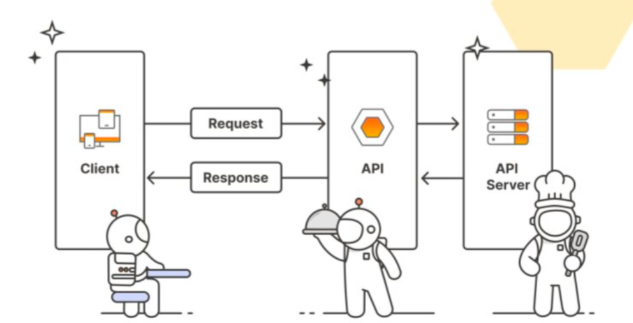
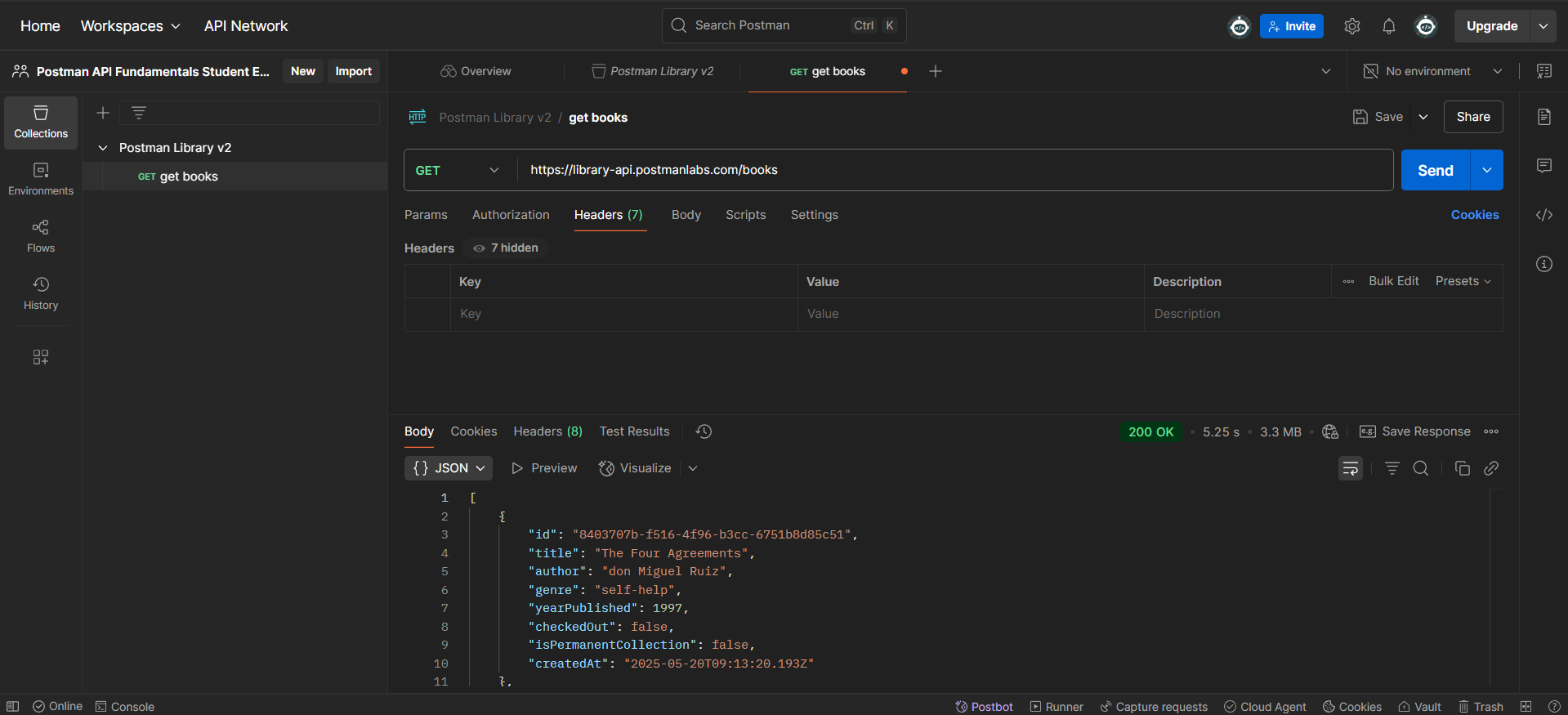
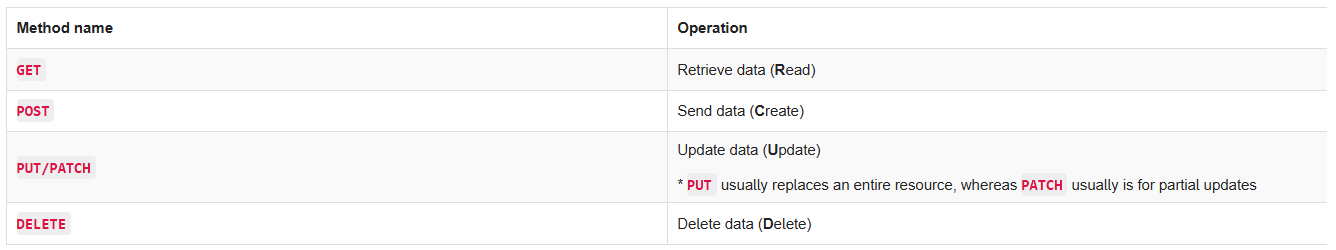
# **Postman API Fundamentals Student Expert certification** **API:** Application Programming Interfaces -

* You can think of APIs as being like a waiter at a restaurant, serving as a go-between for the customer and the kitchen.   
    
  A customer who wants soup doesn't go into the kitchen to cook. They don't even have to know how to make soup! They only have to know *how to ask* *the waiter* for soup*, expecting the waiter to* bring back soup*.*   
    
    
    
  Types of APIs  
  REST (Representational State Transfer)
* GraphQL
* WebSockets
* webhooks
* SOAP (Simple Object Access Protocol)
* gRPC (Google Remote Procedure Call)
* MQTT (MQ Telemetry Transport)

