Package ‘fomantic.plus’

May 8, 2026

Type Package

Title Add Extra 'Fomantic UI' Components to 'shiny.semantic'

Version 0.1.0

Description Extend 'shiny.semantic' with extra 'Fomantic UI' components.
Create pages in a format similar to 'shiny', form validation and more.

URL https://github.com/ashbaldry/fomantic.plus

BugReports https://github.com/ashbaldry/fomantic.plus/issues

License MIT + file LICENCE

Encoding UTF-8

Imports shiny, shiny.semantic, htmltools, jsonlite

Suggests rmarkdown, knitr, testthat

RoxygenNote 7.1.2

VignetteBuilder knitr

Language en-GB

NeedsCompilation no

Author Ashley Baldry [aut, cre]

Maintainer Ashley Baldry <arbaldry91@gmail.com>

Repository CRAN

Date/Publication 2022-01-24 20:12:52 UTC

Contents

---

addPopup *Create Fomantic UI Popup*

Description

Add a tooltip to an element that on hover will show extra information

`addTooltip` will only use a basic CSS tooltip with a limited amount of functionality, whereas `addPopup` is initialised with JS, and can include more functionality

Usage

```
addPopup(
  el,
  text,
  position = NULL,
  variation = NULL,
  inverted = FALSE,
  title = NULL,
  offset = NULL,
  settings = NULL,
  html = FALSE
)

addTooltip(el, text, position = NULL, variation = NULL, inverted = FALSE)
```

Arguments

`el`	A UI element that the tooltip will be applied to
`text`	Contents of the tooltip. Can either be a character string or an HTML object
`position`	(Optional) Force the popup to appear in a direction relative to `el`. Choose a vertical position from `"top"`, `"bottom"`, `""` and a horizontal from `"left"`, `"center"`, `"right"`, `""`
`variation`	(Optional) Add certain features to the popup `mini`, `tiny`, `small`, `medium`, `large`, `huge` Affect the size of the font in the popup `basic` Removes the pointing arrow of the popup `fixed`, `wide` **(addPopup only)**, `very wide` **(addPopup only)** Affect the width of the popup

inverted	Should the colours of the popup be inverted?
title	(Optional) Add a title to the popup. Only appears when html = FALSE
offset	(Optional) A numeric value of the number of pixel to offset the tooltip by
settings	Named list of settings to be applied to the popup. Check Fomantic UI website for full list. For example list(on = "click") will mean the popup appears on a click rather than a hover.
html	Is text valid HTML code? Defaults to FALSE

Value

addTooltip will return el with extra attributes added to the top level tag.

addPopup will return a shiny.tag.list, first similar to addTooltip an updated version of el. Then a small JS script has been added to enable the popup.

See Also

https://fomantic-ui.com/modules/popup.html

Examples

```
addPopup(
  fui_el$label(id = "help_label", class = "small circular", "?"),
  "This can be used as a help icon in a shiny app",
  inverted = TRUE
)

addTooltip(
  fui_el$label(class = "small circular", "?"),
  "This can be used as a help icon in a shiny app"
)
```

---

darkmode_toggle *Invert Toggle*

Description

Add a toggle to the shiny application that triggers all Fomantic UI elements to become "inverted"

Usage

```
darkmode_toggle(label = "Dark Mode", ..., checked = FALSE)
```

Arguments

label	Labels to add before and after the toggle. By default "Dark Mode" will appear after the toggle
...	Tag attributes (named arguments) and children (unnamed arguments)
checked	Should the application start off in dark mode?

Details

To prevent elements from becoming inverted/removing their inverted state, include `keep-inverted-state` to maintain them in either standard or inverted.

Value

A `shiny.tag` that will provide a toggle style checkbox in the UI of a shiny application.

Examples

```
if (interactive()) {
  library(shiny)
  library(shiny.semantic)
  ui <- semanticPage(
    extendShinySemantic(),
    fui_el$grid(
      fui_el$row(
        class = "two column",
        fui_el$column(
          fui_el$segment(
            class = "purple",
            darkmode_toggle()
          )
        ),
        fui_el$column(
          fui_el$segment(
            class = "red keep-inverted-state"
          )
        )
      )
    ),
    fui_el$cards(
      class = "two",
      fui_el$card(),
      fui_el$card()
    )
  )

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

  shiny::shinyApp(ui, server)
}
```

`extendShinySemantic` *Add* `fomantic.plus` *Dependencies to* `shiny.semantic` *Application*

Description

In order for any of the `fomantic.plus` functionality to work

This will be automatically included in any `xxx_page` function in this package, for example `navbar_page`.

Usage

```
extendShinySemantic()
```

Value

A `shiny.tag.list` containing tags to enable the JS and CSS required for this package.

