

Package: shinyMobile (via r-universe)

September 15, 2024

Type Package

Title Mobile Ready 'shiny' Apps with Standalone Capabilities

Version 2.0.1

Maintainer David Granjon <dgranjon@gmail.com>

Description Develop outstanding 'shiny' apps for 'iOS' and 'Android' as well as beautiful 'shiny' gadgets. 'shinyMobile' is built on top of the latest 'Framework7' template <<https://framework7.io>>. Discover 14 new input widgets (sliders, vertical sliders, stepper, grouped action buttons, toggles, picker, smart select, ...), 2 themes (light and dark), 12 new widgets (expandable cards, badges, chips, timelines, gauges, progress bars, ...) combined with the power of server-side notifications such as alerts, modals, toasts, action sheets, sheets (and more) as well as 3 layouts (single, tabs and split).

Imports shiny, htmltools, jsonlite, magrittr, gplots, lifecycle

License GPL-2

Encoding UTF-8

URL <https://github.com/RinterFace/shinyMobile>,
<https://rinterface.github.io/shinyMobile/>

BugReports <https://github.com/RinterFace/shinyMobile/issues>

RoxygenNote 7.3.1

Suggests knitr, rmarkdown, stats, cli, testthat (>= 2.1.0),
rstudioapi, shinyWidgets, apexcharter, ggplot2, dplyr, bslib,
shinytest2, thematic

VignetteBuilder knitr

Roxygen list(markdown = TRUE)

Config/testthat/parallel true

Config/testthat/edition 3

Repository <https://rinterface.r-universe.dev>

RemoteUrl <https://github.com/rinterface/shinyMobile>

RemoteRef HEAD

RemoteSha 383e865afa8ac7f2ffded3c6e3f37269d829d1e6

Contents

| | |
|--------------------------------|----|
| addF7Popover | 4 |
| add_shinyMobile_deps | 6 |
| f7Accordion | 6 |
| f7ActionSheet | 8 |
| f7Align | 10 |
| f7AutoComplete | 11 |
| f7Back | 14 |
| f7Badge | 14 |
| f7Block | 15 |
| f7Button | 18 |
| f7Card | 21 |
| f7Checkbox | 24 |
| f7CheckboxGroup | 25 |
| f7Chip | 27 |
| f7ColorPicker | 29 |
| f7DatePicker | 30 |
| f7DefaultOptions | 33 |
| f7Dialog | 34 |
| f7DownloadButton | 36 |
| f7Fab | 37 |
| f7FabClose | 38 |
| f7Fabs | 38 |
| f7File | 41 |
| f7Float | 42 |
| f7Form | 43 |
| f7Gallery | 45 |
| f7Gauge | 45 |
| f7Grid | 48 |
| f7Icon | 48 |
| f7Item | 49 |
| f7Items | 50 |
| f7Link | 50 |
| f7List | 51 |
| f7ListGroup | 54 |
| f7ListIndex | 54 |
| f7ListItem | 57 |
| f7Login | 58 |
| f7Margin | 60 |
| f7MessageBar | 61 |
| f7Messages | 63 |
| f7MultiLayout | 65 |

| | |
|-----------------|-----|
| f7Navbar | 67 |
| f7Next | 69 |
| f7Notif | 70 |
| f7Padding | 71 |
| f7Page | 72 |
| f7Panel | 74 |
| f7PanelMenu | 76 |
| f7PhotoBrowser | 77 |
| f7Picker | 78 |
| f7Popup | 81 |
| f7Progress | 84 |
| f7Radio | 85 |
| f7Searchbar | 88 |
| f7Select | 90 |
| f7Sheet | 91 |
| f7SingleLayout | 94 |
| f7Skeleton | 95 |
| f7Slider | 97 |
| f7SmartSelect | 100 |
| f7SplitLayout | 102 |
| f7Stepper | 106 |
| f7SubNavbar | 109 |
| f7Swipeout | 110 |
| f7Swiper | 112 |
| f7Tab | 114 |
| f7TabLayout | 115 |
| f7Table | 118 |
| f7TabLink | 119 |
| f7Tabs | 119 |
| f7TapHold | 122 |
| f7Text | 123 |
| f7Timeline | 126 |
| f7Toast | 128 |
| f7Toggle | 130 |
| f7Toolbar | 131 |
| f7Tooltip | 132 |
| f7Treeview | 136 |
| f7TreeviewGroup | 139 |
| f7TreeviewItem | 140 |
| f7VirtualList | 142 |
| getF7Colors | 147 |
| insertF7Tab | 148 |
| preview_mobile | 148 |
| removeF7Tab | 150 |
| showF7Preloader | 150 |
| updateF7App | 152 |
| updateF7Entity | 155 |
| updateF7Routes | 161 |

| | |
|-------------------------------|-----|
| updateF7Tabs | 161 |
| updateF7VirtualList | 162 |
| validateF7Input | 166 |

| | |
|--------------|------------|
| Index | 169 |
|--------------|------------|

| | |
|--------------|-------------------------------|
| addF7Popover | <i>Add Framework7 popover</i> |
|--------------|-------------------------------|

Description

addF7Popover adds a popover to the given target and show it if enabled by [toggleF7Popover](#).
toggleF7Popover toggles the visibility of popover. See example for use case.

Usage

```
addF7Popover(
  id = NULL,
  selector = NULL,
  options,
  session = shiny::getDefaultReactiveDomain()
)
```

```
toggleF7Popover(
  id = NULL,
  selector = NULL,
  session = shiny::getDefaultReactiveDomain()
)
```

Arguments

| | |
|----------|---|
| id | Popover target id. |
| selector | jQuery selector. Allow more customization for the target (nested tags). |
| options | List of options to pass to the popover. See https://framework7.io/docs/popover.html#popover-parameters . |
| session | Shiny session object. |

Examples

```
library(shiny)
library(shinyMobile)

lorem_ipsum <- "Lorem ipsum dolor sit amet,
               consectetur adipiscing elit. Quisque ac diam ac quam euismod
               porta vel a nunc. Quisque sodales scelerisque est, at porta
               justo cursus ac."

popovers <- data.frame(
```

```

    id = paste0("target_", 1:3),
    content = paste("Popover content", 1:3, lorem_ipsum),
    stringsAsFactors = FALSE
  )

app <- shinyApp(
  ui = f7Page(
    title = "f7Popover",
    f7SingleLayout(
      navbar = f7Navbar(
        title = "f7Popover"
      ),
      f7Block(
        f7Toggle(
          inputId = "toggle",
          "Enable popover",
          color = "green",
          checked = TRUE
        )
      ),
      f7Segment(
        lapply(seq_len(nrow(popovers)), function(i) {
          f7Button(
            inputId = sprintf("target_%s", i),
            sprintf("Popover target %s", i)
          )
        })
      )
    ),
    server = function(input, output, session) {
      # Enable/disable (don't run first)
      observeEvent(input$toggle,
        {
          lapply(
            seq_len(nrow(popovers)),
            function(i) toggleF7Popover(id = popovers[i, "id"])
          )
        },
        ignoreInit = TRUE
      )

      # show
      lapply(seq_len(nrow(popovers)), function(i) {
        observeEvent(input[[popovers[i, "id"]]], {
          addF7Popover(
            id = popovers[i, "id"],
            options = list(
              content = popovers[i, "content"]
            )
          )
        })
      })
    }
  )
)

```

```

    }
  )

  if (interactive() || identical(Sys.getenv("TESTTHAT"), "true")) app

```

add_shinyMobile_deps *shinyMobile dependencies utils*

Description

This function attaches shinyMobile dependencies to the given tag

Usage

```
add_shinyMobile_deps(tag)
```

Arguments

tag Element to attach the dependencies.

f7Accordion *Framework7 accordion container*

Description

f7Accordion creates an interactive accordion container.

f7AccordionItem is to be inserted in [f7Accordion](#).

[updateF7Accordion](#) toggles an [f7Accordion](#) on the client.

Usage

```

f7Accordion(
  ...,
  id = NULL,
  multiCollapse = deprecated(),
  side = c("right", "left")
)

f7AccordionItem(..., title = NULL, open = FALSE)

updateF7Accordion(
  id,
  selected = NULL,
  session = shiny::getDefaultReactiveDomain()
)

```

Arguments

| | |
|---------------|---|
| ... | Item content such as f7Block or any f7 element. |
| id | Accordion instance. |
| multiCollapse | [Deprecated] : removed from Framework7. |
| side | Accordion collapse toggle side. Default to right. |
| title | Item title. |
| open | Whether the item is open at start. FALSE by default. |
| selected | Index of item to select. |
| session | Shiny session object |

Author(s)

David Granjon, <dgranjon@gmail.com>

Examples

```
library(shiny)
library(shinyMobile)

app <- shinyApp(
  ui = f7Page(
    title = "Accordions",
    f7SingleLayout(
      navbar = f7Navbar("Accordions"),
      f7Segment(f7Button(inputId = "go", "Go")),
      f7Accordion(
        id = "myaccordion1",
        f7AccordionItem(
          title = "Item 1",
          f7Block("Item 1 content"),
          open = TRUE
        ),
        f7AccordionItem(
          title = "Item 2",
          f7Block("Item 2 content")
        )
      )
    )
  ),
  server = function(input, output, session) {
    observeEvent(input$go, {
      updateF7Accordion(id = "myaccordion1", selected = 2)
    })
  }
)

if (interactive() || identical(Sys.getenv("TESTTHAT"), "true")) app
```

f7ActionSheet *Framework7 action sheet*

Description

f7ActionSheet creates an action sheet may contain multiple buttons. Each of them triggers an action on the server side. It may be updated later by [updateF7ActionSheet](#).

updateF7ActionSheet updates an [f7ActionSheet](#) from the server.

Usage

```
f7ActionSheet(
  id,
  buttons,
  grid = FALSE,
  ...,
  session = shiny::getDefaultReactiveDomain()
)
```

```
updateF7ActionSheet(id, options, session = shiny::getDefaultReactiveDomain())
```

Arguments

id Unique id. This gives the state of the action sheet. `input$id` is TRUE when opened and inversely. Importantly, if the action sheet has never been opened, `input$id` is NULL.

buttons list of buttons such as

```
buttons <- list(
  list(
    text = "Notification",
    icon = f7Icon("info"),
    color = NULL
  ),
  list(
    text = "Dialog",
    icon = f7Icon("lightbulb_fill"),
    color = NULL
  )
)
```

The currently selected button may be accessed via `input$<sheet_id>_button`. The value is numeric. When the action sheet is closed, `input$<sheet_id>_button` is NULL. This is useful when you want to trigger events after a specific button click.

grid Whether to display buttons on a grid. Default to FALSE.

... Other options. See <https://framework7.io/docs/action-sheet#action-sheet-parameters>.

session Shiny session object.
options Other options. See <https://framework7.io/docs/action-sheet#action-sheet-parameters>.

Examples

```
library(shiny)
library(shinyMobile)

sheetModuleUI <- function(id) {
  ns <- NS(id)
  f7Segment(
    f7Button(inputId = ns("go"), label = "Show action sheet", color = "green"),
    f7Button(inputId = ns("update"), label = "Update action sheet", color = "red")
  )
}

sheetModule <- function(id) {
  moduleServer(
    id,
    function(input, output, session) {
      ns <- session$ns

      observeEvent(input$action1_button, {
        if (input$action1_button == 1) {
          f7Notif(
            text = "You clicked on the first button",
            icon = f7Icon("bolt_fill"),
            title = "Notification",
            titleRightText = "now"
          )
        } else if (input$action1_button == 2) {
          f7Dialog(
            id = ns("test"),
            title = "Click me to launch a Toast!",
            type = "confirm",
            text = "You clicked on the second button",
          )
        }
      })

      observeEvent(input$test, {
        f7Toast(text = paste("Alert input is:", input$test))
      })

      observeEvent(input$go, {
        f7ActionSheet(
          grid = TRUE,
          id = ns("action1"),
          buttons = list(
            list(
              text = "Notification",
              icon = f7Icon("info"),
            )
          )
        )
      })
    }
  )
}
```

```

        color = NULL
      ),
      list(
        text = "Dialog",
        icon = f7Icon("lightbulb_fill"),
        color = NULL
      )
    )
  )
}

observeEvent(input$update, {
  updateF7ActionSheet(
    id = "action1",
    options = list(
      grid = TRUE,
      buttons = list(
        list(
          text = "Plop",
          icon = f7Icon("info"),
          color = "orange"
        )
      )
    )
  )
})
}
)
}

app <- shinyApp(
  ui = f7Page(
    title = "Action sheet",
    f7SingleLayout(
      navbar = f7Navbar("Action sheet"),
      br(),
      sheetModuleUI(id = "sheet1")
    )
  ),
  server = function(input, output, session) {
    sheetModule("sheet1")
  }
)

if (interactive() || identical(Sys.getenv("TESTTHAT"), "true")) app

```

f7Align

Framework7 align utility

Description

f7Align is an alignment utility for items.

Usage

```
f7Align(tag, side = c("left", "center", "right", "justify"))
```

Arguments

| | |
|------|--|
| tag | Tag to align. |
| side | Side to align: "left", "center", "right" or "justify". |

Author(s)

David Granjon, <dgranjon@ymail.com>

Examples

```
if(interactive()){
  library(shiny)
  library(shinyMobile)

  shinyApp(
    ui = f7Page(
      title = "Align",
      f7SingleLayout(
        navbar = f7Navbar(title = "f7Align"),
        f7Row(
          f7Align(h1("Left"), side = "left"),
          f7Align(h1("Center"), side = "center"),
          f7Align(h1("Right"), side = "right")
        )
      )
    ),
    server = function(input, output) {}
  )
}
```

f7AutoComplete

Framework7 autocomplete input

Description

f7AutoComplete generates a Framework7 autocomplete input.

updateF7AutoComplete changes the value of an autocomplete input on the client.

Usage

```
f7AutoComplete(
  inputId,
  label = NULL,
  placeholder = NULL,
  value = NULL,
  choices,
  openIn = c("popup", "page", "dropdown"),
  typeahead = TRUE,
  expandInput = deprecated(),
  closeOnSelect = FALSE,
  dropdownPlaceholderText = NULL,
  multiple = FALSE,
  limit = NULL,
  style = list(media = NULL, description = NULL, floating = FALSE, outline = FALSE),
  ...
)

updateF7AutoComplete(
  inputId,
  value = NULL,
  choices = NULL,
  ...,
  session = shiny::getDefaultReactiveDomain()
)
```

Arguments

| | |
|--------------------------------------|--|
| <code>inputId</code> | Autocomplete input id. |
| <code>label</code> | Autocomplete label. |
| <code>placeholder</code> | Text to write in the container. |
| <code>value</code> | Autocomplete initial value, if any. |
| <code>choices</code> | Autocomplete choices. |
| <code>openIn</code> | Defines how to open Autocomplete, can be page or popup (for Standalone) or dropdown. |
| <code>typeahead</code> | Enables type ahead, will prefill input value with first item in match. Only if <code>openIn</code> is "dropdown". |
| <code>expandInput</code> | [Deprecated] : removed from Framework7. |
| <code>closeOnSelect</code> | Set to true and autocomplete will be closed when user picks value. Not available if multiple is enabled. Only works when <code>openIn</code> is 'popup' or 'page'. |
| <code>dropdownPlaceholderText</code> | Specify dropdown placeholder text. Only if <code>openIn</code> is "dropdown". |
| <code>multiple</code> | Whether to allow multiple value selection. Only works when <code>openIn</code> is 'popup' or 'page'. |
| <code>limit</code> | Limit number of maximum displayed items in autocomplete per query. |

| | |
|---------|--|
| style | Autocomplete styling parameters. Only available when openIn is "dropdown". |
| ... | Extra options. See https://framework7.io/docs/autocomplete#autocomplete-parameters |
| session | The Shiny session object. |

Note

Contrary to [f7Text](#), this input can't be cleared.

Author(s)

David Granjon, <dgranjon@gmail.com>

Examples

```
library(shiny)
library(shinyMobile)

app <- shinyApp(
  ui = f7Page(
    title = "My app",
    f7SingleLayout(
      navbar = f7Navbar(title = "Update autocomplete"),
      f7Block(f7Button(inputId = "update", label = "Update autocomplete")),
      f7Block(
        inset = TRUE,
        strong = TRUE,
        f7BlockTitle("Autocomplete input"),
        f7AutoComplete(
          inputId = "myautocomplete",
          placeholder = "Some text here!",
          openIn = "dropdown",
          label = "Type a fruit name",
          choices = c(
            "Apple", "Apricot", "Avocado", "Banana", "Melon",
            "Orange", "Peach", "Pear", "Pineapple"
          ),
          style = list(
            outline = TRUE,
            media = f7Icon("house"),
            description = "typeahead input",
            floating = TRUE
          )
        )
      )
    ),
    f7Block(verbatimTextOutput("autocompleteval"))
  ),
  server = function(input, output, session) {
    output$autocompleteval <- renderText(input$myautocomplete)

    observeEvent(input$update, {
      updateF7AutoComplete(
```

```
        inputId = "myautocomplete",
        value = "plip",
        choices = c("plip", "plap", "ploup")
    )
})
}
)

if (interactive() || identical(Sys.getenv("TESTTHAT"), "true")) app
```

f7Back

Framework7 back button

Description

[f7Back](#) is a button to go back in [f7Tabs](#).

Usage

```
f7Back(targetId)
```

Arguments

targetId [f7Tabs](#) id.

f7Badge

Framework7 badge

Description

Container to highlight important information with color.

Usage

```
f7Badge(..., color = NULL)
```

Arguments

... Badge content. Avoid long text.
color Badge color: see here for valid colors <https://framework7.io/docs/badge.html>.

Author(s)

David Granjon, <dgranjon@gmail.com>

Examples

```

if (interactive()) {
  library(shiny)
  library(shinyMobile)

  colors <- getF7Colors()

  shinyApp(
    ui = f7Page(
      title = "Badges",
      f7SingleLayout(
        navbar = f7Navbar(title = "f7Badge"),
        f7Block(
          strong = TRUE,
          f7Badge("Default"),
          lapply(seq_along(colors), function(i) {
            f7Badge(colors[[i]], color = colors[[i]])
          })
        )
      )
    ),
    server = function(input, output) {}
  )
}

```

f7Block

Framework7 block

Description

f7Block creates a block container.

f7BlockTitle creates a title for [f7Block](#).

f7BlockHeader creates a header content for [f7Block](#).

[f7BlockFooter](#) creates a footer content for [f7Block](#).

Usage

```

f7Block(
  ...,
  hairlines = deprecated(),
  strong = FALSE,
  inset = FALSE,
  tablet = FALSE,
  outline = FALSE
)

f7BlockTitle(title, size = NULL)

```

```
f7BlockHeader(text)
```

```
f7BlockFooter(text)
```

Arguments

| | |
|-----------|---|
| ... | Block content. Also for f7BlockHeader and f7BlockFooter . |
| hairlines | [Deprecated] : removed from Framework7. |
| strong | Add white background so that text is highlighted. FALSE by default. |
| inset | Whether to set block inset. FALSE by default. Works only if strong is TRUE. |
| tablet | Whether to make block inset only on large screens. FALSE by default. |
| outline | Block border. Default to FALSE. |
| title | Block title. |
| size | Block title size. NULL by default or "medium", "large". |
| text | Any text. |

Author(s)

David Granjon, <dgranjon@ymail.com>

Examples

```
if (interactive()) {
  library(shiny)
  library(shinyMobile)

  shinyApp(
    ui = f7Page(
      title = "Blocks",
      f7SingleLayout(
        navbar = f7Navbar(title = "f7Block"),
        f7BlockTitle(title = "A large title", size = "large"),
        f7Block(
          f7BlockHeader(text = "Header"),
          "Here comes paragraph within content block.
          Donec et nulla auctor massa pharetra
          adipiscing ut sit amet sem. Suspendisse
          molestie velit vitae mattis tincidunt.
          Ut sit amet quam mollis, vulputate
          turpis vel, sagittis felis.",
          f7BlockFooter(text = "Footer")
        ),
        f7BlockTitle(title = "A medium title", size = "medium"),
        f7Block(
          strong = TRUE,
          outline = TRUE,
          f7BlockHeader(text = "Header"),
          "Here comes paragraph within content block.
```



```

Donec et nulla auctor massa pharetra
adipiscing ut sit amet sem. Suspendisse
molestie velit vitae mattis tincidunt.
Ut sit amet quam mollis, vulputate
turpis vel, sagittis felis.",
    f7BlockFooter(text = "Footer")
),
f7BlockTitle(title = "A normal title", size = NULL),
f7Block(
    inset = TRUE,
    strong = TRUE,
    f7BlockHeader(text = "Header"),
    "Here comes paragraph within content block.
Donec et nulla auctor massa pharetra
adipiscing ut sit amet sem. Suspendisse
molestie velit vitae mattis tincidunt.
Ut sit amet quam mollis, vulputate
turpis vel, sagittis felis.",
    f7BlockFooter(text = "Footer")
),
f7Block(
    inset = TRUE,
    tablet = TRUE,
    strong = TRUE,
    f7BlockHeader(text = "Header"),
    "Here comes paragraph within content block.
Donec et nulla auctor massa pharetra
adipiscing ut sit amet sem. Suspendisse
molestie velit vitae mattis tincidunt.
Ut sit amet quam mollis, vulputate
turpis vel, sagittis felis.",
    f7BlockFooter(text = "Footer")
),
f7Block(
    inset = TRUE,
    strong = TRUE,
    outline = TRUE,
    f7BlockHeader(text = "Header"),
    "Here comes paragraph within content block.
Donec et nulla auctor massa pharetra
adipiscing ut sit amet sem. Suspendisse
molestie velit vitae mattis tincidunt.
Ut sit amet quam mollis, vulputate
turpis vel, sagittis felis.",
    f7BlockFooter(text = "Footer")
)
)
),
server = function(input, output) {}
}

```

f7Button*Framework7 action button*

Description

f7Button generates a Framework7 action button.

updateF7Button updates an [f7Button](#).

A Framework7 segmented button container for [f7Button](#).

Usage

```
f7Button(  
  inputId = NULL,  
  label = NULL,  
  href = NULL,  
  color = NULL,  
  fill = TRUE,  
  outline = FALSE,  
  shadow = FALSE,  
  rounded = FALSE,  
  size = NULL,  
  active = FALSE,  
  tonal = FALSE,  
  icon = NULL  
)  
  
updateF7Button(  
  inputId,  
  label = NULL,  
  color = NULL,  
  fill = NULL,  
  outline = NULL,  
  shadow = NULL,  
  rounded = NULL,  
  size = NULL,  
  tonal = NULL,  
  icon = NULL,  
  session = shiny::getDefaultReactiveDomain()  
)  
  
f7Segment(  
  ...,  
  container = deprecated(),  
  shadow = FALSE,  
  rounded = FALSE,  
  strong = FALSE  
)
```

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 or f7Icon . |
| href | Button link. |
| color | Button color. Not compatible with outline. See here for valid colors https://framework7.io/docs/badge.html . |
| fill | Fill style. TRUE by default. Not compatible with outline |
| outline | Outline style. FALSE by default. Not compatible with fill. |
| shadow | Button shadow. FALSE by default. Only for material design. |
| rounded | Round style. FALSE by default. |
| size | Button size. NULL by default but also "large" or "small". |
| active | Button active state. Default to FALSE. This is useful when used in f7Segment with the strong parameter set to TRUE. |
| tonal | Button tonal style. Default to FALSE |
| icon | Button icon. Expect f7Icon . |
| session | The Shiny session object, usually the default value will suffice. |
| ... | Slot for f7Button . |
| container | [Deprecated] : removed from Framework7. |
| strong | Add white background so that text is highlighted. FALSE by default. |

Author(s)

David Granjon, <dgranjon@gmail.com>

Examples

```
library(shiny)
library(shinyMobile)

app <- shiny::shinyApp(
  ui = f7Page(
    title = "Update f7Button",
    f7SingleLayout(
      navbar = f7Navbar(title = "Update f7Button"),
      f7Block(f7Button("update", "Update Button")),
      f7Block(
        f7Button(
          "button",
          "My button",
          color = "orange",
          outline = FALSE,
          fill = TRUE,
          shadow = FALSE,
          rounded = FALSE,
```

```

        icon = f7Icon("speedometer")
      )
    )
  ),
server = function(input, output, session) {
  observeEvent(input$update, {
    updateF7Button(
      inputId = "button",
      label = "Updated label",
      color = "purple",
      shadow = TRUE,
      rounded = TRUE,
      outline = TRUE,
      fill = FALSE,
      tonal = TRUE,
      size = "large",
      icon = f7Icon("speaker_zzz")
    )
  })
}
)

if (interactive() || identical(Sys.getenv("TESTTHAT"), "true")) app
if (interactive()) {
  library(shiny)
  library(shinyMobile)

  shinyApp(
    ui = f7Page(
      options = list(dark = FALSE),
      title = "Button Segments",
      f7SingleLayout(
        navbar = f7Navbar(title = "f7Segment, f7Button"),
        f7BlockTitle(title = "Simple Buttons in a segment"),
        f7Segment(
          f7Button(color = "blue", label = "My button", fill = FALSE),
          f7Button(color = "green", label = "My button", fill = FALSE),
          f7Button(color = "yellow", label = "My button", fill = FALSE)
        ),
        f7BlockTitle(title = "Tonal buttons"),
        f7Segment(
          f7Button(color = "blue", label = "My button", tonal = TRUE),
          f7Button(color = "green", label = "My button", tonal = TRUE),
          f7Button(color = "yellow", label = "My button", tonal = TRUE)
        ),
        f7BlockTitle(title = "Filled Buttons in a segment/rounded container"),
        f7Segment(
          rounded = TRUE,
          f7Button(color = "black", label = "My button"),
          f7Button(color = "green", label = "My button"),
          f7Button(color = "yellow", label = "My button")
        ),
      ),
    ),
  )
}

```

```

f7BlockTitle(title = "Outline Buttons in a segment/shadow container"),
f7Segment(
  shadow = TRUE,
  f7Button(label = "My button", outline = TRUE, fill = FALSE),
  f7Button(label = "My button", outline = TRUE, fill = FALSE),
  f7Button(label = "My button", outline = TRUE, fill = FALSE)
),
f7BlockTitle(title = "Buttons in a segment/strong container"),
f7Segment(
  strong = TRUE,
  f7Button(label = "My button", fill = FALSE),
  f7Button(label = "My button", fill = FALSE),
  f7Button(label = "My button", fill = FALSE, active = TRUE)
)
)
),
server = function(input, output) {}
)
}

```

f7Card

Framework7 card

Description

f7Card creates a simple card container.

f7ExpandableCard is a card that can expand. Ideal for a gallery.

updateF7Card maximizes an [f7ExpandableCard](#) on the client.

Usage

```

f7Card(
  ...,
  image = NULL,
  title = NULL,
  footer = NULL,
  outline = FALSE,
  height = NULL,
  raised = FALSE,
  divider = FALSE
)

```

```

f7ExpandableCard(
  ...,
  id = NULL,
  title = NULL,
  subtitle = NULL,

```

```

    color = NULL,
    image = NULL,
    fullBackground = FALSE,
    buttonColor = "white"
  )

  updateF7Card(id, session = shiny::getDefaultReactiveDomain())

```

Arguments

| | |
|----------------|--|
| ... | Card content. |
| image | Card background image url. Tje JPG format is preferred. Not compatible with the color argument. |
| title | Card title. |
| footer | Footer content, if any. Must be wrapped in a tagList. |
| outline | Outline style. FALSE by default. |
| height | Card height. NULL by default. |
| raised | Card shadow. FALSE by default. |
| divider | Card header/footer dividers. FALSE by default. |
| id | Card id. |
| subtitle | Card subtitle. |
| color | Card background color. See https://framework7.io/docs/cards.html . Not compatible with the img argument. |
| fullBackground | Whether the image should cover the entire card. |
| buttonColor | Color of the close button. Default is "white". |
| session | Shiny session object. |

Note

For [f7ExpandableCard](#), image and color are not compatible. Choose one of them.

