<https://learning.postman.com/docs/writing-scripts/intro-to-scripts/>

**Scripting in postman**

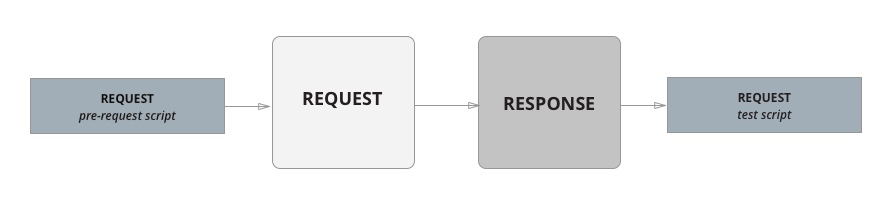
Pre-requests scripts

Test scripts

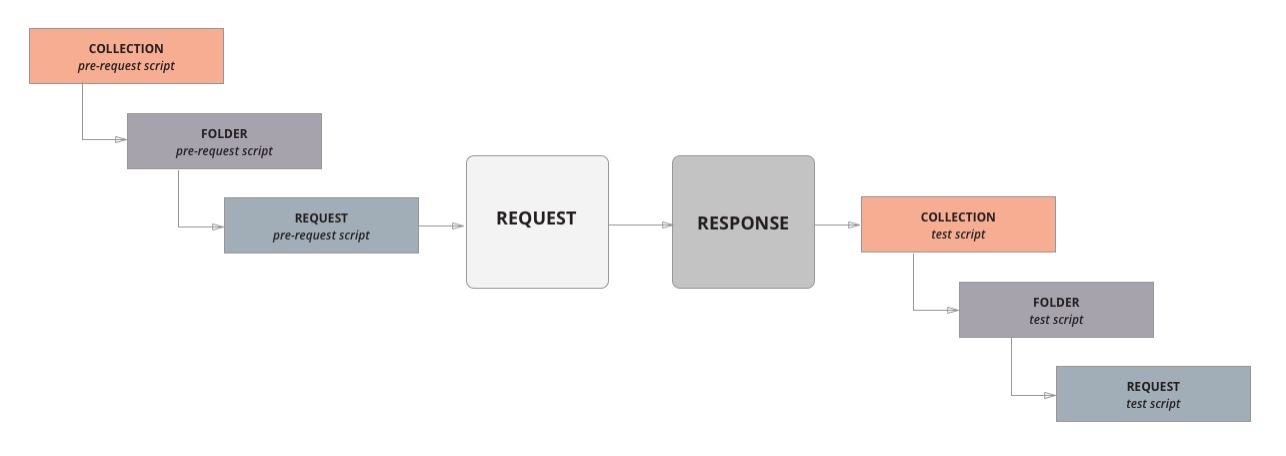
Above can be added to a collection, a folder, a request within a collection or a request not saved to a collection

