SurrealismUI

SurrealismUI是一个完全使用Slint进行构建的Slint第三方组件库

SurrealismUI is a third-party component library built entirely using Slint

About Doc Icon

- ⛔ ： do not use

Themes

Built in 6 theme colors in SurrealismUI

- primary
- success
- info
- warning
- error
- dark

themes-color

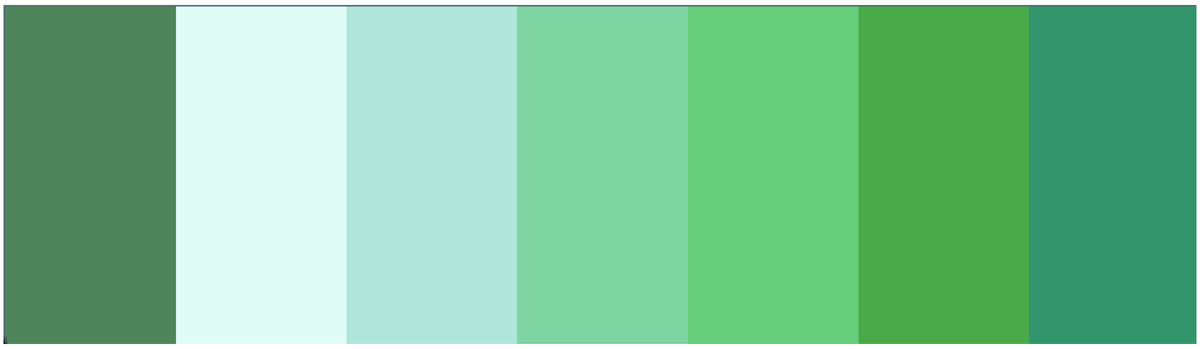
primary

1. opacity：#1A5BE988
2. font：#bbdbf6
3. weakest：#96C4ED
4. weaker：#4584E9
5. normal：#1A5BE9
6. deeper：#0F3CC9
7. deepest：#1d2f7a

success

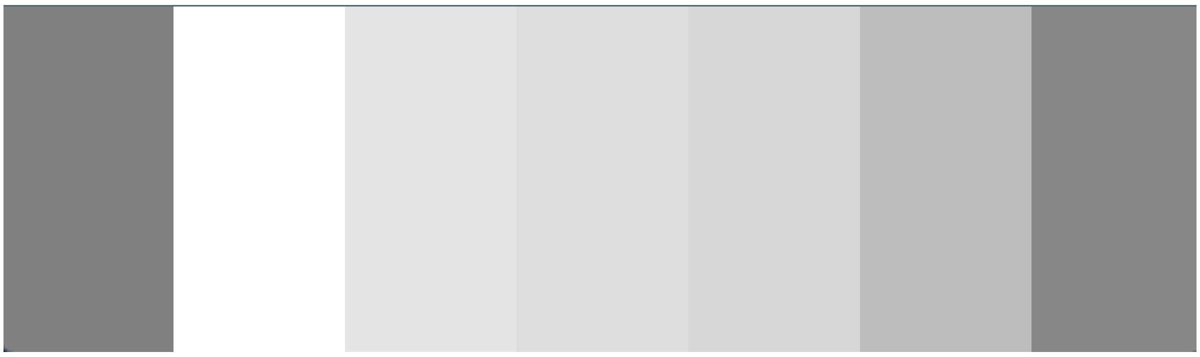
1. opacity：#7de39187
2. font：#e0fcf7
3. weakest：#B0E5DC
4. weaker：#7FD5A2

5. normal：#66CD7A
6. deeper：#4aa949
7. deepest：#33956B



info

1. opacity：#d7d7d788
2. font：#ffffff
3. weakest：#E6E5E6
4. weaker：#DEDDDE
5. normal：#d7d7d7
6. deeper：#bcbcbc
7. deepest：#878787

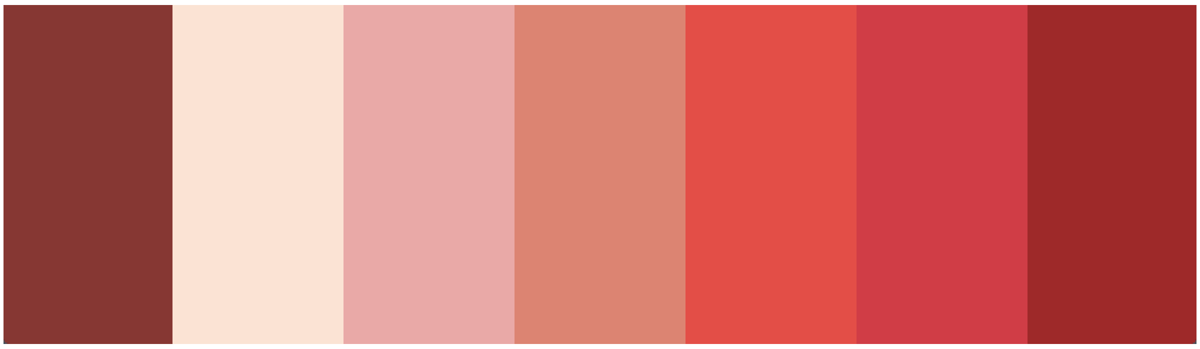


warning

1. opacity：#f06b4288
2. font：#fdd1c3
3. weakest：#e48d73
4. weaker：#f07651
5. normal：#f06b42
6. deeper：#e95a2e
7. deepest：#e63819

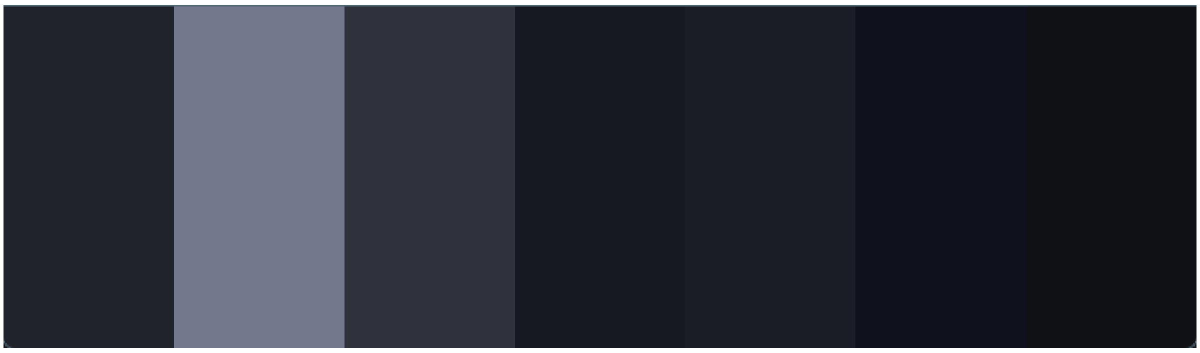
error

1. opacity：#e34e4788
2. font：#fbe3d4
3. weakest：#e9a9a7
4. weaker：#DC8472
5. normal：#e34e47
6. deeper：#D03D46
7. deepest：#9e2929



dark

1. opacity：#262a3987
2. font：#73788c
3. weakest：#2f323d
4. weaker：#171922
5. normal：#1a1c26
6. deeper：#0f121c
7. deepest：#101114