Author(s)

David Granjon, <dgranjon@ymail.com>

Examples

```

# Simple card
if (interactive()) {
  library(shiny)
  library(shinyMobile)

  shinyApp(
    ui = f7Page(
      title = "Cards",
      options = list(dark = FALSE),

```

```

f7SingleLayout(
  navbar = f7Navbar(title = "f7Card"),
  f7Card("This is a simple card with plain text,
but cards can also contain their own header,
footer, list view, image, or any other element."),
  f7Card(
    title = "Card header",
    raised = TRUE,
    outline = TRUE,
    divider = TRUE,
    div(class = "date", "March 16, 2024"),
    "This is a simple card with plain text,
but cards can also contain their own header,
footer, list view, image, or any other element.",
    footer = "Card footer"
  ),
  f7Card(
    title = "Card header",
    image = "https://cdn.framework7.io/placeholder/nature-1000x600-3.jpg",
    "This is a simple card with plain text,
but cards can also contain their own header,
footer, list view, image, or any other element.",
    footer = tagList(
      f7Link("Link 1", href = "https://google.com"),
      f7Badge("Badge", color = "green")
    )
  )
)
)
),
server = function(input, output) {}
)
}

library(shiny)
library(shinyMobile)

app <- shinyApp(
  ui = f7Page(
    title = "Expandable Cards",
    f7SingleLayout(
      navbar = f7Navbar(title = "Expandable Cards"),
      f7ExpandableCard(
        id = "card1",
        title = "Expandable Card 1",
        image = "https://i.pinimg.com/originals/73/38/6e/73386e0513d4c02a4fbb814cadfba655.jpg",
        "Framework7 - is a free and open source HTML mobile framework
to develop hybrid mobile apps or web apps with iOS or Android
native look and feel. It is also an indispensable prototyping apps tool
to show working app prototype as soon as possible in case you need to."
      ),
      hr(),
      f7BlockTitle(title = "Click below to expand the card!") %>% f7Align(side = "center"),
      f7Button(inputId = "go", label = "Go"),

```

```

    br(),
    f7ExpandableCard(
      id = "card2",
      title = "Expandable Card 2",
      fullBackground = TRUE,
      image = "https://cdn.pixabay.com/photo/2017/10/03/18/55/mountain-2813667_960_720.png",
      "Framework7 - is a free and open source HTML mobile framework
      to develop hybrid mobile apps or web apps with iOS or Android
      native look and feel. It is also an indispensable prototyping apps tool
      to show working app prototype as soon as possible in case you need to."
    )
  ),
  server = function(input, output, session) {
    observeEvent(input$go, {
      updateF7Card(id = "card2")
    })
  }
)

if (interactive() || identical(Sys.getenv("TESTTHAT"), "true")) app

```

f7Checkbox

Framework7 checkbox

Description

[f7Checkbox](#) creates a checkbox input.

`updateF7Checkbox` changes the value of a checkbox input on the client.

Usage

```
f7Checkbox(inputId, label, value = FALSE)
```

```

updateF7Checkbox(
  inputId,
  label = NULL,
  value = NULL,
  session = shiny::getDefaultReactiveDomain()
)

```

Arguments

| | |
|----------------------|---|
| <code>inputId</code> | The input slot that will be used to access the value. |
| <code>label</code> | Display label for the control, or <code>NULL</code> for no label. |
| <code>value</code> | Initial value (<code>TRUE</code> or <code>FALSE</code>). |
| <code>session</code> | The Shiny session object. |

Examples

```

library(shiny)
library(shinyMobile)

app <- shinyApp(
  ui = f7Page(
    f7SingleLayout(
      navbar = f7Navbar(title = "updateF7Checkbox"),
      f7Block(f7Button("update", "Toggle checkbox")),
      f7Checkbox(
        inputId = "checkbox",
        label = "Checkbox",
        value = FALSE
      )
    )
  ), server = function(input, output, session) {
    observeEvent(input$update, {
      updateF7Checkbox("checkbox", value = !input$checkbox)
    })
  }
)

if (interactive() || identical(Sys.getenv("TESTTHAT"), "true")) app

```

f7CheckboxGroup

Framework7 checkbox group

Description

f7CheckboxGroup creates a checkbox group input

Custom choice item for [f7CheckboxGroup](#).

Usage

```

f7CheckboxGroup(
  inputId,
  label,
  choices = NULL,
  selected = NULL,
  position = c("left", "right"),
  style = list(inset = FALSE, outline = FALSE, dividers = FALSE, strong = FALSE)
)

f7CheckboxChoice(..., title, subtitle = NULL, after = NULL)

```

Arguments

| | |
|----------|--|
| inputId | Input id. |
| label | Input label |
| choices | List of choices. Can be a simple vector or named list or a list of f7RadioChoice or f7CheckboxChoice |
| selected | Selected element. NULL by default. If you pass f7RadioChoice or f7CheckboxChoice in choices, selected must be a numeric value corresponding to the index of the element to select. |
| position | Check mark side. "left" or "right". |
| style | Input style. Inherit from f7List options such as outline, inset, strong and dividers. |
| ... | Choice content. Text is striped if too long. |
| title | Item title. |
| subtitle | Item subtitle. |
| after | Display at the right of title. |

Examples

```
library(shiny)
library(shinyMobile)

app <- shinyApp(
  ui = f7Page(
    title = "My app",
    f7SingleLayout(
      navbar = f7Navbar(title = "f7CheckboxGroup"),
      f7BlockTitle("Simple choices", size = "large"),
      f7CheckboxGroup(
        inputId = "checkboxgroup",
        label = "Choose a variable:",
        choices = colnames(mtcars)[-1],
        selected = "disp",
        position = "right"
      ),
      tableOutput("data"),
      f7BlockTitle("Custom choices: f7CheckboxChoice", size = "large"),
      f7CheckboxGroup(
        inputId = "checkboxgroup2",
        label = "Custom choices",
        choices = list(
          f7CheckboxChoice(
            "Lorem ipsum dolor sit amet, consectetur adipiscing elit.
            Nulla sagittis tellus ut turpis condimentum,
            ut dignissim lacus tincidunt",
            title = "Choice 1",
            subtitle = "David",
            after = "March 16, 2024"
          )
        )
      )
    )
  )
)
```

```

    f7CheckboxChoice(
      "Cras dolor metus, ultrices condimentum sodales sit
      amet, pharetra sodales eros. Phasellus vel felis tellus.
      Mauris rutrum ligula nec dapibus feugiat",
      title = "Choice 2",
      subtitle = "Veerle",
      after = "March 17, 2024"
    )
  ),
  selected = 2,
  style = list(
    inset = TRUE,
    outline = TRUE,
    dividers = TRUE,
    strong = TRUE
  )
),
textOutput("selected")
)
),
server = function(input, output) {
  output$data <- renderTable(
    {
      mtcars[, c("mpg", input$checkboxgroup), drop = FALSE]
    },
    rownames = TRUE
  )
  output$selected <- renderText(input$checkboxgroup2)
}
)

if (interactive() || identical(Sys.getenv("TESTTHAT"), "true")) app

```

f7Chip

Framework7 chips

Description

f7Chip is an improved badge container.

Usage

```

f7Chip(
  label = NULL,
  image = NULL,
  icon = NULL,
  outline = FALSE,
  status = NULL,
  iconStatus = NULL,
  closable = FALSE
)

```

Arguments

| | |
|------------|--|
| label | Chip label. |
| image | Chip image, if any. |
| icon | Icon, if any. IOS and Material icons available. |
| outline | Whether to outline chip. FALSE by default. |
| status | Chip color: see here for valid colors https://framework7.io/docs/chips.html . |
| iconStatus | Chip icon color: see here for valid colors https://framework7.io/docs/chips.html . |
| closable | Whether to close the chip. FALSE by default. |

Author(s)

David Granjon, <dgranjon@ymail.com>

Examples

```
if(interactive()){
  library(shiny)
  library(shinyMobile)

  shinyApp(
    ui = f7Page(
      title = "Chips",
      f7SingleLayout(
        navbar = f7Navbar(title = "f7Chip"),
        f7Block(
          strong = TRUE,
          f7Chip(label = "simple Chip"),
          f7Chip(label = "outline Chip", outline = TRUE),
          f7Chip(label = "icon Chip", icon = f7Icon("plus_circle_fill"), iconStatus = "pink"),
          f7Chip(label = "image Chip", image = "https://loremflickr.com/g/320/240/london"),
          f7Chip(label = "closable Chip", closable = TRUE),
          f7Chip(label = "colored Chip", status = "green"),
          f7Chip(label = "colored outline Chip", status = "green", outline = TRUE)
        )
      )
    ),
    server = function(input, output) {}
  )
}
```

f7ColorPicker

*Create a Framework7 color picker input***Description**

Create a Framework7 color picker input

Usage

```
f7ColorPicker(
  inputId,
  label,
  value = "#ff0000",
  placeholder = NULL,
  modules = f7ColorPickerModules,
  palettes = f7ColorPickerPalettes,
  sliderValue = TRUE,
  sliderValueEditable = TRUE,
  sliderLabel = TRUE,
  hexLabel = TRUE,
  hexValueEditable = TRUE,
  groupedModules = TRUE,
  style = list(outline = FALSE, inset = FALSE, strong = FALSE, dividers = FALSE),
  ...
)
```

Arguments

| | |
|---------------------|--|
| inputId | Color picker input. |
| label | Color picker label. |
| value | Initial picker value in hex. |
| placeholder | Color picker placeholder. |
| modules | Picker color modules. Choose at least one. |
| palettes | Picker color predefined palettes. Must be a list of color vectors, each value specified as HEX string. |
| sliderValue | When enabled, it will display sliders values. |
| sliderValueEditable | When enabled, it will display sliders values as elements to edit directly. |
| sliderLabel | When enabled, it will display sliders labels with text. |
| hexLabel | When enabled, it will display HEX module label text, e.g. HEX. |
| hexValueEditable | When enabled, it will display HEX module value as element to edit directly. |
| groupedModules | When enabled it will add more exposure to sliders modules to make them look more separated. |

| | |
|-------|--|
| style | Input style. Inherit from f7List options such as outline, inset, strong and dividers. |
| ... | Other options to pass to the picker. See https://framework7.io/docs/color-picker#color-picker-parameters . |

Value

The return value is a list and includes hex, rgb, hsl, hsb, alpha, hue, rgba, and hsla values. See <https://framework7.io/docs/color-picker#color-picker-value>.

Examples

```
library(shiny)
library(shinyMobile)

app <- shinyApp(
  ui = f7Page(
    title = "My app",
    f7SingleLayout(
      navbar = f7Navbar(title = "f7ColorPicker"),
      f7ColorPicker(
        inputId = "mycolorpicker",
        placeholder = "Some text here!",
        label = "Select a color"
      ),
      "The picker hex value is:",
      textOutput("colorPickerVal"),
      "The picker rgb value is:",
      textOutput("colorPickerValRgb")
    )
  ),
  server = function(input, output) {
    output$colorPickerVal <- renderText(input$mycolorpicker$hex)
    output$colorPickerValRgb <- renderText(unlist(paste(input$mycolorpicker$rgb, collapse = ",")))
  }
)

if (interactive() || identical(Sys.getenv("TESTTHAT"), "true")) app
```

f7DatePicker

Framework7 date picker

Description

f7DatePicker creates a Framework7 date picker input.

updateF7DatePicker changes the value of a date picker input on the client.

Usage

```
f7DatePicker(
  inputId,
  label,
  value = NULL,
  multiple = FALSE,
  direction = c("horizontal", "vertical"),
  minDate = NULL,
  maxDate = NULL,
  dateFormat = "yyyy-mm-dd",
  openIn = c("auto", "popover", "sheet", "customModal"),
  scrollToInput = FALSE,
  closeByOutsideClick = TRUE,
  toolbar = TRUE,
  toolbarCloseText = "Done",
  header = FALSE,
  headerPlaceholder = "Select date",
  style = list(outline = FALSE, inset = FALSE, strong = FALSE, dividers = FALSE),
  ...
)

updateF7DatePicker(
  inputId,
  value = NULL,
  ...,
  session = shiny::getDefaultReactiveDomain()
)
```

Arguments

| | |
|----------------------------------|--|
| <code>inputId</code> | Date input id. |
| <code>label</code> | Input label. |
| <code>value</code> | Array with initial selected dates. Each array item represents selected date. If <code>timePicker</code> enabled, the value needs to be an object of type <code>POSIXct</code> . |
| <code>multiple</code> | If <code>TRUE</code> allow to select multiple dates. |
| <code>direction</code> | Months layout direction, could be <code>'horizontal'</code> or <code>'vertical'</code> . |
| <code>minDate</code> | Minimum allowed date. |
| <code>maxDate</code> | Maximum allowed date. |
| <code>dateFormat</code> | Date format: <code>"yyyy-mm-dd"</code> , for instance. |
| <code>openIn</code> | Can be <code>auto</code> , <code>popover</code> (to open calendar in popover), <code>sheet</code> (to open in sheet modal) or <code>customModal</code> (to open in custom Calendar modal overlay). In case of <code>auto</code> will open in sheet modal on small screens and in popover on large screens. |
| <code>scrollToInput</code> | Scroll viewport (page-content) to input when calendar opened. |
| <code>closeByOutsideClick</code> | If enabled, picker will be closed by clicking outside of picker or related input element. |

| | |
|-------------------|--|
| toolbar | Enables calendar toolbar. |
| toolbarCloseText | Text for Done/Close toolbar button. |
| header | Enables calendar header. |
| headerPlaceholder | Default calendar header placeholder text. |
| style | Input style. Inherit from f7List options such as outline, inset, strong and dividers. |
| ... | Other options to pass to the picker. See https://framework7.io/docs/calendar#calendar-parameters . |
| session | The Shiny session object, usually the default value will suffice. |

Value

a Date vector.

Examples

```
library(shiny)
library(shinyMobile)

app <- shinyApp(
  ui = f7Page(
    title = "My app",
    f7SingleLayout(
      navbar = f7Navbar(title = "Update date picker"),
      f7Segment(
        f7Button(inputId = "update", label = "Update"),
        f7Button(inputId = "rmToolbar", label = "Remove toolbar"),
        f7Button(inputId = "addToolbar", label = "Add toolbar")
      ),
      f7Segment(
        f7Button(inputId = "removeTime", label = "Remove time"),
        f7Button(inputId = "addTime", label = "Add time")
      ),
      f7DatePicker(
        inputId = "picker",
        label = "Choose a date and time",
        value = as.POSIXct("2024-03-24 09:00:00 UTC"),
        openIn = "auto",
        direction = "horizontal",
        timePicker = TRUE,
        dateFormat = "yyyy-mm-dd, HH:mm"
      ),
      f7Block(verbatimTextOutput("pickerval"))
    )
  ),
  server = function(input, output, session) {
    output$pickerval <- renderPrint(input$picker)
  }
)
```



```

observeEvent(input$update, {
  updateF7DatePicker(
    inputId = "picker",
    value = as.POSIXct("2024-03-23 10:00:00 UTC"),
    timePicker = TRUE,
    dateFormat = "yyyy-mm-dd, HH:mm" # preserve date format
  )
})

observeEvent(input$rmToolbar, {
  updateF7DatePicker(
    inputId = "picker",
    timePicker = TRUE,
    toolbar = FALSE,
    dateFormat = "yyyy-mm-dd, HH:mm" # preserve date format
  )
})

observeEvent(input$addToolbar, {
  updateF7DatePicker(
    inputId = "picker",
    timePicker = TRUE,
    toolbar = TRUE,
    dateFormat = "yyyy-mm-dd, HH:mm" # preserve date format
  )
})

observeEvent(input$removeTime, {
  updateF7DatePicker(
    inputId = "picker",
    value = as.Date(input$picker),
    timePicker = FALSE,
    dateFormat = "yyyy-mm-dd" # new date format
  )
})

observeEvent(input$addTime, {
  updateF7DatePicker(
    inputId = "picker",
    value = as.POSIXct(input$picker),
    timePicker = TRUE,
    dateFormat = "yyyy-mm-dd, HH:mm" # preserve date format
  )
})
}
)

if (interactive() || identical(Sys.getenv("TESTTHAT"), "true")) app

```

Description

List of default custom options.

Usage

```
f7DefaultOptions()
```

Value

A list of options to pass in [f7Page](#).

| | |
|----------|---------------------------------|
| f7Dialog | <i>Framework7 dialog window</i> |
|----------|---------------------------------|

Description

f7Dialog generates a modal window.

Usage

```
f7Dialog(
  id = NULL,
  title = NULL,
  text,
  type = c("alert", "confirm", "prompt", "login"),
  session = shiny::getDefaultReactiveDomain()
)
```

Arguments

| | |
|---------|--|
| id | Input associated to the alert. Works when type is one of "confirm", "prompt" or "login". |
| title | Dialog title |
| text | Dialog text. |
| type | Dialog type: c("alert", "confirm", "prompt", "login"). |
| session | shiny session. |

Examples

```
library(shiny)
library(shinyMobile)

app <- shinyApp(
  ui = f7Page(
    title = "Dialogs",
    f7SingleLayout(
      navbar = f7Navbar(title = "f7Dialog"),
```

```

    f7Block(
      f7Grid(
        cols = 4,
        f7Button(inputId = "alert", "Alert"),
        f7Button(inputId = "confirm", "Confirm"),
        f7Button(inputId = "prompt", "Prompt"),
        f7Button(inputId = "login", "Login")
      ),
      f7Grid(
        cols = 2,
        uiOutput("prompt_res"),
        uiOutput("login_res")
      )
    )
  ),
),
server = function(input, output, session) {
  observeEvent(input$alert, {
    f7Dialog(
      title = "Dialog title",
      text = "This is an alert dialog"
    )
  })

  observeEvent(input$confirm, {
    f7Dialog(
      id = "confirm_dialog",
      title = "Dialog title",
      type = "confirm",
      text = "This is an alert dialog"
    )
  })

  observeEvent(input$confirm_dialog, {
    f7Toast(text = paste("Alert input is:", input$confirm_dialog))
  })

  observeEvent(input$prompt, {
    f7Dialog(
      id = "prompt_dialog",
      title = "Dialog title",
      type = "prompt",
      text = "This is a prompt dialog"
    )
  })
  output$prompt_res <- renderText({
    req(input$prompt_dialog)
    input$prompt_dialog
  })

  observeEvent(input$login, {
    f7Dialog(
      id = "login_dialog",

```

```

        title = "Dialog title",
        type = "login",
        text = "This is an login dialog"
      )
    })

    output$login_res <- renderUI({
      req(input$login_dialog$user, input$login_dialog$password)
      img(src = "https://media2.giphy.com/media/12gfL8Xrhv7C1fXiV/giphy.gif")
    })
  }
)

if (interactive() || identical(Sys.getenv("TESTTHAT"), "true")) app

```

f7DownloadButton *Create a download button*

Description

Use these functions to create a download button; when clicked, it will initiate a browser download. The filename and contents are specified by the corresponding shiny downloadHandler() defined in the server function.

Usage

```
f7DownloadButton(outputId, label = "Download", class = NULL, ...)
```

Arguments

| | |
|----------|--|
| outputId | The name of the output slot that the downloadHandler is assigned to. |
| label | The label that should appear on the button. |
| class | Additional CSS classes to apply to the tag, if any. |
| ... | Other arguments to pass to the container tag function. |

Examples

```

if (interactive()) {
  library(shiny)
  library(shinyMobile)
  ui = f7Page(
    f7SingleLayout(
      navbar = f7Navbar(title = "File handling"),
      f7Block(f7DownloadButton("download", "Download!"))
    )
  )
)

server = function(input, output, session) {
  # Our dataset

```

```

data <- mtcars

output$download = downloadHandler(
  filename = function() {
    paste("data-", Sys.Date(), ".csv", sep="")
  },
  content = function(file) {
    write.csv(data, file)
  }
)
}

shinyApp(ui, server)
}

```

f7Fab

*Framework7 floating action button (FAB)***Description**

f7Fab generates a nice button to be put in [f7Fabs](#).

updateF7Fab changes the label of an [f7Fab](#) input on the client.

Usage

```
f7Fab(inputId, label, width = NULL, ..., flag = NULL)
```

```
updateF7Fab(inputId, label = NULL, session = shiny::getDefaultReactiveDomain())
```

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. |
| width | The width of the input, e.g. '400px', or '100%'; see validateCssUnit() . |
| ... | Named attributes to be applied to the button or link. |
| flag | Additional text displayed next to the button content. Only works if f7Fabs position parameter is not starting with center-... |
| session | The Shiny session object, usually the default value will suffice. |

Author(s)

David Granjon, <dgranjon@ymail.com>

| | |
|------------|-----------------------------|
| f7FabClose | <i>Framework7 FAB close</i> |
|------------|-----------------------------|

Description

f7FabClose indicates that the current tag should close the [f7Fabs](#).

Usage

```
f7FabClose(tag)
```

Arguments

| | |
|-----|-------------|
| tag | Target tag. |
|-----|-------------|

| | |
|--------|--|
| f7Fabs | <i>Framework7 container for floating action button (FAB)</i> |
|--------|--|

Description

f7Fabs hosts multiple [f7Fab](#).

updateF7Fabs toggles [f7Fabs](#) on the server side.

f7FabMorphTarget convert a tag into a target morphing. See <https://framework7.io/docs/floating-action-button#fab-morph>.

Usage

```
f7Fabs(
  ...,
  id = NULL,
  position = c("right-top", "right-center", "right-bottom", "left-top", "left-center",
    "left-bottom", "center-center", "center-top", "center-bottom"),
  color = NULL,
  extended = FALSE,
  label = NULL,
  sideOpen = c("left", "right", "top", "bottom", "center"),
  morph = deprecated(),
  morphTarget = NULL,
  global = FALSE
)

updateF7Fabs(id, session = shiny::getDefaultReactiveDomain())

f7FabMorphTarget(tag)
```

Arguments

| | |
|-------------|---|
| ... | Slot for f7Fab . |
| id | Optional: access the current state of the f7Fabs container. |
| position | Container position. |
| color | Container color. |
| extended | If TRUE, the FAB will be wider. This allows to use a label (see below). |
| label | Container label. Only if extended is TRUE. |
| sideOpen | When the container is pressed, indicate where buttons are displayed. |
| morph | [Deprecated] : removed from Framework7. |
| morphTarget | CSS selector of the morph target: ". toolbar" for instance. |
| global | If FABs are used in f7TabLayout and this argument is set to TRUE, the FAB will be displayed on all tabs. If FALSE, the default, it will be displayed only on tab it is called from. |
| session | The Shiny session object, usually the default value will suffice. |
| tag | Target tag. |

Note

The background color might be an issue depending on the parent container. Consider it experimental.

Author(s)

David Granjon, <dgranjon@gmail.com>

Examples

```
library(shiny)
library(shinyMobile)

app <- shinyApp(
  ui = f7Page(
    title = "Update f7Fabs",
    f7SingleLayout(
      navbar = f7Navbar(title = "Update f7Fabs"),
      toolbar = f7Toolbar(
        position = "bottom",
        lapply(1:3, function(i) f7Link(label = i, href = "#") |> f7FabClose())
      ) |> f7FabMorphTarget(),
      # put an empty f7Fabs container
      f7Fabs(
        id = "fabsMorph",
        extended = TRUE,
        label = "Open",
        position = "center-bottom",
        color = "yellow",
        sideOpen = "right",
```

```

    morphTarget = ".toolbar"
  ),
  f7Block(f7Button(inputId = "toggle", label = "Toggle Fabs")),
  f7Fabs(
    position = "center-center",
    id = "fabs",
    lapply(1:3, function(i) f7Fab(inputId = i, label = i))
  ),
  f7BlockTitle("Output"),
  f7Block(
    textOutput("res")
  ),
  f7BlockTitle("Page Content"),
  f7Block(
    p("Lorem ipsum dolor sit amet, consectetur adipiscing elit. Suspendisse
      hendrerit magna non sem iaculis, ac rhoncus est pulvinar. Interdum et
      malesuada fames ac ante ipsum primis in faucibus. In sagittis vel lacus
      ac bibendum. Maecenas mollis, diam nec fermentum sollicitudin, massa
      lectus ullamcorper orci, in laoreet lectus quam nec lacus.
      Nulla sollicitudin imperdiet metus, quis mollis justo finibus varius.
      In mattis malesuada enim in tincidunt. Nulla vehicula dui lacus,
      iaculis condimentum dui dapibus ac. Cras elit nunc, auctor vestibulum
      odio id, iaculis posuere arcu. Mauris dignissim id lectus sit amet
      vestibulum. Nam rutrum sit amet augue vel interdum. Donec sed orci vitae
      eros eleifend posuere vitae id nibh. Donec faucibus erat in placerat
      feugiat. Sed sodales facilisis eros, porta viverra purus pretium eu.
      Morbi vehicula metus lacus, id commodo mauris posuere nec. Vivamus
      ornare et lacus et lobortis. Etiam tristique elit id eros ornare,
      vel faucibus mauris hendrerit. Nulla elit nulla, consequat sit amet
      neque et, ultrices elementum diam. Etiam dignissim elit a arcu pulvinar,
      ut dapibus elit maximus. Mauris ultricies nulla in mauris laoreet, at
      lacinia lorem maximus. Nulla sed enim diam. In ac felis dignissim,
      euismod augue nec, tempus augue. Maecenas eget aliquam mi.
      In tincidunt massa a velit suscipit, ac dapibus mi laoreet. Vestibulum
      lacinia nulla lorem, nec blandit quam sollicitudin at. Pellentesque
      in vehicula lacus. Etiam vitae lectus malesuada, hendrerit mauris eu,
      placerat elit. Mauris vehicula dictum pharetra. Etiam interdum vehicula
      urna, ac blandit lectus posuere id. Nullam facilisis tincidunt sem et
      pretium. Praesent pulvinar feugiat augue, quis pretium nunc vestibulum a.
      Morbi id eros eget lectus placerat placerat. Morbi dapibus viverra
      orci nec pellentesque. Vestibulum mollis gravida sem, quis tincidunt
      sem maximus gravida. Nam id egestas augue, sit amet egestas orci. Duis
      porttitor lectus sit amet efficitur auctor. Quisque dui ante, eleifend
      eget nibh a, tincidunt interdum nisi. Integer varius tempor erat, in
      commodo neque elementum ut. Maecenas eu lorem ultrices, posuere neque ac,
      aliquam ante. Maecenas eu volutpat arcu. Morbi hendrerit sem sed vehicula
      sodales. Quisque ultrices massa erat, vel accumsan risus vehicula eu.
      Donec laoreet aliquet est, a consequat odio viverra lacinia. Suspendisse
      id iaculis risus. Vestibulum posuere dignissim lacus quis ornare. Nam
      dapibus efficitur neque sed tristique.")
  )
)
)
)

```



```

    ),
    server = function(input, output, session) {
      output$res <- renderText(input[["1"]])

      observeEvent(input$toggle, {
        updateF7Fabs(id = "fabs")
      })
    }
  )

  if (interactive() || identical(Sys.getenv("TESTTHAT"), "true")) app

```

f7File

File Upload Control

Description

Create a file upload control that can be used to upload one or more files.

