API SPECIFICATION LANGUAGE ESSENTIALS

Daniel Schröder, Freelancer, Web + Dev + Ops

OUTLINE

1. Intro
2. Tools
    1. OpenAPI
    2. RAML
    3. API Blueprint
3. Examples
4. Synopsis
5. References

DESIGN FIRST APPROACH

So why do we need to specify our API?

Because otherwise this will happen!

BENEFITS OF SPECIFYING AN API

- interface between two or more eco systems
  - try-and-error is pain in the ass
  - minimise future legacy payload
  - better performance, usability and reliability
  - optimize number of necessary API calls
  - bigger changes often results in bc-breaks
  - easier human and machine consumption

API SPECIFICATION CONTENTS

- Metadata
- Versioning
- Caching
- Authorization (access policy definition)
- Authentication (identity verification)
- Request methods and parameters
- Response status and payload
- Error handling

ADVANTAGES OF NOT USING NOTEPAD FOR SPECIFICATION

- tools are using standardized source formats
  - linting, syntax checks, validation, ...
  - can be rendered as HTML for end users
- is readable by machines
  - to serve some mock data
  - available for code generators (interfaces, schemas, ...)
  - ready to get used for automated tests

API SPECIFICATION TOOLS

	OpenAPI	RAML	API Blueprint
Format	JSON	YAML	Markdown
Version	2.0	1.0	1A9
Released	Sep 2014	Jul 2016	Jun 2015
License	Apache-2.0	Apache-2.0	MIT
Stars	6.6k	2.8k	5.5k

OPENAPI

- API specification defined in JSON
- Swagger core tools:
  - Swagger Editor: online editor
  - Swagger Codegen: code generation binary
  - Swagger UI: renderer, browser and sandbox
- community driven tools:
  - Spectacle: HTML renderer
  - pokemock: mock server
  - got-swag: automated tests
  - and many more...

RAML

- API specification defined in YAML
- official tools:
  - API Workbench: editor (Atom plugin)
- project library:
  - raml2html: RAML to HTML generator
  - Osprey Mock Service: API mock service
  - Abao: automated testing tool
  - and many more...

API BLUEPRINT

- API specification defined in Markdown
- popular tools:
  - Apiary: editor, mocks, tests, inspections
  - Aglio: HTML renderer
  - Drakov: mock server
  - Dredd: testing framework
  - And many more...

API SPECIFICATION EXAMPLES

OPENAPI

```
{
  "swagger": "2.0",
  "info": {
    "title": "TodoMVC",
    "description": "TodoMVC API is a Todo storage backend.",
    "version": "1.0.0"
  },
  "host": "api.todomvc.com",
  "schemes": [
    "http"
  ],
  "definitions": {},
  "paths": {}
}
```

EXAMPLES

RAML

```
#%RAML 1.0
title: TodoMVC
description: TodoMVC API is a Todo storage backend.
version: 1.0.0
baseUri: http://api.todomvc.com

types:
  ...

...
```

EXAMPLES

API BLUEPRINT

```
FORMAT: 1A
HOST: http://api.todomvc.com

# TodoMVC API documentation

TodoMVC API is a Todo storage backend.

# Data Structures

...
```

SYNOPSIS

- best tool? depends!
  - every tool has integrations for common languages
  - all of these tools have: renderers, mock servers, testing utilities, ...
- OpenAPI seems to be the oldest and most popular tool
- some RAML tools are not ready for version 1.0 yet
- API Blueprint has the advantage that it is human readable without conversion

QUESTIONS?

Thank you!

REFERENCES

- http://imgur.com/gallery/Gv3UU40
- http://www.memegen.com/meme/9kf03c
- https://github.com/
- https://www.openapis.org/
- https://swagger.io/
- https://raml.org/
- http://apiworkbench.com/
- https://apiblueprint.org/
- https://apiary.io/