Package: a11yShiny (via r-universe)

May 27, 2026

Type Package

Title Accessibility Enhancements to Popular R Shiny Functions

Version 0.1.4

Description Accessible wrappers for popular 'shiny' UI components, enforcing ARIA attributes and structural requirements in line with BITV 2.0 (Barrierefreie-Informationstechnik-Verordnung) and WCAG 2.1 AA. Covers action buttons, text and select inputs, fluid page layouts with HTML landmarks and skip links, 'DT' data tables, and bar and line graphs from 'ggplot2'. Components validate label presence, expose keyboard-accessible ARIA states, and provide a high-contrast toggle. This package was developed by d-fine GmbH on behalf of the German Federal Ministry of Research, Technology and Space (BMFTR).

License EUPL-1.2

Encoding UTF-8

Depends R (>= 4.1)

Imports shiny (>= 1.9.0), htmltools (>= 0.5.0), DT, ggplot2, rlang

Suggests devtools, covr, knitr, rmarkdown, bslib (>= 0.5.0), testthat (>= 3.0.0)

URL https://gitlab.opencode.de/bmbf/datenlabor/barrierefrei-r

BugReports https://gitlab.opencode.de/bmbf/datenlabor/barrierefrei-r/-/issues

VignetteBuilder knitr

RoxygenNote 7.3.3

Config/testthat/edition 3

NeedsCompilation no

Author Jan Liebnitzky [aut, cre]

Maintainer Jan Liebnitzky <datenlabor@bmftr.bund.de>

Config/pak/sysreqs cmake make libuv1-dev zlib1g-dev

Repository https://datenlabor-bmftr.r-universe.dev

Date/Publication 2026-05-27 08:51:35 UTC

RemoteUrl https://github.com/cran/a11yShiny

RemoteRef HEAD

RemoteSha db51121ea968d32441e67a035277866f5c0f2244

Contents

`a11y_actionButton` *Accessible action button*

Description

A wrapper for `shiny::actionButton()` with ARIA attributes according to BITV 2.0, enforced visible label or `aria-label`, and custom CSS class.

Usage

```
a11y_actionButton(
  inputId,
  label = NULL,
  icon = NULL,
  aria_label = NULL,
  aria_controls = NULL,
  ...
)
```

Arguments

`inputId`	Button ID
`label`	(optional) Visible label
`icon`	(optional) Icon, e.g. `shiny::icon("check")`
`aria_label`	(optional, but required if `label` is missing or empty) ARIA label for buttons without text
`aria_controls`	(optional) ARIA controls attribute
`...`	Additional arguments for `shiny::actionButton()`

Value

HTML `<button>` tag with appropriate ARIA attributes

Examples

```
# Button with a visible label
a11y_actionButton("btn1", label = "Submit")

# Icon-only button (aria_label required)
a11y_actionButton("btn2",
  icon = shiny::icon("search"),
  aria_label = "Search"
)

# Button with aria-controls linking to another element
a11y_actionButton("btn3",
  label = "Toggle",
  aria_controls = "panel1"
)

# Button with visible label, icon, and additional aria-label
a11y_actionButton("refresh",
  label = "Refresh",
  icon = shiny::icon("refresh"),
  aria_label = "Click to refresh data"
)
```

`a11y_column` *Accessible column*

Description

A wrapper for `shiny::column()` with optional ARIA attributes according to BITV 2.0 and custom CSS class. Ensures responsive layout by validating that columns have appropriate widths.

Usage

```
a11y_column(width, ..., offset = NULL, id = NULL, aria_label = NULL)
```

Arguments

width	Column width (as in shiny::column(): 1–12)
...	Content for column
offset	(optional) Column offset (as in shiny::column(), 1–12)
id	(optional) Column ID
aria_label	(optional) aria-label for the column/region

Value

HTML <div> tag with classes for grid and a11y

Examples

```
# Simple column
a11y_column(6, htmltools::p("Half width content"))

# Column with offset and aria-label
a11y_column(4,
  offset = 2,
  aria_label = "Sidebar",
  htmltools::p("Offset column")
)

# Column with a descriptive aria-label for a form section
a11y_column(4,
  aria_label = "Histogram settings",
  htmltools::p("Configuration controls go here")
)
```

---

a11y_dateInput *Accessible date input*

Description

A wrapper for shiny::dateInput() with ARIA attributes according to BITV 2.0, enforced visible label, optional heading annotation, custom CSS class, and sr-only description support.

