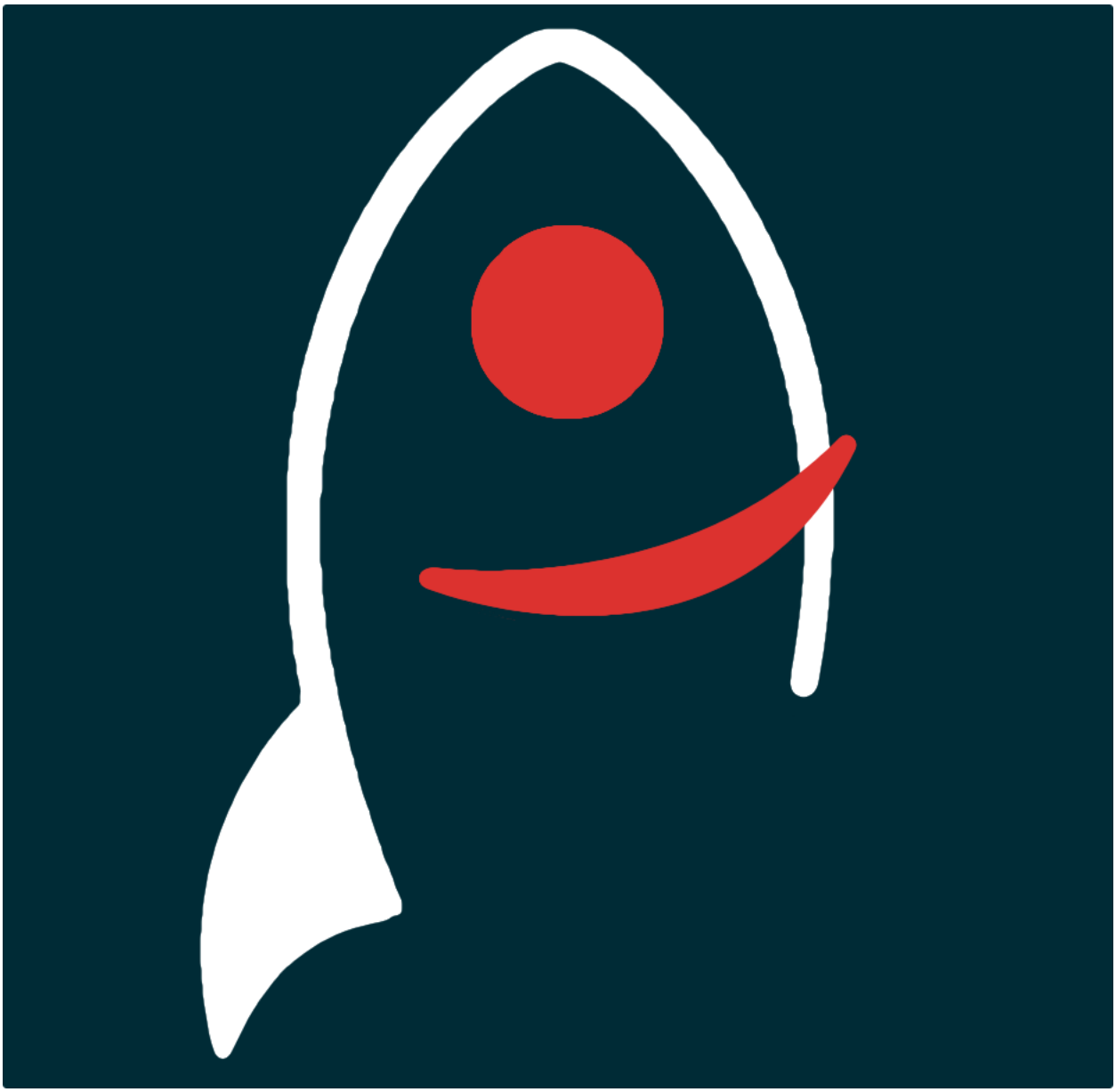

khufu Documentation

Release v1.3.3

Dave Young

2022

TABLE OF CONTENTS

Twitter Bootstrap elements via Python.

Documentation for khufu is hosted by Read the Docs (development version and master version). The code lives on github. Please report any issues you find here.

CHAPTER

ONE

FEATURES

-

1.1 Installation

The easiest way to install khufu is to use `pip` (here we show the install inside of a conda environment):

```
conda create -n khufu python=3.7 pip
conda activate khufu
pip install khufu
```

Or you can clone the github repo and install from a local version of the code:

```
git clone git@github.com:thespacedoctor/khufu.git
cd khufu
python setup.py install
```

To upgrade to the latest version of khufu use the command:

```
pip install khufu --upgrade
```

To check installation was successful run `khufu -v`. This should return the version number of the install.

1.1.1 Development

If you want to tinker with the code, then install in development mode. This means you can modify the code from your cloned repo:

```
git clone git@github.com:thespacedoctor/khufu.git
cd khufu
python setup.py develop
```

Pull requests are welcomed!

1.2 Initialisation

Before using khufu you need to use the `init` command to generate a user settings file. Running the following creates a yaml settings file in your home folder under `~/.config/khufu/khufu.yaml`:

```
khufu init
```

The file is initially populated with khufu's default settings which can be adjusted to your preference.

If at any point the user settings file becomes corrupted or you just want to start afresh, simply trash the `khufu.yaml` file and rerun `khufu init`.

1.2.1 Modifying the Settings

Once created, open the settings file in any text editor and make any modifications needed.

1.2.2 Basic Python Setup

If you plan to use `khufu` in your own scripts you will first need to parse your settings file and set up logging etc. One quick way to do this is to use the `fundamentals` package to give you a logger, a settings dictionary and a database connection (if connection details given in settings file):

```
## SOME BASIC SETUP FOR LOGGING, SETTINGS ETC
from fundamentals import tools
from os.path import expanduser
home = expanduser("~")
settingsFile  = home + "/.config/khufu/khufu.yaml"
su = tools(
    arguments={"settingsFile": settingsFile},
    docString=__doc__,
)
arguments, settings, log, dbConn = su.setup()
```

1.3 Todo List

Todo:

- [ ] when complete, clean popover function
- [ ] when complete add logging
- [ ] when complete, decide whether to abstract function to another module

(The *original entry* is located in /home/docs/checkouts/readthedocs.org/user_builds/khufu/envs/develop/lib/python3.7/site-packages/khufu-1.3.3-py3.7.egg/khufu/addons/popover.py:docstring of khufu.addons.popover.popover, line 22.)

Todo:

- [ ] when complete, clean popover function
- [ ] when complete add logging

- [ ] when complete, decide whether to abstract function to another module

(The original entry is located in /home/docs/checkouts/readthedocs.org/user_builds/khufu/envs/develop/lib/python3.7/site-packages/khufu-1.3.3-py3.7.egg/khufu/addons/popover.py:docstring of khufu.addons.popover.popover, line 22.)

Todo:

- nice!

(The *original entry* is located in /home/docs/checkouts/readthedocs.org/user_builds/khufu/checkouts/develop/docs/source/_template_.mo
line 1.)

