

# Package ‘shinyGovstyle’

February 22, 2022

**Title** Custom Gov Style Inputs for Shiny

**Version** 0.0.8

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

**Depends** R (>= 3.1.0)

**License** GPL-3

**Encoding** UTF-8

**RoxygenNote** 7.1.2

**URL** <https://github.com/moj-analytical-services/shinyGovstyle>

**BugReports** <https://github.com/moj-analytical-services/shinyGovstyle/issues>

**Imports** shiny (>= 0.14), htmltools, shinyjs, jsonlite

**Suggests** testthat

**NeedsCompilation** no

**Author** Ross Wyatt [aut, cre]

**Maintainer** Ross Wyatt <[ross.wyatt@justice.gov.uk](mailto:ross.wyatt@justice.gov.uk)>

**Repository** CRAN

**Date/Publication** 2022-02-22 11:00:02 UTC

## R topics documented:

accordion . . . . .	2
backlink_Input . . . . .	3
banner . . . . .	4
button_Input . . . . .	5
checkboxbox_Input . . . . .	6
cookieBanner . . . . .	8
date_Input . . . . .	9
details . . . . .	11
error_off . . . . .	12

error_on . . . . .	13
error_summary . . . . .	14
error_summary_update . . . . .	15
file_Input . . . . .	16
font . . . . .	18
footer . . . . .	19
govTable . . . . .	20
govTabs . . . . .	21
gov_layout . . . . .	22
gov_summary . . . . .	23
header . . . . .	24
heading_text . . . . .	25
input_field . . . . .	26
insert_text . . . . .	27
label_hint . . . . .	28
layouts . . . . .	29
noti_banner . . . . .	31
panel_output . . . . .	32
radio_button_Input . . . . .	33
run_example . . . . .	35
select_Input . . . . .	35
tag_Input . . . . .	36
text_area_Input . . . . .	37
text_Input . . . . .	38
warning_text . . . . .	39
word_count . . . . .	40

**Index** **42**

---

accordion	<i>Accordion Function</i>
-----------	---------------------------

---

**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 object

**Examples**

```
if (interactive()) {  
  
  ui <- fluidPage(  
    shinyGovstyle::header(  
      main_text = "Example",  
      secondary_text = "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",  
    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) {}  
  
  shinyApp(ui = ui, server = server)  
}
```

---

backlink\_Input

*Back Link Function*

---

**Description**

This function adds a back link to the page

**Usage**

```
backlink_Input(inputId)
```

**Arguments**

inputId            The input slot that will be used to access the value.

**Value**

a backlink html shiny object

**Examples**

```

if (interactive()) {
  ui <- fluidPage(
    header(
      main_text = "Example",
      secondary_text = "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")
    })
  }
  shinyApp(ui = ui, server = server)
}

```

---

 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	test to display.

**Value**

a banner html shiny object

**Examples**

```
if (interactive()) {

  ui <- fluidPage(
    shinyGovstyle::header(
      main_text = "Example",
      secondary_text = "User Examples",
      logo="shinyGovstyle/images/moj_logo.png"),
    shinyGovstyle::banner(
      inputId = "banner", type = "beta", 'This is a new service')
  )

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

  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 object

**Examples**

```

if (interactive()) {
  ui <- fluidPage(
    shinyGovstyle::header(
      main_text = "Example",
      secondary_text = "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) {}
  shinyApp(ui = ui, server = server)
}

```

---

`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**

<code>inputId</code>	Input id for the group of checkboxes
<code>cb_labels</code>	Add the names of the options that will appear
<code>checkboxIds</code>	Add the values for each checkbox
<code>label</code>	Insert the text for the checkbox group.
<code>hint_label</code>	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 object

### Examples

```

if (interactive()) {

  ui <- fluidPage(
    # Required for error handling function
    shinyjs::useShinyjs(),
    shinyGovstyle::header(
      main_text = "Example",
      secondary_text = "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")
      }
    })
  })
}

```

