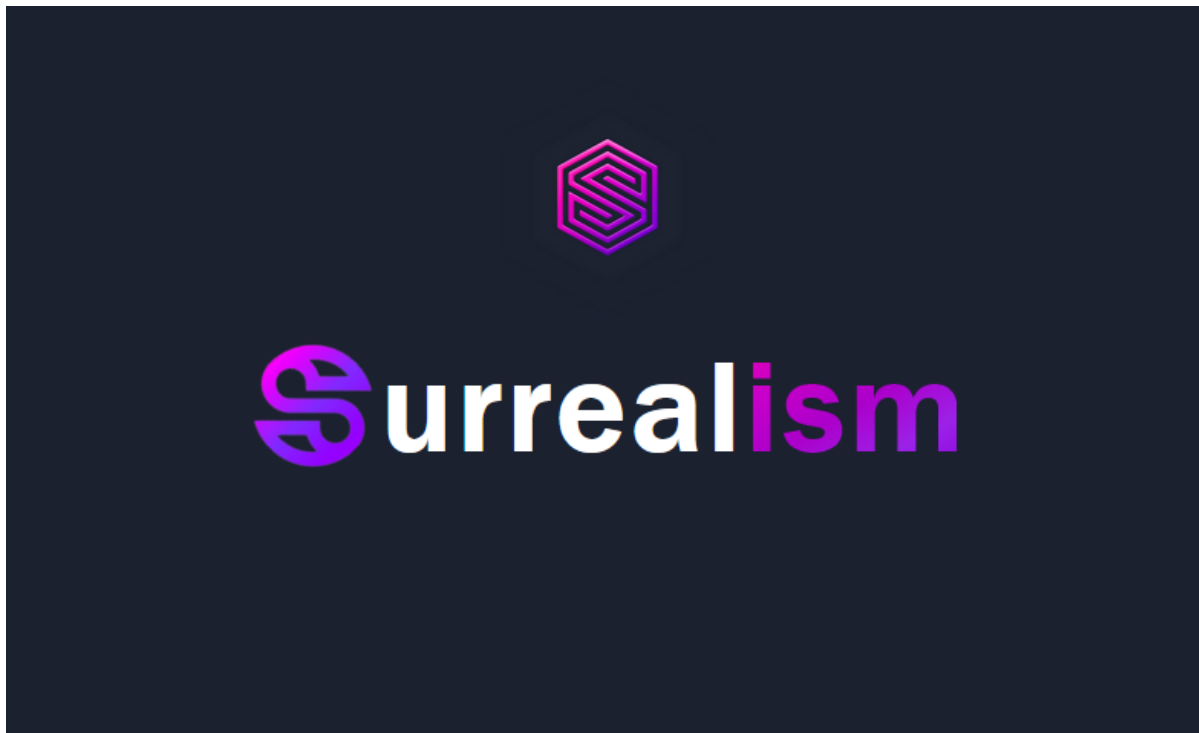


# SurrealismUI

---

- author: [syf20020816@outlook.com](mailto:syf20020816@outlook.com)
- createDate: 20230908
- updateDate: 20230910
- version: 0.1.0
- email: [syf20020816@outlook.com](mailto:syf20020816@outlook.com)





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
-  : Recommended use

## Updates

- V0.1.0
  - Adopting Fluent2's component design style
  - Multiple default methods are provided for consumers to call (see index.slint which on the outermost side)
  - Decoupling functions and components
  - Fix some style errors
  - add `SURLink` and `SURAvatar`

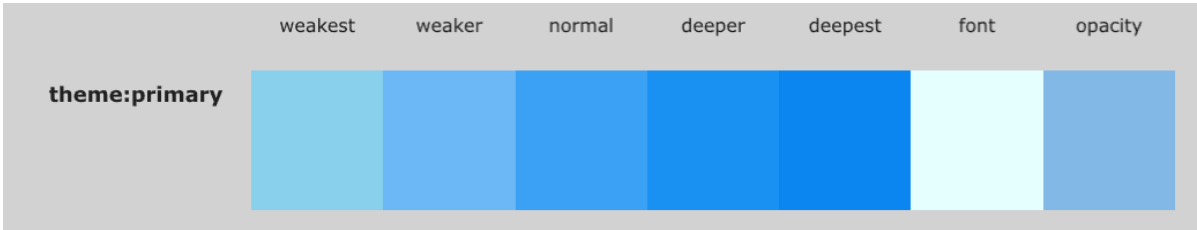
# Themes

Built in 7 theme colors in SurrealismUI

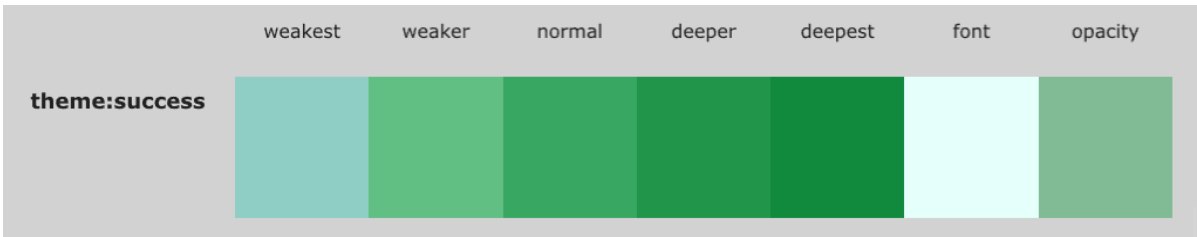
- primary
- success
- info
- warning
- error
- dark
- light

