

# 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](https://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](https://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:

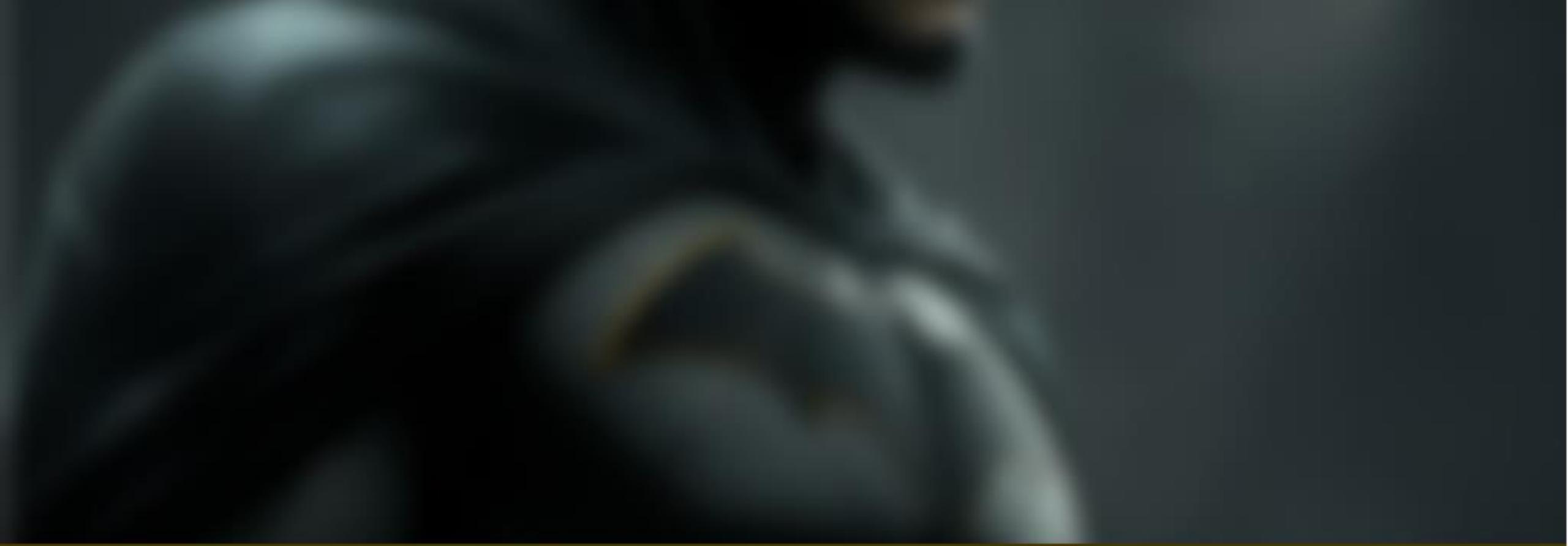
- Endpoint
- Method
- Status

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                              \"#philosophy\",  
                              \"#science\"\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



Screenshot of the GitHub repository page for `cypress-ajv-schema-validator`.

The repository is public and has 59 commits across 3 branches and 6 tags. The latest commit was made 2 weeks ago by `sclavijosuero`, updating the `readme.md` file.

The repository details include:

- About:** Lightweight JSON Schema validator for Cypress using AJV
- Readme:** Available
- MIT license:** Available
- Activity:** 10 stars, 1 watching, 1 fork
- Releases:** 6, with the latest being v1.4.0 (2 weeks ago)
- Packages:** No packages published, with a link to "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`.

The page shows the following details:

- Version:** 1.4.0 • Public • Published 12 days ago
- Dependencies:** 4 (Code: Beta)
- Dependents:** 0
- Versions:** 6
- Settings:** Settings

**Description:** 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](https://github.com/sclavijosuero/cypress-ajv-schema-validator)

**Homepage:** [github.com/sclavijosuero/cypress-ajv-schema-validator](https://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

**Code Examples:**

**Notes:** For a detailed guide on setting up and using this plugin to maximize its benefits, please refer to my articles: [link](#)

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



**1. Install and Configure Plugin in**



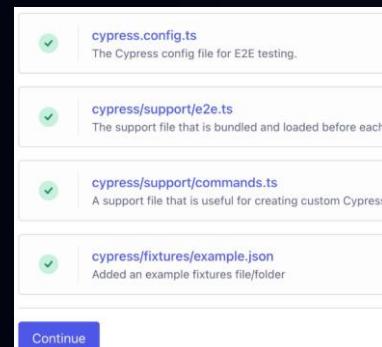
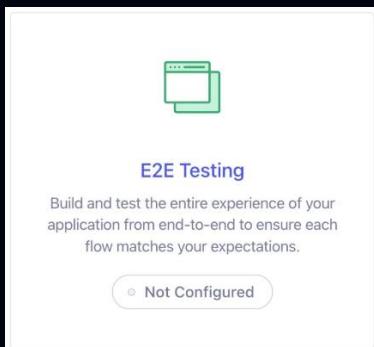
# Install *cypress-ajv-schema-validator*

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` repository 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.

