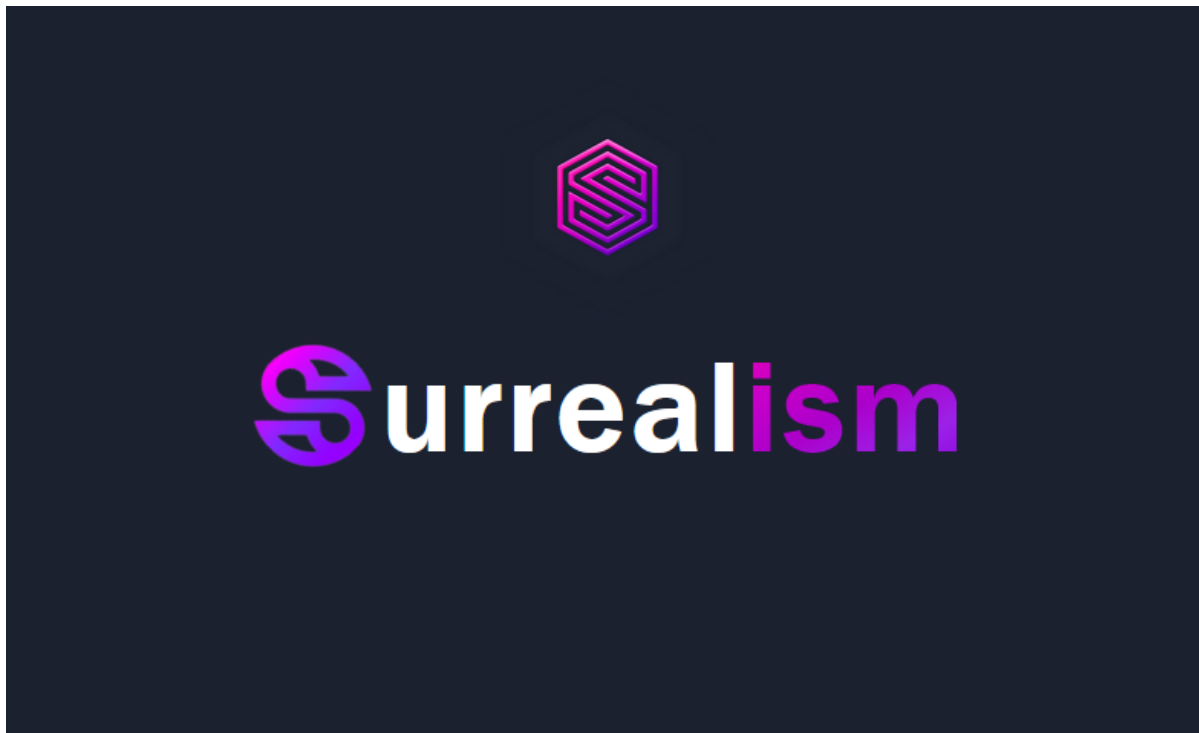# SurrealismUI

- author：syf20020816@outlook.com

- createDate：20230908

- updateDate：20230930

- version：0.1.5

- email：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

- V0.1.5
  - add `SURMenu`
  - enhance `SURTip` (the location of the tip can be changed now and you can show it with hover！)
- V0.1.4
  - add `SURTip`
  - add `SURLoading`
  - add `SURDialog`
- V0.1.3

- add `SURBadge`
      - add `Progress`
      - add `Persona`
  - V0.1.2
      - rebuild components (have `SURIcon`)
      - rebuild `SURIcon`
      - rebuild file structure
      - solve memery overflow issue
      - use minimize import principle (remove inner loop to judge component show!) ❗
      - test use Rust ✅
  - V0.1.1
      - add `SURRadio`
      - add `SURDivider`
      - add `SURCollection`
      - add `SURPopup`
  - 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

```
1              default
2    ———————————————————————————————
3    |  logic control layer (Rust|C++)  |
4    ———————————————————————————————
5                  ⇕
6    ———————————————————————————————
```

```
|     UI layer (write components)    |
--------------------------------------

            SurrealismUI
--------------------------------------
|   logic control layer (Rust|C++)   |
--------------------------------------
                ⇕
--------------------------------------
|       UI Styles Wrapper layer      |   <-- What SurrealismUI do , see ①
--------------------------------------
|    UI layer (write components)     |
--------------------------------------
```

①: define a lot replaceable theme styles and binding styles use theme property , can be customized in slint file or logic control layer , means: all system components are wrapped (Customizing themes in third-party component libraries is very affordable as it acts on the UI layer. SLINT is like an integration of HTML and CSS, so I use this way)(By binding global singleton variables to styles, any component that uses variables can change styles simultaneously)

```
               System support (like iced)
--------------------------------------      ----------------
|             logic control          | -->  | Theme::Light |
--------------------------------------      ----------------
|               UI layer             |              ↓
--------------------------------------       |------------|
                     ↑              ↓              ↓
                 import    ← Light_Theme Styles   Dark_Theme Styles
```
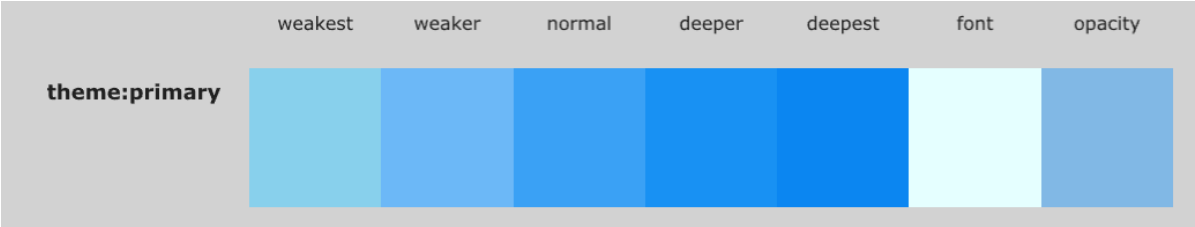
Diff
Slint differs from other GUI frameworks in that the UI layer is completed through. slint, which I believe is good and brings many advantages (compatibility with different platforms, instant preview, maintainability, parallel development, etc.). But this also leads to SLIT being unable to easily customize the theme of the component. Theme customization and switching are dynamic to static processes, which require a lot of logical processing, and this is also same as (HTML+CSS+js | ts)
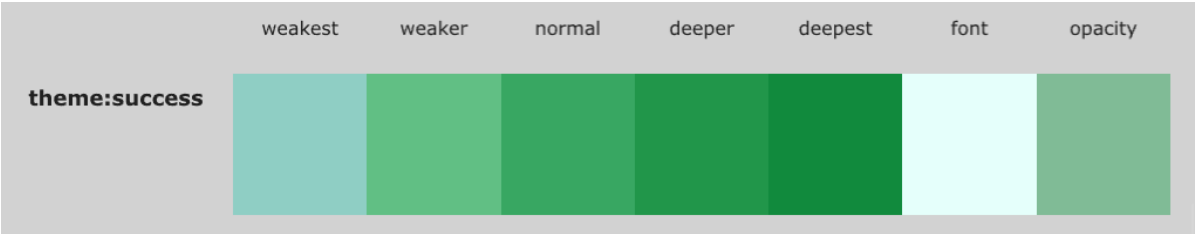
Slint be careful
Slint's work on topic definition will simultaneously affect the built-in components currently provided and other languages' API. Although this feature may seem simple, it may bring significant risks, which I believe need to be weighed and considered. Is this necessary? Because for third-party component libraries, although the workload of helping users define themes is large, it is not complex. As the author of SurrealismUI, it is evident that we have successfully implemented the definition of themes in static slint language and can be modified at the logical level. Users customize themes through static file overwriting.
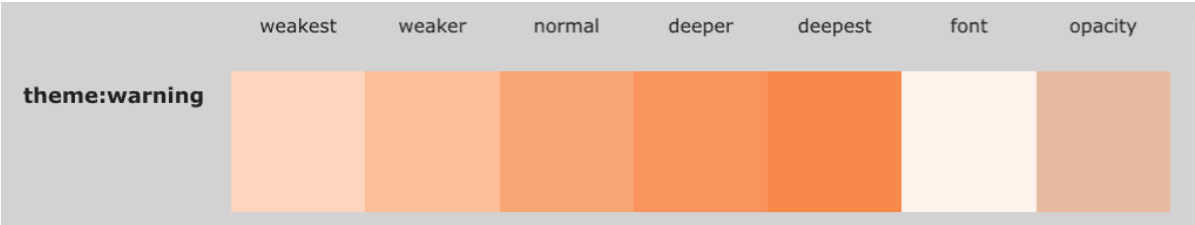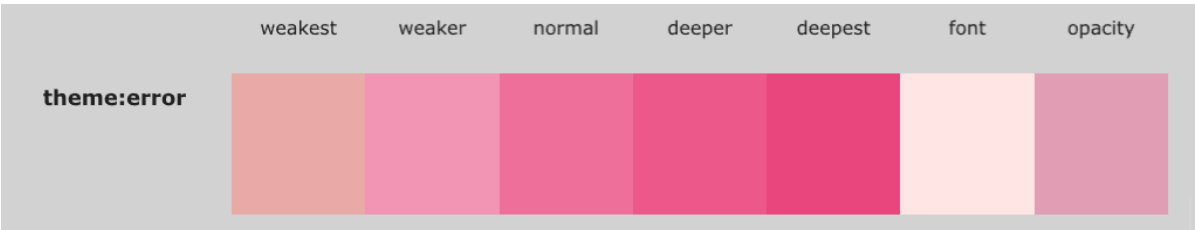
primary

