From Technical Writer to API Documentation Specialist

Learn the basics of APIs and why they matter; why great API documentation is important; how REST, OpenAPI, and Swagger made it possible for technical writers to get involved; and how you can jump in on this opportunity

Eve Karp

APIs Enable Innovation Through Integration

When software manufacturers moved away from distributing via CDs, and enabled businesses and consumers to simply download the product, it was a game-changer. The change made it so much easier to install new software, obtain technical support, and update with new releases.

But the various software remained disconnected from one another. This kept internal business operations inefficient and prevented startups from further innovating on the technology of large companies—until APIs entered the scene.

Today, APIs connect devices, software, and data all around us. They enable developers to focus on building new solutions, instead of reinventing the wheel over and over again. For example, when the developers of Yelp set out to create a business rating app, they tapped into the Google Maps API instead of having to create their own maps from scratch.

An Application Programming Interface—or API—is a set of instructions that tells different software how to interact. Large companies started producing APIs in the 1990's for their internal and customer's needs, but a lack of standardization hindered wide-spread development and adoption. Then in 2000, an engineer proposed the architectural style REST, which set a standard for the core properties that would make up an API.

REST APIs are simpler to develop and adopt

The new standard spurred more companies to start releasing their own APIs, and motivated their clients to integrate the software into their business operations. In contrast, Salesforce chose to not adopt the REST standard for its first API. It required a 400 page PDF manual to install and operate, and as a result, did not gain much traction (ReadMe, 2016).

Software companies dedicate significant resources towards developing and maintaining an API. It can serve as another revenue channel, such as in the case of Google Maps API, which starts at $10,000 per year for developer access (Olaf, 2017).

Alternatively, an API can be used to increase the value that a business provides to its clients. Developers can use the GitHub API connect the version control system to a project management tool such as Trello.

A third objective for releasing an API can be to promote and increase adoption of the core product. Twitter leveraged its API to grow rapidly—at one point, 75% of content came in through the API (Willmott, 2010).

As of January 2018 there are over 19 thousand APIs, according to ProgrammableWeb, which maintains the world's largest directory of public APIs. Currently, about 200 new APIs are released each month.

An API's documentation is as important as the capabilities

As the number of APIs has grown, so have the expectations of developers, who demand that APIs be simple to understand and adopt. While it is crucial that the API performs as expected, the documentation is what makes adoption possible in the first place. Simply put by John Musser, one of the founders of ProgrammableWeb, "the number one reason developers hate your API is because your documentation sucks."

Here lies the catch-22: the skills necessary for producing well-written work are very different from those used for engineering. People can become knowledgeable in one area, but it's unlikely to find (and afford to pay) someone who can do both.

The industry tends to use a "code first, document last" approach. In order to produce great documentation, an engineer must first develop the API, then dedicate additional time to explaining the code to a professional writer. Not only would the engineer have to describe what the code does, but also explain what information an external developer would need in order to understand how to use the API. The writer would have to use their best judgement on how to organize the information and what to include. This approach is obviously time-consuming. Even worse, it risks documentation that is poorly structured and incomplete.

Swagger and OpenAPI standardized documentation

While REST architecture had become a common agreement on the characteristics of an API, the industry still lacked a uniform vocabulary and order for those components. Documentation varied drastically from one company to the next; developers had to spend significant time just figuring out the capabilities of an API and how to implement it.

In 2011, one of the founders of Wordnik created the Swagger API project, as a software that would automate documentation, along with a specification for how the API would be described. It was purchased by SmartBear, then released as open source (Gardiner, 2018).The specification was accepted as the industry standard for describing REST APIs. It was later renamed to the OpenAPI Specification to distinguish it from the software. The name Swagger today refers to a set of tools for designing, developing, and documenting APIs.

Swagger streamlined documentation for APIs, simplifying the work for developers who create APIs and for those who use them. It also made possible for technical writers to join the development team to support engineers and improve the user experience.

