Write the Docs 2017

Lessons and Recommendations

Kristen Martin
June 14, 2017

WTD Overview

- Community of software documentarians
- Local Meetup group
- Annual conferences in NA and Europe

www.writethedocs.org

Purposes of Documentation

For Everyone

- Minimize reliance on tribal knowledge
- Reduce re-work and repetitive conversations
- Illustrate patterns that can reveal deeper problems
- Bring “slow-boil” issues to attention and resolution

For Sysadmins

- Help with planning and incident response
- Create knowledge base for using managed services and vendor software/hardware
  - Ex.: Customizations of off-the-shelf products
- Track unexpected occurrences during routine tasks
  - Ex.: Problems when deploying from Stage to Production

For Support Agents

- Offer a toolkit for fixing common problems
- Address troubleshooting from different perspectives
  - Assuming docs are written with cross-functional support
- Replace boring conversations with interesting ones

Documentation Structure and Process

Doc Site Navigation

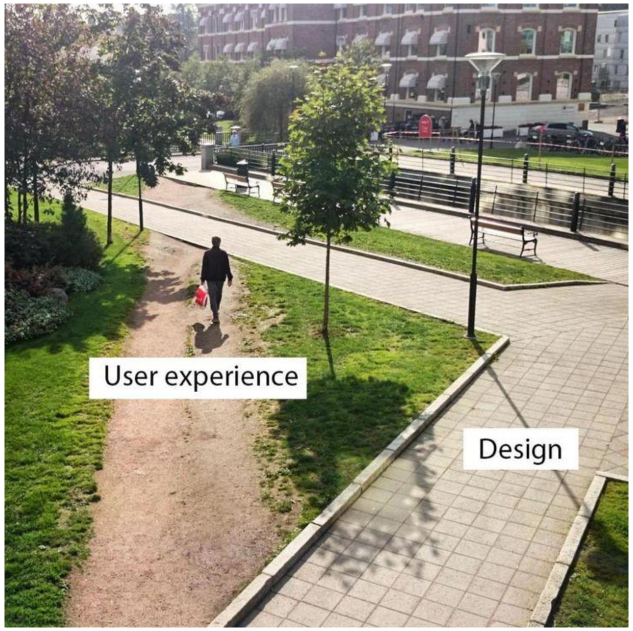
- Navigation and usability are just as important as content
- Principles of navigation:
  - Hierarchy
  - Progressive disclosure
  - Desire line
  - Modularity
  - Wayfinding

UX Principles in Doc Design

- *Personas*: representation of users/readers
  - Think about different scenarios that bring readers to your document
  - Consider emotional state of the reader who is having trouble
- Docs exist not just to tell users what buttons to click, but how to accomplish something
  - Use task-oriented structure and language

UX Principles in Doc Design

- Conduct usability testing of docs
- Track how docs are used



Source: Pinterest

Style Guides

- Offer authoritative answers to recurring questions about formatting, abbreviations, etc.
- Should be living documents
- Are a tool, not a policing mechanism

*Twitter: @_grammar_*

Document Critiques

- General process:
  - Identify stakeholders
  - Establish scope of critique
  - Provide guidelines for participants
  - Listen and repeat
  - Evaluate results
- Critique vs. criticism

Document Critiques

- Tips for giving feedback:
  - Deliver a compliment sandwich
  - Pose feedback as a question, e.g., *What if we moved element X to this side of the page?*
- Tips for receiving feedback:
  - Remember that everybody wants to make the product better
  - Evaluate feedback in the context of serving the project's goals

Document Versioning

- Docs can be treated like code
  - Draft/staging
  - Testing
  - Deployment
  - Automation via web authoring tools, Git integration

Document Versioning

- Docs can be considered part of the larger project and developed in tandem with software
  - Allows writing to occur when knowledge is fresh
- Ask at the outset: *How will docs be maintained and versioned?*

Document Content

Considering Your Audience

- Who will be using the document?
  - Customers/end users
  - Developers
  - Partner organizations
- Remember UX design principles

Collaborating on Content

- Everybody has knowledge to contribute
  - Technical people can write content, and writers can polish that content
- Offer templates when asking team members to contribute
- Understand how to motivate people to contribute
  - Most knowledgeable people tend to write least documentation

Testing Content

- Measures of quality documentation:
  - Completeness
  - Consistency
  - Correctness
  - Contributability
- Specific tests may include:
  - Required elements
  - Brand guidelines
  - Acceptable acronyms
  - Weak words, passive verbs

