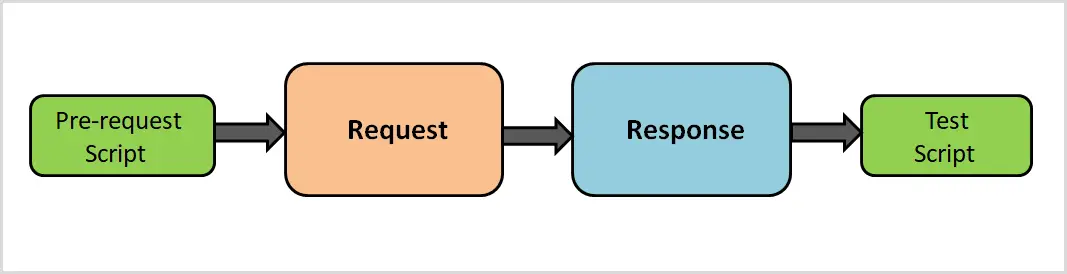
# **API Test Cases in Postman using JavaScript**

In the previous articles on [Postman Tutorial](https://www.softwaretestingmaterial.com/postman-tutorial/), we have covered “[**How To Run Collections Using Newman**](https://www.softwaretestingmaterial.com/postman-tutorial/#How_To_Run_Collections_using_Newman)“

In this “API Test Cases in Postman using JavaScript” article, I will be demonstrating as to how you can implement this concept and get a tight grip over this.

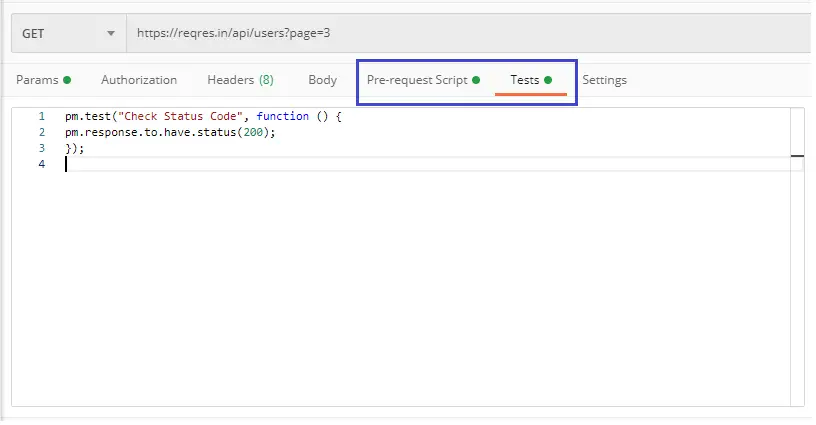
With Postman, we can add scripts to our request and write tests.

Code added under the **Pre-request Script tab** will execute before the request is sent, and code added under the **Tests tab** will execute after the response is received.



We use **pre-request scripts** for all the things we need to do before the execution of the request such as setting variables, clearing the variables, or set the environment variable in the environment.

We write **Test scripts** to validate API’s error handling by sending a request with incomplete data. Through Test scripts, we can use dynamic variables and can carry out test assertions on response data.



Postman has a PM API (known as the pm.\*API) which is the powerful way to write tests in Test tab.

### **pm.test():**

The pm.test() function is used to write test specifications. It accepts 2 parameters, the name of the test (as a string) and a function to return a boolean value. It can be used only in the Tests tab after the primary Postman request has been sent.

**Example:**

pm.test("response should be okay to process", function () {

pm.response.to.not.be.error;

pm.response.to.have.jsonBody("");

pm.response.to.not.have.jsonBody("error");

});

### **pm.expect():**

The pm.expect() assertion function makes it easy to write readable tests, and we can deal with assertions of data from a response or variables.

**Example:**

pm.test("environment to be production", function () {

pm.expect(pm.environment.get("env")).to.equal("production");

});

### **pm.response.to.be.\* :**

The pm.resonse.to.be object provides tests for response status types and body variations.

**Example:**

pm.test("response is ok", function () {

pm.response.to.have.status(200);

});

### **Test Cases:**

We need to add any of the assertions inside a pm.test callback.

**Example:**

pm.test("Your test name", function () {

var jsonData = pm.response.json();

pm.expect(jsonData.value).to.eql(100);

});

### **1) Status Code:**

#### 1.1 Check if the status code is 200:

pm.test("Status code is 200", function () {

pm.response.to.have.status(200);

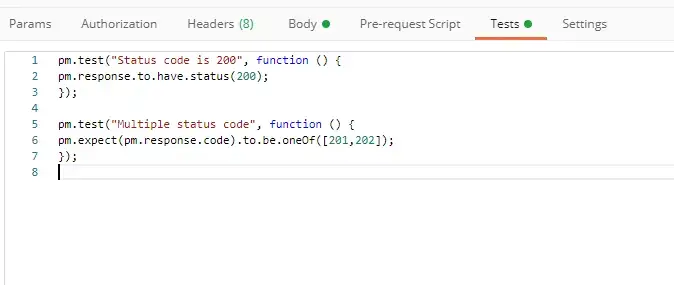
});

#### 1.2 Checking multiple status codes:

pm.test("Multiple Status Codes”, function() {

pm.expect(pm.response.code).to.be.oneOf([201,202]);

