React UI Accessibility on TV

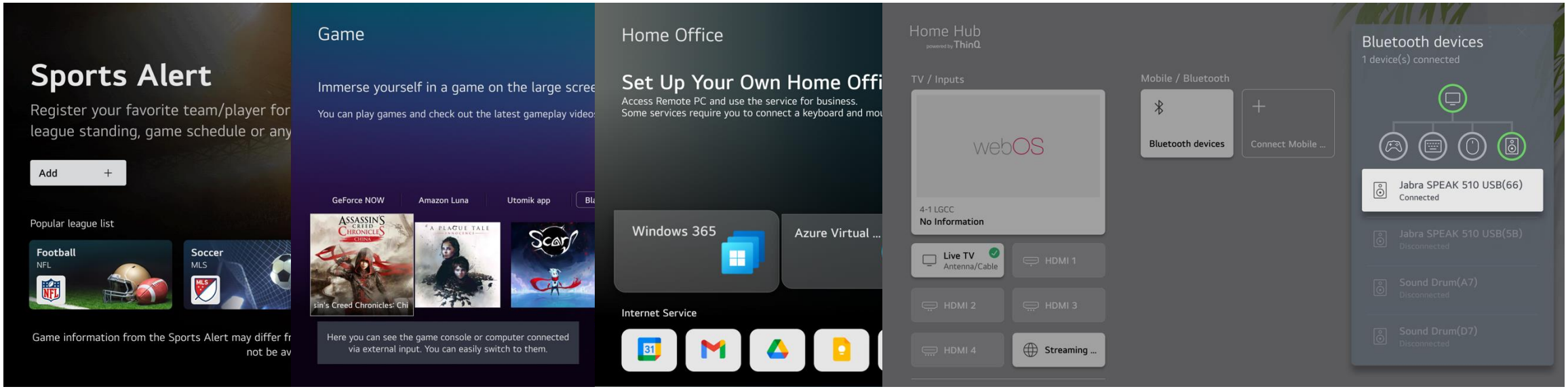
Practical Case Study in Real Production

Seungho Park(@hohpark, https://enactjs.com)

LG Electronics

Enact

- Enact is a React-based app framework that supports TV UI components, spatial navigation, a11y, i18n, and webOS APIs. It is shipped on tens of millions of LG webOS TVs every year



Agenda

- Key Navigation
- ARIA attributes & roles
- Use Cases – Practical examples on TV
  - Read Icon
  - Order matters: Title + Subtitle + Item
  - Range Widget: Title + Hint + Value
  - Alert Popup: Read without focus
  - Tab Menu: N of Total
  - Virtual List: N of Total
  - Announce any time you need
- Tools
- References

What is Accessibility

- The Web is fundamentally designed to work for all people, whatever their hardware, software, language, location, or ability. When the Web meets this goal, it is accessible to people with a diverse range of hearing, movement, sight, and cognitive ability. W3C – Accessibility[1)]

- Laws & Regulations[2)]
  - ADA (Americans with Disabilities Act)
  - EAA (European Accessibility Act)
  - 2024 updates – explicit ruling: Websites and mobile apps must be accessible

1) https://www.w3.org/standards/webdesign/accessibility
2) https://gitnation.com/contents/accessibility-in-2024

