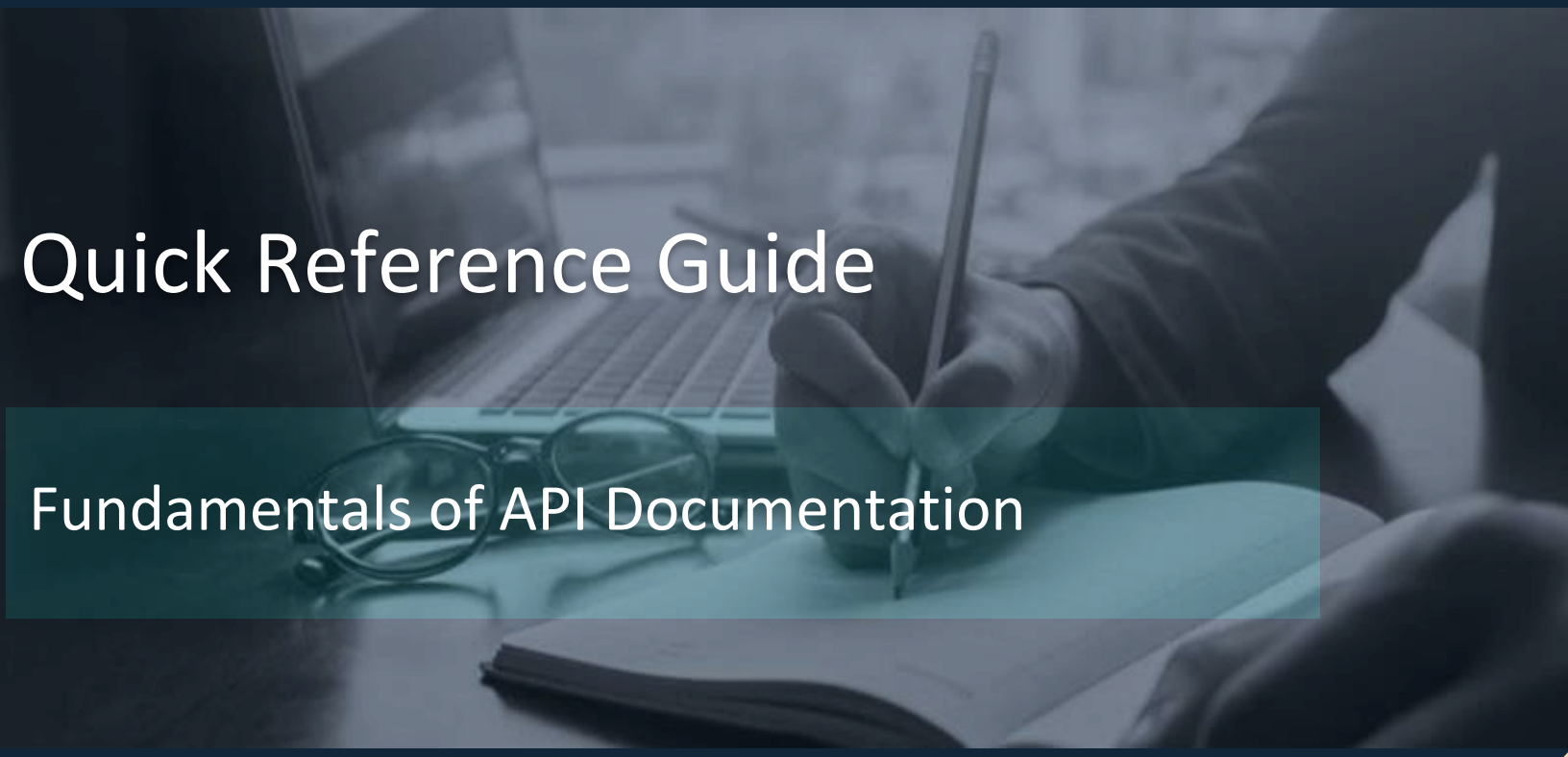
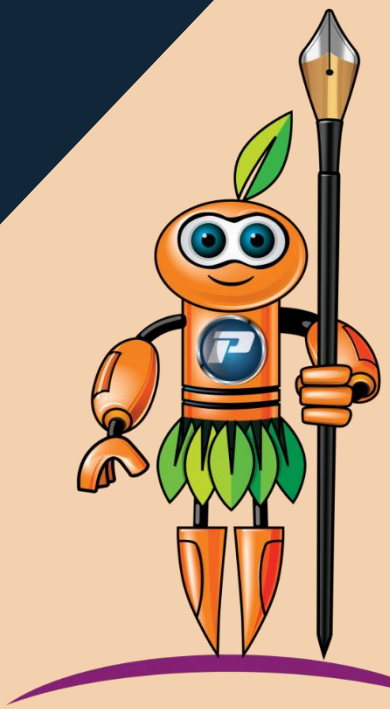
Tech Writer's Tribe



Quick Reference Guide

Fundamentals of API Documentation



Fundamental Course on API Documentation

Table of Contents

Introduction

The Fundamental Course on API Documentation is designed for technical writers who want to learn how to document the APIs. The course covers the basics of APIs and best practices while documenting the APIs.

