Knocksinput

knocksinput and knockselinput is a super components for taking user input and processing it following the developers adjustments.
The core difference between knocksinput and knockselinput is the CSS Framework, the first is based on Materialize-Css, while the other is based on Element-UI, both of them has a totally different DOM structure so splitting them into 2 components was just perfect.

Features

1. Validate the input if it is required.
2. Validate the input if it has a maximum length.
3. Validate the input if it has a minimum length.
4. Validate the input if it is numeric.
5. Validate the input if it matches a specific regex.
6. Validate if the input equals another forbidden values.
7. Validate the input live by performing remote XHR, and react based on a specific responses, eg : check a username on user input.
8. Get data sets remotely and represent it for the user to perform an autocomplete process.
9. React on a specific scopes following the developer’s adjustments (Knocks Scopes: a concept by Knocks that performs faster and wider than HTML Forms).
10. Auto reset it self after a successful submit by the user.
11. Customizing the error messages for each validation factor based a default values, or developer’s adjustments.
12. Easier bounding for icons.
13. Translating the placeholders and error messages automatically based on the user’s language.
14. Auto alignment for all the text output based on the user’s language.
15. Auto alignment and styling for user’s input based on what language he types.
16. Customization options for the developer so he can change the whole component styling.

17. A place for the developer to bound an extra content inside the component as a Slot.
18. Handling browsers autocomplete styling.
19. Handling many CSS-frameworks clashes between each others to get an elegant appearance.
20. Focusing on mount option for the developer for more UX priorities.
21. Remote alteration for the input value following the knocks_input_update API and Scopes.
22. Initialize with a specific value as an App autofill.
23. Enter or Return button listening to perform a quick submit (optional).
24. Input event on change.
25. Autocomplete event on data sets arrival.
26. Autocomplete output limits customizations.
27. Autocomplete limits for user input to start performing XHRs.
28. Check live Events carrying the response, and the components analysis for it.

Events API

Global Events

- **knocks_submit**

  requires a scope, it asks the component to declare about it’ s status.

  eg : `App.$emit(‘knocks_submit’,[‘my_scope’]);`

- **knocks_input_reset**

  requires a scope, it resets the component as it’ s empty.

  eg : `App.$emit(‘knocks_input_reset’,[‘my_scope’]);`

- **knocks_input_update**

  requires an object that contains a scope, new value and isFired status(optional).

  eg :
```
App.$emit(‘knocks_input_reset’ ,
                                   {
                                     scope : [‘my_scope’] ,
                                     value : ‘foo’ ,
                                     isFired : true
                                    });
```

Local Events

- **focus**

  the component emits this event once the user focuses on the input, the event has no payloads.

- **blur**

  The component emits this event once the user leaves the input, the event has no payloads.
- **input**

  The component emits this event on the user input, carries the new value as a payload.
- **change**

  The component emits this event on the user input and on every change happens to the input value, the event carries the new value as a payload.
- **loading**

  The component emits this event on check live XHR loading.
- **loading_done**

  The component emits this event on check live XHR loading done.
- **autocomplete**

  In knocksinput, this event is being emitted with a visual representation on displaying autocomplete.

  In knockselinput, this event is being emitted on the XHR response with no visual representation.

  In both knocksinput and knockselinput the event has the XHR response as a payload.

