**Tutorial Document: Introduction to APIs and Postman Basics**

**1. Introduction to APIs**

**What is an API?**  
An **API (Application Programming Interface)** allows different software systems to communicate with each other. APIs define how requests are made and responses are returned, enabling systems to share information and functionality without direct user intervention.

**Key Concepts:**

* **Endpoints**: Specific URLs that allow access to a particular part of an API.
* **HTTP Methods**: APIs commonly use HTTP methods (GET, POST, PUT, DELETE) to perform actions on the server.
* **HTTP Status Codes**: Codes that indicate the success or failure of an API request (e.g., 200 OK, 404 Not Found).

**Example:**  
Think of an API as a waiter taking your order at a restaurant and returning your meal from the kitchen.

**2. Understanding HTTP and JSON**

**HTTP Basics**  
HTTP (HyperText Transfer Protocol) allows communication between clients (e.g., your computer) and servers. HTTP requests have a **method** (e.g., GET, POST) and often require headers, parameters, and/or a body to complete.

**JSON (JavaScript Object Notation)**  
JSON is a lightweight data format used for data exchange. It’s easy for humans to read and write and for machines to parse and generate.

**Example of JSON:**

json

{

"name": "Alice",

"age": 30,

"is\_student": false

}

In this example, name, age, and is\_student are keys, and their values are "Alice", 30, and false, respectively.

**3. Getting Started with Postman**

Postman is a popular tool for testing APIs. It allows users to send HTTP requests, view responses, and interact with APIs without writing code.

**Postman Setup:**

1. Download and install Postman.
2. Open Postman, and create a free account (optional, but recommended for saving your work).

**4. Exploring the Postman Interface**

The Postman interface consists of several key elements:

* **Workspace**: A collection of your saved requests and projects.
* **Collections**: Organized folders for your requests.
* **Request Builder**: Where you create and send HTTP requests.

**Step-by-Step Practical Guide**

**4.1. Building a Simple GET Request**

A GET request retrieves information from the server. We’ll use a sample public API to practice.

1. **Open Postman** and click **+ New Request** in the workspace.
2. **Set the Request Method** to GET.
3. **Enter the Request URL**: https://jsonplaceholder.typicode.com/posts/1
   * This sample API will return data for a single post in JSON format.
4. Click **Send**.
5. **View the Response**: Observe the JSON response in the lower panel.

**Example of Response:**

json

{

"userId": 1,

"id": 1,

"title": "Sample Title",

"body": "This is the body of the post."

}

**4.2. Building a POST Request**

A POST request sends data to the server to create a new resource.

1. **Create a New Request** with the POST method.
2. **Enter the URL**: https://jsonplaceholder.typicode.com/posts
3. **Add Headers**:
   * Key: Content-Type
   * Value: application/json
4. **Add a Request Body** (select raw and JSON format):

json

{

"title": "Sample Post",

"body": "This is the content of the new post",

"userId": 1

}

1. Click **Send** and check the response.

**4.3. Building a PUT Request**

A PUT request updates an existing resource with new data.

1. Create a New Request with the **PUT** method.
2. Enter the URL: https://jsonplaceholder.typicode.com/posts/1.
   * This will update the post with ID 1.
3. Add Headers:
   * **Key:** Content-Type
   * **Value:** application/json
4. Add a Request Body (select **raw** and **JSON** format):

json

{

"id": 1,

"title": "Updated Title",

"body": "This is the updated content of the post",

"userId": 1

}

1. Click **Send** and view the response.

**Example of Response:**

json

Copy code

{

"id": 1,

"title": "Updated Title",

"body": "This is the updated content of the post",

"userId": 1

}

**4.4. Building a DELETE Request**

A DELETE request removes a resource from the server.

1. Create a New Request with the **DELETE** method.
2. Enter the URL: https://jsonplaceholder.typicode.com/posts/1.
   * This will delete the post with ID 1.
3. Click **Send**.
4. Check the Response Status Code (e.g., 200 OK or 204 No Content) to confirm the deletion.

**Note:** Since JSONPlaceholder is a mock API, the deletion isn’t actually persisted.

**4.5. Building a PATCH Request**

A PATCH request updates specific fields in an existing resource.

1. Create a New Request with the **PATCH** method.
2. Enter the URL: https://jsonplaceholder.typicode.com/posts/1.
   * This will partially update the post with ID 1.
3. Add Headers:
   * **Key:** Content-Type
   * **Value:** application/json
4. Add a Request Body (select **raw** and **JSON** format):

json

{

"title": "Partially Updated Title"

}

1. Click **Send** and view the response.

**Example of Response:**

json

{

"id": 1,

"title": "Partially Updated Title",

"body": "This is the body of the post.",

"userId": 1

}

**5. Introduction to HTTP Status Codes**

Understanding the status codes returned by an API response helps you interpret the outcome of your requests:

* **200 OK:** Success.
* **201 Created:** New resource created.
* **400 Bad Request:** Client-side error.
* **401 Unauthorized:** Requires authentication.
* **404 Not Found:** Resource doesn’t exist.
* **500 Internal Server Error:** Server-side error.

**6. Practical Exercise: Experimenting with JSON Data**

**Objective**: To familiarize yourself with JSON data structure and response interpretation.

1. Make a GET request to https://jsonplaceholder.typicode.com/users/1.
2. Identify:
   * The user’s name
   * Email
   * Address details
3. Change the ID number in the URL (e.g., /users/2) to retrieve data for different users.

**Challenge**: Try to change parts of the JSON in a POST request and see how different data inputs alter the response.

**7. Summary and Key Takeaways**

* **API Basics**: APIs are interfaces for communication between software systems.
* **Postman Setup**: Use Postman to send and inspect HTTP requests.
* **Request Types**: GET for retrieving data, POST for creating new data.
* **JSON**: A format used to exchange data.
* **HTTP Status Codes**: Help identify the success or error in requests.

**8. Assignment**

1. **Create a Collection**: Organize your GET and POST requests into a collection.
2. **Experiment with Endpoints**:
   * Use https://jsonplaceholder.typicode.com/todos for GET requests.
   * Use https://jsonplaceholder.typicode.com/posts for POST requests.
3. **Document Findings**: Record responses and HTTP status codes for each request in a shared document or spreadsheet.