**Tutorial Document: Test Scripts, API Documentation, Swagger, and Authentication**

**1. Using API Documentation and Swagger**

API documentation is essential for understanding how to interact with different endpoints, the data they require, and the expected responses. Swagger is a tool that helps with visualizing and interacting with API documentation in a user-friendly interface.

**Section 1: Understanding API Documentation**

**What is API Documentation?**  
API documentation provides details on how to use an API, listing endpoints, required parameters, response structures, and potential errors. It often includes:

* **Endpoints**: Available actions and their paths.
* **Parameters**: Inputs required or optional for a request.
* **Headers**: Metadata like authorization or content type.
* **Sample Requests and Responses**: Examples to guide API calls.

**How to Read Documentation Effectively**:

1. Look for **Base URL** (the main part of the API’s URL).
2. Review **available endpoints** and the HTTP methods (GET, POST, etc.).
3. Check for **required parameters** and **authentication** requirements.
4. View **sample responses** to understand the expected data format.

**Here are the detailed steps to create documentation in Postman:**

**1. Create a Collection**

* **Step 1**: Click on **Collections** in the left panel of Postman.
* **Step 2**: Click **New Collection** and give it a meaningful name (e.g., "JSONPlaceholder API Tests").
* **Step 3**: Add requests to this collection by either creating new ones or moving existing requests into it. This structure will make it easier to generate organized documentation.

**2. API Testing Basics using Test Scripts (Test Snippets)**

**GET Request**

**Endpoint**: https://jsonplaceholder.typicode.com/posts

1. **Setup**:
   * Select Add Request Option by clicking the three dots right to the JSONPlaceholder API Tests Collection.
   * Enter the URL in the Postman address bar.
   * Method: GET
2. **Add Test Snippet**:
   * Click on the **Scripts** tab.
   * Select the snippet **"Status code: Code is 200"**.

**Test Script**:

pm.test("Status code is 200", function () {

pm.response.to.have.status(200);

});

1. **Send Request**:
   * Click **Send** and verify the status code is 200 in the test results.

**POST Request**

**Endpoint**: https://jsonplaceholder.typicode.com/posts

1. **Setup**:
   * Select Add Request Option by clicking the three dots right to the JSONPlaceholder API Tests Collection.
   * Method: POST
   * Body: Select **raw** → **JSON**

json

{

"title": "foo",

"body": "bar",

"userId": 1

}

1. **Add Test Snippet**:
   * Open the **Scripts** tab.
   * Select the snippet **"Response body: Contains string"**.

**Test Script**:

pm.test("Response body contains title", function () {

pm.expect(pm.response.text()).to.include("foo");

});

1. **Send Request**:
   * Click **Send** and verify the response contains "title": "foo".

**PUT Request**

**Endpoint**: https://jsonplaceholder.typicode.com/posts/1

1. **Setup**:
   * Select Add Request Option by clicking the three dots right to the JSONPlaceholder API Tests Collection.
   * Method: PUT
   * Body: Select **raw** → **JSON**

{

"id": 1,

"title": "updated title",

"body": "updated body",

"userId": 1

}

1. **Add Test Snippet**:
   * Open the **Scripts** tab.
   * Select the snippet **"Response time is less than 200ms"**.

**Test Script**:

pm.test("Response time is less than 200ms", function () {

pm.expect(pm.response.responseTime).to.be.below(200);

});

1. **Send Request**:
   * Click **Send** and ensure the response time is below 200ms.

**DELETE Request**

**Endpoint**: https://jsonplaceholder.typicode.com/posts/1

1. **Setup**:
   * Select Add Request Option by clicking the three dots right to the JSONPlaceholder API Tests Collection.
   * Method: DELETE
2. **Add Test Snippet**:
   * Open the **Scripts** tab.
   * Select the snippet **"Status code: Code is 204"**.

**Test Script**:

pm.test("Status code is 204", function () {

pm.response.to.have.status(204);

});

1. **Send Request**:
   * Click **Send** and verify the status code is 204 (No Content).

**PATCH Request**

**Endpoint**: https://jsonplaceholder.typicode.com/posts/1

1. **Setup**:
   * Select Add Request Option by clicking the three dots right to the JSONPlaceholder API Tests Collection.
   * Method: PATCH
   * Body: Select **raw** → **JSON**

{

"title": "patched title"

}

1. **Add Test Snippet**:
   * Open the **Scripts** tab.
   * Select the snippet **"Response body: JSON value check"**.