## themes-color

### primary



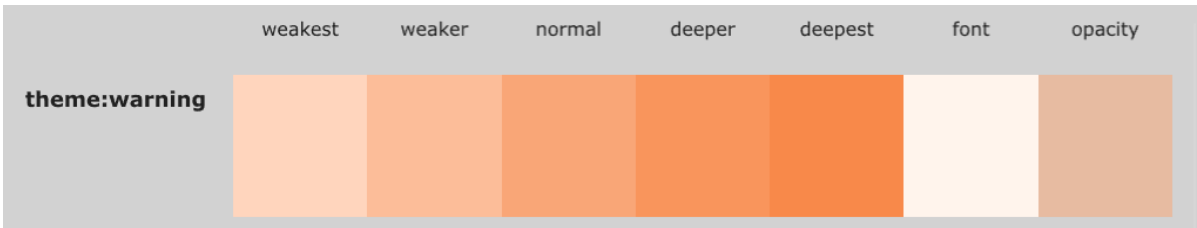
### success



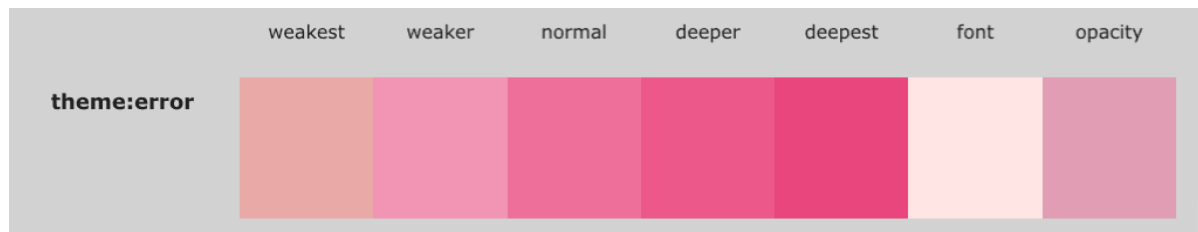
### info



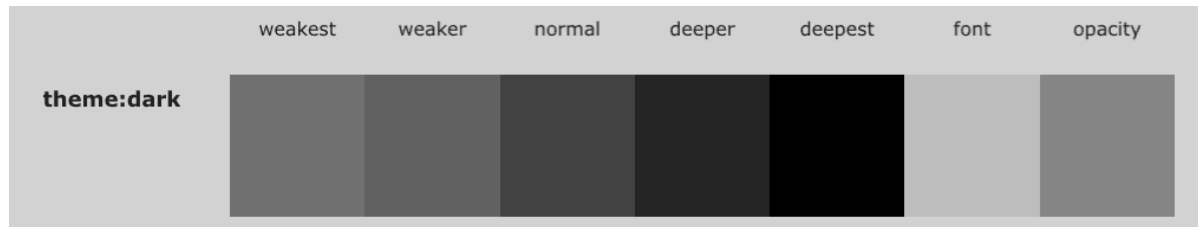
### warning



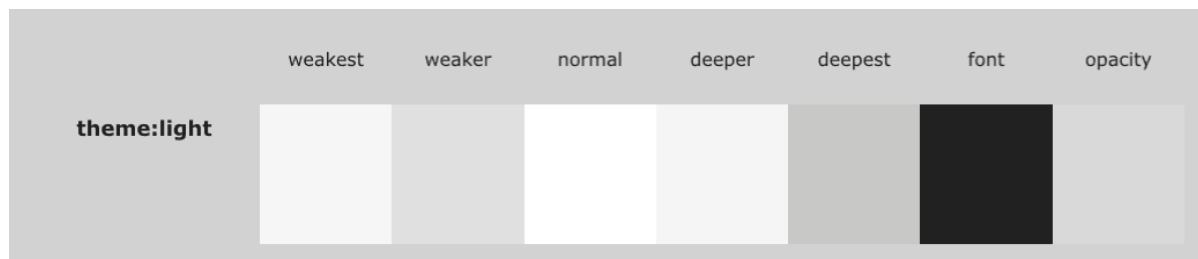
## error



## dark



## light



# 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

```
1 import {SURText} from "../../index.slint";
2 import {Themes} from "../../themes/index.slint";
3
4 component TestWindow inherits window {
5   height: 400px;
6   width: 400px;
7   background:#ddd;
8   SURText {
```

```

9      x: 100px;
10     y: 20px;
11     content: "hello world";
12   }
13   SURText {
14     x:100px;
15     y:100px;
16     theme:Themes.Error;
17   }
18
19 }

```


hello world

SURText

## SURIcon

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

### properties:

- `in property <Icons> icon` : icon types
- `out property <bool> has-hover` : has hover or not
- `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

```

1  import {SURIcon} from "../..../index.slint";
2  import {Icons,Size,Themes} from "../..../themes/index.slint";
3  export component TestIcon inherits window {
4    height: 400px;
5    width: 400px;
6    GridLayout {
7      spacing: 40px;
8      Row{
9        SURIcon{
10          height: 30px;
11          width: 30px;
12          icon: Icons.Abnormal;

