Documentation for On-Screen keyboard

Prerequisites

```
npm i angular-simple-keyboard

npm i simple-keyboard-layouts
```

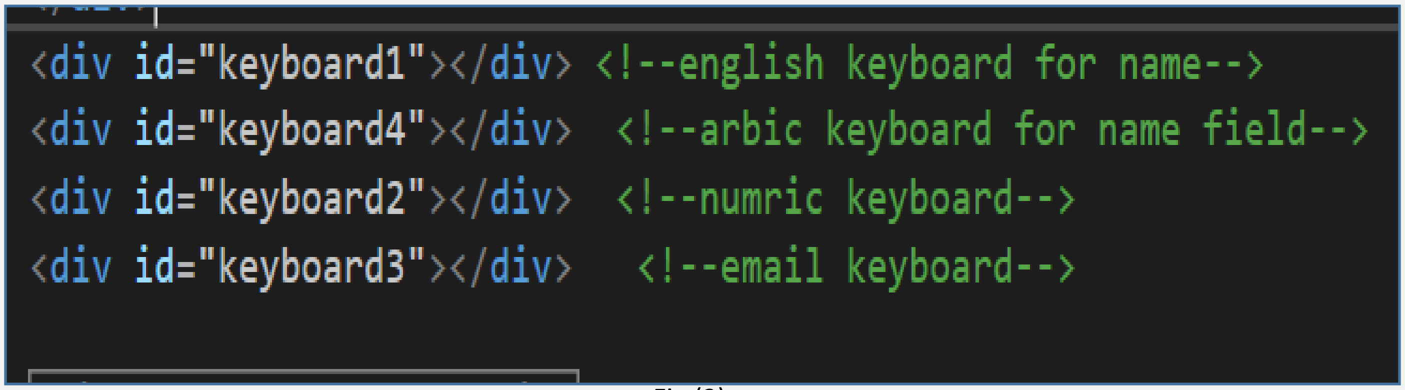
Background about the Next Code

The below code is used to show a popup onscreen keyboard to help the user to enter input without an external keyboard each keyboard instance handle one input field and its form validations also that configuration makes the external keyboard typing on that input doesn't hold the value.
There are some extra confirmations on the below keyboard (tab button, enter button)
To override on its the basic functions.
I tried to simplify the code into separatable functions.

Structure

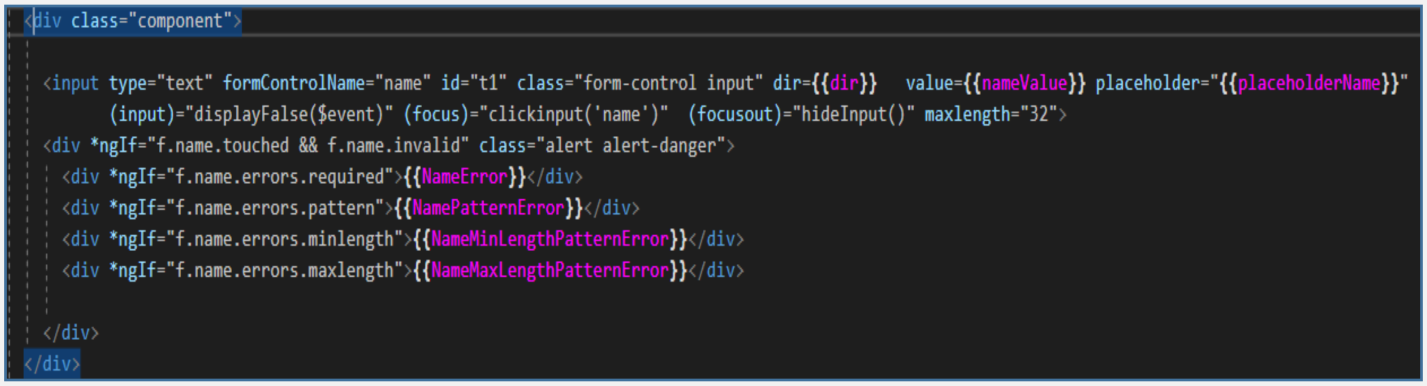
To set up the keyboard on the project we add multiple code lines in each file.
HTML File:

Fig (1)



```
<div id="keyboard1"></div> <!--english keyboard for name-->
<div id="keyboard4"></div>  <!--arbic keyboard for name field-->
<div id="keyboard2"></div>  <!--numric keyboard-->
<div id="keyboard3"></div>   <!--email keyboard-->
```

Fig (2)



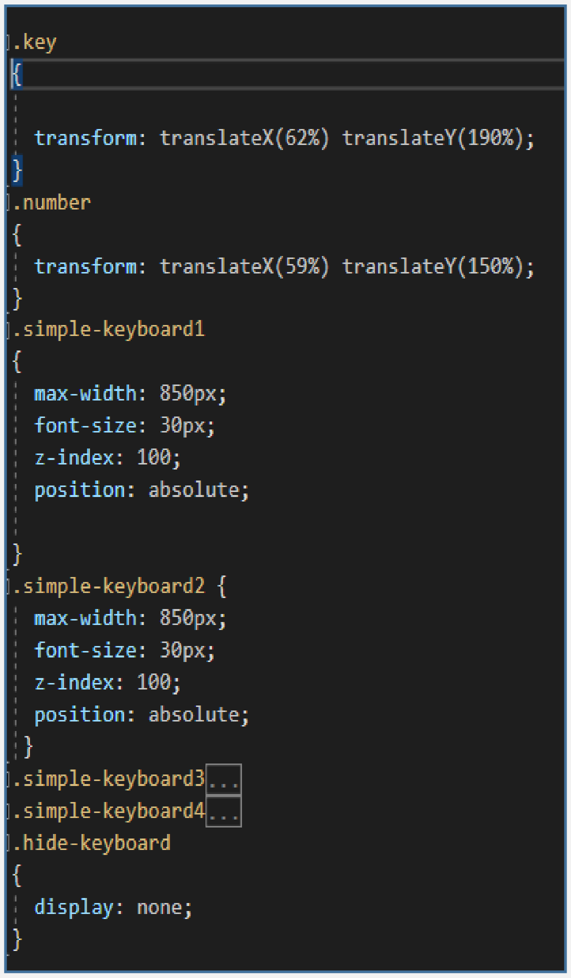
```
<div class="component">

  <input type="text" formControlName="name" id="t1" class="form-control input" dir={{dir}}   value={{nameValue}} placeholder="{{placeholderName}}"
        (input)="displayFalse($event)" (focus)="clickinput('name')"  (focusout)="hideInput()" maxlength="32">
  <div *ngIf="f.name.touched && f.name.invalid" class="alert alert-danger">
    <div *ngIf="f.name.errors.required">{{NameError}}</div>
    <div *ngIf="f.name.errors.pattern">{{NamePatternError}}</div>
    <div *ngIf="f.name.errors.minlength">{{NameMinLengthPatternError}}</div>
    <div *ngIf="f.name.errors.maxlength">{{NameMaxLengthPatternError}}</div>

  </div>
</div>
```

In figure (1), the keyboard Placeholder added with different ids for each mode and language needed
It preferred to be at the end of the file.

In figure (2), is binding the keyboard functions on the input tag in HTML
Here we will know each parameter passed in the input tag

- ❖ Value => will bind the keyboard value of each button clicked to be shown in the area.
- ❖ Focus=> shows the specific keyboard assigned to the input previously and appears in the selector passed to the function as a parameter.
- ❖ Focus out=>event serves for hiding the keyboard when it's in use and also fixes the bug that appears when the keyboard loses the bond between keyboard and input where you can't delete what you typed anymore.
- ❖ Input=> listen for the change that happens in the keyboard and sync it with the input value. (will be demonstrated more later in typescript file code)
- ❖ Id=> used in the input function above.

