

Package: shinyMobile (via r-universe)

September 6, 2024

Type Package

Title Mobile Ready 'shiny' Apps with Standalone Capabilities

Version 1.0.0.9001

Maintainer David Granjon <dgranjon@gmail.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 7.1.1

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

VignetteBuilder knitr

Repository <https://tanho63.r-universe.dev>

RemoteUrl <https://github.com/tanho63/shinyMobile>

RemoteRef HEAD

RemoteSha 2add4d6f5ce4bf825f82d01fbfa7c62bef058ad

Contents

addF7Popover	4
add_dependencies	6
add_f7icons_dependencies	6
add_framework7_deps	7
add_pwacompat_deps	7
add_pwa_deps	7
add_shinyMobile_deps	8
createSelectOptions	8
create_app_ui	8
f7Accordion	9
f7ActionSheet	11
f7Align	16
f7AppBar	17
f7AutoComplete	19
f7Badge	22
f7Block	23
f7BlockFooter	25
f7BlockTitle	25
f7Button	26
f7Card	28
f7Checkbox	32
f7CheckboxGroup	34
f7Chip	35
f7Col	37
f7ColorPicker	37
f7DatePicker	39
f7Dialog	42
f7DownloadButton	45
f7Fab	46
f7FabClose	47
f7Fabs	47
f7File	50
f7Flex	52
f7Float	53
f7Found	54
f7Gallery	54
f7Gauge	55
f7HideOnEnable	57
f7HideOnSearch	58
f7Icon	58
f7Item	59
f7Items	60
f7Link	60
f7List	61
f7ListGroup	63
f7ListIndex	63

f7ListIndexItem	65
f7ListItem	65
f7Login	66
f7Margin	70
f7Menu	71
f7MessageBar	73
f7Messages	74
f7Navbar	77
f7NotFound	79
f7Notif	79
f7Padding	81
f7Page	82
f7Panel	83
f7PanelMenu	86
f7Password	87
f7PhotoBrowser	88
f7Picker	89
f7Popup	92
f7Progress	94
f7Radio	95
f7Row	97
f7Searchbar	98
f7SearchbarTrigger	101
f7SearchIgnore	101
f7Segment	102
f7Select	104
f7Shadow	105
f7Sheet	107
f7SingleLayout	109
f7Skeleton	110
f7Slide	112
f7Slider	112
f7SmartSelect	116
f7SplitLayout	118
f7Stepper	120
f7SubNavbar	124
f7Swipeout	125
f7Swiper	127
f7Tab	128
f7TabLayout	129
f7Table	132
f7TabLink	133
f7Tabs	134
f7TapHold	137
f7Text	138
f7TextArea	140
f7Timeline	142
f7Toast	144

f7Toggle	145
f7Toolbar	147
f7Tooltip	148
f7VirtualList	150
getF7Colors	153
insertF7Tab	153
preview_mobile	155
removeF7Tab	156
showF7Preloader	157
updateF7App	159
updateF7Entity	161
updateF7Tabs	162
updateF7VirtualList	165
validateF7Input	169

Index	171
--------------	------------

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

Description

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

Usage

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

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

Arguments

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

Examples

```

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

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

  popovers <- data.frame(
    id = paste0("target_", 1:10),
    content = paste("Popover content", 1:10, lorem_ipsum),
    stringsAsFactors = FALSE
  )

  shinyApp(
    ui = f7Page(
      options = list(theme = "ios"),
      title = "f7Popover",
      f7SingleLayout(
        navbar = f7Navbar(
          title = "f7Popover",
          subNavbar = f7SubNavbar(
            f7Toggle(
              inputId = "toggle",
              "Enable popover",
              color = "green",
              checked = TRUE
            )
          )
        )
      ),
      f7Segment(
        lapply(seq_len(nrow(popovers)), function(i) {
          f7Button(
            inputId = sprintf("target_%s", i),
            sprintf("Target %s", i)
          )
        })
      )
    ),
    server = function(input, output, session) {
      # Enable/disable (don't run first)
      observeEvent(input$toggle, {
        lapply(seq_len(nrow(popovers)), function(i) toggleF7Popover(id = popovers[i, "id"]) )
      }, ignoreInit = TRUE)

      # show
      lapply(seq_len(nrow(popovers)), function(i) {
        observeEvent(input[[popovers[i, "id"]]], {

```

```

    addF7Popover(
      id = popovers[i, "id"],
      options = list(
        content = popovers[i, "content"]
      )
    )
  })
}
)
}

```

add_dependencies *Attach all created dependencies in the ./R directory to the provided tag*

Description

This function only works if there are existing dependencies. Otherwise, an error is raised.

Usage

```
add_dependencies(tag, deps = NULL)
```

Arguments

tag	Tag to attach the dependencies.
deps	Dependencies to add. Expect a vector of names. If NULL, all dependencies are added.

add_f7icons_dependencies
Framework7 icon dependencies

Description

This function attaches icon dependencies to the given tag.

Usage

```
add_f7icons_dependencies(tag)
```

Arguments

tag	Element to attach the dependencies.
-----	-------------------------------------

add_framework7_deps *framework7 dependencies utils*

Description

This function attaches framework7. dependencies to the given tag

Usage

```
add_framework7_deps(tag)
```

Arguments

tag Element to attach the dependencies.

add_pwacompat_deps *pwacompat dependencies utils*

Description

This function attaches pwacompat. dependencies to the given tag

Usage

```
add_pwacompat_deps(tag)
```

Arguments

tag Element to attach the dependencies.

add_pwa_deps *PWA dependencies utils*

Description

This function attaches PWA manifest and icons to the given tag

Usage

```
add_pwa_deps(tag)
```

Arguments

tag Element to attach the dependencies.

add_shinyMobile_deps *shinyMobile dependencies utils*

Description

This function attaches shinyMobile dependencies to the given tag

Usage

```
add_shinyMobile_deps(tag)
```

Arguments

tag Element to attach the dependencies.

createSelectOptions *Create option html tag based on choice input*

Description

Used by [f7SmartSelect](#) and [f7Select](#)

Usage

```
createSelectOptions(choices, selected)
```

Arguments

choices Vector of possibilities.
selected Default selected value.

create_app_ui *Create the app UI*

Description

Internal

Usage

```
create_app_ui(iframe, device, color, landscape)
```


Arguments

iframe	iframe tag designed by preview_mobile .
device	See preview_mobile input.
color	See preview_mobile input.
landscape	See preview_mobile input.

f7Accordion

Framework7 accordion container

Description

f7Accordion creates an interactive accordion container.

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

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

Usage

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

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

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

Arguments

...	Item content such as f7Block or any f7 element.
id	Accordion instance.
multiCollapse	Whether to open multiple items at the same time. FALSE by default.
title	Item title.
open	Whether the item is open at start. FALSE by default.
selected	Index of item to select.
session	Shiny session object

Author(s)

David Granjon, <dgranjon@gmail.com>

Examples

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

  shinyApp(
    ui = f7Page(
      title = "Accordions",
      f7SingleLayout(
        navbar = f7Navbar("Accordions"),
        f7Accordion(
          id = "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
          )
        )
      })
    }
  )
}

# Update accordion
if (interactive()) {
```

```
library(shiny)
library(shinyMobile)

shinyApp(
  ui = f7Page(
    title = "Accordions",
    f7SingleLayout(
      navbar = f7Navbar("Accordions"),
      f7Button(inputId = "go", "Go"),
      f7Accordion(
        id = "myaccordion1",
        f7AccordionItem(
          title = "Item 1",
          f7Block("Item 1 content"),
          open = TRUE
        ),
        f7AccordionItem(
          title = "Item 2",
          f7Block("Item 2 content")
        )
      )
    )
  ),
  server = function(input, output, session) {

    observeEvent(input$go, {
      updateF7Accordion(id = "myaccordion1", selected = 2)
    })

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

Description

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

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

Usage

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

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

Arguments

id	Unique id. This gives the state of the action sheet. <code>input\$id</code> is TRUE when opened and inversely. Importantly, if the action sheet has never been opened, <code>input\$id</code> is NULL.
buttons	list of buttons such as <pre>buttons <- list(list(text = "Notification", icon = f7Icon("info"), color = NULL), list(text = "Dialog", icon = f7Icon("lightbulb_fill"), color = NULL))</pre> <p>The currently selected button may be accessed via <code>input\$<sheet_id>_button</code>. The value is numeric. When the action sheet is closed, <code>input\$<sheet_id>_button</code> is NULL. This is useful when you want to trigger events after a specific button click.</p>
grid	Whether to display buttons on a grid. Default to FALSE.
...	Other options. See https://v5.framework7.io/docs/action-sheet.html#action-sheet-parameters .
session	Shiny session object.
options	Other options. See https://v5.framework7.io/docs/action-sheet.html#action-sheet-parameters .

Examples

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

```

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

    observe({
      print(list(
        sheetOpen = input$action1,
        button = input$action1_button
      ))
    })

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

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

    observeEvent(input$go, {
      f7ActionSheet(
        grid = TRUE,
        id = "action1",
        buttons = list(
          list(
            text = "Notification",
            icon = f7Icon("info"),
            color = NULL
          ),
          list(
            text = "Dialog",
            icon = f7Icon("lightbulb_fill"),

```

```

        color = NULL
      )
    )
  }
})
}
)

### in shiny module
library(shiny)
library(shinyMobile)

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

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

  ns <- session$ns

  observe({
    print(list(
      sheetOpen = input$action1,
      button = input$action1_button
    ))
  })

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

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

  observeEvent(input$go, {
    f7ActionSheet(
      grid = TRUE,

```

```

      id = ns("action1"),
      buttons = list(
        list(
          text = "Notification",
          icon = f7Icon("info"),
          color = NULL
        ),
        list(
          text = "Dialog",
          icon = f7Icon("lightbulb_fill"),
          color = NULL
        )
      )
    )
  })
}

shinyApp(
  ui = f7Page(
    title = "Action sheet",
    f7SingleLayout(
      navbar = f7Navbar("Action sheet"),
      br(),
      sheetModuleUI(id = "sheet1")
    )
  ),
  server = function(input, output, session) {
    callModule(sheetModule, "sheet1")
  }
)
}
if (interactive()) {
  library(shiny)
  library(shinyMobile)

  shinyApp(
    ui = f7Page(
      title = "Update Action sheet",
      f7SingleLayout(
        navbar = f7Navbar("Update Action sheet"),
        br(),
        f7Segment(
          f7Button(inputId = "go", label = "Show action sheet", color = "green"),
          f7Button(inputId = "update", label = "Update action sheet", color = "red")
        )
      )
    ),
    server = function(input, output, session) {

      observe({
        print(list(
          sheetOpen = input$action1,
          button = input$action1_button
        ))
      })
    }
  )
}

```


Arguments

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

Author(s)

David Granjon, <dgranjon@gmail.com>

Examples

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

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

f7AppBar

Framework7 appBar

Description

f7AppBar is displayed on top of an [f7Navbar](#). [f7AppBar](#) can also trigger [f7Panel](#).

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

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

Usage

```
f7AppBar(..., leftPanel = FALSE, rightPanel = FALSE)
```

```
f7Back(targetId)
```

```
f7Next(targetId)
```

Arguments

...	Any UI content such as f7Searchbar , f7Next and f7Back . It is best practice to wrap f7Next and f7Back in an f7Flex .
leftPanel	Whether to enable the left panel. FALSE by default.
rightPanel	Whether to enable the right panel. FALSE by default.
targetId	f7Tabs id.

Examples

```

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

  cities <- names(precip)

  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("envelope"),
            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

Framework7 autocomplete input

Description

f7AutoComplete generates a Framework7 autocomplete input.

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

Usage

```

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

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

```

Arguments

inputId	The id of the input object.
label	Autocomplete label.
placeholder	Text to write in the container.
value	New value.
choices	Autocomplete choices.
openIn	Defines how to open Autocomplete, can be page or popup (for Standalone) or dropdown.

typeahead	Enables type ahead, will prefill input value with first item in match. Only if openIn is "dropdown".
expandInput	If TRUE then input which is used as item-input in List View will be expanded to full screen wide during dropdown visible. Only if openIn is "dropdown".
closeOnSelect	Set to true and autocomplete will be closed when user picks value. Not available if multiple is enabled. Only works when openIn is 'popup' or 'page'.
dropdownPlaceholderText	Specify dropdown placeholder text. Only if openIn is "dropdown".
multiple	Whether to allow multiple value selection. Only works when openIn is 'popup' or 'page'.
session	The Shiny session object.

Note

You cannot update choices yet.

Author(s)

David Granjon, <dgranjon@gmail.com>

Examples

```
# Autocomplete input
if(interactive()){
  library(shiny)
  library(shinyMobile)

  shinyApp(
    ui = f7Page(
      title = "My app",
      f7SingleLayout(
        navbar = f7Navbar(title = "f7AutoComplete"),
        f7AutoComplete(
          inputId = "myautocomplete1",
          placeholder = "Some text here!",
          dropdownPlaceholderText = "Try to type Apple",
          label = "Type a fruit name",
          openIn = "dropdown",
          choices = c('Apple', 'Apricot', 'Avocado', 'Banana', 'Melon',
                     'Orange', 'Peach', 'Pear', 'Pineapple')
        ),
        textOutput("autocompleteval1"),
        f7AutoComplete(
          inputId = "myautocomplete2",
          placeholder = "Some text here!",
          openIn = "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)
}
)
}

# Update autocomplete
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!",
            openIn = "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(
          inputId = "myautocomplete",
          value = "Banana"
        )
      })
    }
  )
}

```

```
)  
}
```

f7Badge

Framework7 badge

Description

Container to highlight important information with color.

Usage

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

Arguments

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

Author(s)

David Granjon, <dgranjon@ymail.com>

Examples

```
if(interactive()){  
  library(shiny)  
  library(shinyMobile)  
  
  colors <- getF7Colors()  
  
  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

Framework7 block

Description

f7Block creates a block container.

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

Usage

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

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

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.
text	Any text.

