Package 'shinyGovstyle'

April 13, 2026

Title Custom Gov Style Inputs for Shiny

Version 0.2.0

Description Collection of 'shiny' application styling that are based on the GOV.UK Design System. See <https://design-system.service.gov.uk/components/> for details.

License GPL (>= 3)

URL https://github.com/dfe-analytical-services/shinyGovstyle

BugReports https://github.com/dfe-analytical-services/shinyGovstyle/issues

Depends R (>= 4.1.0)

Imports htmltools, jsonlite, lifecycle, purrr, reactable, readODS, readr, shiny (>= 0.14), shinyjs, stringr, writexl

Suggests bslib, knitr, rlang, rmarkdown, shinytest2, testthat

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

NeedsCompilation no

Author Ross Wyatt [aut], Cam Race [aut, cre], Sarah Wong [aut], Richard Bielby [ctb], Charlotte Foster [ctb], Jeni Martin [ctb], Andrew Baker [ctb]

Maintainer Cam Race <camrace8@gmail.com>

Repository CRAN

Date/Publication 2026-04-13 12:40:02 UTC

Contents

`accordion` *Accordion Function*

Description

This function inserts a accordion

Usage

```
accordion(inputId, titles, descriptions)
```

Arguments

`inputId`	Input Id for the accordion
`titles`	Add the titles for the accordion
`descriptions`	Add the main text for the accordion

Value

an accordion HTML shiny tag object

See Also

Other Govstyle tables tabs and accordions: `govReactable()`, `govReactable-shiny`, `govTable()`, `govTabs()`

Examples

```
ui <- shiny::fluidPage(
  shinyGovstyle::header(
    org_name = "Example",
    service_name = "User Examples",
    logo = "shinyGovstyle/images/moj_logo.png",
    logo_alt_text = "Ministry of Justice logo"
  ),
  shinyGovstyle::banner(
    inputId = "banner", type = "beta", 'This is a new service'
  ),
  shinyGovstyle::gov_layout(
```

```
    size = "two-thirds",
    accordion(
      "acc1",
      c(
        "Writing well for the web",
        "Writing well for specialists",
        "Know your audience",
        "How people read"
      ),
      c(
        "This is the content for Writing well for the web.",
        "This is the content for Writing well for specialists.",
        "This is the content for Know your audience.",
        "This is the content for How people read."
      )
    )
  ),
  shinyGovstyle::footer(full = TRUE)
)

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

if (interactive()) shiny::shinyApp(ui = ui, server = server)
```

---

backlink_Input *Back Link Function*

Description

This function adds a back link to the page

Usage

```
backlink_Input(inputId, label = "Back")
```

Arguments

inputId	The input slot that will be used to access the value
label	The link text for the backlink, default is "Back"

Value

a backlink HTML shiny tag object

See Also

Other Govstyle navigation: contents_link(), service_navigation(), update_service_navigation()

Examples

```
ui <- shiny::fluidPage(
  header(
    org_name = "Example",
    service_name = "User Examples",
    logo = "shinyGovstyle/images/moj_logo.png"
  ),
  shiny::navlistPanel(
    "",
    id = "nav",
    widths = c(2, 10),
    well = FALSE,
    # Create first panel
    shiny::tabPanel(
      "Select Types",
      value = "panel1",
      gov_layout(
        size = "two-thirds",
        backlink_Input("link1"),
        shiny::tags$br(),
        shiny::tags$br()
      )
    ),
    shiny::tabPanel(
      "Tab2",
      value = "panel2"
    )
  ),
  shinyGovstyle::footer(full = TRUE)
)

server <- function(input, output, session) {
  # Slightly confused in that it goes forward rather than back
  # but shows how to use
  observeEvent(input$link1, {
    updateTabsetPanel(session, "nav", selected = "panel2")
  })
}

if (interactive()) shinyApp(ui = ui, server = server)
```

---

bad_link_text *Lookup for bad link text*

Description

A single column data frame, listing out known examples of bad link text that check for in the `external_link()` function.

Usage

```
bad_link_text
```

Format

`bad_link_text`:

A data frame with 53 rows and 1 columns:

bad_link_text Lower cased examples of non-descriptive link text

Details

We've started curating this list so we can create automated checks to help all link text to be as descriptive as possible in line with WCAG 2.2 success criteria 2.4.4: Link Purpose (In Context).

Source

Curated by explore.statistics@education.gov.uk

---

banner *Banner Function*

Description

This function create a detail component that you can click for further details.

Usage

```
banner(inputId, type, label)
```

Arguments

`inputId`	The input slot that will be used to access the value
`type`	Main type of label e.g. alpha or beta. Can be any word
`label`	text to display

Value

a banner HTML shiny tag object

See Also

Other Govstyle page structure: `cookieBanner()`, `footer()`, `gov_layout()`, `header()`, `layouts`, `skip_to_main()`

Examples

```
ui <- shiny::fluidPage(
  shinyGovstyle::header(
    org_name = "Example",
    service_name = "User Examples",
    logo = "shinyGovstyle/images/moj_logo.png"
  ),
  shinyGovstyle::banner(
    inputId = "banner", type = "Beta", 'This is a new service'
  )
)

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

if (interactive()) shinyApp(ui = ui, server = server)
```

`button_Input` *Button Function*

Description

This function create a gov style button

Usage

```
button_Input(inputId, label, type = "default")
```

Arguments

`inputId`	The input slot that will be used to access the value
`label`	Display label for the control, or `NULL` for no label
`type`	The type of button. Options are default, start, secondary and warning. Defaults to "default"

Value

a HTML button shiny tag object

See Also

Other Govstyle select inputs: `checkbox_Input()`, `file_Input()`, `radio_button_Input()`, `select_Input()`

Examples

```
ui <- shiny::fluidPage(
  shinyGovstyle::header(
    org_name = "Example",
    service_name = "User Examples",
    logo = "shinyGovstyle/images/moj_logo.png"
  ),
  shinyGovstyle::gov_layout(
    size = "two-thirds",
    shinyGovstyle::button_Input(
      inputId = "btn1",
      label = "Continue",
      type = "default"
    )
  ),
  shinyGovstyle::footer(full = TRUE)
)

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

if (interactive()) shinyApp(ui = ui, server = server)
```

---

case_data *Case data*

Description

Simple example dataset containing case management data, used to demonstrate tabs in the example app.

Usage

```
case_data
```

Format

A data frame with 12 rows and 4 variables:

tabs The tab the data belongs to

case_manager The name of the case manager

cases_open Number of open cases

cases_closed Number of closed cases

Source

Random data generated for the shinyGovstyle package

checkbox_Input *Checkbox Function*

Description

This function inserts a checkbox group

Usage

```
checkbox_Input(
  inputId,
  cb_labels,
  checkboxIds,
  label,
  hint_label = NULL,
  small = FALSE,
  error = FALSE,
  error_message = NULL
)
```

Arguments

inputId	Input Id for the group of checkboxes
cb_labels	Add the names of the options that will appear
checkboxIds	Add the values for each checkbox
label	Insert the text for the checkbox group
hint_label	Insert optional hint/secondary text. Defaults to NULL
small	change the sizing to a small version of the checkbox. Defaults to FALSE
error	Whenever you want to include error handle on the component
error_message	If you want a default error message

Value

a checkbox HTML shiny tag object

See Also

Other Govstyle select inputs: button_Input(), file_Input(), radio_button_Input(), select_Input()

Examples

```
ui <- shiny::fluidPage(
  # Required for error handling function
  shinyjs::useShinyjs(),
  shinyGovstyle::header(
    org_name = "Example",
```

```
    service_name = "User Examples",
    logo="shinyGovstyle/images/moj_logo.png"),
  shinyGovstyle::banner(
    inputId = "banner", type = "beta", 'This is a new service'),
  shinyGovstyle::gov_layout(size = "two-thirds",
    # Simple checkbox
    shinyGovstyle::checkbox_Input(
      inputId = "check1",
      cb_labels = c("Option 1", "Option 2", "Option 3"),
      checkboxIds = c("op1", "op2", "op3"),
      label = "Choice option"
    ),
    # Error checkbox
    shinyGovstyle::checkbox_Input(
      inputId = "check2",
      cb_labels = c("Option 1", "Option 2", "Option 3"),
      checkboxIds = c("op1", "op2", "op3"),
      label = "Choice option",
      hint_label = "Select the best fit",
      error = TRUE,
      error_message = "Select one"
    ),
    # Button to trigger error
    shinyGovstyle::button_Input(inputId = "submit", label = "Submit")
  ),
  shinyGovstyle::footer(full = TRUE)
)

server <- function(input, output, session) {
  # Trigger error on blank submit of eventId2
  observeEvent(input$submit, {
    if (is.null(input$check2)){
      shinyGovstyle::error_on(inputId = "check2")
    } else {
      shinyGovstyle::error_off(inputId = "check2")
    }
  })
}

if (interactive()) shinyApp(ui = ui, server = server)
```

`contents_link` *Contents link function*

Description

[Deprecated]

Usage

```
contents_link(link_text, input_id, subcontents_text_list, subcontents_id_list)
```

Arguments

link_text	vector of link text for contents
input_id	contents button Id
subcontents_text_list	vector of link text for subcontents
subcontents_id_list	vector of link Ids for subcontents. If missing automatically matches to Id in heading_text()

Details

This function creates an action link to nav between tabs and optionally link to subcontents headers.