A11y Minimum Requirement

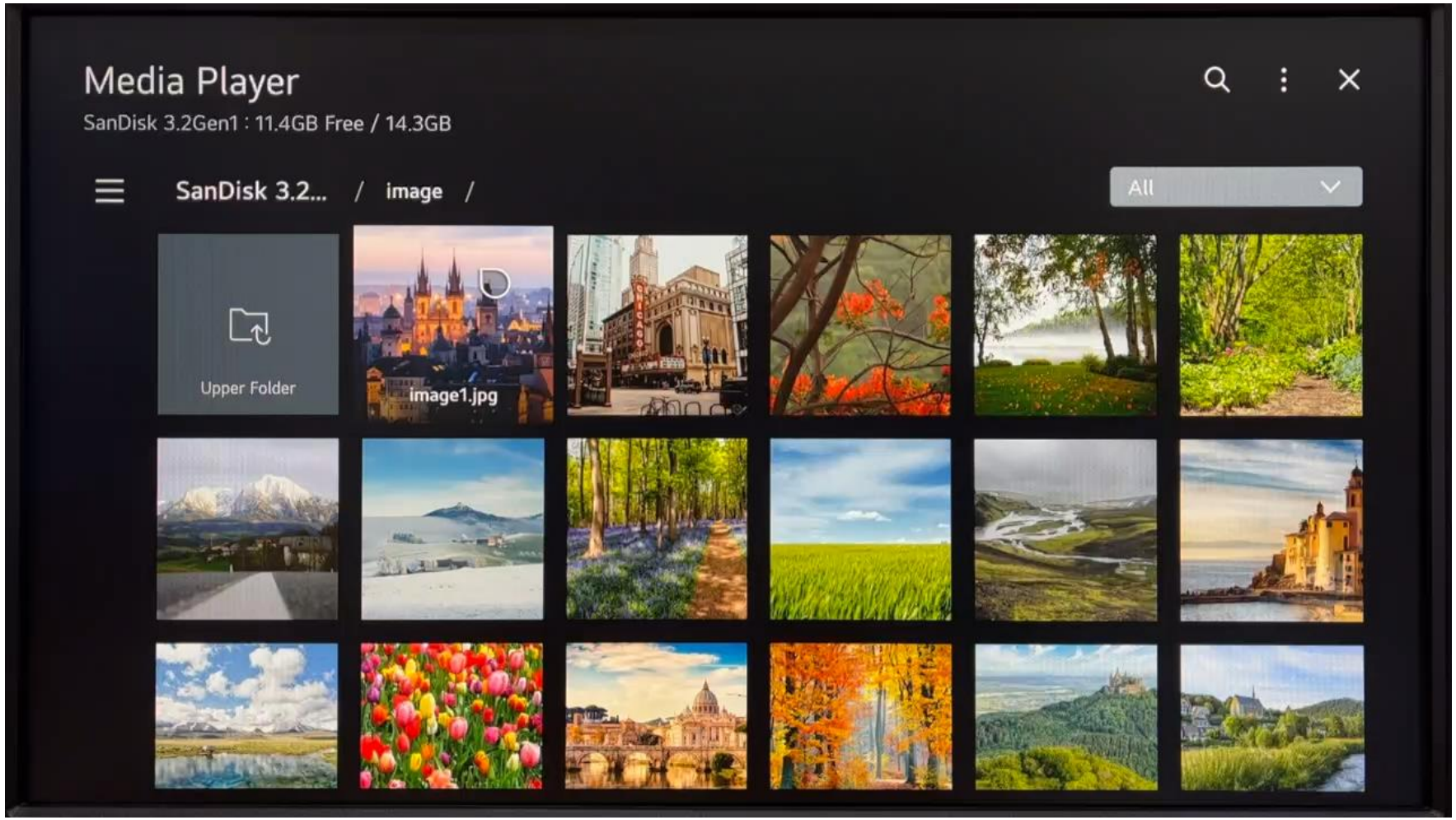
- Keyboard Navigable[1)]
  - , → **tabindex**

- @enact/spotlight[2)]
  - An extensible utility that enables users to navigate applications using a keyboard or television remote control. Responding to input from the UP, DOWN, LEFT, RIGHT, and ENTER keys.

1) https://developer.mozilla.org/en-US/docs/Web/Accessibility/Keyboard-navigable_JavaScript_widgets#general_guidelines
2) https://github.com/enactjs/enact/tree/master/packages/spotlight

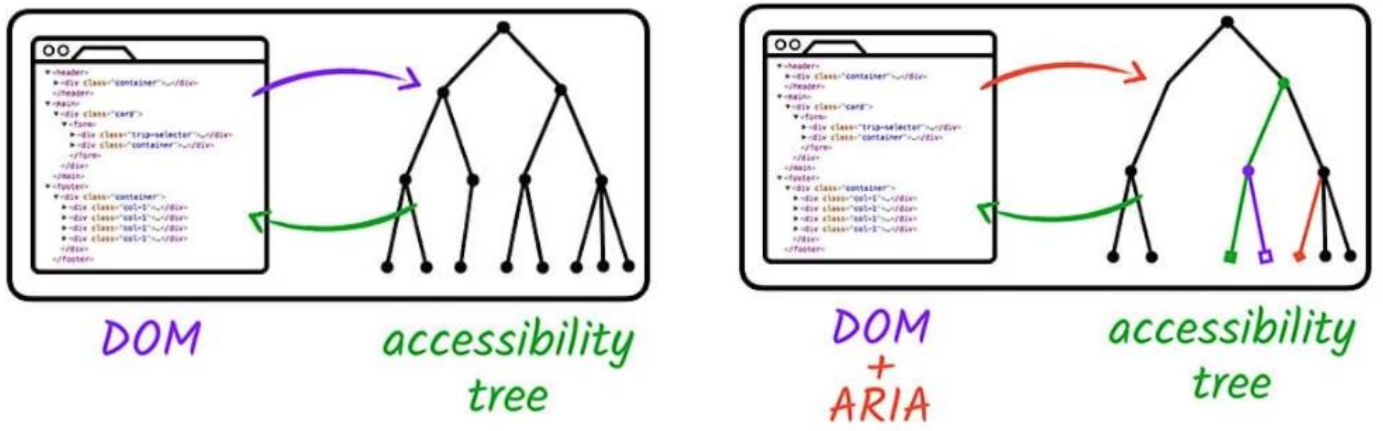
A11y Minimum Requirement

- @enact/spotlight



ARIA - Accessible Rich Internet Applications

- ARIA is a set of roles and attributes that define ways to make web content and web applications (especially those developed with JavaScript) more accessible to people with disabilities.[1)]
- **Roles**: ARIA roles can be used to describe elements that don't natively exist in HTML.
- **Attributes**: ARIA attributes enable modifying an element's states and properties as defined in the accessibility tree.



