# **How to create Tests in Postman**

[**HOME**](https://qaautomation.expert/)

In this tutorial, we will see how we can ***create Tests in the Postman***.

***Table of Contents***

1. [***Create a request in Postman***](https://qaautomation.expert/2023/08/14/how-to-create-tests-in-postman/#create-a-request-in-postman)
2. [***Create Tests for the request***](https://qaautomation.expert/2023/08/14/how-to-create-tests-in-postman/#create-tests-for-the-request)
3. [***Type of Tests***](https://qaautomation.expert/2023/08/14/how-to-create-tests-in-postman/#type-of-tests)
4. [***Test to check Status code***](https://qaautomation.expert/2023/08/14/how-to-create-tests-in-postman/#1-test-to-check-status-code)
5. [***Test to check the Status code from a list of codes***](https://qaautomation.expert/2023/08/14/how-to-create-tests-in-postman/#2-test-to-check-the-status-code-from-a-list-of-codes)
6. [***Test to check status code is not 404***](https://qaautomation.expert/2023/08/14/how-to-create-tests-in-postman/#3-test-to-check-status-code-is-not-404)
7. [***Test to check the status code name***](https://qaautomation.expert/2023/08/14/how-to-create-tests-in-postman/#4-test-to-check-the-status-code-name)
8. [***Test to check Content-Type is present***](https://qaautomation.expert/2023/08/14/how-to-create-tests-in-postman/#5-test-to-check-content-type-is-present)
9. [***Test to check the expected header***](https://qaautomation.expert/2023/08/14/how-to-create-tests-in-postman/#6-test-to-check-the-expected-header)
10. [***Test to check response body contains string***](https://qaautomation.expert/2023/08/14/how-to-create-tests-in-postman/#7-test-to-check-response-body-contains-string)
11. [***Test to check response body contains JSON value***](https://qaautomation.expert/2023/08/14/how-to-create-tests-in-postman/#8-test-to-check-response-body-contains-json-value)
12. [***Test to check field value in response***](https://qaautomation.expert/2023/08/14/how-to-create-tests-in-postman/#9-test-to-check-field-value-in-response)
13. [***Test to check response time***](https://qaautomation.expert/2023/08/14/how-to-create-tests-in-postman/#10-test-to-check-response-time)
14. [***Test an assertion value type***](https://qaautomation.expert/2023/08/14/how-to-create-tests-in-postman/#11-test-an-assertion-value-type)
15. [***Test to check multiple assertions***](https://qaautomation.expert/2023/08/14/how-to-create-tests-in-postman/#12-test-to-check-multiple-assertions)
16. [***Test to check nested field value in response***](https://qaautomation.expert/2023/08/14/how-to-create-tests-in-postman/#13-test-to-check-nested-field-value-in-response)
17. [***Test to check nested array value in response***](https://qaautomation.expert/2023/08/14/how-to-create-tests-in-postman/#14-test-to-check-nested-array-value-in-response)

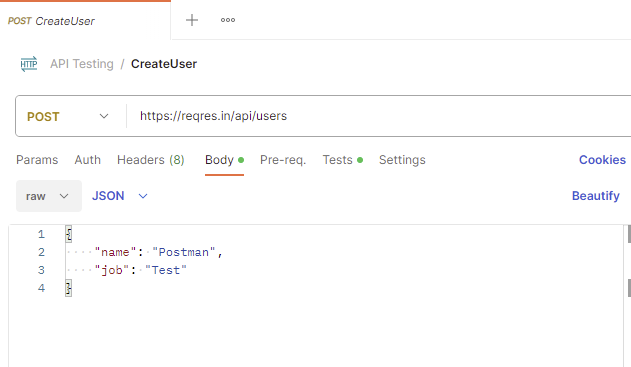
Consider a sample JSON which has 4 sets of data. This data we will be passing to a POST request.

### ***Create a request in Postman***

Create a request in Postman. Below is the URL of the request.

|  |  |
| --- | --- |
| 1 | https://reqres.in/api/users |

Paste the body of the request in the Body part of the Postman.



