

API testing with Bruno

German perl workshop 2025

<https://usebruno.com>

What is Bruno?

- What postman used to be or should be
- Rest api client gui.
- Text based storage (use git and diff)
- Command line tool for automated testing
- Open source: <https://github.com/usebruno/bruno/>
- But some paid features

Install Bruno desktop app

- From <https://www.usebruno.com/>
- Runs local on windows, linux, macOs
- Built using Next.js and React.
Uses electron to ship a desktop version
- Tipp (for all electron apps): Ctrl-Shift-I for dev-tools (console)
- Demo data:
 - <https://dummyjson.com>
 - <https://restful-api.dev>
 - <https://reqbin.com>

Hello world

- Create collection
- Create request
- Run request



NEW REQUEST

Type

☒ HTTP ☐ GraphQL ☐ From cURL

Name

reqbin echo

URL

GET ▼

https://reqbin.com/echo

Cancel

Create

GET

▼ <https://dummyjson.com/products/:ID?TOKEN={{TOKEN}}>

Params²

Body

Headers

Auth

Vars

Script *

Assert¹

Tests

Docs

Query

Name	Path	
TOKEN	{{TOKEN}}	☒ 
l p	{{PRICE}}	☐ 

[+ Add Param](#)

Path 

Name	Value
ID	3

GET

POST

PUT

DELETE

PATCH

OPTIONS

HEAD

Environments

⌚ 🏃 👁 ⚙ 🌐 dev ▾

Collection Environments

- 🗄 dev
- 🗄 dev.sample
- 🗄 prod
- 🗄 No Environment
- ⚙ Configure

ENVIRONMENTS

dev

dev.sample

prod

[+ Create](#)

🗄 dev

[✎](#) [📄](#) [🗑](#)

Enabled	Name	Value	Secret	
☒	ID	1	☐	🗑
☒	TOKEN	asdfasdf	☐	🗑

[+ Add Variable](#)

Response

Headers ²²

Timeline

Tests ¹



200 OK

363ms

1.43KB

```
1 {  
2   "id": 3,  
3   "title": "Powder Canister",  
4   "description": "The Powder Canister is a finely milled setting po  
wder designed to set makeup and control shine. With a lightweight a  
nd translucent formula, it provides a smooth and matte finish.",  
5   "category": "beauty",  
6   "price": 14.99,  
7   "discountPercentage": 9.84,  
8   "rating": 4.64,  
9   "stock": 89,  
10  "tags": [  
11    "beauty",  
12    "face powder"  
13  ],
```

Response Headers ²² Timeline Tests ¹

[Clear Timeline](#)

200 OK GET [2025-05-03T10:01:13.145Z]

2 minutes ago

https://dummyjson.com/products/3?TOKEN=asdfasdf

Request

Response

Network Logs

https://dummyjson.com/products/3?TOKEN=asdfasdf

▼ Headers

No Headers found

▼ Body

No Body found

200 OK GET [2025-05-01T13:43:56.649Z]

2 days ago

https://dummyjson.com/products/4?TOKEN=asdfasdf

Create, change and delete in testing

POST  https://dummyjson.com/products/add   

Params Body • Headers Auth • Vars

Script Assert Tests Docs

JSON  Prettify

```
1 {
2   "title": "Powder Canister",
3   "description": "The Powder.",
4   "category": "beauty"
5 }
```

Response Headers ²² Timeline Tests

  201 Created 368ms 84B

```
1 {
2   "id": 195,
3   "title": "Powder Canister",
4   "description": "The Powder.",
5   "category": "beauty"
6 }
```

Store response data in variable

The screenshot shows a REST client interface. On the left is a sidebar with a search bar and a list of requests. The main area displays a POST request to `https://dummyjson.com/products/add` with the `Script` tab selected. The script contains two lines of code to store the response data in a variable named `DELID`.

Search requests ...