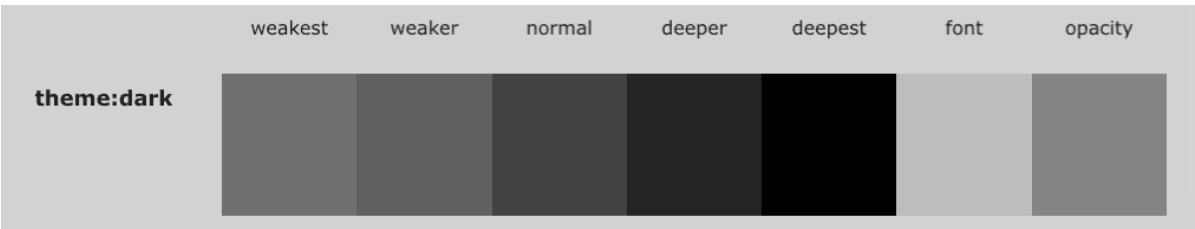|	weakest	weaker	normal	deeper	deepest	font	opacity
theme:primary							

success

	weakest	weaker	normal	deeper	deepest	font	opacity
theme:success							

info

	weakest	weaker	normal	deeper	deepest	font	opacity
theme:info							

warning

	weakest	weaker	normal	deeper	deepest	font	opacity
theme:warning							

error

	weakest	weaker	normal	deeper	deepest	font	opacity
theme:error							

dark

	weakest	weaker	normal	deeper	deepest	font	opacity
theme:dark							

light

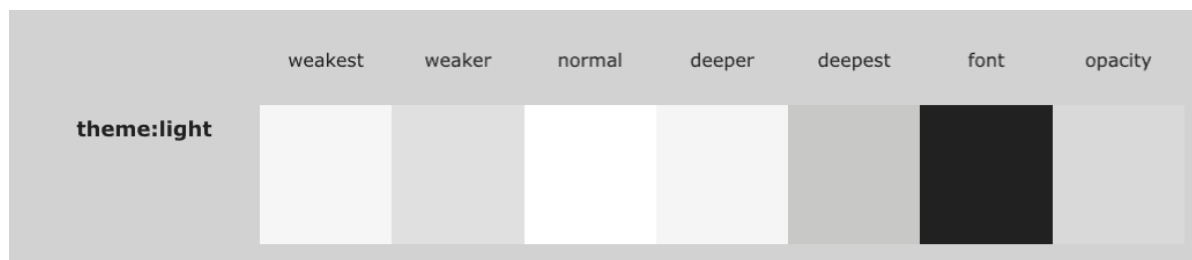|	weakest	weaker	normal	deeper	deepest	font	opacity
theme:light							

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:#fff;
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

SURIcon

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

properties:

- `in property <image> 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 {IconSources,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: @image-url("../../icons/sd-card.svg");
13          theme: Themes.Primary;
14        }
15        SURIcon{
16          height: 30px;
17          width: 30px;
18          icon: @image-url("../../icons/add-computer.svg");
19          theme: Themes.Success;
20        }
21        SURIcon{
22          height: 30px;
23          width: 30px;
24          icon: @image-url("../../icons/yep.svg");
```

```
25            theme: Themes.Error;
26          }
27        SURIcon{
28          icon: @image-url("../../icons/t-shirt.svg");
29          theme: Themes.Dark;
30          height: 30px;
31          width: 30px;
32        }
33        SURIcon{
34          height: 30px;
35          width: 30px;
36          icon: @image-url("../../icons/video-conference.svg");
37          theme: Themes.Info;
38        }
39        SURIcon{
40          height: 30px;
41          width: 30px;
42          icon:@image-url("../../icons/vacation.svg");
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: @image-url("../../icons/cake-five.svg");
57          theme: Themes.Primary;
58        }
59        SURIcon{
60          height: 30px;
61          width: 30px;
62          icon: @image-url("../../icons/label.svg");
63          theme: Themes.Success;
64        }
65        SURIcon{
66          height: 30px;
67          width: 30px;
68          icon: @image-url("../../icons/wifi.svg");
69          theme: Themes.Error;
70        }
71        SURIcon{
72          icon: @image-url("../../icons/wallet-one.svg");
73          theme: Themes.Dark;
74          height: 30px;
75          width: 30px;
76        }
77        SURIcon{
78          height: 30px;
79          width: 30px;
80          icon: @image-url("../../icons/game-console.svg");
```

```
 81          theme: Themes.Info;
 82        }
 83      SURIcon{
 84        height: 30px;
 85        width: 30px;
 86        icon: @image-url("../../icons/qiyehao.svg");
 87        theme: Themes.Warning;
 88      }
 89    }
 90    Row{
 91      SURIcon{
 92        height: 30px;
 93        width: 30px;
 94        icon: @image-url("../../icons/scanning-two.svg");
 95        theme: Themes.Primary;
 96      }
 97      SURIcon{
 98        height: 30px;
 99        width: 30px;
100        icon: @image-url("../../icons/oceanengine.svg");
101        theme: Themes.Success;
102      }
103      SURIcon{
104        height: 30px;
105        width: 30px;
106        icon: @image-url("../../icons/zoom-internal.svg");
107        theme: Themes.Error;
108      }
109      SURIcon{
110        icon: @image-url("../../icons/zip.svg");
111        theme: Themes.Dark;
112        height: 30px;
113        width: 30px;
114      }
115      SURIcon{
116        height: 30px;
117        width: 30px;
118        icon: @image-url("../../icons/f-eight-key.svg");
119        theme: Themes.Info;
120      }
121      SURIcon{
122        height: 30px;
123        width: 30px;
124        icon: @image-url("../../icons/pacifier.svg");
125        theme: Themes.Warning;
126      }
127    }
128  }
129 }
```
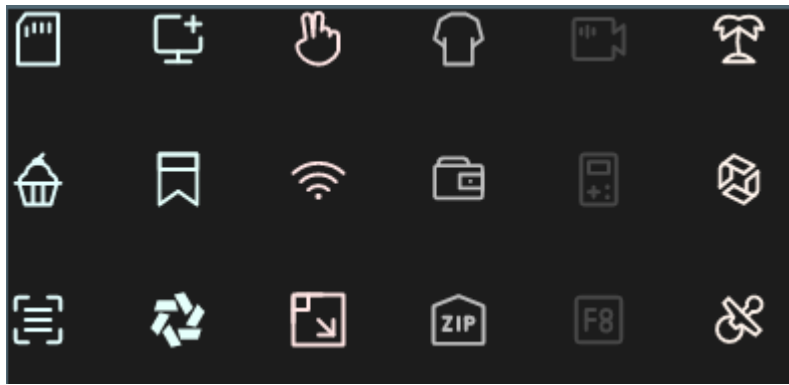
SURCard

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} 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  }
```
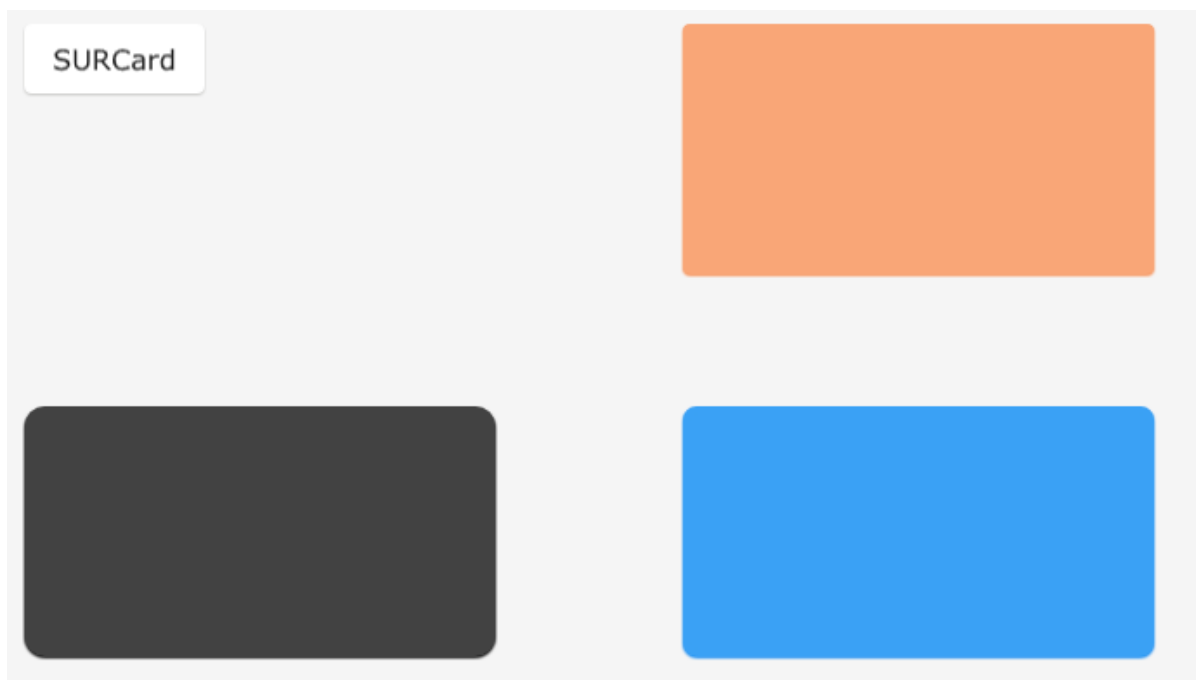