Images) https://web.dev/semantics-aria/

1) https://developer.mozilla.org/en-US/docs/Web/Accessibility/ARIA

ARIA in JSX

- aria-*, role DOM props are supported for all built-in component[1)]

```
<div
      {...rest}
      aria-checked={selected}
      aria-disabled={disabled}
      role="checkbox"
/>
```

1) https://react.dev/reference/react-dom/components/common#common-props

Use Case - 1 : Read Icon

Use Case - 1

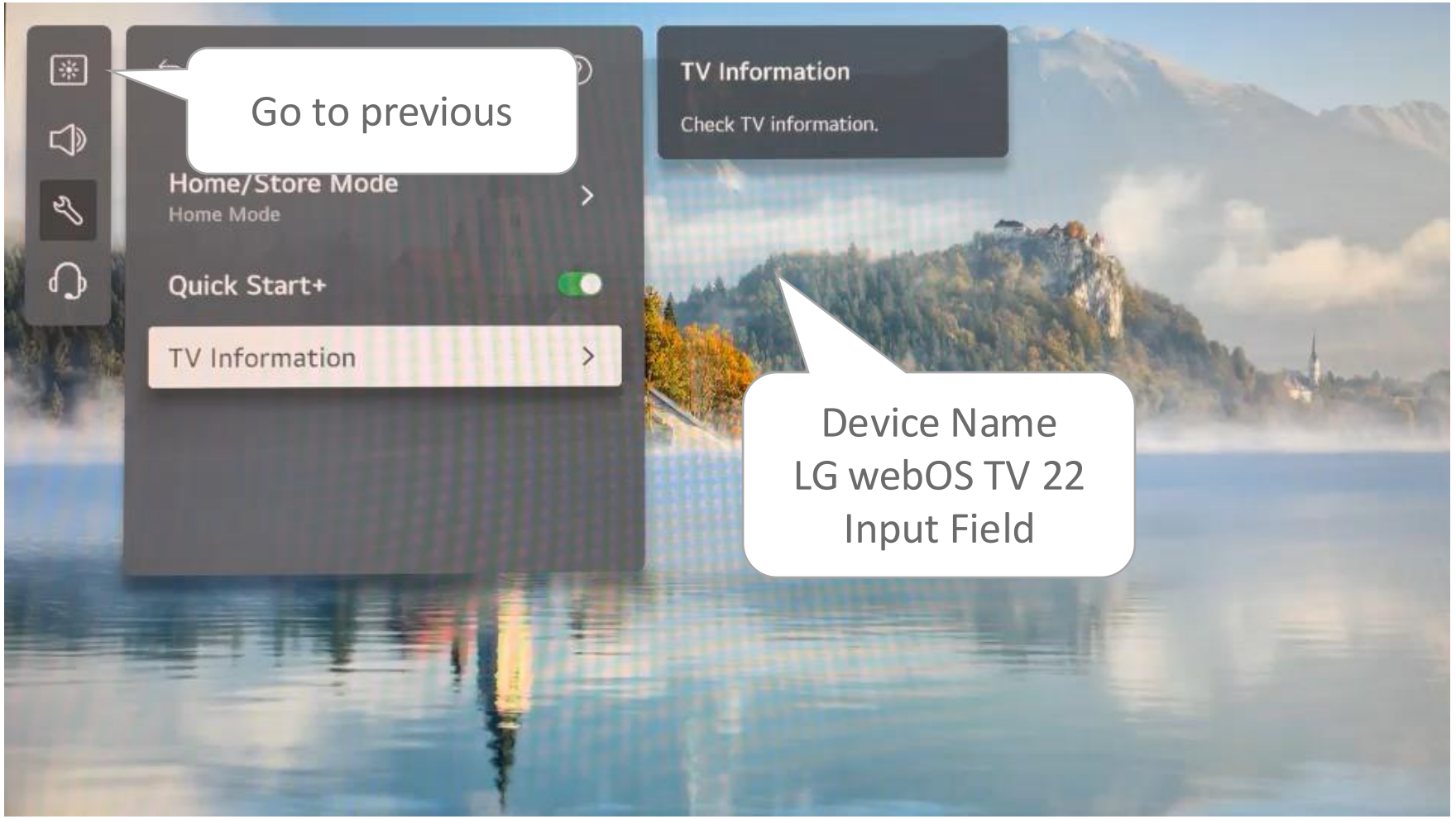
- **aria-label** : Defines a string value that labels an interactive element.

```
<Button
      aria-label="Increase"
      icon="plus"
/>
```