Usage

```

f7File(
  inputId,
  label,
  multiple = FALSE,
  accept = NULL,
  width = NULL,
  buttonLabel = "Browse...",
  placeholder = "No file selected"
)

```

Arguments

| | |
|-------------|---|
| inputId | The input slot that will be used to access the value. |
| label | Display label for the control, or NULL for no label. |
| multiple | Whether the user should be allowed to select and upload multiple files at once. Does not work on older browsers, including Internet Explorer 9 and earlier. |
| accept | A character vector of MIME types; gives the browser a hint of what kind of files the server is expecting. |
| width | The width of the input, e.g. 400px. |
| buttonLabel | The label used on the button. Can be text or an HTML tag object. |
| placeholder | The text to show before a file has been uploaded. |

Examples

```
if (interactive()) {
  library(shiny)
  library(shinyMobile)

  ui <- f7Page(
    f7SingleLayout(
      navbar = f7Navbar(title = "File handling"),
      f7Block(f7File("up", "Upload!"))
    )
  )

  server <- function(input, output) {
    data <- reactive(input$up)
    observe(print(data()))
  }

  shinyApp(ui, server)
}
```

f7Float

Framework7 float utility

Description

f7Float is an alignment utility for items.

Usage

```
f7Float(tag, side = c("left", "right"))
```

Arguments

| | |
|------|-----------------------------------|
| tag | Tag to float. |
| side | Side to float: "left" or "right". |

Author(s)

David Granjon, <dgranjon@gmail.com>

Examples

```
if(interactive()){
  library(shiny)
  library(shinyMobile)

  shinyApp(
    ui = f7Page(
      title = "Float",
```

```

f7SingleLayout(
  navbar = f7Navbar(title = "f7Float"),
  f7Float(h1("Left"), side = "left"),
  f7Float(h1("Right"), side = "right")
)
),
server = function(input, output) {}
)
}

```

f7Form

Creates an input form

Description

Instead of having shiny return one input at a time, a form is a collection of related inputs. The form returns a list with all sub-inputs as elements. This avoids to have to deal with too many inputs.

[updateF7Form](#) update form inputs on the server.

Usage

```
f7Form(id, ...)
```

```
updateF7Form(id, data, session = shiny::getDefaultReactiveDomain())
```

Arguments

| | |
|---------|---|
| id | Form unique id. Using <code>input\$<id></code> gives the form result. |
| ... | A list of input elements. |
| data | New form data. |
| session | Shiny session objects. |

Details

This only works with elements having an input HTML tag.

Examples

```

library(shiny)
library(shinyMobile)

app <- shinyApp(
  ui = f7Page(
    f7SingleLayout(
      navbar = f7Navbar(title = "Inputs form"),
      f7Block(f7Button("update", "Click me")),
      f7BlockTitle("A list of inputs in a form"),

```

```

f7List(
  inset = TRUE,
  dividers = FALSE,
  strong = TRUE,
  f7Form(
    id = "myform",
    f7Text(
      inputId = "text",
      label = "Text input",
      value = "Some text",
      placeholder = "Your text here",
      style = list(
        description = "A cool text input",
        outline = TRUE,
        media = f7Icon("house"),
        clearable = TRUE,
        floating = TRUE
      )
    ),
    f7TextArea(
      inputId = "textarea",
      label = "Text Area",
      value = "Lorem ipsum dolor sit amet, consectetur
        adipiscing elit, sed do eiusmod tempor incididunt ut
        labore et dolore magna aliqua",
      placeholder = "Your text here",
      resize = TRUE,
      style = list(
        description = "A cool text input",
        outline = TRUE,
        media = f7Icon("house"),
        clearable = TRUE,
        floating = TRUE
      )
    ),
    f7Password(
      inputId = "password",
      label = "Password:",
      placeholder = "Your password here",
      style = list(
        description = "A cool text input",
        outline = TRUE,
        media = f7Icon("house"),
        clearable = TRUE,
        floating = TRUE
      )
    )
  ),
  verbatimTextOutput("form")
),
server = function(input, output, session) {

```

```

output$form <- renderPrint(input$myform)

observeEvent(input$update, {
  updateF7Form(
    "myform",
    data = list(
      "text" = "New text",
      "textarea" = "New text area",
      "password" = "New password"
    )
  )
})
}
)

if (interactive() || identical(Sys.getenv("TESTTHAT"), "true")) app

```

f7Gallery

Launch the shinyMobile Gallery

Description

A gallery of all components available in shinyMobile.

Usage

```
f7Gallery()
```

Examples

```

if (interactive()) {
  f7Gallery()
}

```

f7Gauge

Framework7 gauge

Description

f7Gauge creates a gauge instance.

updateF7Gauge updates a framework7 gauge from the server side.

Usage

```

f7Gauge(
  id,
  type = "circle",
  value,
  size = 200,
  bgColor = "transparent",
  borderBgColor = "#eeeeee",
  borderColor = "#000000",
  borderWidth = "10",
  valueText = NULL,
  valueTextColor = "#000000",
  valueFontSize = "31",
  valueFontWeight = "500",
  labelText = NULL,
  labelTextColor = "#888888",
  labelTextSize = "14",
  labelTextWeight = "400"
)

updateF7Gauge(
  id,
  value = NULL,
  labelText = NULL,
  size = NULL,
  bgColor = NULL,
  borderBgColor = NULL,
  borderColor = NULL,
  borderWidth = NULL,
  valueText = NULL,
  valueTextColor = NULL,
  valueFontSize = NULL,
  valueFontWeight = NULL,
  labelTextColor = NULL,
  labelTextSize = NULL,
  labelTextWeight = NULL,
  session = shiny::getDefaultReactiveDomain()
)

```

Arguments

| | |
|----------------------|---|
| <code>id</code> | Gauge ID. |
| <code>type</code> | Gauge type. Can be "circle" or "semicircle". Default is "circle." |
| <code>value</code> | Gauge value/percentage. Must be a number between 0 and 100. |
| <code>size</code> | Generated SVG image size (in px). Default is 200. |
| <code>bgColor</code> | Gauge background color. Can be any valid color string, e.g. #ff00ff, rgb(0,0,255), etc. Default is "transparent". |

| | |
|------------------------------|---|
| <code>borderBgColor</code> | Main border/stroke background color. |
| <code>borderColor</code> | Main border/stroke color. |
| <code>borderWidth</code> | Main border/stroke width. |
| <code>valueText</code> | Gauge value text (large text in the center of gauge). |
| <code>valueTextColor</code> | Value text color. |
| <code>valueFontSize</code> | Value text font size. |
| <code>valueFontWeight</code> | Value text font weight. |
| <code>labelText</code> | Gauge additional label text. |
| <code>labelTextColor</code> | Label text color. |
| <code>labelFontSize</code> | Label text font size. |
| <code>labelFontWeight</code> | Label text font weight. |
| <code>session</code> | Shiny session object. |

Author(s)

David Granjon <dgranjon@ymail.com>

Examples

```
library(shiny)
library(shinyMobile)

app <- shinyApp(
  ui = f7Page(
    title = "Gauges",
    f7SingleLayout(
      navbar = f7Navbar(title = "f7Gauge"),
      f7Block(
        f7Gauge(
          id = "mygauge",
          type = "semicircle",
          value = 50,
          borderColor = "#2196f3",
          borderWidth = 10,
          valueFontSize = 41,
          valueTextColor = "#2196f3",
          labelText = "amount of something"
        )
      ),
      f7Block(f7Button("update", "Update Gauge"))
    )
  ),
  server = function(input, output, session) {
    observeEvent(input$update, {
      updateF7Gauge(id = "mygauge", value = 75, labelText = "New label!")
    })
  }
)
```

```

    }
  )

  if (interactive() || identical(Sys.getenv("TESTTHAT"), "true")) app

```

f7Grid *Framework7 grid container*

Description

Grid container for elements

Usage

```
f7Grid(..., cols, gap = TRUE)
```

Arguments

| | |
|------|--|
| ... | Row content. |
| cols | Columns number. Numeric between 1 and 20. |
| gap | Whether to display gap between columns. TRUE by default. |

Author(s)

David Granjon, <dgranjon@gmail.com>

f7Icon *Framework7 icons*

Description

Use Framework7 icons in shiny applications, see complete list of icons here : <https://framework7.io/icons/>.

Usage

```
f7Icon(..., lib = NULL, color = NULL, style = NULL)
```

Arguments

| | |
|-------|---|
| ... | Icon name and f7Badge . |
| lib | Library to use: NULL, "ios" or "md". Leave NULL by default. Specify, md or ios if you want to hide/show icons on specific devices. If you choose "md" be sure to include the corresponding fonts as they are not provided by shinyMobile. You can get them at https://github.com/marella/material-icons/ . |
| color | Icon color, if any. |
| style | CSS styles to be applied on icon, for example use font-size: 56px; to have a bigger icon. |

Author(s)

David Granjon, <dgranjon@gmail.com>

Examples

```
if (interactive()) {
  library(shiny)
  library(shinyMobile)

  shinyApp(
    ui = f7Page(
      title = "Icons",
      f7SingleLayout(
        navbar = f7Navbar(title = "icons"),
        f7List(
          f7ListItem(
            title = tagList(
              f7Icon("envelope")
            )
          ),
          f7ListItem(
            title = tagList(
              f7Icon("envelope_fill", color = "green")
            )
          ),
          f7ListItem(
            title = f7Icon("house", f7Badge("1", color = "red"))
          ),
          f7ListItem(
            title = f7Icon("home", lib = "md"),
            "Only for material design"
          )
        )
      )
    ),
    server = function(input, output) {}
  )
}
```

f7Item

Framework7 body item

Description

Similar to [f7Tab](#) but for the [f7SplitLayout](#).

Usage

```
f7Item(..., tabName)
```

Arguments

... Item content.
 tabName Item id. Must be unique, without space nor punctuation symbols.

Author(s)

David Granjon, <dgranjon@ymail.com>

f7Items *Framework7 item container*

Description

Build a Framework7 wrapper for [f7Item](#)

Usage

f7Items(...)

Arguments

... Slot for wrapper for [f7Item](#).

Author(s)

David Granjon, <dgranjon@ymail.com>

f7Link *Framework7 link*

Description

Link to point toward external content.

Usage

f7Link(label = NULL, href, icon = NULL, routable = FALSE)

Arguments

label Optional link text.
 href Link source, url.
 icon Link icon, if any. Must pass [f7Icon](#).
 routable Whether to make the link handled by the framework 7 router. Default to FALSE which opens a new page in a new tab.

Author(s)

David Granjon, <dgranjon@gmail.com>

Examples

```
if (interactive()) {
  library(shiny)
  library(shinyMobile)

  shinyApp(
    ui = f7Page(
      title = "Links",
      f7SingleLayout(
        navbar = f7Navbar(title = "f7Link"),
        f7Link(label = "Google", href = "https://www.google.com"),
        f7Link(href = "https://www.twitter.com", icon = f7Icon("bolt_fill"))
      )
    ),
    server = function(input, output) {}
  )
}
```

f7List

Create a framework 7 list view

Description

Create a framework 7 list view

Usage

```
f7List(
  ...,
  mode = NULL,
  inset = FALSE,
  outline = FALSE,
  dividers = FALSE,
  strong = FALSE,
  id = NULL
)
```

Arguments

| | |
|-------|--|
| ... | Slot for f7ListGroup or f7ListItem . |
| mode | List mode. NULL, "simple", "links", "media" or "contacts". |
| inset | Whether to display a card border. FALSE by default. |

| | |
|----------|--|
| outline | Outline style. Default to FALSE. |
| dividers | Dividers style. Default to FALSE. |
| strong | Strong style. Default to FALSE. |
| id | Optional id, which can be used as a target for f7ListIndex . |

Examples

```

library(shiny)
library(shinyMobile)

app <- shinyApp(
  ui = f7Page(
    title = "My app",
    f7TabLayout(
      navbar = f7Navbar(title = "f7List"),

      f7Tabs(
        f7Tab(
          title = "Lists",
          tabName = "list",

          # simple list
          f7List(
            mode = "simple",
            lapply(1:3, function(j) tags$li(letters[j]))
          ),

          # list with complex items
          f7List(
            strong = TRUE,
            outline = TRUE,
            inset = TRUE,
            lapply(1:3, function(j) {
              f7ListItem(
                letters[j],
                media = f7Icon("alarm_fill"),
                header = "Header",
                footer = "Footer"
              )
            })
          ),

          # list with complex items
          f7List(
            mode = "media",
            lapply(1:3, function(j) {
              f7ListItem(
                title = letters[j],
                subtitle = "subtitle",
                "Lorem ipsum dolor sit amet, consectetur adipiscing elit.
                Nulla sagittis tellus ut turpis condimentum, ut dignissim
                lacus tincidunt. Cras dolor metus, ultrices condimentum sodales

```

```

sit amet, pharetra sodales eros. Phasellus vel felis tellus.
Mauris rutrum ligula nec dapibus feugiat. In vel dui laoreet,
commodo augue id, pulvinar lacus.",
    media = tags$img(
        src = paste0(
            "https://cdn.framework7.io/placeholder/people-160x160-", j, ".jpg"
        )
    ),
    right = "Right Text"
)
})
),

# list with links
f7List(
    mode = "links",
    lapply(1:3, function(j) {
        tags$li(
            f7Link(label = letters[j], href = "https://google.com")
        )
    })
)
),
f7Tab(
    title = "Group",
    tabName = "groupedlists",

    # grouped lists
    f7List(
        id = "mycontacts",
        mode = "contacts",
        lapply(1:3, function(i) {
            f7ListGroup(
                title = LETTERS[i],
                lapply(1:10, function(j) f7ListItem(letters[j]))
            )
        })
    )
),
f7Tab(
    title = "Other group",
    tabName = "groupedlists2",

    # grouped lists
    f7List(
        id = "myothercontacts",
        mode = "contacts",
        lapply(4:6, function(i) {
            f7ListGroup(
                title = LETTERS[i],
                lapply(10:20, function(j) f7ListItem(letters[j]))
            )
        })
    )
)

```

```

    )
  )
)

)
),
server = function(input, output) {
  f7ListIndex(id = "contacts", target = "#mycontacts", label = TRUE)
  f7ListIndex(id = "othercontacts", target = "#myothercontacts", label = TRUE)

}
)

if (interactive() || identical(Sys.getenv("TESTTHAT"), "true")) app

```

| | |
|-------------|---|
| f7ListGroup | <i>Create a framework 7 group of contacts</i> |
|-------------|---|

Description

Create a framework 7 group of contacts

Usage

```
f7ListGroup(..., title)
```

Arguments

| | |
|-------|---------------------------------------|
| ... | slot for f7ListItem . |
| title | Group title. |

| | |
|-------------|--|
| f7ListIndex | <i>Create a Framework 7 list index</i> |
|-------------|--|

Description

List index must be attached to an existing list view.

Usage

```
f7ListIndex(id, target, ..., session = shiny::getDefaultReactiveDomain())
```

Arguments

| | |
|---------|---|
| id | Unique id. |
| target | Related list element. CSS selector like .class, #id, ... |
| ... | Other options (see https://framework7.io/docs/list-index#list-index-parameters). |
| session | Shiny session object. |

Note

While you can also supply a class as target, we advise to use an id to avoid conflicts.

Examples

```
library(shiny)
library(shinyMobile)

app <- shinyApp(
  ui = f7Page(
    title = "My app",
    f7TabLayout(
      navbar = f7Navbar(title = "f7List"),

      f7Tabs(
        f7Tab(
          title = "Lists",
          tabName = "list",

          # simple list
          f7List(
            mode = "simple",
            lapply(1:3, function(j) tags$li(letters[j]))
          ),

          # list with complex items
          f7List(
            strong = TRUE,
            outline = TRUE,
            inset = TRUE,
            lapply(1:3, function(j) {
              f7ListItem(
                letters[j],
                media = f7Icon("alarm_fill"),
                header = "Header",
                footer = "Footer"
              )
            })
          ),

          # list with complex items
          f7List(
            mode = "media",
            lapply(1:3, function(j) {
              f7ListItem(
                title = letters[j],
                subtitle = "subtitle",
                "Lorem ipsum dolor sit amet, consectetur adipiscing elit.
                Nulla sagittis tellus ut turpis condimentum, ut dignissim
                lacus tincidunt. Cras dolor metus, ultrices condimentum sodales
                sit amet, pharetra sodales eros. Phasellus vel felis tellus.
                Mauris rutrum ligula nec dapibus feugiat. In vel dui laoreet,
```

```

    commodo augue id, pulvinar lacus.",
    media = tags$img(
      src = paste0(
        "https://cdn.framework7.io/placeholder/people-160x160-", j, ".jpg"
      )
    ),
    right = "Right Text"
  )
})
),

# list with links
f7List(
  mode = "links",
  lapply(1:3, function(j) {
    tags$li(
      f7Link(label = letters[j], href = "https://google.com")
    )
  })
)
),
f7Tab(
  title = "Group",
  tabName = "groupedlists",

  # grouped lists
  f7List(
    id = "mycontacts",
    mode = "contacts",
    lapply(1:3, function(i) {
      f7ListGroup(
        title = LETTERS[i],
        lapply(1:10, function(j) f7ListItem(letters[j]))
      )
    })
  )
),
f7Tab(
  title = "Other group",
  tabName = "groupedlists2",

  # grouped lists
  f7List(
    id = "myothercontacts",
    mode = "contacts",
    lapply(4:6, function(i) {
      f7ListGroup(
        title = LETTERS[i],
        lapply(10:20, function(j) f7ListItem(letters[j]))
      )
    })
  )
)
)

```



```

    )
  )
),
server = function(input, output) {
  f7ListIndex(id = "contacts", target = "#mycontacts", label = TRUE)
  f7ListIndex(id = "othercontacts", target = "#myothercontacts", label = TRUE)
}
)

if (interactive() || identical(Sys.getenv("TESTTHAT"), "true")) app

```

f7ListItem

*Create a Framework 7 contact item***Description**

Create a Framework 7 contact item

Usage

```

f7ListItem(
  ...,
  id = NULL,
  title = NULL,
  subtitle = NULL,
  header = NULL,
  footer = NULL,
  href = NULL,
  media = NULL,
  right = NULL,
  routable = FALSE
)

```

Arguments

| | |
|----------|---|
| ... | Item text. |
| id | Optional id for item. |
| title | Item title. |
| subtitle | Item subtitle. Only work if the f7List mode is media. |
| header | Item header. Do not use when f7List mode is not NULL. |
| footer | Item footer. Do not use when f7List mode is not NULL. |
| href | Item external link. |
| media | Expect f7Icon or <code>img</code> . |
| right | Right content if any. |

| | |
|----------|--|
| routable | Works when href is not NULL. Default to FALSE. If TRUE, the list item may point to another page. See f7MultiLayout . Can also be used in combination with href = "#" to make items appear as links, but not actually navigate anywhere, which is useful for custom click events. |
|----------|--|

f7Login

Framework7 login screen

Description

Provide a UI template for authentication

f7LoginServer is demonstration module to test the [f7Login](#) page. We do not recommend using it in production, since there is absolutely no security over the passed credentials. On the JS side, the login is closed as soon as a user and password are provided but no validity checks are made.

updateF7Login toggles a login page.

Usage

```
f7Login(
  ...,
  id,
  title,
  label = "Sign In",
  footer = NULL,
  startOpen = TRUE,
  cancellable = FALSE
)

f7LoginServer(id, ignoreInit = FALSE, trigger = NULL)

updateF7Login(
  id = deprecated(),
  user = NULL,
  password = NULL,
  cancel = FALSE,
  session = shiny::getDefaultReactiveDomain()
)
```

Arguments

| | |
|--------|--|
| ... | Slot for inputs like password, text, ... |
| id | [Deprecated] . |
| title | Login page title. |
| label | Login confirm button label. |
| footer | Optional footer. |

| | |
|-------------|--|
| startOpen | Whether to open the login page at start. Default to TRUE. There are some cases where it is interesting to set up to FALSE, for instance when you want to have authentication only in a specific tab of your app (See example 2). |
| cancellable | Whether to show a cancel button to close the login modal. Default to FALSE. |
| ignoreInit | If TRUE, then, when this observeEvent is first created/initialized, ignore the handlerExpr (the second argument), whether it is otherwise supposed to run or not. The default is FALSE. |
| trigger | Reactive trigger to toggle the login page state. Useful, when one wants to set up local authentication (for a specific section). See example 2. |
| user | Value of the user input. |
| password | Value of the password input. |
| cancel | Whether to close the login. Default to FALSE. |
| session | Shiny session object. |

Note

There is an input associated with the login status, namely `input$login`. It is linked to an action button, `input$submit`, which is 0 when the application starts. As soon as the button is pressed, its value is incremented which may be used to call [updateF7Login](#). `input$user` and `input$password` contains values passed by the user in these respective fields and can be forwarded to [updateF7Login](#). `input$cancel` is incremented whenever the login is closed when cancellable. You can access the value and trigger other actions on the server, as shown in [f7LoginServer](#).

Examples

```
library(shiny)
library(shinyMobile)

app <- shinyApp(
  ui = f7Page(
    title = "Login module",
    f7SingleLayout(
      navbar = f7Navbar(
        title = "Login Example"
      ),
      toolbar = f7Toolbar(
        position = "bottom",
        f7Link(label = "Link 1", href = "https://www.google.com"),
        f7Link(label = "Link 2", href = "https://www.google.com")
      ),
      f7Login(id = "login", title = "Welcome", cancellable = TRUE),
      # main content
      f7BlockTitle(
        title = HTML(paste("Welcome", textOutput("user"))),
        size = "large"
      )
    )
  ),
  server = function(input, output, session) {
```

```
loginData <- f7LoginServer(id = "login")

exportTestValues(
  status = loginData$status(),
  user = loginData$user(),
  password = loginData$password(),
  authenticated = loginData$authenticated(),
  cancelled = loginData$cancelled()
)

output$user <- renderText({
  req(loginData$user)
  loginData$user()
})
}
)

if (interactive() || identical(Sys.getenv("TESTTHAT"), "true")) app
```

f7Margin

Framework7 margin utility

Description

f7Margin adds a margin to the given tag.

Usage

```
f7Margin(tag, side = NULL)
```

Arguments

| | |
|------|--|
| tag | Tag to apply the margin. |
| side | margin side: "left", "right", "top", "bottom", "vertical" (top and bottom), "horizontal" (left and right). Leave NULL to apply on all sides. |

Author(s)

David Granjon, <dgranjon@ymail.com>

Examples

```
if(interactive()){
  library(shiny)
  library(shinyMobile)

  cardTag <- f7Card(
    title = "Card header",
    "This is a simple card with plain text,
```

```

but cards can also contain their own header,
footer, list view, image, or any other element.",
footer = tagList(
  f7Button(color = "blue", label = "My button", href = "https://www.google.com"),
  f7Badge("Badge", color = "green")
)
)

shinyApp(
  ui = f7Page(
    title = "Margins",
    f7SingleLayout(
      navbar = f7Navbar(title = "f7Margin"),
      f7Margin(cardTag),
      cardTag
    )
  ),
  server = function(input, output) {}
)
}

```

f7MessageBar

Framework7 message bar.

Description

f7MessageBar creates a message text container to type new messages. Insert before [f7Messages](#). See examples.

updateF7MessageBar updates message bar content on the server side.

Usage

```
f7MessageBar(inputId, placeholder = "Message")
```

```

updateF7MessageBar(
  inputId,
  value = NULL,
  placeholder = NULL,
  session = shiny::getDefaultReactiveDomain()
)

```

Arguments

| | |
|-------------|---|
| inputId | f7MessageBar unique id. |
| placeholder | New placeholder value. |
| value | New value. |
| session | Shiny session object. |

Examples

```

library(shiny)
library(shinyMobile)

app <- shinyApp(
  ui = f7Page(
    title = "Update message bar",
    f7SingleLayout(
      navbar = f7Navbar(
        title = "Message bar",
        hairline = FALSE
      ),
      toolbar = f7Toolbar(
        position = "bottom",
        f7Link(label = "Link 1", href = "https://www.google.com"),
        f7Link(label = "Link 2", href = "https://www.google.com")
      ),
      # main content
      f7Segment(
        f7Button("updateMessageBar", "Update value"),
        f7Button("updateMessageBarPlaceholder", "Update placeholder")
      ),
      f7Block(
        title = "Message bar",
        f7MessageBar(inputId = "mymessagebar", placeholder = "Message")
      ),
      uiOutput("messageContent")
    )
  ),
  server = function(input, output, session) {
    output$messageContent <- renderUI({
      req(input$mymessagebar)
      tagList(
        f7BlockTitle("Message Content", size = "large"),
        f7Block(strong = TRUE, inset = TRUE, input$mymessagebar)
      )
    })
    observeEvent(input$updateMessageBar, {
      updateF7MessageBar(
        inputId = "mymessagebar",
        value = "sjsjsj"
      )
    })
    observeEvent(input$updateMessageBarPlaceholder, {
      updateF7MessageBar(
        inputId = "mymessagebar",
        placeholder = "Enter your message"
      )
    })
  }
)

```

```
if (interactive() || identical(Sys.getenv("TESTTHAT"), "true")) app
```

f7Messages

Framework7 messages container

Description

f7Messages is an empty container targeted by [updateF7Messages](#) to include multiple [f7Message](#).

f7Message creates a message item to be inserted in [f7Messages](#) with [updateF7Messages](#).

updateF7Messages add messages to a [f7Messages](#) container.

Usage

```
f7Messages(  
  id,  
  title = NULL,  
  autoLayout = TRUE,  
  newMessagesFirst = FALSE,  
  scrollMessages = TRUE,  
  scrollMessagesOnEdge = TRUE  
)  
  
f7Message(  
  text,  
  name,  
  type = c("sent", "received"),  
  header = NULL,  
  footer = NULL,  
  avatar = NULL,  
  textHeader = NULL,  
  textFooter = NULL,  
  image = NULL,  
  imageSrc = NULL,  
  cssClass = NULL  
)  
  
updateF7Messages(  
  id,  
  messages,  
  showTyping = FALSE,  
  session = shiny::getDefaultReactiveDomain()  
)
```

Arguments

id Reference to [f7Messages](#) container.

| | |
|----------------------|---|
| title | Optional messages title. |
| autoLayout | Enable Auto Layout to add all required additional classes automatically based on passed conditions. |
| newMessagesFirst | Enable if you want to use new messages on top, instead of having them on bottom. |
| scrollMessages | Enable/disable messages auto scrolling when adding new message. |
| scrollMessagesOnEdge | If enabled then messages auto scrolling will happen only when user is on top/bottom of the messages view. |
| text | Message text. |
| name | Sender name. |
| type | Message type - sent or received. |
| header | Single message header. |
| footer | Single message footer. |
| avatar | Sender avatar URL string. |
| textHeader | Message text header. |
| textFooter | Message text footer. |
| image | Message image HTML string, e.g. <code></code> . Can be used instead of imageSrc parameter. |
| imageSrc | Message image URL string. Can be used instead of image parameter. |
| cssClass | Additional CSS class to set on message HTML element. |
| messages | List of f7Message . |
| showTyping | Show typing when a new message comes. Default to FALSE. Does not work yet... |
| session | Shiny session object |

Examples

```

library(shiny)
library(shinyMobile)

app <- shinyApp(
  ui = f7Page(
    title = "Messages",
    f7SingleLayout(
      navbar = f7Navbar(
        title = "Messages",
        hairline = FALSE
      ),
      toolbar = f7MessageBar(inputId = "mymessagebar", placeholder = "Message"),
      # main content
      f7Messages(id = "mymessages", title = "My message")
    )
  ),

```



```

server = function(input, output, session) {
  # Send a message
  observeEvent(input[["mymessagebar-send"]], {
    updateF7Messages(
      id = "mymessages",
      list(
        f7Message(
          text = input$mymessagebar,
          name = "David",
          type = "sent",
          header = "Message Header",
          footer = "Message Footer",
          textHeader = "Text Header",
          textFooter = "text Footer",
          avatar = "https://cdn.framework7.io/placeholder/people-100x100-7.jpg"
        )
      )
    )
  })

  # Receive a message
  observeEvent(TRUE, {
    updateF7Messages(
      id = "mymessages",
      showTyping = FALSE, # DOES NOT WORK YET WHEN TRUE ...
      list(
        f7Message(
          text = "Some message",
          name = "Victor",
          type = "received",
          avatar = "https://cdn.framework7.io/placeholder/people-100x100-9.jpg"
        )
      )
    )
  })
}

if (interactive() || identical(Sys.getenv("TESTTHAT"), "true")) app

```

Description

[Experimental] Experimental multi pages layout. This has to be used with the brochure R package. See in the corresponding pkgdown article.

