Taskbar Customization

Overview

Electron has APIs to configure the app's icon in the Windows taskbar. This API supports both Windows-only features like creation of a `JumpList`, custom thumbnails and toolbars, icon overlays, and the so-called "Flash Frame" effect, and cross-platform features like recent documents and application progress.

JumpList

Windows allows apps to define a custom context menu that shows up when users right-click the app's icon in the taskbar. That context menu is called `JumpList`. You specify custom actions in the `Tasks` category of JumpList, as quoted from MSDN:

> *Applications define tasks based on both the program's features and the key things a user is expected to do with them. Tasks should be context-free, in that the application does not need to be running for them to work. They should also be the statistically most common actions that a normal user would perform in an application, such as compose an email message or open the calendar in a mail program, create a new document in a word processor, launch an application in a certain mode, or launch one of its subcommands. An application should not clutter the menu with advanced features that standard users won't need or one-time actions such as registration. Do not use tasks for promotional items such as upgrades or special offers.*
>
> *It is strongly recommended that the task list be static. It should remain the same regardless of the state or status of the application. While it is possible to vary the list dynamically, you should consider that this could confuse the user who does not expect that portion of the destination list to change.*

> *NOTE: The screenshot above is an example of general tasks of Internet Explorer*

Unlike the dock menu in macOS which is a real menu, user tasks in Windows work like application shortcuts. For example, when a user clicks a task, the program will be executed with specified arguments.

To set user tasks for your application, you can use app.setUserTasks API.

Examples

Set user tasks

Starting with a working application from the Quick Start Guide, update the `main.js` file with the following lines:

```
const { app } = require('electron')

app.setUserTasks([
  {
    program: process.execPath,
    arguments: '--new-window',
    iconPath: process.execPath,
    iconIndex: 0,
    title: 'New Window',
    description: 'Create a new window'
```

```
  }
])
```

Clear tasks list

To clear your tasks list, you need to call `app.setUserTasks` with an empty array in the `main.js` file.

```
const { app } = require('electron')

app.setUserTasks([])
```

> *NOTE: The user tasks will still be displayed even after closing your application, so the icon and program path specified for a task should exist until your application is uninstalled.*

Thumbnail Toolbars

On Windows, you can add a thumbnail toolbar with specified buttons to a taskbar layout of an application window. It provides users with a way to access a particular window's command without restoring or activating the window.

As quoted from MSDN:

> *This toolbar is the familiar standard toolbar common control. It has a maximum of seven buttons. Each button's ID, image, tooltip, and state are defined in a structure, which is then passed to the taskbar. The application can show, enable, disable, or hide buttons from the thumbnail toolbar as required by its current state.*
>
> *For example, Windows Media Player might offer standard media transport controls such as play, pause, mute, and stop.*


player

> *NOTE: The screenshot above is an example of thumbnail toolbar of Windows Media Player*

To set thumbnail toolbar in your application, you need to use BrowserWindow.setThumbarButtons

Examples

Set thumbnail toolbar

Starting with a working application from the Quick Start Guide, update the `main.js` file with the following lines:

```
const { BrowserWindow } = require('electron')
const path = require('path')

const win = new BrowserWindow()

win.setThumbarButtons([
  {
    tooltip: 'button1',
    icon: path.join(__dirname, 'button1.png'),
    click () { console.log('button1 clicked') }
  }, {
    tooltip: 'button2',
    icon: path.join(__dirname, 'button2.png'),
    flags: ['enabled', 'dismissonclick'],
```

```
    click () { console.log('button2 clicked.') }
  }
])
```

Clear thumbnail toolbar

To clear thumbnail toolbar buttons, you need to call `BrowserWindow.setThumbarButtons` with an empty array in the `main.js` file.

```
const { BrowserWindow } = require('electron')

const win = new BrowserWindow()
win.setThumbarButtons([])
```

Icon Overlays in Taskbar

On Windows, a taskbar button can use a small overlay to display application status.

As quoted from MSDN:

> *Icon overlays serve as a contextual notification of status, and are intended to negate the need for a separate notification area status icon to communicate that information to the user. For instance, the new mail status in Microsoft Outlook, currently shown in the notification area, can now be indicated through an overlay on the taskbar button. Again, you must decide during your development cycle which method is best for your application. Overlay icons are intended to supply important, long-standing status or notifications such as network status, messenger status, or new mail. The user should not be presented with constantly changing overlays or animations.*

 Overlay on taskbar button

> *NOTE: The screenshot above is an example of overlay on a taskbar button*

To set the overlay icon for a window, you need to use the BrowserWindow.setOverlayIcon API.

Example

Starting with a working application from the Quick Start Guide, update the `main.js` file with the following lines:

```
const { BrowserWindow } = require('electron')

const win = new BrowserWindow()

win.setOverlayIcon('path/to/overlay.png', 'Description for overlay')
```

Flash Frame

On Windows, you can highlight the taskbar button to get the user's attention. This is similar to bouncing the dock icon in macOS.

As quoted from MSDN:

> *Typically, a window is flashed to inform the user that the window requires attention but that it does not currently have the keyboard focus.*

To flash the BrowserWindow taskbar button, you need to use the BrowserWindow.flashFrame API.

Example

Starting with a working application from the Quick Start Guide, update the `main.js` file with the following lines:

```
const { BrowserWindow } = require('electron')

const win = new BrowserWindow()

win.once('focus', () => win.flashFrame(false))
win.flashFrame(true)
```

> *NOTE: Don't forget to call `win.flashFrame(false)` to turn off the flash. In the above example, it is called when the window comes into focus, but you might use a timeout or some other event to disable it.*