**REST APIs: Access** -  
 1. **Public APIs (aka Open APIs)**  
Consumed by anyone who discovers the API

**2. Private APIs**  
Consumed only within an organization and not made public

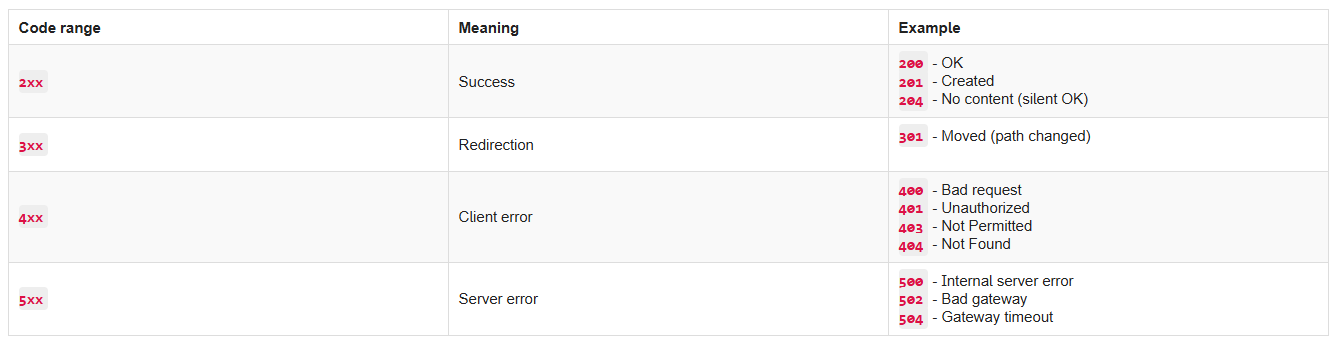
**3. Partner APIs**  
Consumed between one or more organizations that have an established relationship  
  
  
• REST API allows you to [**CRUD**](https://en.wikipedia.org/wiki/Create,_read,_update_and_delete) (**C**reate, **R**ead, **U**pdate, **D**elete)  
Click Workspace -> Create Collection ->click + or new or Collection -> rename according to you -> Create a new request by either clicking **Add a request or three dots** -> name u r request -> put request Link ->click send -> you will see a json file of list of books  
  


More requests: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Methods>A request URL has three parts: a **protocol**   
(such as **http://** or **https://**), **host** (location of the server), and **path** (route on the server). In REST APIs, the path often points to a reference entity, like "books".

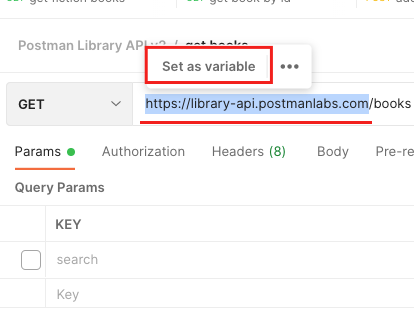
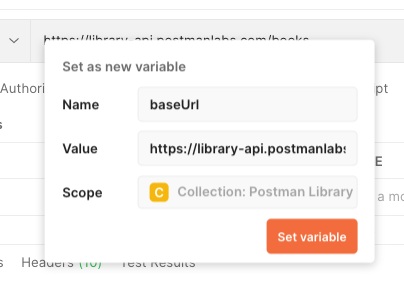
|  |  |  |
| --- | --- | --- |
| Protocol | Host | Path |
| **https://** | **library-api.postmanlabs.com** | **/books** |

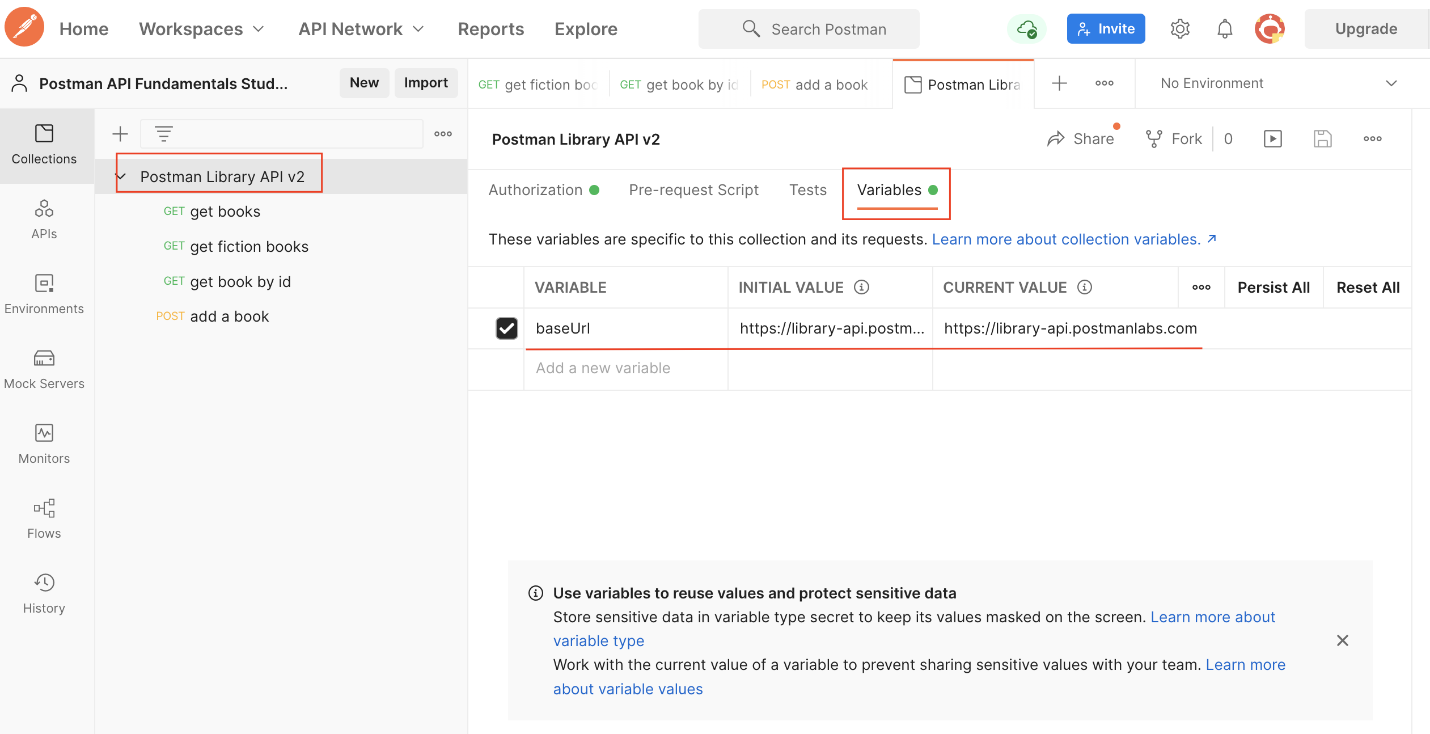
Paths and complete URLs are also sometimes called **API endpoints**.

