POSTMAN

Postman is one of the most popular software testing tools which is used for API testing. With the help of this tool, developers can easily create, test, share, and document APIs.

* Postman is a standalone software testing API (Application Programming Interface) platform to build, test, design, modify, and document APIs. It is a simple Graphic User Interface for sending and viewing HTTP requests and responses.
* While using Postman, for testing purposes, one doesn't need to write any HTTP client network code. Instead, we build test suites called collections and let Postman interact with the API.
* Postman can make various types of HTTP requests like GET, POST, PUT, PATCH.

# Benefits of postman

* After installing Postman, it can be accessed from any device using user account.
* We can add checkpoints and verifications
* Automation testing is also possible with postman
* The design of multiple environments results in less replication of tests as one can use the same collection but for a different setting.

# Creating the first request

<https://google.com/articles/articlename>

articlename🡪 URN ( **U**niform **R**esource **N**ame)

google.com/articles/articlename🡪 URI (**U**niform **R**esource **I**dentifier)

<https://google.com/articles/articlename-->> URL (**U**niform **R**esource **L**ocator)

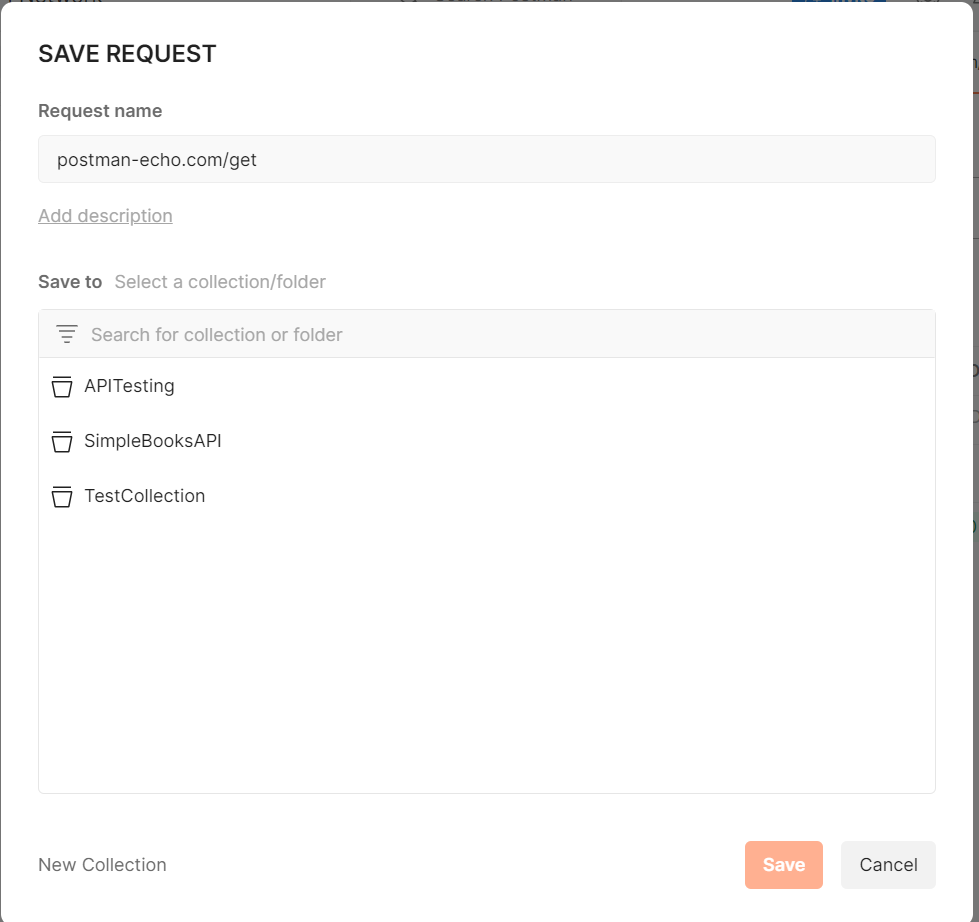
1. Create a collection
2. Add a request
3. Try with the below request URL:

postman-echo.com/get

## Collections

Collection in Postman means a group of API requests. We can arrange these requests in a folder.