@enact/sandstone/Button

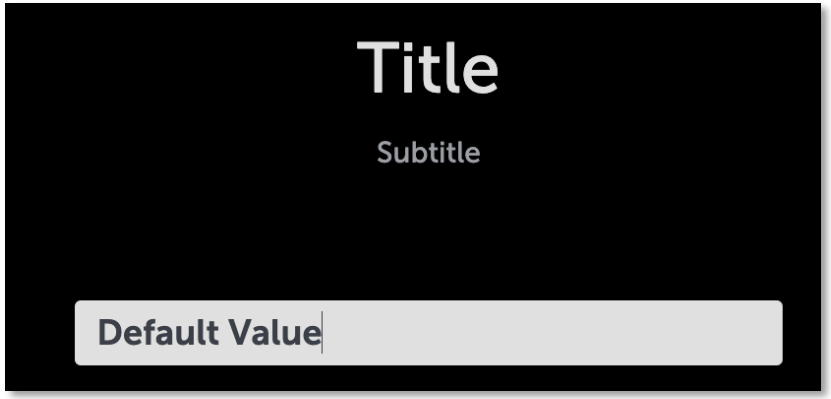
"Increase Button"

Use Case - 2 : Order matters



Use Case - 2

- **aria-labelledby**: Identifies the element(s) that labels the element it is applied to
- The **aria-labelledby** property value order matters. When more than one element is referenced by aria-labelledby, the content from each referenced element is combined in the order that they are referenced in the aria-labelledby value.




```
const ariaLabelledBy = `${id}_title ${id}_subtitle`;

return (
      <Popup
            aria-labelledby={ariaLabelledBy}
            onClose={onClose}
            onShow={onShow}
            open={open}
      />
);
```