### Response status codes

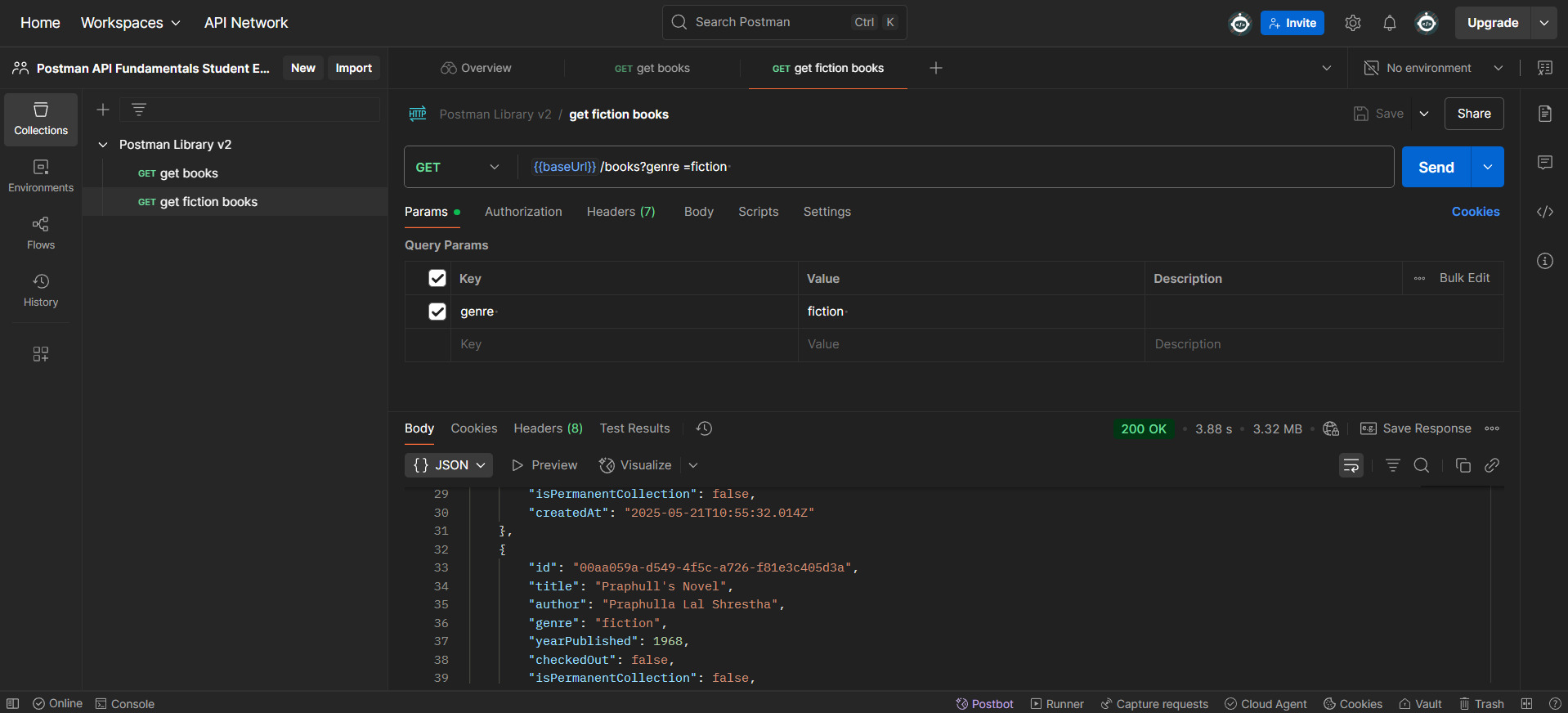
* any status code starting with a "2xx" (a "200-level response") represents a successful call.  
    
  

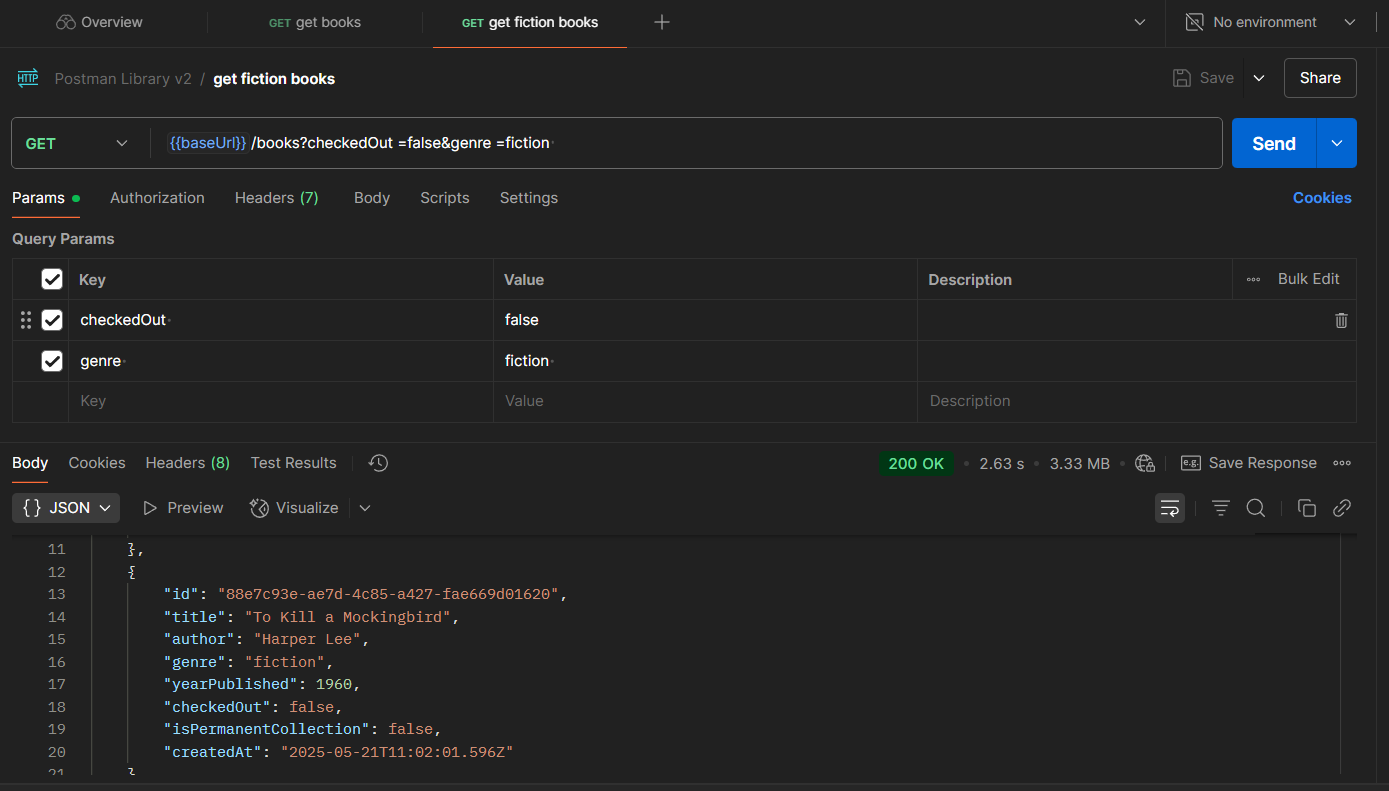
The **request** is sent over a **network** to some **server**. In our case, we made a request over the public internet to a server located at the address [**https://library-api.postmanlabs.com**](https://library-api.postmanlabs.com)

**•** [**https://library-api.postmanlabs.com/books**](https://library-api.postmanlabs.com/books)  
Click **"Set as variable"** to save the base URL to a variable.  
  
