Who is responsible for destroying the font passed in the WM_SETFONT message?

devblogs.microsoft.com/oldnewthing/20080912-00

September 12, 2008

Raymond Chen

The `WM_SETFONT` message tells a control which font you would like the control to use for its text. This message is a convention, not a rule. For example, our scratch program doesn't pay attention to the `WM_SETFONT` message. If somebody sends it a `WM_SETFONT`, nothing happens. The scratch program just ignores the caller and uses whatever font it wants.

Although supporting the `WM_SETFONT` message is optional, if you do choose to support it, you would be well-served to adhere to the following convention: The `WM_SETFONT` message does not alter font ownership. In other words, whoever was responsible for destroying the font before the `WM_SETFONT` message is sent is still responsible for destroying it after the message is sent.

Corollary: If you tell somebody to use a specific font, don't destroy the font while they're still using it, because they're counting on you to keep the font valid.

Example: The dialog manager creates the dialog box font and sends the `WM_SETFONT` message to each control to tell it what font it should use. The dialog manager keeps the font valid until the dialog box is destroyed, at which point the font is destroyed as well.

But what font does a control use before it receives a `WM_SETFONT` message? Whatever font it wants. Some controls have particular ideas about the font they will use by default; the list view, for example, uses the icon label font. In those cases, it is the control's responsibility to destroy that default font when the control is destroyed. This is true even if the parent window creates another font and sends a `WM_SETFONT` to change to that font. There are now two fonts involved: the original default font (which is the control's responsibility to destroy) and the font set by `WM_SETFONT` (which is the parent's responsibility to destroy).

Now this may all sound complicated, but remember the basic rule: The `WM_SETFONT` message does not change who is responsible for destroying a font. Let's look at that scenario again, but take out the `WM_SETFONT` message. (I've crossed it out below.) A control creates a default font. The parent window creates another font. ~~The parent window sends the `WM_SETFONT` message to the control.~~

Now, without that crossed-out sentence, the ownership rules are crystal clear. The control is responsible for destroying the default font it created, and the parent is responsible for destroying its own font.

The same principle applies to the `WM_GETFONT` message. You can send `WM_SETFONT` and `WM_GETFONT` messages all day long; it has no effect on who is responsible for destroying the font at the end of the day.

Raymond Chen

Follow