Fig (3)



```css
.key
{

  transform: translateX(62%) translateY(190%);
}
.number
{
  transform: translateX(59%) translateY(150%);
}
.simple-keyboard1
{
  max-width: 850px;
  font-size: 30px;
  z-index: 100;
  position: absolute;

}
.simple-keyboard2 {
  max-width: 850px;
  font-size: 30px;
  z-index: 100;
  position: absolute;
 }
.simple-keyboard3...
.simple-keyboard4...
.hide-keyboard
{
  display: none;
}
```

CSS File

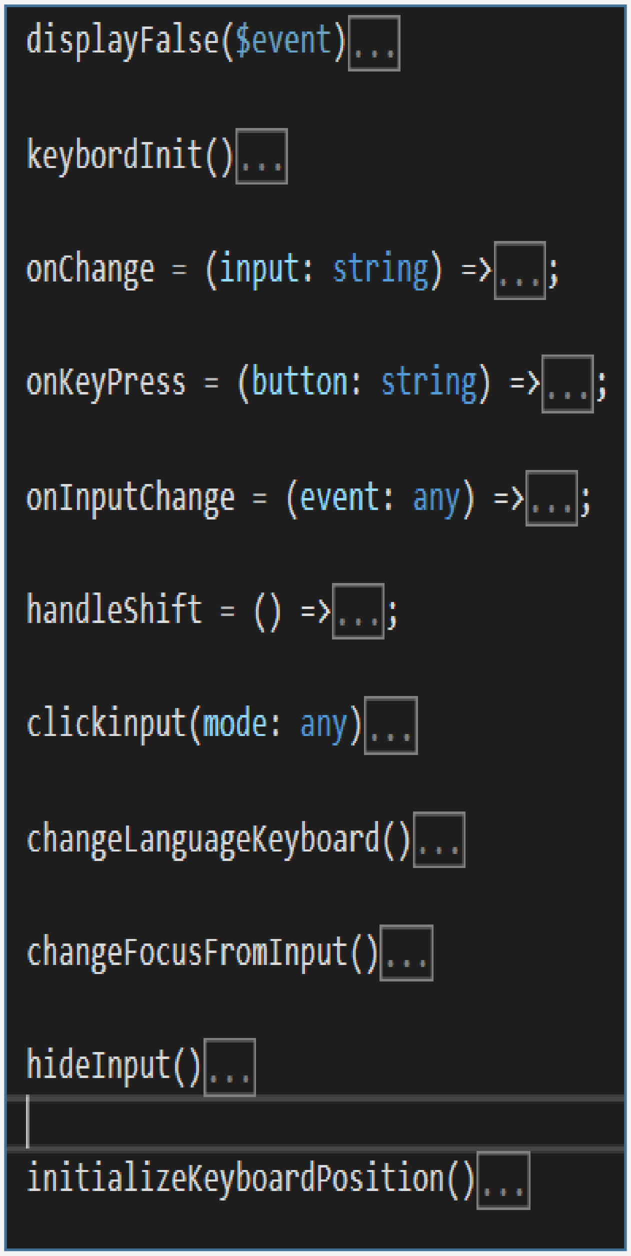
All CSS classes in figure 3 can be customized for the project Needs,

- ❖ Classes (key, number) =>is just for positing the keyboard in the screen keys for standard keyboard layout, a number for numbers keyboard layout.
- ❖ Simple-keyboard 'n'=>acts as a placeholder for the keyboard layout and the (1,2, 3,..) is a custom style for each keyboard bonded with each input.
- ❖ Hide-keyboards =>hides the specific keyboard layout where it isn't in use.

The incoming screenshots for the actual functions for the keyboard, will be divided into multiple sections.

❖ Section 1 (Function declarations)

Function name	Description
Display False	This function calls (onInputChange) and pass the event object to get the value.
Keyboard Init	Initializes all keyboard instances, hides them, and initializes their positions.
On Change	Callback called on keyboard configuration get the mapped of keypress and converts it to the string value.
On Keypress	Callback called on keyboard configuration get the key pressed and decide if the key is special (tab, enter) one or not.
On Input Change	Gets the event element and assign the character pressed to the input value.
Handle Shift	Changes the default layout of the keyboard to shift layout
Click input	Switching the keyboards according to selector which changes according to which input is selected and active.
Change Language Keyboard	Special function to switch the keyboard layout from English to Arabic simulates the (win+space) function of windows
Change focus From Input	Special function for tab button to switch focus from input to the next input.
Hide Input	Hides the keyboard when it loses focus from the input.
initialize Keyboard Position	Initializes the keyboards layouts positioning on the screen.



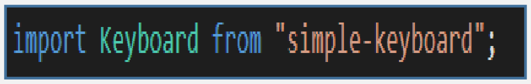
```
displayFalse($event)...

keybordInit()...

onChange = (input: string) =>...;

onKeyPress = (button: string) =>...;

onInputChange = (event: any) =>...;

handleShift = () =>...;

clickinput(mode: any)...

changeLanguageKeyboard()...

changeFocusFromInput()...

hideInput()...

initializeKeyboardPosition()...
```

❖ Requirements and initializations needed for the functions

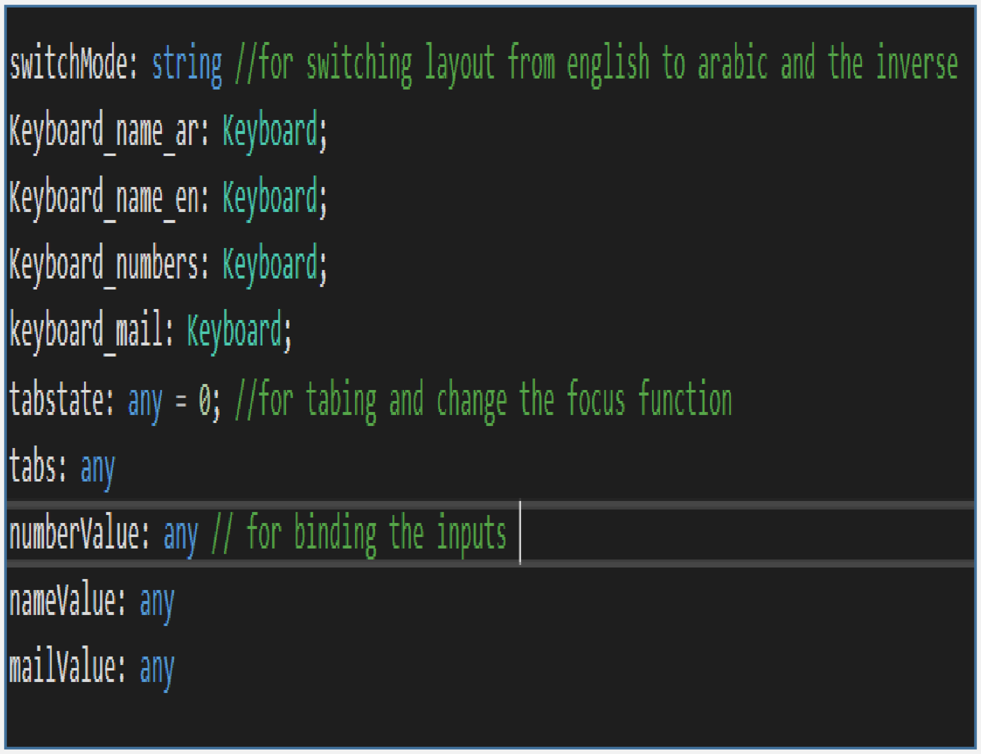
the 3 figures are initializations used in the functions
also, custom layout initializations for the keyboard and appending the values that binded into the form validations

