

Id found tests

The image displays two screenshots of the Postman application interface, showing the configuration and testing of a GET request to a placeholder API.

Top Screenshot: Pre-request Script Tab

- Request:** GET `{{baseUrl}}/posts/1`
- Pre-request Script:**

```
1 pm.sendRequest({
2   url: "https://jsonplaceholder.typicode.com/posts/1",
3   method: "GET"
4 },function(err,response){
5   const user = response.json();
6   pm.variables.set("idFound",false);
7   if(response.statusCode === 200){
8     pm.variables.set("userId",user.userId);
9     pm.variables.set("id",user.id);
10    pm.variables.set("title",user.title);
11    pm.variables.set("idFound",true);
12  }
13  else{
14    pm.variables.set("idFound",false);
15  }
16 });
```

Bottom Screenshot: Tests Tab

- Request:** GET `{{baseUrl}}/posts/1`
- Tests:**

```
30 console.log("id found",idFound);
31
32 if(idFound){
33   pm.test("User id should not be empty",function(){
34     pm.expect(userId).to.be.not.null;
35   });
36
37 pm.test("Id should not be empty",function(){
38   pm.expect(id).to.be.not.null;
39 });
40
41 pm.test("title should not be empty",function(){
42   pm.expect(title).to.be.not.empty;
43 });
44
45 }
46 else{
47   pm.test("id is not found",function(){
48     pm.expect(idFound).to.eql(false);
49   });
50 }
```

Postman interface showing a collection named "JsonPlaceholderTest" with a sub-collection "allposttest". The "allposttest" sub-collection contains a "GET allposttest" request. The "Test Results (9/9)" tab is active, displaying a table of test results:

All	Passed	Skipped	Failed
PASS	status code is 200		
PASS	Response body as string		
PASS	id=1		
PASS	Content-Type is present		
PASS	Response time is less than 200ms		
PASS	Status code name has string		
PASS	User id should not be empty		
PASS	id should not be empty		
PASS	title should not be empty		

The interface also shows a sidebar with "Collections" and "Environments". The "Environments" section is active, displaying "jsonPlaceholderEnv".

Postman interface showing the "Console" tab. The console displays the following log entries:

```
1
1
"sunt aut facere repellat provident occaecati excepturi optio reprehenderit"
"id found" true
> GET https://jsonplaceholder.typicode.com/posts/1 200 81 ms </>
> GET https://jsonplaceholder.typicode.com/posts/ 200 72 ms
1
1
"sunt aut facere repellat provident occaecati excepturi optio reprehenderit"
"id found" true
```

The interface also shows a sidebar with "Collections" and "Environments". The "Environments" section is active, displaying "jsonPlaceholderEnv".

Id not found

The screenshot shows the Postman interface with a workspace named 'Assignment'. The left sidebar displays a collection 'JsonPlaceholder' with several endpoints. The 'allpoststest' endpoint is selected. The main panel shows a GET request to 'https://jsonplaceholder.typicode.com/posts/1'. The 'Pre-request Script' tab is active, containing a JavaScript function that sets variables based on the response status. The status bar at the bottom indicates a successful 200 OK response.

```
1 pm.sendRequest({
2   url: "https://jsonplaceholder.typicode.com/posts/1",
3   method: "GET"
4 },function(response){
5   const user = response.json();
6   pm.variables.set("idFound",false);
7   if(response.status === 200){
8     pm.variables.set("userId",user.userId);
9     pm.variables.set("id",user.id);
10    pm.variables.set("title",user.title);
11    pm.variables.set("idFound",true);
12  }
13  else{
14    pm.variables.set("idFound",false);
15  }
16 });
```

This screenshot shows the same Postman interface, but with the 'Tests' tab active. It contains JavaScript code to verify the response. The status bar shows a 200 OK response. The 'Test Results' section at the bottom indicates that all 7 tests passed.

```
30 console.log("id found",idFound);
31
32 if(idFound){
33   pm.test("User id should not be empty",function(){
34     pm.expect(userId).to.be.not.null;
35   });
36
37   pm.test("Id should not be empty",function(){
38     pm.expect(id).to.be.not.null;
39   });
40
41   pm.test("title should not be empty",function(){
42     pm.expect(title).to.be.not.empty;
43   });
44 }
45 else{
46   pm.test("id is not found",function(){
47     pm.expect(idFound).to.eql(false);
48   });
49 }
50 }
```

