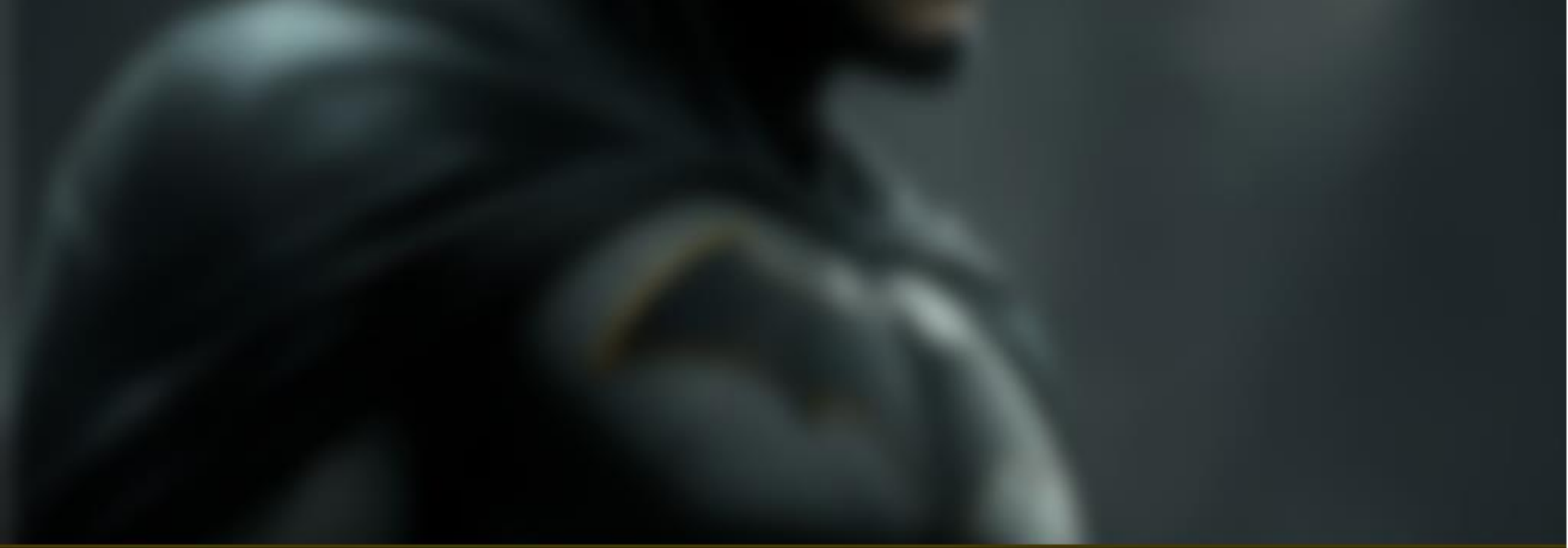
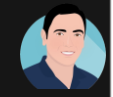
CYPRESS-AJV-SCHEMA-VALIDATOR PLUGIN VIDEO TUTORIAL



JSON Schemas



JSON Schema

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

(*) Source: json-schema.org

```json
{
    "title": "New Blog Post",
    "content": "This is the content of the blog post...",
    "publishedDate": "2023-08-25T15:00:00Z",
    "author": {
      "username": "authoruser",
      "email": "author@example.com"
    },
    "tags": ["Technology", "Programming"]
  }
```

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.