We can create a request and when we save Postman will ask for the creation of a new collection.



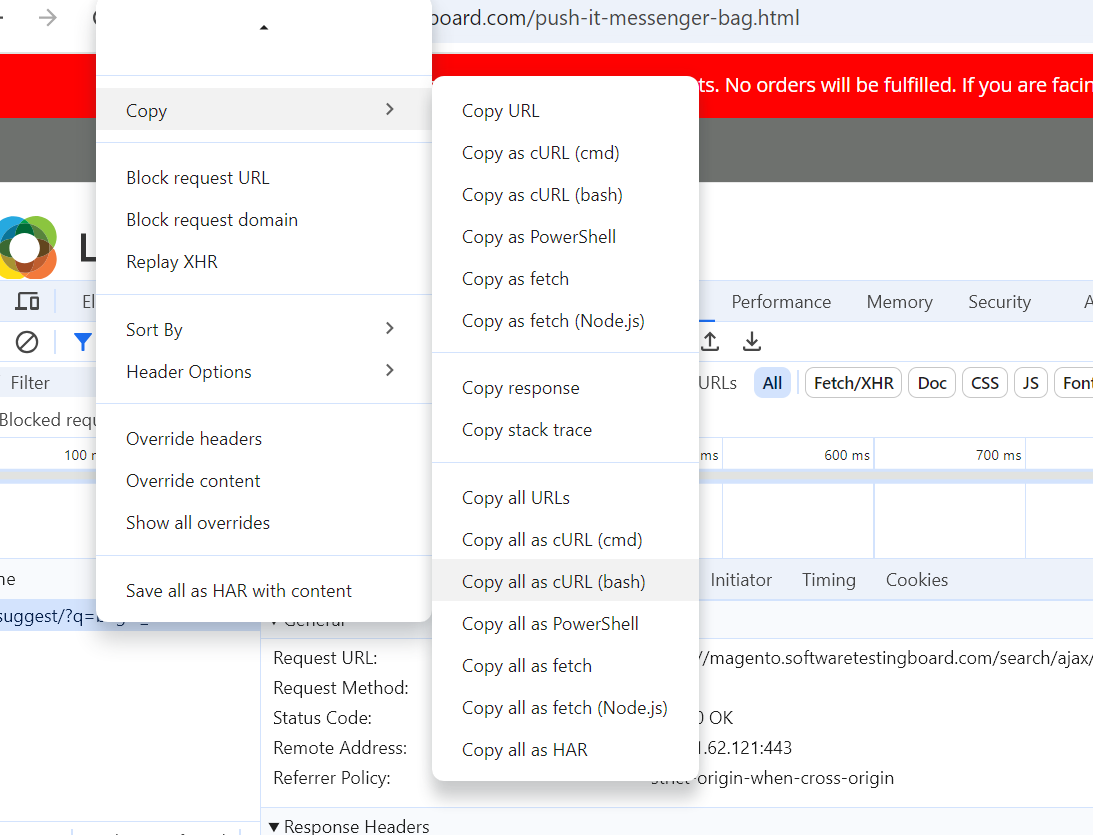
* We can change the Request name, or by default, the Request URL will be displayed as the Request name.
* We can also write the Request description as an optional step.
* We can either create a new collection or add the request in existing collection.
* Save the collection by clicking on the Save button.
* Now, after saving, the collection appears in the sidebar under the Collection tab.
* Within a collection, any number of requests can be added.

Creating GET, POST, and PUT requests in the collection

# How to import requests from browser to Postman

Sometimes we may want to get the request from the browser and check in Postman to modify and check the response.