```

    }
  shinyApp(ui = ui, server = server)
}

```

---

 cookieBanner

*Cookie Banner Function*


---

### Description

This function creates a cookie banner. You need to have shinyjs::useShinyjs() enabled to work. All the ids are pre set. 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 object.

### Examples

```

if (interactive()) {
ui <- fluidPage(
  shinyGovstyle::header(
    main_text = "Example",
    secondary_text = "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) {

  #Need these 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 you
    #updateTabsetPanel to have a cookie page for instance
  })

}
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 object

**Examples**

```

if (interactive()) {

  ui <- fluidPage(
    # Required for error handling function.
    shinyjs::useShinyjs(),
    shinyGovstyle::header(
      main_text = "Example",
      secondary_text = "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")
      }
    })
  }
  shinyApp(ui = ui, server = server)
}

```

---

details	<i>Details Function</i>
---------	-------------------------

---

**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 object

**Examples**

```
if (interactive()) {
  ui <- fluidPage(
    shinyGovstyle::header(
      main_text = "Example",
      secondary_text = "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) {}
shinyApp(ui = ui, server = 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 inputId to turn error handling iff for on for.

**Value**

no return value. This toggles off error css

**Examples**

```
## Only run examples in interactive R sessions
if (interactive()) {

  ui <- fluidPage(
    # Required for error handling function
    shinyjs::useShinyjs(),
    shinyGovstyle::header(
      main_text = "Example",
      secondary_text = "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")
  }
})
}

# Run the application
shinyApp(ui = ui, server = server)
}

```

---

error\_on

*Error on Function*


---

## Description

This function turns on the the error o 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

## Examples

```

## Only run examples in interactive R sessions
if (interactive()) {

  ui <- fluidPage(
    # Required for error handling function
    shinyjs::useShinyjs(),
    shinyGovstyle::header(
      main_text = "Example",
      secondary_text = "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")
    }
  })
}

# Run the application
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 object

**Examples**

```

if (interactive()) {
  ui <- fluidPage(
    shinyjs::useShinyjs(),
    shinyGovstyle::header(
      main_text = "Example",
      secondary_text = "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
    })
  }
  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 inputid 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

**Examples**

```

if (interactive()) {
  ui <- fluidPage(
    shinyjs::useShinyjs(),
    shinyGovstyle::header(
      main_text = "Example",
      secondary_text = "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
    })
  }
  shinyApp(ui = ui, server = server)
}

```

---

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. It doesn't look like the www.gov.uk/ style one, although this www.gov.uk/ doesn't seem to have a settle style as, for example it changes between Firefox and Chrome



**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 include error handling Defaults to FALSE.
error_message	Message to display on error. Defaults to NULL

**Value**

a file input html shiny object

**Examples**

```
if (interactive()) {
  ui <- fluidPage(
    # Required for error handling function
    shinyjs::useShinyjs(),
    shinyGovstyle::header(
      main_text = "Example",
      secondary_text = "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")
    }
  })
}
shinyApp(ui = ui, server = server)
}

```

font

*Font Function***Description**

This function adds rge nta fonts to the app. See <https://design-system.service.gov.uk/styles/typography/> for when they are allowed.

**Usage**

```
font()
```

**Value**

no value returned. This loads the font css file

**Examples**

```

if (interactive()) {

  ui <- fluidPage(
    font(),
    shinyGovstyle::header(
      main_text = "Example",
      secondary_text = "User Examples",
      logo="shinyGovstyle/images/moj_logo.png")
  )
}

```

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

---

footer

*Footer Function*

---

## Description

This function create a gov style footer for your page

## Usage

```
footer(full = FALSE)
```

## Arguments

**full** Whenever you want to have blank footer or official gov version. Defaults to FALSE

## Value

a footer html shiny object

## Examples

```
if (interactive()) {  
  ui <- fluidPage(  
    shinyGovstyle::header(  
      main_text = "Example",  
      secondary_text = "User Examples",  
      logo="shinyGovstyle/images/moj_logo.png"),  
    shinyGovstyle::banner(  
      inputId = "banner", type = "beta", 'This is a new service'),  
    tags$br(),  
    tags$br(),  
    shinyGovstyle::footer(full = TRUE)  
  )  
  server <- function(input, output, session) {}  
  shinyApp(ui = ui, server = server)  
}
```

---

govTable	<i>Table Function</i>
----------	-----------------------

---

### 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. Options are three-quarters, two-thirds, one-half, one-third, one-quarter. Default is NULL.

### Value

an table html shiny object

### Examples