```

```

13         theme: Themes.Primary;
14     }
15     SURIcon{
16         height: 30px;
17         width: 30px;
18         icon: Icons.Add;
19         theme: Themes.Success;
20     }
21     SURIcon{
22         height: 30px;
23         width: 30px;
24         icon: Icons.Baby-car-seat;
25         theme: Themes.Error;
26     }
27     SURIcon{
28         icon: Icons.T-shirt;
29         theme: Themes.Dark;
30         height: 30px;
31         width: 30px;
32     }
33     SURIcon{
34         height: 30px;
35         width: 30px;
36         icon: Icons.Baby-meal;
37         theme: Themes.Info;
38     }
39     SURIcon{
40         height: 30px;
41         width: 30px;
42         icon: Icons.Vacation;
43         theme: Themes.Warning;
44         clicked=>{
45             debug("clicked");
46             self.theme= Themes.Error;
47             self.height += 2px;
48             self.width += 2px;
49         }
50     }
51 }
52 Row{
53     SURIcon{
54         height: 30px;
55         width: 30px;
56         icon: Icons.Eagle;
57         theme: Themes.Primary;
58     }
59     SURIcon{
60         height: 30px;
61         width: 30px;
62         icon: Icons.Earth;
63         theme: Themes.Success;
64     }
65     SURIcon{
66         height: 30px;
67         width: 30px;
68         icon: Icons.waistline;

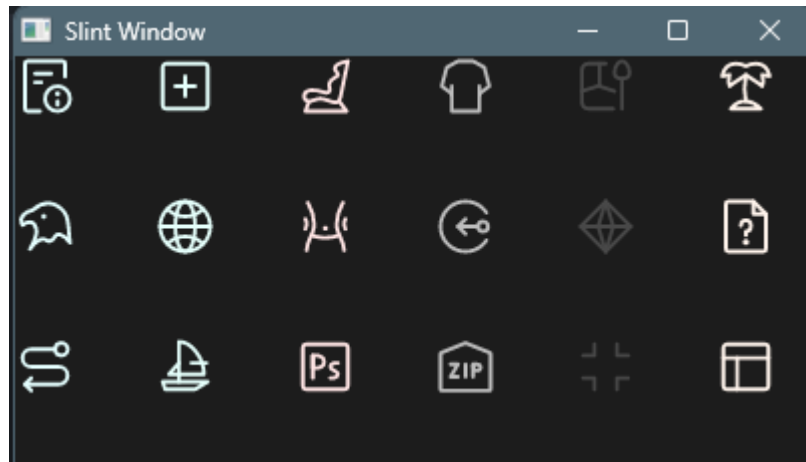
```

```
69         theme: Themes.Error;
70     }
71     SURIcon{
72         icon: Icons.Warehousing;
73         theme: Themes.Dark;
74         height: 30px;
75         width: 30px;
76     }
77     SURIcon{
78         height: 30px;
79         width: 30px;
80         icon: Icons.Quadrangular-pyramid;
81         theme: Themes.Info;
82     }
83     SURIcon{
84         height: 30px;
85         width: 30px;
86         icon: Icons.File-question;
87         theme: Themes.Warning;
88     }
89 }
90 Row{
91     SURIcon{
92         height: 30px;
93         width: 30px;
94         icon: Icons.S-turn-right;
95         theme: Themes.Primary;
96     }
97     SURIcon{
98         height: 30px;
99         width: 30px;
100        icon: Icons.Sailing;
101        theme: Themes.Success;
102    }
103    SURIcon{
104        height: 30px;
105        width: 30px;
106        icon: Icons.Adobe-photoshop;
107        theme: Themes.Error;
108    }
109    SURIcon{
110        icon: Icons.Zip;
111        theme: Themes.Dark;
112        height: 30px;
113        width: 30px;
114    }
115    SURIcon{
116        height: 30px;
117        width: 30px;
118        icon: Icons.Off-screen;
119        theme: Themes.Info;
120    }
121    SURIcon{
122        height: 30px;
123        width: 30px;
124        icon: Icons.Page;
```

```

125         theme: Themes.Warning;
126     }
127 }
128 }
129 }

```



## 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 Themes
- in property `<length>` `card-height` : card height 👍
- in property `<length>` `card-width` : card width 👍
- in property `<PaddingSize>` `padding-size` : card padding size
- in property `<Shadows>` `shadow` : card shadow type
- in property `<Borders>` `border` : card border type
- in-out property `<PaddingItem>` `card-padding` : card padding

### example

```

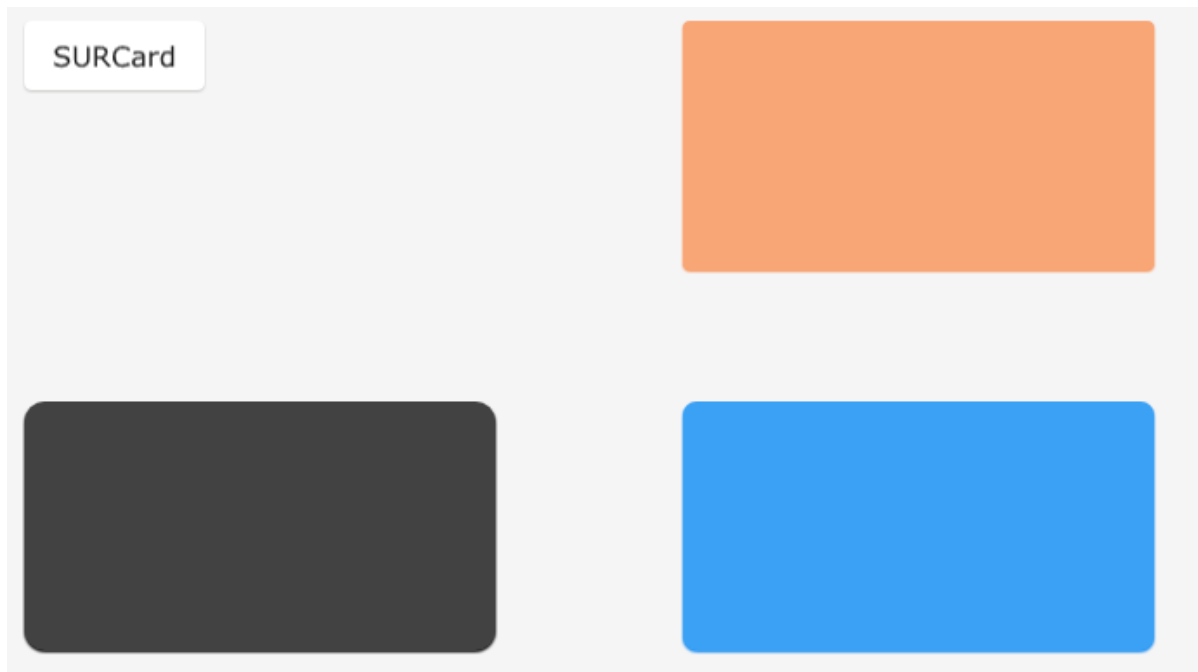
1  import {SURButton,SURCard,SURText} from "../../index.slint";
2  import {Themes,Icons} from "../../themes/index.slint";
3
4  component TestCard inherits window {
5      height: 560px;
6      width: 900px;
7      background: #F5F5F5;
8
9      SURCard {
10         x:20px;
11         y: 20px;
12         card-width:text.width;
13         text:=SURText {
14             content: "SURCard";
15         }
16     }

```

```

17  SURCard {
18      x:400px;
19      y: 20px;
20      card-width:240px;
21      card-height:120px;
22      theme: Themes.Warning;
23  }
24  SURCard {
25      x:20px;
26      y: 240px;
27      theme: Themes.Dark;
28      card-width:240px;
29      card-height:120px;
30      border: X-Large;
31  }
32
33  SURCard {
34      x:400px;
35      y: 240px;
36      theme: Themes.Primary;
37      card-width:240px;
38      card-height:120px;
39      border: Large;
40  }
41  }

```



## SURButton

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

### properties (card + icon + text)

- in property `<Icons> icon`: Button icon
- in property `<length> font-size`: button font size
- in property `<length> letter-spacing`: button letter spacing