Components

SURText

It is the simplest and most common component in SurrealismUI

properties:

- `in property <Themes> theme` : Surrealism themes
- `in-out property <string> content` : the content in SURText

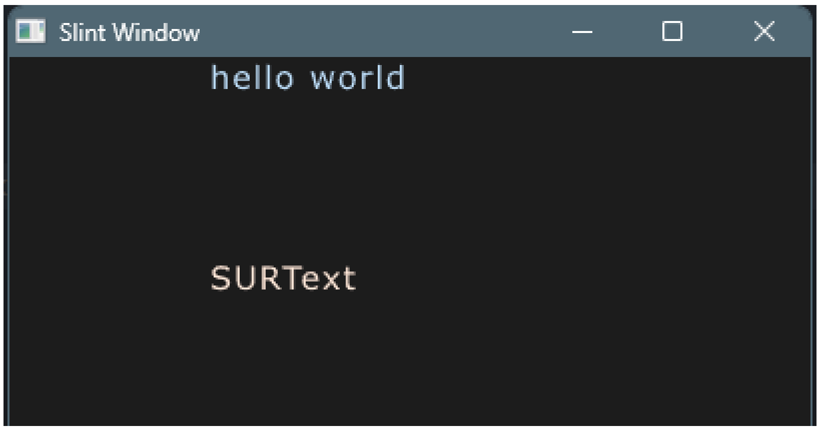
callbacks:

functions:

- `pure public function get()->string` : get content
- `public function set(content:string)` : set content

example

```
import {SURText} from "../../components/index.slint";
import {Themes} from "../../components/themes/index.slint";

component TestWindow inherits Window {
  height: 400px;
  width: 400px;
  SURText {
    x: 100px;
    y: 0;
    content: "hello world";
  }
  SURText {
    x:100px;
    y:100px;
    theme:Themes.Error;
  }

}
```



SURIcon

there are 2658 different icons in SURIcon from : https://github.com/bytedance/iconpark

properties:

- `in property <Icons> icon` : icon types
- `in property <Themes> theme` : Surrealism theme
- `in-out property <brush> icon-color` : icon color
- `private property <[IconItem]> icon-datas` : source icon datas ⛔

callbacks:

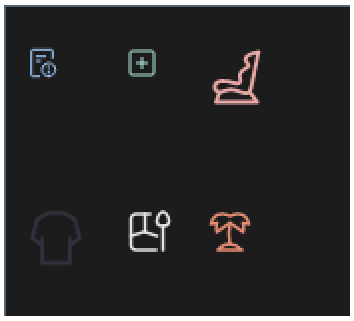
- `callback clicked` : run if you click the icon

functions:

- `pure function get_icon(item:IconItem)->image` : get icon src from for iter item ⛔

example

```
import {SURIcon} from "../../components/index.slint";
import {Icons,Size,Themes} from "../../components/themes/index.slint";
component TestWindow inherits Window {
  height: 400px;
  width: 400px;
  SURIcon{
    x: 10px;
    y: 20px;
    icon: Icons.Abnormal;
    theme: Themes.Primary;
  }
  SURIcon{
    x: 60px;
    y: 20px;
    icon: Icons.Add;
    theme: Themes.Success;
  }
  SURIcon{
    x: 100px;
    y: 20px;
    height: 30px;
    width: 30px;
    icon: Icons.Baby-car-seat;
    theme: Themes.Error;

  }
  SURIcon{
    x: 10px;
    y: 100px;
    icon: Icons.T-shirt;
    theme: Themes.Dark;
    height: 30px;
    width: 30px;
  }
  SURIcon{
    height: 24px;
    width: 24px;
    x: 60px;
    y: 100px;
    icon: Icons.Baby-meal;
    theme: Themes.Info;
  }
  SURIcon{
    height: 24px;
    width: 24px;
    x: 100px;
    y: 100px;
    icon: Icons.Vacation;
    theme: Themes.Warning;
    clicked=>{
      debug("clicked");
      self.theme= Themes.Error;
      self.height += 2px;
      self.width += 2px;
    }
  }
}
```



SURCard

A very simple universal card without any layout or restrictions
you can add anything you want to the card

properties

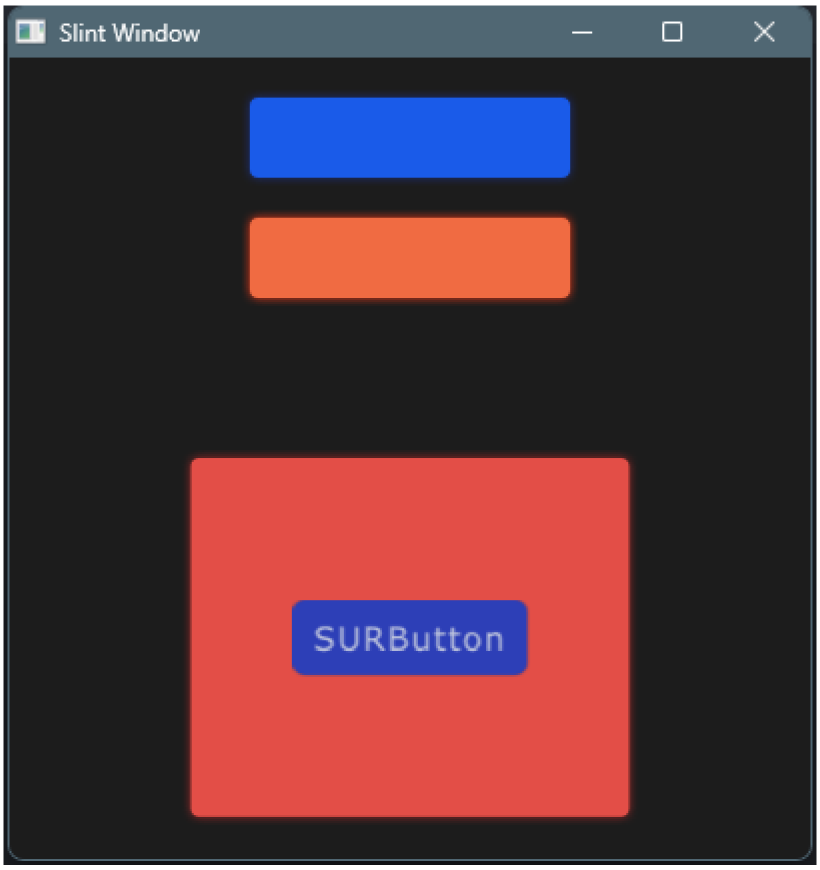
- `in property <Themes> theme` : Surrealism theme

