SOFTWARE DEVELOPMENT IN SCIENCE
DOCUMENTATION

19/20 NOVEMBER 2019 I WOUTER KLIJN

OVERVIEW

- Homepage or Github.io landing page
- Tutorials
- How-to guides
- Technical reference and API documentation

HOMEPAGE OR GITHUB.IO LANDING PAGE

- First contact, should answer:
  - What is the software?
  - Is it important for me, should I care?
  - How much effort is it to learn?
- Further reading:
  - License
  - Bug tracker
  - Source code
  - Further documentation
- Download (button)!

http://stevelosh.com/blog/2013/09/teach-dont-tell/

TUTORIALS

- First entry point for new users
  - Check technical level with a real (new) user
  - You as a developer cannot assess the level
- **The very first tutorial**
  - No rocket science
  - Do one thing, from beginning to end and

Give the user a working example within 30 minutes

http://blog.parse.com/learn/engineering/designing-great-api-docs/
https://jacobian.org/writing/what-to-write/

HOW-TO GUIDE

- How to perform a specific action or function
- Do not double information
  - Refer back to tutorials and other how-to with hyperlinks
- In depth and comprehensive
- Not a dry summations of functions
  - It is not an API documentation

https://jacobian.org/writing/what-to-write/

TECHNICAL/REFERENCE/API DOCUMENTATION

- The details of your project
- List all your public functions and classes
  - Function_Name(Arguments)
  - Description
  - Source file, inheritance hierarchy, example
- Typically generated from source-code
- Expensive
  - E-mail to develops can be a alternative for some projects?

https://jacobian.org/writing/what-to-write/
http://en.cppreference.com/w/

- Other interesting sources of documentation:
  - Source code
  - Issue tracker
  - Internal chat server
  - Meeting notes
  - Email
  - Tests !!!
- Articles series on documentation:
  - https://jacobian.org/writing/great-documentation/

https://medium.com/@MrJamesFisher/documentation-black-holes-facd0c3b9ed7

Questions?