(****Important**)

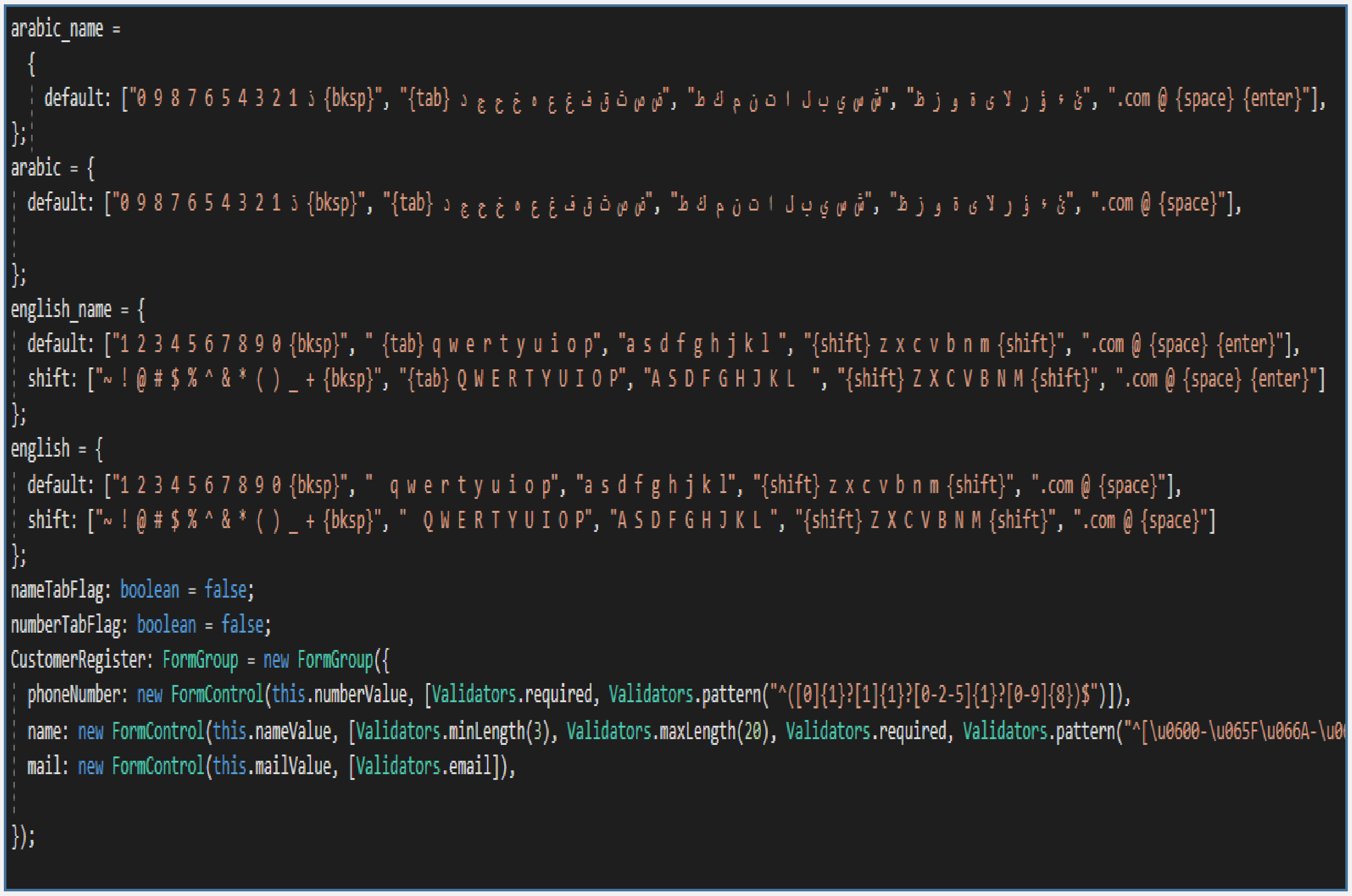
At **ngOnInit ()** function we call **keybordInit();** and get the local language (‘ar’, ‘en’) and assign it to **switchMode**.



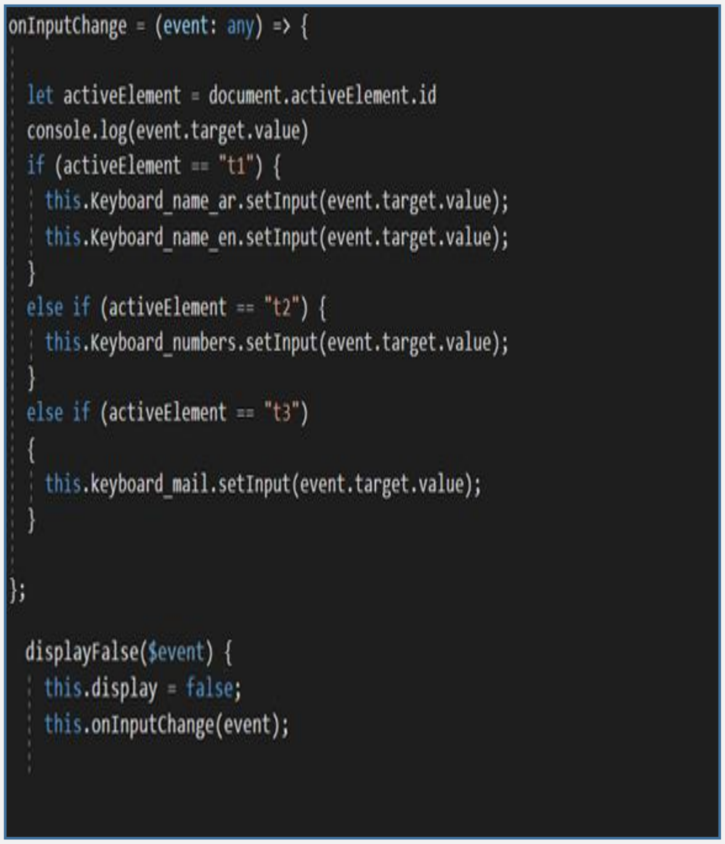
```
import Keyboard from "simple-keyboard";
```



```
switchMode: string //for switching layout from english to arabic and the inverse
Keyboard_name_ar: Keyboard;
Keyboard_name_en: Keyboard;
Keyboard_numbers: Keyboard;
keyboard_mail: Keyboard;
tabstate: any = 0; //for tabing and change the focus function
tabs: any
numberValue: any // for binding the inputs
nameValue: any
mailValue: any
```

