関西 Ruby 会議 06

API server/client development using JSON Schema

Masayuki IZUMI - @izumin5210

Masayuki IZUMI - @izumin5210

うさみみ東大院生（M1）/ うさみみ高専生

Rekimoto Lab. at the University of Tokyo
(2008-2015: Akashi-NCT)

Enginner at **Wantedly, Inc.**
(2014.9-2015.2: Dmetlabel, Inc.)

Kobe.rb

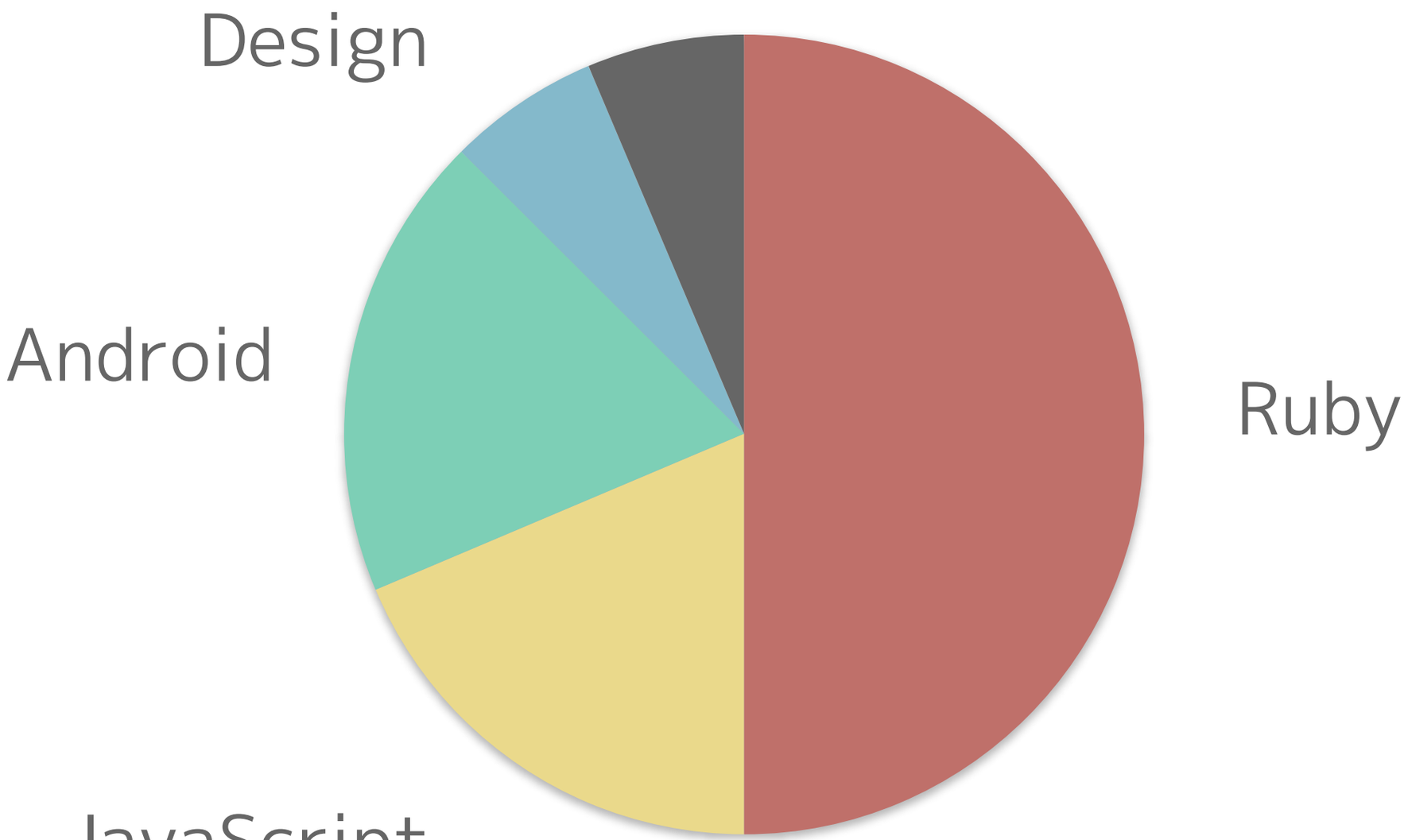
Asakusa.rb

https://www.flickr.com/photos/takkanm/3978417669/
https://www.flickr.com/photos/suzumenonamida/8213183210/

We are hiring !

https://www.wantedly.com/companies/wantedly/projects

pry(main)> izumin.skill_ratio



ぃずみん @izumin5210 · Jun 19

趣味: JSON SchemaとRuby on Railsを使ったAPIの設計・開発

3

「JSON Schema ?」

ときどき見かけるけど，何がうれしいの？

Q: What does it do?

JSON Schema ってなにすんの？

A: It describes your JSON data format[1].

JSON のフォーマット記述するやつやで.

[1] http://json-schema.org/

What is JSON Schema ?

* Defines structure of JSON data
  - JSON Schema Core

* Provides structual validation
  - JSON Schema Validation

* Defines hyper{text,media} related things
  - JSON Hyper Schema

ここから JSON Schema の簡単な説明！

e.g. JSON Schema Core

```yaml
title: User
properties:
  email:
    description: email address
    type: string
  name:
    description: user name
    type: string
```

```json
{
  "email": "syaro@example.com",
  "name": "Syaro Kirima"
}
```

e.g. JSON Schema Core

```yaml
title: User
properties:
  email:
    description: email address
    type: string
  name:
    description: user name
    type: string
```

```json
{
  "email": "syaro@example.com",
  "name": "Syaro Kirima"
}
```

e.g. JSON Schema Core

```yaml
title: User
properties:
  email:
    description: email address
    type: string
  name:
    description: us
    type: string
```

```
7 primitive types:
    array, boolean, integer,
    number, null, object, string
```

```json
{
  "email": "syaro@example.com",
  "name": "Syaro Kirima"
```

e.g. JSON Schema Validation

```yaml
title: User
properties:
  email:
    description: email address
    type: string
    format: email
  name:
    description: user name
    type: string
    maxLength: 32
required:
  - email
  - name
```

```json
{
  "email": "syaro@example.com",
  "name": "Syaro Kirima"
}
```

e.g. JSON Schema Validation

```yaml
title: User
properties:
  email:
    description: email address
    type: string
    format: email
  name:
    description: user name
    type: string
    maxLength: 32
required:
  - email
  - name
```

```json
{
  "email": "syaro@example.com",
  "name": "Syaro Kirima"
}
```

e.g. JSON Schema Validation

```
title: User
properties:
  email:
    description: email address
    type: string
    format: email
  name:
    description: user na
    type: string
    maxLength: 32
required:
  - email
  - name
```

```
{
                                    ample.com",
                                    ima"
```

Defined validation keywords:

maximum, maxLength, pattern,
maxItems, enum, allOf, anyOf,
definitions, ...

