Package 'shinyMobile'

November 30, 2019

Type Package

Title Mobile Ready 'shiny' Apps with Standalone Capabilities

Version 0.1.0

Maintainer David Granjon <dgranjon@ymail.com>

Description

Develop outstanding 'shiny' apps for 'iOS', 'Android', desktop 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

License GPL-2

Encoding UTF-8

URL https://github.com/RinteRface/shinyMobile, https://rinterface.github.io/shinyMobile/

BugReports https://github.com/RinteRface/shinyMobile/issues

LazyData true

RoxygenNote 7.0.0

Suggests knitr, rmarkdown, plotly, stats, testthat (>= 2.1.0), V8, cli, rstudioapi, shinyWidgets, echarts4r

VignetteBuilder knitr

NeedsCompilation no

Author David Granjon [aut, cre], Victor Perrier [aut], John Coene [ctb], Isabelle Rudolf [aut], Framework7 [ctb, cph] (Full featured HTML framework for building iOS & Android apps)

Repository CRAN

Date/Publication 2019-11-30 11:50:03 UTC

R topics documented:

`create_manifest` *Create a manifest for your shiny app*

Description

This is a central piece if you want to have your app standalone for instance

Usage

```
create_manifest(
  path,
  name = "My App",
  shortName = "My App",
  description = "What it does!",
  lang = "en-US",
  startUrl,
  display = c("minimal-ui", "standalone", "fullscreen", "browser"),
  icons
)
```

Arguments

path	package path.
name	App name.
shortName	App short name.
description	App description
lang	App language (en-US by default).
startUrl	Page to open at start.
display	Display mode. Choose among `c("minimal-ui","standalone","fullscreen","browser")`. In practice, you want the standalone mode so that the app looks like a native app.
icons	Dataframe containing icons specs for instance `data.frame(src = rep("icons/128x128.png",10),siz = rep("128x128",10),types = rep("image/png",10))`. src gives the icon path (in the www folder for instance), sizes gives the size and types the type.

Value

This function creates a www folder for your shiny app. Must specify the path. It also creates 2 additional folders to contain icons and splashscreen (splashscreen creation is not automatic for iOS). It also creates the manifest.json file in this www folder. You still have to create your own icons and splashscreen!

Note

See https://developer.mozilla.org/en-US/docs/Web/Manifest for more informations.

Examples

```
create_manifest(
  path = tempdir(),
  name = "My App",
  shortName = "My App",
  description = "What it does!",
  lang = "en-US",
  startUrl = "https://www.google.com/",
  display = "standalone",
  icons = data.frame(
    src = rep("icons/128x128.png", 10),
    sizes = rep("128x128", 10),
    types = rep("image/png", 10)
  )
)
```

f7Accordion *Create a Framework7 accordion*

Description

Build a Framework7 accordion

Usage

```
f7Accordion(..., inputId = NULL, multiCollapse = FALSE)
```

Arguments

...	Slot for f7AccordionItem.
inputId	Optional id to recover the state of the accordion.
multiCollapse	Whether to open multiple items at the same time. FALSE by default.

Author(s)

David Granjon, <dgranjon@ymail.com>

Examples

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

 shiny::shinyApp(
  ui = f7Page(
    title = "Accordions",
    f7SingleLayout(
     navbar = f7Navbar("Accordions"),
     f7Accordion(
      inputId = "myaccordion1",
      f7AccordionItem(
       title = "Item 1",
       f7Block("Item 1 content"),
       open = TRUE
      ),
      f7AccordionItem(
       title = "Item 2",
       f7Block("Item 2 content")
      )
     ),
     f7Accordion(
      multiCollapse = TRUE,
      inputId = "myaccordion2",
      f7AccordionItem(
       title = "Item 1",
```

```
          f7Block("Item 1 content")
         ),
         f7AccordionItem(
          title = "Item 2",
          f7Block("Item 2 content")
         )
        )
       )
     ),
     server = function(input, output, session) {
      observe({
       print(
        list(
         accordion1 = input$myaccordion1,
         accordion2 = input$myaccordion2
        )
       )
      })
     }
    )
   }
```

`f7AccordionItem` *Create a Framework7 accordion item*

Description

Build a Framework7 accordion item

Usage

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

Arguments

`...`	Item content such as f7Block or any f7 element.
`title`	Item title.
open	Whether the item is open at start. FALSE by default.

Author(s)

David Granjon, <dgranjon@ymail.com>

f7ActionSheet *Create a framework7 action sheet*

Description

Create a framework7 action sheet

Usage

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

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.
session	Shiny session object.
grid	Whether to display buttons on a grid. Default to FALSE.
buttons	dataframe of buttons such as `buttons <-data.frame( text = c('Button 1','Button 2'),color = c(NA,NA) )`. The currently selection button can be accessed via `input$button`. The value is numeric. When the action sheet is closed,`input$button` is NULL. This is useful when you want to trigger events after a specific button click.
icons	A list of icons for buttons. Expect f7Icon.

Examples

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

 shiny::shinyApp(
   ui = f7Page(
     title = "Action sheet",
     f7SingleLayout(
       navbar = f7Navbar("Action sheet"),
       br(),
       f7Button(inputId = "go", "Show action sheet", color = "red")
     )
   ),
   server = function(input, output, session) {
```

```
    observe({
      print(list(
        sheetOpen = input$action1,
        button = input$button
      ))
    })

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

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

    observeEvent(input$go, {
      f7ActionSheet(
        grid = TRUE,
        id = "action1",
        icons = list(f7Icon("info"), f7Icon("lightbulb_fill")),
        buttons = data.frame(
          text = c('Notification', 'Dialog'),
          color = c(NA, NA)
        )
      )
    })
  }
 )
}
```

---

f7Align *Create a Framework7 align*

Description

Build a Framework7 align

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)

 shiny::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) {}
 )
}
```

---

f7Appbar *Create a Framework 7 appbar*

Description

Displayed on top of f7Navbar. Interestingly, f7Appbar can also trigger f7Panel.

Usage

```
f7Appbar(..., left_panel = FALSE, right_panel = FALSE, maximizable = FALSE)
```

Arguments

...	Any UI content such as f7Searchbar, f7Next and f7Back. It is best practice to wrap f7Next and f7Back in a f7Flex.
left_panel	Whether to enable the left panel. FALSE by default.
right_panel	Whether to enable the right panel. FALSE by default.
maximizable	Whether to allow fullscreen. FALSE by default. This should only be used when the app runs on a desktop.

Examples

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

 cities <- names(precip)

 shiny::shinyApp(
   ui = f7Page(
     title = "My app",
     f7Appbar(
       f7Flex(f7Back(targetId = "tabset"),f7Next(targetId = "tabset")),
       f7Searchbar(id = "search1", inline = TRUE)
     ),
     f7TabLayout(
       navbar = f7Navbar(
         title = "f7Appbar",
         hairline = FALSE,
         shadow = TRUE
       ),
       f7Tabs(
         animated = FALSE,
         swipeable = TRUE,
         id = "tabset",
         f7Tab(
           tabName = "Tab 1",
           icon = f7Icon("email"),
           active = TRUE,
           "Text 1"
         ),
         f7Tab(
           tabName = "Tab 2",
           icon = f7Icon("today"),
           active = FALSE,
           "Text 2"
         ),
         f7Tab(
           tabName = "Tab 3",
           icon = f7Icon("cloud_upload"),
           active = FALSE,
           "Text 3"
         )
```

```
      )
    )
  ),
  server = function(input, output) {}
)
}
```

`f7AutoComplete` *Create a Framework7 autocomplete input*

Description

Build a Framework7 autocomplete input

Usage

```
f7AutoComplete(
  inputId,
  label,
  placeholder = NULL,
  value = choices[1],
  choices,
  typeahead = TRUE,
  expandInput = TRUE,
  type = c("popup", "page", "dropdown"),
  dropdownPlaceholderText = NULL,
  multiple = FALSE
)
```

Arguments

`inputId`	Autocomplete input id.
`label`	Autocomplete label.
`placeholder`	Text to write in the container.
`value`	Autocomplete initial value, if any.
`choices`	Autocomplete choices.
`typeahead`	Enables type ahead, will prefill input value with first item in match.
`expandInput`	If TRUE then input which is used as item-input in List View will be expanded to full screen wide during dropdown visible.
`type`	Defines how to open Autocomplete, can be page or popup (for Standalone) or dropdown.
`dropdownPlaceholderText`	Specify dropdown placeholder text.
`multiple`	Whether to allow multiple value selection. Only works when type is 'popup' or 'page'.

Author(s)

David Granjon, <dgranjon@ymail.com>

Examples

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

 shinyApp(
   ui = f7Page(
    title = "My app",
    f7SingleLayout(
     navbar = f7Navbar(title = "f7Picker"),
     f7AutoComplete(
      inputId = "myautocomplete1",
      placeholder = "Some text here!",
      dropdownPlaceholderText = "Try to type Apple",
      label = "Type a fruit name",
      type = "dropdown",
      choices = c('Apple', 'Apricot', 'Avocado', 'Banana', 'Melon',
       'Orange', 'Peach', 'Pear', 'Pineapple')
     ),
     textOutput("autocompleteval1"),
     f7AutoComplete(
      inputId = "myautocomplete2",
      placeholder = "Some text here!",
      type = "popup",
      multiple = TRUE,
      label = "Type a fruit name",
      choices = c('Apple', 'Apricot', 'Avocado', 'Banana', 'Melon',
                  'Orange', 'Peach', 'Pear', 'Pineapple')
     ),
     verbatimTextOutput("autocompleteval2")
    )
   ),
   server = function(input, output) {
    observe({
     print(input$myautocomplete1)
     print(input$myautocomplete2)
    })
    output$autocompleteval1 <- renderText(input$myautocomplete1)
    output$autocompleteval2 <- renderPrint(input$myautocomplete2)
   }
 )
}
```

---

f7Back *Create a framework 7 back button*

Description

This buttons allows to switch between multiple f7Tab.

Usage

```
f7Back(targetId)
```

Arguments

targetId	f7Tabs id.

---

f7Badge *Create a Framework7 badge*

Description

Build a Framework7 badge

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@ymail.com>

Examples

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

 colors <- getF7Colors()

 shiny::shinyApp(
   ui = f7Page(
     title = "Badges",
     f7SingleLayout(
       navbar = f7Navbar(title = "f7Badge"),
       f7Block(
         strong = TRUE,
         lapply(seq_along(colors), function(i) {
```

```
            f7Badge(colors[[i]], color = colors[[i]])
          })
        )
      )
    ),
    server = function(input, output) {}
  )
}
```

---

f7Block *Create a Framework7 block*

Description

Build a Framework7 block

Usage

```
f7Block(..., hairlines = TRUE, strong = FALSE, inset = FALSE, tablet = FALSE)
```

Arguments

...	Block content. Also for f7BlockHeader and f7BlockFooter.
hairlines	Whether to allow hairlines. TRUE by default.
strong	Whether to put the text in bold. 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.

Author(s)

David Granjon, <dgranjon@ymail.com>

Examples

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

 shiny::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,
 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(
 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,
 hairlines = FALSE,
 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) {}
  )
}
```

`f7BlockFooter` *Create a Framework7 block footer*

Description

Build a Framework7 block footer

Usage

```
f7BlockFooter(text = NULL)
```

Arguments

`text`	Footer text.

Author(s)

David Granjon, <dgranjon@ymail.com>

`f7BlockHeader` *Create a Framework7 block header*

Description

Build a Framework7 block header

Usage

```
f7BlockHeader(text = NULL)
```

Arguments

`text`	Header text.

Author(s)

David Granjon, <dgranjon@ymail.com>

f7BlockTitle *Create a Framework7 block title*

Description

Build a Framework7 block title

Usage

```
f7BlockTitle(title = NULL, size = NULL)
```

Arguments

title	Block title.
size	Block title size. NULL by default or "medium", "large".

Author(s)

David Granjon, <dgranjon@ymail.com>

f7Button *Create a Framework7 button*

Description

Build a Framework7 button

Usage

```
f7Button(
  inputId = NULL,
  label = NULL,
  src = NULL,
  color = NULL,
  fill = TRUE,
  outline = FALSE,
  shadow = FALSE,
  rounded = FALSE,
  size = NULL
)
```

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.
src	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".

Author(s)

David Granjon, <dgranjon@ymail.com>

---

f7Card *Create a Framework7 card*

Description

Build a Framework7 card

Usage

```
f7Card(
  ...,
  img = NULL,
  title = NULL,
  footer = NULL,
  outline = FALSE,
  height = NULL
)
```

Arguments

...	Card content.
img	Card image if any. Displayed in the header.
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.

Author(s)

David Granjon, <dgranjon@ymail.com>

Examples

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

 shiny::shinyApp(
  ui = f7Page(
    title = "Cards",
    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",
     "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", src = "https://www.google.com"),
      f7Badge("Badge", color = "green")
     )
    ),
    f7Card(
     title = "Card header",
     img = "https://lorempixel.com/1000/600/nature/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.",
     footer = tagList(
      f7Button(color = "blue", label = "My button", src = "https://www.google.com"),
      f7Badge("Badge", color = "green")
     )
    )
    )
  ),
  server = function(input, output) {}
 )
}
```

---

f7checkBox *Create a F7 Checkbox*

Description

Create a F7 Checkbox

Usage

```
f7checkBox(inputId, label, value = FALSE)
```

Arguments

inputId	The input slot that will be used to access the value.
label	Display label for the control, or NULL for no label.
value	Initial value (TRUE or FALSE).

