How can I detect programmatically whether Windows is an N or KN version?

devblogs.microsoft.com/oldnewthing/20221229-00

December 29, 2022

Raymond Chen

A customer was dealing with a problem that occurred only when running on an N or KN version of Windows. These versions of Windows are special versions that omit the multimedia features.[1] Microsoft is legally required to offer these versions of Windows in certain jurisdictions, although in practice, the number of people who buy them is vanishingly small. (It sometimes feels like the sole reason those customers exist is to file bugs that reproduce only on N and KN versions.)

The customer wanted to know how to detect that the user is running an N or KN version of Windows, so they could disable the features of the product that depend upon multimedia support.

The answer is that you don't check whether you are on an N or KN version of Windows. Rather, you check whether multimedia support is present.

Because the user, after buying an N or KN version of Windows (perhaps inadvertently), can later download and install the Media Feature Pack and restore multimedia support. In that case, they expect your program to enable its multimedia features.

So really, what you want to do is detect whether multimedia support is present. One way to do that is to see if you can call the `MFStartup` function, and whether it succeeds. If not, then Media Foundation is not available, and multimedia features are not available.

This particular customer had a Web-based app, in which case they can use `HTMLMediaElement.canPlayType` to detect whether the system can play their media, and skip the video if so.

Bonus chatter: The bug was that if you asked to see the training video, the video didn't play (expected), but the app also hung (not expected). The reason is that the app tried to play the video roughly like this:

```
var video = document.queryselector("#training-video");
video.src = "/videos/training.mp4";
video.addEventListener("error", onVideoFinished);
video.addEventListener("ended", onVideoFinished);
```

If multimedia support is not present, the `error` event is raised immediate upon setting the `src` property. But the code hasn't registered a handler for that event, so the `error` event is raised, but nobody is there to handle it. The app later adds a handler, but it's too late. That's why the app appeared to hang.

To deal with errors that occur immediately, the app should register the event handlers *before* setting the source.

(Note that `canPlayType` is still useful, even after they fix this race condition. That lets them detect that the system cannot play the training video at all, and they can remove the "Play training video" option from their interface, or replace it with an explanation of why the training video is not available.)

[1] The KN version also omits Windows Messenger, but since Windows Messenger itself has been discontinued, the distinction is meaningless in practice.

Raymond Chen

Follow