SURButton

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

properties (card + icon + text)

- `in property <image> 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,IconSources} 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:@image-url("../../icons/safe-retrieval.svg");
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:@image-url("../../icons/magic-hat.svg");
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 <image> 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    SURInput{
27      y: 140px;
28      theme:Themes.Error;
29      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  }
```
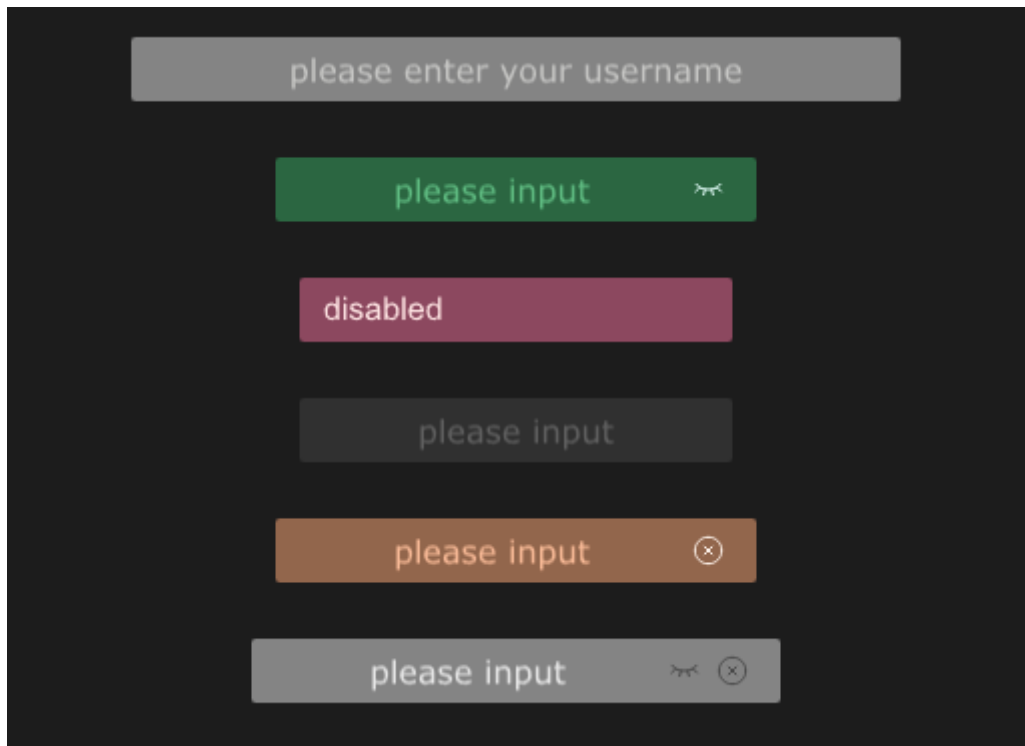

SURStar

 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 👍
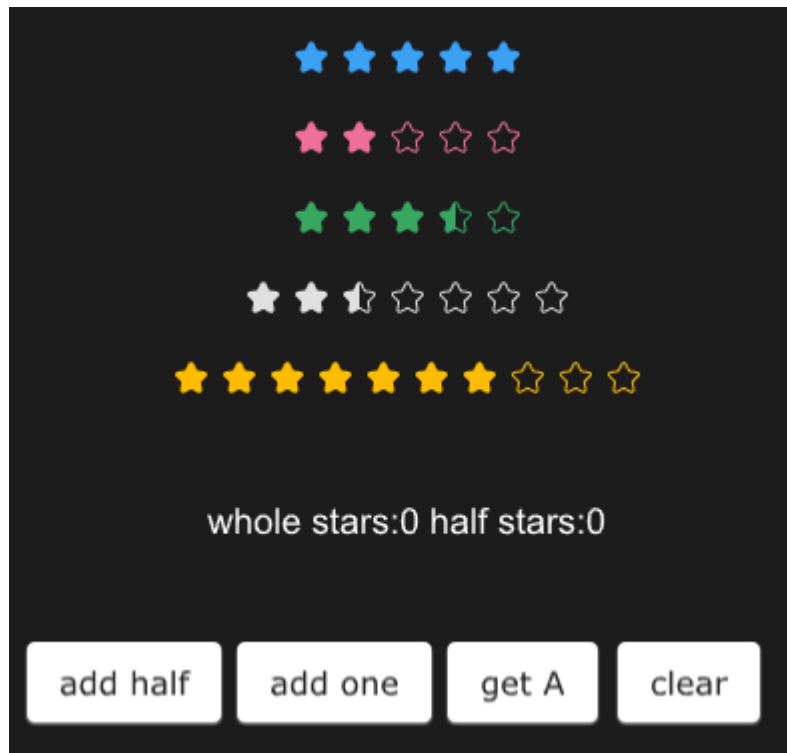- `public function add-half()` : add half stars 👍

callbacks

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

example

```
 1   import {SURStar,SURButton} from "../../index.slint";
 2   import {Themes,IconSources} 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 {
```

```
     y: 320px;
     x: 115px;
     content: "add one";
     clicked => {
       os.add-one();
     }
   }
   fs:=SURStar {
     max-score : 10;
     score : 7.2;
     y: 180px;
     no-theme:true;
     clicked(whole,half) => {
       t.n = whole;
       t.m = half;
     }
   }
   SURButton {
     y: 320px;
     x: 220px;
     content: "get A";
     clicked => {
       fs.full();
     }
   }
   SURButton {
     y: 320px;
     x: 305px;
     content: "clear";
     clicked => {
       fs.clear();
     }
   }
   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

```
1  import {SURTag} from "../../index.slint";
2  import {Themes} 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  }
```
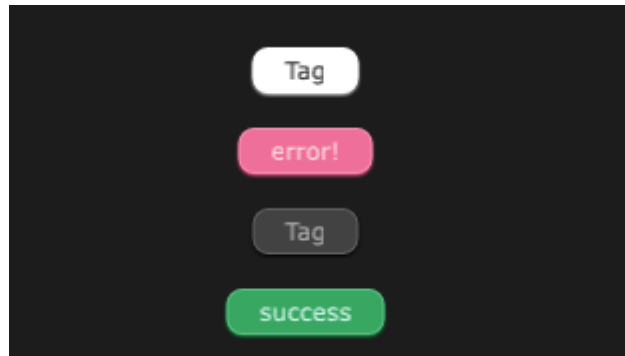

SURHeader

 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} 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  }
```
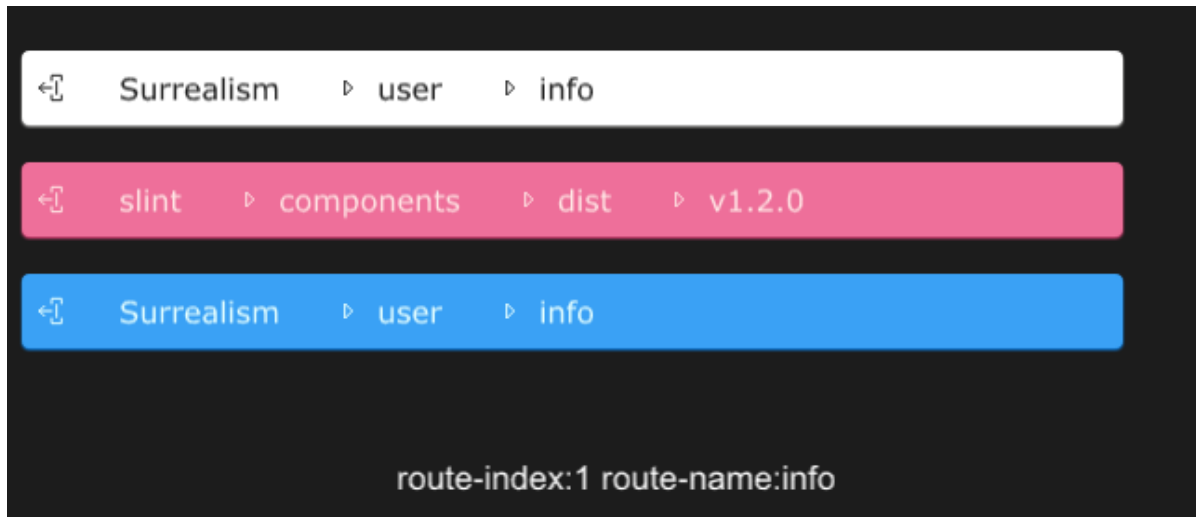

SURTable

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} 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  }
```

