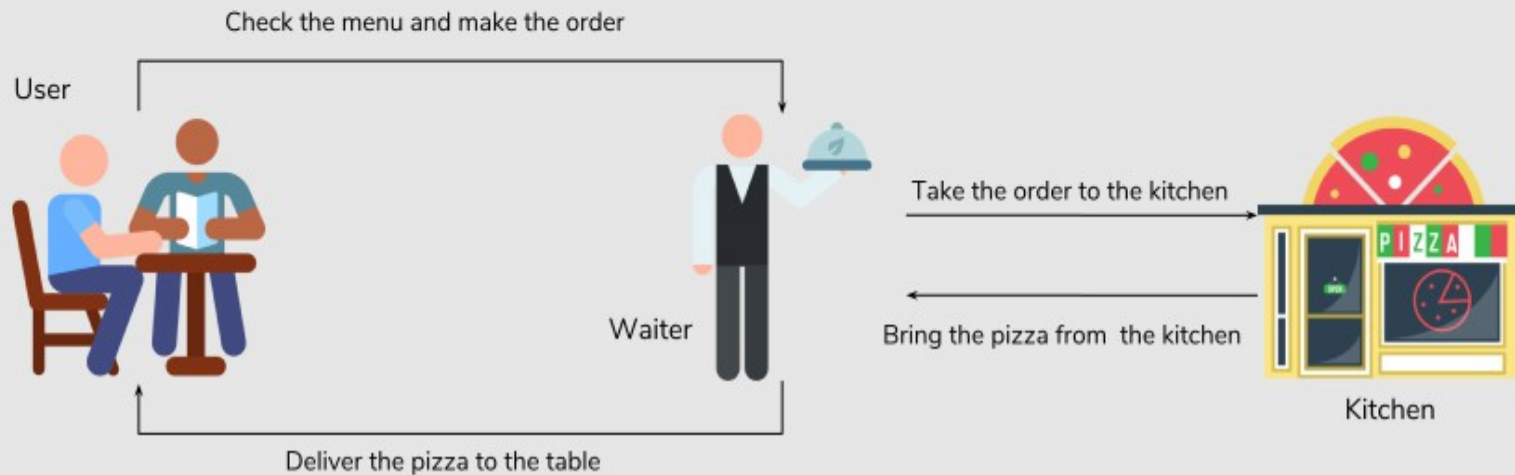
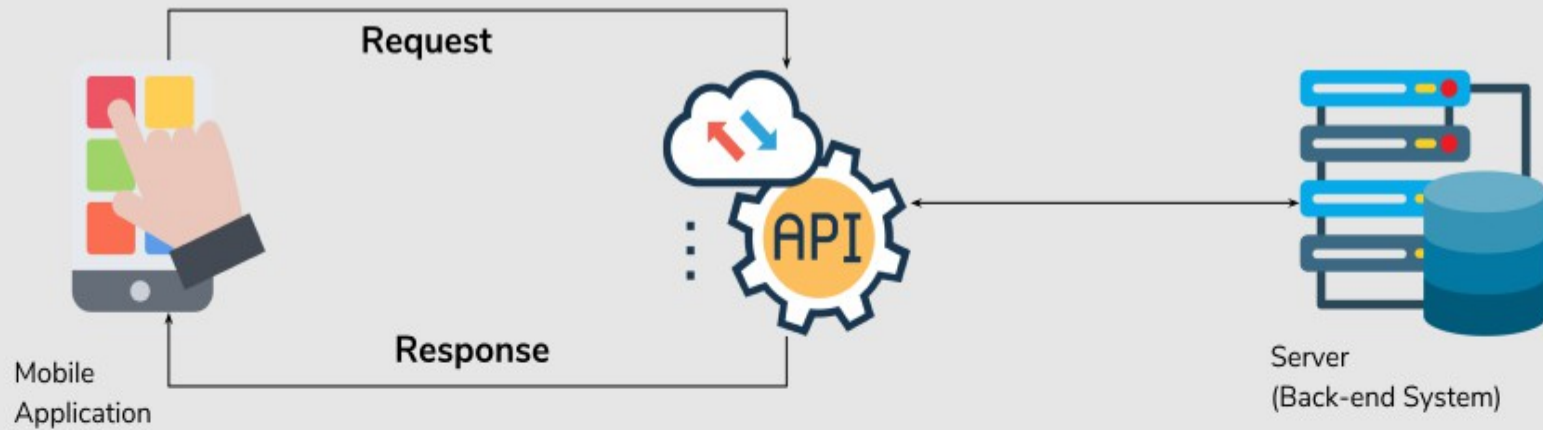