  
  
  
  
  
  
  
  
  
Name your new variable "**baseUrl**" and select "**Collection**" as the scope, then click **Set variable.**  
  
  
Click on your collection, then the **Variables** tab. Here you can view and edit your variables.

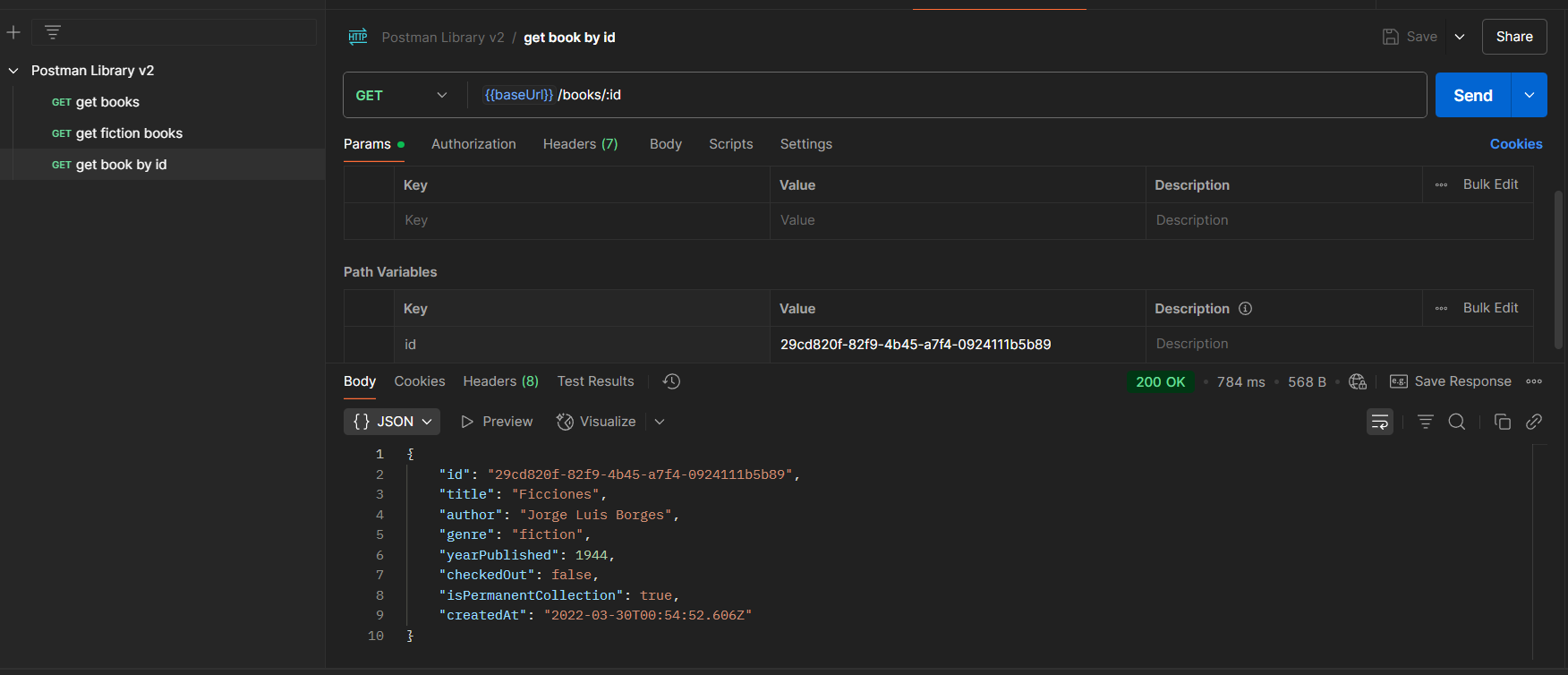
a request method (**GET**/**POST**/**PUT**/**PATCH**/**DELETE**, etc) and requested url  
Query parameter syntax

start with a question mark **?** followed by the key-value pairs in the format: **<key>=<value>**  
**Ex. GET** [**https://some-api.com/photos?orientation=landscape**](https://some-api.com/photos?orientation=landscape)**if there are multiple query it is separated by &**   
**Ex. GET** [**https://some-api.com/photos?orientation=landscape&size=500x400**](https://some-api.com/photos?orientation=landscape&size=500x400)ou can change your search directly from the URL by changing the value for the query parameter **q=<something else!>**

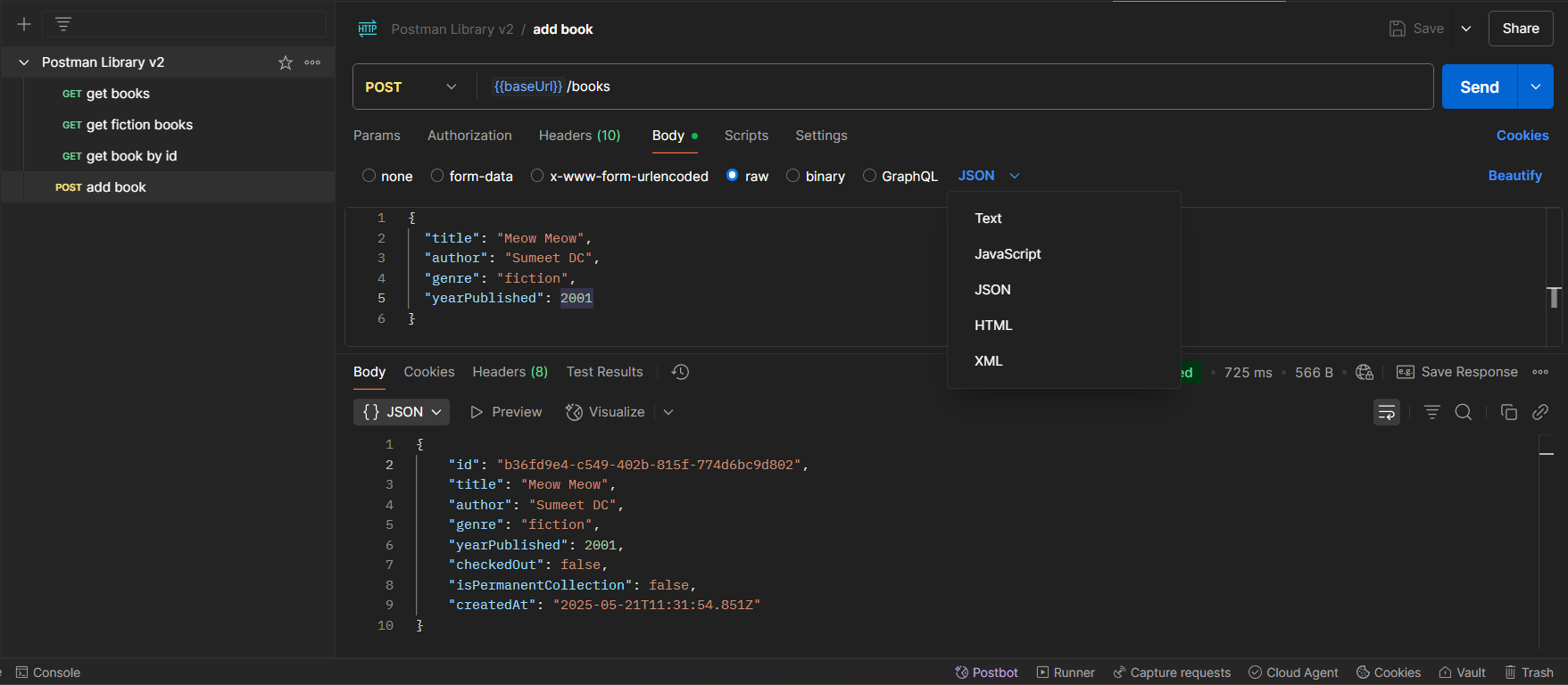
**In Query Parameter :**  
**Dublicate the get request rename it and then go to params in key enter genre value fiction you get all books of fiction in link you can see key value pair ?genre=fiction**