functions

- `pure public function count-height(h:length) -> length` : a cheap way to calculate height
- `pure public function count-width(w:length) -> length` : a cheap way to calculate width

example

```
import {SURButton,SURCard} from "../../components/index.slint";
import {Themes,Icons} from "../../components/themes/index.slint";

component TestWindow inherits Window {
  height: 400px;
  width: 400px;
  SURCard {
    y: 20px;
    height: 40px;
    width: 160px;
   }
   SURCard {
    y: 80px;
    height: 40px;
    width: 160px;
    theme: Themes.Warning;
   }
   SURCard {
    y: 200px;
    height: self.count-height(160px);
    width: self.count-width(200px);
    theme: Themes.Error;
    SURButton {

    }
   }
}
```



SURButton

SURButton is a button component that you can freely perform regular attribute operations on

properties

- `in property <Themes> theme`: Surrealism Themes
- `in property <Icons> icon` : Icons.Null : do button has icon
- `in-out property <brush> font-color`: button content color
- `in-out property <brush> icon-color` : button icon color
- `in property <length> font-size` : font size
- `in property <int> font-weight` : font weight
- `in property <bool> font-italic`: font italic
- `in property <string> font-family` : font family
- `in property <bool> circle` : set the button as a rounded button
- `private property <length> letter-spacing` : content letter-spacing ⛔
- `in-out property <string> content` : the content of the button

functions

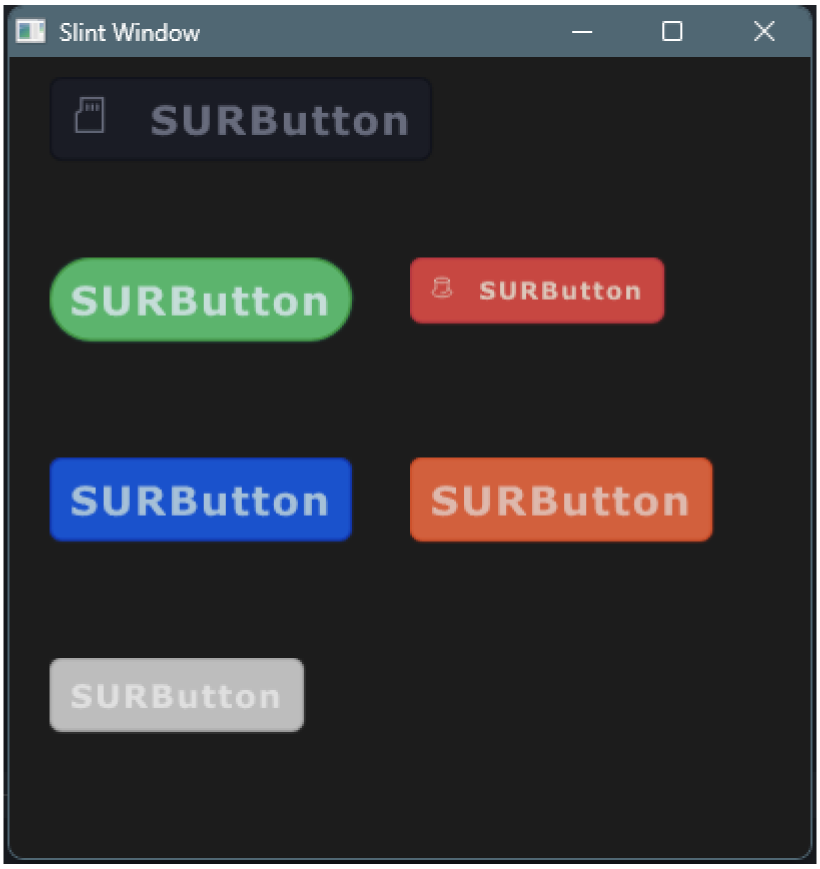
callbacks

- `clicked` : run if you click the button

example

```
import {SURButton} from "../../components/index.slint";
import {Themes,Icons} from "../../components/themes/index.slint";
component TestWindow inherits Window {
  height: 400px;
  width: 400px;

  SURButton {
    x: 20px;
    y: 10px;
    font-size: 20px;
    font-weight:700;
    theme:Themes.Dark;
    icon:Icons.Mini-sd-card;
    clicked => {
      self.content = "clicked"
    }
  }
  SURButton {
    x: 20px;
    y: 100px;
    font-size: 20px;
    font-weight:700;
    theme:Themes.Success;
    circle:true;
  }
```

```
    SURButton {
      x: 20px;
      y: 200px;
      font-size: 20px;
      font-weight:700;
      theme:Themes.Primary;
    }
    SURButton {
      x: 20px;
      y: 300px;
      font-weight:700;
      theme:Themes.Info;
    }
    SURButton {
      x: 200px;
      y: 100px;
      font-size: 12px;
      font-weight:700;
      theme:Themes.Error;
      icon:Icons.Magic-hat;
    }
    SURButton {
      x: 200px;
      y: 200px;
      font-size: 20px;
      font-weight:700;
      theme:Themes.Warning;
    }
  }
```



SURInput

This is a basic input box, often used in forms, divided into two types` : text and password

properties :