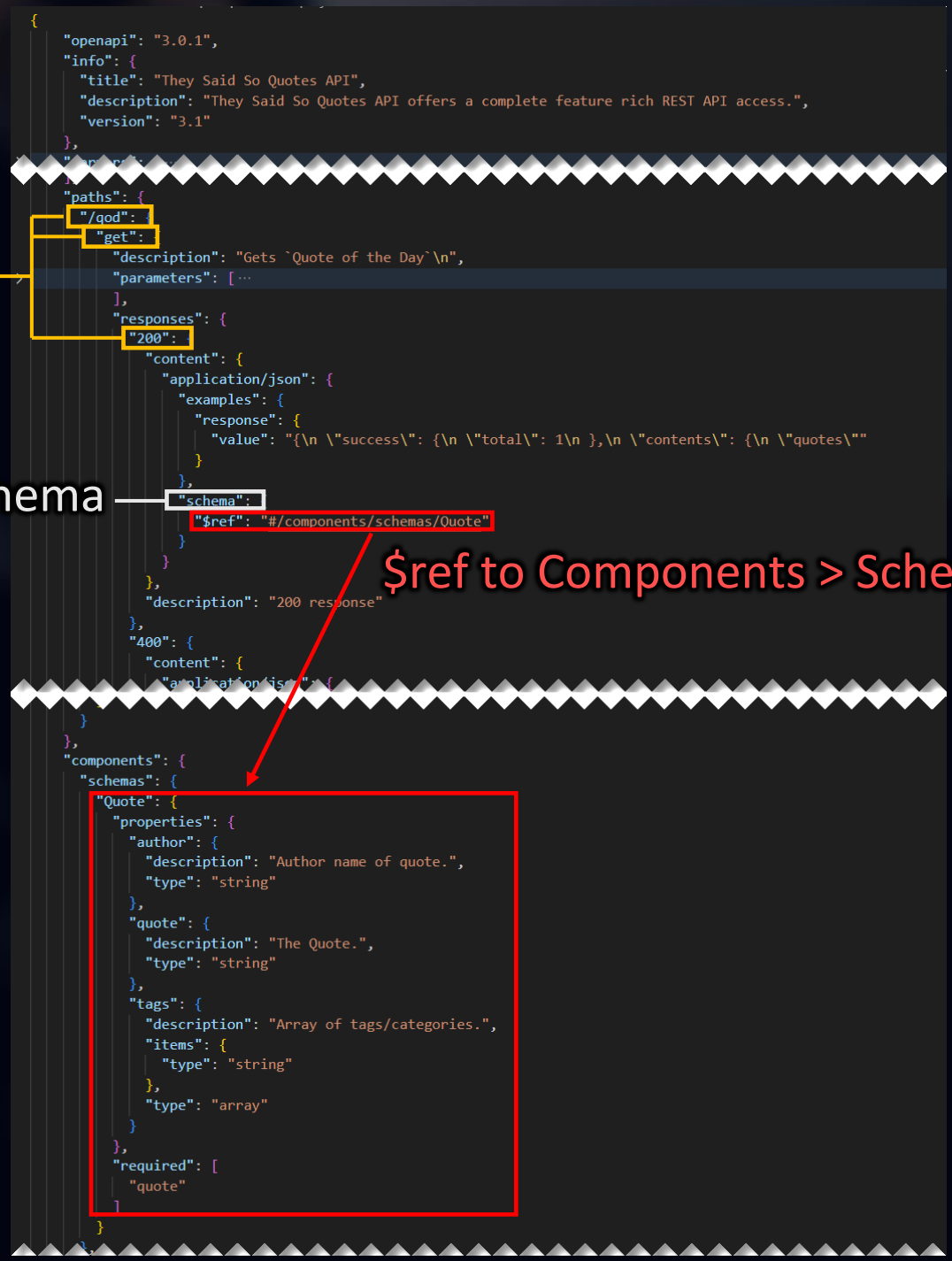
OpenAPI Sample

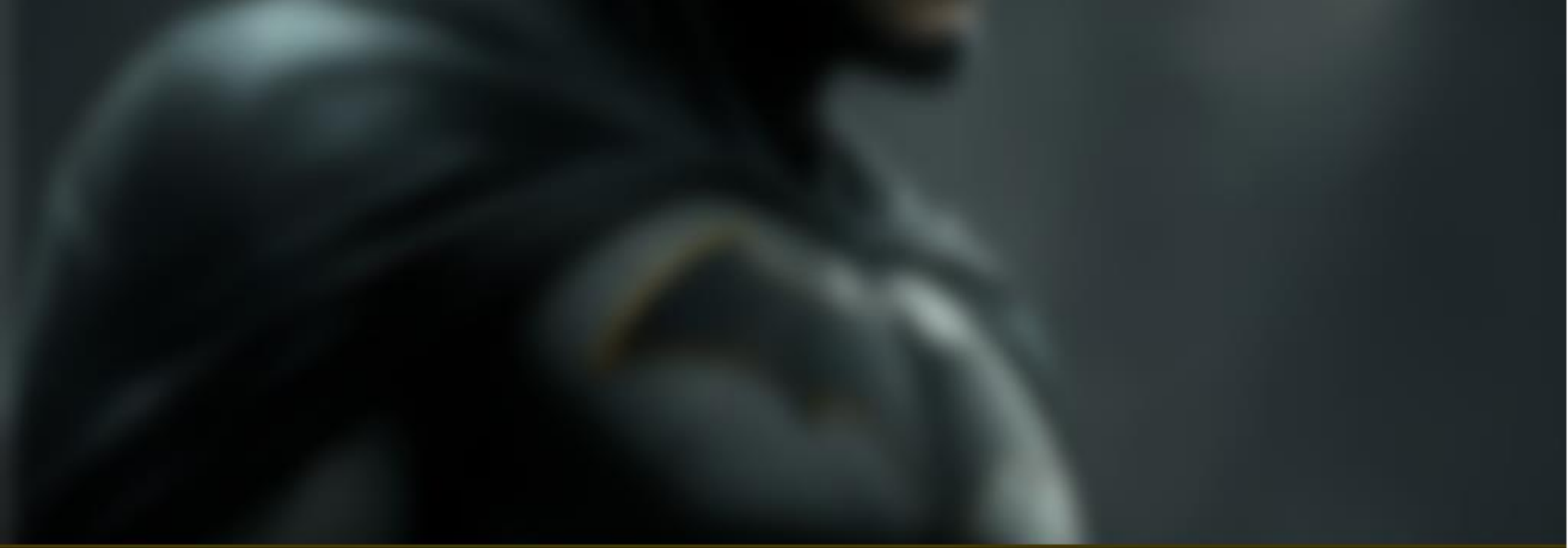
Path:

- Endpoint
- Method
- Status

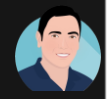
Schema

$ref to Components > Schema





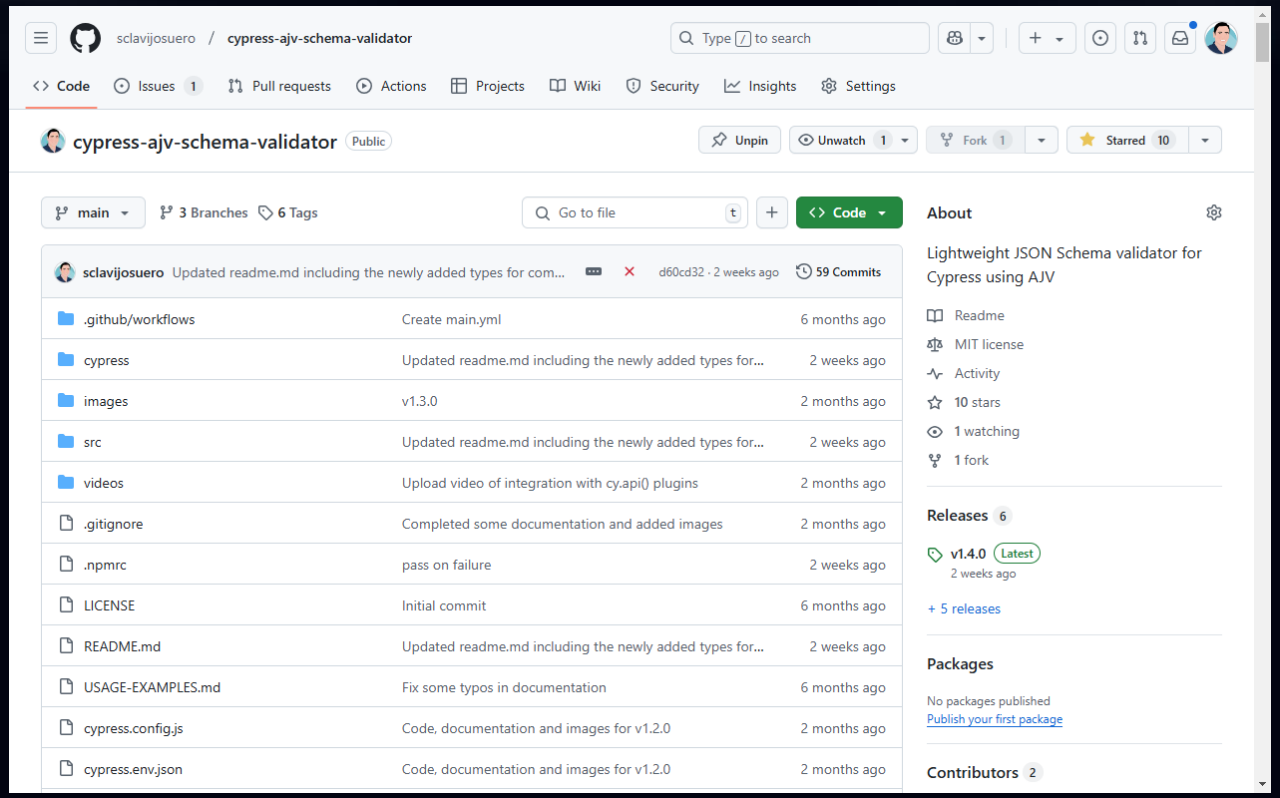
What is cypress-ajv-schema-validator?



