POSTMAN

Testing, Automation & Reporting with Postman

PRESENTED BY

Danny Dainton - Senior Quality Engineer

Trent McCann - Engineering Manager, Quality

Sept 11, 2019

POSTMAN

Speakers

🇬🇧

Danny Dainton
Senior Quality Engineer

🇨🇦

Trent McCann
Engineering Manager, Quality

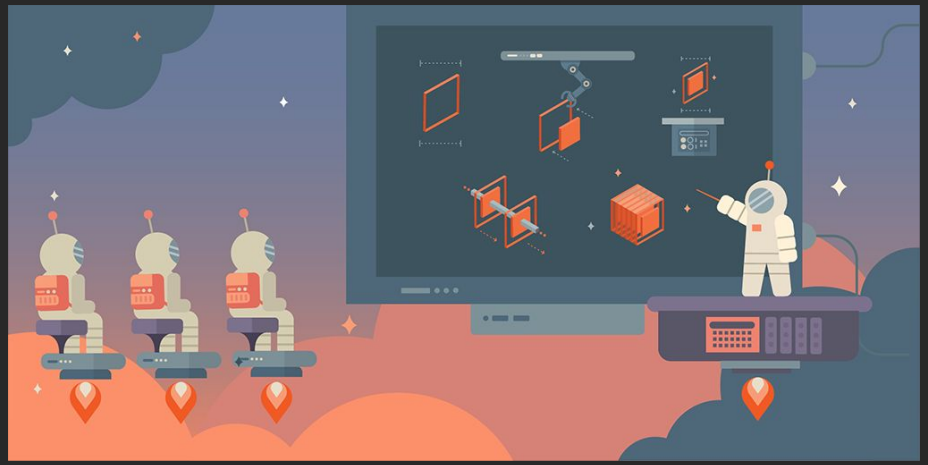
POSTMAN

Housekeeping

- Wifi - **PostmanWIFI** p/**POSTCON!**
- Washrooms / Beverages
- Facilitators
- Breaks

POSTMAN

Agenda



- Testing Module Part I
- Testing: Practical Exercises
- **Break**
- Testing Module Part II
- Testing: Practical Exercise
- **Break**
- Automation
- Reporting
- Workshop Wrap-up

POSTMAN

Prerequisites

POSTMAN

Import Collections

1. Navigate to bit.ly/POSTCON2019WORKSHOP
2. Copy the first Collection link and open up the Postman app
3. Select the Import button at the top of the app.
4. Select Import From Link tab
5. Paste the URL into the field
6. Click Import
7. Repeat for the second link

POSTMAN

Assumptions

This session assumes that attendees:

- Are comfortable making Requests in Postman
- Possess basic working knowledge of JavaScript
- Have the Latest Postman Desktop App Installed for your OS

POSTMAN

Testing Concepts

POSTMAN

Overview

POSTMAN

What the Hell is API Testing?

- People *generally* think that API testing is a form of software testing that you need to know about before getting started
- It is testing but in a different context
- You are still:
  - Using the same test skills and techniques but applying them differently
  - Focused on mitigating risk
  - Providing information about the software under test

POSTMAN

Why Write API Tests?

- Determine if the application under test meets expectations for **functionality**, **reliability**, **performance**, and **security**
- Ensure consistent high quality as your application grows
- Critical component for automating testing – APIs are the primary interface to application logic
- Tests at the API layer are less brittle and easier to maintain
- Most stable interface - GUI tests are not sufficient to validate all functional paths of a multi-tier architecture

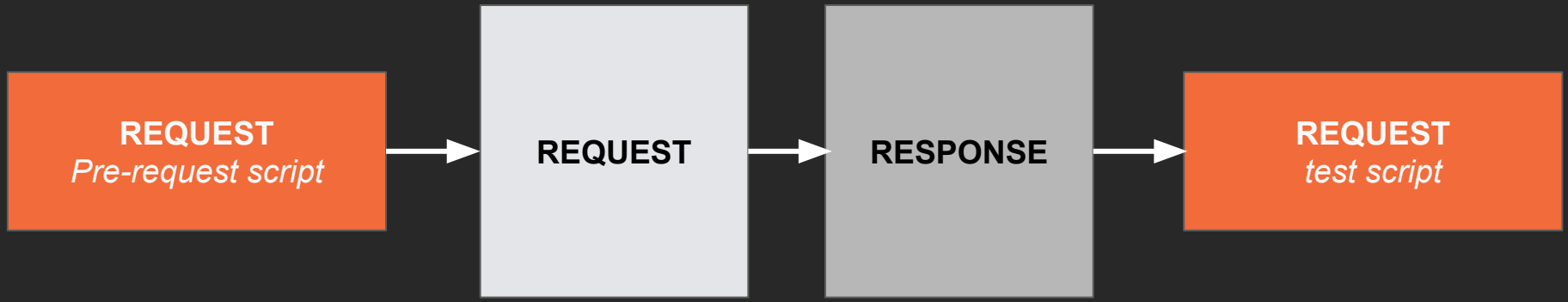
POSTMAN

Scripts

- Postman contains a powerful runtime based on Node.js that allows you to add dynamic behavior to requests and collections
- Allows you to write test suites, build requests that can contain dynamic parameters, pass data between requests and more
- You can add JavaScript code to execute during 2 events in the flow:
  - Before a request is sent to the server, under the *Pre-request Script* tab
  - After a response is received, as a test script under the *Tests* tab

POSTMAN

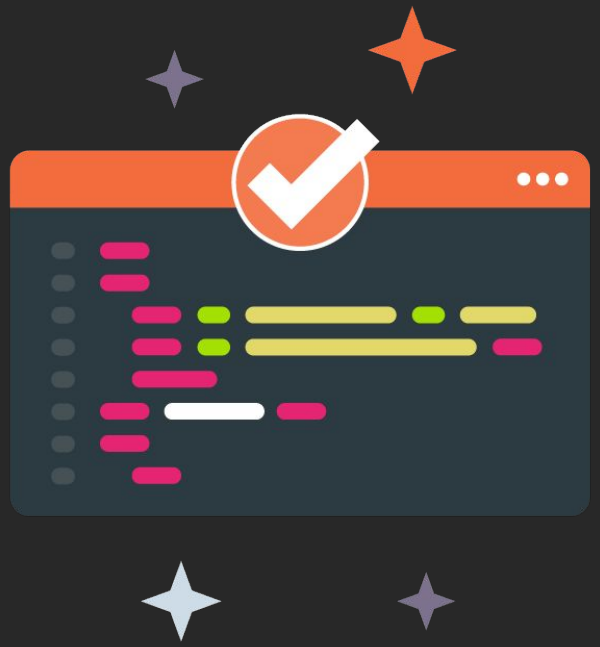
Execution Order of Scripts



- A *Pre-request Script* associated with a request will execute before the request is sent
- A *Test* script associated with a request will execute after the request is sent

POSTMAN

Collection and Folder Scripts



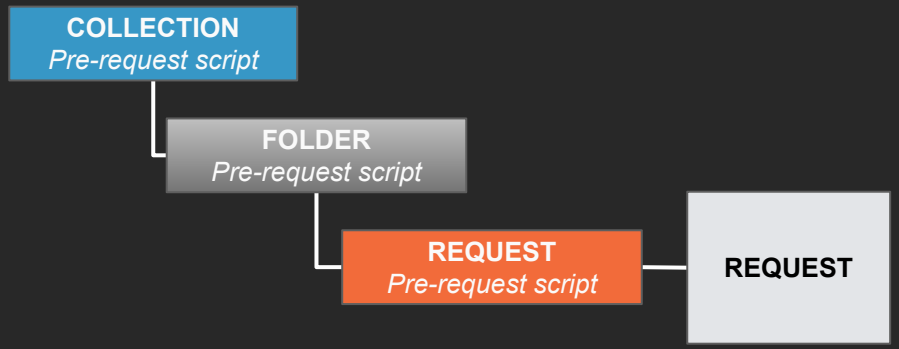
POSTMAN

Pre-Request Scripts

- Snippets of code associated with a collection request that are executed before the request is sent
- Example use cases:
  - Including the timestamp in the request headers *or*
  - Sending a random alphanumeric string in the URL parameters
- Can be added to a collection, a folder, or a single request within a collection
- Runs prior to each request in the respective collection or folder, allowing for commonly used code to be executed prior to every request

