

CYPRESS-AJV-SCHEMA-VALIDATOR PLUGIN VIDEO TUTORIAL



JSON Schemas {



JSON Schema {X}

“JSON Schema is a declarative language for defining structure and constraints for JSON data.”

(*) Source: json-schema.org

```
{  
  "$schema": "https://json-schema.org/draft/2020-12/schema",  
  "title": "Example Schema",  
  "type": "object",  
  "properties": {  
    "id": {  
      "type": "integer",  
      "description": "An integer value representing the ID"  
    },  
    "name": {  
      "type": "string",  
      "description": "A string value representing the name"  
    },  
    "tags": {  
      "type": "array",  
      "description": "An array of strings representing tags",  
      "items": {  
        "type": "string"  
      }  
    },  
    "required": [  
      "id",  
      "name"  
    ]  
  }  
}
```



OpenAPI Document



“An OpenAPI Document is a single JSON or YAML document that conforms to the OpenAPI Specification.”

(*) Source: spec.openapis.org

- The OpenAPI Specification (formerly Swagger Specification) are schema documents to describe your entire API specification.
- It will contain multiple schemas definitions, one for each supported combination of **Endpoint - Method - Status** (also called **Path**) by that API.
- Swagger 2.0 is an older version of OpenAPI 3.



OpenAPI Sample



Path:

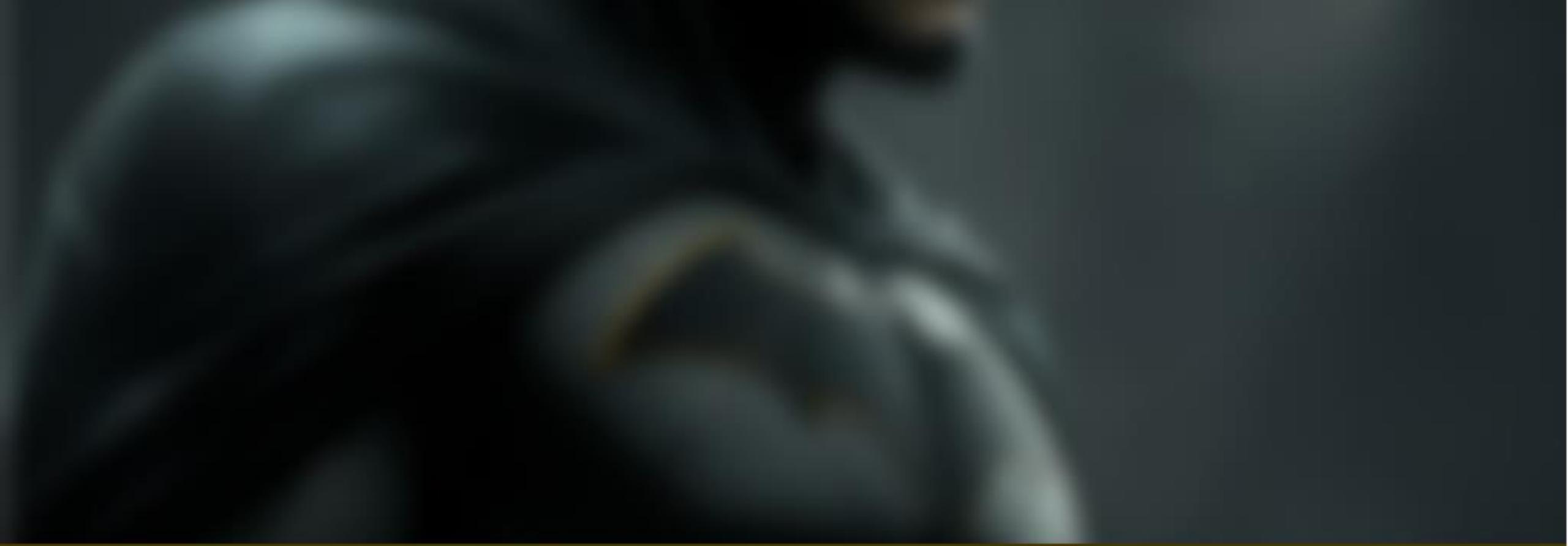
- Endpoint
- Method
- Status

Schema

```
{  
  "openapi": "3.0.1",  
  "info": {  
    "title": "They Said So Quotes API",  
    "description": "They Said So Quotes API offers a complete feature rich REST API access.",  
    "version": "3.1"  
  },  
  "paths": {  
    "/god": {  
      "get": {  
        "description": "Gets `Quote of the Day`\n",  
        "parameters": [ ... ],  
        "responses": {  
          "200": {  
            "content": {  
              "application/json": {  
                "examples": {  
                  "response": {  
                    "value": "{\n                      \"success\": {\n                        \"total\": 1\n                      },\n                      \"contents\": {\n                        \"quotes\": [\n                          {\n                            \"author\": \"Albert Einstein\",  
                            \"quote\": \"The only source of knowledge is experience.\",  
                            \"tags\": [\n                              \"Science\",  
                              \"Philosophy\",  
                              \"Wisdom\"\n                            ]  
                          }\n                        ]\n                      }\n                    }  
                  }  
                }  
              }  
            },  
            "description": "200 response"  
          },  
          "400": {  
            "content": {  
              "application/json": {  
                "examples": {  
                  "response": {  
                    "value": "{\n                      \"error\": \"Bad Request\"\n                    }  
                  }  
                }  
              }  
            }  
          }  
        }  
      }  
    }  
  },  
  "components": {  
    "schemas": {  
      "Quote": {  
        "properties": {  
          "author": {  
            "description": "Author name of quote.",  
            "type": "string"  
          },  
          "quote": {  
            "description": "The Quote.",  
            "type": "string"  
          },  
          "tags": {  
            "description": "Array of tags/categories.",  
            "items": {  
              "type": "string"  
            },  
            "type": "array"  
          }  
        },  
        "required": [  
          "quote"  
        ]  
      }  
    }  
  }  
}
```

\$ref to Components > Schema





What is cypress-ajv-schema-validator?



