A pre-request script is a piece of JavaScript code that runs before your request is sent. You can use it to set variables, parameters, headers, or body data for your request. You can also use it to perform some calculations or validations before sending the request.

There are different ways to get the data for the pre-request script, depending on your use case. Here are some common sources of data:

* **Environment variables**: You can store and access data in environment variables using pm.environment.get() and pm.environment.set() methods. Environment variables are useful for storing data that is specific to a particular environment, such as development, testing, or production.
* **Global variables**: You can store and access data in global variables using pm.globals.get() and pm.globals.set() methods. Global variables are useful for storing data that is shared across all environments and requests, such as authentication tokens or user IDs.
* **Data files**: You can import data from CSV or JSON files and access them using pm.iterationData.get() method. Data files are useful for running multiple iterations of the same request with different data values, such as testing different scenarios or edge cases.
* **Request data**: You can access data from the request itself, such as the URL, method, headers, parameters, or body, using pm.request object. Request data is useful for manipulating or validating the request before sending it, such as generating dynamic values or checking the request format.

To select a data file in JSON format from the body when you run a collection in Postman, you need to do the following steps:

* Create a JSON file with an array of key-value pairs that correspond to the variables you want to use in your request body. For example, if you want to send different names and emails in your PUT request, you can create a file like this:

[

{

"name": "Alice",

"email": "alice@example.com"

},

{

"name": "Bob",

"email": "bob@example.com"

},

{

"name": "Charlie",

"email": "charlie@example.com"

}

]

* Save the file with a .json extension and store it in a convenient location.
* In Postman, create a request with the PUT method and the URL of the API endpoint that you want to update. In the Body tab, select raw and JSON as the format. Use the same variable names as in your JSON file inside curly braces. For example:

{

"name": "{{name}}",

"email": "{{email}}"

}

* Save the request to a collection or create a new collection for it.
* Open the Collection Runner window by clicking on the arrow next to the collection name and then on the Run button.
* In the Collection Runner window, select the collection and the request that you want to run. Click on the Select File button under the Data section and choose the JSON file that you created earlier.
* Click on the Preview button to see how the data file values are mapped to the request body. You can also edit the iteration count, delay, and log options as per your preference.
* Click on the Run button to start the collection run. You can view the results in the Run Results tab or in the Console window.

The values that you put in the data file depend on the purpose of your test. If you want to test how the API handles different types of input, you can use **variable** values that cover various scenarios, such as valid, invalid, empty, or long values. For example:

[

{

"name": "Alice",

"email": "alice@example.com"

},

{

"name": "",

"email": "bob@example.com"

},

{

"name": "Charlie",

"email": "charlie@invalid"

},

{

"name": "David",

"email": "david@example.com"

},

{

"name": "Eve",

"email": "eve@example.com"

},

{

"name": "Frank",

"email": "frank@example.com"

},

{

"name": "This is a very long name that exceeds the maximum length allowed by the API",

"email": "george@example.com"

}

]

If you want to test how the API handles the same input repeatedly, you can use **random** values that are generated dynamically in the pre-request script. For example, you can use the faker library to generate random names and emails:

var faker = require('faker');

pm.environment.set("name", faker.name.findName());

pm.environment.set("email", faker.internet.email());

And then use the environment variables in the request body:

{

"name": "{{name}}",

"email": "{{email}}"

}