e.g. JSON Hyper-Schema

```yaml
title: User
links:
  - href: "/api/users"
    method: GET
    rel: instances
    title: List
properties:
  # define some properties
```

```
[
  {
    "email": "cocoa@example.com",
    "name": "Cocoa Hoto"
  },
  {
    "email": "syaro@example.com",
    "name": "Syaro Kirima"
  }
]
```

e.g. JSON Hyper-Schema

```yaml
title: User
links:
  - href: "/api/users"
    method: GET
    rel: instances
    title: List
properties:
  # define some properties
```

response body from `GET /api/users`

```
[
  {
    "email": "cocoa@example.com",
    "name": "Cocoa Hoto"
  },
  {
    "email": "syaro@example.com",
    "name": "Syaro Kirima"
  }
]
```

e.g. JSON Hyper-Schema

```yaml
title: User
links:
  - href: "/api/users"
    method: POST
    rel: craete
    schema:
      properties:
        name:
          "$ref": "#/properties/name"
    title: Create
properties:
  # define some properties
```

e.g. JSON Hyper-Schema

```yaml
title: User
links:
  - href: "/api/users"
    method: POST
    rel: craete
    schema:
      properties:
        name:
          "$ref": "#/properties/name"
    title: Create
properties:
  # define some properties
```

JSON Schema(subschema) for request parameter

e.g. JSON Hyper-Schema

```
title: User
links:
  - href: "/api/users"
    method: POST
    rel: craete
    schema:
      properties:
        name:
          "$ref": "#/properties/name"
    title: Create
properties:
  # define some properties
```

```
"$ref": "#/foo/bar"  JSON Reference[2]
#/foo/bar            JSON Pointer[3]
```

[2] http://tools.ietf.org/html/draft-pbryan-zyp-json-ref-03

[3] http://tools.ietf.org/html/rfc6901

e.g. JSON Hyper-Schema

```yaml
title: User
links:
  - href: "/api/users/{(%23%2Fproperties%2Fname)}"
    method: GET
    rel: instance
    title: self
properties:
  # define some properties
```

e.g. JSON Hyper-Schema

```yaml
title: User
links:
  - href: "/api/users/{(%23%2Fproperties%2Fname)}"
    method: GET
    rel: instance
    title: self
properties:
  # define some properties
```

```
"/foo/bar/{(%23baz)}"      URI Template[4]
```