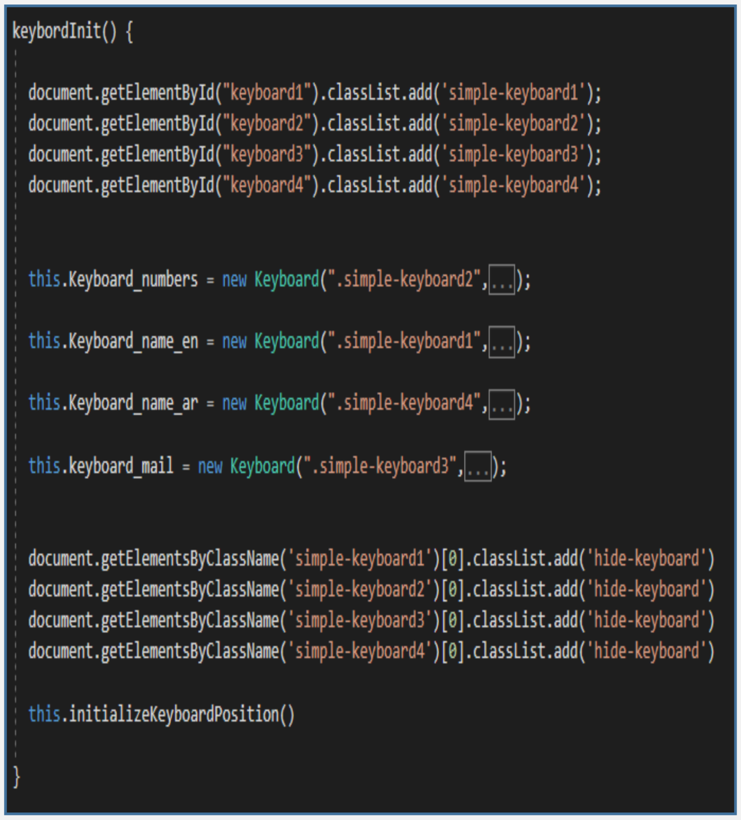


```
arabic_name =
  {
   default: ["ذ 1 2 3 4 5 6 7 8 9 0 {bksp}", "{tab} ض ص ث ق ف غ ع ه خ ح ج د", "ش س ي ب ل ا ت ن م ك ط", "ئ ء ؤ ر لا ى ة و ز ظ", ".com @ {space} {enter}"],
};
arabic = {
  default: ["ذ 1 2 3 4 5 6 7 8 9 0 {bksp}", "{tab} ض ص ث ق ف غ ع ه خ ح ج د", "ش س ي ب ل ا ت ن م ك ط", "ئ ء ؤ ر لا ى ة و ز ظ", ".com @ {space}"],


};
english_name = {
  default: ["1 2 3 4 5 6 7 8 9 0 {bksp}", " {tab} q w e r t y u i o p", "a s d f g h j k l ", "{shift} z x c v b n m {shift}", ".com @ {space} {enter}"],
  shift: ["~ ! @ # $ % ^ & * ( ) _ + {bksp}", "{tab} Q W E R T Y U I O P", "A S D F G H J K L  ", "{shift} Z X C V B N M {shift}", ".com @ {space} {enter}"]
};
english = {
  default: ["1 2 3 4 5 6 7 8 9 0 {bksp}", "  q w e r t y u i o p", "a s d f g h j k l", "{shift} z x c v b n m {shift}", ".com @ {space}"],
  shift: ["~ ! @ # $ % ^ & * ( ) _ + {bksp}", "  Q W E R T Y U I O P", "A S D F G H J K L ", "{shift} Z X C V B N M {shift}", ".com @ {space}"]
};
nameTabFlag: boolean = false;
numberTabFlag: boolean = false;
CustomerRegister: FormGroup = new FormGroup({
  phoneNumber: new FormControl(this.numberValue, [Validators.required, Validators.pattern("^([0]{1}?[1]{1}?[0-2-5]{1}?[0-9]{8})$")]),
  name: new FormControl(this.nameValue, [Validators.minLength(3), Validators.maxLength(20), Validators.required, Validators.pattern("^[\u0600-\u065F\u066A-\u0
  mail: new FormControl(this.mailValue, [Validators.email]),

});
```



```
onInputChange = (event: any) => {

  let activeElement = document.activeElement.id
  console.log(event.target.value)
  if (activeElement == "t1") {
    this.Keyboard_name_ar.setInput(event.target.value);
    this.Keyboard_name_en.setInput(event.target.value);
  }
  else if (activeElement == "t2") {
    this.Keyboard_numbers.setInput(event.target.value);
  }
  else if (activeElement == "t3")
  {
    this.keyboard_mail.setInput(event.target.value);
  }

};

  displayFalse($event) {
    this.display = false;
    this.onInputChange(event);
```

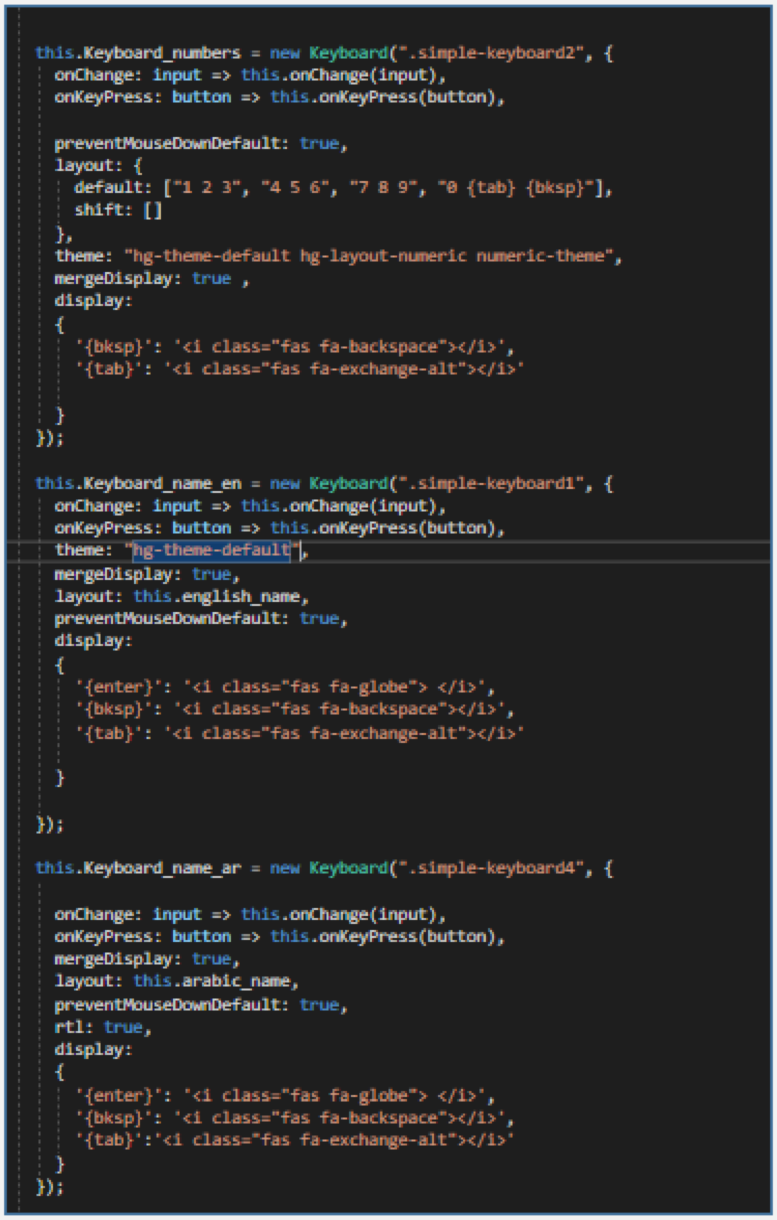
displayFalse hides the form validation error and call inputchange function which gets the active input and assigns the value clicked from the keyboard instances.
(setinput)=> act as bind that value and input
To be able to delete the value or add on it,
Its built-in function.
Also creates a bond between each input and its Keyboard (each keyboard must be bonded with Single input only).