**Execution order of scripts**

Execution order for a Single Request

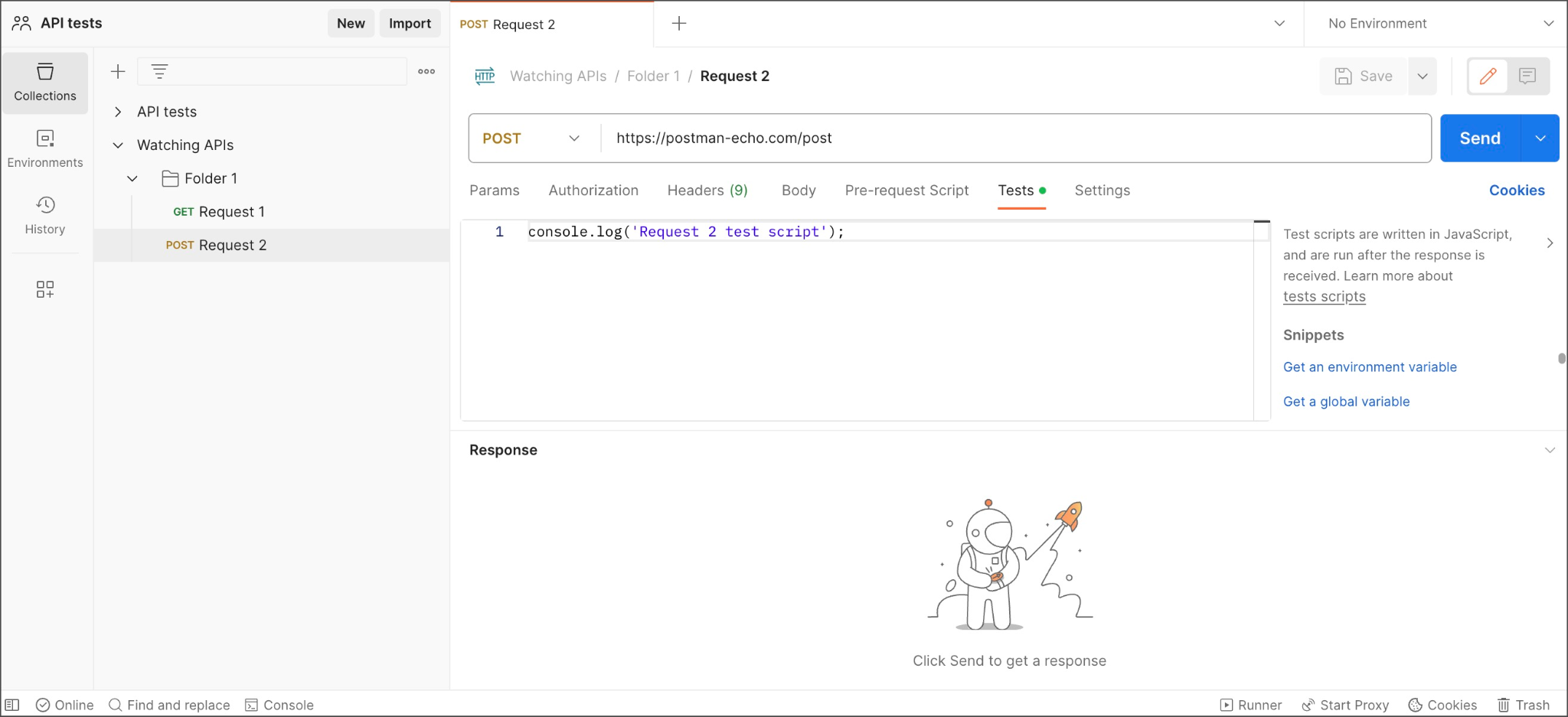


Execution order for every request in collection



For every request in a collection, the scripts will always run according to the same hierarchy. Collection-level scripts (if any) will run first, then folder-level scripts (if any), and then request-level scripts (if any). Note that this order of execution applies to both pre-request and test scripts.

For example, imagine you had the following collection structured with a single folder and two requests within the folder.