1.4 Release Notes

v1.3.3 - May 10, 2022

- **FIXED** doc fixes

v1.3.2 - December 3, 2020

- **fixed** image wells not getting imported correctly

v1.3.1 - June 25, 2020

- **fixed** small Python 2 to 3 issues

v1.3.0 - May 25, 2020

- Now compatible with Python 3.*

CHAPTER

TWO

API REFERENCE

2.1 Classes

khufu.forms.login_form	*The worker class for the login_form module*
khufu.images.imageWell	*Framework for a bootstrap style well containing thumbnail images that can be clicked on to reveal a modal of more imformation*
khufu.images.imagingModal	*An image and modal – click on the image to present the modal of the larger image with download options*
khufu.modals.modalForm	*The worker class for the modalForm module*
khufu.scaffolding.login_page	*The worker class for the login_page module*
khufu.tables.sortable_table	*The worker class for the sortable_table module*

2.1.1 login_form *(class)*

class login_form(*log*, *iconPath*, *message*)

Bases: object

The worker class for the login_form module

Key Arguments

- log – logger
- iconPath – path to webapp icon
- message – message to display (warning)

Methods

close()	
get()	*get the login_form object*

get()

get the login_form object

Return

- formContent – the content of the login form

2.1.2 imageWell *(class)*

class imageWell(*log*, *title='Title of Image Well'*, *description='Description of the content of the image well'*, *imageDisplay='rounded'*)

Bases: object

Framework for a bootstrap style well containing thumbnail images that can be clicked on to reveal a modal of more imformation

Key Arguments

- log – logger
- title – Title of Image Well
- description – Description of the content of the image well
- imageDisplay – [rounded | circle | polaroid | False]

Methods

appendImage(imagePath, imageTitle[, ...])	*append an image to the image well*
close()	
get()	*get the image well*

appendImage(*imagePath*, *imageTitle*, *modalHeaderContent=''*, *modalFooterContent=''*, *modalFooterButtons=[]*)

append an image to the image well

Key Arguments

- imagePath – path to the image to add to the well
- imageTitle – text to tag the image with
- modalHeaderContent – the heading for the modal
- modalFooterContent – the footer (usually buttons)

Return

- None

get()

get the image well

Return

- imageWellRow – the html text

2.1.3 imagingModal *(class)*

class imagingModal(*log*, *dbConn=False*, *imagePath=False*, *display=False*, *modalHeaderContent=''*, *modalFooterContent=''*, *modalFooterButtons=[]*, *stampWidth=180*, *modalImageWidth=400*, *downloadLink=False*)

Bases: `object`

An image and modal – click on the image to present the modal of the larger image with download options

Key Arguments

- `dbConn` – mysql database connection
- `log` – logger
- `display` – [rounded | circle | polaroid | False]
- `imagePath` – path to the image to be displayed
- `modalHeaderContent` – the heading for the modal
- `modalFooterContent` – the footer (usually buttons)
- `stampWidth` – 180
- `modalImageWidth` – 400
- `downloadLink` – False

Methods

`close`()	
`get`()	*get the object*

get()

get the object

Return

- `imageModal`

2.1.4 modalForm *(class)*

class modalForm(*log*, *title*, *postToScriptUrl*, *reloadToUrl*, *formClassName=False*)

Bases: `object`

The worker class for the modalForm module

Key Arguments

- `log` – logger
- `title` – title
- `postToScriptUrl` – postToScriptUrl
- `reloadToUrl` – reloadToUrl
- `formClassName` – give a class name to form (if required by CSS or JS)

Methods

add_form_object(formObject[, label, hidden])	*add a form objec to the modal form*
add_hidden_parameter_value(key, value)	*add hidden parameter value to the form (to be submitted with the form but does not need user input)*
close()	

get()	*get the modalForm object*
get_form_action_buttons(formId)	*get form action buttons*
set_hidden_parameters()	*get hidden parameters*

add_form_object (*formObject*, *label=''*, *hidden=False*)
add a form objec to the modal form

Key Arguments

- formObject – the object to add to the form
- label – label to assign to the object
- hidden – is the form object hidden initially?

add_hidden_parameter_value (*key*, *value*)
add hidden parameter value to the form (to be submitted with the form but does not need user input)

Key Arguments

- key – the key for the hidden value (will be appended to query string when form submitted)
- value – the value of the hidden parameter

get ()
get the modalForm object

Return

- modalForm

get_form_action_buttons (*formId*)
get form action buttons

Key Arguments

- formId – the HTML id of the form

Return

- actionButtons – the action buttos for the form (cancel, submit)

set_hidden_parameters ()
get hidden parameters

2.1.5 login_page *(class)*

class login_page(*log*, *mainCssFilePath='/static/styles/main.css'*, *jsFilePath='/static/js/main-ck.js'*, *pageTitle='Login'*, *iconPath=''*, *came_from='/'*, *message=''*)

Bases: `object`

The worker class for the login_page module

Key Arguments

- `log` – logger
- `mainCssFilePath` – the filepath of the main CSS file
- `jsFilePath` – the filepath of the main JS file
- `pageTitle` – pageTitle
- `iconPath` – webapp icon path
- `came_from` – the url this login page was triggered from
- `message` – message to display as notification

Methods

`close`()	
`get`()	*get the login_page object*

get()

get the login_page object

Return

- `login_page` – the html login page

2.1.6 sortable_table *(class)*

class sortable_table(*log*, *dbConn=False*, *currentPageUrl=''*, *columnsToDisplay=[]*, *defaultSort=False*, *tableRowsDictionary={}*)

Bases: `object`

The worker class for the sortable_table module

Key Arguments

- `dbConn` – mysql database connection
- `log` – logger
- `currentPageUrl` – the baseurl for the webpage
- `columnsToDisplay` – a list of column objects for the table
- `defaultSort` – the column to sort on by default
- `tableRowsDictionary` – dictionary of column names and values (e.g. a mysql query result)

Methods

close()

get()	*get the sortable_table object*
get_table_body()	*get table body*
get_table_head()	*get table head*

get()
get the sortable_table object

Return

- sortable_table

get_table_body()
get table body

Return

- tableBody

get_table_head()
get table head

Return

- tableHead – the table head

2.2 Functions

khufu.addons.closeIcon	*Get close icon.*
khufu.addons.mediaObject	*Generate an abstract object style for building various types of components (like blog comments, Tweets, etc) that feature a left- or right-aligned image alongside textual content.*
khufu.addons.popover	*popover to provide helper text or some secondary info about an element*
khufu.addons.well	*Get well.*
khufu.buttons.button	*Generate a button - TBS style*
khufu.buttons.buttonGroup	*Generate a buttonGroup - TBS style*
khufu.commonutils.getpackagepath	*getpackagepath*
khufu.dropdowns.dropdown	*get a toggleable, contextual menu for displaying lists of links.*
khufu.dropdowns.dropdownLinkList	*dropdownLinkList*
khufu.forms.checkbox	*Generate a checkbox - TBS style*
khufu.forms.controlRow	*generate a form row*
khufu.forms.form	*Generate a form - TBS style*
khufu.forms.formActions	*Generate a formActions - TBS style*
khufu.forms.formInput	*Generate a form input - TBS style*
khufu.forms.horizontalFormControlGroup	*Generate a horizontal form control group (row) - TBS style*
khufu.forms.horizontalFormControlLabel	*set a horizontal form label*