**API Reference**

### Custom Commands

`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.

Specs

Search specs

cypress\e2e

`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%)

First Test PASS

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 suite contains two tests: one that passes and one that fails.

**Test Suite Results:**

- Test will PASS - Plain JSON Schema Validation (Green checkmark)
- Test will FAIL - Plain JSON Schema Validation (Red X)

**Test Body (Failed Test):**

```
1 request GET 200 https://api.zippopotam.us/us/33162
2   ✘ FAILED - THE RESPONSE BODY IS NOT VALID AGAINST THE SCHEMA (Number of schema errors: 1).
3     ● { "instancePath": "/country abbreviation", "schemaPath": "#/properties/country$20abbreviation/type", "keyword": "type", "params": { "type": "string" }, "message": "must be string" }
4   then function(){}
5   ! Error
The response body is not valid against the schema!
```

**Browser Output:**

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.

**Console Output (DevTools):**

Second Test FAIL (1 schema error):

Property "country abbreviation" must be string

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" }  
places: Array(1)  
0:  
latitude: "25.9286"  
longitude: "-80.183"  
place name: "Miami"  
state: "Florida"  
state abbreviation: "FL"  
[[Prototype]]: Object  
length: 1  
[[Prototype]]: Array(0)  
post code: "33162"  
[[Prototype]]: Object

## 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  
        }  
    }  
}
```

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  
    }  
}
```

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"  
                            }  
                        }  
                    }  
                }  
            }  
        }  
    }  
}
```



# 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 tests: "test-openapi-schema.cy.js" and "test-plainjson-schema.cy.js".
- Test Result:** The "test-openapi-schema.cy.js" test is marked as failed (red icon) and has a duration of 136ms.
- Test Body:** The code block shows a log statement followed by an assertion that fails:

```
log '*** MOCKING REQUEST WITH PROVIDED FIXTURE ***', Object(3)
wrap Object(3)

✖ FAILED - THE RESPONSE BODY IS NOT VALID AGAINST THE SCHEMA (Number of schema errors: 5).
```
- Console:** The bottom section shows the test results in the console tab. It includes:
  - 7 messages, 7 user messages, 0 errors, 0 warnings, 7 info, 0 verbose.
  - Command: colorLog-e34040
  - Displayname: FAILED
  - Message: \*\*THE RESPONSE BODY IS NOT VALID AGAINST THE SCHEMA (Number of schema errors: 5).
  - Number\_of\_schema\_errors: 5
  - Schema\_errors: (5) [{...}, {...}, {...}, {...}, {...}]
  - Mismatches\_in\_data: {
    - id: '123e4567-e89b-12d3-a456-426614174000', name: 'null must be string', status: 'Lost', email: 'invalid-email-format' must match format "\email\"
    - age: 30
    - createdAt: 'Missing property 'createdAt''
    - email: 'invalid-email-format' must match format "\email\"'
    - id: '123e4567-e89b-12d3-a456-426614174000'
    - isActive: true
    - name: 'null must be string'
    - preferences: {notifications: 'yes' must be boolean, theme: 'dark', itemsPerPage: 20}
    - status: 'Lost'

A large red arrow points from the error message in the test body to the corresponding error message in the console output. Another red arrow points from the "Schema\_errors" section in the console to the specific validation errors listed under "Mismatches\_in\_data".

**First Test FAIL (5 schema errors)**

# 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 the following details:

- Test Overview:** Specs → test-openapi-schema.cy.js (136ms).  
Case: "/service1" - "post" - 401
- Test Body:** A red box highlights the failing code:

```
1 log *** MOCKING REQUEST WITH PROVIDED FIXTURE *** , Object(3)
2 wrap Object(3)
3
4   ✘ FAILED - THE RESPONSE BODY IS NOT VALID AGAINST THE SCHEMA (Number of schema errors: 1).
5     {
6       "instancePath": "/message",
7       "schemaPath": "#/components/schemas/ErrorResponse/properties/message/type",
8       "keyword": "type",
9       "params": { "type": "string" },
10      "message": "must be string"
11    }
12  then function() {}
13  !
14  Error
15
16 The response body is not valid against the schema!
```
- Browser Preview:** Default blank page. Shows a "cy" logo and the message: "This page was cleared by navigating to about:blank. All active session data (cookies, localStorage and sessionStorage) across all domains are cleared."
- Console:** Shows the error message and stack trace:

```
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: ▶ [{}]
Matches_in_data: ▶ {code: 401, message: 'null must be string'} ▶
code: 401
message: "null must be string"
[[Prototype]]: Object
```

Stack trace (index-BdCmY7-e.js:96637):  
index-BdCmY7-e.js:96637  
index-BdCmY7-e.js:96637  
index-BdCmY7-e.js:96637  
index-BdCmY7-e.js:96637  
index-BdCmY7-e.js:96637  
index-BdCmY7-e.js:96637

A large red arrow points from the failing test code in the test runner to the corresponding error message in the console.

**Second Test FAIL (1 schema error)**

# 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 screenshot illustrates a Cypress test runner interface on the left and a browser window on the right. The test runner shows a file tree with several test files, including one named `test-petstore-with-cyApi-gleb.cy.js`. This specific test has failed due to Swagger2 Schema Validation errors. The browser window displays the validation results for the `/pet/findByStatus?status=pending` endpoint. The validation output is as follows:

```
Request:  
{  
  "url": "https://petstore.swagger.io/v2/pet/findByStatus?status=pending",  
  "headers": {  
    "Content-Type": "application/json"  
  }  
}  
  
Response: 200 87ms  
  
[  
  {  
    "id": "82901748 must be string",  
    "category": {  
      "id": 7962040,  
      "name": "culpa fugiat consequat' must be integer",  
      "color": "Missing property 'color'"  
    },  
    "name": "'doggie' must be integer",  
    "photouris": [  
      "elit proident non",  
      "Excepteur ex in incididunt enim"  
    ],  
    "tags": [  
      {  
        "id": -75108271,  
        "name": "sit anim",  
        "type": "Missing property 'type'"  
      },  
      {  
        "id": -85364496,  
        "name": "ipsum",  
        "type": "Missing property 'type'"  
      }  
    ],  
    "status": "'pending' must be equal to one of the allowed values",  
    "age": "Missing property 'age'"  
  },  
  {  
    "id": "9223372036854775000 must be string",  
    "category": {  
      "id": 7962040,  
      "name": "culpa fugiat consequat' must be integer",  
      "color": "Missing property 'color'"  
    },  
    "name": "'doggie' must be integer",  
    "photouris": [  
      "elit proident non",  
      "Excepteur ex in incididunt enim"  
    ],  
    "tags": [  
      {  
        "id": -75108271,  
        "name": "sit anim",  
        "type": "Missing property 'type'"  
      },  
      {  
        "id": -85364496,  
        "name": "ipsum",  
        "type": "Missing property 'type'"  
      }  
    ],  
    "status": "'pending' must be equal to one of the allowed values",  
    "age": "Missing property 'age'"  
  }  
]
```

Annotations highlight two specific errors:

- A yellow box labeled "Schema error" points to the first validation error: `"id": "82901748 must be string"`.
- A red box labeled "Missing Property" points to another validation error: `"type": "Missing property 'type'"`.



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

The image shows two screenshots illustrating a schema validation process for a Swagger2 API.

**Cypress Test Runner Screenshot:** 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 lists numerous validation errors across various properties like `age`, `color`, `name`, and `type`.

**Browser Screenshot:** On the right, a browser window shows the results of a `GET https://petstore.swagger.io/v2/pet/findByStatus?status=pending` request. The 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 border and the error message "Missing property 'age'".
- A red callout labeled "Missing Property" points to the `type` field of the second pet object, also with a red border and the error message "Missing property 'type'".

The JSON response body is partially visible:

```
1: [
2:   {
3:     "id": 82901748,
4:     "category": {
5:       "color": "black",
6:       "name": "culpa fugiat consequat"
7:     },
8:     "name": "doggie",
9:     "photoUrls": [
10:       "elit proident non",
11:       "Exepteur ex in incididunt enim"
12:     ],
13:     "tags": [
14:       {
15:         "type": "Missing property 'type'"
16:       }
17:     ],
18:     "status": "pending"
19:   },
20:   {
21:     "id": -75108271,
22:     "name": "sit anim"
23:   },
24:   {
25:     "id": -85364496,
26:     "name": "ipsum"
27:   }
28]
```



# 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 'disableSchemaValidation' environment variable. To the right, a browser preview window shows a blank white page with the Cypress logo and a message indicating it's a default blank page.

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 API SCHEMA VALIDATION DISABLED ▲ - The Cypress environment variable "disableSchemaValidation" has been set to true.
```

Test will FAIL - Plain JSON Schema Validation

TEST BODY

```
1 request GET 200 https://api.zippopotam.us/us/33162
2 API SCHEMA VALIDATION DISABLED ▲ - The Cypress environment variable "disableSchemaValidation" has been set to true.
```

Chrome 131 1280x800 (49%)

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.



# 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*)