Usage

```
a11y_dateInput(
  inputId,
  label,
  value = NULL,
  min = NULL,
  max = NULL,
  format = "yyyy-mm-dd",
  startview = "month",
  weekstart = 1,
  language = "en",
  width = NULL,
  ...,
  describedby = NULL,
  describedby_text = NULL,
  heading_level = NULL,
  aria_controls = NULL
)
```

Arguments

`inputId`	Input ID
`label`	Visible label (required)
`value`	Initial date value (Date or string, optional)
`min`	Minimum date (optional)
`max`	Maximum date (optional)
`format`	Date format (as in `shiny::dateInput()`, default: `"yyyy-mm-dd"`)
`startview`	Initial view (`"month"`, `"year"`, `"decade"`; default: `"month"`)
`weekstart`	First day of the week (1 = Monday, default: `1`)
`language`	Language for the datepicker (default: `"en"`)
`width`	Control width (optional)
`...`	Additional arguments for `shiny::dateInput()`
`describedby`	ID of an element for `aria-describedby` (optional)
`describedby_text`	Creates an sr-only `<div>` that serves as a description and is linked via `aria-describedby`. If `describedby` is set, its ID is used, otherwise an ID is generated (`<inputId>-desc`). (optional)
`heading_level`	1–6, marks the visible `<label>` as a heading via `role="heading"` and `aria-level` (optional)
`aria_controls`	(optional) ARIA controls attribute

Value

HTML tag of the input component (possibly with sr-only description)

Examples

```
# Basic date input
a11y_dateInput("start", "Start date")

# With German datepicker, custom format and description
a11y_dateInput("birthday", "Date of birth",
  format = "dd.mm.yyyy",
  language = "de",
  describedby_text = "Enter your date of birth"
)

# Date input with heading-level annotation
a11y_dateInput("mydate", "Select date:",
  language = "de",
  heading_level = 2
)
```

---

a11y_fluidPage — *Accessible fluid page*

Description

A wrapper for `shiny::fluidPage()` with required `title`/`lang`, one main landmark, optional header/nav/aside/footer, skip link, and flexible layout CSS.

Usage

```
a11y_fluidPage(
  ...,
  title,
  lang = NULL,
  main = NULL,
  main_id = "main-content",
  header = NULL,
  nav = NULL,
  aside = NULL,
  footer = NULL
)
```

Arguments

`...`	Content to go inside <main> when `main` is `NULL`, or extra children to append to `main`
`title`	Page title (required)
`lang`	Language code (required)

`main`	(optional) <main> tag (`tags$main(...)`); if provided, it is normalized (id/class ensured)
`main_id`	(optional) ID to assign if the main has no id yet (default `"main-content"`)
`header`	(optional) Header content or `tags$header(...)`
`nav`	(optional) Nav content or `tags$nav(...)`
`aside`	(optional) Aside content or `tags$aside(...)`
`footer`	(optional) Footer content or `tags$footer(...)`

Details

- Enforces `title` and `lang`
- Ensures exactly one `<main id="main-content">`
- Attaches the CSS dependency (`use_a11y`) for flexible layout (`.a11y-flow`)
- Allows optional header/nav/aside/footer blocks

Value

A Shiny UI tag (page)

Examples

```
# Minimal accessible page
a11y_fluidPage(
  title = "My App",
  lang  = "en",
  htmltools::h1("Welcome")
)

# Page with header and footer
a11y_fluidPage(
  title  = "Dashboard",
  lang   = "de",
  header = htmltools::tags$header(htmltools::h1("Dashboard")),
  footer = htmltools::tags$footer("Footer content"),
  htmltools::p("Main content goes here")
)
```

`a11y_fluidRow` *Accessible fluid row*

Description

A wrapper for `shiny::fluidRow()` with optional section landmarks, ARIA attributes according to BITV 2.0, and custom CSS class. Ensures responsive layout by validating that all columns are `a11y_columns` and their widths sum to 12.