continues on next page

Table 8 – continued from previous page

khufu.forms.radio	*Generate a radio - TBS style*
khufu.forms.searchForm	*Generate a search-form - TBS style*
khufu.forms.select	*Generate a select - TBS style*
khufu.forms.textarea	*Generate a textarea - TBS style*
khufu.forms.uneditableInput	*Generate a uneditableInput - TBS style*
khufu.helpers.hide_from_device	*hide from device)*
khufu.helpers.unescape_html	*Unescape a string previously escaped with cgi.escape()*
khufu.images.image	*Create an HTML image (with ot without link).
khufu.images.thumbnail_div	*Generate a thumbnail - TBS style*
khufu.images.thumbnails	*Generate a thumbnail - TBS style*
khufu.labelsAndBadges.alert	*Generate a alert - TBS style*
khufu.labelsAndBadges.badge	*Generate a badge - TBS style*
khufu.labelsAndBadges.label	*Generate a label - TBS style*
khufu.labelsAndBadges.progressBar	*Generate a progress bar - TBS style*
khufu.labelsAndBadges.stackedProgressBar	*Generate a progress bar - TBS style*
khufu.modals.modal	*generate a modal to by generated with a js event*
khufu.navigation.is_navStyle_active	*is navStyle active*
khufu.navigation.navBar	*Generate a navBar - TBS style*
khufu.navigation.nav_list	*Create an html list of navigation items from the required python list*
khufu.navigation.pagination	*Generate pagination - TBS style.
khufu.navigation.responsive_navigation_bar	*Create a twitter bootstrap responsive nav-bar component*
khufu.navigation.searchbox	*Create a Search box*
khufu.navigation.tabbableNavigation	Generate a tabbable Navigation
khufu.plots.svg	*svg*
khufu.scaffolding.body	*Generate an HTML body*
khufu.scaffolding.grid_column	*Get a column block for the Twiiter Bootstrap static layout grid.*
khufu.scaffolding.grid_row	*Create a row using the Twitter Bootstrap static layout grid.
khufu.scaffolding.head	*Generate an html head element for your webpage*
khufu.scaffolding.htmlDocument	*The doctype and html tags*
khufu.scaffolding.row_adjustable	*row adjustable*
khufu.tables.table	*Generate a table - TBS style*
khufu.tables.tableCaption	*Generate a table caption - TBS style*
khufu.tables.tbody	*Generate a table body - TBS style*
khufu.tables.td	*Generate a table data cell - TBS style*
khufu.tables.th	*Generate a table header cell - TBS style*
khufu.tables.thead	*Generate a table head - TBS style*
khufu.tables.tr	*Generate a table row - TBS style*
khufu.typography.a	*Generate an anchor - TBS style*
khufu.typography.abbr	*Get HTML5 Abbreviation*
khufu.typography.address	*Get The HTML5 address element*
khufu.typography.blockquote	*Get HTML5 Blockquote*
khufu.typography.coloredText	*Colour text a given colour*
khufu.typography.descriptionLists	*A list of definitions.*
khufu.typography.emphasizeText	*Get HTML's default emphasis tags with lightweight styles.*

continues on next page

Table 8 – continued from previous page

khufu.typography.heroUnit	*Generate a heroUnit - TBS style*
khufu.typography.li	*Generate a li - TBS style*
khufu.typography.ol	*An ordered list*
khufu.typography.p	*Get a Paragraph element*
khufu.typography.pageHeader	*Generate a pageHeader - TBS style*
khufu.typography.ul	*Get An unordered list – can be used for navigation, stacked tab and pill*
khufu.urls.default_fields	*default feilds*

2.2.1 closeIcon (*function*)

closeIcon()

Get close icon. The generic close icon for dismissing content like modals and alerts.

Key Arguments

Return

- closeIcon – the closeIcon

2.2.2 mediaObject (*function*)

mediaObject(*displayType='div'*, *imagePath=''*, *headlineText=''*, *otherContent=False*, *nestedMediaObjects=False*)

Generate an abstract object style for building various types of components (like blog comments, Tweets, etc) that feature a left- or right-aligned image alongside textual content.

Key Arguments

- displayType – the display style of the media object [“div” | “li”]
- img – the image to include
- headlineText – the headline text for the object
- otherContent – other content to be displayed inside the media object
- nestedMediaObjects – nested media objects to be appended

Return

- media – the media object

2.2.3 popover (*function*)

popover(*tooltip=False*, *placement=False*, *trigger=False*, *title=False*, *content=False*, *delay=200*, *after=False*)

popover to provide helper text or some secondary info about an element

Key Arguments

- tooltip – use tooltip instead of popover
- placement – direction popover expands into [top | bottom | left | right]
- trigger – the trigger for the popover [False | click | hover | focus | manual]
- title – the popover title

- `content` – the popover content
- `delay` – delay in ms
- `after` – place the div required by the

Return

- `popover` - the popover helper text to be added to an element

Todo:

- [ ] when complete, clean popover function
- [ ] when complete add logging
- [ ] when complete, decide whether to abstract function to another module

---

2.2.4 well (*function*)

`well`(*wellText=''*, *wellSize='default'*, *htmlId=False*, *htmlClass=False*)

Get well. Use the well as a simple effect on an element to give it an inset effect.

Key Arguments

- `wellText` – the text to be displayed in the well
- `wellSize` – the size of the well ["default" | "large" | "small"]

Return

- `well` – the well

2.2.5 button (*function*)

`button`(*buttonText=''*, *buttonStyle='default'*, *buttonSize='default'*, *htmlId=False*, *htmlClass=False*, *href=False*, *pull=False*, *submit=False*, *block=False*, *disable=False*, *dataToggle=False*, *popover=False*, *postInBackground=False*, *notification=False*, *close=False*, *formId=False*)

Generate a button - TBS style

Key Arguments

- `buttonText` – the text to display on the button
- `buttonStyle` – the style of the button required [default | primary | info | success | warning | danger | inverse | link]
- `buttonSize` – the size of the button required [large | small | mini]
- `htmlId` – the htmlId for the button
- `href` – link the button to another location?
- `pull` – left, right or center
- `submit` – set to true if a form button [true | false]
- `block` – create block level buttons—those that span the full width of a parent [True | False]
- `disable` – this class is only for aesthetic; you must use custom JavaScript to disable links here
- `dataToggle` – for use with js to launch, for example, a modal

- popover – add a popover element for this button

Return

- button – the button

2.2.6 buttonGroup (*function*)

buttonGroup(*buttonList=[]*, *format='default'*, *pull=False*)
Generate a buttonGroup - TBS style

Key Arguments

- buttonList – a list of buttons
- format – format of the button [default | toolbar | vertical]

Return

- buttonGroup – the buttonGroup

2.2.7 getpackagepath (*function*)

getpackagepath()
getpackagepath

Return

- packagePath – path to the host package

2.2.8 dropdown (*function*)

dropdown(*buttonSize='default'*, *buttonColor='default'*, *linkNotButton=False*, *menuTitle='#'*, *splitButton=False*, *splitButtonHref=False*, *linkList=[]*, *separatedLinkList=False*, *pull=False*, *htmlId=False*, *htmlClass=False*, *direction='down'*, *popover=False*, *onPhone=True*, *onTablet=True*, *onDesktop=True*)
get a toggleable, contextual menu for displaying lists of links. Made interactive with the dropdown JavaScript plugin. You need to wrap the dropdown's trigger and the dropdown menu within .dropdown, or another element that declares position: relative

- buttonSize – size of button [mini | small | default | large]
- buttonColor – [default | sucess | error | warning | info]
- menuTitle – the title of the menu
- splitButton – split the button into a separate action button and a dropdown
- splitButtonHref – link for the split button
- linkList – a list of (linked) items items that the menu should display
- separatedLinkList – a list of (linked) items items that the menu should display below divider
- pull – [false | right | left] (e.g Add right to a .dropdown-menu to right align the dropdown menu.)
- direction – drop [down | up]
- popover – add a popover for this dropdown
- onPhone – does this container get displayed on a phone sized screen