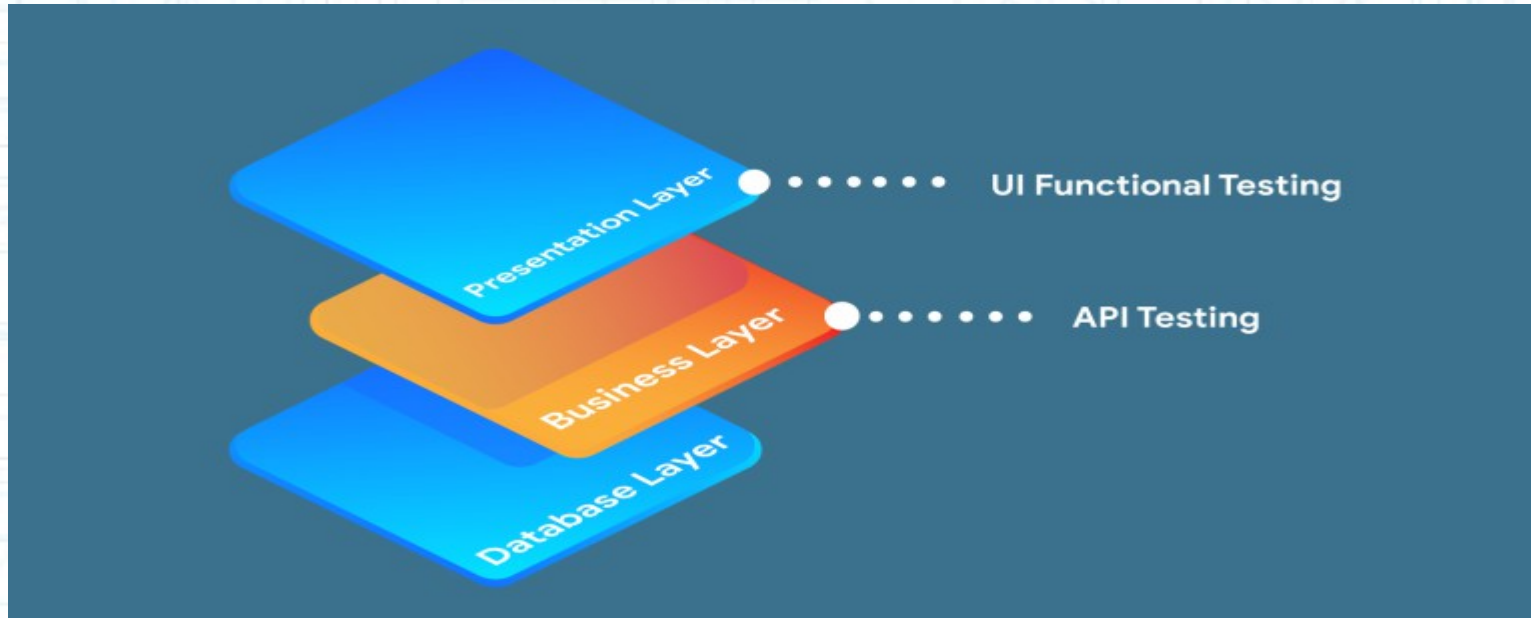