Examples

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

 shiny::shinyApp(
   ui = f7Page(
    title = "My app",
    f7SingleLayout(
     navbar = f7Navbar(title = "f7checkBox"),
     f7Card(
      f7checkBox(
       inputId = "check",
       label = "Checkbox",
       value = FALSE
      ),
      verbatimTextOutput("test")
     )
    )
   ),
   server = function(input, output) {
    output$test <- renderPrint({input$check})
   }
 )
}
```

---

f7checkBoxGroup *Create an f7 checkbox group input*

Description

Create an f7 checkbox group input

Usage

```
f7checkBoxGroup(inputId, label, choices = NULL, selected = NULL)
```

Arguments

inputId	Checkbox group input.
label	Checkbox group label.
choices	Checkbox group choices.
selected	Checkbox group selected value.

Examples

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

  shiny::shinyApp(
    ui = f7Page(
     title = "My app",
     f7SingleLayout(
      navbar = f7Navbar(title = "f7checkBoxGroup"),
      f7checkBoxGroup(
       inputId = "variable",
       label = "Choose a variable:",
       choices = colnames(mtcars)[-1],
       selected = NULL
      ),
      tableOutput("data")
     )
    ),
    server = function(input, output) {
     output$data <- renderTable({
      mtcars[, c("mpg", input$variable), drop = FALSE]
      }, rownames = TRUE)
    }
  )
 }
```

---

f7Chip *Create a Framework7 chips*

Description

Build a Framework7 chips

Usage

```
f7Chip(
  label = NULL,
  img = NULL,
  icon = NULL,
  outline = FALSE,
```

```
  status = NULL,
  icon_status = NULL,
  closable = FALSE
)
```

Arguments

label	Chip label.
img	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.
icon_status	Chip icon color: see here for valid colors https://framework7.io/docs/chips.html.
closable	Whether to close the chip. FALSE by default.

Note

Not ready for production color and icon isssues.

Author(s)

David Granjon, <dgranjon@ymail.com>

Examples

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

 shiny::shinyApp(
  ui = f7Page(
    title = "Chips",
    init = f7Init(theme = "light", skin = "md"),
    f7SingleLayout(
      navbar = f7Navbar(title = "f7Navbar"),
      f7Block(
        strong = TRUE,
        f7Chip(label = "simple Chip"),
        f7Chip(label = "outline Chip", outline = TRUE),
        f7Chip(label = "icon Chip", icon = f7Icon("add_round"), icon_status = "pink"),
        f7Chip(label = "image Chip", img = "https://lorempixel.com/64/64/people/9/"),
        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) {}
 )
}
```

`f7Col` *Create a Framework7 column container*

Description

Build a Framework7 column container

Usage

```
f7Col(...)
```

Arguments

`...`	Column content. The width is automatically handled depending on the number of columns.

Note

The dark theme does not work for items embedded in a column. Use f7Flex instead.

Author(s)

David Granjon, <dgranjon@ymail.com>

`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
)
```

Arguments

inputId	Color picker input.
label	Color picker label.
value	Color picker value. hex, rgb, hsl, hsb, alpha, hue, rgba, hsla are supported.
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 <input> 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 <input> element to edit directly.
groupedModules	When enabled it will add more exposure to sliders modules to make them look more separated.

Examples

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

 shinyApp(
   ui = f7Page(
     title = "My app",
     f7SingleLayout(
       navbar = f7Navbar(title = "f7ColorPicker"),
       f7ColorPicker(
         inputId = "mycolorpicker",
         placeholder = "Some text here!",
         label = "Select a color"
       ),
       "The picker value is:",
       textOutput("colorPickerVal")
     )
   ),
```

```
  server = function(input, output) {
    output$colorPickerVal <- renderText(input$mycolorpicker)
  }
 )
}
```

`f7DatePicker` *Create a Framework7 date input*

Description

Create a Framework7 date input

Usage

```
f7DatePicker(
  inputId,
  label,
  value = NULL,
  min = NULL,
  max = NULL,
  format = "yyyy-mm-dd"
)
```

Arguments

`inputId`	Date input id.
`label`	Input label.
`value`	Start value.
`min`	Minimum date.
`max`	Maximum date.
`format`	Date format: "yyyy-mm-dd", for instance.

Examples

```
if (interactive()) {
 library(shiny)
 library(shinyMobile)
 shinyApp(
   ui = f7Page(
     preloader = FALSE,
     color = "pink",
     title = "My app",
     f7SingleLayout(
       navbar = f7Navbar(title = "f7DatePicker"),
       f7DatePicker(
         inputId = "date",
```

```
          label = "Choose a date",
          value = "2019-08-24"
        ),
        "The selected date is",
        textOutput("selectDate")
      )
    ),
    server = function(input, output, session) {
      output$selectDate <- renderText(input$date)
    }
  )
}
```

---

f7Dialog *Create a Framework7 dialog window*

Description

Create a Framework7 dialog window

Usage

```
f7Dialog(
  inputId = NULL,
  title = NULL,
  text,
  type = c("alert", "confirm", "prompt", "login"),
  session
)
```

Arguments

inputId	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

```
# simple alert
if (interactive()) {
  library(shiny)
  library(shinyMobile)
  shinyApp(
    ui = f7Page(
      title = "My App",
```

```
      f7SingleLayout(
        navbar = f7Navbar(title = "f7Dialog"),
        f7Button(inputId = "goButton", "Go!")
      )
    ),
    server = function(input, output, session) {
      shiny::observeEvent(input$goButton,{
        f7Dialog(
         title = "Dialog title",
         text = "This is an alert dialog",
         session = session
        )
      })
    }
  )
}
# confirm alert
if (interactive()) {
 library(shiny)
 library(shinyMobile)
 shinyApp(
   ui = f7Page(
     title = "My App",
     f7SingleLayout(
       navbar = f7Navbar(title = "f7Dialog"),
       f7Button(inputId = "goButton", "Go!")
     )
   ),
   server = function(input, output, session) {

     observeEvent(input$goButton,{
       f7Dialog(
         inputId = "test",
         title = "Dialog title",
         type = "confirm",
         text = "This is an alert dialog",
         session = session
       )
     })

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

   }
 )
}
# prompt dialog
if (interactive()) {
 library(shiny)
 library(shinyMobile)
 shinyApp(
   ui = f7Page(
```

```
      title = "My App",
      f7SingleLayout(
        navbar = f7Navbar(title = "f7Dialog"),
        f7Button(inputId = "goButton", "Go!"),
        uiOutput("res")
      )
    ),
    server = function(input, output, session) {

      observe({
        print(input$prompt)
      })

      observeEvent(input$goButton,{
        f7Dialog(
          inputId = "prompt",
          title = "Dialog title",
          type = "prompt",
          text = "This is a prompt dialog",
          session = session
        )
      })

      output$res <- renderUI(f7BlockTitle(title = input$prompt, size = "large"))
    }
  )
}

# login dialog
if (interactive()) {
  library(shiny)
  library(shinyMobile)
  shinyApp(
    ui = f7Page(
      title = "My App",
      f7SingleLayout(
        navbar = f7Navbar(title = "f7Dialog"),
        f7Button(inputId = "goButton", "Go!"),
        uiOutput("ui")
      )
    ),
    server = function(input, output, session) {

      observe({
        print(input$login)
      })

      observeEvent(input$goButton,{
        f7Dialog(
          inputId = "login",
          title = "Dialog title",
          type = "login",
          text = "This is an login dialog",
```

```
        session = session
      )
    })

    output$ui <- renderUI({
      req(input$login$user == "David" & input$login$password == "prout")
      img(src = "https://media2.giphy.com/media/12gfL8Xxrhv7C1fXiV/giphy.gif")
    })
  }
 )
}
```

`f7ExpandableCard` *Create a Framework7 expandable card*

Description

Build a Framework7 expandable card

Usage

```
f7ExpandableCard(
  ...,
  id = NULL,
  title = NULL,
  subtitle = NULL,
  color = NULL,
  img = NULL,
  fullBackground = FALSE
)
```

Arguments

`...`	Card content.
`id`	Unique card id. Useful to handle multiple cards in the DOM.
`title`	Card title.
`subtitle`	Card subtitle.
`color`	Card background color. See http://framework7.io/docs/cards.html. Not compatible with the img argument.
`img`	Card background image url. Tje JPG format is prefered. Not compatible with the color argument.
`fullBackground`	Whether the image should cover the entire card.

Note

img and color are not compatible. Choose one of them.

Author(s)

David Granjon, <dgranjon@ymail.com>

Examples

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

 shiny::shinyApp(
  ui = f7Page(
    title = "Expandable Cards",
    f7SingleLayout(
      navbar = f7Navbar(
       title = "Expandable Cards",
       hairline = FALSE,
       shadow = TRUE
      ),
      f7ExpandableCard(
       id = "card1",
       title = "Expandable Card 1",
       color = "blue",
       subtitle = "Click on me pleaaaaase",
       "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."
      ),
      f7ExpandableCard(
       id = "card2",
       title = "Expandable Card 2",
       color = "green",
       "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."
      ),
      f7ExpandableCard(
       id = "card3",
       title = "Expandable Card 3",
     img = "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."
      ),
      f7ExpandableCard(
       id = "card4",
       title = "Expandable Card 4",
       fullBackground = TRUE,
       img = "https://i.ytimg.com/vi/8q_kmxwK5Rg/maxresdefault.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."
      )
    )
  ),
  server = function(input, output) {}
 )
}
```

---

f7Fab *Create a Framework7 floating action button (FAB)*

Description

Build a Framework7 floating action button (FAB)

Usage

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

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-...

Author(s)

David Granjon, <dgranjon@ymail.com>

f7FabClose *Indicate that the current tag should close the f7Fabs*

Description

Indicate that the current tag should close the f7Fabs

Usage

```
f7FabClose(tag)
```

Arguments

tag	Target tag.

f7FabMorphTarget *Convert a tag into a target morphing*

Description

Convert a tag into a target morphing

Usage

```
f7FabMorphTarget(tag)
```

Arguments

tag	Target tag.

Examples

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

 shiny::shinyApp(
   ui = f7Page(
     f7SingleLayout(
       navbar = f7Navbar(title = "f7Fabs Morph"),
       toolbar = f7Toolbar(
         position = "bottom",
         lapply(1:3, function(i) f7Link(i) %>% f7FabClose())
       ) %>% f7FabMorphTarget(),
       # put an empty f7Fabs container
       f7Fabs(
         extended = TRUE,
```

```
          label = "Open",
          position = "center-top",
          color = "yellow",
          sideOpen = "right",
          morph = TRUE,
          morphTarget = ".toolbar"
        )
      )

    ),
    server = function(input, output) {}
 )
}
```

---

f7Fabs *Create a Framework7 container for floating action button (FAB)*

Description

Build a Framework7 container for floating action button (FAB)

Usage

```
f7Fabs(
  ...,
 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 = FALSE,
  morphTarget = NULL
)
```

Arguments

...	Slot for f7Fab.
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	Whether to allow the FAB to transofrm into another UI element.
morphTarget	CSS selector of the morph target: ".toolbar" for instance.

Note

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

Author(s)

David Granjon, <dgranjon@ymail.com>

Examples

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

 shiny::shinyApp(
  ui = f7Page(
    color = "pink",
    title = "Floating action buttons",
    f7SingleLayout(
     navbar = f7Navbar(title = "f7Fabs"),
     f7Fabs(
      extended = TRUE,
      label = "Menu",
      position = "center-top",
      color = "yellow",
      sideOpen = "right",
      lapply(1:4, function(i) f7Fab(paste0("btn", i), i))
    ),
    lapply(1:4, function(i) verbatimTextOutput(paste0("res", i))),

    f7Fabs(
      position = "center-center",
      color = "purple",
      sideOpen = "center",
      lapply(5:8, function(i) f7Fab(paste0("btn", i), i))
    ),
    lapply(5:8, function(i) verbatimTextOutput(paste0("res", i))),

    f7Fabs(
      position = "left-bottom",
      color = "pink",
      sideOpen = "top",
      lapply(9:12, function(i) f7Fab(paste0("btn", i), i))
    )
    )

  ),
  server = function(input, output) {
    lapply(1:12, function(i) {
      output[[paste0("res", i)]] <- renderPrint(input[[paste0("btn", i)]])
    })
  }
```

```
 )
}
```

---

f7Flex *Create a Framework7 flex container*

Description

Build a Framework7 flex container

Usage

```
f7Flex(...)
```

Arguments

...	Items.

Author(s)

David Granjon, <dgranjon@ymail.com>

Examples

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

 shiny::shinyApp(
   ui = f7Page(
    title = "Align",
    f7SingleLayout(
     navbar = f7Navbar(title = "f7Flex"),
     f7Flex(
     f7Block(strong = TRUE),
     f7Block(strong = TRUE),
     f7Block(strong = TRUE)
    )
    )
   ),
   server = function(input, output) {}
 )
}
```

f7Float *Create a Framework7 float*

Description

Build a Framework7 float

Usage

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

Arguments

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

Author(s)

David Granjon, <dgranjon@ymail.com>

Examples

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

 shiny::shinyApp(
   ui = f7Page(
    title = "Float",
    f7SingleLayout(
     navbar = f7Navbar(title = "f7Float"),
     f7Float(h1("Left"), side = "left"),
    f7Float(h1("Right"), side = "right")
    )
   ),
   server = function(input, output) {}
 )
}
```

f7Found *Utility to display an item when the search is successful.*

Description

Use with f7Searchbar.