- `onTablet` – does this container get displayed on a tablet sized screen
- `onDesktop` – does this container get displayed on a desktop sized screen*

Return

- `dropdown` – the dropdown menu

2.2.9 dropdownLinkList (*function*)

`dropdownLinkList`(*linkDictionary={}*, *title='dropdown'*, *dropDirection='down'*)
dropdownLinkList

Key Arguments

- `log` – logger
- `linkDictionary` – { text : href }
- `title` – title for the dropdown
- `dropDirection` – up or down

Return

- `thisDropdown`

2.2.10 checkbox (*function*)

`checkbox`(*optionText=''*, *inline=False*, *optionNumber=1*, *htmlId=False*, *inlineHelpText=False*, *blockHelpText=False*, *disabled=False*, *checked=False*)
Generate a checkbox - TBS style

Key Arguments

- `optionText` – the text associated with this checkbox
- `inline` – display the checkboxes inline?
- `optionNumber` – option number of inline
- `htmlId` – htmlId
- `inlineHelpText` – inline and block level support for help text that appears around form controls
- `blockHelpText` – a longer block of help text that breaks onto a new line and may extend beyond one line
- `disabled` – add the disabled attribute on an input to prevent user input
- `checked` – the default checked/unchecked state of the box

Return

- `checkbox` – the checkbox

2.2.11 controlRow (*function*)

controlRow(*inputList=[]*)
generate a form row

Key Arguments

- `inputList` – list of inputs for the control row

Return

- `controlRow` – the controlRow

2.2.12 form (*function*)

form(*content=''*, *formType='inline'*, *postToScript=''*, *htmlId=False*, *htmlClass=False*, *navBarPull=False*, *postInBackground=False*, *redirectUrl=False*, *span=False*, *offset=False*, *openInNewTab=False*)
Generate a form - TBS style

Key Arguments

- `content` – the content
- `formType` – the type if the form required [“inline” | “horizontal” | “search” | “navbar-form” | “navbar-search”]
- `postToScript` – the script to post the form values to
- `htmlId` – the id for the form
- `navBarPull` – align the form is in a navBar [false | right | left]
- `postInBackground` – submit form in background without refreshing page
- `redirectUrl` – url to redirect to after form is submitted

Return

- `inlineForm` – the inline form

2.2.13 formActions (*function*)

formActions(*primaryButton=''*, *button2=False*, *button3=False*, *button4=False*, *button5=False*, *inlineHelpText=False*, *blockHelpText=False*)
Generate a formActions - TBS style

Key Arguments

- `primaryButton` – the primary button
- `button2` – another button
- `button3` – another button
- `button4` – another button
- `button5` – another button
- `inlineHelpText` – inline and block level support for help text that appears around form controls
- `blockHelpText` – a longer block of help text that breaks onto a new line and may extend beyond one line

Return

- `formActions` – the formActions

2.2.14 formInput (*function*)

formInput(*ttype='text'*, *placeholder=''*, *span=2*, *htmlId=False*, *searchBar=False*, *pull=False*, *prepend=False*, *append=False*, *button1=False*, *button2=False*, *prependDropdown=False*, *appendDropdown=False*, *inlineHelpText=False*, *blockHelpText=False*, *focusedInputText=False*, *rightText=False*, *required=False*, *disabled=False*, *defaultValue=False*, *hidden=False*, *divWrap=True*)

Generate a form input - TBS style

Key Arguments

- `ttype` – [text | password | datetime | datetime-local | date | month | time | week | number | float | email | url | search | tel | color]
- `placeholder` – the placeholder text
- `span` – column span
- `htmlId` – html id
- `searchBar` – is this input a searchbar?
- `pull` – [false | right | left] align form
- `prepend` – prepend text to the input.
- `append` – append text to the input.
- `button1` – do you want a button associated with the input?
- `button2` – as above for a 2nd button
  - `appendDropdown` – do you want a appended button-dropdown associated with the input?
  - `prependDropdown` – do you want a prepended button-dropdown associated with the input?
  - `inlineHelpText` – inline and block level support for help text that appears around form controls
  - `blockHelpText` – a longer block of help text that breaks onto a new line and may extend beyond one line
  - `focusedInputText` – make the input focused by providing some initial editable input text
  - `required` – required attribute if the field is not optional
  - `disabled` – add the disabled attribute on an input to prevent user input
  - `defaultValue` – a default value to be passed to action script
  - `hidden` – hide the CG from the user?
  - `divWrap` – wrap in div

Return

- `input` – the input

2.2.15 horizontalFormControlGroup (*function*)

horizontalFormControlGroup(*content=''*, *validationLevel=False*, *hidden=False*)
Generate a horizontal form control group (row) - TBS style

Key Arguments

- `content` – the content
- `validationLevel` – validation level [warning | error | info | success]
- `hidden` – hide the CG from the user?

Return

- `horizontalFormControlGroup` – the horizontal form control group

2.2.16 horizontalFormControlLabel (*function*)

horizontalFormControlLabel(*labelText=''*, *forId=False*, *sideLabel=False*, *location='left'*)
set a horizontal form label

Key Arguments

- `labelText` – the label text
- `forId` – what is the label for (id of the associated object)?

Return

- `horizontalFormRowLabel` – the horizontalFormRowLabel

2.2.17 radio (*function*)

radio(*optionText=''*, *optionNumber=1*, *htmlId=False*, *inlineHelpText=False*, *blockHelpText=False*, *disabled=False*, *checked=False*)
Generate a radio - TBS style

Key Arguments

- `optionText` – the text associated with this checkbox
- `optionNumber` – the order in the option list
- `htmlId` – the html id of the element
- `inlineHelpText` – inline and block level support for help text that appears around form controls
- `blockHelpText` – a longer block of help text that breaks onto a new line and may extend beyond one line
- `disabled` – add the disabled attribute on an input to prevent user input
- `checked` – is the radio button checked by default

Return

- `radio` – the radio

2.2.18 searchForm (*function*)

searchForm(*buttonText=''*, *span=2*, *inlineHelpText=False*, *blockHelpText=False*, *focusedInputText=False*, *htmlId=False*)

Generate a search-form - TBS style

Key Arguments

- `buttonText` – the button text
- `span` – column span
- `inlineHelpText` – inline and block level support for help text that appears around form controls
- `blockHelpText` – a longer block of help text that breaks onto a new line and may extend beyond one line
- `focusedInputText` – make the input focused by providing some initial editable input text
- `htmlId` – htmlId

Return

- `searchForm` – the search-form

2.2.19 select (*function*)

select(*optionList=[]*, *valueList=False*, *multiple=False*, *span=2*, *htmlId=False*, *htmlClass=False*, *inlineHelpText=False*, *blockHelpText=False*, *required=False*, *disabled=False*, *popover=False*, *extraAttributeTupleList=False*, *defaultOption=False*)

Generate a select - TBS style

Key Arguments

- `optionList` – the list of options
- `multiple` – display all the options at once?
- `span` – column span
- `htmlId` – the html id of the element
- `inlineHelpText` – inline and block level support for help text that appears around form controls
- `blockHelpText` – a longer block of help text that breaks onto a new line and may extend beyond one line
- `required` – required attribute if the field is not optional
- `disabled` – add the disabled attribute on an input to prevent user input
- `popover` – add helper text to the select
- `defaultOption` – option to select as default