Value

an action button HTML shiny tag object

See Also

Other Govstyle navigation: backlink_Input(), service_navigation(), update_service_navigation()

Examples

```
ui <- shiny::fluidPage(
  gov_row(
    # Nav columns
    shiny::column(
      width = 3,
      id = "nav", # DO NOT REMOVE ID

      # Contents box
      shiny::tags$div(
        id = "govuk-contents-box", # DO NOT REMOVE ID
        class = "govuk-contents-box", # DO NOT REMOVE CLASS

        shiny::tags$h2("Contents"),

        # Text types tab
        contents_link(
          "Text Types",
          "text_types_button",
          subcontents_text_list = c(
            "date_Input",
            "text_Input",
            "text_area_Input",
            "button_Input"
          ),
          subcontents_id_list = c(NA, NA, NA, "button_input_text_types")
        ),

        # Tables tabs and accordions tab
```

```
    contents_link(
      "Tables, tabs and accordions",
      "tables_tabs_and_accordions_button",
      subcontents_text_list = c(
        "govTable",
        "govTabs",
        "accordions",
        "button_Input"
      ),
      subcontents_id_list = c(
        NA,
        NA,
        NA,
        "button_input_tables_tabs_accordions"
      )
    ),

    contents_link(
      "Cookies",
      "cookies_button"
    ),
  )
),

shiny::column(
  width = 9,
  id = "main_col", # DO NOT REMOVE ID

  # Set up a nav panel so everything not on single page
  shiny::tabsetPanel(
    type = "hidden",
    id = "tab-container", # DO NOT REMOVE ID

    shiny::tabPanel(
      "Text Types",
      value = "text_types",
      gov_layout(
        size = "two-thirds",
        backlink_Input("back1"),
        heading_text("Page 2", size = "l"),
        label_hint(
          "label2",
          paste(
            "These are some examples of the",
            "types of user text inputs that you can use"
          )
        ),
        heading_text("date_Input", size = "s"),
        date_Input(
          inputId = "date1",
          label = "What is your date of birth?",
          hint_label = "For example, 31 3 1980"
        ),
```

```
    heading_text("text_Input", size = "s"),
    text_Input(
      inputId = "txt1",
      label = "Event name"
    ),
    heading_text("text_area_Input", size = "s"),
    text_area_Input(
      inputId = "text_area1",
      label = "Can you provide more detail?",
      hint_label = paste(
        "Do not include personal or financial",
        "information, like your National Insurance",
        "number or credit card details."
      )
    ),
    text_area_Input(
      inputId = "text_area2",
      label = "How are you today?",
      hint_label = "Leave blank to trigger error",
      error = TRUE,
      error_message = "Please do not leave blank",
      word_limit = 300
    ),
    heading_text(
      "button_Input",
      size = "s",
      id = "button_input_text_types"
    ),
    button_Input("btn2", "Go to next page"),
    button_Input(
      "btn3",
      "Check for errors",
      type = "warning"
    )
  )
),

shiny::tabPanel(
  "Tables, tabs and accordions",
  value = "tables_tabs_and_accordions",
  gov_layout(
    size = "two-thirds",
    backlink_Input("back2"),
    heading_text("Page 3", size = "l"),
    label_hint(
      "label3",
      paste(
        "These are some examples of using tabs",
        "type tables"
      )
    ),
    heading_text("govTable", size = "s"),
    heading_text("govTabs", size = "s"),
```

```
            heading_text("accordions", size = "s"),
            shinyGovstyle::accordion(
              "acc1",
              c(
                "Writing well for the web",
                "Writing well for specialists",
                "Know your audience",
                "How people read"
              ),
              c(
                paste(
                  "This is the content for Writing well",
                  "for the web."
                ),
                paste(
                  "This is the content for Writing well",
                  "for specialists."
                ),
                paste(
                  "This is the content for",
                  "Know your audience."
                ),
                "This is the content for How people read."
              )
            ),

            heading_text(
              "button_Input",
              size = "s",
              id = "button_input_tables_tabs_accordions"
            ),
            button_Input("btn4", "Go to next page"),
          )
        ),

        ##################### Create cookie panel #########
        shiny::tabPanel(
          "Cookies",
          value = "panel-cookies",
          gov_layout(
            size = "two-thirds",
            heading_text("Cookie page", size = "l"),
            label_hint(
              "label-cookies",
              "This an example cookie page"
            )
          )
        )
      )
    )
  ), # end of main_col
  footer(TRUE)
) # end of gov_row
```

```
server <- function(input, output, session) {
  # Tab nav
  shiny::observeEvent(input$back2, {
    shiny::updateTabsetPanel(
      session,
      "tab-container",
      selected = "text_types"
    )
  })

  shiny::observeEvent(input$tables_tabs_and_accordions_button, {
    shiny::updateTabsetPanel(
      session,
      "tab-container",
      selected = "tables_tabs_and_accordions"
    )
  })

  shiny::observeEvent(input$cookies_button, {
    shiny::updateTabsetPanel(
      session,
      "tab-container",
      selected = "panel-cookies"
    )
  })
} # end of server

if (interactive()) shiny::shinyApp(ui = ui, server = server)
```

`cookieBanner` *Cookie Banner Function*

Description

This function creates a cookie banner. You need to have `shinyjs::useShinyjs()` enabled for this to work. All the Ids are preset. See example for how to structure.

Usage

```
cookieBanner(service_name)
```

Arguments

`service_name`	Name for this service to add to banner

Value

a cookie banner HTML shiny tag object

See Also

Other Govstyle page structure: banner(), footer(), gov_layout(), header(), layouts, skip_to_main()

Examples

```
ui <- shiny::fluidPage(
  shinyGovstyle::header(
    org_name = "Example",
    service_name = "User Examples",
    logo="shinyGovstyle/images/moj_logo.png"),
  #Needs shinyjs to work
  shinyjs::useShinyjs(),
  shinyGovstyle::cookieBanner("The best thing"),
  shinyGovstyle::gov_layout(size = "two-thirds"),
  shinyGovstyle::footer(full = TRUE)
)

server <- function(input, output, session) {
  # Set of observeEvent to create a path through the cookie banner
  observeEvent(input$cookieAccept, {
    shinyjs::show(id = "cookieAcceptDiv")
    shinyjs::hide(id = "cookieMain")
  })

  observeEvent(input$cookieReject, {
    shinyjs::show(id = "cookieRejectDiv")
    shinyjs::hide(id = "cookieMain")
  })

  observeEvent(input$hideAccept, {
    shinyjs::toggle(id = "cookieDiv")
  })

  observeEvent(input$hideReject, {
    shinyjs::toggle(id = "cookieDiv")
  })

  observeEvent(input$cookieLink, {
    # Need to link here to where further info is located. You can use
    # updateTabsetPanel to have a cookie page for instance
  })

}
if (interactive()) shinyApp(ui = ui, server = server)
```

---

date_Input *Date Input Function*

Description

This function create a date input that follows GDS component

Usage

```
date_Input(
  inputId,
  label,
  hint_label = NULL,
  error = FALSE,
  error_message = NULL,
  day = NULL,
  month = NULL,
  year = NULL
)
```

Arguments

inputId	The input slot that will be used to access the value
label	Display label for the control, or NULL for no label
hint_label	Display hint label for the control, or NULL for no hint label
error	Whenever to include error components.Defaults to FALSE
error_message	Error handling message? Defaults to NULL
day	Select a default day on start up. Defaults to NULL
month	Select a default month on start up. Defaults to NULL
year	Select a default year on start up. Defaults to NULL

Value

a data input HTML shiny tag object

See Also

Other Govstyle text types: gov_list(), heading_text(), input_field(), text_Input(), text_area_Input(), word_count()

Examples

```
ui <- shiny::fluidPage(
  # Required for error handling function.
  shinyjs::useShinyjs(),
  shinyGovstyle::header(
    org_name = "Example",
    service_name = "User Examples",
    logo = "shinyGovstyle/images/moj_logo.png"
  ),
  shinyGovstyle::banner(
    inputId = "banner", type = "beta", 'This is a new service'
  ),
  shinyGovstyle::gov_layout(
    size = "two-thirds",
    # Simple date input
```

```
    shinyGovstyle::date_Input(
      inputId = "dob_input",
      label = "Please enter your birthday"
    ),
    # Error date input
    shinyGovstyle::date_Input(
      inputId = "dob_input2",
      label = "Please enter your birthday",
      hint_label = "For example, 12 11 2007",
      error = TRUE
    ),
    # Button to trigger error
    shinyGovstyle::button_Input(inputId = "submit", label = "Submit")
  ),
  shinyGovstyle::footer(full = TRUE)
)

server <- function(input, output, session) {
  # Trigger error on blank submit of dob_input2
  observeEvent(input$submit, {
    if (input$dob_input2 == "//") {
      shinyGovstyle::error_on(inputId = "dob_input2")
    } else {
      shinyGovstyle::error_off(inputId = "dob_input2")
    }
  })
}

if (interactive()) shinyApp(ui = ui, server = server)
```

`details` *Details Function*

Description

This function create a detail component that you can click for further details.

Usage

```
details(inputId, label, help_text)
```

Arguments

`inputId`	The input slot that will be used to access the value
`label`	Main label text
`help_text`	Additional help information in the component

Value

a details box HTML shiny tag object

See Also

Other Govstyle feedback types: gov_summary(), insert_text(), label_hint(), noti_banner(), panel_output(), tag_Input(), value_box(), warning_text()

Examples

```
ui <- shiny::fluidPage(
  shinyGovstyle::header(
    org_name = "Example",
    service_name = "User Examples",
    logo = "shinyGovstyle/images/moj_logo.png"
  ),
  shinyGovstyle::gov_layout(
    size = "two-thirds",
    shinyGovstyle::details(
      inputId = "help_div",
      label = "Help with form",
      help_text = "To complete the form you need to fill it in..."
    )
  ),
  shinyGovstyle::footer(full = TRUE)
)

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

if (interactive()) shinyApp(ui = ui, server = server)
```

---

download_button *Download button*

Description

The download_button() provides a standard way to provide a download link, which facilitates important accessible / positive user experience elements, namely:

- file type
- file size

These are necessary in order for users to understand what they are downloading, both in terms of being able to decide if they are comfortable with downloading the file over their current connection and if it's in a form they're able to deal with once it is downloaded. If the exact file size is not easily determined, then it may be acceptable to provide an estimate or an upper limit.