# What is API?



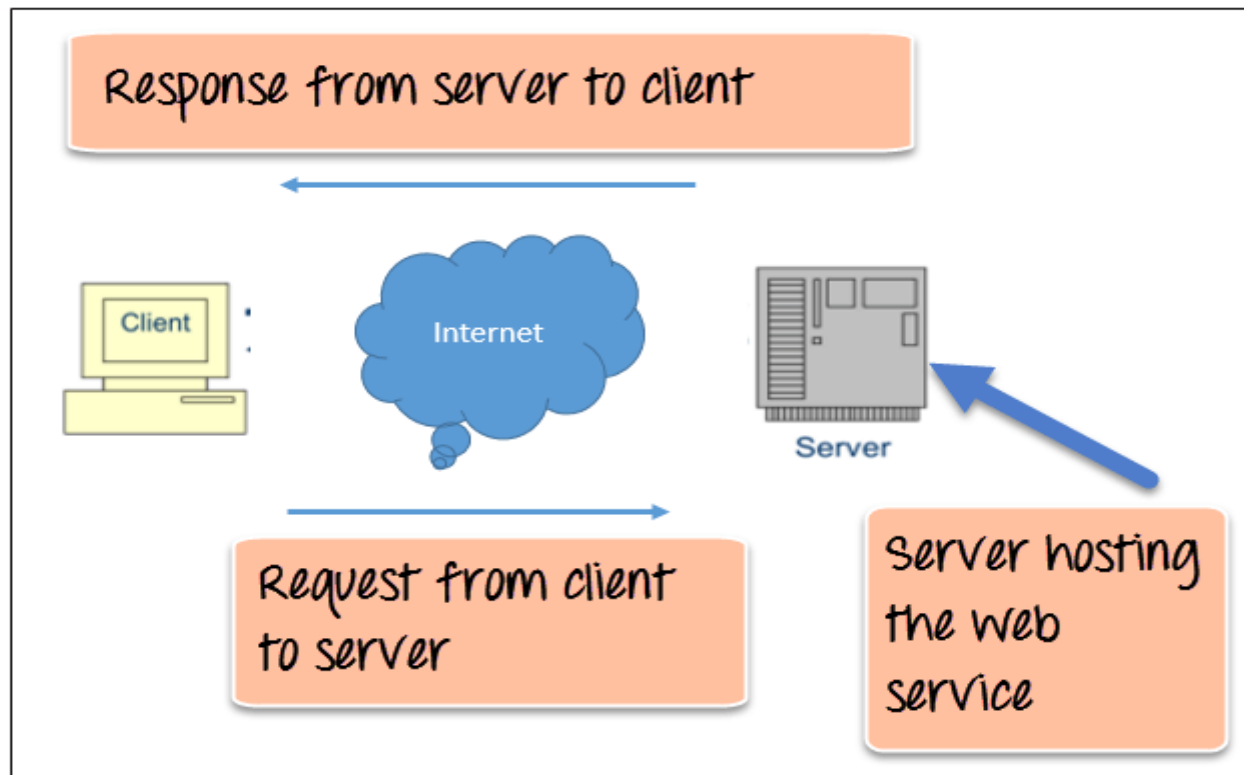
# Defination of API

- API is nothing but Application Programming Interface
- It is a interface that allows two applications to interact with each other without any user intervention.



# WebServices

- Web service is a standardized medium to provide communication between the client and server applications over the internet
- Example Travel ticketing booking