An Open-source JSON Schema Validator Plugin for

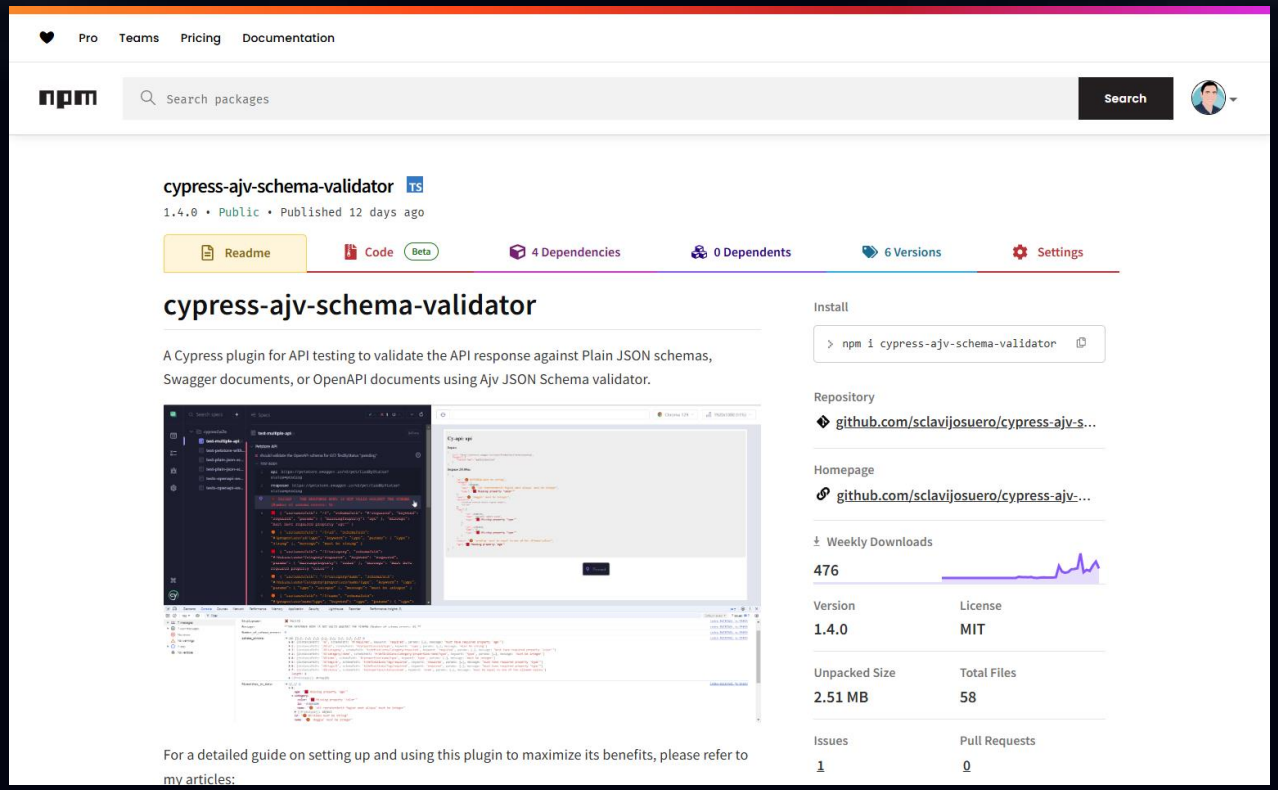
- 👀 The ultimate “vigilante” for your API Schemas
- ⚖️ Leverages the capabilities of the **AJV** plugin
- {x} Supports **JSON Schema**, and **OpenAPI 3/Swagger 2 documents**
- 📖 Results provided in a comprehensive and user-friendly way
- 🤝 Seamless integration with **@bahmutov/cy-api** and **cypress-plugin-api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

cypress-ajv-schema-validator



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

cypress-ajv-schema-validator



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

1. Install and Configure Plugin in Cypress

Install *cypress-ajv-schema-validator*

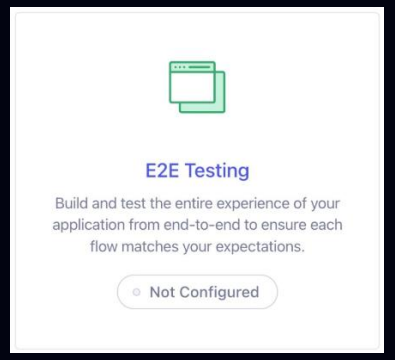
1. Prerequisite: Node v18

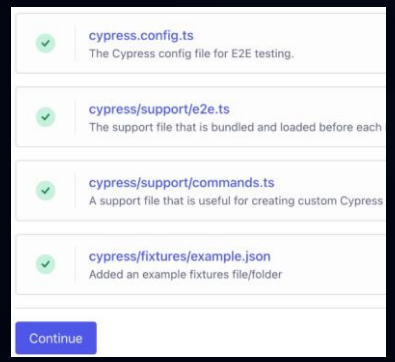
2. Create Cypress Project

```
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

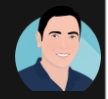
```
npm install -D cypress-ajv-schema-validator
```

Configure *cypress-ajv-schema-validator*

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

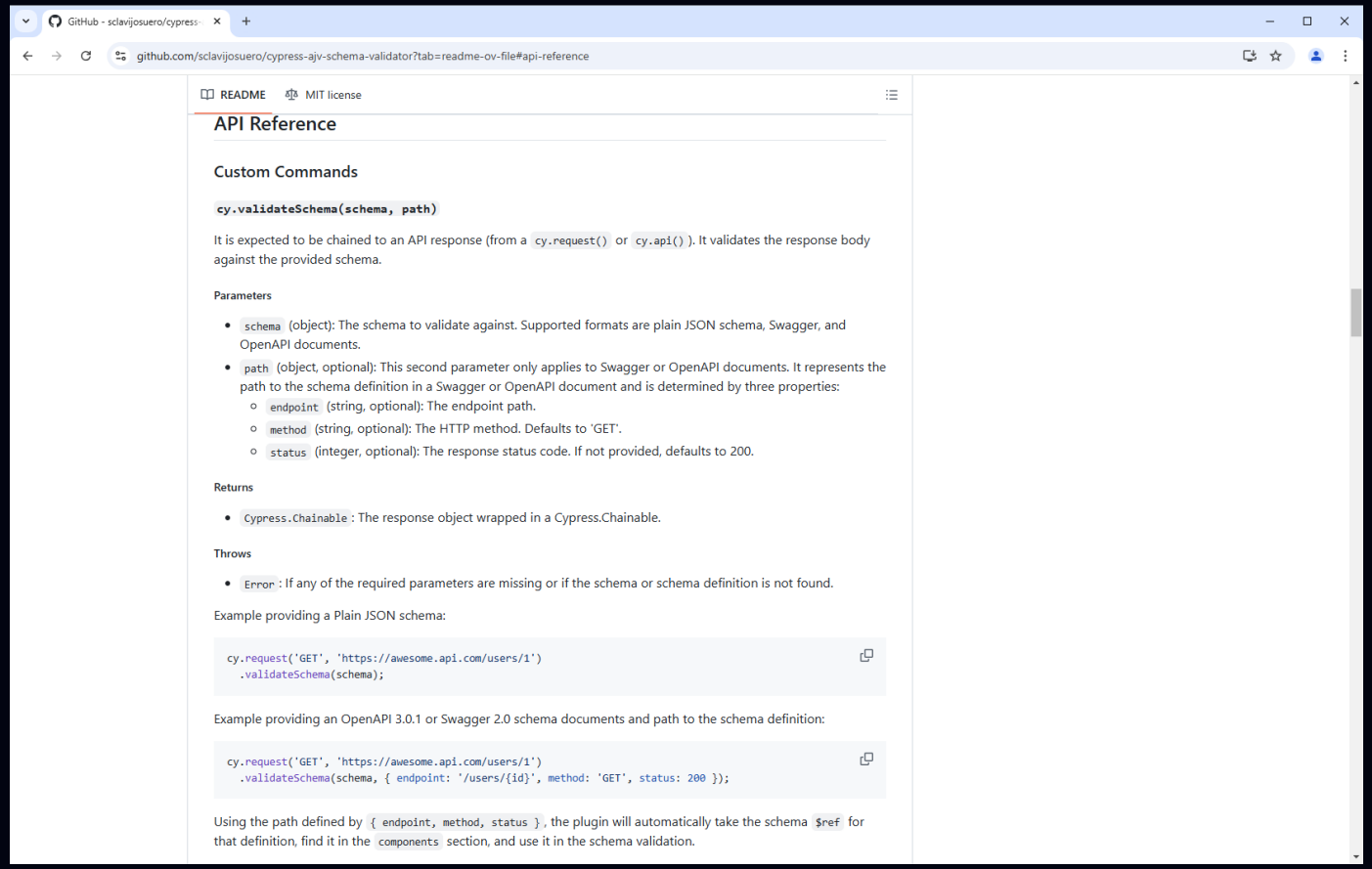
```
// ***********************************************
// This example commands.js shows you how to
// create various custom commands and overwrite
// existing commands.
//
// For more comprehensive examples of custom
// commands please read more here:
// https://on.cypress.io/custom-commands
// ***********************************************