Examples

```
if (interactive()) {
  library(shiny)
  library(shiny.semantic)
  library(fomantic.plus)

  ui <- semanticPage(
    title = "Hello Fomantic UI",
    tags$head(
      extendShinySemantic()
    )
  )
}
```

---

field_validation *Field Validation for Fomantic UI*

Description

A field validation assigns a series of rules that have been assigned to a particular input and checks, upon the form submission, whether or not the input meets all specified criteria.

Usage

```
field_validation(id, ..., extra_params = NULL)

field_rule(rule, prompt = NULL, value = NULL)
```

Arguments

`id`	HTML id of the field to be validated
`...`	A series of `field_rules` that will be applied to the field
`extra_params`	A named list of extra parameters that can be added to the field validation. For example `optional = TRUE` means the field will only be checked if non-empty

`rule`	The type of rule to be applied. Valid rules are available in Details.
`prompt`	Text to be displayed in the UI if the validation fails. Leave `NULL` if want to use default text.
`value`	Certain fields require a value to check validation. Check Details if the `rule` requires a value.

Details

If it fails, then the field will be highlighted and the failures will either be specified as a message below the field or a label. Once the failure(s) has been rectified, the highlighting will disappear.

The following `rules` are allowed:

`empty` A field is not empty

`checked` A checkbox field is checked

`email` A field is a valid e-mail address

`url` A field is a url

`integer` A field is an integer value or matches an integer range*

`decimal` A field must be a decimal number or matches a decimal range*

`number` A field is any number or matches a number range*

`regExp` Matches against a regular expression

`creditCard` A field is a valid credit card**

`contains`, `doesntContain` A field (doesn't) contain text (case insensitive)

`containsExactly`, `doesntContainExactly` A field (doesn't) contain text (case sensitive)

`is`, `not` A field is (not) a value (case insensitive)

`isExactly`, `notExactly` A field is (not) a value (case sensitive)

`minLength`, `exactLength`, `maxLength` A field is at least/exactly/at most a set length

`match`, `different` A field should (not) match the value of another validation field. Use the field ID as the value

`minCount`, `exactCount`, `maxCount` A multiple select field contains at least/exactly/at most a set number of selections

* For ranges, include the parameter `value = "x..y"` where `x` is the minimum value and `y` is the maximum value. Leave either side blank to not have a lower/upper limit

** Include comma separated string of card providers if required e.g. `value = "visa,mastercard"`

Value

A structured list of the `field_rules` that can be recognised by `form_validation`.

References

https://fomantic-ui.com/behaviors/form.html

See Also

form_validation

Examples

```
# E-mail validations
field_validation("email", field_rule("email"))

# Password validation
field_validation(
  "password",
  field_rule("empty"),
  field_rule("minLength", value = 8),
  field_rule("regExp", "Must contain at least one special character", "\\W")
)
```

---

form_button *Fomantic UI Button*

Description

Creates a button specifically for Fomantic UI forms in order to check all inputs meet validation rules

Usage

```
form_button(input_id, label, icon = NULL, width = NULL, ...)
```

Arguments

input_id	The input slot that will be used to access the value
label	The contents of the button, can either be character string or HTML tags
icon	An optional icon to appear on the button
width	Width of the input
...	Named attributes to be applied to the button or remaining parameters passed to button, like class

Value

A shiny.tag that will show a submit button in the UI of a shiny application.

See Also

form_validation, action_button

Examples

```
form_button("submit", "Submit")
```

form_validation *Form Validation for Fomantic UI*

Description

A form validation behaviour checks data against a set of criteria before passing it along to the server.

Usage

```
form_validation(
  id,
  ...,
  submit_label = "Submit",
  submit_class = "",
  include_button = TRUE,
  inline = FALSE
)
```

Arguments

id	ID of the parent form
...	A series of field_validation whose id are inputs contained within the form
submit_label	Label to give the submission button at the end of the form (included in returned UI with input value {id}_submit)
submit_class	Additional classes to give the submission button
include_button	Logical, should the submit button be included? Defaults to TRUE. If FALSE, a action_button will be required in the form somewhere with "submit form-button" included as part of the class in order for the validation to run.
inline	Logical, do you want the field validation errors as in-line labels (TRUE), or in a message box at the bottom of the form (FALSE)?

Details

In order for the validation to work, the form_validation must be a direct child of the form.

The "Submit" button has an input value of {id}_submit and will only trigger server-side events if all the fields pass validation.

NB If you do not include either form validation input as part of the server-side code then the inputs will pass through to the server as if there were no validation.

Value

A shiny.tag.list containing the inline JS to perform the form validation in the shiny UI.

If include_button = TRUE then a button will also be included to appear in the UI.

