

BEYOND DOCUMENTATION

WITH

OPENAPI

GET /SPEAKERS/BOYAN

 Boyan Yordanov

 @specter_bg

 PHP Varna and VarnaJS organizer

 VarnaLab member

 Developer@ShtrakBG

***WHO LIKES WRITING
DOCUMENTATION?***

DOCUMENTATION THAT IS:

- "too long to read"
- "totally useless"
- "not updated"

JUST BORING

API SPECIFICATIONS

- machine readable
- easy to write
- more than documentation

*I still have to write it,
though.*

- me

I can do so many cool things with this definition.

- also me (a couple of months later)

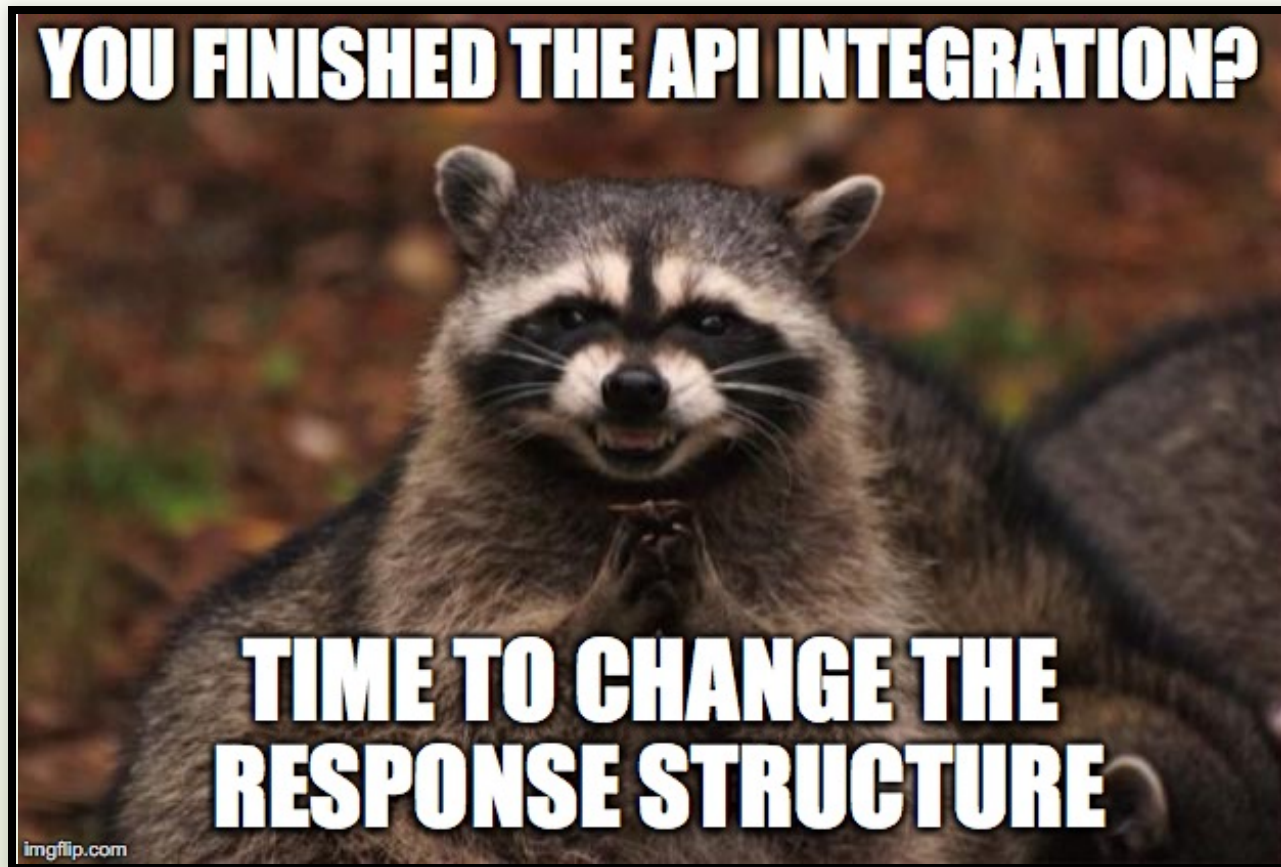
***WHY BOTHER WITH AN
API SPECIFICATION?***

HAVE YOU BEEN THIS PERSON?