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,IconSources} 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 }
```
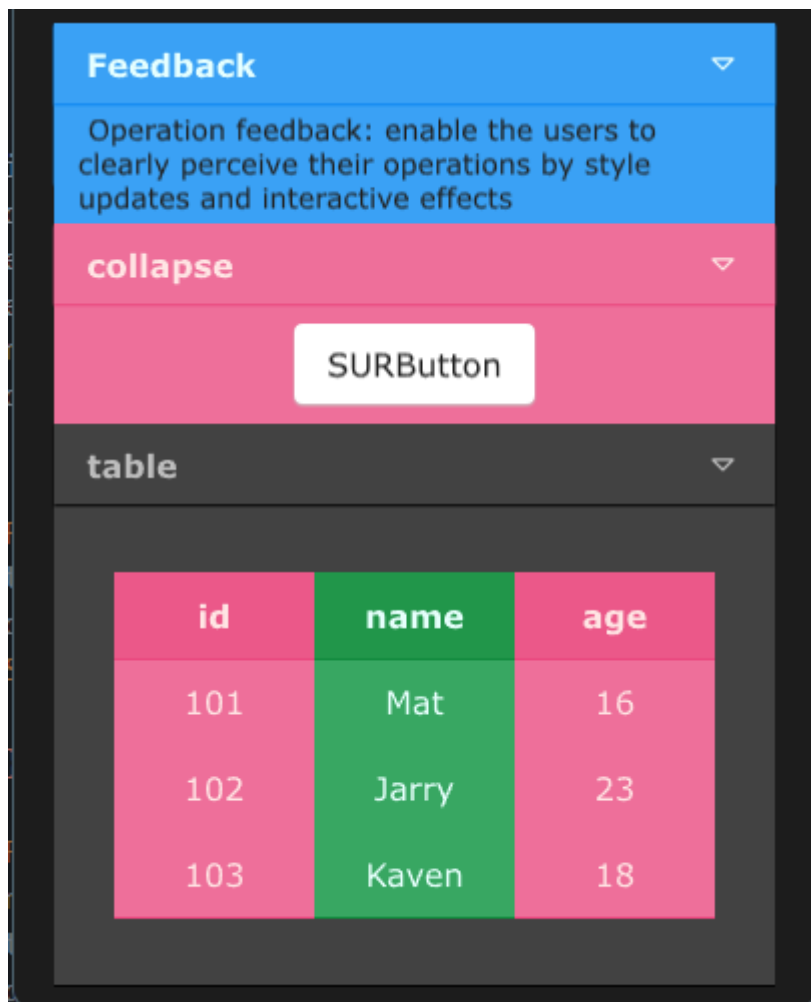

SURResult

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

```slint
1   import {SURResult,ResType} from "../../index.slint";
2   import {Themes} 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  }
```
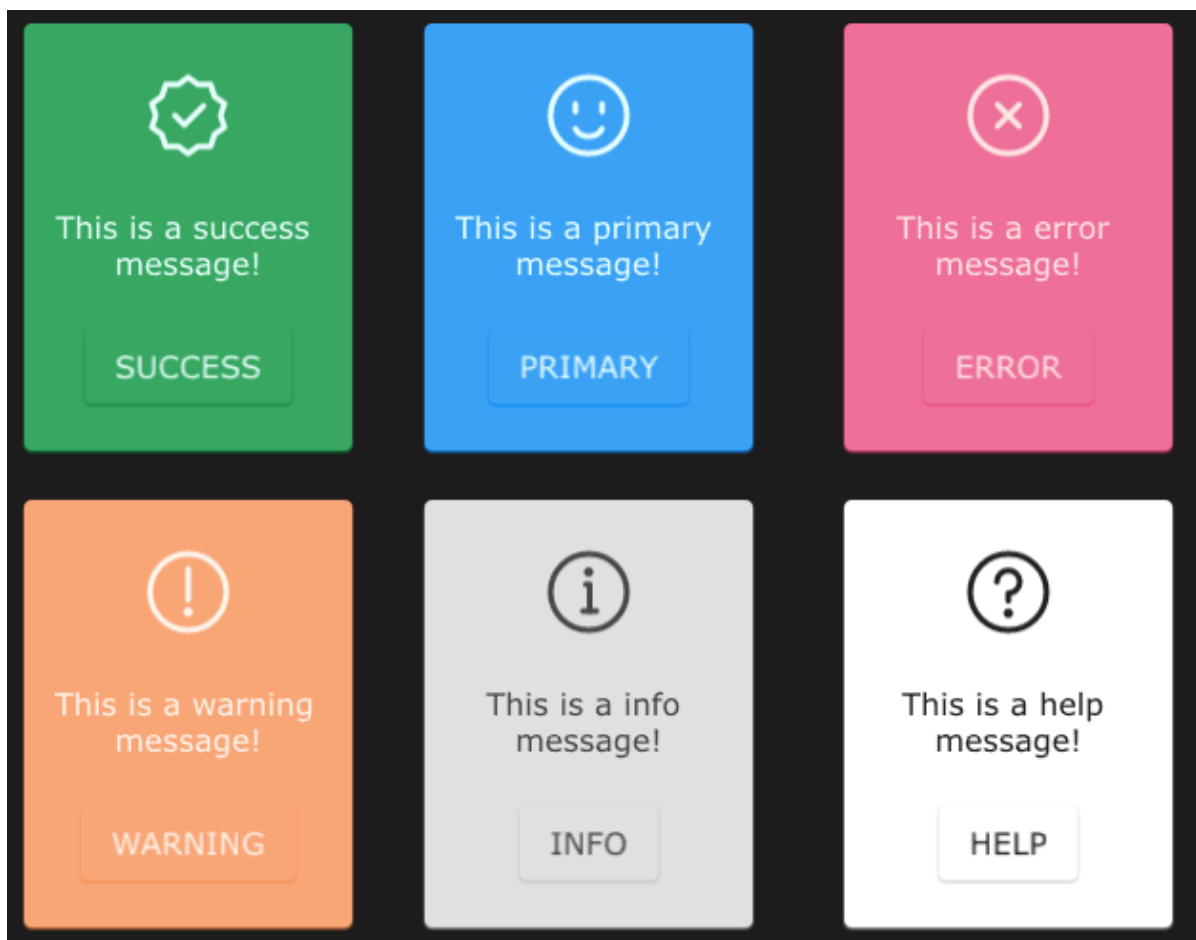
SURSelect

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} 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:
    {}",index,id,label,value,vt);
48    }
```

```
49    }
```
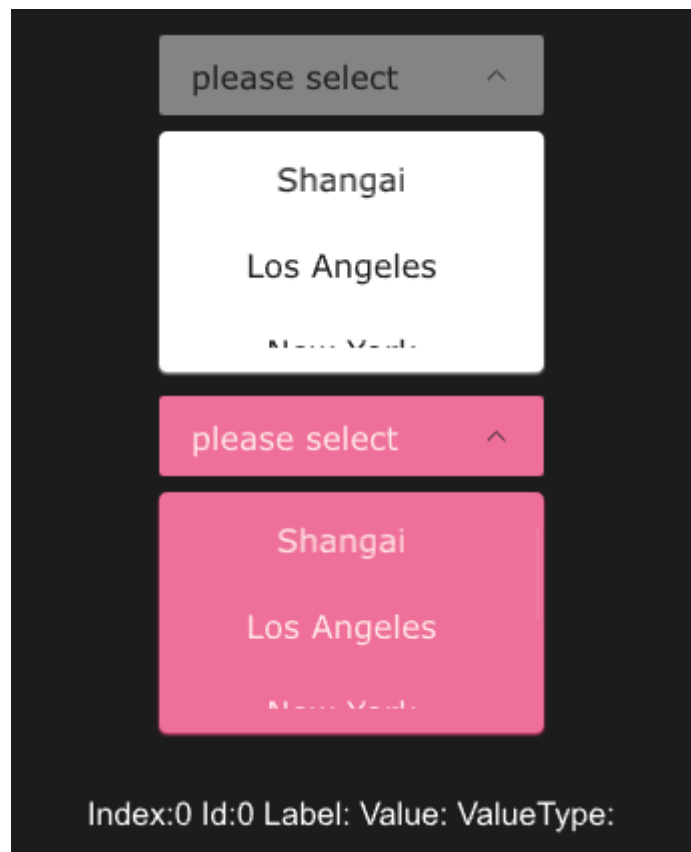