Technical writers contribute at each stage of the development process

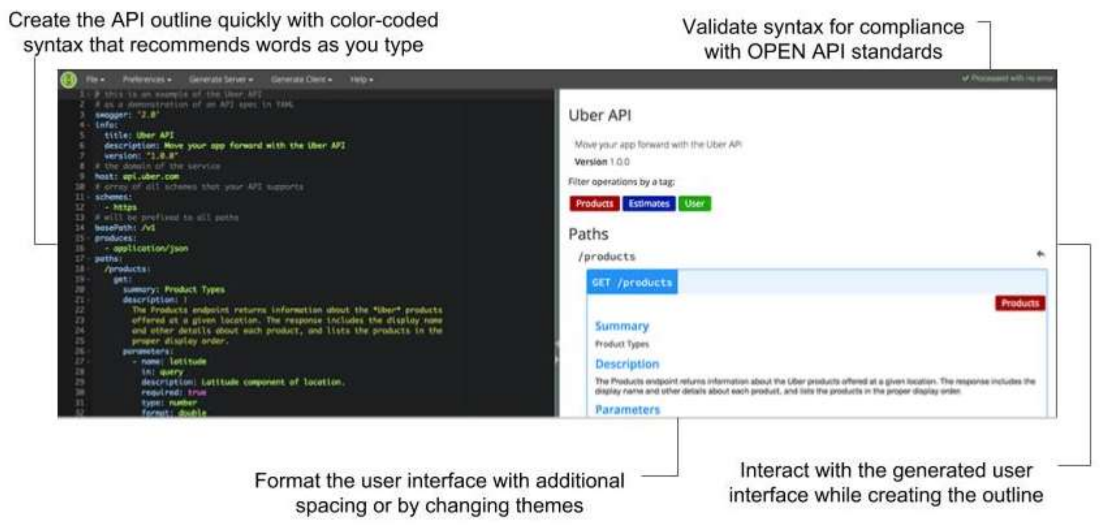
The combination of the OpenAPI Specifications and Swagger tools enable technical writers to take a more active role at each stage of the development process.

Planning

In the planning phase, a technical writer learns about business needs and requirements from the project manager, then works with developers to identify endpoints and agree on terminology. Before developers write any code, a writer first creates an outline for the API in Swagger Editor to serve as a reference for the developers.

Per the specifications, the outline is written in JSON, a syntax that consists of name-value pairs. It contains information on objects, their order, and contents. Swagger Editor ensures the outline is REST compliant and enables developers to interact with the API to understand how each object should behave.

Swagger Editor


Create the API outline quickly with color-coded syntax that recommends words as you type
Validate syntax for compliance with OPEN API standards
Format the user interface with additional spacing or by changing themes
Interact with the generated user interface while creating the outline
File
Preferences
Generate Server
Generate Client
Help
Processed with no error
Products
Estimates
User

```
# this is an example of the Uber API
# as a demonstration of an API spec in YAML
swagger: "2.0"
info:
  title: Uber API
  description: Move your app forward with the Uber API
  version: "1.0.0"
# the domain of the service
host: api.uber.com
# array of all schemes that your API supports
schemes:
  - https
# will be prefixed to all paths
basePath: /v1
produces:
  - application/json
paths:
  /products:
    get:
      summary: Product Types
      description: |
        The Products endpoint returns information about the *Uber* products
        offered at a given location. The response includes the display name
        and other details about each product, and lists the products in the
        proper display order.
      parameters:
        - name: latitude
          in: query
          description: Latitude component of location.
          required: true
          type: number
          format: double
```

Uber API

Move your app forward with the Uber API

Version 1.0.0

Filter operations by a tag:

Paths

/products

GET /products

Products

Summary

Product Types

Description

The Products endpoint returns information about the Uber products offered at a given location. The response includes the display name and other details about each product, and lists the products in the proper display order.

Parameters

Production

In this stage, the developers build and test code. Writers are responsible for producing two types of documentation: reference (pertaining to the code) and non-reference (everything else).