Return

- `select` – the select

2.2.20 textarea (*function*)

textarea(*rows=''*, *span=2*, *placeholder=''*, *htmlId=False*, *inlineHelpText=False*, *blockHelpText=False*, *focusedInputText=False*, *required=False*, *disabled=False*, *prepopulate=False*)

Generate a textarea - TBS style

Key Arguments

- `rows` – the number of rows the text area should span
- `span` – column span
- `placeholder` – the placeholder text
- `htmlId` – html id for item
- `inlineHelpText` – inline and block level support for help text that appears around form controls
- `blockHelpText` – a longer block of help text that breaks onto a new line and may extend beyond one line
- `focusedInputText` – make the input focused by providing some initial editable input text
- `required` – required attribute if the field is not optional
- `disabled` – add the disabled attribute on an input to prevent user input

Return

- `textarea` – the textarea

2.2.21 uneditableInput (*function*)

uneditableInput(*placeholder=''*, *span=2*, *inlineHelpText=False*, *blockHelpText=False*)

Generate a uneditableInput - TBS style

Key Arguments

- `placeholder` – the placeholder text
- `span` – column span
- `inlineHelpText` – inline and block level support for help text that appears around form controls
- `blockHelpText` – a longer block of help text that breaks onto a new line and may extend beyond one line

Return

- `uneditableInput` – an uneditable input - the user can see but not interact

2.2.22 hide_from_device (*function*)

hide_from_device(*content=''*, *onPhone=True*, *onTablet=True*, *onDesktop=True*)

hide from device)

Key Arguments

- `content` - content to hide/show
- `onPhone` - onPhone?
- `onTablet` - onTablet?

- `onDesktop` - onDesktop?

Return

- `span` – span containings content with show/hide parameters

2.2.23 unescape_html (*function*)

`unescape_html`(*html*)

Unescape a string previously escaped with cgi.escape()

Key Arguments

- `dbConn` – mysql database connection
- `log` – logger
- `html` – the string to be unescaped

Return

- `html` – the unescaped string

2.2.24 image (*function*)

`image`(*src='http://placehold.it/200x200'*, *href=False*, *display=False*, *pull='left'*, *htmlClass=False*, *htmlId=False*, *thumbnail=False*, *width=False*, *height=False*, *onPhone=True*, *onTablet=True*, *onDesktop=True*, *clickToModal=False*, *openInNewTab=False*, *modal=False*)

Create an HTML image (with ot without link). Based on the Twitter bootstrap setup.

Key Arguments

- `src` – image url
- `href` – image link url
- `display` – how the image is to be displayed [rounded | circle | polaroid]
- `pull` – how to align the image if within a `<div>` [“left” | “right” | “center”]
- `htmlId` – the id of the image
- `htmlClass` – the class of the image
- `width` – the width of the image
- `onPhone` – does this container get displayed on a phone sized screen
- `onTablet` – does this container get displayed on a tablet sized screen
- `onDesktop` – does this container get displayed on a desktop sized screen
- `clickToModal` – if you want to display the image in a modal when clicked?
- `openInNewTab` – open image link in new tab?
- `modal` – is this linked to a modal?

Return

- `image` - the formatted image

2.2.25 thumbnail_div (*function*)

thumbnail_div(*div_content=''*)
Generate a thumbnail - TBS style

Key Arguments

- `div_content` – the html content of the thumbnail

Return

- `thumbnail` – the thumbnail with HTML content

2.2.26 thumbnails (*function*)

thumbnails(*listItems=[]*)
Generate a thumbnail - TBS style

Key Arguments

- `htmlContent` – the html content of the thumbnail

Return

- `thumbnail` – the thumbnail with HTML content

2.2.27 alert (*function*)

alert(*alertText=''*, *alertHeading=''*, *extraPadding=False*, *alertLevel='warning'*)
Generate a alert - TBS style

Key Arguments

- `alertText` – the text to be displayed in the alert
- `extraPadding` – for longer messages, increase the padding on the top and bottom of the alert wrapper
- `alertLevel` – the level of the alert [“warning” | “error” | “success” | “info”]

Return

- `alert` – the alert

2.2.28 badge (*function*)

badge(*text=''*, *level='default'*)
Generate a badge - TBS style

Key Arguments

- `text` – the text content
- `level` – the level colour of the badge [“default” | “success” | “warning” | “important” | “info” | “inverse”]

Return

- `badge` – the badge

2.2.29 label (*function*)

label(*text=''*, *level='default'*)
Generate a label - TBS style

Key Arguments

- `text` – the text content
- `level` – the level colour of the label ["default" | "success" | "warning" | "important" | "info" | "inverse"]

Return

- `label` – the label

2.2.30 progressBar (*function*)

progressBar(*barStyle='plain'*, *precentageWidth='10'*, *barLevel='info'*)
Generate a progress bar - TBS style

Key Arguments

- `barStyle` – style of the progress bar ["plain" | "striped" | "striped-active"]
- `precentageWidth` – the current progress of the bar
- `barLevel` – the level color of the bar ["info" | "warning" | "success" | "error"]

Return

- `progressBar` – the progressBar

2.2.31 stackedProgressBar (*function*)

stackedProgressBar(*barStyle='plain'*, *infoWidth='10'*, *successWidth='10'*, *warningWidth='10'*, *errorWidth='10'*)
Generate a progress bar - TBS style

Key Arguments

- `barLevel` – the level/color of progress ["info" | "success" | "warning" | "danger"]
- `barStyle` – style of the progress bar ["plain" | "striped" | "striped-active"]
- `infoWidth` – the precentage width of the info level bar
- `successWidth` – the precentage width of the success level bar
- `warningWidth` – the precentage width of the warning level bar
- `errorWidth` – the precentage width of the error level bar

Return

- `progressBar` – the progressBar

2.2.32 modal (*function*)

modal(*modalHeaderContent=''*, *modalBodyContent=''*, *modalFooterContent=''*, *htmlId=False*, *centerContent=False*, *htmlClass=False*)
generate a modal to by generated with a js event

Key Arguments

- `modalHeaderContent` – the heading for the modal
- `modalBodyContent` – the content (form or text)
- `modalFooterContent` – the foot (usually buttons)
- `htmlId` – id for button to hook onto with href
- `centerContent` - center the content in the form?
- `htmlClass` - htmlClass for the form

Return

- `modal` – the modal

2.2.33 is_navStyle_active (*function*)

is_navStyle_active(*log*, *thisPageName*, *thisPageId*)
is navStyle active

Key Arguments

- `log` – logger
- `thisPageName` – the thisPageName of the page
- `thisPageId` – the Id of this page

Return

- `navStyle` – boolean, true if the navStyle should be active, i.e. the link is to the currently viewed page

2.2.34 navBar (*function*)

navBar(*brand=''*, *contentList=[]*, *contentListPull=False*, *dividers=False*, *forms=False*, *fixedOrStatic=False*, *location='top'*, *responsive=False*, *dark=False*, *transparent=False*, *htmlClass=False*)
Generate a navBar - TBS style

Key Arguments

- `brand` – the website brand [image | text]
- `contentList` – the content list of li and dropdowns
- `contentListPull` – False, right, left
- `fixedOrStatic` – Fix the navbar to the top or bottom of the viewport, or create a static full-width navbar that scrolls away with the page [False | fixed | static]
- `location` – location of the navigation bar if fixed or static
- `dark` – Modify the look of the navbar by making it dark
- `transparent` – make the bar see-through

Return

- navBar – the navBar

2.2.35 nav_list (*function*)

nav_list(*itemList=[]*, *pull=False*, *onPhone=True*, *onTablet=True*, *onDesktop=True*)

Create an html list of navigation items from the required python list

Key Arguments

- itemList – items to be included in the navigation list
- pull – float the nav-list [False | 'right' | 'left']
- onPhone – does this container get displayed on a phone sized screen
- onTablet – does this container get displayed on a tablet sized screen
- onDesktop – does this container get displayed on a desktop sized screen

Return

- navList

2.2.36 pagination (*function*)

pagination(*listItems=''*, *size='default'*, *align='left'*)

Generate pagination - TBS style. Simple pagination inspired by Rdio, great for apps and search results.