<http://www.commitstrip.com>

HOW ABOUT THIS?



YOU FINISHED THE API INTEGRATION?

**TIME TO CHANGE THE
RESPONSE STRUCTURE**

***OUR PROCESS HAS
FAILED US***

1. LITTLE UPFRONT DESIGN WORK

2. UNANNOUNCED OR MISCOMMUNICATED CHANGES

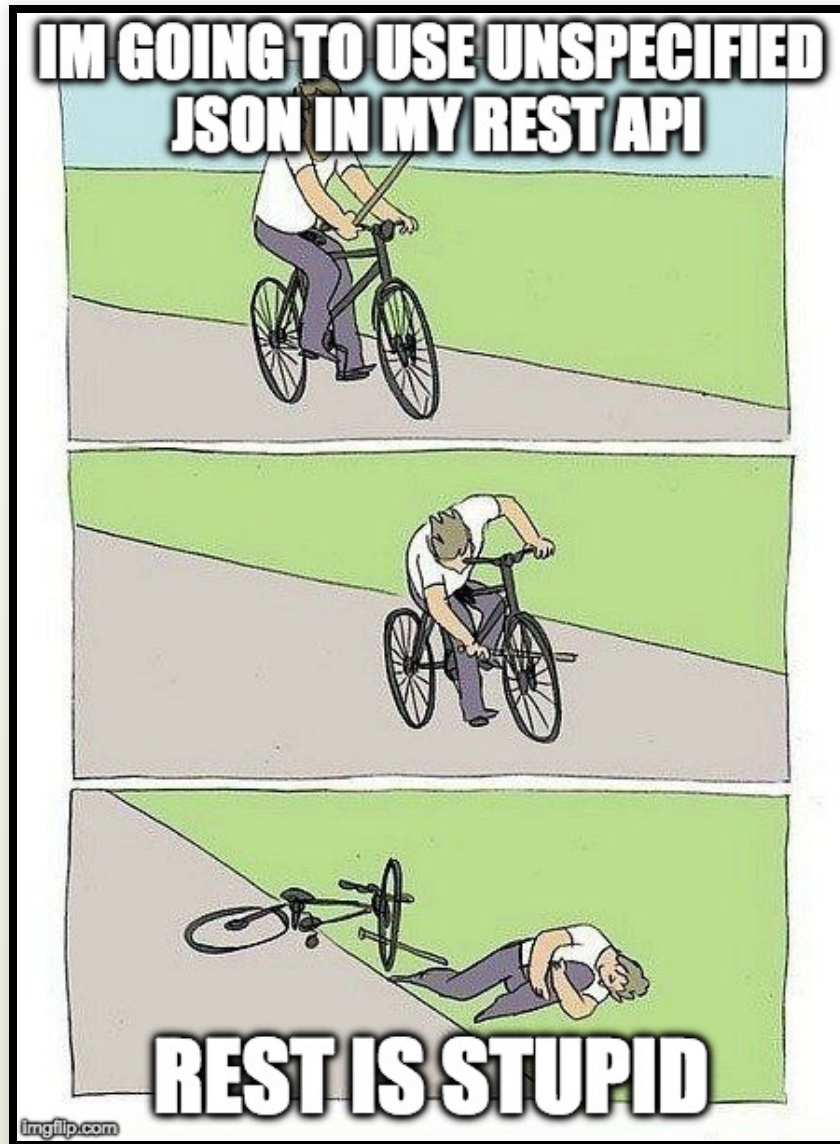
WASTED TIME

AND

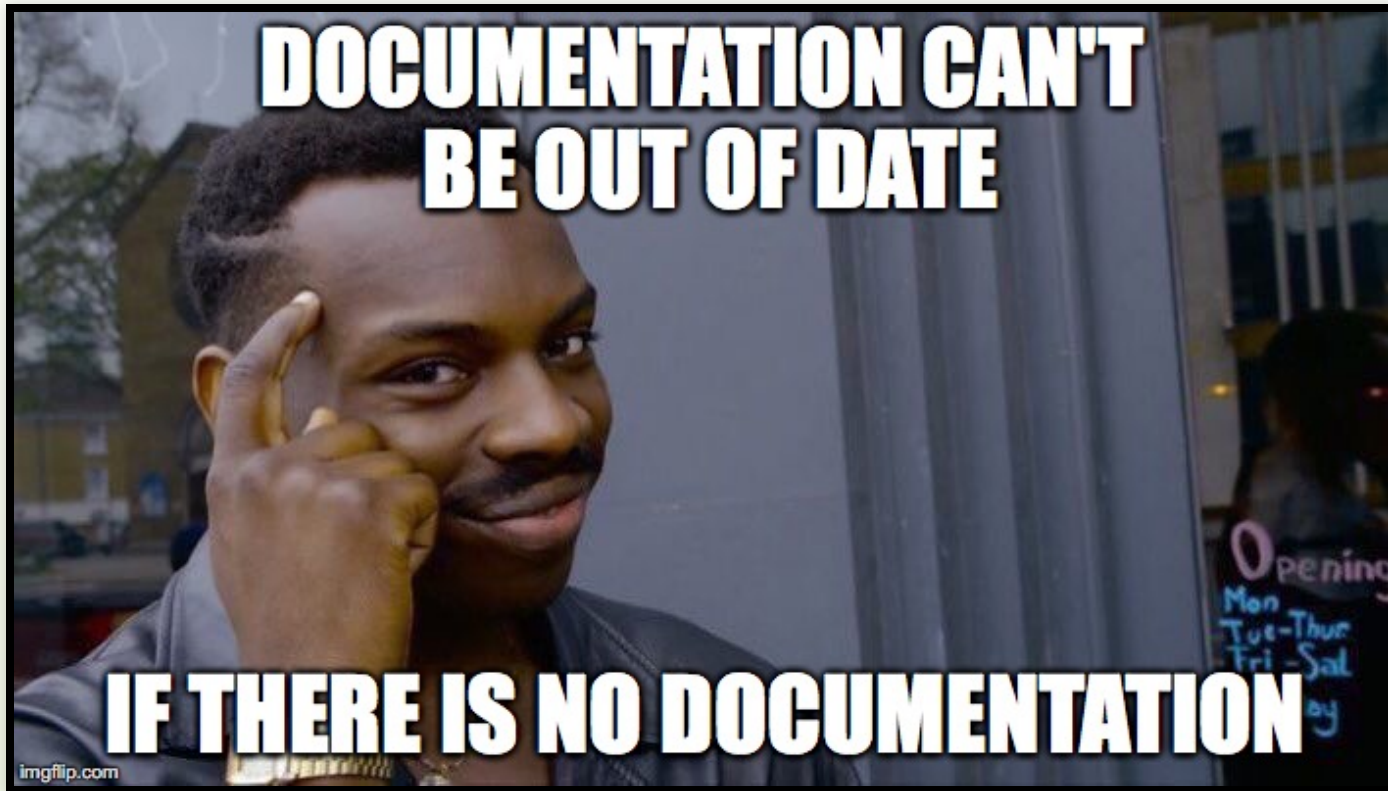
A LOT OF FRUSTRATION

COMMON PROBLEMS

- changes aren't properly tested
- clients constantly wait for the API
- documentation and code don't match
- SDKs have another opinion



<https://blog.apisyouwonthate.com/why-do-people-dislike-json-a7d67c8d38c1>



**DOCUMENTATION CAN'T
BE OUT OF DATE**

IF THERE IS NO DOCUMENTATION

imgflip.com

WHAT CAN WE DO?

