DOCS AS CODE

WHAT INFORMATION DESIGNERS NEED TO KNOW ABOUT THE NEW WAVE OF API DOCUMENTATION

While the quality of the user manual isn't likely to factor into a consumer's choice of one toaster over another, the same can no longer be said for APIs.

by Sybil Scott

MAKING API DOCUMENTATION EASY TO USE, CREATE, AND MAINTAIN

Introduction

Creating and maintaining API documentation can be costly, and it is sometimes hard to judge the return on investment. Therefore, not much attention has been paid to quality documentation historically; having something--anything--was good enough.

That's changing, however. As Uddin and Robillard stated in their *IEEE Software* article, "How API Documentation Fails" (2015), "Because good documentation can help developers work efficiently, it can even serve to promote the API. In contrast, documentation that doesn't meet its readers' expectations can lead to frustration, major loss of time, and even abandonment of the API."

Companies are realizing that there are marketing (i.e. profit) considerations with documentation; as a result, efforts to make it both easy to create and easy to use are ramping up.

Historically, developers have a reputation of being bad at documentation, disinterested in it, or both (Thomchick, 2018).

Enter docs as code.

Since developers are both creators and end users of API documentation, this new philosophy seeks to make documentation as much like code as possible. It allows developers to create and use documentation using the methods and tools with which they are already comfortable and efficient. The task should be less painful because it's closer to their usual processes and skill set.

MAKING API DOCUMENTATION EASY TO USE, CREATE, AND MAINTAIN

Documentation Issues

In Uddin and Robillard's research on the documentation issues developers face (2015), a few presentation factors surfaced, all having to do with the needed content being too hard to find.

- **Bloat** and **Excessive Structural Information:** too much information means users can't separate the content they need from that they don't. Restating structural information found elsewhere buries the important content.
- **Fragmentation:** when the information isn't arranged logically and is therefore spread out over multiple pages. Leads to frustrated searching and too many clicks.
- **Tangled information:** when multiple usage scenarios are included in one description so it's hard to separate out the needed scenario from the others.

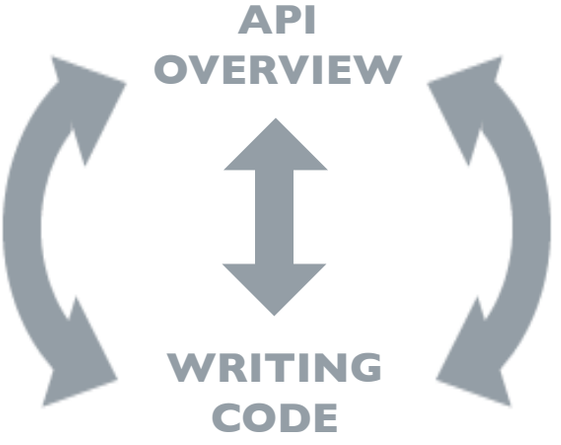
MAKING API DOCUMENTATION EASY TO USE, CREATE, AND MAINTAIN

Further Audience Considerations

While developers are the main audience of API docs, there is still a wide range of factors to consider.

Documentation needs to be usable by both novice and experienced developers, and by those both familiar and unfamiliar with the specific API being used.

That means providing introductory and overview information without getting in the way of advanced users who need to pinpoint specific information.



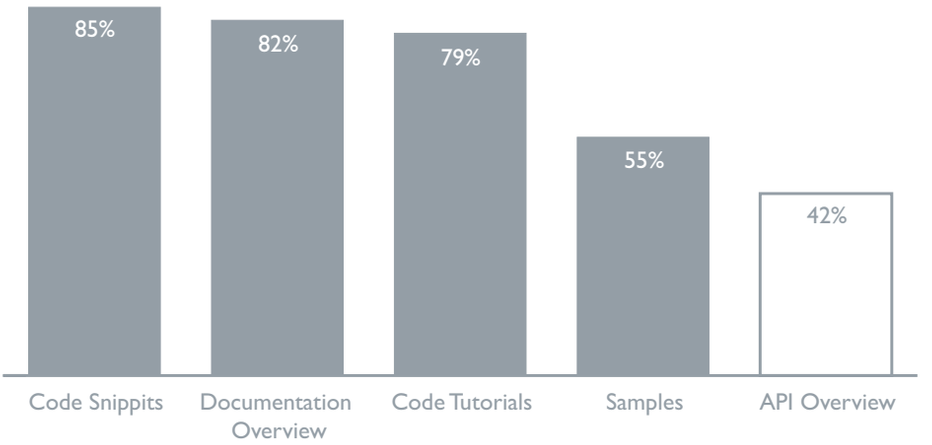
In addition, as outlined in the observation study carried out by Meng et al. (2019) on how developers use documentation, the docs need to be usable by developers who engage different strategies as they approach the API:

- **Systematic developers** tend to read more overview information up front to understand the API more thoroughly before starting a task.
- **Opportunistic developers** try to start coding immediately in a trial-and-error fashion, then seek information (specifically, code examples) that directly relates to the task.
- **Pragmatic developers** use a combination of systematic and opportunistic approaches.

MAKING API DOCUMENTATION EASY TO USE, CREATE, AND MAINTAIN

Key Elements

For their survey of open-source API documentation, Watson et al. (2013) compiled past research and determined five elements that developers consider key to good documentation. They then evaluated 33 of the most popular open-source software projects to determine if they included these elements.



Four of the five elements were indeed found in over half of the sources, three in fairly high percentages. A correlation can therefore be presumed between the elements' inclusion and the source's popularity among developers. Good documentation, then, includes these elements:

- **Documentation Overview**
- **API Overview**
- **Code Snippets**
- **Code Tutorials**
- **Samples**

MAKING API DOCUMENTATION EASY TO USE, CREATE, AND MAINTAIN

Categorization

API Documentation generally includes two categories: **reference and conceptual**. Having both types and organizing it well helps developers of any experience, strategy, and task find what they need.

