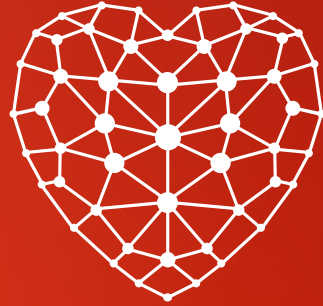




Design Driven APIs with Node.js and Swagger

Jeremy Whitlock (@whitlockjc)
Software Engineer



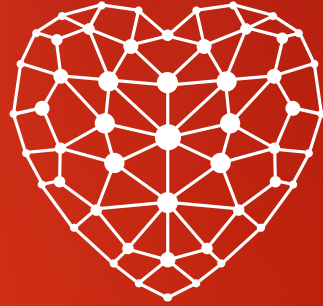


What is an API?

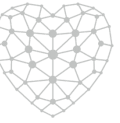


laugh
just kidding





API Documentation

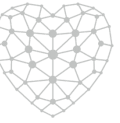


API Documentation is “Critical”



"While APIs are meant for computers, applications, and other systems, integrations all begin with helping other humans understand the what, and how of what an API does, making API docs a critical point in the API journey."





READ THE SOURCE LUKE



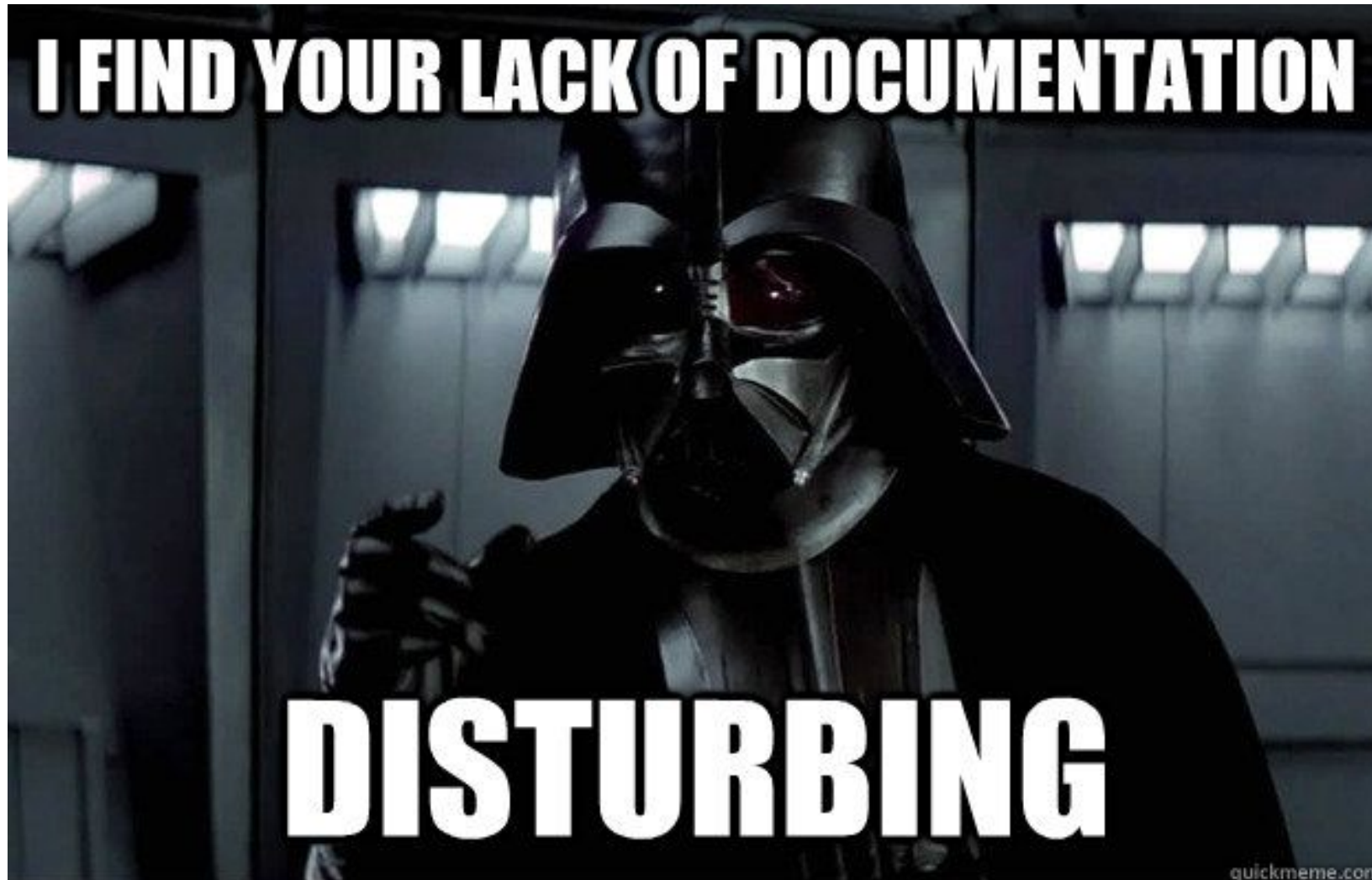
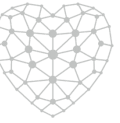
If It Isn't Documented, It Doesn't Exist



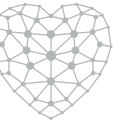
“No matter how wonderful your library is and how intelligent its design, if you're the only one who understands it, it doesn't do any good. Documentation means not just autogenerated API references, but also annotated examples and in-depth tutorials. You need all three to make sure your library can be easily adopted.”