SURLink

SURLink is commonly used to represent text connections or sharing

properties

- `in property <image> 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
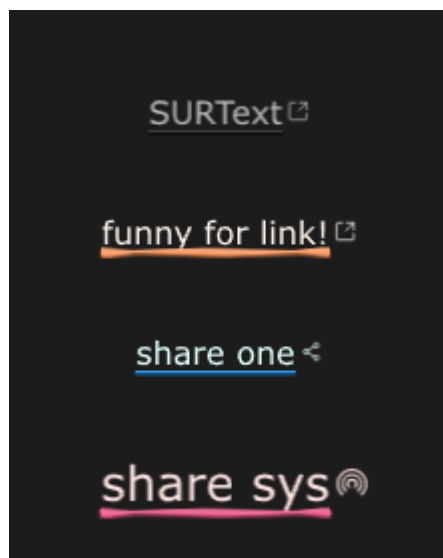
exeample

```
1    import {SURLink} from "../../index.slint";
2    import {Themes,IconSources} 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: @image-url("../../icons/share-one.svg");
22      content: "share one";
23    }
24    SURLink {
25      y: 280px;
26      funny:true;
27      theme: Error;
28      icon : @image-url("../../icons/share-sys.svg");
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,IconSources,ROOT-STYLES} 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-STYLES.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-STYLES.sur-size.small * 2;
28      padding-size : Small;
29      theme: Error;
30    }
31    SURAvatar {
32      y: 300px;
33      avatar-size : ROOT-STYLES.sur-size.large * 2;
34      padding-size : Large;
35      theme: Dark;
36      avatar:@image-url("../.E:\Rust\try\surrealism-ui\README\imgs\logo.png");
37    }
38
39  }
```
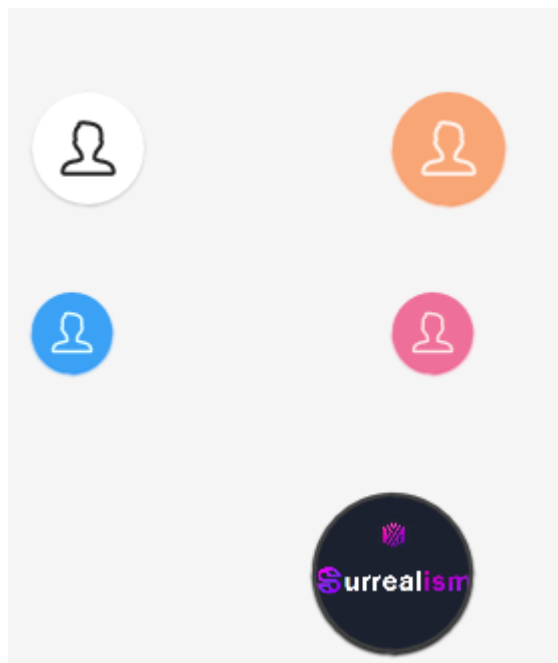
SURRadio

Radio let people select a single item

properties (card)

- `in-out property <bool> has-clicked` : the radio is clicked or not

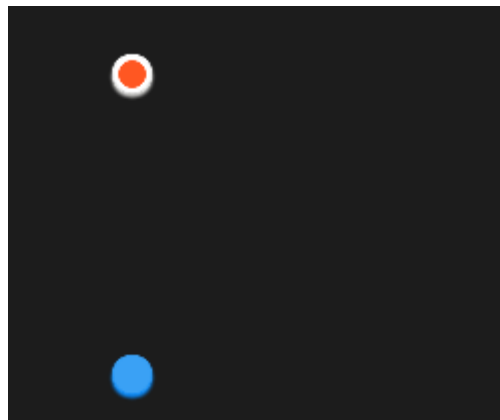- `in-out property <brush> active-color` : radio activecolor

functions

callbacks

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

example

```
1  import {SURRadio} from "../../index.slint";
2  import {Themes} from "../../themes/index.slint";
3
4  component TestCollection inherits Window {
5    height: 560px;
6    width: 600px;
7
8    SURRadio{
9      y: 60px;
10   }
11
12   SURRadio{
13     y: 180px;
14     active-color : #4affae;
15     theme:Primary;
16   }
17 }
```


SURPopup

A masked pop-up layer appears in the current window

And users will not be able to use the pop-up layer to cover the components under it. Clicking on the pop-up layer again will close it

properties

- `in-out property <bool> is-show` : the popup layer is show or not

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

functions

- `public function open()` : open the popup

- `public function close()` : close the popup

callbacks

example

```slint
1   import {SURPopup,SURButton} from "../../index.slint";
2   import {Themes} from "../../themes/index.slint";
3
4   component TestDivider inherits Window {
5     height: 800px;
6     width: 800px;
7     background: #535353;
8
9     SURButton {
10
11      content: "show";
12      clicked => {
13        p.open();
14
15        debug("sds1")
16      }
17    }
18
19
20    p:=SURPopup {
21      SURButton {
22        content: "you can add anything in Popup";
23        y: 160px;
24      }
25    }
26  }
```
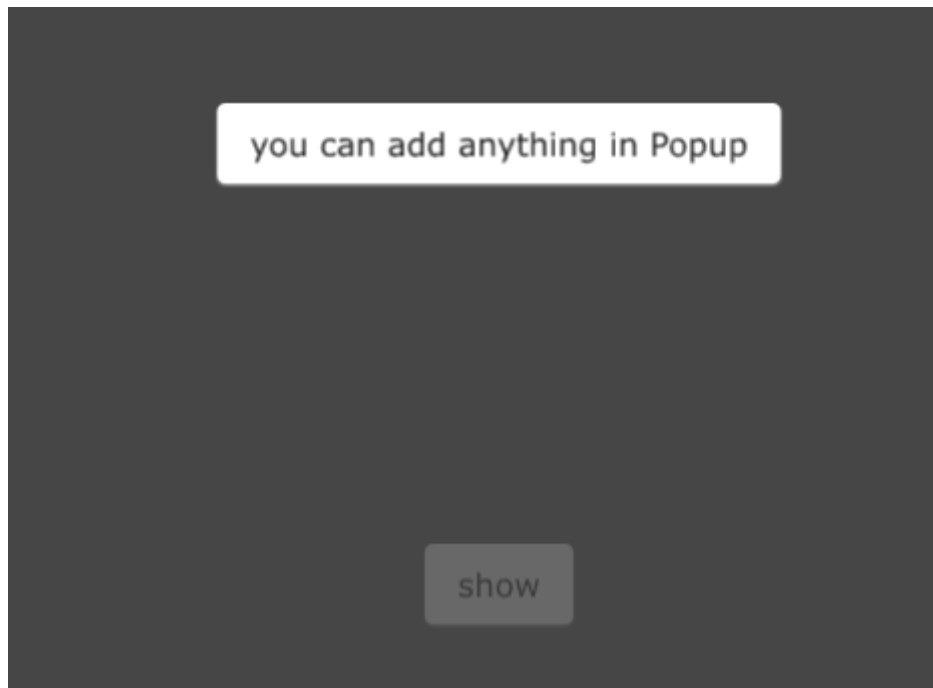
SURDivider

A divider groups sections of content to create visual rhythm and hierarchy.

Use dividers along with spacing and headers to organize content in your layout.

properties

- `in property <string> content` : divider content

- `in property <image> icon` : divider icon

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

functions

- `function show-what()->int` : show icon or content 🛑

example

```
1   import {SURDivider} from "../../index.slint";
2   import {Themes,IconSources} from "../../themes/index.slint";
3
4   component TestDivider inherits Window {
5     height: 400px;
6     width: 400px;
7     background: #535353;
8
9     SURDivider {
10      y: 60px;
11      width: 380px;
12    }
13    SURDivider {
14      y: 120px;
15      width: 380px;
16      icon:@image-url("../../icons/nail-polish-one.svg");
17      theme:Themes.Error;
18    }
```

```
19    SURDivider {
20      y: 180px;
21      width: 380px;
22      icon:@image-url("../../icons/earth.svg");
23      theme:Themes.Dark;
24    }
25    SURDivider {
26      y: 240px;
27      width: 380px;
28      content:"";
29      theme:Themes.Light;
30    }
31  }
```
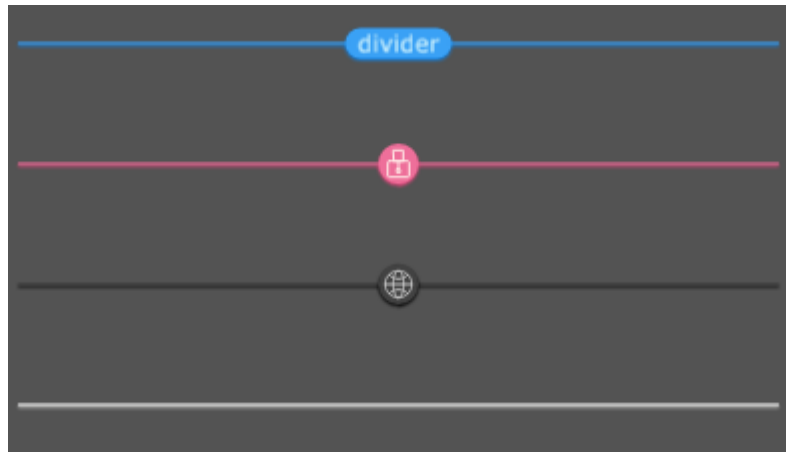

SURCollection

SURCollection is a grid storage box, but in reality it is not based on grid layout.