Author(s)

David Granjon, <dgranjon@ymail.com>

Examples

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

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

```

    f7BlockFooter(text = "Footer")
  ),

  f7BlockTitle(title = "A medium title", size = "medium"),
  f7Block(
    strong = TRUE,
    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

Framework7 block footer

Description

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

Usage

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

Arguments

text Any text.

Author(s)

David Granjon, <dgranjon@ymail.com>

f7BlockTitle

Framework7 block title

Description

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

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*Framework7 action button*

Description

f7Button generates a Framework7 action button.

updateF7Button updates an [f7Button](#).

Usage

```
f7Button(  
  inputId = NULL,  
  label = NULL,  
  href = NULL,  
  color = NULL,  
  fill = TRUE,  
  outline = FALSE,  
  shadow = FALSE,  
  rounded = FALSE,  
  size = NULL,  
  active = FALSE  
)  
  
updateF7Button(  
  inputId,  
  label = NULL,  
  color = NULL,  
  fill = NULL,  
  outline = NULL,  
  shadow = NULL,  
  rounded = NULL,  
  size = NULL,  
  session = shiny::getDefaultReactiveDomain()  
)
```

Arguments

inputId	The input slot that will be used to access the value.
label	The contents of the button or link—usually a text label, but you could also use any other HTML, like an image or f7Icon .
href	Button link.
color	Button color. Not compatible with outline. See here for valid colors https://framework7.io/docs/badge.html .
fill	Fill style. TRUE by default. Not compatible with outline
outline	Outline style. FALSE by default. Not compatible with fill.

shadow	Button shadow. FALSE by default. Only for material design.
rounded	Round style. FALSE by default.
size	Button size. NULL by default but also "large" or "small".
active	Button active state. Default to FALSE. This is useful when used in f7Segment with the strong parameter set to TRUE.
session	The Shiny session object, usually the default value will suffice.

Author(s)

David Granjon, <dgranjon@ymail.com>

Examples

```

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

  shiny::shinyApp(
    ui = f7Page(
      title = "Update f7Button",
      f7SingleLayout(
        navbar = f7Navbar(title = "Update f7Button"),
        f7Button(
          "test",
          "Test",
          color = "orange",
          outline = FALSE,
          fill = TRUE,
          shadow = FALSE,
          rounded = FALSE,
          size = NULL),
        f7Toggle("prout", "Update Button")
      )
    ),
    server = function(input, output, session) {
      observe(print(input$test))
      observeEvent(input$prout, {
        if (input$prout) {
          updateF7Button(
            inputId = "test",
            label = "Updated",
            color = "purple",
            shadow = TRUE,
            rounded = TRUE,
            size = "large"
          )
        }
      })
    }
  )
}

```

f7Card

Framework7 card

Description

f7Card creates a simple card container.

f7SocialCard is a special card for social content.

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

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

Usage

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

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

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

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

Arguments

...	Card content.
image	Card background image url. Tje JPG format is preferred. Not compatible with the color argument.
title	Card title.
footer	Footer content, if any. Must be wrapped in a tagList.
outline	Outline style. FALSE by default.
height	Card height. NULL by default.

author	Author.
date	Date.
id	Card id.
subtitle	Card subtitle.
color	Card background color. See https://framework7.io/docs/cards.html . Not compatible with the <code>img</code> argument.
fullBackground	Whether the image should cover the entire card.
session	Shiny session object.

Note

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

Author(s)

David Granjon, <dgranjon@gmail.com>

Examples

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

  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"),
            f7Badge("Badge", color = "green")
          )
        )
      ),
    f7Card(
      title = "Card header",
      image = "https://loremflickr.com/320/240",
      "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"),
```

```

        f7Badge("Badge", color = "green")
      )
    )
  ),
  server = function(input, output) {}
)
}

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

  shinyApp(
    ui = f7Page(
      title = "Social Card",
      f7SingleLayout(
        navbar = f7Navbar(title = "f7SocialCard"),
        f7SocialCard(
          image = "https://loremflickr.com/g/320/240/paris",
          author = "John Doe",
          date = "Monday at 3:47 PM",
          "What a nice photo i took yesterday!",
          img(src = "https://loremflickr.com/g/320/240/paris", width = "100%"),
          footer = tagList(
            f7Badge("1", color = "yellow"),
            f7Badge("2", color = "green"),
            f7Badge("3", color = "blue")
          )
        )
      )
    ),
    server = function(input, output) {}
  )
}

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

  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",
    image = "https://i.pinimg.com/originals/73/38/6e/73386e0513d4c02a4fbb814cadfba655.jpg",
    "Framework7 - is a free and open source HTML mobile framework
    to develop hybrid mobile apps or web apps with iOS or Android
    native look and feel. It is also an indispensable prototyping apps tool
    to show working app prototype as soon as possible in case you need to."
  ),
  f7ExpandableCard(
    id = "card4",
    title = "Expandable Card 4",
    fullBackground = TRUE,
    image = "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) {}
}

# Update expandable card
if (interactive()) {
  library(shiny)
  library(shinyMobile)

  shinyApp(
    ui = f7Page(
      title = "Expandable Cards",
      f7SingleLayout(
        navbar = f7Navbar(
          title = "Expandable Cards",

```

```

        hairline = FALSE,
        shadow = TRUE
    ),
    f7ExpandableCard(
        id = "card1",
        title = "Expandable Card 1",
        image = "http://i.pining.com/originals/73/38/6e/73386e0513d4c02a4fbb814cadfba655.jpg",
        "Framework7 - is a free and open source HTML mobile framework
        to develop hybrid mobile apps or web apps with iOS or Android
        native look and feel. It is also an indispensable prototyping apps tool
        to show working app prototype as soon as possible in case you need to."
    ),

    hr(),
    f7BlockTitle(title = "Click below to expand the card!") %>% f7Align(side = "center"),
    f7Button(inputId = "go", label = "Go"),
    br(),
    f7ExpandableCard(
        id = "card2",
        title = "Expandable Card 2",
        fullBackground = TRUE,
        image = "http://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")
    })

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

```

f7Checkbox

Framework7 checkbox

Description

[f7Checkbox](#) creates a checkbox input.

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

Usage

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

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

Arguments

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.
session	The Shiny session object.

Examples

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

  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})
    }
  )
}

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

shinyApp(ui, server)
}

```

f7CheckboxGroup

Framework7 checkbox group

Description

f7CheckboxGroup creates a 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

Framework7 chips

Description

f7Chip is an improved badge container.

Usage

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

```

    status = NULL,
    iconStatus = NULL,
    closable = FALSE
  )

```

Arguments

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

Author(s)

David Granjon, <dgranjon@gmail.com>

Examples

```

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

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

```

f7Col	<i>Framework7 column container</i>
-------	------------------------------------

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@gmail.com>

f7ColorPicker	<i>Create a Framework7 color picker input</i>
---------------	---

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

<code>inputId</code>	Color picker input.
<code>label</code>	Color picker label.
<code>value</code>	Color picker value. hex, rgb, hsl, hsb, alpha, hue, rgba, hsla are supported.
<code>placeholder</code>	Color picker placeholder.
<code>modules</code>	Picker color modules. Choose at least one.
<code>palettes</code>	Picker color predefined palettes. Must be a list of color vectors, each value specified as HEX string.
<code>sliderValue</code>	When enabled, it will display sliders values.
<code>sliderValueEditable</code>	When enabled, it will display sliders values as <code><input></code> elements to edit directly.
<code>sliderLabel</code>	When enabled, it will display sliders labels with text.
<code>hexLabel</code>	When enabled, it will display HEX module label text, e.g. HEX.
<code>hexValueEditable</code>	When enabled, it will display HEX module value as <code><input></code> element to edit directly.
<code>groupedModules</code>	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	<i>Framework7 date picker</i>
--------------	-------------------------------

Description

f7DatePicker creates a Framework7 date picker input.

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

Usage

```
f7DatePicker(  
  inputId,  
  label,  
  value = NULL,  
  multiple = FALSE,  
  direction = c("horizontal", "vertical"),  
  minDate = NULL,  
  maxDate = NULL,  
  dateFormat = "yyyy-mm-dd",  
  openIn = c("auto", "popover", "sheet", "customModal"),  
  scrollToInput = FALSE,  
  closeByOutsideClick = TRUE,  
  toolbar = TRUE,  
  toolbarCloseText = "Done",  
  header = FALSE,  
  headerPlaceholder = "Select date"  
)  
  
updateF7DatePicker(  
  inputId,  
  value = NULL,  
  ...,  
  session = shiny::getDefaultReactiveDomain()  
)
```

Arguments

inputId	The id of the input object.
label	Input label.
value	The new value for the input.
multiple	If TRUE allow to select multiple dates.
direction	Months layout direction, could be 'horizontal' or 'vertical'.
minDate	Minimum allowed date.
maxDate	Maximum allowed date.

dateFormat	Date format: "yyyy-mm-dd", for instance.
openIn	Can be auto, popover (to open calendar in popover), sheet (to open in sheet modal) or customModal (to open in custom Calendar modal overlay). In case of auto will open in sheet modal on small screens and in popover on large screens.
scrollToInput	Scroll viewport (page-content) to input when calendar opened.
closeByOutsideClick	If enabled, picker will be closed by clicking outside of picker or related input element.
toolbar	Enables calendar toolbar.
toolbarCloseText	Text for Done/Close toolbar button.
header	Enables calendar header.
headerPlaceholder	Default calendar header placeholder text.
...	Parameters used to update the date picker, use same arguments as in f7DatePicker .
session	The Shiny session object, usually the default value will suffice.

Value

a Date vector.

Examples

```
# Date picker
if (interactive()) {
  library(shiny)
  library(shinyMobile)

  shinyApp(
    ui = f7Page(
      title = "My app",
      f7SingleLayout(
        navbar = f7Navbar(title = "f7DatePicker"),
        f7DatePicker(
          inputId = "date",
          label = "Choose a date",
          value = "2019-08-24"
        ),
        "The selected date is",
        verbatimTextOutput("selectDate"),
        f7DatePicker(
          inputId = "multipleDates",
          label = "Choose multiple dates",
          value = Sys.Date() + 0:3,
          multiple = TRUE
        ),
        "The selected date is",
        verbatimTextOutput("selectMultipleDates"),
        f7DatePicker(
```



```

        inputId = "default",
        label = "Choose a date",
        value = NULL
      ),
      "The selected date is",
      verbatimTextOutput("selectDefault")
    )
  ),
  server = function(input, output, session) {

    output$selectDate <- renderPrint(input$date)
    output$selectMultipleDates <- renderPrint(input$multipleDates)
    output$selectDefault <- renderPrint(input$default)

  }
}
# Update date picker
if (interactive()) {
  library(shiny)
  library(shinyMobile)

  shinyApp(
    ui = f7Page(
      title = "My app",
      f7SingleLayout(
        navbar = f7Navbar(title = "Update date picker"),
        f7Card(
          f7Button(inputId = "selectToday", label = "Select today"),
          f7Button(inputId = "rmToolbar", label = "Remove toolbar"),
          f7Button(inputId = "addToolbar", label = "Add toolbar"),
          f7DatePicker(
            inputId = "mypicker",
            label = "Choose a date",
            value = Sys.Date() - 7,
            openIn = "auto",
            direction = "horizontal"
          ),
          verbatimTextOutput("pickerval")
        )
      )
    ),
    server = function(input, output, session) {

      output$pickerval <- renderPrint(input$mypicker)

      observeEvent(input$selectToday, {
        updateF7DatePicker(
          inputId = "mypicker",
          value = Sys.Date()
        )
      })
    }
  )
}

```

```

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

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

```

f7Dialog

Framework7 dialog window

Description

f7Dialog generates a modal window.

Usage

```

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

```

Arguments

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

Examples

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

# confirm alert
if (interactive()) {
  library(shiny)
  library(shinyMobile)
  shinyApp(
    ui = f7Page(
      title = "Confirm Dialog",
      f7SingleLayout(
        navbar = f7Navbar(title = "f7Dialog"),
        f7Button(inputId = "goButton", "Go!")
      )
    ),
    server = function(input, output, session) {

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

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

```

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

      observe({
        print(input$prompt)
      })

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

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

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

      observe({
        print(input$login)
      })
    }
  )
}

```

```

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

output$ui <- renderUI({
  req(input$login$user == "David" & input$login$password == "prout")
  img(src = "https://media2.giphy.com/media/12gfl8Xxrhv7C1fXiV/giphy.gif")
})
}
)
}

```

f7DownloadButton	<i>Create a download button</i>
------------------	---------------------------------

Description

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

Usage

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

Arguments

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

Examples

```

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

```

```

server = function(input, output, session) {
  # Our dataset
  data <- mtcars

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

shinyApp(ui, server)
}

```

f7Fab

Framework7 floating action button (FAB)

Description

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

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

Usage

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

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

Arguments

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

Author(s)

David Granjon, <dgranjon@gmail.com>

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("trigger", label = "Don't click me")
    })
  }
  shinyApp(ui, server)
}
```

f7FabClose

Framework7 FAB close

Description

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

Usage

```
f7FabClose(tag)
```

Arguments

tag Target tag.

f7Fabs

Framework7 container for floating action button (FAB)

Description

f7Fabs hosts multiple [f7Fab](#).

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

f7FabMorphTarget convert a tag into a target morphing. See <https://v5.framework7.io/docs/floating-action-button.html#fab-morph>.

Usage

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

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

```
f7FabMorphTarget(tag)
```

Arguments

...	Slot for f7Fab .
id	The id of the input object.
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 transform into another UI element.
morphTarget	CSS selector of the morph target: ". toolbar" for instance.
session	The Shiny session object, usually the default value will suffice.
tag	Target tag.

Note

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

Author(s)

David Granjon, <dgranjon@ymail.com>

Examples

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



```

shinyApp(
  ui = f7Page(
    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)]]))
      })
    }
  )
}

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

  shinyApp(
    ui = f7Page(
      title = "Update f7Fabs",
      f7SingleLayout(
        navbar = f7Navbar(title = "Update f7Fabs"),
        f7Button(inputId = "toggleFabs", label = "Toggle Fabs"),
        f7Fabs(
          position = "center-center",

```

```

        id = "fabs",
        lapply(1:3, function(i) f7Fab(inputId = i, label = i))
      )
    )
  ),
  server = function(input, output, session) {
    observe(print(input$fabs))
    observeEvent(input$toggleFabs, {
      updateF7Fabs(id = "fabs")
    })
  }
)
}
if (interactive()) {
  library(shiny)
  library(shinyMobile)

  shinyApp(
    ui = f7Page(
      f7SingleLayout(
        navbar = f7Navbar(title = "f7Fabs Morph"),
        toolbar = f7Toolbar(
          position = "bottom",
          lapply(1:3, function(i) f7Link(href = 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) {}
  )
}

```

f7File

File Upload Control

Description

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

Usage

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

Arguments

<code>inputId</code>	The input slot that will be used to access the value.
<code>label</code>	Display label for the control, or <code>NULL</code> for no label.
<code>multiple</code>	Whether the user should be allowed to select and upload multiple files at once. Does not work on older browsers, including Internet Explorer 9 and earlier.
<code>accept</code>	A character vector of MIME types; gives the browser a hint of what kind of files the server is expecting.
<code>width</code>	The width of the input, e.g. 400px, or 100%.
<code>buttonLabel</code>	The label used on the button. Can be text or an HTML tag object.
<code>placeholder</code>	The text to show before a file has been uploaded.