Usage

```
f7Found(tag)
```

Arguments

tag	tag to display. When using f7Searchbar, one must wrap the items to search in inside f7Found.

f7Gallery *Launch the shinyMobile Gallery*

Description

A gallery of all components available in shinyMobile.

Usage

```
f7Gallery()
```

Examples

```
if (interactive()) {

 f7Gallery()

}
```

f7Gauge *Create a Framework7 gauge*

Description

Build a Framework7 gauge

Usage

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

Arguments

id	Gauge ID.
type	Gauge type. Can be "circle" or "semicircle". Default is "circle."
value	Gauge value/percentage. Must be a number between 0 and 100.
valueText	Gauge value text (large text in the center of gauge).
size	Generated SVG image size (in px). Default is 200.
bgColor	Gauge background color. Can be any valid color string, e.g. #ff00ff, rgb(0,0,255), etc. Default is "transparent".
borderBgColor	Main border/stroke background color.
borderColor	Main border/stroke color.
borderWidth	Main border/stroke width.
valueTextColor	Value text color.
valueFontSize	Value text font size.
valueFontWeight	Value text font weight.

labelText	Gauge additional label text.
labelTextColor	Label text color.
labelFontSize	Label text font size.
labelFontWeight	Label text font weight.

Author(s)

David Granjon and Isabelle Rudolf, <dgranjon@ymail.com>

Examples

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

 shiny::shinyApp(
  ui = f7Page(
    title = "Gauges",
    f7SingleLayout(
     navbar = f7Navbar(title = "f7Gauge"),
     f7Block(
      f7Gauge(
       id = "mygauge",
       type  = "semicircle",
       value = 50,
       borderColor = "#2196f3",
       borderWidth = 10,
       valueText = "50%",
       valueFontSize = 41,
       valueTextColor = "#2196f3",
       labelText = "amount of something"
      )
     )
    )
  ),
  server = function(input, output) {}
 )
}
```

---

f7HideOnEnable *Utility to hide a given tag when f7Searchbar is enabled.*

Description

Use with f7Searchbar.

Usage

```
f7HideOnEnable(tag)
```

Arguments

tag	tag to hide.

---

f7HideOnSearch *Utility to hide a given tag on search*

Description

Use with f7Searchbar.

Usage

```
f7HideOnSearch(tag)
```

Arguments

tag	tag to hide.

---

f7Icon *Create a Framework7 icon*

Description

Build a Framework7 icon

Usage

```
f7Icon(..., lib = NULL, fill = FALSE)
```

Arguments

...	Icon name and f7Badge.
lib	Library to use: NULL, "ios" or "md".
fill	Whether to fill or not. FALSE by default.

Author(s)

David Granjon, <dgranjon@ymail.com>

Examples

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

 shiny::shinyApp(
  ui = f7Page(
    title = "Icons",
    f7ListCard(
     f7ListCardItem(
       title = tagList(
        f7Icon("email_fill", lib = "ios"),
        "This does not appear for material design devices."
       )
     ),
     f7ListCardItem(
       title = f7Icon("home", f7Badge("1", color = "red"))
     ),
     f7ListCardItem(
       title = f7Icon("email", lib = "md")
     )
    )
  ),
  server = function(input, output) {}
 )
}
```

---

f7Init *Custom initialization*

Description

Use inside f7Page. Mandatory!

Usage

```
f7Init(
  skin = c("ios", "md", "auto", "aurora"),
  theme = c("dark", "light"),
  filled = FALSE,
  color = NULL,
  tapHold = TRUE,
  iosTouchRipple = FALSE,
  iosCenterTitle = TRUE,
  iosTranslucentBars = FALSE,
  hideNavOnPageScroll = TRUE,
  hideTabsOnPageScroll = FALSE,
  serviceWorker = NULL
)
```

Arguments

skin	App skin: "ios", "md", "auto" or "aurora".
theme	App theme: "light" or "dark".
filled	Whether to fill the f7Navbar and f7Toolbar with the current selected color. FALSE by default.
color	Color theme: See http://framework7.io/docs/color-themes.html. Expect a name like blue or red. If NULL, use the default color.
tapHold	It triggers (if enabled) after a sustained, complete touch event. By default it is disabled. Note, that Tap Hold is a part of built-in Fast Clicks library, so Fast Clicks should be also enabled.
iosTouchRipple	Default to FALSE. Enables touch ripple effect for iOS theme.
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.
iosTranslucentBars	Enable translucent effect (blur background) on navigation bars for iOS theme (on iOS devices). FALSE by default.
hideNavOnPageScroll	Default to TRUE. Will hide Navbars on page scroll.
hideTabsOnPageScroll	Default to FALSE. Will hide tabs on page scroll.
serviceWorker	Object with service worker module parameters. (Use for PWA).

Author(s)

David Granjon, <dgranjon@ymail.com>

---

f7InsertTab *Insert a f7Tab in a f7Tabs*

Description

Insert a f7Tab in a f7Tabs

Usage

```
f7InsertTab(
  inputId,
  tab,
  target,
  position = c("before", "after"),
  select = FALSE,
  session = shiny::getDefaultReactiveDomain()
)
```

Arguments

inputId	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.

Examples

```
if (interactive()) {
 library(shiny)
 library(shinyMobile)
 shiny::shinyApp(
   ui = f7Page(
     title = "Insert a tab Before the target",
     f7TabLayout(
       panels = tagList(
      f7Panel(title = "Left Panel", side = "left", theme = "light", "Blabla", effect = "cover"),
      f7Panel(title = "Right Panel", side = "right", theme = "dark", "Blabla", effect = "cover")
       ),
       navbar = f7Navbar(
         title = "Tabs",
         hairline = FALSE,
         shadow = TRUE,
         left_panel = TRUE,
         right_panel = TRUE
       ),
       f7Tabs(
         animated = TRUE,
         id = "tabs",
         f7Tab(
           tabName = "Tab 1",
           icon = f7Icon("email"),
           active = TRUE,
           "Tab 1",
           f7Button(inputId = "go", label = "Go")
         ),
         f7Tab(
           tabName = "Tab 2",
           icon = f7Icon("today"),
           active = FALSE,
           "Tab 2"
         )
       )
     )
   ),
   server = function(input, output, session) {
     observeEvent(input$go, {
       f7InsertTab(
```

```
            inputId = "tabs",
            position = "before",
            target = "Tab 2",
            tab = f7Tab (tabName = paste0("tab_", input$go), "Test"),
            select = TRUE
          )
        })
      }
    )
  }
```

`f7Item` *Create a Framework7 f7Item.*

Description

Similar to f7Tab but for the f7SplitLayout.

Usage

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

Arguments

`...`	Item content.
`tabName`	Item id. Must be unique.

Author(s)

David Granjon, <dgranjon@ymail.com>

`f7Items` *Create a Framework7 wrapper for f7Item*

Description

Build a Framework7 wrapper for f7Item

Usage

```
f7Items(...)
```

Arguments

`...`	Slot for wrapper for f7Item.

Author(s)

David Granjon, <dgranjon@ymail.com>

`f7Link` *Create a Framework7 link*

Description

Build a Framework7 link

Usage

```
f7Link(label = NULL, icon = NULL, src = NULL, external = FALSE)
```

Arguments

`label`	Link text.
`icon`	Link icon, if any.
`src`	Link source, url.
`external`	Whether switch to an external link. FALSE by default.

Author(s)

David Granjon, <dgranjon@ymail.com>

Examples

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

 shiny::shinyApp(
   ui = f7Page(
    title = "Links",
    f7SingleLayout(
     navbar = f7Navbar(title = "f7Link"),
     f7Link(label = "Google", src = "https://www.google.com"),
     f7Link(label = "Google", src = "https://www.google.com", external = TRUE)
    )
   ),
   server = function(input, output) {}
 )
}
```

f7List *Create a framework 7 contact list*

Description

Create a framework 7 contact list

Usage

```
f7List(..., mode = NULL, inset = FALSE)
```

Arguments

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

Examples

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

shiny::shinyApp(
  ui = f7Page(
    title = "My app",
    f7SingleLayout(
      navbar = f7Navbar(title = "f7List"),

      # simple list
      f7List(
        lapply(1:3, function(j) f7ListItem(letters[j]))
      ),

      # list with complex items
      f7List(
        lapply(1:3, function(j) {
          f7ListItem(
            letters[j],
            media = f7Icon("alarm_fill"),
            right = "Right Text",
            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(
          lapply(1:3, function(j) {
            f7ListItem(url = "https://google.com", letters[j])
          })
        ),

        # grouped lists
        f7List(
          mode = "contacts",
          lapply(1:3, function(i) {
            f7ListGroup(
              title = LETTERS[i],
              lapply(1:3, function(j) f7ListItem(letters[j]))
            )
          })
        )
      )
    ),
    server = function(input, output) {}
  )
}
```

`f7ListGroup` *Create a framework 7 group of contacts*

Description

Create a framework 7 group of contacts

Usage

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

Arguments

...	slot for f7ListItem.
title	Group title.

---

f7ListIndex *Create a Framework 7 list index*

Description

Create a Framework 7 list index

Usage

```
f7ListIndex(..., id)
```

Arguments

...	Slot for f7ListGroup.
id	Unique id.

Note

For some reason, unable to get more than 1 list index working. See example below. The second list does not work.

Examples

```
if (interactive()) {
 library(shiny)
 library(shinyMobile)
 shiny::shinyApp(
   ui = f7Page(
     title = "My app",
     f7TabLayout(
       navbar = f7Navbar(
         title = "f7ListIndex",
         hairline = FALSE,
         shadow = TRUE
       ),
       f7Tabs(
         f7Tab(
           tabName = "List 1",
           f7ListIndex(
             id = "listIndex1",
```

```
              lapply(seq_along(LETTERS), function(i) {
                f7ListGroup(
                  title = LETTERS[i],
                  lapply(1:3, function(j) {
                    f7ListIndexItem(letters[j])
                  })
                )
              })
            )
          ),
          f7Tab(
            tabName = "List 2",
            f7ListIndex(
              id = "listIndex2",
              lapply(seq_along(LETTERS), function(i) {
                f7ListGroup(
                  title = LETTERS[i],
                  lapply(1:3, function(j) {
                    f7ListIndexItem(letters[j])
                  })
                )
              })
            )
          )
        )
      )
    ),
    server = function(input, output) {}
  )
}
```

`f7ListIndexItem` *Create a Framework 7 list index item*

Description

Create a Framework 7 list index item

Usage

```
f7ListIndexItem(..., .noWS = NULL)
```

Arguments

`...`	Attributes and children of the element. Named arguments become attributes, and positional arguments become children. Valid children are tags, single-character character vectors (which become text nodes), raw HTML (see `HTML`), and `html_dependency` objects. You can also pass lists that contain tags, text nodes, or HTML. To use boolean attributes, use a named argument with a `NA` value. (see example)

.noWS	A character vector used to omit some of the whitespace that would normally be written around this tag. Valid options include before, after, outside, after-begin, and before-end. Any number of these options can be specified.

---

f7ListItem *Create a Framework 7 contact item*

Description

Create a Framework 7 contact item

Usage

```
f7ListItem(
  ...,
  title = NULL,
  subtitle = NULL,
  header = NULL,
  footer = NULL,
  url = NULL,
  media = NULL,
  right = NULL
)
```

Arguments

...	Item text.
title	Item title.
subtitle	Item subtitle.
header	Item header. Do not use when f7List mode is not NULL.
footer	Item footer. Do not use when f7List mode is not NULL.
url	Item url.
media	Expect f7Icon or img.
right	Right content if any.

f7Margin *Create a Framework7 margin*

Description

Build a Framework7 margin

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", src = "https://www.google.com"),
    f7Badge("Badge", color = "green")
  )
 )

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

```
}
```

`f7Message` *Create a framework7 message*

Description

Create a framework7 message

Usage

```
f7Message(
  content,
  src = NULL,
  author = NULL,
  date = NULL,
  state = c("sent", "received"),
  type = c("text", "img")
)
```

Arguments

`content`	Message content. If type is ing, content must be an url or path.
`src`	Message author avatar
`author`	Message author
`date`	Message date
`state`	Whether it is a received or sent message.
`type`	Whether it is a text or image.

`f7Messages` *Create a Framework7 messages container*

Description

Create a Framework7 messages container

Usage

```
f7Messages(..., id)
```

Arguments

`...`	Slot for f7Message.
`id`	Container id.

