



# - shinyWidgets -



# Qu'est-ce que shinyWidgets ?

C'est un package **R** qui vous permet de customiser vos applications Shiny en mettant à votre disposition de nouveaux inputs.

```
install.packages("shinyWidgets")
```

## Comment fait-il?

shinyWidgets intègre des librairies JavaScripts dans Shiny :

- Bootstrap switch : <http://bootstrapswitch.com/examples.html>
- Bootstrap select : <https://silviomoreto.github.io/bootstrap-select/examples/>
- Plein d'autres trucs de Bootstrap...

Mais aussi : [sweetalert](#), [animate.css](#), [bttm.css](#),...

# À quoi cela ressemble ?

shiny

☒ Choice A

☒ Option 1 ☐ Option 2 ☐ Option 3 ☐ Option 4

Nationalité :

Royaume-Uni

Allemagne

Etats-Unis

Belgique

Chine

Espagne

Pays-Bas

Mexique

shinyWidgets

Success ☒

ON

☐ Option 1 ☒ Option 2 ☐ Option 3 ☐ Option 4


☐ Option 1 ☒ Option 2 ☐ Option 3 ☐ Option 4


Nationalité :


Choisissez une nationalité


Tout sélectionner


Tout désélectionner


 Royaume-Uni


 Allemagne


 Etats-Unis

 Belgique

 Chine

 Espagne

 Pays-Bas

 Mexique

# Comment l'utiliser ?

Comme les inputs de Shiny :

```
# ui
materialSwitch(
  inputId = "id", label = "Label", status = "primary"
)

# server
input$id

updateMaterialSwitch(session = session, inputId = "id", value = FALSE)
```

Exceptions :

- Dropdown : uniquement UI
- ProgressBars : définition dans l'UI, update dans le server
- sweetAlert : initialisation dans l'UI, utilisation dans le server

**Plus d'exemples :**

**shinyWidgets::shinyWidgetsGallery()**

# Comment cela fonctionne ?

1. Intégration de bibliothèques JavaScript et CSS, à inclure une fois dans le head de l'application.
2. Définition de nouveaux inputs controls, les fonctions qui seront utilisées dans le UI.
3. Bindings JavaScript pour faire dialoguer client – server, principalement dans le sens client vers server, mais aussi dans le sens server vers client (pour les update d'inputs)

# Intégration de librairies JavaScript et CSS

Il faut utiliser un répertoire `www` dans lequel on puisse sourcer les librairies, il doit se trouver dans le répertoire `inst/` du package.

Déclaration de l'emplacement du répertoire :

```
addResourcePath(  
  prefix = 'shinyWidgets',  
  directoryPath = system.file('www', package = 'shinyWidgets')  
)
```

Déclaration des dépendances :

```
htmltools::htmlDependency(  
  name = "selectPicker",  
  version = "1.11.0",  
  src = c(href="shinyWidgets/selectPicker"), # inst/www/selectPicker  
  script = "js/bootstrap-select.min.js",  
  stylesheet = "css/bootstrap-select.min.css"  
)
```

# Définition de nouveaux inputs controls

Génération de code HTML depuis R :

```
tags$select(  
  class = "form-control",  
  lapply(  
    X = seq_len(5),  
    FUN = tags$option  
  )  
)
```



```
<select class="form-control">  
  <option>1</option>  
  <option>2</option>  
  <option>3</option>  
  <option>4</option>  
  <option>5</option>  
</select>
```

## Template type customInput.R

```
customInput <- function(inputId, ...) {  
  mytag <- tagList(  
    ...  
  )  
  htmltools::attachDependencies(mytag, mydep)  
}
```