Usage

```
f7MultiLayout(
  ...,
  toolbar = NULL,
  title = NULL,
  options = f7DefaultOptions(),
  allowPWA = FALSE,
  basepath = "/"
)
```

Arguments

| | |
|---------|---|
| ... | Pages. Must be an element like <code>shiny::tags\$div(class = "page", ...)</code> |
| toolbar | Contrary to f7SingleLayout or any other layout, the multi page layout can have a common toolbar for all pages. See more at https://framework7.io/docs/toolbar-tabbar#common-toolbar . You can pass f7Toolbar in this slot or f7Tabs but if you do so, don't pass any toolbar in the different pages elements. |
| title | Page title. |
| options | shinyMobile configuration. See f7DefaultOptions and https://framework7.io/docs/app.html . Below are the most notable options. General options: <ul style="list-style-type: none"> • theme: App skin: "ios", "md", or "auto". • dark: Dark layout. TRUE, FALSE, or "auto". The default is "auto". If set to "auto" automatically enables dark theme based on user system color scheme preference. • skeletonsOnLoad: Whether to display skeletons on load. This is a preloading effect. Not compatible with preloader. • preloader: Loading spinner. Not compatible with skeletonsOnLoad. • filled: Whether to fill the f7Navbar and f7Toolbar with the current selected color. FALSE by default. • color: Color theme: See https://framework7.io/docs/color-themes.html. Expect a name like blue, red or hex code like #FF0000. If NULL, use the default color. If a name is specified it must be accepted either by col2hex or getF7Colors (valid Framework 7 color names). • pullToRefresh: Whether to active the pull to refresh feature. Default to FALSE. See https://framework7.io/docs/pull-to-refresh#examples. • iosTranslucentBars: Enable translucent effect (blur background) on navigation bars for iOS theme (on iOS devices). FALSE by default. Touch module options https://framework7.io/docs/app#param-touch : <ul style="list-style-type: none"> • touchClicksDistanceThreshold: Distance threshold (in px) to prevent short swipes. So if tap/move distance is larger than this value then "click" will not be triggered. • tapHold: It triggers (if enabled) after a sustained, complete touch event. By default it is enabled. See f7TapHold for usage. • tapHoldDelay: Determines how long (in ms) the user must hold their tap before the taphold event is fired on the target element. Default to 750 ms. |

- `tapHoldPreventClicks`: When enabled (by default), then click event will not be fired after tap hold event.
- `iosTouchRipple`: Default to FALSE. Enables touch ripple effect for iOS theme.
- `mdTouchRipple`: Default to TRUE. Enables touch ripple effect for MD theme.

Navbar options <https://framework7.io/docs/navbar#navbar-app-parameters>:

- `iosCenterTitle`: Default to TRUE. When enabled then it will try to position title at the center in iOS theme. Sometime (with some custom design) it may not needed.
- `hideOnPageScroll`: Default to FALSE. Will hide Navbars on page scroll.

Toolbar options <https://framework7.io/docs/toolbar-tabbar#toolbar-app-parameters>:

- `hideOnPageScroll`: Default to FALSE. Will hide tabs on page scroll.

In any case, you must follow the same structure as provided in the function arguments.

| | |
|-----------------------|---|
| <code>allowPWA</code> | Whether to include PWA dependencies. Default to FALSE. |
| <code>basepath</code> | Useful when the app is deployed on a server like <code>https://user.shinyapps.io/base_path</code> . |

f7Navbar

Framework7 Navbar

Description

Build a navbar layout element to insert in `f7SingleLayout`, `f7TabLayout` or `f7SplitLayout`.

`updateF7Navbar` toggles an `f7Navbar` component from the server.

Usage

```
f7Navbar(
  ...,
  subNavbar = NULL,
  title = NULL,
  subtitle = deprecated(),
  hairline = TRUE,
  shadow = deprecated(),
  bigger = FALSE,
  transparent = FALSE,
  leftPanel = FALSE,
  rightPanel = FALSE
)

updateF7Navbar(
  animate = TRUE,
  hideStatusbar = FALSE,
  session = shiny::getDefaultReactiveDomain()
)
```

Arguments

| | |
|---------------|--|
| ... | Slot for f7SearchbarTrigger . Not compatible with f7Panel . |
| subNavbar | f7SubNavbar slot, if any. |
| title | Navbar title. |
| subtitle | [Deprecated] : removed from Framework7. |
| hairline | Whether to display a thin border on the top of the navbar. TRUE by default, for ios. |
| shadow | [Deprecated] : removed from Framework7. |
| bigger | Whether to display bigger title. FALSE by default. Title becomes smaller when scrolling down the page. |
| transparent | Whether the navbar should be transparent. FALSE by default. Only works if bigger is TRUE. |
| leftPanel | Whether to enable the left panel. FALSE by default. You can also pass a list of shiny tag with <code>shiny::tagList</code> , such as an icon or text. This is useful when using the yet experimental routable API with f7MultiLayout . |
| rightPanel | Whether to enable the right panel. FALSE by default. You can also pass a list of shiny tags with <code>shiny::tagList</code> , such as an icon or text. This is useful when using the yet experimental routable API with f7MultiLayout . |
| animate | Whether it should be hidden with animation or not. By default is TRUE. |
| hideStatusBar | When FALSE (default) it hides navbar partially keeping space required to cover statusbar area. Otherwise, navbar will be fully hidden. |
| session | Shiny session object. |

Note

Currently, bigger parameters does mess with the CSS.

Author(s)

David Granjon, <dgranjon@ymail.com>

Examples

```
library(shiny)
library(shinyMobile)

app <- shinyApp(
  ui = f7Page(
    title = "Sub Navbar",
    options = list(
      dark = FALSE,
      navbar = list(
        hideOnPageScroll = TRUE,
        mdCenterTitle = TRUE
      )
    )
  ),
```

```

f7SingleLayout(
  panels = tagList(
    f7Panel(
      title = "Left Panel",
      side = "left",
      f7Block("Blabla"),
      effect = "cover"
    ),
    f7Panel(
      title = "Right Panel",
      side = "right",
      f7Block("Blabla"),
      effect = "cover"
    )
  ),
  navbar = f7Navbar(
    subNavbar = f7SubNavbar(
      f7Button(label = "My button"),
      f7Button(label = "My button"),
      f7Button(label = "My button")
    ),
    title = "Title",
    leftPanel = TRUE,
    rightPanel = TRUE
  ),
  f7Block(f7Button(inputId = "toggle", "Toggle navbar")),
  f7Block(
    lapply(1:20, f7Card)
  )
),
server = function(input, output, session) {
  observeEvent(input$toggle, {
    updateF7Navbar()
  })
}
)

if (interactive() || identical(Sys.getenv("TESTTHAT"), "true")) app

```

f7Next

Framework7 next button

Description

[f7Next](#) is a button to go next in [f7Tabs](#).

Usage

```
f7Next(targetId)
```

Arguments

targetId [f7Tabs](#) id.

f7Notif *Framework7 notification*

Description

Notification with title, text, icon and more.

Usage

```
f7Notif(
  text,
  icon = NULL,
  title = NULL,
  titleRightText = NULL,
  subtitle = NULL,
  closeTimeout = 5000,
  closeButton = FALSE,
  closeOnClick = TRUE,
  swipeToClose = TRUE,
  ...,
  session = shiny::getDefaultReactiveDomain()
)
```

Arguments

| | |
|----------------|--|
| text | Notification content. |
| icon | Notification icon. |
| title | Notification title. |
| titleRightText | Notification right text. |
| subtitle | Notification subtitle |
| closeTimeout | Time before notification closes. |
| closeButton | Whether to display a close button. FALSE by default. |
| closeOnClick | Whether to close the notification on click. TRUE by default. |
| swipeToClose | If enabled, notification can be closed by swipe gesture. |
| ... | Other options. See https://framework7.io/docs/notification.html . |
| session | shiny session. |

Examples

```
if (interactive()) {
  library(shiny)
  library(shinyMobile)
  shinyApp(
    ui = f7Page(
      title = "My app",
      f7SingleLayout(
        navbar = f7Navbar(title = "f7Notif"),
        f7Block(f7Button(inputId = "goButton", "Go!"))
      )
    ),
    server = function(input, output, session) {
      observeEvent(input$goButton, {
        f7Notif(
          text = "test",
          icon = f7Icon("bolt_fill"),
          title = "Notification",
          subtitle = "A subtitle",
          titleRightText = "now"
        )
      })
    }
  )
}
```

f7Padding

Framework7 padding utility

Description

f7Padding adds padding to the given tag.

Usage

```
f7Padding(tag, side = NULL)
```

Arguments

| | |
|------|---|
| tag | Tag to apply the padding. |
| side | padding side: "left", "right", "top", "bottom", "vertical" (top and bottom), "horizontal" (left and right). Leave NULL to apply on all sides. |

Author(s)

David Granjon, <dgranjon@ymail.com>

Examples

```

if(interactive()){
  library(shiny)
  library(shinyMobile)

  cardTag <- f7Card(
    title = "Card header",
    f7Padding(
      p("The padding is applied here.")
    ),
    footer = tagList(
      f7Button(color = "blue", label = "My button", href = "https://www.google.com"),
      f7Badge("Badge", color = "green")
    )
  )

  shinyApp(
    ui = f7Page(
      title = "Padding",
      f7SingleLayout(navbar = f7Navbar(title = "f7Padding"), cardTag)
    ),
    server = function(input, output) {}
  )
}

```

f7Page

Framework7 page container

Description

f7Page is the main app container.

Usage

```
f7Page(..., title = NULL, options = f7DefaultOptions(), allowPWA = FALSE)
```

Arguments

| | |
|---------|--|
| ... | Slot for shinyMobile skeleton elements: f7SingleLayout , f7TabLayout , f7SplitLayout . |
| title | Page title. |
| options | shinyMobile configuration. See f7DefaultOptions and https://framework7.io/docs/app.html . Below are the most notable options. General options: <ul style="list-style-type: none"> • theme: App skin: "ios", "md", or "auto". • dark: Dark layout. TRUE, FALSE, or "auto". The default is "auto". If set to "auto" automatically enables dark theme based on user system color scheme preference. |

- `skeletonsOnLoad`: Whether to display skeletons on load. This is a preloading effect. Not compatible with `preloader`.
- `preloader`: Loading spinner. Not compatible with `skeletonsOnLoad`.
- `filled`: Whether to fill the `f7Navbar` and `f7Toolbar` with the current selected color. `FALSE` by default.
- `color`: Color theme: See <https://framework7.io/docs/color-themes.html>. Expect a name like `blue`, `red` or hex code like `#FF0000`. If `NULL`, use the default color. If a name is specified it must be accepted either by `col2hex` or `getF7Colors` (valid Framework 7 color names).
- `pullToRefresh`: Whether to active the pull to refresh feature. Default to `FALSE`. See <https://framework7.io/docs/pull-to-refresh#examples>.
- `iosTranslucentBars`: Enable translucent effect (blur background) on navigation bars for iOS theme (on iOS devices). `FALSE` by default.

Touch module options <https://framework7.io/docs/app#param-touch>:

- `touchClicksDistanceThreshold`: Distance threshold (in px) to prevent short swipes. So if tap/move distance is larger than this value then "click" will not be triggered.
- `tapHold`: It triggers (if enabled) after a sustained, complete touch event. By default it is enabled. See `f7TapHold` for usage.
- `tapHoldDelay`: Determines how long (in ms) the user must hold their tap before the `taphold` event is fired on the target element. Default to 750 ms.
- `tapHoldPreventClicks`: When enabled (by default), then click event will not be fired after tap hold event.
- `iosTouchRipple`: Default to `FALSE`. Enables touch ripple effect for iOS theme.
- `mdTouchRipple`: Default to `TRUE`. Enables touch ripple effect for MD theme.

Navbar options <https://framework7.io/docs/navbar#navbar-app-parameters>:

- `iosCenterTitle`: Default to `TRUE`. When enabled then it will try to position title at the center in iOS theme. Sometime (with some custom design) it may not be needed.
- `hideOnPageScroll`: Default to `FALSE`. Will hide Navbars on page scroll.

Toolbar options <https://framework7.io/docs/toolbar-tabbar#toolbar-app-parameters>:

- `hideOnPageScroll`: Default to `FALSE`. Will hide tabs on page scroll.

In any case, you must follow the same structure as provided in the function arguments.

`allowPWA` Whether to include PWA dependencies. Default to `FALSE`.

Author(s)

David Granjon, <dgranjon@gmail.com>

f7Panel

*Framework7 panel***Description**

f7Panel is a sidebar element. It may be used as a simple sidebar or as a container for [f7PanelMenu](#) in case of [f7SplitLayout](#).

updateF7Panel toggles an [f7Panel](#) from the server.

Usage

```
f7Panel(
  ...,
  id = NULL,
  title = NULL,
  side = c("left", "right"),
  theme = deprecated(),
  effect = c("reveal", "cover", "push", "floating"),
  resizable = FALSE,
  options = list()
)

updateF7Panel(id, session = shiny::getDefaultReactiveDomain())
```

Arguments

| | |
|-----------|--|
| ... | Panel content. Slot for f7PanelMenu , if used as a sidebar. |
| id | Panel unique id. |
| title | Panel title. |
| side | Panel side: "left" or "right". |
| theme | [Deprecated] : removed from Framework7. |
| effect | Whether the panel should behave when opened: "cover", "reveal", "floating" or "push". |
| resizable | Whether to enable panel resize. FALSE by default. |
| options | Other panel options. See https://framework7.io/docs/panel#panel-parameters . |
| session | Shiny session object. |

Author(s)

David Granjon, <dgranjon@gmail.com>

Examples

```
library(shiny)
library(shinyMobile)

app <- shinyApp(
  ui = f7Page(
    title = "Panels",
    options = list(dark = FALSE),
    f7SingleLayout(
      navbar = f7Navbar(
        title = "f7Panel",
        leftPanel = TRUE,
        rightPanel = TRUE
      ),
      panels = tagList(
        f7Panel(
          id = "mypanel1",
          side = "left",
          effect = "push",
          title = "Left panel",
          resizable = TRUE,
          f7Block("A panel with push effect"),
          f7PanelMenu(
            id = "panelmenu",
            f7PanelItem(
              tabName = "tab1",
              title = "Tab 1",
              icon = f7Icon("envelope"),
              active = TRUE
            ),
            f7PanelItem(
              tabName = "tab2",
              title = "Tab 2",
              icon = f7Icon("house")
            )
          )
        ),
        f7Panel(
          id = "mypanel2",
          side = "right",
          effect = "floating",
          title = "Right panel",
          f7Block(
            "A panel with cover effect"
          ),
          options = list(swipe = TRUE)
        )
      ),
      toolbar = f7Toolbar(
        position = "bottom",
        icons = TRUE,
        f7Link(label = "Link 1", href = "https://www.google.com"),
```

```

        f7Link(label = "Link 2", href = "https://www.google.com")
      ),
      # main content
      f7Block(
        f7Button(inputId = "toggle", "Toggle panel 1")
      )
    )
  ),
  server = function(input, output, session) {
    observeEvent(input$mypanel2, {
      state <- if (input$mypanel2) "open" else "closed"

      f7Toast(
        text = paste0("Right panel is ", state),
        position = "center",
        closeTimeout = 1000,
        closeButton = FALSE
      )
    })

    observeEvent(input$toggle, {
      updateF7Panel(id = "mypanel1")
    })
  }
)

if (interactive() || identical(Sys.getenv("TESTTHAT"), "true")) app

```

f7PanelMenu

Framework7 sidebar menu

Description

f7PanelMenu creates a menu for [f7Panel](#). It may contain multiple [f7PanelItem](#).
 f7PanelItem creates a Framework7 sidebar menu item for [f7SplitLayout](#).

Usage

```

f7PanelMenu(
  ...,
  id = NULL,
  mode = "links",
  inset = FALSE,
  outline = FALSE,
  dividers = FALSE,
  strong = FALSE
)

f7PanelItem(title, tabName, icon = NULL, active = FALSE)

```

Arguments

| | |
|----------|--|
| ... | Slot for f7PanelItem . |
| id | Unique id to access the currently selected item. |
| mode | List mode. NULL, "simple", "links", "media" or "contacts". |
| inset | Whether to display a card border. FALSE by default. |
| outline | Outline style. Default to FALSE. |
| dividers | Dividers style. Default to FALSE. |
| strong | Strong style. Default to FALSE. |
| title | Item name. |
| tabName | Item unique tabName. Must correspond to what is passed to f7Item . |
| icon | Item icon. |
| active | Whether the item is active at start. Default to FALSE. |

Author(s)

David Granjon, <dgranjon@gmail.com>

f7PhotoBrowser

Framework7 photo browser

Description

A nice photo browser.

Usage

```
f7PhotoBrowser(
  photos,
  theme = c("light", "dark"),
  type = c("popup", "standalone", "page"),
  ...,
  id = NULL,
  session = shiny::getDefaultReactiveDomain()
)
```

Arguments

| | |
|---------|---|
| photos | List of photos |
| theme | Browser theme: choose either light or dark. |
| type | Browser type: choose among c("popup", "standalone", "page"). |
| ... | Other options to pass to the photo browser. See https://framework7.io/docs/photo-browser#photo-browser-parameters for more details. |
| id | Unique id. Useful to leverage updateF7Entity on the server. |
| session | Shiny session object. |

Examples

```

library(shiny)
library(shinyMobile)

app <- shinyApp(
  ui = f7Page(
    title = "f7PhotoBrowser",
    f7SingleLayout(
      navbar = f7Navbar(title = "f7PhotoBrowser"),
      f7Block(
        f7Button(inputId = "togglePhoto", "Open photo")
      )
    )
  ),
  server = function(input, output, session) {
    observeEvent(input$togglePhoto, {
      f7PhotoBrowser(
        id = "photobrowser1",
        theme = "dark",
        type = "page",
        photos = list(
          list(url = "https://cdn.framework7.io/placeholder/sports-1024x1024-1.jpg"),
          list(url = "https://cdn.framework7.io/placeholder/sports-1024x1024-2.jpg"),
          list(url = "https://cdn.framework7.io/placeholder/sports-1024x1024-3.jpg",
            caption = "Me cycling")
          )
        ),
        thumbs = c(
          "https://cdn.framework7.io/placeholder/sports-1024x1024-1.jpg",
          "https://cdn.framework7.io/placeholder/sports-1024x1024-2.jpg",
          "https://cdn.framework7.io/placeholder/sports-1024x1024-3.jpg"
        )
      )
    })
  }
)

if (interactive() || identical(Sys.getenv("TESTTHAT"), "true")) app

```

f7Picker

Framework7 picker input

Description

f7Picker generates a picker input.

updateF7Picker changes the value of a picker input on the client.

Usage

```
f7Picker(
```

```

    inputId,
    label,
    placeholder = NULL,
    value = choices[1],
    choices,
    rotateEffect = TRUE,
    openIn = "auto",
    scrollToInput = FALSE,
    closeByOutsideClick = TRUE,
    toolbar = TRUE,
    toolbarCloseText = "Done",
    sheetSwipeToClose = FALSE,
    style = list(inset = FALSE, outline = FALSE, strong = FALSE, dividers = FALSE),
    ...
)

updateF7Picker(
  inputId,
  value = NULL,
  choices = NULL,
  rotateEffect = NULL,
  openIn = NULL,
  scrollToInput = NULL,
  closeByOutsideClick = NULL,
  toolbar = NULL,
  toolbarCloseText = NULL,
  sheetSwipeToClose = NULL,
  ...,
  session = shiny::getDefaultReactiveDomain()
)

```

Arguments

| | |
|----------------------------------|---|
| <code>inputId</code> | Picker input id. |
| <code>label</code> | Picker label. |
| <code>placeholder</code> | Text to write in the container. |
| <code>value</code> | Picker initial value, if any. |
| <code>choices</code> | Picker choices. |
| <code>rotateEffect</code> | Enables 3D rotate effect. Default to TRUE. |
| <code>openIn</code> | Can be auto, popover (to open picker in popover), sheet (to open in sheet modal). In case of auto will open in sheet modal on small screens and in popover on large screens. Default to auto. |
| <code>scrollToInput</code> | Scroll viewport (page-content) to input when picker opened. Default to FALSE. |
| <code>closeByOutsideClick</code> | If enabled, picker will be closed by clicking outside of picker or related input element. Default to TRUE. |

| | |
|-------------------|--|
| toolbar | Enables picker toolbar. Default to TRUE. |
| toolbarCloseText | Text for Done/Close toolbar button. |
| sheetSwipeToClose | Enables ability to close Picker sheet with swipe. Default to FALSE. |
| style | Input style. Inherit from <code>f7List</code> options such as outline, inset, strong and dividers. |
| ... | Other options to pass to the picker. See https://framework7.io/docs/picker#picker-parameters . |
| session | The Shiny session object, usually the default value will suffice. |

Author(s)

David Granjon, <dgranjon@ymail.com>

Examples

```
library(shiny)
library(shinyMobile)

app <- shinyApp(
  ui = f7Page(
    title = "My app",
    f7TabLayout(
      navbar = f7Navbar(title = "Update pickers"),
      f7Tabs(
        f7Tab(
          title = "Standalone",
          tabName = "standalone",
          f7Segment(
            f7Button(inputId = "update", label = "Update picker"),
            f7Button(
              inputId = "removeToolbar",
              label = "Remove picker toolbars",
              color = "red"
            )
          )
        ),
        f7Picker(
          inputId = "picker",
          placeholder = "Some text here!",
          label = "Picker Input",
          choices = c("a", "b", "c"),
          options = list(sheetPush = TRUE),
          style = list(strong = TRUE)
        ),
        f7Block(verbatimTextOutput("pickerval"))
      ),
      f7Tab(
        title = "List",
        tabName = "list",
        f7List(
```



```

        strong = TRUE,
        f7Picker(
          inputId = "picker2",
          placeholder = "Some text here!",
          label = "Picker Input",
          choices = c("a", "b", "c"),
          options = list(sheetPush = TRUE)
        )
      ),
      f7Block(verbatimTextOutput("pickerval2"))
    )
  )
),
server = function(input, output, session) {
  output$pickerval <- renderText(input$picker)
  output$pickerval2 <- renderText(input$picker2)

  observeEvent(input$update, {
    updateF7Picker(
      inputId = "picker",
      value = "b",
      choices = letters,
      openIn = "sheet",
      toolbarCloseText = "Close me",
      sheetSwipeToClose = TRUE
    )
  })

  observeEvent(input$removeToolbar, {
    updateF7Picker(
      inputId = "picker",
      value = "b",
      choices = letters,
      openIn = "sheet",
      toolbar = FALSE
    )
  })
}
)

if (interactive() || identical(Sys.getenv("TESTTHAT"), "true")) app

```

Description

f7Popup creates a popup window with any UI content that pops up over App's main content. Popup as all other overlays is part of so called "Temporary Views".

Usage

```
f7Popup(
  ...,
  id,
  title = NULL,
  backdrop = TRUE,
  closeByBackdropClick = TRUE,
  closeOnEscape = FALSE,
  animate = TRUE,
  swipeToClose = FALSE,
  fullsize = FALSE,
  closeButton = TRUE,
  push = TRUE,
  page = FALSE,
  session = shiny::getDefaultReactiveDomain()
)
```

Arguments

| | |
|----------------------|---|
| ... | UI elements for the body of the popup window. |
| id | Popup unique id. Useful if you want to access the popup state. <code>input\$<id></code> is TRUE when the popup is opened and inversely. |
| title | Title for the popup window, use NULL for no title. |
| backdrop | Enables Popup backdrop (dark semi transparent layer behind). Default to TRUE. |
| closeByBackdropClick | When enabled, popup will be closed on backdrop click. Default to TRUE. |
| closeOnEscape | When enabled, popup will be closed on ESC keyboard key press. Default to FALSE. |
| animate | Whether the Popup should be opened/closed with animation or not. Default to TRUE. |
| swipeToClose | Whether the Popup can be closed with swipe gesture. Can be true to allow to close popup with swipes to top and to bottom. Default to FALSE. |
| fullsize | Open popup in full width or not. Default to FALSE. |
| closeButton | Add or not a button to easily close the popup. Default to TRUE. |
| push | Push effect. Default to TRUE. |
| page | Allow content to be scrollable, as a page. Default to FALSE. |
| session | Shiny session object. |

Examples

```
library(shiny)
library(shinyMobile)

app <- shinyApp(
  ui = f7Page(
```

```

title = "Popup",
f7SingleLayout(
  navbar = f7Navbar(
    title = "f7Popup"
  ),
  f7Block(f7Button("toggle1", "Toggle Popup")),
  br(),
  f7Block(f7Button("toggle2", "Toggle Page Popup"))
)
),
server = function(input, output, session) {

  output$res1 <- renderPrint(input$text)
  output$res2 <- renderPrint(input$text2)

  observeEvent(input$toggle1, {
    f7Popup(
      id = "popup1",
      title = "My first popup",
      f7Text(
        "text1", "Popup content",
        "This is my first popup ever, I swear!"
      ),
      verbatimTextOutput("res1")
    )
  })

  observeEvent(input$toggle2, {
    f7Popup(
      id = "popup2",
      title = "My first popup",
      page = TRUE,
      f7Text(
        "text2", "Popup content",
        "Look at me, I can scroll!"
      ),
      verbatimTextOutput("res2"),
      p("Lorem ipsum dolor sit amet, consectetur adipiscing elit. Suspendisse
hendrerit magna non sem iaculis, ac rhoncus est pulvinar. Interdum et
malesuada fames ac ante ipsum primis in faucibus. In sagittis vel lacus
ac bibendum. Maecenas mollis, diam nec fermentum sollicitudin, massa
lectus ullamcorper orci, in laoreet lectus quam nec lacus.
Nulla sollicitudin imperdiet metus, quis mollis justo finibus varius.
In mattis malesuada enim in tincidunt. Nulla vehicula dui lacus,
iaculis condimentum dui dapibus ac. Cras elit nunc, auctor vestibulum
odio id, iaculis posuere arcu. Mauris dignissim id lectus sit amet
vestibulum. Nam rutrum sit amet augue vel interdum. Donec sed orci vitae
eros eleifend posuere vitae id nibh. Donec faucibus erat in placerat
feugiat. Sed sodales facilisis eros, porta viverra purus pretium eu.
Morbi vehicula metus lacus, id commodo mauris posuere nec. Vivamus
ornare et lacus et lobortis. Etiam tristique elit id eros ornare,
vel faucibus mauris hendrerit. Nulla elit nulla, consequat sit amet
neque et, ultrices elementum diam. Etiam dignissim elit a arcu pulvinar,

```

```

    ut dapibus elit maximus. Mauris ultricies nulla in mauris laoreet, at
    lacinia lorem maximus. Nulla sed enim diam. In ac felis dignissim,
    euismod augue nec, tempus augue. Maecenas eget aliquam mi.
    In tincidunt massa a velit suscipit, ac dapibus mi laoreet. Vestibulum
    lacinia nulla lorem, nec blandit quam sollicitudin at. Pellentesque
    in vehicula lacus. Etiam vitae lectus malesuada, hendrerit mauris eu,
    placerat elit. Mauris vehicula dictum pharetra. Etiam interdum vehicula
    urna, ac blandit lectus posuere id. Nullam facilisis tincidunt sem et
    pretium. Praesent pulvinar feugiat augue, quis pretium nunc vestibulum a.
    Morbi id eros eget lectus placerat placerat. Morbi dapibus viverra
    orci nec pellentesque. Vestibulum mollis gravida sem, quis tincidunt
    sem maximus gravida. Nam id egestas augue, sit amet egestas orci. Duis
    porttitor lectus sit amet efficitur auctor. Quisque dui ante, eleifend
    eget nibh a, tincidunt interdum nisi. Integer varius tempor erat, in
    commodo neque elementum ut. Maecenas eu lorem ultrices, posuere neque ac,
    aliquam ante. Maecenas eu volutpat arcu. Morbi hendrerit sem sed vehicula
    sodales. Quisque ultrices massa erat, vel accumsan risus vehicula eu.
    Donec laoreet aliquet est, a consequat odio viverra lacinia. Suspendisse
    id iaculis risus. Vestibulum posuere dignissim lacus quis ornare. Nam
    dapibus efficitur neque sed tristique."
  )
)
})

observeEvent(input$popup1, {
  popupStatus <- if (input$popup1) "opened" else "closed"

  f7Toast(
    position = "top",
    text = paste("Popup1 is", popupStatus)
  )
})
}
)

if (interactive() || identical(Sys.getenv("TESTTHAT"), "true")) app

```

f7Progress

Framework7 progress bar

Description

f7Progress creates a progress bar.

updateF7Progress update a framework7 progress bar from the server side

Usage

```
f7Progress(id, value = NULL, color)
```

```
updateF7Progress(id, value, session = shiny::getDefaultReactiveDomain())
```

Arguments

| | |
|---------|---|
| id | Progress id. Must be unique. |
| value | Progress value. Between 0 and 100. If NULL the progress bar is infinite. |
| color | Progress color. See https://framework7.io/docs/progressbar.html . |
| session | Shiny session object. |

Examples

```
library(shiny)
library(shinyMobile)

app <- shinyApp(
  ui = f7Page(
    title = "Update Progress",
    f7SingleLayout(
      navbar = f7Navbar(title = "f7Progress"),
      f7BlockTitle("Progress with value"),
      f7Block(
        f7Progress(id = "pg1", value = 10, color = "blue")
      ),
      f7Slider(
        inputId = "obs",
        label = "Progress value",
        max = 100,
        min = 0,
        value = 50,
        scale = TRUE
      ),
      f7BlockTitle("Infinite progress"),
      f7Block(
        f7Progress(id = "pg2", value = NULL, color = "red")
      )
    )
  ),
  server = function(input, output, session) {
    observeEvent(input$obs, {
      updateF7Progress(id = "pg1", value = input$obs)
    })
  }
)

if (interactive() || identical(Sys.getenv("TESTTHAT"), "true")) app
```

f7Radio

*Framework7 radio input***Description**

f7Radio creates a radio button input.

updateF7Radio updates a radio button input.