SPECIFICATION FIRST

Nothing gets implemented until the
API is defined

SPECIFICATIONS AS CONTRACTS

verify that the API does what it says



GAME PLAN

1. produce the API specification
2. generate docs and mock API
3. refine the specification
4. use it in integration tests
5. work on the actual server code

POTENTIAL BENEFITS

1. THE SPEC IS THE
SINGLE SOURCE
OF TRUTH FOR
OUR DESIGN

2. ALWAYS UP TO

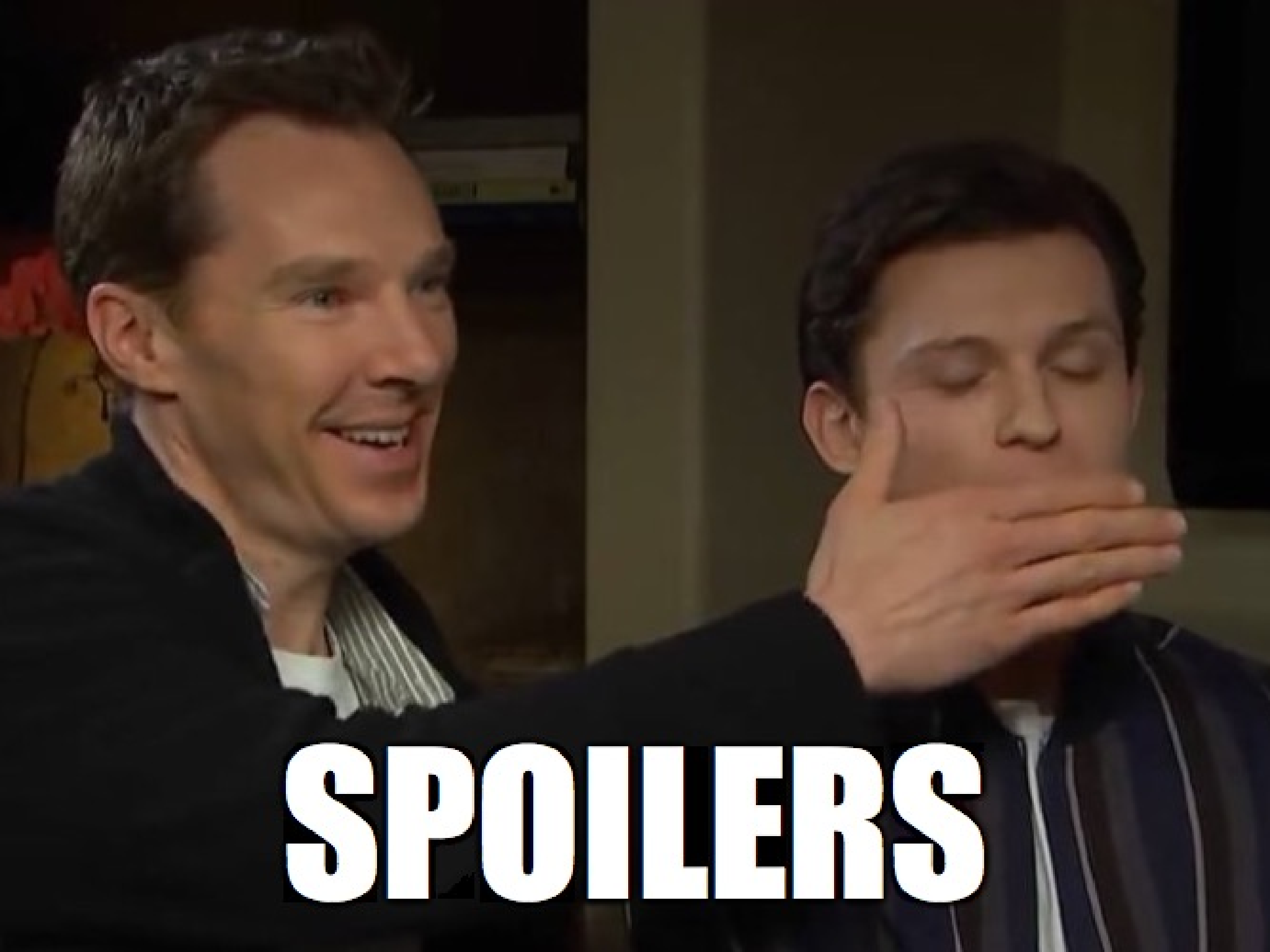
DATE

DOCUMENTATION

AND TESTS

3. WORK ON
BACKEND AND
CLIENTS
INDEPENDENTLY

API SPECIFICATION FORMATS



SPOILERS

API BLUEPRINT

- markdown based
- development is open on GitHub
- it's quite mature
- there's tooling for the most essential things
- mostly documentation oriented