# Exemple : searchInput

Enter your text

A placeholder



```
searchInput <- function(inputId, label = NULL, value = "", placeholder = NULL,
                        btnSearch = NULL, btnReset = NULL, width = NULL) {
  if (!is.null(btnSearch)) {
    btnSearch <- tags$button(
      class="btn btn-default", id = paste0("search", inputId), type="button", btnSearch
    )
  }
  if (!is.null(btnReset)) {
    btnReset <- tags$button(
      class="btn btn-default", id = paste0("reset", inputId), type="button", btnReset
    )
  }
  searchTag <- div(
    class="form-group shiny-input-container",
    style = if (!is.null(width))
      paste0("width: ", validateCssUnit(width), ";"),
    if (!is.null(label)) tags$label(label, `for` = inputId),
    div(
      id = inputId, class = "input-group search-text",
      tags$input(id = paste0("se", inputId),
        style = "border-radius: 0.5em 0 0 0.5em !important;",
        type = "text", class = "form-control", value = value,
        placeholder = placeholder),
      tags$span(class="input-group-btn", btnReset, btnSearch)
    )
  )
  attachShinyWidgetsDep(searchTag)
}
```



# Bindings JavaScript

Pour pouvoir récupérer la valeur de l'input côté server, il y a deux solutions :

- Utiliser les bindings déjà existants de Shiny (checkbox, radio buttons, ...)
- En créer de nouveaux

## Template type binding.js

```
// création  
var binding = new Shiny.InputBinding();  
  
// définition  
$.extend(binding, {  
  
    ...  
  
});  
  
// enregistrement  
Shiny.inputBindings.register(binding);
```

# Bindings JavaScript

## Principal methods

```
find: function(scope) {  
  ...  
},  
  
getValue: function(el) {  
  ...  
},  
  
setValue: function(el, value) {  
  ...  
},  
  
subscribe: function(el, callback) {  
  ...  
},  
  
receiveMessage: function(el, data) {  
  ...  
}
```



- Comment trouver l'input dans la page HTML ?
- Comment récupérer la valeur de l'input ?
- Comment définir la valeur de l'input ?
- Comment savoir que la valeur de l'input a été modifiée ?
- Que faire lorsque l'on veut updater l'input depuis le server ?

# Exemple : searchInput

- find : on cherche la classe « seach-text » qui permet d'identifier l'input barre de recherche :

```
find: function(scope) {  
    return $(scope).find('.search-text');  
}
```

- subscribe : màj de l'input quand on tape sur Entrée ou clique que le bouton :

```
subscribe: function(el, callback) {  
    $('#se' + el.id).on(  
        'keyup.searchInputBinding input.searchInputBinding',  
        function(event) {  
            if(event.keyCode == 13) { // if enter  
                callback();  
            }  
        });  
    $('#search' + el.id).on('click', function(event) { // on click  
        callback();  
    });  
    $('#reset' + el.id).on('click', function(event) { // on click  
        $('#se' + el.id).val('');  
        callback();  
    });  
}
```

# Définition d'un input sans bindings



On peut également créer un input sans écrire de bindings, par exemple pour créer des évènements dans un htmlwidget. On utilise la fonction :

```
Shiny.onInputChange(<id>, <value>)
```

Id de l'input

Valeur prise par l'input, peut être un élément atomique ou un JSON

## Démo

# Envoi de messages au client depuis le server



## UI (User Interface)

```
Shiny.addCustomMessageHandler(  
  '<id>', function(data) {  
    ...  
  });
```

## server

```
session$sendCustomMessage(  
  type = "<id>",  
  message = list("<value>")  
)
```

Côté serveur, le message peut être un élément atomique ou une liste.  
S'il s'agit d'une liste, « data » est un JSON côté UI, on accède à ses éléments avec un point « . » au lieu d'un « \$ ».



# -MERCI-

Pour me mettre des stars :

<https://github.com/dreamRs/shinyWidgets>

Pour me contacter :

Mail : [victor.perrier@dreamrs.fr](mailto:victor.perrier@dreamrs.fr)

Twitter : [@pvictorr](https://twitter.com/pvictorr)

Web : <https://www.dreamrs.fr/>