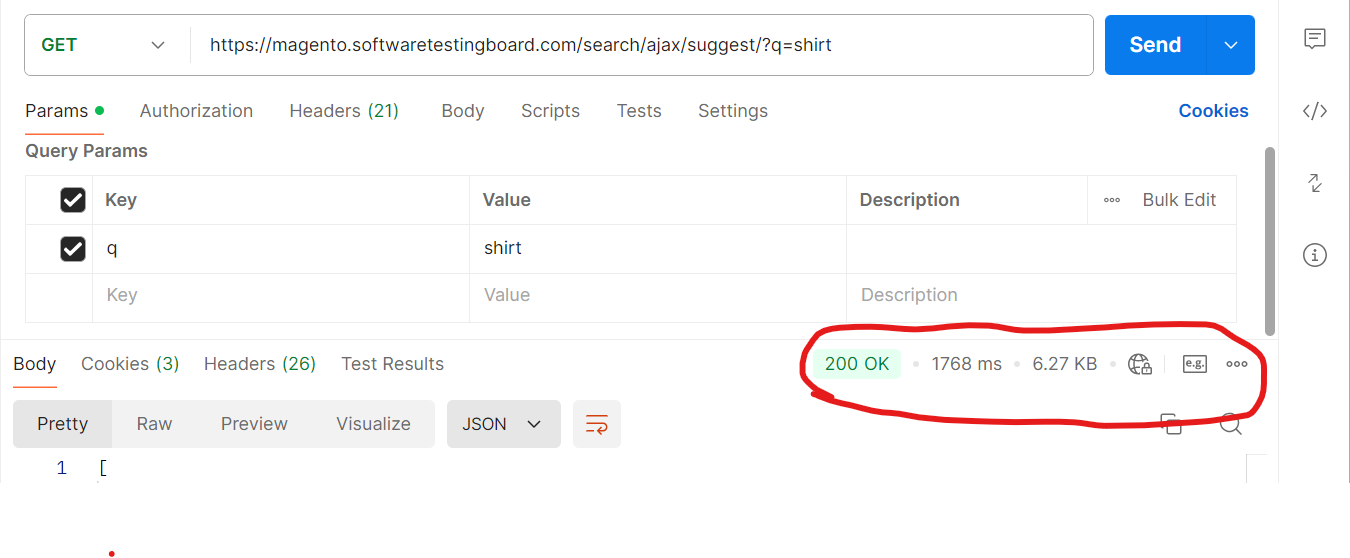
Postman allows us to import the request from the browser.



* 1. Copy the curl bash
  2. In postman, click on import
  3. Paste the curl
  4. Select a collection
  5. Save the request
  6. Now we can change the request data and verify the response

# Inspecting the HTTP responses

When we send a request, the response information appears in the bottom pane. This includes the response body, status code, time it took to send the request, and response size.



# Cookies Manager

A cookie is a packet of data that a computer receives and then sends back without changing or altering it.

To view the cookies associated with a request, we can check the cookies tab.

User can Add, modify or delete cookies by clicking on cookies under send button.

In the **Cookies** window, select the **Manage Cookies** tab to display a list of domains and the cookies associated with each one.

## Create Cookies

To add a new cookie for a domain, select **+ Add Cookie** under the domain. A pre-generated cookie string is created.

Postman supports the following attirbutes for cookies:

* **cookieName** = **cookieValue** - The name of the cookie and the value stored in it.
* **Domain** - The domain Postman will send the cookie to.
* **Path** - The URL path that the cookie is restricted to. If the path is /, the cookie will be sent to all requests in the specified domain.
* **HttpOnly** - If present, the cookie won't be accessible to the client-side scripts run on the page (for example, with document.cookie in JavaScript). The cookie will only be added to the cookie header in requests that are made. This field doesn't have an effect on Postman's behavior.
* **Secure** - If present, the cookie is only sent when the URL begins with https:// and won't be sent over an insecure connection.
* **Expires** - The time after which the cookie will expire and not be sent by Postman.

## Edit or Delete cookies

To update an existing cookie for a domain, select the cookie you want to edit, edit any attribute, then select **Save**.

To delete a domain and all cookies associated with it, select Close icon next to the domain. To delete an individual cookie, select Close icon next to the cookie.

# Postman Scripts

Postman is not only used for manual testing, but we can also do automation testing of API

* Before sending a request, a pre-request script will run and,
* After receiving a response, **test scripts** will run.