Key Arguments

- listItems – the numbered items to be listed within the <ul> of the pagination block
- size – additional pagination block sizes ["mini" | "small" | "default" | "large"]
- align – change the alignment of pagination links ["left" | "center" | "right"]

Return

- pagination – the pagination

2.2.37 responsive_navigation_bar (*function*)

responsive_navigation_bar(*shade='dark'*, *brand=False*, *brandLink='#'*, *loginDetails=False*, *outsideNavList=False*, *insideNavList=False*, *htmlId=False*, *onPhone=True*, *onTablet=True*, *onDesktop=True*)

Create a twitter bootstrap responsive nav-bar component

Key Arguments

- shade – if dark then colors are inverted [False | 'dark']
- brand – the website brand [image | text]
- outsideNavList – nav-list to be contained outside collapsible content
- insideNavList – nav-list to be contained inside collapsible content
- htmlId –
- onPhone – does this container get displayed on a phone sized screen
- onTablet – does this container get displayed on a tablet sized screen

- `onDesktop` – does this container get displayed on a desktop sized screen

Return

- `navBar` –

2.2.38 searchbox (*function*)

searchbox(*size='medium'*, *htmlId=''*, *placeHolder=False*, *button=False*, *buttonSize='small'*, *buttonColor='grey'*, *navBar=False*, *pull=False*, *actionScript='#'*)

Create a Search box

Key Arguments

- `size` – size = mini | small | medium | large | xlarge | xxlarge
- `htmlId` – the html id of the search bar
- `placeholder` – placeholder text
- `button` – do you want a search button?
- `buttonSize`
- `buttonColor`
- `actionScript` – the script used to action the search text

Return

- `markup` – markup for the searchbar

2.2.39 tabbableNavigation (*function*)

tabbableNavigation(*contentDictionary={}*, *fadeIn=True*, *direction='top'*, *htmlClass=False*, *htmlId=False*, *uniqueNavigationId=False*, *contentCount={}*)

Generate a tabbable Navigation

Key Arguments

- `contentDictionary` – the content dictionary { name : content }
- `fadeIn` – make tabs fade in
- `direction` – the position of the tabs [above | below | left | right]
- `uniqueNavigationId` – a unique id for this navigation block if more than one on page

Return

- `tabbableNavigation` – the tabbableNavigation

2.2.40 svg (*function*)

svg(*htmlClass=False*, *dataUrl='#'*, *dataFormat='json'*, *disable=False*, *htmlId=False*, *chartType=''*, *span=12*, *height=False*)
svg

Key Arguments

- `htmlClass` – the extra html classes required
- `disable` – disable the plot (can enable via javascript)
- `htmlId` – the html id if required
- `csvUrl` – url to a csv file/csv data
- `chartType` – the type of chart required (determines which javascript function to trigger)
- `span` – span of chart area

Return

- `svg` – the svg element

2.2.41 body (*function*)

body(*navBar=False*, *content=''*, *htmlId=''*, *extraAttr=''*, *relativeUrlBase=False*, *responsive=True*, *googleAnalyticsCode=False*, *jsFilePath='main.js'*)
Generate an HTML body

Key Arguments

- `navBar` – the top navigation bar
- `htmlId` – *id* attribute of the body
- `content` – body content built from smaller HTML code blocks
- `extraAttr` – an extra attributes to be added to the body definition
- `relativeUrlBase` – how to get back to the document root
- `responsive` – should the webpage be responsive to screen-size?
- `googleAnalyticsCode` – google analytics code for the website
- `jsFilePath` – the name of the main javascript file

Return

- `body` – the body

2.2.42 grid_column (*function*)

grid_column(*span=1*, *offset=0*, *content=''*, *htmlId=False*, *htmlClass=False*, *pull=False*, *onPhone=True*, *onTablet=True*, *onDesktop=True*, *dataspy=False*)
Get a column block for the Twiiter Bootstrap static layout grid.

Key Arguments

- `log` – logger
- `span` – the relative width of the column
- `offset` – increase the left margin of the column by this amount

- `htmlId` – the id of the column
- `htmlClass` – the class of the column
- `pull` – left, right, or center
- `onPhone` – does this column get displayed on a phone sized screen
- `onTablet` – does this column get displayed on a tablet sized screen
- `onDesktop` – does this column get displayed on a desktop sized screen

Return

- `column` – the column

2.2.43 grid_row (*function*)

grid_row(*responsive=True*, *columns=''*, *htmlId=False*, *htmlClass=False*, *onPhone=True*, *onTablet=True*, *onDesktop=True*)

Create a row using the Twitter Bootstrap static layout grid. The static Bootstrap grid system utilizes 12 columns.

Key Arguments

- `responsive` – fluid layout if true, fixed if false
- `columns` – coulmns to be included in this row
- `htmlId` – the id of the row
- `htmlClass` – the class of the row
- `onPhone` – does this row get displayed on a phone sized screen
- `onTablet` – does this row get displayed on a tablet sized screen
- `onDesktop` – does this row get displayed on a desktop sized screen

Return

- `row` – the row

2.2.44 head (*function*)

head(*relativeUrlBase=False*, *mainCssFilePath='main.css'*, *pageTitle=''*, *extras=''*, *faviconLocation=False*, *assetsDirectory=False*)

Generate an html head element for your webpage

Key Arguments

`relativeUrlBase` – relative base url for js, css, image folders `pageTitle` – well, the page title! `mainCssFilePath` – css file path `extras` – any extra info to be included in the `head` element `faviconLocation` – path to faviconLocation if not in document root

Return

- `head` – the head

2.2.45 htmlDocument (*function*)

htmlDocument(*contentType=False*, *content=''*, *attachmentSaveAsName=False*)
The doctype and html tags

Key Arguments

- `content` – the head and body of the html page
- `attachmentSaveAsName` – save file as this name instead of opening in browser

Return

- `contentType` – the content type ["text/html"]
- `doctype` – the HTML5 doctype

2.2.46 row_adjustable (*function*)

row_adjustable(*span=12*, *offset=0*, *content=''*, *htmlId=False*, *htmlClass=False*, *onPhone=True*, *onTablet=True*, *onDesktop=True*)
row adjustable

Key Arguments

- `span` – the relative width of the column
- `offset` – increase the left margin of the column by this amount
- `content` – content for the row
- `htmlId` – the id of the column
- `htmlClass` – the class of the column
- `onPhone` – does this column get displayed on a phone sized screen
- `onTablet` – does this column get displayed on a tablet sized screen
- `onDesktop` – does this column get displayed on a desktop sized screen

Return

- `row` – the adjustable row

2.2.47 table (*function*)

table(*caption=''*, *thead=''*, *tbody=''*, *striped=True*, *bordered=False*, *hover=True*, *condensed=False*, *span=False*)
Generate a table - TBS style

Key Arguments

- `caption` – the table caption
- `thead` – the table head
- `tbody` – the table body
- `striped` – Adds zebra-striping to any odd table row
- `bordered` – Add borders and rounded corners to the table.
- `hover` – Enable a hover state on table rows within a `<tbody>`

- `condensed` – Makes tables more compact by cutting cell padding in half.