Nicholas Zakas

<http://blog.codinghorror.com/if-it-isnt-documented-it-doesnt-exist/>



Good API Documentation



URL

TOKEN

API REFERENCE

pet

- Add a new pet to the store
- Update an existing pet
- Finds Pets by status**
- Finds Pets by tags
- Deletes a pet
- Find pet by ID
- Updates a pet in the store with form data
- uploads an image

store

user

Swagger Petstore

This is a sample server Petstore server.
You can find out more about Swagger at <http://swagger.io> or on irc.freenode.net,

Finds Pets by status

GET /pet/findByStatus

Multiple status values can be provided with comma seperated strings

Parameters

status

Status values that need to be considered for filter

Test this endpoint

TRY **OAUTH**

Response Type

Response Messages

400 Invalid status value

RESPONSE SAMPLE

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

RESPONSE SCHEMA

{ } Pet

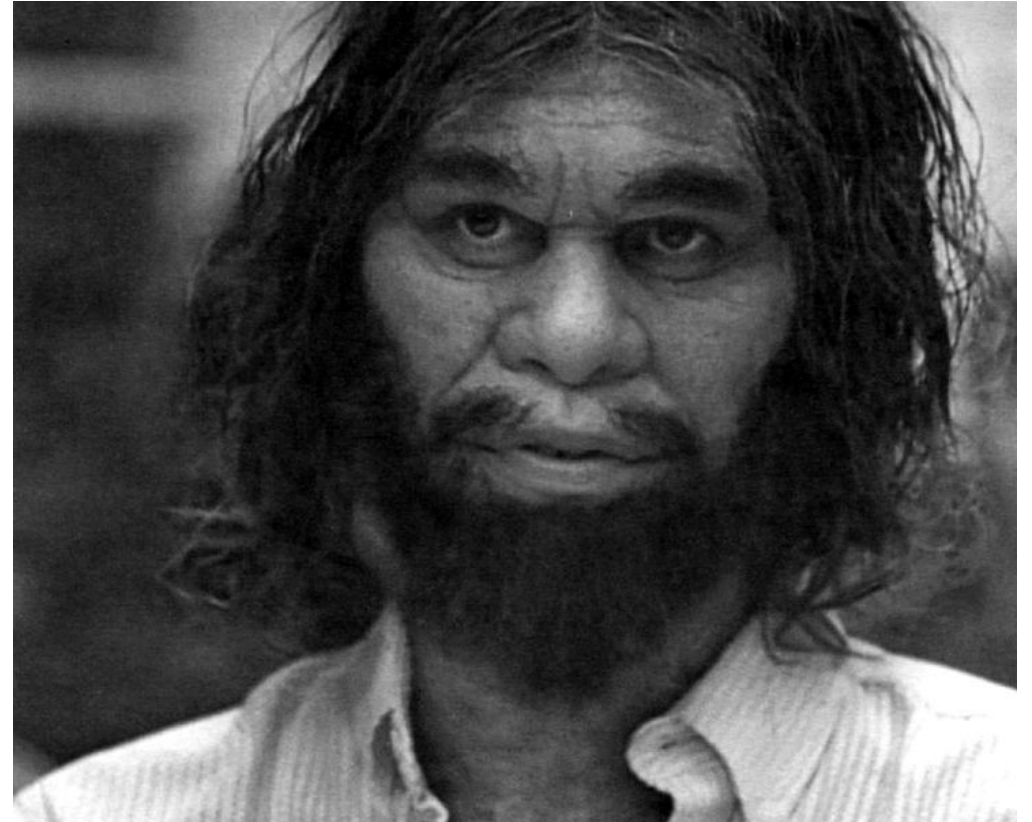
id
 (optional)

category
 (optional)

name

photoUrls

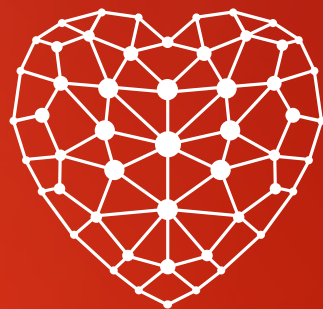
Good API Documentation



API Documentation should be easy to understand.



WHY ALL THE API DOCUMENTATION TALK?