- `in` property `<bool>` `font-italic`: button font italic
- `in` property `<int>` `font-weight`: button font weight
- `in` property `<string>` `font-family`: button font family
- `in-out` property `<string>` `content`: the content of the button;

## functions

## callbacks

- `clicked`: run if you click the button

## example

```

1  import {SURButton} from "/index.slint";
2  import {Themes,Icons} from "/themes/index.slint";
3  component TestButton inherits Window {
4      height: 400px;
5      width: 400px;
6      SURButton {
7          x: 20px;
8          y: 10px;
9
10         theme:Themes.Dark;
11         icon:Icons.Mini-sd-card;
12         clicked => {
13             self.content = "clicked"
14         }
15     }
16     SURButton {
17         x: 260px;
18         y: 10px;
19
20         content:"Save";
21         clicked => {
22             self.content = "clicked"
23         }
24     }
25     SURButton {
26         x: 20px;
27         y: 100px;
28         content:"Success";
29         theme:Themes.Success;
30
31     }
32     SURButton {
33         x: 20px;
34         y: 200px;
35         content:"Primary";
36         theme:Themes.Primary;
37     }
38     SURButton {
39         x: 20px;
40         y: 300px;
41         content:"Info";

```

```

42     theme:Themes.Info;
43 }
44 SURButton {
45     x: 200px;
46     y: 100px;
47     content:"Error?";
48     theme:Themes.Error;
49     icon:Icons.Magic-hat;
50 }
51 SURButton {
52     x: 200px;
53     y: 200px;
54
55     theme:Themes.Warning;
56 }
57 }

```



## 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

```

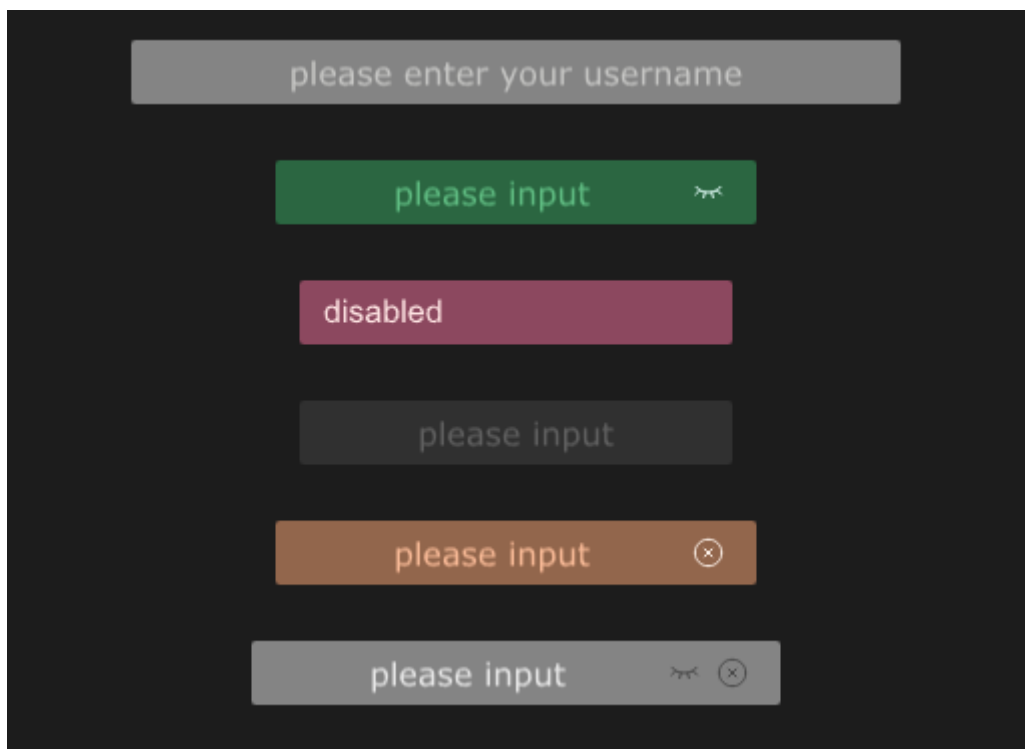
1  import {SURText,SURInput,SURButton, SURIcon} from "../..index.slint";
2  import {Themes} from "../..themes/index.slint";
3
4  export component TestInput inherits window {
5      height: 500px;
6      width: 600px;
7
8      SURInput{
9          y: 20px;
10         placeholder : "please enter your username";
11         input-width: 360px;
12         accepted(res)=>{
13             debug("content in input:" + res);
14         }
15         changed(change-res)=>{
16             debug(change-res);
17         }
18     }
19
20     w:=SURInput{
21         y: 80px;
22         theme: Themes.Success;
23         type: InputType.password;
24         password: true;
25     }
26
27     SURInput{
28         y: 140px;
29         theme: Themes.Error;
30         disabled: true;

```

```

30     content:"disabled";
31   }
32   SURInput{
33     y: 200px;
34     theme:Themes.Dark;
35   }
36
37   SURInput{
38     y: 260px;
39     theme:Themes.Warning;
40     clearable:true;
41   }
42   SURInput{
43     y: 320px;
44     theme:Themes.Info;
45     type:InputType.password;
46     clearable:true;
47     password:true;
48   }
49
50 }

```



## 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 🚫
- `public function full()` : star all 👍
- `public function clear()` : no star 👍
- `public function add-one()` : add one star 👍
- `public function add-half()` : add half stars 👍