### ***Create Tests for the request***

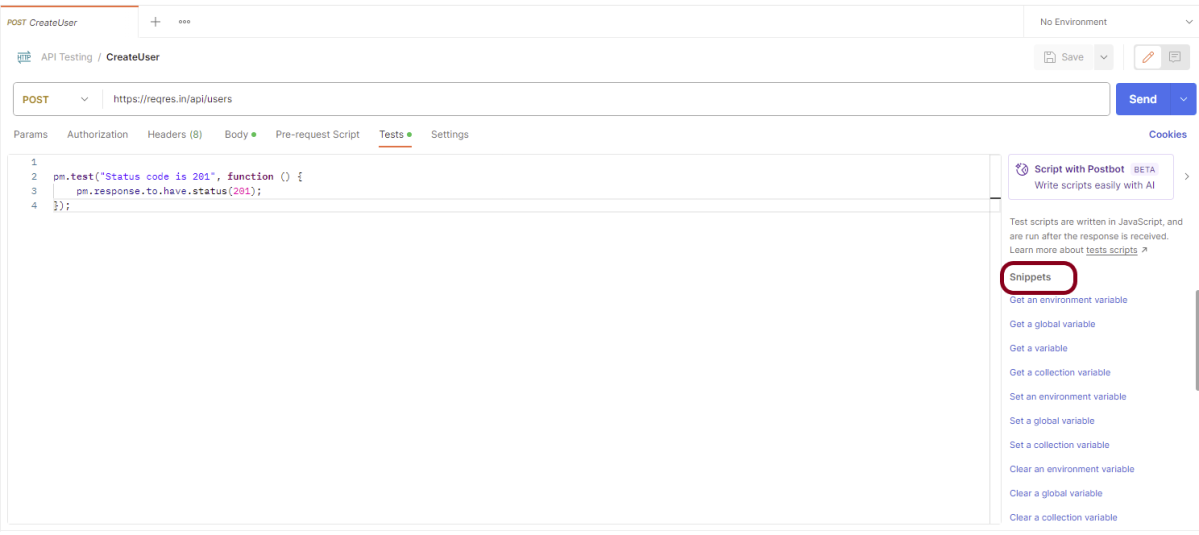
Click on the “Tests”. Write the tests as shown below:

Advertisements

REPORT THIS AD

* ***pm.test****: This is the function for writing test specifications,****pm****here refers to Postman api and****test****refers to the specification of the function which is for testing purposes*
* ***Status code is 201 :****This line is just a string which is the test name. When your test is performed, this string will be written in front of the result.*
* ***function(){}:****The next parameter is the function that is passed to perform the test*
* ***pm.response :****This is used to capture the response received and perform assertions on it such as status code, headers etc.*

If you need help in creating the test, you can refer to the snippets present on the right side of the console.



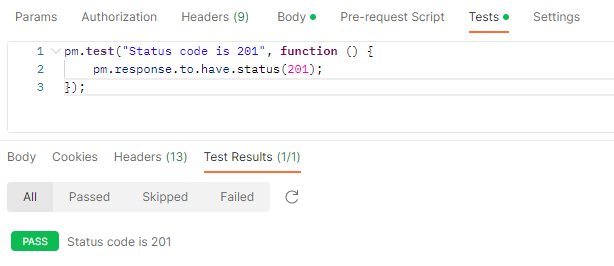
### ***Type of Tests***

#### ***1.Test to check Status code***

This test check the status code of the response. Here, we are checking status code if the status code is 201, then pass the test.

|  |  |
| --- | --- |
| 1  2  3 | pm.test("Status code is 201", function () {      pm.response.to.have.status(201);  }); |