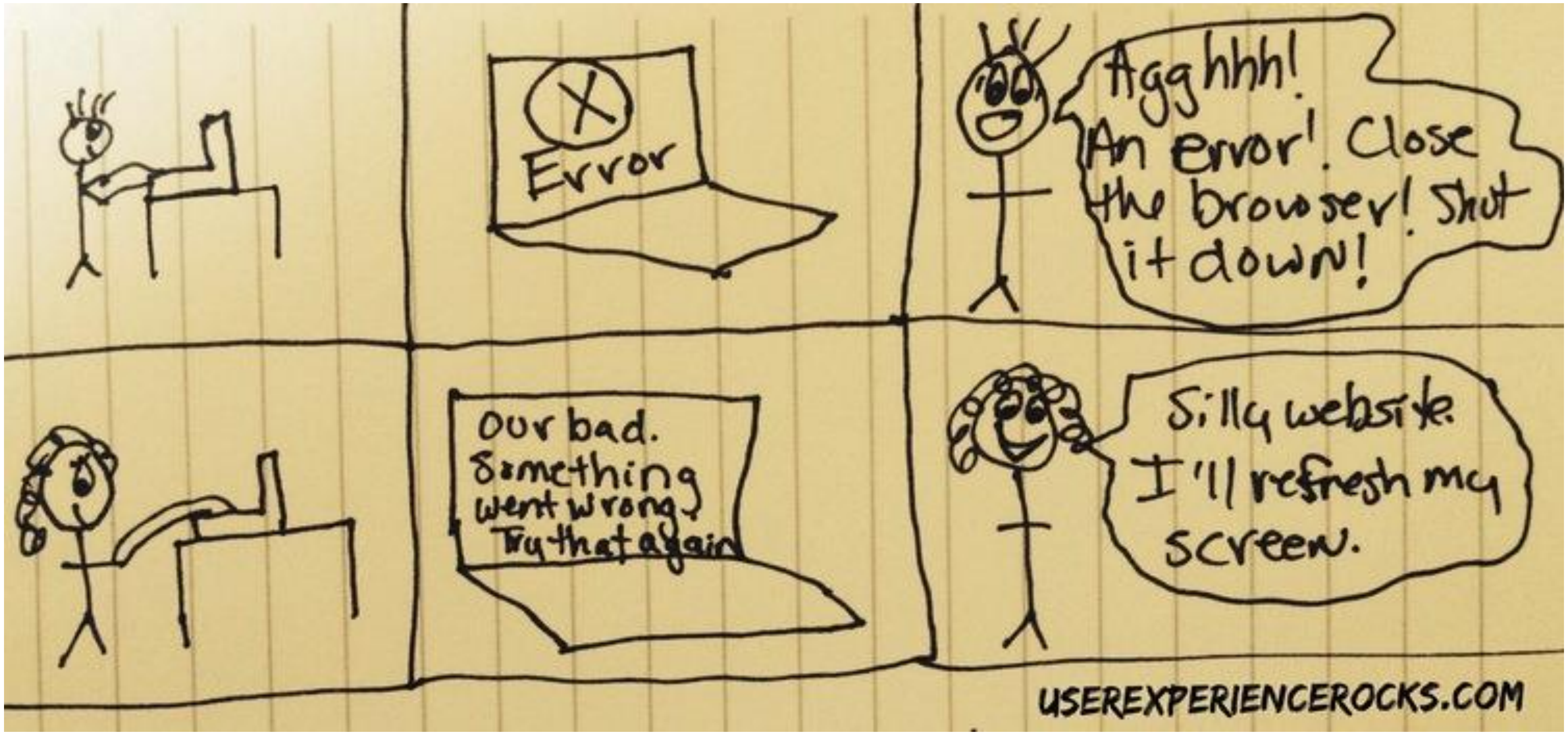
Using Minimalism

- Focus on readers' goals
- Eliminate vague and passive language
- Use simple words for ESL readers
- Make titles and headings clear and concise
- Avoid unnecessary contextual information
- Turn walls of text into lists
    - **Bullets** convey independent bits of information
    - **Numbers** describe a procedure

Writing Error Messages

- Be humble: apologize for unexpected behavior
- Be helpful: instruct user what to do next
  - Focus on object first, action second
- Be human: don’t talk like a robot
- Be careful with humor: don’t be dismissive of the issue

Writing Error Messages


Error
Agghhh! An error! Close the browser! Shut it down!
Our bad. Something went wrong. Try that again
Silly website. I'll refresh my screen.
USEREXPERIENCEROCKS.COM

Source: Jennifer Aldrich

Writing Error Messages

- Don’t rely on just one method to convey info
  - Icons/graphics
  - Text
  - Color

Naming Software Elements

- Applications, variables, functions, *et al.*
- Good names…
  - Contextualize
  - Illuminate
  - Empower
  - Explain
- Bad names…
  - Confuse
  - Obscure

Naming Software Elements

- Why is it hard to choose good names?
  - Names are reductive
  - Purpose of object can be unclear until it’s in use
  - Names evoke different things for different users
- Why do bad names persist?
  - People dislike change
  - Engineers overestimate effort/impact of change
  - Engineers don’t understand how other people may be confused

Making Docs Accessible

- Many Americans have disabilities
  - Learning (e.g., dyslexia)
  - Physical (e.g., visual impairment)
- Interactions with assistive tech devices
  - Screen readers
  - Specialized keyboards
- Using diverse methods to deliver a message
  - Image alt text
  - Contrasting colors
  - etc.

Making Docs Intelligent

- Principles based on replicability crisis in scientific literature
- Keep docs in sync with code base
- Ensure that instructions are reproducible
- Embed executable code in the document
  - Sample code visible to readers should be minimal
  - Full executable code stored on back end

Recommendations

Suggestions for a small department with informal documentation

1. Define and prioritize documentation needs for different user segments

- Internal users:
  - Developers
  - Systems administrators
  - QA testers
- External users:
  - Customers
  - Vendors

2. Create a Knowledge Base for each app that any team member can contribute to

- This could include best practices, configuration tips, and known bugs and their workarounds
- No matter which tool is used, it must have a structure and guidelines for contributing

3. Move user-facing documents to a centralized location where they can be updated each time a new iteration is released

- Revision author and timestamp would be on each page
- There would be a way to highlight which pages/articles have recently been changed

4. Develop a style guide for technical documentation, including:

- What names to use for different applications that may be referred to colloquially
- How to format references to UI elements, such as button labels and link text
- How to format references to databases, tables, and fields