**Test Script**:

pm.test("Response title is updated", function () {

var jsonData = pm.response.json();

pm.expect(jsonData.title).to.eql("patched title");

});

1. **Send Request**:

Click **Send** and confirm the title is updated in the response.

**Now, Run the whole collection by Clicking on “Run Collection” option. View and interpret the search results.**

**3. Add Descriptions in the Documentation Tab**

* **Step 1**: Open a request in your collection.
* **Step 2**: Go to the **Documentation Tab** (found in the right panel of the request editor).
* **Step 3**: Write a detailed description of what the request does, such as the endpoint’s purpose, method (GET, POST), and expected outcomes.

**3. Add Comments for Parameters, Headers, and Body Content**

* **Step 1**: Under the **Params**, **Headers**, and **Body** sections, add comments for each parameter and header explaining its purpose and usage.
* **Step 2**: This helps users understand the data required for each request, especially if certain headers or parameters are essential for authentication or specifying data formats.

**4. Generate Shareable Documentation**

* **Step 1**: Go to the collection's options (click on the three dots next to the collection name).
* **Step 2**: Select **View documentation**.
* **Step 3**: Postman will generate an HTML documentation page with all the details you've provided.
* **Step 4**: You can share this documentation via a public URL, or keep it private and share it with team members.

This documentation can then be shared as a comprehensive reference for developers and team members, complete with request details, descriptions, and example responses.

**Section 2: Exploring Swagger**

**What is Swagger?**  
Swagger is an interactive API documentation tool. It provides a way to view and test REST API endpoints without writing code.

**Swagger in Postman: Step-by-Step Basics**

**Step 1: Access Swagger Documentation**

* Navigate to a public API with Swagger documentation, e.g., Pet Store API (<https://petstore.swagger.io/> ).
* Explore the API structure:
  + **Endpoints**: /pet, /store, /user, etc.
  + **Methods**: GET, POST, PUT, DELETE.
  + **Parameters**: Query parameters, headers, or body content.
  + **Responses**: Status codes and example responses.

**Step 2: Import Swagger Specification into Postman**

1. Open Postman.
2. Click the **Import** button (top-left corner).
3. **Paste URL**: Copy the Swagger file URL (often .json) and paste it into Postman's import tool (<https://petstore.swagger.io/v2/swagger.json> ).
4. Click **Continue** to complete the import.
5. A new collection will appear in Postman with all endpoints pre-configured.

**Working with Swagger in Postman: Examples**

**Example 1: Testing a GET Request**

* **API**: Pet Store API
* **Endpoint**: /pet/findByStatus
* **Steps**:
  1. In Postman, open the imported collection and select GET /pet/findByStatus.
  2. Add a query parameter: status=available.
  3. Click **Send**.
  4. View the response:
     + **Status Code**: 200 OK.
     + **Response Body**: List of pets with status available.

**Example 2: Testing a POST Request**

* **API**: Pet Store API
* **Endpoint**: /pet
* **Steps**:
  1. In Postman, open POST /pet.
  2. Navigate to the **Body** tab and select raw -> JSON.
  3. Add the following JSON:

json

{

"id": 12345,

"name": "Buddy",

"status": "available"

}

* 1. Click **Send**.
  2. View the response:
     + **Status Code**: 200 OK.
     + **Response Body**: Confirmation of the new pet's creation.

**Exploring Swagger Documentation**

1. **Endpoints**:
   * URLs that represent specific API functionalities.
2. **Methods**:
   * HTTP methods: GET (retrieve data), POST (create data), PUT (update data), DELETE (remove data).
3. **Parameters**:
   * **Query parameters** (e.g., ?status=available).
   * **Path parameters** (e.g., /pet/{petId}).
4. For example, consider the following URL: https://api.example.com/users/{userId}/orders/{orderId}  
   The path parameters here are {userId} and {orderId}.
5. Now take this URL, https://api.example.com/users/123/orders/456  
   When path parameters {userId} and {orderId} are replaced with actual values (123 and 456), they form the **endpoint**.
6. **Responses**:
   * Status codes (e.g., 200 OK, 404 Not Found).
   * Example response bodies.

**Section 3: Authentication in Postman**

Many APIs require authentication, which verifies your identity to access protected resources. Common authentication methods include API keys, Bearer tokens, and Basic authentication.