An Open-source JSON Schema Validator Plugin for



- ⚖️ The ultimate “vigilante” for your API Schemas
- ⚖️ Leverages the capabilities of the **AJV** plugin
- 📦 Supports **JSON Schema**, and **OpenAPI 3/Swagger 2 documents**
- 📖 Results provided in a comprehensive and user-friendly way
- 🤝 Seamless integration with **@bahmutov/cy-api** and **cypress-plugin-api** plugins



cypress-ajv-schema-validator

A screenshot of a GitHub repository page. The repository name is "cypress-ajv-schema-validator" by user "sclavijosuero". The "Code" tab is selected. The main content area shows a list of commits from the "main" branch. The commits are as follows:

- sclavijosuero Updated readme.md including the newly added types for com... d60cd32 · 2 weeks ago 59 Commits
- .github/workflows Create main.yml 6 months ago
- cypress Updated readme.md including the newly added types for... 2 weeks ago
- images v1.3.0 2 months ago
- src Updated readme.md including the newly added types for... 2 weeks ago
- videos Upload video of integration with cy.api() plugins 2 months ago
- .gitignore Completed some documentation and added images 2 months ago
- .npmrc pass on failure 2 weeks ago
- LICENSE Initial commit 6 months ago
- README.md Updated readme.md including the newly added types for... 2 weeks ago
- USAGE-EXAMPLES.md Fix some typos in documentation 6 months ago
- cypress.config.js Code, documentation and images for v1.2.0 2 months ago
- cypress.env.json Code, documentation and images for v1.2.0 2 months ago

The right sidebar contains sections for "About", "Releases", "Packages", and "Contributors".

About

Lightweight JSON Schema validator for Cypress using AJV

Readme

MIT license

Activity

10 stars

1 watching

1 fork

Releases 6

v1.4.0 Latest 2 weeks ago

+ 5 releases

Packages

No packages published

Publish your first package

Contributors 2

<https://github.com/sclavijosuero/cypress-ajv-schema-validator>



cypress-ajv-schema-validator



Screenshot of the npm package page for `cypress-ajv-schema-validator`.

Readme (selected)

Code (Beta)

4 Dependencies

0 Dependents

6 Versions

Settings

cypress-ajv-schema-validator

A Cypress plugin for API testing to validate the API response against Plain JSON schemas, Swagger documents, or OpenAPI documents using Ajv JSON Schema validator.

Install:

```
> npm i cypress-ajv-schema-validator
```

Repository: github.com/sclavijosuero/cypress-ajv-schema-validator

Homepage: github.com/sclavijosuero/cypress-ajv-schema-validator

Weekly Downloads: 476

Version: 1.4.0 License: MIT

Unpacked Size: 2.51 MB Total Files: 58

Issues: 1 Pull Requests: 0

For a detailed guide on setting up and using this plugin to maximize its benefits, please refer to my articles:

<https://www.npmjs.com/package/cypress-ajv-schema-validator>



1. Install and Configure Plugin in 



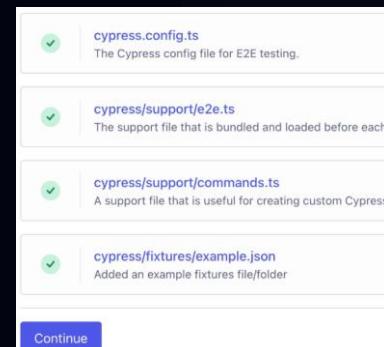
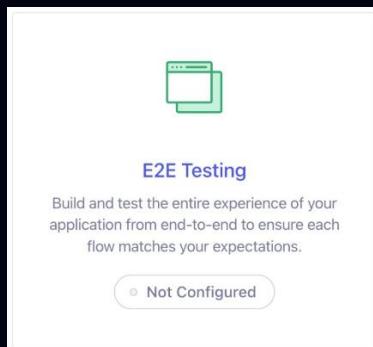
Install *cypress-ajv-schema-validator*

1. Prerequisite: Node v18

```
npm init
```

```
npm install --save-dev cypress
```

```
npx cypress open
```



3. *cypress.config.js*

```
const { defineConfig } = require("cypress");

module.exports = defineConfig({
  viewportWidth: 1280,
  viewportHeight: 800,

  e2e: {
    setupNodeEvents(on, config) {
      // implement node event listeners here
    },
    baseUrl: 'https://petstore.swagger.io/v2',
  },
});
```

4. *.gitignore*

```
node_modules/
cypress/screenshots/
cypress/videos/
cypress/downloads/
```

5. Install plugin

```
npm install -D cypress-ajv-schema-validator
```



Configure *cypress-ajv-schema-validator*

1. *Commands.js (or e2e.js)*

```
// ****
// This example commands.js shows you how to
// create various custom commands and overwrite
// existing commands.
//
// For more comprehensive examples of custom
// commands please read more here:
// https://on.cypress.io/custom-commands
// ****

import 'cypress-ajv-schema-validator';
```



