Q #1) How can you set headers for all the requests that are in a particular Postman collection?

Answer: Postman collections allow adding pre-request scripts at both the collection as well as individual request level. To add any script that applies to all the requests that are present in the collection, we will need to add a pre-request script at the collection level.

Please follow the below steps to add a collection level pre-request script for adding a header to all the requests.

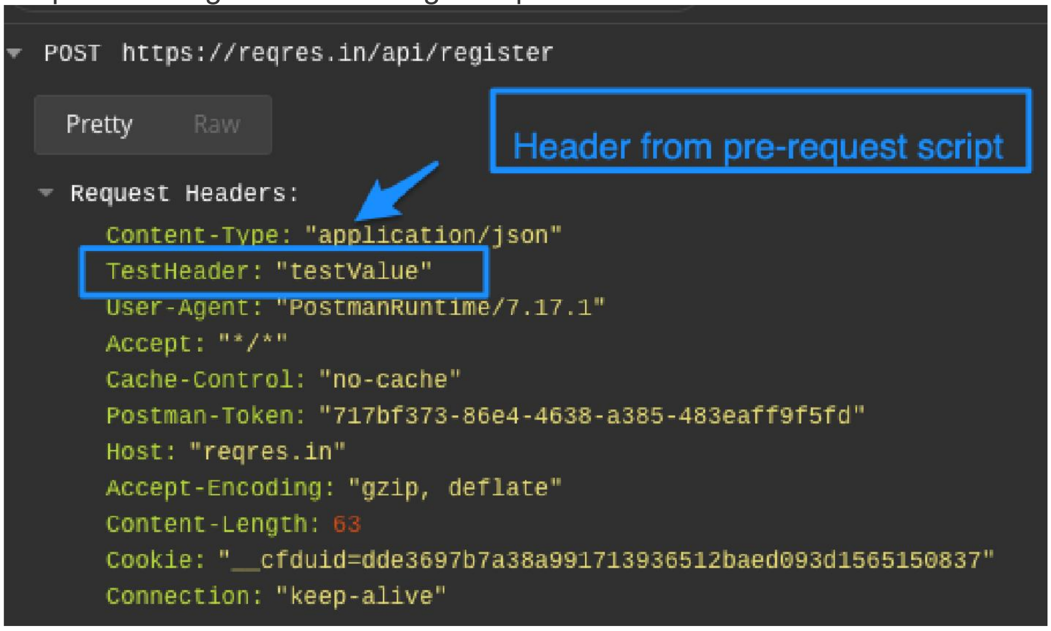
a) Open collection options by right-clicking the collection and navigate the pre-request script tab.

b) Now add the below script for adding a request header for all the requests.

```
pm.request.headers.add({

   key: 'TestHeader',

   value: 'testValue'

});
```

c) Click **Update** to save the collection level pre-request script.

d) Now execute any request in the collection (directly or through collection runner) and view the request details in the Postman console debugger to validate if the pre-request script is working fine and adding the specified header.



```
▾ POST https://reqres.in/api/register
  ▾ Request Headers:
      Content-Type: "application/json"
      TestHeader: "testValue"
      User-Agent: "PostmanRuntime/7.17.1"
      Accept: "*/*"
      Cache-Control: "no-cache"
      Postman-Token: "717bf373-86e4-4638-a385-483eaff9f5fd"
      Host: "reqres.in"
      Accept-Encoding: "gzip, deflate"
      Content-Length: 63
      Cookie: "__cfduid=dde3697b7a38a991713936512baed093d1565150837"
      Connection: "keep-alive"
```

Q #2) What's the use of Workspaces in Postman?

Answer: Postman workspaces are nothing but collaboration areas or space for one or many people to work on the same collection or set of collections. It's a way to logically separate the collections or requests from each other.
In other words, it is simply an abstraction in terms of logical separation of requests.

2 types of Workspaces are supported by Postman i.e. Team, and Personal.

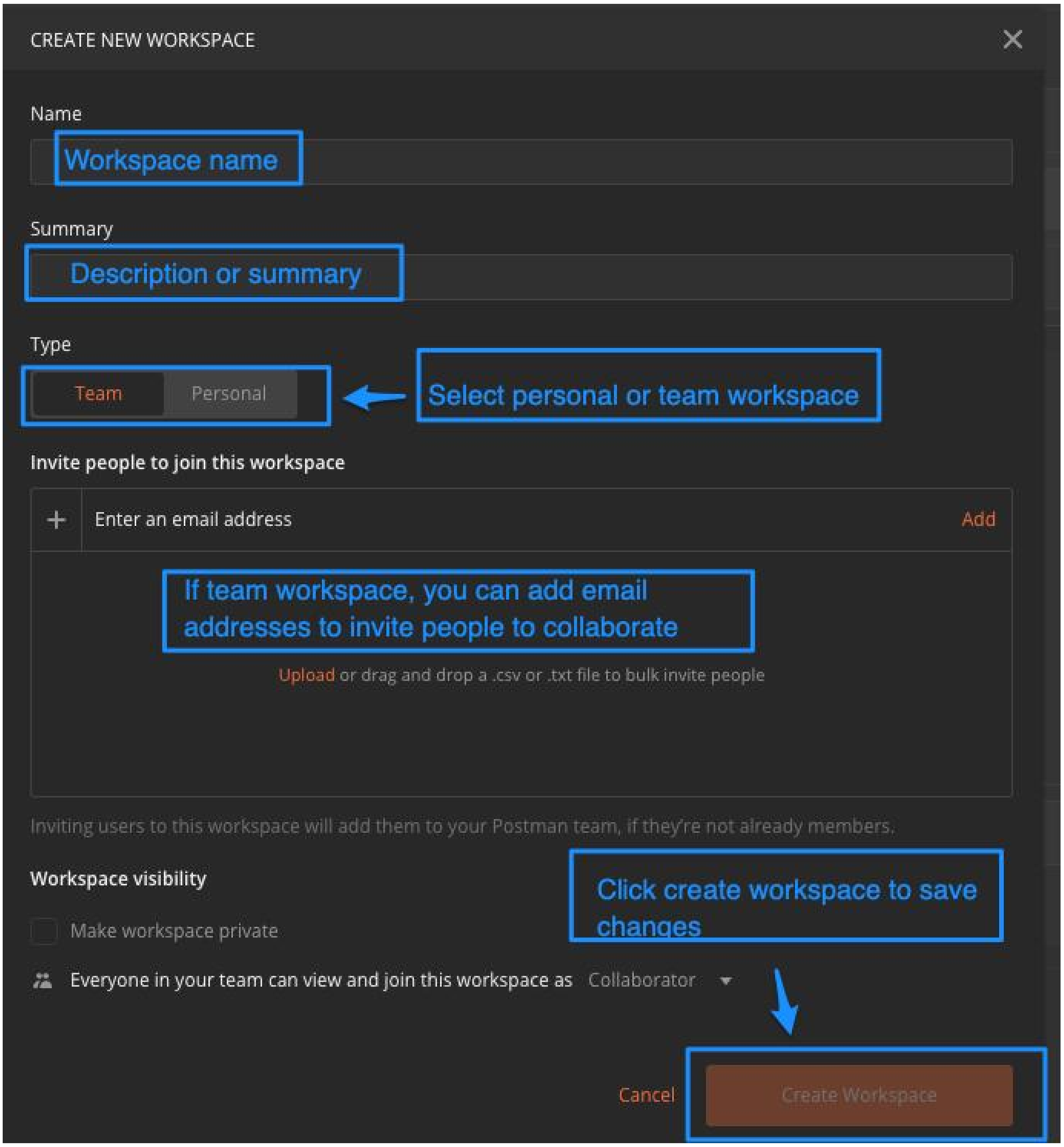
#1) Team Workspaces are created for collaborating with multiple people that are a part of the same team. Look at it from the perspective of a common shared repository in git, where anyone can pull the repository code and contribute.

