Package 'shinyNextUI'

July 23, 2025

Title 'HeroUI' 'React' Template for 'shiny' Apps

Version 0.1.0

Description A set of user interface components to create outstanding 'shiny' apps <https://shiny.posit.co/>, with the power of 'React' 'JavaScript' <https://react.dev/>. Seamlessly support dark and light themes, customize CSS with 'tailwind' <https://tailwindcss.com/>.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

URL https://rinterface.github.io/shinyNextUI/

Imports htmltools, shiny, shiny.react, jsonlite

Suggests testthat (>= 3.0.0), shinytest2, purrr, thematic, shiny.router, roxy.shinylive

Config/testthat/edition 3

Depends R (>= 2.10)

LazyData true

NeedsCompilation no

Author David Granjon [aut, cre], Next UI Inc [ctb, cph] (HeroUI template: <https://www.heroui.com/>)

Maintainer David Granjon <dgranjon@ymail.com>

Repository CRAN

Date/Publication 2025-03-17 08:50:02 UTC

Contents

accordion *accordion*

Description

Accordion display a list of high-level options that can expand/collapse to reveal more information.

Usage

```
accordion(inputId, ..., value = default_value)

accordion_item(...)

update_accordion(session = shiny::getDefaultReactiveDomain(), inputId, ...)
```

Arguments

inputId	ID of the component.
...	Props to pass to the component. The allowed props are listed below in the **Details** section.
value	Starting value.
session	Object passed as the session argument to Shiny server.

Details

- **children**. Type: ReactNode OR ReactNode[]. Default: NA.
- **variant**. Type: light OR shadow OR bordered OR splitted. Default: "light".
- **selectionMode**. Type: none OR single OR multiple. Default: NA.
- **selectionBehavior**. Type: toggle OR replace. Default: "toggle".
- **isCompact**. Type: boolean. Default: false.
- **isDisabled**. Type: boolean. Default: false.
- **showDivider**. Type: boolean. Default: true.
- **dividerProps**. Type: DividerProps. Default: NA.
- **hideIndicator**. Type: boolean. Default: false.
- **disableAnimation**. Type: boolean. Default: false.
- **disableIndicatorAnimation**. Type: boolean. Default: false.
- **disallowEmptySelection**. Type: boolean. Default: false.
- **keepContentMounted**. Type: boolean. Default: false.
- **fullWidth**. Type: boolean. Default: true.
- **motionProps**. Type: MotionProps. Default: NA.
- **disabledKeys**. Type: React.Key[]. Default: NA.

- **itemClasses**. Type: `AccordionItemClassnames`. Default: `NA`.
- **selectedKeys**. Type: `all` OR `React.Key[]`. Default: `NA`.
- **defaultSelectedKeys**. Type: `all` OR `React.Key[]`. Default: `NA`.
- **onSelectionChange**. Type: `(keys: "all" OR Set<React.Key>) => any`. Default: `NA`.
- **children**. Type: `ReactNode`. Default: `NA`.
- **title**. Type: `ReactNode`. Default: `NA`.
- **subtitle**. Type: `ReactNode`. Default: `NA`.
- **indicator**. Type: `IndicatorProps`. Default: `NA`.
- **startContent**. Type: `ReactNode`. Default: `NA`.
- **motionProps**. Type: `MotionProps`. Default: `NA`.
- **isCompact**. Type: `boolean`. Default: `false`.
- **isDisabled**. Type: `boolean`. Default: `false`.
- **keepContentMounted**. Type: `boolean`. Default: `false`.
- **hideIndicator**. Type: `boolean`. Default: `false`.
- **disableAnimation**. Type: `boolean`. Default: `false`.
- **disableIndicatorAnimation**. Type: `boolean`. Default: `false`.
- **HeadingComponent**. Type: `React.ElementType`. Default: `"h2"`.
- **classNames**. Type: `AccordionItemClassnames`. Default: `NA`.
- **onFocus**. Type: `(e: FocusEvent) => void`. Default: `NA`.
- **onBlur**. Type: `(e: FocusEvent) => void`. Default: `NA`.
- **onFocusChange**. Type: `(isFocused: boolean) => void`. Default: `NA`.
- **onKeyDown**. Type: `(e: KeyboardEvent) => void`. Default: `NA`.
- **onKeyUp**. Type: `(e: KeyboardEvent) => void`. Default: `NA`.
- **onPress**. Type: `(e: PressEvent) => void`. Default: `NA`.
- **onPressStart**. Type: `(e: PressEvent) => void`. Default: `NA`.
- **onPressEnd**. Type: `(e: PressEvent) => void`. Default: `NA`.
- **onPressChange**. Type: `(isPressed: boolean) => void`. Default: `NA`.
- **onPressUp**. Type: `(e: PressEvent) => void`. Default: `NA`.
- **onClick**. Type: `MouseEventHandler`. Default: `NA`.

Value

An object of class `shiny.tag` containing the necessary configuration and including options such as JavaScript dependencies to instantiate a HeroUI accordion component.

See Also

See https://heroui.com/docs/components/accordion.

Examples

```
library(shiny)
library(shinyNextUI)
library(shiny.react)

ui <- nextui_page(
  debug_react = TRUE,
  p(class = "font-extrabold text-2xl uppercase", "accordion()"),
  accordion(
    inputId = "accordion1",
    value = "2",
    accordion_item(
      startContent = avatar(
        isBordered = TRUE,
        color = "primary",
        radius = "lg",
        src = "https://i.pravatar.cc/150?u=a042581f4e29026024d"
      ),
      onPress = JS("(e) => alert('Your pressed me')"),
      "plop",
      title = "Element 1",
      key = "1",
      subtitle = "subtitle"
    ),
    accordion_item("plop", title = "Element 2", key = "2")
  ),
  spacer(y = 2),
  p(class = "font-extrabold text-2xl uppercase", "shadow variant accordion()"),
  accordion(
    inputId = "accordion2",
    isCompact = TRUE,
    variant = "shadow",
    value = "3",
    accordion_item("plop", title = "Element 1", key = "1", subtitle = "subtitle"),
    accordion_item("plop", title = "Element 2", key = "2"),
    accordion_item("plop", title = "Element 3", key = "3")
  ),
  spacer(y = 2),
  p(class = "font-extrabold text-2xl uppercase", "bordered variant accordion()"),
  accordion(
    inputId = "accordion3",
    isCompact = TRUE,
    variant = "bordered",
    accordion_item("plop", title = "Element 1", key = "1", subtitle = "subtitle"),
    accordion_item("plop", title = "Element 2", key = "2")
  ),
  spacer(y = 2),
  p(class = "font-extrabold text-2xl uppercase", "splitted variant accordion()"),
  accordion(
    inputId = "accordion4",
    isCompact = TRUE,
    variant = "splitted",
```

```
  motionProps= JS(
    '{
    variants: {
      enter: {
        y: 0,
        opacity: 1,
        height: "auto",
        transition: {
          height: {
            type: "spring",
            stiffness: 500,
            damping: 30,
            duration: 1,
          },
          opacity: {
            easings: "ease",
            duration: 1,
          },
        },
      },
      exit: {
        y: -10,
        opacity: 0,
        height: 0,
        transition: {
          height: {
            easings: "ease",
            duration: 0.25,
          },
          opacity: {
            easings: "ease",
            duration: 0.3,
          },
        },
      },
    },
  }'),
  value = "2",
  accordion_item("plop", title = "Element 1", key = "1", subtitle = "subtitle"),
  accordion_item("plop", title = "Element 2", key = "2")
),
spacer(y = 2),
p(class = "font-extrabold text-2xl uppercase", "Update accordion"),
select_input(
  "select",
  label = "Accordion to open:",
  select_item(key = 1, value = "1", "1"),
  select_item(key = 2, value = "2", "2")
),
spacer(y = 2),
accordion(
  inputId = "accordion5",
  isCompact = TRUE,
```

```
    variant = "bordered",
    accordion_item("plop1", title = "Element 1", key = "1", subtitle = "subtitle"),
    accordion_item("plop2", title = "Element 2", key = "2")
  )
)

server <- function(input, output, session) {
  observeEvent(input$select, {
    update_accordion(
      session,
      "accordion5",
      value = input$select
    )
  })
  observe(
    print(input$accordion1)
  )
}

if (interactive() || is_testing()) shinyApp(ui, server)
```

`actionButton` *Action button*

Description

This is a higher level wrapper of action_button to match vanilla's Shiny syntax and parameters.

This is a higher level wrapper of update_action_button to match vanilla's Shiny syntax and parameters.

Buttons allow users to perform actions and choose with a single tap.

Usage

```
actionButton(inputId, label, icon = NULL, width = NULL, ...)

updateActionButton(
  session = getDefaultReactiveDomain(),
  inputId,
  label = NULL,
  icon = NULL
)

button(...)

action_button(inputId, ..., value = default_value)

update_action_button(session = shiny::getDefaultReactiveDomain(), inputId, ...)
```

Arguments

inputId	The input slot that will be used to access the value.
label	The contents of the button or link–usually a text label, but you could also use any other HTML, like an image.
icon	An optional icon() to appear on the button.
width	Not used with NextUI but left for compatibility.
...	Named attributes to be applied to the button or link.
session	The session object passed to function given to shinyServer. Default is getDefaultReactiveDomain()
value	Starting value.

Details

- **children**. Type: ReactNode. Default: NA.
- **variant**. Type: solid OR bordered OR light OR flat OR faded OR shadow OR ghost. Default: "solid".
- **color**. Type: default OR primary OR secondary OR success OR warning OR danger. Default: "default".
- **size**. Type: sm OR md OR lg. Default: "md".
- **radius**. Type: none OR sm OR md OR lg OR full. Default: NA.
- **startContent**. Type: ReactNode. Default: NA.
- **endContent**. Type: ReactNode. Default: NA.
- **spinner**. Type: ReactNode. Default: NA.
- **spinnerPlacement**. Type: start OR end. Default: "start".
- **fullWidth**. Type: boolean. Default: false.
- **isIconOnly**. Type: boolean. Default: false.
- **isDisabled**. Type: boolean. Default: false.
- **isLoading**. Type: boolean. Default: false.
- **disableRipple**. Type: boolean. Default: false.
- **disableAnimation**. Type: boolean. Default: false.
- **onPress**. Type: (e: PressEvent) => void. Default: NA.
- **onPressStart**. Type: (e: PressEvent) => void. Default: NA.
- **onPressEnd**. Type: (e: PressEvent) => void. Default: NA.
- **onPressChange**. Type: (isPressed: boolean) => void. Default: NA.
- **onPressUp**. Type: (e: PressEvent) => void. Default: NA.
- **onKeyDown**. Type: (e: KeyboardEvent) => void. Default: NA.
- **onKeyUp**. Type: (e: KeyboardEvent) => void. Default: NA.
- **onClick**. Type: MouseEventHandler. Default: NA.
- **children**. Type: ReactNode OR ReactNode[]. Default: NA.

- **variant**. Type: `solid` OR `bordered` OR `light` OR `flat` OR `faded` OR `shadow` OR `ghost`. Default: `"solid"`.
- **color**. Type: `default` OR `primary` OR `secondary` OR `success` OR `warning` OR `danger`. Default: `"default"`.
- **size**. Type: `sm` OR `md` OR `lg`. Default: `"md"`.
- **radius**. Type: `none` OR `sm` OR `md` OR `lg` OR `full`. Default: `"xl"`.
- **fullWidth**. Type: `boolean`. Default: `false`.
- **isDisabled**. Type: `boolean`. Default: `false`.

Value

Object with `shiny.tag` class suitable for use in the UI of a Shiny app. The update functions return nothing (called for side effects).

An object of class `shiny.tag` containing the necessary configuration and including options such as JavaScript dependencies to instantiate a HeroUI button component.

See Also

See https://heroui.com/docs/components/button and action_button to get the list of possible parameters.

See https://heroui.com/docs/components/button.

Examples

```
library(shiny)
library(shinyNextUI)
library(shiny.react)

ui <- nextui_page(
  reactOutput("button")
)

server <- function(input, output, session) {
  n_click <- reactiveVal(0)
  observeEvent(input$clicked, {
    n_click(n_click() + 1)
  })

  output$button <- renderReact({
    action_button(
      inputId = "clicked",
      color = "primary",
      shadow = TRUE,
      sprintf(
        "Test Button. You clicked: %s times.",
        n_click()
      )
    )
  })
```

```
  exportTestValues(n_click = n_click())
}

if (interactive() || is_testing()) shinyApp(ui, server)
library(shiny)
library(shinyNextUI)
library(shiny.react)

ui <- nextui_page(
  reactOutput("button")
)

server <- function(input, output, session) {
  n_click <- reactiveVal(0)
  observeEvent(input$clicked, {
    n_click(n_click() + 1)
  })

  output$button <- renderReact({
    action_button(
      inputId = "clicked",
      color = "primary",
      shadow = TRUE,
      sprintf(
        "Test Button. You clicked: %s times.",
        n_click()
      )
    )
  })

  exportTestValues(n_click = n_click())
}

if (interactive() || is_testing()) shinyApp(ui, server)
library(shiny)
library(shinyNextUI)
library(shiny.react)

ui <- nextui_page(
  reactOutput("button")
)

server <- function(input, output, session) {
  n_click <- reactiveVal(0)
  observeEvent(input$clicked, {
    n_click(n_click() + 1)
  })

  output$button <- renderReact({
    action_button(
      inputId = "clicked",
      color = "primary",
```

```
      shadow = TRUE,
      sprintf(
        "Test Button. You clicked: %s times.",
        n_click()
      )
    )
  })

  exportTestValues(n_click = n_click())
}

if (interactive() || is_testing()) shinyApp(ui, server)
```

`autocomplete` *autocomplete*

Description

An autocomplete combines a text input with a listbox, allowing users to filter a list of options to items matching a query.

Usage

```
autocomplete(inputId, ..., value = default_value)

autocomplete_section(...)

autocomplete_item(...)

update_autocomplete(session = shiny::getDefaultReactiveDomain(), inputId, ...)
```

Arguments

`inputId`	ID of the component.
`...`	Props to pass to the component. The allowed props are listed below in the **Details** section.
`value`	Starting value.
`session`	Object passed as the `session` argument to Shiny server.

Details

1. Autocomplete Props
   - **children***. Type: `ReactNode[]`. Default: `NA`.
   - **label**. Type: `ReactNode`. Default: `NA`.
   - **name**. Type: `string`. Default: `NA`.
   - **variant**. Type: `flat` `OR` `bordered` `OR` `faded` `OR` `underlined`. Default: `"flat"`.

- **color**. Type: `default` OR `primary` OR `secondary` OR `success` OR `warning` OR `danger`. Default: `"default"`.
- **size**. Type: `sm` OR `md` OR `lg`. Default: `"md"`.
- **radius**. Type: `none` OR `sm` OR `md` OR `lg` OR `full`. Default: `NA`.
- **items**. Type: `Iterable<T>`. Default: `NA`.
- **defaultItems**. Type: `Iterable<T>`. Default: `NA`.
- **inputValue**. Type: `string`. Default: `NA`.
- **defaultInputValue**. Type: `string`. Default: `NA`.
- **allowsCustomValue**. Type: `boolean`. Default: `false`.
- **allowsEmptyCollection**. Type: `boolean`. Default: `true`.
- **shouldCloseOnBlur**. Type: `boolean`. Default: `true`.
- **placeholder**. Type: `string`. Default: `NA`.
- **description**. Type: `ReactNode`. Default: `NA`.
- **menuTrigger**. Type: `focus` OR `input` OR `manual`. Default: `"focus"`.
- **labelPlacement**. Type: `inside` OR `outside` OR `outside-left`. Default: `"inside"`.
- **selectedKey**. Type: `React.Key`. Default: `NA`.
- **defaultSelectedKey**. Type: `React.Key`. Default: `NA`.
- **disabledKeys**. Type: `all` OR `React.Key[]`. Default: `NA`.
- **errorMessage**. Type: `ReactNode` OR `((v: ValidationResult) => ReactNode)`. Default: `NA`.
- **validate**. Type: `(value: { inputValue: string, selectedKey: React.Key }) => ValidationError` OR `true` OR `null` OR `undefined`. Default: `NA`.
- **validationBehavior**. Type: `native` OR `aria`. Default: `"native"`.
- **startContent**. Type: `ReactNode`. Default: `NA`.
- **endContent**. Type: `ReactNode`. Default: `NA`.
- **autoFocus**. Type: `boolean`. Default: `false`.
- **defaultFilter**. Type: `(textValue: string, inputValue: string) => boolean`. Default: `NA`.
- **filterOptions**. Type: `Intl.CollatorOptions`. Default: `"{ sensitivity: 'base'}"`.
- **maxListboxHeight**. Type: `number`. Default: `"256"`.
- **itemHeight**. Type: `number`. Default: `"32"`.
- **isVirtualized**. Type: `boolean`. Default: `"undefined"`.
- **isReadOnly**. Type: `boolean`. Default: `false`.
- **isRequired**. Type: `boolean`. Default: `false`.
- **isInvalid**. Type: `boolean`. Default: `false`.
- **isDisabled**. Type: `boolean`. Default: `false`.
- **fullWidth**. Type: `boolean`. Default: `true`.
- **selectorIcon**. Type: `ReactNode`. Default: `NA`.
- **clearIcon**. Type: `ReactNode`. Default: `NA`.
- **showScrollIndicators**. Type: `boolean`. Default: `true`.
- **scrollRef**. Type: `React.RefObject<HTMLElement>`. Default: `NA`.
- **inputProps**. Type: `InputProps`. Default: `NA`.
- **popoverProps**. Type: `PopoverProps`. Default: `NA`.