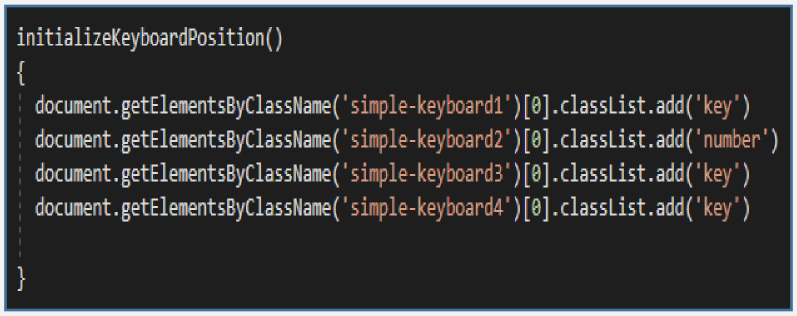
```
keybordInit() {

  document.getElementById("keyboard1").classList.add('simple-keyboard1');
  document.getElementById("keyboard2").classList.add('simple-keyboard2');
  document.getElementById("keyboard3").classList.add('simple-keyboard3');
  document.getElementById("keyboard4").classList.add('simple-keyboard4');


  this.Keyboard_numbers = new Keyboard(".simple-keyboard2",...);

  this.Keyboard_name_en = new Keyboard(".simple-keyboard1",...);

  this.Keyboard_name_ar = new Keyboard(".simple-keyboard4",...);

  this.keyboard_mail = new Keyboard(".simple-keyboard3",...);


  document.getElementsByClassName('simple-keyboard1')[0].classList.add('hide-keyboard')
  document.getElementsByClassName('simple-keyboard2')[0].classList.add('hide-keyboard')
  document.getElementsByClassName('simple-keyboard3')[0].classList.add('hide-keyboard')
  document.getElementsByClassName('simple-keyboard4')[0].classList.add('hide-keyboard')

  this.initializeKeyboardPosition()

}
```

KeyboardInit() is the core function for initialize all Instances of the keyboards

The attached screenshot is the keyboard configuration set where you bind the (**on change, on keypress**).
PreventMouseDownDefault => used to control the keyboard focus and focus out event listener.
Layout => passing the custom layout.
Display => change the (backspace, tab) into custom style to render as icons.

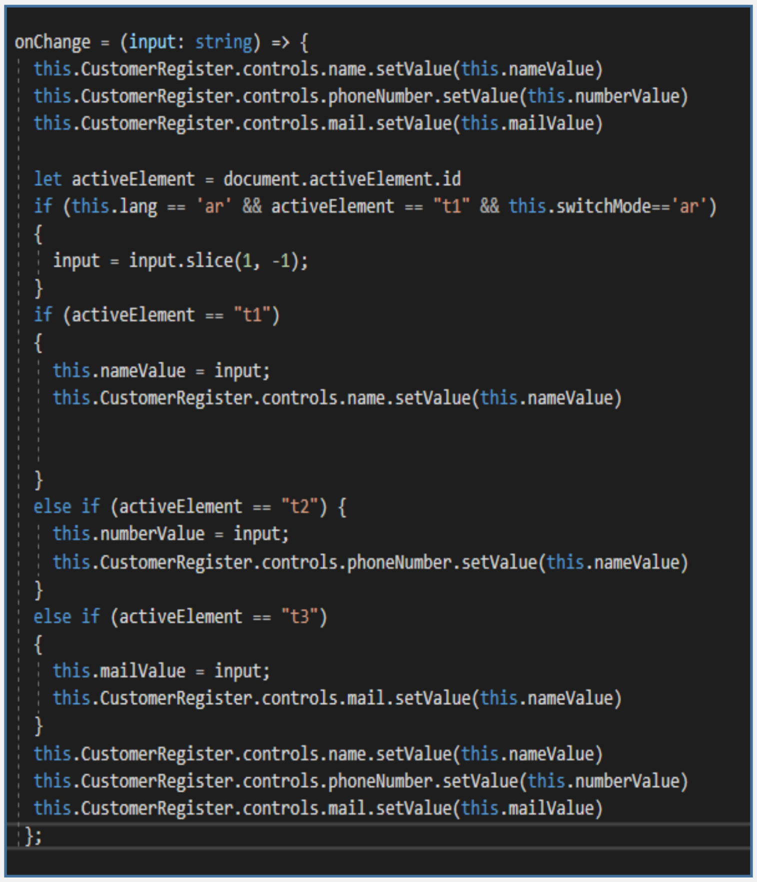


```
this.Keyboard_numbers = new Keyboard(".simple-keyboard2", {
  onChange: input => this.onChange(input),
  onKeyPress: button => this.onKeyPress(button),

  preventMouseDownDefault: true,
  layout: {
    default: ["1 2 3", "4 5 6", "7 8 9", "0 {tab} {bksp}"],
    shift: []
  },
  theme: "hg-theme-default hg-layout-numeric numeric-theme",
  mergeDisplay: true ,
  display:
  {
    '{bksp}': '<i class="fas fa-backspace"></i>',
    '{tab}': '<i class="fas fa-exchange-alt"></i>'

  }
});

this.Keyboard_name_en = new Keyboard(".simple-keyboard1", {
  onChange: input => this.onChange(input),
  onKeyPress: button => this.onKeyPress(button),
  theme: "hg-theme-default",
  mergeDisplay: true,
  layout: this.english_name,
  preventMouseDownDefault: true,
  display:
  {
    '{enter}': '<i class="fas fa-globe"> </i>',
    '{bksp}': '<i class="fas fa-backspace"></i>',
    '{tab}': '<i class="fas fa-exchange-alt"></i>'

  }

});

this.Keyboard_name_ar = new Keyboard(".simple-keyboard4", {

  onChange: input => this.onChange(input),
  onKeyPress: button => this.onKeyPress(button),
  mergeDisplay: true,
  layout: this.arabic_name,
  preventMouseDownDefault: true,
  rtl: true,
  display:
  {
    '{enter}': '<i class="fas fa-globe"> </i>',
    '{bksp}': '<i class="fas fa-backspace"></i>',
    '{tab}':'<i class="fas fa-exchange-alt"></i>'
  }
});
```



```
initializeKeyboardPosition()
{
  document.getElementsByClassName('simple-keyboard1')[0].classList.add('key')
  document.getElementsByClassName('simple-keyboard2')[0].classList.add('number')
  document.getElementsByClassName('simple-keyboard3')[0].classList.add('key')
  document.getElementsByClassName('simple-keyboard4')[0].classList.add('key')

}
```

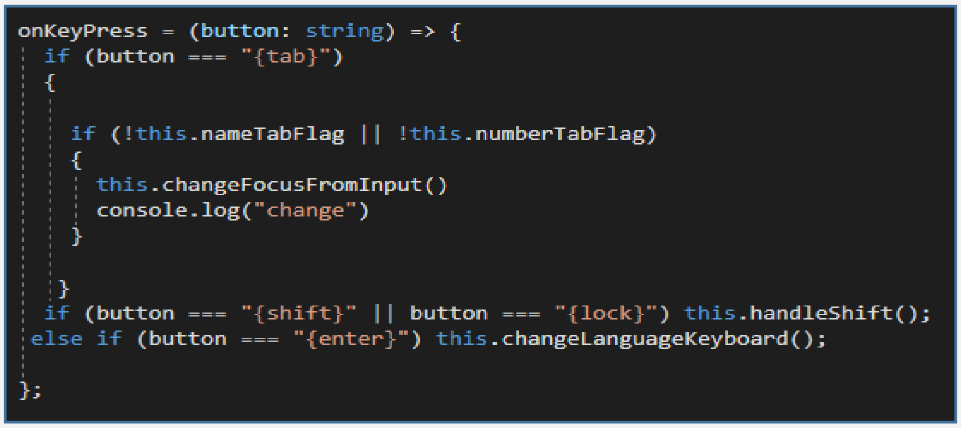
initializeKeyboardPosition=> used for set every keyboard instance in its position.