Usage

```
download_button(outputId, button_label, file_type = "CSV", file_size = NULL)
```

Arguments

outputId	The name of the output slot that the shiny::downloadHandler() is assigned to
button_label	Text that will appear describing the download action. Vague text like 'click here' or 'here' will cause an error, as will ending in a full stop. Leading and trailing white space will be automatically trimmed. If the string is shorter than 7 characters a console warning will be thrown. There is no way to hush this other than providing more detail
file_type	The file type to be download (default: CSV)
file_size	The file size if known. Needs to be a string ending in one of KB, MB, GB or rows

Value

shiny tag object

See Also

Other Govstyle actions: download_link(), download_radios(), download_radios_handler(), external_link()

Examples

```
ui <- shiny::fluidPage(
  gov_text("Choose a data set to download."),
  select_Input(
    "dataset",
    "Data set",
    select_text = c("Car road tests", "New York air quality"),
    select_value = c("mtcars", "airquality")
  ),
  download_button(
    "download_data",
    "Download selected data set",
    file_size = "4 KB"
  )
)

server <- function(input, output) {
  # The requested data set
  data <- reactive({
    get(input$dataset)
  })

  output$download_data <- downloadHandler(
    filename = function() {
      # Use the selected dataset as the suggested file name
      paste0(input$dataset, ".csv")
    },
    content = function(file) {
```

```
      # Write the dataset to the `file` that will be downloaded
      write.csv(data(), file)
    }
  )
}

if (interactive()) shiny::shinyApp(ui, server)
```

---

download_link *Download link*

Description

The download_link() provides a standard way to provide a download link, which facilitates important accessible / positive user experience elements, namely:

- file type
- file size

These are necessary in order for users to understand what they are downloading, both in terms of being able to decide if they are comfortable with downloading the file over their current connection and if it's in a form they're able to deal with once it is downloaded. If the exact file size is not easily determined, then it may be acceptable to provide an estimate or an upper limit.

Usage

```
download_link(outputId, link_text, file_type = "CSV", file_size = NULL)
```

Arguments

outputId	The name of the output slot that the shiny::downloadHandler() is assigned to
link_text	Text that will appear describing the download action. Vague text like 'click here' or 'here' will cause an error, as will ending in a full stop. Leading and trailing white space will be automatically trimmed. If the string is shorter than 7 characters a console warning will be thrown. There is no way to hush this other than providing more detail
file_type	The file type to be download (default: CSV)
file_size	The file size if known. Needs to be a string ending in one of KB, MB, GB or rows

Value

shiny tag object

See Also

Other Govstyle actions: download_button(), download_radios(), download_radios_handler(), external_link()

Examples

```
ui <- shiny::fluidPage(
  gov_text("Choose a data set to download."),
  select_Input(
    "dataset",
    "Data set",
    select_text = c("Car road tests", "New York air quality"),
    select_value = c("mtcars", "airquality")
  ),
  gov_text(
    download_link(
      "download_data",
      "Download selected data set",
      file_size = "4 KB"
    )
  )
)

server <- function(input, output) {
  # The requested data set
  data <- reactive({
    get(input$dataset)
  })

  output$download_data <- downloadHandler(
    filename = function() {
      # Use the selected dataset as the suggested file name
      paste0(input$dataset, ".csv")
    },
    content = function(file) {
      # Write the dataset to the `file` that will be downloaded
      write.csv(data(), file)
    }
  )
}

if (interactive()) shiny::shinyApp(ui, server)
```

---

download_radios *Download with extension radios*

Description

Download with extension radios

Usage

```
download_radios(
  id = "download_radios",
  download_type = "table",
```

```
  file_types = c("CSV", "ODS", "XLSX"),
  file_sizes = c("< 1 GB", "< 1 GB", "< 1 GB"),
  small = FALSE
)
```

Arguments

id	Shiny element Id. Default is "download_radios", but must be customised to be unique if multiple instances of this module are being used in a single app. Must match up to the ID of a `download_radios_helper()` instance in server code
download_type	Element being downloaded. Expected to be along the lines of "underlying data", "table", "current data view". Default is "table"
file_types	File formats to offer, can be any combination of "CSV", "XLSX", "ODS". Default is a vector of all 3
file_sizes	Estimated file sizes for each file format. This needs to be a vector of the same length as file_types
small	Set radio buttons to small size (logical, default: FALSE)

Value

UI containing radio selection and download button

See Also

Other Govstyle actions: `download_button()`, `download_link()`, `download_radios_handler()`, `external_link()`

Examples

```
ui <- shiny::fluidPage(
  download_radios("download_file",
    file_types = c("CSV", "ODS"),
    file_sizes = c("2 KB", "5 KB")
  )
)

server <- function(input, output, session) {
  output$download_file <- download_radios_handler(
    "download_file",
    file_name = "simple_data_frame",
    file_contents = mtcars
  )
}

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

`download_radios_handler` *Download with extension radios handler*

Description

Download with extension radios handler

Usage

```
download_radios_handler(id = "download_radios", file_name, file_contents)
```

Arguments

`id`	Shiny element Id. Default is "download_radios", but must be customised to be unique if multiple instances of this module are being used in a single app. Must match up to the ID of a `download_radios()` instance in UI code
`file_name`	Name of the file to be downloaded
`file_contents`	Contents to write to the download file

Value

Output for use with `download_radios()`

See Also

Other Govstyle actions: `download_button()`, `download_link()`, `download_radios()`, `external_link()`

Examples

```
ui <- shiny::fluidPage(
  download_radios("download_file",
    file_types = c("CSV", "ODS"),
    file_sizes = c("2 KB", "5 KB")
  )
)

server <- function(input, output, session) {
  output$download_file <- download_radios_handler(
    "download_file",
    file_name = "simple_data_frame",
    file_contents = mtcars
  )
}

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

error_off *Error off Function*

Description

This function turns off the the error o the component, once issues have been sorted.

Usage

```
error_off(inputId)
```

Arguments

inputId	The input Id to turn error handling on for

Value

no return value. This toggles off error CSS

See Also

Other Govstyle errors: error_on(), error_summary(), error_summary_update()

Examples

```
ui <- shiny::fluidPage(
  # Required for error handling function
  shinyjs::useShinyjs(),
  shinyGovstyle::header(
    org_name = "Example",
    service_name = "User Examples",
    logo = "shinyGovstyle/images/moj_logo.png"
  ),
  shinyGovstyle::banner(
    inputId = "banner", type = "beta", 'This is a new service'
  ),
  shinyGovstyle::gov_layout(
    size = "two-thirds",
    # Error text box
    shinyGovstyle::text_Input(
      inputId = "eventId",
      label = "Event Name",
      error = TRUE
    ),
    # Button to trigger error
    shinyGovstyle::button_Input(inputId = "submit", label = "Submit")
  ),
  shinyGovstyle::footer(full = TRUE)
)
```

```
server <- function(input, output, session) {
  # Trigger error on blank submit of eventId2
  observeEvent(input$submit, {
    if (input$eventId != "") {
      shinyGovstyle::error_off(inputId = "eventId")
    } else {
      shinyGovstyle::error_on(
        inputId = "eventId",
        error_message = "Please complete"
      )
    }
  })
}

if (interactive()) shinyApp(ui = ui, server = server)
```

`error_on` *Error on Function*

Description

This function turns on the error for the component. Can be used to validate inputs.

Usage

```
error_on(inputId, error_message = NULL)
```

Arguments

`inputId`	The input id that you to to turn the error on for
`error_message`	if you want to add an additional error message Defaults to NULL, showing the original designed error message

Value

no return value. This toggles on error CSS

See Also

Other Govstyle errors: `error_off()`, `error_summary()`, `error_summary_update()`

Examples

```
ui <- shiny::fluidPage(
  # Required for error handling function
  shinyjs::useShinyjs(),
  shinyGovstyle::header(
    org_name = "Example",
    service_name = "User Examples",
```

```
    logo="shinyGovstyle/images/moj_logo.png"
  ),
  shinyGovstyle::banner(
    inputId = "banner", type = "beta", 'This is a new service'
  ),
  shinyGovstyle::gov_layout(size = "two-thirds",
    # Error text box
    shinyGovstyle::text_Input(
      inputId = "eventId",
      label = "Event Name",
      error = TRUE),
    # Button to trigger error
    shinyGovstyle::button_Input(inputId = "submit", label = "Submit")
  ),
  shinyGovstyle::footer(full = TRUE)
)

server <- function(input, output, session) {
  # Trigger error on blank submit of eventId2
  observeEvent(input$submit, {
    if (input$eventId != ""){
      shinyGovstyle::error_off(inputId = "eventId")
    } else {
      shinyGovstyle::error_on(
        inputId = "eventId",
        error_message = "Please complete"
      )
    }
  })
}

if (interactive()) shinyApp(ui = ui, server = server)
```

`error_summary` *Error Summary Function*

Description

This function loads the error summary component to display error text. This replicates the gov style error boxes linked below: https://design-system.service.gov.uk/components/error-summary/

Usage

```
error_summary(inputId, error_title, error_list)
```

Arguments

`inputId`	The input slot that will be used to access the value
`error_title`	The title for the error summary
`error_list`	A list of text values to be displayed in the error body

Value

an error_summary HTML shiny tag object

See Also

Other Govstyle errors: `error_off()`, `error_on()`, `error_summary_update()`

Examples

```
ui <- shiny::fluidPage(
  shinyjs::useShinyjs(),
  shinyGovstyle::header(
    org_name = "Example",
    service_name = "User Examples",
    logo = "shinyGovstyle/images/moj_logo.png"
  ),
  shinyGovstyle::gov_layout(
    size = "two-thirds",
    error_summary(
      inputId = "errorId",
      error_title = "Error title",
      error_list = c("error item1", "error item2")
    )
  ),
  shinyGovstyle::button_Input("btn1", "Change error summary"),
  shinyGovstyle::footer(full = TRUE)
)

server <- function(input, output, session) {
  shiny::observeEvent(input$btn1, {
    error_summary_update(
      "errorId",
      c("error item1", "error item2", "error item3")
    )},
    ignoreInit = TRUE
  )
}

if (interactive()) shinyApp(ui = ui, server = server)
```

`error_summary_update` *Error Summary Update Function*

Description

This function changes the text that displays in the error summary box. Requires `shinyjs::useShinyjs()` to work.