Similarly, for all the people who are part of the team, the workspace gets shared and everyone can contribute. You can also invite new users to collaborate with your collection by sharing their email id and when someone joins or accepts that invite they will be able to collaborate with that collection.

#2) Personal Workspaces are a way to logically separate collections (or projects) from one another. These are useful when you are working with multiple projects and you wish to separate the associated requests/collections from each other. then you can create separate workspaces for both the projects.
To create a new workspace (either team or personal), simply click the workspace icon and then click “Create New”.

Once the workspace properties window opens, select whether you want to create a personal or team workspace. For team workspace, you can choose to invite people with their email addresses by asking them to collaborate on the workspace.

This is how the workspace properties window will look like.



CREATE NEW WORKSPACE

Name

Workspace name

Summary

Description or summary

Type

Team | Personal

Select personal or team workspace

Invite people to join this workspace

Enter an email address — Add

If team workspace, you can add email addresses to invite people to collaborate

Upload or drag and drop a .csv or .txt file to bulk invite people

Inviting users to this workspace will add them to your Postman team, if they're not already members.

Workspace visibility

Make workspace private

Everyone in your team can view and join this workspace as Collaborator

Click create workspace to save changes

Cancel | Create Workspace

Q #3) How can Postman collections run through the command line?

Answer: Postman has a command-line integration tool called Newman with which you can run any existing Postman collection.

Newman is a nodejs based package, which requires just a node environment to execute the collection and has full parity with the Postman collection runner i.e. the Newman collection runner supports the Postman capabilities like Running assertions, Pre-request scripts or any other scripts that are associated with the requests that are a part of the collection.

To use Newman:

- You need to have node installed.
- Now the Newman package needs to be installed through npm using the command.

```
npm install -g newman
```

- The collection needs to be executed and the associated environment configuration should be first exported to its JSON form through the Postman application
- Now run the below command to run the Postman collection through Newman.

```
newman run {{path to collection json}} -e {{path to environment json if any}}
```

Q #4) How can you generate HTML based reports running tests through the Postman?

Answer: Newman uses the concept of reporters and templates to generate HTML reports for the executed collection.
Hence, to generate HTML reports, you first need to install a reporter. You can install any of the available HTML reporters like Newman-reporter-html as a node package through the below command.

```
npm install -g newman-reporter-html
```

Once the HTML reporter is installed, we can use the Newman command to run the collection with -r flag i.e. the reporter flag and specify the reporter name as HTML.