Examples

```
if (interactive()) {
 library(shiny)
 library(shinyMobile)
 shinyApp(
   ui = f7Page(
     preloader = FALSE,
     title = "My app",
     f7SingleLayout(
       navbar = f7Navbar(title = "f7Messages"),
       f7Messages(
         id = "messagelist",
         f7Message(
           "Lorem ipsum dolor sit amet,
           consectetur adipiscing elit.
           Duo Reges: constructio interrete",
           src = "https://cdn.framework7.io/placeholder/people-100x100-7.jpg",
           author = "David",
           date = "2019-09-12",
           state = "received",
           type = "text"
         ),
         f7Message(
           "https://cdn.framework7.io/placeholder/cats-200x260-4.jpg",
           src = "https://cdn.framework7.io/placeholder/people-100x100-9.jpg",
           author = "Lia",
           date = NULL,
           state = "sent",
           type = "img"
         ),
         f7Message(
           "Hi Bro",
           src = "https://cdn.framework7.io/placeholder/people-100x100-9.jpg",
           author = NULL,
           date = "2019-08-15",
           state = "sent",
           type = "text"
         )
       )
     )
   ),
   server = function(input, output, session) {
   }
 )
}
```

---

f7Navbar *Create a Framework7 Navbar*

Description

Build a Framework7 Navbar

Usage

```
f7Navbar(
  ...,
  subNavbar = NULL,
  title = NULL,
  subtitle = NULL,
  hairline = TRUE,
  shadow = TRUE,
  bigger = FALSE,
  transparent = FALSE,
  left_panel = FALSE,
  right_panel = FALSE
)
```

Arguments

...	Slot for f7SearchbarTrigger. Not compatible with f7Panel.
subNavbar	f7SubNavbar slot, if any.
title	Navbar title.
subtitle	Navbar subtitle. Not compatible with bigger.
hairline	Whether to display a thin border on the top of the navbar. TRUE by default.
shadow	Whether to display a shadow. TRUE by default.
bigger	Whether to display bigger title. FALSE by default. Not compatible with subtitle.
transparent	Whether the navbar should be transparent. FALSE by default. Only works if bigger is TRUE.
left_panel	Whether to enable the left panel. FALSE by default.
right_panel	Whether to enable the right panel. FALSE by default.

Note

Currently, bigger parameters does mess with the CSS.

Author(s)

David Granjon, <dgranjon@ymail.com>

f7NavbarHide *Hide a framework7 navbar*

Description

Hide a framework7 navbar

Usage

```
f7NavbarHide(
  session = shiny::getDefaultReactiveDomain(),
  animate = TRUE,
  hideStatusbar = FALSE
)
```

Arguments

session	Shiny session object.
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.

Examples

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

 shiny::shinyApp(
    ui = f7Page(
       title = "Accordions",
       f7SingleLayout(
          navbar = f7Navbar("Hide/Show navbar"),
          f7Segment(
             f7Button(inputId = "hide", "Hide navbar", color = "red"),
             f7Button(inputId = "show", "Show navbar", color = "green"),
          )
       )
    ),
    server = function(input, output, session) {

       observeEvent(input$hide, {
          f7NavbarHide()
       })

       observeEvent(input$show, {
          f7NavbarShow()
       })
    }
```

```
 )
}
```

---

f7NavbarShow *Show a framework7 navbar*

Description

Show a framework7 navbar

Usage

```
f7NavbarShow(session = shiny::getDefaultReactiveDomain(), animate = TRUE)
```

Arguments

session	Shiny session object.
animate	Whether it should be hidden with animation or not. By default is TRUE.

Examples

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

 shiny::shinyApp(
    ui = f7Page(
       title = "Accordions",
       f7SingleLayout(
          navbar = f7Navbar("Hide/Show navbar"),
          f7Segment(
             f7Button(inputId = "hide", "Hide navbar", color = "red"),
             f7Button(inputId = "show", "Show navbar", color = "green"),
          )
       )
    ),
    server = function(input, output, session) {

       observeEvent(input$hide, {
          f7NavbarHide()
       })

       observeEvent(input$show, {
          f7NavbarShow()
       })
    }
 )
}
```

f7Next *Create a framework 7 next button*

Description

This buttons allows to switch between multiple f7Tab.

Usage

```
f7Next(targetId)
```

Arguments

targetId	f7Tabs id.

f7NotFound *Utility to display an item when the search is unsuccessful.*

Description

Use with f7Searchbar.

Usage

```
f7NotFound(tag)
```

Arguments

tag	tag to use.

f7Notif *Create a Framework7 notification*

Description

Create a Framework7 notification

Usage

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

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.
session	shiny session.

Examples

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

```
            subtitle = "A subtitle",
            titleRightText = "now",
            session = session
          )
        })
      }
    )
  }
```

---

f7Padding *Create a Framework7 padding*

Description

Build a Framework7 padding

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", src = "https://www.google.com"),
    f7Badge("Badge", color = "green")
  )
 )

 shiny::shinyApp(
   ui = f7Page(
```

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

---

f7Page *Create a Framework7 page*

Description

Build a Framework7 page

Usage

```
f7Page(
  ...,
  init = f7Init(skin = "auto", theme = "light"),
  title = NULL,
  preloader = FALSE,
  loading_duration = 3
)
```

Arguments

`...`	Slot for shinyMobile skeleton elements: f7Appbar, f7SingleLayout, f7TabLayout, f7SplitLayout.
`init`	App configuration. See f7Init.
`title`	Page title.
`preloader`	Whether to display a preloader before the app starts. FALSE by default.
`loading_duration`	Preloader duration.

Author(s)

David Granjon, <dgranjon@ymail.com>

`f7Panel` *Create a Framework7 panel*

Description

Build a Framework7 panel

Usage

```
f7Panel(
  ...,
  inputId = NULL,
  title = NULL,
  side = c("left", "right"),
  theme = c("dark", "light"),
  effect = c("reveal", "cover"),
  resizable = FALSE
)
```

Arguments

`...`	Panel content. Slot for f7PanelMenu, if used as a sidebar.
`inputId`	Panel unique id. This is to access the input$id giving the panel state, namely open or closed.
`title`	Panel title.
`side`	Panel side: "left" or "right".
`theme`	Panel background color: "dark" or "light".
`effect`	Whether the panel should behave when opened: "cover" or "reveal".
`resizable`	Whether to enable panel resize. FALSE by default.

Author(s)

David Granjon, <dgranjon@ymail.com>

Examples

```
if (interactive()) {
 library(shiny)
 library(shinyMobile)
 shiny::shinyApp(
   ui = f7Page(
     title = "My app",
     init = f7Init(skin = "ios", theme = "light"),
     f7SingleLayout(
       navbar = f7Navbar(
         title = "Single Layout",
```

```
      hairline = FALSE,
      shadow = TRUE,
      left_panel = TRUE,
      right_panel = TRUE
    ),
    panels = tagList(
      f7Panel(side = "left", inputId = "mypanel1"),
      f7Panel(side = "right", inputId = "mypanel2")
    ),
    toolbar = f7Toolbar(
      position = "bottom",
      icons = TRUE,
      hairline = FALSE,
      shadow = FALSE,
      f7Link(label = "Link 1", src = "https://www.google.com"),
      f7Link(label = "Link 2", src = "https://www.google.com", external = TRUE)
    ),
    # main content
    f7Shadow(
      intensity = 10,
      hover = TRUE,
      f7Card(
        title = "Card header",
        sliderInput("obs", "Number of observations", 0, 1000, 500),
        h1("You only see me by opening the left panel"),
        plotOutput("distPlot"),
        footer = tagList(
        f7Button(color = "blue", label = "My button", src = "https://www.google.com"),
          f7Badge("Badge", color = "green")
        )
      )
    )
  )
),
server = function(input, output, session) {

  observeEvent(input$mypanel2, {

    state <- if (input$mypanel2) "open" else "closed"

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

  output$distPlot <- renderPlot({
    if (input$mypanel1) {
      dist <- rnorm(input$obs)
      hist(dist)
```

```
        }
      })
    }
  )
}
```

`f7PanelItem` *Create a Framework7 sidebar menu item*

Description

Build a Framework7 sidebar menu item for f7SplitLayout.

Usage

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

Arguments

`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@ymail.com>

`f7PanelMenu` *Create a Framework7 sidebar menu*

Description

Build a Framework7 sidebar menu

Usage

```
f7PanelMenu(..., id = NULL)
```

Arguments

`...`	Slot for f7PanelItem.
`id`	Unique id to access the currently selected item.

Author(s)

David Granjon, <dgranjon@ymail.com>

`f7Password` *Create an f7 password input*

Description

Create an f7 password input

Usage

```
f7Password(inputId, label, value = "", placeholder = NULL)
```

Arguments

`inputId`	Text input id.
`label`	Text input label.
`value`	Text input value.
`placeholder`	Text input placeholder.

Examples

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

 shiny::shinyApp(
   ui = f7Page(
     title = "My app",
     f7SingleLayout(
      navbar = f7Navbar(title = "f7Password"),
      f7Password(
       inputId = "password",
       label = "Password:",
       placeholder = "Your password here"
      ),
      verbatimTextOutput("value")
     )
   ),
   server = function(input, output) {
     output$value <- renderPrint({ input$password })
   }
 )
}
```

`f7PhotoBrowser` *Create a fremework7 photo browser*

Description

Create a fremework7 photo browser

Usage

```
f7PhotoBrowser(
  id,
  label,
  photos,
  theme = c("light", "dark"),
  type = c("popup", "standalone", "page")
)
```

Arguments

`id`	Unique id.
`label`	Trigger label
`photos`	List of photos
`theme`	Browser theme: choose either light or dark.
`type`	Browser type: choose among `c("popup","standalone","page")`.

Examples

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

 shiny::shinyApp(
   ui = f7Page(
     title = "f7PhotoBrowser",
     f7SingleLayout(
       navbar = f7Navbar(title = "f7PhotoBrowser"),
       f7PhotoBrowser(
         id = "photobrowser1",
         label = "Open",
         theme = "light",
         type = "standalone",
         photos = 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"
         )
       )
     )
   )
```

```
    ),
    server = function(input, output, session) {}
  )
}
```

---

f7Picker *Create a Framework7 picker input*

Description

Build a Framework7 picker input

Usage

```
f7Picker(inputId, label, placeholder = NULL, value = choices[1], choices)
```

Arguments

inputId	Picker input id.
label	Picker label.
placeholder	Text to write in the container.
value	Picker initial value, if any.
choices	Picker choices.

Author(s)

David Granjon, <dgranjon@ymail.com>

Examples

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

 shinyApp(
   ui = f7Page(
    title = "My app",
    f7SingleLayout(
     navbar = f7Navbar(title = "f7Picker"),
     f7Picker(
      inputId = "mypicker",
      placeholder = "Some text here!",
      label = "Picker Input",
      choices = c('a', 'b', 'c')
     ),
     textOutput("pickerval")
    )
   ),
```

```
  server = function(input, output) {
   output$pickerval <- renderText(input$mypicker)
  }
 )
}
```

---

f7Popover *Create a framework 7 popover*

Description

f7Popover has to be used in an oberve or observeEvent context. Only works for input elements!

Usage

```
f7Popover(targetId, content, session)
```

Arguments

targetId	Target to put the popover on.
content	Popover content.
session	shiny session.

Examples

```
if (interactive()) {
 library(shiny)
 library(shinyMobile)
 shinyApp(
   ui = f7Page(
     title = "f7Popover",
     f7SingleLayout(
       navbar = f7Navbar(title = "f7Popover"),
       f7PopoverTarget(
         f7Button(
           inputId = "goButton",
           "Go!"
         ),
         targetId = "test"
       ),
       br(),
       br(),
       f7PopoverTarget(
         f7Slider(
           inputId = "slider",
           label = "Value",
           value = 10,
           min = 0,
```

```
            max = 20
          ),
          targetId = "test2"
        )
      )
    ),
    server = function(input, output, session) {
      observe({
        f7Popover(
          targetId = "test",
          content = "This is a f7Button",
          session
        )
      })

      observe({
        f7Popover(
          targetId = "test2",
          content = "This is a f7Slider",
          session
        )
      })
    }
  )
}
```

`f7PopoverTarget` *Define a popover target*

Description

This must be used in combination of f7Popover. Only works for input elements!

Usage

```
f7PopoverTarget(tag, targetId)
```

Arguments

`tag`	Tag that will be targeted. Must be a f7Input element.
`targetId`	Popover id. Must correspond to the f7PopovertargetId.

f7Popup *Create a f7 popup*

Description

Create a f7 popup

Usage

```
f7Popup(..., id, label = "Open", title)
```

Arguments

...	Content.
id	Popup unique id.
label	Popup trigger label.
title	Title.

Examples

```
if (interactive()) {
 library(shiny)
 library(shinyMobile)
 shiny::shinyApp(
   ui = f7Page(
     color = "pink",
     title = "My app",
     f7SingleLayout(
      navbar = f7Navbar(
        title = "f7Popup",
        hairline = FALSE,
        shadow = TRUE
      ),
      f7Popup(
       id = "popup1",
       label = "Open",
       title = "My first popup",
       "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"
      )
     )
   ),
   server = function(input, output) {}
 )
}
```

`f7Progress` *Create a Framework7 progress bar*

Description

Build a Framework7 progress bar

Usage

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

Arguments

`id`	Progress id. Must be unique.
`value`	Progress value. Between 0 and 100.
`color`	Progress color. See http://framework7.io/docs/progressbar.html.