While developers work on the code, writers create non-reference documentation. This commonly includes an API overview, getting started guide, directions on how to get API keys (a user-unique passcode to use the API), status and error codes, glossary, and FAQs (Johnson, n.d.).

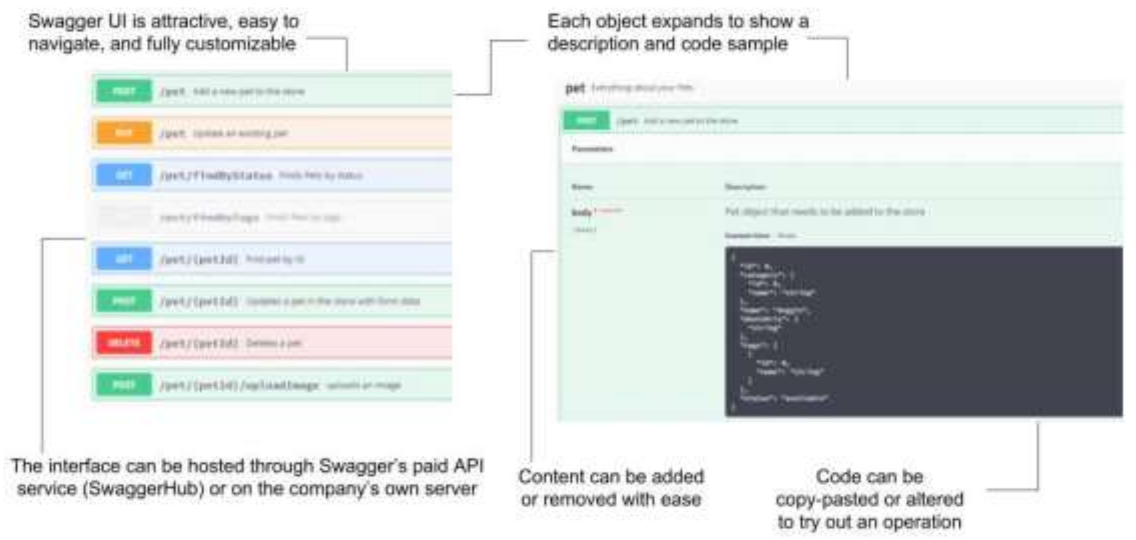
Once the code is complete, it gets plugged into Swagger Inspector. The tool scans through the code to validate that the API has been built to REST standards. More importantly for writers, Swagger Inspector generates a user interface with rudimentary documentation.

Finally, the writer expands on the generated documentation to include comprehensive information on what each component does, how to integrate it, and code samples. The result is organized, attractive, and complete documentation.

Management

Once a completed API is published, the work still isn't over. Companies don't wait for a perfect version of the software before releasing it; they generally put out an acceptable version, then allocate resources to addressing user feedback and complaints. Each time developers modify the code, the writer will need to update the accompanying documentation. Swagger UI enables companies to publish interactive documentation—and to easily update it later. In addition to maintaining up-to-date documentation, a writer may also revise content in response to user feedback or to optimize it for search engines.

Swagger UI


Swagger UI is attractive, easy to navigate, and fully customizable
Each object expands to show a description and code sample
The interface can be hosted through Swagger's paid API service (SwaggerHub) or on the company's own server
Content can be added or removed with ease
Code can be copy-pasted or altered to try out an operation

The OpenAPI specifications inform a writer on how to document an API, while Swagger tools support the writer at each stage. While the tools and tasks may be new to a technical writer, he or she already possess most of the skills necessary to produce great documentation.

Writing API Documentation Is Lucrative and Utilizes Skills That Professional Writers Already Have

According to technical writer recruiter Andrew Davis, companies pay annual salaries between $120 to $150 thousand for full-time API documentation writers and $60 to

$110 per hour for contractors. The Bureau of Labor Statistics confirms that web services will fuel demand for writers: "Employment of technical writers is projected to grow 11 percent from 2016 to 2026, faster than the average for all occupations. The continuing expansion of Web-based product support will result in a greater need for those who can write instruction manuals and communicate information clearly to users."