2. Plugin API



cypress-ajv-schema-validator API Reference

<https://github.com/sclavijosuero/cypress-ajv-schema-validator?tab=readme-ov-file#api-reference>

The screenshot shows a browser window displaying the API Reference for the *cypress-ajv-schema-validator* plugin on GitHub. The URL in the address bar is <https://github.com/sclavijosuero/cypress-ajv-schema-validator?tab=readme-ov-file#api-reference>. The page has a dark theme. At the top, there are links to 'README' and 'MIT license'. Below that is a red horizontal bar with the text 'API Reference'. Under this bar, the section 'Custom Commands' is visible. It contains a code snippet for the `cy.validateSchema(schema, path)` command. A note states: 'It is expected to be chained to an API response (from a `cy.request()` or `cy.api()`). It validates the response body against the provided schema.' Below this, the 'Parameters' section lists two required parameters: 'schema' (object) and 'path' (object, optional). The 'schema' parameter is described as validating against plain JSON, Swagger, and OpenAPI documents. The 'path' parameter is described as applying to Swagger or OpenAPI documents, defining the path to the schema definition. It includes properties for 'endpoint' (string, optional), 'method' (string, optional, defaulting to 'GET'), and 'status' (integer, optional, defaulting to 200). The 'Returns' section indicates that it returns a `Cypress.Chainable` object. The 'Throws' section notes that an `Error` is thrown if required parameters are missing or if the schema or schema definition is not found. Below these sections, examples are provided for providing a Plain JSON schema and an OpenAPI 3.0.1 or Swagger 2.0 schema document. Both examples show how to chain the command to an API request. A note at the bottom explains that the plugin will automatically take the schema `$ref` for definitions found in the `components` section of the schema document.

`cy.validateSchema(schema, path)`

It is expected to be chained to an API response (from a `cy.request()` or `cy.api()`). It validates the response body against the provided schema.

Parameters

- `schema` (object): The schema to validate against. Supported formats are plain JSON schema, Swagger, and OpenAPI documents.
- `path` (object, optional): This second parameter only applies to Swagger or OpenAPI documents. It represents the path to the schema definition in a Swagger or OpenAPI document and is determined by three properties:
 - `endpoint` (string, optional): The endpoint path.
 - `method` (string, optional): The HTTP method. Defaults to 'GET'.
 - `status` (integer, optional): The response status code. If not provided, defaults to 200.

Returns

- `Cypress.Chainable` : The response object wrapped in a Cypress.Chainable.

Throws

- `Error` : If any of the required parameters are missing or if the schema or schema definition is not found.

Example providing a Plain JSON schema:

```
cy.request('GET', 'https://awesome.api.com/users/1')
.validateSchema(schema);
```

Example providing an OpenAPI 3.0.1 or Swagger 2.0 schema documents and path to the schema definition:

```
cy.request('GET', 'https://awesome.api.com/users/1')
.validateSchema(schema, { endpoint: '/users/{id}', method: 'GET', status: 200 });
```

Using the path defined by `{ endpoint, method, status }`, the plugin will automatically take the schema `$ref` for that definition, find it in the `components` section, and use it in the schema validation.



cypress-ajv-schema-validator API

`cy.validateSchema(schema, path)`

- **schema**: Object with the schema to validate. Can be Plain JSON schema, Swagger and OpenAPI documents.
- **path**: (optional) Object with the path to the schema definition within a Swagger or OpenAPI document.



cypress-ajv-schema-validator API cy.validateSchema(schema, path)

PATH (optional)

Only for Swagger & OpenAPI Documents

- **endpoint**: The endpoint path.
- **method**: The HTTP method. Defaults to 'GET'.
- **status**: The response status code. If not provided, defaults to 200.

```
title: "User API",
version: "1.0.0"
},
paths: {
"/users/{id)": {
"get": {
summary: "Get a user by ID",
parameters: [
{
name: "id",
in: "path",
required: true,
schema: {
type: "integer"
}
}
],
responses: {
"200": {
description: "User details",
content: {
"application/json": {
schema: {
"$ref": "#/components/schemas/User"
}
}
}
},
"401": {
description: "Unauthorized"
}
}
},
"components": {
"schemas": {
"User": {
type: "object"
}
}
}
}
```

Find definition automatically in components



3. First Test (Plain JSON)



Example: Plain JSON Schema (API to test)

API Endpoint URL

<https://api.zippopotam.us/us/33162>

```
{  
    "post code": "33162",  
    "country": "United States",  
    "country abbreviation": "US",  
    "places": [  
        {  
            "place name": "Miami",  
            "longitude": "-80.183",  
            "state": "Florida",  
            "state abbreviation": "FL",  
            "latitude": "25.9286"  
        }  
    ]  
}
```

Plain JSON Schema

```
{  
    "type": "object",  
    "properties": {  
        "post code": {  
            "type": "string"  
        },  
        "country": {  
            "type": "string"  
        },  
        "country abbreviation": {  
            "type": "string"  
        },  
        "places": {  
            "type": "array",  
            "items": {  
                "type": "object",  
                "properties": {  
                    "place name": {  
                        "type": "string"  
                    },  
                    "longitude": {  
                        "type": "string"  
                    },  
                    "state": {  
                        "type": "string"  
                    },  
                    "state abbreviation": {  
                        "type": "string"  
                    },  
                    "latitude": {  
                        "type": "string"  
                    }  
                },  
                "required": [  
                    "place name",  
                    "longitude",  
                    "state",  
                    "state abbreviation",  
                    "latitude"  
                ]  
            }  
        },  
        "required": [  
            "post code",  
            "country",  
            "country abbreviation",  
            "places"  
        ]  
    }  
}
```