Author(s)

David Granjon, <dgranjon@ymail.com>

Examples

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

 shiny::shinyApp(
   ui = f7Page(
    title = "Progress",
    f7SingleLayout(
     navbar = f7Navbar(title = "f7Progress"),
     f7Block(
     f7Progress(id = "pg1", value = 10, color = "pink"),
     f7Progress(id = "pg2", value = 100, color = "green"),
     f7Progress(id = "pg3", value = 50, color = "orange")
    )
    )
   ),
   server = function(input, output) {}
 )
}
```

f7ProgressInf *Create a Framework7 infinite progress bar*

Description

Build a Framework7 infinite progress bar

Usage

```
f7ProgressInf(color = NULL)
```

Arguments

color	Progress color. See http://framework7.io/docs/progressbar.html.

Note

Buggy display when status is not NULL.

Author(s)

David Granjon, <dgranjon@ymail.com>

Examples

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

 shiny::shinyApp(
   ui = f7Page(
    title = "Progress Infinite",
    f7SingleLayout(
     navbar = f7Navbar(title = "f7ProgressInf"),
     f7Block(
     f7ProgressInf(),
     f7ProgressInf(color = "yellow")
    )
    )
   ),
   server = function(input, output) {}
 )
}
```

f7Radio *Create an f7 radio button input*

Description

Create an f7 radio button input

Usage

```
f7Radio(inputId, label, choices = NULL, selected = NULL)
```

Arguments

inputId	Radio input id.
label	Radio label
choices	List of choices.
selected	Selected element. NULL by default.

Examples

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

 shiny::shinyApp(
   ui = f7Page(
    title = "My app",
    f7SingleLayout(
     navbar = f7Navbar(title = "f7Radio"),
     f7Radio(
      inputId = "radio",
      label = "Choose a fruit:",
      choices = c("banana", "apple", "peach"),
      selected = "apple"
     ),
     plotOutput("plot")
    )
   ),
   server = function(input, output) {
    output$plot <- renderPlot({
     if (input$radio == "apple") hist(mtcars[, "mpg"])
    })
   }
 )
}
```

f7RemoveTab *Remove a f7Tab in a f7Tabs*

Description

Remove a f7Tab in a f7Tabs

Usage

```
f7RemoveTab(inputId, target, session = shiny::getDefaultReactiveDomain())
```

Arguments

inputId	f7Tabs id.
target	f7Tab to remove.
session	Shiny session object.

Examples

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

 ui <- f7Page(
  title = "Remove a tab",
  f7TabLayout(
   panels = tagList(
    f7Panel(title = "Left Panel", side = "left", theme = "light", "Blabla", effect = "cover"),
    f7Panel(title = "Right Panel", side = "right", theme = "dark", "Blabla", effect = "cover")
   ),
   navbar = f7Navbar(
     title = "Tabs",
     hairline = FALSE,
     shadow = TRUE,
     left_panel = TRUE,
     right_panel = TRUE
   ),
   f7Tabs(
     id = "tabset1",
     f7Tab(
       tabName = "Tab 1",
       active = TRUE,
       p("Text 1"),
       f7Button("remove1","Remove tab 1")
     ),
     f7Tab(
       tabName = "Tab 2",
       active = FALSE,
       p("Text 2")
```

```
      ),
      f7Tab(
        tabName = "Tab 3",
        active = FALSE,
        p("Text 3")
      )
    )
   )
  )

  server <- function(input, output, session) {
    observe(print(input$tabset1))
    observeEvent(input$remove1, {
      f7RemoveTab(
        inputId = "tabset1",
        target = "Tab 1"
      )
    })
  }
  shinyApp(ui, server)
}
```

---

f7Row *Create a Framework7 row container*

Description

Build a Framework7 row container

Usage

```
f7Row(..., gap = TRUE)
```

Arguments

...	Row content.
gap	Whether to display gap between columns. TRUE by default.

Author(s)

David Granjon, <dgranjon@ymail.com>

Examples

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

 shiny::shinyApp(
  ui = f7Page(
```

```
    title = "Grid",
    f7SingleLayout(
     navbar = f7Navbar(title = "f7Row, f7Col"),
     f7Row(
     f7Col(
      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."
      )
     ),
     f7Col(
      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", "My button", src = "https://www.google.com"),
        f7Badge("Badge", color = "green")
       )
      )
     )
    )
    )
  ),
  server = function(input, output) {}
 )
}
```

---

f7Searchbar *Create a Framework 7 searchbar*

Description

Create a Framework 7 searchbar

Usage

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

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 a f7Searchbar in a f7Appbar. Notice that utilities like f7HideOnSearch and f7NotFound are not compatible with this mode.

Examples

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

cars <- rownames(mtcars)

shiny::shinyApp(
  ui = f7Page(
    title = "My app",
    f7SingleLayout(
      navbar = f7Navbar(
        title = "f7Searchbar",
        hairline = FALSE,
        shadow = TRUE,
        subNavbar = f7SubNavbar(
          f7Searchbar(id = "search1")
        )
      ),
      f7Block(
        "This block will be hidden on search.
        Lorem ipsum dolor sit amet, consectetur adipisicing elit."
      ) %>% f7HideOnSearch(),
      f7List(
        lapply(seq_along(cars), function(i) {
          f7ListItem(cars[i])
        })
      ) %>% f7Found(),

      f7Block(
        p("Nothing found")
      ) %>% f7NotFound()

    )
  ),
  server = function(input, output) {}
)

# Expandable searchbar with trigger
cities <- names(precip)

shiny::shinyApp(
  ui = f7Page(
```

```
    title = "My app",
    f7SingleLayout(
      navbar = f7Navbar(
        title = "f7Searchbar with trigger",
        hairline = FALSE,
        shadow = TRUE,
        f7SearchbarTrigger(targetId = "search1"),
        subNavbar = f7SubNavbar(
          f7Searchbar(id = "search1", expandable = TRUE)
        )
      ),
      f7Block(
        "This block will be hidden on search.
        Lorem ipsum dolor sit amet, consectetur adipisicing elit."
      ) %>% f7HideOnSearch(),
      f7List(
        lapply(seq_along(cities), function(i) {
          f7ListItem(cities[i])
        })
      ) %>% f7Found(),

      f7Block(
        p("Nothing found")
      ) %>% f7NotFound()

    )
  ),
  server = function(input, output) {}
)

# Searchbar in \link{f7Appbar}
shiny::shinyApp(
 ui = f7Page(
   title = "My app",
   f7Appbar(
     f7Searchbar(id = "search1", inline = TRUE)
   ),
   f7SingleLayout(
     navbar = f7Navbar(
       title = "f7Searchbar in f7Appbar",
       hairline = FALSE,
       shadow = TRUE
     ),
     f7List(
       lapply(seq_along(cities), function(i) {
         f7ListItem(cities[i])
       })
     ) %>% f7Found()
   )
 ),
 server = function(input, output) {}
)
}
```

f7SearchbarTrigger *Create a Framework 7 searchbar trigger*

Description

Create a Framework 7 searchbar trigger

Usage

```
f7SearchbarTrigger(targetId)
```

Arguments

targetId	Id of the f7Searchbar

Examples

```
if (interactive()) {

}
```

f7SearchIgnore *Utility to ignore an item from search.*

Description

Use with f7Searchbar.

Usage

```
f7SearchIgnore(tag)
```

Arguments

tag	tag to ignore.

f7Segment *Create a Framework7 segmented button container*

Description

Build a Framework7 segmented button container

Usage

```
f7Segment(
  ...,
  container = c("segment", "row"),
  shadow = FALSE,
  rounded = FALSE
)
```

Arguments

...	Slot for f7Button.
container	Either "row" or "segment".
shadow	Button shadow. FALSE by default. Only for material design and if the container is segment.
rounded	Round style. FALSE by default. Only if the container is segment.

Author(s)

David Granjon, <dgranjon@ymail.com>

Examples

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

 shiny::shinyApp(
  ui = f7Page(
    title = "Button Segments",
    f7SingleLayout(
     navbar = f7Navbar(title = "f7Segment, f7Button"),
     f7BlockTitle(title = "Simple Buttons in a row container"),
    f7Segment(
     container = "row",
     f7Button(color = "blue", label = "My button", fill = FALSE),
   f7Button(color = "green", label = "My button", src = "http://www.google.com", fill = FALSE),
     f7Button(color = "yellow", label = "My button", fill = FALSE)
    ),
    f7BlockTitle(title = "Filled Buttons in a segment/rounded container"),
    f7Segment(
```

```
      rounded = TRUE,
      container = "segment",
      f7Button(color = "black", label = "Action Button", inputId = "button2"),
      f7Button(color = "green", label = "My button", src = "http://www.google.com"),
      f7Button(color = "yellow", label = "My button")
     ),
     f7BlockTitle(title = "Outline Buttons in a segment/shadow container"),
     f7Segment(
      shadow = TRUE,
      container = "segment",
      f7Button(label = "My button", outline = TRUE),
      f7Button(label = "My button", outline = TRUE),
      f7Button(label = "My button", outline = TRUE)
     ),
     f7BlockTitle(title = "Rounded Buttons in a segment container"),
     f7Segment(
      container = "segment",
      f7Button(color = "blue", label = "My button", rounded = TRUE),
      f7Button(color = "green", label = "My button", rounded = TRUE),
      f7Button(color = "yellow", label = "My button", rounded = TRUE)
     ),
     f7BlockTitle(title = "Buttons of different size in a row container"),
     f7Segment(
      container = "row",
      f7Button(color = "pink", label = "My button", shadow = TRUE),
      f7Button(color = "purple", label = "My button", size = "large", shadow = TRUE),
      f7Button(color = "orange", label = "My button", size = "small", shadow = TRUE)
     ),

     br(), br(),
     f7BlockTitle(title = "Click on the black action button to update the value"),
     verbatimTextOutput("val")
     )
   ),
   server = function(input, output) {
    output$val <- renderPrint(input$button2)
   }
  )
 }
```

`f7Select` *Create an f7 select input*

Description

Create an f7 select input

Usage

```
f7Select(inputId, label, choices)
```

Arguments

inputId	Select input id.
label	Select input label.
choices	Select input choices.

Examples

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

 shiny::shinyApp(
   ui = f7Page(
     title = "My app",
     f7SingleLayout(
      navbar = f7Navbar(title = "f7Select"),
      f7Select(
       inputId = "variable",
       label = "Choose a variable:",
       choices = colnames(mtcars)[-1]
      ),
      tableOutput("data")
     )
   ),
   server = function(input, output) {
     output$data <- renderTable({
       mtcars[, c("mpg", input$variable), drop = FALSE]
     }, rownames = TRUE)
   }
 )
}
```

---

f7Shadow *Create a Framework7 shadow effect*

Description

Build a Framework7 shadow effect

Usage

```
f7Shadow(tag, intensity, hover = FALSE, pressed = FALSE)
```

Arguments

tag	Tag to apply the shadow on.
intensity	Shadow intensity. Numeric between 1 and 24. 24 is the highest elevation.
hover	Whether to display the shadow on hover. FALSE by default.
pressed	Whether to display the shadow on click. FALSE by default.

Author(s)

David Granjon, <dgranjon@ymail.com>

Examples

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

 shiny::shinyApp(
   ui = f7Page(
    title = "Shadows",
    f7SingleLayout(
     navbar = f7Navbar(title = "f7Shadow"),
     f7Shadow(
     intensity = 16,
     hover = TRUE,
     pressed = TRUE,
     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", src = "https://www.google.com"),
       f7Badge("Badge", color = "green")
      )
     )
    )
    )
   ),
   server = function(input, output) {}
 )
}
```

---

f7Sheet *Create an f7 sheet modal*

Description

Create an f7 sheet modal

Usage

```
f7Sheet(
  ...,
  hiddenItems,
  id,
```

```
  label = "Open",
  orientation = c("top", "bottom"),
  swipeToClose = FALSE,
  swipeToStep = FALSE,
  backdrop = FALSE,
  closeByOutsideClick = TRUE,
  swipeHandler = TRUE
)
```

Arguments

...	Sheet content. If wipeToStep is TRUE, these items will be visible at start.
hiddenItems	Put items you want to hide inside. Only works when wipeToStep is TRUE.
id	Sheet unique id.
label	Trigger label.
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 and Aurora themes 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.

Examples

```
if (interactive()) {
 library(shiny)
 library(shinyMobile)
 shiny::shinyApp(
    ui = f7Page(
       title = "My app",
       f7SingleLayout(
          navbar = f7Navbar(title = "f7Sheet"),
          f7Sheet(
             id = "sheet1",
             label = "More",
             orientation = "bottom",
             swipeToClose = TRUE,
             swipeToStep = 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",
             hiddenItems = tagList(
                f7Segment(
```

```
                    container = "segment",
                    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)
                 ),
                 f7Flex(
                    f7Badge(32, color = "blue"),
                    f7Badge("Badge", color = "green")
                 ),
                 f7Flex(
                    f7Gauge(
                       id = "mygauge",
                       type  = "semicircle",
                       value = 10,
                       borderColor = "#2196f3",
                       borderWidth = 10,
                       valueText = "50%",
                       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) {
        observe({print(input$sheet1)})
        output$distPlot <- renderPlot({
           hist(rnorm(input$obs))
        })
        observeEvent(input$obs, {
           updateF7Gauge(session, id = "mygauge", value = input$obs)
        })
     }
  )
 }
```

---

f7SingleLayout *Create a Framework7 single layout*

Description

Build a Framework7 single layout

Usage

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

Arguments

...	Content.
navbar	Slot for f7Navbar.
toolbar	Slot for f7Toolbar.
panels	Slot for f7Panel. Wrap in tagList if multiple panels.
appbar	Slot for f7Appbar.

Author(s)

David Granjon, <dgranjon@ymail.com>

Examples