Examples

```
if (interactive()) {  
  library(shiny)  
  library(shinyMobile)  
  
  ui = f7Page(  
    f7SingleLayout(  
      navbar = f7Navbar(title = "File handling"),  
      f7File("up", "Upload!")  
    )  
  )  
  
  server = function(input, output) {  
    data <- reactive(input$up)  
    observe(print(data()))  
  }  
  
  shinyApp(ui, server)  
}
```

f7Flex

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)

  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	<i>Framework7 float utility</i>
---------	---------------------------------

Description

f7Float is an alignment utility for items.

Usage

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

Arguments

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

Author(s)

David Granjon, <dgranjon@ymail.com>

Examples

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

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

f7Found	<i>Utility to display an item when the search is successful.</i>
---------	--

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	<i>Launch the shinyMobile Gallery</i>
-----------	---------------------------------------

Description

A gallery of all components available in shinyMobile.

Usage

```
f7Gallery()
```

Examples

```
if (interactive()) {  
  f7Gallery()  
}
```

Description

f7Gauge creates a gauge instance.

updateF7Gauge updates a framework7 gauge from the server side.

Usage

```
f7Gauge(  
  id,  
  type = "circle",  
  value,  
  size = 200,  
  bgColor = "transparent",  
  borderBgColor = "#eeeeee",  
  borderColor = "#000000",  
  borderWidth = "10",  
  valueText = NULL,  
  valueTextColor = "#000000",  
  valueFontSize = "31",  
  valueFontWeight = "500",  
  labelText = NULL,  
  labelTextColor = "#888888",  
  labelTextSize = "14",  
  labelTextWeight = "400"  
)  
  
updateF7Gauge(  
  id,  
  value = NULL,  
  labelText = NULL,  
  size = NULL,  
  bgColor = NULL,  
  borderBgColor = NULL,  
  borderColor = NULL,  
  borderWidth = NULL,  
  valueText = NULL,  
  valueTextColor = NULL,  
  valueFontSize = NULL,  
  valueFontWeight = NULL,  
  labelTextColor = NULL,  
  labelTextSize = NULL,  
  labelTextWeight = NULL,  
  session = shiny::getDefaultReactiveDomain()  
)
```

Arguments

id	Gauge id.
type	Gauge type. Can be "circle" or "semicircle". Default is "circle."
value	New value. Numeric between 0 and 100.
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.
valueText	Gauge value text (large text in the center of gauge).
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.
session	Shiny session object.

Author(s)

David Granjon <dgranjon@ymail.com>

Examples

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

  shinyApp(
    ui = f7Page(
      title = "Gauges",
      f7SingleLayout(
        navbar = f7Navbar(title = "f7Gauge"),
        f7Block(
          f7Gauge(
            id = "mygauge",
            type = "semicircle",
            value = 50,
            borderColor = "#2196f3",
            borderWidth = 10,
```



```

        valueFontSize = 41,
        valueTextColor = "#2196f3",
        labelText = "amount of something"
      )
    )
  ),
  server = function(input, output) {}
)
}

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

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

```

f7HideOnEnable

Utility to hide a given tag when [f7Searchbar](#) is enabled.

Description

Use with [f7Searchbar](#).

Usage

```
f7HideOnEnable(tag)
```

Arguments

tag	tag to hide.
-----	--------------

f7HideOnSearch	<i>Utility to hide a given tag on search</i>
----------------	--

Description

Use with [f7Searchbar](#).

Usage

```
f7HideOnSearch(tag)
```

Arguments

tag	tag to hide.
-----	--------------

f7Icon	<i>Framework7 icons</i>
--------	-------------------------

Description

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

Usage

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

Arguments

...	Icon name and f7Badge .
lib	Library to use: NULL, "ios" or "md". Leave NULL by default. Specify, md or ios if you want to hide/show icons on specific devices.
color	Icon color, if any.
style	CSS styles to be applied on icon, for example use font-size: 56px; to have a bigger icon.
old	Deprecated. This was to handle old and new icons but shinyMobile only uses new icons from now. This parameter will be removed in a future release.

Author(s)

David Granjon, <dgranjon@gmail.com>

Examples

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

  shinyApp(
    ui = f7Page(
      title = "Icons",
      f7SingleLayout(
        navbar = f7Navbar(title = "icons"),
        f7List(
          f7ListItem(
            title = tagList(
              f7Icon("envelope")
            )
          ),
          f7ListItem(
            title = tagList(
              f7Icon("envelope_fill", color = "green")
            )
          ),
          f7ListItem(
            title = f7Icon("home", f7Badge("1", color = "red"))
          ),
          f7ListItem(
            title = f7Icon("envelope", lib = "md"),
            "This will not appear since only for material design"
          )
        )
      )
    ),
    server = function(input, output) {}
  )
}
```

f7Item

Framework7 body item

Description

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

Usage

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

Arguments

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

Author(s)

David Granjon, <dgranjon@ymail.com>

f7Items *Framework7 item container*

Description

Build a Framework7 wrapper for [f7Item](#)

Usage

f7Items(...)

Arguments

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

Author(s)

David Granjon, <dgranjon@ymail.com>

f7Link *Framework7 link*

Description

Link to point toward external content.

Usage

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

Arguments

label Optional link text.
 href Link source, url.
 icon Link icon, if any. Must pass [f7Icon](#).

Author(s)

David Granjon, <dgranjon@gmail.com>

Examples

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

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

f7List

Create a framework 7 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)

  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)
  shinyApp(
    ui = f7Page(
      title = "List Index",
      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, before-end, and inside. 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,
  href = 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.
href	Item external link.
media	Expect f7Icon or <code>img</code> .
right	Right content if any.

f7Login

*Framework7 login screen***Description**

Provide a template for authentication

f7LoginServer is a useful server elements to fine tune the [f7Login](#) page.

updateF7Login toggles a login page.

Usage

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

```
f7LoginServer(input, output, session, ignoreInit = FALSE, trigger = NULL)
```

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

Arguments

...	Slot for inputs like password, text, ...
id	f7Login unique id.
title	Login page title.
label	Login confirm button label.
footer	Optional footer.
startOpen	Whether to open the login page at start. Default to TRUE. There are some cases where it is interesting to set up to FALSE, for instance when you want to have authentication only in a specific tab of your app (See example 2).

input	Shiny input object.
output	Shiny output object.
session	Shiny session object.
ignoreInit	If TRUE, then, when this observeEvent is first created/initialized, ignore the handlerExpr (the second argument), whether it is otherwise supposed to run or not. The default is FALSE.
trigger	Reactive trigger to toggle the login page state. Useful, when one wants to set up local authentication (for a specific section). See example 2.
user	Value of the user input.
password	Value of the password input.

Details

This function does not provide the backend features. You would need to store credentials in a database for instance.

Note

There is an input associated with the login status, namely `input$login`. It is linked to an action button, which is 0 when the application starts. As soon as the button is pressed, its value is incremented which might fire a `observeEvent` listening to it (See example below). Importantly, the login page is closed only if the text and password inputs are filled. The `f7LoginServer` contains a piece of server logic that does this work for you.

Examples

```
if (interactive()) {
  # global authentication
  library(shiny)
  library(shinyMobile)
  shinyApp(
    ui = f7Page(
      title = "Login module",
      f7SingleLayout(
        navbar = f7Navbar(
          title = "Login Example",
          hairline = FALSE,
          shadow = TRUE
        ),
        toolbar = f7Toolbar(
          position = "bottom",
          f7Link(label = "Link 1", href = "https://www.google.com"),
          f7Link(label = "Link 2", href = "https://www.google.com")
        ),
        f7Login(id = "loginPage", title = "Welcome"),
        # main content
        f7BlockTitle(
          title = HTML(paste("Welcome", textOutput("user"))),
          size = "large"
        )
      )
    )
  )
}
```

```

    ) %>% f7Align("center")
  )
),
server = function(input, output, session) {

  loginData <- callModule(f7LoginServer, id = "loginPage")

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

# section specific authentication
library(shiny)
library(shinyMobile)
shinyApp(
  ui = f7Page(
    title = "Local access restriction",
    f7TabLayout(
      navbar = f7Navbar(
        title = "Login Example for Specific Section",
        hairline = FALSE,
        shadow = TRUE
      ),
      f7Tabs(
        id = "tabs",
        f7Tab(
          tabName = "Tab 1",
          "Without authentication"
        ),
        f7Tab(
          tabName = "Restricted",
          # main content
          f7BlockTitle(
            title = HTML(paste("Welcome", textOutput("user"))),
            size = "large"
          ) %>% f7Align("center")
        )
      ),
      f7Login(id = "loginPage", title = "Welcome", startOpen = FALSE)
    )
  ),
  server = function(input, output, session) {

    # trigger
    trigger <- reactive({
      req(input$tabs)
    })

    # do not run first since the login page is not yet visible
    loginData <- callModule(

```

```

    f7LoginServer,
    id = "loginPage",
    ignoreInit = TRUE,
    trigger = trigger
  )

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

# with 2 different protected sections
library(shiny)
library(shinyMobile)
shinyApp(
  ui = f7Page(
    title = "Multiple restricted areas",
    f7TabLayout(
      navbar = f7Navbar(
        title = "Login Example for 2 Specific Section",
        hairline = FALSE,
        shadow = TRUE
      ),
      f7Tabs(
        id = "tabs",
        f7Tab(
          tabName = "Tab 1",
          "Without authentication"
        ),
        f7Tab(
          tabName = "Restricted",
          # main content
          f7BlockTitle(
            title = "1st restricted area",
            size = "large"
          ) %>% f7Align("center")
        ),
        f7Tab(
          tabName = "Restricted 2",
          # main content
          f7BlockTitle(
            title = "2nd restricted area",
            size = "large"
          ) %>% f7Align("center")
        )
      ),
      f7Login(id = "loginPage", title = "Welcome", startOpen = FALSE),
      f7Login(id = "loginPage2", title = "Welcome", startOpen = FALSE)
    )
  ),

```

```
server = function(input, output, session) {  
  
  trigger1 <- reactive({  
    req(input$tabs == "Restricted")  
  })  
  
  trigger2 <- reactive({  
    req(input$tabs == "Restricted 2")  
  })  
  
  # do not run first since the login page is not yet visible  
  callModule(  
    f7LoginServer,  
    id = "loginPage",  
    ignoreInit = TRUE,  
    trigger = trigger1  
  )  
  
  callModule(  
    f7LoginServer,  
    id = "loginPage2",  
    ignoreInit = TRUE,  
    trigger = trigger2  
  )  
  
}  
)  
}
```

f7Margin

Framework7 margin utility

Description

f7Margin adds a margin to the given tag.

Usage

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

Arguments

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

Author(s)

David Granjon, <dgranjon@ymail.com>

Examples

```

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

  cardTag <- f7Card(
    title = "Card header",
    "This is a simple card with plain text,
    but cards can also contain their own header,
    footer, list view, image, or any other element.",
    footer = tagList(
      f7Button(color = "blue", label = "My button", href = "https://www.google.com"),
      f7Badge("Badge", color = "green")
    )
  )

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

```

f7Menu*Framework7 menu container*

Description

f7Menu is a container for [f7MenuItem](#) and/or [f7MenuDropdown](#).

f7MenuItem creates a special action button for [f7Menu](#).

f7MenuDropdown creates a dropdown menu for [f7Menu](#).

f7MenuDropdownDivider creates a dropdown divider for [f7MenuDropdown](#).

updateF7MenuDropdown toggles [f7MenuDropdown](#) on the client.

Usage

```
f7Menu(...)
```

```
f7MenuItem(inputId, label)
```

```
f7MenuDropdown(..., id = NULL, label, side = c("left", "center", "right"))
```

```
f7MenuDropdownDivider()
```

```
updateF7MenuDropdown(id, session = shiny::getDefaultReactiveDomain())
```

Arguments

...	Slot for <code>f7MenuItem</code> and <code>f7MenuDropdownDivider</code> .
<code>inputId</code>	Menu item input id.
<code>label</code>	Button label.
<code>id</code>	Menu to target.
<code>side</code>	Dropdown opening side. Choose among <code>c("left", "center", "right")</code> .
<code>session</code>	Shiny session object.

Examples

```
# Menu container
if (interactive()) {
  library(shiny)
  library(shinyMobile)
  shinyApp(
    ui = f7Page(
      title = "Menus",
      f7SingleLayout(
        navbar = f7Navbar(
          title = "f7Menu",
          hairline = FALSE,
          shadow = TRUE
        ),
        f7Button(inputId = "toggle", label = "Toggle menu"),
        f7Menu(
          f7MenuDropdown(
            id = "menu1",
            label = "Menu 1",
            f7MenuItem(inputId = "item1", "Item 1"),
            f7MenuItem(inputId = "item2", "Item 2"),
            f7MenuDropdownDivider(),
            f7MenuItem(inputId = "item3", "Item 3")
          )
        )
      )
    ),
    server = function(input, output, session) {
      observeEvent(input$toggle, {
        updateF7MenuDropdown("menu1")
      })

      observeEvent(input$item1, {
        f7Notif(text = "Well done!")
      })
    }
  )
}
```



```
    observe({
      print(input$item1)
      print(input$menu1)
    })
  }
)
```

f7MessageBar

Framework7 message bar.

Description

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

updateF7MessageBar updates message bar content on the server side.