What is an API

- API is an acronym for Application Programming Interface
- It is an interface between two applications.
- Wikipedia says:
  An **Application Programming Interface** (**API**) is a computing interface which defines interactions between multiple software intermediaries.
- **Example**: In a restaurant, a waiter is the mediator or an API between you and the kitchen. He gets all the information or food, you want from the kitchen. So, you make a **REQUEST** and you get a **RESPONSE**.

What are Microservices

Microservices are small modules of programs that interact with each other to build an application. These modules interact with each other through APIs.

What are Web services

- Services that you request over the web => Web Services
- When you request for any page of the website using the URL, you call the website through the APIs. The web browser sends a request to the website and in response the page is displayed.
- **All web services are APIs but not all APIs are web services**. The non- webservices APIs are the ones that are used in microservices.

What is SOAP

- Acronym for Simple Object Access Protocol
- Requires XML
- Message format is usually defined through a WSDL (Web Services Description Language) file.
- The SOAP protocol is fully described at http://www.w3.org/TR/SOAP.

What is JSON

- Acronym of JavaScript Object Notation.
- A lightweight text-based open standard designed for human-readable data interchange.
- JSON is a language-independent data format
- Douglas Crockford originally specified the JSON format in the early 2000s. JSON was first standardized in 2013, as ECMA-404. RFC 8259 and published in 2017.
- Data is represented in key-value pairs.
- Curly braces hold objects, and each name is followed by a colon.

- The key-value pairs are separated by a comma.
- Square brackets hold arrays and values are separated by a comma.

JSON Data Types

Most common JSON types:

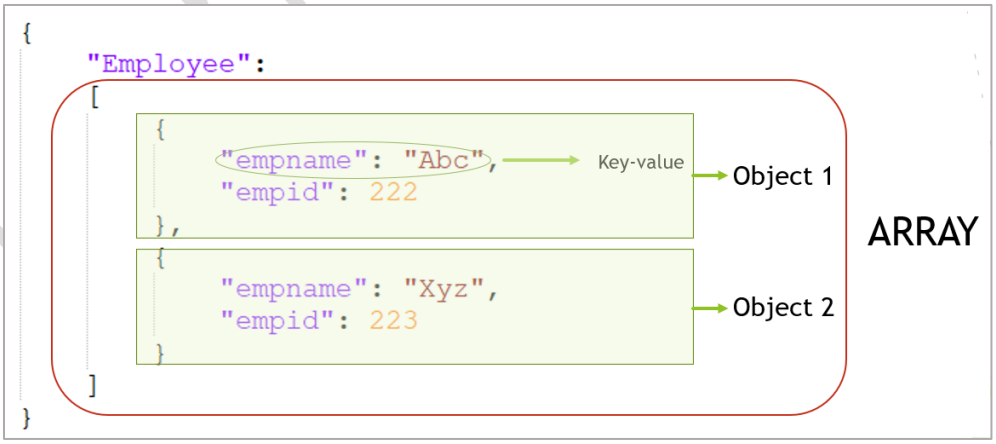
Data Type	Description	Example
String	Always in double quotes. Can consist of numbers, alphanumeric and special characters.	"employee", "name", "2345", "Ver_1"
Numbers	Only numeric characters	345, 678
Boolean	Either of True or False	true
Null	Empty value	

JSON Arrays

- A list of objects. Always enclosed in square brackets [ ].
- Array values must be of type string, number, object, array, Boolean or null.

JSON Objects

- Objects are JSON's dictionaries.
- JSON objects are surrounded by curly braces { }.
- JSON objects are written in key/value pairs.
- Keys must be strings, and values must be a valid JSON data type (string, number, object, array, boolean or null).
- Keys and values are separated by a colon.
- Each key/value pair is separated by a comma.
- **Example**:
  {"name" : "employee name", "employeeid" : 245, "present" : true}



What are REST APIs

- Architectural style to design APIs. Therefore, also known as RESTful APIs.
- Acronym for Representational State Transfer.
- Uses HTTP as a transport protocol.
- Message format is JSON (JavaScript Object Notation)/XML/text/image).

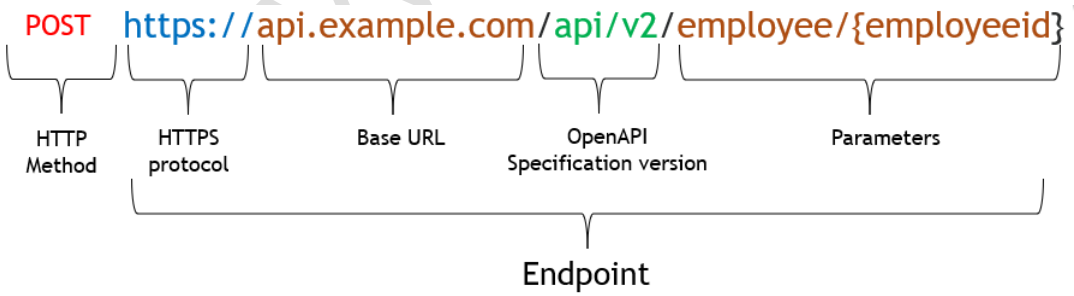
Authentication and Authorization

- Authentication refers to proving the correct identity.
- Authorization refers to allowing access to the server to perform actions.