The output of the above test is

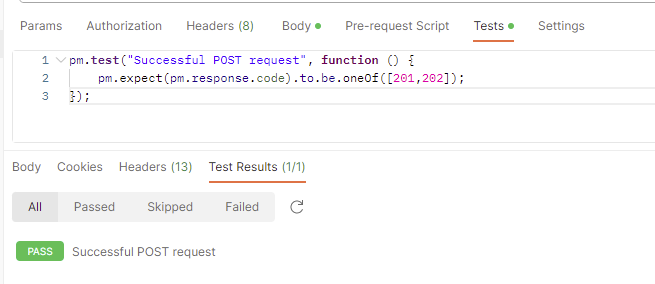


#### ***2. Test to check the Status code from a list of codes***

This test check the status code of the response. Here, we are checking status code if the status code is either 201 or 202, then pass the test.

|  |  |
| --- | --- |
| 1  2  3 | pm.test("Successful POST request", function () {      pm.expect(pm.response.code).to.be.oneOf([201,202]);  }); |

The output of the above test is

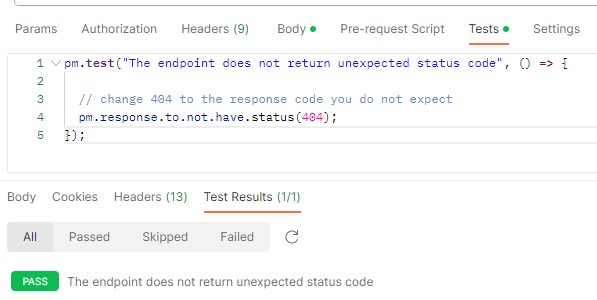


#### ***3. Test to check status code is not 404***

This test check the status code of the response. Here, we are checking status code if the status code is not 404, then pass the test.

|  |  |
| --- | --- |
| 1  2  3  4  5 | pm.test("The endpoint does not return unexpected status code", () => {      // change 404 to the response code you do not expect    pm.response.to.not.have.status(404);  }); |

The output of the above test is

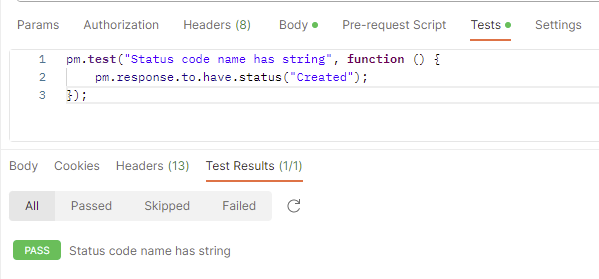


#### ***4. Test to check the status code name***

This test check the name of the status of the response. Here, we are checking if the name of the status is ***“Created”***, then pass the test.

|  |  |
| --- | --- |
| 1  2  3 | pm.test("Status code name has string", function () {      pm.response.to.have.status("Created");  }); |

The output of the above test is

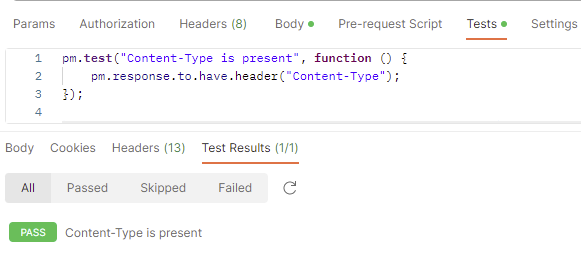


#### ***5. Test to check Content-Type is present***

This test check if the response contains header as Content-Type, then pass the test.

|  |  |
| --- | --- |
| 1  2  3 | pm.test("Content-Type is present", function () {      pm.response.to.have.header("Content-Type");  }); |

The output of the above test is

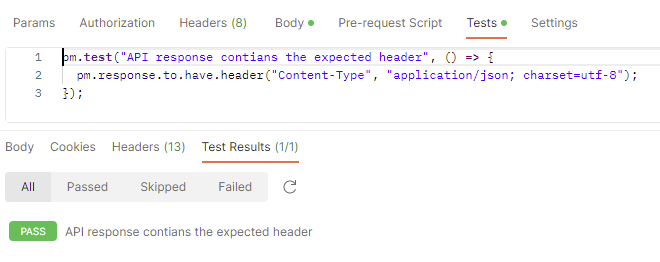


#### ***6. Test to check the expected header***

This test check the headers like content-type, accept, token and so on. Here, we have checked if the content-type is “application/json” or not. If yes, then pass the test.

