If accessibility cannot easily be disabled by grouping, developers can place all background components in an array and cycle through the array to disable accessibility for each component individually.

Note: In addition to disabling accessibility for hidden elements, keyboard access should also be restricted to current, active elements. See "Keyboard accessibility" on page 15 for information on how to remove keyboard access from hidden content.

Exclude hidden or inactive elements

Elements that are hidden behind other elements, provided only for decoration, or located off-screen should be excluded from the reading order while they are invisible or inactive. Any active elements that can still be seen by sighted users should remain in the reading order, even if they are read-only (not editable) or disabled. Hidden elements include rollover content that has not been activated and elements that are hidden behind modal pop-up windows. In addition, inactive decorative elements that do not convey information should not be included in the reading order.

Setting the `accessibilityEnabled` property to `false` on a form, grouping, or container component will disable accessibility for all child components; for example:

```
// turn accessibility support off for this form when it is hidden behind a modal pop-up
frmSignInForm.accessibilityEnabled = false;
```

When the property is set on an individual component, the setting affects the accessibility of just that particular component. When accessibility is disabled for a component, it will not be exposed to assistive technology via MSAA.

Provide controls for disruptive elements

Screen readers may interpret any automatically updating content on the screen as a change that must be reported to the user. This can cause the screen reader to start over at the top of the page every time there is a change on the screen. Examples of elements that can be disruptive to screen reader users when accessibility is not implemented correctly include:

- Animations
- Weather and progress updates
- Clocks
- Scoreboards
- Timers

It is not a good practice to remove these components from the reading order to prevent screen readers from announcing the information repeatedly. This approach makes any dynamic content inaccessible to screen reader users, and does nothing for other users who may simply want to stop, pause, or hide the dynamic content. See "Controlling dynamic and automatically updating content" on page 14 for guidelines on handling such content.

Avoid redundant elements

When screen reader software encounters a Flex component, such as a form field, it announces the component's accessible name. Screen reader users do not require visual labels for form fields that have an accessible name property, but other users do need the visual label. A problem arises for screen reader users when both the visual label and the accessible name are exposed to assistive technology. In these cases, the screen reader reads both. If the label and accessible name are identical, the user hears everything twice; if they are not identical, the results can be even more confusing. In Flex, this is likely to happen with TextInput, ComboBox, List, DataGrid, and TextArea components. To avoid this redundancy, it is best to hide the visual label for such components from the screen reader by setting the `accessibilityEnabled` property to `false`; for example:

```
lblFirstName.accessibilityEnabled = false;
```

Silence form items and form headings

In Flex 4, FormItem and FormHeading components expose accessibility properties; prior to Flex 4 they were not exposed by default. Because of this change, label content present in the FormItem containers and