Usage

```
a11y_fluidRow(..., id = NULL, aria_label = NULL)
```

Arguments

`...`	Content to place into the row (columns, tags)
`id`	(optional) Row ID
`aria_label`	(optional) `aria-label` for the row (helpful for sections)

Value

HTML <section> tag (row)

Examples

```
# A row with two columns (widths must sum to 12)
a11y_fluidRow(
  a11y_column(6, htmltools::p("Left")),
  a11y_column(6, htmltools::p("Right"))
)

# Labeled section row
a11y_fluidRow(
  a11y_column(12, htmltools::p("Full width")),
  aria_label = "Results section"
)
```

---

a11y_ggplot2_bar *Accessible bar chart*

Description

A minimal wrapper for a simple bar chart with accessibility features using ggplot2::ggplot2. Provides a high-contrast color palette, black bar outlines, and a minimal theme by default.

Usage

```
a11y_ggplot2_bar(
  data,
  x,
  y,
  accessible_colors = NULL,
  border_color = "#000000",
  bar_width = 0.9,
  legend_title = NULL,
  ...
)
```

Arguments

`data`	Data frame
`x`	Column name for categories (unquoted)
`y`	Column name for bar heights (unquoted)
`accessible_colors`	(optional) Character vector of colors for the palette
`border_color`	(optional) Color for bar outlines (default: `"#000000"`)
`bar_width`	(optional) Bar width (default: `0.9`)
`legend_title`	(optional) Title for the legend (default: `NULL`)
`...`	Additional arguments passed to `ggplot2::labs()` (e.g. `title`, axis labels)

Value

A `ggplot2::ggplot()` object

Examples

```
df <- data.frame(
  category = c("Alpha", "Beta", "Gamma"),
  count    = c(23, 17, 31)
)
a11y_ggplot2_bar(
  data = df, x = category, y = count,
  title = "Counts by Category"
)
```

---

a11y_ggplot2_line *Accessible line chart*

Description

A minimal wrapper for a multi-line chart with accessibility features using ggplot2::ggplot2. Provides a high-contrast color palette, distinct marker shapes, and a minimal theme by default.

Usage

```
a11y_ggplot2_line(
  data,
  x,
  y,
  group = NULL,
  accessible_colors = NULL,
  marker_shapes = NULL,
  line_width = 1,
```

```
  marker_size = 2,
  legend_title = NULL,
  ...
)
```

Arguments

`data`	Data frame
`x`	Column name for the x-axis (unquoted)
`y`	Column name for the y-axis (unquoted)
`group`	(optional) Column name for grouping lines (unquoted)
`accessible_colors`	(optional) Character vector of colors for the palette
`marker_shapes`	(optional) Numeric vector of point shapes, e.g. `c(16, 17, 15, 3, 7, 4, 10)`
`line_width`	(optional) Line width (default: `1`)
`marker_size`	(optional) Point size (default: `2`)
`legend_title`	(optional) Title for the legend
`...`	Additional arguments passed to `ggplot2::labs()`

Value

A `ggplot2::ggplot()` object

Examples

```
# Simple line chart (no grouping)
df <- data.frame(year = 2020:2024, value = c(10, 14, 13, 17, 20))
a11y_ggplot2_line(
  data = df, x = year, y = value,
  title = "Trend"
)

# Grouped line chart
df2 <- data.frame(
  year  = rep(2020:2024, 2),
  value = c(10, 14, 13, 17, 20, 8, 9, 11, 12, 15),
  grp   = rep(c("A", "B"), each = 5)
)
a11y_ggplot2_line(
  data = df2, x = year, y = value,
  group = grp, title = "Trend by Group"
)

# With legend title and additional ggplot2 layers
p <- a11y_ggplot2_line(
  data = df2, x = year, y = value, group = grp,
  legend_title = "Group",
  title = "Trend by Group"
```

```
)
p + ggplot2::geom_hline(yintercept = 12, linetype = "dashed") +
  ggplot2::labs(
    x = "Year", y = "Value",
    subtitle = "With custom axis labels"
  )
```