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



cypress-ajv-schema-validator API

cy.validateSchema(schema, path)

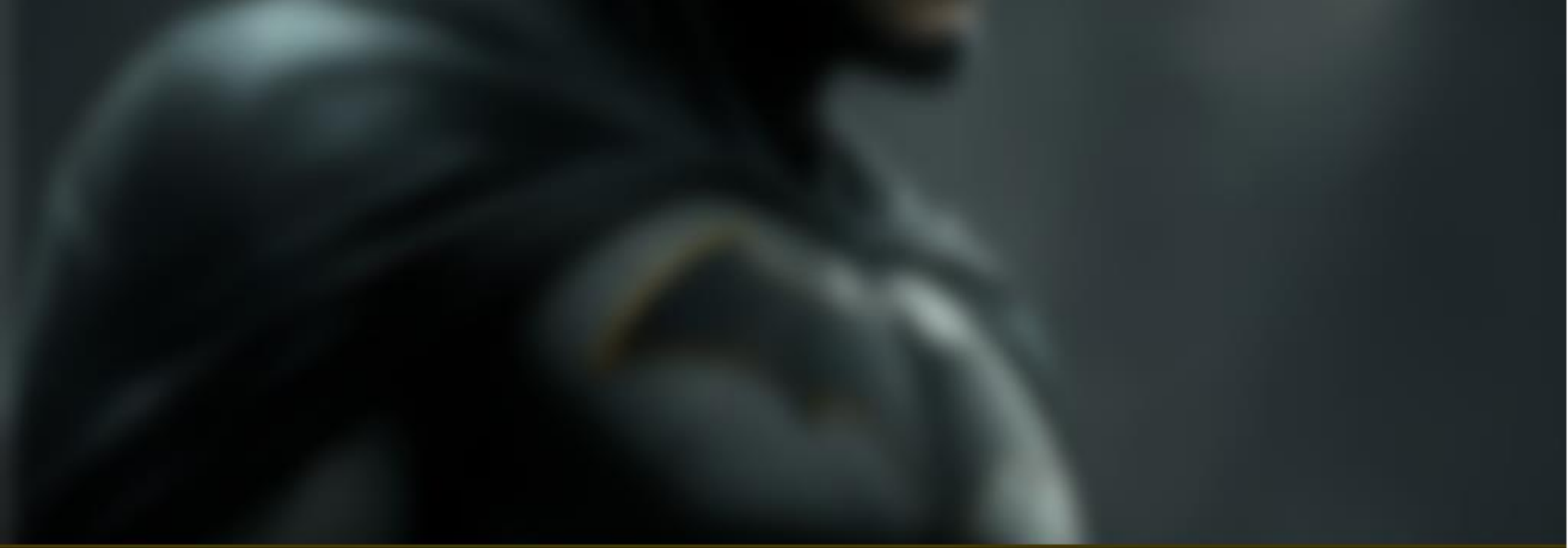
- **schema**: Object with the schema to validate. Can be Plain JSON schema, Swagger and OpenAPI documents.
- **path**: (optional) Object with the path to the schema definition within a Swagger or OpenAPI document.

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

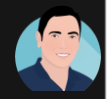
PATH (optional)

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.



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"
        }
    ]
}
```

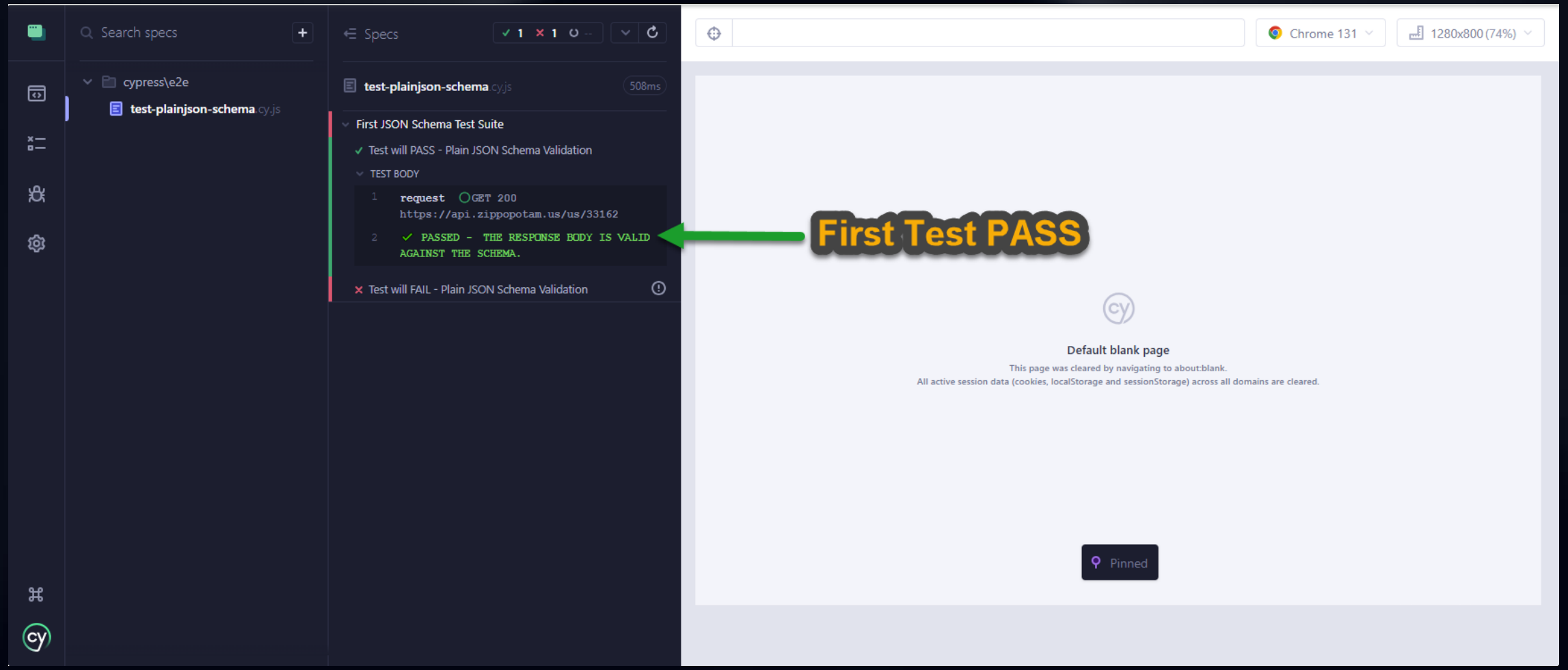
Plain JSON Schema