Swagger

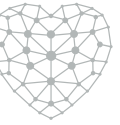
Sample Swagger Document



- Written in JSON or YAML
- Clearly describe:
 - Available APIs
 - API input contract
 - API output contract
 - Required authn/authz
 - Success/Error responses
 - ...
- Extensible

```
11  paths:
12    /weather:
13      get:
14        x-swagger-router-controller: "Weather"
15        operationId: "getWeather"
16        tags:
17          - "/weather"
18        description: "Returns the current weather for the requested location using the requested unit."
19        parameters:
20          - name: "location"
21            in: "query"
22            description: "The MSN Weather location search string."
23            required: true
24            type: "string"
25          - name: "unit"
26            in: "query"
27            description: "The unit, either 'C' or 'F'."
28            required: true
29            type: "string"
30            enum:
31              - "C"
32              - "F"
33            default: "F"
34        responses:
35          200:
36            description: "Successful request."
37            schema:
38              $ref: "#/definitions/Weather"
39          default:
40            description: "Invalid request."
41            schema:
42              $ref: "#/definitions/Error"
```

Swagger Emerging as an Industry Standard



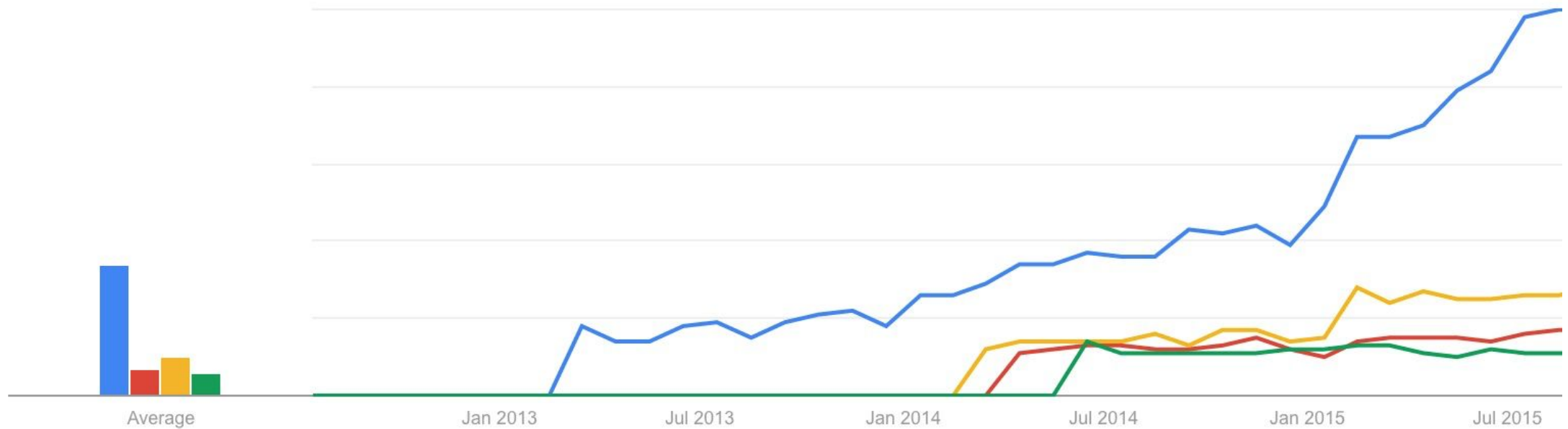
swagger api

raml api

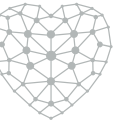
api blueprint

wadl api

Interest over time



There is More to Swagger Than This!

Explore

Swagger Petstore

This is a sample server Petstore server. You can find out more about Swagger at <http://swagger.io> or on irc.freenode.net, #swagger. For this sample, you can use the api key "special-key" to test the authorization filters

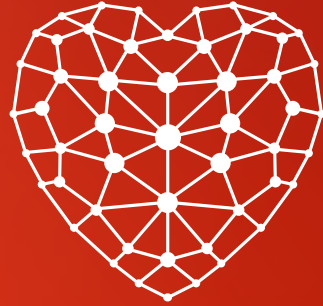
More documentations
Find out more about Swagger

<http://swagger.io>
[Contact the developer](#)
[Apache 2.0](#)

pet : Everything about your Pets

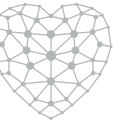
Show/Hide | List Operations | Expand Operations

POST	/pet	Add a new pet to the store
PUT	/pet	Update an existing pet
GET	/pet/findByStatus	Finds Pets by status
GET	/pet/findByTags	Finds Pets by tags
DELETE	/pet/{petId}	Deletes a pet
GET	/pet/{petId}	Find pet by ID
POST	/pet/{petId}	Updates a pet in the store with form data



How Do I Use Swagger?