In Postman, we can write the tests, pass the data between the requests, and change the parameters. It allows adding test script and pre-request script to a collection, a folder, a request.

[Postman](https://www.javatpoint.com/postman) sandbox has been written in [Javascript](https://www.javatpoint.com/javascript-tutorial). It is an execution environment. Since the sandbox has written in Javascript, therefore, it will only receive a script written in Javascript.

## Execution Order

Response

Request

REQUEST  
Pre-request Script

Request Test Script

## For Collections

* Before every request in the collection, a pre-request script associated with a collection will run.
* Before every request in the folder, a pre-request script associated with a folder will run.
* After every request in the collection, a test script associated with a collection will run.
* After a request in the folder, a test script associated with a folder will run.

The order of execution will be

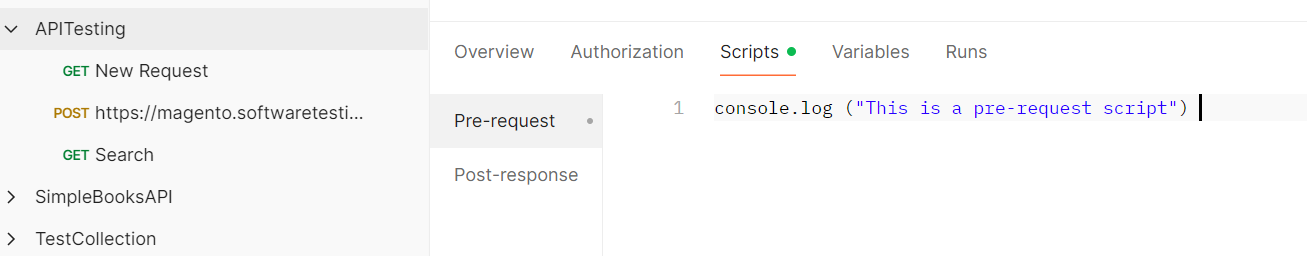
* Collection level script
* Folder level script
* Request level script

The same order will be applied for both pre-request and test scripts.

# Re-requisite script

Pre-request scripts are a piece of code that will run before the execution of a request. It runs in the [Postman](https://www.javatpoint.com/postman) sandbox

We can use the pre-request script for a pre-processing task like setting parameters, variable values, body data, and headers. We may also use pre-request scripts to debug the code, such as by logging output to the console.



# Post Test Scripts

Test scripts execute after the server receives the response.

Test script is used to test whether your API is working accordingly or not,

Test Scripts are also called Post-Request script.

## Testing the API

For testing the API, we can use pm.test() method.

This function writes the conditions of the test in the postman test sandbox.

This function has two arguments, first is the test name (as a string), and second is a function which returns a Boolean value.

For Example, to verify the status code, we can use:

pm.test("Status code is 200", ()=>

{

    pm.response.to.have.status(200);

});

--------------------------------------------------------------------------------

pm.test("Response", () => {

     // assert that the status code is 200

     pm.response.to.be.ok;

     pm.response.to.be.withBody;

});

# Postman variables

Postman variables work in the same way as that of the programming variables. These are the symbols capable of taking various values. You can store the values in variables and can use it throughout in requests, environments, collections, and scripts.

## Different types of variables

1. Global variables
2. Collection level variables
3. Environment variables
4. Local variable
5. Data variable

### Global variables

* Global variables are accessible in the entire workspace.
* To add a global variable, we can click on Environments 🡪 Globals
* Enter the Variable Name and Initial value.
* Use the variable using “{{“

Example: {{baseURL}}/users?page=1

### Collection level variable

* Collection-level variables apply to the entire collection.
* To add a collection variable, click on the collection
* Go to the variable tab
* Add variable name and value.

### Environment variables

Environment variables are applicable to environments. Environment is a set of variables that differentiate among the requests, for example, we can have an environment for testing, one for development, and another for production. We can change the value of the variables to pass the data between requests and tests.

**Creating Environment**

To create environment, we click on Environments and click on ‘+’ to add new environment.