POSTMAN

Pre-Request Script Execution

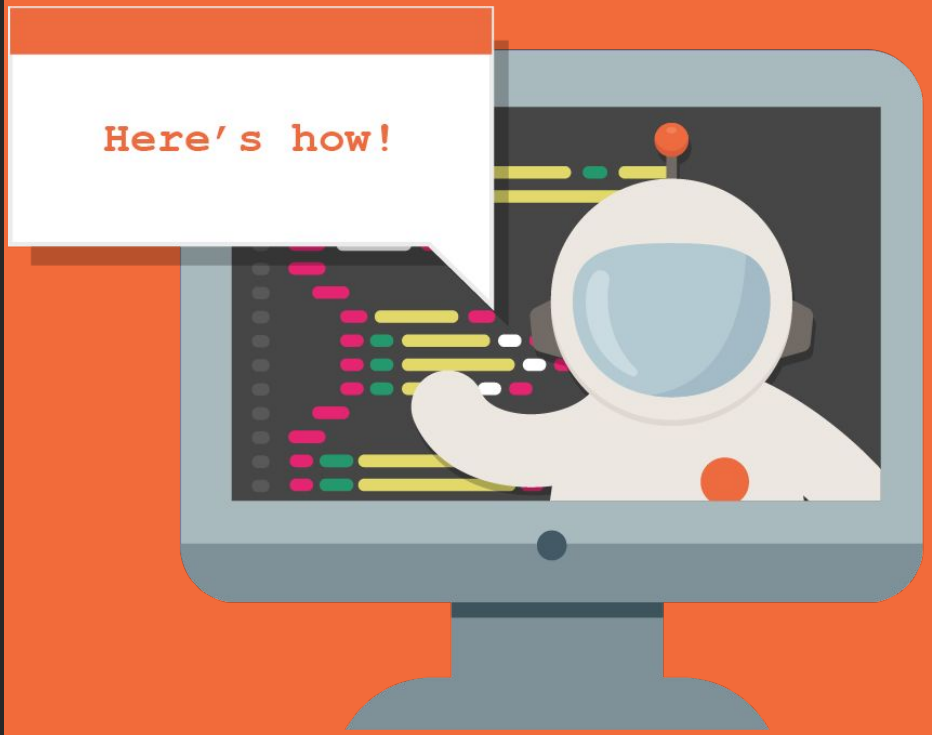


For every request in a collection, scripts will execute in the following order:

- A pre-request script associated with a collection or folder will run prior to every request in the respective collection or folder

POSTMAN

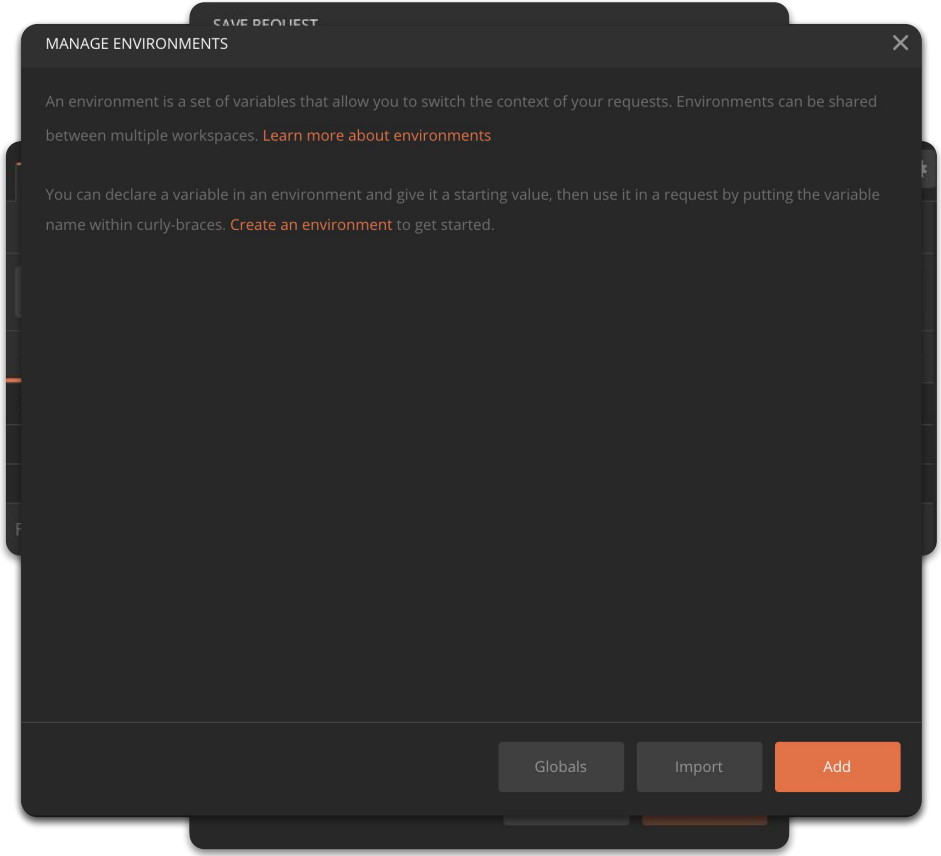
Practical Exercise: Pre-Request Scripts



POSTMAN

Steps

1. Click on the New button
2. Select Request
3. Enter a Request Name
4. Create Collection and enter name
5. Enter URL https://postman-echo.com/get and hit Send
6. Click on Gear Icon
7. Click on the Add button
8. Enter a name and click Add
9. Select Environment from Dropdown



