Postman Cheatsheet

This guide refers to the Postman App. Postman Cheat Sheet is based on the official Postman documentation and own experience.

For a detailed documentation on each feature, check out https://www.getpostman.com/docs.

Variables

All variables can be manually set using the Postman GUI and are scoped.

The code snippets can be used for working with variables in scripts (pre-request, tests).

Learn more about the different variables scopes in this tutorial.

Getting variables in the Request Builder

Depending on the closest scope:

Syntax: `{{myVariable}}`

Examples:

Request URL: `http://{{domain}}/users/{{userId}}`

Headers (key:value): `X-{{myHeaderName}}:foo`

Request body: `{"id": "{{userId}}", "name": "John Doe"}`

Global variables

General purpose variables, ideal for quick results and prototyping.

Please consider using one of the more specialized variables below. Delete variables once they are no longer needed.

When to use:

- passing data to other requests

Setting

```
pm.globals.set('myVariable', MY_VALUE);
```

Getting

```
pm.globals.get('myVariable');
```

Alternatively, depending on the scope:

```
pm.variables.get('myVariable');
```

Removing

Remove one variable

```
pm.globals.unset('myVariable');
```

Remove ALL global variables (rather unusual)

```
pm.globals.clear();
```

Collection variables

When to use:

- good alternative to global variables or environment variables
- for URLs / authentication credentials if only one environment exists

Setting

```
pm.collectionVariables.set('myVariable', MY_VALUE);
```

Getting

```
pm.collectionVariables.get('myVariable');
```

Removing

```
pm.collectionVariables.unset('myVariable');
```

Environment variables

Environment variables are tied to the selected environment. Good alternative to global variables as they have a narrower scope.

When to use:

- storing environment specific information
- URLs, authentication credentials
- passing data to other requests

Setting

```
pm.environment.set('myVariable', MY_VALUE);
```

Getting

```
pm.environment.get('myVariable');
```

Depending on the closest scope:

```
pm.variables.get('myVariable');
```

Removing

Remove one variable

```
pm.environment.unset("myVariable");
```

Remove ALL environment variables

```
pm.environment.clear();
```

Examples:

```
pm.environment.set('name', 'John Doe');
console.log(pm.environment.get('name'));
console.log(pm.variables.get('name'));
```

** Detecting the environment name **

If you need to know inside scripts which environment is currently active (locahost, production, ...) you can use the name property:

```
pm.environment.name
```

Data variables

Exist only during the execution of an iteration (created by the Collection Runner or Newman).

When to use:

- when multiple data-sets are needed

Setting

Can only be set from a CSV or a JSON file.

Getting

```
pm.iterationData.get('myVariable);
```

Depending on the closest scope:

```
pm.variables.get('myVariable');
```

Removing

Can only be removed from within the CSV or JSON file.

Local variables

Local variables are only available withing the request that has set them or when using Newman / Collection runner during the entire execution.

When to use:

- whenever you would like to override all other variable scopes—for whatever reason. Not sure though then this is needed.

Setting

```
pm.variables.set('myVariable', MY_VALUE);
```

Getting

```
pm.variables.get('myVariable', MY_VALUE);
```

Removing

Local variables are automatically removed once the tests have been executed.

Dynamic variables

All dynamic variables can be combined with strings, in order to generate dynamic / unique data.

Example JSON body:

```
{"name": "John Doe", "email": "john.doe.{{$timestamp}}@example.com"}
```

If you want to use dynamic variables in scripts, you can use the *replaceIn* starting with Postman v7.6.0.

```
pm.variables.replaceIn('{{$randomFirstName}}'); // returns a String
```

For more details please see the section dedicated to Dynamic variables

Logging / Debugging variables

Open Postman Console and use *console.log* in your test or pre-request script.

Example:

```
var myVar = pm.globals.get("myVar");
console.log(myVar);
```

Assertions

Note: You 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);
});
```

Status code

Check if status code is 200:

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

Checking multiple status codes:

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

Response time

Response time below 100ms:

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

Headers

Header exists:

```
pm.response.to.have.header('X-Cache');
```

Header has value:

```
pm.expect(pm.response.headers.get('X-Cache')).to.eql('HIT');
```

Cookies

Cookie exists:

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

Cookie has value:

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

Body

Any content type / HTML responses

Exact body match:

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

Partial body match / body contains:

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

JSON responses

Parse body (need for all assertions):

```
const response = pm.response.json();
```

Simple value check:

```
pm.expect(response.age).to.eql(30);
pm.expect(response.name).to.eql('John');
```

Nested value check:

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

XML responses

Convert XML body to JSON:

```
const response = xml2Json(responseBody);
```

Note: see assertions for JSON responses.

Skipping tests

You can use *pm.test.skip* to skip a test. Skipped tests will be displayed in reports.

Simple example

```
pm.test.skip("Status code is 200", () => {
    pm.response.to.have.status(200);
});
```

Conditional skip

```
const shouldBeSkipped = true; // some condition

(shouldBeSkipped ? pm.test.skip : pm.test)("Status code is 200", () => {
    pm.response.to.have.status(200);
});
```

Failing tests

You can fail a test from the scripts without writing an assertion:

```
pm.expect.fail('This failed because ...');
```

Postman Sandbox

pm

this is the object containing the script that is running, can access variables and has access to a read-only copy of the request or response.

pm.sendRequest

Allows to send **simple HTTP(S) GET requests** from tests and pre-request scripts. Example:

```
pm.sendRequest('https://httpbin.org/get', (error, response) => {
    if (error) throw new Error(error);
    console.log(response.json());
});
```

Full-option **HTTP POST request with JSON body**:

```
const payload = { name: 'John', age: 29};

const options = {
    method: 'POST',
    url: 'https://httpbin.org/post',
    header: 'X-Foo:foo',
    body: {
        mode: 'raw',
        raw: JSON.stringify(payload)
    }
};
pm.sendRequest(options, (error, response) => {
    if (error) throw new Error(error);
    console.log(response.json());
});
```

Form-data POST request (Postman will add the multipart/form-data header):

```
const options = {
    'method': 'POST',
    'url': 'https://httpbin.org/post',
    'body': {
            'mode': 'formdata',
            'formdata': [
                {'key':'foo', 'value':'bar'},
                {'key':'bar', 'value':'foo'}
            ]
    }
};
pm.sendRequest(options, (error, response) => {
    if (error) throw new Error(error);
    console.log(response.json());
});
```

Sending a file with form-data POST request

Due to security precautions, it is not possible to upload a file from a script using pm.sendRequest. You cannot read or write files from scripts.

Postman Echo

Helper API for testing requests. Read more at: https://docs.postman-echo.com.

Get Current UTC time in pre-request script

```
pm.sendRequest('https://postman-echo.com/time/now', function (err, res) {
    if (err) { console.log(err); }
    else {
        var currentTime = res.stream.toString();
        console.log(currentTime);
        pm.environment.set("currentTime", currentTime);
    }
});
```

Workflows

Only work with automated collection runs such as with the Collection Runner or Newman. It will NOT have any effect when using inside the Postman App.

Additionaly it is important to note that this will only affect the next request being executed. Even if you put this inside the pre-request script, it will NOT skip the current request.

Set which will be the next request to be executed

```
postman.setNextRequest(“Request name");
```

Stop executing requests / stop the collection run

```
postman.setNextRequest(null);
```