Postman interface showing a collection named "JsonPlaceholderTest" with a sub-collection "allposttest". The selected test is "allposttest" (GET). The URL is `{{baseUri}}/posts/`. The test results show 7/7 tests passed:

- STATUS: 200 OK, 70 ms, 27.98 KB
- Test Results (7/7):
 - PASS: Status code is 200
 - PASS: Response body as string
 - PASS: id=1
 - PASS: Content-Type is present
 - PASS: Response time is less than 200ms
 - PASS: Status code name has string
 - PASS: Id is not found

Activate Windows
Go to Settings to activate Windows.

Postman interface showing the Console output for the "allposttest" test. The output shows the test results and the status of the "id found" variable:

```
1
1
"sunt aut facere repellat provident occaecati excepturi optio reprehenderit"
"id found" true
> GET https://jsonplaceholder.typicode.com/posts/1000 404 888 ms </>
> GET https://jsonplaceholder.typicode.com/posts/ 200 70 ms
undefined
undefined
undefined
undefined
"id found" false
```

Activate Windows
Go to Settings to activate Windows.

Creating post

The image displays two screenshots of the Postman application interface, illustrating the process of creating and testing a POST request.

Top Screenshot: Pre-request Script

- URL:** `{{baseUrl}}/posts`
- Method:** `POST`
- Pre-request Script:**

```
1 const requestPayload = {
2   "userId": 1,
3   "id": 101,
4   "title": "cosica",
5   "body": "empty"
6 };
7
8 pm.sendRequest({
9   url: "https://jsonplaceholder.typicode.com/posts",
10  method: "POST",
11  header: {
12    "Content-Type": "application/json"
13  },
14  body: {
15    mode: "raw",
16    raw: JSON.stringify(requestPayload)
17  }
18 }, function(err, response){
19   if(response && response.code === 201){
20     const createPost = response.json();
21     pm.variables.set("postCreatedUserId", createPost.userId);
22   }
23 });
```

Bottom Screenshot: Tests

- Tests:**

```
10 console.log(userId);
11 console.log(id);
12 console.log(title);
13 console.log(body);
14
15 pm.test("User id should not be empty", function(){
16   pm.expect(userId).to.be.not.null;
17 });
18
19 pm.test("Id should not be empty", function(){
20   pm.expect(id).to.be.not.null;
21 });
22
23 pm.test("title should not be empty", function(){
24   pm.expect(title).to.be.not.empty;
25 });
26
27 pm.test("body should not be empty", function(){
28   pm.expect(body).to.be.not.empty;
29 });
30
```

Home Workspaces API Network Explore Search Postman Invite Upgrade

Assignment New Import

JsonPlaceholderTest / addpost_test

POST {{baseUri}}/posts Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

Body Cookies Headers (25) Test Results (5/5) 201 Created 318 ms 1.25 KB Save as example

All Passed Skipped Failed

PASS Successful POST request

PASS User id should not be empty

PASS Id should not be empty

PASS title should not be empty

PASS body should not be empty

Activate Windows
Go to Settings to activate Windows.

Online Find and replace Console Postbot Runner Start Proxy Cookies Trash

Type here to search

Home Workspaces API Network Explore Search Postman Invite Upgrade

Assignment New Import

JsonPlaceholderTest / addpost_test

POST {{baseUri}}/posts Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

Body Cookies Headers (25) Test Results (5/5) 201 Created 318 ms 1.25 KB Save as example

All Passed Skipped Failed

PASS Successful POST request

All Logs Clear

POST https://jsonplaceholder.typicode.com/posts 201 872 ms

POST https://jsonplaceholder.typicode.com/posts 201 319 ms

1

101

"comics"

"empty"

Activate Windows
Go to Settings to activate Windows.