Example: Plain JSON Schema (Tests)

```
/// <reference types="cypress" />

// Load Plain JSON schema
import schema from '../fixtures/schemas/plainjson-schema.json'

describe(`First JSON Schema Test Suite`, () => {

  Open Cypress | Set ".only"
  it(`Test will PASS - Plain JSON Schema Validation`, () => {
    // Expected Response body is:
    // {
    //   "post code": "33162",
    //   "country": "United States",
    //   "country abbreviation": "US",
    //   "places": [
    //     {
    //       "place name": "Miami",
    //       "longitude": "-80.183",
    //       "state": "Florida",
    //       "state abbreviation": "FL",
    //       "latitude": "25.9286"
    //     }
    //   ]
    // }

    cy.request('https://api.zippopotam.us/us/33162') // (SCHEMA VALIDATION WILL PASS)
      .validateSchema(schema)
  })

  Open Cypress | Set ".only"
  it(`Test will FAIL - Plain JSON Schema Validation`, () => {
    cy.request('https://api.zippopotam.us/us/33162').then((response) => {
      // Modify 'country abbreviation' property in the response body to be a number (expected a string)
      response.body["country abbreviation"] = 1
    }).validateSchema(schema) // (SCHEMA VALIDATION WILL FAIL)
  })
})
```



Example: Plain JSON Schema (Results – First Test PASS)

The screenshot shows the Cypress Test Results interface. On the left, the file structure is visible with a file named `test-plainjson-schema.cy.js` selected. The main pane displays the test results for the "First JSON Schema Test Suite". One test, "Test will PASS - Plain JSON Schema Validation", has passed, indicated by a green checkmark and the message "PASSED - THE RESPONSE BODY IS VALID AGAINST THE SCHEMA.". A green arrow points from the text "First Test PASS" to this message. Another test, "Test will FAIL - Plain JSON Schema Validation", is listed below it. The browser window on the right shows a blank page with the text "Default blank page" and a note about session data being cleared. A pinned status is shown at the bottom right.

First Test PASS

Specs

test-plainjson-schema cy.js

First JSON Schema Test Suite

✓ Test will PASS - Plain JSON Schema Validation

TEST BODY

1 request GET 200 https://api.zippopotam.us/us/33162

2 ✓ PASSED - THE RESPONSE BODY IS VALID AGAINST THE SCHEMA.

✗ Test will FAIL - Plain JSON Schema Validation

Chrome 131 1280x800 (74%)

cy

Default blank page

This page was cleared by navigating to about:blank.
All active session data (cookies, localStorage and sessionStorage) across all domains are cleared.

Pinned



Example: Plain JSON Schema (Results – Second Test FAIL)

The screenshot illustrates a failing JSON schema validation test in Cypress. The test file is `test-plainjson-schema.cy.js`, and the failing test is "Test will FAIL - Plain JSON Schema Validation".

Cypress Test Runner:

- Specs:** Shows the test results: 1 PASS, 1 FAIL, 0 Warnings, 0 Errors.
- TEST BODY:** Contains the failing test code:

```
1 request GET 200 https://api.zippopotam.us/us/33162
2
3   { "instancePath": "/country abbreviation", "schemaPath": "#/properties/country$20abbreviation/type", "keyword": "type", "params": { "type": "string" }, "message": "must be string" }
4
5 then function() {}
6
7 Error
```
- Console:** Displays the error message: "The response body is not valid against the schema!"

Browser Output:

- Title:** Default blank page
- Message:** This page was cleared by navigating to about:blank.
All active session data (cookies, localStorage and sessionStorage) across all domains are cleared.
- Status Bar:** Pinned

DevTools Console:

- Errors:** Shows the error details:
 - Displayname: FAILED
 - Message: **THE RESPONSE BODY IS NOT VALID AGAINST THE SCHEMA (Number of schema errors: 1)
 - Number_of_schema_errors: 1
 - Schema_errors: Array(1)
 - Mismatches_in_data: Object { country: "United States", country abbreviation: "1 must be string" }
- Annotations:** A red box highlights the error message "country abbreviation: \"1 must be string\"", which corresponds to the failing schema rule in the Cypress test code.

Annotations:

- A red arrow points from the Cypress test runner's error message to the browser's error message.
- A large red arrow points from the browser's error message to the detailed error output in the DevTools console.
- A callout bubble highlights the specific schema error: "Property 'country abbreviation' must be string".

Text Labels:

- Second Test FAIL (1 schema error):** Located above the browser output.
- Property "country abbreviation" must be string**: Located above the DevTools error details.

4. Second Test (OpenAPI 3 doc)



Example: OpenAPI 3 Schema Doc (API to test)

API Endpoint 1: GET – 200 Status (Mocked to fail)

[fixtures\mock-data\openapi3\service1-schema-fail_GET.json](#)

```
{  
    "status": 200,  
    "headers": {  
        "content-type": "application/json"  
    },  
    "body": {  
        "id": "123e4567-e89b-12d3-a456-426614174000",  
        "name": null,  
        "status": "Lost",  
        "email": "invalid-email-format",  
        "isActive": true,  
        "age": 30,  
        "tags": [  
            "tag1",  
            2,  
            "tag3"  
        ],  
        "preferences": {  
            "notifications": "yes",  
            "theme": "dark",  
            "itemsPerPage": 20  
        }  
    }  
}
```