Return

- `table` – the table

2.2.48 tableCaption (*function*)

tableCaption(*content=''*)
Generate a table caption - TBS style

Key Arguments

- `content` – the content

Return

- `tableCaption` – the table caption

2.2.49 tbody (*function*)

tbody(*trContent=''*)
Generate a table body - TBS style

Key Arguments

- `trContent` – the table row content

Return

- `tbody` – the table body

2.2.50 td (*function*)

td(*content=False*, *color=False*, *span=False*)
Generate a table data cell - TBS style

Key Arguments

- `content` – the content
- `color` – [sucess | error | warning | info]

Return

- `td` – the table data cell

2.2.51 th (*function*)

th(*content=''*, *color=False*, *href=False*, *popover=False*, *span=False*, *columnWidth=False*)
Generate a table header cell - TBS style

Key Arguments

- `content` – the content
- `color` – [sucess | error | warning | info]
- `href` – add a link for the header cell (to sort for example)

- `popover` – add helper text

Return

- `th` – the table header cell

2.2.52 thead (*function*)

thead(*trContent=''*)
Generate a table head - TBS style

Key Arguments

- `trContent` – the table row content

Return

- `thead` – the table head

2.2.53 tr (*function*)

tr(*cellContent=''*, *color=False*, *href=False*, *popover=False*, *span=False*)
Generate a table row - TBS style

Key Arguments

- `cellContent` – the content - either `<td>`s or `<th>`s
- `color` – [sucess | error | warning | info]
- `href` – add a link for the whole table row

Return

- `tr` – the table row

2.2.54 a (*function*)

a(*content=''*, *href=False*, *tableIndex=False*, *thumbnail=False*, *pull=False*, *triggerStyle=False*, *htmlClass=False*, *htmlId=False*, *notification=False*, *postInBackground=False*, *openInNewTab=False*, *popover=False*)
Generate an anchor - TBS style

Key Arguments

- `content` – the content
- `href` – the href link for the anchor
- `tableIndex` – table index for the dropdown menus [False | -1]
- `pull` – direction to float the link (esp if image)
- `triggerStyle` – link to be used as a dropDown or tab trigger? [False | "dropdown" | "tab" | "thumbnail"]
- `htmlClass` – the class of the link
- `htmlId` – the html id of the anchor
- `postInBackground` – post to the href in the background, to fire data off to a cgi script to action without leaving page

- `notification` – a notification to be displayed on webpage
- `openInNewTab` – open the link in a new tab?

Return

- `a` – the a

2.2.55 abbr (*function*)

abbr(*abbreviation=''*, *fullWord=''*)
Get HTML5 Abbreviation

Key Arguments

- `abbreviation` – the abbreviation
- `fullWord` – the full word

Return

- abbr

2.2.56 address (*function*)

address(*name=False*, *addressLine1=False*, *addressLine2=False*, *addressLine3=False*, *phone=False*, *email=False*, *twitterHandle=False*)
Get The HTML5 address element

Key Arguments

- `name` – name of person
- `addressLine1` – first line of the address
- `addressLine2` – second line of the address
- `addressLine3` – third line of the address
- `phone` – telephone number
- `email` – email address
- `twitterHandle` – twitter handle

Return

- address

2.2.57 blockquote (*function*)

blockquote(*content=''*, *source=False*, *pullRight=False*)
Get HTML5 Blockquote

Key Arguments

- `content` – content to be quoted
- `source` – source of quote

Return

- None

2.2.58 coloredText (*function*)

coloredText(*text=''*, *color='red'*, *htmlClass=''*, *pull=False*, *size=False*, *addBackgroundColor=False*)
Colour text a given colour

Key Arguments

- `text` – the text to color
- `color` – the color
- `htmlClass` – the class for the text
- `size` – the relative size of the text
- `addBackgroundColor` – add a complimentary background color to the text

Return

- `text` – the coloured text span

2.2.59 descriptionLists (*function*)

descriptionLists(*orderedDictionary={}*, *sideBySide=False*)
A list of definitions.

Key Arguments

- `orderedDictionary` – the ordered dictionary of the terms and their definitions
- `sideBySide` – Make terms and descriptions in `<dl>` line up side-by-side.

Return

- `descriptionLists`

2.2.60 emphasizeText (*function*)

emphasizeText(*style='em'*, *text=''*)
Get HTML's default emphasis tags with lightweight styles.

Key Arguments

- `style` – the emphasis tag ["small" | "strong" | "em"]
- `text` – the text to emphasise

Return

- `emphasizeText` – the emphasized text

2.2.61 heroUnit (*function*)

heroUnit(*headline=''*, *tagline=''*, *buttonStyle='primary'*, *buttonText=''*, *buttonHref='#'*)
Generate a heroUnit - TBS style

Key Arguments

- `headline` – the headline text
- `tagline` – the tagline text for below the headline
- `buttonStyle` – the style of the button to be used
- `buttonText` – the text for the button
- `buttonHref` – the anchor link for the button

Return

- `heroUnit` – the heroUnit

2.2.62 li (*function*)

li(*content=''*, *span=False*, *disabled=False*, *submenuTitle=False*, *divider=False*, *navStyle=False*, *navDropDown=False*, *pager=False*, *pull=False*, *onPhone=True*, *onTablet=True*, *onDesktop=True*, *indent=False*, *hidden=False*)
Generate a li - TBS style

Key Arguments

- `content` – the content (if a subMenu for dropdown this should be `<ul>`)
- `span` – the column span [False | 1-12]
- `disabled` – add the disabled attribute on an grey out this list item. Note you can optionally swap anchors for spans to remove click functionality.
- `submenuTitle` – if a submenu (`<ul>`) is to be included as content, use this as the title.
- `divider` – if true this list item shall be a line
- `navStyle` – how is the navigation element to be displayed? [active | header]
- `navDropDown` – true if the list item is to be used as a dropdown in navigation
- `pager` – use the `<li>` within a pager navigation? [False | "previous" | "next"]

Return

- `li` – the li

2.2.63 ol (*function*)

ol(*itemList=[]*)
An ordered list

Key Arguments

- `itemList` – a list of items to be included in the ordered list

Return

- ol

2.2.64 p (*function*)

p(*content=''*, *lead=False*, *textAlign=False*, *color=False*, *navBar=False*, *onPhone=True*, *onTablet=True*, *onDesktop=True*, *htmlId=False*, *htmlClass=False*)

Get a Paragraph element

Key Arguments

- `content` – content of the paragraph
- `lead` – is this a lead paragraph?
- `textAlign` – how to align paragraph text [left | center | right]
- `color` – colored text for emphasis [muted | warning | info | error | success]
- `navBar` – is this <p> for a navbar?
- `onPhone` – does this container get displayed on a phone sized screen
- `onTablet` – does this container get displayed on a tablet sized screen
- `onDesktop` – does this container get displayed on a desktop sized screen

Return

- `p` – the html paragraph element

2.2.65 pageHeader (*function*)

pageHeader(*headline=''*, *tagline=''*)

Generate a pageHeader - TBS style

Key Arguments

- `headline` – the headline text
- `tagline` – the tagline text for below the headline