It achieves a flexible grid through a combination of dual for loops and horizontal and vertical layouts

Clicking on the pop-up layer again will close it

properties (card)

- `in property <length> font-size` : font size

- `in property <int> column-num` : column number

- `in property <int> row-num` : row number

- `in-out property <[[CollectionData]]>` data : collection data , this is the real data!

- `in property <length> row-height` : row height

- `in property <length> column-width` : column width

- `in property <length> row-spaceing` : row spaceing

- `in property <length> column-spacing` : column spacing

functions

callbacks

- `clicked(CollectionData)` : run if you click item in SURCollection

```slint
import {SURButton,SURCollection} from "../../index.slint";
import {Themes} from "../../themes/index.slint";

component TestCollection inherits Window {
  height: 560px;
  width: 900px;

  SURCollection{
    card-height: 300px;
    card-width: 300px;
    column-num: 3;
    font-size : 16px;
    theme: Dark;
    data: [
      [
        {id:0,name:"box1",source:@image-url("./collection_imgs/box1.svg")},
        {id:1,name:"box2",source:@image-url("./collection_imgs/box2.svg")},
        {id:2,name:"box3",source:@image-url("./collection_imgs/box3.svg")}
      ],
      [
        {id:3,name:"box4",source:@image-url("./collection_imgs/box4.svg")},

      ],
      [
        {id:4,name:"box6",source:@image-url("./collection_imgs/box6.svg")},
        {id:5,name:"box7",source:@image-url("./collection_imgs/box7.svg")},
      ]
    ];
    clicked(item)=>{
      debug(item.name);
      debug(item.id);
    }
  }

}
```
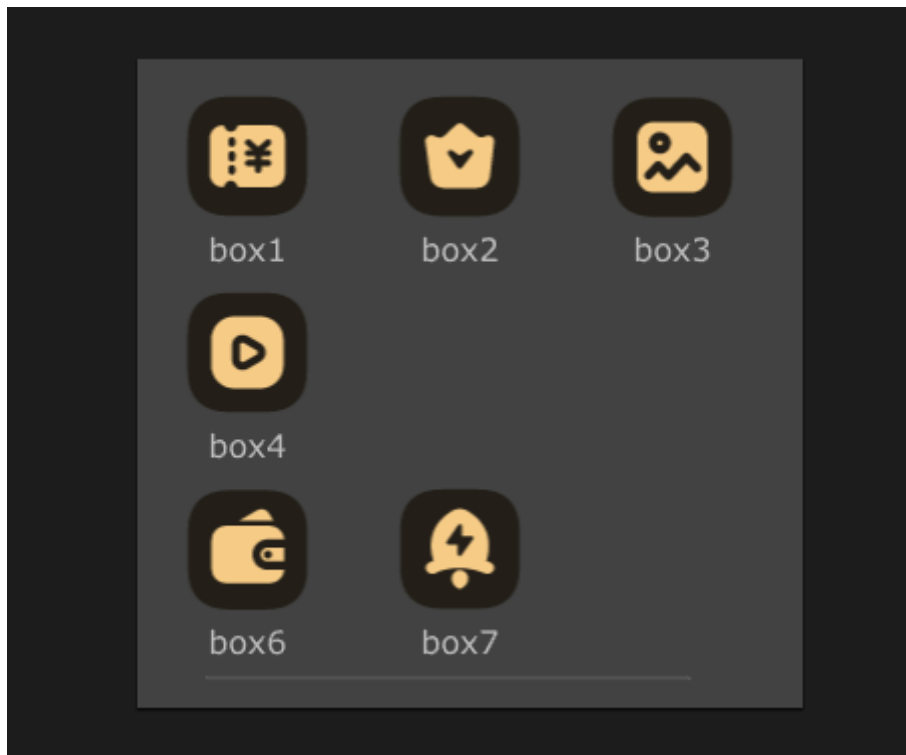
SURPersona

This component is used to display simple user introduction information

properties (card)

- `in property <string> name` : person name
- `in property <string> des` : person description
- `in property <string> btn` : click button content
- `in property <image> avatar` : avatar image
- `in property <length> name-font-size` : name font size
- `in property <length> des-font-size` : des font size
- `in property <length> avatar-height` : avatar height
- `in property <length> name-height` : name height
- `in property <length> des-height` : des height
- `in property <Themes> avatar-theme` : avatar theme
- `in property <Themes> name-theme` : name theme
- `in property <Themes> des-theme` : des theme
- `in property <Themes> btn-theme` : btn theme

functions

callbacks

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

example

```
1  import {SURPersona} from "../../index.slint";
2  import {Themes} from "../../themes/index.slint";
3
4  component TestCollection inherits Window {
5    height: 700px;
6    width: 400px;
7
8    SURPersona {
9      y: 10px;
10   }
11   SURPersona {
12     y: 350px;
13     btn-theme:Dark;
14     theme:Themes.Dark;
15     name-theme:Themes.Dark;
16     des-theme:Themes.Light;
17     avatar : @image-url("../../README/imgs/logo.png");
18     avatar-height:160px;
19     card-height: 310px;
20     clicked=>{
21       debug("view page!")
22     }
23   }
24 }
```
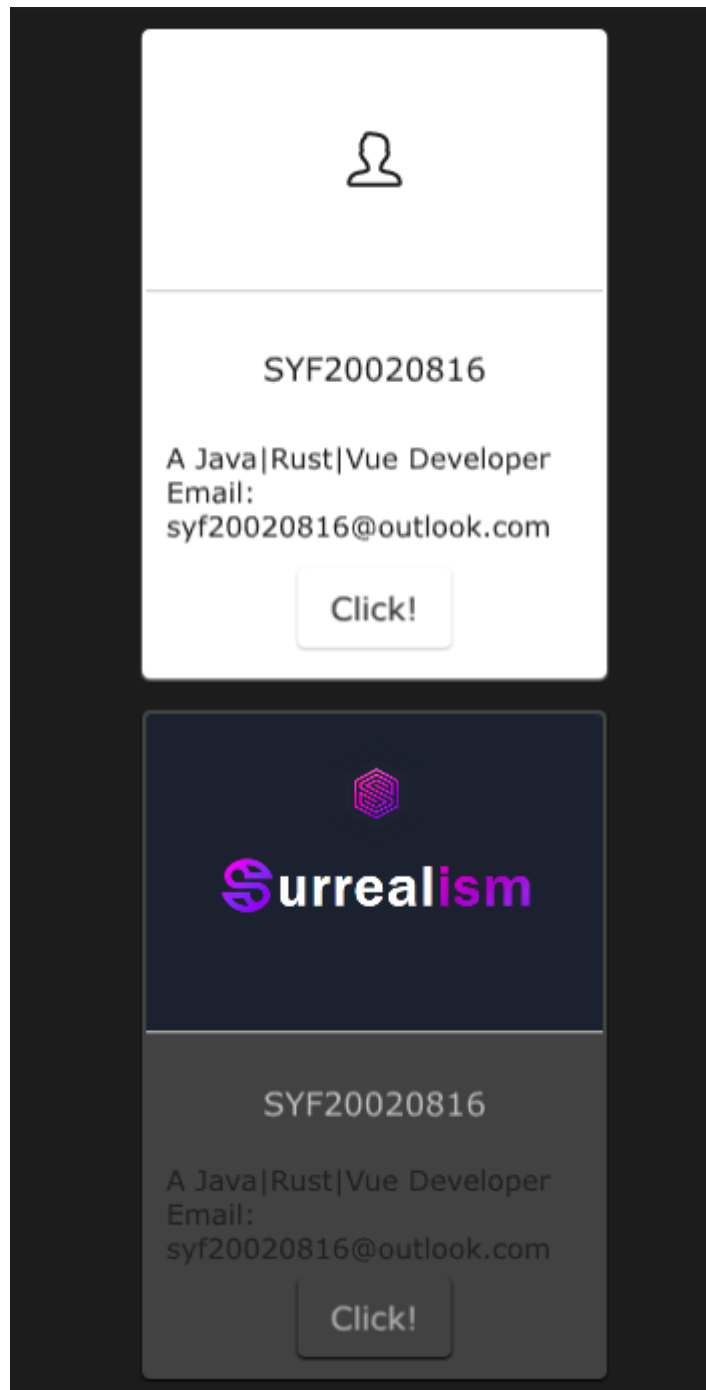
SURBadge

SURBadge is a quick way to display user status or events

properties (card)