OpenAPI Schema Doc

```
{  
    "openapi": "3.0.1",  
    "info": {  
        "title": "Example API",  
        "version": "1.0.0",  
        "description": "An example API with GET and POST methods."  
    },  
    "servers": [  
        {  
            "url": "https://api.example.com"  
        }  
    ],  
    "paths": {  
        "/service1": {  
            "get": {  
                "summary": "Get details for service 1",  
                "operationId": "getService1",  
                "responses": {  
                    "200": {  
                        "description": "Successful response",  
                        "content": {  
                            "application/json": {  
                                "schema": {  
                                    "$ref": "#/components/schemas/Service1Response"  
                                }  
                            }  
                        }  
                    },  
                    "401": {  
                        "description": "Unauthorized",  
                        "content": {  
                            "application/json": {  
                                "schema": {  
                                    "$ref": "#/components/schemas/ErrorResponse"  
                                }  
                            }  
                        }  
                    }  
                }  
            },  
            "post": {  
                "summary": "Create a new entry for service 1",  
                "operationId": "createService1",  
                "requestBody": {  
                    "content": {  
                        "application/json": {  
                            "schema": {  
                                "$ref": "#/components/schemas/Service1Request"  
                            }  
                        }  
                    }  
                }  
            }  
        }  
    }  
}
```

API Endpoint 2: POST – 400 Status (Mocked to fail)

[fixtures\mock-data\openapi3\service1-schema-fail_POST.json](#)

```
{  
    "status": 401,  
    "headers": {  
        "content-type": "application/json"  
    },  
    "body": {  
        "code": 401,  
        "message": null  
    }  
}
```



Example: OpenAPI 3 Schema Doc (*requestMock* Custom Method)

Commands.js

```
// ****
// This example commands.js shows you how to
// create various custom commands and overwrite
// existing commands.
//
// For more comprehensive examples of custom
// commands please read more here:
// https://on.cypress.io/custom-commands
// ****

import 'cypress-ajv-schema-validator';

// MOCKING JAVASCRIPT LIBRARIES:
// - Mockoon (https://github.com/mockoon/mockoon)
// - Mock Service Worker (https://github.com/mswjs/msw)
// - MirageJS (https://github.com/miragejs/miragejs)
// - Nock (https://github.com/nock/nock)

Cypress.Commands.add('requestMock', (method = 'GET', url) => {
  const fixturePath = `${url}_${method.toUpperCase()}`

  cy.fixture(fixturePath).then((responseData) => {
    cy.log(`👀👀👀 *** MOCKING REQUEST WITH PROVIDED FIXTURE *** `, responseData)
    return cy.wrap(responseData)
  })
})
```



Example: OpenAPI 3 Schema Doc (Tests)

```
/// <reference types="cypress" />

import schema from '../fixtures/schemas/openapi3-schema.json'

describe(`OpenAPI3 Document Test Suite`, () => {

  Open Cypress | Set ".only"
  it(`Test will FAIL - OpenAPI3 Schema Validation - Use Case: "/service1" - "get" - 200`, () => {
    cy.requestMock('GET', '/mock-data/openapi3/service1-schema-fail') // Mock the request (WILL FAIL)
      .validateSchema(schema, { endpoint: "/service1", method: "get", status: 200 })
  })

  Open Cypress | Set ".only"
  it(`Test will FAIL - OpenAPI3 Schema Validation - Use Case: "/service1" - "post" - 401`, () => {
    cy.requestMock('POST', '/mock-data/openapi3/service1-schema-fail') // Mock the request (WILL FAIL)
      .validateSchema(schema, { endpoint: "/service1", method: "post", status: 401 })
  })
})
```



Example: OpenAPI 3 Schema Doc (Results – First Test FAIL)

/service1 – GET – 200 Status (Mocked to fail)

The screenshot shows the Cypress DevTools interface with the following details:

- Specs:** A sidebar on the left lists two spec files: `cypress/e2e/test-openapi-schema.cy.js` and `cypress/e2e/test-plainjson-schema.cy.js`.
- Test Result:** The main pane displays a test named "test-openapi-schema.cy.js" which failed with a status of 136ms. The error message is: "Test will FAIL - OpenAPI3 Schema Validation - Use Case: "/service1" - 'get' - 200".
- Test Body:** The test body contains code for mocking a request and then logs an error message: "FAILED - THE RESPONSE BODY IS NOT VALID AGAINST THE SCHEMA (Number of schema errors: 5)".
- Console:** The bottom pane shows the Cypress console output. It includes:
 - 7 messages, 7 user messages, 0 errors, 0 warnings, 7 info, and 0 verbose.
 - A command log entry: "colorLog-e34040".
 - An error message: "FAILED - THE RESPONSE BODY IS NOT VALID AGAINST THE SCHEMA (Number of schema errors: 5)".
 - Details about the schema errors, including the number of errors (5) and a list of specific validation failures.
 - Stack traces for each error, pointing to file "index-BdCmY7-e.js" at line 96637.
- Annotations:** Red boxes highlight several error messages in both the test body and the console output, specifically:
 - "Missing property 'createdAt'"
 - "'invalid-email-format' must match format \\"email\\""
 - "null must be string"
 - "'yes' must be boolean"
 - "'dark' must be string"
 - "'2' must be string"
- Text Overlay:** A large yellow text overlay "First Test FAIL (5 schema errors)" is positioned in the center-right area of the screenshot.