- **listboxProps**. Type: `ListboxProps`. Default: `NA`.
- **scrollShadowProps**. Type: `ScrollShadowProps`. Default: `NA`.
- **selectorButtonProps**. Type: `ButtonProps`. Default: `NA`.
- **clearButtonProps**. Type: `ButtonProps`. Default: `NA`.
- **isClearable**. Type: `boolean`. Default: `true`.
- **disableClearable**. Type: `boolean`. Default: `false`.
- **disableAnimation**. Type: `boolean`. Default: `true`.
- **disableSelectorIconRotation**. Type: `boolean`. Default: `false`.
- **classNames**. Type: `Partial<Record<'base' OR 'listboxWrapper' OR 'listbox' OR 'popoverContent' OR 'endContentWrapper' OR 'clearButton' OR 'selectorButton', string>>`. Default: `NA`.

2. Autocomplete Events

- **onOpenChange**. Type: `(isOpen: boolean, menuTrigger?: MenuTriggerAction) => void`. Default: `NA`.
- **onInputChange**. Type: `(value: string) => void`. Default: `NA`.
- **onSelectionChange**. Type: `(key: React.Key) => void`. Default: `NA`.
- **onFocus**. Type: `(e: FocusEvent<HTMLInputElement>) => void`. Default: `NA`.
- **onBlur**. Type: `(e: FocusEvent<HTMLInputElement>) => void`. Default: `NA`.
- **onFocusChange**. Type: `(isFocused: boolean) => void`. Default: `NA`.
- **onKeyDown**. Type: `(e: KeyboardEvent) => void`. Default: `NA`.
- **onKeyUp**. Type: `(e: KeyboardEvent) => void`. Default: `NA`.
- **onClose**. Type: `() => void`. Default: `NA`.

Value

An object of class `shiny.tag` containing the necessary configuration and including options such as JavaScript dependencies to instantiate a HeroUI autocomplete component.

See Also

See https://heroui.com/docs/components/autocomplete.

Examples

```
library(shiny)
library(shinyNextUI)
library(shiny.react)

animals <- list(
  list(
    label = "Bulbasaur",
    value = "bulbasaur",
    description = "Blabla",
   avatar = "https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/1.png"
  ),
  list(
    label = "Pikachu",
```

```
    value = "pikachu",
    description = "Electric mouse",
   avatar = "https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/25.png"
  )
)

items <- lapply(animals, function(animal) {
  autocomplete_item(
    key = animal[["value"]],
    value = animal[["value"]],
    startContent = avatar(src = animal[["avatar"]]),
    animal[["label"]]
  )
})

ui <- nextui_page(
  debug_react = TRUE,
  action_button("update", "Update to Pikachu?"),
  spacer(y = 4),
  autocomplete(
    "autocomplete",
    label = "Select a pokemon",
    isRequired = TRUE,
    value = "bulbasaur",
    autocomplete_section(
      title = "Default pokemons",
      items
    )
  ),
  textOutput("res")
)

server <- function(input, output, session) {
  output$res <- renderText(input$autocomplete)
  observeEvent(input$autocomplete, {
    print(input$autocomplete)
  })

  observeEvent(input$update, {
    update_autocomplete(session, "autocomplete", value = "pikachu")
  })
}

if (interactive() || is_testing()) shinyApp(ui, server)
```

`avatar` *avatar*

Description

The Avatar component is used to represent a user, and displays the profile picture, initials or fallback icon.

Usage

```
avatar(...)

avatar_group(...)
```

Arguments

...	Props to pass to the component. The allowed props are listed below in the **Details** section.

Details

- **src**. Type: `string`. Default: `NA`.
- **color**. Type: `default` OR `primary` OR `secondary` OR `success` OR `warning` OR `danger`. Default: `"default"`.
- **radius**. Type: `none` OR `sm` OR `md` OR `lg` OR `full`. Default: `"full"`.
- **size**. Type: `sm` OR `md` OR `lg`. Default: `"md"`.
- **name**. Type: `string`. Default: `NA`.
- **icon**. Type: `ReactNode`. Default: `NA`.
- **fallback**. Type: `ReactNode`. Default: `NA`.
- **isBordered**. Type: `boolean`. Default: `false`.
- **isDisabled**. Type: `boolean`. Default: `false`.
- **isFocusable**. Type: `boolean`. Default: `false`.
- **showFallback**. Type: `boolean`. Default: `false`.
- **ImgComponent**. Type: `React.ElementType`. Default: `"img"`.
- **imgProps**. Type: `ImgComponentProps`. Default: `NA`.
- **classNames**. Type: `Partial<Record<"base" OR "img" OR "fallback" OR "name" OR "icon", string>>`. Default: `NA`.
- **max**. Type: `number`. Default: `"5"`.
- **total**. Type: `number`. Default: `NA`.
- **size**. Type: `AvatarProps['size']`. Default: `NA`.
- **color**. Type: `AvatarProps['color']`. Default: `NA`.
- **radius**. Type: `AvatarProps['radius']`. Default: `NA`.
- **isGrid**. Type: `boolean`. Default: `false`.
- **isDisabled**. Type: `boolean`. Default: `NA`.
- **isBordered**. Type: `boolean`. Default: `NA`.
- **renderCount**. Type: `(count: number) => ReactNode`. Default: `NA`.
- **classNames**. Type: `Partial<Record<"base" OR "count", string>>`. Default: `NA`.

Value

An object of class `shiny.tag` containing the necessary configuration and including options such as JavaScript dependencies to instantiate a HeroUI avatar component.

See Also

See https://heroui.com/docs/components/avatar.

Examples

```
library(shiny)
library(shinyNextUI)

avatar_config <- data.frame(
  size = c("xs", "sm", "md", "lg", "xl"),
  src = c(
    "https://i.pravatar.cc/150?u=a042581f4e29026024d",
    "https://i.pravatar.cc/150?u=a042581f4e29026704d",
    "https://i.pravatar.cc/150?u=a04258114e29026702d",
    "https://i.pravatar.cc/150?u=a048581f4e29026701d",
    "https://i.pravatar.cc/150?u=a092581d4ef9026700d"
  ),
  radius = c(rep("full", 2), "lg", "md", "sm"),
  disabled = c(rep(FALSE, 4), TRUE),
  bordered = c(rep(TRUE, 3), rep(FALSE, 2)),
  color = c(
    "primary",
    "secondary",
    "danger",
    "success",
    "warning"
  ),
  fallback = rep(TRUE, 5)
)

avatar_factory <- function(src, size, disabled, bordered, radius, color, fallback) {
  avatar(
    src = src,
    size = size,
    isDisabled = disabled,
    isBordered = bordered,
    radius = radius,
    color = color,
    showFallback = fallback
  )
}

avatars <- purrr::pmap(avatar_config, avatar_factory)

ui <- nextui_page(
  debug_react = TRUE,
  class = "container mx-auto px-4",
```

```
  p("avatar()"),
  spacer(y = 1),
  div(
    class = "flex gap-3 items-center",
    avatars
  ),
  spacer(y = 2),
  p("avatar_group()"),
  spacer(y = 1),
  div(
    class = "flex",
    avatar_group(
      isBordered = TRUE,
      max = 3,
      total = 10,
      lapply(avatar_config$src, function(link) avatar(src = link))
    )
  )
)

server <- function(input, output, session) {}

if (interactive() || is_testing()) shinyApp(ui, server)
```

---

badge *badge*

Description

Badges are used as a small numerical value or status descriptor for UI elements.

Usage

```
badge(...)
```

Arguments

...	Props to pass to the component. The allowed props are listed below in the **Details** section.

Details

- **children**. Type: `ReactNode`. Default: `NA`.
- **content**. Type: `string` `OR` `number` `OR` `ReactNode`. Default: `NA`.
- **variant**. Type: `solid` `OR` `flat` `OR` `faded` `OR` `shadow`. Default: `"solid"`.
- **color**. Type: `default` `OR` `primary` `OR` `secondary` `OR` `success` `OR` `warning` `OR` `danger`. Default: `"default"`.
- **size**. Type: `sm` `OR` `md` `OR` `lg`. Default: `"md"`.

- **shape**. Type: `circle` OR `rectangle`. Default: `"rectangle"`.
- **placement**. Type: `top-right` OR `top-left` OR `bottom-right` OR `bottom-left`. Default: `"top-right"`.
- **showOutline**. Type: `boolean`. Default: `true`.
- **disableOutline**. Type: `boolean`. Default: `false`.
- **disableAnimation**. Type: `boolean`. Default: `false`.
- **isInvisible**. Type: `boolean`. Default: `false`.
- **isOneChar**. Type: `boolean`. Default: `false`.
- **isDot**. Type: `boolean`. Default: `false`.
- **classNames**. Type: `Partial<Record<"base"OR"badge", string>>`. Default: `NA`.

Value

An object of class `shiny.tag` containing the necessary configuration and including options such as JavaScript dependencies to instantiate a HeroUI badge component.

See Also

See https://heroui.com/docs/components/badge.

Examples

```
library(shiny)
library(shinyNextUI)

badge_config <- data.frame(
  size = c(rep("sm", 2), rep("md", 2), rep("lg", 2)),
  color = c(
    "default",
    "primary",
    "secondary",
    "success",
    "warning",
    "danger"
  ),
  disable_outline = c(rep(FALSE, 3), rep(TRUE, 3))
)

variants <- c("solid", "flat", "faded", "shadow")
placement <- c("top-right", "bottom-right", "top-left", "bottom-left")
shape <- c("rectangle", "circle")

badge_factory <- function(size, color, disable_outline) {
  badge(
    class = "mx-5",
    size = size,
    color = color,
    disableOutline = disable_outline,
    content = 1,
```

```
    avatar()
  )
}

badges <- purrr::pmap(badge_config, badge_factory)

ui <- nextui_page(
  p(class = "font-extrabold text-2xl uppercase", "Badges"),
  div(
    class = "flex flex-row",
    badges
  ),
  spacer(y = 2),
  p(class = "font-extrabold text-2xl uppercase", "Badge content"),
  div(
    class = "flex flex-row",
    badge(
      color = "danger",
      content = "New",
      placement = "top-left",
      avatar(
        bordered = TRUE,
        squared = TRUE,
        color = "secondary",
        size = "lg",
        src = "https://i.pravatar.cc/300?u=a042581f4e29026707d"
      )
    )
  ),
  spacer(y = 2),
  p(class = "font-extrabold text-2xl uppercase", "Solid variant"),
  div(
    class = "flex flex-row",
    badge(variant = "solid", color = "success", size = "lg", content = 5, avatar())
  ),
  spacer(y = 2),
  p(class = "font-extrabold text-2xl uppercase", "Flat variant"),
  div(
    class = "flex flex-row",
    badge(variant = "flat", color = "success", size = "lg", content = 5, avatar())
  ),
  spacer(y = 2),
  p(class = "font-extrabold text-2xl uppercase", "Faded variant"),
  div(
    class = "flex flex-row",
    badge(variant = "faded", color = "success", size = "lg", content = 5, avatar())
  ),
  spacer(y = 2),
  p(class = "font-extrabold text-2xl uppercase", "Shadow variant"),
  div(
    class = "flex flex-row",
    badge(variant = "shadow", color = "success", size = "lg", content = 5, avatar())
  )
```

```
)

server <- function(input, output, session) {}

if (interactive() || is_testing()) shinyApp(ui, server)
```

`card` *card*

Description

Card is a container for text, photos, and actions in the context of a single subject.

Usage

```
card(...)

card_body(...)

card_header(...)

card_footer(...)
```

Arguments

`...`	Props to pass to the component. The allowed props are listed below in the **Details** section.

Details

- **children**. Type: `ReactNode OR ReactNode[]`. Default: `NA`.
- **shadow**. Type: `none OR sm OR md OR lg`. Default: `"md"`.
- **radius**. Type: `none OR sm OR md OR lg`. Default: `"lg"`.
- **fullWidth**. Type: `boolean`. Default: `false`.
- **isHoverable**. Type: `boolean`. Default: `false`.
- **isPressable**. Type: `boolean`. Default: `false`.
- **isBlurred**. Type: `boolean`. Default: `false`.
- **isFooterBlurred**. Type: `boolean`. Default: `false`.
- **isDisabled**. Type: `boolean`. Default: `false`.
- **disableAnimation**. Type: `boolean`. Default: `false`.
- **disableRipple**. Type: `boolean`. Default: `false`.
- **allowTextSelectionOnPress**. Type: `boolean`. Default: `false`.
- **classNames**. Type: `Partial<Record<'base' OR 'header' OR 'body' OR 'footer', string>>`. Default: `NA`.

- **onPress**. Type: `(e: PressEvent) => void`. Default: `NA`.
- **onPressStart**. Type: `(e: PressEvent) => void`. Default: `NA`.
- **onPressEnd**. Type: `(e: PressEvent) => void`. Default: `NA`.
- **onPressChange**. Type: `(isPressed: boolean) => void`. Default: `NA`.
- **onPressUp**. Type: `(e: PressEvent) => void`. Default: `NA`.

Value

An object of class `shiny.tag` containing the necessary configuration and including options such as JavaScript dependencies to instantiate a HeroUI card component.

See Also

See https://heroui.com/docs/components/card.

Examples

```
library(shiny)
library(shinyNextUI)
library(shiny.react)

ui <- nextui_page(
  div(
    class = "grid gap-4 grid-cols-3 grid-rows-3 m-5",
    card(card_body("Simple card without anything")),
    card(
      variant = "bordered",
      card_header("Card title"),
      divider(),
      card_body(h1("Card body")),
      divider(),
      card_footer("Card Footer")
    ),
    card(
      #isBlurred = TRUE,
      isPressable = TRUE,
      onPress = JS("() => alert('You pressed me')"),
      shadow = "sm",
      className = "border-none bg-background/60 dark:bg-default-100/50 max-w-[610px]",
      card_body("Press me!")
    )
  )
)

server <- function(input, output, session) {

}

if (interactive() || is_testing()) shinyApp(ui, server)
```

checkbox *checkbox*

Description

Checkboxes allow users to select multiple items from a list of individual items, or to mark one individual item as selected.

Usage

```
checkbox_input(inputId, ..., value = default_value)

update_checkbox_input(
  session = shiny::getDefaultReactiveDomain(),
  inputId,
  ...
)
```

Arguments

inputId	ID of the component.
...	Props to pass to the component. The allowed props are listed below in the **Details** section.
value	Starting value.
session	Object passed as the session argument to Shiny server.

Details

- **children**. Type: ReactNode. Default: NA.
- **icon**. Type: CheckboxIconProps. Default: NA.
- **value**. Type: string. Default: NA.
- **name**. Type: string. Default: NA.
- **size**. Type: sm OR md OR lg. Default: "md".
- **color**. Type: default OR primary OR secondary OR success OR warning OR danger. Default: "primary".
- **radius**. Type: none OR sm OR md OR lg OR full. Default: NA.
- **lineThrough**. Type: boolean. Default: false.
- **isSelected**. Type: boolean. Default: NA.
- **defaultSelected**. Type: boolean. Default: NA.
- **isRequired**. Type: boolean. Default: false.
- **isReadOnly**. Type: boolean. Default: NA.
- **isDisabled**. Type: boolean. Default: false.

