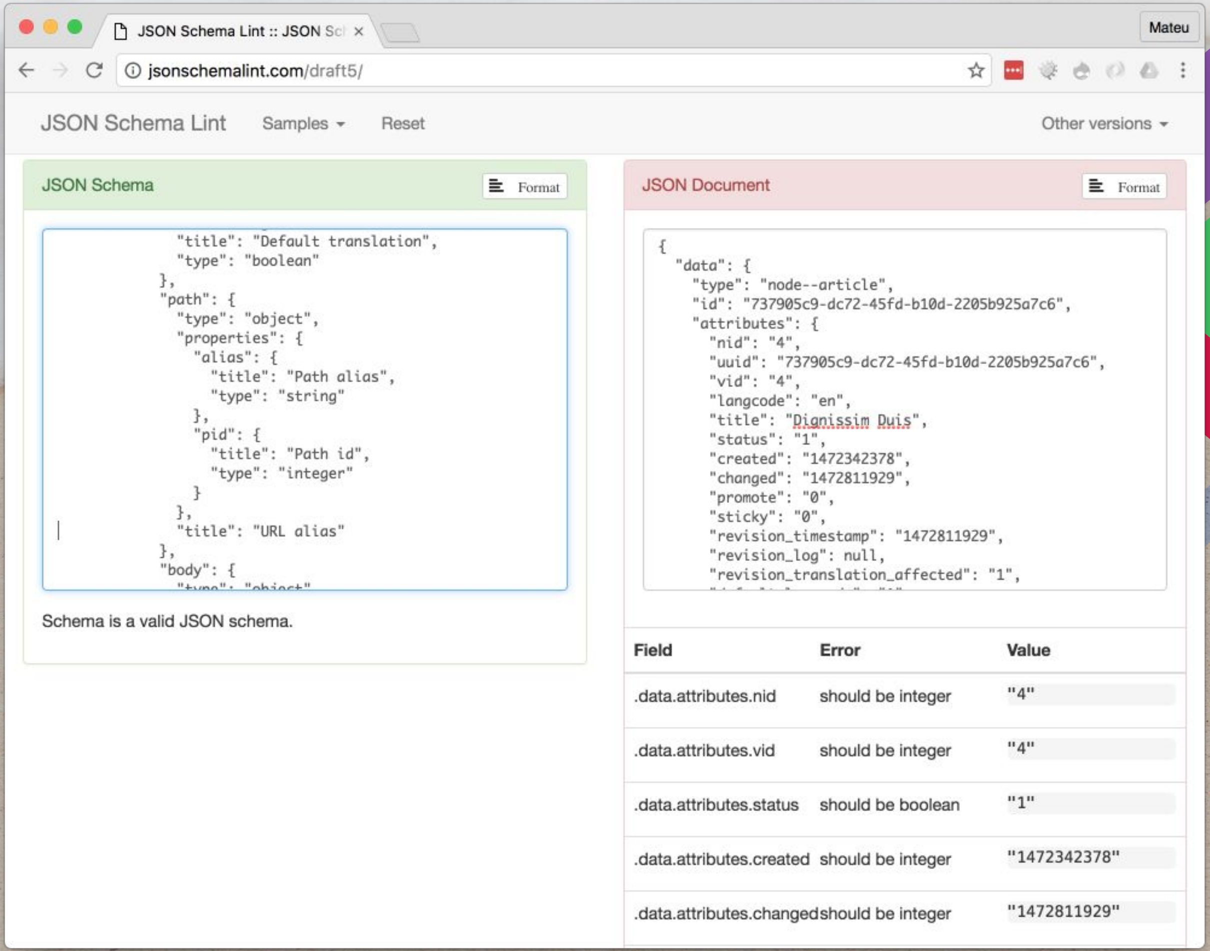
Email: leeroy@palsforlife.com ✓

Email will be used for evil.

Comment: At least i have chicken.

Schema usages?

VALIDATE DATA



```
                "title": "Default translation",
                "type": "boolean"
              },
              "path": {
                "type": "object",
                "properties": {
                  "alias": {
                    "title": "Path alias",
                    "type": "string"
                  },
                  "pid": {
                    "title": "Path id",
                    "type": "integer"
                  }
                },
                "title": "URL alias"
              },
              "body": {
```