In similar way you can perform many other query look at img  


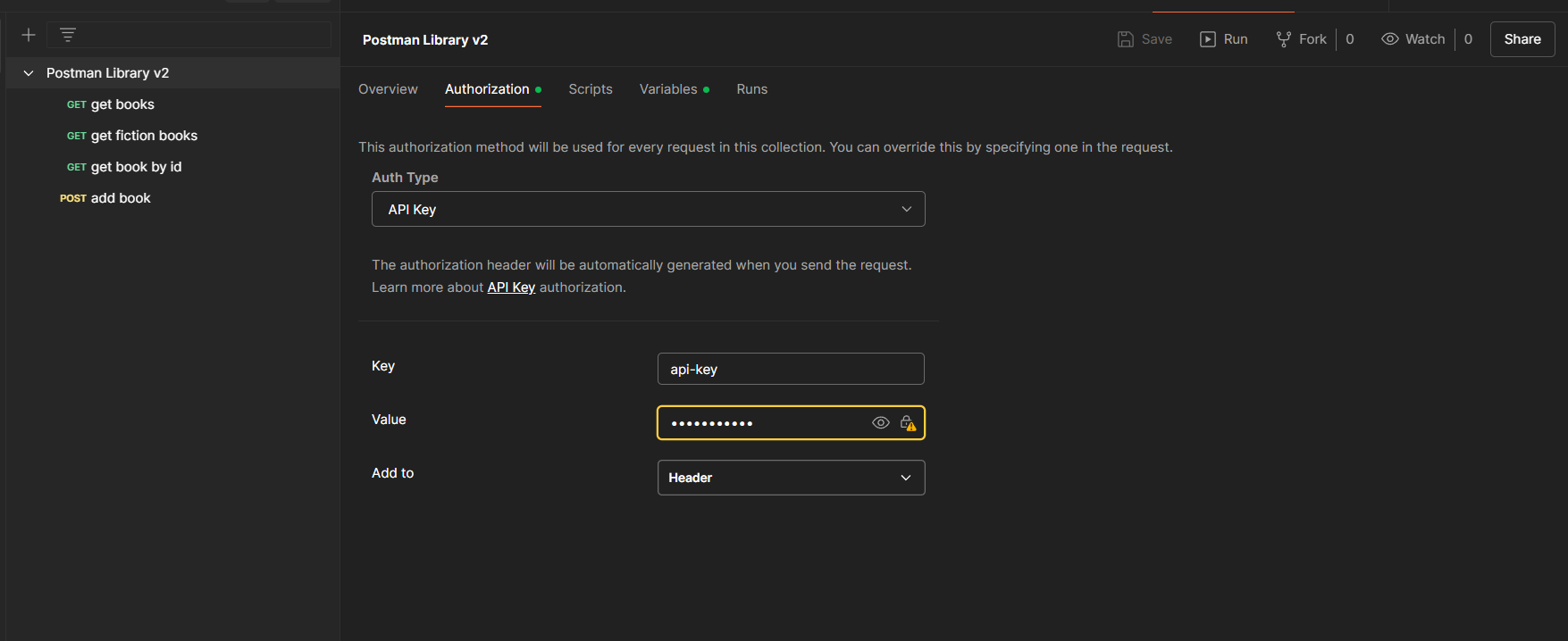
### Path Variable syntax A path variable is a dynamic section of a path and is often used for IDs and entity names such as usernames. The path variable comes immediately after a slash in the path. **GET** [**https://api.github.com/users/{username**](https://api.github.com/users/%7busername)**} -** API call with a value for **{username}** will fetch data about that user. You can have multiple path variables in a single request - **GET** [**https://api.github.com/repos/{owner}/{repoName**](https://api.github.com/repos/%7bowner%7d/%7brepoName)**}**

In the **Params** tab of the request, paste the **id** for "Ficciones" (**29cd820f-82f9-4b45-a7f4-0924111b5b89**) as the **value** for the path variable named **id**. *Make sure not to add any whitespace around the id value*.  


Post request  
to add a book via **POST** request with a **JSON** **Body** to submit book data   
what is the **Body**?

you're sending a request to add a new customer to a database, you might include the customer details in [JSON](https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/JSON) data format. Typically, you will use body data with **PUT**, **POST**, and **PATCH** requests. The **Body** tab in Postman enables you to specify the data you need to send with a request. use raw body data to send anything you can enter as text. There is dropdown menu indicate the format of your data (**Text**, **JavaScript**, **JSON**, **HTML**, or **XML**)  
  
To add new data you need to do authorization else shows error 400 / 401  
  
There are multiple methods for authorizing a request. Some examples are **Basic Auth** (username and password), **OAuth** (delegated authorization), and **API Keys** (secret strings registered to a developer from an API portal).   
  
Getting an API Key

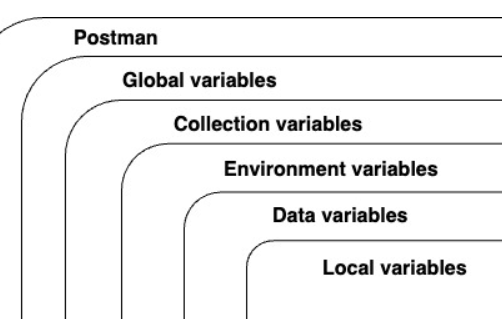
APIs that use API Key auth usually allow developers to sign up in a developer portal, where they will receive a random API Key that can be used to authorize their requests to the API.  
Take the : Key and Value of API click header enter key and value save and send request.  
  
Delete the **api-key** header

Click the Postman lib. select Authorization -> api key -> enter the key value -> save go to post send request.  
  
Authentication is directly added to collection level not to a doc level.