Type	Authentication	Authorization	Examples
No Auth	No	No	Google search
Basic Auth	Yes	No	Email account
Bearer Token	No	Yes	Company specific
OAuth	Yes	Yes	
Two-factor Authentication	Yes	Yes	Bank accounts

Anatomy of an HTTP Request

- Endpoint: Address where API is hosted on the server.
- Resources: Represent API/Collection which can be accessed from the server.



HTTP Methods

- HTTP methods are commonly used to communicate with Rest APIs.
  Popular HTTP methods:
  - **GET:** Used to extract information from the given server using a given URI. Usually Query Parameters are used.
  - **POST:** Used to send data to the server contained in the request body.
  - **PUT:** Used to update or modify the current data in the server contained in the request body.
  - **DELETE:** Used to delete all current data of the target server.

Also known as **CRUD** methods.

Method	Action
POST	CREATE
GET	READ
PUT	UPDATE
DELETE	DELETE

Parameters

Variable in a URL:

- **Query parameters:** Parameters used to get the information on the resources. These are usually followed by a question mark (?).
- **Path parameters**: Points to specific resource. These are usually within curly braces.
- **Headers**: Headers represent the meta-data and authorization information associated with the API request and response.

OpenAPI Specifications

The OpenAPI Specification (OAS) defines a standard, language-agnostic interface to RESTful APIs which allows both humans and computers to discover and understand the capabilities of the service without access to source code, documentation, or through network traffic inspection. When properly defined, a consumer can understand and interact with the remote service with a minimal amount of implementation logic.

An OpenAPI definition can then be used by documentation generation tools to display the API, code generation tools to generate servers and clients in various programming languages, testing tools, and many other use cases.

Reference: https://swagger.io/specification

Documenting the APIs

Once you understand an API, it is time for documenting the API. So, here are the must-haves and the best practices explained.

Identify your Audience

People consuming or using the APIs are not just the developers but other stakeholders as well. So, we can categorize the audience as:

- **Decision makers**: The CEOs, product managers, software architects, business managers. These people individually or collectively decide whether they need the services or data from your company. These people must understand as to how your product or APIs are useful for them.
- **Developers**: The developers who will actually use the APIs. Though the developers are technical, but keep in mind that they may be freshers or from different domain. The document should be equally helpful for the developers of your company who are constantly working on improving the APIs.

Structure of an API Document

The API document consists of two sections:

- API Overview
- API Reference

Let us study more about these sections.

API Overview

This section should have thorough introductions to the concepts, standards, and ideas in an API's data structures and functionality.

- **Concepts should explain**:
    - The details about your product.
    - How can the APIs solve the developer's problem?
- **Workflow/Getting Started should explain:**
    - How can the developers use the APIs?
    - High-level architecture diagram (if possible)
    - Workflow diagram (if possible)
- **Prerequisites**: Any prerequisites to consume the APIs.
- **Return Status codes**: All return status codes used in the APIs.
- **Throttling limits/ Rate Limits**: Used to control the usage of APIs by consumers during a given period. For example, you can limit the number of total API requests as 10000/day.

API Reference

Though this section would be majorly used by the developers, use a simple, concise, and easy-to-understand language. This is where you will give complete information about on API that the developer will use.

This section has the following:

- URI – Brief description about what does the API do
- URL – Request URL format
- Method – State the HTTP method used for the API
- Headers – Stating
    - Type of Authorization/Authentication
    - Application type – whether it is JSON/XML/text/image/HTML
- Example URL: An example of the URL
- Table of parameters

Parameter (mention whether they are mandatory/optional)	Data Type	Default Values (if there are any)	Description

- Example Request: Include an example request as a code block.
- Example Response: Include an example response as a code block.

Best Practices

Now, that you know how to document your APIs, let's see how to beautify your document and make it flawless. Here are some tips and tricks:

- Test the API before you start documenting. Technical writing is not just about writing anymore. Wear the hat of a user and test your APIs.
- While documenting the parameters, check if the data type of a parameter matches the value given in the sample request or sample response payload. If you are not clear, go back to the developer to clarify.
- Always check the sample request and response bodies for any personal information being mentioned such as name, phone number, or email IDs.
- The parameters may be case-sensitive. Best practice is - always document the parameter with the exact case used by the developer.
- Keep the language easy to understand and simple to comprehend.
- Try 'not' to use the CRUD function names in the descriptions.
- For some developers, examples play a more important role in getting started with an API than the explanations of calls and parameters. Therefore, always include example requests and responses.

All the Best !!!