- `in property <string> placeholder` : default placeholder which you wanna show when no content
- `in property <Themes> theme` : Surrealism themes
- `in property <Icons> icon` : icon you wanna show in front (use >= v0.1.0) ⛔
- `in property <length> input-width` : Please do not use width to adjust the length of the input box , use this property to instead
- `in property <length> font-size` : font size
- `in property <bool> disabled` : can input be edited
- `in property <bool> clearable` : can input be cleared
- `in property <bool> password` : can the password input display the password
- `out property <bool> has-focus` : input is focused or not
- `private property <brush> placeholder-color` : placeholder color
- `in-out property <InputType> type` : input type (text or password)
- `in-out property <brush> font-color` : font color
- `in-out property <brush> icon-color` : icon color
- `in-out property <string> content` : the content of the input

functions :

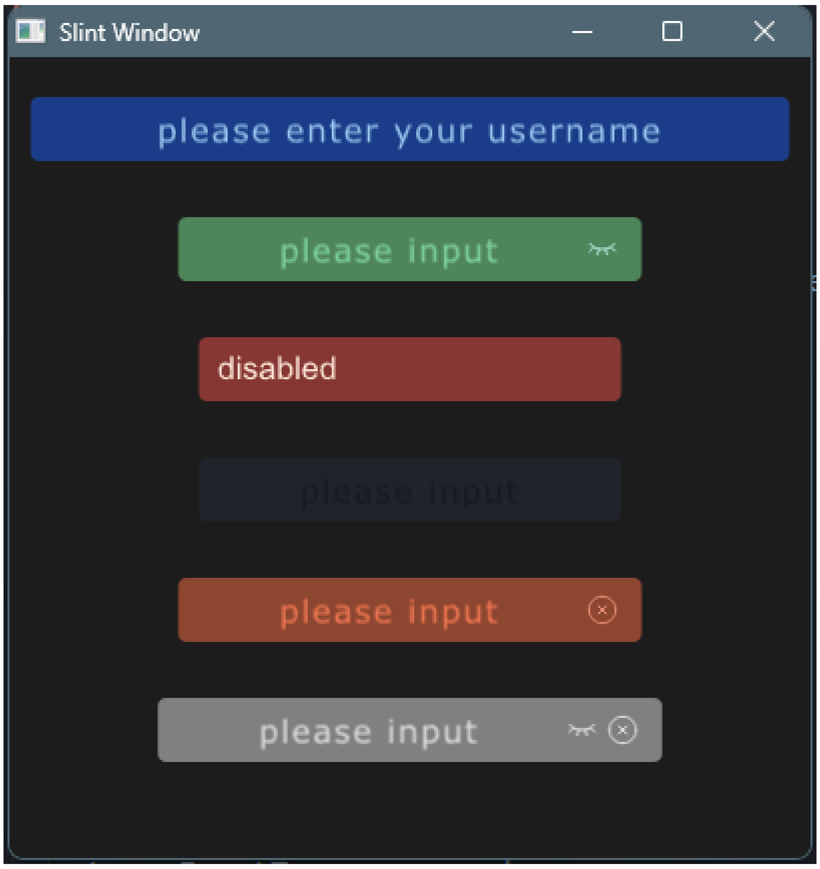
- `pure public function get() ->string` : get content
- `public function set(content` :string) ` : set content
- `pure public function count-width()->length` : count input real width ⛔

callbacks :

- `callback accepted(string)` : run when pressed down Enter key
- `callback changed(string)` : run when content changed
- `callback clear()` : empty content

example

```
import {SURText,SURInput,SURButton, SURIcon} from
"../../components/index.slint";
import {Themes} from "../../components/themes/index.slint";

component TestWindow inherits Window {
  height: 400px;
  width: 400px;

  SURInput{
    y: 20px;
    placeholder :"please enter your username";
    input-width:360px;
    accepted(res)=>{
      debug("content in input:" + res);
    }
    changed(change-res)=>{
      debug(change-res);
    }

  }
  w:=SURInput{
    y: 80px;
    theme:Themes.Success;
    type:InputType.password;
    password:true;
  }
  SURInput{
    y: 140px;
    theme:Themes.Error;
    disabled:true;
    content:"disabled";
  }
  SURInput{
    y: 200px;
    theme:Themes.Dark;
  }

  SURInput{
    y: 260px;
    theme:Themes.Warning;
    clearable:true;
  }
  SURInput{
    y: 320px;
    theme:Themes.Info;
    type:InputType.password;
    clearable:true;
    password:true;
  }

}
```



SURStar

SURStar is a scoring component

properties

- `in property <bool> no-theme` : use Surrealism Theme or not
- `in property <float> score` : the real score
- `in property <Themes> theme` : Themes.Primary;
- `in property <bool> disabled` : can be scored if disabled is false
- `in property <float> max-score` : max score (how many stars you wanna show)

functions

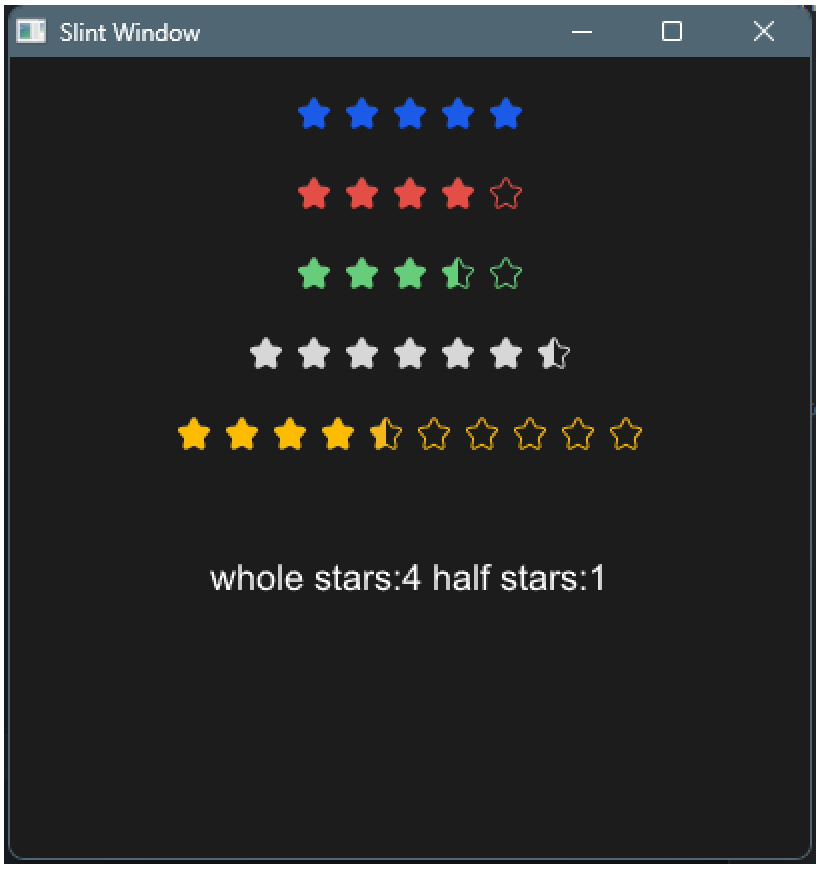
- `pure function get-half-stars()->bool`: count the number of half stars ⛔
- `pure function get-whole-stars()->int` : count the number of whole stars ⛔
- `pure function get-empty-stars()->int` : count the number of empty stars ⛔