- **isIndeterminate**. Type: `boolean`. Default: `NA`.
- **isInvalid**. Type: `boolean`. Default: `false`.
- **validationState**. Type: `valid` `OR` `invalid`. Default: `NA`.
- **disableAnimation**. Type: `boolean`. Default: `false`.
- **classNames**. Type: `Partial<Record<"base"OR "wrapper"OR "icon"OR "label", string>>`. Default: `NA`.
- **onChange**. Type: `React.ChangeEvent<HTMLInputElement>`. Default: `NA`.
- **onValueChange**. Type: `(isSelected: boolean) => void`. Default: `NA`.

Value

An object of class `shiny.tag` containing the necessary configuration and including options such as JavaScript dependencies to instantiate a HeroUI checkbox component.

See Also

See https://heroui.com/docs/components/checkbox.

Examples

```
library(shiny)
library(shinyNextUI)

ui <- nextui_page(
  div(
    class = "flex gap-1",
    action_button("update", "Toggle checkbox"),
    spacer(y = 2),
    checkbox_input(
      inputId = "checkbox",
      value = TRUE,
      "My checkbox",
      isRounded = TRUE,
      color = "warning",
      lineThrough = TRUE
    ),
    textOutput("check_val")
  )
)

server <- function(input, output, session) {
  output$check_val <- renderText(input$checkbox)
  observeEvent(input$update, {
    update_checkbox_input(session, "checkbox", value = !input$checkbox)
  })
}

if (interactive() || is_testing()) shinyApp(ui, server)
```

checkboxgroup_input *Checkbox group input*

Description

Checkbox group input

Usage

```
checkboxgroup_input(inputId, ..., choices, selected = NULL)

update_checkboxgroup_input(
  session = shiny::getDefaultReactiveDomain(),
  inputId,
  ...,
  choices = NULL,
  selected = NULL
)
```

Arguments

inputId	Unique input id.
...	Props.
choices	Radio choices.
selected	Default selected value.
session	Shiny session.

Details

See https://heroui.com/docs/components/checkbox-group to get the list of parameters to pass in ....

Value

Object with shiny.tag class suitable for use in the UI of a Shiny app. The update functions return nothing (called for side effects).

See Also

See https://heroui.com/docs/components/checkbox-group.

Examples

```
library(shiny)
library(shinyNextUI)
library(shiny.react)

ui <- nextui_page(
  debug_react = TRUE,
  div(
    class = "flex flex-col gap-1",
    spacer(y = 2),
    select_input(
      "select",
      label = "Tab to select:",
      value = JS("['sydney']"),
      selectionMode = "multiple",
      select_item(key = "buenos-aires", value = "buenos-aires", "Buenos Aires"),
      select_item(key = "sydney", value = "sydney", "Sydney")
    ),
    checkboxgroup_input(
      inputId = "checkbox_group",
      label = "Checkbox Group",
      choices = c(
        "buenos-aires" = "Buenos Aires",
        "sydney" = "Sydney"
      ),
      orientation = "horizontal",
      color = "secondary"
    ),
    textOutput("checkbox_group_val")
  )
)

server <- function(input, output, session) {
  observeEvent(input$select, {
    update_checkboxgroup_input(session, "checkbox_group", selected = input$select)
  }, ignoreNULL = FALSE)
  output$checkbox_group_val <- renderText(input$checkbox_group)
}

if (interactive() || is_testing()) shinyApp(ui, server)
```

---

chip *chip*

Description

A Chip is a small block of essential information that represent an input, attribute, or action.

Usage

```
chip(...)
```

Arguments

...	Props to pass to the component. The allowed props are listed below in the **Details** section.

Details

- **children**. Type: `ReactNode`. Default: `NA`.
- **variant**. Type: `solid` OR `bordered` OR `light` OR `flat` OR `faded` OR `shadow` OR `dot`. Default: `"solid"`.
- **color**. Type: `default` OR `primary` OR `secondary` OR `success` OR `warning` OR `danger`. Default: `"default"`.
- **size**. Type: `sm` OR `md` OR `lg`. Default: `"md"`.
- **radius**. Type: `none` OR `sm` OR `md` OR `lg` OR `full`. Default: `"full"`.
- **avatar**. Type: `ReactNode`. Default: `NA`.
- **startContent**. Type: `ReactNode`. Default: `NA`.
- **endContent**. Type: `ReactNode`. Default: `NA`.
- **isDisabled**. Type: `boolean`. Default: `false`.
- **classNames**. Type: `Partial<Record<"base" OR "content" OR "dot" OR "avatar" OR "closeButton", string>>`. Default: `NA`.
- **onClose**. Type: `(e: PressEvent) => void`. Default: `NA`.

Value

An object of class `shiny.tag` containing the necessary configuration and including options such as JavaScript dependencies to instantiate a HeroUI chip component.

See Also

See https://heroui.com/docs/components/chip.

Examples

```
library(shiny)
library(shinyNextUI)
library(shiny.react)

ui <- nextui_page(
  div(
    class = "flex gap-2 my-2",
    chip(
      id = "plop",
      "A chip",
      onClose = JS("(e) => {
        var chipEl = e.target.offsetParent;
        Shiny.setInputValue(chipEl.id, false);
        $(chipEl).remove();
```

```
        }")
      ),
      chip(
        "A chip",
        color = "success",
        size = "lg",
        radius = "sm",
        variant = "flat",
        startContent = icon("home"),
        endContent = icon("bell")
      ),
      chip(
      avatar = avatar(name = "JW", src = "https://i.pravatar.cc/300?u=a042581f4e29026709d"),
        "hello"
      ),
      reactOutput("modal")
    )
  )

  server <- function(input, output, session) {

    modalVisible <- reactiveVal(FALSE)
    observeEvent({
      input$plop
    }, {
      if (!input$plop) modalVisible(TRUE)
    })

    observeEvent(input$modal_closed, {
      modalVisible(FALSE)
    })

    output$modal <- renderReact({
      modal(
        scrollBehavior = input$scroll,
        isOpen = modalVisible(),
        size = "sm",
        placement = "top",
        modal_content(
          modal_header("Congrats"),
          modal_body(
            p("You closed me!")
          )
        ),
       onClose = JS("() => Shiny.setInputValue('modal_closed', true, {priority: 'event'})")
      )
    })
  }

  if (interactive() || is_testing()) shinyApp(ui, server)
```

`circular_progress` *circular-progress*

Description

Circular progress indicators are utilized to indicate an undetermined wait period or visually represent the duration of a process.

Usage

```
circular_progress(...)
```

Arguments

`...`	Props to pass to the component. The allowed props are listed below in the **Details** section.

Details

- **label**. Type: `ReactNode`. Default: `NA`.
- **size**. Type: `sm OR md OR lg`. Default: `"md"`.
- **color**. Type: `default OR primary OR secondary OR success OR warning OR danger`. Default: `"primary"`.
- **value**. Type: `number`. Default: `NA`.
- **valueLabel**. Type: `ReactNode`. Default: `NA`.
- **minValue**. Type: `number`. Default: `"0"`.
- **maxValue**. Type: `number`. Default: `"100"`.
- **formatOptions**. Type: `Intl.NumberFormat`. Default: `"{style: 'percent'}"`.
- **isIndeterminate**. Type: `boolean`. Default: `true`.
- **showValueLabel**. Type: `boolean`. Default: `true`.
- **strokeWidth**. Type: `number`. Default: `"2"`.
- **isDisabled**. Type: `boolean`. Default: `false`.
- **disableAnimation**. Type: `boolean`. Default: `false`.
- **classNames**. Type: `Partial<Record<'base'OR'svgWrapper'OR'svg'OR'track'OR'indicator'OR'value'OR'lab string>>`. Default: `NA`.

Value

An object of class `shiny.tag` containing the necessary configuration and including options such as JavaScript dependencies to instantiate a HeroUI circular-progress component.

See Also

See https://heroui.com/docs/components/circular-progress.

Examples

```
library(shiny)
library(shinyNextUI)

ui <- nextui_page(
  dark_mode = TRUE,
  div(
    class = "flex gap-4",
    card(
      class = "",
      card_body(
        class = "grid grid-cols-2 gap-4",
        circular_progress(
          value = 3,
          showValueLabel = TRUE,
          strokeWidth = 4,
          size = "lg",
          minValue = 0,
          maxValue = 150,
          valueLabel = div(icon("battery-full", class = "mx-1"), "2%"),
          color = "danger"
        ),
        circular_progress(
          value = 120,
          showValueLabel = TRUE,
          strokeWidth = 4,
          size = "lg",
          minValue = 0,
          maxValue = 150,
          valueLabel = div(icon("mobile-screen-button", class = "mx-1"), "80%"),
          color = "success"
        )
      )
    )
  )
)

server <- function(input, output, session) {

}

if (interactive() || is_testing()) shinyApp(ui, server)
```

---

code_block *code*

Description

Code is a component used to display inline code.

Usage

```
code_block(...)
```

Arguments

...	Props to pass to the component. The allowed props are listed below in the **Details** section.

Details

- **children**. Type: `ReactNode`. Default: `NA`.
- **size**. Type: `sm` OR `md` OR `lg`. Default: `"sm"`.
- **color**. Type: `default` OR `primary` OR `secondary` OR `success` OR `warning` OR `danger`. Default: `"default"`.
- **radius**. Type: `none` OR `sm` OR `md` OR `lg` OR `full`. Default: `"sm"`.

Value

An object of class `shiny.tag` containing the necessary configuration and including options such as JavaScript dependencies to instantiate a HeroUI code component.

See Also

See https://heroui.com/docs/components/code.

Examples

```
library(shiny)
library(shinyNextUI)
library(shiny.react)

ui <- nextui_page(
  div(
    class = "flex gap-2 my-2",
    code_block(
      size = "lg",
      color = "secondary",
      radius = "full",
      "npm install @nextui-org/react"
    )
  )
)

server <- function(input, output, session) {

}

if (interactive() || is_testing()) shinyApp(ui, server)
```

createReactShinyInput *Create a reactR shiny input element*

Description

This is used to create custom react element for R. Specifically for radio and checkboxgroup which don't work with shiny.react.

Usage

```
createReactShinyInput(
  inputId,
  class,
  default = NULL,
  configuration = list(),
  container = htmltools::tags$div,
  dependencies = NULL
)
```

Arguments

inputId	Unique input id.
class	Element class. Must match the JavaScript class counterpart.
default	Default value.
configuration	Props.
container	Default container.
dependencies	Deps.

Value

A list of tags.

divider *divider*

Description

Divider is a component that separates content in a page.

Usage

```
divider(...)
```

Arguments

...	Props to pass to the component. The allowed props are listed below in the **Details** section.

Details

- **orientation**. Type: `` `horizontal` OR `vertical` ``. Default: `` "`horizontal`" ``.

Value

An object of class `shiny.tag` containing the necessary configuration and including options such as JavaScript dependencies to instantiate a HeroUI divider component.

See Also

See https://heroui.com/docs/components/divider.

Examples

```
library(shiny)
library(shinyNextUI)
library(shiny.react)

ui <- nextui_page(
  debug_react = TRUE,
  card(
    card_header("Header"),
    divider(),
    card_body("Body"),
    divider(),
    card_footer("Footer")
  )
)

server <- function(input, output, session) {

}

if (interactive() || is_testing()) shinyApp(ui, server)
```

`drawer` *drawer*

Description

Displays a panel that slides in from the edge of the screen, containing supplementary content.

Usage

```
drawer(...)

drawer_content(...)

drawer_header(...)

drawer_body(...)

drawer_footer(...)
```

Arguments

...	Props to pass to the component. The allowed props are listed below in the **Details** section.

Details

- **children**. Type: `ReactNode`. Default: `NA`.
- **size**. Type: `xs` OR `sm` OR `md` OR `lg` OR `xl` OR `2xl` OR `3xl` OR `4xl` OR `5xl` OR `full`. Default: `"md"`.
- **radius**. Type: `none` OR `sm` OR `md` OR `lg`. Default: `"lg"`.
- **placement**. Type: `left` OR `right` OR `top` OR `bottom`. Default: `"right"`.
- **isOpen**. Type: `boolean`. Default: `NA`.
- **defaultOpen**. Type: `boolean`. Default: `NA`.
- **isDismissable**. Type: `boolean`. Default: `true`.
- **isKeyboardDismissDisabled**. Type: `boolean`. Default: `false`.
- **shouldBlockScroll**. Type: `boolean`. Default: `true`.
- **hideCloseButton**. Type: `boolean`. Default: `false`.
- **closeButton**. Type: `ReactNode`. Default: `NA`.
- **motionProps**. Type: `MotionProps`. Default: `NA`.
- **portalContainer**. Type: `HTMLElement`. Default: `"document.body"`.
- **disableAnimation**. Type: `boolean`. Default: `false`.
- **classNames**. Type: `Partial<Record<'wrapper' OR 'base' OR 'backdrop' OR 'header' OR 'body' OR 'footer' OR 'closeButton', string>>`. Default: `NA`.
- **onOpenChange**. Type: `(isOpen: boolean) => void`. Default: `NA`.
- **onClose**. Type: `() => void`. Default: `NA`.

Value

An object of class `shiny.tag` containing the necessary configuration and including options such as JavaScript dependencies to instantiate a HeroUI drawer component.

See Also

See https://heroui.com/docs/components/drawer.

Examples

```
library(shiny)
library(shinyNextUI)
library(shiny.react)

ui <- nextui_page(
  action_button(
    inputId = "show_drawer",
    color = "primary",
    shadow = TRUE,
    "Show drawer"
  ),
  reactOutput("drawer")
)

server <- function(input, output, session) {
  drawerVisible <- reactiveVal(FALSE)
  observeEvent(input$show_drawer, {
    drawerVisible(TRUE)
  })

  observeEvent(input$hide_drawer, {
    drawerVisible(FALSE)
  })

  observeEvent(input$drawer_closed, {
    drawerVisible(FALSE)
  })

  output$drawer <- renderReact({
    drawer(
      scrollBehavior = input$scroll,
      isOpen = drawerVisible(),
      size = "sm",
      backdrop = "transparent",
      placement = "right",
      motionProps = JS(
        "{
          variants: {
            enter: {
              opacity: 1,
              x: 0,
              duration: 10,
            },
            exit: {
              x: 100,
              opacity: 0,
              duration: 10,
```

```
      },
    },
  }"
),
drawer_content(
  drawer_header("My drawer"),
  drawer_body(
    p(
      "Cras mattis consectetur purus sit amet fermentum. Cras justo odio,
      dapibus ac facilisis in, egestas eget quam. Morbi leo risus, porta
      ac consectetur ac, vestibulum at eros."
    ),
    p(
      "Lorem ipsum dolor sit amet, consectetur adipiscing elit.
      Nullam pulvinar risus non risus hendrerit venenatis.
      Pellentesque sit amet hendrerit risus, sed porttitor quam.
      Magna exercitation reprehenderit magna aute tempor cupidatat
      consequat elit dolor adipisicing. Mollit dolor eiusmod sunt ex
      incididunt cillum quis. Velit duis sit officia eiusmod Lorem
      aliqua enim laboris do dolor eiusmod. Et mollit incididunt
      nisi consectetur esse laborum eiusmod pariatur proident Lorem
      eiusmod et. Culpa deserunt nostrud ad veniam."
    ),
    p(
      "Mollit dolor eiusmod sunt ex incididunt cillum quis. Velit
       duis sit officia eiusmod Lorem aliqua enim laboris do dolor
       eiusmod. Et mollit incididunt nisi consectetur esse laborum
       eiusmod pariatur proident Lorem eiusmod et. Culpa deserunt
       nostrud ad veniam. Lorem ipsum dolor sit amet, consectetur
       adipiscing elit. Nullam pulvinar risus non risus hendrerit
       venenatis. Pellentesque sit amet hendrerit risus, sed
       porttitor quam. Magna exercitation reprehenderit magna aute
       tempor cupidatat consequat elit dolor adipisicing. Mollit
       dolor eiusmod sunt ex incididunt cillum quis. Velit duis sit
       officia eiusmod Lorem aliqua enim laboris do dolor eiusmod. Et
       mollit incididunt nisi consectetur esse laborum eiusmod
       pariatur proident Lorem eiusmod et. Culpa deserunt nostrud ad
       veniam."
    ),
    p(
      "Mollit dolor eiusmod sunt ex incididunt cillum quis. Velit
       duis sit officia eiusmod Lorem aliqua enim laboris do dolor
       eiusmod. Et mollit incididunt nisi consectetur esse laborum
       eiusmod pariatur proident Lorem eiusmod et. Culpa deserunt
       nostrud ad veniam. Lorem ipsum dolor sit amet, consectetur
       adipiscing elit. Nullam pulvinar risus non risus hendrerit
       venenatis. Pellentesque sit amet hendrerit risus, sed
       porttitor quam. Magna exercitation reprehenderit magna aute
       tempor cupidatat consequat elit dolor adipisicing. Mollit
       dolor eiusmod sunt ex incididunt cillum quis. Velit duis sit
       officia eiusmod Lorem aliqua enim laboris do dolor eiusmod. Et
       mollit incididunt nisi consectetur esse laborum eiusmod
       pariatur proident Lorem eiusmod et. Culpa deserunt nostrud ad
```

```
            veniam."
          )
        ),
        drawer_footer(
          action_button(
            inputId = "hide_drawer",
            color = "danger",
            shadow = TRUE,
            "Close drawer"
          )
        )
      ),
      onClose = JS(
        "() => Shiny.setInputValue('drawer_closed', true, {priority: 'event'})"
      )
    )
  })

  exportTestValues(
    drawer_state = drawerVisible()
  )
}

if (interactive() || is_testing()) shinyApp(ui, server)
```

---

dropdow_menu *Dropdown menu*

Description

Dropdown menu

Usage

```
dropdow_menu(inputId, ..., choices = NULL, selected = NULL)

dropdown_item(...)

dropdown_section(...)

update_dropdown(session = shiny::getDefaultReactiveDomain(), inputId, ...)
```

Arguments

inputId	Unique input id.
...	Props.
choices	Radio choices.
selected	Default selected value.
session	Shiny session.