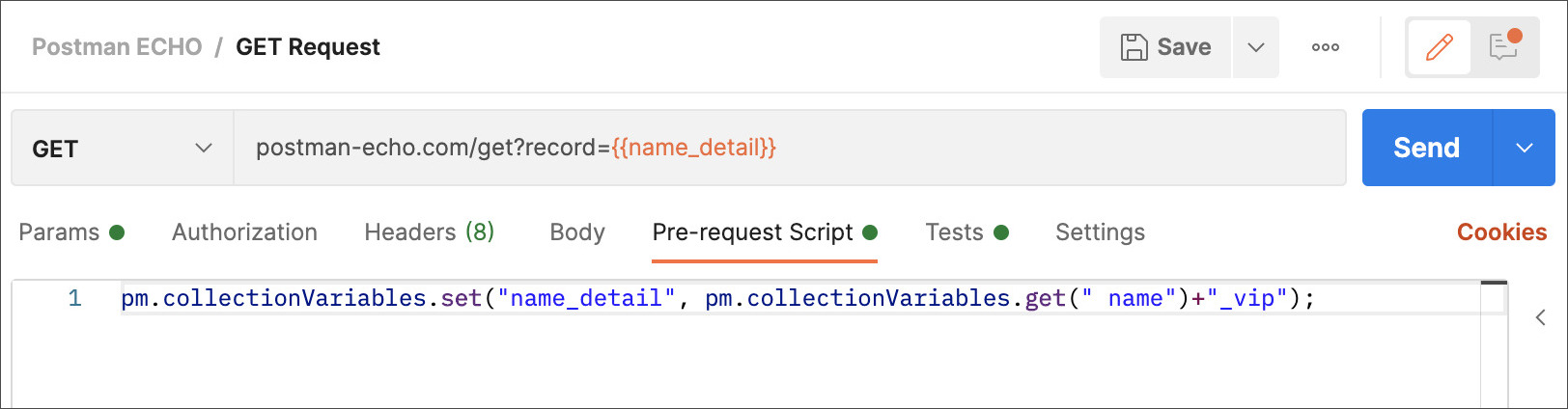


If you created log statements in the pre-request and test script sections for the collection, folder, and requests, the execution order is returned in the [Postman Console](https://learning.postman.com/docs/sending-requests/troubleshooting-api-requests/).