Professional writers already know how to tailor content and style to different audiences. Generally speaking, they have previous experience producing technical documentation on topics through independent research and working with SMEs. Many have become accustomed to learning a new collaboration or publishing tool each time they change jobs. It may seem like a daunting transition, but the skills needed are not unlike those technical writers already use.

API documentation is just user manual

The documentation for an API is similar to many of the other technical documents a writer will produce throughout their career. API documentation is most comparable to a user manual: it begins with an overview of what the API does and the expected outcome, followed by a getting started guide that explains how to obtain the keys. Each component of the API is accompanied by an explanation for integration, as well as examples and troubleshooting advice.

Developers appreciate documentation that consists of short, simple phrases and code samples. Thanks to the Swagger tools, a writer doesn't need to be a programmer, and with the OpenAPI specification, a writer doesn't need to think like one either. That being said, there are some new skills a writer should learn in preparation for their new work.

Professional writers need to learn just a few tools and skills

Many writers are already experienced with markup languages, such as HTML or XML (for those who need a refresher on markup languages, W3Schools has comprehensive documentation for both HTML and XML. Swagger publishes documentation either in YAML or JSON, two syntaxes that are incredibly easy to learn. Recruiter Andrew Davis recommends writers also familiarize themselves with static site generators—Hugo, Jekyll, Sphinx are a few examples.

In addition to Swagger tools, API documentation writers commonly use Github. Github is a version-control tool used primarily by developers to make collaborating and tracking changes easier. For writers, it is useful to understand how projects are stored and accessed in Github, as well as how to determine who wrote which lines of code, should any questions come up. Github offers several free courses to help beginners grasp the basics.

When it comes to knowledge of programming languages there are two beliefs. One has already been mentioned several times: companies hire developers to write code, so there's no expectation that a writer can program. On the other hand, many companies state a preference in job ads for writers with familiarity with one or more of the following object-oriented languages: Python, JavaScript, C++, and Java.

Sarah Maddox, a technical writer worked for several tech companies and is now with Google, puts it this way: "in some companies, the bar is set quite high – you'll need to be able to write your own code samples and review the code written by the developers. In other companies, it's enough just to read code."

As Python is the most popular programming language—as well as the easiest to learn—there isn't a better time to gain a new skill. Python has a huge online community, and extensive documentation. There is a plethora of tutorials and courses online to pick up the basics.

References

API DOCUMENTATION. (n.d.). Retrieved from SwaggerHub: http://swaggerhub.com/wp-content/uploads/2017/06/Whitepaper_APIDocumentationDX.pdf

Bureau of Labor Statistics. (2017, October 24). Retrieved from Technical Writers: https://www.bls.gov/ooh/media-and-communication/technical-writers.htm#tab-6

Gardiner, M. (2018, 7 26). Better API Desgin with OpenAPI.

Johnson, T. (n.d.). *Integrating Swagger UI with the rest of your docs*. Retrieved from I'd Rather Be Writing: http://idratherbewriting.com/learnapidoc/pubapis_combine_swagger_and_guide.html

Maddox, S. (2013, Aug 17). *Ffeathers: a technical writing and fiction blog by Sarah Maddox*. Retrieved from How to become an API technical writer: https://ffeathers.wordpress.com/2013/08/17/how-to-become-an-api-technical-writer

Olaf. (2017, 3 10). *Monetization: Unlock More Value from Your APIs*. Retrieved from apigee: https://apigee.com/about/tags/api-monetization

ProgrammableWeb Research Center. (2018). Retrieved from ProgrammableWeb: https://www.programmableweb.com/api-research

What is Swagger and Why it Matters. (2016, July 10). Retrieved from Readme.io: https://blog.readme.io/what-is-swagger-and-why-it-matters/