# Difference between API and Webservices

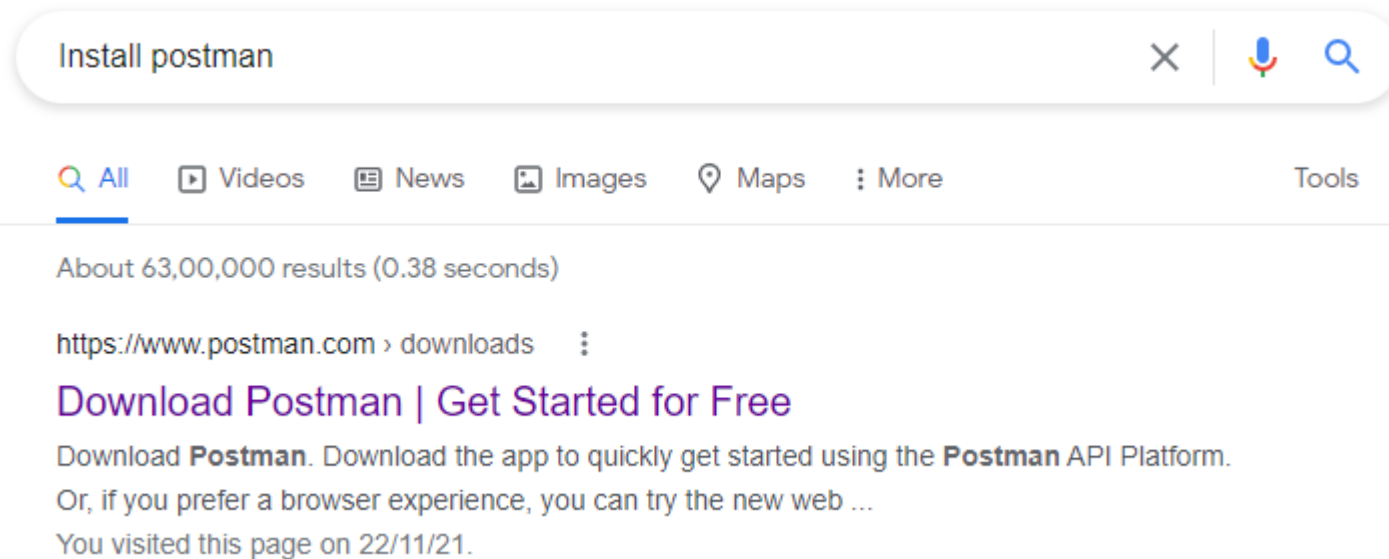
- Webservices
  - All webservices are API's
  - It support only SOAP, REST and XML-RPC
  - Web Service requires a network to work
- API
  - All API's are not webservices
  - It support XML and JSON or any other format
  - API can operate both over the network and not over the network.

# Types of Webservices

- SOAP
  - SOAP stands for Simple Object Application protocol