Example: OpenAPI 3 Schema Doc (Results – Second Test FAIL)

/service1 – POST – 401 Status (Mocked to fail)

The screenshot shows the Cypress Test Runner interface with a failing test case. The test file is `test-openapi-schema.cy.js`, and the failing test is titled "Test will FAIL - OpenAPI3 Schema Validation - Use Case: /service1 - 'post' - 401". The test code includes logging and a `wrap` operation that fails due to schema validation errors. A red arrow points from this error message in the test runner to the corresponding error message in the Cypress DevTools console.

Second Test FAIL (1 schema error)

Console output:

```
Console was cleared
Command: colorLog-e34040
Displayname: ✘ FAILED -
Message: **THE RESPONSE BODY IS NOT VALID AGAINST THE SCHEMA (Number of schema errors: 1).**
Number_of_schema_errors: 1
Schema_errors: ▶ [{...}]
Mismatches_in_data: ▶ {code: 401, message: '● null must be string'} ⓘ
code: 401
message: "● null must be string"
[[Prototype]]: Object
```

5. Integration with cy.api()

@bahmutov/cy-api and *cypress-plugin-api*



Configure plugin to be used with *cy.api()*

1. Install @bahmutov/cy-api

```
npm install --save-dev @bahmutov/cy-api
```

2. Install cypress-plugin-api

```
npm install cypress-plugin-api
```

3. Set environment variable

enableMismatchesOnUI to *true*

- Either in *cypress.env.json*

```
{  
  "enableMismatchesOnUI": true  
}
```

- Or in *cypress.config.js*

```
const { defineConfig } = require("cypress");  
  
module.exports = defineConfig({  
  
  env: {  
    enableMismatchesOnUI: true,  
  }  
});
```

- Or in CLI

```
npx cypress open --env enableMismatchesOnUI=true
```



Example: Swagger2 Schema Doc with *cy.api()* plugins (Test)

```
/// <reference types="cypress" />

import '@bahmutov/cy-api' // Or: import 'cypress-plugin-api'

import petstoreSchema from '../fixtures/schemas/petstore-swagger2-schema-errors.json'

describe(`Petstore Swagger2 Test Suite - '@bahmutov/cy-api' cy.api()`, () => {

  Open Cypress | Set ".only"
  it(`Test will FAIL - Swagger2 Schema Validation - Use Case: "/pet/findByStatusstatus=pending" - "get" - 200`, () => {
    const findByStatusReq = {
      url: 'https://petstore.swagger.io/v2/pet/findByStatus?status=pending',
      headers: { 'Content-Type': 'application/json' }
    }

    cy.api(findByStatusReq)
      .validateSchema(petstoreSchema, { endpoint: '/pet/findByStatus', method: 'get', status: 200 })
  })
})
```



Example: Swagger2 Schema Doc (Results with Gleb's cy.apy())

The image shows two screenshots illustrating a failing Swagger2 schema validation test.

Left Screenshot (Cypress Test Runner):

- Shows a file tree under "cypress/e2e":
 - test-openapi-schema.cy.js
 - test-petstore-with-cyApi-filip.cy.js
 - test-petstore-with-cyApi-gleb.cy.js** (highlighted)
 - test-plainjson-schema.cy.js
- The "test-petstore-with-cyApi-gleb.cy.js" file contains a failing test:

```
Test will FAIL - Swagger2 Schema Validation - Use Case: "/pet/findByStatusstatus=pending" - "get" - 200
```

TEST BODY

```
1 api https://petstore.swagger.io/v2/pet/findByStatus?status=pending
2 response https://petstore.swagger.io/v2/pet/findByStatus?status=pending
3 ✘ FAILED - THE RESPONSE BODY IS NOT VALID AGAINST THE SCHEMA (Number of schema errors: 48).
4   { "instancePath": "/0",
      "schemaPath": "#/required", "keyword": "required", "params": {
        "missingProperty": "age" }, "message": "must have required property 'age'" }
5   { "instancePath": "/0/id",
      "schemaPath": "#/properties/id/type",
      "keyword": "type", "params": { "type": "string" }, "message": "must be string" }
6   { "instancePath": "/0/category",
      "schemaPath": "#/definitions/Category/required",
      "keyword": "required", "params": {
        "missingProperty": "color" },
      "message": "must have required property 'color'" }
7   { "instancePath": "/0/category/name",
      "schemaPath": "#/definitions/Category/properties/name/type",
      "keyword": "type", "params": { "type": "integer" },
      "message": "must be integer" }
8   { "instancePath": "/0/name",
      "schemaPath": "#/properties/name/type",
      "keyword": "type", "params": { "type": "integer" },
      "message": "must be integer" }
```



Example: Swagger2 Schema Doc (Results with Filip's cy.apy())

The image shows two screenshots illustrating a schema validation process. On the left, the Cypress Test Runner interface displays a test file named `test-petstore-with-cyApi-filip.cy.js`. The test suite is titled "Petstore Swagger2 Test Suite - 'cypress-plugin-api' cy.api". A red error message indicates that the test will fail due to Swagger2 Schema Validation, specifically for the endpoint `/pet/findByStatus?status=pending`. The error count is 48, and it details various schema violations across properties like `age`, `color`, `name`, and `type`.