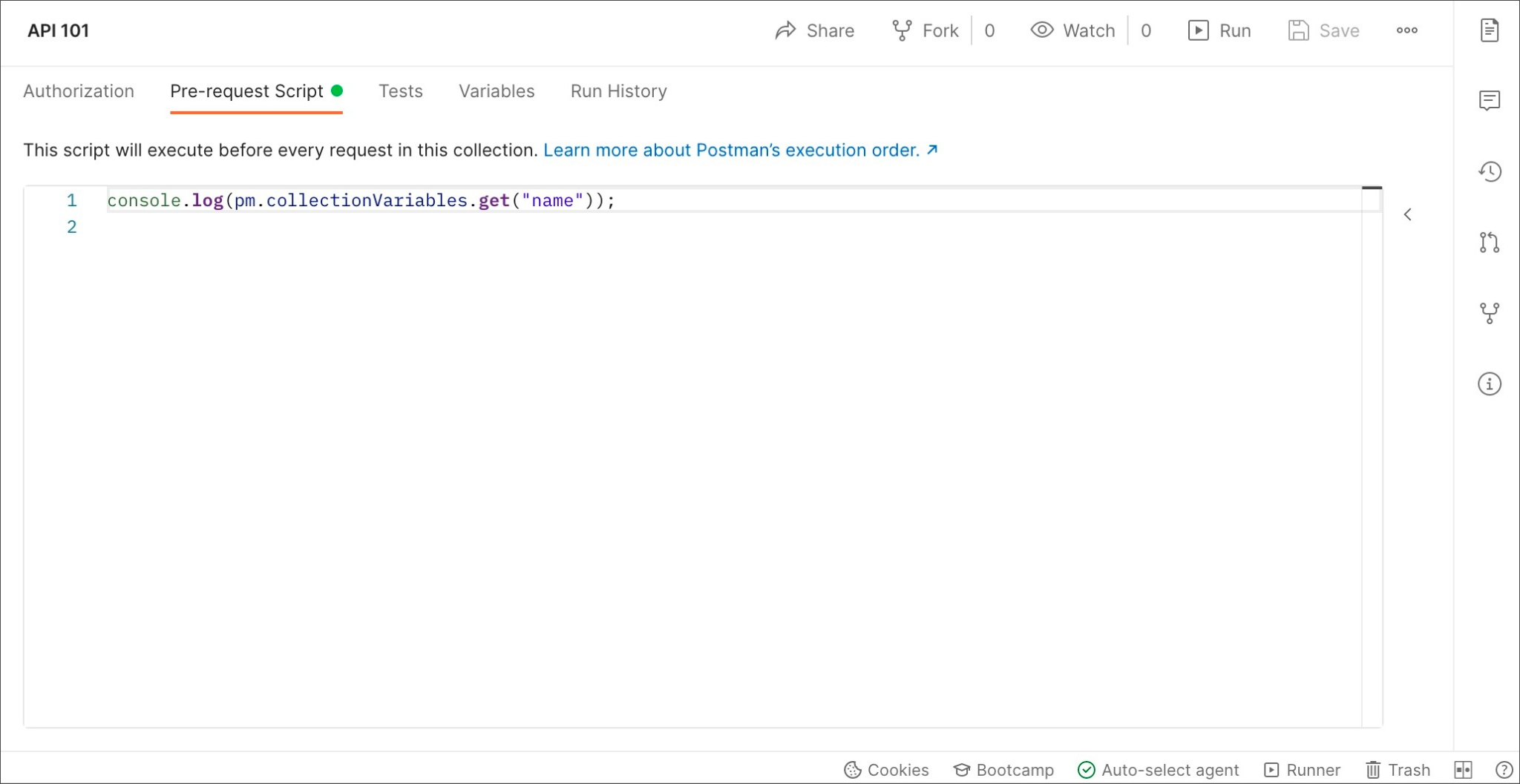


**Pre-request scripting example**

**Request before**



Collection or folder level



**Manage API data and workflos using postman javascript objects**

**The pm object**

Postman JavaScript API -

Object

pm.\*

* This provides access to request and response data and variables.
* Access and manipulate variables at each scope in postman using pm api
* Accessing global, collection, environment variables specifically and

**Variable scope:**

Variable scope determines the precedence postman gives to variables when you reference them

* Global
* Collection
* Environment
* Data
* Local

pm.variables.get(“name”)

pm.variables.set(“name”, ‘daniel’)

You can also access variables defined in the individual scopes with’

Pm.environment

pm.collectionVariables

Pm.globals

**Using environment variables in scripts**

Pm.environment.name

pm.environment.has(“variableName”)

pm.environment.get(“variableName”)

pm.environment.set(“variableName”, variableValue)

Remove a variable from the active environment

pm.environment.unset(“variableName”)

Clear all variables in the active environment

pm.environment.clear()

**Using collection variables in scripts**

pm.collectionVariables

pm.collectionVariables.has(“variableName”)

pm.collectionVariables.get(“variableName”)

pm.collectionVariables.set(“variableName”,variableValue)

Return the resolved value of a dynamic variable inside a script

Const stringWithVars =

pm.collectioncollectionVariables.replaceIn(“Hi, my name is {{firstname}} and I am ((age}}.”)

console.log(stringWithVars)

Return all variables with their values in the collection in an object

pm.colectioncollectionVariables.toObject()

pm.collectioncollectionVariables.unset(“variableName”)

pm.collectionVariables.clear()

**Using global variables in scripts**

Pm.globals

pm.globals

pm.globals.has(“variableName”)

pm.globals.get(“variableName”)

pm.globals.set(“variableName”,variableValue)

**Using data variables in scripts**

pm.iterationData

Method to access and manipulate variables from [data files during a collection run](https://learning.postman.com/docs/collections/running-collections/working-with-data-files/).

pm.iterationData.has(“variableName”)

pm.iterationData.get(“variableName”)

pm.iterationData.toObject()

pm.iterationData.toJSON()

pm.iterationData.unset(“key”)

**Scripting with request and response data**

Pm.request

Pm.response

Pm.info

Pm.cookies

Pm.sendrequest

**Scripting with request data**

Pm.request object provides access to the data for the request the script is running within.

For pre-request script, this is the request that is about to run, for a test script this is the request that has already run

The request URL

Pm.request.url

Pm.request.headers

Pm.request.method

Pm.request.body

Add a header with the specified name and value for the current request

pm.request.headers.add({

Key: “client-id”,

Value: “absde”

})

Delete the request header with the specified name

pm.request.headers.remov(“headerName”)

Insert or update header

pm.request.headers.upsert({

Key: “client-id”,

Value: “absde”

})

**Scripting with response data**

Pm.response.code

Pm.response.status

Pm.response.headers

pm.response.responseSize

pm.response.text()

pm.response.json()

**Writing test assertions**