Usage

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

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

Arguments

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

Examples

```
if (interactive()) {
  library(shiny)
  library(shinyMobile)
  shinyApp(
    ui = f7Page(
      title = "Update message bar",
      f7SingleLayout(
        navbar = f7Navbar(
          title = "Message bar",
```

```

        hairline = FALSE,
        shadow = TRUE
    ),
    toolbar = f7Toolbar(
        position = "bottom",
        f7Link(label = "Link 1", href = "https://www.google.com"),
        f7Link(label = "Link 2", href = "https://www.google.com")
    ),
    # main content
    f7Segment(
        container = "segment",
        f7Button("updateMessageBar", "Update value"),
        f7Button("updateMessageBarPlaceholder", "Update placeholder")
    ),
    f7MessageBar(inputId = "mymessagebar", placeholder = "Message"),
    uiOutput("messageContent")
)
),
server = function(input, output, session) {

    output$messageContent <- renderUI({
        req(input$mymessagebar)
        tagList(
            f7BlockTitle("Message Content", size = "large"),
            f7Block(strong = TRUE, inset = TRUE, input$mymessagebar)
        )
    })

    observeEvent(input$updateMessageBar, {
        updateF7MessageBar(
            inputId = "mymessagebar",
            value = "sjsjsj"
        )
    })

    observeEvent(input$updateMessageBarPlaceholder, {
        updateF7MessageBar(
            inputId = "mymessagebar",
            placeholder = "Enter your message"
        )
    })
}
)
}

```

f7Messages

Framework7 messages container

Description

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

f7Message creates a message item to be inserted in `f7Messages` with `updateF7Messages`.
`updateF7Messages` add messages to an `f7Messages` container.

Usage

```
f7Messages(
  id,
  title = NULL,
  autoLayout = TRUE,
  newMessagesFirst = FALSE,
  scrollMessages = TRUE,
  scrollMessagesOnEdge = TRUE
)

f7Message(
  text,
  name,
  type = c("sent", "received"),
  header = NULL,
  footer = NULL,
  avatar = NULL,
  textHeader = NULL,
  textFooter = NULL,
  image = NULL,
  imageSrc = NULL,
  cssClass = NULL
)

updateF7Messages(
  id,
  messages,
  showTyping = FALSE,
  session = shiny::getDefaultReactiveDomain()
)
```

Arguments

<code>id</code>	Reference to <code>linkf7Messages</code> container.
<code>title</code>	Optional messages title.
<code>autoLayout</code>	Enable Auto Layout to add all required additional classes automatically based on passed conditions.
<code>newMessagesFirst</code>	Enable if you want to use new messages on top, instead of having them on bottom.
<code>scrollMessages</code>	Enable/disable messages auto scrolling when adding new message.
<code>scrollMessagesOnEdge</code>	If enabled then messages auto scrolling will happen only when user is on top/bottom of the messages view.

text	Message text.
name	Sender name.
type	Message type - sent or received.
header	Single message header.
footer	Single message footer.
avatar	Sender avatar URL string.
textHeader	Message text header.
textFooter	Message text footer.
image	Message image HTML string, e.g. <code></code> . Can be used instead of <code>imageSrc</code> parameter.
imageSrc	Message image URL string. Can be used instead of <code>image</code> parameter.
cssClass	Additional CSS class to set on message HTML element.
messages	List of f7Messages .
showTyping	Show typing when a new message comes. Default to FALSE. Does not work yet...
session	Shiny session object

Examples

```

if (interactive()) {
  library(shiny)
  library(shinyMobile)
  shinyApp(
    ui = f7Page(
      title = "Messages",
      f7SingleLayout(
        navbar = f7Navbar(
          title = "Messages",
          hairline = FALSE,
          shadow = TRUE
        ),
        toolbar = f7MessageBar(inputId = "mymessagebar", placeholder = "Message"),
        # main content
        f7Messages(id = "mymessages", title = "My message")
      )
    ),
    server = function(input, output, session) {
      observe({
        print(input[["mymessagebar-send"]])
        print(input$mymessages)
      })
      observeEvent(input[["mymessagebar-send"]], {
        updateF7Messages(
          id = "mymessages",
          list(
            f7Message(

```

```

        text = input$mymessagebar,
        name = "David",
        type = "sent",
        header = "Message Header",
        footer = "Message Footer",
        textHeader = "Text Header",
        textFooter = "text Footer",
        avatar = "https://cdn.framework7.io/placeholder/people-100x100-7.jpg"
    )
  )
}
})

observe({
  invalidateLater(5000)
  names <- c("Victor", "John")
  name <- sample(names, 1)

  updateF7Messages(
    id = "mymessages",
    list(
      f7Message(
        text = "Some message",
        name = name,
        type = "received",
        avatar = "https://cdn.framework7.io/placeholder/people-100x100-9.jpg"
      )
    )
  )
})

}
)
}

```

f7Navbar

Framework7 Navbar

Description

Build a navbar layout element to insert in [f7SingleLayout](#), [f7TabLayout](#) or [f7SplitLayout](#).
 updateF7Navbar toggles an [f7Navbar](#) component from the server.

Usage

```

f7Navbar(
  ...,
  subNavbar = NULL,
  title = NULL,
  subtitle = NULL,

```

```

    hairline = TRUE,
    shadow = TRUE,
    bigger = FALSE,
    transparent = FALSE,
    leftPanel = FALSE,
    rightPanel = FALSE
  )

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

```

Arguments

...	Slot for f7SearchbarTrigger . Not compatible with f7Panel .
subNavbar	f7SubNavbar slot, if any.
title	Navbar title.
subtitle	Navbar subtitle. Not compatible with <code>bigger</code> .
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 <code>subtitle</code> .
transparent	Whether the navbar should be transparent. FALSE by default. Only works if <code>bigger</code> is TRUE.
leftPanel	Whether to enable the left panel. FALSE by default.
rightPanel	Whether to enable the right panel. FALSE by default.
animate	Whether it should be hidden with animation or not. By default is TRUE.
hideStatusbar	When FALSE (default) it hides navbar partially keeping space required to cover statusbar area. Otherwise, navbar will be fully hidden.
session	Shiny session object.

Note

Currently, `bigger` parameters does mess with the CSS.

Author(s)

David Granjon, <dgranjon@gmail.com>

Examples

```

# Toggle f7Navbar
if (interactive()) {
  library(shiny)
  library(shinyMobile)

```

```
shinyApp(  
  ui = f7Page(  
    title = "Show navbar",  
    f7SingleLayout(  
      navbar = f7Navbar("Hide/Show navbar"),  
      f7Button(inputId = "toggle", "Toggle navbar", color = "red")  
    )  
  ),  
  server = function(input, output, session) {  
  
    observeEvent(input$toggle, {  
      updateF7Navbar()  
    })  
  }  
)
```

f7NotFound

Utility to display an item when the search is unsuccessful.

Description

Use with [f7Searchbar](#).

Usage

```
f7NotFound(tag)
```

Arguments

tag tag to use.

f7Notif

Framework7 notification

Description

Notification with title, text, icon and more.

Usage

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

Arguments

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

Examples

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

```



```
        title = "Notification",
        subtitle = "A subtitle",
        titleRightText = "now"
      )
    })
  }
}
```

f7Padding*Framework7 padding utility*

Description

f7Padding adds padding to the given tag.

Usage

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

Arguments

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

Author(s)

David Granjon, <dgranjon@gmail.com>

Examples

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

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

  shinyApp(
    ui = f7Page(
```

```

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

```

f7Page

Framework7 page container

Description

f7Page is the main app container.

Usage

```

f7Page(
  ...,
  title = NULL,
  preloader = FALSE,
  loading_duration = 3,
  options = list(theme = c("ios", "md", "auto", "aurora"), dark = TRUE, filled = FALSE,
    color = "#007aff", touch = list(tapHold = TRUE, tapHoldDelay = 750, iosTouchRipple =
      FALSE), iosTranslucentBars = FALSE, navbar = list(iosCenterTitle = TRUE,
        hideOnPageScroll = TRUE), toolbar = list(hideOnPageScroll = FALSE), pullToRefresh =
          FALSE),
  allowPWA = FALSE
)

```

Arguments

...	Slot for shinyMobile skeleton elements: f7AppBar , f7SingleLayout , f7TabLayout , f7SplitLayout .
title	Page title.
preloader	Whether to display a preloader before the app starts. FALSE by default.
loading_duration	Preloader duration.
options	shinyMobile configuration. See https://framework7.io/docs/app.html . Below are the most notable options. General options: <ul style="list-style-type: none"> • theme: App skin: "ios", "md", "auto" or "aurora". • dark: Dark layout. TRUE or FALSE. • filled: Whether to fill the f7Navbar and f7Toolbar with the current selected color. FALSE by default. • color: Color theme: See https://framework7.io/docs/color-themes.html. Expect a name like blue or red. If NULL, use the default color.

- `pullToRefresh`: Whether to active the pull to refresh feature. Default to `FALSE`. See <https://v5.framework7.io/docs/pull-to-refresh.html#examples>.
- `iosTranslucentBars`: Enable translucent effect (blur background) on navigation bars for iOS theme (on iOS devices). `FALSE` by default.

Touch module options <https://v5.framework7.io/docs/app.html#app-parameters>:

- `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.
- `tapHoldDelay`: Determines how long (in ms) the user must hold their tap before the taphold event is fired on the target element. Default to 750 ms.
- `iosTouchRipple`: Default to `FALSE`. Enables touch ripple effect for iOS theme.

Navbar options <https://v5.framework7.io/docs/navbar.html#navbar-app-parameters>:

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

Toolbar options <https://v5.framework7.io/docs/toolbar-tabbar.html#toolbar-app-parameters>:

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

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

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

Author(s)

David Granjon, <dgranjon@gmail.com>

f7Panel

Framework7 panel

Description

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

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

Usage

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

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

Arguments

...	Panel content. Slot for f7PanelMenu , if used as a sidebar.
id	Panel unique id.
title	Panel title.
side	Panel side: "left" or "right".
theme	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.
session	Shiny session object.

Author(s)

David Granjon, <dgranjon@gmail.com>

Examples

```
if (interactive()) {
  library(shiny)
  library(shinyMobile)
  shinyApp(
    ui = f7Page(
      title = "Panels",
      f7SingleLayout(
        navbar = f7Navbar(
          title = "Single Layout",
          hairline = FALSE,
          shadow = TRUE,
          leftPanel = TRUE,
          rightPanel = TRUE
        ),
        panels = tagList(
          f7Panel(side = "left", id = "mypanel1"),
          f7Panel(side = "right", id = "mypanel2")
        )
      )
    )
  }
```

```

toolbar = f7Toolbar(
  position = "bottom",
  icons = TRUE,
  hairline = FALSE,
  shadow = FALSE,
  f7Link(label = "Link 1", href = "https://www.google.com"),
  f7Link(label = "Link 2", href = "https://www.google.com")
),
# 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", href = "https://www.google.com"),
      f7Badge("Badge", color = "green")
    )
  )
)
),
server = function(input, output, session) {

  observeEvent(input$mypanel2, {

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

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

  output$distPlot <- renderPlot({
    if (input$mypanel1) {
      dist <- rnorm(input$obs)
      hist(dist)
    }
  })
}
}
# Toggle panel
if (interactive()) {
  library(shiny)
  library(shinyMobile)
  shinyApp(

```

```

ui = f7Page(
  title = "Update panel menu",
  f7SingleLayout(
    navbar = f7Navbar(
      title = "Single Layout",
      hairline = FALSE,
      shadow = TRUE,
      leftPanel = TRUE,
      rightPanel = TRUE
    ),
    panels = tagList(
      f7Panel(side = "left", id = "mypanel1", theme = "light", effect = "cover"),
      f7Panel(side = "right", id = "mypanel2", theme = "light")
    ),
    toolbar = f7Toolbar(
      position = "bottom",
      icons = TRUE,
      hairline = FALSE,
      shadow = FALSE,
      f7Link(label = "Link 1", href = "https://www.google.com"),
      f7Link(label = "Link 2", href = "https://www.google.com")
    )
  )
),
server = function(input, output, session) {

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

  observe({
    invalidateLater(2000)
    updateF7Panel(id = "mypanel1")
  })

}
}

```

f7PanelMenu

Framework7 sidebar menu

Description

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

f7PanelItem creates a Framework7 sidebar menu item for [f7SplitLayout](#).

Usage

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

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

Arguments

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

f7Password

Create an f7 password input

Description

Create an f7 password input

Usage

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

Arguments

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)

  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

Framework7 photo browser

Description

A nice photo browser.

Usage

```

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

```

Arguments

photos	List of photos
theme	Browser theme: choose either light or dark.
type	Browser type: choose among c("popup", "standalone", "page").
...	Other options.
session	Shiny session object.

Examples

```

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

  shinyApp(

```



```

ui = f7Page(
  title = "f7PhotoBrowser",
  f7SingleLayout(
    navbar = f7Navbar(title = "f7PhotoBrowser"),
    f7Button(inputId = "togglePhoto", "Open photo")
  )
),
server = function(input, output, session) {
  observeEvent(input$togglePhoto, {
    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"
      )
    )
  })
}
)
}

```

f7Picker

Framework7 picker input

Description

f7Picker generates a picker input.

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

Usage

```

f7Picker(
  inputId,
  label,
  placeholder = NULL,
  value = choices[1],
  choices,
  rotateEffect = TRUE,
  openIn = "auto",
  scrollToInput = FALSE,
  closeByOutsideClick = TRUE,
  toolbar = TRUE,
  toolbarCloseText = "Done",
  sheetSwipeToClose = FALSE
)

```

```

)

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

```

Arguments

inputId	The id of the input object.
label	Picker label.
placeholder	Text to write in the container.
value	Picker initial value, if any.
choices	New picker choices.
rotateEffect	Enables 3D rotate effect. Default to TRUE.
openIn	Can be auto, popover (to open picker in popover), sheet (to open in sheet modal). In case of auto will open in sheet modal on small screens and in popover on large screens. Default to auto.
scrollToInput	Scroll viewport (page-content) to input when picker opened. Default to FALSE.
closeByOutsideClick	If enabled, picker will be closed by clicking outside of picker or related input element. Default to TRUE.
toolbar	Enables picker toolbar. Default to TRUE.
toolbarCloseText	Text for Done/Close toolbar button.
sheetSwipeToClose	Enables ability to close Picker sheet with swipe. Default to FALSE.
session	The Shiny session object, usually the default value will suffice.