## callbacks

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

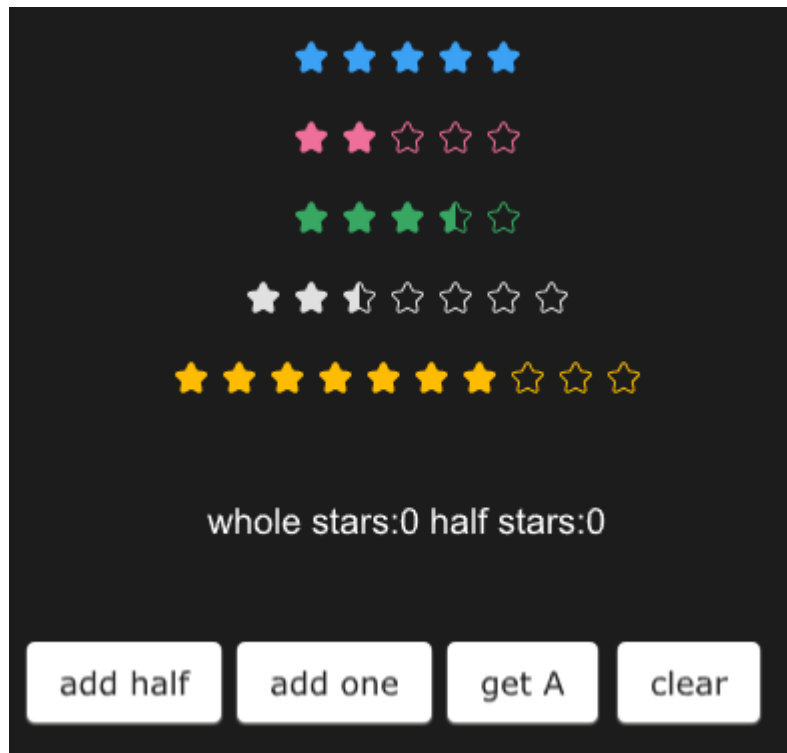
## example

```
1 import {SURStar,SURButton} from "../../index.slint";
2 import {Themes,Icons} from "../../themes/index.slint";
3
4 component TestWindow inherits window {
5     height: 400px;
6     width: 400px;
7     SURStar {
8         y: 20px;
9     }
10    hs:=SURStar {
11        score: 2.2;
12        y: 60px;
13        theme: Error;
14    }
15
16    SURButton {
17        y: 320px;
18        x:10px;
19        content: "add half";
20        clicked => {
21            hs.add-half();
22        }
23    }
24    SURStar {
25        score : 3.8;
26        disabled: true;
27        y: 100px;
28        theme: Success;
29    }
30    os:=SURStar {
31        max-score : 7;
32        score : 2.8;
33        y: 140px;
34        theme: Info;
35    }
36    SURButton {
```

```

37     y: 320px;
38     x: 115px;
39     content: "add one";
40     clicked => {
41         os.add-one();
42     }
43 }
44 fs:=SURStar {
45     max-score : 10;
46     score : 7.2;
47     y: 180px;
48     no-theme:true;
49     clicked(whole,half) => {
50         t.n = whole;
51         t.m = half;
52     }
53 }
54 SURButton {
55     y: 320px;
56     x: 220px;
57     content: "get A";
58     clicked => {
59         fs.full();
60     }
61 }
62 SURButton {
63     y: 320px;
64     x: 305px;
65     content: "clear";
66     clicked => {
67         fs.clear();
68     }
69 }
70 t:=Text{
71     y: 250px;
72     font-size: 18px;
73     in-out property <int> n;
74     in-out property <int> m;
75     text: "whole stars:"+ n + " half stars:" + m;
76 }
77 }

```



## 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

```

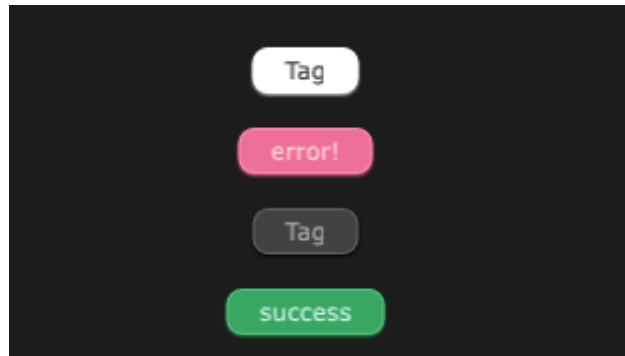
1  import {SURTag} from "../../index.slint";
2  import {Themes,Icons} from "../../themes/index.slint";
3
4  component Testwindow inherits window {
5      height: 400px;
6      width: 400px;
7      SURTag {
8          y: 40px;
9      }
10     SURTag {
11         content:"error!";
12         y:80px;
13         theme:Themes.Error;
14     }
15     SURTag {
16         y:120px;

```

```

17     theme:Themes.Dark;
18     clicked=>{
19         self.font-color= #ddff00;
20     }
21 }
22 SURTag {
23     content:"success";
24     y:160px;
25
26     theme:Themes.Success;
27 }
28 }


```



## SURHeader

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

### properties

- in property `<length> spacing` : spacing of the header 
- 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

```

1  import {SURHeader,Route} from "../../index.slint";
2  import {Themes,Icons} from "../../themes/index.slint";
3
4  component Testwindow inherits window {
5      height: 400px;
6      width: 660px;
7      SURHeader {
8          x:10px;
9          y: 40px;
10     }
11     SURHeader {

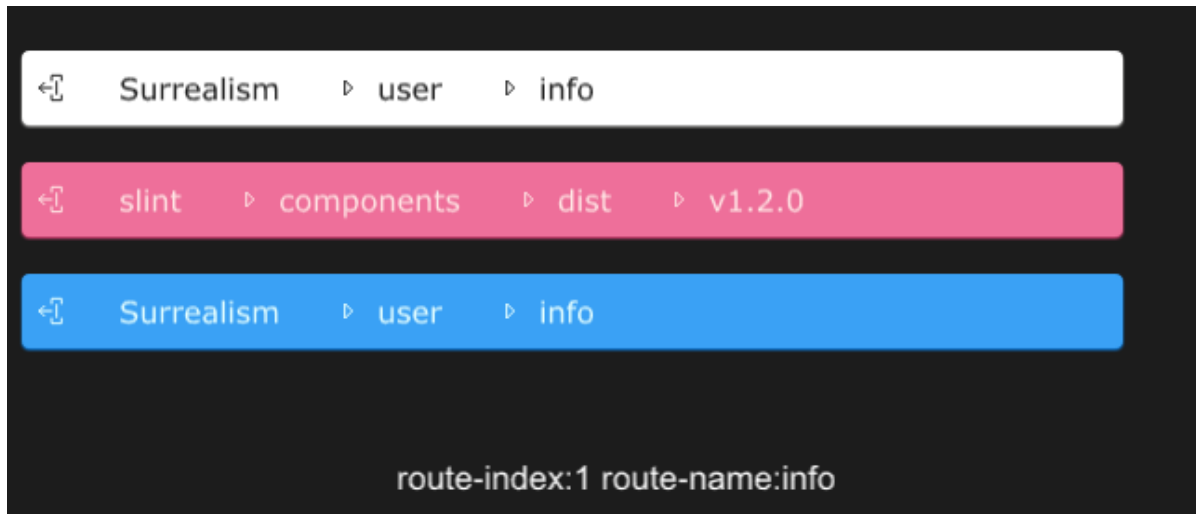
```



```

12     x:10px;
13     y: 100px;
14     theme: Error;
15     route:{
16         home:"slint",routes:["components","dist","v1.2.0"]
17     };
18 }
19 SURHeader {
20     x:10px;
21     y: 160px;
22     theme: Primary;
23     to(index,route)=>{
24         txt.name = route;
25         txt.index = index;
26     }
27     back=>{
28         txt.name = "back";
29     }
30 }
31 txt:=Text{
32     y: 260px;
33     font-size: 18px;
34     in-out property <int> index;
35     in-out property <string> name;
36     text: "route-index:" + index + " route-name:" + name;
37 }
38 }