  - SOAP uses WSDL (Web Services Description Language) is an XML document used to describe the function of a web service
- REST
  - REST stands for Representational State Transfer
  - REST works with plain text,xml and json

# Installation of PostMan

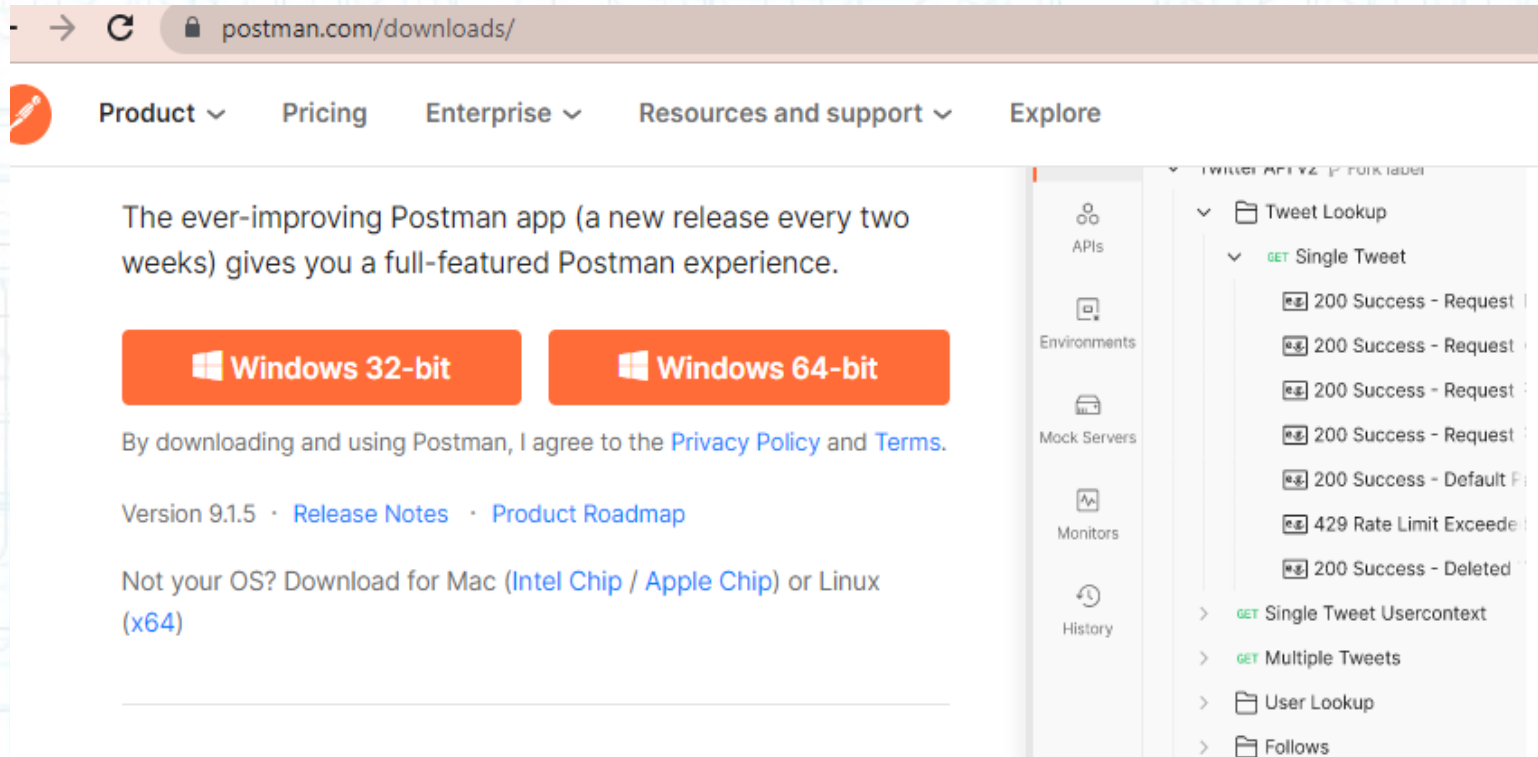
- Go to google.com and search install postman
- Click on the first link of [www.postman.com](https://www.postman.com)