[4] http://tools.ietf.org/html/rfc6570

「何がうれしいの？」

JSON データのフォーマット定義出来るだけ？

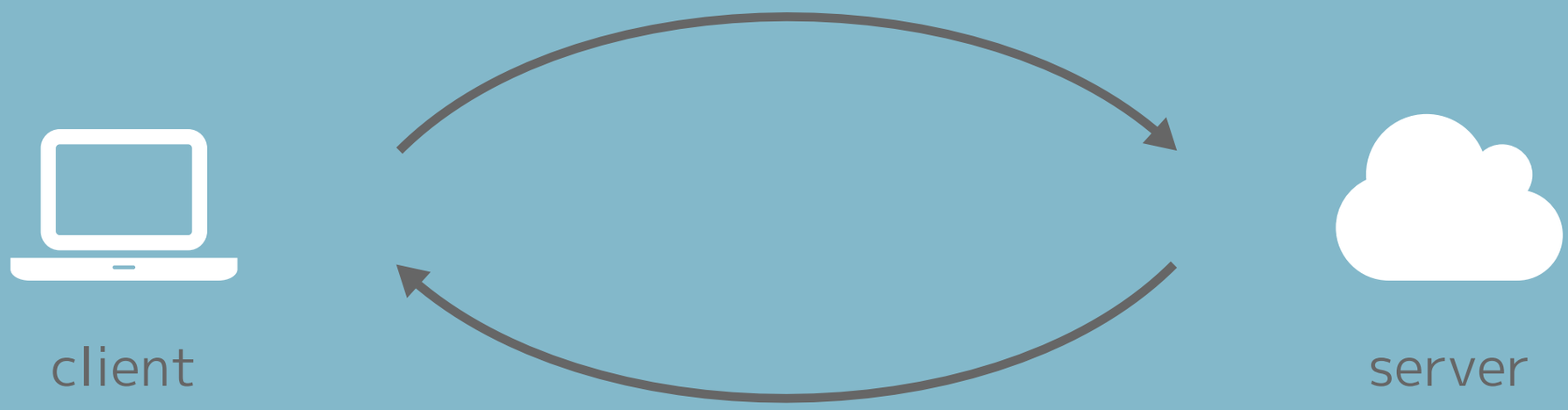
API のドキュメントにもなるよー的な？

できること その①

Validate request parameters

POST /api/posts

{ "body": " 大阪なう ", "user_id": 1 }



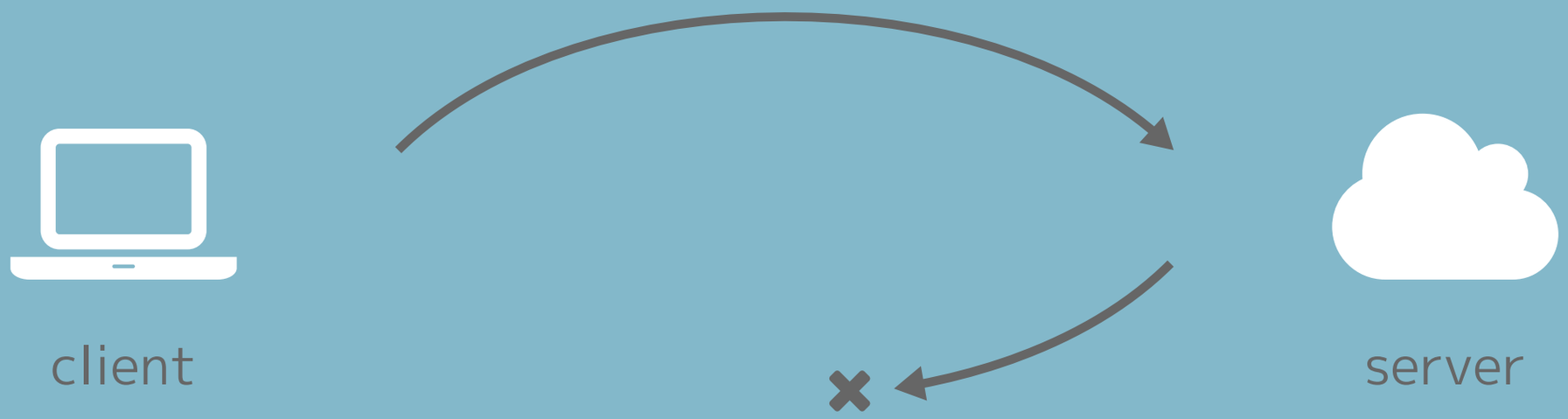
よく取り上げられる JSON Schema 利用法シリーズ

できること その②

Validate API response body

POST /api/posts

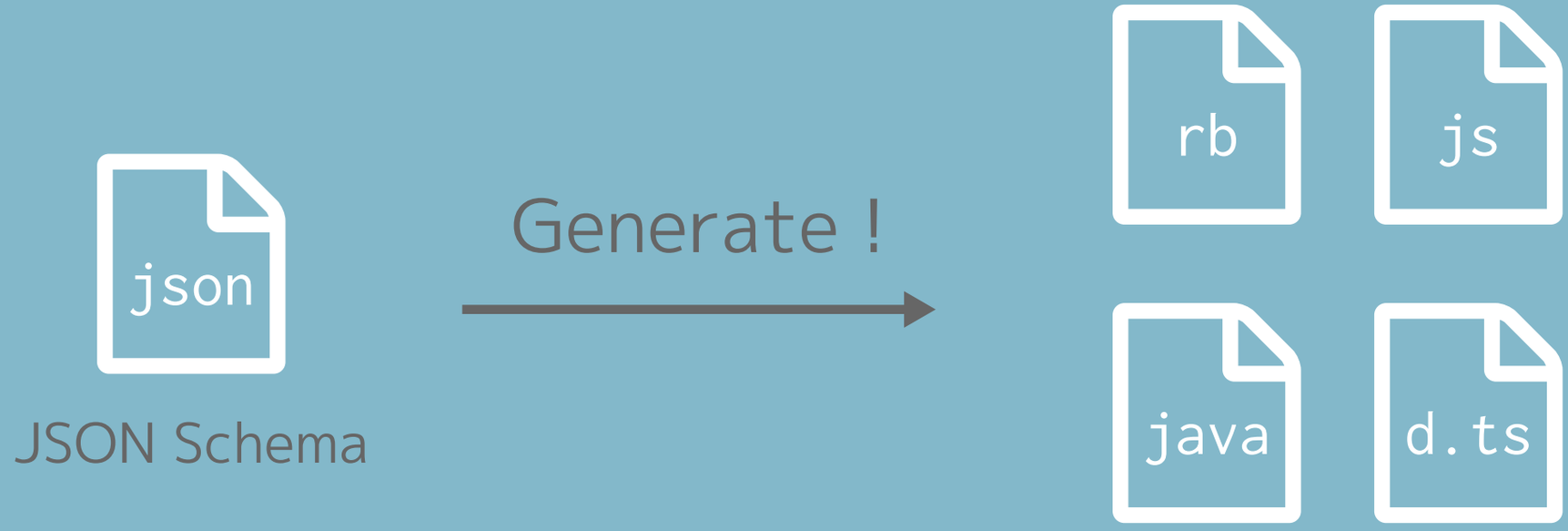
{ "body": " 大阪なう " }



レスポンスの形式間違ってるから 500

できること その③

Generate API client



できること その④

Generate API document

What can JSON Schema do ?

* Validates request parameter
  - rack-json_schema が便利
* Validates response body
  - rack-json_schema が便利
* Generates API client
  - いろんな言語，環境に存在
* Generates API document
  - prmd, jdoc がいい感じ

What can **JSON Schema** do ?

\ Strong Parameter でいいじゃん /

* Validates request parameter

\ テスト書いてたらいらないじゃん /

* Validates response body

\ API の変更あったらどうすんの /

* Generates API client

* Generates API document

「つまり、JSONSchema はオワコンだったんだよ！！」

ΩΩΩ < な、なんだってー !!

~~「つまり、JSONSchemaはオワコンだったんだよ！！」~~

~~ΩΩΩ＜な、なんだってー!!~~

「ホントにうれしいポイント」

値の validation ？ クライアントやドキュメントの生成？

JSON Schema による Web API 設計の最大の利点はそこじゃない！

突然ですが…

Ruby のイケてるところ といえば？

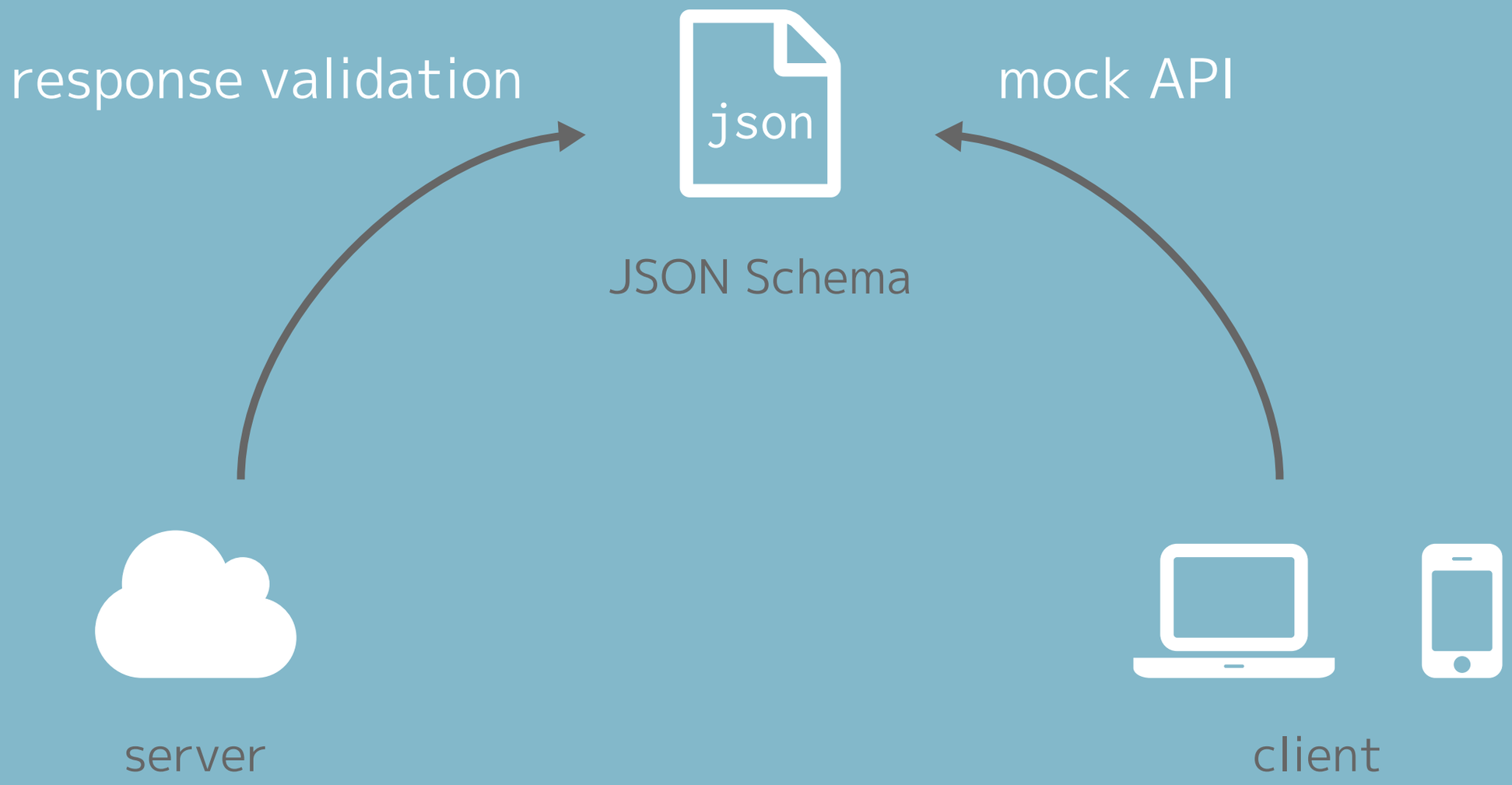
突然ですが…

Ruby のイケてるところ といえば？

» **テストの書きやすさ** ですよね？ [要出典]