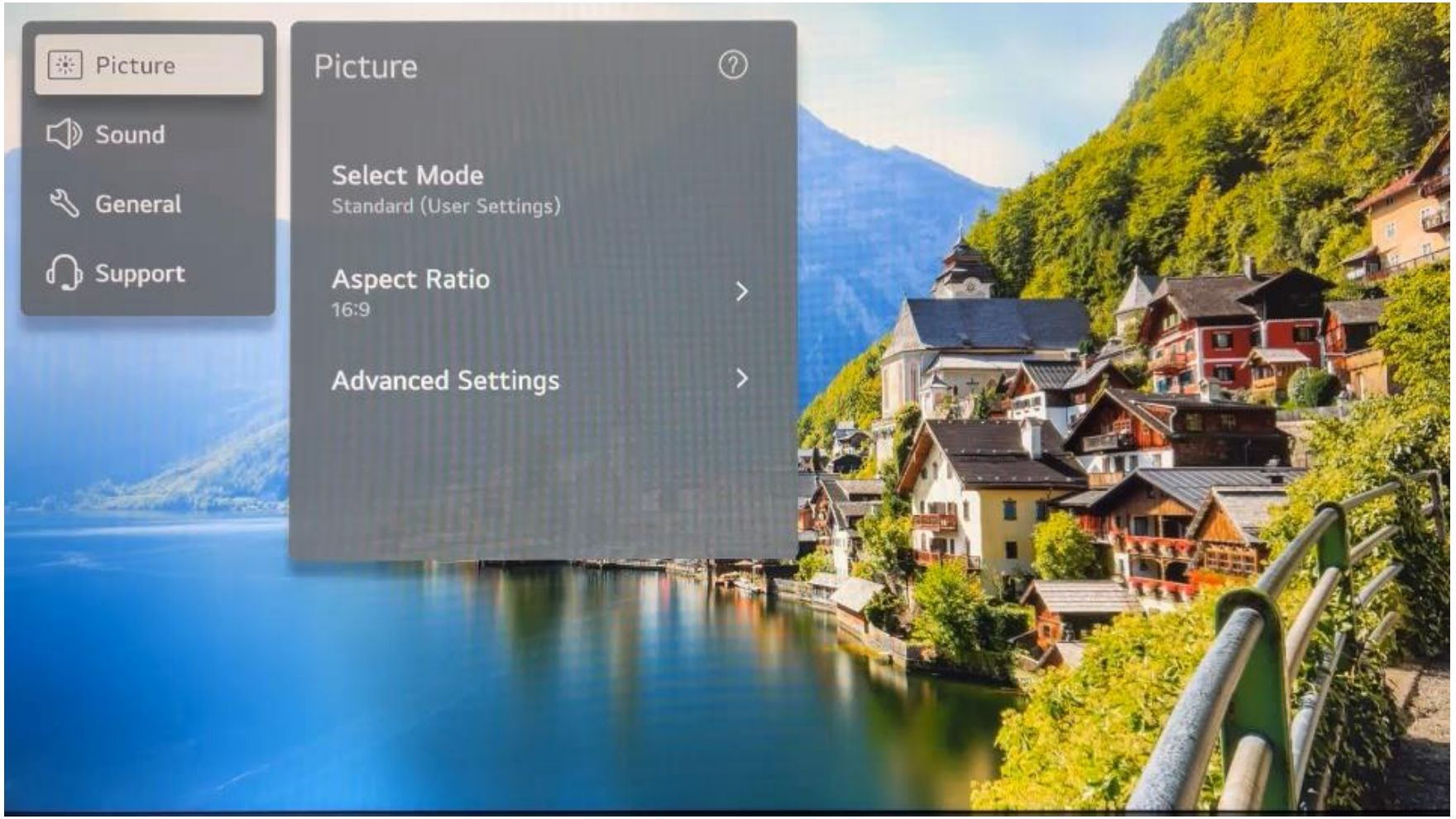
Use Case - 3 : Range widget

Use Case - 3

- **aria-valuenow:** Defines the current value for a range widget. (slider, spinbutton, etc.)
- **aria-valuetext**: Defines the human readable text alternative of aria-valuenow for a range widget.

```
getValueText () {
        const valueText = value;
        const verticalHint = `${valueText} ${$L('change a value with up down button')}`;
        const horizontalHint = `${valueText} ${$L('change a value with left right button')}`;
        return orientation === 'horizontal' ? horizontalHint : verticalHint;
}
return (
        <div
                role="slider"
                aria-valuetext={this.getValueText()}
        />
);
```

Use Case - 4 : Alert Popup



Use Case - 4

- **alert role**: The alert role is used to communicate an important and usually time-sensitive message to the user.
- **aria-live**: The global aria-live attribute indicates that an element will be updated, and describes the types of updates the user agents, assistive technologies, and user can expect from the live region.

```
return (
      <ContextualPopupRoot aria-live="off" role="alert" {...rest}>
            {children}
      </ContextualPopupRoot>
);
```