```
{
  "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"
  ]
}
```

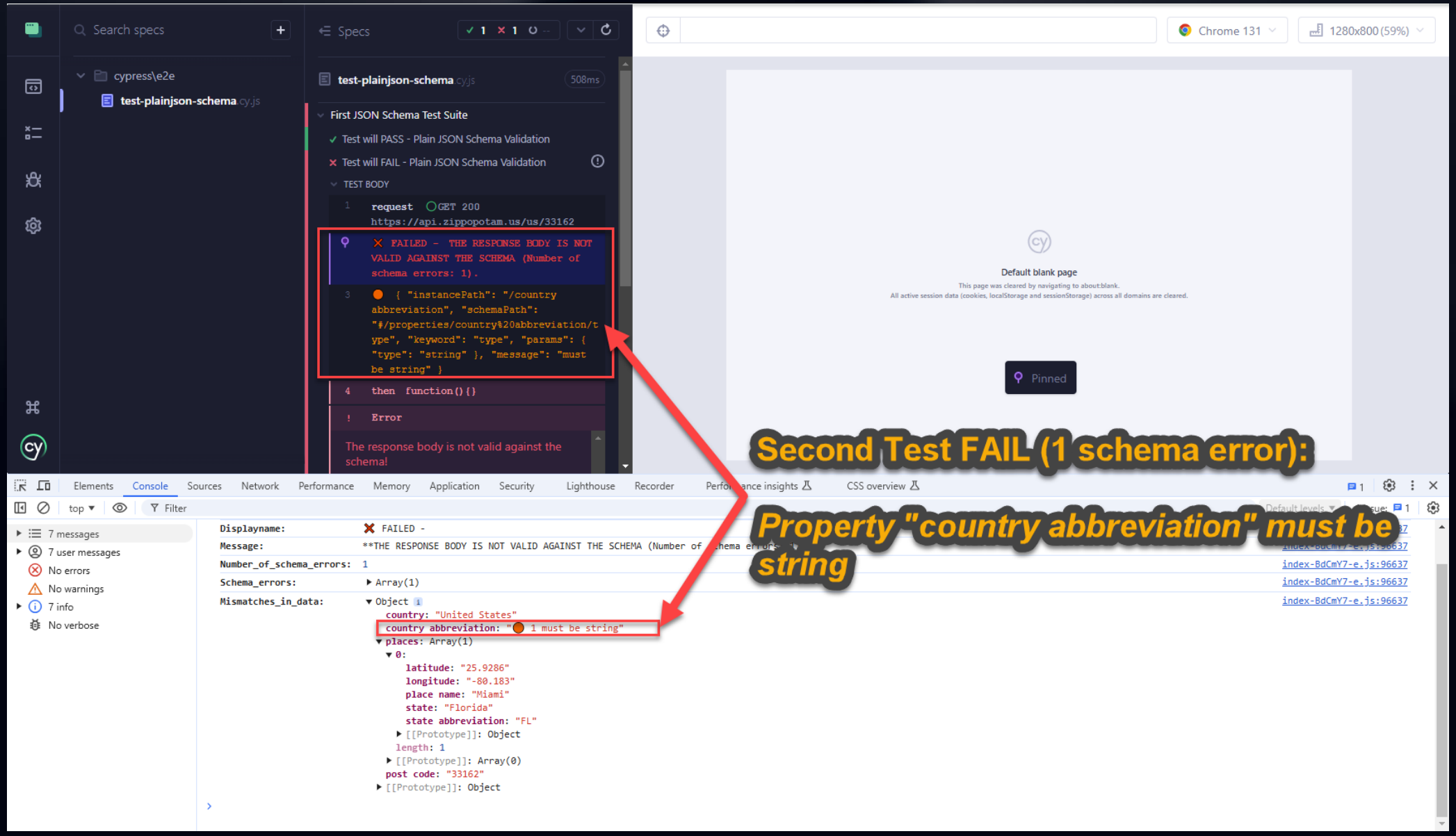
Example: Plain JSON Schema (Tests)

```
/// <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)
    })

})
```

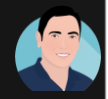
Example: Plain JSON Schema (Results – First Test PASS)



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



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": {
```


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

Commands.js

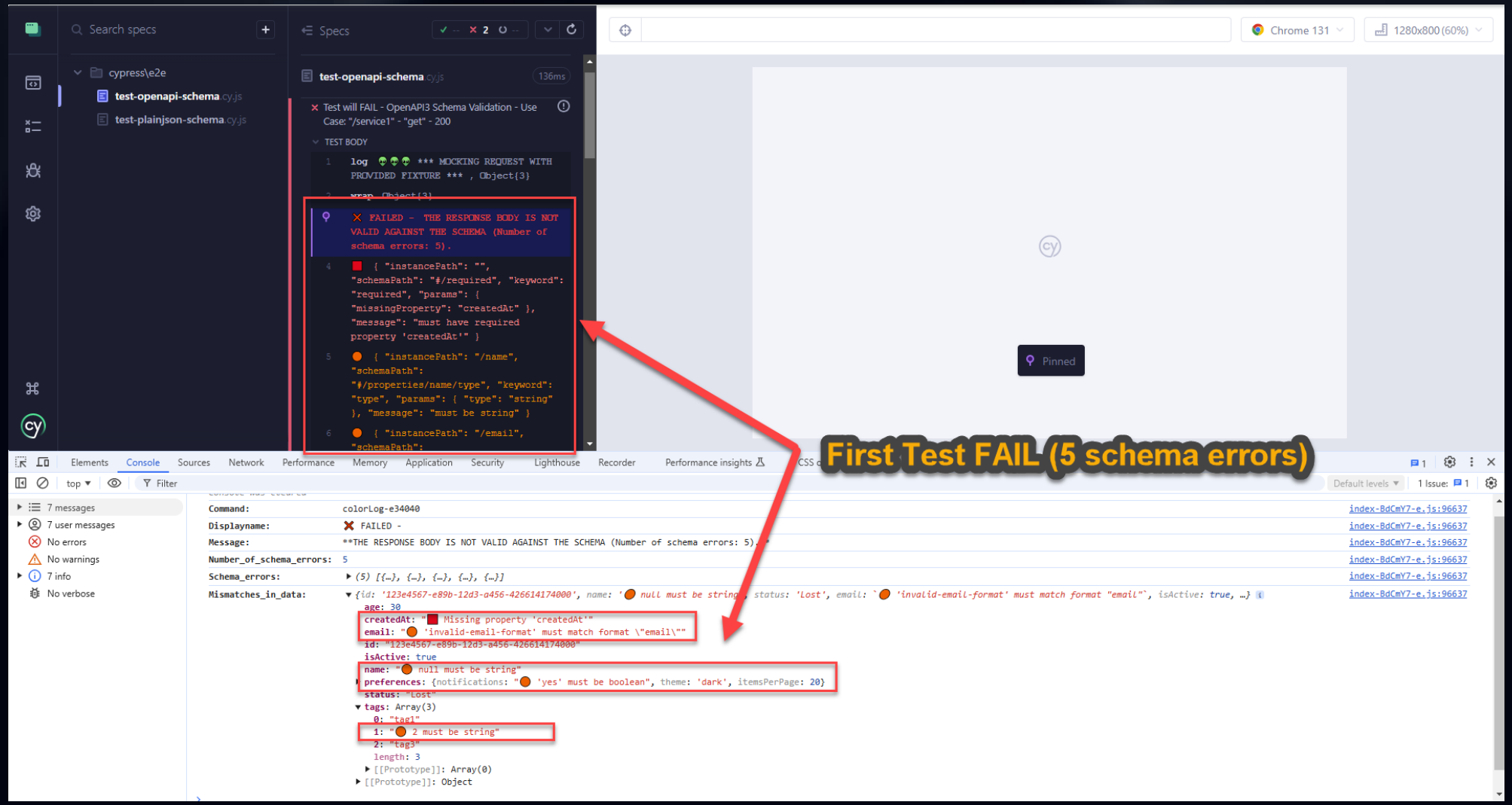
```
// ***********************************************
// This example commands.js shows you how to
// create various custom commands and overwrite
// existing commands.
//
// For more comprehensive examples of custom
// commands please read more here:
// https://on.cypress.io/custom-commands
// ***********************************************

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 })
    })
})
```