Code Driven Swagger

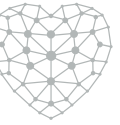


Swagger

Duplication

```
18 exports.getWeather = {  
19   spec: {  
20     "description" : "Operations about pets",  
21     "path" : "/weather",  
22     "notes" : "Returns the current weather for the requested location using the requested unit.",  
23     "summary" : "",  
24     "method": "GET",  
25     "params" : [  
26       params.query("location", "The MSN Weather location search string.", "", true, false, ""),  
27       params.query("unit", "The unit, either 'C' or 'F'.", "", true, false, "")  
28     ],  
29     "type" : "Weather",  
30     "responseMessages" : [errors.invalid('id'), errors.notFound('Weather')],  
31     "nickname" : "getWeather"  
32   },  
33   action: function (req, res) {  
34     var location = req.params.location;  
35     var unit = req.params.unit;  
36  
37     if (typeof location === 'undefined') {  
38       return next('\location is a required argument');  
39     } else if (typeof unit !== 'undefined' && validUnits.indexOf(unit) === -1) {  
40       return next('\ + unit + \' is not a valid unit');  
41     }  
42  
43     if (typeof unit === 'undefined') {  
44       unit = 'F';  
45     }  
46  
47     // Code necessary to consume the Weather API and respond  
48     weather.find({  
49       search: location,  
50       degreeType: unit  
51     }, function(err, result) {  
52       if (err) {  
53         console.log(err.stack);  
54         return next(err.message);  
55       }  
56  
57       res.setHeader('Content-Type', 'application/json');  
58       res.end(JSON.stringify(result[0] || {}, null, 2));  
59     });  
60   }  
61 };
```

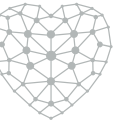
Code Driven Swagger



Swagger

```
3 import com.wordnik.swagger.annotations.*;
4 import io.swagger.model.Weather;
5
6 import javax.ws.rs.GET;
7 import javax.ws.rs.Path;
8 import javax.ws.rs.QueryParam;
9 import javax.ws.rs.core.Response;
10
11 @Path("/weather")
12 @Api(value = "/weather", description = "the weather API")
13 public class WeatherApi {
14
15     @GET
16     @ApiOperation(value = "",
17         notes = "Returns the current weather for the requested location using the requested unit.",
18         response = Weather.class)
19     @ApiResponses(value = {
20         @ApiResponse(code = 200,
21             message = "Successful request."),
22         @ApiResponse(code = 0,
23             message = "Invalid request.")
24     })
25     public Response getWeather(@QueryParam("location")
26         @ApiParam(value = "The MSN Weather location search string.", required=true)
27         String location,
28         @QueryParam("unit")
29         @ApiParam(value = "The unit, either 'C' or 'F'.", required=true)
30         String unit)
31     throws NotFoundException {
32         return Response.ok().entity(new ApiResponseMessage(ApiResponseMessage.OK, "magic!")).build();
33     }
34
35 }
36
```

Swagger Driven Code



```
5 module.exports.getWeather = function getWeather (req, res, next) {  
6   // Code necessary to consume the Weather API and respond  
7   weather.find({  
8     search: req.swagger.params.location.value,  
9     degreeType: req.swagger.params.unit.value  
10  }, function(err, result) {  
11    if (err) {  
12      console.log(err.stack);  
13      return next(err.message);  
14    }  
15  }  
16  res.setHeader('Content-Type', 'application/json');  
17  res.end(JSON.stringify(result[0] || {}, null, 2));  
18  });  
19  };
```


Before (43 lines)

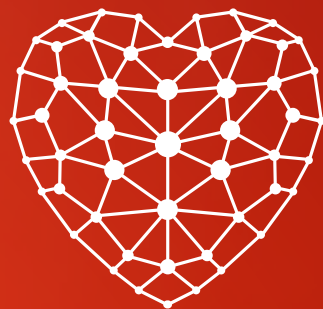
```
18 exports.getWeather = {
19   'spec': {
20     "description": "Operations about pets",
21     "path": "/weather",
22     "notes": "Returns the current weather for the requested location using the requested unit.",
23     "summary": "",
24     "method": "GET",
25     "params": [
26       {
27         name: "location",
28         description: "The MSN Weather location search string.",
29         type: "string",
30         required: true,
31         format: "url",
32       },
33       {
34         name: "unit",
35         description: "The unit, either 'C' or 'F'.",
36         type: "string",
37         required: true,
38         format: "url",
39       },
40     ],
41     "type": "Weather",
42     "responseMessages": [errors.invalid('id'), errors.notFound('Weather')],
43     "nickname": "getWeather"
44   },
45   'action': function (req, res) {
46     var location = req.params.location;
47     var unit = req.params.unit;
48
49     if (typeof location === 'undefined') {
50       return next('\location is a required argument\');
51     } else if (typeof unit !== 'undefined' && validUnits.indexOf(unit) === -1) {
52       return next('\unit is not a valid unit\');
53     }
54
55     if (typeof unit === 'undefined') {
56       unit = 'F';
57     }
58
59     // Code necessary to consume the Weather API and respond
60     weather.find({
61       search: location,
62       degreeType: unit
63     }, function(err, result) {
64       if (err) {
65         console.log(err.stack);
66         return next(err.message);
67       }
68
69       res.setHeader('Content-Type', 'application/json');
70       res.end(JSON.stringify(result[0] || {}, null, 2));
71     });
72   }
73 }
```