Author(s)

David Granjon, <dgranjon@ymail.com>

Examples

```

# Picker input
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)
  }
)

# Update picker input
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"),
          br(),
          f7Button(inputId = "removeToolbar", label = "Remove picker toolbar", color = "red")
        )
      )
    ),
    server = function(input, output, session) {

      output$pickerval <- renderText(input$mypicker)

      observeEvent(input$update, {
        updateF7Picker(
          inputId = "mypicker",

```

```

        value = "b",
        choices = letters,
        openIn = "sheet",
        toolbarCloseText = "Prout",
        sheetSwipeToClose = TRUE
    )
})

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

}
)
}

```

f7Popup

Framework7 popup

Description

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

Usage

```

f7Popup(
  ...,
  id,
  title = NULL,
  backdrop = TRUE,
  closeByBackdropClick = TRUE,
  closeOnEscape = FALSE,
  animate = TRUE,
  swipeToClose = FALSE,
  fullsize = FALSE,
  closeButton = TRUE
)

```

Arguments

...	UI elements for the body of the popup window.
id	Popup unique id.

title	Title for the popup window, use NULL for no title.
backdrop	Enables Popup backdrop (dark semi transparent layer behind). Default to TRUE.
closeByBackdropClick	When enabled, popup will be closed on backdrop click. Default to TRUE.
closeOnEscape	When enabled, popup will be closed on ESC keyboard key press. Default to FALSE.
animate	Whether the Popup should be opened/closed with animation or not. Default to TRUE.
swipeToClose	Whether the Popup can be closed with swipe gesture. Can be true to allow to close popup with swipes to top and to bottom. Default to FALSE.
fullsize	Open popup in full width or not. Default to FALSE.
closeButton	Add or not a button to easily close the popup. Default to TRUE.

Examples

```

if (interactive()) {
  library(shiny)
  library(shinyMobile)
  shinyApp(
    ui = f7Page(
      title = "Popup",
      f7SingleLayout(
        navbar = f7Navbar(
          title = "f7Popup",
          hairline = FALSE,
          shadow = TRUE
        ),
        f7Button("togglePopup", "Toggle Popup"),
        f7Popup(
          id = "popup1",
          title = "My first popup",
          f7Text("text", "Popup content", "This is my first popup ever, I swear!"),
          verbatimTextOutput("popupContent")
        )
      )
    ),
    server = function(input, output, session) {

      output$popupContent <- renderPrint(input$text)

      observeEvent(input$togglePopup, {
        updateF7Popup(id = "popup1")
      })

      observeEvent(input$popup1, {

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

        f7Toast(
          position = "top",

```

```
      text = paste("Popup is", popupStatus)
    )
  })
}
)
```

f7Progress

Framework7 progress bar

Description

f7Progress creates a progress bar.

updateF7Progress update a framework7 progress bar from the server side

Usage

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

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

Arguments

id	Unique progress bar id.
value	New value.
color	Progress color. See https://framework7.io/docs/progressbar.html .
session	Shiny session object.

Author(s)

David Granjon, <dgranjon@ymail.com>

Examples

```
# Progress bars
if(interactive()){
  library(shiny)
  library(shinyMobile)

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

```

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

# Update progress
if (interactive()) {
  library(shiny)
  library(shinyMobile)

  shinyApp(
    ui = f7Page(
      title = "Update 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$obs, {
        updateF7Progress(id = "pg1", value = input$obs)
      })
    }
  )
}

```

f7Radio

Framework7 radio input

Description

f7Radio creates a radio button input.

updateF7Radio updates a radio button input.

Usage

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

```
updateF7Radio(
```

```

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

```

Arguments

inputId	Radio input id.
label	New radio label
choices	New list of choices.
selected	New selected element. NULL by default.
session	Shiny session object.

Examples

```

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

  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"])
      })
    }
  )
}

# Update radio
if (interactive()) {
  library(shiny)
  library(shinyMobile)

  shinyApp(
    ui = f7Page(
      title = "Update radio",
      f7SingleLayout(

```



```
navbar = f7Navbar(title = "Update f7Radio"),
f7Button("go", "Update radio"),
f7Radio(
  inputId = "radio",
  label = "Choose a fruit:",
  choices = c("banana", "apple", "peach"),
  selected = "apple"
),
textOutput("radio_value")
)
),
server = function(input, output, session) {
  output$radio_value <- renderText(input$radio)

  observeEvent(input$go, {
    updateF7Radio(
      session,
      inputId = "radio",
      label = "New label",
      choices = colnames(mtcars),
      selected = colnames(mtcars)[1]
    )
  })
}
)
```

f7Row

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@gmail.com>

Examples

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

  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", label = "My button"),
                f7Badge("Badge", color = "green")
              )
            )
          )
        )
      ),
      server = function(input, output) {}
    )
  }
```

f7Searchbar

Framework 7 searchbar

Description

Searchbar to filter elements in a page.

Usage

```
f7Searchbar(
  id,
  placeholder = "Search",
```

```

    expandable = FALSE,
    inline = FALSE,
    options = NULL
  )

```

Arguments

id	Necessary when using f7SearchbarTrigger . NULL otherwise.
placeholder	Searchbar placeholder.
expandable	Whether to enable the searchbar with a target link, in the navbar. See f7SearchbarTrigger .
inline	Useful to add an f7Searchbar in an f7AppBar . Notice that utilities like f7HideOnSearch and f7NotFound are not compatible with this mode.
options	Search bar options. See https://v5.framework7.io/docs/searchbar.html#searchbar-parameters . If no options are provided, the searchbar will search in list elements by item title. This may be changed by updating the default searchContainer and searchIn.

Examples

```

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

  cars <- rownames(mtcars)

  shinyApp(
    ui = f7Page(
      title = "Simple searchbar",
      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)

shinyApp(
  ui = f7Page(
    title = "Expandable searchbar",
    f7SingleLayout(
      navbar = f7Navbar(
        title = "f7Searchbar with trigger",
        hairline = FALSE,
        shadow = TRUE,
        subNavbar = f7SubNavbar(
          f7Searchbar(id = "search1", expandable = TRUE)
        )
      ),
      f7Block(
        f7SearchbarTrigger(targetId = "search1")
      ) %>% f7HideOnSearch(),
      f7List(
        lapply(seq_along(cities), function(i) {
          f7ListItem(cities[i])
        })
      ) %>% f7Found(),

      f7Block(
        p("Nothing found")
      ) %>% f7NotFound()
    )
  ),
  server = function(input, output) {}
)

# Searchbar in \link{f7AppBar}
shinyApp(
  ui = f7Page(
    title = "Searchbar in appbar",
    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 *Framework 7 searchbar trigger*

Description

Element that triggers the searchbar.

Usage

```
f7SearchbarTrigger(targetId)
```

Arguments

targetId Id of the [f7Searchbar](#).

f7SearchIgnore *Utility to ignore an item from search.*

Description

Use with [f7Searchbar](#).

Usage

```
f7SearchIgnore(tag)
```

Arguments

tag tag to ignore.

f7Segment

Framework7 segmented button container

Description

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

Usage

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

Arguments

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

Author(s)

David Granjon, <dgranjon@ymail.com>

Examples

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

  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", href = "https://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", href = "https://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, fill = FALSE),
  f7Button(label = "My button", outline = TRUE, fill = FALSE),
  f7Button(label = "My button", outline = TRUE, fill = FALSE)
),
f7BlockTitle(title = "Buttons in a segment/strong container"),
f7Segment(
  strong = TRUE,
  container = "segment",
  f7Button(label = "My button", fill = FALSE),
  f7Button(label = "My button", fill = FALSE),
  f7Button(label = "My button", fill = FALSE, active = 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

*Framework7 select input***Description**

f7Select creates a select input.

updateF7Select changes the value of a select input on the client

Usage

```
f7Select(inputId, label, choices, selected = NULL, width = NULL)
```

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

Arguments

inputId	The id of the input object.
label	Select input label.
choices	Select input choices.
selected	New value.
width	The width of the input, e.g. 400px, or 100%.
session	The Shiny session object, usually the default value will suffice.

Examples

```
# Select input
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],
          selected = "hp"
        ),
        tableOutput("data")
      )
    )
  )
}
```



```

    ),
    server = function(input, output) {
      output$data <- renderTable({
        mtcars[, c("mpg", input$variable), drop = FALSE]
      }, rownames = TRUE)
    }
  }
}
# Update select input
if (interactive()) {
  library(shiny)
  library(shinyMobile)

  shinyApp(
    ui = f7Page(
      title = "My app",
      f7SingleLayout(
        navbar = f7Navbar(title = "updateF7Select"),
        f7Card(
          f7Button(inputId = "update", label = "Update select"),
          br(),
          f7Select(
            inputId = "variable",
            label = "Choose a variable:",
            choices = colnames(mtcars)[-1],
            selected = "hp"
          ),
          verbatimTextOutput("test")
        )
      )
    ),
    server = function(input, output, session) {

      output$test <- renderPrint(input$variable)

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

```

Description

Creates a shadow effect to apply on UI elements like [f7Card](#).

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@gmail.com>

Examples

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

  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", href = "https://www.google.com"),
              f7Badge("Badge", color = "green")
            )
          )
        )
      )
    ),
    server = function(input, output) {}
  )
}
```

f7Sheet

*Framework7 sheet***Description**

f7Sheet creates an f7 sheet modal window.

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

Usage

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

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

Arguments

...	Sheet content. If wipeToStep is TRUE, these items will be visible at start.
id	Sheet id.
hiddenItems	Put items you want to hide inside. Only works when swipeToStep is TRUE. Default to NULL.
orientation	"top" or "bottom".
swipeToClose	If TRUE, it can be closed by swiping down.
swipeToStep	If TRUE then sheet will be opened partially, and with swipe it can be further expanded.
backdrop	Enables Sheet backdrop (dark semi transparent layer behind). By default it is TRUE for MD 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.
session	Shiny session object

Note

The sheet modal has to be used in combination with [updateF7Sheet](#). Yet, if you need a specific trigger, simply add ``data-sheet` = paste0("#", id)`, to the tag of your choice (a button), where id refers to the sheet unique id.

Examples

```
# Toggle sheet modal
if (interactive()) {
  library(shiny)
  library(shinyMobile)
  shinyApp(
    ui = f7Page(
      title = "Update f7Sheet",
      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",
          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(
              f7Gauge(
                id = "mygauge",
                type = "semicircle",
                value = 10,
                borderColor = "#2196f3",
                borderWidth = 10,
                valueFontSize = 41,
                valueTextColor = "#2196f3",
                labelText = "amount of something"
              )
            ),
            f7Slider(
              inputId = "obs",
              label = "Number of observations",
              max = 100,
              min = 0,

```

```

        value = 10,
        scale = TRUE
      ),
      plotOutput("distPlot")
    )
  )
),
server = function(input, output, session) {
  observe({print(input$sheet1)})
  output$distPlot <- renderPlot({
    hist(rnorm(input$obs))
  })
  observeEvent(input$obs, {
    updateF7Gauge(id = "mygauge", value = input$obs)
  })
  observeEvent(input$go, {
    updateF7Sheet(id = "sheet1")
  })
}
)
}

```

f7SingleLayout

Framework7 single layout

Description

f7SingleLayout provides a simple page 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 <code>tagList</code> if multiple panels.
appbar	Slot for f7Appbar .

Author(s)

David Granjon, <dgranjon@gmail.com>

Examples

```

if(interactive()){
  library(shiny)
  library(shinyMobile)
  shinyApp(
    ui = f7Page(
      title = "Single layout",
      f7SingleLayout(
        navbar = f7Navbar(
          title = "Single Layout",
          hairline = FALSE,
          shadow = TRUE
        ),
        toolbar = f7Toolbar(
          position = "bottom",
          f7Link(label = "Link 1", href = "https://www.google.com"),
          f7Link(label = "Link 2", href = "https://www.google.com")
        ),
        # main content
        f7Shadow(
          intensity = 10,
          hover = TRUE,
          f7Card(
            title = "Card header",
            f7Slider("obs", "Number of observations", 0, 1000, 500),
            plotOutput("distPlot"),
            footer = tagList(
              f7Button(color = "blue", label = "My button", href = "https://www.google.com"),
              f7Badge("Badge", color = "green")
            )
          )
        )
      )
    ),
    server = function(input, output) {
      output$distPlot <- renderPlot({
        dist <- rnorm(input$obs)
        hist(dist)
      })
    }
  )
}

```

Description

Nice loading overlay for UI elements.

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)

  shinyApp(
    ui = f7Page(
      title = "Skeletons",
      f7SingleLayout(
        navbar = f7Navbar(title = "f7Skeleton"),
        f7Card(
          title = "Card header",
          "This is a simple card with plain text,
          but cards can also contain their own header,
          footer, list view, image, or any other element.",
          footer = tagList(
            f7Button(color = "blue", label = "My button", href = "https://www.google.com"),
            f7Badge("Badge", color = "green")
          )
        ) %>% f7Skeleton(),

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

f7Slide	<i>Framework7 slide</i>
---------	-------------------------

Description

f7Slide is an [f7Swiper](#) element.

Usage

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

Arguments

... Slide content. Any element.

f7Slider	<i>Framework7 range slider</i>
----------	--------------------------------

Description

f7Slider creates a f7 slider input.

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

Usage

```
f7Slider(  
  inputId,  
  label,  
  min,  
  max,  
  value,  
  step = 1,  
  scale = FALSE,  
  scaleSteps = 5,  
  scaleSubSteps = 0,  
  vertical = FALSE,  
  verticalReversed = FALSE,  
  labels = NULL,  
  color = NULL,  
  noSwipping = TRUE  
)
```

```
updateF7Slider(  
  inputId,  
  min = NULL,
```



```

    max = NULL,
    value = NULL,
    scale = FALSE,
    scaleSteps = NULL,
    scaleSubSteps = NULL,
    step = NULL,
    color = NULL,
    session = shiny::getDefaultReactiveDomain()
  )

```

Arguments

inputId	The id of the input object.
label	Slider label.
min	Slider minimum range.
max	Slider maximum range
value	Slider value or a vector containing 2 values (for a range).
step	Slider increase step size.
scale	Slider scale.
scaleSteps	Number of scale steps.
scaleSubSteps	Number of scale sub steps (each step will be divided by this value).
vertical	Whether to apply a vertical display. FALSE by default.
verticalReversed	Makes vertical range slider reversed (vertical must be also enabled). FALSE by default.
labels	Enables additional label around range slider knob. List of 2 f7Icon expected.
color	See getF7Colors for valid colors.
noSwipping	Prevent swiping when slider is manipulated in an f7TabLayout .
session	The Shiny session object.

Note

labels option only works when vertical is FALSE!