Usage

```
error_summary_update(inputId, error_list)
```

Arguments

inputId	The input Id of the error summary you want to update
error_list	An updated list of text values to be displayed in the error body

Value

an update error summary box

See Also

Other Govstyle errors: error_off(), error_on(), error_summary()

Examples

```
ui <- shiny::fluidPage(
  shinyjs::useShinyjs(),
  shinyGovstyle::header(
    org_name = "Example",
    service_name = "User Examples",
    logo = "shinyGovstyle/images/moj_logo.png"
  ),
  shinyGovstyle::gov_layout(
    size = "two-thirds",
    error_summary(
      inputId = "errorId",
      error_title = "Error title",
      error_list = c("error item1", "error item2")
    )
  ),
  shinyGovstyle::button_Input("btn1", "Change error summary"),
  shinyGovstyle::footer(full = TRUE)
)

server <- function(input, output, session) {
  shiny::observeEvent(input$btn1, {
    error_summary_update(
      "errorId",
      c("error item1", "error item2", "error item3")
    )
  },
  ignoreInit = TRUE
  )
}

if (interactive()) shinyApp(ui = ui, server = server)
```

`external_link` *External link*

Description

It is commonplace for external links to open in a new tab, and when we do this we should be careful...

This function automatically adds the following to your link:

- `target="_blank"` to open in new tab
- `rel="noopener noreferrer"` to prevent reverse tabnabbing

By default this function also adds "(opens in new tab)" to your link text to warn users of the behaviour.

This also adds "This link opens in a new tab" as a visually hidden span element within the HTML outputted to warn non-visual users of the behaviour.

The function will error if you end with a full stop, give a warning for particularly short link text and will automatically trim any leading or trailing white space inputted into link_text.

If you are displaying lots of links together and want to save space by avoiding repeating (opens in new tab), then you can set add_warning = FALSE and add a line of text above all of the links saying something like 'The following links open in a new tab'.

Related links and guidance:

- Government digital services guidelines on the use of links
- Anchor tag HTML element and its properties
- WCAG 2.2 success criteria 2.4.4: Link Purpose (In Context)
- Web Accessibility standards link text behaviour

Usage

```
external_link(href, link_text, add_warning = TRUE, footer = FALSE)
```

Arguments

`href`	URL that you want the link to point to
`link_text`	Text that will appear describing your link, must be descriptive of the page you are linking to. Vague text like 'click here' or 'here' will cause an error, as will ending in a full stop. Leading and trailing white space will be automatically trimmed. If the string is shorter than 7 characters a console warning will be thrown. There is no way to hush this other than providing more detail
`add_warning`	Boolean for adding "(opens in new tab)" at the end of the link text to warn users of the behaviour. Be careful and consider accessibility before removing the visual warning
`footer`	Apply standard GDS footer CSS styling. Logical, default = FALSE

Details

Intentionally basic wrapper for HTML anchor elements making it easier to create safe external links with standard and accessible behaviour. For more information on how the tag is generated, see `htmltools::tags()`.

Value

shiny tag object

See Also

Other Govstyle actions: `download_button()`, `download_link()`, `download_radios()`, `download_radios_handler()`

Examples

```
external_link("https://shiny.posit.co/", "R Shiny")

external_link(
  "https://shiny.posit.co/",
  "R Shiny",
  add_warning = FALSE
)

# This will trim and show as 'R Shiny'
external_link("https://shiny.posit.co/", "  R Shiny")

# Example of within text
shiny::tags$p(
  "Oi, ", external_link("https://shiny.posit.co/", "R Shiny"), " is great."
)

# Example of multiple links together
shiny::tags$h2("Related resources")
shiny::tags$p("The following links open in a new tab.")
shiny::tags$ul(
  shiny::tags$li(
    external_link(
      "https://shiny.posit.co/",
      "R Shiny documentation",
      add_warning = FALSE
    )
  ),
  shiny::tags$li(
    external_link(
      "https://www.python.org/",
      "Python documentation",
      add_warning = FALSE
    )
  ),
  shiny::tags$li(
    external_link(
      "https://nextjs.org/",
```

```
      "Next.js documentation",
      add_warning = FALSE
    )
  )
)
```

`file_Input` *File Input Function*

Description

This function create a file upload component. It uses the basis of the shiny fileInput function, but restyles the label and adds error onto it.

Usage

```
file_Input(
  inputId,
  label,
  multiple = FALSE,
  accept = NULL,
  width = NULL,
  buttonLabel = "Choose file",
  placeholder = "No file chosen",
  error = FALSE,
  error_message = NULL
)
```

Arguments

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

Value

a file input HTML shiny tag object

See Also

Other Govstyle select inputs: button_Input(), checkbox_Input(), radio_button_Input(), select_Input()

Examples

```
ui <- shiny::fluidPage(
  # Required for error handling function
  shinyjs::useShinyjs(),
  shinyGovstyle::header(
    org_name = "Example",
    service_name = "User Examples",
    logo="shinyGovstyle/images/moj_logo.png"
  ),
  shinyGovstyle::banner(
    inputId = "banner", type = "beta", 'This is a new service'
  ),
  shinyGovstyle::gov_layout(size = "two-thirds",
    # Simple file input
    shinyGovstyle::file_Input(inputId = "file1", label = "Upload a file"),
    # Error file
    shinyGovstyle::file_Input(
      inputId = "file2",
      label = "Upload a file",
      error = TRUE
    ),
    # Button to trigger error
    shinyGovstyle::button_Input(inputId = "submit", label = "Submit")
  ),
  shinyGovstyle::footer(full = TRUE)
)

server <- function(input, output, session) {
  # Trigger error on blank submit of file2
  observeEvent(input$submit, {
    if (is.null(input$file2)){
      shinyGovstyle::error_on(inputId = "file2")
    } else {
      shinyGovstyle::error_off(
        inputId = "file2"
      )
    }
  })
}
if (interactive()) shinyApp(ui = ui, server = server)
```

---

font *Font Function*

Description

Loads the GDS Transport font for use in your app. GDS Transport is a restricted typeface that must only be used on GOV.UK domains. If your app is not hosted on a GOV.UK domain, do not call this function — the GOV.UK Frontend CSS will fall back to Helvetica or Arial automatically.

Usage

```
font()
```

Details

See the GOV.UK typeface guidance for full details on when GDS Transport is permitted.

Value

no value returned. This loads the font CSS file

See Also

Other Govstyle styling: `full_width_overrides()`

Examples

```
ui <- shiny::fluidPage(
  font(),
  shinyGovstyle::header(
    org_name = "Example",
    service_name = "User Examples",
    logo="shinyGovstyle/images/moj_logo.png")
)

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

if (interactive()) shinyApp(ui = ui, server = server)
```

---

footer *Footer Function*

Description

This function create a gov style footer for your page

Usage

```
footer(full = FALSE, links = NULL)
```

Arguments

full	Whenever you want to have blank footer or official gov version. Defaults to FALSE
links	A vector of actionLinks to be added to the footer, inputIDs are auto-generated and are the snake case version of the link text, e.g. "Accessibility Statement" will have an inputID of accessibility_statement

Details

You can add links in the footer to content either within the dashboard or content external to the dashboard using the links_list argument.

Links in the footer should be used sparingly and are usually for supporting information pages such as the accessibility statement, privacy notice, cookies information or link to a statement of voluntary adoption of the statistics code of practice.

If adding a link to an internal page, generally you will be controlling a hidden tabset so to the end user it looks like it is a new page.

Value

a footer HTML shiny tag object

See Also

Other Govstyle page structure: banner(), cookieBanner(), gov_layout(), header(), layouts, skip_to_main()

Examples

```
ui <- shiny::fluidPage(
  shinyGovstyle::header(
    org_name = "Example",
    service_name = "User Examples",
    logo = "shinyGovstyle/images/moj_logo.png",
    logo_alt_text = "Ministry of Justice logo"
  ),
  shinyGovstyle::banner(
    inputId = "banner", type = "beta", "This is a new service"
  ),
  shiny::tags$br(),
  shiny::tags$br(),
  shinyGovstyle::footer(full = TRUE)
)

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

if (interactive()) shinyApp(ui = ui, server = server)

# Add links
footer(links = c("Accessibility statement", "Cookies"))
```

```
# Add links, internal and external
footer(
  links = c(
    "Accessibility statement",
    "Cookies",
    `Government Digital Service` =
    "https://www.gov.uk/government/organisations/government-digital-service"
  )
)

# Full app with link controlling a hidden tab and a link to an external page
ui <- shiny::fluidPage(
  shinyGovstyle::header(
    org_name = "Example",
    service_name = "User Examples",
    logo = "shinyGovstyle/images/moj_logo.png",
    logo_alt_text = "Ministry of Justice logo"
  ),
  shinyGovstyle::banner(
    inputId = "banner", type = "beta", "This is a new service"
  ),
  shiny::tabsetPanel(
    type = "hidden",
    id = "tabs",
    shiny::tabPanel(
      "Main content",
      value = "main",
      heading_text("Hello world!")
    ),
    shiny::tabPanel(
      "Accessibility statement",
      value = "accessibility-panel",
      heading_text("Accessibility statement")
    ),
    shiny::tabPanel(
      "Cookies",
      value = "cookies-panel",
      heading_text("Cookies")
    ),
  ),
  shinyGovstyle::footer(
    full = TRUE,
    links = c(
      `Accessibility statement` = "accessibility_footer_link",
      `Cookies` = "cookies_footer_link",
      `Government Digital Service` =
       paste0(
         "https://www.gov.uk/government/",
         "organisations/government-digital-service"
       )
    )
  )
)
```

```
server <- function(input, output, session) {
  shiny::observeEvent(input$accessibility_footer_link, {
    shiny::updateTabsetPanel(
      session,
      "tabs",
      selected = "accessibility-panel"
    )
  })
  shiny::observeEvent(input$cookies_footer_link, {
    shiny::updateTabsetPanel(
      session,
      "tabs",
      selected = "cookies-panel"
    )
  })
}

if (interactive()) shinyApp(ui = ui, server = server)
```

`full_width_overrides` *Styling overrides for to give full width*

Description

[Experimental] This is an experimental function that will likely be removed in future releases, as we work through updating the core package styling and adding full width options into the components themselves. This is not well tested and may cause unexpected styling issues when combined with components from other packages, use at your own risk.