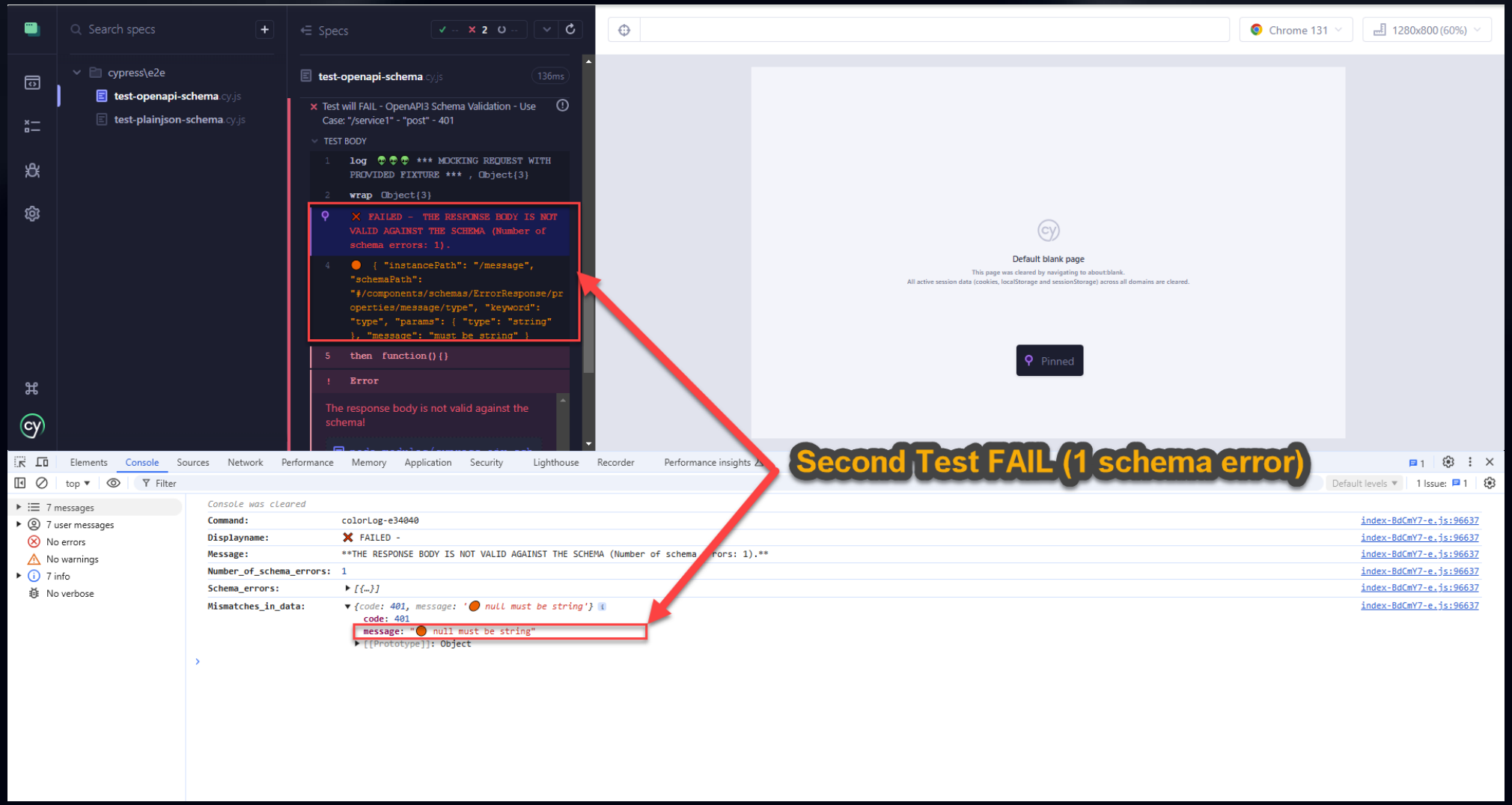
Example: OpenAPI 3 Schema Doc (Results – First Test FAIL)

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



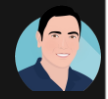
Example: OpenAPI 3 Schema Doc (Results – Second Test FAIL)