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

Examples

```

# Slider input
if(interactive()){
  library(shiny)
  library(shinyMobile)

  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,
      scaleSteps = 5,
      scaleSubSteps = 3,
      scale = TRUE,
      color = "orange",
      labels = tagList(
        f7Icon("circle"),
        f7Icon("circle_fill")
      )
    ),
    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)

  shinyApp(
    ui = f7Page(
      title = "My app",
      f7SingleLayout(
        navbar = f7Navbar(title = "f7Slider Range"),
        f7Card(
          f7Slider(
            inputId = "obs",
            label = "Range values",
            max = 500,
            min = 0,
            value = c(50, 100),
            scale = FALSE
          ),
          verbatimTextOutput("test")
        )
      )
    )
  )
}

```

```

    )
  },
  server = function(input, output) {
    output$test <- renderPrint({input$obs})
  }
)
}

# Update f7Slider
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,
            step = 1,
            color = "deeppurple",
            value = c(50, 100)
          ),
          verbatimTextOutput("test")
        )
      )
    ),
    server = function(input, output, session) {

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

      observeEvent(input$update, {
        updateF7Slider(
          inputId = "obs",
          value = c(1, 5),
          min = 0,
          scaleSteps = 10,
          scaleSubSteps = 5,
          step = 0.1,
          max = 10,
          color = "teal"
        )
      })
    }
  )
}

```

f7SmartSelect

*Framework7 smart select***Description**

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

Usage

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

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

Arguments

inputId	The id of the input object.
label	Select input label.
choices	The new choices.
selected	The new value for the input.
openIn	Smart select type: either c("sheet", "popup", "popover"). Note that the search bar is only available when the type is popup.
searchbar	Whether to enable the search bar. TRUE by default.
multiple	Whether to allow multiple values.
maxLength	Maximum items to select when multiple is TRUE.

virtualList	Enable Virtual List for smart select if your select has a lot of options. Default to FALSE.
...	Parameters used to update the smart select, use same arguments as in f7SmartSelect .
maxLength	Maximum items to select when multiple is TRUE.
session	The Shiny session object, usually the default value will suffice.

Examples

```
# Smart select input
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],
          openIn = "popup"
        ),
        tableOutput("data"),
        f7SmartSelect(
          inputId = "variable2",
          label = "Group variables:",
          choices = list(
            `East Coast` = list("NY", "NJ", "CT"),
            `West Coast` = list("WA", "OR", "CA"),
            `Midwest` = list("MN", "WI", "IA")
          ),
          openIn = "sheet"
        ),
        textOutput("var")
      )
    ),
    server = function(input, output) {
      output$var <- renderText(input$variable2)
      output$data <- renderTable({
        mtcars[, c("mpg", input$variable), drop = FALSE]
      }, rownames = TRUE)
    }
  )
}

# Update smart select
if (interactive()) {
  library(shiny)
  library(shinyMobile)
```

```

shinyApp(
  ui = f7Page(
    title = "My app",
    f7SingleLayout(
      navbar = f7Navbar(title = "Update f7SmartSelect"),
      f7Button("updateSmartSelect", "Update Smart Select"),
      f7SmartSelect(
        inputId = "variable",
        label = "Choose a variable:",
        selected = "drat",
        choices = colnames(mtcars)[-1],
        openIn = "popup"
      ),
      tableOutput("data")
    )
  ),
  server = function(input, output, session) {
    output$data <- renderTable({
      mtcars[, c("mpg", input$variable), drop = FALSE]
    }, rownames = TRUE)

    observeEvent(input$updateSmartSelect, {
      updateF7SmartSelect(
        inputId = "variable",
        openIn = "sheet",
        selected = "hp",
        choices = c("hp", "gear"),
        multiple = TRUE,
        maxLength = 3
      )
    })
  }
)
}

```

f7SplitLayout

Framework7 split layout

Description

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

Usage

```

f7SplitLayout(
  ...,
  navbar,
  sidebar,
  toolbar = NULL,
  panels = NULL,

```

```

    appBar = NULL
  )

```

Arguments

...	Content.
navbar	Slot for f7Navbar .
sidebar	Slot for f7Panel . Particularly we expect the following code: <code>f7Panel(title = "Sidebar", side = "left", theme = "light", "Blabla", style = "reveal")</code>
toolbar	Slot for f7Toolbar .
panels	Slot for f7Panel . Expect only a right panel, for instance: <code>f7Panel(title = "Left Panel", side = "right", theme = "light", "Blabla", style = "cover")</code>
appBar	Slot for f7AppBar .

Author(s)

David Granjon, <dgranjon@gmail.com>

Examples

```

if(interactive()){
  library(shiny)
  library(shinyMobile)
  shinyApp(
    ui = f7Page(
      title = "Split layout",
      f7SplitLayout(
        sidebar = f7Panel(
          inputId = "sidebar",
          title = "Sidebar",
          side = "left",
          theme = "light",
          f7PanelMenu(
            id = "menu",
            f7PanelItem(tabName = "tab1", title = "Tab 1", icon = f7Icon("envelope"), active = TRUE),
            f7PanelItem(tabName = "tab2", title = "Tab 2", icon = f7Icon("house"))
          ),
          effect = "reveal"
        ),
        navbar = f7Navbar(
          title = "Split Layout",
          hairline = FALSE,
          shadow = TRUE
        ),
        toolbar = f7Toolbar(
          position = "bottom",
          f7Link(label = "Link 1", href = "https://www.google.com"),
          f7Link(label = "Link 2", href = "https://www.google.com")
        ),
        # main content

```

```

    f7Items(
      f7Item(
        tabName = "tab1",
        f7Slider("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

Framework7 stepper input

Description

f7Stepper creates a stepper input.

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

Usage

```

f7Stepper(
  inputId,
  label,
  min,
  max,
  value,
  step = 1,
  fill = FALSE,
  rounded = FALSE,
  raised = FALSE,
  size = NULL,
  color = NULL,

```



```

    wraps = FALSE,
    autorepeat = TRUE,
    manual = FALSE,
    decimalPoint = 4,
    buttonsEndInputMode = TRUE
  )

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

```

Arguments

inputId	The id of the input object.
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 relief around the stepper. FALSE by default.
size	Stepper size: "small", "large" or NULL.
color	Stepper color: NULL or "red", "green", "blue", "pink", "yellow", "orange", "grey" and "black".
wraps	In wraps mode incrementing beyond maximum value sets value to minimum value, likewise, decrementing below minimum value sets value to maximum value. FALSE by default.
autorepeat	Pressing and holding one of its buttons increments or decrements the stepper's value repeatedly. With dynamic autorepeat, the rate of change depends on how long the user continues pressing the control. TRUE by default.

<code>manual</code>	It is possible to enter value manually from keyboard or mobile keypad. When click on input field, stepper enter into manual input mode, which allow type value from keyboar and check fractional part with defined accuracy. Click outside or enter Return key, ending manual mode. TRUE by default.
<code>decimalPoint</code>	Number of digits after dot, when in manual input mode.
<code>buttonsEndInputMode</code>	Disables manual input mode on Stepper's minus or plus button click.
<code>session</code>	The Shiny session object, usually the default value will suffice.

Note

While updating, the autorepeat field does not work correctly.

Examples

```
# Stepper input
if(interactive()){
  library(shiny)
  library(shinyMobile)

  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)
    }
  )
}
```

```

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

```

```
)  
}
```

f7SubNavbar

Framework7 sub navbar

Description

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

Usage

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

Arguments

... Any elements.

Examples

```
if (interactive()) {  
  library(shiny)  
  library(shinyMobile)  
  
  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,  
          leftPanel = TRUE,  
          rightPanel = TRUE,  
          subNavbar = f7SubNavbar(  
            f7Button(label = "My button"),  
            f7Button(label = "My button"),  
            f7Button(label = "My button")  
          )  
        ),  
      f7Tabs(  
        animated = TRUE,  
        #swipeable = TRUE,  
        f7Tab(  
          tabName = "Tab 1",  
          icon = f7Icon("envelope"),  

```

```

        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

Framework7 swipeout element

Description

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

f7SwipeoutItem is inserted in [f7Swipeout](#).

Usage

```

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

```

```

f7SwipeoutItem(id, label, color = NULL)

```

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".
id	Item unique id.
label	Item label.
color	Item color.

Examples

```

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

  shinyApp(
    ui = f7Page(
      title = "Swipeout",
      f7SingleLayout(
        navbar = f7Navbar(title = "Swipeout"),
        # 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"
        )
      })

      observeEvent(input$alert, {
        f7Dialog(
          title = "Dialog title",

```

```
      text = "This is an alert dialog"
    )
  })
}
)
```

f7Swiper

Framework7 swiper

Description

f7Swiper creates a Framework7 swiper container (like carousel).

Usage

```
f7Swiper(
  ...,
  id,
  options = list(speed = 400, spaceBetween = 50, slidesPerView = "auto", centeredSlides
    = TRUE, pagination = TRUE)
)
```

Arguments

...	Slot for f7Slide .
id	Swiper unique id.
options	Other options. Expect a list.

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 = "Swiper",
    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)
  }
)
```


Description

Build a Framework7 tab item

Usage

```
f7Tab(..., tabName, icon = NULL, active = FALSE, hidden = 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.
hidden	Whether to hide the tab. This is useful when you want to add invisible tabs (that do not appear in the tabbar) but you can still navigate with updateF7Tabs .

Author(s)

David Granjon, <dgranjon@ymail.com>

f7TabLayout	<i>Framework7 tab layout</i>
-------------	------------------------------

Description

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

Usage

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

Arguments

...	Slot for f7Tabs .
navbar	Slot for f7Navbar .
messagebar	Slot for f7MessageBar .
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)
  library(shinyWidgets)

  shinyApp(
    ui = f7Page(
      title = "Tab layout",
      f7TabLayout(
        tags$head(
          tags$script(
            "$(function(){
              $('#tapHold').on('taphold', function () {
                app.dialog.alert('Tap hold fired!');
              });
            });"
          )
        ),
        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,
          leftPanel = TRUE,
          rightPanel = TRUE
        ),
        f7Tabs(
          animated = FALSE,
          swipeable = TRUE,
          f7Tab(
            tabName = "Tab 1",
            icon = f7Icon("envelope"),
            active = TRUE,
            f7Shadow(
              intensity = 10,
              hover = TRUE,
              f7Card(
                title = "Card header",
                f7Stepper(
                  "obs1",
                  "Number of observations",
                  min = 0,
                  max = 1000,
                  value = 500,
                  step = 100
                ),
                plotOutput("distPlot1"),

```

```

        footer = tagList(
          f7Button(inputId = "tapHold", label = "My button"),
          f7Badge("Badge", color = "green")
        )
      )
    ),
  f7Tab(
    tabName = "Tab 2",
    icon = f7Icon("today"),
    active = FALSE,
    f7Shadow(
      intensity = 10,
      hover = TRUE,
      f7Card(
        title = "Card header",
        f7Select(
          inputId = "obs2",
          label = "Distribution type:",
          choices = c(
            "Normal" = "norm",
            "Uniform" = "unif",
            "Log-normal" = "lnorm",
            "Exponential" = "exp"
          )
        )
      ),
      plotOutput("distPlot2"),
      footer = tagList(
        f7Button(label = "My button", href = "https://www.google.com"),
        f7Badge("Badge", color = "orange")
      )
    )
  ),
  f7Tab(
    tabName = "Tab 3",
    icon = f7Icon("cloud_upload"),
    active = FALSE,
    f7Shadow(
      intensity = 10,
      hover = TRUE,
      f7Card(
        title = "Card header",
        f7SmartSelect(
          inputId = "variable",
          label = "Variables to show:",
          c("Cylinders" = "cyl",
            "Transmission" = "am",
            "Gears" = "gear"),
          multiple = TRUE,
          selected = "cyl"
        )
      ),
      tableOutput("data"),

```

```

        footer = tagList(
          f7Button(label = "My button", href = "https://www.google.com"),
          f7Badge("Badge", color = "green")
        )
      )
    )
  )
),
server = function(input, output) {
  output$distPlot1 <- renderPlot({
    dist <- rnorm(input$obs1)
    hist(dist)
  })

  output$distPlot2 <- renderPlot({
    dist <- switch(
      input$obs2,
      norm = rnorm,
      unif = runif,
      lnorm = rlnorm,
      exp = rexp,
      rnorm
    )

    hist(dist(500))
  })

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

```

f7Table

Framework7 table

Description

Creates a table container.

Usage

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

Arguments

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

Examples

```

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

```

f7TabLink

Special button/link to insert in the tabbar

Description

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

Usage

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

Arguments

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

f7Tabs

*Create a Framework7 tabs***Description**

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

Usage

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

Arguments

...	Slot for f7Tab .
.items	Slot for other items that could be part of the toolbar such as buttons or f7TabLink . This may be useful to open an f7Sheet from the tabbar.
id	Optional to get the id of the currently selected f7Tab .
swipeable	Whether to allow finger 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.
style	Tabs style: c("toolbar", "segmented", "strong"). If style is toolbar, then f7Tab have a toolbar behavior.

Author(s)

David Granjon, <dgranjon@gmail.com>

Examples

```
if (interactive()) {
  # tabs as toolbar
  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",
    f7Sheet(
      id = "sheet",
      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"
    )
  ),
  f7Tab(tabName = "Tab 2", "tab 2 text"),
  f7Tab(tabName = "Tab 3", "tab 3 text"),
  .items = f7TabLink(
    icon = f7Icon("bolt_fill"),
    label = "Toggle Sheet",
    `data-sheet` = "#sheet",
    class = "sheet-open"
  )
)
),
server = function(input, output) {
  output$selected <- renderText(input$tabdemo)
}
)
# standalone tabs
library(shiny)
library(shinyMobile)
shiny::shinyApp(
  ui = f7Page(
    title = "My app",
    f7SingleLayout(
      navbar = f7Navbar(
        title = "Standalone tabs",
        hairline = FALSE,
        shadow = TRUE
      ),
      f7Tabs(
        id = "tabs",
        style = "strong", animated = FALSE, swipeable = TRUE,
        f7Tab(
          tabName = "Tab 1",
          icon = f7Icon("envelope"),
          active = TRUE,
          f7Shadow(

```

```

intensity = 10,
hover = TRUE,
f7Card(
  title = "Card header",
  f7Stepper(
    "obs1",
    "Number of observations",
    min = 0,
    max = 1000,
    value = 500,
    step = 100
  ),
  plotOutput("distPlot")
)
),
f7Tab(
  tabName = "Tab 2",
  icon = f7Icon("today"),
  active = FALSE,
  f7Shadow(
    intensity = 10,
    hover = TRUE,
    f7Card(
      title = "Card header",
      f7Select(
        inputId = "obs2",
        label = "Distribution type:",
        choices = c(
          "Normal" = "norm",
          "Uniform" = "unif",
          "Log-normal" = "lnorm",
          "Exponential" = "exp"
        )
      ),
      plotOutput("distPlot2")
    )
  ),
),
f7Tab(
  tabName = "Tab 3",
  icon = f7Icon("cloud_upload"),
  active = FALSE,
  f7Shadow(
    intensity = 10,
    hover = TRUE,
    f7Card(
      title = "Card header",
      f7SmartSelect(
        inputId = "variable",
        label = "Variables to show:",
        c("Cylinders" = "cyl",
          "Transmission" = "am",

```