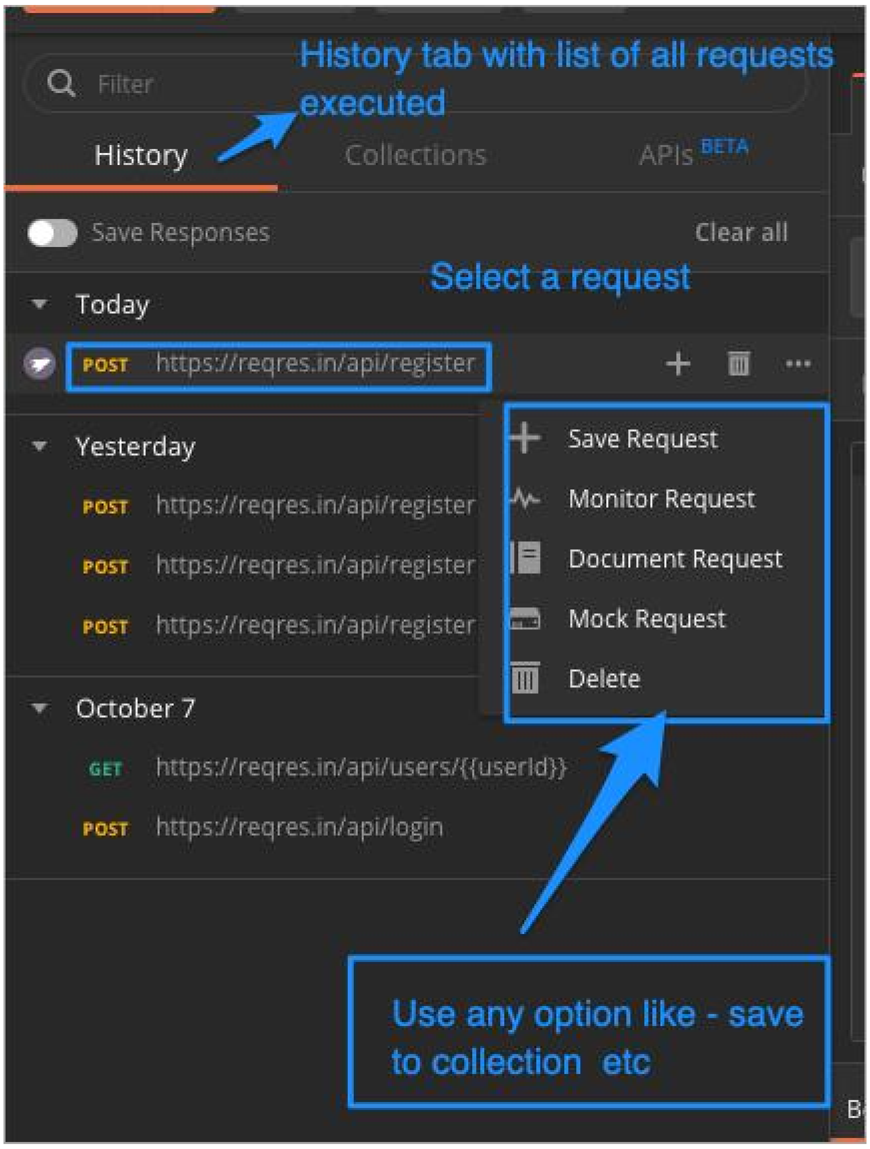
The below command is used:

```
newman run {{path to collection json}} -e {{path to environment json if any}} -r html
```

Please note that as we have not mentioned the name or folder where we want the reports to get generated, by default the reports will be generated in a folder named "Newman" that gets created in the same directory where the Newman command is executed from.

Q #5) How can we use Postman history and save requests from the Postman history to the existing or new collections?

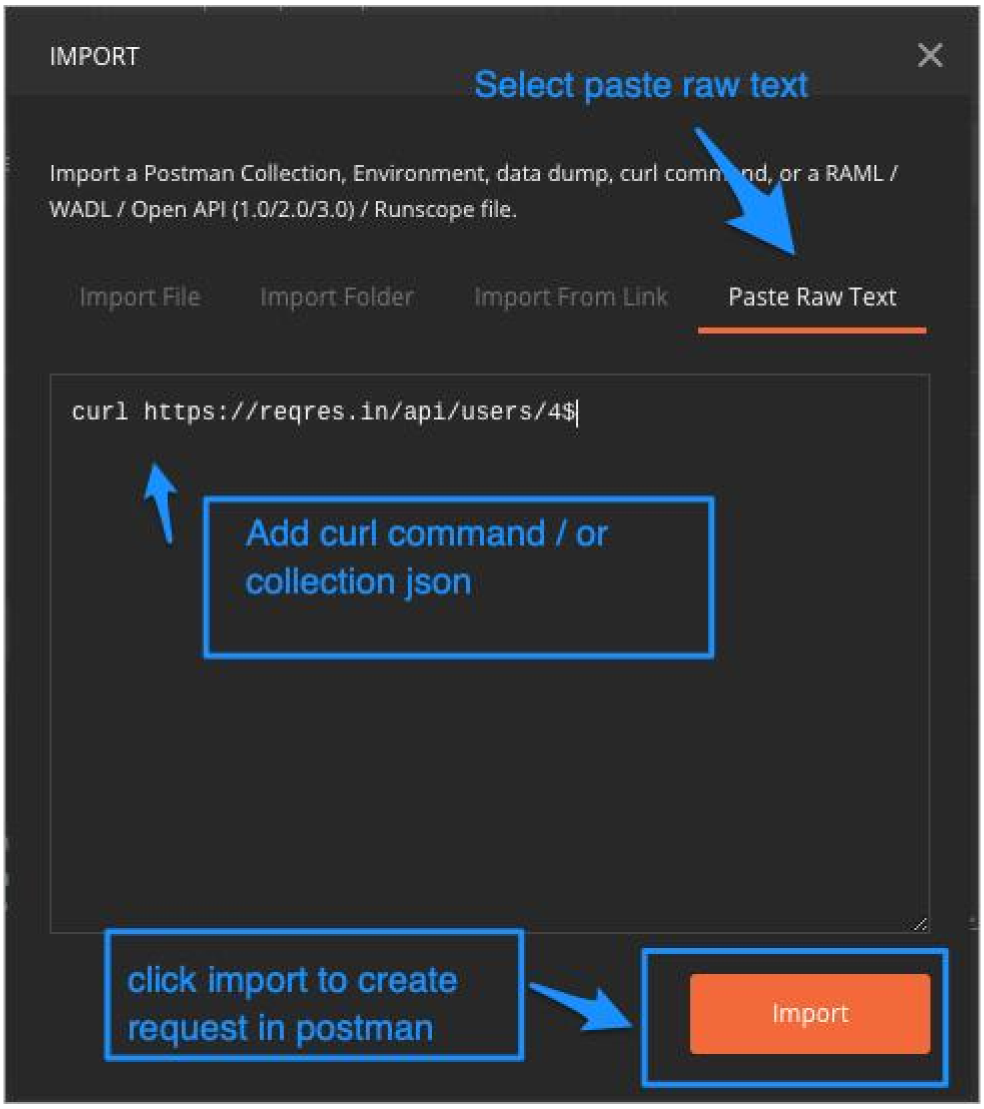
Answer: Any request that gets executed through the Postman application, is available for reference in the History section of the application. So in case, the request was not saved to a collection before it was executed, we can always go back to the history section to fetch the executed request and save it to the collection.
Refer to the below screenshot for more details.


