Clang-Repl and Xeus-Clang-Repl documentation

Project Statement

Project Title: Clang-Repl and Xeus-Clang-Repl documentation

Detailed Description:

General

- I’ll be looking at these technologies from a beginner’s perspective, without prior experience with compilers. My technical background will help emulate your typical user who is interested in using or developing these technologies.

Overall Approach

Note: Following is a sample of document auditing approach, a lot of the references to the Clang documentation below are out of scope for this project.

- Orientation session to introduce the technologies (e.g. ROOT, Cling, etc.) that the writer should become familiar with, including internal and external learning resources.
- Sprint/Feature Demos to introduce new features to a technical audience and capture their responses (and any subsequent changes in feature design as a result).
- Conduct live walkthroughs by non-Developer resources (e.g., QA). This will help the writer:
  - Read between the instruction lines, grasp missing context
  - Capture any challenges faced by the non-dev user in troubleshooting documentation
- Video Content: Byte-sized (less than 5 minutes) screen-grabs of activities (like installation, configuration, etc.) can be recorded, edited, hosted on YouTube and referenced in documentation (to replace the long, outdated demos).
- Tools like ReadTheDocs may be evaluated for improving documentation.
- Knowledge gaps and outdated feature descriptions may be identified after training completion.

Specific Document Audit Examples

- Main page could use some organization. For example, grouping similar topics (e.g., sanitizers), adding brief intros and displaying them in card format (like modern websites).
- Several documents presume that the reader has prior knowledge of technologies (e.g., Clad) and other technical jargon. This reduces readability, making the content seem denser than it is.
- Relevant terminology may need to be introduced at the beginning of an article.
- Inter-linking articles or context-sensitive help (hover to show tooltip) may also help.
- Assumptions/ Pre-requisites should be highlighted in the beginning (like here: It assumes you already know how to compile the code in question...).
- Topic names (example) don’t always explain what the document teaches.
• **Topic not defined** in introduction (e.g., what is Language Extension needs to be established for a new user, before jumping into feature checks).

• Text in the Table of Contents doesn’t always match the Page Headings (the intent may have been to add more context, but these links look like external links, adding to the confusion).

• Many articles jump right into the action items. **Intros** may be improved by providing context on what an article will help the user learn. **Outros** may help summarize the learnings.

• Need to identify missed opportunities to add context or introduce **Terminology**.

• Need to identify **incomplete sections** (example).

• Need to identify **tasks mixed with documentation** content (e.g., 4 instances of TODO).