Usage

```
f7Radio(
  inputId,
  label,
  choices = NULL,
  selected = NULL,
  position = c("left", "right"),
  style = list(inset = FALSE, outline = FALSE, dividers = FALSE, strong = FALSE)
)

updateF7Radio(
  inputId,
  label = NULL,
  choices = NULL,
  selected = NULL,
  session = shiny::getDefaultReactiveDomain()
)

f7RadioChoice(..., title, subtitle = NULL, after = NULL)
```

Arguments

| | |
|----------|--|
| inputId | Input id. |
| label | Input label |
| choices | List of choices. Can be a simple vector or named list or a list of f7RadioChoice or f7CheckboxChoice |
| selected | Selected element. NULL by default. If you pass f7RadioChoice or f7CheckboxChoice in choices, selected must be a numeric value corresponding to the index of the element to select. |
| position | Check mark side. "left" or "right". |
| style | Input style. Inherit from f7List options such as outline, inset, strong and dividers. |
| session | Shiny session object. |
| ... | Choice content. Text is striped if too long. |
| title | Item title. |
| subtitle | Item subtitle. |
| after | Display at the right of title. |

Examples

```
library(shiny)
library(shinyMobile)

app <- shinyApp(
  ui = f7Page(
    title = "Update radio",
```

```

f7SingleLayout(
  navbar = f7Navbar(title = "Update f7Radio"),
  f7Block(f7Button("update", "Update radio")),
  f7Block(
    f7Radio(
      inputId = "radio",
      label = "Choose a fruit:",
      choices = c("banana", "apple", "peach"),
      selected = "apple",
      position = "right"
    ),
    textOutput("res")
  ),
  f7Block(
    f7Radio(
      inputId = "radio2",
      label = "Custom choices",
      choices = list(
        f7RadioChoice(
          "Lorem ipsum dolor sit amet, consectetur adipiscing elit.
          Nulla sagittis tellus ut turpis condimentum,
          ut dignissim lacus tincidunt",
          title = "Choice 1",
          subtitle = "David",
          after = "March 16, 2024"
        ),
        f7RadioChoice(
          "Cras dolor metus, ultrices condimentum sodales sit
          amet, pharetra sodales eros. Phasellus vel felis tellus.
          Mauris rutrum ligula nec dapibus feugiat",
          title = "Choice 2",
          subtitle = "Veerle",
          after = "March 17, 2024"
        )
      ),
      selected = 2,
      style = list(
        outline = TRUE,
        strong = TRUE,
        inset = TRUE,
        dividers = TRUE
      )
    ),
    textOutput("res2")
  )
),
server = function(input, output, session) {
  output$res <- renderText(input$radio)
  output$res2 <- renderText(input$radio2)

  observeEvent(input$update, {
    updateF7Radio(

```

```

        session,
        inputId = "radio",
        label = "New label",
        choices = colnames(mtcars),
        selected = colnames(mtcars)[1]
    )
})
}
)

if (interactive() || identical(Sys.getenv("TESTTHAT"), "true")) app

```

f7Searchbar

Framework 7 searchbar

Description

Searchbar to filter elements in a page.

f7SearchbarTrigger: Element that triggers the searchbar.

f7HideOnSearch: elements with such class on page will be hidden during search

f7HideOnEnable: elements with such class on page will be hidden when searchbar is enabled

f7NotFound: elements with such class are hidden by default and become visible when there is not any search results

f7Found: elements with such class are visible by default and become hidden when there is not any search results.

f7SearchIgnore: searchbar will not consider this elements in search results.

Usage

```

f7Searchbar(
  id,
  placeholder = "Search",
  expandable = FALSE,
  inline = FALSE,
  options = NULL
)

f7SearchbarTrigger(targetId)

f7HideOnSearch(tag)

f7HideOnEnable(tag)

f7NotFound(tag)

f7Found(tag)

f7SearchIgnore(tag)

```


Arguments

| | |
|-------------|---|
| id | Necessary when using f7SearchbarTrigger . NULL otherwise. |
| placeholder | Searchbar placeholder. |
| expandable | Whether to enable the searchbar with a target link, in the navbar. See f7SearchbarTrigger . |
| inline | Useful to add an f7Searchbar in a navbar. Notice that utilities like f7HideOnSearch and f7NotFound are not compatible with this mode. |
| options | Search bar options. See https://framework7.io/docs/searchbar.html#searchbar-parameters . If no options are provided, the searchbar will search in list elements by item title. This may be changed by updating the default searchContainer and searchIn. |
| targetId | Id of the f7Searchbar . |
| tag | tag to ignore. |

Examples

```

library(shiny)
library(shinyMobile)

cities <- names(precip)

app <- shinyApp(
  ui = f7Page(
    title = "Expandable searchbar",
    f7SingleLayout(
      navbar = f7Navbar(
        title = "f7Searchbar with trigger",
        subNavbar = f7SubNavbar(
          f7Searchbar(id = "search1", expandable = TRUE)
        )
      ),
      f7Block(
        f7SearchbarTrigger(targetId = "search1")
      ) %>% f7HideOnSearch(),
      f7List(
        lapply(seq_along(cities), function(i) {
          f7ListItem(cities[i])
        })
      ) %>% f7Found(),
      f7Block(
        p("Nothing found")
      ) %>% f7NotFound()
    )
  ),
  server = function(input, output) {}
)

if (interactive() || identical(Sys.getenv("TESTTHAT"), "true")) app

```

`f7Select`*Framework7 select input*

Description

`f7Select` creates a select input.

`updateF7Select` changes the value of a select input on the client

Usage

```
f7Select(  
  inputId,  
  label,  
  choices,  
  selected = NULL,  
  width = NULL,  
  style = list(media = NULL, description = NULL, outline = FALSE)  
)
```

```
updateF7Select(  
  inputId,  
  selected = NULL,  
  session = shiny::getDefaultReactiveDomain()  
)
```

Arguments

| | |
|-----------------------|---|
| <code>inputId</code> | Text input id. |
| <code>label</code> | Text input label. |
| <code>choices</code> | Select input choices. |
| <code>selected</code> | Select input default selected value. |
| <code>width</code> | The width of the input, e.g. 400px, or 100%. |
| <code>style</code> | Input style. A list with <code>media</code> (image or icon), <code>description</code> (text), <code>floating</code> , <code>outline</code> and <code>clearable</code> (booleans). |
| <code>session</code> | The Shiny session object, usually the default value will suffice. |

Note

Contrary to [f7Text](#), [f7Select](#) can't be cleared and label can't float.

Examples

```
library(shiny)  
library(shinyMobile)
```

```

app <- shinyApp(
  ui = f7Page(
    title = "f7Select",
    f7SingleLayout(
      navbar = f7Navbar(title = "updateF7Select"),
      f7Card(
        f7Button(inputId = "update", label = "Update select"),
        br(),
        f7List(
          f7Select(
            inputId = "select",
            label = "Choose a variable:",
            choices = colnames(mtcars)[-1],
            selected = "hp",
            style = list(
              description = "A basic select input",
              media = f7Icon("car_fill"),
              outline = TRUE
            )
          )
        ),
        verbatimTextOutput("test")
      )
    ),
  server = function(input, output, session) {
    output$test <- renderPrint(input$select)

    observeEvent(input$update, {
      updateF7Select(
        inputId = "select",
        selected = "gear"
      )
    })
  }
)

if (interactive() || identical(Sys.getenv("TESTTHAT"), "true")) app

```

f7Sheet

Framework7 sheet

Description

f7Sheet creates an f7 sheet modal window. The sheet modal has to be used in combination with [updateF7Sheet](#). If you need another trigger, simply add `data-sheet = paste0("#", id)` to the tag of your choice (a button), where id refers to the sheet unique id as well as the class "sheet-open". Inversely, if you need a custom element to close a sheet, give it the "sheet-close" class.

updateF7Sheet toggles an [f7Sheet](#) on the client.

Usage

```
f7Sheet(
  ...,
  id,
  hiddenItems = NULL,
  orientation = c("top", "bottom"),
  swipeToClose = FALSE,
  swipeToStep = FALSE,
  backdrop = FALSE,
  closeByOutsideClick = TRUE,
  swipeHandler = TRUE,
  options = list()
)

updateF7Sheet(id, session = shiny::getDefaultReactiveDomain())
```

Arguments

| | |
|---------------------|---|
| ... | Sheet content. If wipeToStep is TRUE, these items will be visible at start. |
| id | Sheet unique id. |
| hiddenItems | Put items you want to hide inside. Only works when swipeToStep is TRUE. Default to NULL. |
| orientation | "top" or "bottom". |
| swipeToClose | If TRUE, it can be closed by swiping down. |
| swipeToStep | If TRUE then sheet will be opened partially, and with swipe it can be further expanded. |
| backdrop | Enables Sheet backdrop (dark semi transparent layer behind). By default it is true for MD theme and false for iOS theme. |
| closeByOutsideClick | When enabled, sheet will be closed on when click outside of it. |
| swipeHandler | Whether to display a swipe handler. TRUE by default. Need either swipeToClose or swipeToStep set to TRUE to work. |
| options | Other parameters. See https://framework7.io/docs/sheet-modal#sheet-parameters |
| session | Shiny session object |

Examples

```
library(shiny)
library(shinyMobile)

app <- shinyApp(
  ui = f7Page(
    title = "Update f7Sheet",
    f7SingleLayout(
      navbar = f7Navbar(title = "f7Sheet"),
      f7Block(f7Button(inputId = "toggle", label = "Open sheet")),
    )
  )
)
```

```

f7Sheet(
  id = "sheet",
  orientation = "bottom",
  swipeToClose = TRUE,
  swipeToStep = TRUE,
  backdrop = TRUE,
  options = list(push = TRUE, breakpoints = c(0.33, 0.66)),
  "Lorem ipsum dolor sit amet, consectetur adipiscing elit.
  Quisque ac diam ac quam euismod porta vel a nunc. Quisque sodales
  scelerisque est, at porta justo cursus ac",
  hiddenItems = tagList(
    f7Segment(
      rounded = TRUE,
      f7Button(color = "blue", label = "My button 1", rounded = TRUE),
      f7Button(color = "green", label = "My button 2", rounded = TRUE),
      f7Button(color = "yellow", label = "My button 3", rounded = TRUE)
    ),
    f7Grid(
      cols = 1,
      f7Gauge(
        id = "mygauge",
        type = "semicircle",
        value = 10,
        borderColor = "#2196f3",
        borderWidth = 10,
        valueFontSize = 41,
        valueTextColor = "#2196f3",
        labelText = "amount of something"
      )
    ),
    f7Slider(
      inputId = "obs",
      label = "Number of observations",
      max = 100,
      min = 0,
      value = 10,
      scale = TRUE
    ),
    plotOutput("distPlot")
  )
),
server = function(input, output, session) {
  output$distPlot <- renderPlot({
    hist(rnorm(input$obs))
  })
  observeEvent(input$obs, {
    updateF7Gauge(id = "mygauge", value = input$obs)
  })
  observeEvent(input$toggle, {
    updateF7Sheet(id = "sheet")
  })
}

```

```

    }
  )

  if (interactive() || identical(Sys.getenv("TESTTHAT"), "true")) app

```

f7SingleLayout *Framework7 single layout*

Description

f7SingleLayout provides a simple page layout.

Usage

```
f7SingleLayout(..., navbar, toolbar = NULL, panels = NULL)
```

Arguments

| | |
|---------|--|
| ... | Content. |
| navbar | Slot for f7Navbar . |
| toolbar | Slot for f7Toolbar . |
| panels | Slot for f7Panel . Wrap in tagList if multiple panels. |

Author(s)

David Granjon, <dgranjon@gmail.com>

Examples

```

if (interactive()) {
  library(shiny)
  library(shinyMobile)
  shinyApp(
    ui = f7Page(
      title = "Single layout",
      f7SingleLayout(
        navbar = f7Navbar(
          title = "Single Layout"
        ),
        toolbar = f7Toolbar(
          position = "bottom",
          f7Link(label = "Link 1", href = "https://www.google.com"),
          f7Link(label = "Link 2", href = "https://www.google.com")
        ),
        # main content
        f7Card(
          title = "Card header",
          f7Slider("obs", "Number of observations", 0, 1000, 500),
          plotOutput("distPlot"),

```

```

        footer = tagList(
          f7Button(
            color = "blue",
            label = "My button",
            href = "https://www.google.com"
          ),
          f7Badge("Badge", color = "green")
        )
      )
    )
  ),
  server = function(input, output) {
    output$distPlot <- renderPlot({
      dist <- rnorm(input$obs)
      hist(dist)
    })
  }
}

```

f7Skeleton

Framework 7 skeleton effect

Description

Nice loading overlay for UI elements. You can also set `skeletonsOnLoad TRUE` in the app main options (see example) to show skeletons on load.

Usage

```

f7Skeleton(
  target,
  effect = c("fade", "blink", "pulse"),
  duration = NULL,
  session = shiny::getDefaultReactiveDomain()
)

```

Arguments

| | |
|-----------------------|---|
| <code>target</code> | CSS selector on which to apply the effect. In general, you apply the effect on a wrapper such as a card, such that all nested elements receive the skeleton. |
| <code>effect</code> | Choose between "fade", "blink" or "pulse". |
| <code>duration</code> | Effect duration. NULL by default. If you know exactly how much time your most time consuming output takes to render, you can pass an explicit duration. In other cases, leave it to NULL. |
| <code>session</code> | Shiny session object. |

Details

This function is expected to be called from an `observeEvent`, you may also have to increase the observer priority (see example).

Examples

```
if (interactive()) {
  library(shiny)
  library(shinyMobile)

  shinyApp(
    ui = f7Page(
      title = "Skeletons",
      options = list(skeletonsOnLoad = TRUE),
      f7SingleLayout(
        navbar = f7Navbar(title = "f7Skeleton"),
        f7Block(
          f7Button("update", "Update card")
        ),
        f7Card(
          title = "Card header",
          textOutput("test"),
        ),
        f7List(
          f7ListItem(
            href = "https://www.google.com",
            title = "Item 1"
          ),
          f7ListItem(
            href = "https://www.google.com",
            title = "Item 2"
          )
        )
      )
    ),
    server = function(input, output, session) {
      txt <- eventReactive(input$update,
        {
          Sys.sleep(3)
          "This is a simple card with plain text,
          but cards can also contain their own header,
          footer, list view, image, or any other element."
        },
        ignoreNULL = FALSE
      )
      output$test <- renderText(txt())
      observeEvent(input$update,
        {
          f7Skeleton(".card", "fade")
        },
        priority = 1000
      )
    }
  )
}
```



```
    }  
  )  
}
```

f7Slider

Framework7 range slider

Description

f7Slider creates a f7 slider input.

updateF7Slider changes the value of a slider input on the client.

Usage

```
f7Slider(  
  inputId,  
  label,  
  min,  
  max,  
  value,  
  step = 1,  
  scale = FALSE,  
  scaleSteps = 5,  
  scaleSubSteps = 0,  
  vertical = FALSE,  
  verticalReversed = FALSE,  
  labels = NULL,  
  color = NULL,  
  noSwipping = TRUE,  
  showLabel = TRUE,  
  ...,  
  style = list(inset = FALSE, outline = FALSE, strong = FALSE)  
)  
  
updateF7Slider(  
  inputId,  
  min = NULL,  
  max = NULL,  
  value = NULL,  
  scale = FALSE,  
  scaleSteps = NULL,  
  scaleSubSteps = NULL,  
  step = NULL,  
  color = NULL,  
  ...,  
  session = shiny::getDefaultReactiveDomain()  
)
```

Arguments

| | |
|------------------|--|
| inputId | Slider input id. |
| label | Slider label. |
| min | Slider minimum range. |
| max | Slider maximum range. |
| value | Slider value or a vector containing 2 values (for a range). |
| step | Slider increase step size. |
| scale | Slider scale. |
| scaleSteps | Number of scale steps. |
| scaleSubSteps | Number of scale sub steps (each step will be divided by this value). |
| vertical | Whether to apply a vertical display. FALSE by default. |
| verticalReversed | Makes vertical range slider reversed (vertical must be also enabled). FALSE by default. |
| labels | Enables additional label around range slider knob. List of 2 f7Icon expected. |
| color | See getF7Colors for valid colors. |
| noSwipping | Prevent swiping when slider is manipulated in an f7TabLayout . |
| showLabel | Allow bubble containing the slider value. Default to TRUE. |
| ... | Other options to pass to the widget. See https://framework7.io/docs/range-slider#range-slider-parameters . |
| style | Allows to style the input. inset, outline and strong are available. |
| session | The Shiny session object. |

Note

labels option only works when vertical is FALSE!

Important: you cannot transform a range slider into a simple slider and inversely.

Examples

```
library(shiny)
library(shinyMobile)

app <- shinyApp(
  ui = f7Page(
    title = "My app",
    f7SingleLayout(
      navbar = f7Navbar(title = "updateF7Slider"),
      f7Block(
        f7BlockTitle("Simple slider with custom style", size = "large"),
        f7Button(inputId = "update_slider", label = "Update slider"),
        f7Slider(
          inputId = "slider",
          label = "Number of observations",
```

```

      max = 1000,
      min = 0,
      value = 100,
      scaleSteps = 5,
      scaleSubSteps = 3,
      scale = TRUE,
      color = "orange",
      labels = tagList(
        f7Icon("circle"),
        f7Icon("circle_fill")
      ),
      style = list(inset = TRUE, strong = TRUE, outline = TRUE)
    ),
    textOutput("slider_res")
  ),
  f7Block(
    f7BlockTitle("Range slider", size = "large"),
    f7Button(inputId = "update_range", label = "Update slider"),
    f7Slider(
      inputId = "range",
      label = "Range values",
      max = 500,
      min = 0,
      step = 0.01,
      color = "deeppurple",
      value = c(50, 100)
    ),
    textOutput("range_res")
  )
),
server = function(input, output, session) {
  output$slider_res <- renderText({
    input$slider
  })

  observeEvent(input$update_slider, {
    updateF7Slider(
      inputId = "slider",
      value = 0.05,
      min = 0,
      max = 0.01,
      scale = FALSE,
      step = 0.001,
      color = "pink"
    )
  })

  output$range_res <- renderText({
    input$range
  })

  observeEvent(input$update_range, {

```

```

    updateF7Slider(
      inputId = "range",
      value = c(1, 5),
      min = 0,
      scale = TRUE,
      step = 0.01,
      max = 10,
      color = "teal"
    )
  })
}
)

if (interactive() || identical(Sys.getenv("TESTTHAT"), "true")) app

```

f7SmartSelect

Framework7 smart select

Description

f7SmartSelect is smarter than the classic [f7Select](#), allows for choices filtering, ...
 updateF7SmartSelect changes the value of a smart select input on the client.

Usage

```

f7SmartSelect(
  inputId,
  label,
  choices,
  selected = NULL,
  openIn = c("page", "sheet", "popup", "popover"),
  searchbar = TRUE,
  multiple = FALSE,
  maxLength = NULL,
  virtualList = FALSE,
  ...
)

updateF7SmartSelect(
  inputId,
  selected = NULL,
  choices = NULL,
  multiple = NULL,
  maxLength = NULL,
  ...,
  session = shiny::getDefaultReactiveDomain()
)

```

Arguments

| | |
|-------------|--|
| inputId | Select input id. |
| label | Select input label. |
| choices | Select input choices. |
| selected | Default selected item. If NULL, the first item is selected. |
| openIn | Smart select type: either c("sheet", "popup", "popover"). Note that the search bar is only available when the type is popup. |
| searchbar | Whether to enable the search bar. TRUE by default. |
| multiple | Whether to allow multiple values. FALSE by default. |
| maxLength | Maximum items to select when multiple is TRUE. |
| virtualList | Enable Virtual List for smart select if your select has a lot of options. Default to FALSE. |
| ... | Other options. See https://framework7.io/docs/smart-select#smart-select-parameters . |
| session | The Shiny session object, usually the default value will suffice. |

Examples

```

library(shiny)
library(shinyMobile)

app <- shinyApp(
  ui = f7Page(
    title = "Update f7SmartSelect",
    f7SingleLayout(
      navbar = f7Navbar(title = "Update f7SmartSelect"),
      f7Block(f7Button("update", "Update Smart Select")),
      f7List(
        inset = TRUE,
        strong = TRUE,
        outline = TRUE,
        f7SmartSelect(
          inputId = "smartselect",
          label = "Choose a variable:",
          choices = split(colnames(mtcars[-1]), rep(1:5)),
          openIn = "popup"
        )
      ),
      tableOutput("data")
    ),
  ),
  server = function(input, output, session) {
    output$data <- renderTable(
      mtcars[, c("mpg", input$smartselect), drop = FALSE],
      rownames = TRUE
    )

    observeEvent(input$update, {
      updateF7SmartSelect(

```

```

        inputId = "smartselect",
        openIn = "sheet",
        selected = "hp",
        choices = c("hp", "gear", "carb"),
        multiple = TRUE,
        maxLength = 2
    )
})
}
)

if (interactive() || identical(Sys.getenv("TESTTHAT"), "true")) app

```

f7SplitLayout

Framework7 split layout

Description

This is a modified version of the [f7SingleLayout](#). It is intended to be used with tablets.

Usage

```
f7SplitLayout(..., navbar, sidebar, toolbar = NULL, panel = NULL)
```

Arguments

| | |
|---------|--|
| ... | Content. |
| navbar | Slot for f7Navbar . We expect the following: <code>f7Navbar(title = "Navbar", leftPanel = TRUE)</code> |
| sidebar | Slot for f7Panel . Particularly we expect the following: <code>f7Panel(title = "Sidebar", side = "left", theme = "light", "Blabla", effect = "reveal")</code> . At a minimal app width (1024 px) the sidebar becomes always visible. You can override this behavior by setting <code>options = list(visibleBreakpoint = 1024)</code> to the desired width in f7Panel . |
| toolbar | Slot for f7Toolbar . |
| panel | Slot for f7Panel . Expect only a right panel, for instance: <code>f7Panel(title = "Right Panel", side = "right", theme = "light", "Blabla", effect = "cover")</code> |

Author(s)

David Granjon, <dgranjon@gmail.com>

Examples

```
library(shiny)
library(ggplot2)
library(shinyMobile)
library(apexcharter)
library(thematic)

fruits <- data.frame(
  name = c("Apples", "Oranges", "Bananas", "Berries"),
  value = c(44, 55, 67, 83)
)

thematic_shiny(font = "auto")

new_mtcars <- reshape(
  data = head(mtcars),
  idvar = "model",
  varying = list(c("drat", "wt")),
  times = c("drat", "wt"),
  direction = "long",
  v.names = "value",
  drop = c("mpg", "cyl", "hp", "dist", "qsec", "vs", "am", "gear", "carb")
)

app <- shinyApp(
  ui = f7Page(
    title = "Split layout",
    options = list(
      dark = FALSE
    ),
    f7SplitLayout(
      sidebar = f7Panel(
        title = "Sidebar",
        side = "left",
        effect = "push",
        options = list(
          visibleBreakpoint = 1024
        ),
      ),
      f7PanelMenu(
        id = "menu",
        strong = TRUE,
        f7PanelItem(
          tabName = "tab1",
          title = "Tab 1",
          icon = f7Icon("equal_circle"),
          active = TRUE
        ),
        f7PanelItem(
          tabName = "tab2",
          title = "Tab 2",
          icon = f7Icon("equal_circle")
        ),
      ),
    ),
  server = function(input, output, session) {
  }
)
```

```

    f7PanelItem(
      tabName = "tab3",
      title = "Tab 3",
      icon = f7Icon("equal_circle")
    )
  ),
  uiOutput("selected_tab")
),
panel = f7Panel(
  side = "right",
  effect = "floating",
  "Blablabla"
),
navbar = f7Navbar(
  title = "Split Layout",
  hairline = FALSE,
  leftPanel = TRUE,
  rightPanel = TRUE
),
toolbar = f7Toolbar(
  position = "bottom",
  f7Link(label = "Link 1", href = "https://www.google.com"),
  f7Link(label = "Link 2", href = "https://www.google.com")
),
# main content
f7Items(
  f7Item(
    tabName = "tab1",
    f7Button("toggleSheet", "Plot parameters"),
    f7Sheet(
      id = "sheet1",
      label = "Plot Parameters",
      orientation = "bottom",
      swipeToClose = TRUE,
      backdrop = TRUE,
      f7Slider(
        "obs",
        "Number of observations:",
        min = 0, max = 1000,
        value = 500
      )
    )
  ),
  br(),
  plotOutput("distPlot")
),
f7Item(
  tabName = "tab2",
  apexchartOutput("radar")
),
f7Item(
  tabName = "tab3",
  f7Toggle(
    inputId = "plot_show",

```



```

        label = "Show Plot?",
        checked = TRUE
      ),
      apexchartOutput("multi_radial")
    )
  )
),
server = function(input, output, session) {
  observeEvent(input$toggleSheet, {
    updateF7Sheet(id = "sheet1")
  })

  observeEvent(input$obs, {
    if (input$obs < 500) {
      f7Notif(
        text = paste0(
          "The slider value is only ", input$obs, ". Please
          increase it"
        ),
        icon = f7Icon("bolt_fill"),
        title = "Alert",
        titleRightText = Sys.Date()
      )
    }
  })

  output$radar <- renderApexchart({
    apex(
      data = new_mtcars,
      type = "radar",
      mapping = aes(
        x = model,
        y = value,
        group = time
      )
    )
  })

  output$selected_tab <- renderUI({
    HTML(paste0("Currently selected tab: ", strong(input$menu)))
  })

  output$distPlot <- renderPlot({
    dist <- rnorm(input$obs)
    hist(dist)
  })

  output$multi_radial <- renderApexchart({
    if (input$plot_show) {
      apex(data = fruits, type = "radialBar", mapping = aes(x = name, y = value))
    }
  })
}

```

```
    })
  }
)

if (interactive() || identical(Sys.getenv("TESTTHAT"), "true")) app
```

f7Stepper

Framework7 stepper input

Description

f7Stepper creates a stepper input.

updateF7Stepper changes the value of a stepper input on the client.

