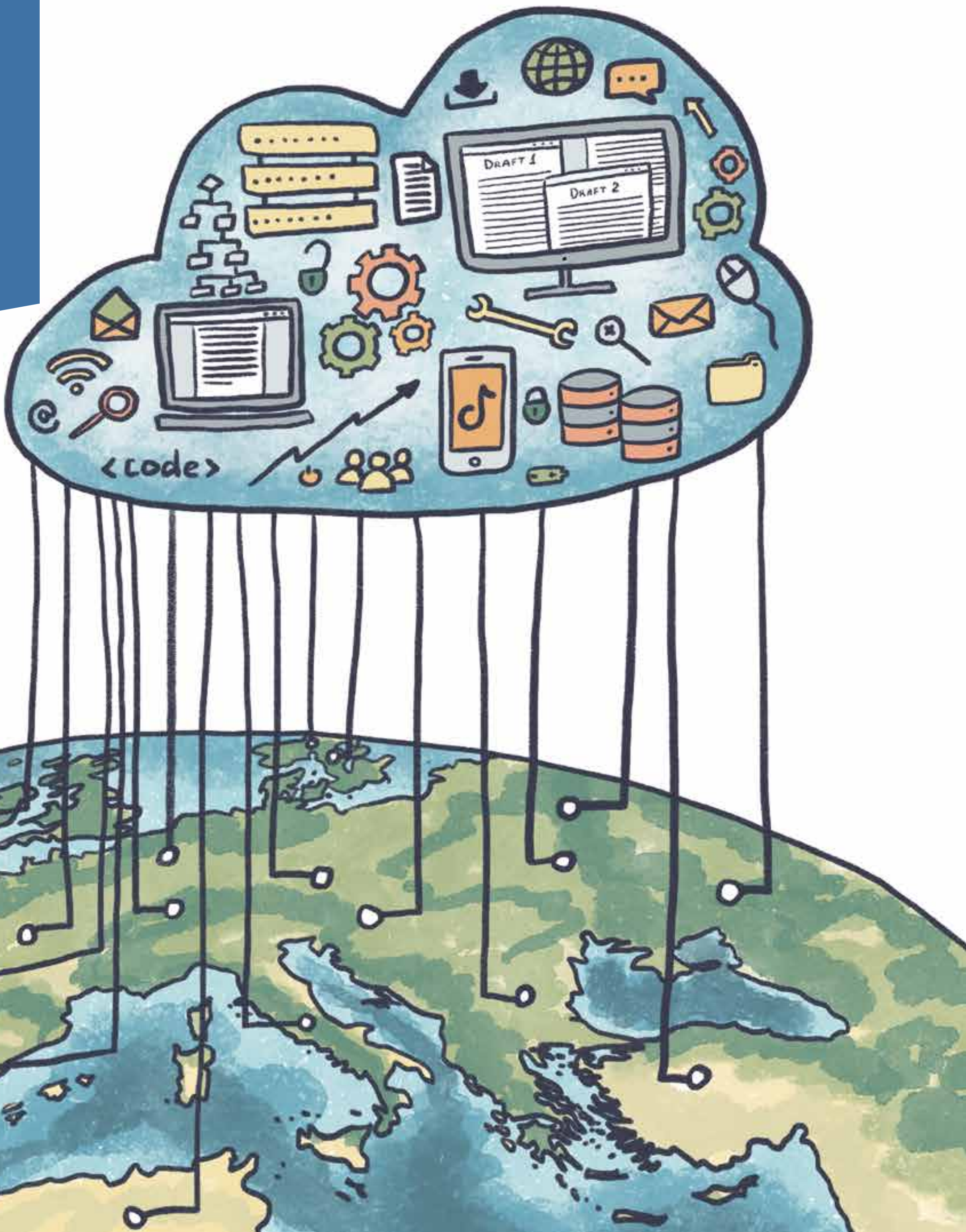
Communicator

The Institute of Scientific and Technical Communicators

Autumn 2023

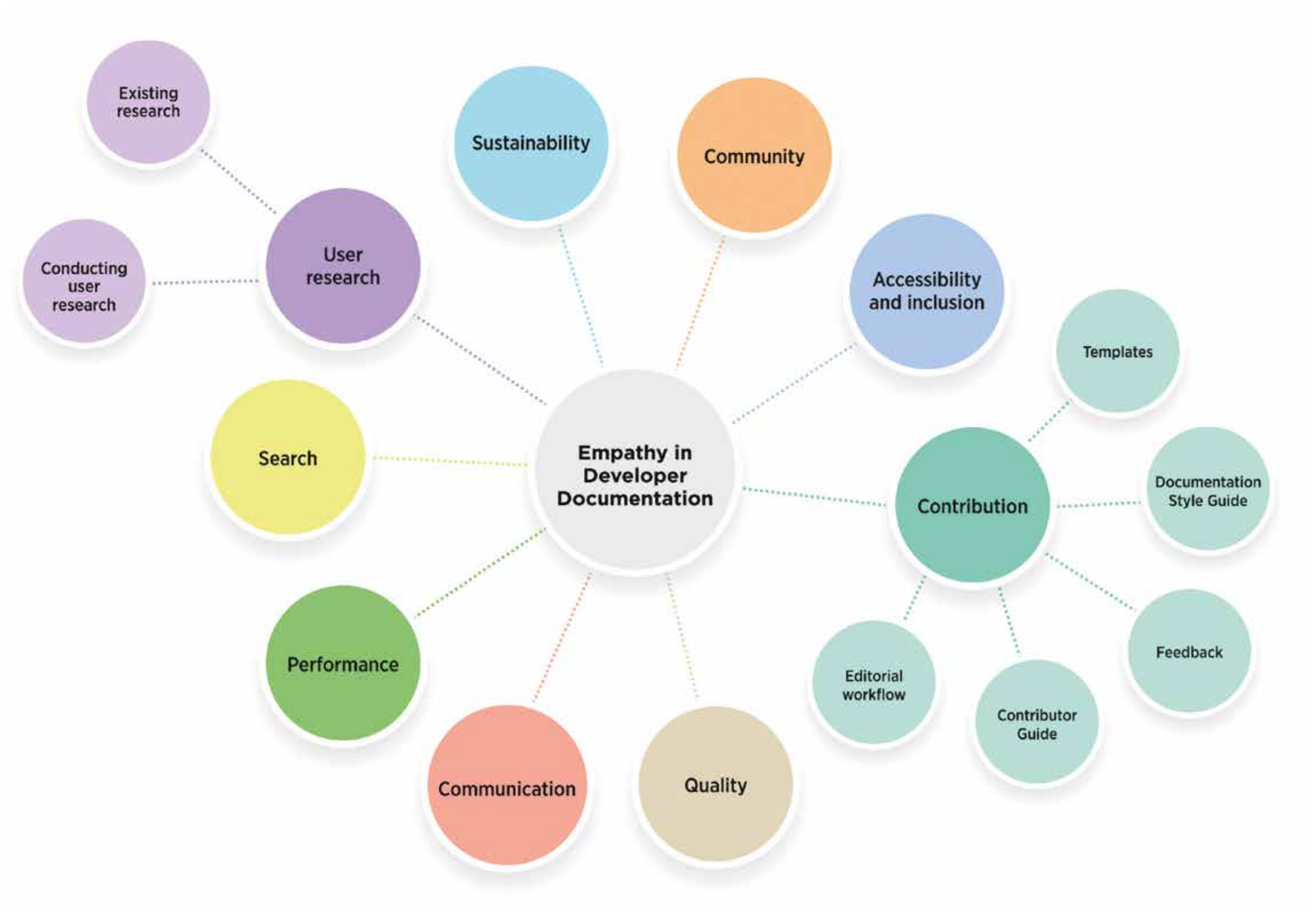
Writing for SaaS

- Establishing a company glossary
- The key to successful developer documentation
- The cost of the cloud



Empathy in Every Detail:

The Key to Successful Developer Documentation by Diana Lakatos.



Developer documentation provides information, guidance, and examples for using APIs, frameworks, and software platforms. It aids in onboarding, and strengthens developer relations by keeping users informed, fostering community, and encouraging collaboration. When you're working on developer documentation, you have the advantage of knowing its primary target audience—developers—from the start. While there are various other audience segments, the majority of users are indeed developers who vary widely in skills, approaches, and learning methods. Because of this, you might find that some of the aspects discussed in this article are developer-centric. However, most of the strategies detailed here are also designed to enhance the experience for other audience segments, including people working in support, product owners, business analysts, project managers, DevRel practitioners, developer advocates, and technical community managers.