Usage

```
full_width_overrides()
```

Value

HTML containing CSS styling overrides

See Also

Other Govstyle styling: `font()`

Examples

```
shinyGovstyle::full_width_overrides()
```

govReactable *Interactive govTable*

Description

This function is opinionated and sets table defaults that are in keeping with the wider GOV.UK design system. Some defaults are overrideable, such as `highlight=TRUE` and `borderless=TRUE`, however some are fixed, such as `showSortIcon=FALSE` as the default sort icon is inaccessible. Additional arguments from `reactable::reactable` can be passed to customise the table.

Usage

```
govReactable(
  df,
  right_col = NULL,
  page_size = 10,
  highlight = TRUE,
  borderless = TRUE,
  min_widths = list(),
  ...
)
```

Arguments

`df`	A dataframe used to generate the table
`right_col`	A vector of column names that should be right-aligned. By default, numeric data is right-aligned, and character data is left-aligned
`page_size`	The default number of rows displayed per page (default: 10)
`highlight`	Highlight table rows on hover
`borderless`	Remove inner borders from table
`min_widths`	Customise minimum column width using a list of columns and minimum width in pixels
`...`	Additional arguments passed to `reactable::reactable`

Details

This function inserts a government-styled table using `reactable`. You can use this in R markdown or Quarto documents, or use renderGovReactable() and govReactableOutput() for tables in R Shiny. govReactableOutput() gives the ability to add a caption, for static tables made using just govReactable(), use heading_text() to add captions to tables.

Value

A `reactable` HTML widget styled with GOV.UK classes

See Also

Other Govstyle tables tabs and accordions: accordion(), govReactable-shiny, govTable(), govTabs()

Examples

```
# Example static table using govReactable
if (interactive()) {
  govReactable(
    iris,
    right_col = c(
      "Sepal.Length", "Sepal.Width", "Petal.Length", "Petal.Width"
    )
  )

  govReactable(
    iris,
    right_col = c(
      "Sepal.Length", "Sepal.Width", "Petal.Length", "Petal.Width"
    ),
    highlight = FALSE,
    page_size = 5,
    min_widths = list(
      Sepal.Length = 75,
      Sepal.Width = 75,
      Petal.Length = 75,
      Petal.Width = 75
    )
  )
}
```

---

govReactable-shiny *Shiny bindings for govReactable Output and render functions for using govReactable within shiny apps*

Description

Shiny bindings for govReactable Output and render functions for using govReactable within shiny apps

Usage

```
govReactableOutput(
  output_table_name,
  caption,
  caption_size = "l",
  heading_level = "h2"
)

renderGovReactable(expr, env = parent.frame(), quoted = FALSE)
```

Arguments

output_table_name	Output variable to read from
caption	Adds a caption to the table as a header
caption_size	Adjust the size of caption Options are s, m, l, xl, with l as the default
heading_level	The HTML heading level for the caption (e.g., "h2", "h3", "h4", "h5"). Default is "h2"
expr	An expression that generates a reactable widget
env	The environment in which to evaluate expr
quoted	Is expr a quoted expression (with quote())? This is useful if you want to save an expression in a variable

Value

govReactableOutput() returns a reactable output element that can be included in a Shiny UI

renderGovReactable() returns a reactable render function that can be assigned to a Shiny output slot

See Also

Other Govstyle tables tabs and accordions: accordion(), govReactable(), govTable(), govTabs()

Examples

```
ui <- shiny::fluidPage(
  govReactableOutput(
    "table",
    caption = "Example table"
  )
)

server <- function(input, output, session) {
  output$table <- renderGovReactable({
    govReactable(iris)
  })
}

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

---

govTable *Table Function*

Description

This function inserts a gov styled table. Format is with header looking rows and columns

Usage

```
govTable(
  inputId,
  df,
  caption,
  caption_size = "l",
  num_col = NULL,
  width_overwrite = NULL
)
```

Arguments

inputId	Input Id for the table
df	expects a dataframe to create a table
caption	adds a caption to the table as a header
caption_size	adjust the size of caption. Options are s, m, l, xl, with l as the default
num_col	adds numeric class format to these columns
width_overwrite	change width. Need to include width for every column. Columns must add up to 1. Options are three-quarters, two-thirds, one-half, one-third, one-quarter. Default is NULL

Value

a table HTML shiny tag object

See Also

Other Govstyle tables tabs and accordions: accordion(), govReactable(), govReactable-shiny, govTabs()

Examples

```
ui <- shiny::fluidPage(
  shinyGovstyle::gov_layout(
    size = "two-thirds",
    shinyGovstyle::govTable(
      "tab1",
      shinyGovstyle::transport_data_small,
      "Test",
      "l",
      num_col = c(2,3),
      width_overwrite = c("one-half", "one-quarter", "one-quarter")
    )
  )
)

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

if (interactive()) shinyApp(ui = ui, server = server)
```

govTabs *Tabs Function*

Description

This function creates a tabs based table. It requires a single dataframe with a grouping variable.

Usage

```
govTabs(inputId, df, group_col)
```

Arguments

inputId	The Id to access the tag
df	A single dataframe with all data. See example for structure
group_col	The column name with the groups to be used as tabs

Value

a tab table HTML shiny tag object

See Also

Other Govstyle tables tabs and accordions: accordion(), govReactable(), govReactable-shiny, govTable()

Examples

```
ui <- shiny::fluidPage(
  shinyGovstyle::header(
    org_name = "Example",
    service_name = "User Examples",
    logo = "shinyGovstyle/images/moj_logo.png",
    logo_alt_text = "Ministry of Justice logo"
  ),
  shinyGovstyle::gov_main_layout(
    size = "two-thirds",
    shinyGovstyle::govTabs("tabs", shinyGovstyle::case_data, "tabs")
  ),
  shinyGovstyle::footer(full = TRUE)
)

server <- function(input, output, session) {}
if (interactive()) shinyApp(ui = ui, server = server)
```

gov_layout *Page Layout Function*

Description

This function loads the page layout, This doesn't work as well as the `gov_main_layout` and associated functions. This is being kept for now as a simpler version where grids are not needed.

Usage

```
gov_layout(..., inputID = "main", size = "full")
```

Arguments

`...`	include the components of the UI that you want within the main page.
`inputID`	ID of the main div. Defaults to "main"
`size`	Layout of the page. Optional are full, one-half, two-thirds, one-third and one-quarter. Defaults to "full"

Value

a HTML shiny layout div

See Also

Other Govstyle page structure: `banner()`, `cookieBanner()`, `footer()`, `header()`, `layouts`, `skip_to_main()`

Examples

```
ui <- shiny::fluidPage(
  shinyGovstyle::header(
    org_name = "Example",
    service_name = "User Examples",
    logo="shinyGovstyle/images/moj_logo.png"
  ),
  shinyGovstyle::gov_layout(
    size = "full",
    shinyGovstyle::panel_output(
      inputId = "panel1",
      main_text = "Application Complete",
      sub_text = paste(
        "Thank you for submitting your application.",
        "Your reference is xvsiq"
      )
    ),
    shinyGovstyle::footer(full = TRUE)
  )
)
```

```
server <- function(input, output, session) {}
if (interactive()) shinyApp(ui = ui, server = server)
```

---

gov_list *Gov List function*

Description

Gov List function

Usage

```
gov_list(list, style = "none")
```

Arguments

list	vector of list
style	options: "none", "bullet", "number". defaults to "none"

See Also

Other Govstyle text types: date_Input(), heading_text(), input_field(), text_Input(), text_area_Input(), word_count()

Examples

```
ui <- shiny::fluidPage(
  shinyGovstyle::header(
    org_name = "Example",
    service_name = "User Examples"
  ),
  shinyGovstyle::banner(
    inputId = "banner", type = "beta", 'This is a new service'
  ),
  shinyGovstyle::gov_layout(
    size = "two-thirds",
    shinyGovstyle::heading_text("gov_list", size = "s"),
    shinyGovstyle::gov_text("List:"),
    gov_list(list = c("a", "b", "c")),
    shinyGovstyle::gov_text("Bulleted list:"),
    gov_list(list = c("a", "b", "c"), style = "bullet"),
    shinyGovstyle::gov_text("Numbered list:"),
    gov_list(list = c("one", "two", "three"), style = "number")
  )
)

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

if (interactive()) shinyApp(ui = ui, server = server)
```

gov_summary *Tabs Function*

Description

This function creates a tabs based table. It requires a single dataframe with a grouping variable.

Usage

```
gov_summary(inputId, headers, info, action = FALSE, border = TRUE)
```

Arguments

inputId	The Id to access the summary list
headers	input for the row headers value
info	summary information values for the table
action	whenever a change link is needed. Sets input to the value of the headers using lowercase and with underscore to replace gaps. Default set to FALSE
border	set if the table should have borders. Default set to TRUE

Value

a summary list table HTML shiny tag object

See Also

Other Govstyle feedback types: details(), insert_text(), label_hint(), noti_banner(), panel_output(), tag_Input(), value_box(), warning_text()

Examples