After (14 lines)

```
5 module.exports.getWeather = function getWeather (req, res, next) {
6   // Code necessary to consume the Weather API and respond
7   weather.find({
8     search: req.swagger.params.location.value,
9     degreeType: req.swagger.params.unit.value
10  }, function(err, result) {
11    if (err) {
12      console.log(err.stack);
13      return next(err.message);
14    }
15
16    res.setHeader('Content-Type', 'application/json');
17    res.end(JSON.stringify(result[0] || {}, null, 2));
18  });
19 }
```

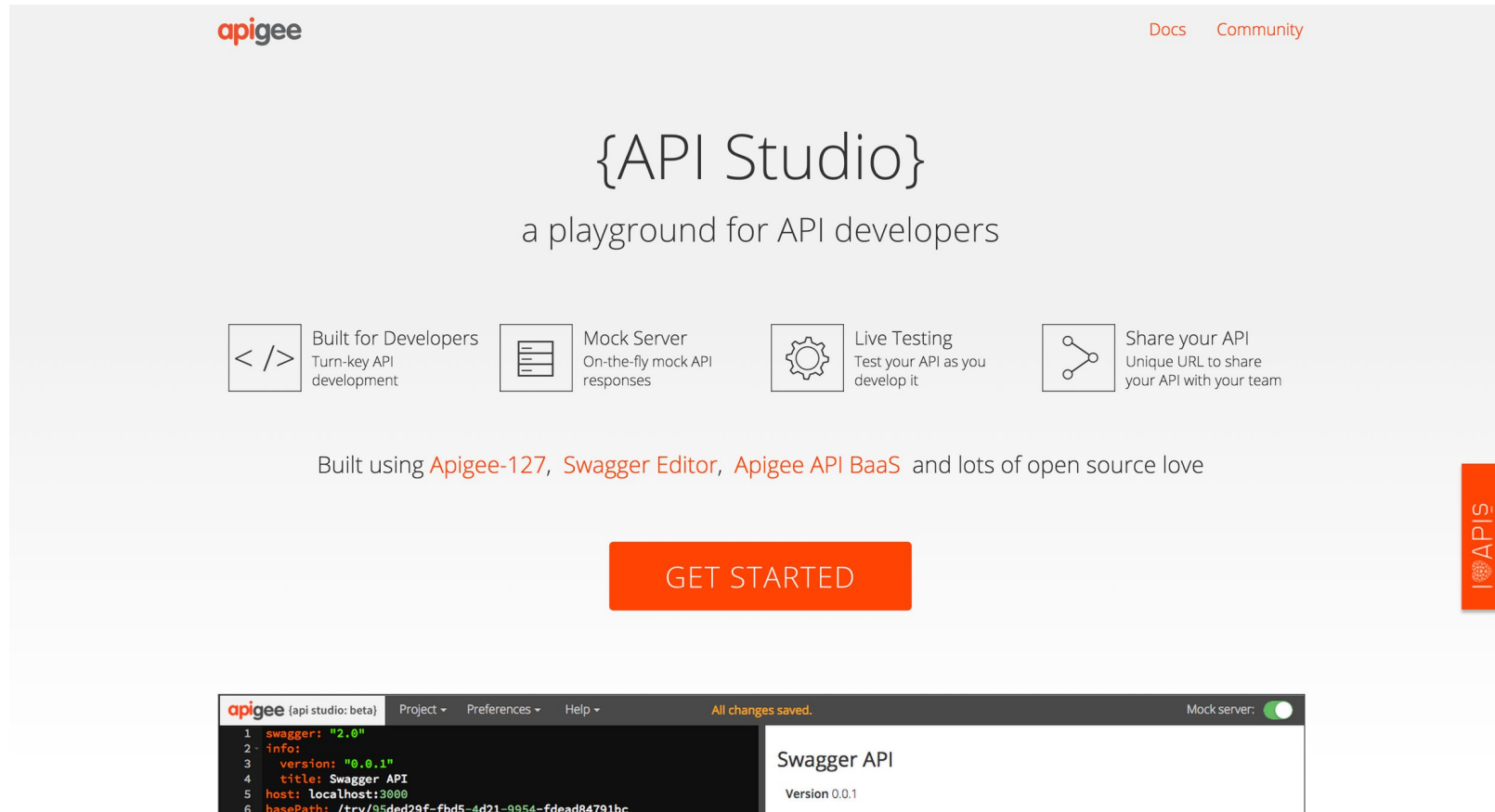


WHICH APPROACH IS BETTER?