Details

See https://heroui.com/docs/components/dropdown to get the list of parameters to pass in ....

Value

Object with `shiny.tag` class suitable for use in the UI of a Shiny app. The update functions return nothing (called for side effects).

Note

Container for related dropdown_item.

See Also

See https://heroui.com/docs/components/dropdown.

Examples

```
library(shiny)
library(shinyNextUI)
library(shiny.react)

items <- list(
  # Dropdown section
  dropdown_section(
    showDivider = TRUE,
    title = "Section 1",
    # Dropdown Items
    list(
      dropdown_item(
        title = "Item 1",
        shortcut = "\u2318N",
        color = "danger",
        description = "Item description",
        startContent = icon("clock")
      ),
      dropdown_item(
        title = "Item 2",
        shortcut = "\u2318N",
        color = "success",
        description = "Item description",
        startContent = icon("home")
      ),
      dropdown_item(
        title = "External link",
        href = "https://heroui.com/",
        target = "_blank",
        description = "Go to nextui documentation"
      )
    )
  ),
```

```
  dropdown_section(
    showDivider = FALSE,
    title = "Section 2",
    # Dropdown Items
    list(
      dropdown_item(
        title = "Item 3",
        color = "warning",
        description = "Item description"
      ),
      dropdown_item(
        title = "Item 4"
      )
    )
  )
)

# You can also skip section
#items <- list(
#  dropdown_item(
#    title = "Item 1",
#    shortcut = "\u2318N",
#    color = "danger",
#    description = "Item description"#,
#    #startContent = icon("clock")
#  ),
#  dropdown_item(
#    title = "Item 2",
#    shortcut = "\u2318N",
#    color = "success",
#    description = "Item description"#,
#    #startContent = icon("home")
#  )
#)

ui <- nextui_page(
  debug_react = TRUE,
  div(
    class = "flex gap-2 my-2",
    dropdow_menu(
      inputId = "dropdown",
      label = "Dropdown menu",
      selected = "Item 2",
      variant = "bordered",
      disabledKeys = c("Item 3", "Item 4"),
      selectionMode = "multiple",
      choices = items
    )
  ),
  verbatimTextOutput("dropdown_val")
)

server <- function(input, output, session) {
```

```
  observe({
    print(input$dropdown)
  })
  output$dropdown_val <- renderText(input$dropdown)
}

if (interactive() || is_testing()) shinyApp(ui, server)
```

`get_examples` *Get all available Shiny app examples*

Description

Get all available Shiny app examples

Usage

```
get_examples()
```

Value

A character vector.

`image` *image*

Description

The Image component is used to display images with support for fallback.

Usage

```
image(...)
```

Arguments

`...`	Props to pass to the component. The allowed props are listed below in the **Details** section.

Details

- **src**. Type: `string`. Default: `NA`.
- **srcSet**. Type: `string`. Default: `NA`.
- **sizes**. Type: `string`. Default: `NA`.
- **alt**. Type: `string`. Default: `NA`.
- **width**. Type: `number`. Default: `NA`.
- **height**. Type: `number`. Default: `NA`.
- **radius**. Type: `none OR sm OR md OR lg OR full`. Default: `"xl"`.
- **shadow**. Type: `none OR sm OR md OR lg`. Default: `"none"`.
- **loading**. Type: `eager OR lazy`. Default: `NA`.
- **fallbackSrc**. Type: `string`. Default: `NA`.
- **isBlurred**. Type: `boolean`. Default: `false`.
- **isZoomed**. Type: `boolean`. Default: `false`.
- **removeWrapper**. Type: `boolean`. Default: `false`.
- **disableSkeleton**. Type: `boolean`. Default: `false`.
- **classNames**. Type: `Partial<Record<"img" OR "wrapper" OR "zoomedWrapper" OR "blurredImg", string>>`. Default: `NA`.
- **onLoad**. Type: `ReactEventHandler<HTMLImageElement>`. Default: `NA`.
- **onError**. Type: `() => void`. Default: `NA`.

Value

An object of class `shiny.tag` containing the necessary configuration and including options such as JavaScript dependencies to instantiate a HeroUI image component.

See Also

See https://heroui.com/docs/components/image.

Examples

```
library(shiny)
library(shinyNextUI)
library(shiny.react)

ui <- nextui_page(
  div(
    class = "flex gap-5 my-2",
    image(
      width = 300,
      alt = "NextUI hero Image",
      src = "https://heroui.com/images/hero-card-complete.jpeg"
    ),
    image(
```

```
      isBlurred = TRUE,
      width = 300,
      alt = "Album cover",
      src = "https://heroui.com/images/hero-card-complete.jpeg"
    ),
    image(
      isZoomed = TRUE,
      width = 300,
      alt = "Fruit image with zoom effect",
      src = "https://heroui.com/images/hero-card-complete.jpeg"
    )
  )
)

server <- function(input, output, session) {
}

if (interactive() || is_testing()) shinyApp(ui, server)
```

`input` *input*

Description

Input is a component that allows users to enter text. It can be used to get user inputs in forms, search fields, and more.

Usage

```
text_input(inputId, ..., value = default_value)

update_text_input(session = shiny::getDefaultReactiveDomain(), inputId, ...)

numeric_input(inputId, ..., value = default_value)

update_numeric_input(session = shiny::getDefaultReactiveDomain(), inputId, ...)

date_input(inputId, ..., value = default_value)

update_date_input(session = shiny::getDefaultReactiveDomain(), inputId, ...)
```

Arguments

`inputId`	ID of the component.
`...`	Props to pass to the component. The allowed props are listed below in the **Details** section.
`value`	Starting value.
`session`	Object passed as the `session` argument to Shiny server.

Details

- **children**. Type: ReactNode. Default: NA.
- **variant**. Type: flat OR bordered OR faded OR underlined. Default: "flat".
- **color**. Type: default OR primary OR secondary OR success OR warning OR danger. Default: "default".
- **size**. Type: sm OR md OR lg. Default: "md".
- **radius**. Type: none OR sm OR md OR lg OR full. Default: NA.
- **label**. Type: ReactNode. Default: NA.
- **value**. Type: string. Default: NA.
- **defaultValue**. Type: string. Default: NA.
- **placeholder**. Type: string. Default: NA.
- **description**. Type: ReactNode. Default: NA.
- **errorMessage**. Type: ReactNode OR ((v: ValidationResult) => ReactNode). Default: NA.
- **validate**. Type: (value: string) => ValidationError OR true OR null OR undefined. Default: NA.
- **validationBehavior**. Type: native OR aria. Default: "native".
- **minLength**. Type: number. Default: NA.
- **maxLength**. Type: number. Default: NA.
- **pattern**. Type: string. Default: NA.
- **type**. Type: text OR email OR url OR password OR tel OR search OR file. Default: "text".
- **startContent**. Type: ReactNode. Default: NA.
- **endContent**. Type: ReactNode. Default: NA.
- **labelPlacement**. Type: inside OR outside OR outside-left. Default: "inside".
- **fullWidth**. Type: boolean. Default: true.
- **isClearable**. Type: boolean. Default: false.
- **isRequired**. Type: boolean. Default: false.
- **isReadOnly**. Type: boolean. Default: false.
- **isDisabled**. Type: boolean. Default: false.
- **isInvalid**. Type: boolean. Default: false.
- **baseRef**. Type: RefObject<HTMLDivElement>. Default: NA.
- **disableAnimation**. Type: boolean. Default: false.
- **classNames**. Type: Partial<Record<'base' OR 'label' OR 'inputWrapper' OR 'innerWrapper' OR 'mainWrapper' OR 'input' OR 'clearButton' OR 'helperWrapper' OR 'description' OR 'errorMessage', string>>. Default: NA.
- **onChange**. Type: React.ChangeEvent<HTMLInputElement>. Default: NA.
- **onValueChange**. Type: (value: string) => void. Default: NA.
- **onClear**. Type: () => void. Default: NA.

Value

An object of class `shiny.tag` containing the necessary configuration and including options such as JavaScript dependencies to instantiate a HeroUI input component.

See Also

See https://heroui.com/docs/components/input.

Examples

```
library(shiny)
library(shinyNextUI)

ui <- nextui_page(
  div(
    class = "flex gap-1",
    text_input(
      inputId = "text",
      value = "Plop",
      placeholder = "Next UI",
      label = "Text input"
    ),
    textOutput("text_val")
  ),
  spacer(y = 5),
  div(
    class = "flex gap-1",
    numeric_input(
      inputId = "numeric",
      value = 10,
      label = "Numeric input"
    ),
    textOutput("numeric_val")
  ),
  spacer(y = 5),
  div(
    class = "flex gap-1",
    date_input(
      inputId = "date",
      value = "2023-12-11",
      label = "Date input"
    ),
    textOutput("date_val")
  )
)

server <- function(input, output, session) {
  output$text_val <- renderText(input$text)
  output$numeric_val <- renderText(input$numeric)
  output$date_val <- renderText(input$date)
}
```

```
if (interactive() || is_testing()) shinyApp(ui, server)
```

`is_testing` *Indicates whether testthat is running*

Description

Indicates whether testthat is running

Usage

```
is_testing()
```

Value

Boolean.

`link` *link*

Description

Links allow users to click their way from page to page. This component is styled to resemble a hyperlink and semantically renders an <a>

Usage

```
link(...)
```

Arguments

`...`	Props to pass to the component. The allowed props are listed below in the **Details** section.

Details

- **size**. Type: `sm` OR `md` OR `lg`. Default: `"md"`.
- **color**. Type: `foreground` OR `primary` OR `secondary` OR `success` OR `warning` OR `danger`. Default: `"primary"`.
- **underline**. Type: `none` OR `hover` OR `always` OR `active` OR `focus`. Default: `"none"`.
- **href**. Type: `string`. Default: `NA`.
- **target**. Type: `HTMLAttributeAnchorTarget`. Default: `NA`.
- **rel**. Type: `string`. Default: `NA`.
- **download**. Type: `boolean` OR `string`. Default: `NA`.

- **ping**. Type: `string`. Default: `NA`.
- **referrerPolicy**. Type: `HTMLAttributeReferrerPolicy`. Default: `NA`.
- **isExternal**. Type: `boolean`. Default: `false`.
- **showAnchorIcon**. Type: `boolean`. Default: `false`.
- **anchorIcon**. Type: `ReactNode`. Default: `NA`.
- **isBlock**. Type: `boolean`. Default: `false`.
- **isDisabled**. Type: `boolean`. Default: `false`.
- **disableAnimation**. Type: `boolean`. Default: `false`.
- **onPress**. Type: `(e: PressEvent) => void`. Default: `NA`.
- **onPressStart**. Type: `(e: PressEvent) => void`. Default: `NA`.
- **onPressEnd**. Type: `(e: PressEvent) => void`. Default: `NA`.
- **onPressChange**. Type: `(isPressed: boolean) => void`. Default: `NA`.
- **onPressUp**. Type: `(e: PressEvent) => void`. Default: `NA`.
- **onKeyDown**. Type: `(e: KeyboardEvent) => void`. Default: `NA`.
- **onKeyUp**. Type: `(e: KeyboardEvent) => void`. Default: `NA`.
- **onClick**. Type: `MouseEventHandler`. Default: `NA`.

Value

An object of class `shiny.tag` containing the necessary configuration and including options such as JavaScript dependencies to instantiate a HeroUI link component.

See Also

See https://heroui.com/docs/components/link.

Examples

```
library(shiny)
library(shinyNextUI)

colors <- c(
  "default",
  "primary",
  "secondary",
  "success",
  "warning",
  "error"
)

link_config <- data.frame(
  color = colors,
  underline = c(rep("none", 2), "hover", "always", "active", "focus"),
  block = c(rep(TRUE, 3), rep(FALSE, 3)),
  href = c(rep("#", 3), rep("https://google.com", 3)),
```

```
  isExternal = c(rep(FALSE, 3), rep(TRUE, 3)),
  size = rep(c("sm", "md", "lg"), 2)
)

link_factory <- function(color, underline, block, href, isExternal, size) {
  link(
    "A super link!",
    color = color,
    underline = underline,
    isBlock = block,
    href = href,
    isExternal = isExternal,
    size = size
  )
}

links <- purrr::pmap(link_config, link_factory)

ui <- nextui_page(
  div(
    class = "grid gap-4 grid-cols-3 grid-rows-2 m-5",
    links
  )
)

server <- function(input, output, session) {}

if (interactive() || is_testing()) shinyApp(ui, server)
```

`listbox` *listbox*

Description

A listbox displays a list of options and allows a user to select one or more of them.

Usage

```
listbox(inputId, ..., value = default_value)

listbox_section(...)

listbox_item(...)

update_listbox(session = shiny::getDefaultReactiveDomain(), inputId, ...)
```

Arguments

`inputId`	ID of the component.

...	Props to pass to the component. The allowed props are listed below in the **Details** section.
value	Starting value.
session	Object passed as the session argument to Shiny server.

Details

- **children***. Type: ReactNode[]. Default: NA.
- **items**. Type: Iterable<T>. Default: NA.
- **variant**. Type: solid OR bordered OR light OR flat OR faded OR shadow. Default: "solid".
- **color**. Type: default OR primary OR secondary OR success OR warning OR danger. Default: "default".
- **selectionMode**. Type: none OR single OR multiple. Default: NA.
- **selectedKeys**. Type: React.Key[]. Default: NA.
- **disabledKeys**. Type: React.Key[]. Default: NA.
- **defaultSelectedKeys**. Type: all OR React.Key[]. Default: NA.
- **disallowEmptySelection**. Type: boolean. Default: false.
- **shouldHighlightOnFocus**. Type: boolean. Default: false.
- **autoFocus**. Type: boolean OR first OR last. Default: false.
- **topContent**. Type: ReactNode. Default: NA.
- **bottomContent**. Type: ReactNode. Default: NA.
- **emptyContent**. Type: ReactNode. Default: "No items.".
- **shouldFocusWrap**. Type: boolean. Default: false.
- **isVirtualized**. Type: boolean. Default: false.
- **virtualization**. Type: Record<"maxListboxHeight" & "itemHeight", number>. Default: NA.
- **hideEmptyContent**. Type: boolean. Default: false.
- **hideSelectedIcon**. Type: boolean. Default: false.
- **disableAnimation**. Type: boolean. Default: false.
- **classNames**. Type: Partial<Record<"base" OR "list" OR "emptyContent", string>>. Default: NA.
- **itemClasses**. Type: Partial<Record<"base" OR "wrapper" OR "title" OR "description" OR "selectedIcon", string>>. Default: NA.
- **onAction**. Type: (key: React.Key) => void. Default: NA.
- **onSelectionChange**. Type: (keys: React.Key[]) => void. Default: NA.
- **children***. Type: ReactNode. Default: NA.
- **title**. Type: string. Default: NA.
- **items**. Type: Iterable<T>. Default: NA.
- **hideSelectedIcon**. Type: boolean. Default: false.