});



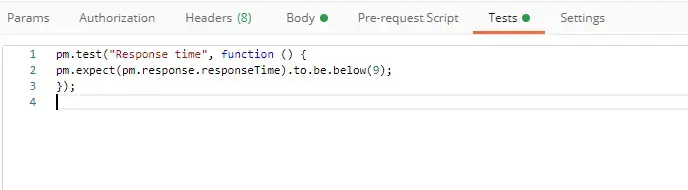
### **2) Response time:**

#### 2.1 Response time below 100ms:

pm.test(“response time”, function() {

pm.expect(pm.response.responseTime).to.be.below(9);

});



### **3) Headers:**

#### 3.1 Header exists:

pm.test(“header exists”, function() {

pm.response.to.have.header(Content-Type);

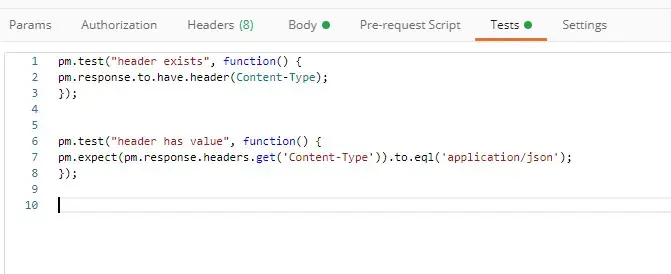
});

#### 3.2 Header has value:

pm.test(“header has value”, function() {

pm.expect(pm.response.headers.get(‘Content-Type’)).to.eql(‘application/json’);

});



### **4) Cookies:**

#### 4.1 Cookie exists:

pm.test(“Cookie exists”, function() {

pm.expect(pm.cookies.has('sessionId')).to.be.true;

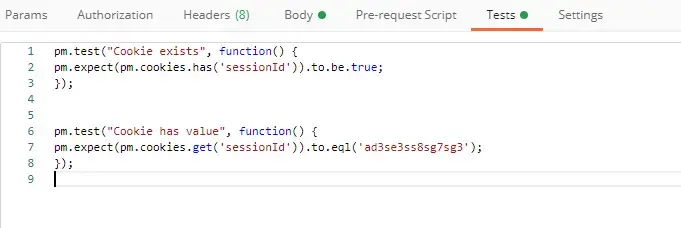
});

#### 4.2 Cookie has value:

pm.test(“Cookie has value”, function() {

pm.expect(pm.cookies.get('sessionId')).to.eql(’ad3se3ss8sg7sg3');

});



### **5) Body (Any content-type / HTML responses):**

#### 5.1 Exact body match:

pm.test(“Match body”, function() {

pm.response.to.have.body("OK");

pm.response.to.have.body('{"success"=true}');

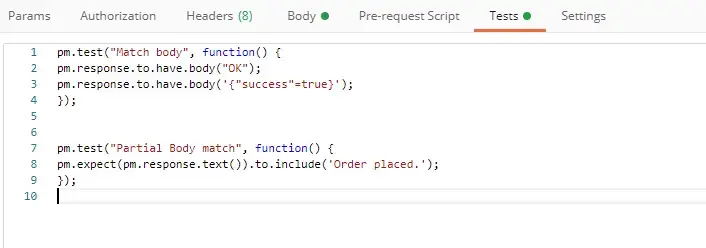
});

#### 5.2 Partial body match/body contains:

pm.test(“Partial Body match”, function() {

pm.expect(pm.response.text()).to.include('Order placed.');

});



### **6) Body (JSON responses):**

**Parse body (need for all assertions)**

const response = pm.response.json();

#### 6.1 Simple value check:

pm.test(“Check value”, function() {

const response = pm.response.json();

pm.expect(response.age).to.eql(30);

pm.expect(response.name).to.eql('John);

});

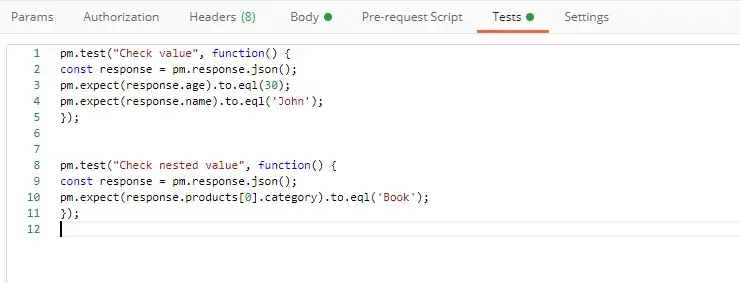
#### 6.2 Nested value check:

pm.test(“Check nested value”, function() {

const response = pm.response.json();

pm.expect(response.products[0].category).to.eql('Book');

});



### **7) Body (XML responses):**

#### Convert XML body to JSON (need for all assertions):

const response = xml2Json(responseBody);

#### 7.1 Simple value check:

pm.test(“Check value”, function() {

const response = xml2Json(responseBody);

pm.expect(response.age).to.eql(30);

pm.expect(response.name).to.eql('John);

});

#### 7.2 Nested value check:

pm.test(“Check value”, function() {

const response = xml2Json(responseBody);

pm.expect(response.products.0.category).to.eql('Book');

});