```
if(interactive()){
 library(shiny)
 library(shinyMobile)
 shiny::shinyApp(
  ui = f7Page(
    title = "My app",
    f7SingleLayout(
      navbar = f7Navbar(
        title = "Single Layout",
        hairline = FALSE,
        shadow = TRUE
      ),
      toolbar = f7Toolbar(
        position = "bottom",
        f7Link(label = "Link 1", src = "https://www.google.com"),
        f7Link(label = "Link 2", src = "https://www.google.com", external = TRUE)
      ),
      # main content
      f7Shadow(
        intensity = 10,
        hover = TRUE,
        f7Card(
          title = "Card header",
          sliderInput("obs", "Number of observations", 0, 1000, 500),
          plotOutput("distPlot"),
          footer = tagList(
           f7Button(color = "blue", label = "My button", src = "https://www.google.com"),
            f7Badge("Badge", color = "green")
          )
```

```
        )
      )
    )
  ),
  server = function(input, output) {
    output$distPlot <- renderPlot({
      dist <- rnorm(input$obs)
      hist(dist)
    })
  }
 )
}
```

---

f7Skeleton *Create a Framework 7 skeleton loading overlay*

Description

Create a Framework 7 skeleton loading overlay

Usage

```
f7Skeleton(tag, effect = "fade", duration = 2)
```

Arguments

tag	Tag to be modified.
effect	Choose between "fade", "blink" or "pulse".
duration	Effect duration: 2s by default.

Examples

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

 shiny::shinyApp(
   ui = f7Page(
     title = "Cards",
     f7SingleLayout(
       navbar = f7Navbar(title = "f7Card"),
       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", src = "https://www.google.com"),
```

```
            f7Badge("Badge", color = "green")
          )
        ) %>% f7Skeleton(),

         f7List(
          f7ListItem(
            url = "https://www.google.com",
            title = "Item 1"
          ) %>% f7Skeleton(effect = "pulse", duration = 5) ,
          f7ListItem(
            url = "https://www.google.com",
            title = "Item 2"
          ) %>% f7Skeleton(effect = "pulse", duration = 5)
         )
      )
    ),
    server = function(input, output) {}
 )
}
```

---

f7Slide *Create a Framework7 slide*

Description

Build a Framework7 slide

Usage

```
f7Slide(...)
```

Arguments

...	Any element.

Author(s)

David Granjon, <dgranjon@ymail.com>

f7Slider *Create a f7 slider*

Description

Create a f7 slider

Usage

```
f7Slider(
  inputId,
  label,
  min,
  max,
  value,
  step = NULL,
  scale = FALSE,
  vertical = FALSE
)
```

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.
vertical	Whether to apply a vertical display. FALSE by default. Does not work yet.

Examples

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

 shiny::shinyApp(
   ui = f7Page(
    title = "My app",
    f7SingleLayout(
     navbar = f7Navbar(title = "f7Slider"),
     f7Card(
      f7Slider(
       inputId = "obs",
       label = "Number of observations",
```

```
        max = 1000,
        min = 0,
        value = 100,
        scale = TRUE
       ),
       verbatimTextOutput("test")
      ),
      plotOutput("distPlot")
     )
    ),
    server = function(input, output) {
     output$test <- renderPrint({input$obs})
     output$distPlot <- renderPlot({
      hist(rnorm(input$obs))
     })
    }
 )
}

# Create a range
if(interactive()){
 library(shiny)
 library(shinyMobile)

 shiny::shinyApp(
   ui = f7Page(
    title = "My app",
    init = f7Init(theme = "auto"),
    f7SingleLayout(
     navbar = f7Navbar(title = "f7Slider"),
     f7Card(
      f7Slider(
       inputId = "obs",
       label = "Range values",
       max = 500,
       min = 0,
       value = c(50, 100),
       scale = TRUE
      ),
      verbatimTextOutput("test")
     )
    )
   ),
   server = function(input, output) {
    output$test <- renderPrint({input$obs})
   }
 )
}
```

---

f7SmartSelect *Create a Framework7 smart select*

Description

It is nicer than the classic f7Select and allows for search.

Usage

```
f7SmartSelect(
  inputId,
  label,
  choices,
  selected = NULL,
  type = c("sheet", "popup", "popover"),
  smart = TRUE,
  multiple = FALSE
)
```

Arguments

inputId	Select input id.
label	Select input label.
choices	Select input choices.
selected	Default selected item.
type	Smart select type: either `c("sheet","popup","popover")`. Note that the search bar is only available when the type is popup.
smart	Whether to enable the search bar. TRUE by default.
multiple	Whether to allow multiple values. FALSE by default.

Examples

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

 shinyApp(
   ui = f7Page(
     title = "My app",
     f7SingleLayout(
       navbar = f7Navbar(title = "f7SmartSelect"),
       f7SmartSelect(
         inputId = "variable",
         label = "Choose a variable:",
         selected = "drat",
         choices = colnames(mtcars)[-1],
         type = "popup"
       ),
       tableOutput("data")
     )
   ),
   server = function(input, output) {
```

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

---

f7SocialCard *Create a Framework7 social card*

Description

Build a Framework7 social card

Usage

```
f7SocialCard(..., author_img = NULL, author = NULL, date = NULL, footer = NULL)
```

Arguments

...	Card content.
author_img	Author img.
author	Author.
date	Date.
footer	Footer content, if any. Must be wrapped in a tagList.

Author(s)

David Granjon, <dgranjon@ymail.com>

Examples

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

 shiny::shinyApp(
  ui = f7Page(
    title = "Social Card",
    f7SingleLayout(
     navbar = f7Navbar(title = "f7SocialCard"),
     f7SocialCard(
     author_img = "http://lorempixel.com/68/68/people/1/",
     author = "John Doe",
     date = "Monday at 3:47 PM",
     "What a nice photo i took yesterday!",
     img(src = "http://lorempixel.com/1000/700/nature/8/", width = "100%"),
     footer = tagList(
```

```
      f7Badge("1", color = "yellow"),
      f7Badge("2", color = "green"),
      f7Badge("3", color = "blue")
     )
    )
    )
  ),
  server = function(input, output) {}
 )
}
```

`f7SplitLayout` *Create a 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,
  panels = NULL,
  appbar = NULL
)
```

Arguments

`...`	Content.
`navbar`	Slot for f7Navbar.
`sidebar`	Slot for f7Panel. Particularly we expect the following code: `f7Panel(title = "Sidebar",side = "left",theme = "light","Blabla",style = "reveal")`
`toolbar`	Slot for f7Toolbar.
`panels`	Slot for f7Panel. Expect only a right panel, for instance: `f7Panel(title = "Left Panel",side = "right",theme = "light","Blabla",style = "cover")`
`appbar`	Slot for f7Appbar.

Author(s)

David Granjon, <dgranjon@ymail.com>

Examples

```
if(interactive()){
 library(shiny)
 library(shinyMobile)
 shiny::shinyApp(
   ui = f7Page(
     title = "My app",
     f7SplitLayout(
       sidebar = f7Panel(
         inputId = "sidebar",
         title = "Sidebar",
         side = "left",
         theme = "light",
         f7PanelMenu(
           id = "menu",
        f7PanelItem(tabName = "tab1", title = "Tab 1", icon = f7Icon("email"), active = TRUE),
           f7PanelItem(tabName = "tab2", title = "Tab 2", icon = f7Icon("home"))
         ),
         effect = "reveal"
       ),
       navbar = f7Navbar(
         title = "Split Layout",
         hairline = FALSE,
         shadow = TRUE
       ),
       toolbar = f7Toolbar(
         position = "bottom",
         f7Link(label = "Link 1", src = "https://www.google.com"),
         f7Link(label = "Link 2", src = "https://www.google.com", external = TRUE)
       ),
       # main content
       f7Items(
         f7Item(
           tabName = "tab1",
           sliderInput("obs", "Number of observations:",
                       min = 0, max = 1000, value = 500
           ),
           plotOutput("distPlot")
         ),
         f7Item(tabName = "tab2", "Tab 2 content")
       )
     )
   ),
   server = function(input, output) {

     observe({
       print(input$menu)
     })

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

```
      })
    }
  )
}
```

`f7Stepper` *Create a F7 radio stepper*

Description

Create a F7 radio stepper

Usage

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

Arguments

`inputId`	Stepper input id.
`label`	Stepper label.
`min`	Stepper minimum value.
`max`	Stepper maximum value.
`value`	Stepper value. Must belong to `\[min,max\]`.
`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 relied 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 accurancy. Click outside or enter Return key, ending manual mode. TRUE by default.

Note

Note that wrap, autorepeat and manual do not work.

Examples

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

 shiny::shinyApp(
   ui = f7Page(
    title = "My app",
    f7SingleLayout(
     navbar = f7Navbar(title = "f7Stepper"),
     f7Stepper(
      inputId = "stepper",
      label = "My stepper",
      min = 0,
      max = 10,
      value = 4
     ),
     verbatimTextOutput("test"),
     f7Stepper(
      inputId = "stepper2",
      label = "My stepper 2",
      min = 0,
      max = 10,
      value = 4,
      color = "orange",
      raised = TRUE,
      fill = TRUE,
      rounded = TRUE
     ),
     verbatimTextOutput("test2")
    )
   ),
   server = function(input, output) {
```

```
    output$test <- renderPrint(input$stepper)
    output$test2 <- renderPrint(input$stepper2)
   }
 )
}
```

---

f7SubNavbar *Create a Framework7 sub navbar*

Description

Create a Framework7 sub navbar

Usage

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

Arguments

...	Any elements.

Examples

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

 shiny::shinyApp(
  ui = f7Page(
     title = "Sub Navbar",
     f7TabLayout(
        panels = tagList(
      f7Panel(title = "Left Panel", side = "left", theme = "light", "Blabla", style = "cover"),
      f7Panel(title = "Right Panel", side = "right", theme = "dark", "Blabla", style = "cover")
        ),
        navbar = f7Navbar(
           title = "SubNavbar",
           hairline = FALSE,
           shadow = TRUE,
           left_panel = TRUE,
           right_panel = TRUE,
           subNavbar = f7SubNavbar(
              f7Button(label = "My button", outline = TRUE),
              f7Button(label = "My button", outline = TRUE),
              f7Button(label = "My button", outline = TRUE)
           )
        ),
        f7Tabs(
           animated = TRUE,
           #swipeable = TRUE,
```

```
            f7Tab(
               tabName = "Tab 1",
               icon = f7Icon("email"),
               active = TRUE,
               "Tab 1"
            ),
            f7Tab(
               tabName = "Tab 2",
               icon = f7Icon("today"),
               active = FALSE,
               "Tab 2"
            ),
            f7Tab(
               tabName = "Tab 3",
               icon = f7Icon("cloud_upload"),
               active = FALSE,
               "Tab 3"
            )
         )
      )
    ),
    server = function(input, output) {}
  )
}
```

`f7Swipeout` *Create a framework7 swipeout element*

Description

To be used in combination with f7ListItem

Usage

```
f7Swipeout(
  tag,
  ...,
  left = NULL,
  right = NULL,
  side = c("left", "right", "both")
)
```

Arguments

`tag`	Tag to be swiped.
`...`	When side is either "right" or "left" use this slot to pass f7SwipeoutItem.
`left`	When side is "both", put the left f7SwipeoutItem.
`right`	When side is "both", put the right f7SwipeoutItem.
`side`	On which side to swipe: "left", "right" or "both".

Examples

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

 shiny::shinyApp(
   ui = f7Page(
     title = "My app",
     f7SingleLayout(
       navbar = f7Navbar(title = "f7List"),
       # simple list
       f7List(
         lapply(1:3, function(j) {
           if (j == 1) {
             f7Swipeout(
               tag = f7ListItem(letters[j]),
               side = "left",
               f7SwipeoutItem(id = "alert", color = "pink", "Alert"),
               f7SwipeoutItem(id = "notification", color = "green", "Notif")
             )
           } else {
             f7ListItem(letters[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",
         session = session
       )
     })

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

```
    }
  )
}
```

`f7SwipeoutItem` *Create a framework7 swipeout item*

Description

Insert in f7Swipeout

Usage

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

Arguments

`id`	Item unique id.
`label`	Item label.
`color`	Item color.

`f7Swiper` *Create a Framework7 swiper*

Description

Build a Framework7 swiper (like carousel)

Usage

```
f7Swiper(
  ...,
  id,
  spaceBetween = 50,
  slidePerView = "auto",
  centered = TRUE,
  speed = 400
)
```

Arguments

`...`	Slot for f7Slide.
`id`	Swiper unique id.
`spaceBetween`	Space between slides. 50 by default. Only if pagination is TRUE.
`slidePerView`	Number of slides at a time. Only if pagination is TRUE. Set to "auto" by default.
`centered`	Whether to center slides. Only if pagination is TRUE.
`speed`	Slides speed. Numeric.

Author(s)

David Granjon, <dgranjon@ymail.com>

Examples

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

 timeline <- f7Timeline(
  sides = TRUE,
  f7TimelineItem(
   "Another text",
   date = "01 Dec",
   card = FALSE,
   time = "12:30",
   title = "Title",
   subtitle = "Subtitle",
   side = "left"
  ),
  f7TimelineItem(
   "Another text",
   date = "02 Dec",
   card = TRUE,
   time = "13:00",
   title = "Title",
   subtitle = "Subtitle"
  ),
  f7TimelineItem(
   "Another text",
   date = "03 Dec",
   card = FALSE,
   time = "14:45",
   title = "Title",
   subtitle = "Subtitle"
  )
 )

 shiny::shinyApp(
   ui = f7Page(
    title = "My app",
    f7SingleLayout(
     navbar = f7Navbar(title = "f7Swiper"),
     f7Swiper(
     id = "my-swiper",
     f7Slide(
      timeline
     ),
     f7Slide(
      f7Toggle(
       inputId = "toggle",
       label = "My toggle",
```

```
        color = "pink",
        checked = TRUE
       ),
       verbatimTextOutput("test")
      )
     )
     )
    ),
    server = function(input, output) {
     output$test <- renderPrint(input$toggle)
    }
  )
}
```

---

f7Tab *Create a Framework7 tab item*

Description

Build a Framework7 tab item

Usage

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

Arguments

...	Item content.
tabName	Item id. Must be unique.
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

Author(s)

David Granjon, <dgranjon@ymail.com>

f7TabLayout *Create a Framework7 page with tab layout*

Description

Build a Framework7 page with tab layout

Usage

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

Arguments

...	Slot for f7Tabs.
navbar	Slot for f7Navbar.
panels	Slot for f7Panel. Wrap in tagList if multiple panels.
appbar	Slot for f7Appbar.

