DOCUMENT REVIEW PROCESS

YVONNE V. RICHARDSON

AZURE INFRASTRUCTURE TEAM

DOCUMENT REVIEW PROCESS

- Update and review existing topics
- Determine what needs to be deleted from the library and/or from the code
- Determine what needs to be added to the library
- Separate new code development from library additions

THE AZURE PORTAL DOCUMENTATION LIBRARY

- Library is at https://github.com/Azure/portaldocs/blob/master/portal-sdk/generated/portalfx-main.md
- Documents are pulled from https://github.com/Azure/portaldocs/blob/dev/portal-sdk/generated/portalfx-main.md
- Documentarian library is https://github.com/Azure/portaldocs/blob/v-yvric/dev/portal-sdk/generated/portalfx-main.md
  - Read-only copy of the master library

4 DOCUMENT UPDATE: SELECTION

- Priority is driven by the oneNote master index
- Other current library documents are self-prioritizing
- New documents will be added to one of the above lists

5 DOCUMENT UPDATE: EXISTING TOPICS

- Find the "main" GitHub document for the topic and update it
- Use the document skeleton that is in GitHub
  - portalfx-skeleton.md
- Guidelines for document updates exist somewhere
  - Complete sentences, active present tense, and so on
- Search GitHub for relevant documents
  - Marked for deletion (archival)
  - Links to and from the new documents

DOCUMENT UPDATE: LINKED TOPICS

- Not necessarily in the main index
- Whatever is linked to should be updated using these processes
- Documents that are linked to will not delay documents that are currently in work
  - They have their own Code Review
  - They are prioritized separately
  - Retaining the current link does no harm
  - Example: portalfx-extensions-samples.md is in work, and it links to portalfx-extension-versioning.md (link is in portalfx-extensions-samples-overview.md)

7 DOCUMENT UPDATE: DRAFT COMPLETION

- Move draft to Documentarian library (Read-only copy, located at https://github.com/Azure/portaldocs/blob/v-yvric/dev/portal-sdk/generated/portalfx-main.md)
- Copy draft to appropriate page in OneNote (see master index)
- Developers can review both copies, and update OneNote copy
  - At the Tuesday meetings on request, or at scheduled working meetings
  - At any time

8 DOCUMENT UPDATE: ONENOTE REVIEW

- Is the content accurate?
  - Example: Document discusses V1 code that still works as described
  - Example: Document discusses code that is about to be removed, but is not yet gone
  - Example: Update document by removing descriptions of V1 code that no longer exists
- Is the content complete?
  - Example: Document does not discuss V2 code, and should include it previous to publication
  - Example: Document does not discuss V3 code, and does not need to in this iteration

DOCUMENT UPDATE: LIBRARY ADDITIONS

- Search library to verify whether topic exists and was missed during rewrite or linking
- Add new topic to library that is related to a document in review
- Worthy of being its own topic
- Do not delay document in review
- Do not delay review process
  - Make a note and prioritize the new document for its own review and a later release

DOCUMENT UPDATE: NEW CODE DEVELOPMENT

- SDK, other releases should not delay initial document release
- Mark the document as "can release to master" and put the SDK changes in the next iteration in the dev environment
- "What's new" document will discuss changes that cannot go to master environment now

DOCUMENT UPDATE: DOC CODE REVIEW

- CodeFlow Process adds its own constraints
- A little more formal than oneNote working meetings and review
- Copy OneNote updates to Documentarian library
  - https://github.com/Azure/portaldocs/blob/v-yvric/dev/portal-sdk/generated/portalfx-main.md
- Use library copy for CodeFlow

DOCUMENT UPDATE: CODEFLOW REVIEW

- Document review is less formal than a program code review
- Invite reviewers as appropriate
- Minimum 2 signoffs in CodeFlow
- Some documents may require more than 2 reviewers
- CodeFlow updates copied to Documentarian library

DOCUMENT UPDATE: PUSH TO MASTER

- When document receives the appropriate number of signoffs, it is ready to be pushed to the dev environment
- Documents copied from dev to master on a regular basis
- Move up documents separately from, and previous to, SDK and similar releases
- Announcements to appropriate communities that the new doc is available in the master branch
- After some amount of time, deprecate the documents that are replaced by the new documents

DOCUMENT UPDATE: ARCHIVAL AND DEPRECATION

- Old documents copied to a sub-directory, renamed, or otherwise archived
- Old document content replaced with a link to the new document
- Copied from documentarian library to dev to master on a schedule