Swagger Driven APIs

API Design



<http://apistudio.io>

API Design

apigee {api studio: beta} File Download Preferences Help All changes saved

```
1 swagger: "2.0"
2 info:
3   version: "0.0.1"
4   title: Swagger API
5   host: playground.apistudio.io
6   basePath: /try/3d9b2768-f877-4dab-adb9-aaed8f99b3ae
7   schemes:
8     - http
9     - https
10 consumes:
11   - application/json
12 produces:
13   - application/json
14 paths:
15   /hello:
16     x-swagger-router-controller: hello_world
17     get:
18       description: Returns greetings to the caller
19       operationId: hello
20       responses:
21         "200":
22           description: Success
23           schema:
24             $ref: "#/definitions/HelloWorldResponse"
25         default:
26           description: Error
27           schema:
28             $ref: "#/definitions/ErrorResponse"
29 definitions:
30   HelloWorldResponse:
31     required:
32       - message
33     properties:
34       message:
35         type: string
36       age:
37         type: number
38   ErrorResponse:
39     required:
40       - message
```

API Response Simulation: ☒

GET /hello

Description

Returns greetings to the caller

Responses

Code	Description	Schema
200	Success	<div>↔</div> <div>▼ HelloWorldResponse { message: string * age: number }</div>
default	Error	<div>↔</div> <div>▼ ErrorResponse { message: string * }</div>