**3.1 Types of Authentication**

1. **API Key**: A simple token passed as a query parameter or header.
2. **Basic Authentication**: Uses a username and password, encoded in base64, in the request header.
3. **Bearer Token**: A token passed in the request header, often used with OAuth 2.0.

**3.2 Implementing Authentication in Postman**

**Example API for Practice**  
We’ll use the sample endpoint https://jsonplaceholder.typicode.com (this API doesn’t require authentication, but you can simulate an API Key for practice).

**3.2.1 Setting Up Postman for API Authentication**

1. **Create a Collection**:
   * Click **New > Collection**.
   * Name it **JSONPlaceholder Authentication**.
2. **Add an Environment**:
   * In the **Environment** tab, create variables for API Key and Base URL.
   * Example:
     + baseUrl = https://jsonplaceholder.typicode.com
     + apiKey = 12345 (simulated key for demo).

**3.2.2 Simulating API Authentication using API Key**

**Example 1: API Key in Query Parameters**

1. **Endpoint**: GET /posts
2. **Steps**:
   * URL: {{baseUrl}}/posts?apiKey={{apiKey}}
   * If already not added, go to **Params** tab and add:
     + apiKey: 12345 (Simulated Key).
   * Click **Send**.
3. **Expected Response**:
   * **Status**: 200 OK.
   * **Body**: List of posts.
4. **Validation Test**:
   * In the **Scripts** tab, add:

pm.test("Status code is 200", function () {

pm.response.to.have.status(200);

});

**Example 2: API Key in Headers**

1. **Endpoint**: GET /posts
2. **Steps**:
   * URL: {{baseUrl}}/posts
   * Go to **Headers** tab and add:
     + Key: Authorization
     + Value: Bearer {{apiKey}} (Simulated Token).
   * Click **Send**.
3. **Expected Response**:
   * **Status**: 200 OK.
   * **Body**: List of posts.

**3.2.3 Simulating API Authentication using Basic Authentication**

**Example 3: Basic Authentication**

1. **Endpoint**: GET /posts/1
2. **Steps**:
   * URL: {{baseUrl}}/posts/1
   * Go to **Authorization** tab and select **Basic Auth**.
     + Username: user
     + Password: password
   * Click **Send**.
3. **Expected Response**:
   * **Status**: 200 OK.
   * **Body**:

{

"userId": 1,

"id": 1,

"title": "sunt aut facere repellat provident occaecati excepturi optio reprehenderit",

"body": "quia et suscipit\nsuscipit recusandae consequuntur expedita et cum\nreprehenderit molestiae ut ut quas totam\nnostrum rerum est autem sunt rem eveniet architecto"

}

**3.2.4 Simulating API Authentication using Bearer Token Authentication**

**Example 4: Bearer Token Authentication**

1. **Endpoint**: POST /posts
2. **Steps**:
   * URL: {{baseUrl}}/posts
   * Go to **Headers** tab and add:
     + Key: Authorization
     + Value: Bearer 12345 (Simulated Token).
   * Go to **Body** tab and select **raw**. Add:

{

"title": "foo",

"body": "bar",

"userId": 1

}

* + Click **Send**.

1. **Expected Response**:
   * **Status**: 201 Created.
   * **Body**:

{

"title": "foo",

"body": "bar",

"userId": 1,

"id": 101

}

**4. Practical Exercises**

**Exercise 1: API Key Simulation**

1. Create a request to https://jsonplaceholder.typicode.com/posts.
2. Use Authorization: Bearer fake\_api\_key in the **Headers**.
3. Verify that you can successfully send the request and view a response.

**Exercise 2: Using Swagger with an Authentication Token**

1. Access a Swagger-based API with authentication requirements.
2. Obtain a sample Bearer token if available.
3. Enter the token in Swagger’s **Authorize** section and interact with the API securely.

**5. Summary and Key Takeaways**

* **Swagger** provides an easy-to-use interface for API documentation and testing.
* **Authentication Types** include API Key, Bearer Token, and Basic Auth.
* **Postman’s Authorization Tab** simplifies the use of these authentication types in requests.

**6. Assignment**

1. **Explore Swagger Documentation**: Use https://petstore.swagger.io/ to create and retrieve data about pets.
2. **Simulate API Key and Bearer Token Authentication in Postman**:
   * Create separate requests for API Key and Bearer Token authentication, substituting with any sample keys/tokens.
3. **Document Findings**: Note the responses and authentication techniques used in a shared document.