Package ‘shinyreforms’

October 14, 2022

Title Add Forms to your 'Shiny' App

Version 0.0.1

Author Piotr Bajger <`piotr.bajger@hotmail.com`>

Maintainer Piotr Bajger <`piotr.bajger@hotmail.com`>

Description Allows to create modular, reusable 'HTML'
forms which can be embedded in your 'shiny' app with minimal effort.
Features include conditional code execution on form submission,
automatic input validation and input tooltips.

URL `https://github.com/piotrbajger/shinyreforms`

Depends R (>= 3.0)

License GPL-3

Imports shiny (>= 1.0.0), htmltools (>= 0.2.6), R6

Suggests testthat, knitr, rmarkdown

VignetteBuilder knitr

BugReports `https://github.com/piotrbajger/shinyreforms`

Encoding UTF-8

LazyData true

RoxygenNote 7.1.0

NeedsCompilation no

Repository CRAN

Date/Publication 2020-05-12 10:10:03 UTC

R topics documented:

`addHelpText` *Adds a help icon to an input.*

Description

Internal function which adds a shinyreforms pop-up with help text to a shiny inputTag. The help text is a div which gets appended to the label for the given input.

Usage

```
addHelpText(tag, helpText, updated = FALSE)
```

Arguments

`tag`	A tag to be modified.
`helpText`	Help text to be added.
`updated`	An internal parameter which is used in recurrent calls to the function.

Value

A modified Shiny tag with a shinyreforms help icon.

Examples

```
addHelpText(
  shiny::textInput("text_input", label = "Label"),
  helpText = "Tooltip"
)
```

`addValidationSuffix` *Appends a validation suffix to a string.*

Description

Appends a validation suffix to a string.

Usage

```
addValidationSuffix(tagId)
```

Arguments

`tagId`	ID of the tag.

Value

A string ‘"tagId-shinyreforms-validation"‘.

`createHelpIcon` *Creates a shinyreforms help icon and pop-up.*

Description

Creates a shinyreforms help icon and pop-up.

Usage

```
createHelpIcon(helpText)
```

Arguments

`helpText`	A tooltip to be displayed.

Value

A shiny div with an icon and pop-up tooltip.

`getInputId` *Returns an ID of an input tag.*

Description

Returns an ID of an input tag.

Usage

```
getInputId(inputTag)
```

Arguments

`inputTag`	A shiny tag to retrieve the ID from.

Value

ID of an input tag.

`ShinyForm` *Class representing a ShinyForm form.*

Description

ShinyForm can be used to include forms in your website. Create a `ShinyForm` object anywhere in your application by defining all the inputs (possibly adding validators) and by specifying callback `onSuccess` and `onError` functions.

Details

Parameters `onSuccess` and `onError` passed to the constructor should be functions with signatures `function(self, input, output)`, where ‘self‘ will refer to the form itself, while `input` and `output` will be the usual Shiny objects.

Public fields

`id` Unique form id which can be used with Shiny input.

`elements` A list of ShinyForm input elements.

`onSuccess` A function with to be run on valid submission, see details.

`onError` A function with to be run on invalid submission, see details.

`submit` A submit Action button/link.

Methods

Public methods:

- `ShinyForm$new()`
- `ShinyForm$ui()`
- `ShinyForm$server()`
- `ShinyForm$getValue()`
- `ShinyForm$clone()`

Method `new()`: Initialises a ShinyForm.

Usage:

```
ShinyForm$new(id, submit, onSuccess, onError, ...)
```

Arguments:

`id` Unique form identifier.

`submit` Submit button label.

`onSuccess` Function to be ran on successful validation.

`onError` Function to be ran on unsuccesful validation.

`...` A list of validated Shiny inputs.

Method `ui()`: Returns the form's UI. To be used inside your App's UI.

Usage:

```
ShinyForm$ui()
```

Method `server()`: Form logic. To be inserted into your App's server function.
Will validate form upon hitting the "Submit" button and run the ‘onSuccess‘ or ‘onError‘ function depending on whether the form is valid.

Usage:

```
ShinyForm$server(input, output)
```

Arguments:

`input` Shiny input.

`output` Shiny output.