Close

Request

Scheme

http

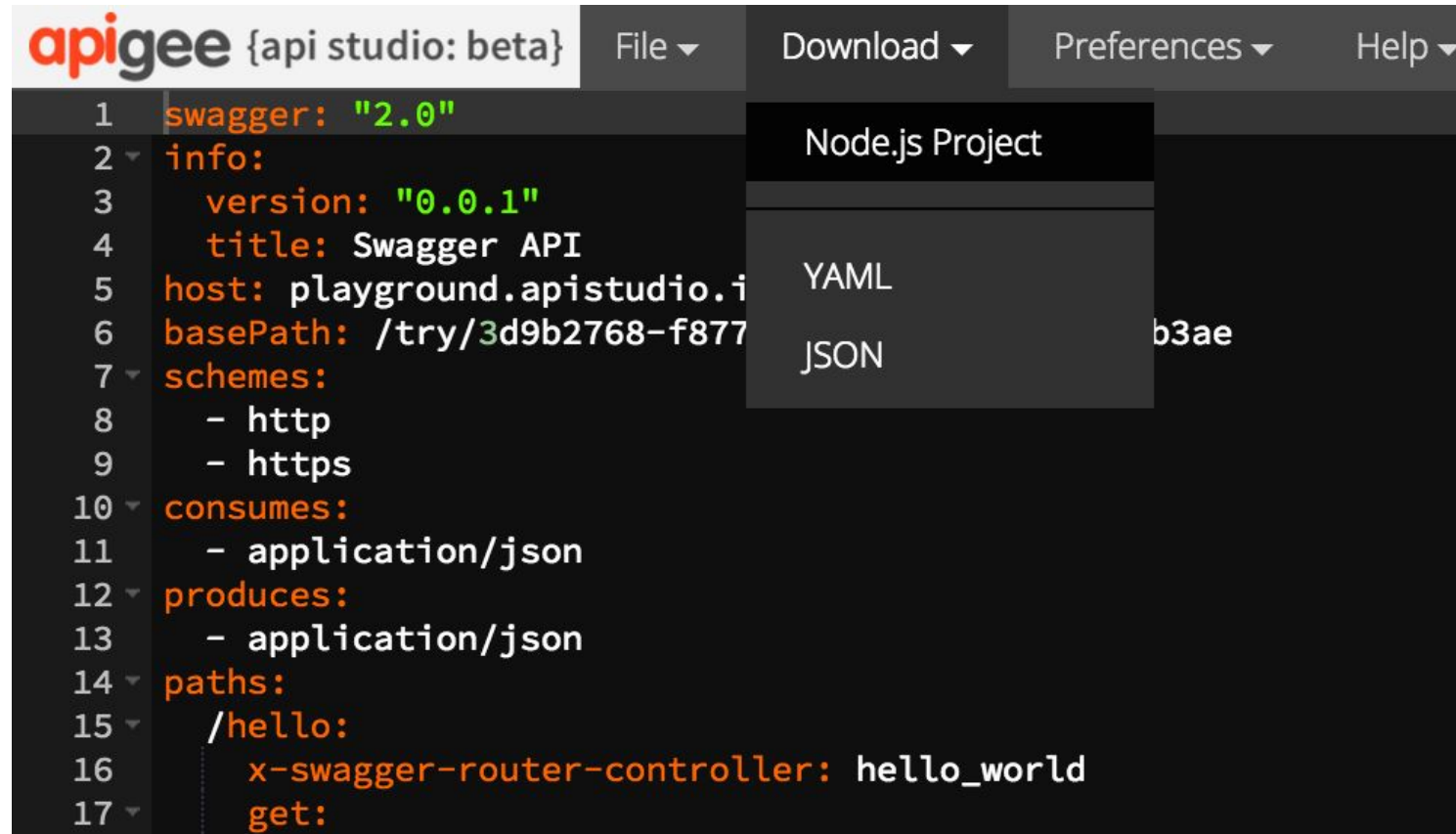
Accept

application/json

GET <http://playground.apistudio.io/try/3d9b2768-f877-4dab-adb9-aaed8f99b3ae/hello>
HTTP/1.1

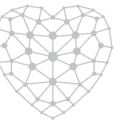
<http://apistudio.io>

API Design

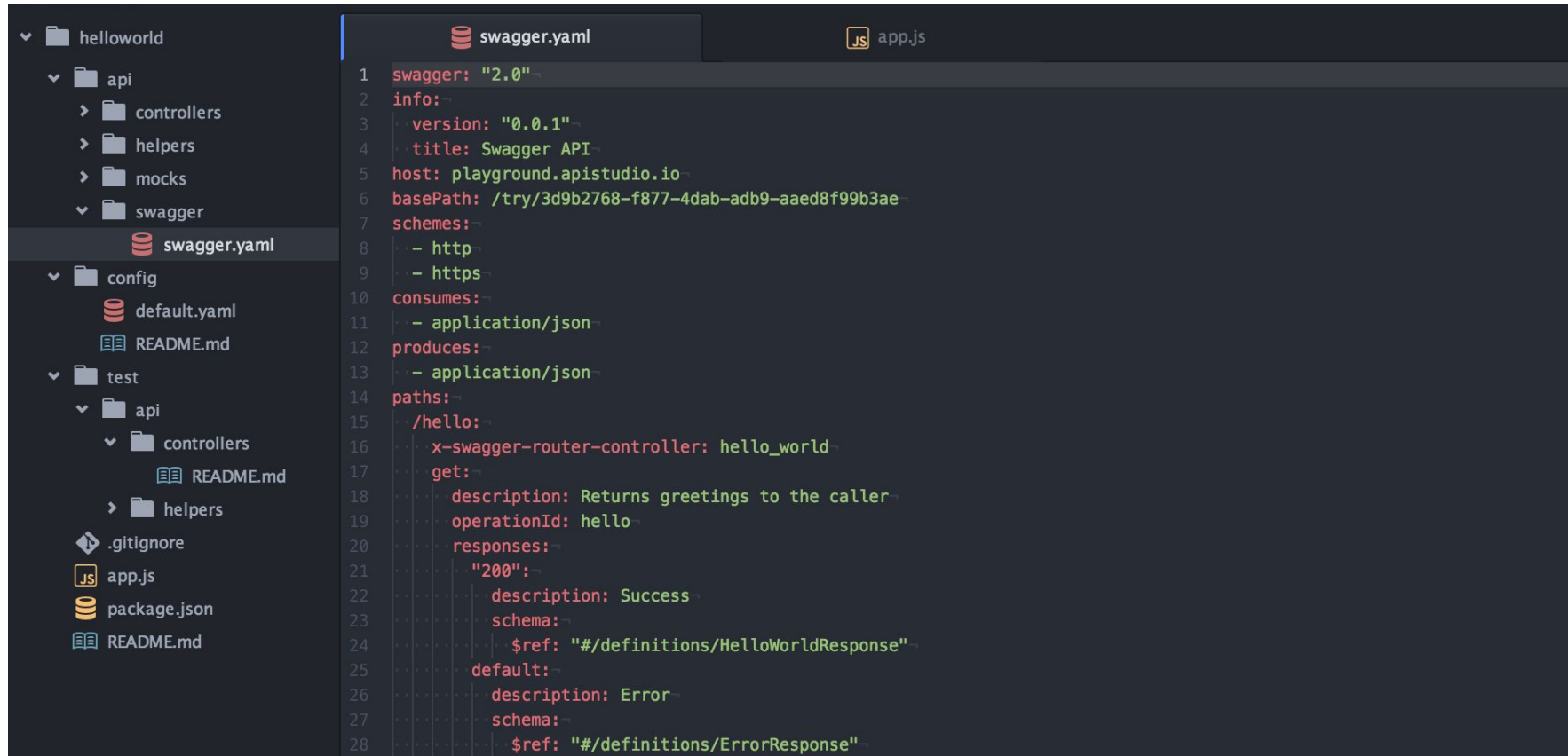


```
1 swagger: "2.0"
2 info:
3   version: "0.0.1"
4   title: Swagger API
5 host: playground.apistudio.io
6 basePath: /try/3d9b2768-f877
7 schemes:
8   - http
9   - https
10 consumes:
11   - application/json
12 produces:
13   - application/json
14 paths:
15   /hello:
16     x-swagger-router-controller: hello_world
17     get:
```