- **showDivider**. Type: boolean. Default: false.
- **dividerProps**. Type: DividerProps. Default: NA.
- **classNames**. Type: Partial<Record<"base" OR "heading" OR "group" OR "divider", string>>. Default: NA.
- **itemClasses**. Type: Partial<Record<"base" OR "wrapper" OR "title" OR "description" OR "shortcut" OR "selectedIcon", string>>. Default: NA.
- **children***. Type: ReactNode. Default: NA.
- **key**. Type: React.Key. Default: NA.
- **title**. Type: string OR ReactNode. Default: NA.
- **textValue**. Type: string. Default: NA.
- **description**. Type: string OR ReactNode. Default: NA.
- **shortcut**. Type: string OR ReactNode. Default: NA.
- **startContent**. Type: ReactNode. Default: NA.
- **endContent**. Type: ReactNode. Default: NA.
- **selectedIcon**. Type: ListboxItemSelectedIconProps. Default: NA.
- **href**. Type: string. Default: NA.
- **target**. Type: HTMLAttributeAnchorTarget. Default: NA.
- **rel**. Type: string. Default: NA.
- **download**. Type: boolean OR string. Default: NA.
- **ping**. Type: string. Default: NA.
- **referrerPolicy**. Type: HTMLAttributeReferrerPolicy. Default: NA.
- **shouldHighlightOnFocus**. Type: boolean. Default: false.
- **hideSelectedIcon**. Type: boolean. Default: false.
- **showDivider**. Type: boolean. Default: false.
- **isDisabled**. Type: boolean. Default: false.
- **isSelected**. Type: boolean. Default: false.
- **isReadOnly**. Type: boolean. Default: false.
- **classNames**. Type: Partial<Record<"base" OR "wrapper" OR "title" OR "description" OR "shortcut" OR "selectedIcon", string>>. Default: NA.
- **onAction**. Type: () => void. Default: NA.
- **onPress**. Type: (e: PressEvent) => void. Default: NA.
- **onPressStart**. Type: (e: PressEvent) => void. Default: NA.
- **onPressEnd**. Type: (e: PressEvent) => void. Default: NA.
- **onPressChange**. Type: (isPressed: boolean) => void. Default: NA.
- **onPressUp**. Type: (e: PressEvent) => void. Default: NA.
- **onKeyDown**. Type: (e: KeyboardEvent) => void. Default: NA.
- **onKeyUp**. Type: (e: KeyboardEvent) => void. Default: NA.
- **onClick**. Type: MouseEventHandler. Default: NA.

Value

An object of class `shiny.tag` containing the necessary configuration and including options such as JavaScript dependencies to instantiate a HeroUI listbox component.

See Also

See https://heroui.com/docs/components/listbox.

Examples

```
library(shiny)
library(shinyNextUI)
library(shiny.react)

animals <- list(
  list(
    label = "Bulbasaur",
    value = "bulbasaur",
    description = "Blabla",
   avatar = "https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/1.png"
  ),
  list(
    label = "Pikachu",
    value = "pikachu",
    description = "Electric mouse",
   avatar = "https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/25.png"
  )
)

items <- lapply(animals, function(animal) {
  listbox_item(
    key = animal[["value"]],
    description = animal[["description"]],
    startContent = avatar(src = animal[["avatar"]]),
    animal[["label"]]
  )
})

ui <- nextui_page(
  debug_react = TRUE,
  spacer(y = 4),
  action_button("update", "Update to Pikachu?"),
  spacer(y = 4),
  div(
    class = "w-full max-w-[260px] border-small px-1 py-2
    rounded-small border-default-200 dark:border-default-100",
    listbox(
      "listbox",
      label = "Select a pokemon",
      value = "bulbasaur",
      variant = "flat",
      selectionMode = "single",
```

```
      listbox_section(
        title = "Default pokemons",
        items
      )
    )
  ),
  textOutput("res")
)

server <- function(input, output, session) {
  output$res <- renderText(input$listbox)
  observeEvent(input$listbox, {
    print(input$listbox)
  })

  observeEvent(input$update, {
    update_listbox(session, "listbox", value = JS("['pikachu']"))
  })
}

if (interactive() || is_testing()) shinyApp(ui, server)
```

`modal` *modal*

Description

Displays a dialog with custom content that requires attention or provides additional information.

Usage

```
modal(...)

modal_content(...)

modal_header(...)

modal_body(...)

modal_footer(...)
```

Arguments

`...`	Props to pass to the component. The allowed props are listed below in the **Details** section.

Details

- **children***. Type: ReactNode. Default: NA.
- **size**. Type: xs OR sm OR md OR lg OR xl OR 2xl OR 3xl OR 4xl OR 5xl OR full. Default: "md".
- **radius**. Type: none OR sm OR md OR lg. Default: "lg".
- **shadow**. Type: none OR sm OR md OR lg. Default: "lg".
- **backdrop**. Type: transparent OR opaque OR blur. Default: "opaque".
- **scrollBehavior**. Type: normal OR inside OR outside. Default: "normal".
- **placement**. Type: auto OR top OR center OR bottom. Default: "auto".
- **isOpen**. Type: boolean. Default: NA.
- **defaultOpen**. Type: boolean. Default: NA.
- **isDismissable**. Type: boolean. Default: true.
- **isKeyboardDismissDisabled**. Type: boolean. Default: false.
- **shouldBlockScroll**. Type: boolean. Default: true.
- **hideCloseButton**. Type: boolean. Default: false.
- **closeButton**. Type: ReactNode. Default: NA.
- **motionProps**. Type: MotionProps. Default: NA.
- **portalContainer**. Type: HTMLElement. Default: "document.body".
- **disableAnimation**. Type: boolean. Default: false.
- **classNames**. Type: Partial<Record<'wrapper' OR 'base' OR 'backdrop' OR 'header' OR 'body' OR 'footer' OR 'closeButton', string>>. Default: NA.
- **onOpenChange**. Type: (isOpen: boolean) => void. Default: NA.
- **onClose**. Type: () => void. Default: NA.

Value

An object of class shiny.tag containing the necessary configuration and including options such as JavaScript dependencies to instantiate a HeroUI modal component.

See Also

See https://heroui.com/docs/components/modal.

Examples

```
library(shiny)
library(shinyNextUI)
library(shiny.react)

ui <- nextui_page(
  action_button(
    inputId = "show_modal",
    color = "primary",
    shadow = TRUE,
```

```
    "Show modal"
  ),
  radio_input(
    "scroll",
    label = "Scroll option",
    choices = c(
      "inside" = "Inside",
      "outside" = "Outside"
    ),
    selected = "inside"
  ),
  reactOutput("modal")
)

server <- function(input, output, session) {
  modalVisible <- reactiveVal(FALSE)
  observeEvent(input$show_modal, {
    modalVisible(TRUE)
  })

  observeEvent(input$modal_closed, {
    modalVisible(FALSE)
  })

  output$modal <- renderReact({
    modal(
      scrollBehavior = input$scroll,
      isOpen = modalVisible(),
      size = "sm",
      modal_content(
        modal_header("My modal"),
        modal_body(
          p(
            "Cras mattis consectetur purus sit amet fermentum. Cras justo odio,
            dapibus ac facilisis in, egestas eget quam. Morbi leo risus, porta
            ac consectetur ac, vestibulum at eros."
          ),
          p(
            "Lorem ipsum dolor sit amet, consectetur adipiscing elit.
            Nullam pulvinar risus non risus hendrerit venenatis.
            Pellentesque sit amet hendrerit risus, sed porttitor quam.
            Magna exercitation reprehenderit magna aute tempor cupidatat
            consequat elit dolor adipisicing. Mollit dolor eiusmod sunt ex
            incididunt cillum quis. Velit duis sit officia eiusmod Lorem
            aliqua enim laboris do dolor eiusmod. Et mollit incididunt
            nisi consectetur esse laborum eiusmod pariatur proident Lorem
            eiusmod et. Culpa deserunt nostrud ad veniam."
          ),
          p(
            "Mollit dolor eiusmod sunt ex incididunt cillum quis. Velit
             duis sit officia eiusmod Lorem aliqua enim laboris do dolor
             eiusmod. Et mollit incididunt nisi consectetur esse laborum
             eiusmod pariatur proident Lorem eiusmod et. Culpa deserunt
```

```
              nostrud ad veniam. Lorem ipsum dolor sit amet, consectetur
              adipiscing elit. Nullam pulvinar risus non risus hendrerit
              venenatis. Pellentesque sit amet hendrerit risus, sed
              porttitor quam. Magna exercitation reprehenderit magna aute
              tempor cupidatat consequat elit dolor adipisicing. Mollit
              dolor eiusmod sunt ex incididunt cillum quis. Velit duis sit
              officia eiusmod Lorem aliqua enim laboris do dolor eiusmod. Et
              mollit incididunt nisi consectetur esse laborum eiusmod
              pariatur proident Lorem eiusmod et. Culpa deserunt nostrud ad
              veniam."
            ),
            p(
              "Mollit dolor eiusmod sunt ex incididunt cillum quis. Velit
              duis sit officia eiusmod Lorem aliqua enim laboris do dolor
              eiusmod. Et mollit incididunt nisi consectetur esse laborum
              eiusmod pariatur proident Lorem eiusmod et. Culpa deserunt
              nostrud ad veniam. Lorem ipsum dolor sit amet, consectetur
              adipiscing elit. Nullam pulvinar risus non risus hendrerit
              venenatis. Pellentesque sit amet hendrerit risus, sed
              porttitor quam. Magna exercitation reprehenderit magna aute
              tempor cupidatat consequat elit dolor adipisicing. Mollit
              dolor eiusmod sunt ex incididunt cillum quis. Velit duis sit
              officia eiusmod Lorem aliqua enim laboris do dolor eiusmod. Et
              mollit incididunt nisi consectetur esse laborum eiusmod
              pariatur proident Lorem eiusmod et. Culpa deserunt nostrud ad
              veniam."
            )
          ),
          modal_footer("Modal footer")
        ),
        onClose = JS("() => Shiny.setInputValue('modal_closed', true, {priority: 'event'})")
      )
    })

    exportTestValues(
      modal_state = modalVisible()
    )
  }

  if (interactive() || is_testing()) shinyApp(ui, server)
```

---

navbar *navbar*

Description

A responsive navigation header positioned on top side of your page that includes support for branding, links, navigation, collapse menu and more.

Usage

```
navbar(...)

navbar_brand(...)

navbar_content(...)

navbar_item(...)

navbar_toggle(...)
```

Arguments

...	Props to pass to the component. The allowed props are listed below in the **Details** section.

Details

- **children***. Type: ReactNode[]. Default: NA.
- **height**. Type: string OR number. Default: "4rem (64px)".
- **position**. Type: static OR sticky. Default: "sticky".
- **maxWidth**. Type: sm OR md OR lg OR xl OR 2xl OR full. Default: "lg".
- **parentRef**. Type: React.RefObject<HTMLElement>. Default: "window".
- **isBordered**. Type: boolean. Default: false.
- **isBlurred**. Type: boolean. Default: true.
- **isMenuOpen**. Type: boolean. Default: false.
- **isMenuDefaultOpen**. Type: boolean. Default: false.
- **shouldHideOnScroll**. Type: boolean. Default: false.
- **motionProps**. Type: MotionProps. Default: NA.
- **disableScrollHandler**. Type: boolean. Default: false.
- **disableAnimation**. Type: boolean. Default: false.
- **classNames**. Type: Partial<Record<'base' OR 'wrapper' OR 'brand' OR 'content' OR 'item' OR 'toggle' OR 'toggleIcon' OR 'menu' OR 'menuItem', string>>. Default: NA.
- **onMenuOpenChange**. Type: (isOpen: boolean) => void. Default: NA.
- **onScrollPositionChange**. Type: (position: number) => void. Default: NA.
- **children***. Type: ReactNode[]. Default: NA.
- **justify**. Type: start OR center OR end. Default: "start".
- **children**. Type: ReactNode. Default: NA.
- **isActive**. Type: boolean. Default: false.
- **icon**. Type: ReactNode OR ((isOpen: boolean OR undefined) => ReactNode). Default: NA.

- **isSelected**. Type: `boolean`. Default: `false`.
- **defaultSelected**. Type: `boolean`. Default: `false`.
- **srOnlyText**. Type: `string`. Default: `"open/close navigation menu"`.
- **onChange**. Type: `(isOpen: boolean) => void`. Default: `NA`.
- **children***. Type: `ReactNode[]`. Default: `NA`.
- **portalContainer**. Type: `HTMLElement`. Default: `"document.body"`.
- **motionProps**. Type: `MotionProps`. Default: `NA`.
- **children**. Type: `ReactNode`. Default: `NA`.
- **isActive**. Type: `boolean`. Default: `false`.

Value

An object of class `shiny.tag` containing the necessary configuration and including options such as JavaScript dependencies to instantiate a HeroUI navbar component.

See Also

See https://heroui.com/docs/components/navbar.

Examples

```
library(shiny)
library(shinyNextUI)
library(shiny.react)
library(shiny.router)

sections <- c("main", "other")

layout <- function(..., content) {
  tags$div(
    css = JS("
      {
        maxW: '100%',
        boxSizing: 'border-box',
      }"
    ),
    ..., # Navbar
    # Content
    tags$div(
      css = JS(
        "{
          boxSizing: 'border-box',
          px: '$12',
          mt: '$8',
          '@xsMax': {px: '$10'}
        }"
      ),
```

```
      content
    )
  )
}

# TO DO: create wrapper for end-user to simplify all this mess.
create_navbar <- function(id) {

  input <- get("input", envir = parent.frame())

  nav_links <- lapply(seq_along(sections), function(i) {
    # Li elements
    link(
      inputId = sprintf("link_%s", i),
      href = route_link(sections[[i]]),
      key = i,
      value = i,
      parent = sprintf("navbar_%s", sections[[i]]),
      isActive = if (is.null(input[[sprintf("navbar_%s", id)]])) {
        if (i == 1) TRUE else FALSE
      } else {
        if (input[[sprintf("navbar_%s", id)]] == i) TRUE else FALSE
      },
      sprintf("Link to %s", sections[[i]])
    )
  })

  nav <- navbar(
    id = sprintf("navbar_%s", id),
    maxWidth = "lg",
    variant = "floating",
    isBordered = TRUE,
    navbar_brand(p(b = TRUE, "Brand", color = "inherit", hideIn = "xs")),
    # Ul element
    navbar_content(
      variant = "highlight",
      activeColor = "success",
      nav_links,
      navbar_item(
        action_button(
          inputId = sprintf("navbar_button-%s", id),
          "Click me",
          auto = TRUE,
          flat = TRUE
        )
      )
    )
  )

  if (is.null(input[[sprintf("navbar_%s", id)]])) {
    tagList(
      tags$script(
        sprintf("Shiny.setInputValue('navbar_%s', 0)", id)
```

```
      ),
      nav
    )
  } else {
    nav
  }
}

page <- function(id, content) {
  layout(
    reactOutput(sprintf("nav_%s", id)),
    content = content
  )
}

home <- page(
  id = "main",
  card(
    numeric_input(
      inputId = "obs",
      label = "Number of observations:",
      value = 500
    ),
    plotOutput("distPlot")
  )
)
other <- page(
  id = "other",
  tableOutput('table')
)

ui <- nextui_page(
  router_ui(
    route("main", home),
    route("other", other)
  )
)

server <- function(input, output, session) {
  observe(print(input$navbar))
  output$nav_main <- renderReact({
    create_navbar("main")
  })

  output$nav_other <- renderReact({
    create_navbar("other")
  })

  output$distPlot <- renderPlot({
    hist(rnorm(input$obs))
  })
  output$table <- renderTable(iris)
```

```
  router_server("main")
}

if (interactive() || is_testing()) shinyApp(ui, server)
```

`nextui_page` *NextUI page wrapper*

Description

Suppressed Bootstrap dependency which is not needed.

Usage

```
nextui_page(..., dark_mode = FALSE, debug_react = FALSE)
```

Arguments

`...`	UI elements.
`dark_mode`	Apply global dark mode. If NULL, no switch is shown.
`debug_react`	Whether to enable react debug mode. Default to FALSE.