POSTMAN

Steps

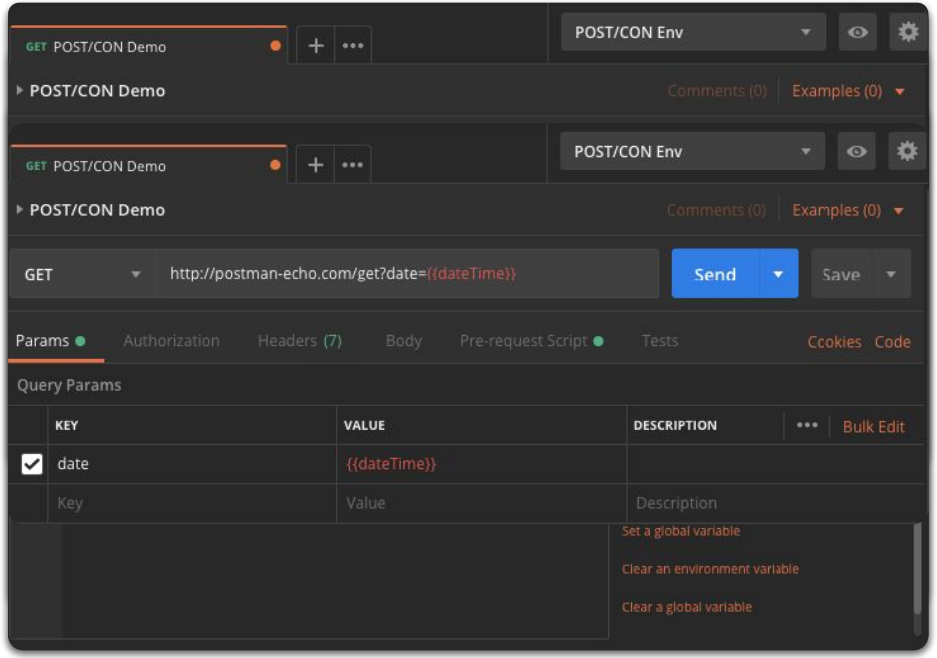
10. Select the Pre-request Script tab

11. Type:
pm.environment.set("dateTime", new Date());

12. Select the Params tab

13. Enter "***date***" for the key
and "***{{dateTime}}***" for the value.

14. Hit Send



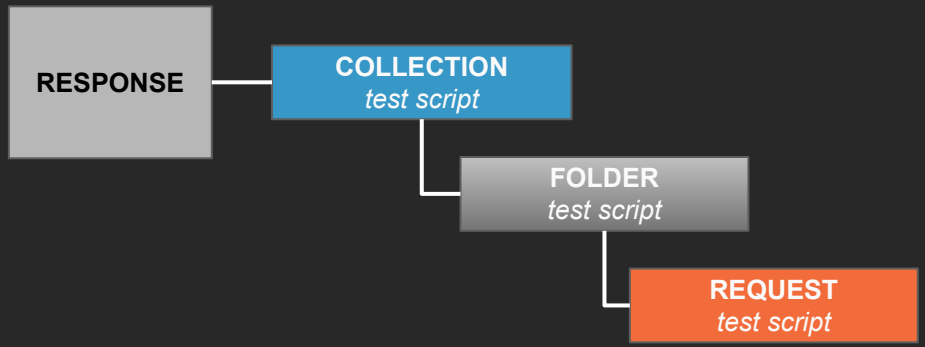
POSTMAN

Test Scripts

- JavaScript code executed **after** the request is sent, allowing access to the **pm.response** object.
- Tests run in a **sandboxed environment**, which is separate from the execution environment of the app
- Commonly used **snippets** are listed next to the editor and are a great way to build test cases
    - Select the snippet you want to add and the appropriate code will populate in the test editor

POSTMAN

Test Script Execution

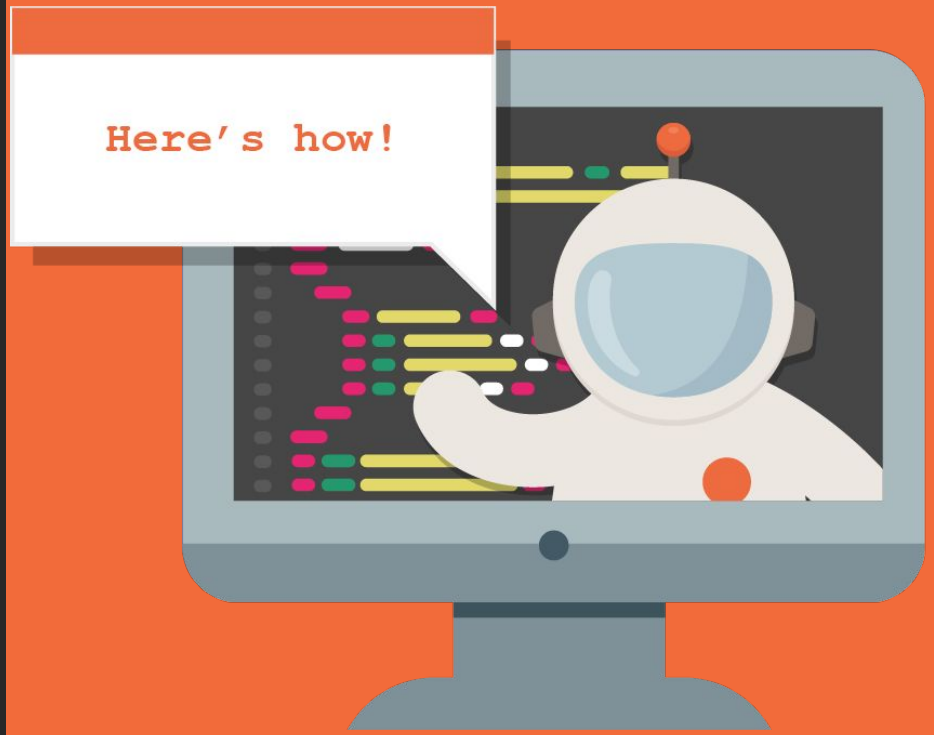


For every request in a collection, scripts will execute in the following order:

- A test script associated with a collection or folder will run after every request in the respective collection or folder

POSTMAN

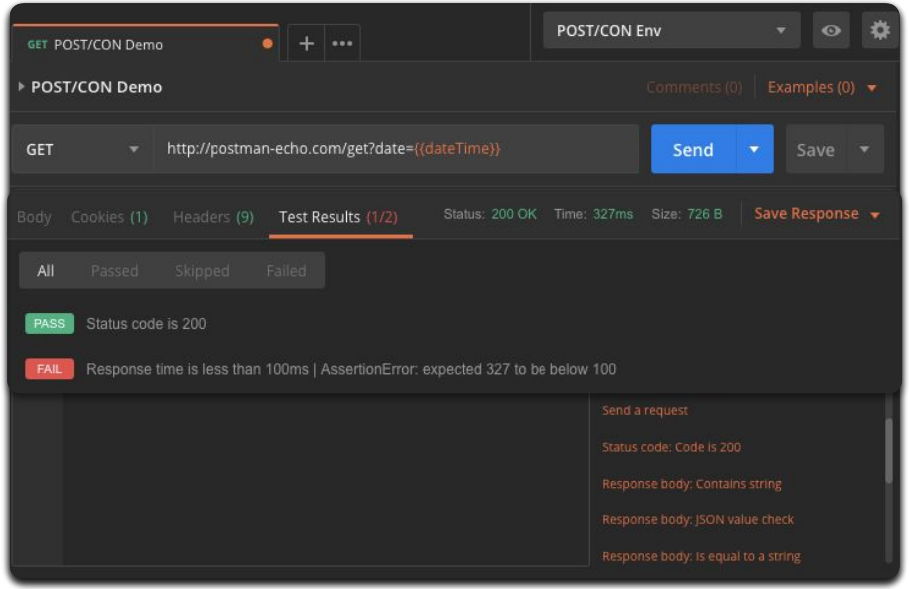
Practical Exercise: Test Scripts



POSTMAN

Steps

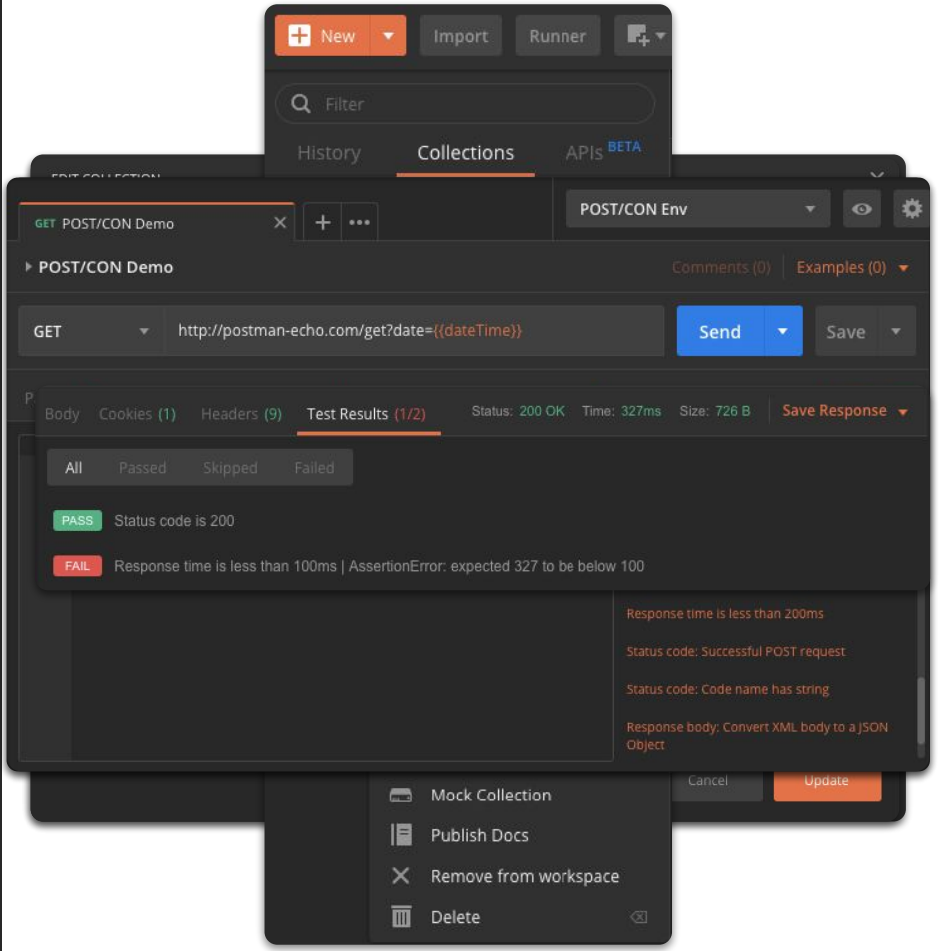
1. Select the Tests tab
2. Select Status Code: Code is 200 Snippet
3. Select Response Time is less than 200ms Snippet
4. Edit 200ms to 100ms
5. Hit Send
6. Select Test Results tab