```



## 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

```
1 import {SURTable,SURTableColumn} from "../../index.slint";
2 import {Themes,Icons} from "../../themes/index.slint";
3
4 export component TestTable inherits window {
5     height: 500px;
6     width: 500px;
7     t1:=SURTable {
8
9         y: 10px;
10        // you can use this way to get height
11        // it depends on how many datas in column
```

```

12     height: col1.get-height();
13     width: 300px;
14
15     col1:=SURTableColumn {
16         border:false;
17         theme:Themes.Error;
18         width: 100px;
19         name:"id";
20         // row-height:60px;
21         datas: ["101","102","103"];
22     }
23     SURTableColumn {
24         theme:Themes.Error;
25         width: 100px;
26         name:"name";
27         datas: ["Mat","Jarry","Kaven"];
28     }
29     SURTableColumn {
30         theme:Themes.Error;
31         width: 100px;
32         name:"age";
33         datas: ["16","23","18"];
34     }
35 }
36 t2:=SURTable {
37
38     y: t1.height + 40px;
39     // you can use this way to get height
40     // it depends on how many datas in column
41     height: tcol1.get-height();
42     width: 350px;
43
44     tcol1:=SURTableColumn {
45         border:false;
46         theme:Themes.Primary;
47         width: 100px;
48         name:"id";
49         // row-height:60px;
50         datas: ["101","102","103"];
51     }
52     SURTableColumn {
53         theme:Themes.Primary;
54         width: 100px;
55         name:"name";
56         datas: ["Mat","Jarry","Kaven"];
57     }
58     SURTableColumn {
59         theme:Themes.Primary;
60         width: 150px;
61         name:"Operations";
62         // cheat datas
63         datas: [" "," "," "];
64         operation-enabled:true;
65     }
66 }
67 }

```

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

```

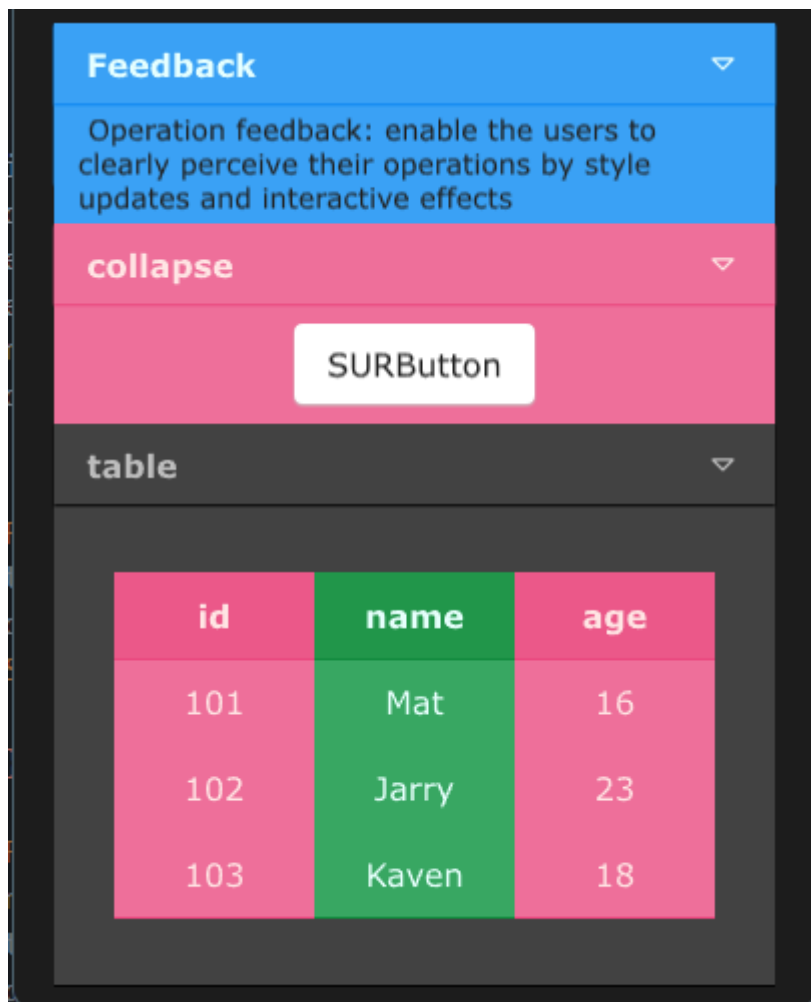
1  import {SURCollapse,SURCollapseItem,SURButton,SURTable,SURTableColumn} from
    "../..../index.slint";
2  import {Themes,Icons} from "../..../themes/index.slint";
3
4
5  component TestWindow inherits window {
6      height: 500px;
7      width: 400px;
8      SURCollapse {
9          y: 10px;
10         // you can set 0 , it has no impact
11         // recommend use the following way
12         height: item1.get-height() * 2;
13         width: 360px;
14         theme: Primary;
15         item1:=SURCollapseItem {
16             name:"Feedback";
17             detail:" operation feedback: enable the users to clearly perceive their
operations by style updates and interactive effects";
18
19         }
20         SURCollapseItem {
21             theme: Themes.Error;
22             define:true;
23             SURButton {
24
25             }
26         }
27         SURCollapseItem {
28             name:"table";
29             theme: Themes.Dark;
30             define:true;
31             item-height:280px;
32             SURTable {
33
34                 height: col1.get-height();
35                 width: 300px;
36                 col1:=SURTableColumn {
37                     border:false;
38                     theme:Themes.Error;
39                     width: 100px;
40                     name:"id";

