```
addEventListener(KeyEvent.KEY _ DOWN, performShortcut);

function performShortcut(e:KeyEvent): void
{
  if (e.keyCode == 63)  // the question mark (?)
  {
    // call same code that button's click handler calls
  }
}
```

Shortcuts to avoid

When choosing shortcut keystrokes for a Flex application, try to avoid the main shortcuts used in browsers, operating systems, and assistive technologies. The following shortcuts may cause difficulty if used in a Flex application:

- Windows: F1 (help), Alt+F4 (close the application), Ctrl+F4 (close the current window), Ctrl+W (close the current window), Ctrl+Tab (switch tabs)
- Internet Explorer and Firefox: F11 (full screen), Alt+D (place the focus in the address bar), Ctrl+P (print), Ctrl+N (new window), Ctrl+T (new tab), Ctrl+F (find), Ctrl+D (Bookmark), Ctrl+H (History)
- JAWS and Window-Eyes: Numpad Plus (toggle forms mode), Ctrl+Shift+A (toggle MSAA mode), Ctrl+Home (move to top of document), Ctrl+End (move to bottom of document), most single letters, Shift+most single letters, Insert+Down Arrow (JAWS, say all), Control+Shift+R (Window-Eyes, read to bottom of document))

While some conflicts are unavoidable, choose shortcuts carefully to avoid those most commonly used.

Note: Screen readers trap most single letter keystrokes in virtual cursor or browse mode. When this mode is activated, Flex applications will not receive keystrokes until the user activates forms mode or MSAA mode, or passes the keystroke through to the application by pressing another keystroke combination.

Reading order

Providing textual equivalents, enabling, keyboard access, and setting accessibility properties is not, by itself, enough to make an application accessible to users of all assistive technology. Developers must also take steps to ensure that content is exposed to assistive technology in a sequence that is meaningful. This sequence is known as reading order and describes the order in which a screen reader reads the content of the screen. The default reading order of any SWF file, whether created in Adobe Flash or with Flex, does not always follow a predictable left-to-right, top-to-bottom order. When the content itself does not explicitly specify a reading order, Flash Player determines the default order based on a formula that uses the x and y coordinates of a component's bound rectangles. For simple Flex applications, reading order may not be an issue; however, for most Flex applications, the reading order should be explicitly specified. If it is not, the content may be difficult or impossible to understand with a screen reader.

The example code below defines a tree component, a data grid, and a text area. In this example, the default reading order flows from the tree control to the data grid and then to the text area (see Figure 2). This is the order in which the items will be encountered by screen reader users, and it is the desired order for the application. Developers can confirm this default order by reviewing the content with a screen reader or testing it in AccProbe. In this case, the screen reader would announce the browser page title, followed by the tree component, the data grid, and the text area. Since this is roughly how a user would follow the application visually, the reading order does not need to be set explicitly.

```
<?xml version="1.0" encoding="utf-8"?>
<s:Application xmlns:fx="http://ns.adobe.com/mxml/2009"
  xmlns:s="library://ns.adobe.com/flex/spark"
  xmlns:mx="library://ns.adobe.com/flex/mx" minWidth="955"
  minHeight="600">
  <mx:Form>
    <mx:Tree accessibilityName="Lunch Items:" width="300" height="150"
```