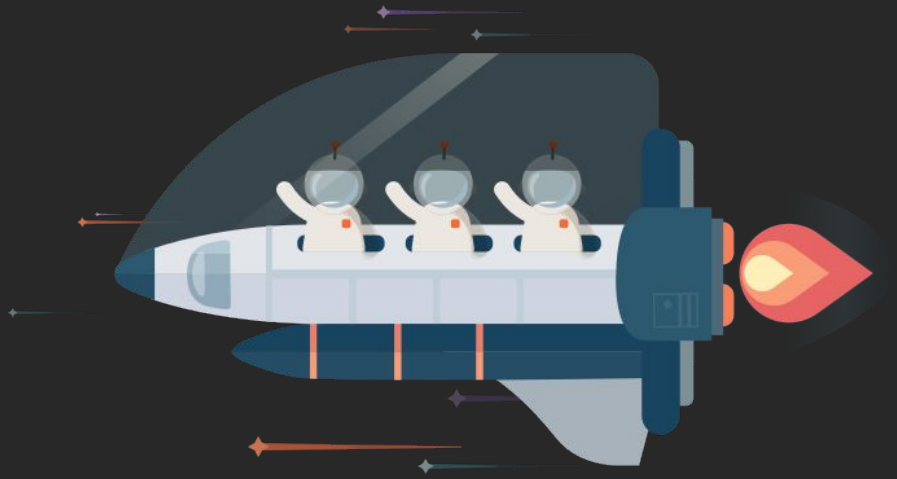
POSTMAN

Steps

7. Highlight and Cut the Tests from Request
8. Save Changes to the Request
9. Edit the Collection
10. Select Tests tab
11. Paste Tests and click Update
12. Hit Send



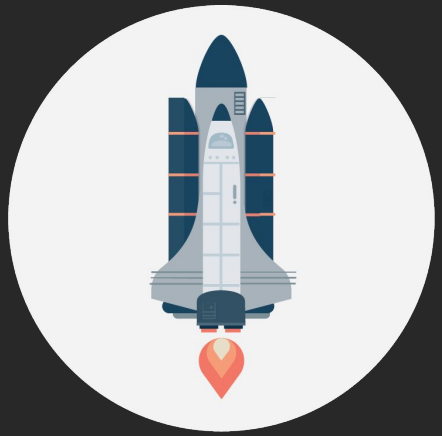
POSTMAN



Break Time

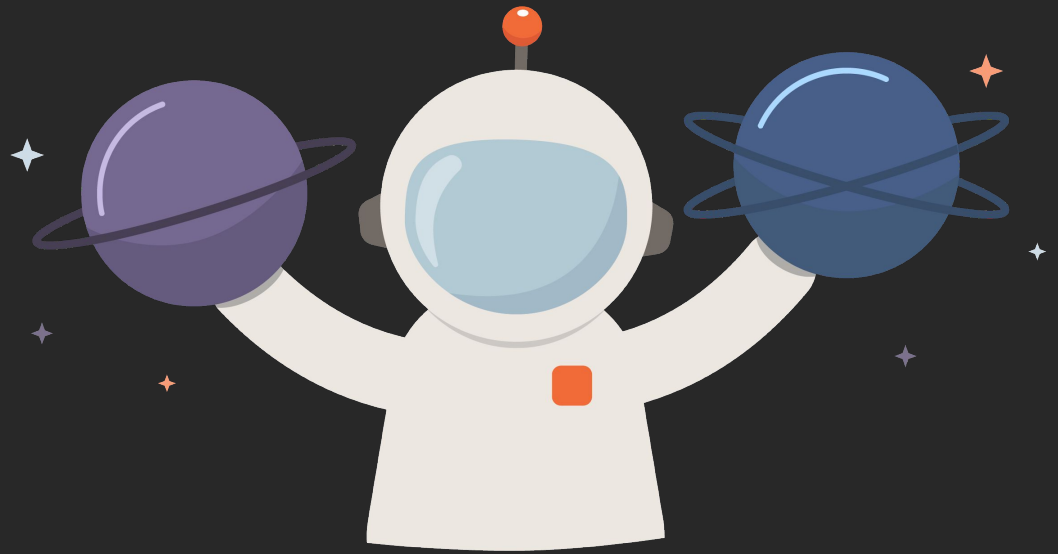
POSTMAN

Level Up Your Testing In Postman



POSTMAN

What do we mean by advanced techniques?



POSTMAN

Dynamic Variables

- Postman has dynamic variables that can be used in requests
- Values are randomly generated during the request/collection run
- Established variables **{{$randomInt}}**, **{{$timestamp}}** and **{{$guid}}** have been included for many versions
- Random fake data can be generated using a subset of the **FakerJS** functions
- Can used in the **URL, Headers**, **Request Body** and also within **Mock Server Examples**

POSTMAN

Chai Assertions

- Using the popular and well known **Chai.js** library
- Chai provides natural language assertions in an expressive and readable style
- Chained together by a series of **getters**
- **be** / **been** / **is** / **and** / **has** / **have** / **not**
- Works with the **pm.expect()** function
- The assertion always starts with the **.to** getter

POSTMAN

Built-In Libraries

- **Lodash**: JS utility library
- **Moment**: Parse, validate, manipulate, and display dates and times in JavaScript.
- **Cheerio**: A fast, lean implementation of the core jQuery API
- **Ajv**: JSON schema validator.
- **CryptoJS**: Standard and secure cryptographic algorithms.
- **xml2Json**: Simple XML to JSON Convertor.

POSTMAN

Postman Console

- The Postman Console can be compared to a browser's Developer console, but tuned for API development
- When debug scripts are written under the Pre-request Script or Test tabs, helpful messages will be logged in the Postman Console
- If an API or API test is not behaving as expected, the console will assist with debugging

POSTMAN

What Does it Do?