Value

Object which can be passed as the UI of a Shiny app.

`pagination` *pagination*

Description

The Pagination component allows you to display active page and navigate between multiple pages.

Usage

```
pagination(inputId, ..., value = default_value)

update_pagination(session = shiny::getDefaultReactiveDomain(), inputId, ...)
```

Arguments

`inputId`	ID of the component.
`...`	Props to pass to the component. The allowed props are listed below in the **Details** section.
`value`	Starting value.
`session`	Object passed as the `session` argument to Shiny server.

Details

- **variant**. Type: `flat OR bordered OR light OR faded`. Default: `"flat"`.
- **color**. Type: `default OR primary OR secondary OR success OR warning OR danger`. Default: `"default"`.
- **size**. Type: `sm OR md OR lg`. Default: `"md"`.
- **radius**. Type: `none OR sm OR md OR lg OR full`. Default: `"xl"`.
- **total**. Type: `number`. Default: `"1"`.
- **dotsJump**. Type: `number`. Default: `"5"`.
- **initialPage**. Type: `number`. Default: `"1"`.
- **page**. Type: `number`. Default: `NA`.
- **siblings**. Type: `number`. Default: `"1"`.
- **boundaries**. Type: `number`. Default: `"1"`.
- **loop**. Type: `boolean`. Default: `false`.
- **isCompact**. Type: `boolean`. Default: `false`.
- **isDisabled**. Type: `boolean`. Default: `false`.
- **showShadow**. Type: `boolean`. Default: `false`.
- **showControls**. Type: `boolean`. Default: `false`.
- **disableCursorAnimation**. Type: `boolean`. Default: `false`.
- **disableAnimation**. Type: `boolean`. Default: `false`.
- **renderItem**. Type: `PaginationItemProps`. Default: `NA`.
- **getItemAriaLabel**. Type: `(page: string) => string`. Default: `NA`.
- **classNames**. Type: `Partial<Record<'base' OR 'wrapper' OR 'prev' OR 'next' OR 'item' OR 'cursor' OR 'forwardIcon' OR 'ellipsis' OR 'chevronNext', string>>`. Default: `NA`.
- **onChange**. Type: `(page: number) => void`. Default: `NA`.

Value

An object of class `shiny.tag` containing the necessary configuration and including options such as JavaScript dependencies to instantiate a HeroUI pagination component.

See Also

See https://heroui.com/docs/components/pagination.

Examples

```
library(shiny)
library(shinyNextUI)
library(shiny.react)
library(thematic)

thematic_shiny()
```

```
max <- 10
cards <- lapply(seq_len(max), function(i) {
  tagList(
    spacer(y = 2),
    card(
      id = sprintf("mycard-%s", i),
      card_header(sprintf("Card %s", i)),
      card_body(
        sprintf("Card content %s", i),
        slider_input(
          sprintf("obs-%s", i),
          "Obs",
          minValue = 0,
          maxValue = 1000,
          value = 500
        ),
        plotOutput(sprintf("distPlot-%s", i))
      )
    )
  )
})

ui <- nextui_page(
  pagination(
    inputId = "pagination",
    loop = TRUE,
    size = "lg",
    variant = "bordered",
    showControls = TRUE,
    page = 1,
    total = max
  )
)

server <- function(input, output, session) {
  history <- reactiveVal(NULL)

  # Dynamically insert cards with the pagination.
  observeEvent(input$pagination, {
    if (!is.null(history()))
      removeUI(sprintf("#mycard-%s", history()), multiple = TRUE)
    insertUI(
      selector = "#pagination",
      where = "afterEnd",
      ui = cards[[input$pagination]]
    )
    history(input$pagination)

    output[[sprintf("distPlot-%s", history())]] <- renderPlot({
      req(input[[sprintf("obs-%s", history())]])
      hist(
        rnorm(input[[sprintf("obs-%s", history())]]),
```

```
          main = sprintf("Super plot %s", history())
        )
      })
    })
}

if (interactive() || is_testing()) shinyApp(ui, server)
```

---

poke_data *Pokemon API data*

Description

Extract of some data from the 151 first Pokemons.

Usage

```
poke_data
```

Format

poke_data**:**

A nested list with 151 entries. Each sublist contains:

- name (char): Pokemon name.
- description (char): Pokemon description.
- shape (char): Pokemon shape.
- sprites (list):
  - front_default (char): front sprite URL.
  - shiny_default (char): front sprite URL (shiny form).
- ...

Note

Have a look to inst/app-doc/data-doc.html to get an interactive overview.

Source

https://pokeapi.co/docs/v2

popover *popover*

Description

Popover is a non-modal dialog that floats around its disclosure. It's commonly used for displaying additional rich content on top of something.

Usage

```
popover(...)

popover_trigger(...)

popover_content(...)
```

Arguments

...	Props to pass to the component. The allowed props are listed below in the **Details** section.

Details

- **children***. Type: `ReactNode[]`. Default: `NA`.
- **size**. Type: `sm` OR `md` OR `lg`. Default: `"md"`.
- **color**. Type: `default` OR `primary` OR `secondary` OR `success` OR `warning` OR `danger`. Default: `"default"`.
- **radius**. Type: `none` OR `sm` OR `md` OR `lg` OR `full`. Default: `"lg"`.
- **shadow**. Type: `none` OR `sm` OR `md` OR `lg`. Default: `"lg"`.
- **backdrop**. Type: `transparent` OR `opaque` OR `blur`. Default: `"transparent"`.
- **placement**. Type: `PopoverPlacement`. Default: `"bottom"`.
- **state**. Type: `OverlayTriggerState`. Default: `NA`.
- **isOpen**. Type: `boolean`. Default: `NA`.
- **defaultOpen**. Type: `boolean`. Default: `NA`.
- **offset**. Type: `number`. Default: `"7"`.
- **containerPadding**. Type: `number`. Default: `"12"`.
- **crossOffset**. Type: `number`. Default: `"0"`.
- **triggerType**. Type: `dialog` OR `menu` OR `listbox` OR `tree` OR `grid`. Default: `"dialog"`.
- **showArrow**. Type: `boolean`. Default: `false`.
- **shouldFlip**. Type: `boolean`. Default: `true`.
- **triggerScaleOnOpen**. Type: `boolean`. Default: `true`.
- **shouldBlockScroll**. Type: `boolean`. Default: `false`.

- **shouldCloseOnScroll**. Type: `boolean`. Default: `false`.
- **isKeyboardDismissDisabled**. Type: `boolean`. Default: `false`.
- **shouldCloseOnBlur**. Type: `boolean`. Default: `false`.
- **motionProps**. Type: `MotionProps`. Default: `NA`.
- **portalContainer**. Type: `HTMLElement`. Default: `"document.body"`.
- **disableAnimation**. Type: `boolean`. Default: `false`.
- **classNames**. Type: `Partial<Record<'base' OR 'trigger' OR 'backdrop' OR 'content', string>>`. Default: `NA`.
- **onOpenChange**. Type: `(isOpen: boolean) => void`. Default: `NA`.
- **shouldCloseOnInteractOutside**. Type: `(e: HTMLElement) => void`. Default: `NA`.
- **onClose**. Type: `() => void`. Default: `NA`.
- **children***. Type: `ReactNode`. Default: `NA`.
- **children**. Type: `ReactNode`. Default: `NA`.

Value

An object of class `shiny.tag` containing the necessary configuration and including options such as JavaScript dependencies to instantiate a HeroUI popover component.

See Also

See https://heroui.com/docs/components/popover.

Examples

```
library(shiny)
library(shinyNextUI)
library(shiny.react)

ui <- nextui_page(
  dark_mode = TRUE,
  div(
    className = "flex align-items-start",
    popover(
      showArrow = TRUE,
      placement = "right",
      backdrop = "blur",
      popover_trigger(button("Click me!", color = "primary")),
      popover_content(
        div(
          className = "px-1 py-2",
          "This is the content of the popover."
        )
      )
    )
  )
```

```
)

server <- function(input, output, session) {
}

if (interactive() || is_testing()) shinyApp(ui, server)
```

---

progress *progress*

Description

The Progress component allows you to view the progress of any activity.

Usage

```
progress(...)
```

Arguments

...	Props to pass to the component. The allowed props are listed below in the **Details** section.

Details

- **label**. Type: `ReactNode`. Default: `NA`.
- **size**. Type: `sm` OR `md` OR `lg`. Default: `"md"`.
- **color**. Type: `default` OR `primary` OR `secondary` OR `success` OR `warning` OR `danger`. Default: `"primary"`.
- **radius**. Type: `none` OR `sm` OR `md` OR `lg` OR `full`. Default: `"full"`.
- **value**. Type: `number`. Default: `NA`.
- **valueLabel**. Type: `ReactNode`. Default: `NA`.
- **minValue**. Type: `number`. Default: `"0"`.
- **maxValue**. Type: `number`. Default: `"100"`.
- **formatOptions**. Type: `Intl.NumberFormat`. Default: `"{style: 'percent'}"`.
- **isIndeterminate**. Type: `boolean`. Default: `false`.
- **isStriped**. Type: `boolean`. Default: `false`.
- **showValueLabel**. Type: `boolean`. Default: `true`.
- **isDisabled**. Type: `boolean`. Default: `false`.
- **disableAnimation**. Type: `boolean`. Default: `false`.
- **classNames**. Type: `Partial<Record<'base' OR 'labelWrapper' OR 'label' OR 'track' OR 'value' OR 'indicator', string>>`. Default: `NA`.

Value

An object of class `shiny.tag` containing the necessary configuration and including options such as JavaScript dependencies to instantiate a HeroUI progress component.

See Also

See https://heroui.com/docs/components/progress.

Examples

```
library(shiny)
library(shinyNextUI)

colors <- c(
  "default",
  "primary",
  "secondary",
  "success",
  "warning",
  "danger"
)

bool_par <- c(rep(FALSE, 3), rep(TRUE, 2), FALSE)

progress_config <- data.frame(
  label = c(rep("A label", 3), rep("", 3)),
  show_value_label = c(rep(FALSE, 3), rep(TRUE, 3)),
  size = c("sm", "sm", rep("md", 2), "lg", "lg"),
  color = colors,
  striped = bool_par,
  radius = c(rep("none", 2), "sm", "md", "lg", "full")
)

progress_factory <- function(
  label,
  show_value_label,
  size,
  color,
  striped,
  radius
) {
  progress(
    label = label,
    showValueLabel = show_value_label,
    value = round(runif(1, 0, 100)),
    size = size,
    color = color,
    isStriped = striped,
    radius = radius
  )
}
```

```
progresses <- purrr::pmap(progress_config, progress_factory)

ui <- nextui_page(
  div(
    class = "grid gap-4 grid-cols-3 grid-rows-3 m-5",
    progresses
  )
)

server <- function(input, output, session) {
}

if (interactive() || is_testing()) shinyApp(ui, server)
```

`radio_input` *Radio input*

Description

Radio input

Usage

```
radio_input(inputId, ..., choices, selected = choices[1])

update_radio_input(
  session = shiny::getDefaultReactiveDomain(),
  inputId,
  ...,
  choices = NULL,
  selected = NULL
)
```

Arguments

`inputId`	Unique input id.
`...`	Props.
`choices`	Radio choices.
`selected`	Default selected value.
`session`	Shiny session.

Details

See https://heroui.com/docs/components/radio-group to get the list of parameters to pass in `...`.

Value

Object with `shiny.tag` class suitable for use in the UI of a Shiny app. The update functions return nothing (called for side effects).

See Also

See https://heroui.com/docs/components/radio-group.

Examples

```
library(shiny)
library(shinyNextUI)
library(shiny.react)

ui <- nextui_page(
  debug_react = TRUE,
  div(
    class = "flex flex-col gap-1",
    spacer(y = 2),
    select_input(
      "select",
      label = "Tab to select:",
      value = JS("['sydney']"),
      disallowEmptySelection = TRUE,
      select_item(key = "buenos-aires", value = "buenos-aires", "Buenos Aires"),
      select_item(key = "sydney", value = "sydney", "Sydney")
    ),
    spacer(y = 2),
    radio_input(
      inputId = "radio",
      label = "Radios",
      description = "Radios are fun.",
      orientation = "horizontal",
      choices = c(
        "buenos-aires" = "Buenos Aires",
        "sydney" = "Sydney"
      )
    ),
    textOutput("radio_val")
  )
)

server <- function(input, output, session) {
  observeEvent(input$select, {
    update_radio_input(session, "radio", selected = input$select)
  })
  output$radio_val <- renderText(input$radio)
}

if (interactive() || is_testing()) shinyApp(ui, server)
```

run_example *Run shinyNextUI example*

Description

Run shinyNextUI example

Usage

```
run_example(name)
```

Arguments

name	Use get_examples to get the available examples.

Value

Runs a Shiny app.

select *select*

Description

A select displays a collapsible list of options and allows a user to select one or more of them.

Usage

```
select_input(inputId, ..., value = default_value)

select_section(...)

select_item(...)

update_select_input(session = shiny::getDefaultReactiveDomain(), inputId, ...)
```

Arguments

inputId	ID of the component.
...	Props to pass to the component. The allowed props are listed below in the **Details** section.
value	Starting value.
session	Object passed as the session argument to Shiny server.

Details

1. Select Props
    - **children***. Type: `ReactNode[]`. Default: `NA`.
    - **items**. Type: `Iterable<T>`. Default: `NA`.
    - **selectionMode**. Type: `single` OR `multiple`. Default: `NA`.
    - **selectedKeys**. Type: `all` OR `Iterable<React.Key>`. Default: `NA`.
    - **disabledKeys**. Type: `Iterable<React.Key>`. Default: `NA`.
    - **defaultSelectedKeys**. Type: `all` OR `Iterable<React.Key>`. Default: `NA`.
    - **variant**. Type: `flat` OR `bordered` OR `faded` OR `underlined`. Default: `"flat"`.
    - **color**. Type: `default` OR `primary` OR `secondary` OR `success` OR `warning` OR `danger`. Default: `"default"`.
    - **size**. Type: `sm` OR `md` OR `lg`. Default: `"md"`.
    - **radius**. Type: `none` OR `sm` OR `md` OR `lg` OR `full`. Default: `NA`.
    - **placeholder**. Type: `string`. Default: `"Select an option"`.
    - **labelPlacement**. Type: `inside` OR `outside` OR `outside-left`. Default: `"inside"`.
    - **label**. Type: `ReactNode`. Default: `NA`.
    - **description**. Type: `ReactNode`. Default: `NA`.
    - **errorMessage**. Type: `ReactNode` OR `((v: ValidationResult) => ReactNode)`. Default: `NA`.
    - **startContent**. Type: `ReactNode`. Default: `NA`.
    - **endContent**. Type: `ReactNode`. Default: `NA`.
    - **selectorIcon**. Type: `ReactNode`. Default: `NA`.
    - **scrollRef**. Type: `React.RefObject<HTMLElement>`. Default: `NA`.
    - **spinnerRef**. Type: `React.RefObject<HTMLElement>`. Default: `NA`.
    - **maxListboxHeight**. Type: `number`. Default: `"256"`.
    - **itemHeight**. Type: `number`. Default: `"32"`.
    - **isVirtualized**. Type: `boolean`. Default: `"undefined"`.
    - **fullWidth**. Type: `boolean`. Default: `true`.
    - **isOpen**. Type: `boolean`. Default: `NA`.
    - **defaultOpen**. Type: `boolean`. Default: `NA`.
    - **isRequired**. Type: `boolean`. Default: `false`.
    - **isDisabled**. Type: `boolean`. Default: `false`.
    - **isMultiline**. Type: `boolean`. Default: `false`.
    - **isInvalid**. Type: `boolean`. Default: `false`.
    - **validationState**. Type: `valid` OR `invalid`. Default: `NA`.
    - **showScrollIndicators**. Type: `boolean`. Default: `true`.
    - **autoFocus**. Type: `boolean`. Default: `false`.
    - **disallowEmptySelection**. Type: `boolean`. Default: `false`.
    - **disableAnimation**. Type: `boolean`. Default: `true`.
    - **disableSelectorIconRotation**. Type: `boolean`. Default: `false`.
    - **hideEmptyContent**. Type: `boolean`. Default: `false`.
    - **popoverProps**. Type: `PopoverProps`. Default: `NA`.