History tab with list of all requests executed
Filter
History
Collections
APIs BETA
Save Responses
Clear all
Select a request
Today
POST https://reqres.in/api/register
Save Request
Monitor Request
Document Request
Mock Request
Delete
Yesterday
POST https://reqres.in/api/register
POST https://reqres.in/api/register
POST https://reqres.in/api/register
October 7
GET https://reqres.in/api/users/{{userId}}
POST https://reqres.in/api/login
Use any option like - save to collection etc

Q #6) How can you import requests in formats other than cURL into Postman?
Answer: Postman supports a lot of common request formats to export requests to. **Example.** Java, C#, Python, PHP, etc. It supports almost all the commonly used libraries and language bindings.
For importing requests, it supports cURL for now. i.e. you can paste a curl command in request import and it gets converted to Postman requests, but the same cannot be done using any other language bindings like Java, Python, etc.

The other way to import multiple requests at once is to import the entire collection directly through a file or collection JSON pasted as raw text in the import window.

Given below is a screenshot of how the import raw text section of the import options will look like.


IMPORT
Select paste raw text
Import a Postman Collection, Environment, data dump, curl command, or a RAML / WADL / Open API (1.0/2.0/3.0) / Runscope file.
Import File
Import Folder
Import From Link
Paste Raw Text
Add curl command / or collection json
click import to create request in postman
Import

```
curl https://reqres.in/api/users/4$
```

Q #7) Is it possible to Log Requests and Responses in Postman?

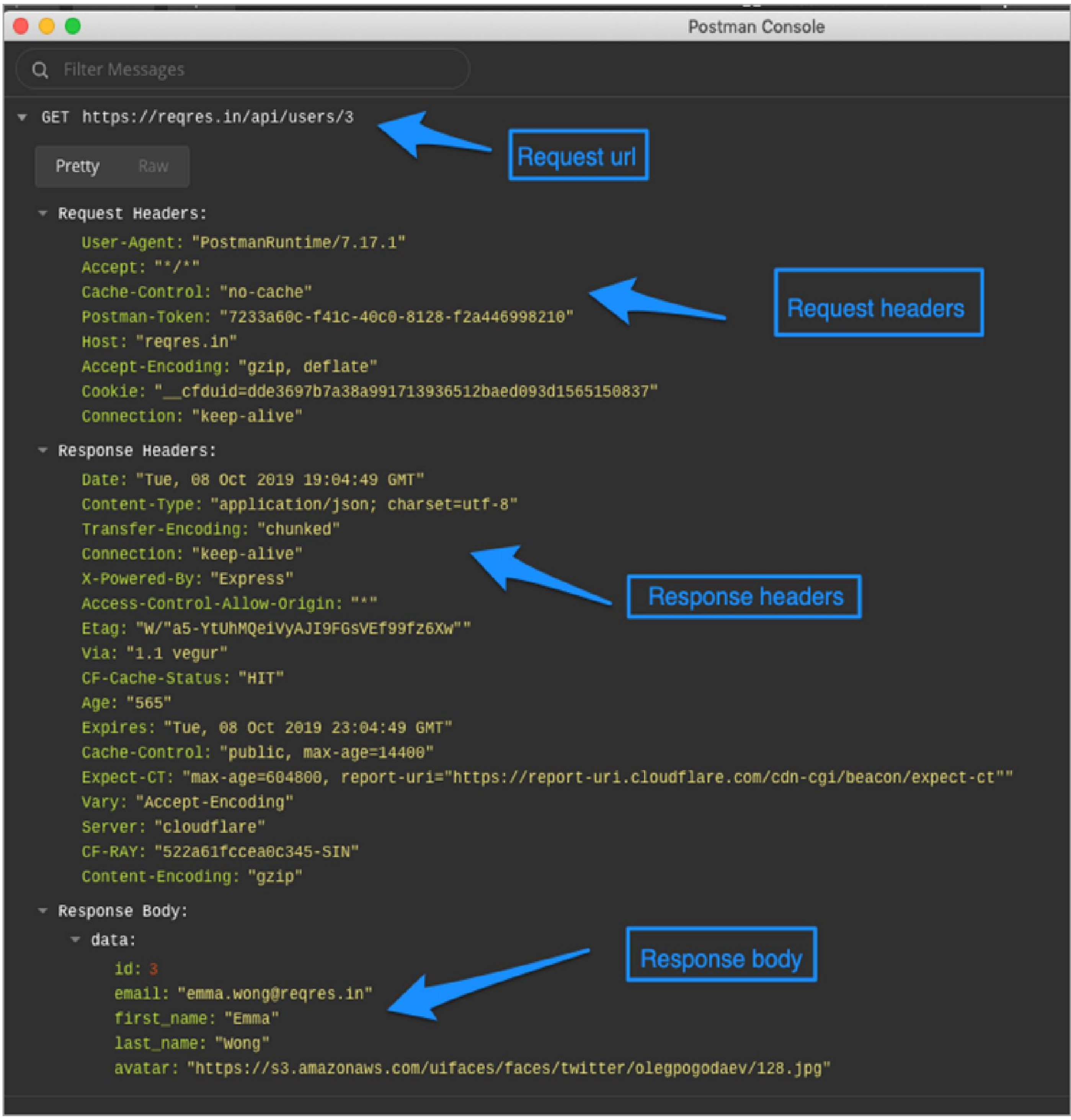
Answer: Postman allows viewing the response body and other request parameters in the application itself.

But there are times when we have applied pre-request scripts and as we are unable to see details about the request URLs and headers that were used while executing the request, and it's always important to see how the actual request looked like.

To view complete Requests and Responses for the executed collection or individual request, Postman provides an additional tool console called "Postman Console" and it can be used to view all the requests/response details.

It's also useful to see the output of any console.log statements that are a part of the pre-request scripts or tests.