RAML

RESTful API Modeling Language

- yaml based
- syntax is quite similar to the OpenAPI Specification
- covers the whole design/development process

JSON SCHEMA

not entirely fair to include it here

- json based
- focused only on the data model
- great for validation and tests
- includes syntax for describing hypermedia controls



OPENAPI

INITIATIVE

don't call it that
please

- formerly known as ~~Swagger~~
- yaml or json based
- very actively developed
- very active community

HOW TO WRITE AN OPENAPI SPECIFICATION

read the spec at swagger.io/specification

BASIC STRUCTURE

```
openapi: "3.0.0"  
info:  
  # ...  
servers:  
  # ...  
paths:  
  # ...  
tags:  
  # ...  
components:  
  # ...
```


***LET'S BUILD A SIMPLE OPENAPI
SPECIFICATION***

BASIC INFORMATION ABOUT THE API

```
info:
  description: >-
    *Imaginary* API for managing meetups.
    Imagined for this **PHPCE 2018** talk.
  version: '0.1'
  title: Meetups Are Awesome API
  contact:
    email: boyan.yordanov.dev@gmail.com
    name: Boyan Yordanov
    url: 'https://boyan.yordanov.com'
  license:
    # ...
```

HOW CAN USERS ACCESS THE API

```
servers:
```

- url: 'https://imaginary-meetups.com/api'
description: Imaginary API for managing Meetups
- url: 'https://staging.imaginary-meetups.com/api'
description: Staging server for the imaginary AP
- url: 'http://localhost:8080'
description: Development version of the API

it also supports templating

```
servers:  
  - url: https://{username}.api-server.com/api/  
    description: User specific URLs  
  variables:  
    username:  
      default: demo  
      description: assigned upon registration
```

WHAT CAN THE API DO

Just a list of paths and the possible requests and responses

```
paths:  
  /meetups  
  # ...  
  /meetups/{id}  
  # ...  
  /meetups/{id}/events  
  # ...
```

A path description

```
/meetups:  
  summary: meetups  
  description: Meetups resource  
  tags:  
    # ..  
  parameters:  
    # ..  
  get:  
    # ...  
  post:  
    # ..
```



LET'S GO DEEPER

DESCRIBING PARAMETERS

```
/meetups:  
# ...  
parameters:  
  - name: city  
    in: query / path / header  
    description: Filter meetups based on city  
    schema:  
      type: string
```

handling array parameters

```
/meetups:  
  # ...  
  parameters:  
    - name: technologies  
      in: query  
      description: Filter meetups based on city  
      schema:  
        type: array  
        items:  
          type: string  
      style: form  
      explode: false
```

Result

```
?technologies=php,apis,javascript
```

POSSIBLE RESPONSES

```
/meetups/{id}
get:
  responses:
    '200':
      content:
        application/json:
          schema:
            # ...
    '404':
      content:
        application/problem+json:
          # ...
```

SCHEMA OBJECTS

- "extended subset" of JSON Schema draft 5
- support references
- in future will support current drafts of JSON Schema and other formats

example schema

```
schema:  
  type: object  
  properties:  
    id:  
      type: string  
      format: uuid  
    name:  
      type: string  
    starting-date:  
      type: string  
      format: date
```

nullable fields

```
city:  
  description: either a string or null  
  type: string  
  nullable: true
```

other schema formats

still a proposal

```
schema:  
  x-oas-draft-alternativeSchema:  
    type: jsonSchema  
    location: ./real-jsonschema.json
```

<https://github.com/OAI/OpenAPI-Specification/issues/1532>

REFERENCE OBJECT

- object with *\$ref* property
- reference objects in the same document
- reference external documents
- replace inline definitions for most OpenAPI components

Replace almost anything in the spec

```
schema:  
  $ref: '#/components/schemas/Meetup'  
# ...  
responses:  
  '200':  
    $ref: 'MeetupsListResponse.yaml'
```

COMPONENTS

```
components:  
  schemas:  
    # ...  
  responses:  
    # ...  
  parameters:  
    # ...  
  requestBodies:  
    # ...  
  headers:  
    # ...  
  examples:  
    #
```

***WE HAVE AN
OPENAPI SPECIFICATION
WHAT DO WE DO WITH IT?***

EASY PICKINGS

HTML DOCUMENTATION

Swagger UI - least favorite

The screenshot displays the Swaggerhub interface for an API titled "Meetups Are Awesome API". The interface includes a top navigation bar with the Swaggerhub logo, user information, and various utility buttons. The main content area shows the API's description, version (0.1 OAS3), and a list of endpoints. The endpoints are categorized into "meetups" and "meetup".