- Logs the request that was sent, including all the underlying **request headers**, **status code** etc.
- Logs the exact **response** sent by the server
- Displays what **proxy** and **certificates** were used for the request
- Shows error logs from **Test** or **Pre-request Scripts**
- Allows the **console.log()** statement to be used from inside sandbox

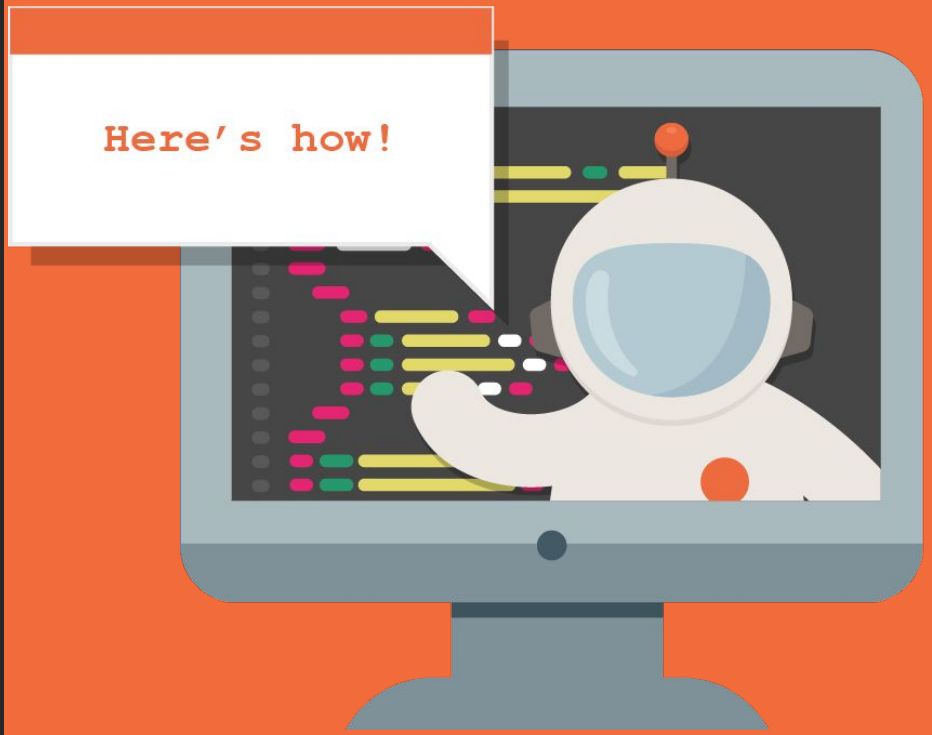
POSTMAN

Console.log(); Your New Best Friend

- JS method used to write messages to the debug console
- Can be used for debugging your test scripts and capturing key information from the test runs

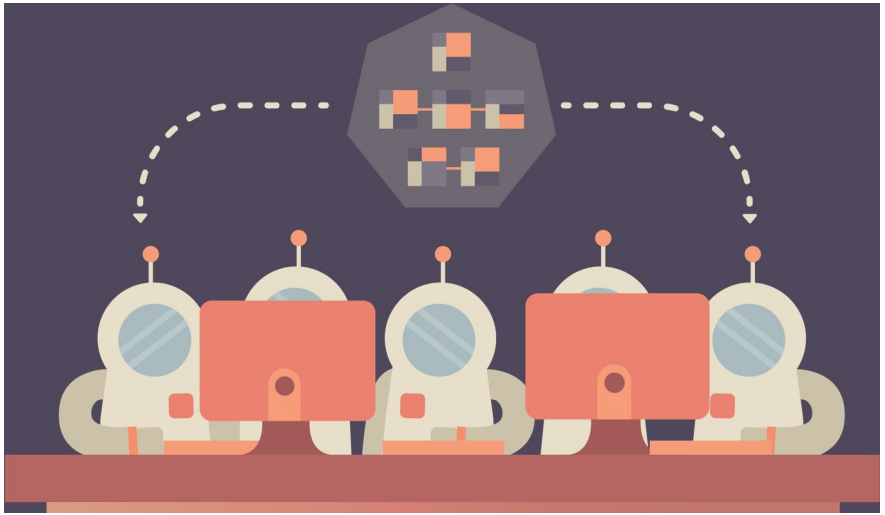
POSTMAN

Practical Exercise: Advanced Techniques

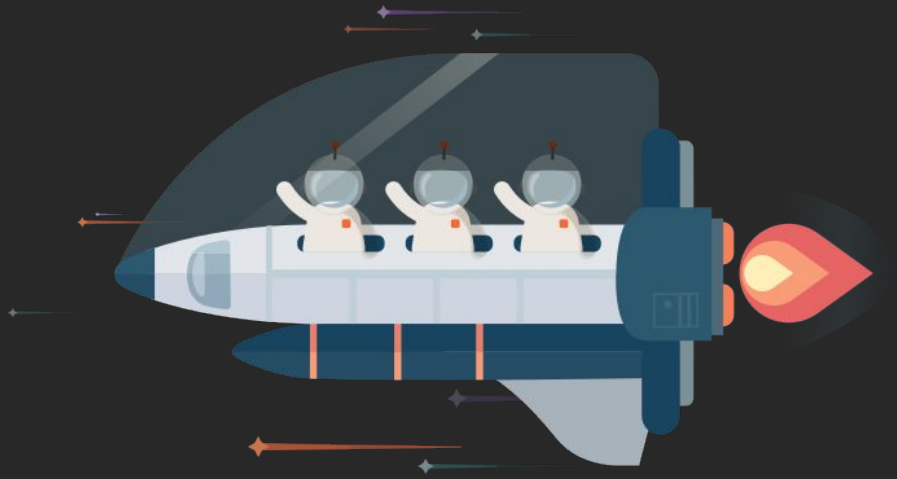


POSTMAN

Module Review



POSTMAN



Break Time

POSTMAN

Automation Made Easy

POSTMAN

Overview

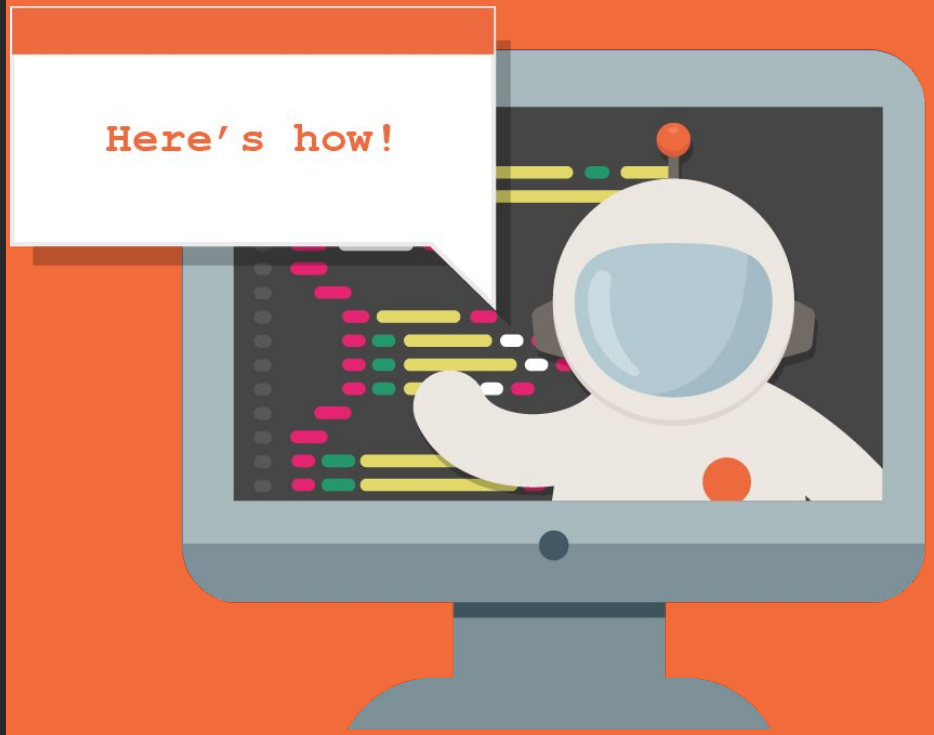
POSTMAN

Collection Runner

- Collections are groups of requests that can be run together as a series, against a corresponding environment
- Enables automation of API test collections
- When a collection is run, all requests in the collection will be sent one after another
- Using scripts allows you to:
  - Build integration test suites
  - Pass data between API requests
  - Build workflows that mirror API use cases