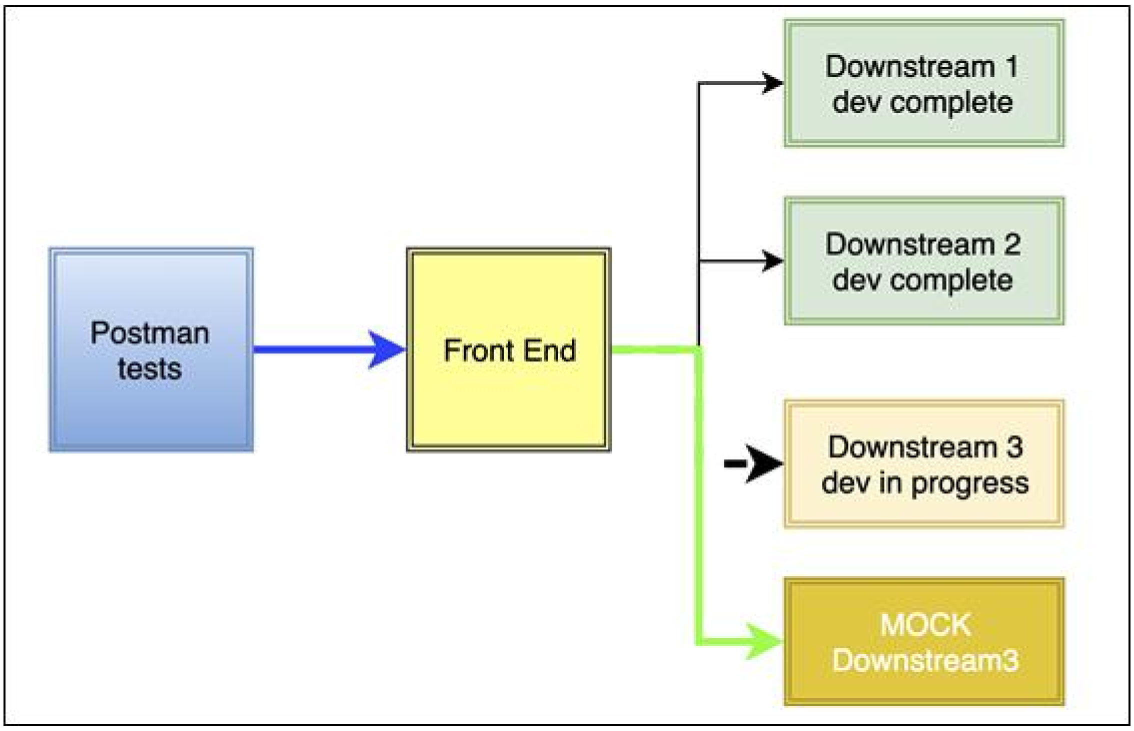
Given below is the screenshot of the Postman console window.



```
GET https://reqres.in/api/users/3

  Request Headers:
     User-Agent: "PostmanRuntime/7.17.1"
     Accept: "*/*"
     Cache-Control: "no-cache"
     Postman-Token: "7233a60c-f41c-40c0-8128-f2a446998210"
     Host: "reqres.in"
     Accept-Encoding: "gzip, deflate"
     Cookie: "__cfduid=dde3697b7a38a991713936512baed093d1565150837"
     Connection: "keep-alive"
  Response Headers:
     Date: "Tue, 08 Oct 2019 19:04:49 GMT"
     Content-Type: "application/json; charset=utf-8"
     Transfer-Encoding: "chunked"
     Connection: "keep-alive"
     X-Powered-By: "Express"
     Access-Control-Allow-Origin: "*"
     Etag: "W/"a5-YtUhMQeiVyAJI9FGsVEf99fz6Xw""
     Via: "1.1 vegur"
     CF-Cache-Status: "HIT"
     Age: "565"
     Expires: "Tue, 08 Oct 2019 23:04:49 GMT"
     Cache-Control: "public, max-age=14400"
     Expect-CT: "max-age=604800, report-uri="https://report-uri.cloudflare.com/cdn-cgi/beacon/expect-ct""
     Vary: "Accept-Encoding"
     Server: "cloudflare"
     CF-RAY: "522a61fccea0c345-SIN"
     Content-Encoding: "gzip"
  Response Body:
     data:
        id: 3
        email: "emma.wong@reqres.in"
        first_name: "Emma"
        last_name: "Wong"
        avatar: "https://s3.amazonaws.com/uifaces/faces/twitter/olegpogodaev/128.jpg"
```

Q #8) How can Postman be used to create Mock Servers?

Answer: Postman allows users to simulate backend servers or any API endpoints that are still under active development and to run an integration test or end to end test, you still need to get some pre-defined response through those endpoints.


Postman tests
Front End
Downstream 1 dev complete
Downstream 2 dev complete
Downstream 3 dev in progress
MOCK Downstream3