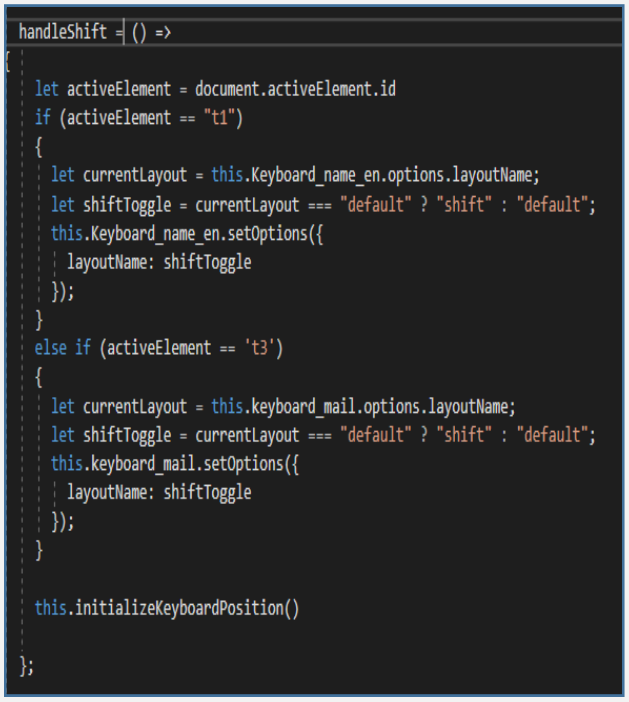


```
onChange = (input: string) => {
  this.CustomerRegister.controls.name.setValue(this.nameValue)
  this.CustomerRegister.controls.phoneNumber.setValue(this.numberValue)
  this.CustomerRegister.controls.mail.setValue(this.mailValue)

  let activeElement = document.activeElement.id
  if (this.lang == 'ar' && activeElement == "t1" && this.switchMode=='ar')
  {
    input = input.slice(1, -1);
  }
  if (activeElement == "t1")
  {
    this.nameValue = input;
    this.CustomerRegister.controls.name.setValue(this.nameValue)


  }
  else if (activeElement == "t2") {
    this.numberValue = input;
    this.CustomerRegister.controls.phoneNumber.setValue(this.nameValue)
  }
  else if (activeElement == "t3")
  {
    this.mailValue = input;
    this.CustomerRegister.controls.mail.setValue(this.nameValue)
  }
  this.CustomerRegister.controls.name.setValue(this.nameValue)
  this.CustomerRegister.controls.phoneNumber.setValue(this.numberValue)
  this.CustomerRegister.controls.mail.setValue(this.mailValue)
};
```

onChange=> is used to sync the values come from the keyboard to each form validator cross ponding to the keyboard
assign each value came from key hit to the input value
and for bug handling in the Arabic layout, the string comes from the input have space at the start and the end of the string.

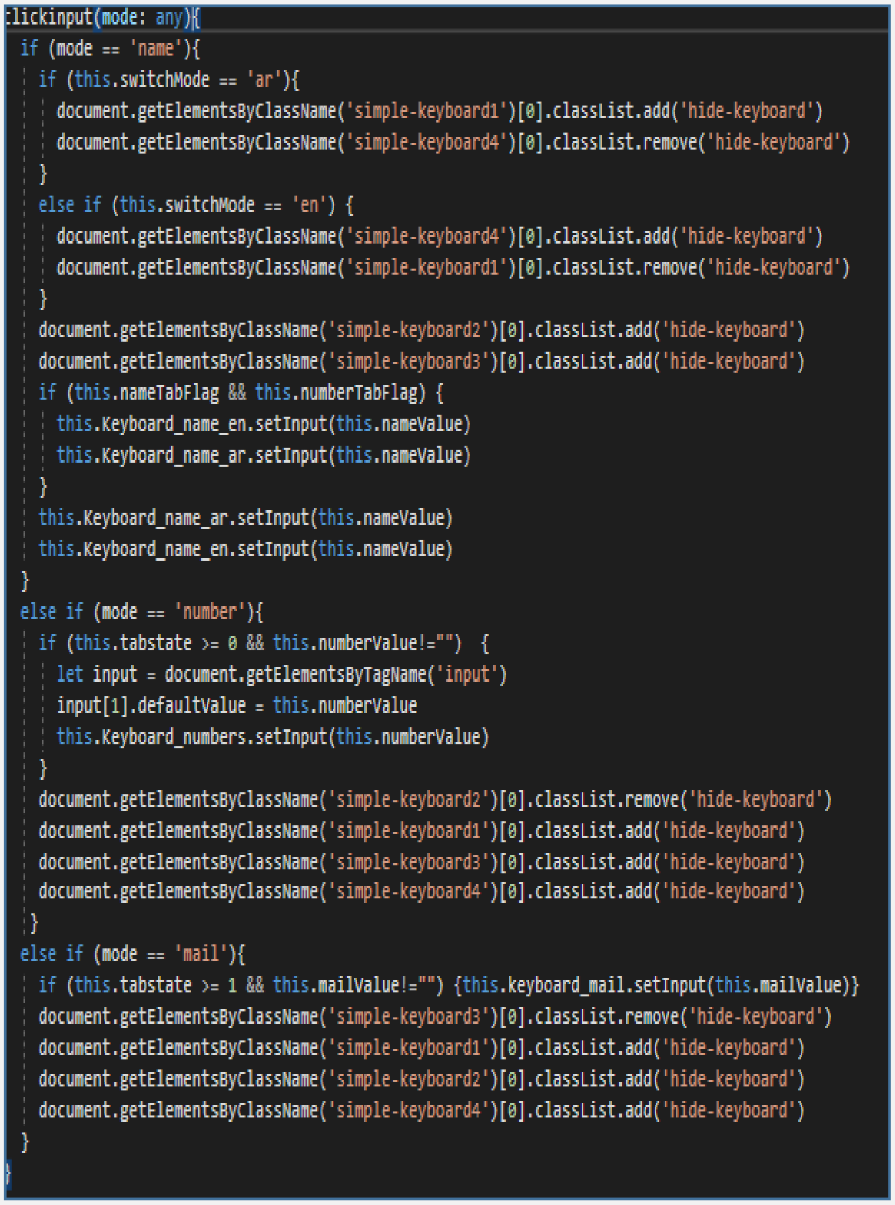
onKeyPress=> is redirecting to other functions when it is found the special button is pressed.