POSTMAN

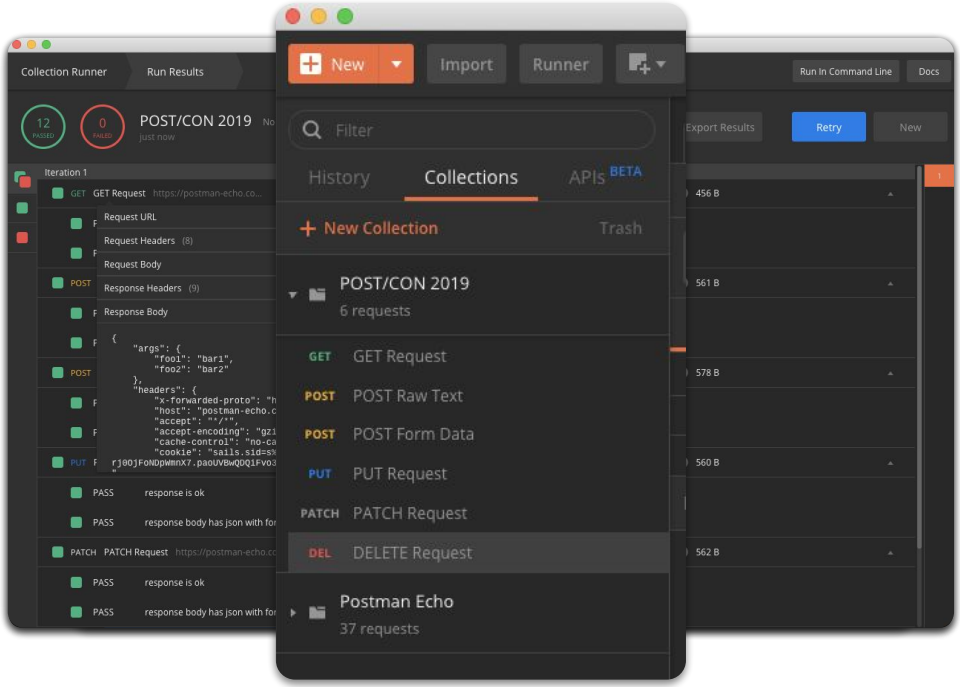
Practical Exercise: Collection Runner



POSTMAN

Steps

1. Navigate to the Collections tab
2. Click on the Navigation arrow and hit Run
3. Click on the Run "*collection name*" button
4. Run Results
5. Click on request name



POSTMAN

Meet Newman

POSTMAN

What is Newman?

- A globally installed NPM package
- A Collection Runner for Postman that runs directly from the command line
- Easily integrates with continuous integration servers and build systems
- Maintains feature parity with the Postman App
- Allows Collections to be run the way they are executed in the Postman app collection runner
- Built as a library - can be extended and used flexibly
- Collection run reports can be generated using different custom reporters

POSTMAN

newman run -h

```
~ newman run -h
Usage: run <collection> [options]

URL or path to a Postman Collection.

Options:
  -e, --environment <path>        Specify a URL or Path to a Postman Environment.
  -g, --globals <path>            Specify a URL or Path to a file containing Postman Globals.
  --folder <path>                 Specify folder to run from a collection. Can be specified multiple times to run multiple folders (default: [])
  --working-dir <path>            The path of the directory to be used as the working directory
  --no-insecure-file-read         Prevents reading the files situated outside of the working directory
  -r, --reporters [reporters]     Specify the reporters to use for this run. (default: ["cli"])
  -n, --iteration-count <n>       Define the number of iterations to run.
  -d, --iteration-data <path>     Specify a data file to use for iterations (either json or csv).
  --export-environment <path>     Exports the environment to a file after completing the run.
  --export-globals <path>         Specify an output file to dump Globals before exiting.
  --export-collection <path>      Specify an output file to save the executed collection
  --postman-api-key <apiKey>      API Key used to load the resources from the Postman API.
  --delay-request [n]             Specify the extent of delay between requests (milliseconds) (default: 0)
  --bail [modifiers]              Specify whether or not to gracefully stop a collection run on encountering an errorand whether to end the run with an error based on
the optional modifier.
  -x , --suppress-exit-code       Specify whether or not to override the default exit code for the current run.
  --silent                        Prevents newman from showing output to CLI.
  --disable-unicode               Forces unicode compliant symbols to be replaced by their plain text equivalents
  --global-var <value>            Allows the specification of global variables via the command line, in a key=value format (default: [])
  --env-var <value>               Allows the specification of environment variables via the command line, in a key=value format (default: [])
  --color <value>                 Enable/Disable colored output. (auto|on|off) (default: "auto")
  --timeout [n]                   Specify a timeout for collection run (in milliseconds) (default: 0)
  --timeout-request [n]           Specify a timeout for requests (in milliseconds). (default: 0)
  --timeout-script [n]            Specify a timeout for script (in milliseconds). (default: 0)
  --ignore-redirects              If present, Newman will not follow HTTP Redirects.
  -k, --insecure                  Disables SSL validations.
  --ssl-client-cert <path>        Specify the path to the Client SSL certificate. Supports .cert and .pfx files.
  --ssl-client-key <path>         Specify the path to the Client SSL key (not needed for .pfx files)
  --ssl-client-passphrase <path>  Specify the Client SSL passphrase (optional, needed for passphrase protected keys).
  --verbose                       Show detailed information of collection run and each request sent
```

POSTMAN

Newman In Action

POSTMAN

```
~ newman run Postman_Echo.json
```

POSTMAN

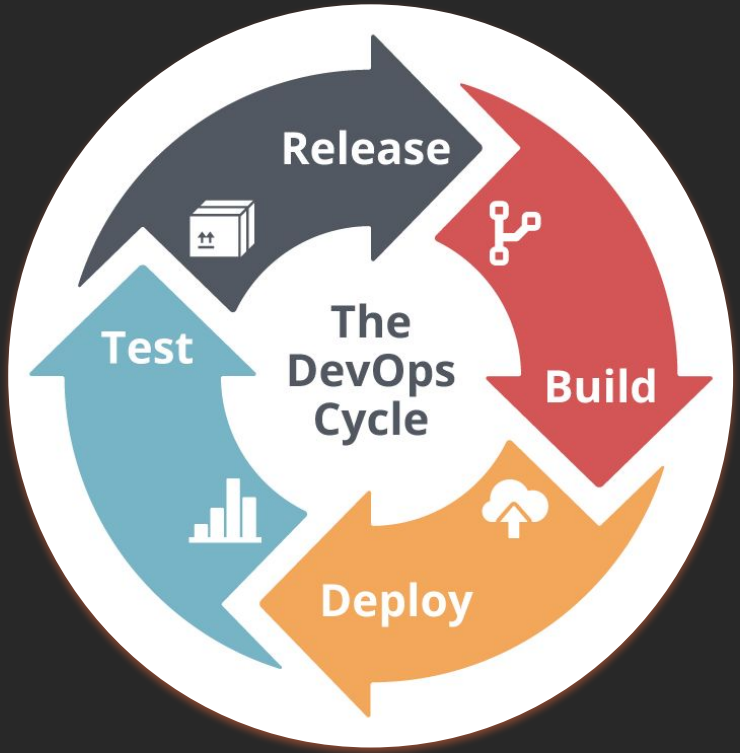
```
~ newman run https://www.getpostman.com/collections/f416b9eb00dbb1a2345a
```

POSTMAN


~ newman run https://www.getpostman.com/collections/ccd909c6a114a2a8738a --env-var baseURL=https://postman-echo.com