Return

- `pageHeader` – the pageHeader

2.2.66 ul (*function*)

ul(*itemList=[]*, *unstyled=False*, *inline=False*, *dropDownMenu=False*, *navStyle=False*, *navPull=False*, *navDirection='horizontal'*, *breadcrumb=False*, *pager=False*, *thumbnails=False*, *mediaList=False*, *htmlId=False*)

Get An unordered list – can be used for navigation, stacked tab and pill

Key Arguments

- `itemList` – a list of items to be included in the unordered list
- `unstyled` – is the list to be unstyled (first children only)
- `inline` – place all list items on a single line with inline-block and some light padding.
- `dropDownMenu` – is this ul to be used in a dropdown menu? [false | true]
- `navStyle` – set the navigation style if used for tabs & pills etc [nav | tabs | pills | list]
- `navPull` – set the alignment of the navigation links [false | left | right]
- `navDirection` – set the direction of the navigation ['default' | 'stacked']

- `breadcrumb` – display breadcrumb across muliple pages? [False | True]
- `pager` – use `<ul>` for a pager
- `thumbnails` – use the `<ul>` for a thumnail block?
- `mediaList` – use the `<ul>` for a media object list?
- `htmlId` – the html id of the ul

Return

- `ul`

2.2.67 default_fields (*function*)

default_fields()
default feilds

Return

- `fieldDict` – a dictionary of { fieldName, defaultValue }

2.3 A-Z Index

Modules

`khufu.addons`	
`khufu.buttons`	
`khufu.code`	*Generate a code section*
`khufu.commonutils`	*common tools used throughout package*
`khufu.dropdowns`	
`khufu.forms`	
`khufu.helpers`	
`khufu.images`	
`khufu.labelsAndBadges`	
`khufu.modals`	
`khufu.navigation`	
`khufu.plots`	
`khufu.scaffolding`	
`khufu.tables`	

continues on next page

Table 9 – continued from previous page

khufu.typography
khufu.urls

Classes

khufu.forms.login_form	*The worker class for the login_form module*
khufu.images.imageWell	*Framework for a bootstrap style well containing thumbnail images that can be clicked on to reveal a modal of more imformation*
khufu.images.imagingModal	*An image and modal – click on the image to present the modal of the larger image with download options*
khufu.modals.modalForm	*The worker class for the modalForm module*
khufu.scaffolding.login_page	*The worker class for the login_page module*
khufu.tables.sortable_table	*The worker class for the sortable_table module*

Functions

khufu.addons.closeIcon	*Get close icon.
khufu.addons.mediaObject	*Generate an abstract object style for building various types of components (like blog comments, Tweets, etc) that feature a left- or right-aligned image alongside textual content.*
khufu.addons.popover	*popover to provide helper text or some secondary info about an element*
khufu.addons.well	*Get well.
khufu.buttons.button	*Generate a button - TBS style*
khufu.buttons.buttonGroup	*Generate a buttonGroup - TBS style*
khufu.commonutils.getpackagepath	*getpackagepath*
khufu.dropdowns.dropdown	*get a toggleable, contextual menu for displaying lists of links.
khufu.dropdowns.dropdownLinkList	*dropdownLinkList*
khufu.forms.checkbox	*Generate a checkbox - TBS style*
khufu.forms.controlRow	*generate a form row*
khufu.forms.form	*Generate a form - TBS style*
khufu.forms.formActions	*Generate a formActions - TBS style*
khufu.forms.formInput	*Generate a form input - TBS style*
khufu.forms.horizontalFormControlGroup	*Generate a horizontal form control group (row) - TBS style*
khufu.forms.horizontalFormControlLabel	*set a horizontal form label*
khufu.forms.radio	*Generate a radio - TBS style*
khufu.forms.searchForm	*Generate a search-form - TBS style*
khufu.forms.select	*Generate a select - TBS style*
khufu.forms.textarea	*Generate a textarea - TBS style*
khufu.forms.uneditableInput	*Generate a uneditableInput - TBS style*
khufu.helpers.hide_from_device	*hide from device)*
khufu.helpers.unescape_html	*Unescape a string previously escaped with cgi.escape()*
khufu.images.image	*Create an HTML image (with ot without link).
khufu.images.thumbnail_div	*Generate a thumbnail - TBS style*

continues on next page

Table 11 – continued from previous page

khufu.images.thumbnails	*Generate a thumbnail - TBS style*
khufu.labelsAndBadges.alert	*Generate a alert - TBS style*
khufu.labelsAndBadges.badge	*Generate a badge - TBS style*
khufu.labelsAndBadges.label	*Generate a label - TBS style*
khufu.labelsAndBadges.progressBar	*Generate a progress bar - TBS style*
khufu.labelsAndBadges.stackedProgressBar	*Generate a progress bar - TBS style*
khufu.modals.modal	*generate a modal to by generated with a js event*
khufu.navigation.is_navStyle_active	*is navStyle active*
khufu.navigation.navBar	*Generate a navBar - TBS style*
khufu.navigation.nav_list	*Create an html list of navigation items from the required python list*
khufu.navigation.pagination	*Generate pagination - TBS style.
khufu.navigation.responsive_navigation_bar	*Create a twitter bootstrap responsive nav-bar component*
khufu.navigation.searchbox	*Create a Search box*
khufu.navigation.tabbableNavigation	Generate a tabbable Navigation
khufu.plots.svg	*svg*
khufu.scaffolding.body	*Generate an HTML body*
khufu.scaffolding.grid_column	*Get a column block for the Twiiter Bootstrap static layout grid.*
khufu.scaffolding.grid_row	*Create a row using the Twitter Bootstrap static layout grid.
khufu.scaffolding.head	*Generate an html head element for your webpage*
khufu.scaffolding.htmlDocument	*The doctype and html tags*
khufu.scaffolding.row_adjustable	*row adjustable*
khufu.tables.table	*Generate a table - TBS style*
khufu.tables.tableCaption	*Generate a table caption - TBS style*
khufu.tables.tbody	*Generate a table body - TBS style*
khufu.tables.td	*Generate a table data cell - TBS style*
khufu.tables.th	*Generate a table header cell - TBS style*
khufu.tables.thead	*Generate a table head - TBS style*
khufu.tables.tr	*Generate a table row - TBS style*
khufu.typography.a	*Generate an anchor - TBS style*
khufu.typography.abbr	*Get HTML5 Abbreviation*
khufu.typography.address	*Get The HTML5 address element*
khufu.typography.blockquote	*Get HTML5 Blockquote*
khufu.typography.coloredText	*Colour text a given colour*
khufu.typography.descriptionLists	*A list of definitions.*
khufu.typography.emphasizeText	*Get HTML's default emphasis tags with lightweight styles.*
khufu.typography.heroUnit	*Generate a heroUnit - TBS style*
khufu.typography.li	*Generate a li - TBS style*
khufu.typography.ol	*An ordered list*
khufu.typography.p	*Get a Paragraph element*
khufu.typography.pageHeader	*Generate a pageHeader - TBS style*
khufu.typography.ul	*Get An unordered list – can be used for navigation, stacked tab and pill*
khufu.urls.default_fields	*default feilds*

CHAPTER

THREE

RELEASE NOTES

v1.3.3 - May 10, 2022

- **FIXED** doc fixes

v1.3.2 - December 3, 2020

- **fixed** image wells not getting imported correctly

v1.3.1 - June 25, 2020

- **fixed** small Python 2 to 3 issues

v1.3.0 - May 25, 2020

- Now compatible with Python 3.*

INDEX

N

O

P

R

S

T

U

W