Variables in Postman :

1. using a variable saved us time and helped reduce redundant copy-paste of the request URL using the double curly brace syntax like this: **{{variableName}}**  
    Reuse values to keep your work [DRY](https://en.wikipedia.org/wiki/Don%27t_repeat_yourself#:~:text=%22Don't%20repeat%20yourself%22,data%20normalization%20to%20avoid%20redundancy.) (Don’t Repeat Yourself)
2. Hide sensitive values like API keys from being shared publicly

### Variable scopes



### Scripting in Postman

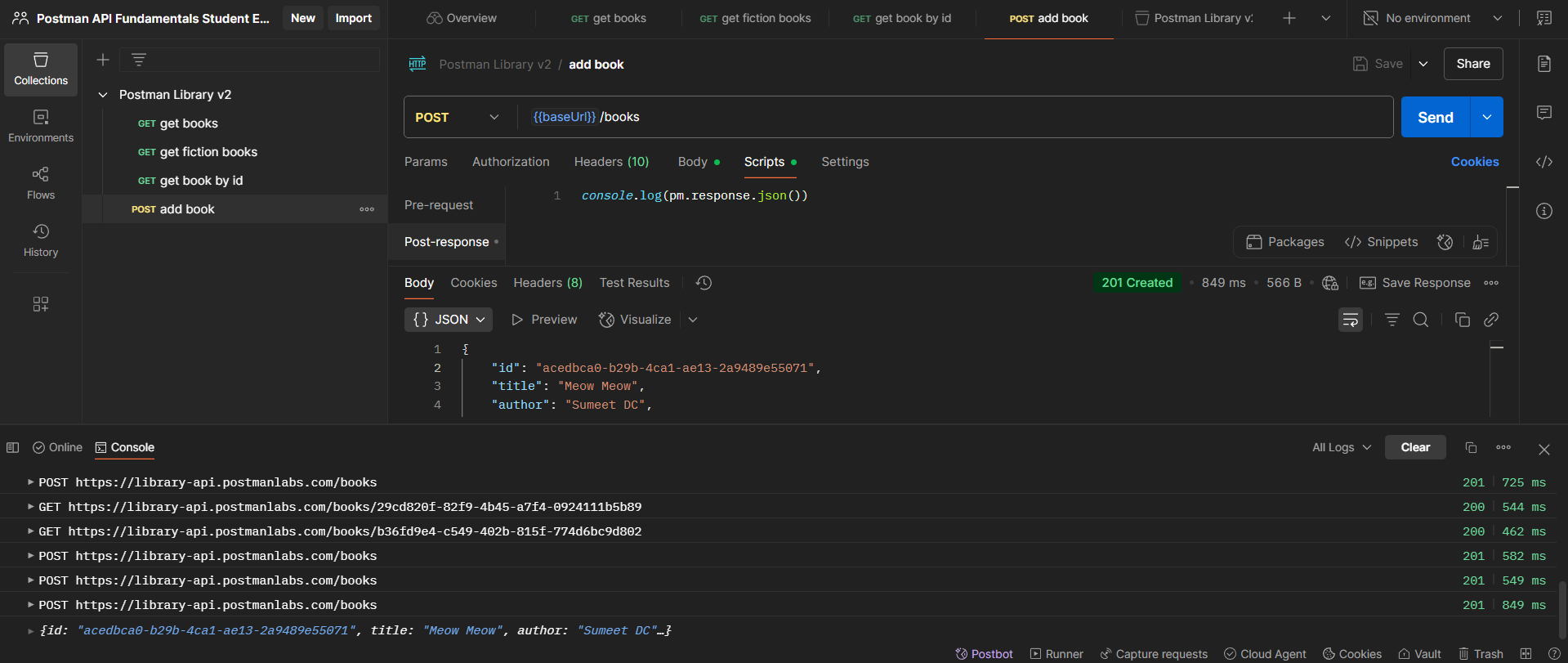
1. Immediately before a request is sent: [pre-request script](https://learning.postman.com/docs/writing-scripts/pre-request-scripts/) (**Pre-request Script** of Scripts tab).
2. Immediately after a response comes back: [post-response script](https://learning.postman.com/docs/writing-scripts/test-scripts/) (**Post-response** of Scripts tab).

The pm object (pm: postman):  
you can access json response body from an API: **pm.response.json()**

get collection variables like the value of **baseUrl: pm.collectionVariables.get(“baseUrl”)**

you can also set them with **pm.collectionVariables.set("variableName", "variableValue")** like this:  
  