References

https://fomantic-ui.com/behaviors/form.html

See Also

field_validation, form_button

Examples

```
if (interactive()) {
  library(shiny)
  library(shiny.semantic)
  library(fomantic.plus)

  ui <- semanticPage(
    tags$head(
      extendShinySemantic()
    ),
    form(
      id = "form",
      field(
        tags$label("Name"),
        text_input("name")
      ),
      field(
        tags$label("E-Mail"),
        text_input("email")
      ),
      form_validation(
        id = "form",
        field_validation("name", field_rule("empty")),
        field_validation("email", field_rule("empty"), field_rule("email"))
      )
    )
  )

  server <- function(input, output) {
  }

  shinyApp(ui, server)
}
```

---

fui_el *Create Fomantic UI Elements*

Description

Create an R object that represents a Fomantic UI Element e.g. segment or container. The contents have remained as minimal as possible to enable the greatest possible flexibility.

Usage

```
fui_el
```

Format

An object of class `list` of length 41.

Details

Most of the elements work just like a standard HTML tag with some pre-defined classes, however there are a few elements which require a value, and so have an extra argument attached:

`emoji` FUI Element: `emoji` - The string of the emoji name

`country` FUI Element: `flag` - Either the country name or 2 character ISO code

`icon` FUI Element: `icon` - The space separated name of the Font Awesome icon

`html_tag` FUI Elements: `header`, `list`, `item` - For certain elements, multiple HTML tags can be used. The default is set to `div`, but can be set to any valid HTML tag.

See Also

https://fomantic-ui.com for styling Fomantic UI elements, builder

Examples

```
# List
fui_el$list(
  fui_el$item("Item 1"),
  fui_el$item("Item 2"),
  fui_el$item("Item 3")
)

# Pink Segment
fui_el$segment(
  class = "pink"
)

# Grid
fui_el$grid(
  fui_el$row(
    class = "two column",
    fui_el$column(),
    fui_el$column()
  )
)

# Flag
fui_el$flag("fr")

# Icon
fui_el$icon("exclamation triangle")
```

navbar_menu *Navbar Menu*

Description

Create a dropdown menu for a navbar_page.

Usage

```
navbar_menu(title, ..., id = title, icon = NULL)
```

Arguments

title	Display title for menu
...	tab_panel elements to include in the page. Can also include strings as section headers, or "—-" as a horizontal separator.
id	The ID of the navbar_menu
icon	Optional icon to appear on the tab. This attribute is only valid when using a tab_panel within a navbar_page.

Value

A structured list of class ssnavmenu, that can be used in navbar_page.

Examples

```
navbar_menu(
  "Menu",
  tab_panel("Summary", shiny::plotOutput("plot")),
  "----",
  "Section header",
  tab_panel("Table", shiny::tableOutput("table"))
)
```

navbar_page *Fomantic UI page with top level navigation bar*

Description

This creates a Fomantic page for use in a Shiny app. It is in the same layout as navbarPage, where a top level navigation bar exists.

Usage

```
navbar_page(
  ...,
  title = "",
  id = NULL,
  selected = NULL,
  position = c("", "top fixed", "bottom fixed"),
  head = NULL,
  header = NULL,
  footer = NULL,
  collapsible = FALSE,
  window_title = title,
  class = "stackable",
  theme = NULL,
  enable_hash_state = TRUE,
  suppress_bootstrap = TRUE
)
```

Arguments

...	Other arguments to be added as attributes of the main div tag wrapper (e.g. style, class etc.)
title	A title to display in the navbar.
id	ID of the navbar menu. Given random ID if none specified.
selected	Which tab should be selected first? If none selected, will automatically have the first tab open.
position	Determines the location and behaviour of the navbar. Padding will be included when pinned to prevent overlap. • ""Default. Top of page, and goes out of view when scrolling • "top fixed"Top of page, pinned when scrolling • "bottom fixed"Bottom of page, pinned when scrolling
head	Optional list of tags to be added to tags$head.
header	Optional list of tags to be added to the top of all tab_panels.
footer	Optional list of tags to be added to the bottom of all tab_panels.
collapsible	TRUE to automatically collapse the navigation elements into a menu when the width of the browser is less than 768 pixels (useful for viewing on smaller touch-screen device)
window_title	A title to display in the browser's title bar. By default it will be the same as the navbar title.
class	Additional classes to be given to the navbar menu. Defaults to "stackable". For optional classes have a look in details
theme	Theme name or path. Full list of supported themes you will find in SUPPORTED_THEMES or at https://semantic-ui-forest.com/themes.

