Microspeak: Party, in various forms

devblogs.microsoft.com/oldnewthing/20140211-00

February 11, 2014

Raymond Chen

Remember, Microspeak includes words and phrases in general use, as long as they are employed at Microsoft at a higher rate than in the general population, or in specific situations that may not be obvious to the uninitiated. They are the words and phrases you need to use in order to fit in.

Today's word is *party*, in various forms, and usually paired with the preposition *on*. In general, it means *to use, change, or modify with few or no constraints*. These aren't genteel tea parties; they're more like wild college parties, the kind that end with the police being called.

> LockBits returns a pointer to the pixel buffer, and the caller can party on the memory inside the rectangle until it calls UnlockBits.

When used with permission verbs like *can* and *may*, the usage indicates that the component has permission to read from and write to the memory, subject to the given constraints.

> The code partied on our data structures because it used a pointer after freeing it.

It is often used in a negative sense to indicate that the component wrote to memory that it should not have. Sort of an unauthorized party. (Compare *fandango on core*.)

> The Contoso notifier injects a DLL into Explorer so it can party on the internal data that keeps track of icons in the notification area and thereby disable the icons of its competitors.

These sorts of unauthorized parties can be malicious and willful as well as merely accidental.

> The *exp* branch is a party branch. You can commit your changes there so we can test it before pulling it into the release branch.

The word *party* can be used to describe an environment in which the normal rules and constraints are reduced or removed entirely. Here, the *party branch* is presumably a branch of the project in which the usual procedures for code changes don't apply, or at least apply less strictly than normal. You can put any experimental changes in the *exp* branch, and then

when a new build comes out the next day, you can run your tests against it to see if they solve the problem. If so, then you can start filling out the necessary paperwork to pull the changes into the release branch.

Many release branches have an experimental offshoot.[1] The idea is that people developing fixes to the product can commit their changes to the experimental branch to see how they work out. If the changes look good, they are pulled into the release branch. If the changes doesn't pass muster, they are rolled back. The developers who use the experimental branch are on their honor to keep the branch in good condition.

Note that this sense of *party* is relative. The experimental branch is a big party compared to the staid and formal release branch, but it's still not a crazy free-for-all. You still need to be judicious about what you put into the party branch so you don't ruin the party for everybody else.

> The Q1 branch is locked down for the beta, but you can party your post-beta fixes into the Q2 branch.

The above example further highlights the relative nature of the term *party*. Even though the Q2 branch is open to post-beta fixes, you still have to go through the usual test and review processes for fixing bugs. It's just that Q2 will accept any approved bug fix, whereas Q1 will accept only fixes for bugs marked beta-blocking.

(That's a little extra Microspeak for you: *blocking*. In Microspeak, a beta blocker is not a pharmacological agent. Rather, it's something that prevents the beta from being released.)

[1] In Windows, the experimental branch associated with a release branch is typically called *cbt*. This officially stands for *Central Build Team*, but some people who live in my house like to joke that it stands for *Can't Be Trusted*.

Raymond Chen

Follow