

Postman interface showing a collection named "jsonplaceholder" and a test suite "jsonplaceholder\_Tests". The selected test is "GET AllList\_Test". The request is a GET to "https://jsonplaceholder.typicode.com/posts". The Pre-request Script tab is active, showing JavaScript code that sets environment variables based on the response.

```
1 pm.sendRequest({
2   url: "https://jsonplaceholder.typicode.com/posts",
3   method: "GET",
4 }, function(err, response){
5   pm.variables.set("userFound", false);
6   if(response){
7     const user = response.json();
8     pm.variables.set("userId", user[0].userId);
9     pm.variables.set("userTitle", user[0].title);
10    pm.variables.set("userFound", true);
11  }
12 });
```

Response status: 200 OK, 72 ms, 27.98 KB. The response body is visible at the bottom.

Postman interface showing the "Tests" tab for the "GET AllList\_Test" test. The test script is active, showing JavaScript code that checks if the user ID and title are not empty.

```
6 const userId = pm.variables.get("userId");
7 const userTitle = pm.variables.get("userTitle");
8 const userFound = pm.variables.get("userFound");
9 console.log("userFound", userFound);
10
11 if(userFound)
12 {
13   pm.test("user Id should not be empty", function(){
14     pm.expect(userId).to.exist;
15   });
16   pm.test("Title should not be empty", function(){
17     pm.expect(userTitle).not.to.be.empty;
18   });
19 }
```

Test Results (3/3) are shown at the bottom, indicating all tests passed.

Postman interface showing the "Tests" tab for the "GET AllList\_Test" test. The test script is active, showing JavaScript code that checks if the user ID and title are not empty.

```
6 const userId = pm.variables.get("userId");
7 const userTitle = pm.variables.get("userTitle");
8 const userFound = pm.variables.get("userFound");
9 console.log("userFound", userFound);
10
11 if(userFound)
12 {
13   pm.test("user Id should not be empty", function(){
14     pm.expect(userId).to.exist;
15   });
16   pm.test("Title should not be empty", function(){
17     pm.expect(userTitle).not.to.be.empty;
18   });
19 }
```

Test Results (3/3) are shown at the bottom, indicating all tests passed.

Postman interface showing a REST client setup for a GET request to `{{baseurl}}/posts/200`. The Pre-request Script tab is active, containing the following JavaScript code:

```
1 pm.sendRequest({
2   url: "https://jsonplaceholder.typicode.com/posts/1",
3   method: "GET",
4 }, function(err, response){
5   if(response){
6     if(response.code === 404)
7     {
8       console.log("user not found. setting default values");
9       pm.variables.set("userNotFound", true);
10    }
11    else
12    {
13      const user = response.json();
```

The right sidebar shows a list of snippets: "Get an environment variable", "Get a global variable", "Get a variable", and "Get a collection variable". The status bar at the bottom indicates a 404 Not Found response with a status of 404, a time of 872 ms, and a size of 1.06 KB.

Postman interface showing the Test Results tab for the GET request. The Test Results section displays the following test results:

Test	Result
Response Empty	PASS
userId should not be empty	PASS
Title should not be empty	PASS

The right sidebar shows a list of snippets: "Get an environment variable", "Get a global variable", "Get a variable", and "Get a collection variable". The status bar at the bottom indicates a 404 Not Found response with a status of 404, a time of 872 ms, and a size of 1.06 KB.

Postman interface showing a REST client setup for a GET request to `{{baseurl}}/posts/200`. The Tests tab is active, containing the following JavaScript code:

```
5 const userNotFound = pm.variables.get("userNotFound");
6 if(userNotFound)
7 {
8   console.log("user not found, handling errors");
9   pm.variables.set("userId", "1");
10  pm.variables.set("userTitle", "my title");
11 }
12
13 const userId = pm.variables.get("userId");
14 const userTitle = pm.variables.get("userTitle");
15 console.log("userId:", userId);
```

The right sidebar shows a list of snippets: "Get an environment variable", "Get a global variable", "Get a variable", and "Get a collection variable". The status bar at the bottom indicates a 404 Not Found response with a status of 404, a time of 872 ms, and a size of 1.06 KB.

Postman interface showing a REST client setup for a POST request to `https://jsonplaceholder.typicode.com/posts`. The request is configured with the following body (JSON):

```
1 const userdata = {
2   "userId": 10,
3   "title": "holiday",
4   "body": "today"
5 };
6 pm.sendRequest({
7   url: "https://jsonplaceholder.typicode.com/posts",
8   method: "POST",
9   header: {
10    "Content-Type": "application/json"
11  },
12  body: {
13    type: "json"
14  }
15 });
```

The console shows the response:

```
{
  "createduserId": 10,
  "createdusertitle": "holiday",
  "createduserbody": "today"
}
```

Response status: 201 Created, 836 ms, 1.26 KB.