- `in property <Position> position` : where the badge show

- `in-out property <image> icon` : icon of the badge

- `in property <brush> icon-color` : icon color

- `in property <brush> font-color` : font color

- `in property <ResType> res-type` : icon Type see result!(but you can define without use this property)

functions

- `pure public function get-x(p_right:length)->length` 👍

- `pure public function get-y(p_bottom:length)->length` 👍

callbacks

example

```
1   import {SURBadge,SURAvatar} from "../../index.slint";
2   import {Themes} from "../../themes/index.slint";
3
4   component TestCollection inherits Window {
5     height: 460px;
6     width: 400px;
7
8     b1:=Rectangle {
9
10      y: 30px;
11      height: avatar.height;
12      width: avatar.width;
13      avatar:=SURAvatar {
14
15      }
16      SURBadge {
17        x: self.get-x(avatar.width);
18        y: self.get-y(avatar.height);
19      }
20    }
21    b2:=Rectangle {
22
23      y: 120px;
24      height: avatar2.height;
25      width: avatar2.width;
26      avatar2:=SURAvatar {
27      }
28      SURBadge {
29        x: self.get-x(avatar2.width);
30        y: self.get-y(avatar2.height);
31        position: Left-Bottom;
32        res-type: Success;
33      }
34    }
35    b3:=Rectangle {
36
37      y: 210px;
38      height: avatar3.height;
39      width: avatar3.width;
40      avatar3:=SURAvatar {
41
42      }
43      SURBadge {
44        x: self.get-x(avatar3.width);
45        y: self.get-y(avatar3.height);
46        position: Left-Top;
```

```
47        res-type: Warning;
48        icon-color:#ff0000;
49        font-color:#ff0000;
50      }
51    }
52    b4:=Rectangle {
53      y: 300px;
54      height: avatar4.height;
55      width: avatar4.width;
56      avatar4:=SURAvatar {
57      }
58      SURBadge {
59        x: self.get-x(avatar4.width);
60        y: self.get-y(avatar4.height);
61        position: Right-Top;
62        res-type: Error;
63      }
64    }
65
66  }
```
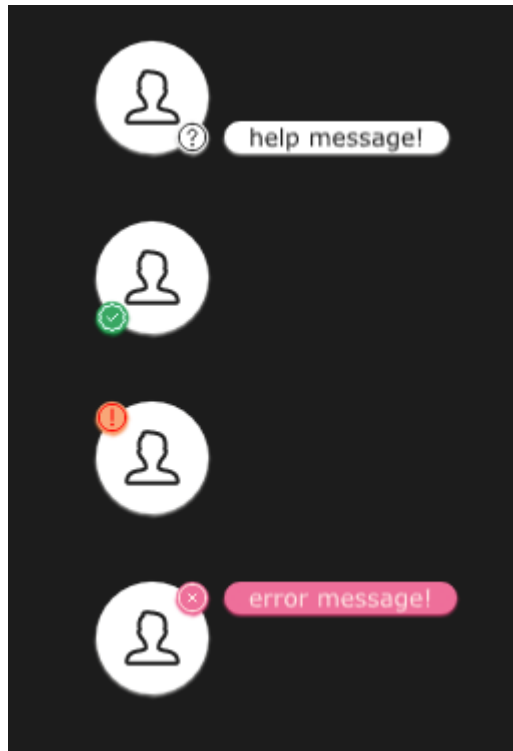

SURProgress

SURProgress is commonly used to display download progress or event processing progress
And you can fully control it through the progress property

properties

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

- `in property <string> content` : what you wanna show to others

- `in-out property <float> progress` : progress

- `private property <length> unit` : unit of progress length

functions

- `pure public function get-progress()` : get timely progress

- `public function full()` : make progress 100%

- `public function clear()` : : make progress 0%

- `public function add(num:float)` : increase progress

callbacks

```slint
1   import {SURProgress,SURButton} from "../../index.slint";
2   import {Themes} from "../../themes/index.slint";
3
4   component TestDivider inherits Window {
5     height: 400px;
6     width: 400px;
7     background: #5b64cd;
8     SURProgress {
9       y: 100px;
10    }
11    a:=SURProgress {
12      y: 200px;
13      theme:Primary;
14    }
15    SURProgress {
16      y: 300px;
17      theme:Dark;
18      progress:86;
19    }
20    SURButton{
21      x: 60px;
22      y: 340px;
23      content: "add";
24      clicked => {
25        a.add(5);
26      }
27    }
28    SURButton{
29      x: 160px;
30      y: 340px;
31      content: "full";
32      clicked => {
33        a.full();
34      }
35    }
36    SURButton{
37      x: 260px;
38      y: 340px;
39      content: "clear";
40      clicked => {
41        a.clear();
42      }
43    }
44  }
```

SURTip

A tip provides supplemental, contextual information elevated near its target component

properties

- `in property <Themes> theme` : Surrealism Theme
- `in property <string> content` : tip content
- `in-out property <bool> show-tip` : hover and tip will show
- `in property <TipPosition> pos` : the position of the tip

functions

- `public function open()` : open the tip
- `public function close()` : close the tip

callbacks

- `callback clicked()` : use to open|close the tip

example

```
1   import {SURTip,SURButton } from "../../index.slint";
2   import {Themes} from "../../themes/index.slint";
3
4
5   component TestWindow inherits Window {
6     height: 400px;
7     width: 400px;
8     SURTip{
9       y: 80px;
10        height:inner0.height;
11        width: inner0.width;
12        theme: Dark;
13        pos:Top;
```

```
14        content:"this is a \n........tip window";
15        show-tip:inner0.has-hover;
16        inner0:=SURButton {
17          content: "click";
18        }
19      }
20    SURTip{
21      height:inner.height;
22      width: inner.width;
23      theme: Dark;
24      pos:Left;
25      content:"this is a \n........tip window";
26      inner:=SURButton {
27        content: "click";
28        clicked => {
29          parent.clicked();
30        }
31      }
32    }
33    SURTip{
34     y: 300px;
35      height:inner2.height;
36      width: inner2.width;
37      theme: Dark;
38      pos:Top;
39      content:"this is a \n........tip window";
40      inner2:=SURButton {
41        content: "click";
42        clicked => {
43          parent.clicked();
44        }
45      }
46    }
47
48  }
```

image-20230916101547495
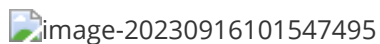
SURLoading (some error in animation)

This is a loading component that you can embed anywhere you want to add a loading animation (now animation have some error)

properties

- `in-out property <bool> is-show` : the popup layer is show or not
- `in property <Themes> theme` : Surrealism Themes
- `in property <image> loading-icon` : loading icon
- `in property <duration> duration` : animation duration
- `in property <bool> an` : open animation or not (error : https://github.com/slint-ui/slint/issues/3494)
- `in property <string> content` : loading content

functions

callbacks

- `callback open()` : open the loading
- `callback close()` : close the loading

example

```
1  import {SURLoading,SURButton,SURCard} from "../../index.slint";
2
3  export component TestLoading inherits Window {
4      height: 600px;
5      width: 400px;
```

```
 6      SURButton {
 7        y: 100px;
 8        content: "show";
 9        clicked => {
10          p.open();
11        }
12      }
13      SURButton {
14        y: 160px;
15        content: "close";
16        clicked => {
17          p.close();
18        }
19      }
20      SURCard{
21        y: 260px;
22        clip: true;
23        card-height: 260px;
24        card-width: 180px;
25        p:=SURLoading { }
26      }
27  }
```
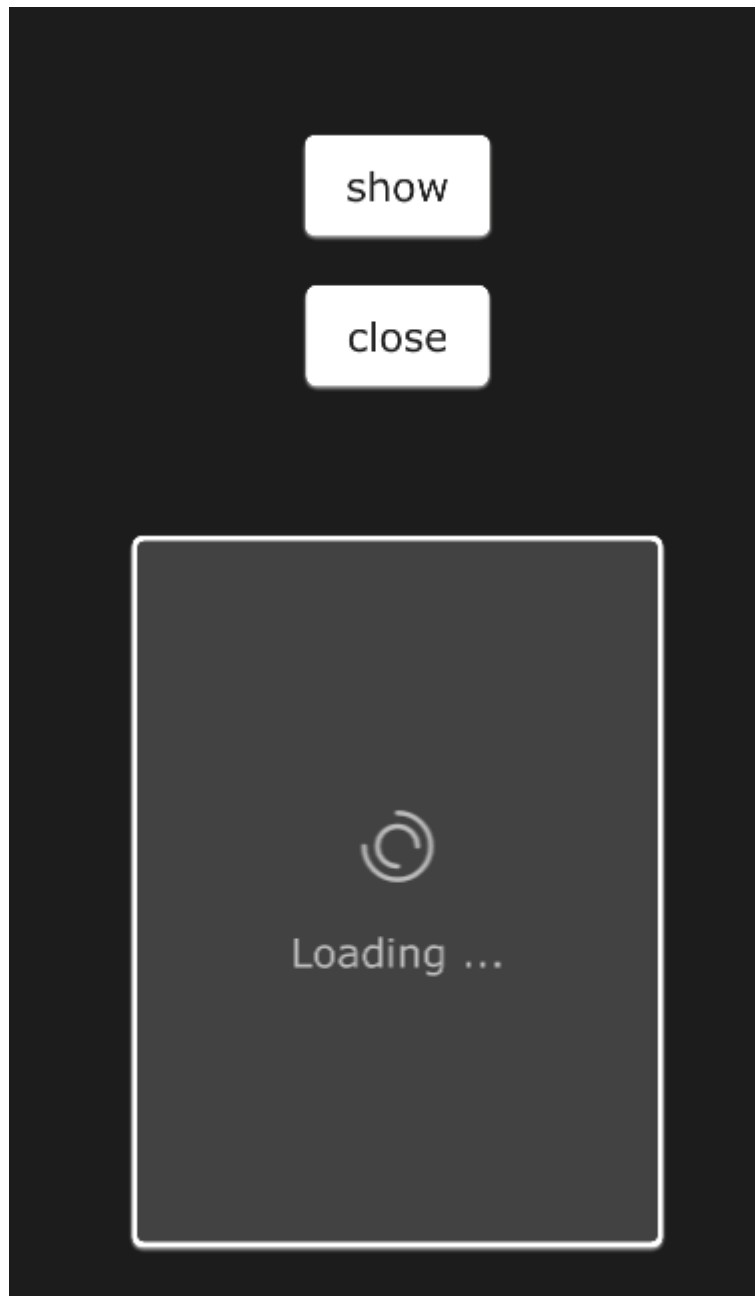
SURDialog