Enter the environment name, variable name, and value. Save the details.

Environment variables apply to the environment selected.

### Local Variables:

In Postman, local variables are temporary variables that are accessed in your request scripts. Local variable values are scoped to a single request or collection run and are no longer available when the run is complete. Local variables are suitable if you need a value to override all other variable scopes but don't want the value to persist once execution has ended.

Ex:

//to set local variables

pm.variables.set(“url\_local”,” [www.google.com](http://www.google.com)”);

https://magento.softwaretestingboard.com /search/ajax/suggest/?q={{search}}

Here, {{search}}, can be taken from pre-script local variable.

pm.variables.**set**("search", "bags");

Same way, we can use **get** method to get the value of local variable.

pm.variables.get("variable\_key");

**Setting global and environment variable in pre-req script**

pm.global.set (“userif”, “2”);

Env Var:

pm.environment.set (“user-env”, “3”);

This will create environment variable for the environment selected

**Collection Variables**

pm.collectionvariables.set (“coll.val”, “3”);

**Removing Variables:**

Local Variables are created at run time and are deleted after request execution

Global and Environment variables are permanently created. To remove them after it is used, include the below line in Test script

pm.globals.unset(“<variable\_name>”);

To use the variables created in pre-req scripts, use the get method

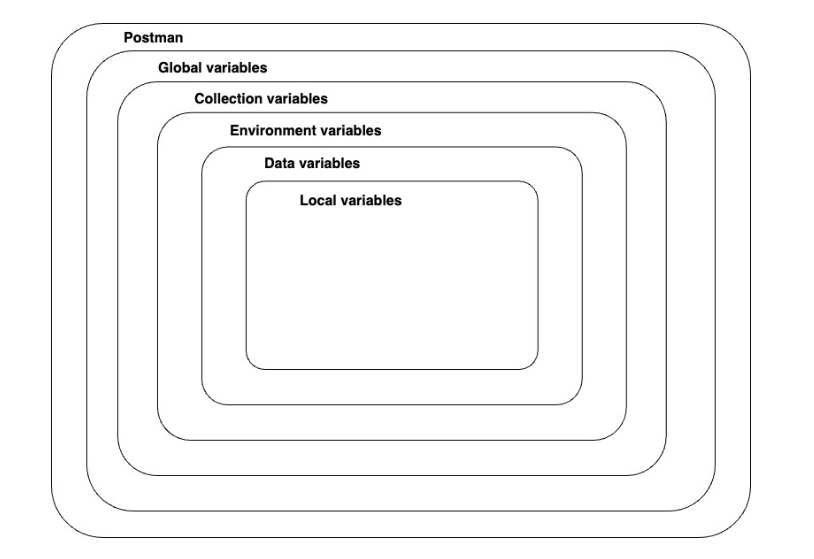
Ex:

Console.log (pm.globals.get(“<varname>”);

### Data Variables

Data variables: These types of variables are external and define the data sets while running collections with the Collection Runner. We can extract this from a CSV or a JSON file. They have current values that don’t persist after a request or collection executes.

## Variable Scope



If a variable with the same name is declared in two different scopes, the value stored in the variable with narrowest scope will be used. For example, if there is a global variable named username and a local variable named username, the local value will be used when the request runs.

## Initial and Current Values of variables

**Initial value** is a value that's set in the element (collection, environment, or globals) where the variable is defined. This value is synced to Postman's servers, and is shared with your team when you share that element. Setting an initial value can be useful when sharing elements with teammates.