callbacks

- `callback clicked(float,float)` : get how many whole stars and half stars

example

```
import {SURStar} from "../../components/index.slint";
import {Themes,Icons} from "../../components/themes/index.slint";

component TestWindow inherits Window {
  height: 400px;
  width: 400px;
  SURStar {
    y: 20px;
  }
  SURStar {
    score: 4.2;
    y: 60px;
    theme: Error;
  }
  SURStar {
    score : 3.8;
    disabled: true;
    y: 100px;
    theme: Success;
  }
  SURStar {
    max-score : 7;
    score : 6.8;
    y: 140px;
    theme: Info;
  }
  SURStar {
    max-score : 10;
    score : 7.2;
    y: 180px;
    no-theme:true;
    clicked(whole,half) => {
      t.n = whole;
      t.m = half;
    }

  }
  t:=Text{
    y: 250px;
    font-size: 18px;
    in-out property <int> n;
    in-out property <int> m;
    text: "whole stars:"+ n + " half stars:" + m;
  }
}
```



SURTag

A small tag used to display data

properties

- `in property <string> content` : the content of the tag
- see card's properties

functions

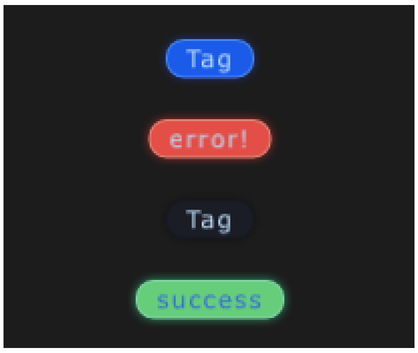
see card's functions

callbacks

- `callback clicked()` : run if you click the tag

example

```
import {SURTag} from "../../components/index.slint";
import {Themes,Icons} from "../../components/themes/index.slint";

component TestWindow inherits Window {
  height: 400px;
  width: 400px;
  SURTag {
    y: 40px;
  }
  SURTag {
    content:"error!";
    y:80px;
    theme:Themes.Error;
  }
  SURTag {
    y:120px;
    theme:Themes.Dark;
    clicked=>{
      self.font-color= #ddff00;
    }
  }
  SURTag {
    content:"success";
    y:160px;
    font-color:#3670d5;
    theme:Themes.Success;
  }
}
```



SURHeader

SURHeader is a simple header component that is generated based on routing information

properties

- `in property <Themes> theme` : Surrealism Themes
- `in property <Route> route` : detail routes , like: `{home:"Surrealism",routes:["user","info"]};`
- `in property <length> font-size` : font size

functions

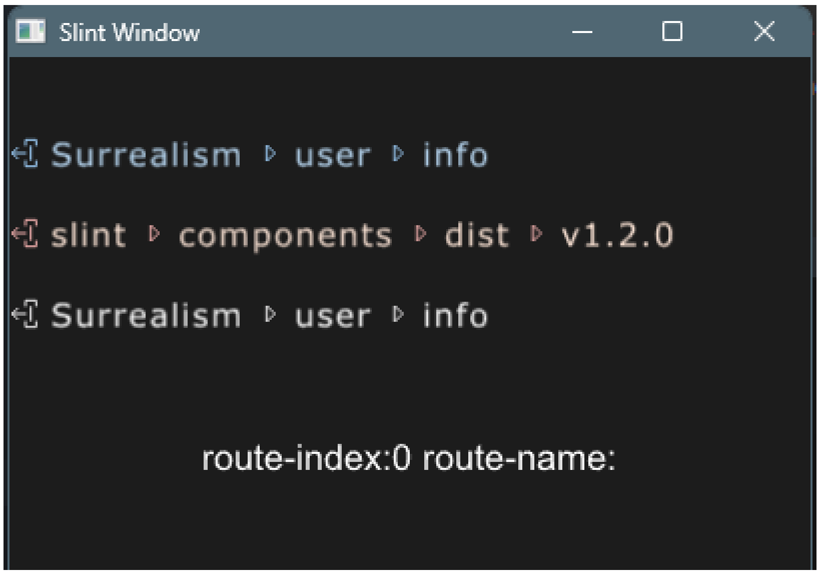
callbacks

- `callback to(int,string)` : to page (it depends on you)
- `callback back()` : back to main page (it depends on you)

example

```
import {SURHeader} from "../../components/index.slint";
import {Themes,Icons} from "../../components/themes/index.slint";

component TestWindow inherits Window {
  height: 400px;
  width: 400px;
  SURHeader {
    x:0px;
```

```
      y: 40px;
    }
    SURHeader {
      x:0px;
      y: 80px;
      theme: Error;
    }
    SURHeader {
      x:0px;
      y: 120px;
      theme: Info;
      to(index,route)=>{
        txt.name = route;
        txt.index = index;
      }
      back=>{
        txt.name = "back";
      }
    }
    txt:=Text{
      font-size: 18px;
      in-out property <int> index;
      in-out property <string> name;
      text: "route-index:" + index + " route-name:" + name;
    }
  }
```



route-index:0 route-name:

SURTable

This is the outter of the Table, and the column data of the table is separated from the outter The outter only serves as a standard layout , this is a zero cost construction

properties

- see SURCard

functions

- see SURCard

callbacks

- see SURCard

SURTableColumn

SURTableColumn is a component of SURTable, and each SURTableColumn forms a complete column of the table
If it's gone, the table will become a card with a horizontal layout

properties

- `in property <bool> border` : add border or not
- `in property <string> name` : table header name
- `in property <[string]> datas` : table datas
- `in property <brush> header-background` : define header background
- `in property <brush> row-background` : define row background
- `in property <Themes> theme` : Surrealism Themes
- `in property <length> header-height`: define header height
- `in property <length> row-height` : define each row height
- `in property <bool> operation-enabled`: enable operation
- `in property <[{name:string,theme:Themes}]> operation` : the operations you wanna do

functions

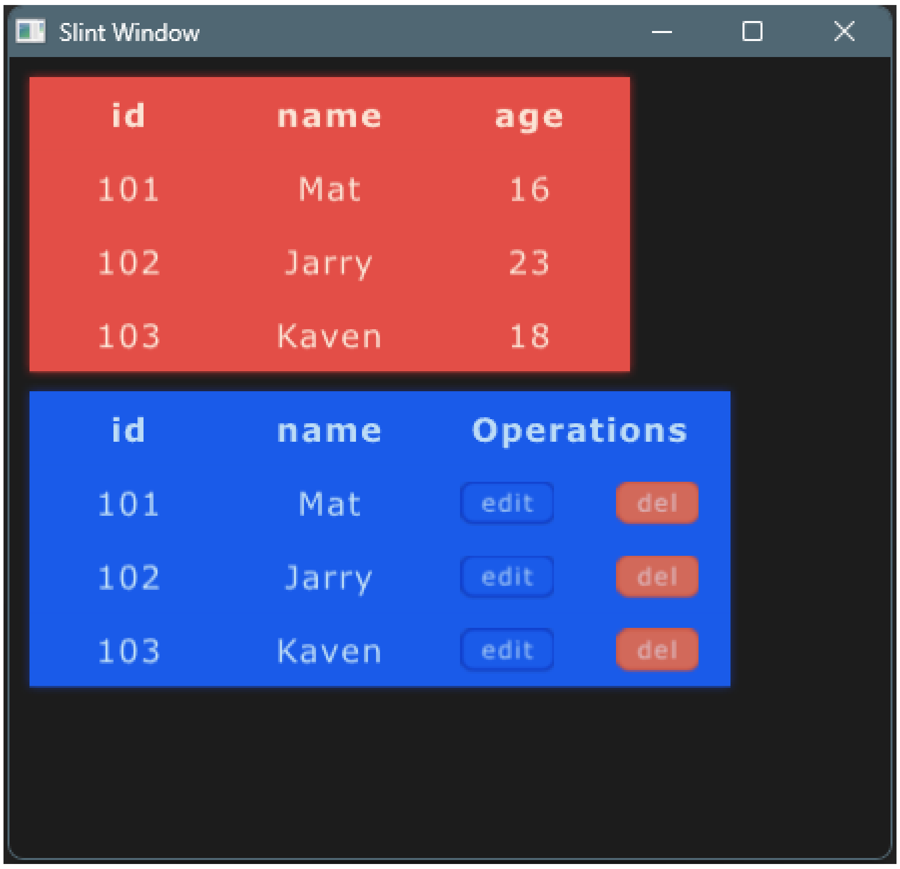
- `function count() ->int` : count the number of row ⛔
- `pure public function get-height()->length` : auto count the height of the table and return height

callbacks

- `callback clicked(int,string)` : run if operation-enabled is true , you will get which operation button you clicked

example

```
import {SURTable,SURTableColumn} from "../../components/index.slint";
import {Themes,Icons} from "../../components/themes/index.slint";

component TestWindow inherits Window {
  height: 400px;
  width: 440px;
  t1:=SURTable {
    x: 10px;
    y: 10px;
    // you can use this way to get height
    // it depends on how many datas in column
    height: col1.get-height();
    width: 300px;
    theme:Themes.Error;
    col1:=SURTableColumn {
      border:false;
      theme:Themes.Error;
      width: 100px;
      name:"id";
      // row-height:60px;
      datas: ["101","102","103"];
    }
    SURTableColumn {
      theme:Themes.Error;
      width: 100px;
      name:"name";
      datas: ["Mat","Jarry","Kaven"];
    }
    SURTableColumn {
      theme:Themes.Error;
      width: 100px;
      name:"age";
      datas: ["16","23","18"];
    }
  }
  t2:=SURTable {
    x: 10px;
    y: t1.height + 20px;
    // you can use this way to get height
    // it depends on how many datas in column
    height: tcol1.get-height();
    width: 350px;
    theme:Themes.Primary;
    tcol1:=SURTableColumn {
      border:false;
      theme:Themes.Primary;
      width: 100px;
      name:"id";
      // row-height:60px;
      datas: ["101","102","103"];
    }
    SURTableColumn {
      theme:Themes.Primary;
      width: 100px;
      name:"name";
      datas: ["Mat","Jarry","Kaven"];
    }
    SURTableColumn {
      theme:Themes.Primary;
      width: 150px;
      name:"Operations";
      // cheat datas
      datas: [" "," "," "];
      operation-enabled:true;
    }
  }
}
```



id	name	age
101	Mat	16
102	Jarry	23
103	Kaven	18

id	name	Operations	
101	Mat	edit	del
102	Jarry	edit	del
103	Kaven	edit	del

SURCollapse

SURCollapse is a foldable panel

This is the outter of the Collapse, what really works is SURCollapseItem

The outter only serves as a standard layout , this is a zero cost construction

properties

- see SURCard

functions

- see SURCard

callbacks

- see SURCard

SURCollapseItem

SURCollapseItem is a component of SURCollapse, without which SURCollapse will not work
You can customize the components or use the default text display method in it

properties

- `in property <length> item-height`: set height of detail
- `in property <string> name` : collapse header;
- `in property <string> detail` : the content of detail

- `in property <bool> define` : define detail or not (if you wanan show something special use true!)
- `in property <Themes> theme` : Surrealism Themes
- `private property <bool> show` : show details or not ⛔