On the right, a browser window shows the results of a `GET https://petstore.swagger.io/v2/pet/findByStatus?status=pending` request. The response status is `200 (OK)`. The response body is a JSON array of pet objects. Two specific errors are highlighted with callouts:

- A yellow callout labeled "Schema error" points to the first pet object in the array. It highlights the `age` field with a red box and the error message "Missing property 'age'".
- A red callout labeled "Missing Property" points to the `type` field of the second pet object, which is annotated with a red box and the error message "Missing property 'type'".

```
GET https://petstore.swagger.io/v2/pet/findByStatus?status=pending Status: 200 (OK) Duration: 83 ms Size: 0 B
```

```
1 [ 2   { 3     "id": 82901748, 4       "category": { 5         "name": "doggie" 6       }, 7       "photoUrls": [ 8         "elit proident non", 9           "Exepteur ex in incididunt enim" 10      ], 11      "tags": [ 12        { 13          "name": "sit anim" 14        } 15      ], 16      "type": -75108271, 17      "name": "ipsum" 18    }, 19    { 20      "id": -85364496, 21      "name": "ipsum" 22    }, 23  ], 24  "status": "pending"
```



6. Disable JSON Schema Validation in your tests



Configure Disable Schema Validation in Tests

1. Set environment variable *disableSchemaValidation* to *true*

- Either in *cypress.env.json*

```
{  
  "disableSchemaValidation": true  
}
```

- Or in *cypress.config.js*

```
const { defineConfig } = require("cypress");  
  
module.exports = defineConfig({  
  
  env: {  
    disableSchemaValidation: true  
  }  
});
```

- Or in CLI

```
npx cypress open --env disableSchemaValidation=true
```



Example: Disable Schema Validation (Results)

The screenshot displays the Cypress Test Runner interface. On the left, the file tree shows several spec files under the 'cypress/e2e' directory, including 'test-openapi-schema.cy.js', 'test-petstore-with-cyApi-filip.cy.js', 'test-petstore-with-cyApi-gleb.cy.js', and 'test-plainjson-schema.cy.js'. The 'test-plainjson-schema.cy.js' file is selected, showing its contents in the central editor area. The test suite contains two tests: one that passes ('Test will PASS - Plain JSON Schema Validation') and one that fails ('Test will FAIL - Plain JSON Schema Validation'). Both failing tests include a note about schema validation being disabled due to the environment variable 'disableSchemaValidation' being set to true. To the right of the editor is a browser preview window showing a blank white page. The browser status bar indicates 'Chrome 131' and a resolution of '1280x800 (49%)'. A small circular icon with the letters 'cy' is visible in the bottom right corner of the browser window.



Sample Project Repo

<https://github.com/sclavijosuero/cypress-ajv-schema-validator-sample-project>

The screenshot shows the GitHub repository page for 'cypress-ajv-schema-validator-sample-project'. The repository is public and has 1 branch and 0 tags. The main commit history shows several initial commits by 'sclavijosuero' related to setting up Cypress and Ajv schema validation. The repository includes a README file and an MIT license. It has 0 stars, 1 watching, and 0 forks. The Languages section shows 100% JavaScript.

Code | **Issues** | **Pull requests** | **Actions** | **Projects** | **Wiki** | **Security** | **Insights** | **Settings**

cypress-ajv-schema-validator-sample-project Public

main · 1 Branch · 0 Tags

Go to file Add file Code

About

CYPRESS-AJV-SCHEMA-VALIDATOR
Plugin Video Tutorial Material

Readme
MIT license
Activity
0 stars
1 watching
0 forks

Releases

No releases published Create a new release

Packages

No packages published Publish your first package

Languages

JavaScript 100.0%

cypress-ajv-schema-validator-sample-project

Welcome to the Cypress-ajv-schema-validator Sample Project repository. This project serves as a demonstration of the features and best practices using the cypress-ajv-schema-validator Plugin, as showcased in the "CYPRESS-AJV-

Related Blogs & Videos:



[CYPRESS-AJV-SCHEMA-VALIDATOR Plugin: The Brave Vigilante for Your API Contracts](#) (*Sebastian Clavijo*)



[CYPRESS-AJV-SCHEMA-VALIDATOR v1.2.0: Boost Debugging Skills from Vigilante to Superhero with Advanced Schema Error Insights!](#) (*Sebastian Clavijo*)



[Elevate Your Cypress Testing: Top 10 Essential Plugins](#) (*Cypress.io - Farah Shalwani*)



[JSON Schema Tooling](#) (*json-schema.org*)



[Schema validation using cypress-ajv-schema-validator vs Optic & Demo comparing API e2e vs Schema testing](#) (*Murat Ozcan*)



[Cypress API Testing: AJV SCHEMA VALIDATOR](#) (*JoanMedia*)

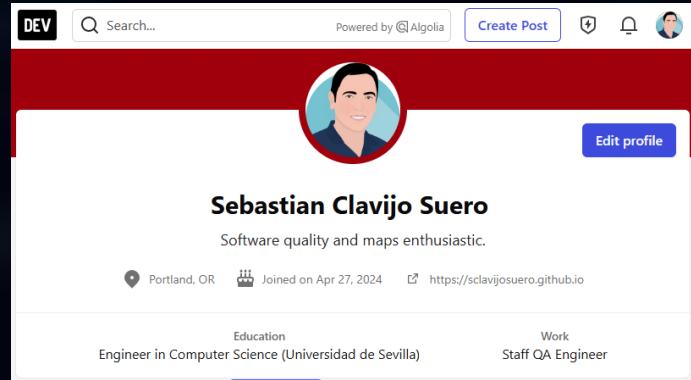


Where to find me?