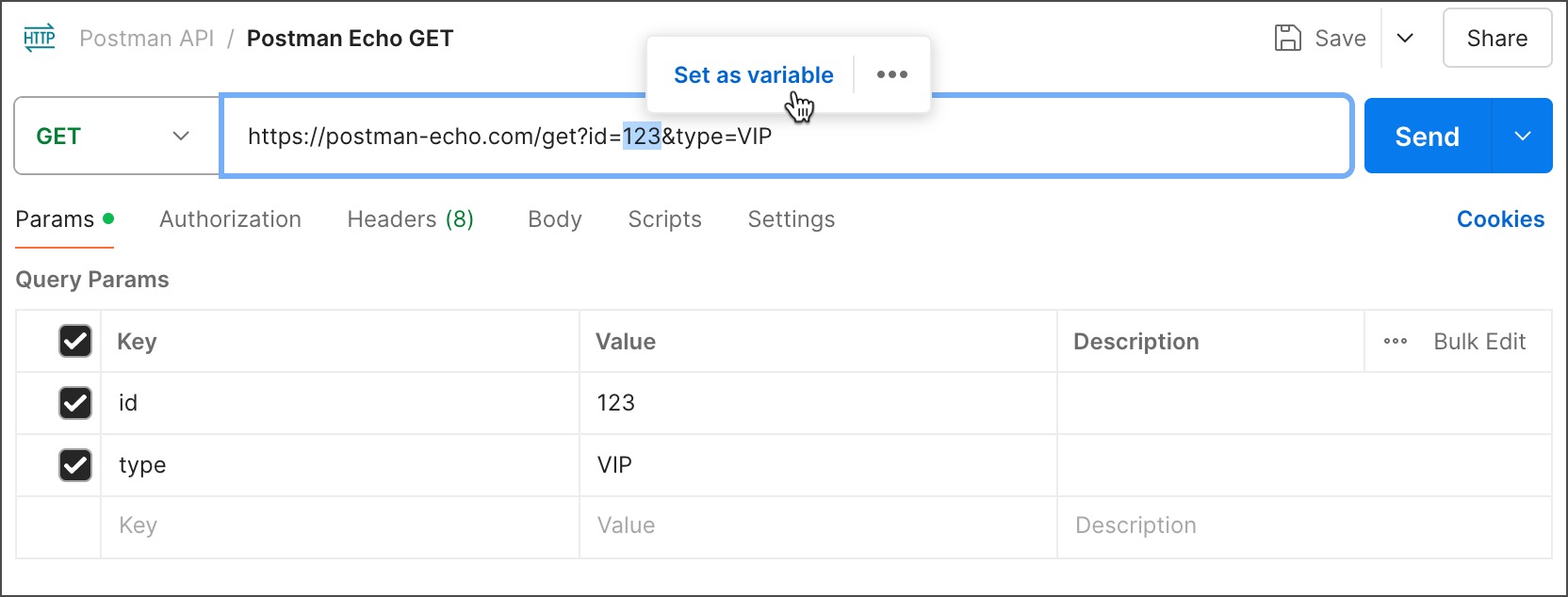
**Current value** is used when sending a request. These are local values, and aren't synced to Postman's servers.

## Variable Types

* **Default** type is automatically assigned to variables. This type is shown as plain text and doesn't have extra properties.
* **Secret** type masks the [initial and current values](https://learning.postman.com/docs/sending-requests/variables/variables/#initial-and-current-values) for all workspace members and can be used to prevent unintentional disclosure of sensitive data, including API secrets, passwords, tokens, and keys.

To define variables at any scope in the request builder, do the following:

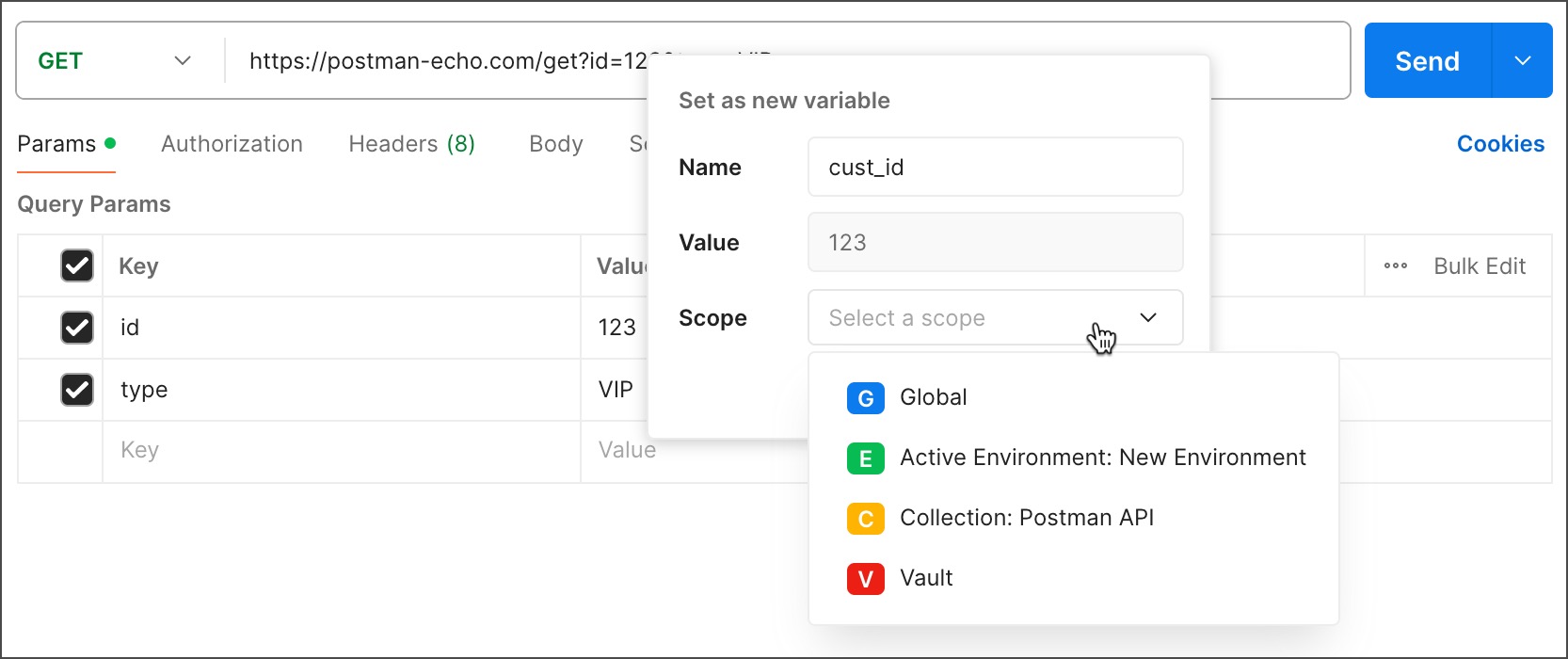
1. Select the data you need, for example in the address, parameters, headers, or body. Select **Set as variable**.