---

a11y_highContrastButton *Accessible high-contrast toggle button*

Description

A toggle button that switches high-contrast mode by adding/removing the CSS class `.high-contrast` on the <body> element, with ARIA attributes according to BITV 2.0.

Usage

```
a11y_highContrastButton(
  inputId = "toggle_contrast",
  label = "Contrast Mode",
  icon = shiny::icon("adjust"),
  aria_label = "Toggle high-contrast mode on or off",
  ...
)
```

Arguments

`inputId`	Button ID (default: `"toggle_contrast"`)
`label`	Visible label (optional, e.g. `"High Contrast"`)
`icon`	Optional icon, e.g. `shiny::icon("adjust")`
`aria_label`	ARIA label (required if `label` is missing/empty)
`...`	Additional arguments for `a11y_actionButton()`

Details

- Uses `a11y_actionButton()` internally (visible label **or** `aria-label`).
- Sets `aria-pressed` and toggles this state via JS.
- Adds the class `.a11y-high-contrast-toggle` to which the JS is attached.

Value

HTML tag of the button component

Examples

```
# Default high-contrast toggle button
a11y_highContrastButton()

# Custom label in English
a11y_highContrastButton(
  label      = "High Contrast",
  aria_label = "Toggle high-contrast mode"
)

# Icon-only toggle (no visible label)
a11y_highContrastButton(
  label      = NULL,
  icon       = shiny::icon("adjust"),
  aria_label = "Toggle high-contrast mode"
)
```

---

a11y_numericInput *Accessible numeric input*

Description

A wrapper for `shiny::numericInput()` with ARIA attributes according to BITV 2.0, enforced visible label, optional heading annotation, custom CSS class, and sr-only description support.

Usage

```
a11y_numericInput(
  inputId,
  label,
  value,
  min = NA,
  max = NA,
  step = NA,
  ...,
  describedby = NULL,
  describedby_text = NULL,
  heading_level = NULL,
  aria_controls = NULL
)
```

Arguments

`inputId`	Input ID
`label`	Visible label (required)
`value`	Initial numeric value

`min`	Minimum value (optional; `NA` for none)
`max`	Maximum value (optional; `NA` for none)
`step`	Step size (optional; `NA` for none)
`...`	Additional arguments for `shiny::numericInput()`
`describedby`	ID of help text for `aria-describedby` (optional)
`describedby_text`	Creates an sr-only `<div>` that serves as a description and is linked via `aria-describedby`. If `describedby` is set, its ID is used, otherwise an ID is generated (`<inputId>-desc`). (optional)
`heading_level`	1–6, marks the visible `<label>` as a heading via `role="heading"` and `aria-level` (optional)
`aria_controls`	(optional) ARIA controls attribute

Value

HTML tag of the input component

Examples

```
# Basic numeric input
a11y_numericInput("age", "Age", value = 30, min = 0, max = 120)

# With heading-level annotation and description
a11y_numericInput("score", "Score",
  value = 0,
  min = 0, max = 100,
  heading_level = 3,
  describedby_text = "Enter a value between 0 and 100"
)

# Linking to an existing help-text element by its ID
a11y_numericInput("seed", "Seed",
  value = 123,
  describedby = "seed_help"
)
```

`a11y_radioButtons` *Accessible radio buttons*

Description

A wrapper for `shiny::radioButtons()` with ARIA attributes according to BITV 2.0, enforced visible label, optional heading annotation, custom CSS class, and sr-only description support.