Dialogs are used to confirm messages or events and display content

properties

- `in property <string> dialog-title` : dialog title;
- `in property <length> dialog-title-size` : dialog title font size;
- `in property <string> dialog-details` : content information in the dialog box;
- `in property <Themes> cancel-btn-theme` : cancel button theme;
- `in property <Themes> confirm-btn-theme` : confirm button theme;
- `in property <string> cancel-btn-content` : cancel button content;
- `in property <string> confirm-btn-content` : confirm button content;
- `in-out property <bool> is-show` : show dialog or not;
- `in property <Themes> theme` : Surrealism Themes
- `in property <float> dialog-height` : Dialog height proportion

- `in property <float> dialog-width` : Dialog width proportion

functions

- `public function open()` : open dialog

- `public function close()` : close dialog

callbacks

- `callback confirm()` : run after confirm button click

- `callback cancel()` : run after cancel button click

example

```
 1  import {SURDialog,SURButton,SURTable,SURTableColumn} from
    "../../index.slint";
 2  import {Themes} from "../../themes/index.slint";
 3
 4  component TestDialog inherits Window {
 5    height: 800px;
 6    width: 800px;
 7    background: #535353;
 8
 9    SURButton {
10
11      content: "show";
12      clicked => {
13        p.open();
14      }
15    }
16
17
18    p:=SURDialog {
19      dialog-details : "";
20      confirm-btn-theme: Success;
21      dialog-width:80%;
22      dialog-height:52%;
23      // do after confirm btn clicked
24      confirm=>{
25        debug("confirm btn clicked~!")
26      }
27      SURTable {
28        // you can use this way to get height
29        // it depends on how many datas in column
30        height: tcol1.get-height();
31        width: 350px;
32
33        tcol1:=SURTableColumn {
34          border:false;
35          theme:Themes.Primary;
36          width: 100px;
37          name:"id";
38          // row-height:60px;
39          datas: ["101","102","103"];
40        }
```

```
41        SURTableColumn {
42          theme:Themes.Primary;
43          width: 100px;
44          name:"name";
45          datas: ["Mat","Jarry","Kaven"];
46        }
47        SURTableColumn {
48          theme:Themes.Primary;
49          width: 150px;
50          name:"Operations";
51          // cheat datas
52          datas: [" "," "," "];
53          operation-enabled:true;
54        }
55      }
56    }
57  }
```
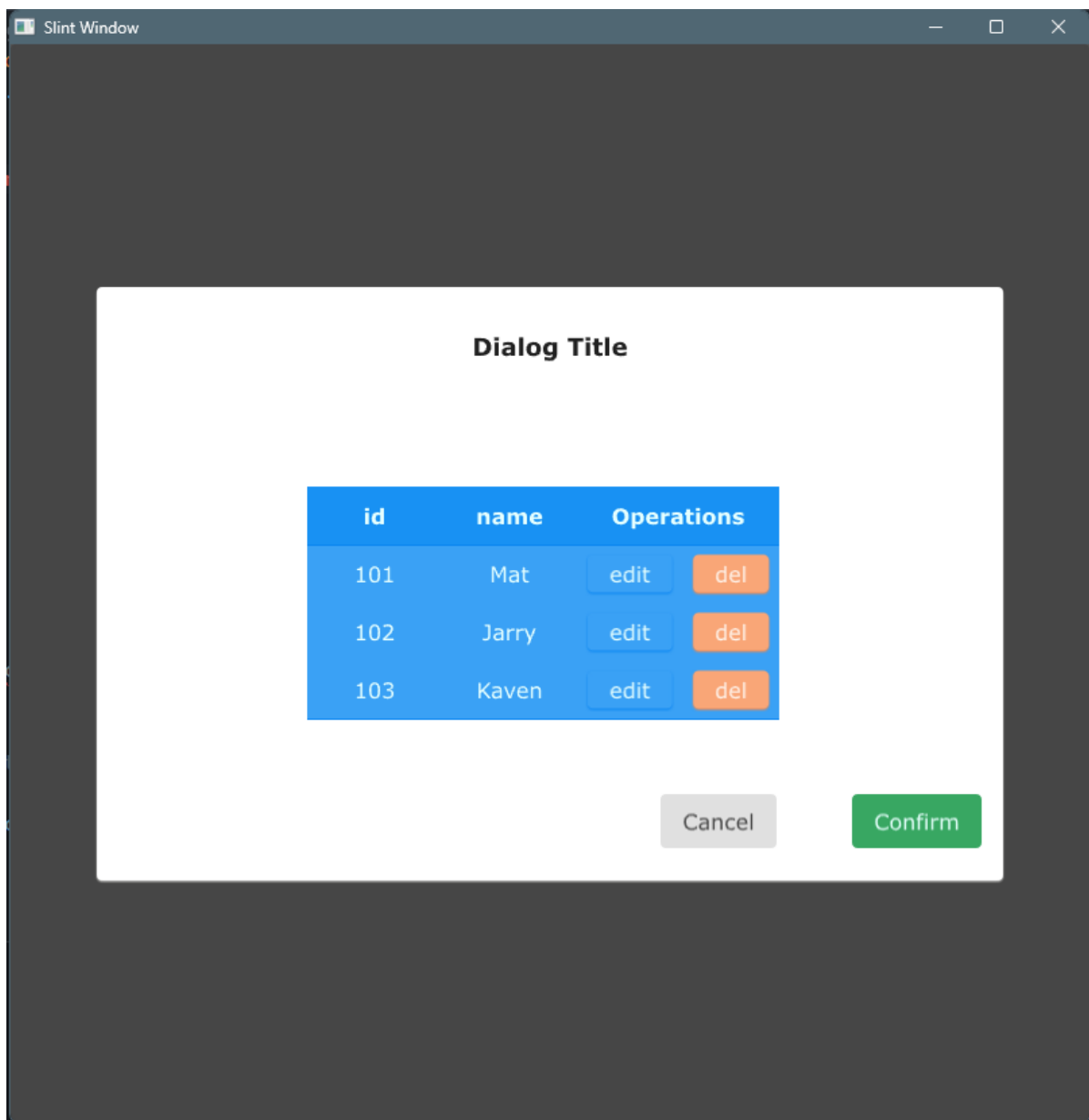
SURMenu

SURMenu is a menu bar located on the left side that you can quickly generate through the menu-data property

properties

- `in-out property <length> icon-box-size` : menu item size ⛔
- `in-out property <length> icon-size` : menu item icon size ⛔;
- `in property <[MenuData]> menu-data` : menu item data (generate menus through it)
- `in-out property <int> active` : which item is active
- `private property <brush> hover-icon-color` : menu item icon color changed when hover

callbacks

- `callback change(int,MenuData)` : run if you click menu item
- `callback clicked-account()` : run if you click account icon
- `callback clicked-setting()` : run if you click setting icon

example

```
1   import { SURMenu , SURIcon} from "../../index.slint";
2   import {IconSources} from "../../themes/index.slint";
3
4   component TestMenu inherits Window {
5       height: 600px;
6       width: 300px;
7       Rectangle {
8         x: 0;
9         y: 0;
10        height:parent.height;
11        width: menu.width;
12        menu:=SURMenu {
13          theme: Dark;
14          change(index,item)=>{
15            debug(index);
16            debug(item);
17          }
18          clicked-account()=>{
19            debug("clicked account");
20          }
21        }
22      }
23   }
```

Slint Window

Search