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



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,
  }
```

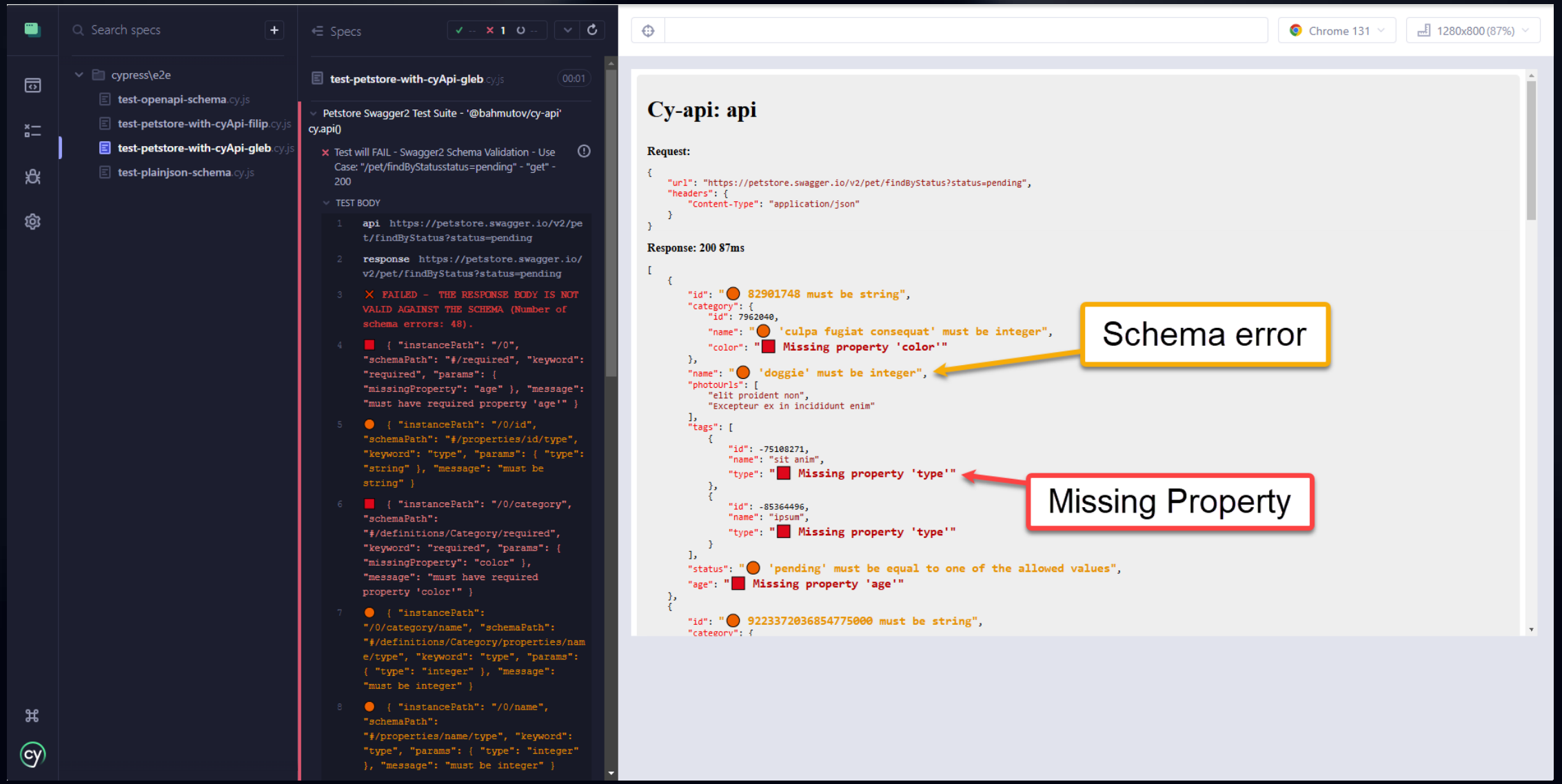
- Or in CLI

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

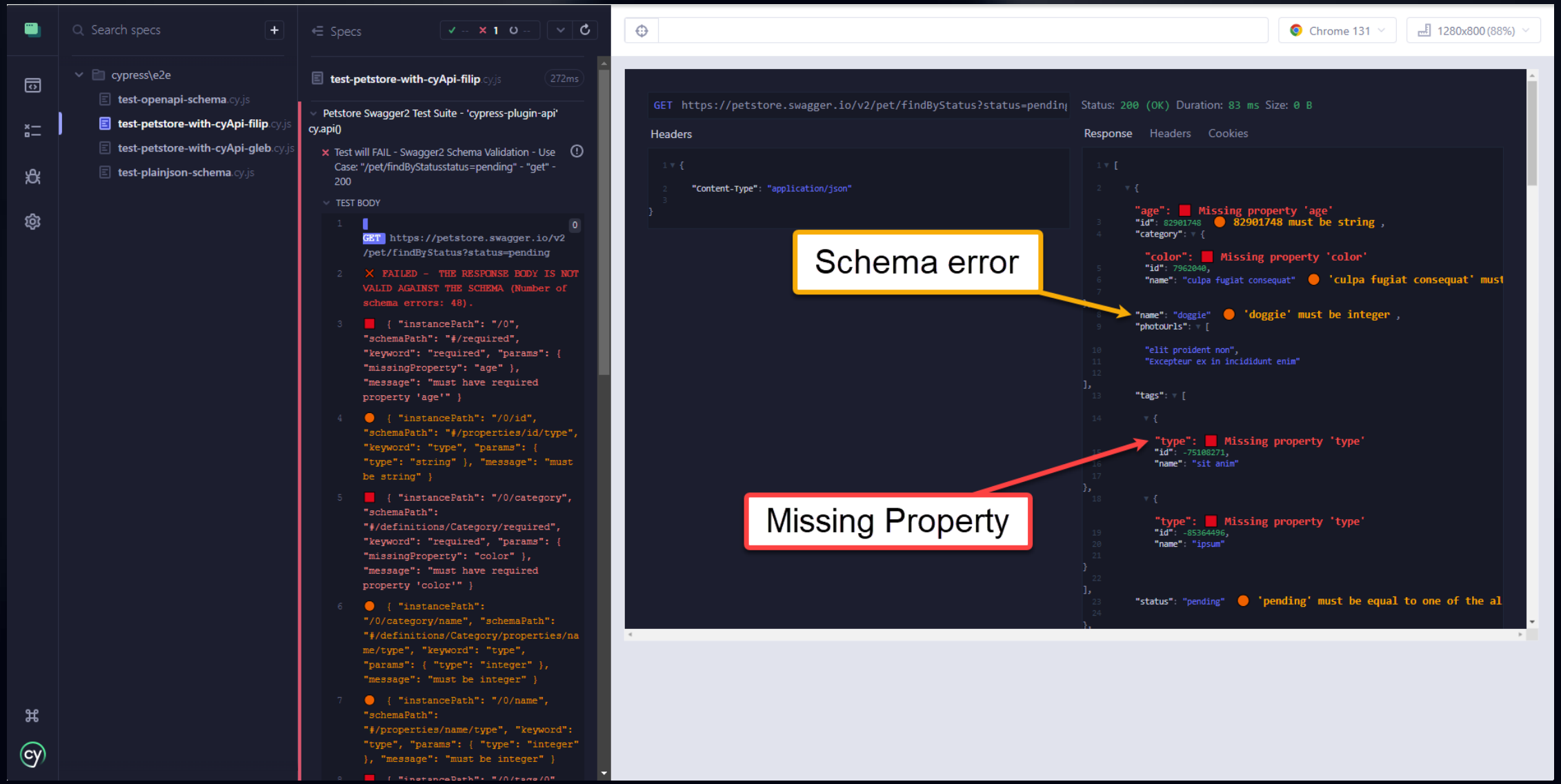
Example: Swagger2 Schema Doc with *cy.apy()* plugins (Test)

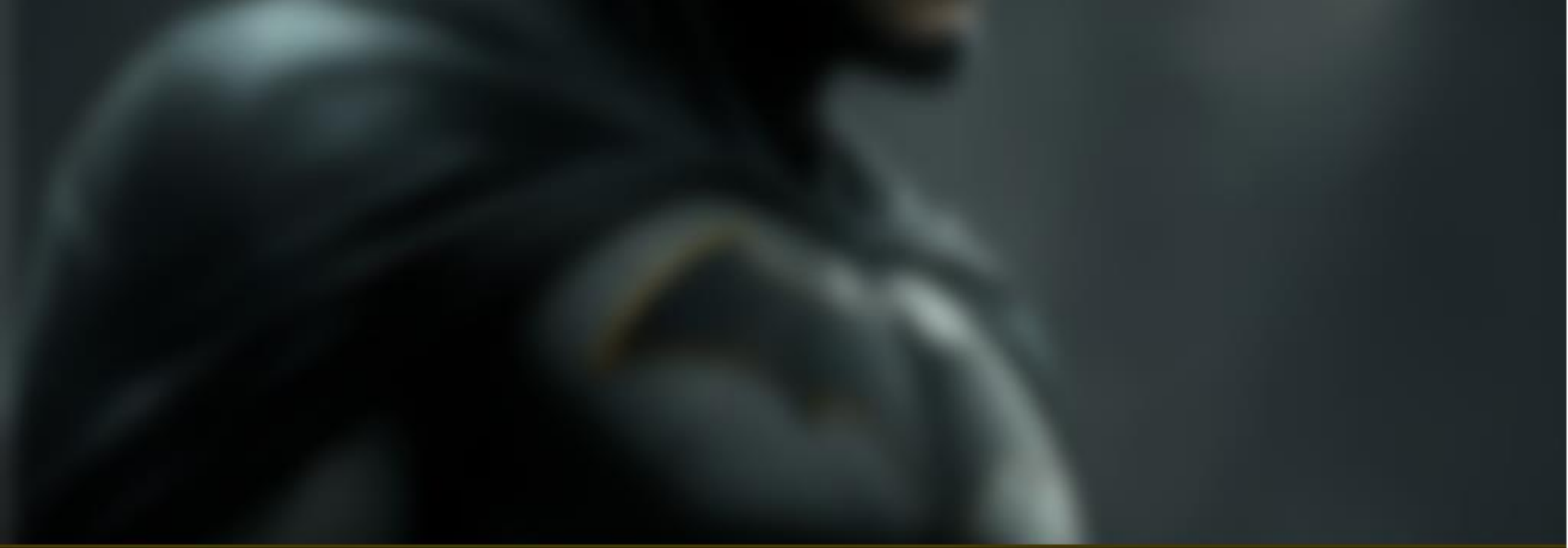
```
/// <reference types="cypress" />

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()*)

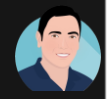


Example: Swagger2 Schema Doc (Results with *Filip's cy.apy()*)