Use Case - 5 : N of Total

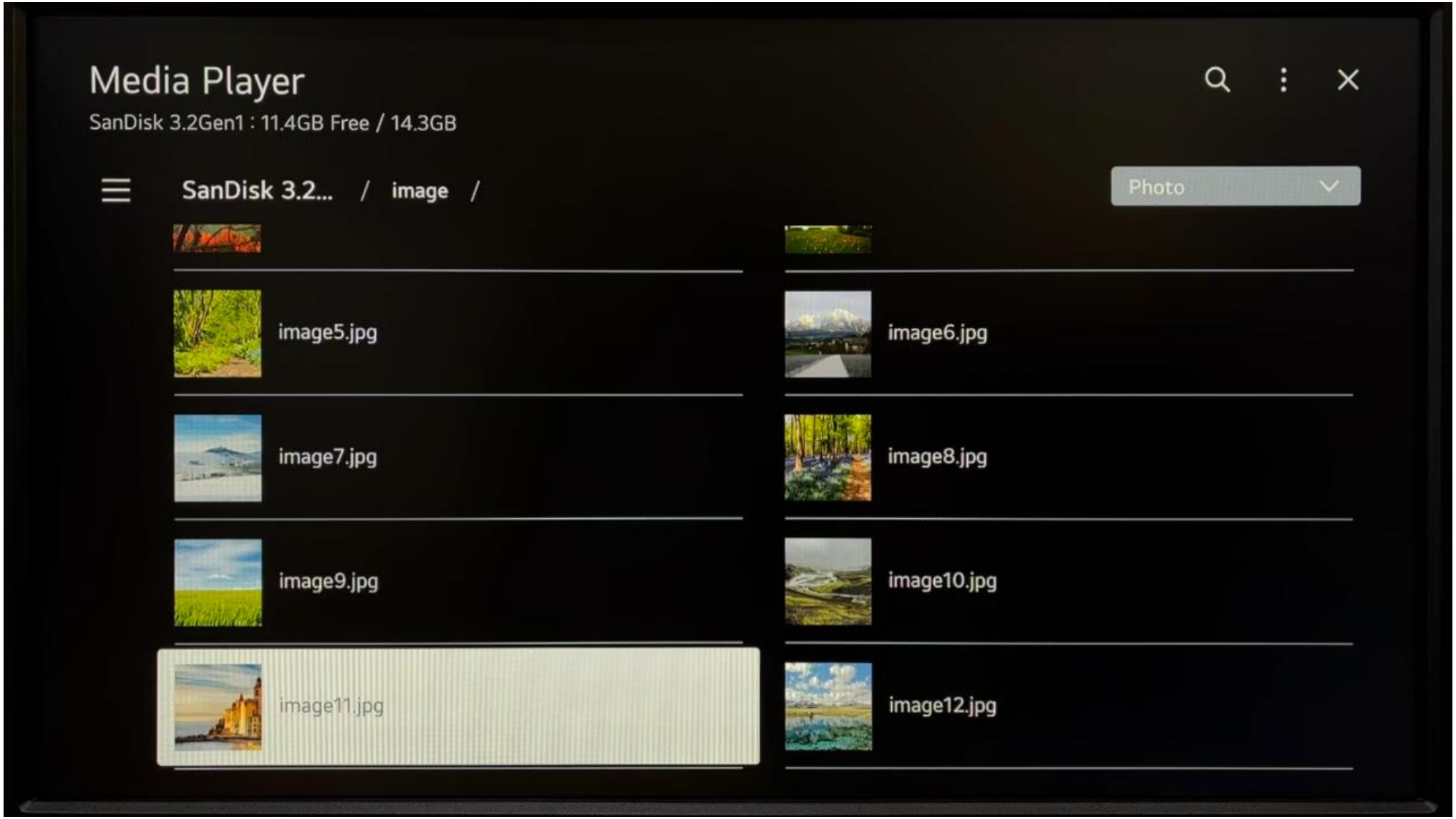
Use Case - 5 : N of Total

- Read N of Total

parent role	children role	screen reader reads
list	listitem	`... + x of y`
listbox	option	`... + seleced x of y`
menu	menuitem	`... + submenu x of y`
	menuitemcheckbox	`... + submenu x of y` or `... + checked x of y`
	menuitemradio	
group	checkbox	`... + unchecked x of y` `... + checked x of y`
	radio	
tablist	tab	`... + tab x of y`
tree	treeitem	`... +level n tree x of y`

```
<div role="tablist">
      <div role="tab">Tab 1</div>
      <div role="tab">Tab 2</div>
      <div role="tab">Tab 3</div>
</div>
```