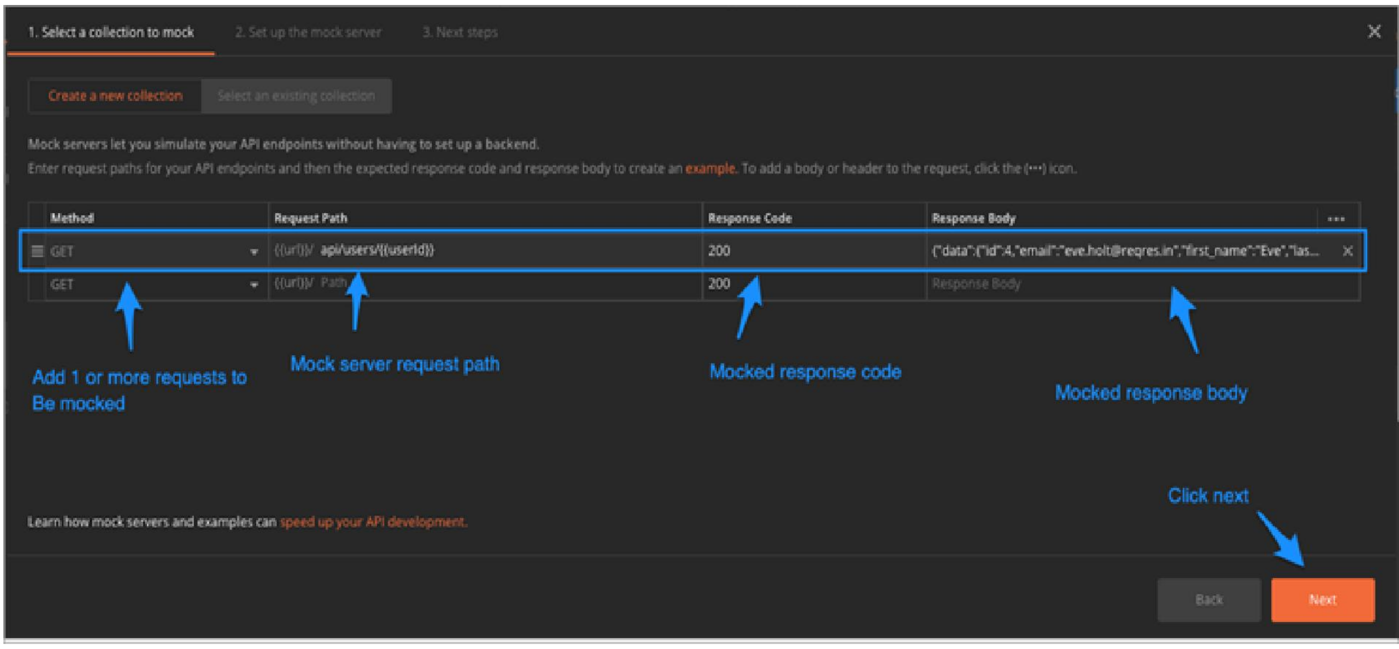
Refer to the above diagram, where a front end server/API has few downstream dependencies, of which one dependency is still a work in progress. To reduce the dependency of the front end being able to use the downstream until its complete, we can create a mock for the downstream and use it till the time the downstream dependency is not complete.

Thus mock servers are nothing but a fake implementation for the backend. To create/use mock servers, a user should be registered with Postman at least for a free account (Postman allows users to register for a free account through the user's email).

Also, please note that for a free account the no. of calls to a mock server is limited to 1000 (This limit can be increased by buying an enterprise plan or purchasing an additional quota from the Postman account usage page).

Here is a Video Tutorial:
To create a mock server, you can use an existing collection i.e if you want to create a mock for your entire collection or add requests when creating a mock server.

Follow the below steps to create a mock server:
a) Click New and select "Mock Server".
b) Add request method(s) to be mocked and add the response code and response body to be returned while the particular API endpoint is called.


1. Select a collection to mock

2. Set up the mock server

3. Next steps

Create a new collection | Select an existing collection

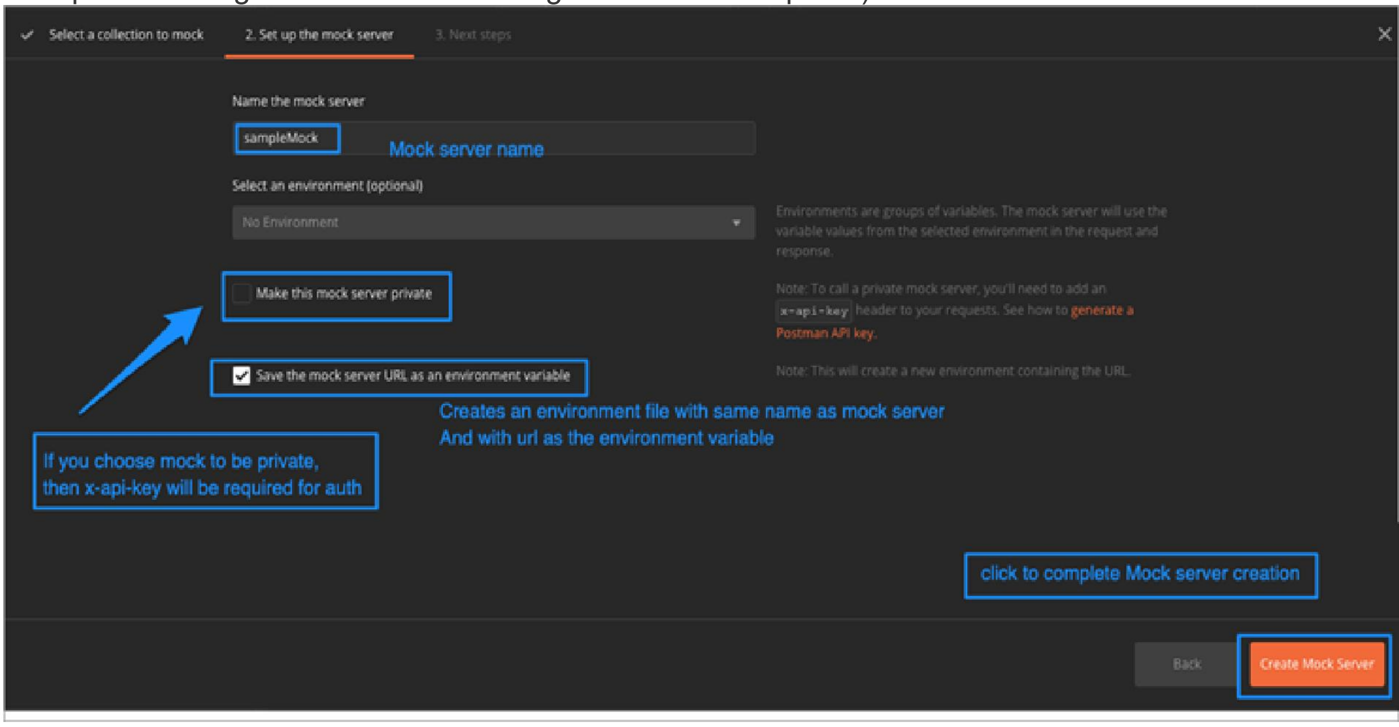
Mock servers let you simulate your API endpoints without having to set up a backend.

Enter request paths for your API endpoints and then the expected response code and response body to create an example. To add a body or header to the request, click the (•••) icon.