- > test
- > validate qs
- > präsi
- > powerfox
- ▼ präsi
 - GET product1
 - POST crud_product_1
 - GET crud_product_2
 - PATCH crud_product_3
 - DEL crud_product_4

POST `https://dummyjson.com/products/add`

Params Body * Headers Auth * Vars Script * Assert Tests

Docs

Pre Request

1

Post Response

```
1 // bru.setVar("DELID", res.body.id);
2 bru.setVar("DELID", 1);
```

Reuse response data in variable

The screenshot shows the Postman interface with a collection named 'präsi' expanded. The request is a DELETE method to the URL 'https://dummyjson.com/products/:ID?TOKEN={{TOKEN}}'. The 'Params' tab is active, showing a path parameter 'TOKEN' with the value '{{TOKEN}}'. Below this, the 'Path' tab is active, showing a path parameter 'ID' with the value '{{DELID}}'.

Collections ↑↓

Search requests ...

- > test
- > validate qs
- > präsi
- > powerfox
- ▼ präsi
 - GET product1
 - POST crud_product_1
 - GET crud_product_2
 - PATCH crud_product_3
 - DEL crud_product_4

DELETE crud_p... x +

DELETE ▼ https://dummyjson.com/products/:ID?TOKEN={{TOKEN}}

Params ² Body Headers Auth Vars Script Assert ¹ Tests

Docs

Query

Name	Path	
TOKEN	{{TOKEN}}	☒

+ Add Param

Path [?]

Name	Value
ID	{{DELID}}

- Simple assertions

- result

+ Add Assertion

Tests (0/0), Passed: 0, Failed: 0

Assertions (3/3), Passed: 3, Failed: 0

✓ res.body.id: eq {{TESTID}}

- ✓ `res.body.title`: `isString`

- ✓ res.body.title: contains Powder Canister

Check result with Javascript (Tests)

Params Body Headers Auth * Vars Script * Assert

Tests * Docs

```
1 test("title should contain 'Mascara'", () => {  
2   expect(res.body.title).toContain('Mascara');  
3 })
```

Response Headers ²² Timeline Tests ¹   200 OK 307ms 1.47KB

Tests (1/1), Passed: 1, Failed: 0

✓ title should contain 'Mascara'

Assertions (0/0), Passed: 0, Failed: 0

Testing in Javascript - Buildins

- atob - Turn base64-encoded ascii data back to binary.
- btoa - Turn binary data to base64-encoded ascii.
- chai - BDD/TDD assertion library for node.js and the browser
<https://www.chaijs.com/>.
- moment - Parse, validate, manipulate, and display dates and times in JavaScript.
- uuid - For the creation of RFC4122 UUIDs.
- nanoid - A tiny, secure, URL-friendly, unique string ID generator for JavaScript.
- crypto-js - JavaScript library of crypto standards.

Debugging tests with console.log()

```
1 test("brand should contain 'Mascara'", () => {  
2   console.log(res.body);  
3   expect(res.body.brand).toContain('Mascara');  
4 })
```

Tests (1/1), Passed: 0, Failed: 1

X brand should contain 'Mascara'
expected 'Essence' to include 'Mascara'

Assertions (0/0), Passed: 0, Failed: 0

The screenshot shows a web browser's developer console with the 'Console' tab selected. At the top, there are icons for Elements, Console, Sources, and Network. The Console shows a single error message with a red 'X' icon and a count of '1'. Below the error message, the JSON response is displayed, expanded to show its properties. The error message is: 'brand should contain 'Mascara'' with the expected value 'Essence' to include 'Mascara'. The JSON response is: {id: 1, title: 'Essence Mascara Lash Princess', description: 'The Essence Mascara Lash Princess is a popular mas... with this long-lasting and cruelty-free formula.', category: 'beauty', price: 9.99, ...}. The 'dimensions' property is expanded to show width: 15.14, height: 13.08, and depth: 22.99. The 'discountPercentage' is 10.48.

index.82df5080.js:681

```
{id: 1, title: 'Essence Mascara Lash Princess', description: 'The E  
ssence Mascara Lash Princess is a popular mas... with this long-lasti  
ng and cruelty-free formula.', category: 'beauty', price: 9.99, ...}
```

availabilityStatus: "In Stock"
brand: "Essence"
category: "beauty"
description: "The Essence Mascara Lash Princess is a popular masca
► dimensions: {width: 15.14, height: 13.08, depth: 22.99}
discountPercentage: 10.48