Usage

```
f7Stepper(  
  inputId,  
  label,  
  min,  
  max,  
  value,  
  step = 1,  
  fill = FALSE,  
  rounded = FALSE,  
  raised = FALSE,  
  size = NULL,  
  color = NULL,  
  wraps = FALSE,  
  autorepeat = TRUE,  
  manual = FALSE,  
  decimalPoint = 4,  
  buttonsEndInputMode = TRUE  
)
```

```
updateF7Stepper(  
  inputId,  
  min = NULL,  
  max = NULL,  
  value = NULL,  
  step = NULL,  
  fill = NULL,  
  rounded = NULL,  
  raised = NULL,  
  size = NULL,  
  color = NULL,  
  wraps = NULL,
```

```

    decimalPoint = NULL,
    autorepeat = NULL,
    manual = NULL,
    session = shiny::getDefaultReactiveDomain()
  )

```

Arguments

| | |
|---------------------|--|
| inputId | Stepper input id. |
| label | Stepper label. |
| min | Stepper minimum value. |
| max | Stepper maximum value. |
| value | Stepper value. Must belong to <code>\[min, max\]</code> . |
| step | Increment step. 1 by default. |
| fill | Whether to fill the stepper. FALSE by default. |
| rounded | Whether to round the stepper. FALSE by default. |
| raised | Whether to put a relief around the stepper. FALSE by default. |
| size | Stepper size: "small", "large" or NULL. |
| color | Stepper color: NULL or "red", "green", "blue", "pink", "yellow", "orange", "grey" and "black". |
| wraps | In wraps mode incrementing beyond maximum value sets value to minimum value, likewise, decrementing below minimum value sets value to maximum value. FALSE by default. |
| autorepeat | Pressing and holding one of its buttons increments or decrements the stepper's value repeatedly. With dynamic autorepeat, the rate of change depends on how long the user continues pressing the control. TRUE by default. |
| manual | It is possible to enter value manually from keyboard or mobile keypad. When click on input field, stepper enter into manual input mode, which allow type value from keyboar and check fractional part with defined accuracy. Click outside or enter Return key, ending manual mode. TRUE by default. |
| decimalPoint | Number of digits after dot, when in manual input mode. |
| buttonsEndInputMode | Disables manual input mode on Stepper's minus or plus button click. |
| session | The Shiny session object, usually the default value will suffice. |

Note

While updating, the autorepeat field does not work correctly.

Examples

```

library(shiny)
library(shinyMobile)

app <- shinyApp(

```

```

ui = f7Page(
  title = "Stepper app",
  f7SingleLayout(
    navbar = f7Navbar(title = "updateF7Stepper"),
    f7Block(f7Button(inputId = "update", label = "Update stepper")),
    f7List(
      strong = TRUE,
      inset = TRUE,
      outline = TRUE,
      f7Stepper(
        inputId = "stepper",
        label = "My stepper",
        min = 0,
        max = 10,
        size = "small",
        value = 4,
        wraps = TRUE,
        autorepeat = TRUE,
        rounded = FALSE,
        raised = FALSE,
        manual = FALSE
      )
    ),
    verbatimTextOutput("test")
  )
),
server = function(input, output, session) {
  output$test <- renderPrint(input$stepper)

  observeEvent(input$update, {
    updateF7Stepper(
      inputId = "stepper",
      value = 0.1,
      step = 0.01,
      size = "large",
      min = 0,
      max = 1,
      wraps = FALSE,
      autorepeat = FALSE,
      rounded = TRUE,
      raised = TRUE,
      color = "pink",
      manual = TRUE,
      decimalPoint = 2
    )
  })
}
)

if (interactive() || identical(Sys.getenv("TESTTHAT"), "true")) app

```

Description

f7SubNavbar creates a nested navbar component for [f7Navbar](#).

Usage

```
f7SubNavbar(...)
```

Arguments

... Any elements.

Examples

```
library(shiny)
library(shinyMobile)

app <- shinyApp(
  ui = f7Page(
    title = "Sub Navbar",
    options = list(
      dark = FALSE,
      navbar = list(
        hideOnPageScroll = TRUE,
        mdCenterTitle = TRUE
      )
    ),
    f7SingleLayout(
      panels = tagList(
        f7Panel(
          title = "Left Panel",
          side = "left",
          f7Block("Blabla"),
          effect = "cover"
        ),
        f7Panel(
          title = "Right Panel",
          side = "right",
          f7Block("Blabla"),
          effect = "cover"
        )
      ),
      navbar = f7Navbar(
        subNavbar = f7SubNavbar(
          f7Button(label = "My button"),
          f7Button(label = "My button"),
          f7Button(label = "My button")
        )
      )
    )
  )
)
```

```

    ),
    title = "Title",
    leftPanel = TRUE,
    rightPanel = TRUE
  ),
  f7Block(f7Button(inputId = "toggle", "Toggle navbar")),
  f7Block(
    lapply(1:20, f7Card)
  )
),
server = function(input, output, session) {
  observeEvent(input$toggle, {
    updateF7Navbar()
  })
}
)

if (interactive() || identical(Sys.getenv("TESTTHAT"), "true")) app

```

f7Swipeout

Framework7 swipeout element

Description

f7Swipeout is designed to be used in combination with [f7ListItem](#).

f7SwipeoutItem is inserted in [f7Swipeout](#).

Usage

```
f7Swipeout(tag, ..., left = NULL, right = NULL, side = deprecated())
```

```
f7SwipeoutItem(id, label, color = NULL)
```

Arguments

| | |
|-------|---|
| tag | Tag to be swiped. |
| ... | [Deprecated] . |
| left | When side is "both", put the left f7SwipeoutItem . |
| right | When side is "both", put the right f7SwipeoutItem . |
| side | [Deprecated] . |
| id | Item unique id. |
| label | Item label. |
| color | Item color. |

Examples

```

if (interactive()) {
  library(shiny)
  library(shinyMobile)

  media_item <- function(j) {
    f7ListItem(
      title = letters[j],
      subtitle = "subtitle",
      "Lorem ipsum dolor sit amet, consectetur adipiscing elit.
        Nulla sagittis tellus ut turpis condimentum, ut dignissim
        lacus tincidunt.",
      media = tags$img(
        src = paste0(
          "https://cdn.framework7.io/placeholder/people-160x160-", j, ".jpg"
        )
      ),
      right = "Right Text"
    )
  }
}
shinyApp(
  ui = f7Page(
    title = "Swipeout",
    f7SingleLayout(
      navbar = f7Navbar(title = "Swipeout"),
      # simple list
      f7List(
        mode = "media",
        strong = TRUE,
        outline = TRUE,
        inset = TRUE,
        lapply(1:3, function(j) {
          if (j == 1) {
            f7Swipeout(
              tag = media_item(j),
              left = tagList(
                f7SwipeoutItem(id = "alert", "Alert"),
                f7SwipeoutItem(id = "notification", color = "green", "Notif")
              ),
              right = f7SwipeoutItem(id = "toast", "Click me!")
            )
          } else {
            media_item(j)
          }
        })
      )
    )
  ),
  server = function(input, output, session) {
    observe({
      print(input$alert)
      print(input$notification)
    })
  }
}

```

```

    })

    observeEvent(input$notification, {
      f7Notif(
        text = "test",
        icon = f7Icon("bolt_fill"),
        title = "Notification",
        subtitle = "A subtitle",
        titleRightText = "now"
      )
    })

    observeEvent(input$alert, {
      f7Dialog(
        title = "Dialog title",
        text = "This is an alert dialog"
      )
    })

    observeEvent(input$toast, {
      f7Toast("This is a toast.")
    })
  }
}

```

f7Swiper

Framework7 swiper

Description

f7Swiper creates a Framework7 swiper container (like carousel).

f7Slide is an [f7Swiper](#) element.

Usage

```

f7Swiper(
  ...,
  id,
  options = list(speed = 400, loop = FALSE, spaceBetween = 50, slidesPerView = "auto",
    centeredSlides = TRUE, navigation = list(nextEl = ".swiper-button-next", prevEl =
    ".swiper-button-prev"), pagination = list(el = ".swiper-pagination", clickable =
    TRUE), scrollbar = list(el = ".swiper-scrollbar", draggable = TRUE))
)

f7Slide(...)

```


Arguments

| | |
|---------|--|
| ... | Slide content. Any element. |
| id | Swiper unique id. |
| options | Other options. Expect a list. See https://swiperjs.com/swiper-api for all available options. |

Author(s)

David Granjon, <dgranjon@gmail.com>

Examples

```
library(shiny)
library(shinyMobile)

app <- shiny::shinyApp(
  ui = f7Page(
    title = "Swiper",
    f7SingleLayout(
      navbar = f7Navbar(title = "f7Swiper"),
      f7Swiper(
        id = "swiper",
        f7Slide(
          f7Card(
            f7Toggle(
              inputId = "toggle",
              label = "My toggle",
              color = "pink",
              checked = TRUE
            ),
            verbatimTextOutput("test")
          )
        ),
        f7Slide(
          f7Card(
            f7Slider(
              inputId = "slider",
              label = "Number of observations",
              max = 1000,
              min = 0,
              value = 100,
              scaleSteps = 5,
              scaleSubSteps = 3,
              scale = TRUE,
              color = "orange",
              labels = tagList(
                f7Icon("circle"),
                f7Icon("circle_fill")
              )
            )
          ),
          textOutput("test2")
        )
      )
    )
  )
```

```

    ),
  )
)
),
server = function(input, output) {
  output$test <- renderPrint(input$toggle)
  output$test2 <- renderText(input$slider)
}
)

if (interactive() || identical(Sys.getenv("TESTTHAT"), "true")) app

```

f7Tab

Create a Framework7 tab item

Description

Build a Framework7 tab item

Usage

```
f7Tab(..., title = NULL, tabName, icon = NULL, active = FALSE, hidden = FALSE)
```

Arguments

| | |
|---------|---|
| ... | Item content. |
| title | Tab title (name). |
| tabName | Item id. Must be unique, without space nor punctuation symbols. |
| icon | Item icon. Expect f7Icon function with the suitable lib argument (either md or ios or NULL for native f7 icons). |
| active | Whether the tab is active at start. Do not select multiple tabs, only the first one will be set to active. |
| hidden | Whether to hide the tab. This is useful when you want to add invisible tabs (that do not appear in the tabbar) but you can still navigate with updateF7Tabs . |

Author(s)

David Granjon, <dgranjon@ymail.com>

`f7TabLayout`*Framework7 tab layout*

Description

`f7TabLayout` create a single page app with multiple tabs, giving the illusion of a multi pages experience.

Usage

```
f7TabLayout(..., navbar, messagebar = NULL, panels = NULL)
```

Arguments

| | |
|-------------------------|---|
| <code>...</code> | Slot for f7Tabs . |
| <code>navbar</code> | Slot for f7Navbar . |
| <code>messagebar</code> | Slot for f7MessageBar . |
| <code>panels</code> | Slot for f7Panel . Wrap in <code>tagList</code> if multiple panels. |

Author(s)

David Granjon, <dgranjon@ymail.com>

Examples

```
library(shiny)
library(shinyMobile)
library(apexcharter)
library(shinyWidgets)

poll <- data.frame(
  answer = c("Yes", "No"),
  n = c(254, 238)
)

app <- shinyApp(
  ui = f7Page(
    title = "Tabs layout",
    f7TabLayout(
      panels = tagList(
        f7Panel(title = "Left Panel", side = "left", "Blabla", effect = "cover"),
        f7Panel(title = "Right Panel", side = "right", "Blabla", effect = "cover")
      ),
      navbar = f7Navbar(
        title = "Tabs",
        leftPanel = TRUE,
        rightPanel = TRUE
      )
    ),
```

```

f7Tabs(
  animated = TRUE,
  # swipeable = TRUE,
  f7Tab(
    title = "Tab 1",
    tabName = "Tab1",
    icon = f7Icon("folder"),
    active = TRUE,
    f7List(
      strong = TRUE,
      prettyRadioButtons(
        inputId = "theme",
        label = "Select a theme:",
        thick = TRUE,
        inline = TRUE,
        selected = "md",
        choices = c("ios", "md"),
        animation = "pulse",
        status = "info"
      ),
      prettyRadioButtons(
        inputId = "dark",
        label = "Select a color:",
        thick = TRUE,
        inline = TRUE,
        selected = "dark",
        choices = c("light", "dark"),
        animation = "pulse",
        status = "info"
      )
    ),
    f7Card(
      title = "Card header",
      apexchartOutput("pie")
    )
  ),
  f7Tab(
    title = "Tab 2",
    tabName = "Tab2",
    icon = f7Icon("keyboard"),
    f7Card(
      title = "Card header",
      apexchartOutput("scatter")
    )
  ),
  f7Tab(
    title = "Tab 3",
    tabName = "Tab3",
    icon = f7Icon("layers_alt"),
    f7Card(
      title = "Card header",
      f7SmartSelect(
        "variable",

```

```

        "Variables to show:",
        c(
          "Cylinders" = "cyl",
          "Transmission" = "am",
          "Gears" = "gear"
        ),
        openIn = "sheet",
        multiple = TRUE
      ),
      tableOutput("data")
    )
  )
),
server = function(input, output, session) {
  # river plot
  dates <- reactive(seq.Date(Sys.Date() - 30, Sys.Date(), by = input$by))

  output$pie <- renderApexchart({
    apex(
      data = poll,
      type = "pie",
      mapping = aes(x = answer, y = n)
    )
  })

  output$scatter <- renderApexchart({
    apex(
      data = mtcars,
      type = "scatter",
      mapping = aes(
        x = wt,
        y = mpg,
        fill = cyl
      )
    )
  })

  # datatable
  output$data <- renderTable(
    {
      mtcars[, c("mpg", input$variable), drop = FALSE]
    },
    rownames = TRUE
  )

  # theme changes
  observe({
    updateF7App(
      options =list(

```

```

        theme = input$theme,
        dark = ifelse(input$dark == "dark", TRUE, FALSE)
      )
    )
  })

}
)

if (interactive() || identical(Sys.getenv("TESTTHAT"), "true")) app

```

f7Table

Framework7 table

Description

Creates a table container.

Usage

```
f7Table(data, colnames = NULL, card = FALSE)
```

Arguments

| | |
|----------|--|
| data | A data.frame. |
| colnames | Column names to use, if NULL uses data column names. |
| card | Whether to use as card. |

Examples

```

library(shiny)
library(shinyMobile)

app <- shinyApp(
  ui = f7Page(
    title = "My app",
    f7SingleLayout(
      navbar = f7Navbar(
        title = "f7Table"
      ),
      uiOutput("table")
    )
  ),
  server = function(input, output) {
    output$table <- renderUI({
      f7Table(mtcars)
    })
  }
)

if (interactive() || identical(Sys.getenv("TESTTHAT"), "true")) app

```

| | |
|-----------|--|
| f7TabLink | <i>Special button/link to insert in the tabbar</i> |
|-----------|--|

Description

Use in the .items slot of [f7Tabs](#).

Usage

```
f7TabLink(..., icon = NULL, label = NULL)
```

Arguments

| | |
|-------|--|
| ... | Any attribute like `data-sheet`, id, ... |
| icon | Expect f7Icon . |
| label | Button label. |

| | |
|--------|---------------------------------|
| f7Tabs | <i>Create a Framework7 tabs</i> |
|--------|---------------------------------|

Description

By default, [f7Tabs](#) are used within the [f7TabLayout](#). However, you may use them as standalone components if you specify a the segmented or strong styles.

Usage

```
f7Tabs(
  ...,
  .items = NULL,
  id = NULL,
  swipeable = FALSE,
  animated = TRUE,
  style = c("toolbar", "segmented", "strong")
)
```

Arguments

| | |
|-----------|--|
| ... | Slot for f7Tab . |
| .items | Slot for other items that could be part of the toolbar such as buttons or f7TabLink . This may be useful to open an f7Sheet from the tabbar. |
| id | Optional to get the id of the currently selected f7Tab . |
| swipeable | Whether to allow finger swipe. FALSE by default. Only for touch-screens. Not compatible with animated. |

| | |
|----------|---|
| animated | Whether to show transition between tabs. TRUE by default. Not compatible with swipeable. |
| style | Tabs style: c("toolbar", "segmented", "strong"). If style is toolbar, then f7Tab have a toolbar behavior. |

Details

For md design, when there is no icons in the tabbar, a tiny horizontal highlight bar is displayed on top of the active tab. Whenever a tab with icon is included, the highlight bar is hidden and a round pill highlights the currently active tab.

Author(s)

David Granjon, <dgranjon@ymail.com>

Examples

```
library(shiny)
library(shinyMobile)

app <- shinyApp(
  ui = f7Page(
    title = "Tabs",
    options = list(dark = FALSE, theme = "ios"),
    f7TabLayout(
      navbar = f7Navbar(
        title = HTML(paste("Currently selected:", textOutput("selected"))),
        subNavbar = f7SubNavbar(
          f7Button("update", "Update", fill = FALSE, outline = TRUE),
          f7Button("remove", "Remove", fill = FALSE, outline = TRUE),
          f7Button("insert", "Insert", fill = FALSE, outline = TRUE)
        )
      ),
      f7Tabs(
        id = "tabs",
        swipeable = TRUE,
        animated = FALSE,
        f7Tab(
          title = "Tab 1",
          tabName = "Tab1",
          icon = f7Icon("house_alt_fill"),
          f7Block("Tab 1 content"),
          f7Sheet(
            id = "sheet",
            label = "More",
            orientation = "bottom",
            swipeToClose = TRUE,
            backdrop = TRUE,
            "Lorem ipsum dolor sit amet, consectetur adipiscing elit.
            Quisque ac diam ac quam euismod porta vel a nunc. Quisque sodales
            scelerisque est, at porta justo cursus ac"
          )
        )
      )
    )
  )
)
```



```

    ),
    f7Tab(
      title = "Tab 2",
      tabName = "Tab2",
      icon = f7Icon("location_circle_fill"),
      f7Block("tab 2 text"),
      active = TRUE
    ),
    f7Tab(
      title = "Tab 3",
      tabName = "Tab3",
      icon = f7Icon("pencil_circle_fill"),
      f7Block("tab 3 text"),
    ),
    .items = f7TabLink(
      icon = f7Icon("bolt_fill"),
      label = "Toggle Sheet",
      `data-sheet` = "#sheet",
      class = "sheet-open"
    )
  )
)
),
server = function(input, output, session) {
  output$selected <- renderText(input$tabs)

  tabs <- reactiveVal(paste0("Tab", 1:3))

  # Update
  observeEvent(input$update, {
    req(length(tabs()) > 0)
    tab_id <- min(tabs())
    updateF7Tabs(
      id = "tabs",
      selected = tab_id
    )
    message(sprintf("Selecting %s", tab_id))
  })

  # Remove max tab
  observeEvent(input$remove, {
    req(length(tabs()) > 0)
    tab_id <- max(tabs())
    removeF7Tab(
      id = "tabs",
      target = tab_id
    )
    message(sprintf("Removing %s", tab_id))
    tabs(tabs()[~which(tabs() == tab_id)])
  })

  # Add
  observeEvent(input$insert, {

```

```

tab_id <- if (length(tabs()) > 0) max(tabs())
new_tab_id <- if (length(tabs()) > 0) {
  as.numeric(strsplit(max(tabs()), "Tab")[[1]][2]) + 1
} else {
  1
}

insertF7Tab(
  id = "tabs",
  position = if (length(tabs()) > 0) "after",
  target = if (length(tabs()) > 0) tab_id,
  tab = f7Tab(
    # Use multiple elements to test for accessor function
    f7Block(sprintf("New tab %s content", new_tab_id)),
    tabName = sprintf("Tab%s", new_tab_id),
    icon = f7Icon("app_badge")
  ),
  select = TRUE
)

message(sprintf("Adding tab %s", new_tab_id))
tabs(c(tabs(), sprintf("Tab%s", new_tab_id)))
})
}
)

if (interactive() || identical(Sys.getenv("TESTTHAT"), "true")) app

```

f7TapHold

Framework7 tapHold module

Description

Framework7 has a so called "tap hold" event. If tapHold is enabled in [f7Page](#), it triggers after a sustained, complete touch event. [f7TapHold](#) is triggered from the server.

Usage

```
f7TapHold(target, callback, session = shiny::getDefaultReactiveDomain())
```

Arguments

| | |
|----------|---|
| target | Element to apply the tapHold event on. Must be a jQuery selector, such as "#id" or ".class", ".class1, .class2", "a"... |
| callback | Javascript callback. |
| session | Shiny session object. |

Examples

```

library(shiny)
library(shinyMobile)

app <- shinyApp(
  ui = f7Page(
    title = "Taphold",
    f7SingleLayout(
      navbar = f7Navbar(title = "f7TapHold"),
      f7Button(inputId = "pressme", label = "Press me")
    )
  ),
  server = function(input, output, session) {
    observe({
      f7TapHold(
        target = "#pressme",
        callback = "app.dialog.alert('Tap hold fired!')"
      )
    })
  }
)

if (interactive() || identical(Sys.getenv("TESTTHAT"), "true")) app

```

f7Text

Framework7 text input

Description

f7Text creates a text input container.

updateF7Text changes the value of a text input on the client.

f7TextArea creates a f7 text area input.

updateF7TextArea changes the value of a text area input on the client.

[f7Password](#) creates a password input.

Usage

```

f7Text(
  inputId,
  label = NULL,
  value = "",
  placeholder = NULL,
  style = list(media = NULL, description = NULL, floating = FALSE, outline = FALSE,
    clearable = TRUE)
)

updateF7Text(

```

```

    inputId,
    label = NULL,
    value = NULL,
    placeholder = NULL,
    session = shiny::getDefaultReactiveDomain()
  )

f7TextArea(
  inputId,
  label,
  value = "",
  placeholder = NULL,
  resize = FALSE,
  style = list(media = NULL, description = NULL, floating = FALSE, outline = FALSE,
    clearable = TRUE)
)

updateF7TextArea(
  inputId,
  label = NULL,
  value = NULL,
  placeholder = NULL,
  session = shiny::getDefaultReactiveDomain()
)

f7Password(
  inputId,
  label,
  placeholder = NULL,
  style = list(media = NULL, description = NULL, floating = FALSE, outline = FALSE,
    clearable = TRUE)
)

```

Arguments

| | |
|--------------------------|---|
| <code>inputId</code> | Text input id. |
| <code>label</code> | Text input label. |
| <code>value</code> | Text input value. |
| <code>placeholder</code> | Text input placeholder. |
| <code>style</code> | Input style. A list with <code>media</code> (image or icon), <code>description</code> (text), <code>floating</code> , <code>outline</code> and <code>clearable</code> (booleans). |
| <code>session</code> | The Shiny session object, usually the default value will suffice. |
| <code>resize</code> | Whether to box can be resized. Default to <code>FALSE</code> . |

Note

Updating label does not work yet.

Examples

```
library(shiny)
library(shinyMobile)

app <- shinyApp(
  ui = f7Page(
    f7SingleLayout(
      navbar = f7Navbar(title = "Text inputs"),
      f7Block(f7Button("update", "Click me")),
      f7BlockTitle("A list of inputs"),
      f7List(
        inset = TRUE,
        dividers = FALSE,
        strong = TRUE,
        f7Text(
          inputId = "text",
          label = "Text input",
          value = "Some text",
          placeholder = "Your text here",
          style = list(
            description = "A cool text input",
            outline = TRUE,
            media = f7Icon("house"),
            clearable = TRUE,
            floating = TRUE
          )
        ),
        f7TextArea(
          inputId = "textarea",
          label = "Text Area",
          value = "Lorem ipsum dolor sit amet, consectetur
            adipiscing elit, sed do eiusmod tempor incididunt ut
            labore et dolore magna aliqua",
          placeholder = "Your text here",
          resize = TRUE,
          style = list(
            description = "A cool text input",
            outline = TRUE,
            media = f7Icon("house"),
            clearable = TRUE,
            floating = TRUE
          )
        ),
        f7Password(
          inputId = "password",
          label = "Password:",
          placeholder = "Your password here",
          style = list(
            description = "A cool text input",
            outline = TRUE,
            media = f7Icon("house"),
            clearable = TRUE,

```

```

        floating = TRUE
      )
    )
  ),
  f7Grid(
    cols = 3,
    f7Block(
      f7BlockTitle("Text value"),
      textOutput("text_value")
    ),
    f7Block(
      f7BlockTitle("Text area value"),
      textOutput("textarea_value")
    ),
    f7Block(
      f7BlockTitle("Password value"),
      textOutput("password_value")
    )
  )
),
server = function(input, output, session) {
  output$text_value <- renderText(input$text)
  output$textarea_value <- renderText(input$textarea)
  output$password_value <- renderText(input$password)

  observeEvent(input$update, {
    updateF7Text(
      inputId = "text",
      value = "Updated Text"
    )
    updateTextAreaInput(
      inputId = "textarea",
      value = "",
      placeholder = "New placeholder"
    )
  })
}
)

if (interactive() || identical(Sys.getenv("TESTTHAT"), "true")) app

```

f7Timeline

Framework7 timeline

Description

f7Timeline is a static timeline container.

f7TimelineItem goes inside [f7Timeline](#).

Usage

```
f7Timeline(  
  ...,  
  sides = FALSE,  
  horizontal = FALSE,  
  calendar = FALSE,  
  year = NULL,  
  month = NULL  
)  
  
f7TimelineItem(  
  ...,  
  date = NULL,  
  card = FALSE,  
  time = NULL,  
  title = NULL,  
  subtitle = NULL,  
  side = NULL  
)
```

Arguments

| | |
|------------|--|
| ... | Item content, text for instance. |
| sides | Enable side-by-side timeline mode. |
| horizontal | Whether to use the horizontal layout. Not compatible with sides. |
| calendar | Special type of horizontal layout with current year and month. |
| year | Current year, only if calendar is TRUE. |
| month | Current month, only if calendar is TRUE. |
| date | Timeline item date. Required. |
| card | Whether to wrap the content in a card. FALSE by default. |
| time | Timeline item time. Optional. |
| title | Timeline item title. Optional. |
| subtitle | Timeline item subtitle. Optional. |
| side | Force element to required side: "right" or "left". Only if sides os TRUE in f7Timeline |

Author(s)

David Granjon <dgranjon@ymail.com>

Examples

```
library(shiny)  
library(shinyMobile)
```

```

items <- tagList(
  lapply(1:5,
    function(i) {
      f7TimelineItem(
        paste0("Another text ", i),
        date = paste0(i, " Dec"),
        card = i %% 2 == 0,
        time = paste0(10 + i, ":30"),
        title = paste0("Title", i),
        subtitle = paste0("Subtitle", i),
        side = ifelse(i %% 2 == 0, "left", "right")
      )
    }
  )
)

app <- shinyApp(
  ui = f7Page(
    title = "Timelines",
    f7SingleLayout(
      navbar = f7Navbar(title = "Timelines"),
      f7BlockTitle(title = "Horizontal timeline", size = "large") %>%
        f7Align(side = "center"),
      f7Timeline(
        sides = FALSE,
        horizontal = TRUE,
        items
      ),
      f7BlockTitle(title = "Vertical side by side timeline", size = "large") %>%
        f7Align(side = "center"),
      f7Timeline(
        sides = TRUE,
        items
      ),
      f7BlockTitle(title = "Vertical timeline", size = "large") %>%
        f7Align(side = "center"),
      f7Timeline(items),
      f7BlockTitle(title = "Calendar timeline", size = "large") %>%
        f7Align(side = "center"),
      f7Timeline(items, calendar = TRUE, year = "2019", month = "December")
    )
  ),
  server = function(input, output) {}
)

if (interactive() || identical(Sys.getenv("TESTTHAT"), "true")) app

```


Description

f7Toast creates a small toast notification from the server side.