- **listboxProps**. Type: `ListboxProps`. Default: `NA`.
- **scrollShadowProps**. Type: `ScrollShadowProps`. Default: `NA`.
- **classNames**. Type: `Partial<Record<"base"OR "label"OR "trigger"OR "mainWrapper" OR "innerWrapper"OR "selectorIcon" OR "value" OR "listboxWrapper"OR "listbox" OR "popoverContent" OR "helperWrapper" OR "description" OR "errorMessage", string>>`. Default: `NA`.

2. Select Events

- **onClose**. Type: `() => void`. Default: `NA`.
- **onOpenChange**. Type: `(isOpen: boolean) => void`. Default: `NA`.
- **onSelectionChange**. Type: `(keys: "all" OR Set<React.Key> & {anchorKey?: string; currentKey?: string}) => void`. Default: `NA`.
- **onChange**. Type: `React.ChangeEvent<HTMLSelectElement>`. Default: `NA`.
- **renderValue**. Type: `RenderValueFunction`. Default: `NA`.

Value

An object of class `shiny.tag` containing the necessary configuration and including options such as JavaScript dependencies to instantiate a HeroUI select component.

See Also

See https://heroui.com/docs/components/select.

Examples

```
library(shiny)
library(shinyNextUI)
library(shiny.react)

animals <- list(
  list(
    label = "Bulbasaur",
    value = "bulbasaur",
    description = "Blabla",
   avatar = "https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/1.png"
  ),
  list(
    label = "Pikachu",
    value = "pikachu",
    description = "Electric mouse",
   avatar = "https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/25.png"
  )
)

select_items <- lapply(animals, function(animal) {
  select_item(
    key = animal[["value"]],
    value = animal[["value"]],
    startContent = avatar(src = animal[["avatar"]]),
    animal[["label"]]
```

```
  )
})

label_placements <- c(
  "inside",
  "outside",
  "outside-left"
)

ui <- nextui_page(
  debug_react = TRUE,
  p(class = "font-extrabold text-2xl uppercase", "Basic select"),
  action_button("update", "Update to bulbasaur?"),
  spacer(y = 2),
  action_button("toggle", "Open select"),
  spacer(y = 2),
  select_input(
    inputId = "select",
    label = "Select an pokemon",
    value = JS("['pikachu']"),
    selectionMode = "multiple",
    description = "This is a select input. You can select multiple values.",
    select_items
  ),
  textOutput("select_val"),
  spacer(y = 5),
  divider(),
  p(class = "font-extrabold text-2xl uppercase", "Variants"),
  lapply(select_variants, function(variant) {
    tagList(
      select_input(
        inputId = sprintf("select-%s", variant),
        label = "Select a pokemon",
        variant = variant,
        value = JS("['pikachu']"),
        description = sprintf("This is a select input with %s variant style", variant),
        select_items
      ),
      spacer(y = 2)
    )
  }),
  spacer(y = 5),
  divider(),
  p(
    class = "font-extrabold text-2xl uppercase",
    "Label placement and validation (no value specified)"
  ),
  lapply(label_placements, function(placement) {
    tagList(
      select_input(
        inputId = sprintf("select-%s", placement),
        label = "Select a pokemon",
        labelPlacement = placement,
```

```
        description = sprintf("This is a select input with %s label placement", placement),
          select_items
        ),
        spacer(y = 10)
      )
    }),
    spacer(y = 5),
    divider(),
    p(
      class = "font-extrabold text-2xl uppercase",
      "Custom render value"
    ),
    select_input(
      inputId = "customselect",
      labelPlacement = "outside-left",
      label = "Pokemon",
      description = "This is a select input. You can select multiple values.",
      items = jsonlite::toJSON(animals),
      select_items
    )
  )

  server <- function(input, output, session) {
    opened <- reactiveVal(FALSE)
    observeEvent(input$update, {
      update_select_input(session, "select", value = JS("['bulbasaur']"))
    })
    observeEvent(input$toggle, {
      opened(!opened())
      update_select_input(session, "select", isOpen = opened())
    })
    output$select_val <- renderText(input$select)
  }

  if (interactive() || is_testing()) shinyApp(ui, server)
```

`sizes` *Available sizes*

Description

Available sizes

Available colors

Available radiuses

Available tabs variants

Available select variants

Usage

```
sizes

colors

radiuses

tabs_variants

select_variants
```

Format

An object of class `character` of length 3.

An object of class `character` of length 6.

An object of class `character` of length 5.

An object of class `character` of length 4.

An object of class `character` of length 4.

`skeleton` *skeleton*

Description

Skeleton is a placeholder to show a loading state and the expected shape of a component.

Usage

```
skeleton(...)
```

Arguments

`...`	Props to pass to the component. The allowed props are listed below in the **Details** section.

Details

- **children**. Type: `ReactNode`. Default: `NA`.
- **isLoaded**. Type: `boolean`. Default: `false`.
- **disableAnimation**. Type: `boolean`. Default: `false`.
- **classNames**. Type: `Partial<Record<"base" OR "content", string>>`. Default: `NA`.

Value

An object of class `shiny.tag` containing the necessary configuration and including options such as JavaScript dependencies to instantiate a HeroUI skeleton component.

See Also

See https://heroui.com/docs/components/skeleton.

Examples

```
library(shiny)
library(shinyNextUI)
library(shiny.react)

ui <- nextui_page(
  debug_react = TRUE,
  reactOutput("skeleton_card")
)

server <- function(input, output, session) {

  is_loaded <- reactiveVal(FALSE)
  observeEvent(req(!is_loaded()), {
    Sys.sleep(4)
    is_loaded(TRUE)
  })

  output$skeleton_card <- renderReact({
    card(
      card_header(
        skeleton(
          isLoaded = is_loaded(),
          user(
            name = "Jane Doe",
            description = "Product Designer",
            avatarProps = JS("{
            src: 'https://i.pravatar.cc/150?u=a04258114e29026702d'
          }")
          )
        )
      ),
      card_body(skeleton("Hello World", isLoaded = is_loaded())),
      card_footer(skeleton("Footer", isLoaded = is_loaded()))
    )
  })
}

if (interactive() || is_testing()) shinyApp(ui, server)
```

`slider` *slider*

Description

A slider allows a user to select one or more values within a range.

Usage

```
slider_input(inputId, ..., value = default_value)

update_slider_input(session = shiny::getDefaultReactiveDomain(), inputId, ...)
```

Arguments

inputId	ID of the component.
...	Props to pass to the component. The allowed props are listed below in the **Details** section.
value	Starting value.
session	Object passed as the session argument to Shiny server.

Details

- **label**. Type: ReactNode. Default: NA.
- **name**. Type: string. Default: NA.
- **size**. Type: sm OR md OR lg. Default: "md".
- **color**. Type: foreground OR primary OR secondary OR success OR warning OR danger. Default: "primary".
- **radius**. Type: none OR sm OR md OR lg OR full. Default: "full".
- **step**. Type: number. Default: "1".
- **value**. Type: number. Default: NA.
- **defaultValue**. Type: number. Default: NA.
- **minValue**. Type: number. Default: "0".
- **maxValue**. Type: number. Default: "100".
- **orientation**. Type: horizontal OR vertical. Default: "horizontal".
- **fillOffset**. Type: number. Default: NA.
- **showSteps**. Type: boolean. Default: false.
- **showTooltip**. Type: boolean. Default: false.
- **marks**. Type: SliderStepMarks. Default: NA.
- **startContent**. Type: ReactNode. Default: NA.
- **endContent**. Type: ReactNode. Default: NA.
- **formatOptions**. Type: Intl.NumberFormat. Default: NA.
- **tooltipValueFormatOptions**. Type: Intl.NumberFormat. Default: NA.
- **tooltipProps**. Type: TooltipProps. Default: NA.
- **showOutline**. Type: boolean. Default: false.
- **hideValue**. Type: boolean. Default: false.
- **hideThumb**. Type: boolean. Default: false.
- **disableThumbScale**. Type: boolean. Default: false.

- **isDisabled**. Type: `boolean`. Default: `false`.
- **disableAnimation**. Type: `boolean`. Default: `false`.
- **getValue**. Type: `(value: SliderValue) => string`. Default: `NA`.
- **renderLabel**. Type: `(props: DOMAttributes<HTMLLabelElement>) => ReactNode`. Default: `NA`.
- **renderValue**. Type: `(props: DOMAttributes<HTMLOutputElement>) => ReactNode`. Default: `NA`.
- **renderThumb**. Type: `(props: DOMAttributes<HTMLDivElement> & {index?: number}) => ReactNode`. Default: `NA`.
- **onChange**. Type: `(value: SliderValue) => void`. Default: `NA`.
- **onChangeEnd**. Type: `(value: SliderValue) => void`. Default: `NA`.

Value

An object of class `shiny.tag` containing the necessary configuration and including options such as JavaScript dependencies to instantiate a HeroUI slider component.

See Also

See https://heroui.com/docs/components/slider.

Examples

```
library(shiny)
library(shinyNextUI)
library(shiny.react)

ui <- nextui_page(
  debug_react = TRUE,
  action_button("update", "Update slider 1"),
  slider_input(
    "slider",
    label = "My slider",
    showTooltip = TRUE,
    radius = "none",
    step = 1,
    maxValue = 10,
    minValue = 0,
    value = 5,
    className = "max-w-md",
    showSteps = TRUE,
    color = "foreground",
    size = "sm",
    marks = JS(
      "
      [
        {
          value: 2,
```

```
        label: 'First mark',
      },
      {
        value: 5,
        label: 'Second mark',
      },
      {
        value: 8,
        label: 'Third mark',
      },
    ]
  "
  )
),
spacer(y = 10),
slider_input(
  "slider2",
  radius = "lg",
  label = "Color offset",
  size = "sm",
  showTooltip = TRUE,
  color = "warning",
  maxValue = 5,
  minValue = -5,
  fillOffset = 0,
  value = 1,
  formatOptions = JS("{signDisplay: 'always'}")
),
spacer(y = 10),
slider_input(
  "slider3",
  label = "Outline",
  color = "foreground",
  showOutline = TRUE,
  minValue = 0,
  maxValue = 10,
  value = 5
),
spacer(y = 10),
slider_input(
  "slider4",
  label = "With start and end content",
  minValue = 0,
  maxValue = 10,
  value = 5,
  startContent = icon("volume-xmark"),
  endContent = icon("volume-high")
),
spacer(y = 10),
slider_input(
  "slider5",
  color = "success",
  step = 0.1,
```

```
    label = "Format value with getValue",
    getValue = JS("(val) => `${val} / 10`"),
    maxValue = 10,
    minValue = 0,
    value = 5
  ),
  spacer(y = 10),
  slider_input(
    "range",
    label = "Range",
    color = "danger",
    minValue = 0,
    maxValue = 10,
    value = c(1, 4)
  ),
  spacer(y = 10),
  div(
    class = "flex flex-row  max-w-md h-[348px] gap-6 w-full",
    slider_input(
      "slider4",
      label = "Vertical",
      size = "sm",
      orientation = "vertical",
      minValue = 0,
      maxValue = 10,
      value = 5,
      startContent = icon("volume-high"),
      endContent = icon("volume-xmark")
    )
  )
)

server <- function(input, output, session) {
  observeEvent(input$update, {
    update_slider_input(session, inputId = "slider", value = 10)
  })

  observeEvent(input$slider, {
    print(class(input$slider))
    print(sprintf("Slider is: %s", input$slider))
  })
  observeEvent(input$range, {
    print(input$range)
  })
}

if (interactive() || is_testing()) shinyApp(ui, server)
```

---

snippet *snippet*

Description

Snippet is a component that can be used to display inline or multiline code snippets.

Usage

```
snippet(...)
```

Arguments

...	Props to pass to the component. The allowed props are listed below in the **Details** section.

Details

- **children**. Type: `ReactNode OR ReactNode[]`. Default: `NA`.
- **size**. Type: `sm OR md OR lg`. Default: `"md"`.
- **radius**. Type: `none OR sm OR md OR lg`. Default: `"lg"`.
- **symbol**. Type: `string OR ReactNode`. Default: `"$"`.
- **timeout**. Type: `number`. Default: `"2000"`.
- **codeString**. Type: `string`. Default: `NA`.
- **tooltipProps**. Type: `TooltipProps`. Default: `NA`.
- **copyIcon**. Type: `ReactNode`. Default: `NA`.
- **checkIcon**. Type: `ReactNode`. Default: `NA`.
- **disableTooltip**. Type: `boolean`. Default: `false`.
- **disableCopy**. Type: `boolean`. Default: `false`.
- **hideCopyButton**. Type: `boolean`. Default: `false`.
- **hideSymbol**. Type: `boolean`. Default: `false`.
- **copyButtonProps**. Type: `ButtonProps`. Default: `NA`.
- **disableAnimation**. Type: `boolean`. Default: `false`.
- **classNames**. Type: `Partial<Record<'base' OR 'content' OR 'pre' OR 'symbol' OR 'copyButton' OR 'checkIcon', string>>`. Default: `NA`.
- **onCopy**. Type: `(value: string OR string[]) => void`. Default: `NA`.

Value

An object of class `shiny.tag` containing the necessary configuration and including options such as JavaScript dependencies to instantiate a HeroUI snippet component.

See Also

See https://heroui.com/docs/components/snippet.

Examples

```
library(shiny)
library(shinyNextUI)
library(shiny.react)

ui <- nextui_page(
  debug_react = TRUE,
  snippet("npm install @nextui-org/react", variant = "bordered"),
  spacer(y = 2),
  snippet("you can't copy me", hideCopyButton = TRUE),
  spacer(y = 2),
  snippet(
    color = "secondary",
    variant = "flat",
    span("npm install @nextui-org/react"),
    span("yarn add @nextui-org/react")
  )
)

server <- function(input, output, session) {

}

if (interactive() || is_testing()) shinyApp(ui, server)
```

---

spacer *spacer*

Description

Spacer is a component used to add space between components.

Usage

```
spacer(...)
```

Arguments

...	Props to pass to the component. The allowed props are listed below in the **Details** section.

Details

- **x**. Type: Space. Default: "1".
- **y**. Type: Space. Default: "1".

Value

An object of class shiny.tag containing the necessary configuration and including options such as JavaScript dependencies to instantiate a HeroUI spacer component.

See Also

See https://heroui.com/docs/layout/spacer.

Examples

```
library(shiny)
library(shinyNextUI)
library(shiny.react)

ui <- nextui_page(
  card(card_body("Card 1")),
  spacer(y = 10),
  card(card_body("Card 2")),
  spacer(y = 2),
  card(card_body("Card 3")),
  spacer(y = 10),
  div(
    class = "flex",
    card(card_body("Card 4")),
    spacer(x = 5),
    card(card_body("Card 5")),
    spacer(x = 1),
    card(card_body("Card 6"))
  )
)

server <- function(input, output, session) {}

if (interactive() || is_testing()) shinyApp(ui, server)
```

`switch` *switch*

Description

The Switch component is used as an alternative between checked and not checked states.

Usage

```
switch_input(inputId, ..., value = default_value)

update_switch_input(session = shiny::getDefaultReactiveDomain(), inputId, ...)
```

Arguments

`inputId`	ID of the component.
`...`	Props to pass to the component. The allowed props are listed below in the **Details** section.
`value`	Starting value.
`session`	Object passed as the `session` argument to Shiny server.

Details

- **children**. Type: ReactNode. Default: NA.
- **value**. Type: string. Default: NA.
- **name**. Type: string. Default: NA.
- **size**. Type: sm OR md OR lg. Default: "md".
- **color**. Type: default OR primary OR secondary OR success OR warning OR danger. Default: "primary".
- **thumbIcon**. Type: ThumbIconProps. Default: NA.
- **startContent**. Type: ReactNode. Default: NA.
- **endContent**. Type: ReactNode. Default: NA.
- **isSelected**. Type: boolean. Default: NA.
- **defaultSelected**. Type: boolean. Default: NA.
- **isReadOnly**. Type: boolean. Default: NA.
- **isDisabled**. Type: boolean. Default: false.
- **disableAnimation**. Type: boolean. Default: false.
- **classNames**. Type: Partial<Record<"base"OR "wrapper"OR "thumb"OR "label" OR "startContent" OR "endContent" OR "thumbIcon" , string>>. Default: NA.
- **onChange**. Type: React.ChangeEvent<HTMLInputElement>. Default: NA.
- **onValueChange**. Type: (isSelected: boolean) => void. Default: NA.