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
}
```

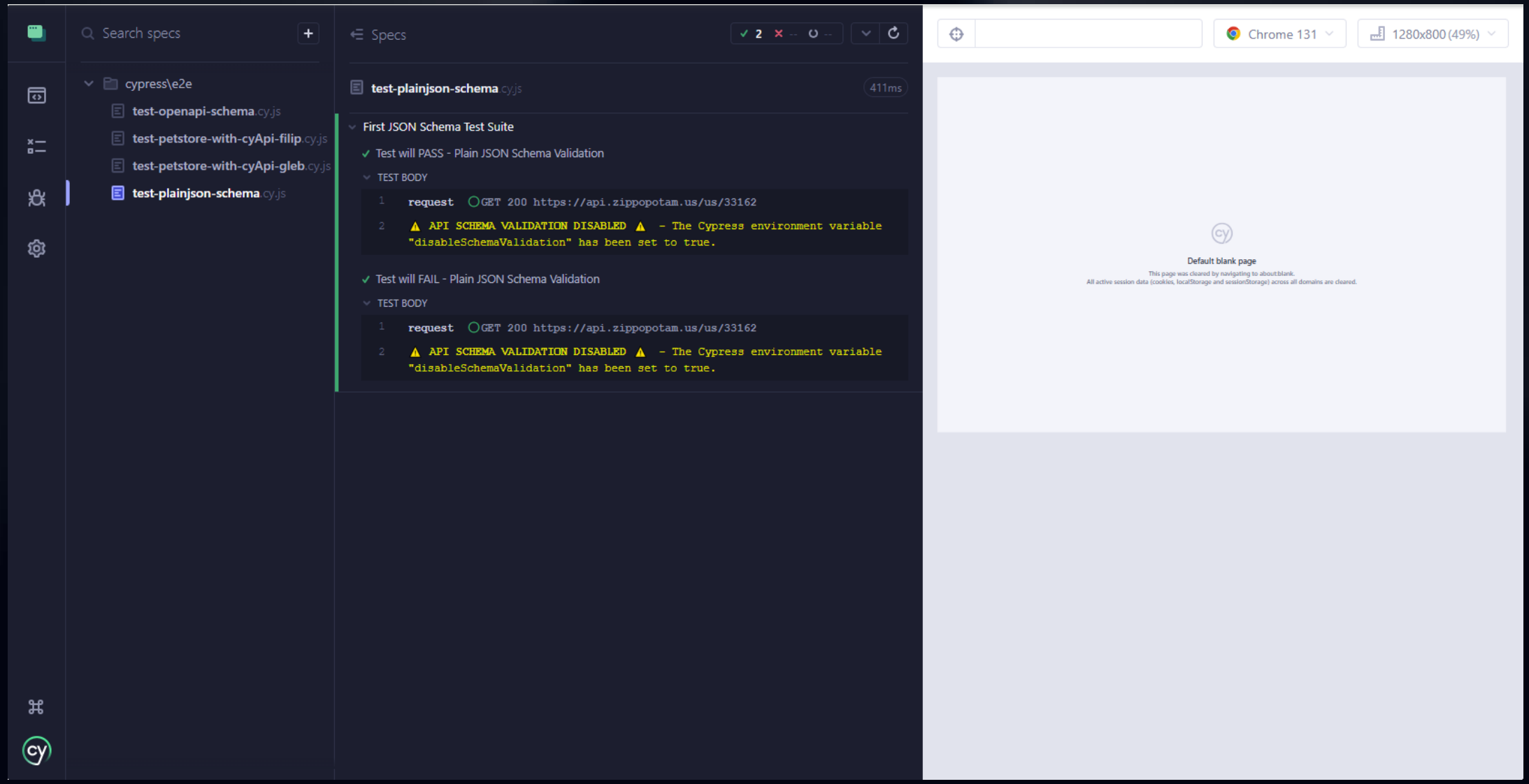
   - Or in *cypress.config.js*

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

module.exports = defineConfig({

  env: {
    disableSchemaValidation: true
  }
```

   - Or in CLI

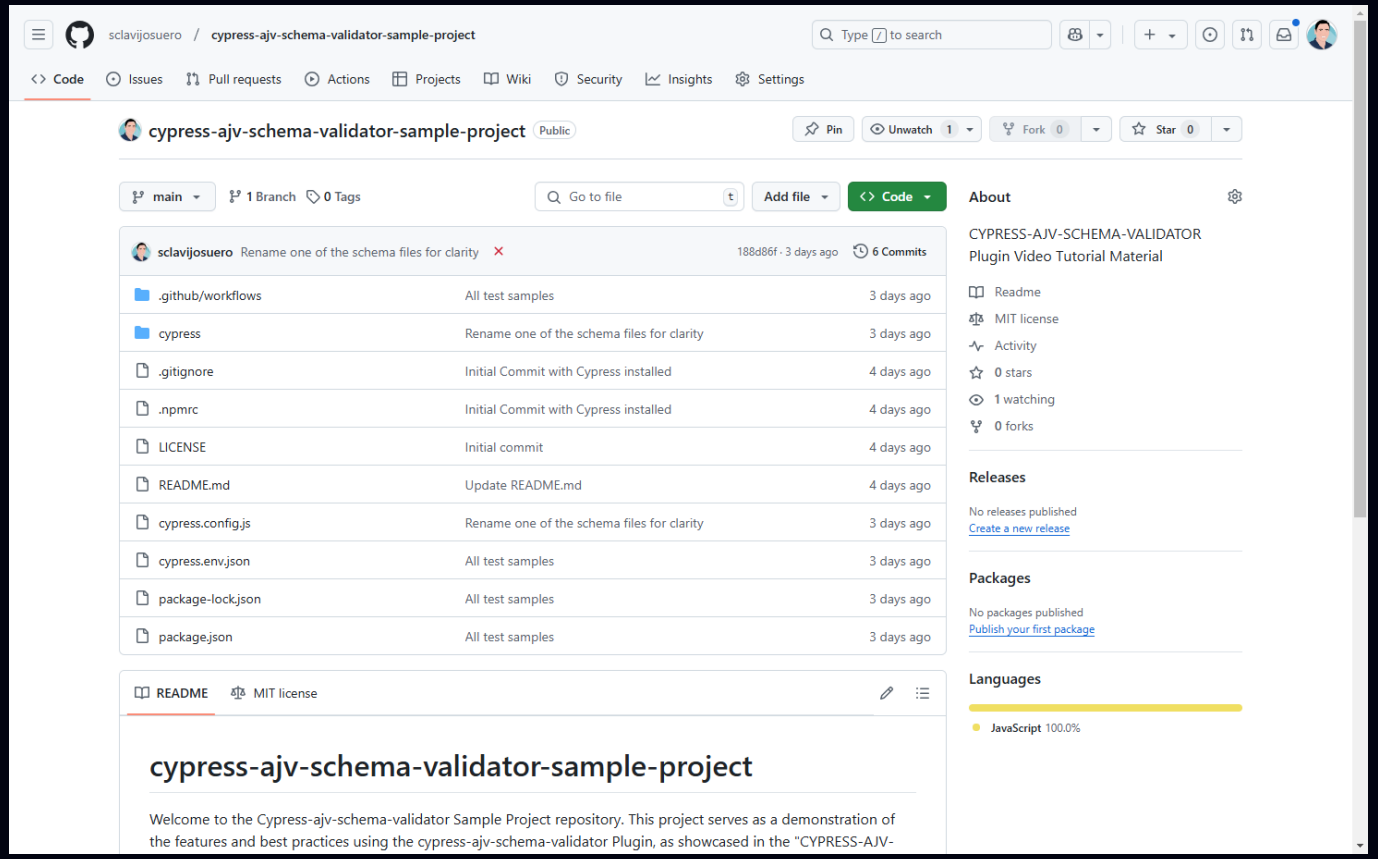
```
npx cypress open --env disableSchemaValidation=true
```

Example: Disable Schema Validation (Results)



Sample Project Repo

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



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)*