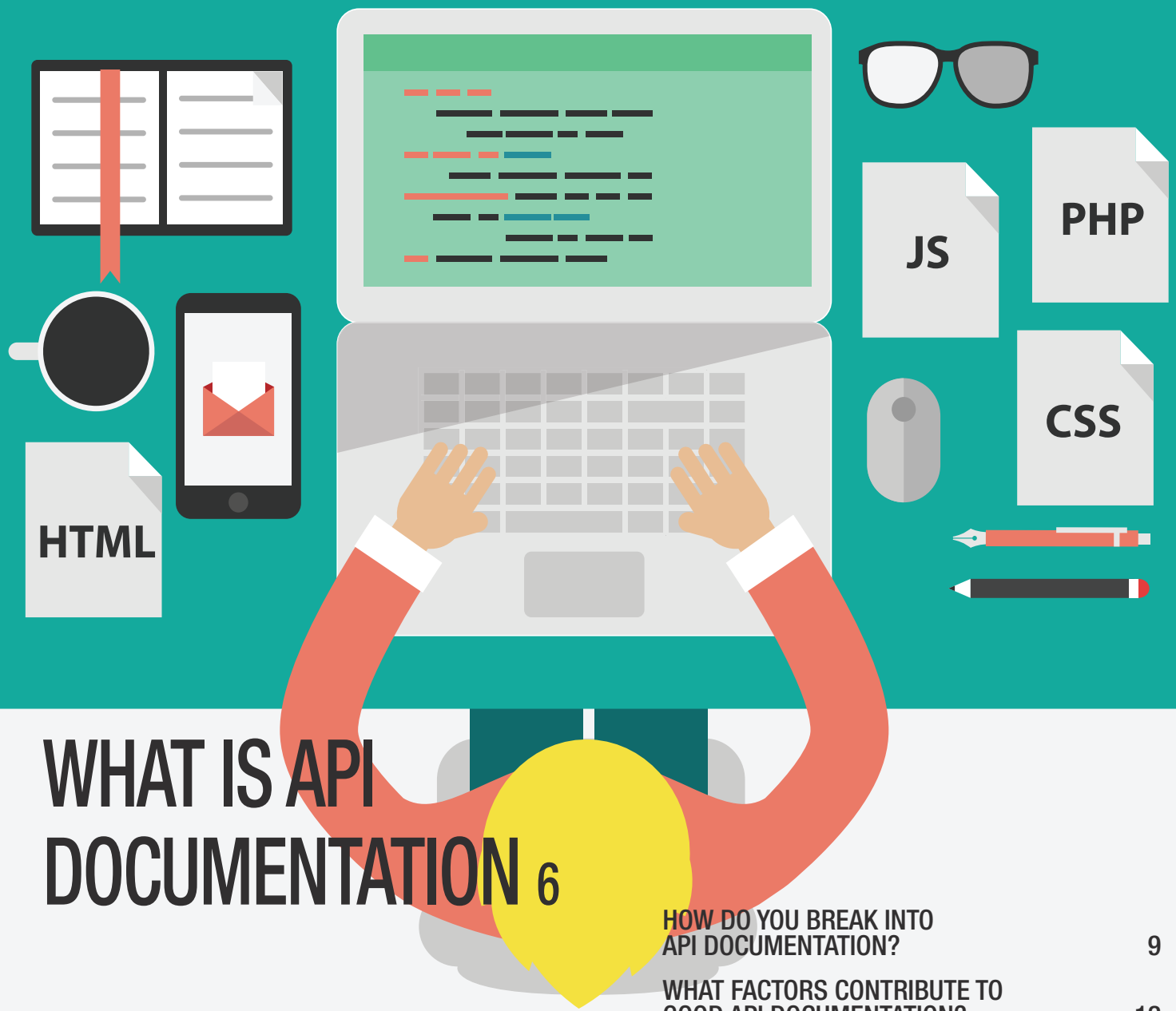
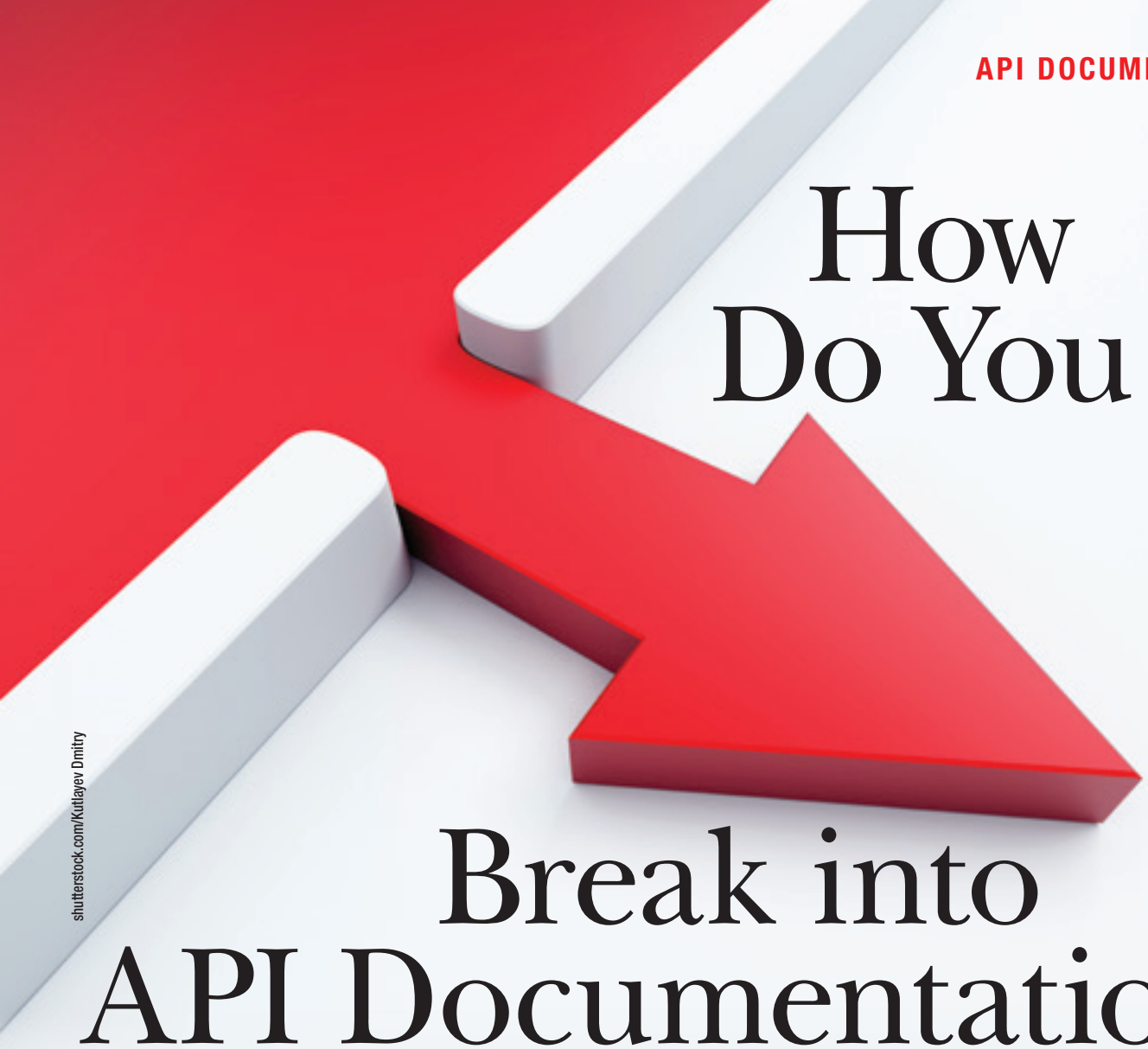
SEPTEMBER 2014