```
# Create an example dataset
headers <- c(
  "Name",
  "Date of birth",
  "Contact information",
  "Contact details"
)
info <- c(
  "Sarah Philips",
  "5 January 1978",
  "72 Guild Street <br> London <br> SE23 6FH",
  "07700 900457 <br> sarah.phillips@example.com"
)

ui <- shiny::fluidPage(
  shinyGovstyle::header(
    org_name = "Example",
```

```
    service_name = "User Examples",
    logo="shinyGovstyle/images/moj_logo.png"
  ),
  shinyGovstyle::gov_layout(
    size = "two-thirds",
    shinyGovstyle::gov_summary("sumID", headers, info, action = FALSE)
  ),
  shinyGovstyle::footer(full = TRUE)
)

server <- function(input, output, session) {}
if (interactive()) shinyApp(ui = ui, server = server)
```

---

header *Header Function*

Description

This function create a header banner. For use at top of the screen

Usage

```
header(
  org_name = "Shiny example app",
  service_name = NULL,
  logo = "shinyGovstyle/images/Dept_logo.svg",
  main_text = lifecycle::deprecated(),
  secondary_text = lifecycle::deprecated(),
  main_link = NULL,
  secondary_link = NULL,
  logo_alt_text = "Departmental logo",
  main_alt_text = NULL,
  secondary_alt_text = NULL,
  logo_width = 66,
  logo_height = 34
)
```

Arguments

org_name	Organisation name that goes in the header
service_name	Service name to supplement the organisation name
logo	Add a link to a logo which will apply in the header. Use crown to use the crown SVG version on GOV UK
main_text	**[Deprecated]** Use org_name instead
secondary_text	**[Deprecated]** Use service_name instead
main_link	Add a link for clicking on main text **[Deprecated]**

secondary_link	Add a link for clicking on secondary header **[Deprecated]**
logo_alt_text	Add alternative text for the logo. Should be used when a logo is used
main_alt_text	Add alternative text for the main link. Should be used when a main link is used **[Deprecated]**
secondary_alt_text	Add alternative text for the secondary link. Should be used when a secondary link is used **[Deprecated]**
logo_width	Change the logo size width CSS to improve fit
logo_height	Change the logo size height CSS to improve fit

Value

a header HTML shiny tag object

See Also

Other Govstyle page structure: banner(), cookieBanner(), footer(), gov_layout(), layouts, skip_to_main()

Examples

```
ui <- shiny::fluidPage(
  shinyGovstyle::header(
    org_name = "Example",
    service_name = "User Examples",
    logo = "shinyGovstyle/images/moj_logo.png",
    logo_alt_text = "Ministry of Justice Logo"
  )
)

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

if (interactive()) shinyApp(ui = ui, server = server)
```

---

heading_text *Heading Text Function*

Description

This function createS heading text

Usage

```
heading_text(text_input, size = "xl", id, level = 1)
```

Arguments

text_input	Text to display
size	Text size using xl, l, m, s. Defaults to xl
id	Custom header id
level	Heading level, integer between 1 and 6. Defaults to 1

Value

a heading text HTML shiny tag object

See Also

Other Govstyle text types: date_Input(), gov_list(), input_field(), text_Input(), text_area_Input(), word_count()

Examples

```
shinyGovstyle::heading_text("This is great text")
shinyGovstyle::heading_text("This is great text", size = "l", level = 2)
```

---

input_field *Input Field Function*

Description

This function inserts number of text inputs. Useful for addresses.

Usage

```
input_field(
  legend,
  labels,
  inputIds,
  widths = NULL,
  types = "text",
  error = FALSE,
  error_message = NULL
)
```

Arguments

legend	Legend that goes above the fieldset
labels	A list of labels for the text inputs
inputIds	A list input slots that will be used to access the value
widths	control the size of the box based on number of characters required. Options are 30, 20, 10, 5, 4, 3, 2. NULL will not limit the size

types	text box types. Will default to text
error	Whenever to icnlud error handling. Defaults to FALSE
error_message	Message to display on error. Defaults to NULL

Value

a input field of HTML as a shiny tag object

See Also

Other Govstyle text types: date_Input(), gov_list(), heading_text(), text_Input(), text_area_Input(), word_count()

Examples

```
ui <- shiny::fluidPage(
  # Required for error handling function
  shinyjs::useShinyjs(),
  shinyGovstyle::header(
    org_name = "Example",
    service_name = "User Examples",
    logo="shinyGovstyle/images/moj_logo.png"
  ),
  shinyGovstyle::banner(
    inputId = "banner", type = "beta", 'This is a new service'
  ),
  shinyGovstyle::gov_layout(
    size = "two-thirds",
    shinyGovstyle::input_field(
      legend ="List of three text boxes in a field",
      labels = c("Field 1", "Field 2", "Field 3"),
      inputIds = c("field1", "field2", "field3"),
      widths = c(30,20,10),
      error = TRUE
    ),
   # Button to trigger error
   shinyGovstyle::button_Input(inputId = "submit", label = "Submit")
  ),
  shinyGovstyle::footer(full = TRUE)
)

server <- function(input, output, session) {
  # Trigger error on blank submit of field2
  observeEvent(input$submit, {
    if (input$field2 == ""){
      shinyGovstyle::error_on(
        inputId = "field2",
        error_message = "Please complete"
      )
    } else {
      shinyGovstyle::error_off(
        inputId = "field2"
```

```
      )
    }
  })
}
if (interactive()) shinyApp(ui = ui, server = server)
```

`insert_text` *Insert Text Function*

Description

This function loads the insert text component to display additional information in a special format.

Usage

```
insert_text(inputId, text)
```

Arguments

`inputId`	The input slot that will be used to access the value
`text`	Text that you want to display on the insert

Value

a insert text HTML shiny tag object

See Also

Other Govstyle feedback types: `details()`, `gov_summary()`, `label_hint()`, `noti_banner()`, `panel_output()`, `tag_Input()`, `value_box()`, `warning_text()`

Examples

```
ui <- shiny::fluidPage(
  shinyGovstyle::header(
    org_name = "Example",
    service_name = "User Examples",
    logo="shinyGovstyle/images/moj_logo.png"
  ),
  shinyGovstyle::gov_layout(
    size = "two-thirds",
    shinyGovstyle::insert_text(
      inputId = "note",
      text = paste(
        "It can take up to 8 weeks to register a lasting power of",
        "attorney if there are no mistakes in the application."
      )
    )
  ),
```

```
  shinyGovstyle::footer(full = TRUE)
)

server <- function(input, output, session) {}
if (interactive()) shinyApp(ui = ui, server = server)
```

`label_hint` *Label with Hint Function*

Description

This function inserts a label and optional hint.

Usage

```
label_hint(inputId, label, hint_input = NULL)
```

Arguments

`inputId`	The input slot that will be used to access the value
`label`	Display label for the control, or `NULL` for no label
`hint_input`	Display hint label for the control, or `NULL` for no hint label

Value

a label hint HTML shiny tag object

See Also

Other Govstyle feedback types: `details()`, `gov_summary()`, `insert_text()`, `noti_banner()`, `panel_output()`, `tag_Input()`, `value_box()`, `warning_text()`

Examples

```
ui <- shiny::fluidPage(
  shinyGovstyle::header(
    org_name = "Example",
    service_name = "User Examples",
    logo="shinyGovstyle/images/moj_logo.png"
  ),
  shinyGovstyle::gov_layout(
    size = "two-thirds",
    label_hint(
      inputId = "label1",
      label = "This is a label",
      hint_input = "This is a hint"
    )
  ),
  shinyGovstyle::footer(full = TRUE)
```

```
)

server <- function(input, output, session) {}
if (interactive()) shinyApp(ui = ui, server = server)
```

`layouts` *Page Layout Functions*

Description

These function loads the page layout in a gov layout. There is a selection of components that can sit within each other. The gov_main_layout is the overarching layout. The gov_row creates a each row and gov_box creates a box within the row. The gov_text is a container for text bodies.

Usage

```
gov_main_layout(..., inputID = "main")

gov_row(...)

gov_box(..., size = "full")

gov_text(...)
```

Arguments

`...`	include the components of the UI that you want within the main page. These components are made to flow through each other. See example
`inputID`	ID of the main div. Defaults to "main"
`size`	size of the box in the row. Optional are full, one-half, two-thirds, one-third and one-quarter. Defaults to "full"

Value

a HTML shiny layout div

See Also

Other Govstyle page structure: `banner()`, `cookieBanner()`, `footer()`, `gov_layout()`, `header()`, `skip_to_main()`

Examples

```
ui <- shiny::fluidPage(
  shinyGovstyle::header(
    org_name = "Example",
    service_name = "User Examples",
    logo="shinyGovstyle/images/moj_logo.png"),
  shinyGovstyle::gov_main_layout(
    shinyGovstyle::gov_row(
      shinyGovstyle::gov_box(
        size = "full",
        shinyGovstyle::gov_text("govuk-grid-column-full")
      )
    ),
    shinyGovstyle::gov_row(
      shinyGovstyle::gov_box(
        size = "one-half",
        shinyGovstyle::gov_text("govuk-grid-column-one-half")
      ),
      shinyGovstyle::gov_box(
        size = "one-half",
        shinyGovstyle::gov_text("govuk-grid-column-one-half")
      )
    ),
    shinyGovstyle::gov_row(
      shinyGovstyle::gov_box(
        size = "one-third",
        shinyGovstyle::gov_text("govuk-grid-column-one-third")
      ),
      shinyGovstyle::gov_box(
        size = "two-third",
        shinyGovstyle::gov_text("govuk-grid-column-two-third")
      )
    ),
    shinyGovstyle::gov_row(
      shinyGovstyle::gov_box(
        size = "one-quarter",
        shinyGovstyle::gov_text("govuk-grid-column-one-quarter")
      ),
      shinyGovstyle::gov_box(
        size = "three-quarters",
        shinyGovstyle::gov_text("govuk-grid-column-three-quarters")
      )
    )
  ),
  shinyGovstyle::footer(full = TRUE)
)