Value

An object of class shiny.tag containing the necessary configuration and including options such as JavaScript dependencies to instantiate a HeroUI switch component.

See Also

See https://heroui.com/docs/components/switch.

Examples

```
library(shiny)
library(shinyNextUI)
library(shiny.react)

base_style <- paste(
  "inline-flex flex-row-reverse w-full max-w-md bg-content1 hover:bg-content2",
  "items-center justify-between cursor-pointer rounded-lg gap-2 p-4 border-2",
  "border-transparent data-[selected=true]:border-primary"
)

thumb_style <- paste(
  "w-6 h-6 border-2 shadow-lg group-data-[hover=true]:border-primary",
  "group-data-[selected=true]:ml-6 group-data-[pressed=true]:w-7",
  "group-data-[selected]:group-data-[pressed]:ml-4"
```

```
)

ui <- nextui_page(
  div(
    class = "flex flex-col",
    action_button("update", "Toggle switch"),
    spacer(y = 2),
    p("Basic"),
    switch_input(
      inputId = "switch",
      value = TRUE,
      size = "xs"
    ),
    textOutput("switch_val"),
    spacer(y = 5),
    p("Custom style"),
    spacer(y = 2),
    switch_input(
      "custom_switch",
      classNames = JS(
        sprintf(
          "{
          base: '%s',
          wrapper: 'p-0 h-4 overflow-visible',
          thumb: '%s'
        }",
          base_style,
          thumb_style
        )
      ),
      div(
        className = "flex flex-col gap-1",
        p(className = "text-medium", "Enable early acces"),
        p(
          className = "text-tiny text-default-400",
          "Get access to new features before they are released."
        )
      )
    )
  )
)

server <- function(input, output, session) {
  output$switch_val <- renderText(input$switch)
  observeEvent(input$update, {
    update_switch_input(session, "switch", value = !input$switch)
  })
}

if (interactive() || is_testing()) shinyApp(ui, server)
```

tabs *tabs*

Description

Tabs organize content into multiple sections and allow users to navigate between them.

Usage

```
tabs(inputId, ..., value = default_value)

update_tabs(session = shiny::getDefaultReactiveDomain(), inputId, ...)

tab(...)
```

Arguments

inputId	ID of the component.
...	Props to pass to the component. The allowed props are listed below in the **Details** section.
value	Starting value.
session	Object passed as the session argument to Shiny server.

Details

- **children***. Type: ReactNode OR ((item: T) => ReactElement). Default: NA.
- **variant**. Type: solid OR bordered OR light OR underlined. Default: "solid".
- **color**. Type: default OR primary OR secondary OR success OR warning OR danger. Default: "default".
- **size**. Type: sm OR md OR lg. Default: "md".
- **radius**. Type: none OR sm OR md OR lg OR full. Default: NA.
- **fullWidth**. Type: boolean. Default: false.
- **items**. Type: Iterable<T>. Default: NA.
- **disabledKeys**. Type: React.Key[]. Default: NA.
- **selectedKey**. Type: React.Key. Default: NA.
- **defaultSelectedKey**. Type: React.Key. Default: NA.
- **shouldSelectOnPressUp**. Type: boolean. Default: true.
- **keyboardActivation**. Type: automatic OR manual. Default: "automatic".
- **motionProps**. Type: MotionProps. Default: NA.
- **disableCursorAnimation**. Type: boolean. Default: false.
- **isDisabled**. Type: boolean. Default: false.
- **disableAnimation**. Type: boolean. Default: false.

- **classNames**. Type: `Partial<Record<"base"OR "tabList"OR "tab"OR "tabContent"OR "cursor" OR "panel" OR "tabWrapper", string>>`. Default: `NA`.
- **placement**. Type: `top OR bottom OR start OR end`. Default: `"top"`.
- **isVertical**. Type: `boolean`. Default: `false`.
- **destroyInactiveTabPanel**. Type: `boolean`. Default: `true`.
- **onSelectionChange**. Type: `(key: React.Key) => any`. Default: `NA`.
- **tabRef**. Type: `RefObject<HTMLButtonElement>`. Default: `NA`.
- **children***. Type: `ReactNode`. Default: `NA`.
- **title**. Type: `ReactNode`. Default: `NA`.
- **titleValue**. Type: `string`. Default: `NA`.
- **href**. Type: `string`. Default: `NA`.
- **target**. Type: `HTMLAttributeAnchorTarget`. Default: `NA`.
- **rel**. Type: `string`. Default: `NA`.
- **download**. Type: `boolean OR string`. Default: `NA`.
- **ping**. Type: `string`. Default: `NA`.
- **referrerPolicy**. Type: `HTMLAttributeReferrerPolicy`. Default: `NA`.
- **shouldSelectOnPressUp**. Type: `boolean`. Default: `NA`.

Value

An object of class `shiny.tag` containing the necessary configuration and including options such as JavaScript dependencies to instantiate a HeroUI tabs component.

See Also

See https://heroui.com/docs/components/tabs.

Examples

```
library(shiny)
library(shinyNextUI)

items <- tagList(
  tab(
    key = 1,
    title = div(
      class = "flex items-center gap-1",
      icon("home"),
      "Tab 1"
    ),
    card(
      card_body(
        "Lorem ipsum dolor sit amet, consectetur adipiscing elit,
          sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
          Ut enim ad minim veniam, quis nostrud exercitation ullamco
```

```
          laboris nisi ut aliquip ex ea commodo consequat."
      )
    )
  ),
  tab(
    key = 2,
    title = "Tab 2",
    card(
      card_body(
        "Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris
          nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in
          reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur."
      )
    )
  ),
  tab(
    key = 3,
    title = "Tab 3",
    card(
      card_body(
        "Excepteur sint occaecat cupidatat non proident, sunt in culpa qui
          officia deserunt mollit anim id est laborum."
      )
    )
  )
)

ui <- nextui_page(
  debug_react = TRUE,
  p(class = "font-extrabold text-2xl uppercase", "Simple tabs"),
  tabs(
    inputId = "tabs1",
    disabledKeys = "2",
    items
  ),
  spacer(y = 2),
  select_input(
    "select",
    label = "Tab to select:",
    value = "1",
    select_item(key = 1, value = "1", "1"),
    select_item(key = 3, value = "3", "3")
  ),
  p("Selected tab is:", textOutput("active_tab", inline = TRUE)),
  spacer(y = 5),
  divider(),
  p(class = "font-extrabold text-2xl uppercase", "Tab size"),
  lapply(sizes, function(size) {
    tagList(
      p(class = "", sprintf("Size: %s", size)),
      tabs(
        inputId = sprintf("tabs-%s", size),
        size = size,
```

```
        items
      )
    )
  }),
  spacer(y = 5),
  divider(),
  p(class = "font-extrabold text-2xl uppercase", "Tab variants"),
  lapply(tabs_variants, function(variant) {
    tagList(
      p(sprintf("Variant: %s", variant)),
      tabs(
        inputId = sprintf("tabs-%s", variant),
        variant = variant,
        items
      )
    )
  }),
  spacer(y = 5),
  divider(),
  p(class = "font-extrabold text-2xl uppercase", "Tab color"),
  lapply(colors, function(color) {
    tagList(
      p(class = "", sprintf("Color: %s", color)),
      tabs(
        inputId = sprintf("tabs-%s", color),
        variant = "bordered",
        color = color,
        items
      )
    )
  })
)

server <- function(input, output, session) {
  output$active_tab <- renderText(input$tabs1)
  observeEvent(input$select, {
    update_tabs(session, inputId = "tabs1", value = input$select)
  })
}

if (interactive() || is_testing()) shinyApp(ui, server)
```

`textarea` *textarea*

Description

Textarea component is a multi-line Input which allows you to write large texts.

Usage

```
textarea_input(inputId, ..., value = default_value)

update_textarea_input(
  session = shiny::getDefaultReactiveDomain(),
  inputId,
  ...
)
```

Arguments

inputId	ID of the component.
...	Props to pass to the component. The allowed props are listed below in the **Details** section.
value	Starting value.
session	Object passed as the session argument to Shiny server.

Details

- **children**. Type: ReactNode. Default: NA.
- **minRows**. Type: number. Default: "3".
- **maxRows**. Type: number. Default: "8".
- **cacheMeasurements**. Type: boolean. Default: false.
- **variant**. Type: flat OR bordered OR faded OR underlined. Default: "flat".
- **color**. Type: default OR primary OR secondary OR success OR warning OR danger. Default: "default".
- **size**. Type: sm OR md OR lg. Default: "md".
- **radius**. Type: none OR sm OR md OR lg OR full. Default: NA.
- **label**. Type: ReactNode. Default: NA.
- **value**. Type: string. Default: NA.
- **defaultValue**. Type: string. Default: NA.
- **placeholder**. Type: string. Default: NA.
- **startContent**. Type: ReactNode. Default: NA.
- **endContent**. Type: ReactNode. Default: NA.
- **description**. Type: ReactNode. Default: NA.
- **errorMessage**. Type: ReactNode OR ((v: ValidationResult) => ReactNode). Default: NA.
- **validate**. Type: (value: string) => ValidationError OR true OR null OR undefined. Default: NA.
- **validationBehavior**. Type: native OR aria. Default: "native".
- **labelPlacement**. Type: inside OR outside OR outside-left. Default: "inside".
- **fullWidth**. Type: boolean. Default: true.

- **isRequired**. Type: `boolean`. Default: `false`.
- **isReadOnly**. Type: `boolean`. Default: `NA`.
- **isDisabled**. Type: `boolean`. Default: `false`.
- **isClearable**. Type: `boolean`. Default: `false`.
- **isInvalid**. Type: `boolean`. Default: `false`.
- **validationState**. Type: `valid` OR `invalid`. Default: `NA`.
- **disableAutosize**. Type: `boolean`. Default: `false`.
- **disableAnimation**. Type: `boolean`. Default: `false`.
- **classNames**. Type: `Partial<Record<"base" OR "label" OR "inputWrapper" OR "innerWrapper" OR "input" OR "description" OR "errorMessage", string>>`. Default: `NA`.
- **onChange**. Type: `React.ChangeEvent<HTMLInputElement>`. Default: `NA`.
- **onValueChange**. Type: `(value: string) => void`. Default: `NA`.
- **onClear**. Type: `() => void`. Default: `NA`.
- **onHeightChange**. Type: `(height: number, meta: { rowHeight: number }) => void`. Default: `NA`.

Value

An object of class `shiny.tag` containing the necessary configuration and including options such as JavaScript dependencies to instantiate a HeroUI textarea component.

See Also

See https://heroui.com/docs/components/textarea.

Examples

```
library(shiny)
library(shinyNextUI)

ui <- nextui_page(
  div(
    class = "flex gap-5",
    action_button("update_text", "Update text"),
    textarea_input(
      inputId = "textarea",
      placeholder = "Enter your amazing ideas.",
      label = "Text area input",
      bordered = TRUE,
      color = "secondary",
      status = "secondary",
      helperColor = "error",
      helperText = "Click on update text"
    )
  ),
  textOutput("textarea_val")
)
```

```
server <- function(input, output, session) {
  output$textarea_val <- renderText(input$textarea)

  observeEvent(input$update_text, {
    update_textarea_input(
      inputId = "textarea",
      value = "Updated value"
    )
  })
}

if (interactive() || is_testing()) shinyApp(ui, server)
```

`theme_switcher` *Theme switcher helper*

Description

Change between light and dark mode

Usage

```
theme_switcher(
  value = TRUE,
  label = "Change theme",
  startContent = sun_icon(),
  endContent = moon_icon()
)
```

Arguments

`value`	Switch status.
`label`	Input label.
`startContent`	Icon when selected.
`endContent`	Icon when not selected.

Value

Object with `shiny.tag` class suitable for use in the UI of a Shiny app.

Examples

```
if (interactive()) {
  library(shiny)
  library(shinyNextUI)
  library(shiny.react)
```

```
  ui <- nextui_page(
    theme_switcher(),
    card(card_body("My card"))
  )

  server <- function(input, output, session) {
    observe({
      print(input$theme)
    })
  }

  shinyApp(ui, server)

}
```

`tooltip` *tooltip*

Description

Tooltips display a brief, informative message that appears when a user interacts with an element.

Usage

```
tooltip(...)
```

Arguments

`...`	Props to pass to the component. The allowed props are listed below in the **Details** section.

Details

- **children***. Type: `ReactNode[]`. Default: `NA`.
- **content**. Type: `ReactNode`. Default: `NA`.
- **size**. Type: `sm` OR `md` OR `lg`. Default: `"md"`.
- **color**. Type: `default` OR `primary` OR `secondary` OR `success` OR `warning` OR `danger`. Default: `"default"`.
- **radius**. Type: `none` OR `sm` OR `md` OR `lg` OR `full`. Default: `"md"`.
- **shadow**. Type: `none` OR `sm` OR `md` OR `lg`. Default: `"sm"`.
- **placement**. Type: `TooltipPlacement`. Default: `"top"`.
- **delay**. Type: `number`. Default: `"0"`.
- **closeDelay**. Type: `number`. Default: `"500"`.
- **isOpen**. Type: `boolean`. Default: `NA`.
- **defaultOpen**. Type: `boolean`. Default: `NA`.

- **offset**. Type: `number`. Default: `"7"`.
- **containerPadding**. Type: `number`. Default: `"12"`.
- **crossOffset**. Type: `number`. Default: `"0"`.
- **showArrow**. Type: `boolean`. Default: `false`.
- **shouldFlip**. Type: `boolean`. Default: `true`.
- **triggerScaleOnOpen**. Type: `boolean`. Default: `true`.
- **isKeyboardDismissDisabled**. Type: `boolean`. Default: `false`.
- **isDismissable**. Type: `boolean`. Default: `false`.
- **shouldCloseOnBlur**. Type: `boolean`. Default: `true`.
- **motionProps**. Type: `MotionProps`. Default: `NA`.
- **portalContainer**. Type: `HTMLElement`. Default: `"document.body"`.
- **updatePositionDeps**. Type: `any[]`. Default: `"[]"`.
- **isDisabled**. Type: `boolean`. Default: `false`.
- **disableAnimation**. Type: `boolean`. Default: `false`.
- **classNames**. Type: `Partial<Record<"base"OR"content", string>>`. Default: `NA`.
- **onOpenChange**. Type: `(isOpen: boolean) => void`. Default: `NA`.
- **shouldCloseOnInteractOutside**. Type: `(e: HTMLElement) => void`. Default: `NA`.
- **onClose**. Type: `() => void`. Default: `NA`.

Value

An object of class `shiny.tag` containing the necessary configuration and including options such as JavaScript dependencies to instantiate a HeroUI tooltip component.

See Also

See https://heroui.com/docs/components/tooltip.

Examples

```
library(shiny)
library(shinyNextUI)
library(shiny.react)

ui <- nextui_page(
  div(
    className = "flex align-items-start",
    tooltip(
      content = "A tooltip...",
      color = "primary",
      showArrow = TRUE,
      closeDelay = 0,
      delay = 0,
      button("Click me!", color = "warning")
    )
```

```
  )
)

server <- function(input, output, session) {
}

if (interactive() || is_testing()) shinyApp(ui, server)
```

`user` *user*

Description

Display user information with avatar and name.

Usage

```
user(...)
```

Arguments

`...`	Props to pass to the component. The allowed props are listed below in the **Details** section.

Details

- **name**. Type: `string`. Default: `NA`.
- **description**. Type: `ReactNode`. Default: `NA`.
- **isFocusable**. Type: `boolean`. Default: `false`.
- **avatarProps**. Type: `AvatarProps`. Default: `NA`.
- **classNames**. Type: `Partial<Record<"base" OR "wrapper" OR "name" OR "description", string>>`. Default: `NA`.

Value

An object of class `shiny.tag` containing the necessary configuration and including options such as JavaScript dependencies to instantiate a HeroUI user component.

See Also

See https://heroui.com/docs/components/user.

Examples

```
library(shiny)
library(shinyNextUI)
library(shiny.react)

ui <- nextui_page(
  div(
    class = "grid gap-4 grid-cols-3 grid-rows-3 m-5",
    user(
      name = "Jane Doe",
      description = "Product Designer",
      avatarProps = JS("{
        src: 'https://i.pravatar.cc/150?u=a04258114e29026702d'
      }")
    )
  )
)

server <- function(input, output, session) {}

if (interactive() || is_testing()) shinyApp(ui, server)
```

Index