Method	Request Path	Response Code	Response Body
GET	{{url}}/ api/users/{{userId}}	200	{"data":{"id":4,"email":"eve.holt@reqres.in","first_name":"Eve","las...
GET	{{url}}/ Path	200	Response Body

Learn how mock servers and examples can speed up your API development.

c) Click Next and choose the mock server name (If you want this mock server to be private, then an authorization header named x-api-key which will be generated for the user profile through which Postman is signed in will be required).


Select a collection to mock

2. Set up the mock server

3. Next steps

Name the mock server: sampleMock

Select an environment (optional): No Environment

Environments are groups of variables. The mock server will use the variable values from the selected environment in the request and response.

Make this mock server private

Note: To call a private mock server, you'll need to add an `x-api-key` header to your requests. See how to generate a Postman API key.

✓ Save the mock server URL as an environment variable

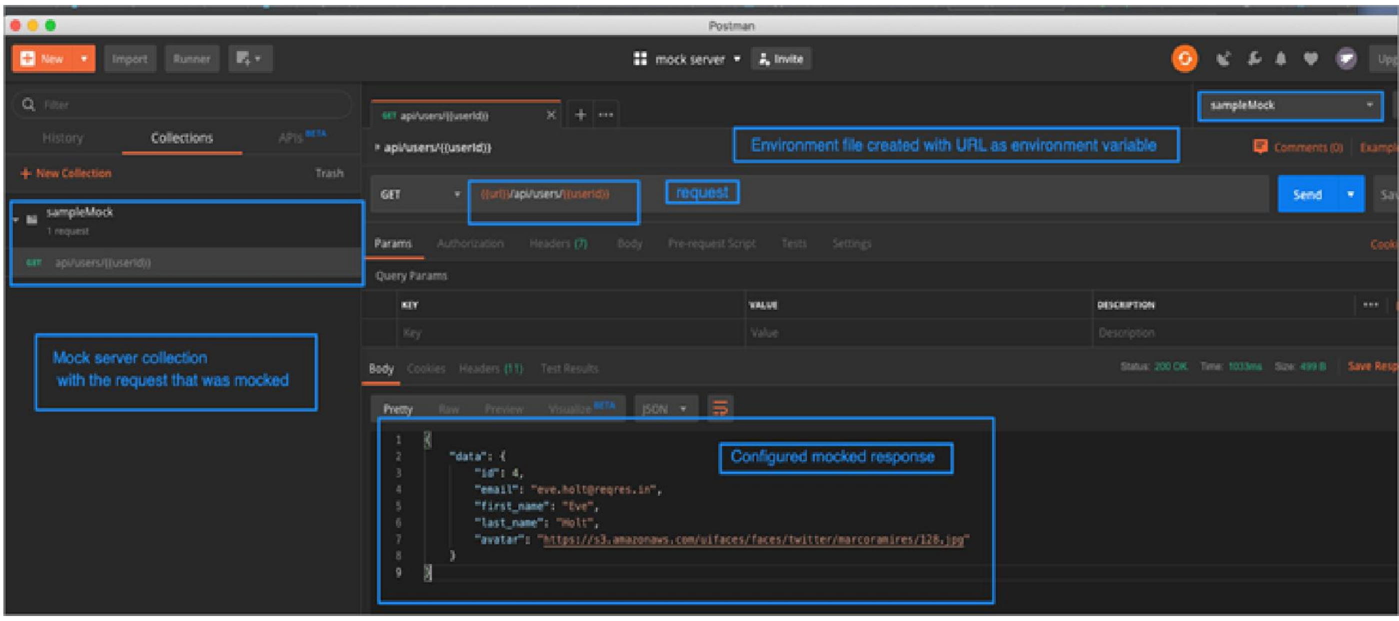
Note: This will create a new environment containing the URL.

Mock server name
If you choose mock to be private, then x-api-key will be required for auth
Creates an environment file with same name as mock server And with url as the environment variable
click to complete Mock server creation
Back
Create Mock Server

d) Click “Create Mock Server”. Essentially this will host your API endpoint on some Postman server and will return the set response whenever the particular endpoint is called.

e) It will also create a new environment file (that was set during the mock server setup) and set the URL of the mocked API endpoint as an environment variable.

f) You are all done and now, you can use this mocked endpoint to send requests to. This mocked implementation can be used for dependent services in the actual code if the real services are still under deployment.



```
{
    "data": {
        "id": 4,
        "email": "eve.holt@reqres.in",
        "first_name": "Eve",
        "last_name": "Holt",
        "avatar": "https://s3.amazonaws.com/uifaces/faces/twitter/marcoramires/128.jpg"
    }
}
```

Q #9) How can we use Custom Javascript Libraries with Postman Pre-request Scripts or Tests?

Answer: Postman sandbox provides a lot of libraries that are built-in and are available for usage. For a complete list of such libraries, refer here to use these libraries, and you will need to add them in pre-request scripts or tests using 'require'.

Here is a Video Tutorial:

Let's see one such example using moment.js and this library provides a lot of helpful functions to perform formatting around time.

Let's say, there is a POST request which has to say, created date for a user and it expects the date format **YYYY-MM-DD**. Though it could be achieved using plain javascript as well, moment.js can do this with one line of code.
Let's see this in action now. In the pre-request script, just add the following line of code, to get the formatted data, stored in an environment variable.

```
var moment = require('moment');

pm.environment.set('formattedDate',moment().format('YYYY-MM-DD'));
```

Another example of the moment could be to add a particular value to the current date and use it in the request body. **For example,** you want to set a field like an expiry date, to current date + 2 days, as well as with formatting to 'YYYY-MM-DD', and you can simply use the script as below.

```
pm.environment.set('expiryDate',moment().add(2,'days').format('YYYY-MM-DD'));
```

In the above script, we can see that we've added or included 'moment.js' library and used the object as a simple Javascript code. Similar to pre-request scripts, these libraries or modules can be used in the post-request scripts or tests as well to do similar stuff.

