**API Learning**

*What? Like it’s Hard?*

[Overview](https://idratherbewriting.com/learnapidoc/) & [Chapter 1](https://idratherbewriting.com/learnapidoc/docapis_intro_to_rest_api_doc.html)

[*INTRODUCTION TO REST API DOCUMENTATION*](https://idratherbewriting.com/learnapidoc/docapis_intro_to_rest_api_doc.html)

API TYPES:

* **Web service APIs:** Sends and receives messages across the web using HTTP to transport the request and response. They are language agnostic.
  + [**REST**](https://idratherbewriting.com/learnapidoc/docapis_introtoapis.html): Representational State Transfer
  + **SOAP**
  + **RPC-based**
  + **Grpc**
  + **GraphQL APIs**
  + **Voice Assistant APIs**
* **Native Library APIs** (or Library based APIs):
  + - Refer to code libraries that devs add directly to their projects
    - Functions incorporated locally within code, expands performable operations w/in proj
    - No need for cloud
    - You need to be familiar to the programming language
    - Most challenging doc task for tech writers
* **Internet of Things (IoT) APIs:** IoT APIs are used by physical devices (such as sensors or wearables) that transmit or receive data to connect the device to an online network. For example, a thermostat sensor in a room might transmit the temperature to a central controller (such as with Nest) via an IoT API. For more detail, see [App nirvana: When the Internet of Things meets the API economy.](https://techbeacon.com/app-dev-testing/app-nirvana-when-internet-things-meets-api-economy) See also [APIs in the world of IoT](https://apifriends.com/api-management/iot-api/).

GODDAMN THAT WAS A LOT AND I STILL DON’T REALLY GET IT

*REST and SOAP are by far the most common, so a good place to start*

APIs, like cars, have different functionalities and pro/cons, developers use different ones to best fit their project.

REST

* REST APIs consist of requests and responses between clients and servers
* Make requests through URL paths
* Req/resp travel through HTTP
* Not required to be a specific prog language
* Resp returned in JSON or XML
* The users make requests for the resources on a web server, and the server returns responses containing the information.
* Both the system initiating the request and the system providing the response can be in any programming language, so long as they transmit the message via HTTP.
* Follow the same protocol as the web
* Don’t need to know a specific programming language to document REST APIs

HISTORY

* eBay’s API in 2005 was one of the first web APIs — the API allowed sellers to manage their products in their eBay stores.
* Now APIs are everywhere and sites are pulling all they need through APIs
  + Rather than building your own payment gateway, you might integrate the [Stripe API](https://stripe.com/docs/api).
  + Rather than building your own e-commerce system, you might use the [Snipcart API](https://docs.snipcart.com/api-reference/introduction).
* Jekyll, a SSG, has a basic program with few integrations. Devs must pull MailChimp for their email and Stripe for payments. This is ideal b/c then every service and extension is the specialized version, powered by experts.

PROCESS

* technical writers should be collaborating with engineers to generate reference documentation through OAS.
* Reference docs account for only part of the needed documentation (maybe half, if that).
* I heavily recommend that technical writers generate reference documentation through the OAS.
* This is covered in the [OpenAPI spec and Swagger section](https://idratherbewriting.com/learnapidoc/pubapis_rest_specification_formats.html)
* many technical writers promote and champion OAS as a standard for creating the reference docs

DOCUMENTATION

* REST follows an *architectural style* rather than an exact protocol standard
* Compare to SOAP APIs, which enforce a specific message format for sending requests and returning responses. As an XML message format, SOAP is very specific and has a WSDL (Web Service Description Language) file that describes how to interact with the API.
* REST APIs, however, do not follow a standard message format. Instead, REST is an architectural *style*, a set of recommended practices for submitting requests and returning responses. To understand the request and response format for REST APIs, you don’t consult the SOAP message specification or look at the WSDL file. Instead, you have to consult the REST API’s *documentation*.
* Each REST API functions a bit differently. There isn’t a single way of doing things, and this flexibility and variety fuel the need for accurate and clear documentation.
* Many employers are looking to hire technical writers who can create not only complete and accurate documentation but who can also create stylish outputs for their documentation.
* With API documentation, there is no GUI interface for users to browse. Instead, the documentation *is* the interface.
  + Employers know this, so they want to make sure they have the right resources to make their API docs stand out as much as possible.
  + Below is DropBox’s Documentation

[A screenshot of a web page

AI-generated content may be incorrect.](https://www.dropbox.com/developers)

* Clearly good UI is a pillar of good documentation
* Documentation is the product interface, so a sharp, modern looking site is necessary.