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

if (interactive()) shinyApp(ui = ui, server = server)
```

noti_banner *Notification Banner Function*

Description

This function creates a notification banner.

Usage

```
noti_banner(
  inputId,
  title_txt = "Important",
  body_txt = NULL,
  type = "standard"
)
```

Arguments

inputId	The input Id for the banner
title_txt	The wording that appears in the title
body_txt	The wording that appears in the banner body
type	The type of banner. Options are standard and success. Standard is default

Value

a notification HTML shiny tag object

See Also

Other Govstyle feedback types: details(), gov_summary(), insert_text(), label_hint(), panel_output(), tag_Input(), value_box(), warning_text()

Examples

```
ui <- shiny::fluidPage(
  shinyGovstyle::header(
    org_name = "Example",
    service_name = "User Examples",
    logo="shinyGovstyle/images/moj_logo.png"
  ),
  shinyGovstyle::noti_banner(
    inputId = "banner", title_txt = "Important", body_txt = "Example text"
  )
)

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

if (interactive()) shinyApp(ui = ui, server = server)
```

panel_output *Panel output*

Description

This function inserts a panel. Normally used for confirmation screens

Usage

```
panel_output(inputId, main_text, sub_text)
```

Arguments

inputId	The input slot that will be used to access the value.
main_text	Add the header for the panel
sub_text	Add the main body of text for the panel

Value

a panel HTML shiny tag object

See Also

Other Govstyle feedback types: details(), gov_summary(), insert_text(), label_hint(), noti_banner(), tag_Input(), value_box(), warning_text()

Examples

```
ui <- shiny::fluidPage(
  shinyGovstyle::header(
    org_name = "Example",
    service_name = "User Examples",
    logo="shinyGovstyle/images/moj_logo.png"
  ),
  shinyGovstyle::gov_layout(size = "full",
    shinyGovstyle::panel_output(
      inputId = "panel1",
      main_text = "Application Complete",
      sub_text = paste(
        "Thank you for submitting your application.",
        "Your reference is xvsiq"
      )
    ),
    shinyGovstyle::footer(full = TRUE)
  )
)

server <- function(input, output, session) {}
if (interactive()) shinyApp(ui = ui, server = server)
```

`radio_button_Input` *Radio Button Function*

Description

This function create radio buttons

Usage

```
radio_button_Input(
  inputId,
  label,
  choices = NULL,
  selected = NULL,
  inline = FALSE,
  small = FALSE,
  choiceNames = NULL,
  choiceValues = NULL,
  hint_label = NULL,
  error = FALSE,
  error_message = NULL,
  custom_class = ""
)
```

Arguments

`inputId`	The `input` slot that will be used to access the value
`label`	Input label
`choices`	List of values to select from (if elements of the list are named then that name rather than the value is displayed to the user)
`selected`	The initially selected value.
`inline`	If you want the radio inline or not, Default is FALSE
`small`	If you want the smaller versions of radio buttons, Default is FALSE
`choiceNames, choiceValues`	Same as in `shiny::checkboxGroupInput()`. List of names and values, respectively, that are displayed to the user in the app and correspond to the each choice (for this reason they must have the same length). If either of these arguments is provided, then the other must be provided and choices must not be provided. The advantage of using both of these over a named list for choices is that choiceNames allows any type of UI object to be passed through (tag objects, icons, HTML code, ...), instead of just simple text
`hint_label`	Additional hint text you may want to display below the label. Defaults to NULL
`error`	Whenever you want to include error handle on the component
`error_message`	If you want a default error message
`custom_class`	If you want to add additional classes to the radio buttons

Value

radio buttons HTML shiny tag object

See Also

Other Govstyle select inputs: button_Input(), checkbox_Input(), file_Input(), select_Input()

Examples

```
ui <- shiny::fluidPage(
  # Required for error handling function
  shinyjs::useShinyjs(),
  shinyGovstyle::header(
    org_name = "Example",
    service_name = "User Examples",
    logo = "shinyGovstyle/images/moj_logo.png"
  ),
  shinyGovstyle::banner(
    inputId = "banner", type = "beta", "This is a new service"
  ),
  shinyGovstyle::gov_layout(
    size = "two-thirds",
    # Simple radio
    shinyGovstyle::radio_button_Input(
      inputId = "radio1",
      choices = c("Yes", "No", "Maybe"),
      label = "Choice option"
    ),
    # Error radio
    shinyGovstyle::radio_button_Input(
      inputId = "radio2",
      choices = c("Yes", "No", "Maybe"),
      label = "Choice option",
      hint_label = "Select the best fit",
      inline = TRUE,
      error = TRUE,
      error_message = "Select one"
    ),
    # Button to trigger error
    shinyGovstyle::button_Input(inputId = "submit", label = "Submit")
  ),
  shinyGovstyle::footer(full = TRUE)
)

server <- function(input, output, session) {
  # Trigger error on blank submit of eventId2
  observeEvent(input$submit, {
    if (is.null(input$radio2)) {
      shinyGovstyle::error_on(inputId = "radio2")
    } else {
      shinyGovstyle::error_off(
        inputId = "radio2"
```

```
      )
    }
  })
}
if (interactive()) shinyApp(ui = ui, server = server)
```

`run_example` *Run examples*

Description

This function runs an example R Shiny app showcasing different parts of the package. Code for the app can be found in the inst/example_app folder in the source code.

Usage

```
run_example()
```

Details

The app uses the bslib package as is generally recommended for Shiny apps.

The app is also deployed at the following URL: https://department-for-education.shinyapps.io/shinygovstyle-example-app/

Value

runs an R Shiny app with examples in

Examples

```
if (interactive()) run_example()
```

`select_Input` *Select Function*

Description

This function inserts a select box

Usage

```
select_Input(inputId, label, select_text, select_value)
```

Arguments

inputId	Input Id for the component
label	Insert the text for the label
select_text	Add the text that will apply in the drop down as a list
select_value	Add the value that will be used for each selection

Value

a select input HTML shiny tag object

See Also

Other Govstyle select inputs: button_Input(), checkbox_Input(), file_Input(), radio_button_Input()

Examples

```
ui <- shiny::fluidPage(
  shinyGovstyle::header(
    org_name = "Example",
    service_name = "User Examples",
    logo = "shinyGovstyle/images/moj_logo.png",
    logo_alt_text = "Ministry of Justice logo"
  ),
  shinyGovstyle::gov_layout(
    size = "full",
    select_Input(
      inputId = "sorter",
      label = "Sort by",
      select_text = c(
        "Recently published",
        "Recently updated",
        "Most views",
        "Most comments"
      ),
      select_value = c("published", "updated", "view", "comments")
    ),
    shiny::tags$br()
  ),
  shinyGovstyle::footer(full = TRUE)
)

server <- function(input, output, session) {}
if (interactive()) shinyApp(ui = ui, server = server)
```

`service_navigation` *Service navigation*

Description

Service navigation component consistent with the GDS service navigation. # nolint

Note: This component uses a hardcoded element ID for the navigation list and mobile menu toggle. Using multiple `service_navigation()` instances on the same page will cause ID conflicts and the mobile menu toggle may not work correctly.

Usage

```
service_navigation(links, service_name = NULL)
```

Arguments

`links`	A vector of actionLinks to be added to the service navigation. inputIDs are auto-generated by lowercasing the link text and replacing all non-alphanumeric characters with underscores, e.g. "Overview page" becomes `overview_page`. For precise control over inputIDs, supply a named vector where names are the page titles and values are the inputIDs.
`service_name`	An optional character string containing the service name to be displayed in the navigation bar

Value

Shiny tag object

See Also

Other Govstyle navigation: `backlink_Input()`, `contents_link()`, `update_service_navigation()`

Examples

```
ui <- shiny::fluidPage(
  shinyGovstyle::header("Title", "Secondary heading"),
  shinyGovstyle::service_navigation(
     c("Summary data", "Detailed stats 1", "User guide")
  ),
  bslib::navset_hidden(
     id = "main_panels",
     bslib::nav_panel(
       "summary_data",
       shiny::tags$h2("Summary data")
     ),
     bslib::nav_panel(
       "detailed_stats_1",
       shiny::tags$h2("Detailed stats 1")
```

```
      ),
      bslib::nav_panel(
        "user_guide",
        shiny::tags$h2("User guide")
      )
    ),
    shinyGovstyle::footer(full = TRUE)
)

server <- function(input, output, session) {
  observeEvent(
    input$summary_data,
    bslib::nav_select("main_panels", "summary_data")
  )
  observeEvent(
    input$detailed_stats_1,
    bslib::nav_select("main_panels", "detailed_stats_1")
  )
  observeEvent(
    input$user_guide,
    bslib::nav_select("main_panels", "user_guide")
  )
}

if (interactive()) shiny::shinyApp(ui = ui, server = server)
```

`skip_to_main` *Skip to main content link*

Description

This function generates a 'Skip to main content' link, which is typically used by keyboard users to bypass content and navigate directly to the main content of a page.

Usage

```
skip_to_main(id = "main")
```

Arguments

`id`	An optional parameter to specify the Id of the main content section, will be automatically preceeded by a hash '#'. Default is "main" to match the "#main" Id within `gov_main()`.

Value

A Shiny tag representing the 'Skip to main content' link

See Also

Other Govstyle page structure: banner(), cookieBanner(), footer(), gov_layout(), header(), layouts

Examples

```
ui <- shiny::fluidPage(
  skip_to_main(),
  header(
    org_name = "Example",
    service_name = "User Examples",
    logo="shinyGovstyle/images/moj_logo.png"
  ),
  gov_main_layout(
    heading_text("Example heading"),
  )
)