# Installation of Postman

- Download the operating specific installation as shown below



- Once it is downloaded , double the click and follow steps to complete installation

## Calling Soap Request through Postman

- Create the Collection and add request in it
- Enter the url as <https://www.dataaccess.com/webservicesserver/NumberConversion.wso>
- Enter the WSDL as below under body

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
```

```
<soap:Body>
```

```
<NumberToWords xmlns="http://www.dataaccess.com/webservicesserver/">
```

```
<ubiNum>500</ubiNum>
```

```
</NumberToWords>
```

```
</soap:Body>
```

```
</soap:Envelope>
```



# Methods/Http Methods

- The primary or most-commonly-used for rest Api are HTTP methods i.e. POST, GET, PUT, and DELETE.
- These correspond to create, read, update, and delete (or CRUD) operations, respectively.
- There are a number of other methods, too, but are utilized less frequently.
- Payload-The response body is also known as the payload. It is usually formatted in JSON. It contains the data that we want to show to the users

# Four common methods

<b>GET</b>	Retrieves data from a remote server. It can be a single resource or a list of resources.
<b>POST</b>	Creates a new resource on the remote server. *
<b>PUT</b>	Updates the data on the remote server.
<b>DELETE</b>	Deletes data from the remote server.

# CURD OPERATIONS

HTTP Verb	CRUD	Entire Collection (e.g. /customers)	Specific Item (e.g. /customers/{id})
POST	Create	201 (Created), 'Location' header with link to /customers/{id} containing new ID.	404 (Not Found), 409 (Conflict) if resource already exists..
GET	Read	200 (OK), list of customers. Use pagination, sorting and filtering to navigate big lists.	200 (OK), single customer. 404 (Not Found), if ID not found or invalid.
PUT	Update/Replace	405 (Method Not Allowed), unless you want to update/replace every resource in the entire collection.	200 (OK) or 204 (No Content). 404 (Not Found), if ID not found or invalid.
PATCH	Update/Modify	405 (Method Not Allowed), unless you want to modify the collection itself.	200 (OK) or 204 (No Content). 404 (Not Found), if ID not found or invalid.
DELETE	Delete	405 (Method Not Allowed), unless you want to delete the whole collection—not often desirable.	200 (OK). 404 (Not Found), if ID not found or invalid.



# Http Status Code

- **Status four code is divided into four groups**
  - 1) The class of 2XX (200, 201, 204) indicates that the data was received and the operation was performed.
  - 2) The class of 3XX (301, 302) redirects the request to another URL.
  - 3) The class of 4XX (403, 404) indicates that the resource was is not found or not available to the client.
  - 4) The class of 5XX (500) to indicate a server error.

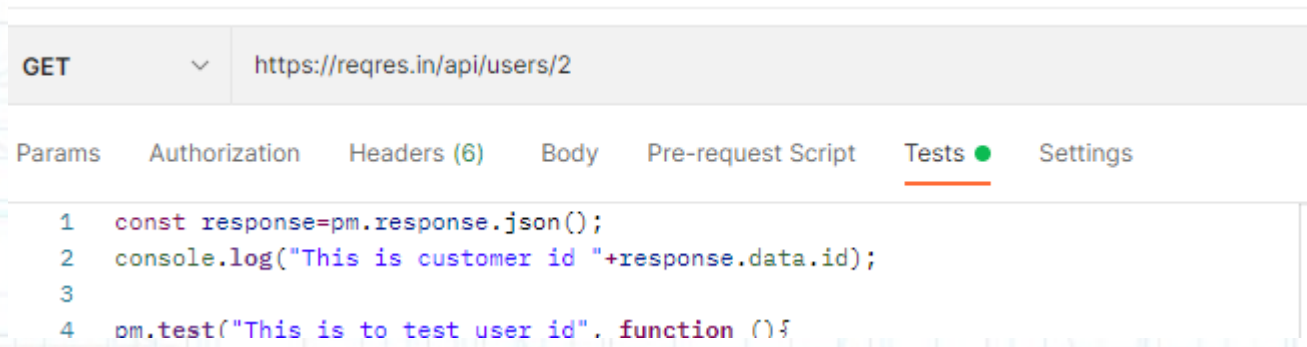
# Http Status Code

- 10 common responses

Code	Message	Description
200	OK	The data was received and the operation was performed.
201	Created	The data was received and a new resource was created. The response needs to return the data in the payload.
204	No content	The operation was successful but no data is returned in the response body. This is useful when deleting a resource.
301	Moved permanently	This and all the future requests should be redirected to a new URL.
302	Moved temporarily	The resource moved temporarily to another URL.
400	Bad request	The server cannot accept the request because something is wrong with the client or the request that it sent.
403	Forbidden	The client is not allowed to use the resource.
404	Not found	The computer is not able to find the resource.