Meetups Are Awesome API
0.1 OAS3

Imaginary API Service or managing meetups and events.
Imagined for this **Bulgaria Web Summit 2018** talk.
[Boyan Yordanov - Website](#)
[Send email to Boyan Yordanov](#)
MIT

OAS 3.0 Tools are in Beta, some features may be missing.

<https://imaginary-meetups.com/api>

bws18 Prepared for Bulgaria Web Summit 2018

openapi OpenAPI specification demo

meetups

- GET** /meetups Find Meetups
- POST** /meetups Start new meetup
- GET** /reports/{id} Generate Report

meetup

- GET** /meetups/{id} Retrieve Meetup

reports

Widdershins + Slate / Shins.js

SHINS
REPLACE THIS WITH YOUR LOGO
IN SOURCE/IMAGES/LOGO.PNG

Q Search

Meetups Are Awesome API v...
Imaginary API Service or mana...

meetups
meetup
reports
Schemas

Meetups Are Awesome API v0.1

Imaginary API Service or managing meetups and events.

Imagined for this **Bulgaria Web Summit 2018** talk.

Base URLs:

- <https://imaginary-meetups.com/api>
- <https://staging.imaginary-meetups.com/api>
- <http://localhost:8080>

Email: [Boyan Yordanov](#) Web: [Boyan Yordanov](#) License: [MIT](#)

meetups

findMeetups

GET /meetups

Find Meetups

Find and search for meetups

Parameters

Parameter	In	Type	Required	Description
city	query	string	false	Filter meetups based on city
technology	query	string	false	Filter meetups by technology

Enumerated Values

Parameter	Value
technology	apis
technology	php

Shell HTTP JavaScript Node.JS Ruby Python Java

Scroll down for code samples, example requests and responses. Select a language for code samples from the tabs above or the mobile navigation menu.

Code samples

```
var headers = {
  'Accept': 'application/json'
};

$.ajax({
  url: 'https://imaginary-meetups.com/api/meetups',
  method: 'get',

  headers: headers,
  success: function(data) {
    console.log(JSON.stringify(data));
  }
})
```

REDOC

Search...

IMAGINARY API SERVICE OR MANAGING MEETUPS AND EVENTS.

BWS18

OPENAPI

MEETUPS

Find Meetups

Start new meetup

Generate Report

MEETUP

REPORTS

Find Meetups

Find and search for meetups

QUERY PARAMETERS

city	string	Filter meetups based on city
technology	string	Enum: "apis" "php" "javascript" "ruby" "java" Filter meetups by technology

Responses

200 Retrieves all meetups in the system

RESPONSE SCHEMA: application/json

meta	object (PaginationControls)
data	Array of object

```
Array [
  {
    id: string <uuid>
    name: string
    city: string
    starting-date: string <date>
      The date the meetup was founded
  }
]
```

Response samples

200

application/json

```
{
  "meta": {
    "total": 3,
    "current-page": 1,
    "total-pages": 1
  },
  "data": [
    {
      "name": "PHP Varna",
      "city": "Varna",
      "starting-date": "13-11-2016"
    },
    {
      "name": "WordPress Varna",
      "city": "Varna"
    },
    {
      "name": ".NET Varna",
      "city": "Varna",
      "starting-date": "05-02-2018"
    }
  ]
}
```


LINTER - SPECCY

<https://github.com/wework/specy>

- lints your specification
- supports rulesets
- docs preview with ReDoc
- resolve spec in one file
- supports external references written in json-schema

EDITORS

everything supporting yaml

VSCode with openapi-lint extension

- validates and lints
- converts Swagger/OpenAPI 2.0 to OpenAPI 3.0
- intellisense for both formats