1. Select **Set as a new variable**.



1. Enter a **Name**, confirm the **Value** is correct, and select a scope. Select **Set variable**.

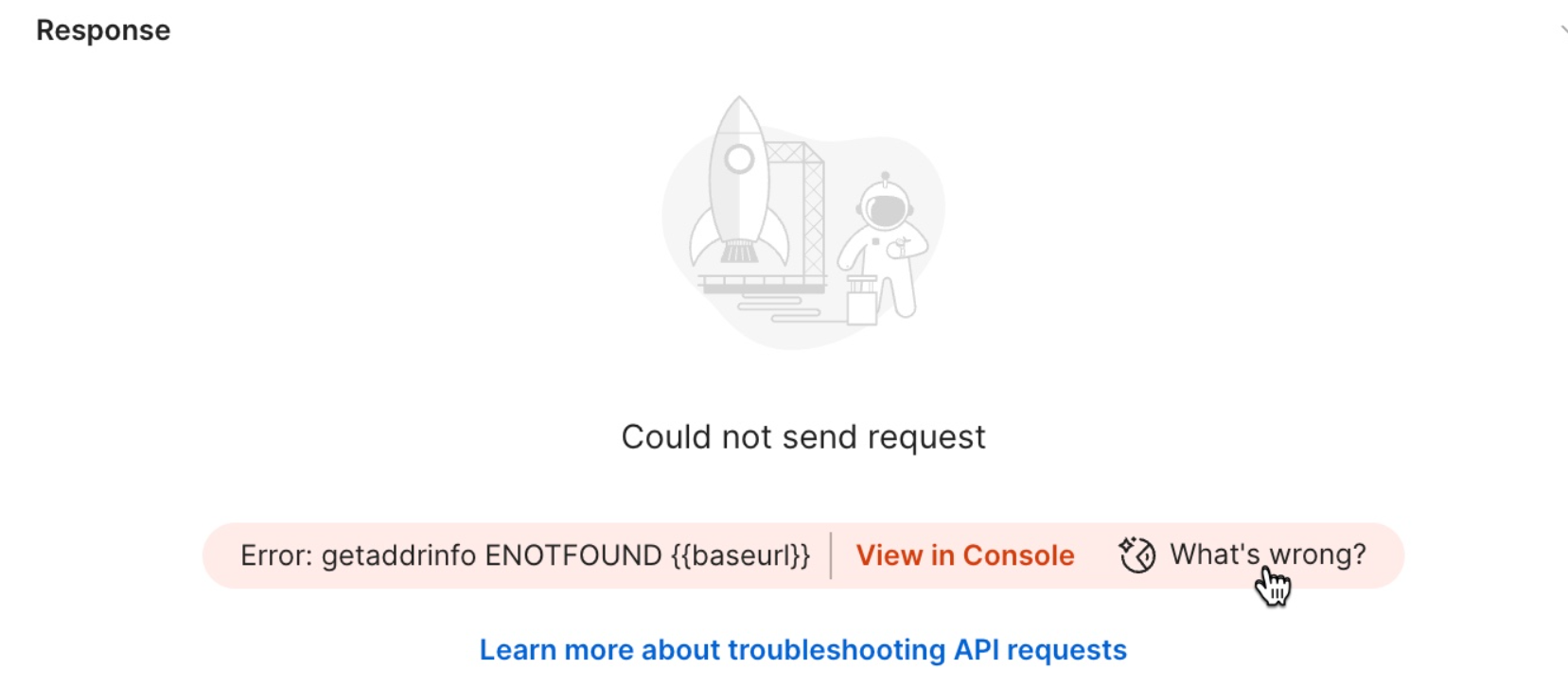


# Debugging in Console

If an API request isn't behaving as expected, there can be many possible reasons. You can ask Postbot to help you find out what the problem is, you can use the Postman Console to troubleshoot the request

## Debbuging using postbot

If you get an unexpected error when you send a request, you can ask Postbot for help. Select **What's wrong?** in the error message. Postbot will tell you about any problems it can identify, and it will offer possible solutions for fixing the issue.

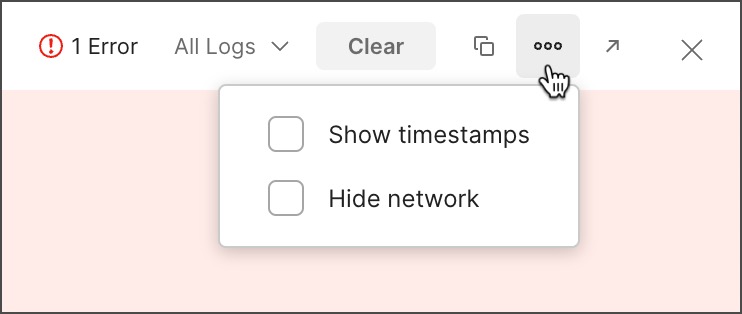


## Debugging in the console

Every request sent by Postman is logged in the Postman Console, so you can view the detail of what happened when you sent a request.

he Postman Console displays network information and the request and response headers and body for each request, together with any Console output messages coming from your scripts.

Filter by log message type under **All Logs**. Select the more actions icon More actions icon to turn timestamps and network information on or off.



The Console will log the last 5,000 messages and 24 hours by default. Select **Clear** to empty the list.

[**Using log statements**](https://learning.postman.com/docs/sending-requests/response-data/troubleshooting-api-requests/#using-log-statements)

Using log statements at appropriate locations in your post-response scripts can help you debug your requests. Postman accepts the following log statements:

* console.log()
* console.info()
* console.warn()
* console.error()
* console.clear()