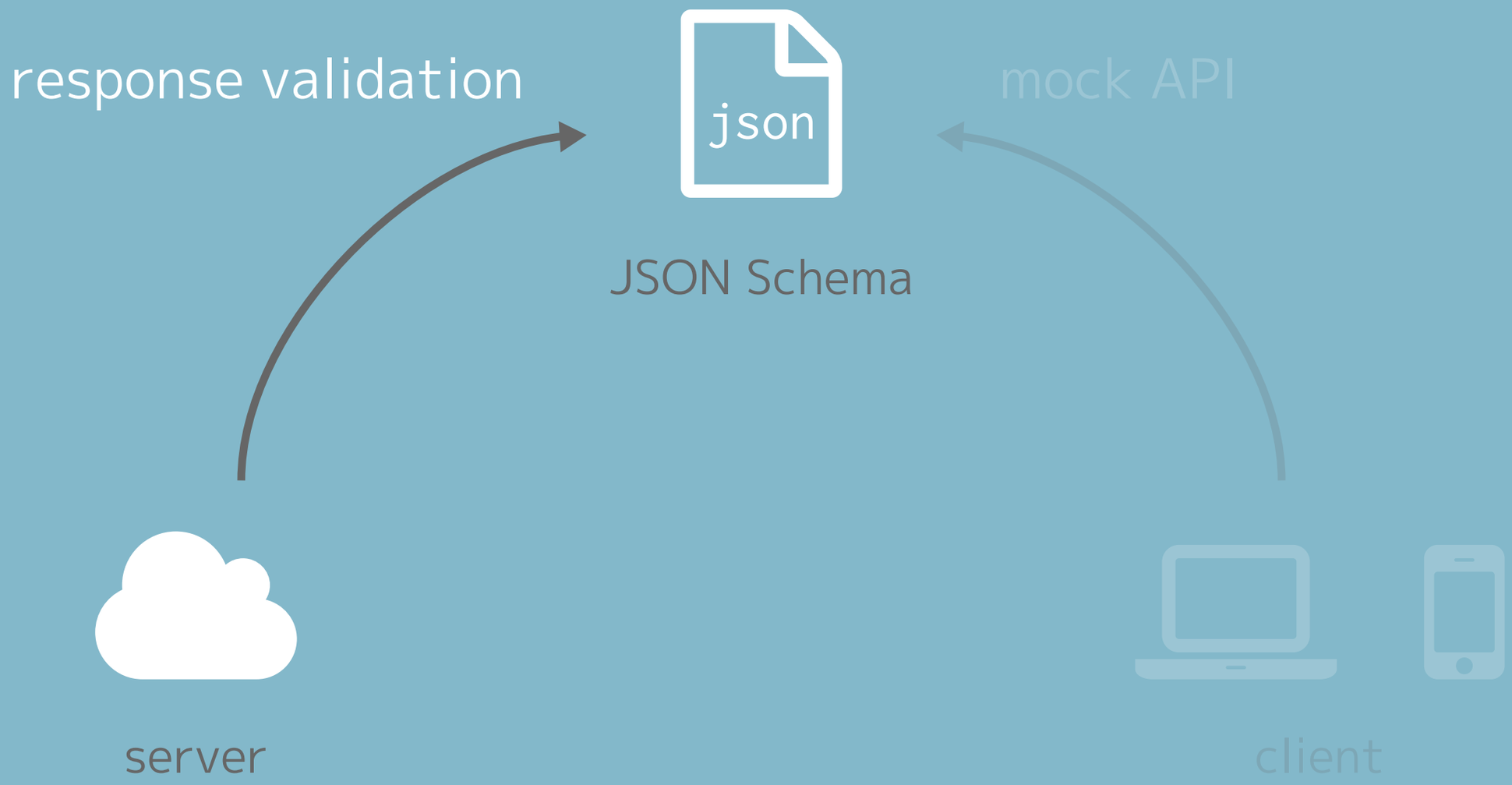
JSON Schema の 一番イケてる使い方

テスト支援ツールとしての JSON Schema



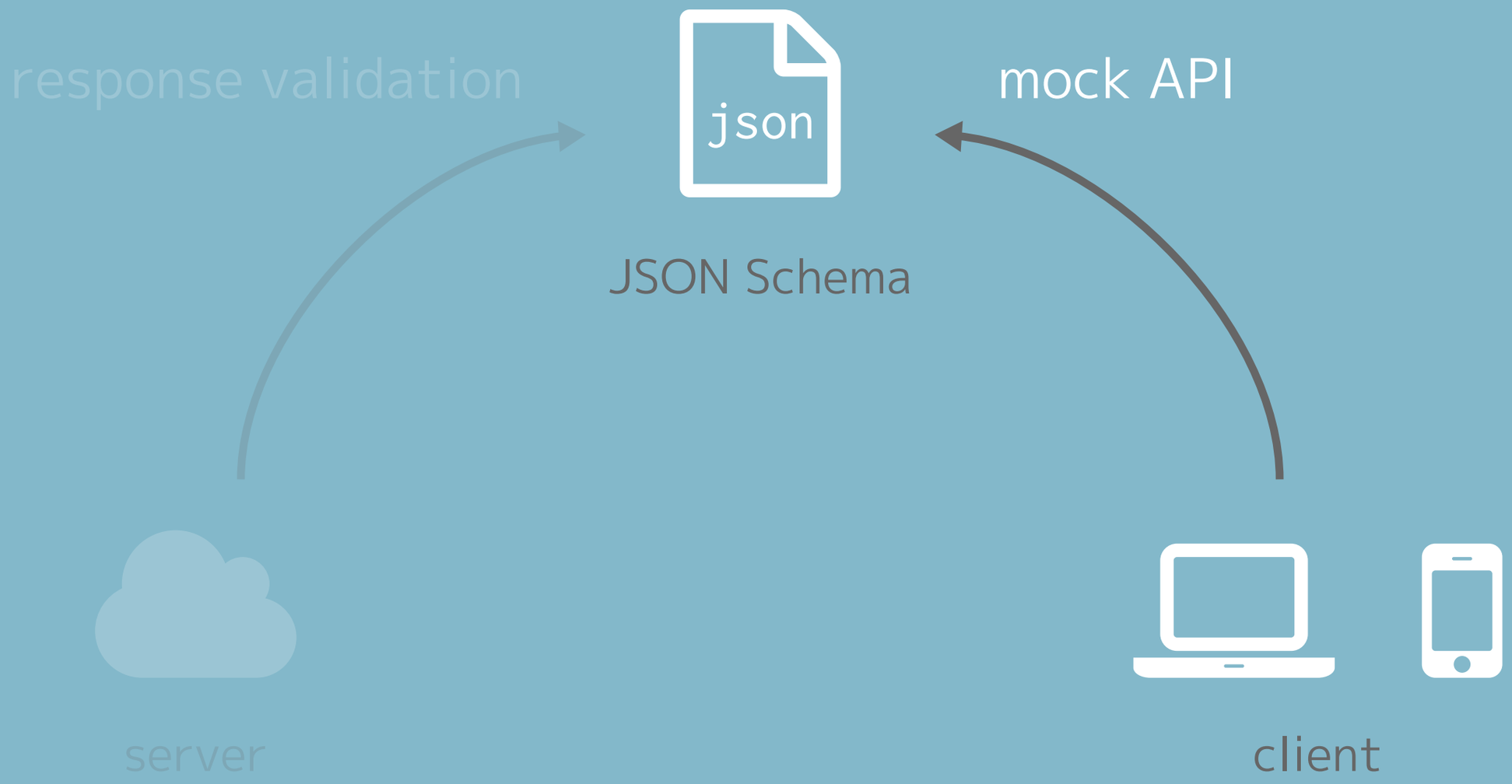
JSON Schema の 一番イケてる使い方

テスト支援ツールとしての JSON Schema



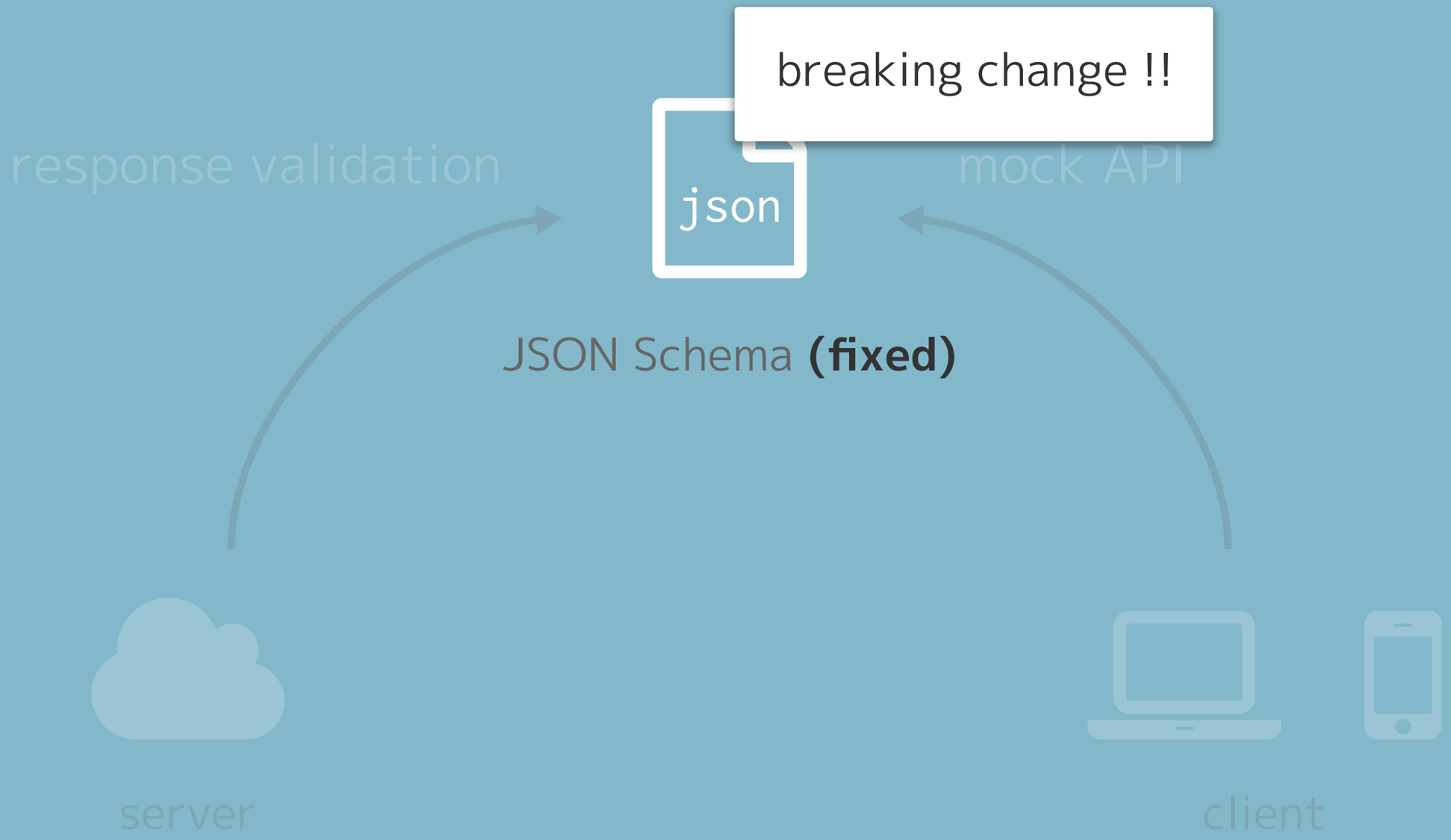
JSON Schema の 一番イケてる使い方

テスト支援ツールとしての JSON Schema



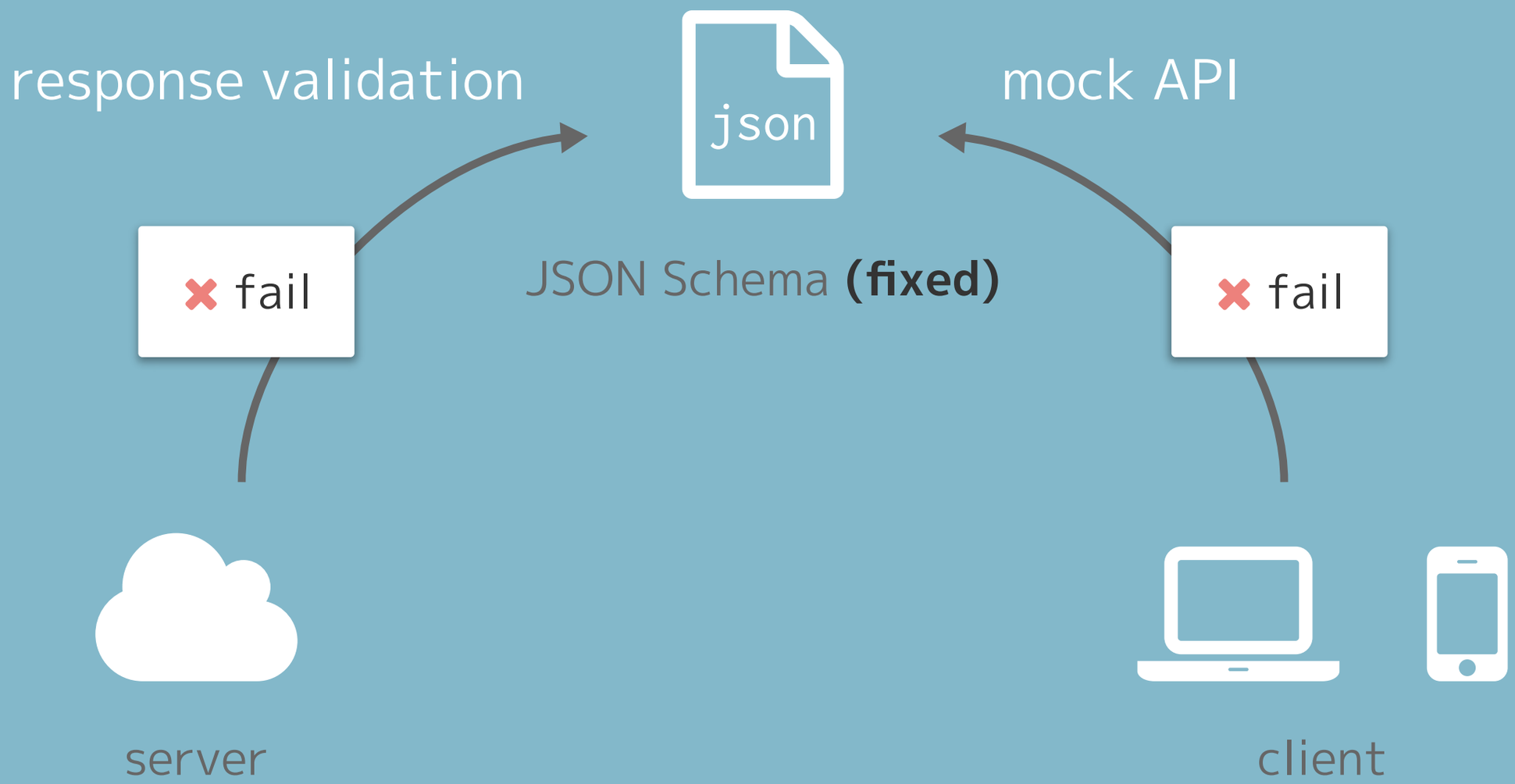
JSON Schema の 一番イケてる使い方

テスト支援ツールとしての JSON Schema



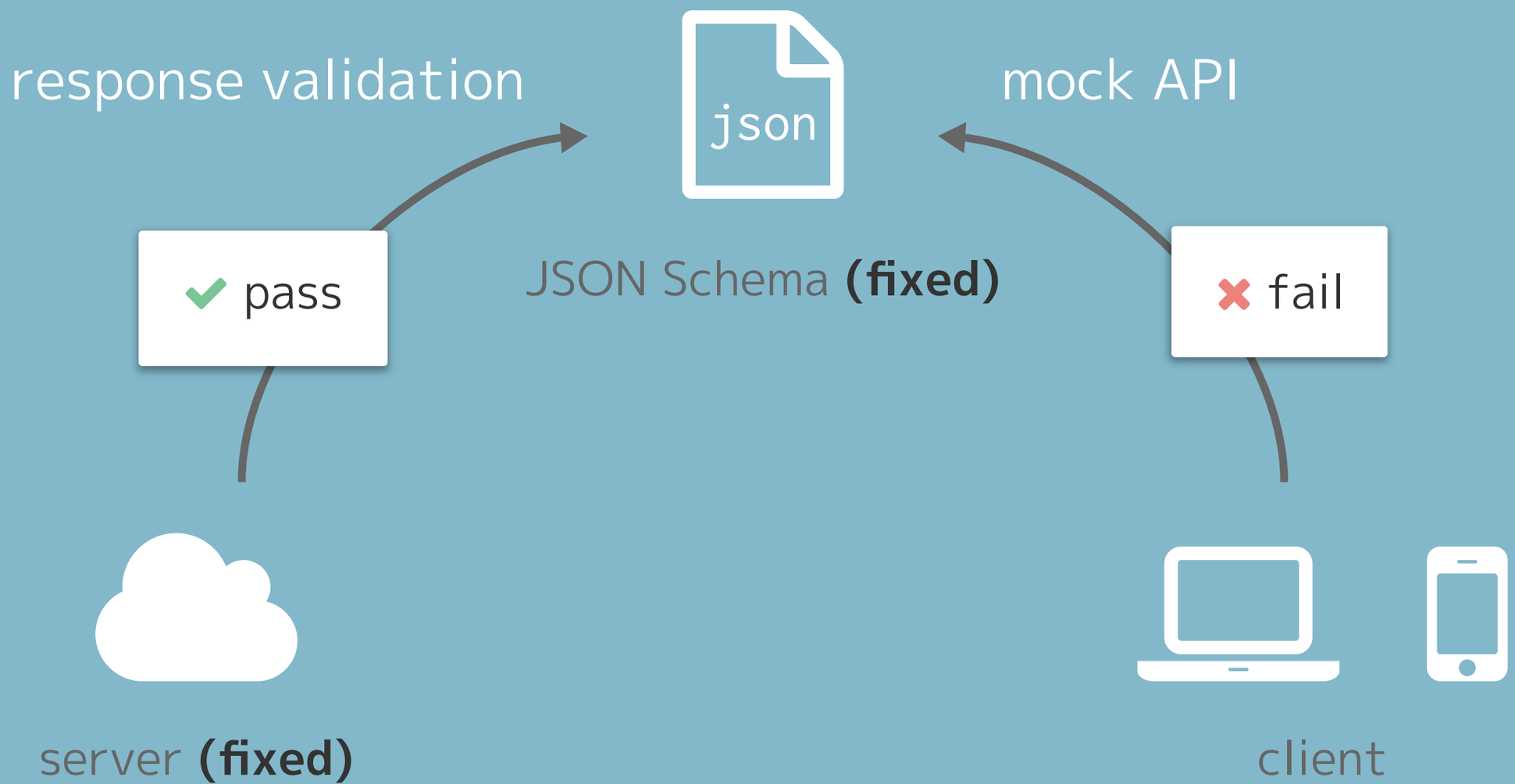