```

```

41 // row-height:60px;
42 datas: ["101","102","103"];
43 }
44 SURTableColumn {
45   theme:Themes.Success;
46   width: 100px;
47   name:"name";
48   datas: ["Mat","Jarry","Kaven"];
49 }
50 SURTableColumn {
51   theme:Themes.Error;
52   width: 100px;
53   name:"age";
54   datas: ["16","23","18"];
55 }
56 }
57 }
58 }
59 }

```



## SURResult

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

## properties

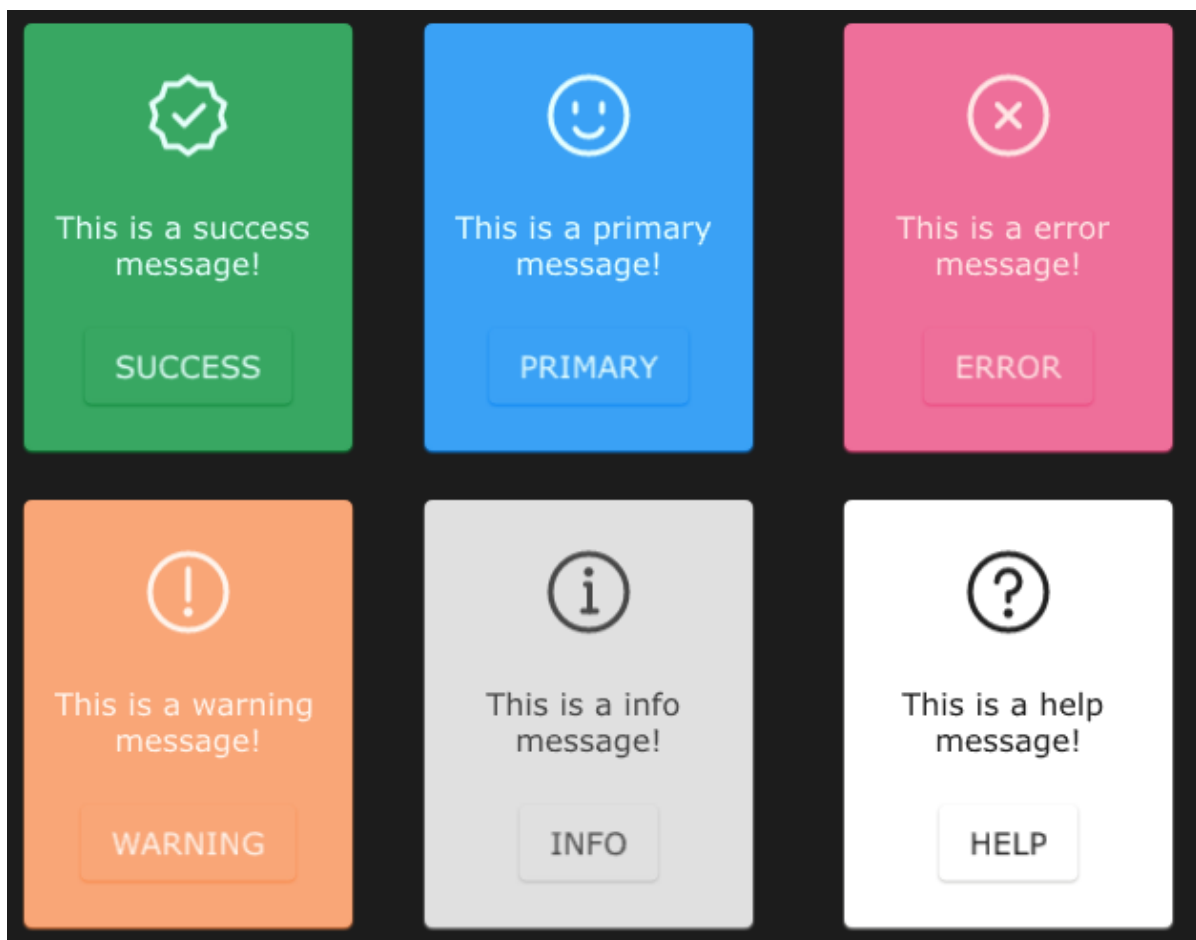
- `in` property `<length> icon-size`: icon size
- `in-out` property `<string> btn`: the content of the button
- `in-out` property `<string> content`: content of the result
- `in` property `<ResType> res-type`: Result type
- `in-out` property `<Icons> icon`: Icon of the result

## functions

### callbacks

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

```
1 import {SURResult, ResType} from "../../index.slint";
2 import {Themes, Icons} from "../../themes/index.slint";
3
4 export component TestResult inherits window {
5     height: 500px;
6     width: 800px;
7     SURResult {
8         x: 10px;
9         y: 10px;
10    }
11    SURResult {
12        x: 220px;
13        y: 10px;
14        res-type: ResType.Primary;
15    }
16    SURResult {
17        x: 220px;
18        y: 260px;
19        res-type: ResType.Info;
20    }
21    SURResult {
22        x: 10px;
23        y: 260px;
24        res-type: ResType.Warning;
25    }
26
27    SURResult {
28        x: 440px;
29        y: 10px;
30        res-type: ResType.Error;
31    }
32    SURResult {
33        x: 440px;
34        y: 260px;
35        res-type: ResType.Help;
36    }
37 }
```