Postman interface showing the same REST client setup, but with the **Tests** tab selected. The test script is as follows:

```
8 const createduserId = pm.variables.get("createduserId");
9 const createdusertitle = pm.variables.get("createdusertitle");
10 const createduserbody = pm.variables.get("createduserbody");
11 console.log("createduserId:", createduserId);
12 console.log("createdusertitle:", createdusertitle);
13 console.log("createduserbody:", createduserbody);
14
15 pm.test("User ID should be present", function() {
16   pm.expect(createduserId).to.exist;
17 });
```

The console shows the test results:

```
1 PASS Successful POST request
2 PASS Response time is less than 200ms
3 PASS User ID should be present
```

Response status: 201 Created, 836 ms, 1.26 KB.

Assignment New Import GET AllList\_Test GET Getone\_notfound\_Test POST create\_Test PUT update\_Test DEL delete\_Test jsonplaceholderEnv

jsonplaceholder\_Tests / update\_Test

PUT {{baseurl}}/posts/{{usno}}

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

```
1 const userdata = {
2   "userId": 10,
3   "title": "holiday",
4   "body": "today"
5 };
6
7 pm.sendRequest({
8   url: "https://jsonplaceholder.typicode.com/posts/1",
9   method: "PUT",
10  header: {
11    "Content-Type": "application/json"
12  },
13 });
```

Pre-request scripts are written in JavaScript, and are run before the request is sent. Learn more about [pre-request scripts](#)

Snippets

- Get an environment variable
- Get a global variable
- Get a variable
- Get a collection variable

Body Cookies Headers (24) Test Results (4/4) 200 OK 320 ms 1.16 KB Save as example

Online Find and replace Console 5 Errors All Logs Clear

PUT https://jsonplaceholder.typicode.com/posts/1 200 811 ms </>

PUT https://jsonplaceholder.typicode.com/posts/1 200 320 ms

"updateduserId": 10

"updatedusertitle": "holiday"

"updateduserbody": "today"

Activate Windows  
Go to Settings to activate Windows.

Assignment New Import GET AllList\_Test GET Getone\_notfound\_Test POST create\_Test PUT update\_Test DEL delete\_Test jsonplaceholderEnv

jsonplaceholder\_Tests / update\_Test

PUT {{baseurl}}/posts/{{usno}}

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

```
6 const updateduserId = pm.variables.get("updateduserId");
7 const updatedusertitle = pm.variables.get("updatedusertitle");
8 const updateduserbody = pm.variables.get("updateduserbody");
9 console.log("updateduserId:", updateduserId);
10 console.log("updatedusertitle:", updatedusertitle);
11 console.log("updateduserbody:", updateduserbody);
12
13 pm.test("User ID should be present", function() {
14   pm.expect(updateduserId).to.exist;
15 });
```

Test scripts are written in JavaScript, and are run after the response is received. Learn more about [tests scripts](#)

Snippets

- Get an environment variable
- Get a global variable

Body Cookies Headers (24) Test Results (4/4) 200 OK 320 ms 1.16 KB Save as example

All Passed Skipped Failed

- PASS Your test name
- PASS User ID should be present
- PASS Title should be present
- PASS Body should be present

Online Find and replace Console Postbot Runner Start Proxy Cookies Trash 11:47 09-12-2023

Type here to search

Postman interface showing the 'delete\_Test' endpoint configuration. The endpoint is a DELETE request to `{{baseurl}}/posts/{{usno}}`. The request body is empty. The 'Pre-request Script' tab is active, showing a JavaScript function that logs a success message if the response status is 200, or an error message otherwise.

```
1 pm.sendRequest({
2   url: "https://jsonplaceholder.typicode.com/posts/1",
3   method: "DELETE",
4   header: {
5     "Content-Type": "application/json"
6   }
7 }, function(err, response) {
8   if (response && response.code === 200) {
9     console.log("User deleted successfully");
10  } else {
11    console.error("Error deleting user:", err);
12  }
13 });
```

The console shows the following output:

```
DELETE https://jsonplaceholder.typicode.com/posts/1
"User deleted successfully"
DELETE https://jsonplaceholder.typicode.com/posts/1
```

Postman interface showing the 'delete\_Test' endpoint configuration. The endpoint is a DELETE request to `{{baseurl}}/posts/{{usno}}`. The request body is empty. The 'Tests' tab is active, showing two JavaScript test functions: one to check if the response body is empty, and another to check if the status code is 200.

```
1 pm.test("Response Empty", function () {
2   pm.response.to.have.body("{}");
3 });
4
5 pm.test("Status code is 200", function () {
6   pm.response.to.have.status(200);
7 });
```

The console shows the following output:

```
DELETE https://jsonplaceholder.typicode.com/posts/1
"User deleted successfully"
DELETE https://jsonplaceholder.typicode.com/posts/1
```

The 'Test Results' tab shows the following results:

Test Name	Result
Response Empty	PASS
Status code is 200	PASS