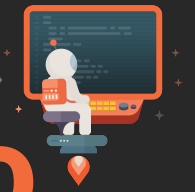
POSTMAN

CI/CD

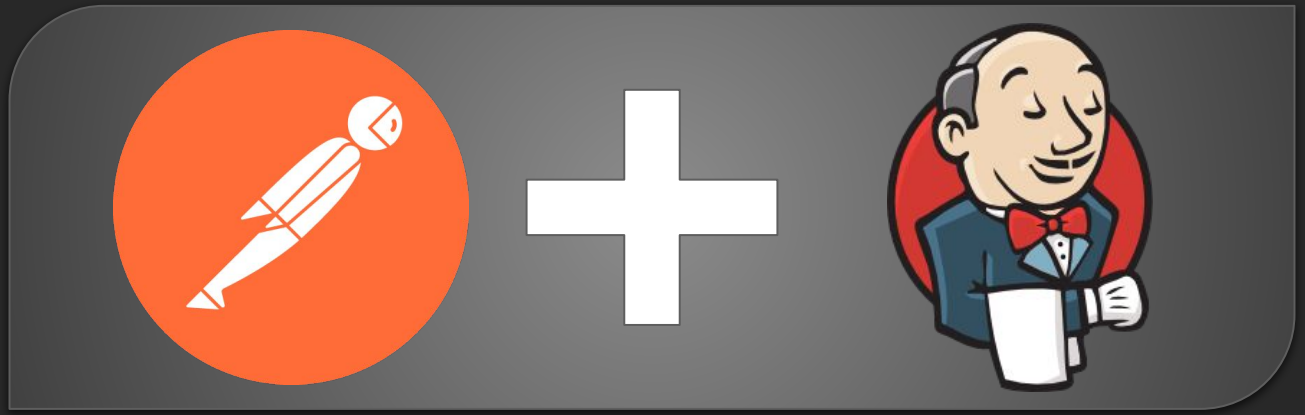


- Newman can run on popular CI/CD frameworks such as Jenkins, Travis, CircleCI and even Azure (via VSTS).
- Provides a streamlined way to run Newman in a Docker container without operating system dependencies or environment setup

POSTMAN

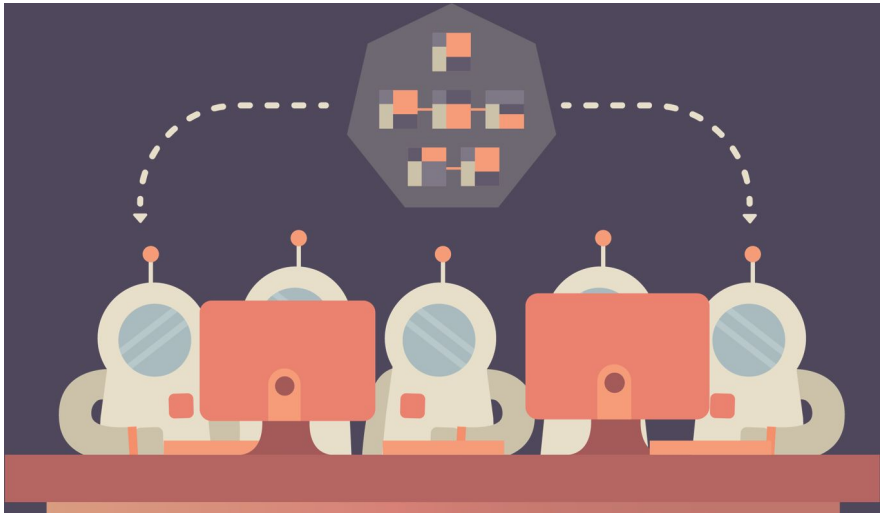


Newman + Jenkins Demo

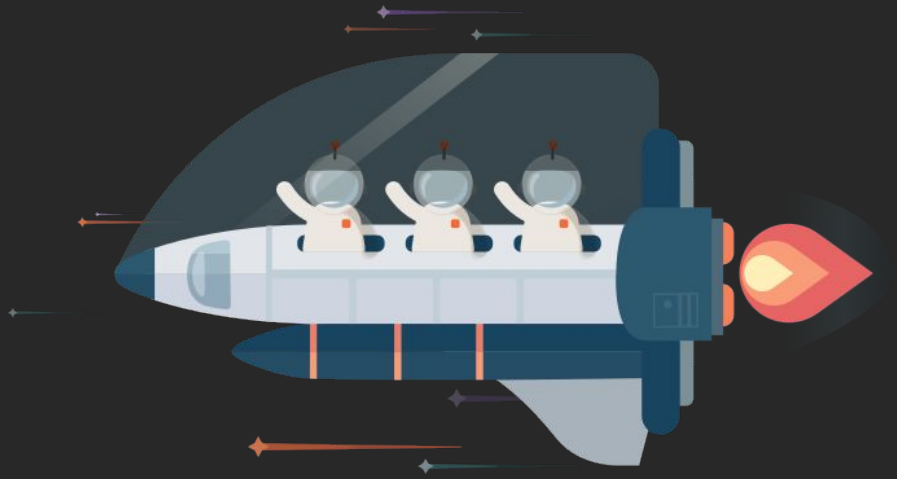


POSTMAN

Module Review



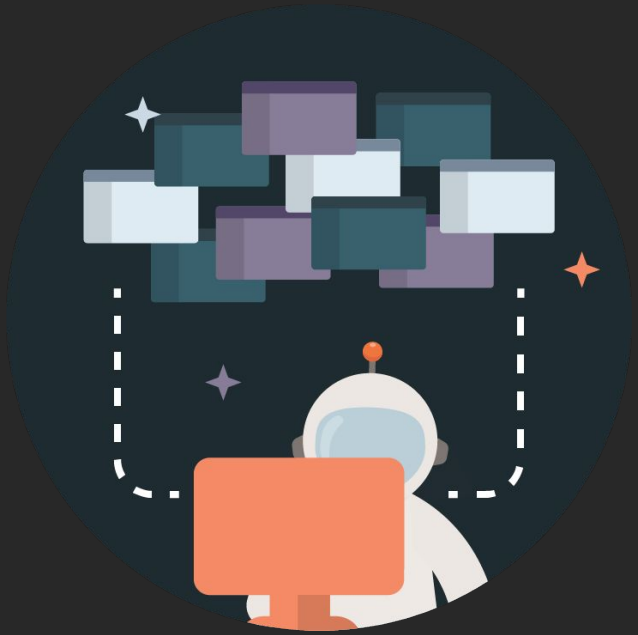
POSTMAN



Break Time

POSTMAN

Reporting: “And that’s the way it is.”



POSTMAN

Overview

POSTMAN

Why Do We Create Reports?

Reports are generally created to **communicate information** to some person, The type of report and the details about what it should contain, will differ from person to person.

Things to consider when creating reports:

- The report's **intended audience**
- The report's **purpose**
- The **type of information** to be communicated

POSTMAN

Reporters Created By Postman

POSTMAN

Standard Newman Reporters

CLI

This is the **default** reporter that comes with Newman and is the standard output that you will see. There are a number of different configurable options/cli arguments available to present that out in specific ways.

JUNIT

This reporter will create an XML file containing the all high level test result data, this can be picked by CI tools like Jenkins to display a graphical representation of the overall test coverage

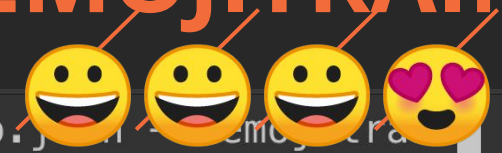
POSTMAN

PROGRESS [========]

```
~ newman run Postman_Echo.json -r progress
```

POSTMAN

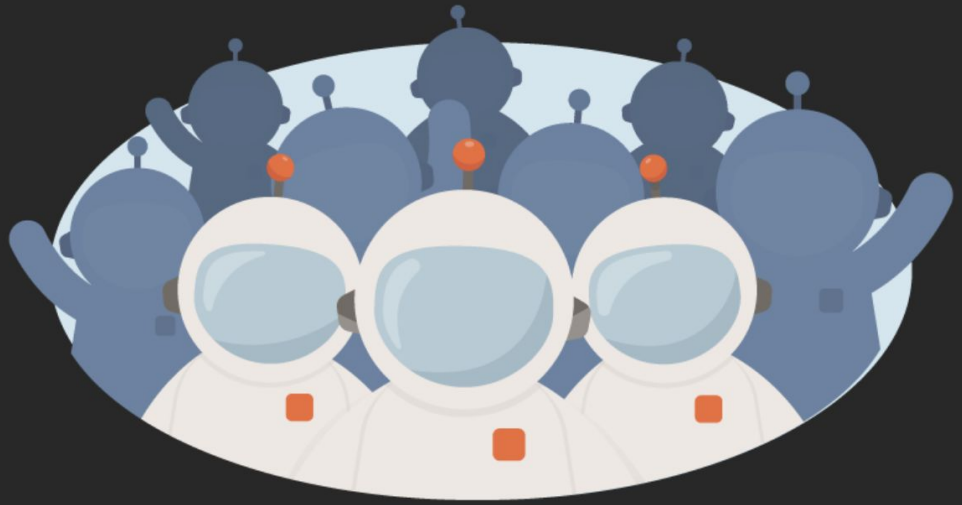
EMOJITRAIN



```
~ newman run Postman_Echo.j
```