<http://apistudio.io>



Swagger Driven Node.js APIs

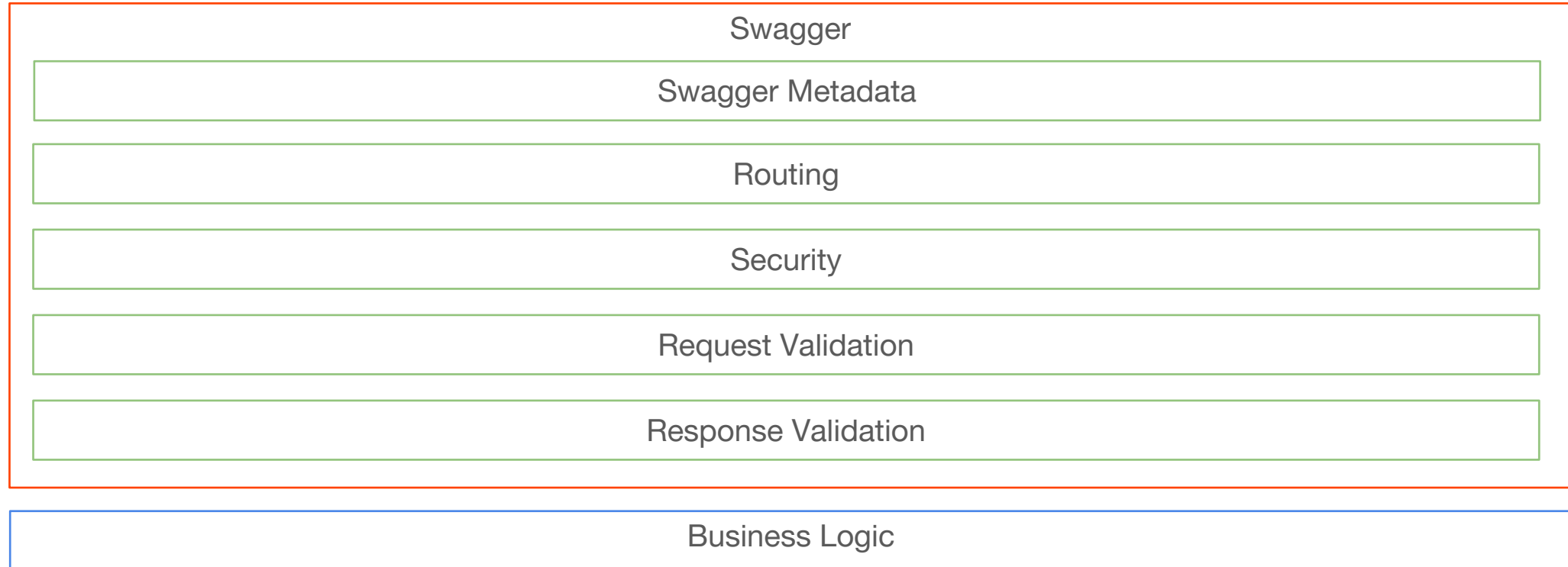


The screenshot shows a code editor with a project structure on the left and a `swagger.yaml` file open in the main editor. The project structure includes a `helloworld` directory with subdirectories `api` (containing `controllers`, `helpers`, `mocks`, and `swagger`) and `config` (containing `default.yaml` and `README.md`). There is also a `test` directory with `api` (containing `controllers` and `helpers`) and other files like `.gitignore`, `app.js`, `package.json`, and `README.md`. The `swagger.yaml` file is configured with the following content:

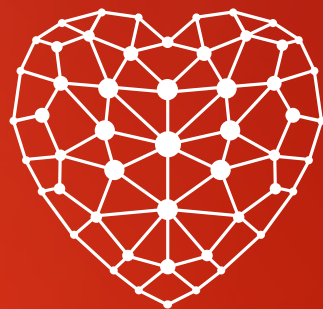
```
1 swagger: "2.0"
2 info:
3   version: "0.0.1"
4   title: Swagger API
5   host: playground.apistudio.io
6   basePath: /try/3d9b2768-f877-4dab-adb9-aaed8f99b3ae
7   schemes:
8     - http
9     - https
10  consumes:
11    - application/json
12  produces:
13    - application/json
14  paths:
15    /hello:
16      x-swagger-router-controller: hello_world
17      get:
18        description: Returns greetings to the caller
19        operationId: hello
20        responses:
21          "200":
22            description: Success
23            schema:
24              $ref: "#/definitions/HelloWorldResponse"
25          default:
26            description: Error
27            schema:
28              $ref: "#/definitions/ErrorResponse"
```

`npm install -g swagger`

Swagger Driven Node.js APIs



```
npm install -g swagger
```

Thank You