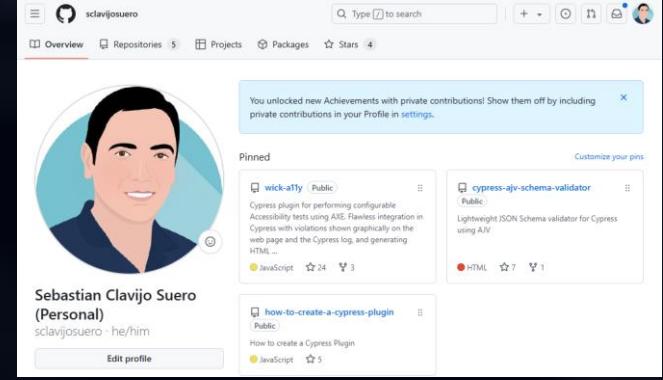
LinkedIn profile of Sebastian Clavijo Suero. It shows a profile picture, a banner image of him in a red "WICK-A11Y" t-shirt, and his bio: "Staff Quality Assurance Engineer @ KUBRA | Engineer in Computer Science | Cypress Independent Contributor | Blogger | Accessibility | Maps Enthusiastic | Plugin-ator". He is located in Portland, OR, joined on April 27, 2024, and has a GitHub link: <https://sclavijosuero.github.io>. He has 500+ connections.

www.linkedin.com/in/sebastianclavijosuero/



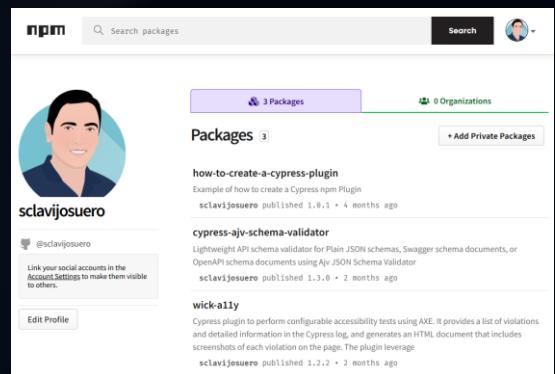
dev.to profile of Sebastian Clavijo Suero. It shows a profile picture, a bio: "Software quality and maps enthusiastic.", location: "Portland, OR", joined on April 27, 2024, and a GitHub link: <https://sclavijosuero.github.io>. It lists "Education" as "Engineer in Computer Science (Universidad de Sevilla)" and "Work" as "Staff QA Engineer".

dev.to/sebastianclavijo



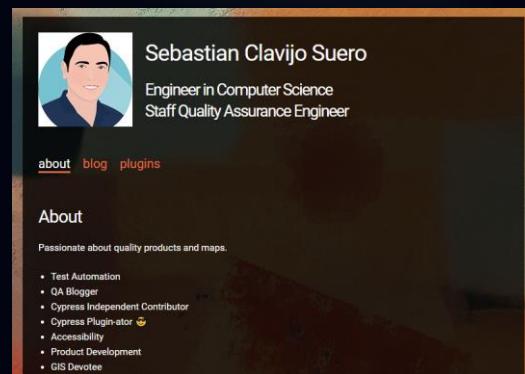
Github profile of Sebastian Clavijo Suero. It shows a profile picture, a bio: "(Personal)", and a GitHub link: <https://github.com/sclavijosuero>. It lists several repositories: "wick-a11y" (Public), "cypress-ajv-schema-validator" (Public), and "how-to-create-a-cypress-plugin" (Public). It also shows achievements and pinned items.

github.com/sclavijosuero



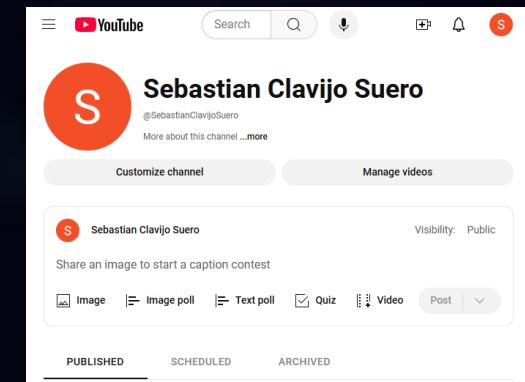
npm profile of Sebastian Clavijo Suero. It shows a profile picture and three packages: "how-to-create-a-cypress-plugin", "cypress-ajv-schema-validator", and "wick-a11y". Each package has a brief description and was published by him.

www.npmjs.com/~sclavijosuero



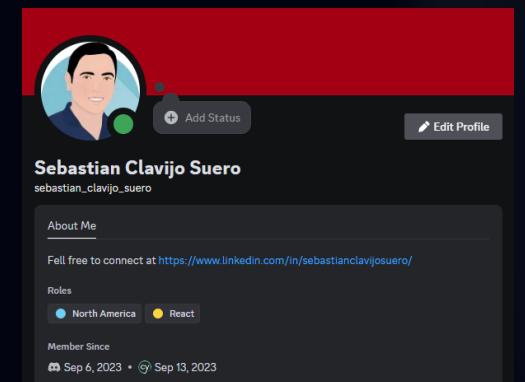
GitHub repository page for "sclavijosuero" showing the "about" page. It features a profile picture, the name "Sebastian Clavijo Suero", and titles: "Engineer in Computer Science" and "Staff Quality Assurance Engineer". It includes sections for "About" and "Plugins".

sclavijosuero.github.io/



YouTube channel page for "Sebastian Clavijo Suero". It shows a profile picture, the channel name, and a bio: "Passionate about quality products and maps.". It includes sections for "About" and "Manage videos".

www.youtube.com/@SebastianClavijoSuero



Discord invite link for "Sebastian Clavijo Suero". It shows a profile picture, the name, and a link: "Free to connect at <https://www.linkedin.com/in/sebastianclavijosuero/>". It also lists roles: "North America" and "React".

discord.com/invite/cypress