JSON Schema の 一番イケてる使い方

テスト支援ツールとしての JSON Schema



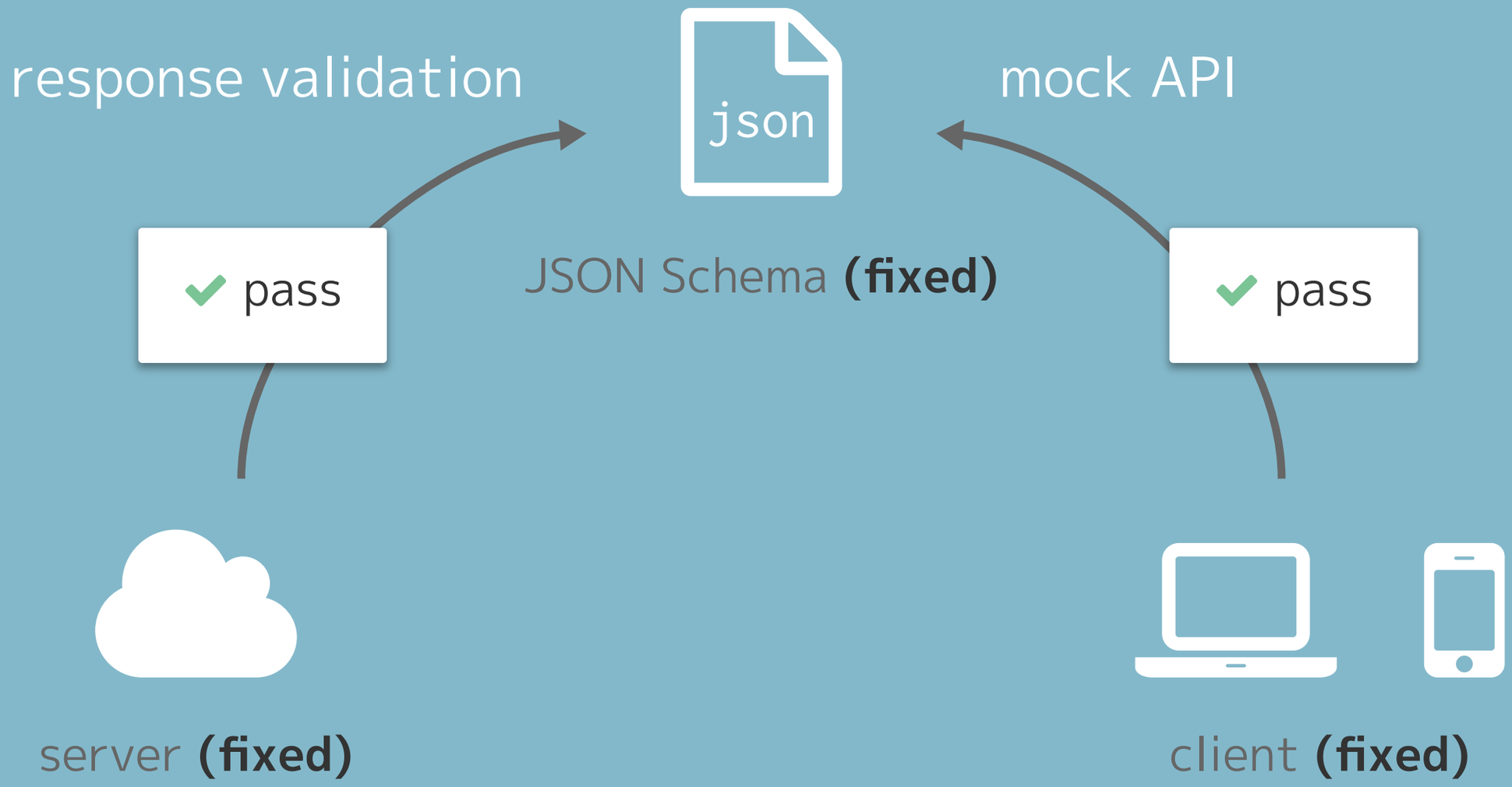
JSON Schema の 一番イケてる使い方

テスト支援ツールとしての JSON Schema



JSON Schema の 一番イケてる使い方

テスト支援ツールとしての JSON Schema



仕様変更は JSON Schema から

Server/Client どちらも **JSON Schema** を
テストに利用する（**validation/mock**）

仕様変更があれば**テストが落ちる**から
対応漏れがなくなる（**CI しよう**）

その他 JSON Schema じょうほう

* Ruby 界隈は JSON Schema がわりと盛ん