- **Reference documentation** generally receives the most attention and most often includes
  - Resource descriptions: the information returned by an API
  - Endpoints and methods: how you access the resource and what interactions are allowed
  - Parameters: options to influence the response
  - Request and response examples: samples showing some parameters and a response schema defining possible response elements
- **Conceptual documentation** is often overlooked but equally important. It can include topics such as
  - Getting started tutorials: concisely details the whole process from beginning to end
  - Authorization information: explains the API key or other authentication process
  - Rate limiting: how often developers can call a given endpoint
  - Status and error codes: the classification of the response (successful, server error, authorization issue, etc.) to assist with troubleshooting
  - Quick reference and glossary: provides a list of the API's endpoints and terms

MAKING API DOCUMENTATION EASY TO USE, CREATE, AND MAINTAIN

Navigation

Good navigation ensures that novice users have all of the information they need while also ensuring experts can skip straight to the helpful content.

- **Web delivery** is preferable to a PDF format (Eker and Rondeau, 2016) for many reasons, including better navigation options.
- **Logical organization** goes hand in hand with the well-categorized content mentioned above. It's also important to have wayfinding elements such as breadcrumbs and easily viewed topics to help users navigate the information from any depth.
- **A robust search tool** helps users pinpoint a specific topic without navigation.

MAKING API DOCUMENTATION EASY TO USE, CREATE, AND MAINTAIN

Layout/Design

The Layout of each section must **easily distinguish different types of content,** such as descriptions, data models, and commands. **Code examples** are one of the most useful aspects of documentation and should easily stand out from other content. They should also have a format that is easily copied and pasted (Meng et al., 2019).

In addition, the basic concepts of good design are in play on every page, but especially on introduction pages, which will likely be more text-heavy. **Clear headers, effective white space,** and adherence to the design principles of **contrast, repetition, alignment, and proximity** (Williams, 2015) will make for the most useful and usable content.

MAKING API DOCUMENTATION EASY TO USE, CREATE, AND MAINTAIN

Systems, Processes, and Workflows

While less directly related to information designers, it's good to understand how docs as code manifests in ways that motivate and help developers (the subject matter experts of APIs) write documentation by using the same systems, processes, and workflows as coding.

- **Use a lightweight markup language** such as Markdown, a simplified form of HTML (*More about Markdown*, n.d.).
  - Uses a code-like syntax
  - Works in text-file formats using any preferred code editor
  - Treats the files with the same workflow and routing as code
  - Focuses on content instead of formatting
- **Store docs in the same version control repository as the code itself,** which is more convenient and encourages documenting while coding instead of after the fact. (Johnson, 2017)
- **Automate when possible**: emerging technologies like Swagger can automate certain processes, allowing developers to focus on the more complex content. (*Swagger,* n.d.)
- **Document iteratively and incrementally** by constantly reviewing, testing, updating, and publishing docs as opposed to leaving the documentation for the end of the process and considering it one-and-done. (*Principle 9*, n.d.)

MAKING API DOCUMENTATION EASY TO USE, CREATE, AND MAINTAIN

References

- Docs as Code—Write the Docs. (n.d.). Retrieved February 22, 2020, from https://www.writethedocs.org/guide/docs-as-code/
- Johnson, T. (2016). Documenting APIs: A guide for technical writers. Documenting APIs: A guide for technical writers| Document REST APIs. I'd Rather Be Writing, 27.
- Johnson, T. (2017, June 2). Limits to the idea of treating docs as code. I'd Rather Be Writing. https://idratherbewriting.com/2017/06/02/when-docs-are-not-like-code/
- Margaret Eker, Jennifer Rondeau - Docs as Code: The Missing Manual. (n.d.). Retrieved February 24, 2020, from https://www.youtube.com/watch?v=JvRd7MmAxPw
- Meng, M., Steinhardt, S., & Schubert, A. (2019). How Developers Use API Documentation: An Observation Study. 11.
- More about Markdown. (n.d.). Documenting APIs. Retrieved March 3, 2020, from https://idratherbewriting.com/learnapidoc/pubapis_markdown.html
- Principle 9: Iterate and increment on content following an agile approach. (n.d.). I'd Rather Be Writing. Retrieved March 3, 2020, from https://idratherbewriting.com/simplifying-complexity/iterative-and-increment-on-content.html
- *The Best APIs are Built with Swagger Tools | Swagger.* (n.d.). Retrieved March 7, 2020, from https://swagger.io/
- Thomchick, R. (2018). Improving access to API documentation for developers with Docs-as-Code-as-a-service. Proceedings of the Association for Information Science and Technology, 55(1), 908–910. https://doi.org/10.1002/pra2.2018.14505501171
- Uddin, G., & Robillard, M. P. (2015). How API Documentation Fails. IEEE Software, 32(4), 68–75. https://doi.org/10.1109/MS.2014.80
- Watson, R., Stamnes, M., Jeannot-Schroeder, J., & Spyridakis, J. H. (2013, September). API documentation and software community values: a survey of open-source API documentation. In Proceedings of the 31st ACM international conference on Design of communication (pp. 165-174).
- Williams, Robin. *The non-designer's design book: design and typographic principles for the visual novice*. Pearson Education, 2015
- XML attributes and examples: Learn API Documentation with JSON and XML. (n.d.). Retrieved February 23, 2020, from https://www.linkedin.com/learning/learn-api-documentation-with-json-and-xml/xml-attributes-and-examples