Microspeak: Inside baseball

devblogs.microsoft.com/oldnewthing/20220628-00

June 28, 2022

Raymond Chen

When we review proposed documentation, a term that you may see in the feedback is *inside baseball*, with the accent on “inside”. (Remember, Microspeak is not just terms used exclusively within Microsoft, but also terms used at Microsoft more often than in the general population.)

> These details about why the Widget cannot be shared between processes sounds like inside baseball. All that the developer needs to know is that cross-process sharing is not supported. They don’t need the gory implementations details that show why it doesn’t work.

> The instructions on how to set things up are in our public documentation, but here’s some inside baseball for the curious.

In a sense, the people who know the most about a feature are the least qualified to write documentation about it because they have been immersed in the topic for so long that they may not realize that they are using terms and phrases in the documentation that presuppose some level of deep internal knowledge of the feature. It’s also often the case that the team internally is a little sloppy with terminology because they understand what the speaker really means and can resolve the ambiguity on their own. However, the people reading the documentation don’t know about these internal shorthands or have the background information necessary to be able to disambiguate the terms automatically.

For example, a team might casually say that “The widget sends a message to another widget”, when really what happens is that the widget *controller* sends the message to the other widget’s controller. To people who work with widgets and widget controllers all day, this is just a handy abbreviation, but for people who are just learning about widgets and widget controllers, this shorthand just adds to the confusion.

Casual synonyms are another source of inside baseball. For example, the documentation might describe an operation as being “retired”, even though the name in the API is “Completed”. This creates confusion to the new reader over whether “retired” is some new stage of the operation separate from “completed”, and it may result in some flipping backward through the pages searching for a definition of “retired”.

Sometimes these inadvertent usages of internal team jargon are relatively easy to decode, but even so, they add another obstacle to people trying to learn about it.

It can be difficult to take off your feature-colored glasses and look at the topic fresh, with the eyes of someone who is just starting to learn about the subject. You may want to find someone from outside your team to read over your documentation and point out places where it is unclear to outsiders.[1]

Another category of inside baseball is the documentation that talks too much about implementation details as a way to justify why something is done. This level of detail can end up causing trouble in the future when the implementation changes. For example, the documentation for `WaitForMultipleObjects` provides an alternative to waiting for more than `MAXIMUM_WAIT_OBJECTS` objects, namely to register a thread pool wait for all of the objects you are interested in. The documentation said, “A wait thread from the thread pool waits on `MAXIMUM_WAIT_OBJECTS` registered objects,” talking about the implementation detail that the thread pool groups all the registered waits into blocks of `MAXIMUM_WAIT_OBJECTS` − 1 handles. But we saw that the implementation detail changed in Windows 8, and now a single thread services all the registered waits. The old documentation provided too much detail about the thread pool’s internal implementation, and became incorrect once that implementation changed. All it really needs to say is “The thread pool waits efficiently on all of the handles,” and the implementation is free to carry out that efficient waiting by whatever means it sees fit.

[1] That’s not to say that your documentation has to be simplified to the point where it can be understood by someone who is brand new to computer programming. You can still assume some level of competency from the reader, but the competency the reader brings to your document is not competency about your specific feature. After all, they’re reading your documentation because they aren’t experts in the subject matter!

Raymond Chen

Follow