Usage

```
f7Toast(  
  text,  
  position = c("bottom", "top", "center"),  
  closeButton = TRUE,  
  closeButtonText = "close",  
  closeButtonColor = "red",  
  closeTimeout = 3000,  
  icon = NULL,  
  ...,  
  session = shiny::getDefaultReactiveDomain()  
)
```

Arguments

| | |
|------------------|--|
| text | Toast content. |
| position | Toast position c("bottom", "top", "center"). |
| closeButton | Whether to close the toast with a button. TRUE by default. |
| closeButtonText | Close button text. |
| closeButtonColor | Close button color. |
| closeTimeout | Time before toast closes. |
| icon | Optional. Expect f7Icon . Warning: Adding icon will hide the close button. |
| ... | Other options. See https://framework7.io/docs/toast.html#toast-parameters . |
| session | Shiny session. |

Examples

```
library(shiny)  
library(shinyMobile)  
  
app <- shinyApp(  
  ui = f7Page(  
    title = "Toast",  
    f7SingleLayout(  
      navbar = f7Navbar(title = "f7Toast"),  
      f7Button(inputId = "toast", label = "Open Toast")  
    )  
  ),  
  server = function(input, output, session) {  
    observeEvent(input$toast, {  
      f7Toast(  
        position = "top",
```

```

        text = "I am a toast. Eat me!"
      )
    })
  }
)

if (interactive() || identical(Sys.getenv("TESTTHAT"), "true")) app

```

f7Toggle

Framework7 toggle input

Description

f7Toggle creates a F7 toggle switch input.

updateF7Toggle changes the value of a toggle input on the client.

Usage

```
f7Toggle(inputId, label, checked = FALSE, color = NULL)
```

```

updateF7Toggle(
  inputId,
  checked = NULL,
  color = NULL,
  session = shiny::getDefaultReactiveDomain()
)

```

Arguments

| | |
|---------|---|
| inputId | Toggle input id. |
| label | Toggle label. |
| checked | Whether to check the toggle. FALSE by default. |
| color | Toggle color: NULL or "primary", "red", "green", "blue", "pink", "yellow", "orange", "purple", "deeppurple", "lightblue", "teal", "lime", "deeporange", "gray", "white", "black". |
| session | The Shiny session object. |

Examples

```

library(shiny)
library(shinyMobile)

app <- shinyApp(
  ui = f7Page(
    title = "f7Toggle",
    f7SingleLayout(

```

```

navbar = f7Navbar(title = "updateF7Toggle"),
f7Card(
  f7Button(inputId = "update", label = "Update toggle"),
  br(),
  f7Toggle(
    inputId = "toggle",
    label = "My toggle",
    color = "pink",
    checked = FALSE
  ),
  verbatimTextOutput("test")
)
),
server = function(input, output, session) {
  output$test <- renderPrint({
    input$update
  })

  observeEvent(input$update, {
    updateF7Toggle(
      inputId = "toggle",
      checked = !input$update,
      color = "green"
    )
  })
}
)

if (interactive() || identical(Sys.getenv("TESTTHAT"), "true")) app

```

f7Toolbar

Framework7 Toolbar

Description

f7Toolbar is a layout element located at the bottom or top. It is internally used by [f7Tabs](#) and can be used in the toolbar slot of [f7Page](#).

Usage

```

f7Toolbar(
  ...,
  position = c("bottom", "top"),
  hairline = deprecated(),
  shadow = deprecated(),
  icons = FALSE,
  scrollable = FALSE
)

```

Arguments

| | |
|------------|--|
| ... | Slot for f7Link or any other element. |
| position | Tabs position: "top" or "bottom". Or use different positions for iOS, MD themes by using: "top-ios", "top-md", "bottom-ios", or "bottom-md". |
| hairline | [Deprecated] : removed from Framework7. |
| shadow | [Deprecated] : removed from Framework7. |
| icons | Whether to use icons instead of text. Either ios or md icons. |
| scrollable | Whether to allow scrolling. FALSE by default. |

Author(s)

David Granjon, <dgranjon@gmail.com>

Examples

```
library(shiny)
library(shinyMobile)

app <- shinyApp(
  ui = f7Page(
    title = "Toolbar",
    toolbar = f7Toolbar(
      icons = TRUE,
      f7Link(
        label = "Link 1",
        href = "https://www.google.com",
        icon = f7Icon("link_circle_fill")
      ),
      f7Link(
        label = "Link 2",
        href = "https://maps.google.com",
        icon = f7Icon("location_circle_fill")
      )
    )
  ),
  server = function(input, output, session) {
  }
)

if (interactive() || identical(Sys.getenv("TESTTHAT"), "true")) app
```

Description

f7Tooltip creates a static tooltip, UI side.

addF7Tooltip adds a dynamic tooltip to the given target. The tooltip can be modified later.

updateF7Tooltip updates a tooltip from the server. Either toggle or update the text content.

Usage

```
f7Tooltip(tag, text)
```

```
addF7Tooltip(  
  id = NULL,  
  selector = NULL,  
  options,  
  session = shiny::getDefaultReactiveDomain()  
)
```

```
updateF7Tooltip(  
  id = NULL,  
  selector = NULL,  
  action = c("toggle", "update"),  
  text = NULL,  
  session = shiny::getDefaultReactiveDomain()  
)
```

Arguments

| | |
|----------|---|
| tag | Tooltip target. |
| text | New tooltip text value. See https://framework7.io/docs/tooltip#tooltip-parameters . |
| id | Tooltip target id. |
| selector | jQuery selector. Allow more customization for the target (nested tags). |
| options | List of options to pass to the tooltip. See https://framework7.io/docs/tooltip#tooltip-parameters . |
| session | Shiny session object. |
| action | Either toggle or update the tooltip. |

Examples

```
library(shiny)  
library(shinyMobile)  
  
lorem_ipsum <- "Lorem ipsum dolor sit amet!"  
  
tooltips <- data.frame(  
  id = paste0("target_", 1:2),  
  text = paste("Tooltip content", 1:2, lorem_ipsum),  
  stringsAsFactors = FALSE  
)
```

```

app <- shinyApp(
  ui = f7Page(
    title = "Tooltip",
    f7SingleLayout(
      navbar = f7Navbar(title = "f7Tooltip"),
      # Static tooltip
      f7Segment(
        f7Tooltip(
          f7Badge("Hover on me", color = "teal"),
          text = "A tooltip!"
        )
      ),
      # Dynamic tooltips
      f7Segment(
        f7Toggle(
          inputId = "toggle",
          "Enable tooltips",
          color = "deeporange",
          checked = TRUE
        )
      ),
      f7Segment(
        lapply(seq_len(nrow(tooltips)), function(i) {
          f7Button(
            inputId = sprintf("target_%s", i),
            sprintf("Target %s", i)
          )
        })
      ),
      f7Text("tooltip_text", "Tooltip new text", placeholder = "Type a text")
    )
  ),
  server = function(input, output, session) {
    # Update content
    observeEvent(input$tooltip_text, {
      lapply(seq_len(nrow(tooltips)), function(i) {
        updateF7Tooltip(
          id = tooltips[i, "id"],
          action = "update",
          text = input$tooltip_text
        )
      })
    }, ignoreInit = TRUE)

    observeEvent(input$toggle, {
      lapply(seq_len(nrow(tooltips)), function(i) {
        updateF7Tooltip(id = tooltips[i, "id"], action = "toggle")
      })
    }, ignoreInit = TRUE)

    # Create
    lapply(seq_len(nrow(tooltips)), function(i) {

```

```

    observeEvent(input[[tooltips[i, "id"]]], {
      addF7Tooltip(
        id = tooltips[i, "id"],
        options = list(
          text = tooltips[i, "text"]
        )
      )
    })
  })
}
)

if (interactive() || identical(Sys.getenv("TESTTHAT"), "true")) app
library(shiny)
library(shinyMobile)

lorem_ipsum <- "Lorem ipsum dolor sit amet!"

tooltips <- data.frame(
  id = paste0("target_", 1:2),
  text = paste("Tooltip content", 1:2, lorem_ipsum),
  stringsAsFactors = FALSE
)

app <- shinyApp(
  ui = f7Page(
    title = "Tooltip",
    f7SingleLayout(
      navbar = f7Navbar(title = "f7Tooltip"),
      # Static tooltip
      f7Segment(
        f7Tooltip(
          f7Badge("Hover on me", color = "teal"),
          text = "A tooltip!"
        )
      ),
      # Dynamic tooltips
      f7Segment(
        f7Toggle(
          inputId = "toggle",
          "Enable tooltips",
          color = "deeporange",
          checked = TRUE
        )
      ),
      f7Segment(
        lapply(seq_len(nrow(tooltips)), function(i) {
          f7Button(
            inputId = sprintf("target_%s", i),
            sprintf("Target %s", i)
          )
        })
      )
    )
  ),

```

```

    f7Text("tooltip_text", "Tooltip new text", placeholder = "Type a text")
  )
),
server = function(input, output, session) {
  # Update content
  observeEvent(input$tooltip_text, {
    lapply(seq_len(nrow(tooltips)), function(i) {
      updateF7Tooltip(
        id = tooltips[i, "id"],
        action = "update",
        text = input$tooltip_text
      )
    })
  }, ignoreInit = TRUE)

  observeEvent(input$toggle, {
    lapply(seq_len(nrow(tooltips)), function(i) {
      updateF7Tooltip(id = tooltips[i, "id"], action = "toggle")
    })
  }, ignoreInit = TRUE)

  # Create
  lapply(seq_len(nrow(tooltips)), function(i) {
    observeEvent(input[[tooltips[i, "id"]]], {
      addF7Tooltip(
        id = tooltips[i, "id"],
        options = list(
          text = tooltips[i, "text"]
        )
      )
    })
  })
}
)

if (interactive() || identical(Sys.getenv("TESTTHAT"), "true")) app

```

f7Treeview

Create a Framework 7 Treeview layout

Description

Create a Framework 7 Treeview layout

Usage

```

f7Treeview(
  ...,
  id,
  selectable = FALSE,

```



```

    withCheckbox = FALSE,
    startExpanded = FALSE
  )

```

Arguments

| | |
|---------------|--|
| ... | Slot for f7TreeviewGroup or f7TreeviewItem . |
| id | Treeview unique id. |
| selectable | Make treeview items selectable. Default is FALSE. |
| withCheckbox | Add a checkbox to each item. Default is FALSE. |
| startExpanded | Whether to expand the treeview at start. |

Examples

```

library(shiny)
library(shinyMobile)

app <- shinyApp(
  ui = f7Page(
    title = "My app",
    f7SingleLayout(
      navbar = f7Navbar(title = "f7Treeview"),

      # simple treeview
      f7BlockTitle("Simple"),
      f7Block(
        f7Treeview(
          id = "simple",
          lapply(1:3, function(i) f7TreeviewItem(label = paste0("Item ", letters[i])))
        )
      ),

      # simple treeview with icons
      f7BlockTitle("Icons"),
      f7Block(
        f7Treeview(
          id = "icons",
          lapply(1:3, function(i) f7TreeviewItem(label = paste0("Item ", letters[i]),
                                                    icon = f7Icon("folder_fill")))
        )
      ),

      # group treeview with icons
      f7BlockTitle("Group"),
      f7Block(
        f7Treeview(
          id = "group",
          startExpanded = TRUE,
          f7TreeviewGroup(
            title = "Images",
            icon = f7Icon("folder_fill"),

```

```

        toggleButton = TRUE,
        lapply(1:3, function(i) f7TreeviewItem(label = paste0("image", i, ".png"),
                                                icon = f7Icon("photo_fill")))
    )
)
),

# group treeview with selectable items
f7BlockTitle("Selectable items"),
f7Block(
  f7Treeview(
    id = "selectable",
    selectable = TRUE,
    f7TreeviewGroup(
      title = "Selected images",
      icon = f7Icon("folder_fill"),
      itemToggle = TRUE,
      lapply(1:3, function(i) f7TreeviewItem(label = paste0("image", i, ".png"),
                                              icon = f7Icon("photo_fill")))
    )
  )
),

# group treeview with checkbox items
f7BlockTitle("Checkbox"),
f7Block(
  f7Treeview(
    id = "checkbox",
    withCheckbox = TRUE,
    f7TreeviewGroup(
      title = "Selected images",
      icon = f7Icon("folder_fill"),
      itemToggle = TRUE,
      lapply(1:3, function(i) f7TreeviewItem(label = paste0("image", i, ".png"),
                                              icon = f7Icon("photo_fill")))
    )
  )
),

# group treeview with checkbox items
f7BlockTitle("With links"),
f7Block(
  f7Treeview(
    id = "links",
    f7TreeviewGroup(
      title = "Links",
      icon = f7Icon("link"),
      itemToggle = TRUE,
      f7TreeviewItem(label = "GitHub",
                     icon = f7Icon("logo_github"),
                     href = "https://github.com/"),
      f7TreeviewItem(label = "CRAN",
                     icon = f7Icon("link"),

```

```

        href = "https://cran.r-project.org/")),
      )
    )
  ),
),
server = function(input, output) {

  observe({
    req(input$selectable)
    print(input$selectable)
  })

  observe({
    req(input$checkbox)
    print(input$checkbox)
  })

}
)

if (interactive() || identical(Sys.getenv("TESTTHAT"), "true")) app

```

f7TreeviewGroup

Create a Framework 7 group for treeview items

Description

Create a Framework 7 group for treeview items

Usage

```
f7TreeviewGroup(..., title, icon, toggleButton = TRUE, itemToggle = FALSE)
```

Arguments

| | |
|--------------|---|
| ... | slot for f7TreeviewItem . |
| title | Group title. |
| icon | Expect f7Icon . |
| toggleButton | Whether or not to display a toggle button. Could be set to FALSE if <code>itemToggle</code> is TRUE. |
| itemToggle | In addition to (or instead of) a toggle button, the whole group can work like a toggle. By default this behaviour is disabled. @example inst/examples/treeview/app.R |

| | |
|----------------|---|
| f7TreeviewItem | <i>Create a Framework 7 Treeview item</i> |
|----------------|---|

Description

Create a Framework 7 Treeview item

Usage

```
f7TreeviewItem(label, icon = NULL, href = NULL)
```

Arguments

| | |
|-------|---------------------------------|
| label | Item label |
| icon | Expect f7Icon . |
| href | Item external link. |

Examples

```
library(shiny)
library(shinyMobile)

app <- shinyApp(
  ui = f7Page(
    title = "My app",
    f7SingleLayout(
      navbar = f7Navbar(title = "f7Treeview"),

      # simple treeview
      f7BlockTitle("Simple"),
      f7Block(
        f7Treeview(
          id = "simple",
          lapply(1:3, function(i) f7TreeviewItem(label = paste0("Item ", letters[i])))
        )
      ),

      # simple treeview with icons
      f7BlockTitle("Icons"),
      f7Block(
        f7Treeview(
          id = "icons",
          lapply(1:3, function(i) f7TreeviewItem(label = paste0("Item ", letters[i]),
                                                    icon = f7Icon("folder_fill")))
        )
      ),

      # group treeview with icons
      f7BlockTitle("Group"),
```

```

f7Block(
  f7Treeview(
    id = "group",
    startExpanded = TRUE,
    f7TreeviewGroup(
      title = "Images",
      icon = f7Icon("folder_fill"),
      toggleButton = TRUE,
      lapply(1:3, function(i) f7TreeviewItem(label = paste0("image", i, ".png"),
                                              icon = f7Icon("photo_fill")))
    )
  )
),

# group treeview with selectable items
f7BlockTitle("Selectable items"),
f7Block(
  f7Treeview(
    id = "selectable",
    selectable = TRUE,
    f7TreeviewGroup(
      title = "Selected images",
      icon = f7Icon("folder_fill"),
      itemToggle = TRUE,
      lapply(1:3, function(i) f7TreeviewItem(label = paste0("image", i, ".png"),
                                              icon = f7Icon("photo_fill")))
    )
  )
),

# group treeview with checkbox items
f7BlockTitle("Checkbox"),
f7Block(
  f7Treeview(
    id = "checkbox",
    withCheckbox = TRUE,
    f7TreeviewGroup(
      title = "Selected images",
      icon = f7Icon("folder_fill"),
      itemToggle = TRUE,
      lapply(1:3, function(i) f7TreeviewItem(label = paste0("image", i, ".png"),
                                              icon = f7Icon("photo_fill")))
    )
  )
),

# group treeview with checkbox items
f7BlockTitle("With links"),
f7Block(
  f7Treeview(
    id = "links",
    f7TreeviewGroup(
      title = "Links",

```

```

        icon = f7Icon("link"),
        itemToggle = TRUE,
        f7TreeViewItem(label = "GitHub",
                      icon = f7Icon("logo_github"),
                      href = "https://github.com/"),
        f7TreeViewItem(label = "CRAN",
                      icon = f7Icon("link"),
                      href = "https://cran.r-project.org/")),
    )
  )
),
server = function(input, output) {

  observe({
    req(input$selectable)
    print(input$selectable)
  })

  observe({
    req(input$checkbox)
    print(input$checkbox)
  })

}
)

if (interactive() || identical(Sys.getenv("TESTTHAT"), "true")) app

```

f7VirtualList

Framework7 virtual list

Description

f7VirtualList is a high performance list container. Use if you have too many components in [f7List](#).

f7VirtualListItem is an item component for [f7VirtualList](#).

Usage

```

f7VirtualList(
  id,
  items,
  rowsBefore = NULL,
  rowsAfter = NULL,
  cache = TRUE,
  mode = NULL,
  inset = FALSE,
  outline = FALSE,

```

```

    dividers = FALSE,
    strong = FALSE
)

f7VirtualListItem(
    ...,
    id = NULL,
    title = NULL,
    subtitle = NULL,
    header = NULL,
    footer = NULL,
    href = NULL,
    media = NULL,
    right = NULL,
    routable = FALSE
)

```

Arguments

| | |
|------------|---|
| id | Optional id for item. |
| items | List items. Slot for f7VirtualListItem . |
| rowsBefore | Amount of rows (items) to be rendered before current screen scroll position. By default it is equal to double amount of rows (items) that fit to screen. |
| rowsAfter | Amount of rows (items) to be rendered after current screen scroll position. By default it is equal to the amount of rows (items) that fit to screen. |
| cache | Disable or enable DOM cache for already rendered list items. In this case each item will be rendered only once and all further manipulations will be with DOM element. It is useful if your list items have some user interaction elements (like form elements or swipe outs) or could be modified. |
| mode | List mode. NULL, "simple", "links", "media" or "contacts". |
| inset | Whether to display a card border. FALSE by default. |
| outline | Outline style. Default to FALSE. |
| dividers | Dividers style. Default to FALSE. |
| strong | Strong style. Default to FALSE. |
| ... | Item text. |
| title | Item title. |
| subtitle | Item subtitle. |
| header | Item header. |
| footer | Item footer. |
| href | Item external link. |
| media | Expect f7Icon or img. |
| right | Right content if any. |

routable Works when href is not NULL. Default to FALSE. If TRUE, the list item may point to another page, but we recommend using [f7List](#) and [f7ListItem](#) instead. See [f7MultiLayout](#). Can also be used in combination with href = "#" to make items appear as links, but not actually navigate anywhere, which is useful for custom click events.

Examples

```
library(shiny)
library(shinyMobile)

app <- shinyApp(
  ui = f7Page(
    title = "Virtual List",
    f7SingleLayout(
      navbar = f7Navbar(
        title = "Virtual Lists"
      ),
      # controls
      f7Segment(
        f7Button(inputId = "appendItem", "Append Item"),
        f7Button(inputId = "prependItems", "Prepend Items"),
        f7Button(inputId = "insertBefore", "Insert before"),
        f7Button(inputId = "replaceItem", "Replace Item")
      ),
      f7Segment(
        f7Button(inputId = "deleteAllItems", "Remove All"),
        f7Button(inputId = "moveItem", "Move Item"),
        f7Button(inputId = "filterItems", "Filter Items")
      ),
      f7Grid(
        cols = 3,
        uiOutput("itemIndexUI"),
        uiOutput("itemNewIndexUI"),
        uiOutput("itemsFilterUI")
      ),
      # searchbar
      f7Searchbar(id = "search1"),
      # main content
      f7VirtualList(
        id = "vlist",
        rowsBefore = 2,
        rowsAfter = 2,
        mode = "media",
        items = lapply(1:1000, function(i) {
          f7VirtualListItem(
            id = paste0("vlist-item-", i),
            title = paste("Title", i),
            subtitle = paste("Subtitle", i),
            header = paste("Header", i),
            footer = paste("Footer", i),
            right = paste("Right", i),
```



```

        paste0("Content", i),
        media = img(style = "border-radius: 8px",
                    src = "https://cdn.framework7.io/placeholder/fashion-88x88-1.jpg")
      )
    })
  )
),
server = function(input, output) {

  output$itemIndexUI <- renderUI({
    req(input$vlist$length > 2)
    f7Stepper(
      inputId = "itemIndex",
      label = "Index",
      min = 1,
      value = 2,
      max = input$vlist$length
    )
  })

  output$itemNewIndexUI <- renderUI({
    req(input$vlist$length > 2)
    f7Stepper(
      inputId = "itemNewIndex",
      label = "New Index",
      min = 1,
      value = 1,
      max = input$vlist$length
    )
  })

  output$itemsFilterUI <- renderUI({
    input$appendItem
    input$prependItems
    input$insertBefore
    input$replaceItem
    input$deleteAllItems
    input$moveItem
    isolate({
      req(input$vlist$length > 2)
      f7Slider(
        inputId = "itemsFilter",
        label = "Items to Filter",
        min = 1,
        max = input$vlist$length,
        value = c(1, input$vlist$length)
      )
    })
  })

  observeEvent(input$appendItem, {
    updateF7VirtualList(

```

```

    id = "vlist",
    action = "appendItem",
    item = f7VirtualListItem(
      title = "New Item Title",
      right = "New Item Right",
      "New Item Content",
      media = img(src = "https://cdn.framework7.io/placeholder/fashion-88x88-3.jpg")
    )
  )
})

observeEvent(input$prependItems, {
  updateF7VirtualList(
    id = "vlist",
    action = "prependItems",
    items = lapply(1:5, function(i) {
      f7VirtualListItem(
        title = paste("Title", i),
        right = paste("Right", i),
        i,
        media = img(src = "https://cdn.framework7.io/placeholder/fashion-88x88-3.jpg")
      )
    })
  )
})

observeEvent(input$insertBefore, {
  updateF7VirtualList(
    id = "vlist",
    action = "insertItemBefore",
    index = input$itemIndex,
    item = f7VirtualListItem(
      title = "New Item Title",
      "New Item Content",
      media = img(src = "https://cdn.framework7.io/placeholder/fashion-88x88-3.jpg")
    )
  )
})

observeEvent(input$replaceItem, {
  updateF7VirtualList(
    id = "vlist",
    action = "replaceItem",
    index = input$itemIndex,
    item = f7VirtualListItem(
      title = "Replacement",
      "Replacement Content",
      media = img(src = "https://cdn.framework7.io/placeholder/fashion-88x88-3.jpg")
    )
  )
})

observeEvent(input$deleteAllItems, {

```

```
      updateF7VirtualList(  
        id = "vlist",  
        action = "deleteAllItems"  
      )  
    })  
  
    observeEvent(input$moveItem, {  
      updateF7VirtualList(  
        id = "vlist",  
        action = "moveItem",  
        oldIndex = input$itemIndex,  
        newIndex = input$itemNewIndex  
      )  
    })  
  
    observeEvent(input$filterItems, {  
      updateF7VirtualList(  
        id = "vlist",  
        action = "filterItems",  
        indexes = input$itemsFilter[1]:input$itemsFilter[2]  
      )  
    })  
  }  
)  
  
if (interactive() || identical(Sys.getenv("TESTTHAT"), "true")) app
```

getF7Colors

Function to get all colors available in shinyMobile

Description

Function to get all colors available in shinyMobile

Usage

```
getF7Colors()
```

Value

A vector containing colors

| | |
|-------------|---------------------------------|
| insertF7Tab | <i>Framework7 tab insertion</i> |
|-------------|---------------------------------|

Description

insertF7Tab inserts an [f7Tab](#) in an [f7Tabs](#).

Usage

```
insertF7Tab(  
  id,  
  tab,  
  target = NULL,  
  position = c("before", "after"),  
  select = FALSE,  
  session = shiny::getDefaultReactiveDomain()  
)
```

Arguments

| | |
|----------|---|
| id | f7Tabs id. |
| tab | f7Tab to insert. |
| target | f7Tab after of before which the new tab will be inserted. |
| position | Insert before or after: c("before", "after"). |
| select | Whether to select the newly inserted tab. FALSE by default. |
| session | Shiny session object. |

See Also

[f7Tabs](#)

| | |
|----------------|---|
| preview_mobile | <i>Allow to preview a given app on different devices.</i> |
|----------------|---|

Description

Allow to preview a given app on different devices.

Usage

```
preview_mobile(  
  appPath = NULL,  
  url = NULL,  
  port = 3838,  
  device = c("iphoneX", "galaxyNote8", "iphone8", "iphone8+", "iphone5s", "iphone5c",  
    "ipadMini", "iphone4s", "nexus5", "galaxyS5", "htcOne"),  
  color = NULL,  
  landscape = FALSE  
)
```

Arguments

| | |
|-----------|---|
| appPath | App to preview if local. |
| url | App to preview if online. |
| port | Default port. Ignored if url is provided. |
| device | Wrapper devices. |
| color | Wrapper color. Only with iphone8 (black, silver, gold), iphone8+ (black, silver, gold), iphone5s (black, silver, gold), iphone5c (white, red, yellow, green, blue), iphone4s (black, silver), ipadMini (black, silver) and galaxyS5 (black, white). |
| landscape | Whether to put the device wrapper in landscape mode. Default to FALSE. |

Value

A shiny app containing an iframe surrounded by the device wrapper.

Note

choose either url or appPath!

Examples

```
if (interactive()) {  
  library(shiny)  
  library(shinyMobile)  
  preview_mobile(appPath = "~/whatever", device = "galaxyNote8")  
  preview_mobile(url = "https://dgranjon.shinyapps.io/miniUI2DemoMd", device = "ipadMini")  
}
```

| | |
|-------------|--------------------------------|
| removeF7Tab | <i>Framework7 tab deletion</i> |
|-------------|--------------------------------|

Description

removeF7Tab removes an [f7Tab](#) in a [f7Tabs](#).

Usage

```
removeF7Tab(id, target, session = shiny::getDefaultReactiveDomain())
```

Arguments

| | |
|---------|----------------------------------|
| id | f7Tabs id. |
| target | f7Tab to remove. |
| session | Shiny session object. |

See Also

[f7Tabs](#)

| | |
|-----------------|-----------------------------|
| showF7Preloader | <i>Framework7 preloader</i> |
|-----------------|-----------------------------|

Description

showF7Preloader shows a preloader. When target is NULL, the overlay applies to the entire view, preventing to perform any actions. When type is not NULL, target is ignored.

updateF7Preloader updates a preloader.

hideF7Preloader hides a preloader.