|  |  |
| --- | --- |
| 1  2  3 | pm.test("API response contians the expected header", () => {    pm.response.to.have.header("Content-Type", "application/json; charset=utf-8");  }); |

The output of the above test is

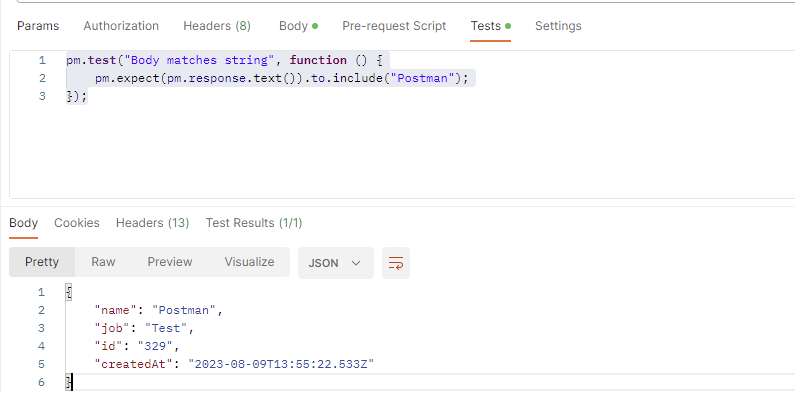


#### ***7. Test to check response body contains string***

This test checks the text in the response. Here, it is checking if the response has text “Postman”, then passes the test.

|  |  |
| --- | --- |
| 1  2  3 | pm.test("Body matches string", function () {      pm.expect(pm.response.text()).to.include("Postman");  }); |

The output of the above test is

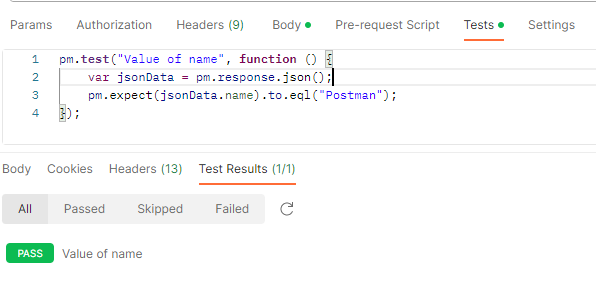


#### ***8. Test to check response body contains JSON value***

This test checks the JSON data in the response. Here, it is checking if the JSON response contains attribute name whose value is “Postman”, then passes the test.

|  |  |
| --- | --- |
| 1  2  3  4 | pm.test("Value of name", function () {      var jsonData = pm.response.json();      pm.expect(jsonData.name).to.eql("Postman");  }); |

The output of the above test is



#### ***9. Test to check field value in response***

This test checks the key-value of the response. Here, it is checking if the response has key as “name “and value as “Postman” and key as job and name as “Test”, then passes the test.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | pm.test("API response contains the expected fields", () => {  **const** response = pm.response.json();      // the line below checks value of the name field is Postman.    pm.expect(response).to.have.property("name", "Postman");      // the line below checks value of the job field is Test (string).    pm.expect(response).to.have.property("job", "Test");  }); |

The output of the above test is

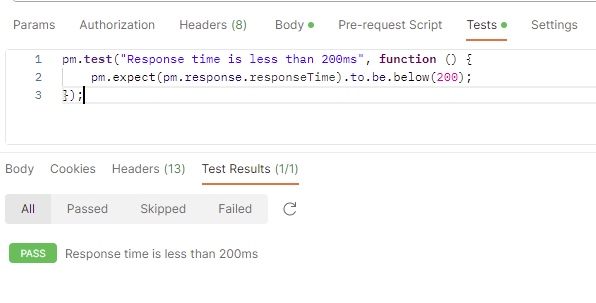


#### ***10. Test to check response time***

This test check the response time of the response. Here, it is checking is the time taken by the response to return is less than 200 ms, then pass the test.