```
if (interactive()) {

  Months <- c("January", "February", "March")
  Bikes <- c("£85", "£75", "£165")
  Cars <- c("£95", "£55", "£125")

  example_data <- data.frame(Months, Bikes, Cars)

  ui <- fluidPage(
    shinyGovstyle::header(
      main_text = "Example",
      secondary_text = "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::govTable(
    "tab1", example_data, "Test", "1", num_col = c(2,3),
    width_overwrite = c("one-half", "one-quarter", "one-quarter"))
  ),

  shinyGovstyle::footer(full = TRUE)
)

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

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 object.

## Examples

```

if (interactive()) {

  # Create an example dataset
  tabs <- c(rep("Past Day", 3),
            rep("Past Week", 3),
            rep("Past Month", 3),
            rep("Past Year", 3))
  Case_manager <- rep(c("David Francis", "Paul Farmer", "Rita Patel"),4)
  Cases_open <- c(3, 1, 2, 24, 16, 24, 98, 122, 126, 1380, 1129, 1539)
}

```

```

Cases_closed <- c(0, 0, 0, 18, 20, 27, 95, 131, 142, 1472, 1083, 1265)
data <- data.frame(tabs, Case_manager, Cases_open, Cases_closed)

ui <- fluidPage(
  shinyGovstyle::header(
    main_text = "Example",
    secondary_text = "User Examples",
    logo="shinyGovstyle/images/moj_logo.png"),
  shinyGovstyle::gov_layout(size = "two-thirds",
    shinyGovstyle::govTabs(data, "tabs")),
  shinyGovstyle::footer(full = TRUE)
)

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

### Examples

```

if (interactive()) {
  ui <- fluidPage(
    shinyGovstyle::header(
      main_text = "Example",
      secondary_text = "User Examples",
      logo="shinyGovstyle/images/moj_logo.png"),
    shinyGovstyle::gov_layout(size = "full",

```

```

    shinyGovstyle::panel_output(
      inputId = "panel1",
      main_text = "Application Complete",
      sub_text = "Thank you for submitting your application.
Your reference is xvsiq")
    ),
    shinyGovstyle::footer(full = TRUE)
  )

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

### Examples

```

if (interactive()) {

  # 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 <- fluidPage(
  shinyGovstyle::header(
    main_text = "Example",
    secondary_text = "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) {}
shinyApp(ui = ui, server = server)
}

```

---

header

*Header Function*


---

## Description

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

## Usage

```

header(
  main_text,
  secondary_text,
  logo = NULL,
  main_link = "#",
  secondary_link = "#",
  logo_width = 36,
  logo_height = 32
)

```

## Arguments

main_text	Main text that goes in the header
secondary_text	Secondary header to supplement the main text
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_link	Add a link for clicking on main text
secondary_link	Add a link for clicking on secondary header.
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 object

**Examples**

```
if (interactive()) {  
  
  ui <- fluidPage(  
    shinyGovstyle::header(  
      main_text = "Example",  
      secondary_text = "User Examples",  
      logo="shinyGovstyle/images/moj_logo.png")  
    )  
  
  server <- function(input, output, session) {}  
  
  shinyApp(ui = ui, server = server)  
}
```

---

heading\_text

*Heading Text Function*

---

**Description**

This function create a heading text

**Usage**

```
heading_text(text_input, size = "xl")
```

**Arguments**

text_input	Text to display
size	Text size using xl, l, m, s. Defaults to xl.