Use Case - 5 : N of Total - Virtual List

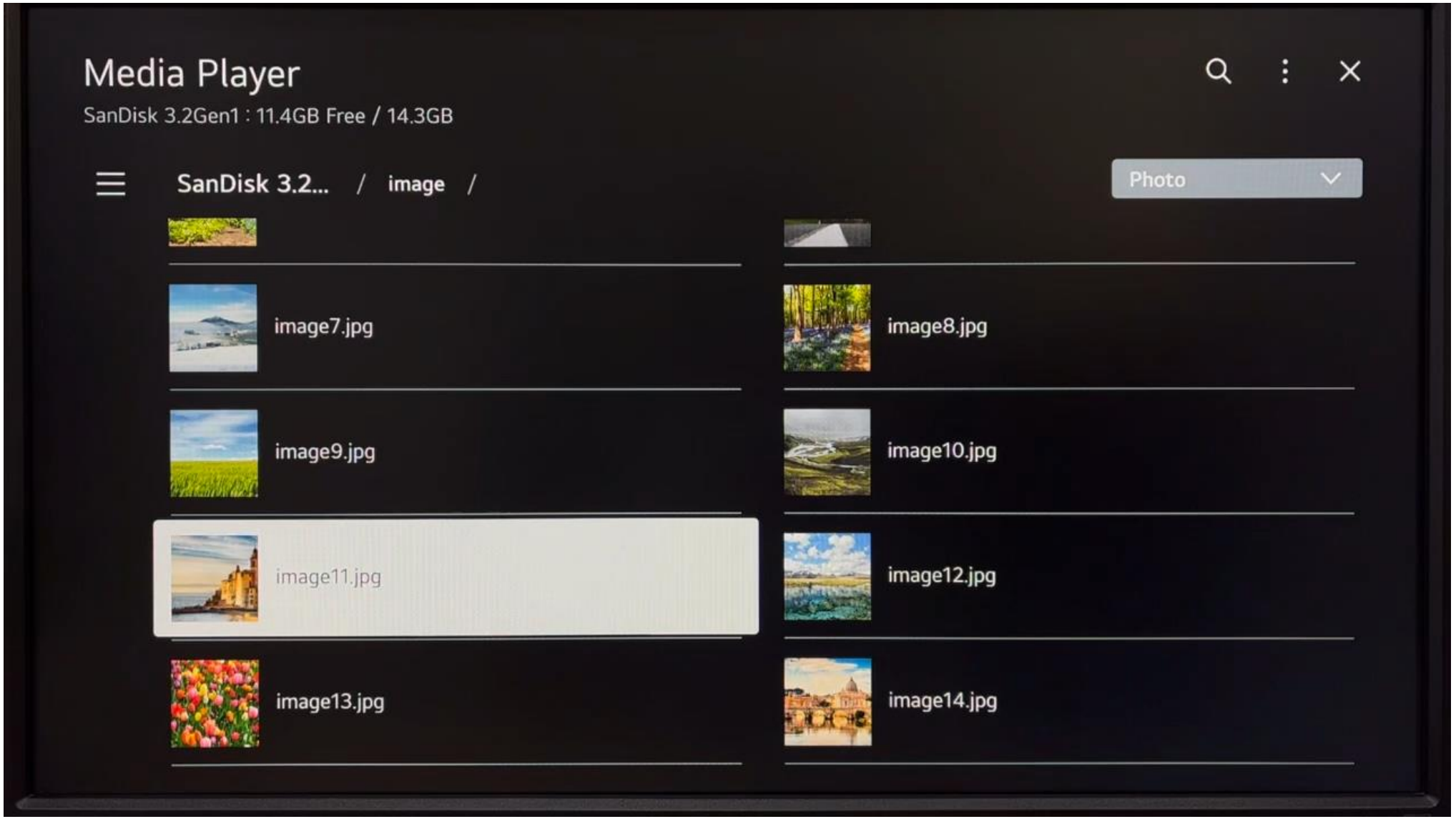


Use Case - 5 : N of Total - Virtual List

- **aria-setsize:** Defines the number of items in the current set of listitems or treeitems when not all items in the set are present in the DOM.
- **aria-posinset**: Defines an element's number or position in the current set of listitems or treeitems when not all items are present in the DOM.

```
<Item
        role="listitem"
        aria-posinset={index + 1}
        aria-setsize={items.length}
>
        {items[index]}
</Item>
```