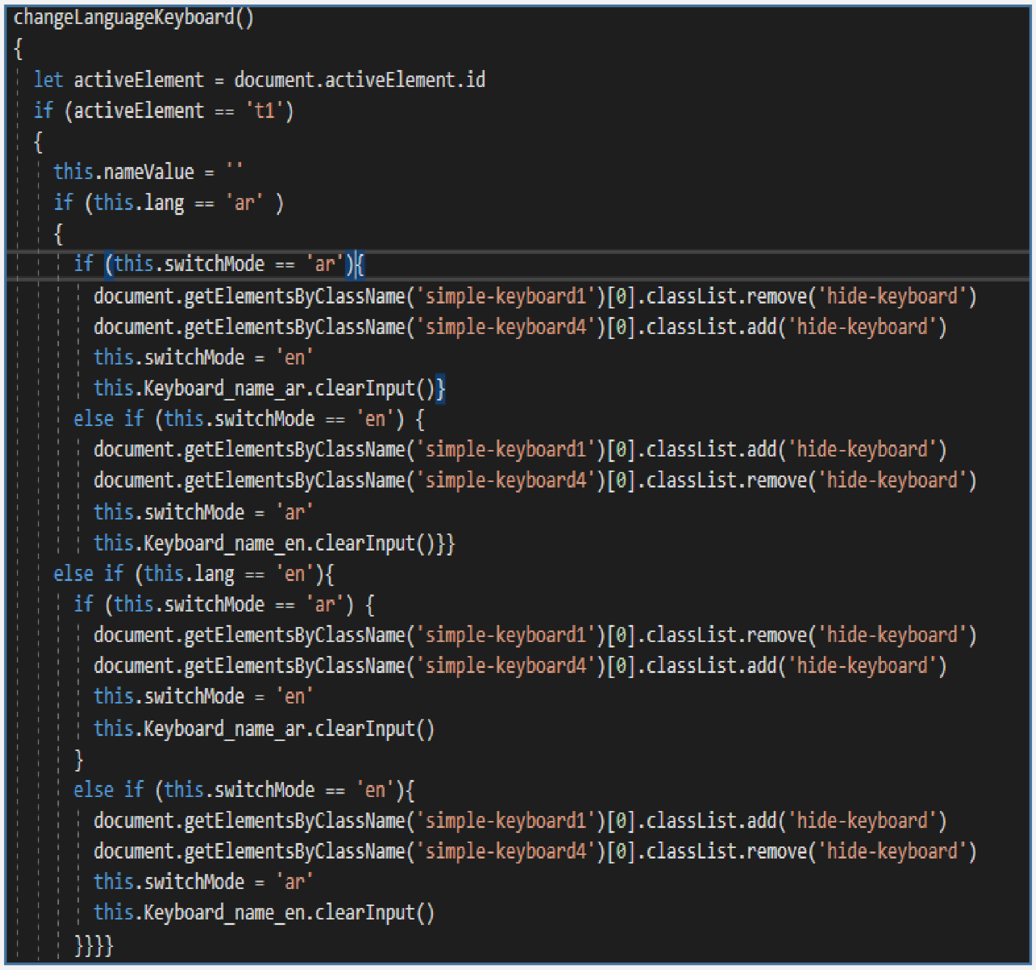

```
onKeyPress = (button: string) => {
  if (button === "{tab}")
  {

    if (!this.nameTabFlag || !this.numberTabFlag)
    {
      this.changeFocusFromInput()
      console.log("change")
    }

   }
  if (button === "{shift}" || button === "{lock}") this.handleShift();
 else if (button === "{enter}") this.changeLanguageKeyboard();

};
```


```
handleShift = () =>
{
  let activeElement = document.activeElement.id
  if (activeElement == "t1")
  {
    let currentLayout = this.Keyboard_name_en.options.layoutName;
    let shiftToggle = currentLayout === "default" ? "shift" : "default";
    this.Keyboard_name_en.setOptions({
      layoutName: shiftToggle
    });
  }
  else if (activeElement == 't3')
  {
    let currentLayout = this.keyboard_mail.options.layoutName;
    let shiftToggle = currentLayout === "default" ? "shift" : "default";
    this.keyboard_mail.setOptions({
      layoutName: shiftToggle
    });
  }

  this.initializeKeyboardPosition()

};
```

handleShift=> is switching the default layout to shift the layout of the active keyboard.

clickInput=> keep track of which keyboard is active and which layout of Arabic or English keyboard for specific input is active.

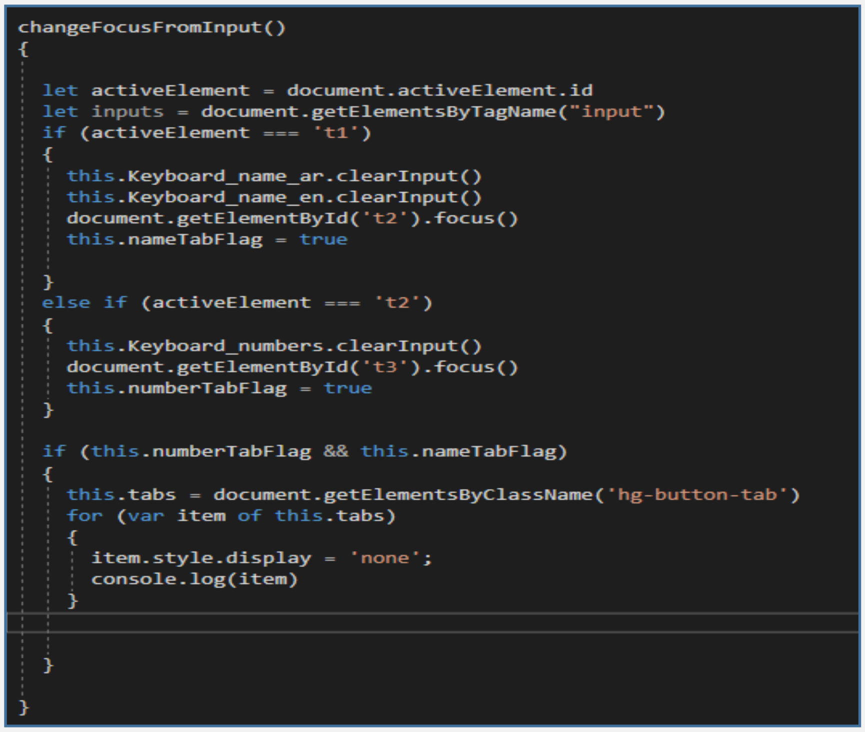

```
:lickinput(mode: any){
  if (mode == 'name'){
    if (this.switchMode == 'ar'){
      document.getElementsByClassName('simple-keyboard1')[0].classList.add('hide-keyboard')
      document.getElementsByClassName('simple-keyboard4')[0].classList.remove('hide-keyboard')
    }
    else if (this.switchMode == 'en') {
      document.getElementsByClassName('simple-keyboard4')[0].classList.add('hide-keyboard')
      document.getElementsByClassName('simple-keyboard1')[0].classList.remove('hide-keyboard')
    }
    document.getElementsByClassName('simple-keyboard2')[0].classList.add('hide-keyboard')
    document.getElementsByClassName('simple-keyboard3')[0].classList.add('hide-keyboard')
    if (this.nameTabFlag && this.numberTabFlag) {
      this.Keyboard_name_en.setInput(this.nameValue)
      this.Keyboard_name_ar.setInput(this.nameValue)
    }
    this.Keyboard_name_ar.setInput(this.nameValue)
    this.Keyboard_name_en.setInput(this.nameValue)
  }
  else if (mode == 'number'){
    if (this.tabstate >= 0 && this.numberValue!="")  {
      let input = document.getElementsByTagName('input')
      input[1].defaultValue = this.numberValue
      this.Keyboard_numbers.setInput(this.numberValue)
    }
    document.getElementsByClassName('simple-keyboard2')[0].classList.remove('hide-keyboard')
    document.getElementsByClassName('simple-keyboard1')[0].classList.add('hide-keyboard')
    document.getElementsByClassName('simple-keyboard3')[0].classList.add('hide-keyboard')
    document.getElementsByClassName('simple-keyboard4')[0].classList.add('hide-keyboard')
   }
  else if (mode == 'mail'){
    if (this.tabstate >= 1 && this.mailValue!="") {this.keyboard_mail.setInput(this.mailValue)}
    document.getElementsByClassName('simple-keyboard3')[0].classList.remove('hide-keyboard')
    document.getElementsByClassName('simple-keyboard1')[0].classList.add('hide-keyboard')
    document.getElementsByClassName('simple-keyboard2')[0].classList.add('hide-keyboard')
    document.getElementsByClassName('simple-keyboard4')[0].classList.add('hide-keyboard')
  }
}
```