  - Heroku さん Increments さんありがとうございます
  - おもしろ gem やイイ感じの知見が多い

その他 JSON Schema じょうほう

* JS 界隈は…
  - JSON（**JavaScript** Object Notation）とは何だったのか
  - 自分で node module 作るハメになった
  - **json-schema-parser** / **json-schema-mockifier**
  - 僕の屍を越えて行ってください

「JSON 手書きつらい…」

なんなんだこの大量の {} と "" は…　ド M かよ…

Write JSON Schema in YAML

interagent/prmd

```
# -y: yaml として出力
$ prmd init -y User > doc/schemata/user.yml
$ prmd init -y Post > doc/schemata/post.yml

# 1 つの .json に結合
$ prmd combine --meta doc/meta.yml -o doc/schema.json doc/schemata
```

YAML も大差ない…

$ref ややこしい…

Write JSON Schema in DSL

r7kamura/json_world

Ruby の DSL で Schema が定義できる

type に JsonWorld なクラスを渡せる

（**$ref** からの脱却！！）

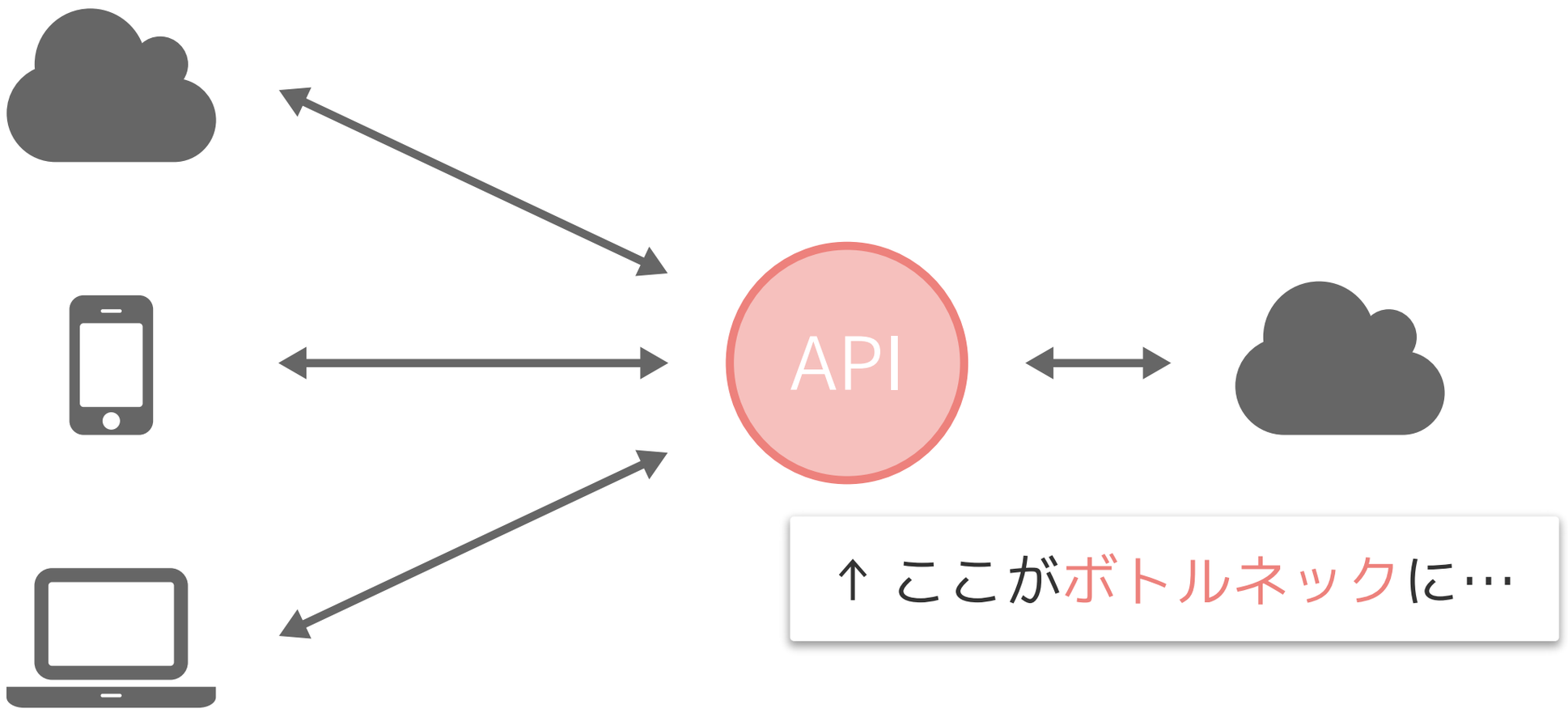
「Conclusion」

JSON Schema たのしい！（ぐるぐる目

Why JSON Schema ?