Usage

```
a11y_radioButtons(
  inputId,
  label,
  choices,
  selected = NULL,
  inline = FALSE,
  ...,
  describedby = NULL,
  describedby_text = NULL,
  heading_level = NULL,
  aria_controls = NULL
)
```

Arguments

`inputId`	Input ID
`label`	Visible label (required)
`choices`	Choices (as in `shiny::radioButtons()`)
`selected`	Preselected value (optional)
`inline`	Display inline (logical, default: `FALSE`)
`...`	Additional arguments for `shiny::radioButtons()`
`describedby`	ID of an element for `aria-describedby` (optional)
`describedby_text`	Creates an sr-only `<div>` that serves as a description and is linked via `aria-describedby`. If `describedby` is set, its ID is used, otherwise an ID is generated (`<inputId>-desc`). (optional)
`heading_level`	1–6, marks the visible `<label>` as a heading via `role="heading"` and `aria-level` (optional)
`aria_controls`	(optional) ARIA controls attribute

Value

HTML tag of the input component (possibly with sr-only description)

Examples

```
# Basic radio buttons
a11y_radioButtons("format", "Output format",
  choices = c("CSV", "Excel", "JSON")
)

# Inline layout with heading annotation
a11y_radioButtons("yesno", "Agree?",
  choices = c("Yes", "No"),
  inline = TRUE,
  heading_level = 4
```

```
)

# With named choices and a screen-reader description
a11y_radioButtons("radio_choice", "Pick one",
  choices = list(
    "Option 1" = 1,
    "Option 2" = 2,
    "Option 3" = 3
  ),
  selected = 1,
  describedby_text = "Select one of the three options"
)
```

---

a11y_renderDataTable *Accessible DataTable renderer*

Description

A wrapper for `DT::renderDataTable()` that enables keyboard navigation (KeyTable extension) by default and provides built-in German/English translations.

Usage

```
a11y_renderDataTable(expr, lang = NULL, dt_language = NULL, ...)
```

Arguments

`expr`	Table expression
`lang`	Language code ("de" or "en"), or `NULL` if `dt_language` is set in `expr()`
`dt_language`	(optional) DT language list (see DT docs); required when using a language other than "de"/"en"
`...`	Other `DT::renderDataTable()` arguments

Value

A Shiny render function

Examples

```
# Inside a Shiny server function
if (interactive()) {
  library(shiny)
  server <- function(input, output, session) {
    output$table <- a11y_renderDataTable(
      expr = mtcars[, 1:5],
      lang = "en"
    )
```

```
  }
}

# German-language table with Buttons extension and accessible export options
if (interactive()) {
  library(shiny)
  server <- function(input, output, session) {
    output$table_de <- a11y_renderDataTable(
      expr = head(iris[, 1:4], 10),
      lang = "de",
      selection = "none",
      extensions = c("Buttons"),
      options = list(
        pageLength = 5,
        dom = "Bfrtip",
        buttons = c("excel", "csv")
      )
    )
  }
}
```

`a11y_selectInput` *Accessible select input*

Description

A wrapper for `shiny::selectInput()` with ARIA attributes according to BITV 2.0, enforced visible label, optional heading annotation, custom CSS class, and sr-only description support.

Usage

```
a11y_selectInput(
  inputId,
  label,
  choices,
  selected = NULL,
  multiple = FALSE,
  ...,
  describedby = NULL,
  describedby_text = NULL,
  heading_level = NULL,
  aria_controls = NULL
)
```

Arguments

`inputId`	Input ID
`label`	Visible label (required)
`choices`	Choice list (as in `shiny::selectInput()`)
`selected`	Preselection (optional)
`multiple`	Multiple selection (default: `FALSE`)
`...`	Additional arguments for `shiny::selectInput()`
`describedby`	ID of an element for `aria-describedby` (optional)
`describedby_text`	Creates an sr-only <`div`> that serves as a description and is linked via `aria-describedby`. If `describedby` is set, its ID is used, otherwise an ID is generated (<`inputId`>`-desc`). (optional)
`heading_level`	1–6, marks the visible <`label`> as a heading via `role="heading"` and `aria-level` (optional)
`aria_controls`	(optional) ARIA controls attribute