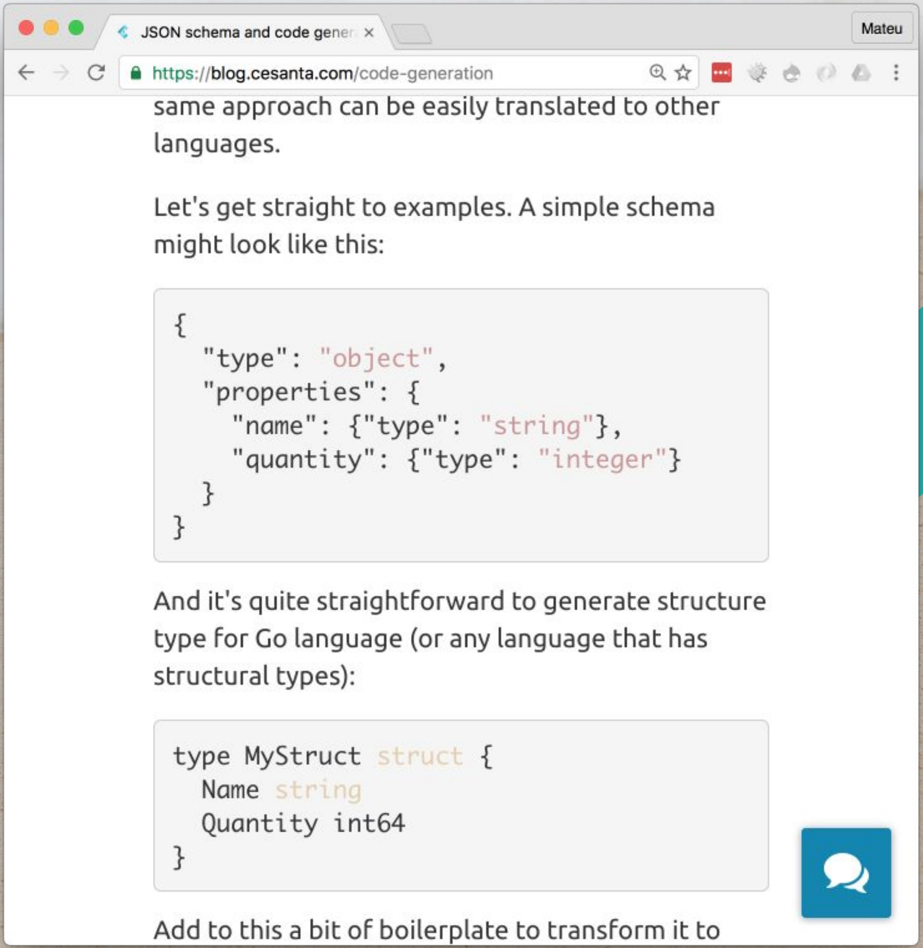
Schema is a valid JSON schema.

```
{
  "data": {
    "type": "node--article",
    "id": "737905c9-dc72-45fd-b10d-2205b925a7c6",
    "attributes": {
      "nid": "4",
      "uuid": "737905c9-dc72-45fd-b10d-2205b925a7c6",
      "vid": "4",
      "langcode": "en",
      "title": "Dignissim Duis",
      "status": "1",
      "created": "1472342378",
      "changed": "1472811929",
      "promote": "0",
      "sticky": "0",
      "revision_timestamp": "1472811929",
      "revision_log": null,
      "revision_translation_affected": "1",
```

Field	Error	Value
.data.attributes.nid	should be integer	"4"
.data.attributes.vid	should be integer	"4"
.data.attributes.status	should be boolean	"1"
.data.attributes.created	should be integer	"1472342378"
.data.attributes.changed	should be integer	"1472811929"

Schema usages?

GENERATE CODE


JSON schema and code gener
Mateu
https://blog.cesanta.com/code-generation

same approach can be easily translated to other languages.

Let's get straight to examples. A simple schema might look like this:

```
{
  "type": "object",
  "properties": {
    "name": {"type": "string"},
    "quantity": {"type": "integer"}
  }
}
```

And it's quite straightforward to generate structure type for Go language (or any language that has structural types):

```
type MyStruct struct {
  Name string
  Quantity int64
}
```

Add to this a bit of boilerplate to transform it to

Limitations

- Multilingual support is not great
- ~~File integration needs some work~~
- Revision support
- Extensible through code only
- Limited to the entity system

Open challenges

- Versioning content model in Drupal
- Responsive images and image styles
- Data pre-processing
- Multiple-operation requests
- Aggregated values

Do you want to help?

Join us for contribution sprints!

- **First Time Sprinter Workshop** - 9:00-12:00 - Room Wicklow 2A
- **Mentored Core Sprint** - 9:00-18:00 - Wicklow Hall 2B
- **General Sprints** - 9:00 - 18:00 - Wicklow Hall 2A

Credits

Special thanks to all the people who made and released these awesome resources for free:

- Presentation template by SlidesCarnival
- Photographs by Startupstockphotos
- Creative Commons images

What did you think?

Evaluate this session

events.drupal.org/dublin2016/schedule

https://events.drupal.org/node/add/session-evaluation?field_eval_session=13193