> *Aligning with the ISTC Communicator Autumn issue's theme of 'Writing for SaaS', I'd like to add that many leading examples of developer documentation come from SaaS companies. Stripe, Twilio, and GitHub, for instance, are frequently cited as benchmark references in developer documentation.*

While this article explores several key aspects, there are naturally many more to consider. The focus is on cultivating a mindset where, at each stage and through various facets, you can empathetically connect with your users and find the best ways to serve their needs.

The type of empathy required here entails recognizing, valuing, and understanding the thoughts, experiences, and emotions of users interacting with your documentation. To be genuinely helpful, you must grasp their experience from their perspective. It's not about making assumptions or guessing their thoughts. Instead, there are various methods available to truly understand them. This leads us to our first aspect: user research.

User research

User research is the foundation for understanding the people who engage with your documentation. Research should be woven into every phase of your documentation process as it continuously offers new insights about user expectations, behaviours, needs, and motivations via systematic investigative methods—enabling you to tailor your content and documentation site features to real user needs.

To gain insights into developers at large and specifically those who engage with your documentation, you can explore existing research and, of course, conduct your own.

Existing research

Delving into existing research provides insights into the tactics and techniques developers employ, their preferred workflows, their interaction with documentation, and their underlying thought processes.

Studies by Clarke[1], Watson[2], Meng et al.[3], and others illuminate how developers engage with documentation. These insights help you grasp their problem-solving strategies, risk assessments, and learning methods. For instance, Clarke identified three primary approaches to coding and learning: systematic, opportunistic, and pragmatic. Systematic developers are cautious and thorough. They prioritise the need to understand technology deeply before usage. They tend to read documentation extensively and adhere to given guidelines, aiming for clean, efficient code. Opportunistic developers dive right in, sourcing information just-in-time for the task at hand. They often veer from standard procedures, searching the web while coding. Pragmatic developers combine systematic and opportunistic approaches. They learn enough to begin and consult resources as challenges arise. Each developer's approach might shift based on the task or context. Providing content for all approaches is a genuine challenge. You can address this by continuously learning about your users and adapting your content, navigation, and documentation site features to meet their needs. Fundamentally, all developers require: clear concepts, procedural information, references, code samples, intuitive navigation, and efficient search functionalities.

Your documentation will cater to a range of developers, varying in experience and specialisation. This includes junior to senior developers, backend to frontend experts, and even your own team members. You also have to consider factors like language proficiency, neurodiversity, and varying abilities. It's vital to gauge and periodically reassess the needs, abilities, and expertise of your audience, as it heavily influences your content, structure, and onboarding strategies.

The patterns gleaned from research offer a framework for understanding developers' thought processes and behaviours. However, it's crucial to remember that every developer is unique. To gain deeper insights into the developers and other target audience segments using your documentation, conducting your own user research is essential.

1. Clarke, S. (2007) What is an end user software engineer? — Dagstuhl Seminar Proceedings, Schloss Dagstuhl-Leibniz-Zentrum für Informatik.
2. Watson, R. B. (2015) The effect of visual design and information content on readers' assessments of API reference topics (Doctoral dissertation)
3. Meng, M., Steinhardt, S., Schubert, A. (2019) How Developers Use API Documentation: An Observation Study — Communication Design Quarterly (January 29, 2019)

Conducting user research

Regular user research should be a continuous activity throughout the documentation development cycle starting from the discovery phase. Several user research methodologies are available, and choosing the right method often depends on the specific phase of documentation development and the problem you are trying to solve. The discovery phase, for instance, benefits from field studies, diary studies, and both user and stakeholder interviews. As the project progresses, workshops can be utilised to develop personas and identify essential tasks. The prototyping phase can be enhanced with usability studies and accessibility evaluations. As the documentation nears completion, surveys and analytical reviews offer valuable feedback and insights.

Regardless of the chosen method, the goal remains consistent: identify patterns across user interactions, comprehend the underlying reasons for specific behaviours, and leverage these insights to guide documentation decisions. Through user research, you can also develop personas—semi-fictional archetypical users whose goals and characteristics represent the needs of target audience segments. Personas can assist in maintaining a user-focused approach during all stages.