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

if (interactive()) shinyApp(ui = ui, server = server)
```

---

tag_Input *Tag Function*

Description

This function creates a tag.

Usage

```
tag_Input(inputId, text, colour = "navy")
```

Arguments

inputId	The Id to access the tag
text	The text in the tag
colour	The colour of the tag. Default is navy. Other options are grey, green, teal, blue, purple, magenta, red, orange and yellow

Value

a tag HTML shiny tag object

See Also

Other Govstyle feedback types: details(), gov_summary(), insert_text(), label_hint(), noti_banner(), panel_output(), value_box(), warning_text()

Examples

```
ui <- shiny::fluidPage(
  shinyGovstyle::header(
    org_name = "Example",
    service_name = "User Examples",
    logo="shinyGovstyle/images/moj_logo.png"
  ),
  shinyGovstyle::gov_layout(
    size = "two-thirds",
    shinyGovstyle::tag_Input("tag1", "Complete"),
    shinyGovstyle::tag_Input("tag2", "Incomplete", "red")
  ),
  shinyGovstyle::footer(full = TRUE)
)

server <- function(input, output, session) {}
if (interactive()) shinyApp(ui = ui, server = server)
```

---

text_area_Input *Text Area Input Function*

Description

This function create a text area input.

Usage

```
text_area_Input(
  inputId,
  label,
  hint_label = NULL,
  row_no = 5,
  error = FALSE,
  error_message = NULL,
  word_limit = NULL
)
```

Arguments

inputId	The input slot that will be used to access the value
label	Display label for the control, or NULL for no label
hint_label	Display hint label for the control, or NULL for no hint label
row_no	Size of the text entry box. Defaults to 5
error	Whenever to include error handling. Defaults to FALSE
error_message	Message to display on error. Defaults to NULL
word_limit	Add a word limit to the display. Defaults to NULL

Value

a text area box HTML shiny tag object

See Also

Other Govstyle text types: date_Input(), gov_list(), heading_text(), input_field(), text_Input(), word_count()

Examples

```
text_area_Input(
  "taId",
  "Can you provide more detail?",
  paste(
    "Do not include personal or financial information, like your",
    "National Insurance number or credit card details."
  )
)
```

---

text_Input *Text Input Function*

Description

This function create a text input.

Usage

```
text_Input(
  inputId,
  label,
  hint_label = NULL,
  type = "text",
  width = NULL,
  error = FALSE,
  error_message = NULL,
  prefix = NULL,
  suffix = NULL
)
```

Arguments

inputId	The input slot that will be used to access the value
label	Display label for the control, or NULL for no label
hint_label	Display hint label for the control, or NULL for no hint label
type	Type of text input to accept. Defaults to text

width	control the size of the box based on number of characters required. Options are 30, 20, 10, 5, 4, 3, 2. NULL will not limit the size
error	Whenever to include error handling. Defaults to FALSE
error_message	Message to display on error. Defaults to NULL
prefix	Add a prefix to the box. Defaults to NULL
suffix	Add a suffix to the box. Defaults to NULL

Value

a text input HTML shiny tag object

See Also

Other Govstyle text types: date_Input(), gov_list(), heading_text(), input_field(), text_area_Input(), word_count()

Examples

```
ui <- shiny::fluidPage(
  # Required for error handling function
  shinyjs::useShinyjs(),
  shinyGovstyle::header(
    org_name = "Example",
    service_name = "User Examples",
    logo = "shinyGovstyle/images/moj_logo.png"
  ),
  shinyGovstyle::banner(
    inputId = "banner", type = "beta", 'This is a new service'
  ),
  shinyGovstyle::gov_layout(
    size = "two-thirds",
    # Simple text box
    shinyGovstyle::text_Input(inputId = "eventId", label = "Event Name"),
    # Error text box
    shinyGovstyle::text_Input(
      inputId = "eventId2",
      label = "Event Name",
      hint_label = "This can be found on the letter",
      error = TRUE
    ),
    # Button to trigger error
    shinyGovstyle::button_Input(inputId = "submit", label = "Submit")
  ),
  shinyGovstyle::footer(full = TRUE)
)

server <- function(input, output, session) {
  # Trigger error on blank submit of eventId2
  observeEvent(input$submit, {
    if (input$eventId2 != "") {
      shinyGovstyle::error_off(inputId = "eventId2")
```

```
    } else {
      shinyGovstyle::error_on(
        inputId = "eventId2",
        error_message = "Please complete"
      )
    }
  })
}

if (interactive()) shinyApp(ui = ui, server = server)
```

`transport_data` *Transport data*

Description

Simple example dataset containing transport data, used to demonstrate various table styles in the example app.

Usage

```
transport_data
```

Format

A data frame with 10 rows and 5 variables:

months The month the data belongs to

colours The colour of the vehicle

bikes Cost of bikes

vans Cost of vans

buses Cost of buses

Source

Random data generated for the shinyGovstyle package

transport_data_small *Transport data small*

Description

Smaller example dataset containing transport data, used to demonstrate various table styles in the example app.

Usage

```
transport_data_small
```

Format

A data frame with 3 rows and 3 variables:

months The month the data belongs to

bikes Cost of bikes

cars Cost of cars

Source

Random data generated for the shinyGovstyle package

update_service_navigation *Update the active item in a service navigation component*

Description

Sends a message to the browser to update the highlighted active item in the service navigation bar.

When you need this function: when navigation is triggered programmatically — for example, via a next / back button or a footer link that points to a main page. In those cases the nav link itself is not clicked, so the JavaScript binding does not fire and the active state does not update automatically. Call `update_service_navigation()` alongside your tab-switching call to keep them in sync.

When you don't need this function: when the user clicks a service navigation link directly. The JavaScript binding updates the active state automatically, so you only need to switch the tab panel in your `observeEvent()`.

Usage

```
update_service_navigation(session, inputId)
```

Arguments

session	The Shiny session object
inputId	The inputId of the service navigation link to set as active

Value

NULL, called for side effects

See Also

Other Govstyle navigation: backlink_Input(), contents_link(), service_navigation()

Examples

```
# Nav link clicked — JS handles active state, just switch the panel.
# Works the same whether you use shiny or bslib tab panels.
if (interactive()) {
  server <- function(input, output, session) {
    # shiny tabsetPanel
    shiny::observeEvent(input$page_two, {
      shiny::updateTabsetPanel(session, "tabs", selected = "page_two")
    })

    # bslib navset
    shiny::observeEvent(input$page_two, {
      bslib::nav_select("tabs", "page_two")
    })
  }
}

# Programmatic navigation (e.g. a next / back button) — the nav link is not
# clicked, so you must also call update_service_navigation() explicitly.
if (interactive()) {
  server <- function(input, output, session) {
    # shiny tabsetPanel
    shiny::observeEvent(input$next_btn, {
      shiny::updateTabsetPanel(session, "tabs", selected = "page_two")
      shinyGovstyle::update_service_navigation(session, "page_two")
    })

    # bslib navset
    shiny::observeEvent(input$next_btn, {
      bslib::nav_select("tabs", "page_two")
      shinyGovstyle::update_service_navigation(session, "page_two")
    })
  }
}
```

value_box *Create a value text box with optional description and colour*

Description

This function generates a value text box with an optional description and customisable colour.

Usage

```
value_box(value = "your value goes here", text = NA, colour = "blue")
```

Arguments

value	Character. The primary value to display in the value box. Defaults to "your value goes here"
text	Character or NA. An optional description to appear below the value. If not provided (default is NA), the description will not be displayed
colour	Character. A colour to apply to the value box. Defaults to "blue". Choose from the following: "grey", "purple", "teal", "blue", "yellow", "orange", "red", "magenta", or "green"

Details

The text box can be used in Shiny applications to display highlighted information, such as statistics or key metrics.

Value

A Shiny `div` tag representing the value box, styled according to the specified parameters

See Also

Other Govstyle feedback types: `details()`, `gov_summary()`, `insert_text()`, `label_hint()`, `noti_banner()`, `panel_output()`, `tag_Input()`, `warning_text()`

Examples

```
value_box(
  value = "1,000,000",
  text = "This is the latest value for the selected inputs.",
  colour = "purple"
)
```

warning_text *Warning Text Function*

Description

This function create warning text.

Usage

```
warning_text(inputId, text)
```

Arguments

inputId	The input slot that will be used to access the value
text	Text that goes in the main

Value

a warning box HTML shiny tag object

See Also

Other Govstyle feedback types: details(), gov_summary(), insert_text(), label_hint(), noti_banner(), panel_output(), tag_Input(), value_box()

Examples

```
shinyGovstyle::warning_text(
  inputId = "warn1",
  text = "You can be fined up to £5,000 if you do not register."
)
```

word_count *Word Count Function*

Description

This function create tracks the word count and should be used with the text area function.

Usage

```
word_count(inputId, input, word_limit = NULL)
```

Arguments

inputId	The input slot of the text area that you want to affect
input	The text input that is associated with the box
word_limit	Change the word limit if needed. Default will keep as what was used in text area component

Value

no value returned. Updates the word count in a shiny app

See Also

Other Govstyle text types: date_Input(), gov_list(), heading_text(), input_field(), text_Input(), text_area_Input()

Examples

```
ui <- shiny::fluidPage(
  shinyjs::useShinyjs(),
  shinyGovstyle::header(
    "Justice", "", logo = "shinyGovstyle/images/moj_logo.png"
  ),
  gov_layout(
    size = "full",
    text_area_Input(
      inputId = "text_area",
      label = "Can you provide more detail?",
      hint_label = paste(
        "Do not include personal or financial information,",
        "like your National Insurance number or credit card details."
      ),
      word_limit = 300
    )
  ),
  footer(TRUE)
)

server <- function(input, output, session) {
  shiny::observeEvent(input$text_area,
    word_count(
      inputId = "text_area",
      input = input$text_area
    )
  )
}
if (interactive()) shinyApp(ui = ui, server = server)
```

Index