Other libraries are available like crypto js which could be useful to convert a text to encrypted value like Base 64 or encoded hash and could be used as a part of the request body.

Q #10) What are Postman Monitors?

Answer: Postman monitors are nothing but collection monitors that are set up and are executed as per the configured frequency. These are generally used when someone wants their collection to run at a particular frequency and the results are required to be monitored with failures being notified through email or slack integration.
Generally, teams with their infrastructure like CI and own cloud servers would not prefer to use Postman defined monitors as it would run only on published or public endpoints or on mocked endpoints (if configured through mock servers).

11) What is Postman?

Postman is an API(application programming interface) development tool which helps to build, test and modify APIs.

12) Why should we use Postman?

– Postman is a free tool.
– Postman has the ability to make various types of HTTP requests(GET, POST, PUT, PATCH), saving environments for later use, converting the API to code for various languages(like JavaScript, Python).
– Postman manages end-to-end API lifecycle, from design, mocking, testing, and deploying, all the way to maintenance and deprecation.
– Postman's Platform Runtime Service can manage API collections, workspaces, examples and environments.
– Postman can be easily integrated with CI/CD tools like Jenkins.
– Postman provides extensive documentation and widely supported by the community.

13) What are the different types of variable scopes available in Postman?.

Postman provides the following environment variable scopes:
– Global Variables: It allow you to access data between collections, requests, test scripts, and environments. Global variables are available throughout a workspace.
– Envrionment Variables: It allows you to tailor your processing to different environments, for example local development vs testing or production.
– Local Variables: They are temporary, and only accessible 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.
– Collection Variables: are available throughout the requests in a collection and are independent of environments, so do not change based on the selected environment.
– Data Variables: come from external CSV and JSON files to define

data sets you can use when running collections via Newman or the Collection Runner.

14) What is the order of preference scope for each Postman variable?
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. Below is the preference of the variables in descending order Local Variables -> Data Variables -> Environment Variables -> Collection Variables -> Global Variables

15) How can you log the variable value in Postman?
Variable values can be logged in the Postman console by using the following command:
console.log(pm.variables.get("variable_key"));

16) What is a dynamic variable and how can you use it in Postman?
Postman provides more than 100 dynamic variables which can generate unique random values. Some of the examples are:
{{$guid}} : A v4 style guid
{{$timestamp}}: The current timestamp (Unix timestamp in seconds)
{{$randomInt}}: A random integer between 0 and 1000

We can use dynamic variables in pre-request script or test scripts using the following script:
pm.variables.replaceIn('{{$randomEmail}}');

17) What are the different authorization options available in Postman?
API Requests can be authorized using the following options:
– API Key
– Bearer Token
– Basic auth
– Digest auth
– Oauth 1.0
– Oauth 2.0
– Hawk Authentication
– AWS Signature
– NTLM Authentication

18) How can you reuse your authentication token for different requests?
We can create a Collection and add all the requests to that collection. We can add the authorization token
in the collection and then we can select "Inherit auth from parent" option as authorization for every request.

19) What are the types of API Requests which is supported by Postman?

GET, POST, PUT, PATCH, DELETE, COPY, HEAD, OPTIONS, LINK, UNLINK, PURGE, LOCK, UNLOCK, PROPFIND and VIEW.

20) What is difference between Query Params and Path Variables?
Path Variables is used to identify a specific resource or resources whereas Query Parameter is used to sort/filter those resources.

21) What is a Collection in Postman?
We can group requests into collections to keep your workspace organized, to collaborate with teammates, to generate API documentation, test suites, and to automate request runs.

22) Write test code to check whether the response status is 200 or not?

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

23) In which type of encoding does postman accept authorization credentials?
Postman accept authorization in Base64 encoding only. This is provided inbuilt in Postman or else you can also refer third party websites to convert the credentials in base64.

24) Can we have two global scope variables with the same name in Postman?
Since global variables have global scope i.e. without any environment, they cannot have duplicate names.

25) What is a workspace in Postman?
Workspaces allow us to organize our Postman work and collaborate with teammates. We can group our projects together, with workspace acting as the single source of truth for related APIs, collections, environments, mocks, monitors, and other linked entities.

26) How many types of workspaces are present in Postman?
There are 2 types of workspaces in Postman:
– Personal Workspace: This workspace is only visible to the account user
– Team Workspace: This workspace can be accessed and shared across the team members

27) Can we import local variables in Postman Monitors?
Yes. Postman monitors allows to import local variables but it does not allow to import global variables.

28) What is binary in Post method in Postman?

Binary form in Postman is designed to send the information in a format that cannot be entered manually. Since everything in a computer is converted to binary, we use these options which cannot be written manually such as an image, a file etc.

29) What are the two types of scripts in Postman?

We can write two types of script in Postman:
– Tests script
– Pre-request script

30) How can Postman collections run through the command line?

Postman collections can be directly run from the command line using the Newman tool. Newman is a nodejs based package, which requires just a node environment to execute the collection and has full parity with the Postman collection runner i.e. the Newman collection runner supports the Postman capabilities like Running assertions, Pre-request scripts or any other scripts that are associated with the requests that are a part of the collection.

31) Which one will be preferred in postman- a global variable or a local variable?

In postman, if 2 variables have the same name(one being local, other global) then the higher priority is of the local variable. it will overwrite the global variable.

32) What is a Collection Runner?

Collection Runner is used to run set of requests in a specified sequence and environment. We can also use external data files to run the requests in multiple iterations. Collection Runner logs all the request test results and scripts can pass data between requests.

33) How can you monitor APIs in Postman?

We can use Collection Monitors to schedule and monitor all the APIs in a collection. Postman will send a email notification for every API Request/Test failure.

34) Which BDD library is used to assert values inside Postman tests?

Chai assertion library is used in Postman tests. It provides many language chains which are used to build assertions within the API Tests.

35) What is a Pre-Request script?

We can use pre-request scripts to execute JavaScript before a request runs. We can perform pre-processing tasks
such as setting variable value, parameters, headers and body-data