**pm.collectionVariables.set(“myVar”, “foo”)**

### Add a script to your request



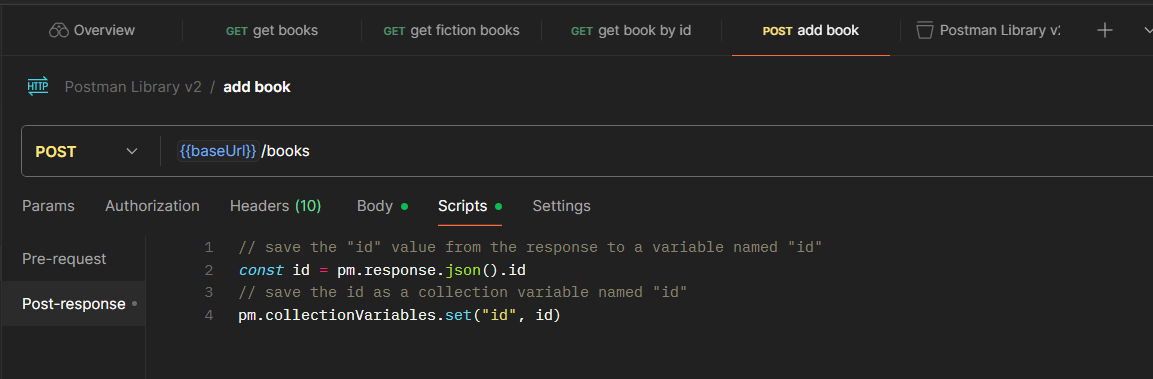
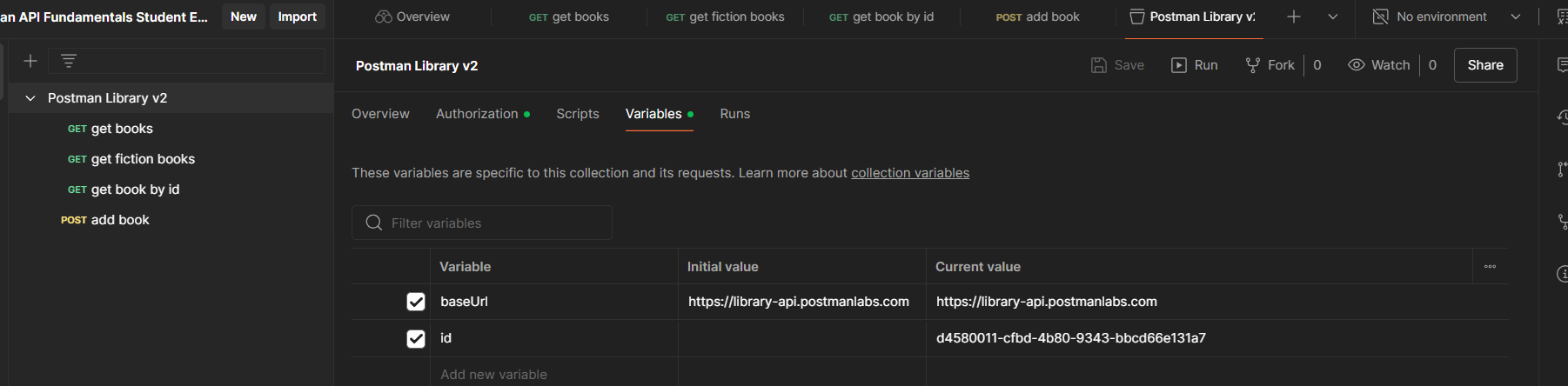
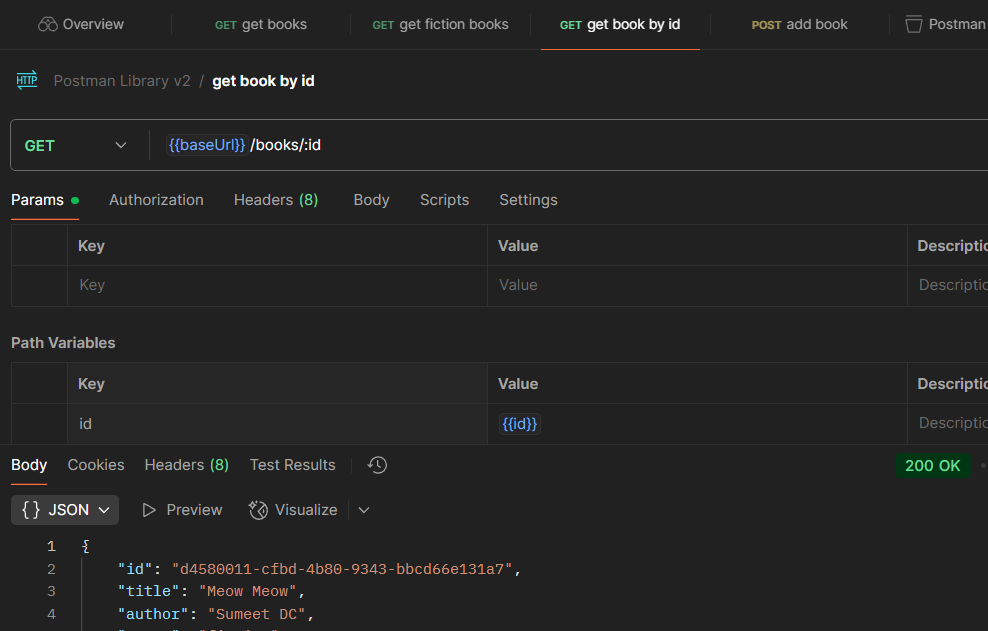
Go in script -> enter your js script -> save and hit send -> left corner click console response of log is done   
  
Setting and getting collection variables

To **set** a collection variable, use the **.set()** method with two parameters: the variable name and the variable value

**pm.collectionVariables.set("variableName", value)**

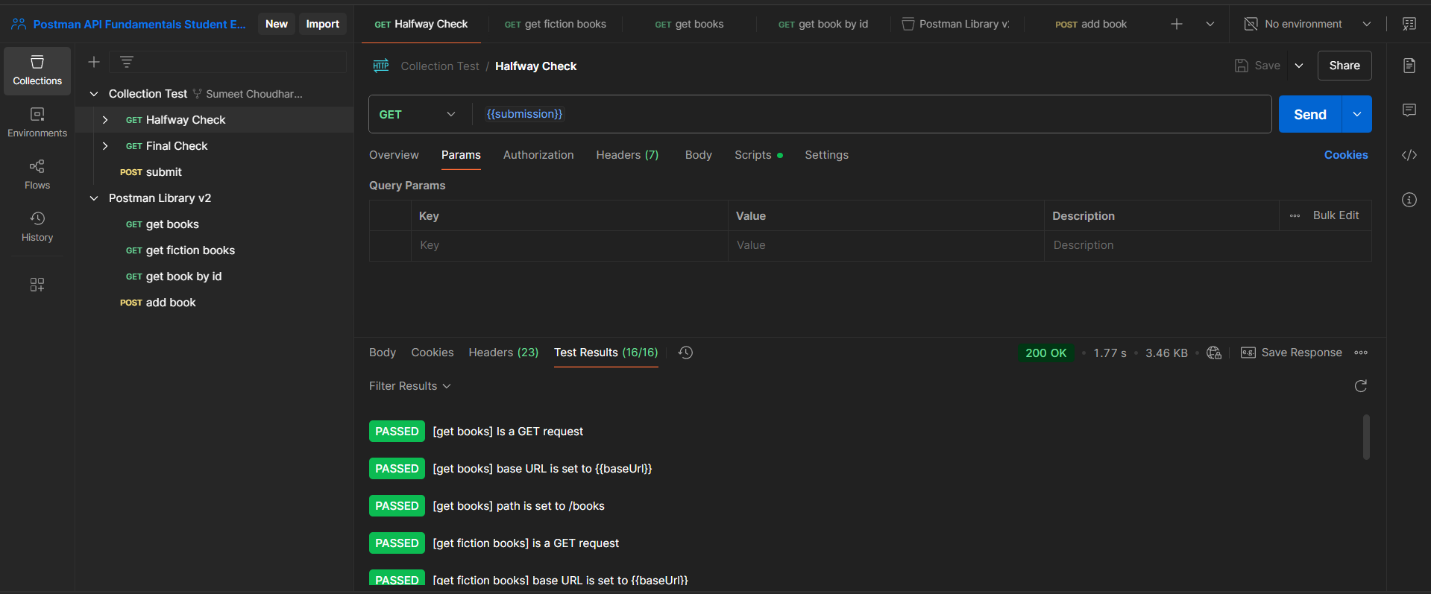
To **get** a collection variable use the **.get()** method and specify the name of the variable you want to retrieve:

**pm.collectionVariables.get("variableName")**

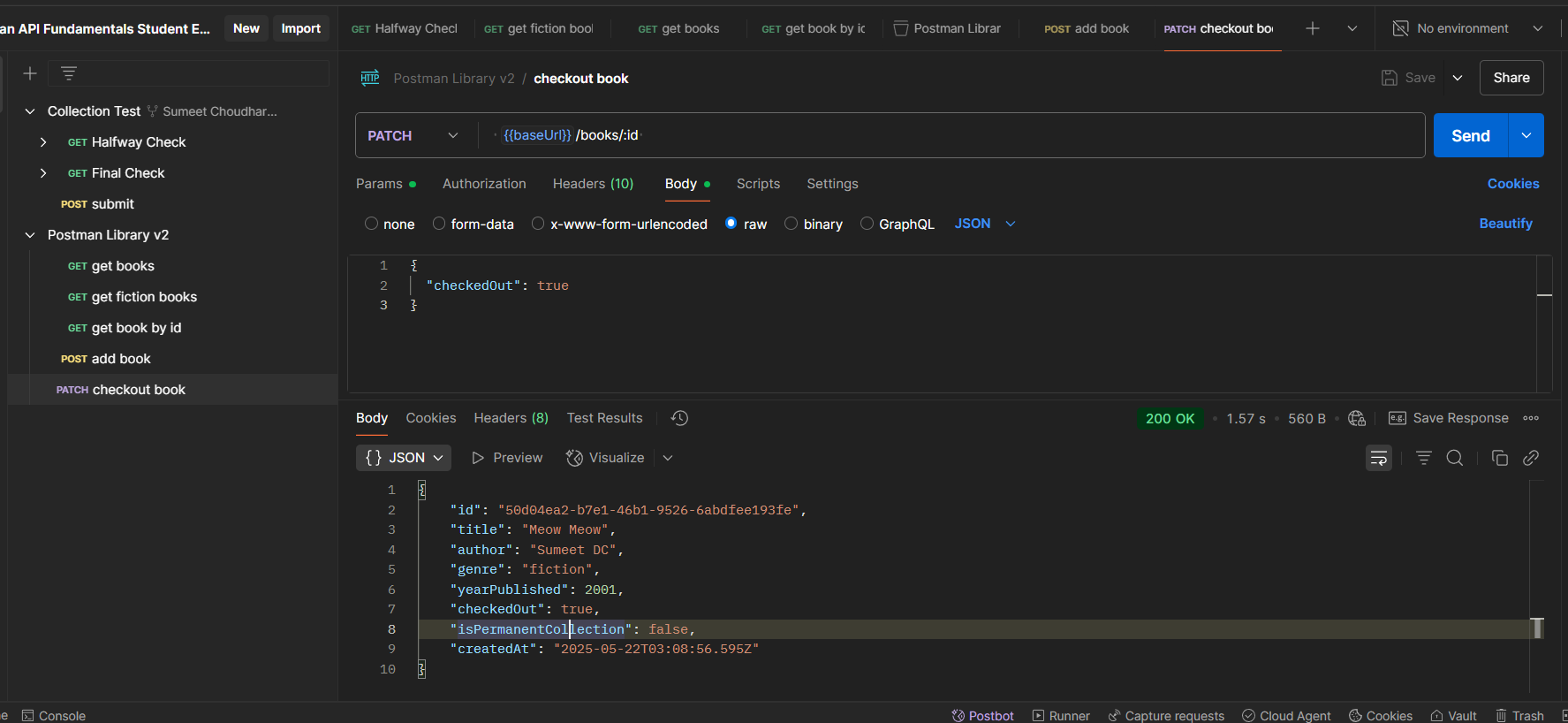
paste the js code in scripts hit send  
  
go the postman lib variables you see id  
go to get by id use that id in value section you don’t need to repeat and you can fetch

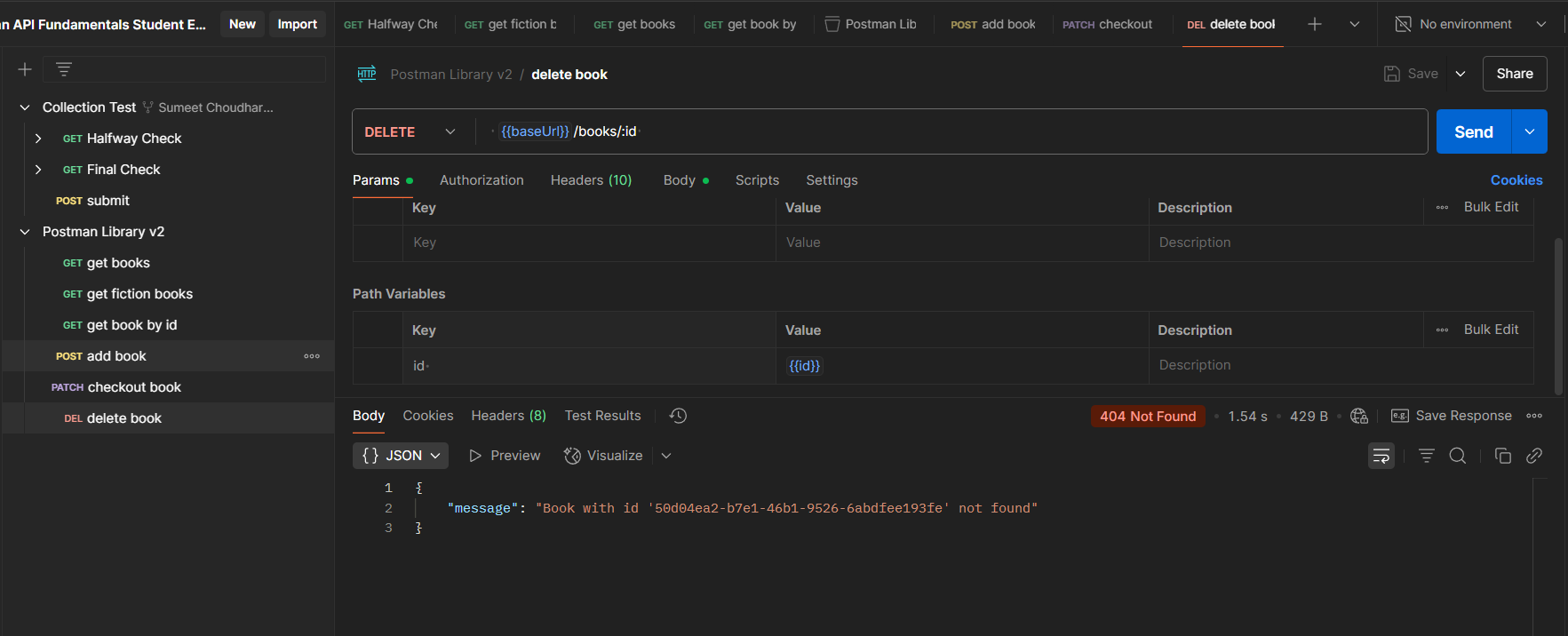
to fork a test collection into your workspace:

1. Get the API link for your **entire Postman Library API v2 :**  
     
   Click **View more actions** on your collection >Share > Via API > Generate New Key > Copy  
     
   This will create a read-only API key that will allow anyone with this link to access a JSON representation of your collection.
2. Navigate to your forked "**Collection Test**" collection, open the "Variables" tab, and **paste** thelink in your clipboard copied in the previous step into the collection variable named "submission"



### Make a request to update the book: Patch



Select Patch-> use base url ->write raw json ->save send  
  
Delete request -> select delete -> take baseurl and id -> save send -> 204 deleted click again u see 404 not found   
  


## Generating code snippets

On right hand side click -> < > and select the language and get code   