## 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

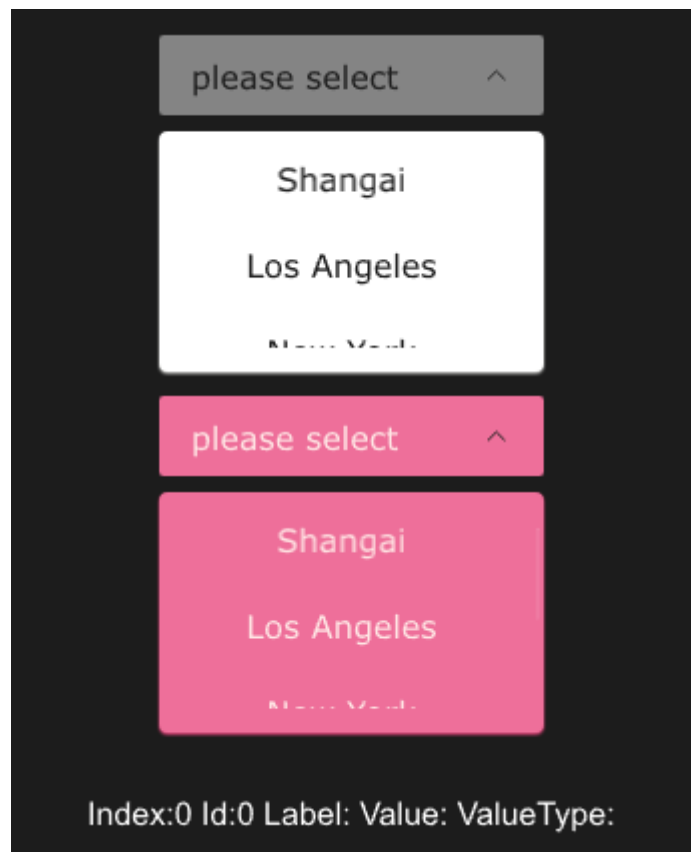


## callbacks

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

## example

```
1 import {SURSelect,valueType} from "../../index.slint";
2 import {Themes,Icons} from "../../themes/index.slint";
3
4 component TestWindow inherits window {
5     height: 440px;
6     width: 400px;
7     SURSelect {
8         y: 20px;
9         ranges-string: [
10             {id:0,label:"Shangai",value:"s01"},
11             {id:1,label:"Los Angeles",value:"l02"},
12             {id:2,label:"New York",value:"n03"},
13             {id:3,label:"Hong Kong",value:"h04"},
14         ];
15     }
16     SURSelect {
17         y: 200px;
18         theme: Error;
19         ranges-float: [
20             {id:0,label:"Shangai",value:0.1},
21             {id:1,label:"Los Angeles",value:0.2},
22             {id:2,label:"New York",value:0.3},
23             {id:3,label:"Hong Kong",value:0.4},
24         ];
25         changed(index,id,label,value,value-type)=>{
26             if(value-type==ValueType.String){
27                 t.vt = "string";
28             }else if(value-type==ValueType.Float){
29                 t.vt = "float"
30             }else{
31                 t.vt = "int"
32             }
33             t.index = index;
34             t.id = id;
35             t.label = label;
36             t.value = value;
37         }
38     }
39     t:=Text{
40         y: 400px;
41         font-size: 16px;
42         in-out property <int> index;
43         in-out property <int> id;
44         in-out property <string> label;
45         in-out property <string> vt;
46         in-out property <string> value;
47         text: @tr("Index:{} Id:{} Label:{} Value:{} ValueType:
48         {}",index,id,label,value,vt);
49     }
```



## SURLink

SURLink is commonly used to represent text connections or sharing

### properties

- `in property <Icons> icon` : share icon you can use whatever you want
- `in property <bool> funny` : Easter egg just funny

### callbacks

- `callback clicked()` : run if you click share icon

### example

```

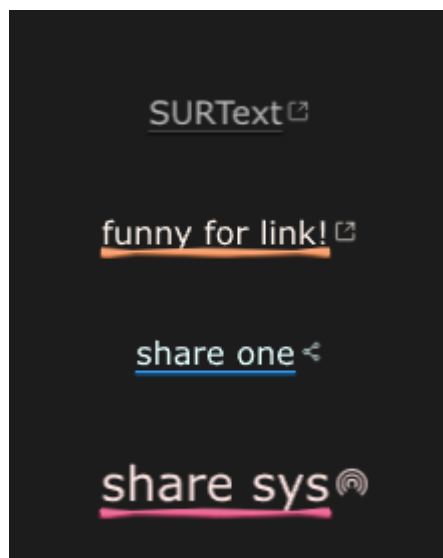
1  import {SURLink} from "../../index.slint";
2  import {Themes,Icons} from "../../themes/index.slint";
3
4  component TestWindow inherits window {
5      height: 420px;
6      width: 400px;
7
8      SURLink {
9          y: 100px;
10         theme: Dark;
11     }
12     SURLink {
13         y: 160px;
14         funny:true;

```

```

15     theme: Warning;
16     content: "funny for link!";
17 }
18 SURLink {
19     y: 220px;
20     theme: Primary;
21     icon: Icons.Share-one;
22     content: "share one";
23 }
24 SURLink {
25     y: 280px;
26     funny:true;
27     theme: Error;
28     icon : Icons.Share-sys;
29     font-size: 24px;
30     content: "share sys";
31     clicked=>{
32         debug("share sys!")
33     }
34 }
35 }

```



## SURAvatar

SURAvatar is a avatar component that defaults to Icons.Avatar when there are no images available

### properties

- in property <length> avatar-size : avatar size
- in property <image> avatar : avatar image

### example

```

1  import {SURAvatar} from "../../index.slint";
2  import {Themes,Icons,ROOT-STYLE} from "../../themes/index.slint";
3
4  component TestWindow inherits window {
5      height: 400px;
6      width: 400px;

```

```

7   background: #F5F5F5;
8   SURAvatar {
9     x: 20px;
10    y: 100px;
11  }
12  SURAvatar {
13    x:20px;
14    y: 200px;
15    avatar-size : ROOT-STYLE.sur-size.small * 2;
16    padding-size : Small;
17    theme: Primary;
18  }
19  SURAvatar {
20    x: 200px;
21    y: 100px;
22    theme: Warning;
23  }
24  SURAvatar {
25    x: 200px;
26    y: 200px;
27    avatar-size : ROOT-STYLE.sur-size.small * 2;
28    padding-size : Small;
29    theme: Error;
30  }
31  SURAvatar {
32    y: 300px;
33    avatar-size : ROOT-STYLE.sur-size.large * 2;
34    padding-size : Large;
35    theme: Dark;
36    avatar:@image-url("../..//README/imgs/logo.png");
37  }
38
39 }

```