Attributes

```
//The CSS classes that will initialize the icon’s part
icon : {
   type : String ,
   default : null
 } ,
 //The unique id for the component, and it’s required.
 gid : {
   type : String ,
   required : true ,
 } ,
 //The Static Message id that will represent the label
 place_holder : {
   type : Number ,
   default : null,
 } ,
//The Static Message body that will represent the label
 placeholder : {
   type : String ,
   default : null
 },



 //Boolean value, once you turn on it hides the label.
 disable_placeholder: {
   type : Boolean ,
   default : false
 },
//The CSS classes that the input element itself will take.
 knocksclass : {
   type : String ,
   default : 'knocks_input_ps'
 } ,
 //The input type, eg: ‘text’ , ‘number’,etc.
 type : {
   type : String ,
   default : 'text'
 },
//Normally the input listens for Enter press to submit, this
  attribute disable this feature.
unsubmitable : {
 type : Boolean ,
         default : false
        },
        //Initializes a special scope for submitting from inputs.
        submit_scope : {
         type : Array ,
         default : null
        },
        //Boolean value turns on the value existence validation.
        is_required : {
          type : Boolean ,
          default : false
        } ,
        //Boolean value turns on the numerical values validation.
        is_numeric : {
          type : Boolean ,
          default : null
        },

       //Check if the entered value is less than the max value
        max : {
          type : Number ,
          default : undefined ,
        },
       //Check if the entered value is greater than the min value
        min : {
          type : Number ,
          default : undefined ,
        },
       //Compares the input length with the max length and validate it
        max_len : {
          type : Number ,
          default : undefined ,
        },
       //Compares the input length with the min length and validate it
        min_len : {
          type : Number ,
          default : undefined ,
        },
       //Compares the input to validate if it matches the given regex
        regex : {
          type : String ,
          default : undefined ,
        } ,
       //A hint to give to the user so he can know the expected format
        regex_example :{
          type : String ,
          default : null ,
        },
       //Boolean value enables Check Live feature
        check_live : {
          type : Boolean ,
          default : false ,
        } ,


       //The url that will be messaged for Checking Live
       check_at : {
          type : String ,
          default : null
        } ,
       //The expected response that you will refuse.
        check_invalid_at :{
          type : String ,
          defualt : null
        },
       //The expected response that you will accept.
        check_valid_at :{
          type : String ,
          defualt : null
        },
       //Boolean value enables autocomplete feature.
        autocomplete : {
          type : Boolean ,
          default : false
        } ,
       //The url that will be messaged for Autocomplete
        autocomplete_from : {
          type : String ,
          default : null
        } ,
       //Specifies when to start performing autocomplete, eg: 2 chars.
        autocomplete_start : {
          type : Number ,
          default : 2
        },
       //Specifies the maximum results number that will be displayed.
        autocomplete_max_results : {
          type : Number ,
          default : 10
        },


       //CSS Classes will be given to the input on success.
       success_class : {
          type : String ,
          default : 'knocks_input_success '
        } ,
       //CSS Classes will be given to the input on error.
        error_class : {
          type : String ,
          default : 'knocks_input_error animated pulse'
        } ,
       //CSS Classes will be given to initialize the icon.
       icon_class : {
         type : String ,
         default : ''
       },
       //CSS Classes will be given to the icon on focus.
       icon_focus : {
         type : String ,
         default : 'knocks_text_dark_active'
```

```
},
//CSS Classes will be given to the icon on success.
icon_success : {
  type : String ,
  default : 'knocks_text_success'
},
//CSS Classes will be given to icon on error.
icon_error : {
  type : String ,
  default : 'knocks_text_error'
},
//Forms and events scope for the component.
scope : {
  type : Array ,
  default : null
},



//A value to compare the input value with, similarity is error.
same_as : {
  type : String ,
  default : null
} ,
//The value placeholder that will be displayed on error.
same_as_name : {
  type : String ,
  default : ''
},
//Boolean Value usually is false
mat_follower : {
  type : Boolean ,
  default : false
},
//Boolean value usually is true
el_follower : {
  type : Boolean ,
  default : true
```

```
        },

        check_live_prefix_msg : {
          type : String ,
          default : '' ,
        },
        //ERROR MESSAGES CUSTOMIZATION
        ////Errors Messages
        is_required_msg : {
          type : String ,
          default : 'This field is required.' ,
        },
        is_numeric_msg : {
          type : String ,
          default : 'This field expects a numeric value.' ,
        },


        max_value_msg : {
          type : String ,
          default : 'The maximum value for this field is' ,
        },
        min_value_msg : {
          type : String ,
          default : 'The minimum value for this field is' ,
        },
        max_length_msg : {
          type : String ,
          default : 'The maximum length for your value shouldn\'t pass'
,
        },
        min_length_msg : {
          type : String ,
          default : 'The minimum length for your value shouldn\'t be
less than' ,
        },
        regex_bus_msg : {
          type : String ,
          default : 'This field expects another formula'
        },
        check_live_msg : {
          type : String ,
          default : 'This value is not available' ,
        },
       samilarity_msg : {
          type : String ,
          default : 'This value should be the same as' ,
        },
        ////Errors Icons
        is_required_icon : {
          type : String ,
          default : 'knocks-alert-circle' ,
        },


        is_numeric_icon : {
          type : String ,
          default : 'knocks-alert-circle' ,
        },
        max_value_icon : {
          type : String ,
          default : 'knocks-alert-circle' ,
        },
        min_value_icon : {
          type : String ,
          default : 'knocks-alert-circle' ,
        },
        max_length_icon : {
          type : String ,
          default : 'knocks-alert-circle' ,
        },
        min_length_icon : {
          type : String ,
          default : 'knocks-alert-circle' ,
        },
        regex_bus_icon : {
          type : String ,
          default : 'knocks-alert-circle' ,
        },
        check_live_icon : {
          type : String ,
          default : 'knocks-alert-circle' ,
        },
       samilarity_icon : {
          type : String ,
          default : 'knocks-alert-circle' ,
        },
        focus_on_mount : {
          type : Boolean ,
          default : false
        },
        //Handle styling on browsers autofill
        handle_autofill : {
          type : Boolean ,
          default : false
        } ,
        //a value to start the input with
        start_as : {
          type : [String , Number, Array , Object] ,
          default : null
        },
        //Boolean flag turns off displaying errors
        hide_errors : {
          type : Boolean ,
          default : false
        },
        //Boolean flag turns on showing a progress division
        show_autocomplete_progress : {
          type : Boolean ,
          default : false
        },
        //Progress message customizations
        autocomplete_progress_message : {
          type : String ,
          default : 'Searching...'
        },
        autocomplete_progress_message_classes : {
          type : String ,
          default : 'blue-text'
        }
```

Perfect Initialization Example

```
<knockselinput
gid = “my_input_unique_id”
v-model = “myInput”
start_as = “hello world”
:scope = “[‘first_form’ , ‘general_form’]”
:submit_scope = “[‘first_form’]”
placeholder = “My Input”
icon = “knocks-icon”
is_required
:max_len = “10”
:min_len = “2”
regex = “foo-bar”
regex_example = “your input must matches foo-bar”
check_live
check_at = "user/check"
check_invalid_at = "exist"
check_valid_at = “not_exist"
check_live_prefix_msg = ", This value is already taken.”
autocomplete
:autocomplete_start = “3"
autocomplete_on_mount
autocomplete_from = “user/search"
@focus = “isOnFocus = true”
@blur = “isOnFocus = false”
@change = “hasChangedOnce = true”
@autocomplete = “handleResults($event)”>
</knockselinput>
```