```

        "Gears" = "gear"),
        multiple = TRUE,
        selected = "cyl"
      ),
      tableOutput("data")
    )
  )
)
),
server = function(input, output) {
  output$distPlot <- renderPlot({
    dist <- rnorm(input$obs1)
    hist(dist)
  })

  output$distPlot2 <- renderPlot({
    dist <- switch(
      input$obs2,
      norm = rnorm,
      unif = runif,
      lnorm = rlnorm,
      exp = rexp,
      rnorm
    )

    hist(dist(500))
  })

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

```

f7TapHold

Framework7 tapHold module

Description

[f7TapHold](#) is triggered after long press on an element, from the server.

Usage

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

Arguments

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

Examples

```

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

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

```

f7Text

Framework7 text input

Description

f7Text creates a text input container.

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

Usage

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

```

updateF7Text(
  inputId,
  label = NULL,
  value = NULL,

```

```

    placeholder = NULL,
    session = shiny::getDefaultReactiveDomain()
  )

```

Arguments

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.
session	The Shiny session object, usually the default value will suffice.

Examples

```

# A text input
if(interactive()){
  library(shiny)
  library(shinyMobile)

  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 })
    }
  )
}

# Update text input
if (interactive()) {
  library(shiny)
  library(shinyMobile)

  ui <- f7Page(
    f7SingleLayout(
      navbar = f7Navbar(title = "updateF7Text"),
      f7Block(f7Button("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("text", value = "Updated Text")
  })
}
shinyApp(ui, server)
}

```

f7TextArea

Framework7 text area input

Description

f7TextArea creates a f7 text area input.

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

Usage

```
f7TextArea(inputId, label, value = "", placeholder = NULL, resize = FALSE)
```

```

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

```

Arguments

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.
resize	Whether to box can be resized. Default to FALSE.
session	The Shiny session object, usually the default value will suffice.

Examples

```

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

  shinyApp(
    ui = f7Page(
      title = "My app",
      f7TextArea(
        inputId = "textarea",
        label = "Text Area",
        value = "Lorem ipsum dolor sit amet, consectetur
        adipiscing elit, sed do eiusmod tempor incididunt ut
        labore et dolore magna aliqua",
        placeholder = "Your text here",
        resize = TRUE
      ),
      textOutput("value")
    ),
    server = function(input, output) {
      output$value <- renderText({ input$textarea })
    }
  )
}

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

  ui <- f7Page(
    f7SingleLayout(
      navbar = f7Navbar(title = "updateF7TextArea"),
      f7Block(f7Button("trigger", "Click me")),
      f7TextArea(
        inputId = "textarea",
        label = "Text Area",
        value = "Lorem ipsum dolor sit amet, consectetur
        adipiscing elit, sed do eiusmod tempor incididunt ut
        labore et dolore magna aliqua",
        placeholder = "Your text here",
        resize = TRUE
      ),
      verbatimTextOutput("value")
    )
  )

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

```

```
}

```

f7Timeline

Framework7 timeline

Description

f7Timeline is a static timeline container.

f7TimelineItem goes inside [f7Timeline](#).

Usage

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

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

Arguments

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

Author(s)

David Granjon <dgranjon@ymail.com>

David Granjon <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"
    )
  )

  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) {}
)
}

```

f7Toast

Framework7 toast

Description

f7Toast creates a small toast notification from the server side.

Usage

```

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

```

Arguments

text	Toast content.
position	Toast position c("bottom", "top", "center").
closeButton	Whether to close the toast with a button. TRUE by default.
closeButtonText	Close button text.
closeButtonColor	Close button color.
closeTimeout	Time before toast closes.

icon	Optional. Expect <code>f7Icon</code> . Warning: Adding icon will hide the close button.
...	Other options. See https://framework7.io/docs/toast.html#toast-parameters .
session	Shiny session.

Examples

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

f7Toggle

Framework7 toggle input

Description

f7Toggle creates a F7 toggle switch input.

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

Usage

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

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

Arguments

inputId	The id of the input object.
label	Toggle label.
checked	Whether the toggle is TRUE or FALSE.
color	Toggle color.
session	The Shiny session object.

Examples

```
# f7Toggle
if(interactive()){
  library(shiny)
  library(shinyMobile)

  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)
    }
  )
}

# Update f7Toggle
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$test <- renderPrint({input$update})

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

```

f7Toolbar

Framework7 Toolbar

Description

f7Toolbar is a layout element located at the bottom or top. It is internally used by [f7Tabs](#).

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@gmail.com>

f7Tooltip

Framework7 tooltip

Description

f7Tooltip creates a static tooltip, UI side.

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

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

Usage

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

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

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

Arguments

tag	Tooltip target.
text	New tooltip text value. See https://v5.framework7.io/docs/tooltip.html#tooltip-parameters .
id	Tooltip target id.
selector	jQuery selector. Allow more customization for the target (nested tags).
options	List of options to pass to the tooltip. See https://v5.framework7.io/docs/tooltip.html#tooltip-parameters .
session	Shiny session object.
action	Either toggle or update the tooltip.

Examples

```

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

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

  lorem_ipsum <- "Lorem ipsum dolor sit amet!"

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

  shinyApp(
    ui = f7Page(
      options = list(theme = "ios"),
      title = "f7Tooltip",
      f7SingleLayout(
        navbar = f7Navbar(
          title = "f7Tooltip",
          subNavbar = f7SubNavbar(
            f7Toggle(
              inputId = "toggle",
              "Enable tooltips",
              color = "green",
              checked = TRUE
            )
          )
        )
      ),
      f7Segment(
        lapply(seq_len(nrow(tooltips)), function(i) {
          f7Button(
            inputId = sprintf("target_%s", i),

```

```

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

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

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

```

f7VirtualList

Framework7 virtual list

Description

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

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

Usage

```
f7VirtualList(id, items, rowsBefore = NULL, rowsAfter = NULL, cache = TRUE)
```

```
f7VirtualListItem(
  ...,
  title = NULL,
  subtitle = NULL,
  header = NULL,
  footer = NULL,
  href = NULL,
  media = NULL,
  right = NULL
)
```

Arguments

id	Virtual list unique id.
items	List items. Slot for f7VirtualListItem .
rowsBefore	Amount of rows (items) to be rendered before current screen scroll position. By default it is equal to double amount of rows (items) that fit to screen.
rowsAfter	Amount of rows (items) to be rendered after current screen scroll position. By default it is equal to the amount of rows (items) that fit to screen.
cache	Disable or enable DOM cache for already rendered list items. In this case each item will be rendered only once and all further manipulations will be with DOM element. It is useful if your list items have some user interaction elements (like form elements or swipe outs) or could be modified.
...	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.
href	Item external link.
media	Expect f7Icon or img.
right	Right content if any.

Examples

```
if (interactive()) {
  library(shiny)
  library(shinyMobile)
  shinyApp(
    ui = f7Page(
      title = "Virtual List",
      f7SingleLayout(
        navbar = f7Navbar(
```

```

    title = "Virtual Lists",
    hairline = FALSE,
    shadow = TRUE
  ),
  # main content
  f7VirtualList(
    id = "vlist",
    rowsBefore = 2,
    rowsAfter = 2,
    items = lapply(1:2000, function(i) {
      f7VirtualListItem(
        title = paste("Title", i),
        subtitle = paste("Subtitle", i),
        header = paste("Header", i),
        footer = paste("Footer", i),
        right = paste("Right", i),
        content = i,
        media = img(src = "https://cdn.framework7.io/placeholder/fashion-88x88-1.jpg")
      )
    })
  )
)
),
server = function(input, output) {
}
)

```

```

# below example will not load with classic f7List
#shinyApp(
#  ui = f7Page(
#    title = "My app",
#    f7SingleLayout(
#      navbar = f7Navbar(
#        title = "Virtual Lists",
#        hairline = FALSE,
#        shadow = TRUE
#      ),
#      # main content
#      f7List(
#        lapply(1:2000, function(i) {
#          f7ListItem(
#            title = paste("Title", i),
#            subtitle = paste("Subtitle", i),
#            header = paste("Header", i),
#            footer = paste("Footer", i),
#            right = paste("Right", i),
#            content = i
#          )
#        })
#      )
#    )
#  ),
# ),

```



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

getF7Colors

Function to get all colors available in shinyMobile

Description

Function to get all colors available in shinyMobile

Usage

```
getF7Colors()
```

Value

A vector containing colors

insertF7Tab

Framework7 tab insertion

Description

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

Usage

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

Arguments

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

Examples

```

if (interactive()) {
  library(shiny)
  library(shinyMobile)
  shinyApp(
    ui = f7Page(
      title = "Insert a tab Before the target",
      f7TabLayout(
        navbar = f7Navbar(
          title = "Tabs",
          hairline = FALSE,
          shadow = TRUE,
          leftPanel = TRUE,
          rightPanel = TRUE
        ),
        f7Tabs(
          animated = TRUE,
          id = "tabs",
          f7Tab(
            tabName = "Tab 1",
            icon = f7Icon("airplane"),
            active = TRUE,
            "Tab 1",
            f7Button(inputId = "add", label = "Add tabs")
          ),
          f7Tab(
            tabName = "Tab 2",
            icon = f7Icon("today"),
            active = FALSE,
            f7Button(inputId="stay", label = "Stay"),
            "Tab 2"
          )
        )
      )
    ),
    server = function(input, output, session) {
      observeEvent(input$stay, {
        f7Toast("Please stay")
      })
      observeEvent(input$add, {
        insertF7Tab(
          id = "tabs",
          position = "after",
          target = "Tab 1",
          tab = f7Tab (
            # Use multiple elements to test for accessor function
            f7Button(inputId = "add_dynamic", label = "Add dyn"),
            f7Text(inputId = "my_text", label ="Enter something", placeholder = "What?"),
            f7Text(inputId = "my_other", label ="Else:", placeholder = "Else ?"),
            tabName = paste0("tabx_", input$go), "Test2",
            icon = f7Icon("app_badge")
          )
        ),

```


Examples

```

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

```

removeF7Tab

Framework7 tab deletion

Description

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

Usage

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

Arguments

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

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,
        leftPanel = TRUE,
        rightPanel = 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, {
    removeF7Tab(
      id = "tabset1",
      target = "Tab 1"
    )
  })
}
shinyApp(ui, server)
}

```

showF7Preloader

Framework7 preloader

Description

showF7Preloader shows a preloader.

f7HidePreloader hides a preloader.

Usage

```

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

```

```

f7HidePreloader(target = NULL, session = shiny::getDefaultReactiveDomain())

```

Arguments

target	Element where preloader overlay will be added.
color	Preloader color.
session	Shiny session object.

Examples

```

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

  # basic preloader with red color
  shinyApp(
    ui = f7Page(
      title = "Preloader",
      f7SingleLayout(
        navbar = f7Navbar(
          title = "Preloader",
          hairline = FALSE,
          shadow = TRUE
        ),
        # main content
        f7Button("showLoader", "Show loader"),
        f7Shadow(
          intensity = 10,
          hover = TRUE,
          f7Card(
            title = "Card header",
            f7Slider("obs", "Number of observations", 0, 1000, 500),
            plotOutput("distPlot")
          )
        )
      )
    ),
    server = function(input, output, session) {
      output$distPlot <- renderPlot({
        dist <- rnorm(input$obs)
        hist(dist)
      })

      observeEvent(input$showLoader, {
        showF7Preloader(color = "red")
        Sys.sleep(2)
        f7HidePreloader()
      })
    }
  )

  # preloader in container
  shinyApp(
    ui = f7Page(

```

```

title = "Preloader in container",
f7SingleLayout(
  navbar = f7Navbar(
    title = "Preloader in container",
    hairline = FALSE,
    shadow = TRUE
  ),
  # main content
  f7Shadow(
    intensity = 10,
    hover = TRUE,
    f7Card(
      title = "Card header",
      f7Slider("obs", "Number of observations", 0, 1000, 500),
      plotOutput("distPlot")
    )
  ),
  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.")
)
),
server = function(input, output, session) {
  output$distPlot <- renderPlot({
    dist <- rnorm(input$obs)
    hist(dist)
  })

  observeEvent(input$obs, {
    showF7Preloader(target = "#distPlot", color = "red")
    Sys.sleep(2)
    f7HidePreloader()
  })
}
)
}

```

updateF7App

Update Framework7 configuration

Description

updateF7App allows to update a shinyMobile app at run time by injecting any configuration inside the current running instance. Useful if you want to share the same behavior across multiple elements.

Usage

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

Arguments

options	List of options.
session	Shiny session object.

Note

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

Examples

```

if (interactive()) {
  library(shiny)
  library(shinyMobile)
  shinyApp(
    ui = f7Page(
      title = "Simple Dialog",
      f7SingleLayout(
        navbar = f7Navbar(title = "f7Dialog"),
        f7Button(inputId = "goButton", "Go!"),
        f7Button(inputId = "update", "Update config")
      )
    ),
    server = function(input, output, session) {
      observeEvent(input$goButton,{
        f7Dialog(
          title = "Dialog title",
          text = "This is an alert dialog"
        )
      })

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

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

```

updateF7Entity	<i>Update Framework7 entity</i>
----------------	---------------------------------

Description

updateF7Entity allows to update any Framework7 instance from the server. For each entity, the list of updatable properties may significantly vary. Please refer to the Framework7 documentation at <https://v5.framework7.io/docs/>.

Usage

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

Arguments

id	Element id.
options	Configuration list. Tightly depends on the entity. See https://v5.framework7.io/docs/ .
session	Shiny session object.

Examples

```
# Update action sheet instance
if (interactive()) {
  library(shiny)
  library(shinyMobile)
  shinyApp(
    ui = f7Page(
      title = "Simple Dialog",
      f7SingleLayout(
        navbar = f7Navbar(title = "Update action sheet instance"),
        f7Button(inputId = "goButton", "Go!"),
        f7Button(inputId = "update", "Update config")
      )
    ),
    server = function(input, output, session) {
      observeEvent(input$goButton, {
        f7ActionSheet(
          grid = TRUE,
          id = "action1",
          buttons = list(
            list(
              text = "Notification",
              icon = f7Icon("info"),
              color = NULL
            ),
            list(
              text = "Dialog",
              icon = f7Icon("lightbulb_fill"),
            )
          )
        )
      })
    }
  )
}
```