Swagger Editor

The image shows the Swagger Editor interface. On the left, a code editor displays a YAML OpenAPI definition for an API named "Meetups Are Awesome API". The definition includes server URLs, contact information, license (MIT), and several endpoints under the "/meetups" path, such as "findMeetups" and "startNewMeetup".

```
1 openapi: 3.0.0
2 servers:
3   - url: 'https://imaginary-meetups.com/api'
4     description: Imaginary API service for managing Meetup's
5   - url: 'https://staging.imaginary-meetups.com/api'
6     description: Staging server for the imaginary API service
7   - url: 'http://localhost:8080'
8     description: Development version of the API
9 info:
10  description: >-
11    ## Imaginary API Service or managing meetups and events.
12
13    Imagined for this **Bulgaria Web Summit 2018** talk.
14  version: '0.1'
15  title: Meetups Are Awesome API
16  contact:
17    email: boyan.yordanov.dev@gmail.com
18    name: Boyan Yordanov
19    url: 'https://boyan.yordanov.com'
20  license:
21    name: MIT
22    url: 'https://opensource.org/licenses/MIT'
23  tags:
24    - name: bws18
25      description: Prepared for Bulgaria Web Summit 2018
26    - name: openapi
27      description: OpenAPI specification demo
28  paths:
29    /meetups:
30      summary: meetups
31      description: Meetups resource
32      get:
33        operationId: findMeetups
34        summary: Find Meetups
35        description: Find and search for meetups
36        tags:
37          - meetups
38        parameters:
39          - name: city
40            in: query
41            description: Filter meetups based on city
42            schema:
43              type: string
44          - name: technology
45            in: query
46            description: Filter meetups by technology
47            schema:
```

On the right, the rendered API page displays the title "Meetups Are Awesome API" and version "0.1 OAS3". It provides a description: "Imaginary API Service or managing meetups and events. Imagined for this Bulgaria Web Summit 2018 talk." and lists contact information for Boyan Yordanov. A dropdown menu shows the selected API URL: "https://imaginary-meetups.com/api". Below this, a list of endpoints is shown with their methods and descriptions:

- bws18** Prepared for Bulgaria Web Summit 2018
- openapi** OpenAPI specification demo
- meetups**
 - GET** /meetups Find Meetups
 - POST** /meetups Start new meetup
 - GET** /reports/{id} Generate Report
- meetup**
 - GET** /meetups/{id} Retrieve Meetup
- reports**

Apicurio

The screenshot displays the Apicurio Studio interface for editing an API. The breadcrumb trail is Dashboard > APIs > Meetups Are Awesome API > Editor. The user is logged in as Boyan Yordanov. The main view shows the API endpoint `/meetups` with a `GET` method. The interface is divided into several sections:

- Paths:** A list of paths including `/meetups` (highlighted with `GET` and `POST` buttons), `/meetups/{id}`, and `/reports/{id}`.
- Definitions:** A list of definitions including `Meetup`, `NewMeetup`, `PaginatedMeetups`, and `PaginationControls`.
- INFO:** The title is "Find Meetups" with a description "Find and search for meetups".
- SERVERS:** A section for defining servers.
- REQUEST BODY:** A section for defining the request body.
- QUERY PARAMETERS:** A list of query parameters:
 - `city`: Filter meetups based on city, type `string`.
 - `technology`: Filter meetups by technology, type `string`.
- RESPONSES:** A list of responses:
 - 200 OK:** Retrieves all meetups in the system. The response body is `application/json` with a type of `PaginatedMeetups`. It includes a table of examples:

Name	Summary	Description
varna-meetups	No Summary	No Description

Buttons for "Add an example" and "Done" are visible at the bottom of the response editor.

OpenAPI GUI

The screenshot displays the OpenAPI GUI v3 interface. At the top, the title "OpenAPI-GUI v3" is on the left, and navigation links "Top", "Settings", "Save", "Undo", "Follow", and "GitHub" are on the right. Below this is a main navigation bar with "About", "Upload", "Header", "Servers", "Security", "Tags", "Main" (active), "Schemas", "Wizards", "Export JSON", and "Export YAML".