Usage

```
showF7Preloader(  
  target = NULL,  
  color = NULL,  
  type = NULL,  
  id = NULL,  
  session = shiny::getDefaultReactiveDomain()  
)
```

```
updateF7Preloader(  
  id,  
  title = NULL,
```

```

    text = NULL,
    progress = NULL,
    session = shiny::getDefaultReactiveDomain()
  )

hideF7Preloader(
  target = NULL,
  id = NULL,
  session = shiny::getDefaultReactiveDomain()
)

```

Arguments

| | |
|----------|---|
| target | Element where preloader overlay will be added. |
| color | Preloader color. |
| type | Leave NULL to use the default preloader or use either "dialog" or "progress". |
| id | When type isn't NULL, an id is required to be able to use updateF7Preloader . |
| session | Shiny session object. |
| title | Dialog title. |
| text | Dialog text. |
| progress | Progress bar content. |

Examples

```

if (interactive()) {
  library(shiny)
  library(shinyMobile)

  # preloader in container
  shinyApp(
    ui = f7Page(
      title = "Preloader in container",
      f7SingleLayout(
        navbar = f7Navbar(
          title = "Preloader in container"
        ),
        # main content
        f7Block(
          f7Button("compute", "Compute")
        ),
        f7Block(textOutput("calc"))
      )
    ),
    server = function(input, output, session) {
      res <- reactiveVal(NULL)
      progress <- reactiveVal(NULL)
      output$calc <- renderText(res())

      observeEvent(input$compute, {

```

```

    res(NULL)
    progress(0)
    showF7Preloader(color = "red", type = "progress", id = "loader")
    for (i in seq_along(1:100)) {
      Sys.sleep(0.025)
      progress(i)
      updateF7Preloader(
        id = "loader",
        title = "Computing ...",
        text = sprintf("Done: %s/100", progress()),
        progress = progress()
      )
    }
    res("Result!")
  })

  observeEvent(res(), {
    hideF7Preloader(id = "loader")
  })
}
)
}

```

updateF7App

Update Framework7 configuration

Description

updateF7App allows to update a shinyMobile app at run time by injecting any configuration inside the current running instance. Useful if you want to share the same behavior across multiple elements. It can also be used to update the app theme, dark mode, or color.

Usage

```
updateF7App(options, session = shiny::getDefaultReactiveDomain())
```

Arguments

| | |
|---------|-----------------------|
| options | List of options. |
| session | Shiny session object. |

Note

This function may not work with all options and is intended for advanced/expert usage.

Examples

```
library(shiny)
library(shinyMobile)

colors <- c(
  lightblue = "#5ac8fa",
  pink = "#ff2d55",
  teal = "#009688",
  yellow = "#ffcc00"
)

app <- shinyApp(
  ui = f7Page(
    title = "Update App",
    options = (
      list(
        color = "#5ac8fa"
      )
    ),
    f7SingleLayout(
      navbar = f7Navbar(title = "Update App"),
      f7BlockTitle("Update f7Dialog configuration"),
      f7Segment(
        f7Button(
          inputId = "goButton",
          "Show f7Dialog"
        ),
        f7Button(
          inputId = "update",
          "Update config"
        )
      ),
      f7BlockTitle("Update theme"),
      f7Segment(
        f7Button(
          inputId = "theme_ios",
          "iOS theme"
        ),
        f7Button(
          inputId = "theme_md",
          "MD theme"
        )
      ),
      f7BlockTitle("Set dark mode"),
      f7Segment(
        f7Button(
          inputId = "enable_darkmode",
          "Enable darkmode"
        ),
        f7Button(
          inputId = "disable_darkmode",
          "Disable darkmode"
        )
      )
    )
  )
)
```

```

    )
  ),
  f7BlockTitle("Change color theme"),
  f7Segment(
    tagList(
      lapply(names(colors),
        function(color) {
          f7Button(
            inputId = paste0("color_", color),
            label = color,
            color = color,
          )
        }
      )
    )
  )
),
server = function(input, output, session) {
  observeEvent(input$goButton, {
    f7Dialog(
      id = "test2",
      title = "Dialog title",
      text = "This is an alert dialog",
      type = "confirm"
    )
  })

  observeEvent(input$update, {
    updateF7App(
      options = list(
        dialog = list(
          buttonOk = "Yeaaaah!",
          buttonCancel = "Ouuups!"
        )
      )
    )

    f7Dialog(
      id = "test",
      title = "Warning",
      type = "confirm",
      text = "Look at me, I have a new buttons!"
    )
  })

  observeEvent(input$theme_ios, {
    updateF7App(
      options = list(
        theme = "ios"
      )
    )
  })
})

```

```

observeEvent(input$theme_md, {
  updateF7App(
    options = list(
      theme = "md"
    )
  )
})

observeEvent(input$enable_darkmode, {
  updateF7App(
    options = list(
      dark = TRUE
    )
  )
})

observeEvent(input$disable_darkmode, {
  updateF7App(
    options = list(
      dark = FALSE
    )
  )
})

lapply(names(colors), function(color) {
  observeEvent(input[[paste0("color_", color)]], {
    updateF7App(
      options = list(
        color = colors[color]
      )
    )
  })
})

}
)

if (interactive() || identical(Sys.getenv("TESTTHAT"), "true")) app

```

updateF7Entity

Update Framework7 entity

Description

updateF7Entity allows to update any Framework7 instance from the server. For each entity, the list of updatable properties may significantly vary. Please refer to the Framework7 documentation at <https://framework7.io/docs/>. Currently, updateF7Entity supports [f7Gauge](#), [f7Swiper](#), [f7Searchbar](#), [f7PhotoBrowser](#), [f7Popup](#), [f7ListIndex](#) and [f7ActionSheet](#).

Usage

```
updateF7Entity(id, options, session = shiny::getDefaultReactiveDomain())
```

Arguments

| | |
|---------|--|
| id | Element id. |
| options | Configuration list. Tightly depends on the entity. See https://framework7.io/docs/ . |
| session | Shiny session object. |

Examples

```
library(shiny)
library(shinyMobile)

app <- shinyApp(
  ui = f7Page(
    title = "Update Entity",
    f7SingleLayout(
      navbar = f7Navbar(title = "Update action sheet instance"),
      f7BlockTitle("Action sheet", size = "large"),
      f7Segment(
        f7Button(
          inputId = "goButton",
          "Show action sheet",
          fill = FALSE,
          outline = TRUE
        ),
        f7Button(
          inputId = "update_action_sheet",
          "Update config",
          fill = FALSE,
          outline = TRUE
        ),
        f7Button(
          inputId = "reset_action_sheet",
          "Reset",
          fill = FALSE,
          outline = TRUE
        )
      ),
      f7BlockTitle("Gauges", size = "large"),
      f7Block(
        f7Gauge(
          id = "mygauge",
          type = "semicircle",
          value = 50,
          borderColor = "#2196f3",
          borderWidth = 10,
          valueFontSize = 41,
          valueTextColor = "#2196f3",
```

```

        labelText = "amount of something"
    )
),
f7Block(f7Button("update_gauge", "Update Gauge")),
f7BlockTitle("Swiper", size = "large"),
f7Swiper(
  id = "swiper",
  lapply(1:20, function(c) {
    f7Slide(
      f7Card(
        title = sprintf("Slide %s", c)
      )
    )
  })
),
f7Block(f7Button("update_swiper", "Update Swiper")),
f7BlockTitle("Photo Browser", size = "large"),
f7Segment(
  f7Button(
    "show_photobrowser",
    "Open photo browser",
    fill = FALSE,
    outline = TRUE
  ),
  f7Button(
    "update_photobrowser",
    "Update photo browser",
    fill = FALSE,
    outline = TRUE
  )
),
f7BlockTitle("Popup", size = "large"),
f7Segment(
  f7Button(
    "toggle",
    "Toggle Popup",
    fill = FALSE,
    outline = TRUE
  ),
  f7Button(
    "update",
    "Update Popup",
    fill = FALSE,
    outline = TRUE
  )
)
),
),
server = function(input, output, session) {
  observeEvent(input$goButton, {
    f7ActionSheet(
      grid = TRUE,
      id = "action1",

```

```

    buttons = list(
      list(
        text = "Notification",
        icon = f7Icon("info"),
        color = NULL
      ),
      list(
        text = "Dialog",
        icon = f7Icon("lightbulb_fill"),
        color = NULL
      )
    )
  )
}

observeEvent(input$update_action_sheet, {
  updateF7Entity(
    id = "action1",
    options = list(
      buttons = list(
        list(
          text = "Notification",
          icon = f7Icon("info"),
          color = NULL
        )
      )
    )
  )
}

observeEvent(input$reset_action_sheet, {
  updateF7Entity(
    id = "action1",
    options = list(
      buttons = list(
        list(
          text = "Notification",
          icon = f7Icon("info"),
          color = NULL
        ),
        list(
          text = "Dialog",
          icon = f7Icon("lightbulb_fill"),
          color = NULL
        )
      )
    )
  )
}

observeEvent(input$update_gauge, {
  new_val <- 75
  updateF7Entity(

```

```

    id = "mygauge",
    options = list(
      # Must be between 0 and 1
      value = new_val / 100,
      valueText = paste0(new_val, "%"),
      labelText = "New label!"
    )
  )
})

```

```

observeEvent(input$update_swiper, {
  updateF7Entity(
    "swiper",
    options = list(
      speed = 100,
      slidesPerView = 2,
      spaceBetween = 10,
      autoplay = TRUE,
      scrollbar = list(
        enabled = FALSE
      ),
      navigation = list(
        enabled = FALSE
      ),
      pagination = list(
        type = "progressbar"
      ),
      grid = list(
        fill = "columns",
        rows = 4
      ),
      thumbs = TRUE
    )
  )
})

```

```

observeEvent(input$show_photobrowser, {
  f7PhotoBrowser(
    id = "photobrowser1",
    theme = "dark",
    type = "page",
    photos = list(
      list(url = "https://cdn.framework7.io/placeholder/sports-1024x1024-1.jpg"),
      list(url = "https://cdn.framework7.io/placeholder/sports-1024x1024-2.jpg"),
      list(
        url = "https://cdn.framework7.io/placeholder/sports-1024x1024-3.jpg",
        caption = "Me cycling"
      )
    )
  ),
  thumbs = c(
    "https://cdn.framework7.io/placeholder/sports-1024x1024-1.jpg",
    "https://cdn.framework7.io/placeholder/sports-1024x1024-2.jpg",
    "https://cdn.framework7.io/placeholder/sports-1024x1024-3.jpg"
  )
})

```

```

    )
  )
})

observeEvent(input$update_photobrowser, {
  updateF7Entity(
    "photobrowser1",
    options = list(
      type = "popup",
      popupPush = TRUE,
      toolbar = FALSE,
      photos = list(
        list(url = "https://cdn.framework7.io/placeholder/sports-1024x1024-1.jpg")
      ),
      thumbs = list("https://cdn.framework7.io/placeholder/sports-1024x1024-1.jpg")
    )
  )
})

observeEvent(input$toggle, {
  f7Popup(
    id = "popup",
    title = "My first popup",
    f7Text(
      "text", "Popup content",
      "This is my first popup ever, I swear!"
    ),
    verbatimTextOutput("res")
  )
})

observeEvent(input$update, {
  updateF7Entity(
    id = "popup",
    options = list(
      swipeToClose = TRUE,
      animate = FALSE,
      closeOnEscape = TRUE,
      # Content must contain the popup
      # layout!!!
      content = '<div class="popup">
        <div class="view">
          <div class="page">
            <div class="navbar">
              <div class="navbar-bg"></div>
              <div class="navbar-inner">
                <div class="title">Popup</div>
                <div class="right">
                  <!-- Link to close popup -->
                  <a class="link popup-close">Close</a>
                </div>
              </div>
            </div>
          </div>
        </div>
      </div>'
    )
  )
})

```



```

        <div class="page-content">
          <div class="block">New content ...</div>
        </div>
      </div>
    </div>
  </div>'
)
)
})
}
)

if (interactive() || identical(Sys.getenv("TESTTHAT"), "true")) app

```

| | |
|----------------|------------------------------------|
| updateF7Routes | <i>Update routes on the server</i> |
|----------------|------------------------------------|

Description

[Experimental] Add a route to existing app routes.

Usage

```
updateF7Routes(routes, session = shiny::getDefaultReactiveDomain())
```

Arguments

| | |
|---------|-----------------------|
| routes | New list of routes. |
| session | Shiny session object. |

| | |
|--------------|---|
| updateF7Tabs | <i>Update a Framework 7 tabsetPanel</i> |
|--------------|---|

Description

Update [f7Tabs](#).

Usage

```
updateF7Tabs(id, selected = NULL, session = shiny::getDefaultReactiveDomain())
```

Arguments

| | |
|----------|---|
| id | Id of the f7Tabs to update. |
| selected | Newly selected tab. |
| session | Shiny session object. |

See Also[f7Tabs](#)

updateF7VirtualList *Update an [f7VirtualList](#) on the server side*

Description

This function wraps all methods from <https://framework7.io/docs/virtual-list.html>

Usage

```
updateF7VirtualList(  
  id,  
  action = c("appendItem", "appendItems", "prependItem", "prependItems", "replaceItem",  
    "replaceAllItems", "moveItem", "insertItemBefore", "filterItems", "deleteItem",  
    "deleteAllItems", "scrollToItem"),  
  item = NULL,  
  items = NULL,  
  index = NULL,  
  indexes = NULL,  
  oldIndex = NULL,  
  newIndex = NULL,  
  session = shiny::getDefaultReactiveDomain()  
)
```

Arguments

| | |
|----------|---|
| id | f7VirtualList to update. |
| action | Action to perform. See https://framework7.io/docs/virtual-list.html . |
| item | If action is one of <code>appendItem</code> , <code>prependItem</code> , <code>replaceItem</code> , <code>insertItemBefore</code> . |
| items | If action is one of <code>appendItems</code> , <code>prependItems</code> , <code>replaceAllItems</code> . |
| index | If action is one of <code>replaceItem</code> , <code>insertItemBefore</code> , <code>deleteItem</code> . |
| indexes | If action if one of <code>filterItems</code> , <code>deleteItems</code> . |
| oldIndex | If action is <code>moveItem</code> . |
| newIndex | If action is <code>moveItem</code> . |
| session | Shiny session. |

Examples

```

library(shiny)
library(shinyMobile)

app <- shinyApp(
  ui = f7Page(
    title = "Virtual List",
    f7SingleLayout(
      navbar = f7Navbar(
        title = "Virtual Lists"
      ),
      # controls
      f7Segment(
        f7Button(inputId = "appendItem", "Append Item"),
        f7Button(inputId = "prependItems", "Prepend Items"),
        f7Button(inputId = "insertBefore", "Insert before"),
        f7Button(inputId = "replaceItem", "Replace Item")
      ),
      f7Segment(
        f7Button(inputId = "deleteAllItems", "Remove All"),
        f7Button(inputId = "moveItem", "Move Item"),
        f7Button(inputId = "filterItems", "Filter Items")
      ),
      f7Grid(
        cols = 3,
        uiOutput("itemIndexUI"),
        uiOutput("itemNewIndexUI"),
        uiOutput("itemsFilterUI")
      ),
      # searchbar
      f7Searchbar(id = "search1"),
      # main content
      f7VirtualList(
        id = "vlist",
        rowsBefore = 2,
        rowsAfter = 2,
        mode = "media",
        items = lapply(1:1000, function(i) {
          f7VirtualListItem(
            id = paste0("vlist-item-", i),
            title = paste("Title", i),
            subtitle = paste("Subtitle", i),
            header = paste("Header", i),
            footer = paste("Footer", i),
            right = paste("Right", i),
            paste0("Content", i),
            media = img(style = "border-radius: 8px",
              src = "https://cdn.framework7.io/placeholder/fashion-88x88-1.jpg")
          )
        })
      )
    )
  )
)

```

```

),
server = function(input, output) {

  output$itemIndexUI <- renderUI({
    req(input$vlist$length > 2)
    f7Stepper(
      inputId = "itemIndex",
      label = "Index",
      min = 1,
      value = 2,
      max = input$vlist$length
    )
  })

  output$itemNewIndexUI <- renderUI({
    req(input$vlist$length > 2)
    f7Stepper(
      inputId = "itemNewIndex",
      label = "New Index",
      min = 1,
      value = 1,
      max = input$vlist$length
    )
  })

  output$itemsFilterUI <- renderUI({
    input$appendItem
    input$prependItems
    input$insertBefore
    input$replaceItem
    input$deleteAllItems
    input$moveItem
    isolate({
      req(input$vlist$length > 2)
      f7Slider(
        inputId = "itemsFilter",
        label = "Items to Filter",
        min = 1,
        max = input$vlist$length,
        value = c(1, input$vlist$length)
      )
    })
  })

  observeEvent(input$appendItem, {
    updateF7VirtualList(
      id = "vlist",
      action = "appendItem",
      item = f7VirtualListItem(
        title = "New Item Title",
        right = "New Item Right",
        "New Item Content",
        media = img(src = "https://cdn.framework7.io/placeholder/fashion-88x88-3.jpg")
      )
    )
  })
}

```

```
    )
  )
})

observeEvent(input$prependItems, {
  updateF7VirtualList(
    id = "vlist",
    action = "prependItems",
    items = lapply(1:5, function(i) {
      f7VirtualListItem(
        title = paste("Title", i),
        right = paste("Right", i),
        i,
        media = img(src = "https://cdn.framework7.io/placeholder/fashion-88x88-3.jpg")
      )
    })
  )
})

observeEvent(input$insertBefore, {
  updateF7VirtualList(
    id = "vlist",
    action = "insertItemBefore",
    index = input$itemIndex,
    item = f7VirtualListItem(
      title = "New Item Title",
      "New Item Content",
      media = img(src = "https://cdn.framework7.io/placeholder/fashion-88x88-3.jpg")
    )
  )
})

observeEvent(input$replaceItem, {
  updateF7VirtualList(
    id = "vlist",
    action = "replaceItem",
    index = input$itemIndex,
    item = f7VirtualListItem(
      title = "Replacement",
      "Replacement Content",
      media = img(src = "https://cdn.framework7.io/placeholder/fashion-88x88-3.jpg")
    )
  )
})

observeEvent(input$deleteAllItems, {
  updateF7VirtualList(
    id = "vlist",
    action = "deleteAllItems"
  )
})

observeEvent(input$moveItem, {
```

```

    updateF7VirtualList(
      id = "vlist",
      action = "moveItem",
      oldIndex = input$itemIndex,
      newIndex = input$itemNewIndex
    )
  })

  observeEvent(input$filterItems, {
    updateF7VirtualList(
      id = "vlist",
      action = "filterItems",
      indexes = input$itemsFilter[1]:input$itemsFilter[2]
    )
  })
}
)

if (interactive() || identical(Sys.getenv("TESTTHAT"), "true")) app

```

 validateF7Input

Framework7 input validation

Description

validateF7Input is a function to validate a given shinyMobile input.

Usage

```

validateF7Input(
  inputId,
  info = NULL,
  pattern = NULL,
  error = NULL,
  session = shiny::getDefaultReactiveDomain()
)

```

Arguments

| | |
|---------|---|
| inputId | Input to validate. |
| info | Additional text to display below the input field. |
| pattern | Pattern for validation. Regex. |
| error | Error text. |
| session | Shiny session object. |

Note

Only works for `f7Text`, `f7Password` and `f7TextArea`. See more at <https://framework7.io/docs/inputs.html>.

Examples

```
library(shiny)
library(shinyMobile)

app <- shinyApp(
  ui = f7Page(
    title = "Validate inputs",
    f7SingleLayout(
      navbar = f7Navbar(title = "validateF7Input"),
      f7Text(
        inputId = "caption",
        label = "Caption",
        value = "Data Summary"
      ),
      verbatimTextOutput("value"),
      hr(),
      f7Text(
        inputId = "caption2",
        label = "Enter a number",
        value = 1
      ),
      hr(),
      f7Password(
        inputId = "password",
        label = "Password"
      )
    )
  ),
  server = function(input, output, session) {
    observe({
      validateF7Input(inputId = "caption", info = "Whatever")
      validateF7Input(
        inputId = "caption2",
        pattern = "[0-9]*",
        error = "Only numbers please!"
      )
      validateF7Input(
        inputId = "password",
        pattern = "^(?=.*[a-z])(?=.*[A-Z])(?=.*\\d)[a-zA-Z\\d]{8,}$",
        error = "Password must contain at least one
          number and one uppercase and lowercase letter,
          and at least 8 or more characters"
      )
    })

    output$value <- renderPrint({
      input$caption
    })
  }
)
```

```
    })  
  }  
)  
  
if (interactive() || identical(Sys.getenv("TESTTHAT"), "true")) app
```


Index

add_shinyMobile_deps, 6
addF7Popover, 4
addF7Tooltip (f7Tooltip), 132

col2hex, 66, 73

f7Accordion, 6, 6
f7AccordionItem (f7Accordion), 6
f7ActionSheet, 8, 8, 155
f7Align, 10
f7AutoComplete, 11
f7Back, 14, 14
f7Badge, 14, 48
f7Block, 7, 15, 15
f7BlockFooter, 15, 16
f7BlockFooter (f7Block), 15
f7BlockHeader, 16
f7BlockHeader (f7Block), 15
f7BlockTitle (f7Block), 15
f7Button, 18, 18, 19
f7Card, 21
f7Checkbox, 24, 24
f7CheckboxChoice, 26, 86
f7CheckboxChoice (f7CheckboxGroup), 25
f7CheckboxGroup, 25, 25
f7Chip, 27
f7ColorPicker, 29
f7DatePicker, 30
f7DefaultOptions, 33, 66, 72
f7Dialog, 34
f7DownloadButton, 36
f7ExpandableCard, 21, 22
f7ExpandableCard (f7Card), 21
f7Fab, 37, 37, 38, 39
f7FabClose, 38
f7FabMorphTarget (f7Fabs), 38
f7Fabs, 37, 38, 38, 39
f7File, 41
f7Float, 42
f7Form, 43
f7Found, 88
f7Found (f7Searchbar), 88
f7Gallery, 45
f7Gauge, 45, 155
f7Grid, 48
f7HideOnEnable, 88
f7HideOnEnable (f7Searchbar), 88
f7HideOnSearch, 88, 89
f7HideOnSearch (f7Searchbar), 88
f7Icon, 19, 48, 50, 57, 98, 114, 119, 129, 139, 140, 143
f7Item, 49, 50, 77
f7Items, 50
f7Link, 50, 132
f7List, 26, 30, 32, 51, 57, 80, 86, 142, 144
f7ListGroup, 51, 54
f7ListIndex, 52, 54, 155
f7ListItem, 51, 54, 57, 110, 144
f7Login, 58, 58
f7LoginServer, 59
f7LoginServer (f7Login), 58
f7Margin, 60
f7Message, 63, 64
f7Message (f7Messages), 63
f7MessageBar, 61, 61, 115
f7Messages, 61, 63, 63
f7MultiLayout, 58, 65, 68, 144
f7Navbar, 66, 67, 67, 73, 94, 102, 109, 115
f7Next, 69, 69
f7NotFound, 88, 89
f7NotFound (f7Searchbar), 88
f7Notif, 70
f7Padding, 71
f7Page, 34, 72, 122, 131
f7Panel, 68, 74, 74, 76, 94, 102, 115
f7PanelItem, 76, 77
f7PanelItem (f7PanelMenu), 76
f7PanelMenu, 74, 76
f7Password, 123, 167

- f7Password (f7Text), 123
- f7PhotoBrowser, 77, 155
- f7Picker, 78
- f7Popup, 81, 155
- f7Progress, 84
- f7Radio, 85
- f7RadioChoice, 26, 86
- f7RadioChoice (f7Radio), 85
- f7Searchbar, 88, 89, 155
- f7SearchbarTrigger, 68, 88, 89
- f7SearchbarTrigger (f7Searchbar), 88
- f7SearchIgnore, 88
- f7SearchIgnore (f7Searchbar), 88
- f7Segment, 19
- f7Segment (f7Button), 18
- f7Select, 90, 90, 100
- f7Sheet, 91, 91, 119
- f7SingleLayout, 66, 67, 72, 94, 102
- f7Skeleton, 95
- f7Slide (f7Swiper), 112
- f7Slider, 97
- f7SmartSelect, 100
- f7SplitLayout, 49, 67, 72, 74, 76, 102
- f7Stepper, 106
- f7SubNavbar, 68, 109
- f7Swipeout, 110, 110
- f7SwipeoutItem, 110
- f7SwipeoutItem (f7Swipeout), 110
- f7Swiper, 112, 112, 155
- f7Tab, 49, 114, 119, 120, 148, 150
- f7TabLayout, 39, 67, 72, 98, 115, 119
- f7Table, 118
- f7TabLink, 119, 119
- f7Tabs, 14, 66, 69, 70, 115, 119, 119, 131, 148, 150, 161, 162
- f7TapHold, 66, 73, 122, 122
- f7Text, 13, 90, 123, 167
- f7TextArea, 167
- f7TextArea (f7Text), 123
- f7Timeline, 126, 126, 127
- f7TimelineItem (f7Timeline), 126
- f7Toast, 128
- f7Toggle, 130
- f7Toolbar, 66, 73, 94, 102, 131
- f7Tooltip, 132
- f7Treeview, 136
- f7TreeviewGroup, 137, 139
- f7TreeviewItem, 137, 139, 140
- f7VirtualList, 142, 142, 162
- f7VirtualListItem, 143
- f7VirtualListItem (f7VirtualList), 142
- getF7Colors, 66, 73, 98, 147
- hideF7Preloader (showF7Preloader), 150
- insertF7Tab, 148
- preview_mobile, 148
- removeF7Tab, 150
- showF7Preloader, 150
- toggleF7Popover, 4
- toggleF7Popover (addF7Popover), 4
- updateF7Accordion, 6
- updateF7Accordion (f7Accordion), 6
- updateF7ActionSheet, 8
- updateF7ActionSheet (f7ActionSheet), 8
- updateF7App, 152
- updateF7AutoComplete (f7AutoComplete), 11
- updateF7Button (f7Button), 18
- updateF7Card (f7Card), 21
- updateF7Checkbox (f7Checkbox), 24
- updateF7DatePicker (f7DatePicker), 30
- updateF7Entity, 77, 155
- updateF7Fab (f7Fab), 37
- updateF7Fabs (f7Fabs), 38
- updateF7Form, 43
- updateF7Form (f7Form), 43
- updateF7Gauge (f7Gauge), 45
- updateF7Login, 59
- updateF7Login (f7Login), 58
- updateF7MessageBar (f7MessageBar), 61
- updateF7Messages, 63
- updateF7Messages (f7Messages), 63
- updateF7Navbar (f7Navbar), 67
- updateF7Panel (f7Panel), 74
- updateF7Picker (f7Picker), 78
- updateF7Preloader, 151
- updateF7Preloader (showF7Preloader), 150
- updateF7Progress (f7Progress), 84
- updateF7Radio (f7Radio), 85
- updateF7Routes, 161
- updateF7Select (f7Select), 90

updateF7Sheet, [91](#)
updateF7Sheet (f7Sheet), [91](#)
updateF7Slider (f7Slider), [97](#)
updateF7SmartSelect (f7SmartSelect), [100](#)
updateF7Stepper (f7Stepper), [106](#)
updateF7Tabs, [114](#), [161](#)
updateF7Text (f7Text), [123](#)
updateF7TextArea (f7Text), [123](#)
updateF7Toggle (f7Toggle), [130](#)
updateF7Tooltip (f7Tooltip), [132](#)
updateF7VirtualList, [162](#)

validateCssUnit(), [37](#)
validateF7Input, [166](#)