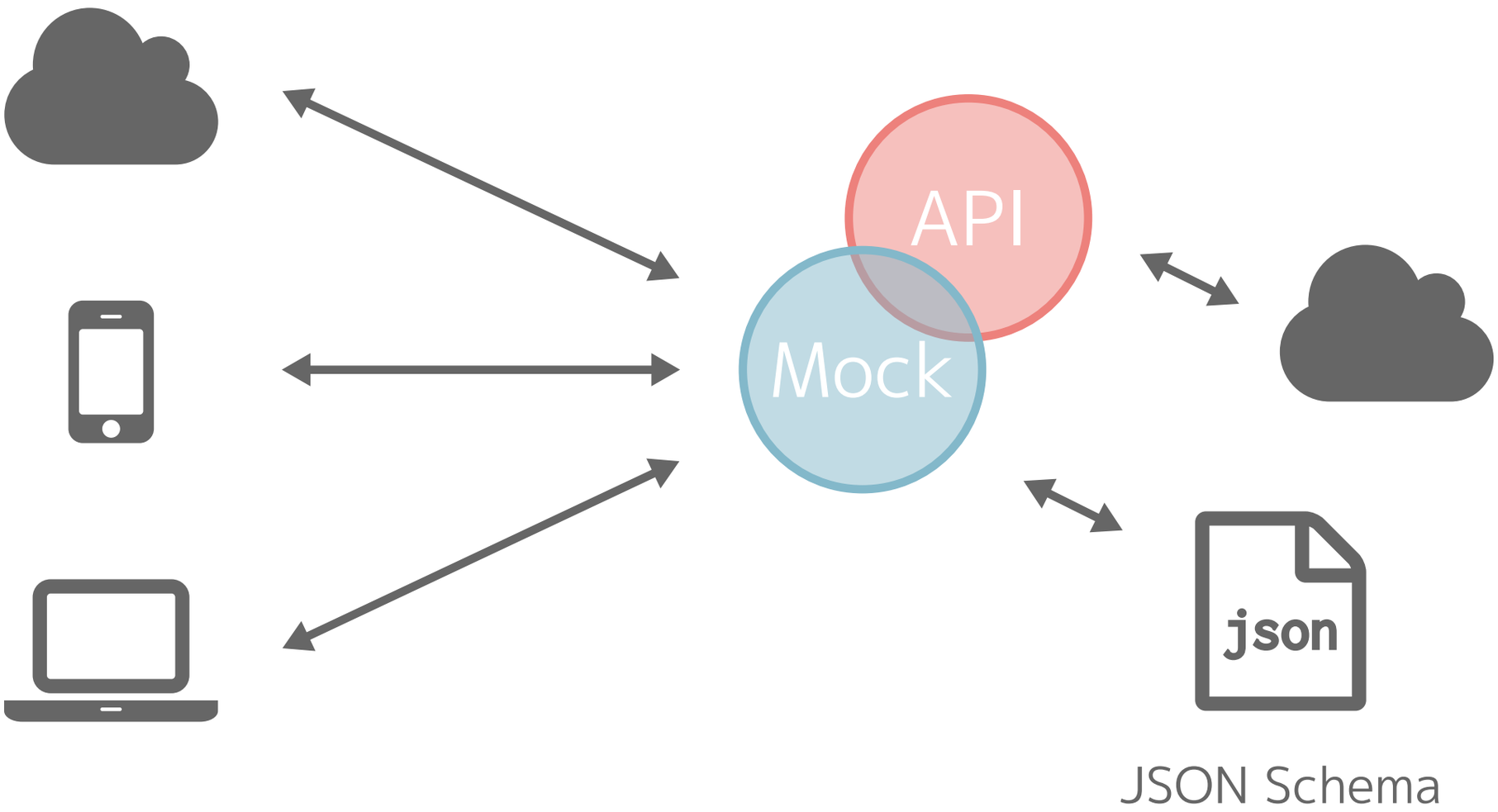
```
KEYWORDS = [
  "microservices", "mobile first", "SPA"
]
```

Why JSON Schema ?



↑ ここがボトルネックに…

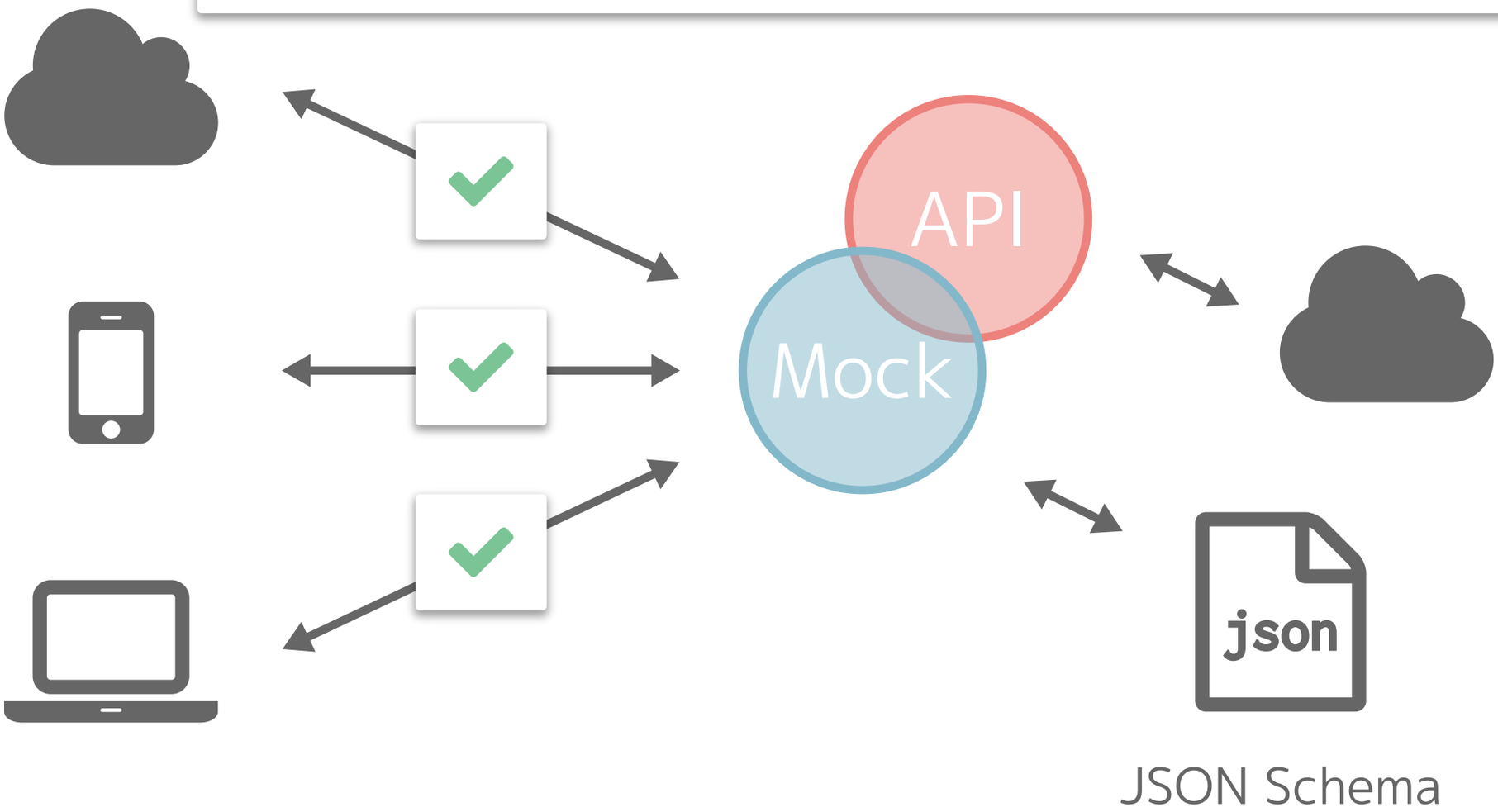
Why JSON Schema ?



JSON Schema

Why JSON Schema ?

テスト・開発が他サービスに影響されない



JSON Schema

ぃずみん @izumin5210 · Jul 9

JSON Schema（Hyper Schema）でドキュメント作りたいんじゃない！
開発者が変更する必要のないコードを生成したいのでもない！
変更に！！！！強い！！！！！！！！テストが！！！！！！！！！！書きたいの！！！！！！！！！！！！！

2

ぃずみん @izumin5210 · Jul 9

その辺の話をSOAPでつらいところいっぱい見てきたオジサンたちとしっかり議論してみたい

2

Conclusion - Why **JSON Schema** ?

JSON Schema の価値

- API をモック化することでボトルネックにならない
- ちゃんとテスト回せば変更にも強い
- クライアント側もちゃんとテスト書こう！
- 実装に JSON Schema という " 制約 " をつける


WANTEDLY
https://www.wantedly.com/companies/wantedly/projects