functions

- `pure public function get-height()->length`: get collapse header height

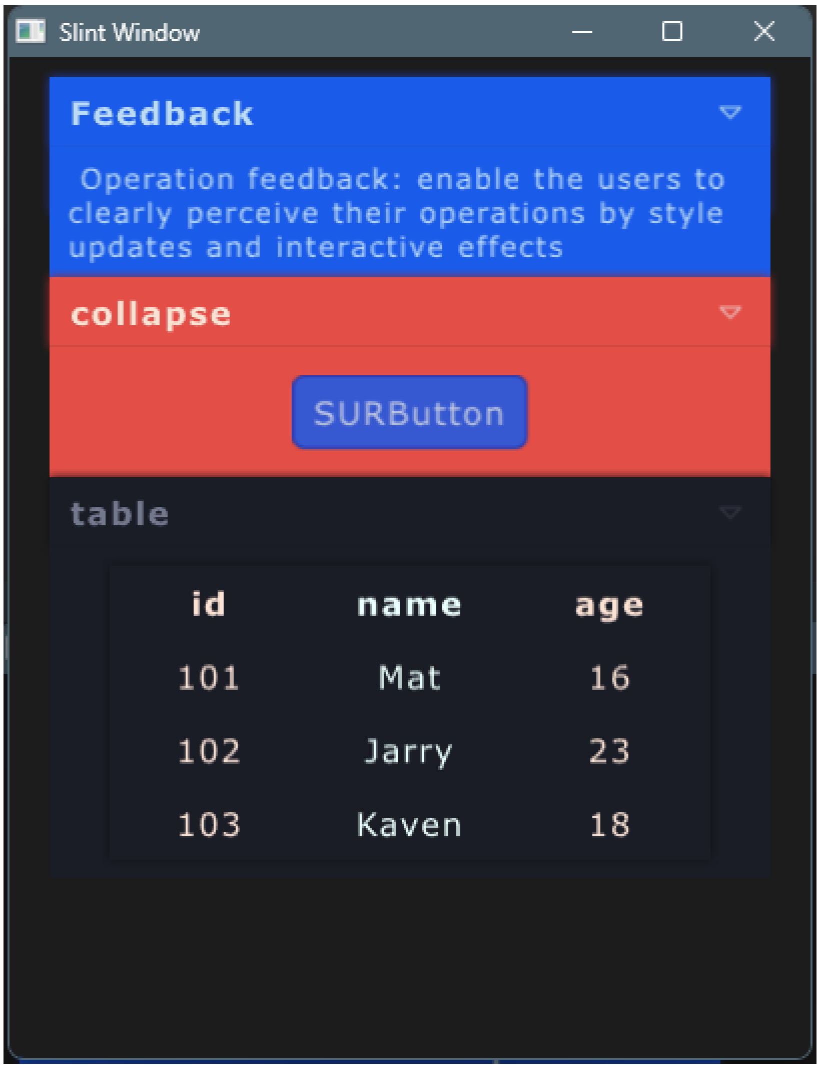
callbacks

- `callback clicked()` : run if you show collapse detail

example

```
import {SURCollapse,SURCollapseItem,SURButton,SURTable,SURTableColumn} from "../../components/index.slint";
import {Themes,Icons} from "../../components/themes/index.slint";


component TestWindow inherits Window {
  height: 500px;
  width: 400px;
  SURCollapse {
    y: 10px;
    // you can set 0 , it has no impact
    // recommend use the following way
    height: item1.get-height() * 2;
    width: 360px;
    item1:=SURCollapseItem {
      name:"Feedback";
      detail:" Operation feedback: enable the users to clearly perceive their operations by style updates and interactive effects";

    }
    SURCollapseItem {
      theme: Themes.Error;
      define:true;
      SURButton {

      }
    }
    SURCollapseItem {
      name:"table";
      theme: Themes.Dark;
      define:true;
      item-height:200px;
      SURTable {

        height: col1.get-height();
        width: 300px;
        theme:Themes.Dark;
        col1:=SURTableColumn {
          border:false;
          theme:Themes.Error;
```

```
            width: 100px;
            name:"id";
            // row-height:60px;
            datas: ["101","102","103"];
          }
          SURTableColumn {
            theme:Themes.Success;
            width: 100px;
            name:"name";
            datas: ["Mat","Jarry","Kaven"];
          }
          SURTableColumn {
            theme:Themes.Error;
            width: 100px;
            name:"age";
            datas: ["16","23","18"];
          }
        }
      }
    }
}
```


Slint Window
Feedback
Operation feedback: enable the users to clearly perceive their operations by style updates and interactive effects
collapse
SURButton
table

id	name	age
101	Mat	16
102	Jarry	23
103	Kaven	18

SURResult

SURResult helps you easily build a quick prompt , you can build it in popup window

properties

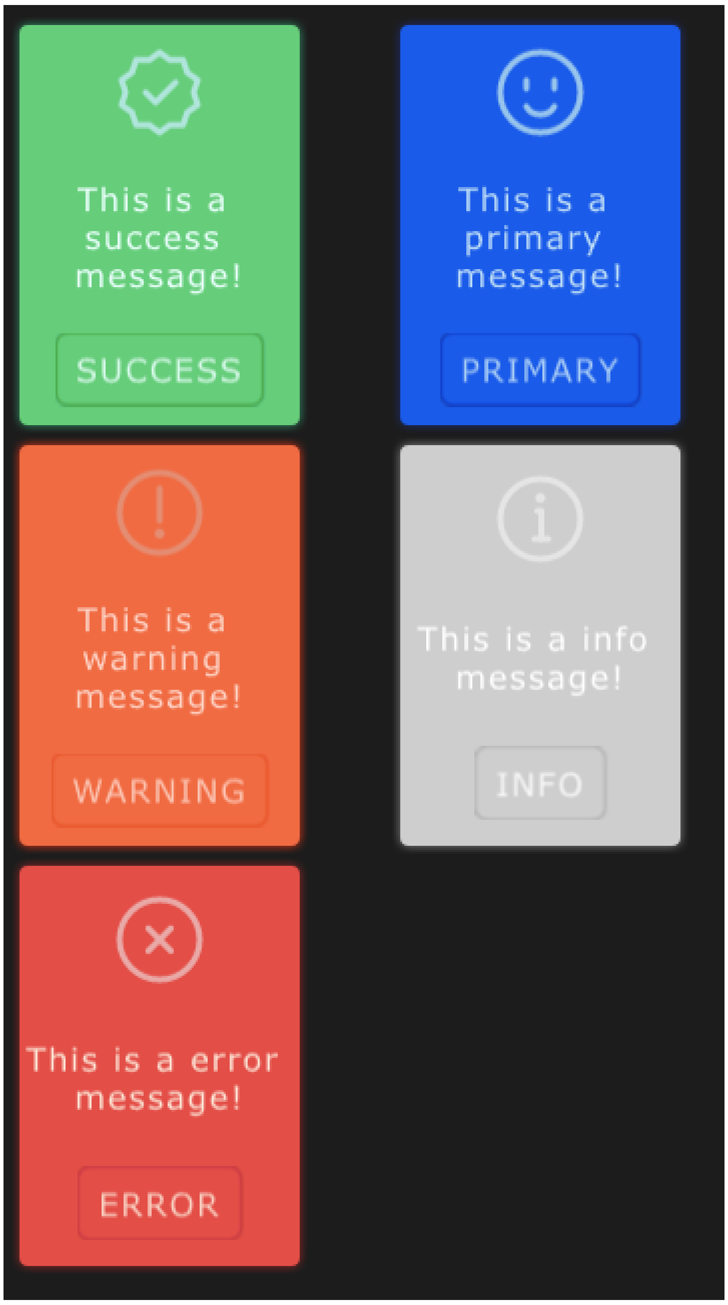
- `in property <length> icon-size`: icon size
- `in-out property <string> btn`: the content of the button
- `in-out property <string> content` : content of the result
- `in property <ResType> res-type` : Result type
- `in-out property <Icons> icon`: Icon of the result