Value

HTML tag of the input component (possibly with sr-only description)

Examples

```
# Basic select input
a11y_selectInput("colour", "Colour",
  choices = c("Red", "Green", "Blue")
)

# With a screen-reader-only description
a11y_selectInput("size", "Size",
  choices = c("S", "M", "L"),
  describedby_text = "Choose a t-shirt size"
)

# Label promoted to a heading (useful for sectioned forms)
a11y_selectInput("n_breaks", "Number of bins",
  choices = c(10, 20, 35, 50),
  selected = 20,
  heading_level = 3
)

# Linking to an existing description element via its ID
htmltools::tags$p(id = "n_breaks_help", "Choose how many bins to display")
a11y_selectInput("n_breaks2", "Number of bins",
  choices = c(10, 20, 35, 50),
  describedby = "n_breaks_help"
)
```

`a11y_textButtonGroup` *Accessible text input with action button*

Description

A composite component combining `a11y_textInput()` and `a11y_actionButton()` with BITV 2.0-conform ARIA attributes. The text input enforces a visible label; the button enforces either a visible label or an `aria-label`. By default the button gets `aria-controls` pointing to the text input.

Usage

```
a11y_textButtonGroup(
  textId,
  buttonId,
  label,
  value = "",
  placeholder = NULL,
  button_label = NULL,
  button_icon = NULL,
  button_aria_label = NULL,
  controls = NULL,
  layout = c("inline", "stack"),
  text_describedby = NULL,
  text_describedby_text = NULL,
  text_heading_level = NULL
)
```

Arguments

`textId`	ID for the inner text input (required)
`buttonId`	ID for the inner action button (required)
`label`	Visible label for the text input (required)
`value`	Initial value for the text input (default: `""`)
`placeholder`	Placeholder for the text input (optional)
`button_label`	Visible label for the button (optional; if omitted/empty, `button_aria_label` is required)
`button_icon`	Icon for the button (e.g. `shiny::icon("search")`, optional)
`button_aria_label`	ARIA label for the button (required if `button_label` is missing or empty)
`controls`	ID referenced by `aria-controls` on the button. If `NULL`, defaults to `textId`.
`layout`	`"inline"` (default) or `"stack"`. Inline places text and button in one row; stack places them under each other.

`text_describedby`
: ID for `aria-describedby` of the text input (optional)

`text_describedby_text`
: Creates an sr-only <div> as description for the text input. If `text_describedby` is set, that ID is used, otherwise <textId>-desc (optional)

`text_heading_level`
: 1–6, marks the visible <label> of the text input as heading via `role="heading"` and `aria-level` (optional)

Value

HTML tag containing the labeled text input and button

Examples

```
# Search box with icon-only button
a11y_textButtonGroup(
    textId = "search_text",
    buttonId = "search_btn",
    label = "Search",
    placeholder = "Enter a keyword",
    button_icon = shiny::icon("search"),
    button_aria_label = "Run search"
)

# Stacked layout with visible button label
a11y_textButtonGroup(
    textId = "comment_text",
    buttonId = "comment_btn",
    label = "Comment",
    button_label = "Send",
    layout = "stack"
)

# With screen-reader description and heading-level on the label
a11y_textButtonGroup(
    textId = "query_text",
    buttonId = "query_btn",
    label = "Search query",
    placeholder = "Enter a keyword",
    button_icon = shiny::icon("search"),
    button_aria_label = "Run search",
    text_describedby_text = "Type a keyword and press the search button",
    text_heading_level = 3
)
```

a11y_textInput *Accessible text input*

Description

A wrapper for `shiny::textInput()` with ARIA attributes according to BITV 2.0, enforced visible label, optional heading annotation, custom CSS class, and sr-only description support.