The left sidebar shows a tree view of API endpoints: `/meetups` (with sub-items `get` and `post`), `/meetups/{id}` (with sub-item `get`), and `/reports/{id}` (with sub-item `get`).

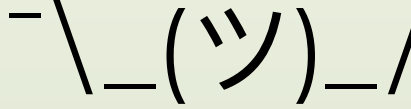
The main content area is divided into two sections. The top section shows the `/meetups` endpoint with two operations: `GET findMeetups` and `POST startMeetup`. The bottom section shows the `/meetups/{id}` endpoint with the `GET findMeetup` operation selected. This operation view includes tabs for "Main", "Description", "Docs", "Parameters", "Request Body", "Responses" (active), "Security", "Links", and "Callbacks".

Under the "Responses" tab, two response entries are visible:

- 200** Status Code: The meetup was found and successfully returned. Below this, the **Media-Type** is set to `application/json`.
- 404** Status Code: Required.

SENYA EDITOR

<https://senya.io>

- PhpStorm (JetBrains) plugin
- free for the time being
- smart completion
- live linting
- handles \$refs well
- preview in Swagger UI 

CODE GENERATION
OPENAPI GENERATOR

- community supported fork of **swagger-codegen**
- supports OpenAPI 3.0
- Updated templates for different languages and frameworks

JANE-PHP/JANE-PHP

<https://github.com/janephp/janephp>

- generates PSR-7 compatible SDKs
- can use different clients based on HTTPPlug
- supports json-schema and OpenAPI
- OpenAPI v3 support was added recently

MOCK SERVERS

- APISprout
- SwaggerHub
- Stoplight.io

POSTMAN COLLECTIONS

Apimatic.io

- upload/download or use API
- transform to and from OpenAPI
- lots of formats including Postman 2.0 Collections

OPENAPI / JSON SCHEMA CONVERSION

<https://github.com/mikunn/openapi2schema>

<https://github.com/wework/json-schema-to-openapi-schema>

WHY CONVERT?

BEST OF BOTH WORLDS

(sort of)

EXAMPLE TEST

<https://github.com/swaggest/php-json-schema>

```
/** @test */  
public function it_tests_with_swaggest_json_schema()  
{  
    $schema = $this->loadSchema('events');  
    $response = $this->get('/events');  
    $this->assertValidResponse($schema, $response);  
}
```

EXAMPLE ASSERTION

<https://github.com/swaggest/php-json-schema>

```
protected function assertValidResponse (
    $schema, $response
) {
    $validator = Schema::import($schema);
    try {
        $data = json_decode($response->getContent());
        $validator->in($data);
        $this->assertTrue(true, 'Something went wrong.')
    } catch (InvalidValue $e) {
        $this->fail($e->getMessage());
    }
}
```


WORKING WITH THE SPECIFICATION

<https://github.com/apiioo/psx-api>

- Parse OpenAPI specification
- Generate from code
- Nice PHP API to work with the spec
- Supports other specifications as well

PROOF OF CONCEPT TIME

<https://github.com/boyanyordanov/php-openapi-testing>

```
class ApiTest extends TestCase
{
    use OpenAPITestTools\OpenAPIAssertions;

    /** @dataProvider getApiTestCases */
    public function test_it_runs_openapi_based_tests (
        $path,
        $resource
    ) {
        $this->assertValidContract($path, $resource);
    }
}
```

```
//...

public function getApiTestCases(): array
{
    $provider = new OpenAPITestTools\SpecDataProvide
        __DIR__ . '/../../openapi.yaml'
    );
    return $provider->getTestCases();
}
}
```

MORE TOOLS

OPENAPI.TOOLS

thanks to Phil Sturgeon and Matthew
Trask

THANK YOU!

QUESTIONS?

<https://joind.in/talk/17ec9>



RESOURCES

- **Specification:** <https://swagger.io/specification>
- <https://github.com/OAI/OpenAPI-Specification> - issues / PRs
- <https://apisyouwonthate.com> - books/blog/slack
- <https://philsturgeon.uk>
- <https://apihandyman.io>
- <https://apievangelist.com>
- <http://json-schema.org>
- <http://www.commitstrip.com/en/2018/02/26/its-good-to-have-experience/>