Placing an element behind another in the z-order (front to back order) to hide it is not recommended, as the element may still appear in the tab order. Thus, developers should be sure to set the visible property on items that are not `visible` to `false`, move elements that are off-screen completely off-screen, and ensure that `tabEnabled` is set to `false` when an element is visible but should not appear in the tab order.

Shortcuts and mnemonics

Frequently used functions of an application should be accessible via keyboard shortcuts, so that they can be accessed as quickly with the keyboard as with a mouse. Keyboard shortcuts typically use an easily remembered letter (or *mnemonic*) from the name of the component or action. The letter is underlined in that name to help users identify and remember it. For example, Ctrl+S is a standard mnemonic for a Save button or Save menu item.

There are no specific guidelines that explain when a shortcut is required, but it is a good practice to add a shortcut for any commonly used component that:

- Requires pressing the Tab key more than five times to reach
- Controls media playback
- Is needed quickly, such as to silence audio or stop repeated content updates

It is important to indicate shortcuts on the screen. Though it is possible to indicate them via the `accessibilityShortcut` property, screen readers do not always make use of this property and the information provided there will be unavailable to keyboard-only users who are not using a screen reader. In addition to underlining shortcut letters in menu items and button names, it is often helpful to make a list of shortcuts available on the screen or in a keyboard-accessible pop-up window. A list of shortcut keystrokes can also be provided in user guides and online help documentation.

Keystrokes to silence audio

It is very difficult for users of a screen reader to listen to an audio track and the screen reader at the same time. Audio that plays automatically when an application is loaded is especially disruptive. Thus, developers must ensure that users of assistive technology can stop the playback of any audio content (including narration, multimedia, or background music) that lasts longer than five seconds.

In the following example, Ctrl+S is used to pause the audio content:

```
addEventListener(KeyEvent.KEY_DOWN, pauseVideo);
function pauseVideo(e: KeyEvent): void
{
  // if ctrl+s is pressed pause the multimedia presentation
  if (e.ctrlKey && e.keyCode == 83)
    myVideoPlayer.pause();
}
```

Implementing keyboard shortcuts

Event listeners for keyboard shortcuts should be placed on the most general component in the application or on the application itself. If the event listener is attached to a specific component, the shortcut keys it handles will only be active when the focus is on that component.

As an example, consider a developer who wants to provide quick access to the help screen for an application by using the question mark (?) key as a shortcut. The developer would add a listener for the `KEY_DOWN` event on the main application object as shown in the following example:

```
// assign the event listener directly on the main application object
// For applications that accept user input (such as into form fields) the
 function below should be designed to verify that the user is not current
 in a form field.  Otherwise a keystroke such as F1 or a keystroke
 combination such as Control+Shift+? could be used rather than the
 single key question mark (?).
```