|  |  |
| --- | --- |
| 1  2  3 | pm.test("Response time is less than 200ms", function () {      pm.expect(pm.response.responseTime).to.be.below(200);  }); |

The output of the above test is



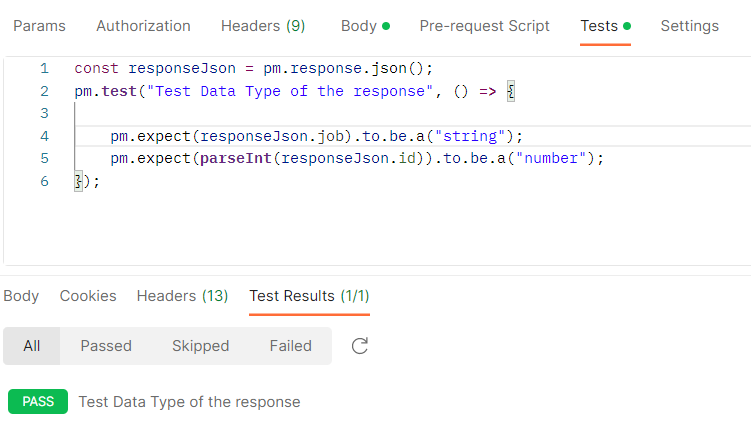
#### ***11. Test an assertion value type***

This test check the test data type of the values of the keys. Here, we are checking if the data type for the id is number, then pass the test.

Test the type of any part of the response:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | **const** responseJson = pm.response.json();  pm.test("Test Data Type of the response", () => {        pm.expect(responseJson.job).to.be.a("string");      pm.expect(parseInt(responseJson.id)).to.be.a("number");  }); |

The output of the above test is

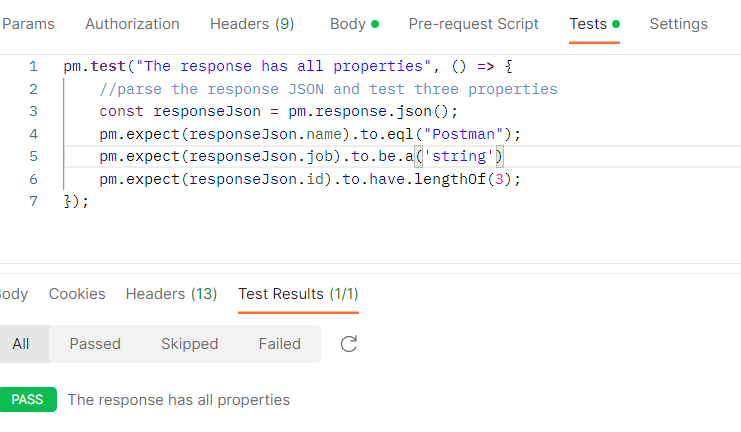


#### ***12. Test to check multiple assertions***

This test checks the multiple assertions in the response.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | pm.test("The response has all properties", () => {      //parse the response JSON and test three properties  **const** responseJson = pm.response.json();      pm.expect(responseJson.name).to.eql("Postman");      pm.expect(responseJson.job).to.be.a('string')      pm.expect(responseJson.id).to.have.lengthOf(3);  }); |