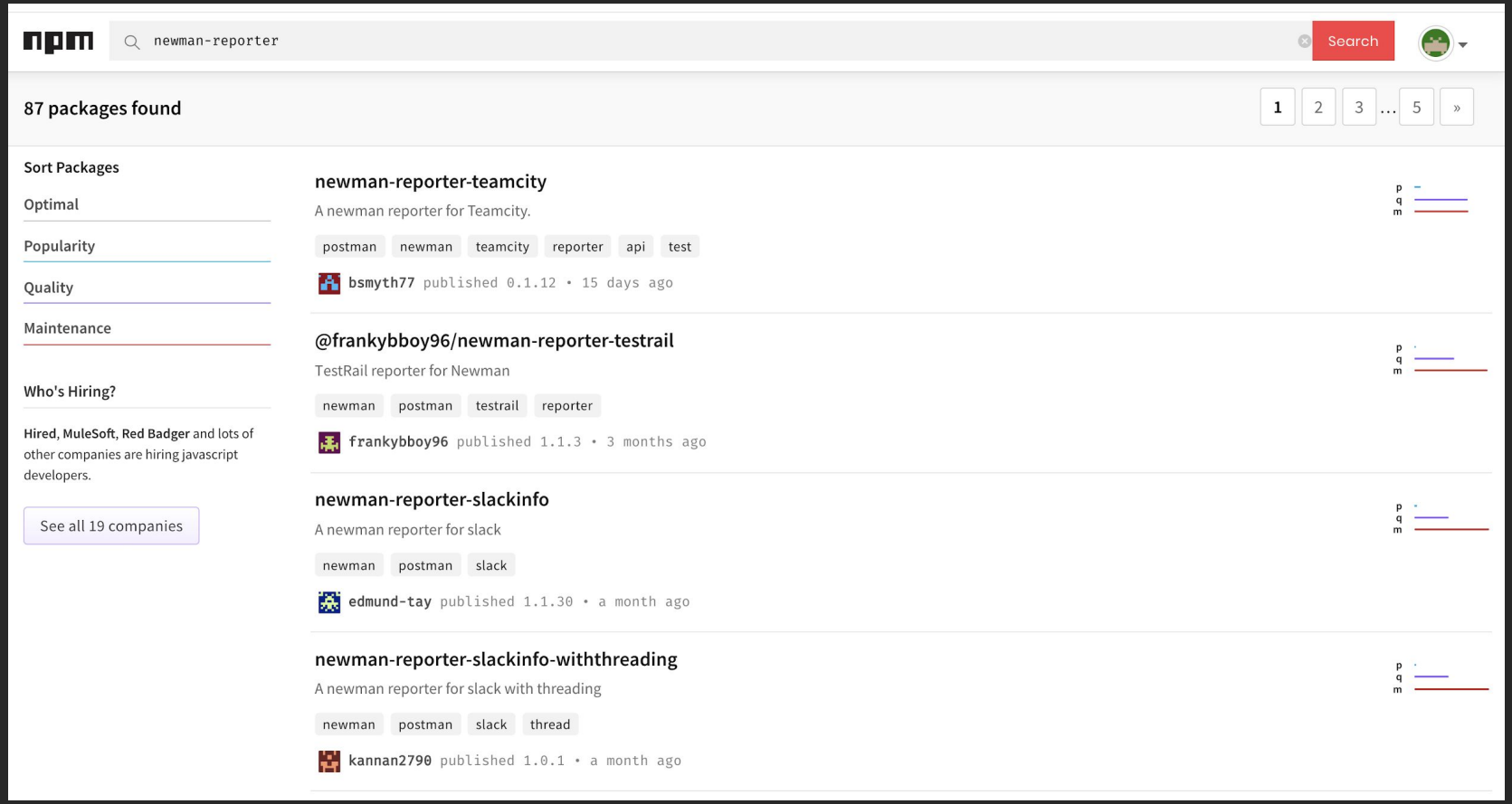
POSTMAN

Reporters Created By The Postman Community



POSTMAN

NPM Packages



POSTMAN

StatsD

POSTMAN

Slack

Newman Slack APP 11:13 AM

POST/CON 2019

	total	failed
iterations	1	0
requests	12	0
testScripts	24	0
prerequestScripts	14	0
assertions	49	0
total run duration	2.4s	

POSTMAN

A new take on a modern classic...

Postman HTML Report

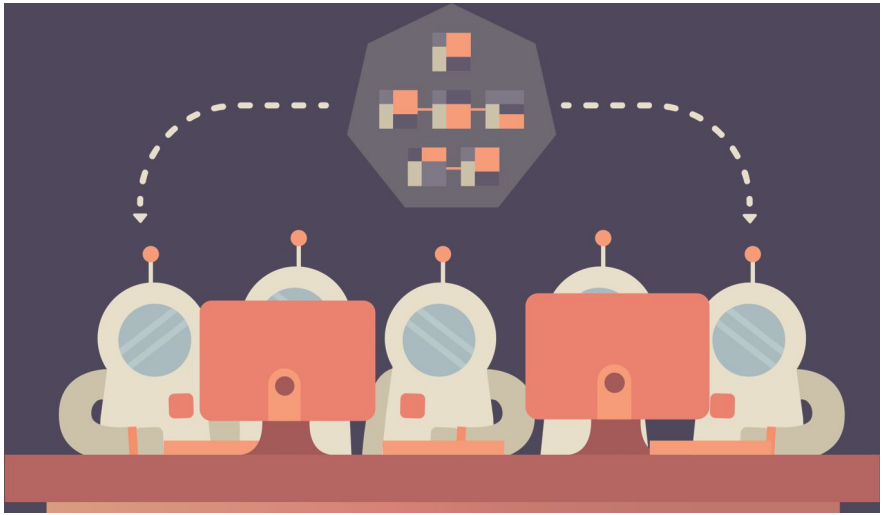
bit.ly/newmanhtml

HTMLEXTRA Report

bit.ly/newmanhtmlextra

POSTMAN

Module Review



POSTMAN

Review

- Testing within Postman can be an effective way to give you confidence that your API endpoints are returning the correct information without having to manually confirm this on each request
- Postman is packed full of different helpers, libraries and utilities that can enable you to create more complex test suites
- Automating a Collection within the App is super simple using the Built In Runner
- You can use Newman to run your Collections from either the Command Line or easily integrate into your CI systems

POSTMAN

Recommended Resources

- Postman Community - https://community.getpostman.com
- Learning Centre - https://learning.getpostman.com
- GitHub Repo - https://github.com/postmanlabs/postman-app-support
- Support Portal - https://support.getpostman.com/hc

Keep In Touch!

Trent


ca/linkedin.com/in/trentmccann


@Postman_Trent

Danny


uk/linkedin.com/in/dannydainton


@DannyDainton

POSTMAN

Workshop Resources

bit.ly/POSTCON2019WORKSHOP

POSTMAN

Thank You and Goodbye