Usage

```
a11y_textInput(
  inputId,
  label,
  value = "",
  width = NULL,
  placeholder = NULL,
  ...,
  describedby = NULL,
  describedby_text = NULL,
  heading_level = NULL,
  aria_controls = NULL
)
```

Arguments

`inputId`	Input ID
`label`	Visible label (required)
`value`	Initial text value (default: `""`)
`width`	Control width (optional)
`placeholder`	Placeholder text (optional)
`...`	Additional arguments for `shiny::textInput()`
`describedby`	ID of an element for `aria-describedby` (optional)
`describedby_text`	Creates an sr-only `<div>` that serves as a description and is linked via `aria-describedby`. If `describedby` is set, its ID is used, otherwise an ID is generated (`<inputId>-desc`). (optional)
`heading_level`	1–6, marks the visible `<label>` as a heading via `role="heading"` and `aria-level` (optional)
`aria_controls`	(optional) ARIA controls attribute

Value

HTML tag of the input component (possibly with sr-only description)

Examples

```
# Basic text input
a11y_textInput("name", "Full name")

# With placeholder and screen-reader description
a11y_textInput("email", "E-mail address",
  placeholder = "user@example.com",
  describedby_text = "We will never share your e-mail"
)

# With heading-level annotation (for sectioned forms)
a11y_textInput("company", "Company name",
  heading_level = 3
)

# Linking to an existing description element via its ID
a11y_textInput("query", "Search query",
  describedby = "query_help"
)
```

---

a11y_textInputsGroup *Accessible group of text inputs*

Description

A wrapper for a group of related text input elements inside a `<fieldset>`/`<legend>` with ARIA attributes according to BITV 2.0. Intended for combined form elements that belong together (e.g. address fields, date parts, etc.).

Usage

```
a11y_textInputsGroup(
  groupId,
  legend,
  inputs,
  describedby = NULL,
  describedby_text = NULL,
  legend_heading_level = NULL
)
```

Arguments

groupId	ID for the fieldset/group (used as id of `<fieldset>`)
legend	Visible group label (used as `<legend>`, required)
inputs	A list of input specifications. Each element must be a list with:
	inputId ID of the text input (required)

label Visible label for this text field (optional)

value Initial value (optional, default: `""`)

placeholder Placeholder text (optional)

width Width for this field (optional, as in `shiny::textInput()`)

aria_label Accessible name via `aria-label` (optional – required if no visible label and no title)

title Title attribute for additional explanation or as accessible name if no visible label (optional)

`describedby` ID of an element used for `aria-describedby` on the group (optional)

`describedby_text` Creates an sr-only `<div>` that serves as a description for the group and is linked via `aria-describedby`. If `describedby` is set, its ID is used, otherwise an ID is generated (`<groupId>-desc`). (optional)

`legend_heading_level` 1–6, marks the visible `<legend>` as a heading via `role="heading"` and `aria-level` (optional)

Details

- The group gets a visible legend (required).
- Each inner text input has its own ID.
- For each inner input you must provide at least **one** of: a visible `label`, an `aria_label`, or a `title` attribute.
- The fieldset is marked as `role="group"` and is linked to the legend via `aria-labelledby`.

Value

HTML tag of the fieldset containing multiple text inputs (possibly with an sr-only group description)

Examples

```
# Address field group
a11y_textInputsGroup(
    groupId = "address",
    legend = "Address",
    inputs = list(
        list(inputId = "street", label = "Street"),
        list(inputId = "city", label = "City"),
        list(inputId = "zip", label = "ZIP code")
    )
)

# With group description and legend promoted to heading level
a11y_textInputsGroup(
    groupId = "address_full",
    legend = "Postal address",
    inputs = list(
        list(inputId = "street2", label = "Street and number"),
```

```
        list(inputId = "zip2", label = "ZIP code", width = "120px"),
        list(inputId = "city2", label = "City"),
        list(inputId = "country2", label = "Country")
    ),
    describedby_text = "Please enter your full postal address.",
    legend_heading_level = 3
)
```

Index