Conducting research for developer documentation is a complex task, so you'll greatly benefit from having a UX Researcher on your team. As you conduct regular user research, you'll discover its profound impact on the quality of your developer documentation, including its relevance and usefulness to your users.

Community

The community-driven documentation approach aims to involve users into every stage of documentation development. This method fosters a culture of open dialogue and transparency between the documentation team and its community. The early and consistent feedback loop ensures immediate testing and validation of all assumptions. With the community playing an active role—offering insights, adding content, and influencing the documentation's trajectory—the final product resonates more closely with their needs. By actively engaging users early and maintaining constant collaboration, you can swiftly validate plans and make necessary adjustments. This approach saves time and money by minimising the risk of producing large sections of content that don't align with user needs.

Maintaining this constant collaboration builds trust. When the entire documentation process is transparent, the community can track the evolution of content and feel confident that their feedback is valued.

Forge strong relationships with your community from the beginning. Ways to involve users include regular user research, engaging with community members across multiple platforms, and providing them various opportunities for contribution.

Contribution

I believe that every piece of input—be it from team members, users, clients, partners, or even individuals from completely different disciplines—that enhances or propels your documentation forward, qualifies as a contribution.

Developers contribute in various ways, including supplying technical writers with information and code examples through drafts, descriptions, or videos. They might add new topics to your documentation or adjust and update existing ones. Additionally, they review changes made by other team or community members, test code examples, report issues, and highlight areas for improvement.

Other contributors can provide invaluable insights. For example, account managers often share feedback from their client interactions, pinpointing areas that require more support. Those in support roles can quickly identify content gaps or areas needing enhancement. External contributors, driven by their passions, may offer suggestions to improve clarity, accessibility, or inclusiveness in the documentation.

Feedback

Consistent communication with your users is essential for refining documentation. By observing their interactions, you'll glean feedback and questions that highlight areas for enhancement. For example, if users routinely ask questions on various channels due to documentation gaps, it's wise to update those sections to provide the necessary information for future users.

Feedback can encompass everything from specific product features and documentation structure to navigation and the clarity of content. To facilitate the feedback collection process, consider embedding a feedback block at the bottom of each documentation page. This could allow for simple inputs, like a smiley rating, but also give users the option to provide detailed suggestions. In our experience, such feedback mechanisms have been helpful in indicating which documentation sections might need further refinement.

Editorial workflow

For content contribution, implementing an efficient editorial workflow is crucial—one that seamlessly accommodates both internal and external contributors, and ensures they receive prompt and precise feedback.

When aiming to encourage contributions from developers, it's essential to establish an editorial workflow that makes it easy for them to contribute. In my experience, Docs as Code is a great fit for developer documentation:

- The primary contributors—developers themselves—are already well-versed in the tools and workflows associated with the Docs as Code approach. This means they can start contributing without the learning curve, thereby eliminating potential barriers.
- For teams and communities spread across multiple time zones, Docs as Code's inherent support for asynchronous communication allows for timely contributions, feedback, and edits without expecting instant responses.
- Docs as Code permits contributors to work within their preferred environments.
- While tailored for developers, you can make this approach work for non-developers, too. This might include simplifying certain processes, using easily understandable markup languages, and offering additional support like online text editors.

Reviews in a Docs as Code environment offer a great opportunity to engage with both internal and external users, fostering collaboration to refine documentation. Often overlooked in comparison to other communication channels, they play a unique role, especially in open-source documentation, reflecting team dynamics and shaping community culture. Throughout the review process, prioritise clear and empathetic communication, advocating for a range of diverse insights, and fostering an atmosphere of mutual respect and understanding.

A Docs as Code approach also enables you to incorporate automated tests and linters into your review process to ensure high-quality content, accessibility, and inclusion in your documentation. Linters are tools that automatically check your code or documentation against specific rules. For instance, you can check for any broken links, scan for grammatical or spelling mistakes, identify non-inclusive language, or assess the readability of your content.

Contributor Guide

In addition to offering a familiar editorial workflow, first-time contributors to your documentation require a clear starting point. A Contributor Guide can provide the essential information they need to dive in. A well-crafted Contributor Guide should be both succinct and comprehensive. It should include details on giving feedback, a tutorial for updating or adding new content, links to the Style Guide with a succinct overview of its main points, and references to templates (if available) along with explanations of their purpose. Also, ensure there's a direct channel for contributors to communicate any questions or concerns.