intercom

THE MAGAZINE OF THE SOCIETY FOR TECHNICAL COMMUNICATION



WHAT IS API DOCUMENTATION 6



shutterstock.com/Kutlayev Dmitry

How Do You Break into API Documentation?

By MARY CONNOR | *Member*

The Market

Large numbers of us broke into technical writing by documenting software that runs on personal computers. Over the years, we have watched sales of those machines and their software packages stagnate. The manuals and help that we created grew gaunt under the relentless pressure to do it faster, cheaper, and lighter. With the rise of support-written knowledge bases (think ZenDesk, *www.zendesk.com/*) and community-generated content, many wondered if conventional documentation jobs were destined to fade.

Now we're witnessing dizzying growth in high tech, driven by the shift to ubiquitous computing through mobile devices and by distributed, cloud-based storage and services. All of this growth depends on there being solid and successful APIs to interconnect our devices and services. A quick look at these resources shows that the range of API offerings is stunning:

- API Directory (ProgrammableWeb, *www.programmableweb.com/apis/directory*)
- Government APIs (data.gov, *http://catalog.data.gov/dataset?res_format=api&_res_format_limit=0*)
- Open APIs (Wikipedia, *http://en.wikipedia.org/wiki/List_of_open_APIs*)

Just how successful these APIs are has everything to do with how quickly and easily programmers can become power users of these new interfaces, and so everything to do with documentation and tutorials, and us! That's the good news: the job market for programmer writers has become habañero hot. Since I've put full-time API documentation experience on my profile, I've started receiving an eye-popping stream of unsolicited invitations for positions with tech companies that are everyday names to the wired crowd.