**Value**

a heading text html shiny object

**Examples**

```
if (interactive()) {  
  ui <- fluidPage(  
    shinyGovstyle::header(  
      main_text = "Example",  
      secondary_text = "User Examples",  
      logo="shinyGovstyle/images/moj_logo.png"),  
    shinyGovstyle::gov_layout(size = "two-thirds",  
      shinyGovstyle::heading_text("This is great text", "m")  
    )  
  )  
  server <- function(input, output, session) {}  
  shinyApp(ui = ui, server = server)  
}
```

```

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

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

```

---

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 object

**Examples**

```

if (interactive()) {

  ui <- fluidPage(
    # Required for error handling function
    shinyjs::useShinyjs(),
    shinyGovstyle::header(
      main_text = "Example",
      secondary_text = "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")
      }
    })
  }
  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 object

**Examples**

```
if (interactive()) {
  ui <- fluidPage(
    shinyGovstyle::header(
      main_text = "Example",
      secondary_text = "User Examples",
      logo="shinyGovstyle/images/moj_logo.png"),
    shinyGovstyle::gov_layout(size = "two-thirds",
      shinyGovstyle::insert_text(
        inputId = "note",
        text = "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) {}
  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 object

**Examples**

```
if (interactive()) {
  ui <- fluidPage(
    shinyGovstyle::header(
      main_text = "Example",
      secondary_text = "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) {}
  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**

<code>...</code>	include the components of the UI that you want within the main page. These components are made to flow through each other. See example
<code>inputID</code>	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

### Examples

```
if (interactive()) {
  ui <- fluidPage(
    shinyGovstyle::header(
      main_text = "Example",
      secondary_text = "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) {}
  shinyApp(ui = ui, server = server)
}
```

---

noti\_banner

*Notification Banner Function*

---

## Description

This function create 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 object

## Examples

```
if (interactive()) {

  ui <- fluidPage(
    shinyGovstyle::header(
      main_text = "Example",
      secondary_text = "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) {}
}
```

```

  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 object

### Examples

```

if (interactive()) {
  ui <- fluidPage(
    shinyGovstyle::header(
      main_text = "Example",
      secondary_text = "User Examples",
      logo="shinyGovstyle/images/moj_logo.png"),
    shinyGovstyle::gov_layout(size = "full",
      shinyGovstyle::panel_output(
        inputId = "panel1",
        main_text = "Application Complete",
        sub_text = "Thank you for submitting your application.
                    Your reference is xvsiq")
      ),
    shinyGovstyle::footer(full = TRUE)
  )
  server <- function(input, output, session) {}
  shinyApp(ui = ui, server = server)
}

```



---

radio_button_Input	<i>Radio Button Function</i>
--------------------	------------------------------

---

### 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 <a href="#">checkboxGroupInput</a> . 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 object

**Examples**

```

if (interactive()) {

  ui <- fluidPage(
    # Required for error handling function
    shinyjs::useShinyjs(),
    shinyGovstyle::header(
      main_text = "Example",
      secondary_text = "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")
      }
    })
  }
  shinyApp(ui = ui, server = server)
}

```

---

`run_example`*Example Function*

---

**Description**

This function runs a shiny example using different parts of the package

**Usage**

```
run_example()
```

**Value**

a 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**

<code>inputId</code>	Input id for the component
<code>label</code>	Insert the text for the label.
<code>select_text</code>	Add the text that will apply in the drop down as a list
<code>select_value</code>	Add the value that will be used for each selection.

**Value**

a select input html shiny object

**Examples**

```

if (interactive()) {
  ui <- fluidPage(
    shinyGovstyle::header(
      main_text = "Example",
      secondary_text = "User Examples",
      logo="shinyGovstyle/images/moj_logo.png"),
    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")),
      tags$br()
    ),
    shinyGovstyle::footer(full = TRUE)
  )

  server <- function(input, output, session) {}
  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, turquoise, blue, purple, pink, red, orange and yellow

**Value**

a tag html shiny object.

**Examples**

```

if (interactive()) {
  ui <- fluidPage(
    shinyGovstyle::header(
      main_text = "Example",
      secondary_text = "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) {}
  shinyApp(ui = ui, server = server)
}

```

---

text_area_Input	<i>Text Area Input Function</i>
-----------------	---------------------------------

---

**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 icnlud 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 object

**Examples**