The output of the above test is

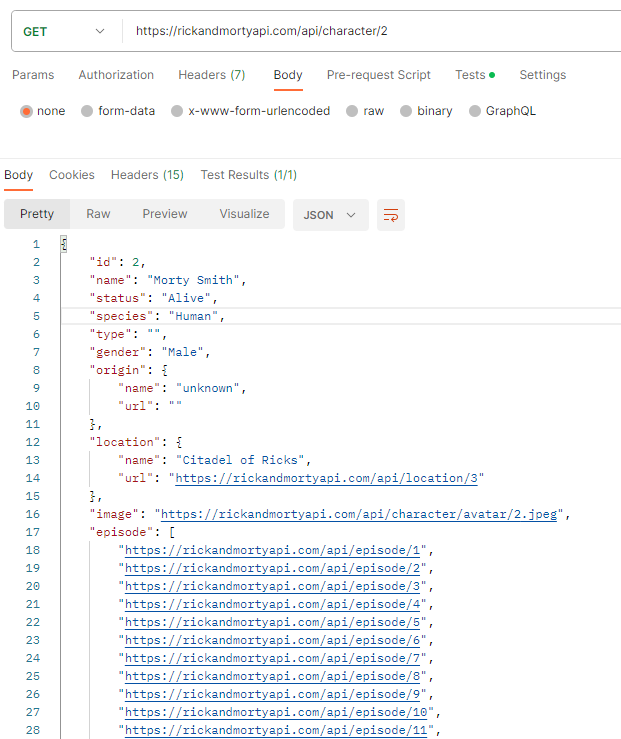


If any of the contained assertions fails, the test as a whole will fail. All assertions must be successful for the test to pass.

#### ***13. Test to check nested field value in response***

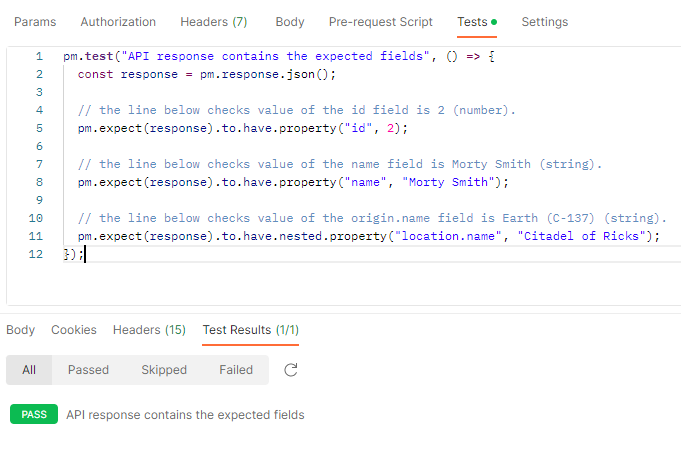
This test checks the nested field value in the response.

Sample Response



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | pm.test("API response contains the expected fields", () => {  **const** response = pm.response.json();      // the line below checks value of the id field is 2 (number).    pm.expect(response).to.have.property("id", 2);      // the line below checks value of the name field is Morty Smith (string).    pm.expect(response).to.have.property("name", "Morty Smith");      // the line below checks value of the origin.name field is Earth (C-137) (string).    pm.expect(response).to.have.nested.property("location.name", "Citadel of Ricks");  }); |

The output of the above test is

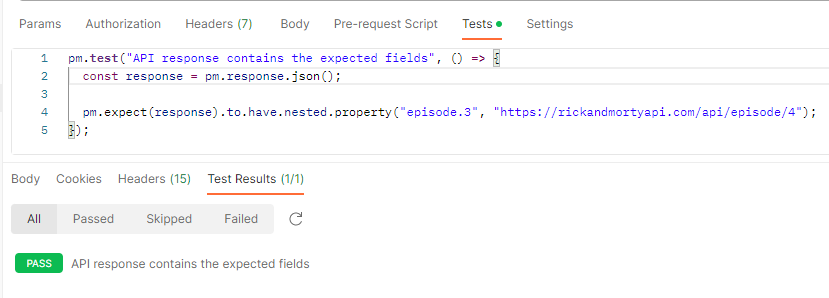


#### ***14. Test to check nested array value in response***

We can use the below script to check the value of the second item in the  episode array of the endpoint.

|  |  |
| --- | --- |
| 1  2  3  4  5 | pm.test("API response contains the expected fields", () => {  **const** response = pm.response.json();      pm.expect(response).to.have.nested.property("episode.3", "<https://rickandmortyapi.com/api/episode/4>");  }); |

The output of the above test is



In JavaScript, the position of the items in lists (array) starts from 0, so the first item has 0 as the position number, the second one 1, and so on.

Advertisements

REPORT THIS AD

We are done! Congratulations on making it through this tutorial and hope you found it useful! Happy Learning!!