[*WHAT IS A REST API?*](https://idratherbewriting.com/learnapidoc/docapis_what_is_a_rest_api.html)

When you go on Google Flights or Kayak the site is reaching out for information from other sites to provide you with a simplified list. It does this through APIs.

A YouTube video imbedded in a website? Yup, API.

When you purchase a pay-per-view movie on your Fire TV and a payment portal opens? Payment doesn’t occur on Fire TV; the app makes API calls out to the payment servers (perhaps Stripe?) and payment is processed in the cloud.

Web Services

*~language agnostic* ♥ *interoperable across different platforms and systems~*

All APIs that use HTTP protocol as the transport format for requests and responses are considered “web services.”

* With web services, the client making the request for the resource and the API server providing the response can use any programming language or platform — it doesn’t matter because the message request and response are made through a common HTTP web protocol.
* It doesn’t matter whether engineers build the API with Java, Ruby, Python, or some other language. The requests are made over HTTP, and the responses are returned through HTTP.

A diagram of a cloud computing process

AI-generated content may be incorrect.

REST is a style, not a standard

* REST is an architectural style, not a standard protocol
  + REST is a general style that the API follows
  + RESTful or REST-like are kinda REST but not perfectly
* Can use message styles like XML, JSON, Atom, RSS, CSV, HTML, and more.
* Most use JSON –it’s easy to work with/through, can be parsed with JavaScript

REST API’s focus on:

* resources accessed through URLs
* *things*, rather than actions
* ways to access the resources

DEFINITIONS:

**Static site generator (SSG):** is a software tool that automatically creates static HTML web pages from content sources like Markdown or other plain text files

**OpenAPI specification (OAS):** formerly known as the Swagger Specification, is a standard for describing REST APIs in a machine-readable format. It uses [YAML or JSON](https://www.google.com/search?sca_esv=ee5c04314dac669e&cs=1&sxsrf=AE3TifOYlKqi4qr1VkPZ17NxlD0lusNvAg%3A1749143016086&q=YAML+or+JSON&sa=X&ved=2ahUKEwjiw6fA4dqNAxU-FVkFHcDTAVoQxccNegQIAxAB&mstk=AUtExfB_lyjdsfgNhYscNShtgx7OH5G_BS1a89Qjpc8jf1HmEp9lEXuV7qpFnllA8LTO18G4h-iTDG1yiYVZ6vY_8ldqOZqiPvph_pEfsMKs6yID9okhKzeU6Pk8Aws6XYCUNgD96GfJkvJBrr9H9auSz5RGW2iEGIn-ZiNgMGg0RktbAhGQ-RB1TF1L_xvx4H10Y8SsSaaZQhK76E-reMwHWRy-Ax0HKGrkeak8-ECpxeCAxP3xWn-bsOW3Fz6xGU-sBBZ0DEBbZl8Iy0QCJdOBgr_7&csui=3) to define the API's structure, including endpoints, parameters, data types, and authentication schemes. This allows developers to understand and interact with the API, even without access to source code.

**Graphical User Interface (GUI):** a visual interface that allows users to interact with software through elements like windows, icons, and menus.

**GUI documentation**: explains how to use a GUI

**Web service:** is a web-based application that provides resources in a format consumable by other computers. Web services include various types of APIs, including both REST and SOAP APIs. Web services are basically request-and-response interactions between clients and servers (a computer requests a resource, and the web service responds to the request). Sends and receives messages across the web using HTTP to transport the request and response. They are language agnostic

**JSON (JavaScript Object Notation):** lightweight, text-based data interchange format. It is derived from JavaScript object syntax but is language-independent, making it widely used for data transmission in web applications and other contexts.