Bruno CLI

- `npm install -g @usebruno/cli`
- `bru run request.bru`
- `bru run --env=dev request.bru`
- `npx @usebruno/cli run --env=dev`
- `npx @usebruno/cli run --env=dev --output results.html --format html`

```
C:\Users\Public\Documents\bruno\praesi>set NODE_OPTIONS="--no-deprecation"
```

```
C:\Users\Public\Documents\bruno\praesi>npx @usebruno/cli run --env=dev
```

Running Folder Recursively

```
product1 (200 OK) - 401 ms
```

```
  ✓ assert: res.body.id: eq {{TESTID}}
```

```
  ✓ assert: res.body.title: isString
```

```
  ✓ assert: res.body.title: contains Powder Canister
```

```
crud_product_1 (201 Created) - 102 ms
```

```
crud_product_2 (200 OK) - 104 ms
```

```
  ✓ brand should contain 'Essence'
```

```
crud_product_3 (200 OK) - 101 ms
```

```
  ✓ assert: res.body.id: eq {{ID}}
```

```
crud_product_4 (200 OK) - 104 ms
```

```
  ✓ assert: res.body.id: eq {{ID}}
```

```
Requests:      5 passed, 5 total
```

```
Tests:         1 passed, 1 total
```

```
Assertions:    5 passed, 5 total
```

```
Ran all requests - 812 ms
```

```
Requests:      5 passed, 5 total
```

```
Tests:         1 passed, 1 total
```

```
Assertions:    5 passed, 5 total
```



Summary

Requests

Iteration 1



Total requests

5



Total errors

0



Total Controls

4



Total Failed Controls

0

Total run duration

0.912 s

SUMMARY ITEM

TOTAL

PASSED

FAILED

SKIPPED

ERROR

Requests

5

5

0

0

0

Assertions

4

4

0

-

-



Summary

Requests

Iteration 1

☐ Show All



product1 - 2/2 Passed

REQUEST INFORMATION

File

product1

Request Method

GET

RESPONSE INFORMATION

Response Code

200

Response time

375 ms

Real live (run in gitlab)

- Gitlab Pipeline schedules
- .gitlab-ci.yml

```
stages: - brunoTest
bruno_test:
script:
  - pushd tests/Bruno/
  - npm install
  - npx @usebruno/cli run --env=dev
  - popd
```

tests/Bruno/package.json

 package.json  131 B

```
1  {
2      "name": "bruno-validate-api-collection",
3      "version": "1.0.0",
4      "dependencies": {
5          "jsonschema": "^1.4.1"
6      }
7  }
```

- Version 1.5.0 is broken
- Install with „npm i“

Variables in

- Requests
 - Collection
 - Bruno Environment
 - .env-file
 - Environment vars
-
- Use: `{{VAR}}` or `{{process.env.ENVVAR}}`

Reusable code in js-module

JS helper.js 2.37 KiB

```
1  const getElem = (res, name, value) => {
2      return res.body.results[0][name].find((e)=>e.validatorName === value);
3  };
4
5  const testSchema = (res) => {
6      var Validator = require('jsonschema').Validator;
7      var v = new Validator();
8      var schema = {
9          "id": "/ValidateResultV1",
10         "type": "object",
11         "properties": {
12             "requestUUID": {
13                 "type": "string",
14                 "pattern": /^[0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{12}$/ ,
15             },
16             "status": {
```

Test with helper module

```
60     const h = require("../lib/helper");
61
62     // schema
63     test("body should be valid schema. check console on failure", () => {
64         expect(h.testSchema(res).errors).to.be.empty;
65     });
66
67     // firstName
68     test("firstName should have validator badword with result false", () => {
69         expect(h.getElem(res, "firstName", "badword").result).to.be.false;
70     });
```

Debugging

- With `console.log`
- With node debugger, but tricky.
- If you need a debugger, maybe Bruno is not the right tool
- If you program a lot, it is useful. But that's not what it's for

Can I do the same in Playwright?



Compare Bruno and Playwright

Bruno

- + easy install Gui and runtime
- + simple API testing
- + GUI to develop tests

Playwright

- + debuggable (e.g. vscode plugin)
- + complex API testing (prepare/cleanup)
- + integration with other testing
- big installation