API design note: Beware of adding an “Other” enum value

devblogs.microsoft.com/oldnewthing/20250217-00

February 17, 2025

Consider the following API:

```
enum class WidgetFlavor
{
    Vanilla,
    Chocolate,
    Strawberry,
    Other,
};

WidgetFlavor GetWidgetFlavor(HWIDGET widget);
```

The idea here is that Widgets come in three flavors today, but we might add new flavors in the future, so we add an `Other` value to cover any future flavors.

This is a problem.

Suppose we do indeed add a new flavor `Mint` in the next version.

```
enum class WidgetFlavor
{
    Vanilla,
    Chocolate,
    Strawberry,
    Other,
    Mint,
};
```

What flavor should you report if somebody calls `GetWidgetFlavor` and the widget is mint?

If you return `WidgetFlavor::Mint`, then this will confuse code written with the Version 1 API, because they expected to get `Other` for anything that isn't vanilla, chocolate, or strawberry. The word "other" means "not mentioned elsewhere", so the presence of an `Other` logically implies that the enumeration is exhaustive.

On the other hand, you obviously should return `WidgetFlavor::Mint` because that's why you added the value to the enum in the first place!

My recommendation is not to have an `Other` at all. Just document that the enumeration is open-ended, and programs should treat any unrecognized values as if they were "Other". Any code that uses this enumeration will therefore put all unrecognized values into an app-defined "Other" category on their own. Now you can add `Mint`: New code will put minty widgets in a "Mint" category, and old code will continue to put them in the app=defined "Other" category.