`enable_hash_state`
: boolean flag that enables a different hash in the URL for each tab, and creates historical events

`suppress_bootstrap`
: boolean flag that suppresses bootstrap when turned on

Details

Inside, it uses two crucial options:

(1) `shiny.minified` with a logical value, tells whether it should attach min or full semantic css or js (TRUE by default). (2) `shiny.custom.semantic` if this option has not NULL character `semanticPage` takes dependencies from custom css and js files specified in this path (NULL by default). Depending on `shiny.minified` value the folder should contain either "min" or standard version. The folder should contain: `semantic.css` and `semantic.js` files, or `semantic.min.css` and `semantic.min.js` in `shiny.minified = TRUE` mode.

The following classes can be applied to the navbar:

- `stackable` - When the width of the webpage becomes too thin, for example on mobile, the navbar will become a stack
- `inverted` - Will create an inverted coloured navbar

Value

A `shiny.tag.list` containing the UI for a shiny application.

Examples

```
navbar_page(
  title = "App Title",
  tab_panel("Plot"),
  tab_panel("Summary"),
  tab_panel("Table")
)

navbar_page(
  title = "App Title",
  tab_panel("Plot"),
  tab_panel("Icon", icon = "r project"),
  navbar_menu(
    "More",
    tab_panel("Summary"),
    "----",
    "Section header",
    tab_panel("Table")
  )
)
```

`runFPlusExample` *Run Fomantic Plus Examples*

Description

Run Fomantic Plus Examples

Usage

```
runFPlusExample(
  example = NA,
  port = getOption("shiny.port"),
  launch.browser = getOption("shiny.launch.browser", interactive()),
  host = getOption("shiny.host", "127.0.0.1"),
  display.mode = c("auto", "normal", "showcase")
)
```

Arguments

`example`	The name of the example to run, or `NA` (the default) to list the available examples.
`port`	The TCP port that the application should listen on. If the port is not specified, and the shiny.port option is set (with options(shiny.port = XX)), then that port will be used. Otherwise, use a random port between 3000:8000, excluding ports that are blocked by Google Chrome for being considered unsafe: 3659, 4045, 5060, 5061, 6000, 6566, 6665:6669 and 6697. Up to twenty random ports will be tried.
`launch.browser`	If true, the system's default web browser will be launched automatically after the app is started. Defaults to true in interactive sessions only.
`host`	The IPv4 address that the application should listen on. Defaults to the shiny.host option, if set, or "127.0.0.1" if not.
`display.mode`	The mode in which to display the example. Defaults to `showcase`, but may be set to normal to see the example without code or commentary.

Value

If `example = NA` then a list of the available examples will be shown, otherwise the selected application will be rendered.

See Also

`runExample`

Examples

```
if (interactive()) {
  runFPlusExample()

  # Fomantic UI Kitchen Sink
  runKitchenSink()
}
```

---

show_tab *Show/Hide Tab*

Description

Dynamically show or hide a tab_panel or navbar_menu

Usage

```
show_tab(session = shiny::getDefaultReactiveDomain(), id, target)

hide_tab(session = shiny::getDefaultReactiveDomain(), id, target)
```

Arguments

session	The session object passed to function given to shinyServer.
id	The id of the navbar object
target	The tab value to toggle visibility

Value

Changes to the visibility of a tab in the shiny UI.

Examples

```
if (interactive()) {
  library(shiny)
  library(shiny.semantic)

  ui <- navbar_page(
    title = "App Title",
    id = "navbar",
    tab_panel(
      "Plot",
      action_button("hide", "Hide Table"),
      action_button("show", "Show Table"),
      value = "plot"
    ),
    tab_panel("Summary", value = "summary"),
```

```
    tab_panel("Table", value = "table")
  )

  server <- function(input, output, session) {
    observeEvent(input$hide, hide_tab(session, "navbar", "table"))
    observeEvent(input$show, show_tab(session, "navbar", "table"))
  }

  shinyApp(ui, server)
}
```

---

tab_panel *Tab Panel*

Description

Create a tab panel

Usage

```
tab_panel(
  title,
  ...,
  value = title,
  icon = NULL,
  type = "bottom attached segment"
)
```

Arguments

title	Display title for tab
...	UI elements to include within the tab
value	The value that should be sent when navbar_menu reports that this tab is selected. If omitted and navbar_menu has an id, then the title will be used.
icon	Optional icon to appear on the tab. This attribute is only valid when using a tab_panel within a navbar_page.
type	Change depending what type of tab is wanted. Default is bottom attached segment.

Value

A tab that can be passed to navbar_menu.

See Also

navbar_menu

Examples

```
navbar_menu(
  tab_panel("Plot", shiny::plotOutput("plot")),
  tab_panel("Summary", shiny::verbatimTextOutput("summary")),
  tab_panel("Table", shiny::tableOutput("table"))
)
```

Index