```

        color = NULL
      )
    )
  }
})

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

```

 updateF7Tabs

Update a Framework 7 tabsetPanel

Description

Update [f7Tabs](#).

Usage

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

Arguments

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

Examples

```

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

  subtabs_ui <- function(id) {
    ns <- NS(id)
  }
}

```

```

tagList(
  f7Toggle(inputId = ns("updateSubTab"), label = "Update SubTab", checked = FALSE),
  f7Tabs(
    id = ns("subtabdemo"),
    style = "strong",
    animated = FALSE,
    f7Tab(tabName = "SubTab 1", "SubTab 1"),
    f7Tab(tabName = "SubTab 2", "SubTab 2", active = TRUE),
    f7Tab(tabName = "SubTab 3", "SubTab 3")
  )
)
}

subtabs <- function(input, output, session) {
  observeEvent(input$updateSubTab, {
    selected <- ifelse(input$updateSubTab, "SubTab 1", "SubTab 2")
    updateF7Tabs(session, id = "subtabdemo", selected = selected)
  })
  return(reactive(input$subtabdemo))
}

shinyApp(
  ui = f7Page(
    title = "Tab Layout",
    f7TabLayout(
      navbar = f7Navbar(
        title =
          f7Flex(
            HTML(paste("Selected Tab:", textOutput("selectedTab"))),
            HTML(paste("Selected Subtab:", textOutput("selectedSubTab")))
          )
      ,
      subNavbar = f7SubNavbar(
        f7Flex(
          f7Toggle(inputId = "updateTab", label = "Update Tab", checked = TRUE),
          subtabs_ui("subtabs1")[[1]]
        )
      )
    ),
    f7Tabs(
      id = "tabdemo",
      swipeable = TRUE,
      animated = FALSE,
      f7Tab(
        tabName = "Tab 1",
        subtabs_ui("subtabs1")[[2]]
      ),
      f7Tab(tabName = "Tab 2", "Tab 2"),
      f7Tab(tabName = "Tab 3", "Tab 3")
    )
  )
),

```

```

server = function(input, output, session) {
  output$selectedTab <- renderText(input$tabdemo)
  observeEvent(input$updateTab, {
    selected <- ifelse(input$updateTab, "Tab 1", "Tab 2")
    updateF7Tabs(id = "tabdemo", selected = selected)
  })
  subtab <- callModule(subtabs, "subtabs1")
  output$selectedSubTab <- renderText(subtab())
}
)
# with hidden tabs
shinyApp(
  ui <- f7Page(
    title = "shinyMobile",
    f7TabLayout(
      navbar = f7Navbar(
        title = "Update Tabs with hidden tab",
        subtitle = "",
        hairline = TRUE,
        shadow = TRUE,
        bigger = FALSE,
        transparent = TRUE
      ),
      f7Tabs(
        id = 'tabs',
        animated = TRUE,
        f7Tab(
          active = TRUE,
          tabName = 'Main tab',
          icon = f7Icon('doc_text'),
          h1("This is the first tab."),
          f7Button(inputId = 'goto', label = 'Go to hidden tab')
        ),
        f7Tab(
          tabName = 'Second tab',
          icon = f7Icon('bolt_horizontal'),
          h1('This is the second tab.')
        ),
        f7Tab(
          tabName = 'Hidden tab',
          hidden = TRUE,
          h1('This is a tab that does not appear in the tab menu.
          Yet, you can still access it.')
        )
      )
    )
  ),
  server = function(input, output, session) {
    observe(print(input$tabs))
    observeEvent(input$goto,{
      updateF7Tabs(session = session, id = 'tabs', selected = 'Hidden tab')
    })
  }
)

```

```
    }  
  )  
}
```

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

Description

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

Usage

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

Arguments

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

Examples

```

if (interactive()) {
  library(shiny)
  library(shinyMobile)
  shinyApp(
    ui = f7Page(
      title = "Update virtual list",
      f7SingleLayout(
        navbar = f7Navbar(
          title = "Virtual Lists",
          hairline = FALSE,
          shadow = TRUE
        ),
        # main content
        f7Segment(
          container = "segment",

          f7Button(inputId = "appendItem", "Append Item"),
          f7Button(inputId = "prependItems", "Prepend Items"),
          f7Button(inputId = "insertBefore", "Insert before"),
          f7Button(inputId = "replaceItem", "Replace Item")
        ),
        f7Segment(
          container = "segment",
          f7Button(inputId = "deleteAllItems", "Remove All"),
          f7Button(inputId = "moveItem", "Move Item"),
          f7Button(inputId = "filterItems", "Filter Items")
        ),
        f7Flex(
          uiOutput("itemIndexUI"),
          uiOutput("itemNewIndexUI"),
          uiOutput("itemsFilterUI")
        ),
        f7VirtualList(
          id = "vlist",
          items = lapply(1:5, function(i) {
            f7VirtualListItem(
              title = paste("Title", i),
              subtitle = paste("Subtitle", i),
              header = paste("Header", i),
              footer = paste("Footer", i),
              right = paste("Right", i),
              content = i,
              media = img(src = "https://cdn.framework7.io/placeholder/fashion-88x88-3.jpg")
            )
          })
        )
      )
    ),
    server = function(input, output, session) {

      output$itemIndexUI <- renderUI({

```

```
    req(input$vlist$length > 2)
  f7Stepper(
    inputId = "itemIndex",
    label = "Index",
    min = 1,
    value = 2,
    max = input$vlist$length
  )
})

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

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

observe(print(input$vlist))

observeEvent(input$appendItem, {
  updateF7VirtualList(
    id = "vlist",
    action = "appendItem",
    item = f7VirtualListItem(
      title = "New Item Title",
      right = "New Item Right",
      content = "New Item Content",
      media = img(src = "https://cdn.framework7.io/placeholder/fashion-88x88-1.jpg")
    )
  )
})
```

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

observeEvent(input$insertBefore, {
  updateF7VirtualList(
    id = "vlist",
    action = "insertItemBefore",
    index = input$itemIndex,
    item = f7VirtualListItem(
      title = "New Item Title",
      content = "New Item Content",
      media = img(src = "https://cdn.framework7.io/placeholder/fashion-88x88-1.jpg")
    )
  )
})

observeEvent(input$replaceItem, {
  updateF7VirtualList(
    id = "vlist",
    action = "replaceItem",
    index = input$itemIndex,
    item = f7VirtualListItem(
      title = "Replacement",
      content = "Replacement Content",
      media = img(src = "https://cdn.framework7.io/placeholder/fashion-88x88-1.jpg")
    )
  )
})

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

observeEvent(input$moveItem, {
  updateF7VirtualList(
    id = "vlist",
```



```

        action = "moveItem",
        oldIndex = input$itemIndex,
        newIndex = input$itemNewIndex
      )
    })

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

```

 validateF7Input

Framework7 input validation

Description

validateF7Input is a function to validate a given shinyMobile input.

Usage

```

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

```

Arguments

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

Note

Only works for [f7Text](#), [f7Password](#), [f7TextArea](#) and [f7Select](#). See more at <https://framework7.io/docs/inputs.html>.

Examples

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

  shinyApp(
    ui = f7Page(
      title = "Validate inputs",
      f7SingleLayout(
        navbar = f7Navbar(title = "validateF7Input"),
        f7Text(
          inputId = "caption",
          label = "Caption",
          value = "Data Summary"
        ),
        verbatimTextOutput("value"),
        hr(),
        f7Text(
          inputId = "caption2",
          label = "Enter a number",
          value = 1
        )
      )
    ),
    server = function(input, output, session) {
      observe({
        validateF7Input(inputId = "caption", info = "Whatever")
        validateF7Input(
          inputId = "caption2",
          pattern = "[0-9]*",
          error = "Only numbers please!"
        )
      })
      output$value <- renderPrint({ input$caption })
    }
  )
}
```

Index

add_dependencies, 6
add_f7icons_dependencies, 6
add_framework7_deps, 7
add_pwa_deps, 7
add_pwacompat_deps, 7
add_shinyMobile_deps, 8
addF7Popover, 4
addF7Tooltip (f7Tooltip), 148

create_app_ui, 8
createSelectOptions, 8

f7Accordion, 9, 9
f7AccordionItem (f7Accordion), 9
f7ActionSheet, 11, 11
f7Align, 16
f7AppBar, 17, 17, 82, 99, 109, 119, 129
f7AutoComplete, 19
f7Back, 17, 18
f7Back (f7AppBar), 17
f7Badge, 22, 58
f7Block, 9, 23, 23, 25
f7BlockFooter, 23, 25, 25
f7BlockHeader, 23
f7BlockHeader (f7Block), 23
f7BlockTitle, 25
f7Button, 26, 26, 102
f7Card, 28, 105
f7Checkbox, 32, 32
f7CheckboxGroup, 34
f7Chip, 35
f7Col, 37
f7ColorPicker, 37
f7DatePicker, 39, 40
f7Dialog, 42
f7DownloadButton, 45
f7ExpandableCard, 28, 29
f7ExpandableCard (f7Card), 28
f7Fab, 46, 46, 47, 48
f7FabClose, 47
f7FabMorphTarget (f7Fabs), 47
f7Fabs, 46, 47, 47
f7File, 50
f7Flex, 18, 37, 52
f7Float, 53
f7Found, 54, 54
f7Gallery, 54
f7Gauge, 55
f7HideOnEnable, 57
f7HideOnSearch, 58, 99
f7HidePreloader (showF7Preloader), 157
f7Icon, 26, 58, 60, 66, 113, 129, 133, 145, 151
f7Item, 59, 60, 87
f7Items, 60
f7Link, 60, 147
f7List, 61, 66, 150, 151
f7ListGroup, 61, 63, 63
f7ListIndex, 63
f7ListIndexItem, 65
f7ListItem, 61, 63, 65, 125
f7Login, 66, 66
f7LoginServer (f7Login), 66
f7Margin, 70
f7Menu, 71, 71
f7MenuDropdown, 71
f7MenuDropdown (f7Menu), 71
f7MenuDropdownDivider, 72
f7MenuDropdownDivider (f7Menu), 71
f7MenuItem, 71, 72
f7MenuItem (f7Menu), 71
f7Message, 74
f7Message (f7Messages), 74
f7MessageBar, 73, 73, 129
f7Messages, 73, 74, 75, 76
f7Navbar, 17, 77, 77, 82, 109, 119, 124, 129
f7Next, 17, 18
f7Next (f7AppBar), 17
f7NotFound, 79, 99
f7Notif, 79

- f7Padding, 81
- f7Page, 82
- f7Panel, 17, 78, 83, 83, 86, 109, 119, 129
- f7PanelItem, 86, 87
- f7PanelItem (f7PanelMenu), 86
- f7PanelMenu, 83, 84, 86
- f7Password, 87, 169
- f7PhotoBrowser, 88
- f7Picker, 89
- f7Popup, 92
- f7Progress, 94
- f7Radio, 95
- f7Row, 97
- f7Searchbar, 18, 54, 57, 58, 79, 98, 99, 101
- f7SearchbarTrigger, 78, 99, 101
- f7SearchIgnore, 101
- f7Segment, 27, 102
- f7Select, 8, 104, 116, 169
- f7Shadow, 105
- f7Sheet, 107, 107, 134
- f7SingleLayout, 77, 82, 109, 118
- f7Skeleton, 110
- f7Slide, 112, 127
- f7Slider, 112
- f7SmartSelect, 8, 116, 117
- f7SocialCard (f7Card), 28
- f7SplitLayout, 59, 77, 82, 83, 86, 118
- f7Stepper, 120
- f7SubNavbar, 78, 124
- f7Swipeout, 125, 125
- f7SwipeoutItem, 125
- f7SwipeoutItem (f7Swipeout), 125
- f7Swiper, 112, 127
- f7Tab, 59, 128, 134, 153, 156
- f7TabLayout, 77, 82, 113, 129, 134
- f7Table, 132
- f7TabLink, 133, 134
- f7Tabs, 17, 18, 129, 133, 134, 134, 147, 153, 156, 162
- f7TapHold, 137, 137
- f7Text, 138, 169
- f7TextArea, 140, 169
- f7Timeline, 142, 142
- f7TimelineItem (f7Timeline), 142
- f7Toast, 144
- f7Toggle, 145
- f7Toolbar, 82, 109, 119, 147
- f7Tooltip, 148
- f7VirtualList, 150, 150, 165
- f7VirtualListItem, 151
- f7VirtualListItem (f7VirtualList), 150
- getF7Colors, 113, 153
- HTML, 65
- insertF7Tab, 153
- preview_mobile, 9, 155
- removeF7Tab, 156
- showF7Preloader, 157
- toggleF7Popover, 4
- toggleF7Popover (addF7Popover), 4
- updateF7Accordion, 9
- updateF7Accordion (f7Accordion), 9
- updateF7ActionSheet, 11
- updateF7ActionSheet (f7ActionSheet), 11
- updateF7App, 159
- updateF7AutoComplete (f7AutoComplete), 19
- updateF7Button (f7Button), 26
- updateF7Card (f7Card), 28
- updateF7Checkbox (f7Checkbox), 32
- updateF7DatePicker (f7DatePicker), 39
- updateF7Entity, 161
- updateF7Fab (f7Fab), 46
- updateF7Fabs (f7Fabs), 47
- updateF7Gauge (f7Gauge), 55
- updateF7Login (f7Login), 66
- updateF7MenuDropdown (f7Menu), 71
- updateF7MessageBar (f7MessageBar), 73
- updateF7Messages, 74, 75
- updateF7Messages (f7Messages), 74
- updateF7Navbar (f7Navbar), 77
- updateF7Panel (f7Panel), 83
- updateF7Picker (f7Picker), 89
- updateF7Progress (f7Progress), 94
- updateF7Radio (f7Radio), 95
- updateF7Select (f7Select), 104
- updateF7Sheet, 108
- updateF7Sheet (f7Sheet), 107
- updateF7Slider (f7Slider), 112
- updateF7SmartSelect (f7SmartSelect), 116
- updateF7Stepper (f7Stepper), 120

updateF7Tabs, [129](#), [162](#)
updateF7Text (f7Text), [138](#)
updateF7TextArea (f7TextArea), [140](#)
updateF7Toggle (f7Toggle), [145](#)
updateF7Tooltip (f7Tooltip), [148](#)
updateF7VirtualList, [165](#)

validateCssUnit(), [46](#)
validateF7Input, [169](#)