Author(s)

David Granjon, <dgranjon@ymail.com>

Examples

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

 shiny::shinyApp(
  ui = f7Page(
    title = "Tab Layout",
    f7TabLayout(
      panels = tagList(
     f7Panel(title = "Left Panel", side = "left", theme = "light", "Blabla", effect = "cover"),
     f7Panel(title = "Right Panel", side = "right", theme = "dark", "Blabla", effect = "cover")
      ),
      navbar = f7Navbar(
        title = "Tabs",
        hairline = FALSE,
        shadow = TRUE,
        left_panel = TRUE,
        right_panel = TRUE
      ),
      f7Tabs(
        animated = TRUE,
        f7Tab(
          tabName = "Tab 1",
          icon = f7Icon("email"),
```

```
      active = TRUE,
      f7Shadow(
        intensity = 10,
        hover = TRUE,
        f7Card(
          title = "Card header",
          sliderInput("obs1", "Number of observations", 0, 1000, 500),
          plotOutput("distPlot1"),
          footer = tagList(
        f7Button(color = "blue", label = "My button", src = "https://www.google.com"),
            f7Badge("Badge", color = "green")
          )
        )
      )
    ),
    f7Tab(
      tabName = "Tab 2",
      icon = f7Icon("today"),
      active = FALSE,
      f7Shadow(
        intensity = 10,
        hover = TRUE,
        f7Card(
          title = "Card header",
          sliderInput("obs2", "Number of observations", 0, 10000, 5000),
          plotOutput("distPlot2"),
          footer = tagList(
        f7Button(color = "blue", label = "My button", src = "https://www.google.com"),
            f7Badge("Badge", color = "green")
          )
        )
      )
    ),
    f7Tab(
      tabName = "Tab 3",
      icon = f7Icon("cloud_upload"),
      active = FALSE,
      f7Shadow(
        intensity = 10,
        hover = TRUE,
        f7Card(
          title = "Card header",
          sliderInput("obs3", "Number of observations", 0, 10, 5),
          plotOutput("distPlot3"),
          footer = tagList(
        f7Button(color = "blue", label = "My button", src = "https://www.google.com"),
            f7Badge("Badge", color = "green")
          )
        )
      )
    )
  )
)
```

```
  ),
  server = function(input, output) {
    output$distPlot1 <- renderPlot({
      dist <- rnorm(input$obs1)
      hist(dist)
    })
    output$distPlot2 <- renderPlot({
      dist <- rnorm(input$obs2)
      hist(dist)
    })
    output$distPlot3 <- renderPlot({
      dist <- rnorm(input$obs3)
      hist(dist)
    })
  }
 )
}
```

`f7Tabs` *Create a Framework7 tabs*

Description

Build a Framework7 tabs

Usage

```
f7Tabs(..., .items = NULL, id = NULL, swipeable = FALSE, animated = TRUE)
```

Arguments

`...`	Slot for f7Tab.
`.items`	Slot for other items that could be part of the toolbar such as buttons or f7Sheet.
`id`	Optional to get the id of the currently selected f7Tab.
`swipeable`	Whether to allow finger swip. 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.

Note

Animated does not work when set to FALSE and swipeable is FALSE.

Author(s)

David Granjon, <dgranjon@ymail.com>

Examples

```
if (interactive()) {
 library(shiny)
 library(shinyMobile)
 shiny::shinyApp(
  ui = f7Page(
    title = "Tab Layout",
    f7TabLayout(
    navbar = f7Navbar(title = HTML(paste("Currently selected:", textOutput("selected")))),
      f7Tabs(
        id = "tabdemo",
        swipeable = TRUE,
        animated = FALSE,
        f7Tab(tabName = "Tab 1", "tab 1 text"),
        f7Tab(tabName = "Tab 2", "tab 2 text"),
        f7Tab(tabName = "Tab 3", "tab 3 text"),
        .items = shiny::tags$a(
         class = "tab-link",
         href = "#",
         f7Icon("bolt_fill"),
         shiny::span(class = "tabbar-label", "Optional Item")
        )
      )
    )
  ),
  server = function(input, output) {
    output$selected <- renderText(input$tabdemo)
  }
 )
}
```

`f7TapHold` *Create a Framework7 tapHold event*

Description

Triggered after long press on an element.

Usage

```
f7TapHold(target, callback, session)
```

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

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

 shinyApp(
   ui = f7Page(
     title = "My app",
     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!');",
       session = session
      )
    })
   }
 )
}
```

---

f7Text *Create an f7 text input*

Description

Create an f7 text input

Usage

```
f7Text(inputId, label, value = "", placeholder = NULL)
```

Arguments

inputId	Text input id.
label	Text input label.
value	Text input value.
placeholder	Text input placeholder.

Examples

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

 shiny::shinyApp(
   ui = f7Page(
     title = "My app",
     f7SingleLayout(
      navbar = f7Navbar(title = "f7Text"),
      f7Text(
       inputId = "caption",
       label = "Caption",
       value = "Data Summary",
       placeholder = "Your text here"
      ),
      verbatimTextOutput("value")
     )
   ),
   server = function(input, output) {
     output$value <- renderPrint({ input$caption })
   }
 )
}
```

`f7Timeline` *Create a Framework7 timeline*

Description

Build a Framework7 timeline

Usage

```
f7Timeline(
  ...,
  sides = FALSE,
  horizontal = FALSE,
  calendar = FALSE,
  year = NULL,
  month = NULL
)
```

Arguments

`...`	Slot for f7TimelineItem.
`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.

Author(s)

David Granjon and Isabelle Rudolf, <dgranjon@ymail.com>

Examples

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

 items <- tagList(
   f7TimelineItem(
     "Another text",
     date = "01 Dec",
     card = FALSE,
     time = "12:30",
     title = "Title",
     subtitle = "Subtitle",
     side = "left"
   ),
   f7TimelineItem(
     "Another text",
     date = "02 Dec",
     card = TRUE,
     time = "13:00",
     title = "Title",
     subtitle = "Subtitle"
   ),
   f7TimelineItem(
     "Another text",
     date = "03 Dec",
     card = FALSE,
     time = "14:45",
     title = "Title",
     subtitle = "Subtitle"
   )
 )

 shiny::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) {}
)
}
```

`f7TimelineItem` *Create a Framework7 timeline item*

Description

Build a Framework7 timeline item

Usage

```
f7TimelineItem(
  ...,
  date = NULL,
  card = FALSE,
  time = NULL,
  title = NULL,
  subtitle = NULL,
  side = NULL
)
```

Arguments

`...`	Item content, text for instance.
`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 and Isabelle Rudolf, <dgranjon@ymail.com>

---

f7Toast *Create a Framework7 toast*

Description

Create a Framework7 toast

Usage

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

Arguments

session	Shiny session.
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.

Examples

```
if (interactive()) {
 library(shiny)
 library(shinyMobile)
 shinyApp(
  ui = f7Page(
    title = "My app",
    f7SingleLayout(
      navbar = f7Navbar(title = "f7Toast"),
      f7Button(inputId = "toast", label = "Open Toast")
    )
  ),
  server = function(input, output, session) {
    observeEvent(input$toast, {
      f7Toast(
        session,
        position = "top",
        text = "I am a toast. Eat me!"
      )
    })
  }
 )
}
```

---

f7Toggle *Create a F7 toggle switch*

Description

Create a F7 toggle switch

Usage

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

Arguments

inputId	Toggle input id.
label	Toggle label.
checked	Whether to check the toggle. FALSE by default.
color	Toggle color: NULL or "red", "green", "blue", "pink", "yellow", "orange", "grey" and "black".

Examples

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

 shiny::shinyApp(
   ui = f7Page(
    title = "My app",
    f7SingleLayout(
     navbar = f7Navbar(title = "f7Toggle"),
     f7Toggle(
      inputId = "toggle",
      label = "My toggle",
      color = "pink",
      checked = TRUE
     ),
     verbatimTextOutput("test"),
     f7Toggle(
      inputId = "toggle2",
      label = "My toggle 2"
     ),
     verbatimTextOutput("test2")
    )
   ),
   server = function(input, output) {
    output$test <- renderPrint(input$toggle)
    output$test2 <- renderPrint(input$toggle2)
   }
 )
}
```

---

f7Toolbar *Create a Framework7 Toolbar*

Description

Build a Framework7 Toolbar

Usage

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

Arguments

...	Slot for f7Link or any other element.
position	Tabs position: "top" or "bottom".
hairline	Whether to display a thin border on the top of the toolbar. TRUE by default.
shadow	Whether to display a shadow. TRUE by default.
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@ymail.com>

---

f7Tooltip *Create a Framework7 tooltip*

Description

This uses the auto init framework 7 tooltip

Usage

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

Arguments

tag	Tooltip target.
text	Tooltip content.

Examples

```
if (interactive()) {
 library(shiny)
 library(shinyMobile)
 shinyApp(
   ui = f7Page(
     title = "My app",
     f7SingleLayout(
       navbar = f7Navbar(title = "f7Tooltip"),
       f7Tooltip(
         f7Badge("Hover on me", color = "pink"),
         text = "A tooltip!"
       )
     )
   ),
   server = function(input, output, session) {
   }
 )
}
```

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

updateF7Accordion *Update a Framework 7 accordion*

Description

Update a Framework 7 accordion

Usage

```
updateF7Accordion(inputId, selected = NULL, session)
```

Arguments

inputId	Accordion instance.
selected	Index of item to select.
session	Shiny session object

Examples

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

 shiny::shinyApp(
   ui = f7Page(
     title = "Accordions",
     f7SingleLayout(
       navbar = f7Navbar("Accordions"),
       f7Button(inputId = "go", "Go"),
       f7Accordion(
         inputId = "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(inputId = "myaccordion1", selected = 2, session = session)
        })

        observe({
          print(
            list(
              accordion1_state = input$myaccordion1$state,
              accordion1_values = unlist(input$myaccordion1$value)
            )
          )
        })
      }
    )
  }
```

`updateF7AutoComplete` *Change the value of an autocomplete input on the client*

Description

Change the value of an autocomplete input on the client

Usage

```
updateF7AutoComplete(session, inputId, value = NULL)
```

Arguments

`session`	The session object passed to function given to the server.
`inputId`	The id of the input object.
`value`	Picker initial value, if any.

Note

You cannot update choices yet.

Examples

```
if (interactive()) {
 library(shiny)
 library(shinyMobile)
 shinyApp(
  ui = f7Page(
    title = "My app",
    f7SingleLayout(
      navbar = f7Navbar(title = "Update autocomplete"),
      f7Card(
        f7Button(inputId = "update", label = "Update autocomplete"),
        f7AutoComplete(
         inputId = "myautocomplete",
         placeholder = "Some text here!",
         type = "dropdown",
         label = "Type a fruit name",
         choices = c('Apple', 'Apricot', 'Avocado', 'Banana', 'Melon',
                     'Orange', 'Peach', 'Pear', 'Pineapple')
        ),
        verbatimTextOutput("autocompleteval")
      )
    )
  ),
  server = function(input, output, session) {

    observe({
     print(input$myautocomplete)
    })

    output$autocompleteval <- renderText(input$myautocomplete)

    observeEvent(input$update, {
      updateF7AutoComplete(
        session,
        inputId = "myautocomplete",
        value = "Banana"
      )
    })
  }
 )
}
```

---

updateF7Card *Update a framework 7 expandable card*

Description

Update a framework 7 expandable card

Usage

```
updateF7Card(id, session)
```

Arguments

id	Card id.
session	Shiny session object.