Documentation Style Guide

A documentation style guide is an essential tool for maintaining consistency, clarity, and effectiveness in your written content. It ensures that your writing remains grammatically sound and clear and is also tailored to the unique needs of your target audience. Beyond text, it offers guidelines for integrating images and videos, making it an indispensable resource for contributors, reviewers, and editors.

The creation of a style guide is iterative; it evolves as you add to it, test its guidelines, and gather feedback throughout the content creation process. This guide will be both a reference and a benchmark, ensuring a consistent quality and approach in your documentation. As questions or decisions arise during content production, adding them to your style guide will organically grow it into a comprehensive resource for all contributors.

Templates

Templates are like blueprints with set content and placeholders. These placeholders actively direct where and how to add varying content, help with the desired format, such as titles, and may include certain parameters like character limits.

By using templates designed in your selected markup language, even contributors who are not familiar with the language can more easily create content. Beyond simplifying the content creation process, templates bolster documentation consistency, ensuring each topic adheres to a uniform structure.

Initial documentation structures often revolve around foundational content types, such as tutorials that delineate tasks, concepts that provide context, and references like API references. As your documentation expands, you'll need to introduce additional templates for content types like release notes and use cases.

While streamlining the contribution process with supportive tools and workflows is essential, it's equally important to consistently recognize and reward contributors for their valuable input. Doing so creates a positive experience for contributors and encourages ongoing engagement and involvement in your project.

Communication

Effective communication is crucial in developing and maintaining documentation. By offering a variety of channels for interaction, you can ensure that your community members remain informed and actively engaged while granting them the liberty to choose the platform they're most comfortable with. Regularly sharing updates on new features, improvements, and other changes helps in fostering a culture of transparency and collaboration.

Real-time communication channels, such as chats, serve as platforms for immediate feedback and discussions. Such direct interactions often yield valuable insights, serving as a direct indicator of the community's needs and concerns, subsequently helping to refine and enhance the documentation further. Hosting video conferences can deepen these interactions, allowing for more structured presentations, feature demonstrations, and strategic discussions, all of which can help align the visions of both the team and the community.

For more structured communication, incorporating a Q&A interface on community platforms can be beneficial. These platforms can spotlight the most pressing concerns, frequently asked questions, or emerging trends within the community, serving as a guide for documentation improvement.

Harnessing the power of varied communication channels is not just beneficial but essential for a documentation's evolution, ensuring it remains responsive to the ever-evolving needs of its community.

Performance

A fast documentation site ensures that users stay engaged and benefit from a superior user experience, helps the site rank higher in search results, and contributes to the site’s sustainability. The Google/SOASTA Research from 2017 revealed that a mere two-second increase in page load time, from 1 to 3 seconds, raises the bounce probability by 32%. If this load time extends from 1 to 5 seconds, the bounce likelihood increases to a staggering 90%.

Performance measurement tools provide insights into a site's performance on both desktop and mobile platforms and offer recommendations for improvement. It's important to understand how these tools calculate performance scores and the impact of specific metrics on the overall score. Conducting regular tests across various pages helps establish clear performance benchmarks tailored to the unique attributes of each page. Prioritising performance from the beginning and implementing consistent refinements is key to achieving optimal results.

Search

Both the research I previously mentioned and the results of the user research we’ve conducted over the years indicate that developers primarily visit your documentation site with a goal: to learn or to address a particular issue. It's imperative that you guide them efficiently to the information they seek. While having a well-organised site navigation is beneficial, the significance of an effective search mechanism cannot be understated. There are two key types of search to consider:

- On-site search: This allows users to easily locate information within the documentation site. Incorporating analytics into this can be invaluable, as it provides insights that can be used to further refine and improve the documentation based on what users are searching for.
- SEO (Search Engine Optimization): When developers search for different topics on search engines, you have to make sure they find relevant information from your site in the search results. This boosts the visibility of your content and ensures that developers are directed to trustworthy and relevant sources.

Information quality

Documentation serves as the bridge between the developer and the product, making the quality of information contained within it essential to developer success. The inherent value of information is determined by its quality, which refers to the attributes and characteristics that make it valuable to the intended audience.