405	Method not allowed	For example, when sending a DELETE request to a server that doesn't support the method.
500	Internal server error	Service unavailable due to error on the server side.

# Verifying Result through code

- Postman has inbuilt assertion library to validate the response . It is Chai assertion library
- To write the asseration , we need to use Test section of request as show below



- Chai library is written in javascript and has one global object that is **pm**



# Verifying Result through code

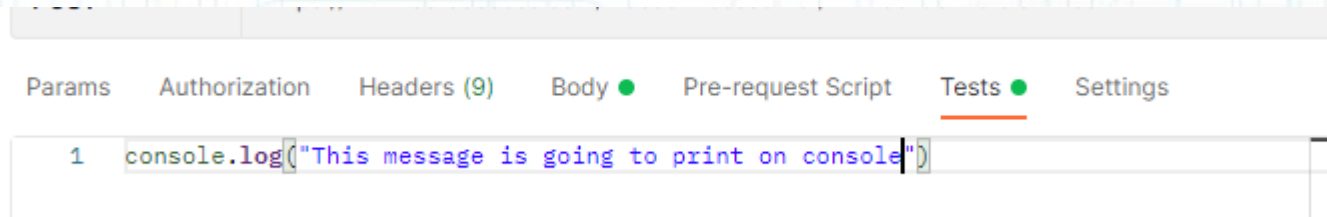
- **Snippet**
- Postman already given some inbuilt code which we need to use in API testing regularly e.g. Checking the response time
- You can find these snippet on the left side within Test Tab like below
- Clicking on the snippets in Test section



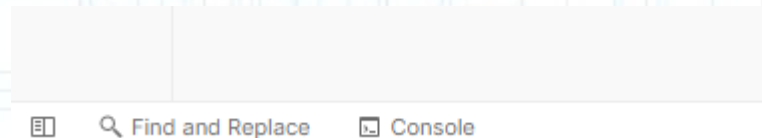
# Verifying Result through code

- **Console Log**

- If you want to print anything on the console then you can use following code , i.e. `console.log("")`



- To see the Postman console ,click on the console option which is at the right side and at the bottom of tool



# Chai Library

- PM object
  - It is global object in Chai library
  - To send the request or response we need to use pm object . Please find the some of the standard asseration methods

1 / 3

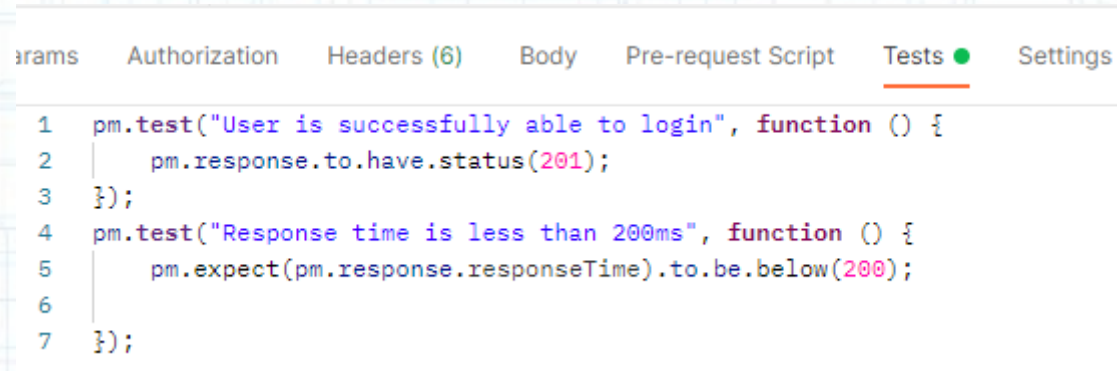
## Response Assertion API in test scripts

- `pm.response.to.have.status(code:Number)`
- `pm.response.to.have.status(reason:String)`
- `pm.response.to.have.header(key:String)`
- `pm.response.to.have.header(key:String, optionalValue:String)`
- `pm.response.to.have.body()`
- `pm.response.to.have.body(optionalValue:String)`
- `pm.response.to.have.body(optionalValue:RegExp)`
- `pm.response.to.have.jsonBody()`
- `pm.response.to.have.jsonBody(optionalExpectEqual:Object)`
- `pm.response.to.have.jsonBody(optionalExpectPath:String)`
- `pm.response.to.have.jsonBody(optionalExpectPath:String, optionalValue:*)`



# Tests

- To mark pass/fail in test result , we need to use test of pm object like below



The screenshot shows the 'Tests' tab in Postman. The interface includes tabs for 'Params', 'Authorization', 'Headers (6)', 'Body', 'Pre-request Script', 'Tests', and 'Settings'. The 'Tests' tab is active, displaying a code editor with the following JavaScript code:

```
1 pm.test("User is successfully able to login", function () {  
2     pm.response.to.have.status(201);  
3 });  
4 pm.test("Response time is less than 200ms", function () {  
5     pm.expect(pm.response.responseTime).to.be.below(200);  
6  
7 });
```

- In above example we are checking the response status as 201 and response time should be below 200 ms



# Test Result

- Postman will marked pass/fail based on criteria under test result as follows

The screenshot displays the Postman interface for a GET request to `https://reqres.in/api/users/2`. The 'Tests' tab is active, showing two test cases. Below the tests, the 'Test Results' tab is selected, showing two failed tests. The status bar at the bottom indicates 'Status: 200 OK' and 'Time: 100ms'.

```
1 pm.test("User is successfully able to login", function () {
2   pm.response.to.have.status(201);
3 });
4 pm.test("Response time is less than 200ms", function () {
5   pm.expect(pm.response.responseTime).to.be.below(200);
6
7 });
```

Body Cookies Headers (19) **Test Results (0/2)** Status: 200 OK Time: 100ms

All Passed Skipped Failed

**FAIL** User is successfully able to login | AssertionError: expected response to have status code 201 but got 200

**FAIL** Response time is less than 200ms | AssertionError: expected 2339 to be below 200