```
text_area_Input("taId", "Can you provide more detail?",
  "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 area 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 object

**Examples**

```

## Only run examples in interactive R sessions
if (interactive()) {

  ui <- fluidPage(
    # Required for error handling function
    shinyjs::useShinyjs(),
    shinyGovstyle::header(
      main_text = "Example",
      secondary_text = "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")
      }
    })
  }

  # Run the application
  shinyApp(ui = ui, server = server)
}

```

**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 object

**Examples**

```
if (interactive()) {
  ui <- fluidPage(
    shinyGovstyle::header(
      main_text = "Example",
      secondary_text = "User Examples",
      logo="shinyGovstyle/images/moj_logo.png"),
    shinyGovstyle::gov_layout(size = "two-thirds",
      shinyGovstyle::warning_text(
        inputId = "warn1",
        text = "You can be fined up to £5,000 if you do not register.")
    ),
    shinyGovstyle::footer(full = TRUE)
  )

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

---

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

**Examples**

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

# Index

- \* **accordion**
  - accordion, 2
- \* **area**
  - text\_area\_Input, 37
- \* **backlink**
  - backlink\_Input, 3
- \* **banner**
  - banner, 4
  - cookieBanner, 8
  - noti\_banner, 31
- \* **box**
  - details, 11
- \* **button**
  - button\_Input, 5
- \* **checkbox**
  - checkbox\_Input, 6
- \* **cookie**
  - cookieBanner, 8
- \* **count**
  - word\_count, 40
- \* **date**
  - date\_Input, 9
- \* **details**
  - details, 11
- \* **error\_summary\_update**
  - error\_summary\_update, 15
- \* **error\_summary**
  - error\_summary, 14
- \* **error**
  - error\_off, 12
  - error\_on, 13
- \* **example**
  - run\_example, 35
- \* **field**
  - input\_field, 26
- \* **file**
  - file\_Input, 16
- \* **font**
  - font, 18
- \* **footer**
  - footer, 19
- \* **header**
  - header, 24
- \* **heading**
  - heading\_text, 25
- \* **input**
  - file\_Input, 16
  - input\_field, 26
  - text\_Input, 38
- \* **inserttext**
  - insert\_text, 27
- \* **label**
  - label\_hint, 28
- \* **list**
  - gov\_summary, 23
- \* **notification**
  - noti\_banner, 31
- \* **panel**
  - panel\_output, 32
- \* **radiobuttons**
  - radio\_button\_Input, 33
- \* **select**
  - select\_Input, 35
- \* **style**
  - gov\_layout, 22
  - layouts, 29
- \* **summary**
  - gov\_summary, 23
- \* **table**
  - govTable, 20
  - govTabs, 21
- \* **tab**
  - govTabs, 21
- \* **tag**
  - tag\_Input, 36
- \* **text**
  - text\_area\_Input, 37
  - text\_Input, 38

- \* **warning**
  - warning\_text, 39
- \* **word**
  - word\_count, 40
- accordion, 2
- backlink\_Input, 3
- banner, 4
- button\_Input, 5
- checkbox\_Input, 6
- checkboxGroupInput, 33
- cookieBanner, 8
- date\_Input, 9
- details, 11
- error\_off, 12
- error\_on, 13
- error\_summary, 14
- error\_summary\_update, 15
- file\_Input, 16
- font, 18
- footer, 19
- gov\_box (layouts), 29
- gov\_layout, 22
- gov\_main\_layout (layouts), 29
- gov\_row (layouts), 29
- gov\_summary, 23
- gov\_text (layouts), 29
- govTable, 20
- govTabs, 21
- header, 24
- heading\_text, 25
- input\_field, 26
- insert\_text, 27
- label\_hint, 28
- layouts, 29
- noti\_banner, 31
- panel\_output, 32
- radio\_button\_Input, 33
- run\_example, 35
- select\_Input, 35
- tag\_Input, 36
- text\_area\_Input, 37
- text\_Input, 38
- warning\_text, 39
- word\_count, 40