The Salaries

Not only are *more* API jobs becoming available, but these are jobs that pay *significantly* higher than conventional ones. In my experience, you would command hourly rates that are 1.5 to 2 times that of non-API jobs. And, while not a

salary issue, another feature of these jobs has tremendous monetary value: API jobs are more likely to offer the option of working remotely. However, the on-site jobs do tend to be concentrated in specific metro areas (such as San Jose), which have a higher than average cost of living.

Perhaps more relevant to our Great-Recession-sobered perspective, being an API writer means being a sought-after talent that's in relatively high demand, with broad options and mobility. And acquiring the skill-base of programming writing is *itself* insurance, as it grants access to jobs in more technically demanding shops, as well as job titles with more technical requirements, such as business analyst. Any way you look at it, your salary security improves.

The Working Environment

I associate the working environment for API writing with being embedded with the development team that codes the API. If you have worked within feature-based teams or on an Agile/SCRUM team, you are already familiar with this dynamic, being the "lone writer" that supports a group of developers and testers, usually in service of a product owner or product author. Your supervisor, however, may be the development manager. Having a developer for a boss can have huge implications for the expectations you face, the tools you get, the deliverables you own, and the content strategy that might be missing or discounted.

The most common Myers-Briggs personality type I've encountered on these teams is INTP, a fact that I bring up for this reason: your ability to relate to these developers and to win their respect means your success or failure in this job. All things are possible, but I think highly extroverted writers, for example, could face an uphill battle with team acceptance. I have witnessed several writers be rejected by developers who were repelled by their manner, and it's not a situation that just resolves itself with time and friendly words. If you don't seek nerdvana among deep thinkers who will call you out on your logical errors, you might want to reconsider this path.

The Skillsets

As an API writer, you can find yourself exercising a surprisingly broad set of skills, a set that you will find either exhilarating or terrifying (or both, if you're human):

- **Development—**You will need to be able to use the shop's chosen IDE (such as Visual Studio or Eclipse), work with their code files, run local builds, master their version control system, learn their code commenting style, and install, update, and *own* the API reference generation. Yes, unfair as it sounds, you will be expected to tackle all of the programmer and QA tools in addition to your own documentation tools.
- **Business Analysis—**You will find yourself in the position to serve as a business analyst for the interface. In practical terms, this means using your communications expertise to devise and argue for stronger, intuitive, and consistent object naming and organization, and using your research to compare the API both with competitors and with implicit or explicit industry standards.
- **User Experience—**As with all documentation jobs, you will still strap on your gloves to fight for a better user experience. In the case of developer documentation, you will look for ways to improve TTFHW (Time to First "Hello World"), how API users get introduced to the product, how they get set up, how they learn, and how they get help, both from support and their peer community. Chances are, you will be the one person on the team with the drive and experience to make headway on UX/CX, perhaps getting involved with support initiatives.
- **Technical Communication—**As if this all weren't enough, you still need to be an excellent communicator, such as Andrew Davis describes so well in his article, "How to Find Good Programmer Writers" (*www.contentrules.com/blog/how-to-find-good-programmer-writers*). Beyond your powerful writing, bring the full package of skills that separates the great from the competent: be empathetic, responsible, curious, humble, resourceful, attentive, autodidactic, passionate, precise—the writer that every project manager would weep to lose.

The Transition

To transition from a regular technical writer who documents GUI apps for end users to a technical writer who creates API help for developers, you need to be well along the path to being a developer in your own right. Yes, a developer, even if you've never taken a programming course in your life.

Happily, most documentation jobs offer several key areas in which you can develop your technical chops while serving the larger good:

- **Authoring Tool Plumbing—**Be the person on the team who knows the authoring tool inside out and backward, who can install it, upgrade it, back up and restore it, configure it, and import and export its content. Be the Web researcher who finds out how to leverage its obscure features to solve a problem in your help system; be the designer who braves editing the complex template files to add Google analytics or to achieve special Javascript effects that make the CEO take notice. Be the tool agnostic who can do proof-of-concept projects in any number of new and unknown tools, without any handholding and on your own initiative. Be the analyst who studies other help systems for useful enhancements and persuades a developer to write you a script that will help you implement one on your own.
- **Build Engineering—**Be the person on the team who figures out how to automate your help builds, even if the tool doesn't support it directly. Find ways to integrate the help source, builds, and outputs into the existing build process used for the code. Figure out how to publish draft help internally as well as in beta and general availability(GA) versions. Come up with a way to fake conditional build behavior so that internal staff have access to additional content that will never be released.