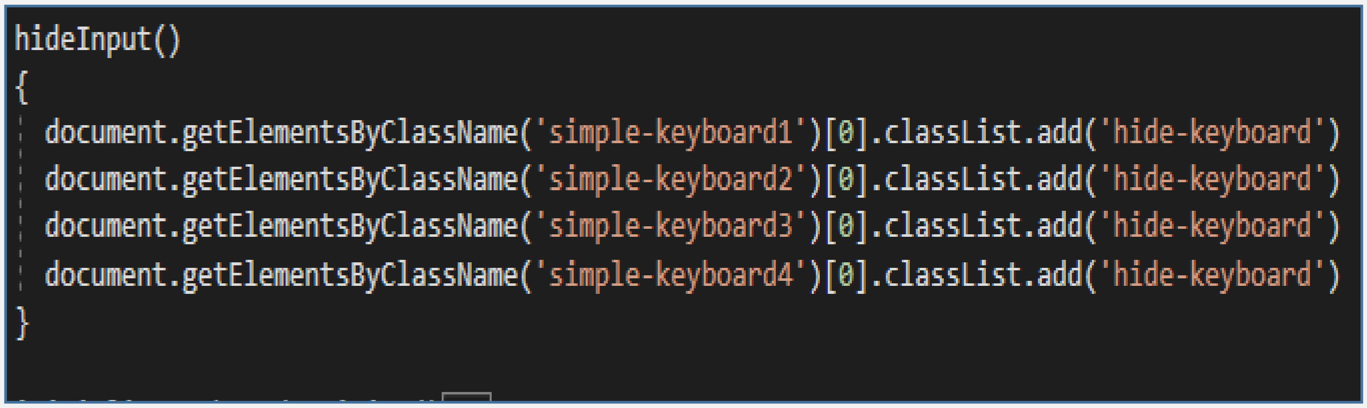
```
changeLanguageKeyboard()
{
  let activeElement = document.activeElement.id
  if (activeElement == 't1')
  {
    this.nameValue = ''
    if (this.lang == 'ar' )
    {
      if (this.switchMode == 'ar'){
        document.getElementsByClassName('simple-keyboard1')[0].classList.remove('hide-keyboard')
        document.getElementsByClassName('simple-keyboard4')[0].classList.add('hide-keyboard')
        this.switchMode = 'en'
        this.Keyboard_name_ar.clearInput()}
      else if (this.switchMode == 'en') {
        document.getElementsByClassName('simple-keyboard1')[0].classList.add('hide-keyboard')
        document.getElementsByClassName('simple-keyboard4')[0].classList.remove('hide-keyboard')
        this.switchMode = 'ar'
        this.Keyboard_name_en.clearInput()}}
    else if (this.lang == 'en'){
      if (this.switchMode == 'ar') {
        document.getElementsByClassName('simple-keyboard1')[0].classList.remove('hide-keyboard')
        document.getElementsByClassName('simple-keyboard4')[0].classList.add('hide-keyboard')
        this.switchMode = 'en'
        this.Keyboard_name_ar.clearInput()
      }
      else if (this.switchMode == 'en'){
        document.getElementsByClassName('simple-keyboard1')[0].classList.add('hide-keyboard')
        document.getElementsByClassName('simple-keyboard4')[0].classList.remove('hide-keyboard')
        this.switchMode = 'ar'
        this.Keyboard_name_en.clearInput()
      }}}}
```

changeLanguageKeyboard=> changes the keyboard layout from English to Arabic layout.

changeFocusFromInput=> handle the change of focus and major bug fixing in binding the input with the keyboard.



```
changeFocusFromInput()
{

  let activeElement = document.activeElement.id
  let inputs = document.getElementsByTagName("input")
  if (activeElement === 't1')
  {
    this.Keyboard_name_ar.clearInput()
    this.Keyboard_name_en.clearInput()
    document.getElementById('t2').focus()
    this.nameTabFlag = true

  }
  else if (activeElement === 't2')
  {
    this.Keyboard_numbers.clearInput()
    document.getElementById('t3').focus()
    this.numberTabFlag = true
  }

  if (this.numberTabFlag && this.nameTabFlag)
  {
    this.tabs = document.getElementsByClassName('hg-button-tab')
    for (var item of this.tabs)
    {
      item.style.display = 'none';
      console.log(item)
    }


  }

}
```

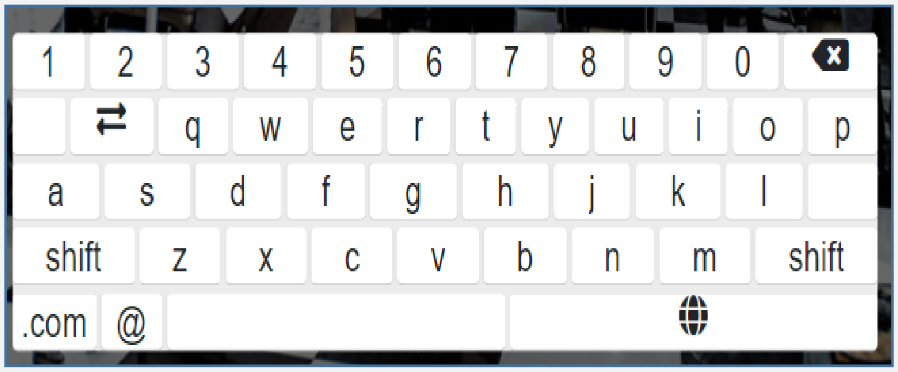


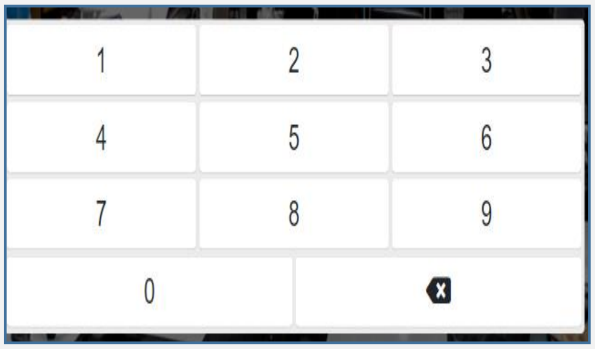
```
hideInput()
{
  document.getElementsByClassName('simple-keyboard1')[0].classList.add('hide-keyboard')
  document.getElementsByClassName('simple-keyboard2')[0].classList.add('hide-keyboard')
  document.getElementsByClassName('simple-keyboard3')[0].classList.add('hide-keyboard')
  document.getElementsByClassName('simple-keyboard4')[0].classList.add('hide-keyboard')
}
```

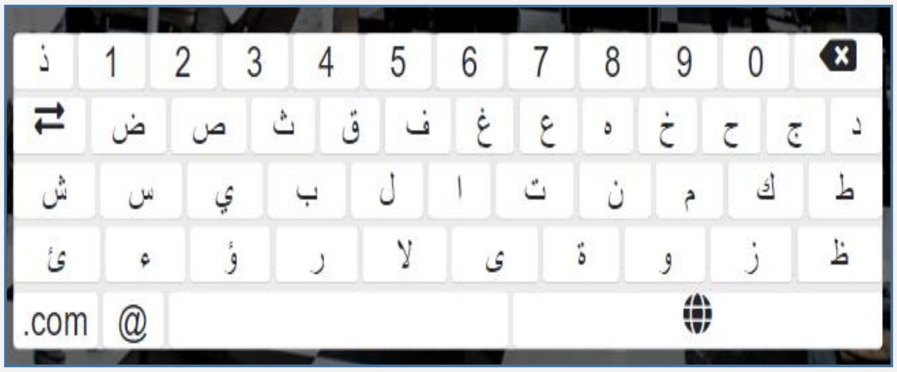
HideInput=> resets all keyboard to be the hidden state.

References:
https://hodgef.com/simple-keyboard/documentation/

Show case for the output product


1 2 3 4 5 6 7 8 9 0
q w e r t y u i o p
a s d f g h j k l
shift z x c v b n m shift
.com @


1 2 3
4 5 6
7 8 9
0


ذ 1 2 3 4 5 6 7 8 9 0
ض ص ث ق ف غ ع ه خ ح ج د
ش س ي ب ل ا ت ن م ك ط
ئ ء ؤ ر لا ى ة و ز ظ
.com @