Examples

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

 shiny::shinyApp(
   ui = f7Page(
     title = "Expandable Cards",
     f7SingleLayout(
       navbar = f7Navbar(
         title = "Expandable Cards",
         hairline = FALSE,
         shadow = TRUE
       ),
       f7ExpandableCard(
         id = "card1",
         title = "Expandable Card 1",
      img = "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,
         img = "https://i.ytimg.com/vi/8q_kmxwK5Rg/maxresdefault.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."
       )
     )
   ),
   server = function(input, output, session) {

     observeEvent(input$go, {
```

```
      updateF7Card(id = "card2", session = session)
    })

    observe({
      list(
        print(input$card1),
        print(input$card2)
      )
    })
  }
 )
}
```

---

updateF7Checkbox *Change the value of a checkbox input on the client*

Description

Change the value of a checkbox input on the client

Usage

```
updateF7Checkbox(session, inputId, label = NULL, value = NULL)
```

Arguments

session	The session object passed to function given to the server.
inputId	The id of the input object.
label	The label to set for the input object.
value	The value to set for the input object.

Examples

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

 ui <- f7Page(
   f7SingleLayout(
    navbar = f7Navbar(title = "updateF7CheckBox"),
    f7Slider(
     inputId = "controller",
     label = "Number of observations",
     max = 10,
     min = 0,
     value = 1,
     step = 1,
     scale = TRUE
    ),
```

```
    f7checkBox(
     inputId = "check",
     label = "Checkbox"
    )
   )
 )

 server <- function(input, output, session) {
   observe({
     # TRUE if input$controller is odd, FALSE if even.
     x_even <- input$controller %% 2 == 1

     if (x_even) {
      showNotification(
       id = "notif",
       paste("The slider is ", input$controller, "and the checkbox is", input$check),
       duration = NULL,
       type = "warning"
      )
     } else {
      removeNotification("notif")
     }

     updateF7Checkbox(session, "check", value = x_even)
   })
 }

shinyApp(ui, server)
}
```

`updateF7Fab` *Change the value of a f7Fab input on the client*

Description

Change the value of a f7Fab input on the client

Usage

```
updateF7Fab(session, inputId, label = NULL)
```

Arguments

`session`	The session object passed to function given to the server.
`inputId`	The id of the input object.
`label`	The label to set for the input object.

Examples

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

 ui <- f7Page(
   f7SingleLayout(
    navbar = f7Navbar(title = "updateF7Fab"),
    f7Fab("trigger", "Click me")
   )
 )

 server <- function(input, output, session) {
   observeEvent(input$trigger, {
     updateF7Fab(session, "trigger", label = "Don't click me")
   })
 }
shinyApp(ui, server)
}
```

`updateF7Gauge` *update a framework7 gauge from the server side*

Description

update a framework7 gauge from the server side

Usage

```
updateF7Gauge(session, id, value)
```

Arguments

`session`	Shiny session object.
`id`	Gauge id.
`value`	New value. Numeric between 0 and 100.

Examples

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


 shiny::shinyApp(
    ui = f7Page(
       title = "Gauges",
       f7SingleLayout(
```

```
            navbar = f7Navbar(title = "update f7Gauge"),
            f7Gauge(
               id = "mygauge",
               type  = "semicircle",
               value = 50,
               borderColor = "#2196f3",
               borderWidth = 10,
               valueText = "50%",
               valueFontSize = 41,
               valueTextColor = "#2196f3",
               labelText = "amount of something"
            ),
            f7Button("go", "Update Gauge")
         )
      ),
      server = function(input, output, session) {
         observeEvent(input$go, {
            updateF7Gauge(session, id = "mygauge", value = 75)
         })
      }
   )
}
```

`updateF7Panel` *Function to programmatically update the state of a f7Panel*

Description

From open to close state and inversely

Usage

```
updateF7Panel(inputId, session)
```

Arguments

`inputId`	Panel unique id. This is to access the input$id giving the panel state, namely open or closed.
`session`	Shiny session object.

Examples

```
if (interactive()) {
 library(shiny)
 library(shinyMobile)
 shiny::shinyApp(
   ui = f7Page(
     title = "My app",
     init = f7Init(skin = "md", theme = "light"),
```

```
    f7SingleLayout(
      navbar = f7Navbar(
        title = "Single Layout",
        hairline = FALSE,
        shadow = TRUE,
        left_panel = TRUE,
        right_panel = TRUE
      ),
      panels = tagList(
       f7Panel(side = "left", inputId = "mypanel1", theme = "light", effect = "cover"),
        f7Panel(side = "right", inputId = "mypanel2", theme = "light")
      ),
      toolbar = f7Toolbar(
        position = "bottom",
        icons = TRUE,
        hairline = FALSE,
        shadow = FALSE,
        f7Link(label = "Link 1", src = "https://www.google.com"),
        f7Link(label = "Link 2", src = "https://www.google.com", external = TRUE)
      )
    )
  ),
  server = function(input, output, session) {

    observe({
      print(
        list(
          panel1 = input$mypanel1,
          panel2 = input$mypanel2
        )
      )
    })

    observe({
      invalidateLater(2000)
      updateF7Panel(inputId = "mypanel1", session = session)
    })

  }
)
}
```

---

updateF7Picker *Change the value of a picker input on the client*

Description

Change the value of a picker input on the client

Usage

```
updateF7Picker(session, inputId, value = NULL, choices = NULL)
```

Arguments

session	The session object passed to function given to the server.
inputId	The id of the input object.
value	Picker initial value, if any.
choices	New picker choices.

Examples

```
if (interactive()) {
 library(shiny)
 library(shinyMobile)
 shinyApp(
  ui = f7Page(
    title = "My app",
    f7SingleLayout(
      navbar = f7Navbar(title = "Update picker"),
      f7Card(
        f7Button(inputId = "update", label = "Update picker"),
        f7Picker(
          inputId = "mypicker",
          placeholder = "Some text here!",
          label = "Picker Input",
          choices = c('a', 'b', 'c')
        ),
        verbatimTextOutput("pickerval")
      )
    )
  ),
  server = function(input, output, session) {

    output$pickerval <- renderText(input$mypicker)

    observeEvent(input$update, {
      updateF7Picker(
        session,
        inputId = "mypicker",
        value = "b",
        choices = letters
      )
    })
  }
 )
}
```

updateF7Progress *update a framework7 progress bar from the server side*

Description

update a framework7 progress bar from the server side

Usage

```
updateF7Progress(session, id, value)
```

Arguments

session	Shiny session object.
id	Unique progress bar id.
value	New value.

Examples

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

 shiny::shinyApp(
   ui = f7Page(
     title = "Progress",
     f7SingleLayout(
       navbar = f7Navbar(title = "f7Progress"),
       f7Block(
         f7Progress(id = "pg1", value = 10, color = "blue")
       ),
       f7Slider(
         inputId = "obs",
         label = "Progress value",
         max = 100,
         min = 0,
         value = 50,
         scale = TRUE
       )
     )
   ),
   server = function(input, output, session) {
     observeEvent(input$obs, {
       updateF7Progress(session, id = "pg1", value = input$obs)
     })
   }
 )
}
```

`updateF7Sheet` *update a framework 7 sheet modal*

Description

update a framework 7 sheet modal

Usage

```
updateF7Sheet(inputId, session)
```

Arguments

`inputId`	Sheet id.
`session`	Shiny session object

Examples

```
if (interactive()) {
 library(shiny)
 library(shinyMobile)
 shiny::shinyApp(
    ui = f7Page(
       color = "pink",
       title = "My app",
       f7SingleLayout(
          navbar = f7Navbar(title = "f7Sheet"),
          f7Button(inputId = "go", label = "Go"),
          f7Sheet(
             id = "sheet1",
             label = "More",
             orientation = "bottom",
             swipeToClose = TRUE,
             swipeToStep = 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"
           )
       )
    ),
    server = function(input, output, session) {
       observe({print(input$sheet1)})
       observeEvent(input$go, {
          updateF7Sheet(inputId = "sheet1", session = session)
       })
    }
 )
}
```

updateF7Slider *Change the value of a slider input on the client*

Description

Change the value of a slider input on the client

Usage

```
updateF7Slider(
  session,
  inputId,
  min = NULL,
  max = NULL,
  value = NULL,
  scale = FALSE
)
```

Arguments

session	The session object passed to function given to the server.
inputId	The id of the input object.
min	Slider minimum range.
max	Slider maximum range
value	Slider value or a vector containing 2 values (for a range).
scale	Slider scale.

Examples

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

 shinyApp(
  ui = f7Page(
    title = "My app",
    f7SingleLayout(
      navbar = f7Navbar(title = "updateF7Slider"),
      f7Card(
        f7Button(inputId = "update", label = "Update slider"),
        f7Slider(
          inputId = "obs",
          label = "Range values",
          max = 500,
          min = 0,
          value = c(50, 100),
          scale = TRUE,
```

```
            vertical = FALSE
          ),
          verbatimTextOutput("test")
        )
      )
    ),
    server = function(input, output, session) {

      output$test <- renderPrint({input$obs})

      observeEvent(input$update, {
        updateF7Slider(
          session,
          inputId = "obs",
          value = c(20, 40),
          min = 10,
          max = 50,
          scale = FALSE
        )
      })
    }
  )
}
```

---

updateF7Stepper *Change the value of a stepper input on the client*

Description

Change the value of a stepper input on the client

Usage

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

Arguments

session	The session object passed to function given to the server.
inputId	The id of the input object.
min	Stepper minimum value.
max	Stepper maximum value.
value	Stepper value. Must belong to \[min, max\].
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 relied 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 accurancy. Click outside or enter Return key, ending manual mode. TRUE by default.

Note

While updating, the autorepeat field does not work correctly.

Examples

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

 shinyApp(
  ui = f7Page(
    title = "My app",
    f7SingleLayout(
      navbar = f7Navbar(title = "updateF7Stepper"),
      f7Card(
        f7Button(inputId = "update", label = "Update stepper"),
        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(
          session,
          inputId = "stepper",
          value = 10,
          size = "large",
          min = 5,
          max = 20,
          wraps = FALSE,
          autorepeat = FALSE,
          rounded = TRUE,
          raised = TRUE,
          color = "pink",
          manual = TRUE
        )
      })
    }
  )
}
```

`updateF7Tabs` *Update a Framework 7 tabsetPanel*

Description

Update f7Tabs.

Usage

```
updateF7Tabs(session, id, selected = NULL)
```

Arguments

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

Examples

```
if (interactive()) {
 library(shiny)
 library(shinyMobile)
 shiny::shinyApp(
   ui = f7Page(
     title = "Tab Layout",
     f7TabLayout(
       navbar = f7Navbar(
        title = HTML(paste("Currently selected:", textOutput("selected"))),
        subNavbar = f7SubNavbar(
         f7Flex(
          f7Toggle(inputId = "updateTab", label = "Update Tab", checked = TRUE),
          f7Toggle(inputId = "updateSubTab", label = "Update SubTab", checked = FALSE)
         )
        )
       ),
       f7Tabs(
         id = "tabdemo",
         swipeable = TRUE,
         animated = FALSE,
         f7Tab(
          tabName = "Tab 1",
          f7Tabs(
           id = "subtabdemo",
           animated = TRUE,
           f7Tab(tabName = "SubTab 1", "SubTab 1"),
           f7Tab(tabName = "SubTab 2", "SubTab 2", active = TRUE),
           f7Tab(tabName = "SubTab 3", "SubTab 3")
          )
         ),
         f7Tab(tabName = "Tab 2", "Tab 2"),
         f7Tab(tabName = "Tab 3", "Tab 3")
       )
     )
   ),
   server = function(input, output, session) {
     observeEvent(input$updateTab, {
       selected <- ifelse(input$updateTab, "Tab 1", "Tab 2")
       updateF7Tabs(session, id = "tabdemo", selected = selected)
     })
     observeEvent(input$updateSubTab, {
       selected <- ifelse(input$updateSubTab, "SubTab 1", "SubTab 2")
       updateF7Tabs(session, id = "subtabdemo", selected = selected)
     })
```

```
  }
 )
}
```

---

updateF7Text *Change the value of a text input on the client*

Description

Change the value of a text input on the client

Usage

```
updateF7Text(session, inputId, label = NULL, value = NULL, placeholder = NULL)
```

Arguments

session	The session object passed to function given to the server.
inputId	The id of the input object.
label	The label to set for the input object.
value	The value to set for the input object.
placeholder	The placeholder to set for the input object.

Examples

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

 ui <- f7Page(
   f7SingleLayout(
    navbar = f7Navbar(title = "updateF7Text"),
    f7Fab("trigger", "Click me"),
    f7Text(
     inputId = "text",
     label = "Caption",
     value = "Some text",
     placeholder = "Your text here"
    ),
    verbatimTextOutput("value")
   )
 )

 server <- function(input, output, session) {
   output$value <- renderPrint(input$text)
   observeEvent(input$trigger, {
     updateF7Text(session, "text", value = "Updated Text")
   })
```

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

---

updateF7Toggle *Change the value of a toggle input on the client*

Description

Change the value of a toggle input on the client

Usage

```
updateF7Toggle(session, inputId, checked = NULL, color = NULL)
```

Arguments

session	The session object passed to function given to the server.
inputId	The id of the input object.
checked	Whether the toggle is TRUE or FALSE.
color	Toggle color.

Examples

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

 shinyApp(
   ui = f7Page(
     title = "My app",
     f7SingleLayout(
       navbar = f7Navbar(title = "updateF7Toggle"),
       f7Card(
         f7Button(inputId = "update", label = "Update toggle"),
         f7Toggle(
           inputId = "toggle",
           label = "My toggle",
           color = "pink",
           checked = FALSE
         ),
         verbatimTextOutput("test")
       )
     )
   ),
   server = function(input, output, session) {

     output$test <- renderPrint({input$toggle})
```

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

Index