A key measure of information quality is its accuracy. Documentation is considered accurate when the content is factually correct, up-to-date, and free from errors. Consistency is equally important, ensuring that no information is contradictory. This extends to sample code in repositories, emphasising the need for rigorous testing and validation.

Completeness is another quality requirement. To cater to user needs efficiently, documentation should encompass all essential details. Whether documenting procedures, API references, or other technical details, the information should be both exhaustive and concise. Completeness ensures users are well-informed, but it's equally vital that they aren't overwhelmed by unnecessary detail.

Readability, on the other hand, determines the ease of consumption. Explaining technical detail requires a blend of precision and simplicity. Write in clear, concise language, with technical jargon adequately explained. Information architecture also plays a crucial role here, guiding users seamlessly through the documentation.

Relevance of information means that it should be applicable to real-world scenarios and use-cases developers might face. To craft relevant content, you must develop a deep understanding of your audience.

Accessibility and inclusion

Accessibility allows people with disabilities to effectively use digital products. Inclusiveness enables people from various backgrounds and circumstances to actively benefit from these products and services.

According to the World Health Organization, approximately 16% of the global population has some form of disability, and nearly all individuals will experience disability at some point, either temporarily or permanently. This indicates the importance of accommodating users who may have visual, auditory, cognitive, or motor impairments, meaning that your documentation should be tailored to accommodate users who may employ screen readers, voice recognition, keyboard navigation, and other assistive tools. Additionally, about 15–20% of people are neurodiverse, adding another layer of consideration in designing documentation that is both accessible and inclusive. As these statistics suggest, members of your team and community perceive, interpret, and engage with your documentation in varied ways. Addressing these differences might introduce distinct challenges to the developer documentation, but the benefits extend to everyone. Practical adaptations, such as including captions for videos, enhancing colour contrasts, and adopting a coherent and logical content structure, can optimise the user experience and reduce cognitive strain for everyone.

When addressing accessibility and inclusion, there are two main areas to concentrate on. First, focus on the technical aspects, such as design adjustments and compliance with accessibility standards, which you can verify using specific tools. Secondly, look at the content itself, ensuring it has a clear structure and is written in an inclusive and easily understandable manner.

Creating accessible documentation not only demonstrates empathy towards specific needs but also enhances the overall user experience for everyone.

Sustainability

The vastness of the Internet and its ever-expanding infrastructure makes it a significant consumer of energy. Every digital action, from streaming a video to browsing a website, leaves a carbon footprint through the servers processing the data, the devices displaying the information, and the energy used in producing and replacing the necessary hardware.

When building your developer documentation site, understanding the environmental implications of web development can steer you towards crafting a more eco-friendly digital environment. The Sustainable Web Manifesto outlines the path to a greener Internet, focusing on renewable energy, efficiency, and transparency, to name a few key principles. By embracing these guidelines during website creation, you not only adopt sustainable online practices but also convey your dedication to environmental responsibility.

Hosting, performance, image management, fonts, web caching, user experience, content management, and search engine optimisation are critical factors to consider when building sustainable websites. Each plays a significant role in optimising the efficiency and environmental impact of your online presence.

Learning more about sustainability best practices and adjusting your website accordingly can benefit both your users and the planet we all live on.

Conclusion

In this article, I aimed to emphasise the importance of adopting an empathetic approach to building developer documentation, with the hope of offering insights that you can incorporate into your own projects. If you'd like to explore this topic further, I highly recommend the presentation titled *Building Empathy-Driven Developer Documentation* by Kat King at Write the Docs Portland 2018. It offers an insightful, relatable, and often delightfully entertaining perspective. I delve deeper into these topics and more in my book, *Crafting Docs for Success: An End-to-End Approach to Developer Documentation*. ■

Diana Lakatos

Diana is an experienced Developer Documentation Specialist dedicated to creating high-quality resources for developers. She has been pivotal in the development of the multiple award-winning platformOS Developer Portal, and spoke about various aspects of building efficient developer docs at conferences like Write The Docs, tcworld, DevRelCon, and API The Docs.

Starting out in technical communication?

ISTC PROFESSIONAL DEVELOPMENT & RECOGNITION

Help your career blast off with the ISTC's mentoring scheme for Junior Members. Join the ISTC today.

Search 'Mentoring Scheme' on the ISTC website for details