Be the person who never stops finding new opportunities to single-source content across file formats and disparate uses. Be the one who learns enough about FTP to automate the publication of content to the Web.

- **Batch and Automation**—Be the person who notices that manual doc processes are being repeated and find ways to automate them, starting with simple DOS batch files and Windows Task Scheduler. Improve upon processes that require you to remember to do the right thing in the right sequence, such as to copy files to network locations or manually rename files to show status; find a better way. Use the scripting tools built into your business apps (such as Visual Basic for Applications) to automate steps and transformations that improve documentation quality, such as Word macros to correct all sizing and formatting problems in hundreds of screenshots. Make full use of variables and templates in your tools, and explore all of the fields available, along with the parameter switch options they offer.
- **Internal Developer Content**—Be the person who volunteers to help write and edit wiki pages to guide new developer hires. Attend every language/tool/methodology training that your shop offers to the development group. Offer to manage, edit, organize, and publish the development artifacts for the group. Help business analysts create clearer and lovelier diagrams and slide decks, and study the content as you go. Attend "brain-dumps" as a scribe, and post your notes and Web conference recordings in the repository (creating it, if need be). Be the go-to person for knowledge management strategy.

What I described above is the path that I journeyed down, which led me to take over API reference projects at my job and then to transition to full-time API documentation responsibility. If those descriptions leave you cold, sweating, and feeling as if you'd want any job but that, you might not enjoy being in the hot seat as an API writer.

The Training

If you are interested in the API career path, strike the word "training" from your consciousness and substitute the word "studying," because the culture of code development is one of unceasing self-education. Even when developers nominally finish training courses on new languages and methodologies, they know that they are never done researching and improving their grasp of the material, which itself will never stop mutating into new versions and forms. Development's tribal code is all about building and maintaining your own technical credibility, without hand-holding.

To make this more real for you, I'll describe how I've witnessed API developer interns being treated. They are assigned to serve a specific team member, who piles them with tasks and problems to solve. To a fair degree, interns are allowed to twist in the wind, to struggle to get their environments working and their projects moving. What at first seems sadistic reveals itself to be highly illuminating: the team gets to observe how quickly the interns teach themselves the new domain, how resiliently they study, acquire help, and handle frustration, and how cleverly they conquer the challenges blocking their projects. Even if the interns resent the intellectual hazing, they emerge wiser and full of confidence that they can handle their next job assignments. They join the tribe.

So, to be a successful API writer, be prepared to join the tribe. Think of yourself as a developer newbie, not one that will threaten to take the job of anyone on the team, but one who will learn just fast enough to be able to teach other newbies. To start, take whatever programming courses are available to you; community colleges and school district community education programs are great resources. But don't let logistical challenges stop you—you can now learn programming completely on the Web, sitting on your couch. Try out courses from the sites below so that you can find some that fit your learning style (see this excellent survey: *http://designzum.com/2014/03/07/best-resources-to-learn-code).*

- Coursera (*https://www.coursera.org/*)
- OpenCourseWare Consortium (*http://ocwconsortium.org/*)
- Codecademy (*http://codecademy.com/*)
- Code Avengers (*http://codeavengers.com/*)
- Code School (*http://codeschool.com/*)
- Treehouse (*http://teamtreehouse.com/*)
- LearnStreet (*http://learnstreet.com/*)
- Udacity (*https://udacity.com/*)
- CodeHS (*http://codehs.com/*)
- Khan Academy (*http://khanacademy.org/cs*)
- Scratch 2.0 (*http://beta.scratch.mit.edu/*)
- SQLZOO (*http://sqlzoo.net/*)

As you complete coding courses, look for opportunities to code solutions to small problems at work, church, or home, and remember to record your completions and outputs on your resume and LinkedIn profile. At the same time, start following the blogs and sites of more technical tech writers, and join relevant LinkedIn groups, such as the one for API documentation (*http://linkedin.com/groups/API-Documentation-3709151*).

The Payoff

Is API documentation more rewarding and fulfilling than creating other types of documentation? For me, yes, for surprisingly non-career reasons. I love having dared to face my personal Everest, my fear that I wasn't capable of doing this work. I love being embedded with developers and testers as a their supporter and peer. And, sweetest for any tech writer, I love that these users, these API users, actually RTFM.

A past president of STC Austin, Mary Connor has an MA in English and has been performing and teaching technical communication for 20 years. Currently at 3M, she has created developer documentation and training materials for REST/SOAP services at Telogis and several generations of API products at Advanced Solutions. She is keen to capture tribal knowledge and to single-source/automate documentation production, which opens space for the important stuff: powerfully clear writing and diagramming. You can contact her through her blog at www.cleverhamster.com.