functions

callbacks

- `callback clicked()` : run if you click the button

```
import {SURResult,ResType} from "../../components/index.slint";
import {Themes,Icons} from "../../components/themes/index.slint";

component TestWindow inherits Window {
  height: 660px;
  width: 400px;
  SURResult {
    x: 10px;
    y: 10px;
  }
  SURResult {
    x: 200px;
    y: 10px;
    res-type:ResType.Primary;
  }
  SURResult {
    x: 200px;
    y: 220px;
    res-type:ResType.Info;
  }
  SURResult {
    x: 10px;
    y: 220px;
    res-type:ResType.Warning;
  }

  SURResult {
    x: 10px;
    y: 430px;
    res-type:ResType.Error;
  }
}
```


This is a success message!
SUCCESS
This is a primary message!
PRIMARY
This is a warning message!
WARNING
This is a info message!
INFO
This is a error message!
ERROR

SURSelect

SURSelect is a selector that provides three types of optional input parameter values

properties

- `in property <Themes> theme` : Surrealism Themes
- `in property <[{id:int,label:string,value:string}]> ranges-string` : select list range (type string)
- `in property <[{id:int,label:string,value:int}]> ranges-int` : select list range (type int)
- `in property <[{id:int,label:string,value:float}]> ranges-float` : select list range (type float)
- `in property <string> placeholder` : placeholder of the select
- `private property <brush> input-color` : the color of the select content ⛔
- `private property <bool> open` : open the select list or not ⛔

- `private property <int> range-type` : the type of the range value ⛔

functions

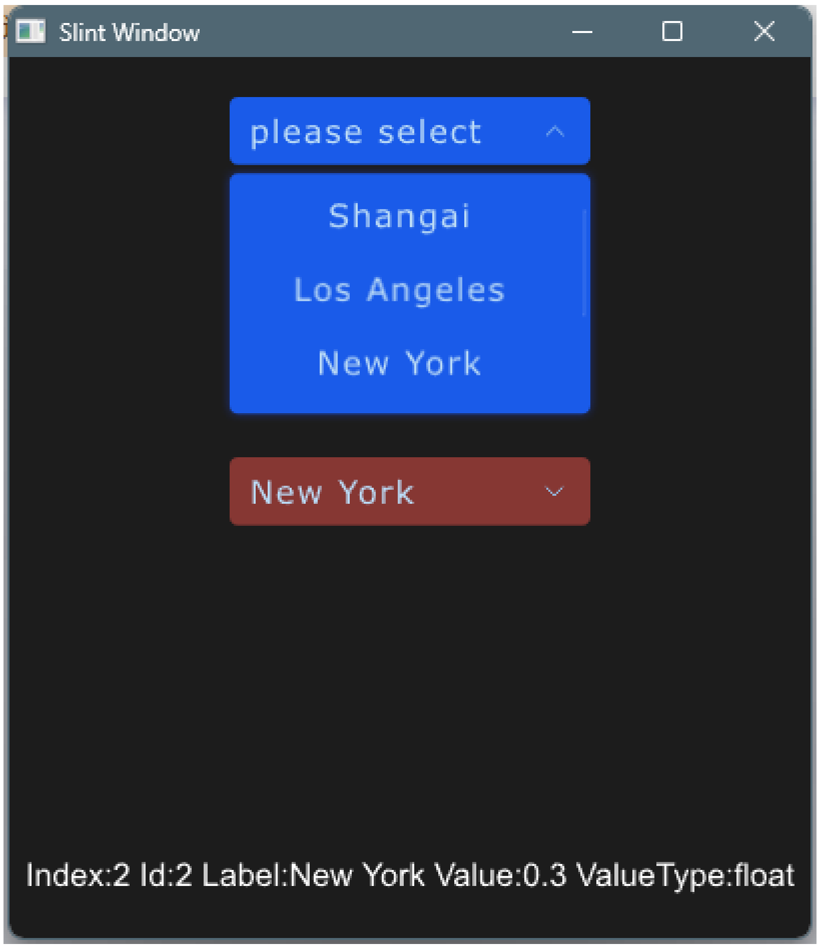
- `pure public function count-width(len:length)->length` : auto count the width of the select

callbacks

- `callback changed(int,int,string,string,ValueType)` : run if you choose an item of list

example

```
import {SURSelect,ValueType} from "../../components/index.slint";
import {Themes,Icons} from "../../components/themes/index.slint";

component TestWindow inherits Window {
  height: 440px;
  width: 400px;
  SURSelect {
    y: 20px;
    ranges-string: [
      {id:0,label:"Shangai",value:"s01"},
      {id:1,label:"Los Angeles",value:"l02"},
      {id:2,label:"New York",value:"n03"},
      {id:3,label:"Hong Kong",value:"h04"},
    ];
  }
  SURSelect {
    y: 200px;
    theme: Error;
    ranges-float: [
      {id:0,label:"Shangai",value:0.1},
      {id:1,label:"Los Angeles",value:0.2},
      {id:2,label:"New York",value:0.3},
      {id:3,label:"Hong Kong",value:0.4},
    ];
    changed(index,id,label,value,value-type)=>{
      if(value-type==ValueType.String){
        t.vt = "string";
      }else if(value-type==ValueType.Float){
        t.vt = "float"
      }else{
        t.vt = "int"
      }
      t.index = index;
      t.id = id;
      t.label = label;
      t.value = value;
    }
  }
  t:=Text{
    y: 400px;
    font-size: 16px;
    in-out property <int> index;
    in-out property <int> id;
    in-out property <string> label;
    in-out property <string> vt;
    in-out property <string> value;
    text: @tr("Index:{} Id:{} Label:{} Value:{} ValueType:
{}",index,id,label,value,vt);
  }
}
```



```
Index:2 Id:2 Label:New York Value:0.3 ValueType:float
```