Method `getValue()`: Returns value of the input element with a given ID.

Usage:

```
ShinyForm$getValue(input, inputId)
```

Arguments:

`input` Shiny input.

`inputId` ID of the input whose value is to be returned.

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
ShinyForm$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`shinyReformsDependency` *Constructs a shinyreforms dependency.*

Description

Constructs a shinyreforms dependency.

Usage

```
shinyReformsDependency()
```

Value

A list containing the shinyreforms style dependency.

`shinyReformsPage` *Adds a shinyreforms dependency to a tag object.*

Description

Adds a shinyreforms dependency to a tag object.

Usage

```
shinyReformsPage(htmlTag)
```

Arguments

`htmlTag`	A shiny HTML tag.

Value

The input htmlTag with the shinyreforms.css dependency added.

Examples

```
if(interactive()){
shinyReformsPage(shiny::fluidPage(...))
}
```

`validatedInput` *Add validator to a Shiny input.*

Description

Use to create shiny input tags with validation. This should only be used in ShinyForm constructor.

Usage

```
validatedInput(tag, helpText = NULL, validators = c())
```

Arguments

`tag`	Tag to be modified.
`helpText`	Tooltip text. If NULL, no tooltip will be added.
`validators`	A vector of ‘Validator‘ objects.

Details

The Shiny tag receives an additional attribute ‘validators‘ which is a vector of ‘Validator‘ objects.

Value

A modified shiny input tag with attached validators and an optional tooltip div.

Examples

```
shinyreforms::validatedInput(
  shiny::textInput("text_input", label = "Username"),
  helpText = "Username must have length between 4 and 12 characters.",
  validators = c(
    shinyreforms::ValidatorMinLength(4),
    shinyreforms::ValidatorMaxLength(12)
  )
)
```

`Validator` *Class representing a Validator.*

Description

Validators are used to validate input fields in a ShinyForm. Validators are to be used with the validatedInput function. A single input field can have several validators.

Details

Package shinyreformss defines a set of commonly used pre-defined Validators. These include:

ValidatorMinLength(minLength): Will fail if string is shorter than minLength.

ValidatorMaxLength(maxLength): Will fail if string is longer than maxLength.

ValidatorNonEmpty(): Will fail if string is empty.

Public fields

`test` Function returning a boolean value which will be used to validate input.

`failMessage` Error message to display when validation fails.

Methods

Public methods:

- `Validator$new()`
- `Validator$check()`
- `Validator$clone()`

Method `new()`**:** Creates a Validator object.

Usage:

```
Validator$new(test, failMessage)
```

Arguments:

`test` A function to test the input. Should take a single value as input and return a boolean.

`failMessage` A fail message to be displayed.

Method `check()`**:** Performs a check on the input.

Usage:

```
Validator$check(value)
```

Arguments:

`value` Input value to be tested.

Returns: TRUE if the check passes, FALSE if otherwise.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
Validator$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
if(interactive()){
Validator(function(value) {
  ...
}, "Validation failed!")
}
```

`ValidatorMaxLength` *Validator enforcing maximum length.*

Description

Will return TRUE for strings longer than the maximum value.

Usage

```
ValidatorMaxLength(maxLength)
```

Arguments

`maxLength`	Maximum length of the input.

Value

A `Validator` checking that the input value does not exceed `maxLength`.

`ValidatorMinLength` *Validator requiring minimum length.*

Description

Will return TRUE for strings longer than the minimum value.

Usage

```
ValidatorMinLength(minLength)
```

Arguments

`minLength`	Minimum length of the input.

Value

A `Validator` checking that the input value is of length at least `minLength`.

`ValidatorNonEmpty` *Validator requiring non-emptiness.*

Description

The validator will return FALSE if the input is NULL, an empty vector, or an empty string ("") and FALSE otherwise.

Usage

```
ValidatorNonEmpty()
```

Value

A `Validator` to check if an input is non-empty.

`ValidatorRequired` *Validator requiring a input (e.g. checked checkbox).*

Description

Will return FALSE if the input value is FALSE (e.g. like for an unchecked textbox.)

Usage

```
ValidatorRequired()
```

Value

A validator which checks that the value is present.

Index