Use Case - 5 : N of Total - Virtual List



Use Case - 6 : Announce

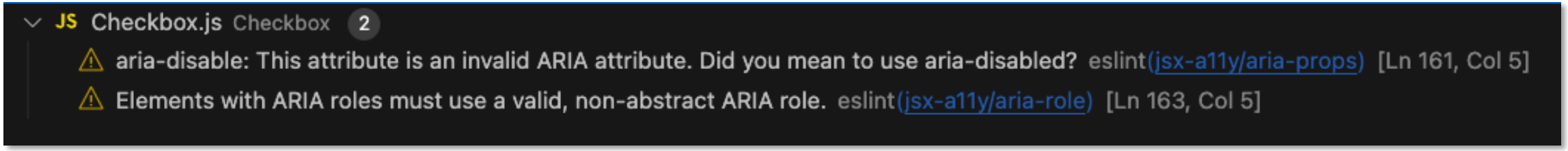
Use Case - 6

- @enact/ui/AnnounceDecorator

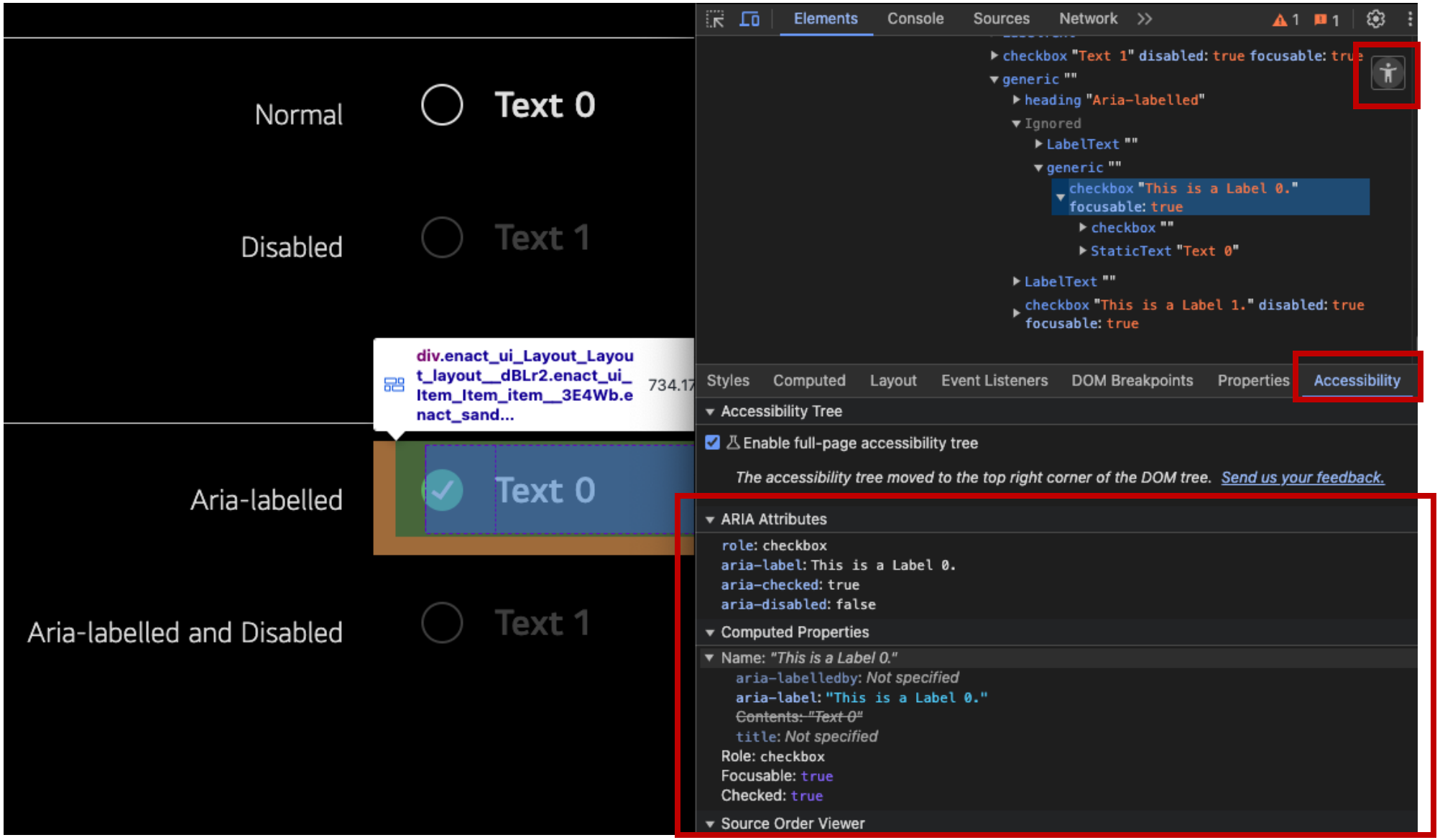
```
announce = (message) => {
        this.alert.setAttribute('aria-label', message);
};

setAlertRef = (node) => {
   this.alert = node;
};

render () {
        return (
                <span ref={this.setAlertRef} role="alert" {...props} />
        );
}
```

Tools

- **eslint-plugin-jsx-a11y**: https://www.npmjs.com/package/eslint-plugin-jsx-a11y
- This plugin does a static evaluation of the JSX to spot accessibility issues in React apps.

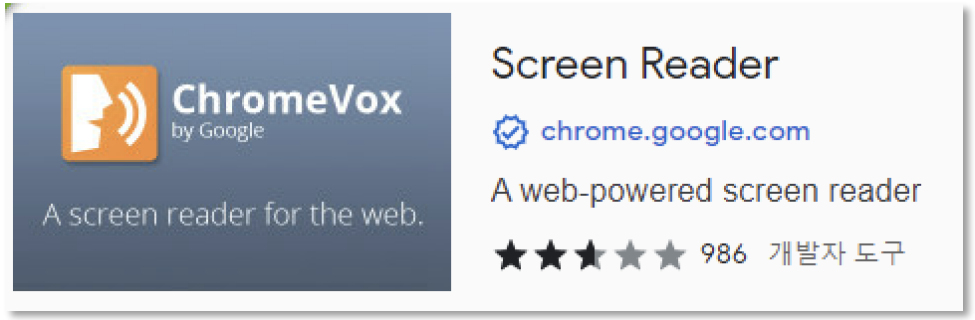


Tools – Chrome DevTools



Tools – Screen Reader

- **JAWS:** You can purchase an additional software program such as JAWS which will allow your computer to read aloud web pages to you.
- **NVDA:** NVDA (Non Visual Desktop Access) is a free screen reader that can read aloud web pages.
- **Narrator:** “Narrator” is a basic built in screen reader available for windows 8 and higher.
- **Mac OS:** Mac OS 10.4 Tiger and higher have a built-in program called VoiceOver which can read aloud text on your computer screen, including web pages.
- **Chrome extension:**



References

- W3C Accessibility: https://www.w3.org/standards/webdesign/accessibility
- MDN Accessibility: https://developer.mozilla.org/docs/Web/Accessibility
- web.dev Learn Accessibility: https://web.dev/learn/accessibility/
- **Enact**
  - Accessibility Guide: https://enactjs.com/docs/developer-guide/accessibility/
  - Spotlight Guide: https://enactjs.com/docs/developer-guide/spotlight/
  - UI components: https://github.com/enactjs/sandstone
  - UI components storybook: https://enactjs.com/sampler
  - Spotlight: https://github.com/enactjs/enact/tree/master/packages/spotlight
  - AnnounceDecorator: https://github.com/enactjs/enact/tree/master/packages/ui/AnnounceDecorator

Q&A

- https://enactjs.com
- https://github.com/enactjs
- https://x.com/EnactJS