



Front End Basic training

Low Code Studio



How to use this tutorial ?

Welcome to your first journey
with Convertigo Low Code Studio.
Let's explore its many features.



Concepts & Definitions

Convertigo uses many concepts
you may not be familiar with.
Find answers with this icon.



Practice time

You prefer to skip the concepts
and start by practice.
Go straight to this icon.



Table of Contents



1 – FRONTEND OVERVIEW

2 – THEMING & STYLING

3 – INTEGRATE PAGES WITH UI COMPONENTS

4 – SHARED COMPONENTS

5 – DIRECTIVES & BINDING DATA

6 – EVENTS, ACTIONS & VARIABLES

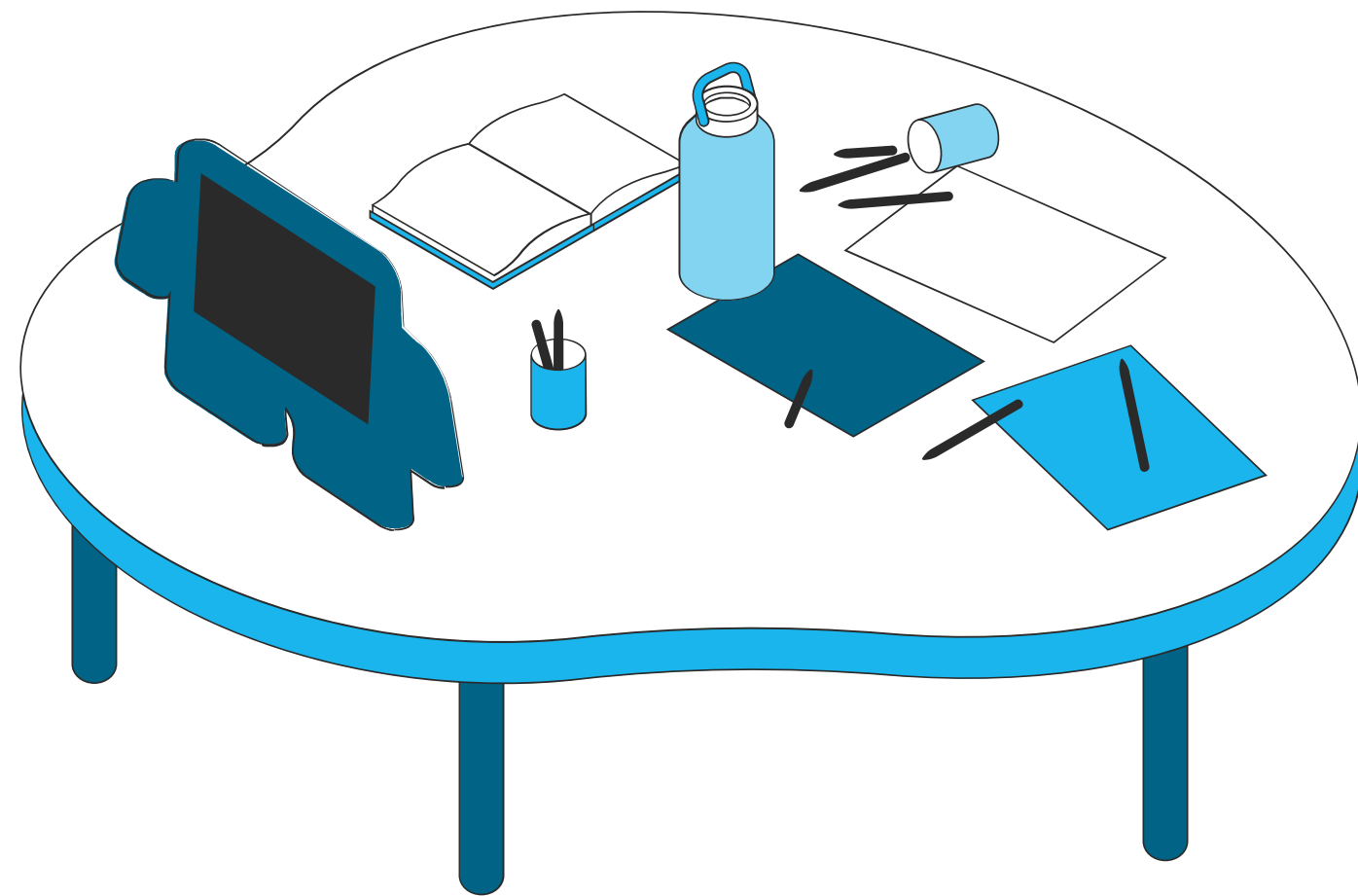
7 – SHARED COMPONENTS EVENTS

8 – SHARED ACTIONS

9 – FINALIZE DATA BINDING

APPENDIX

1 – Frontend Overview



1.1 Technical knowledge

1.2 NgxApp & Application Viewer

1.3 Launching the Application Viewer

1.4 NGX palette

1.5 UI Components

1.6 Application Viewer Controls

1.7 Pages & Menus

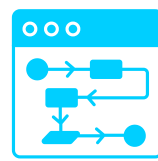
1.1 Technical knowledge

The following concepts are necessary for mastering the studio.



WEB TECHNOLOGIES

- JavaScript & JSON
- Angular 13+
- Ionic 6+



ALGORITHMS

- Pseudocode basics
- Loops, Conditional statements...



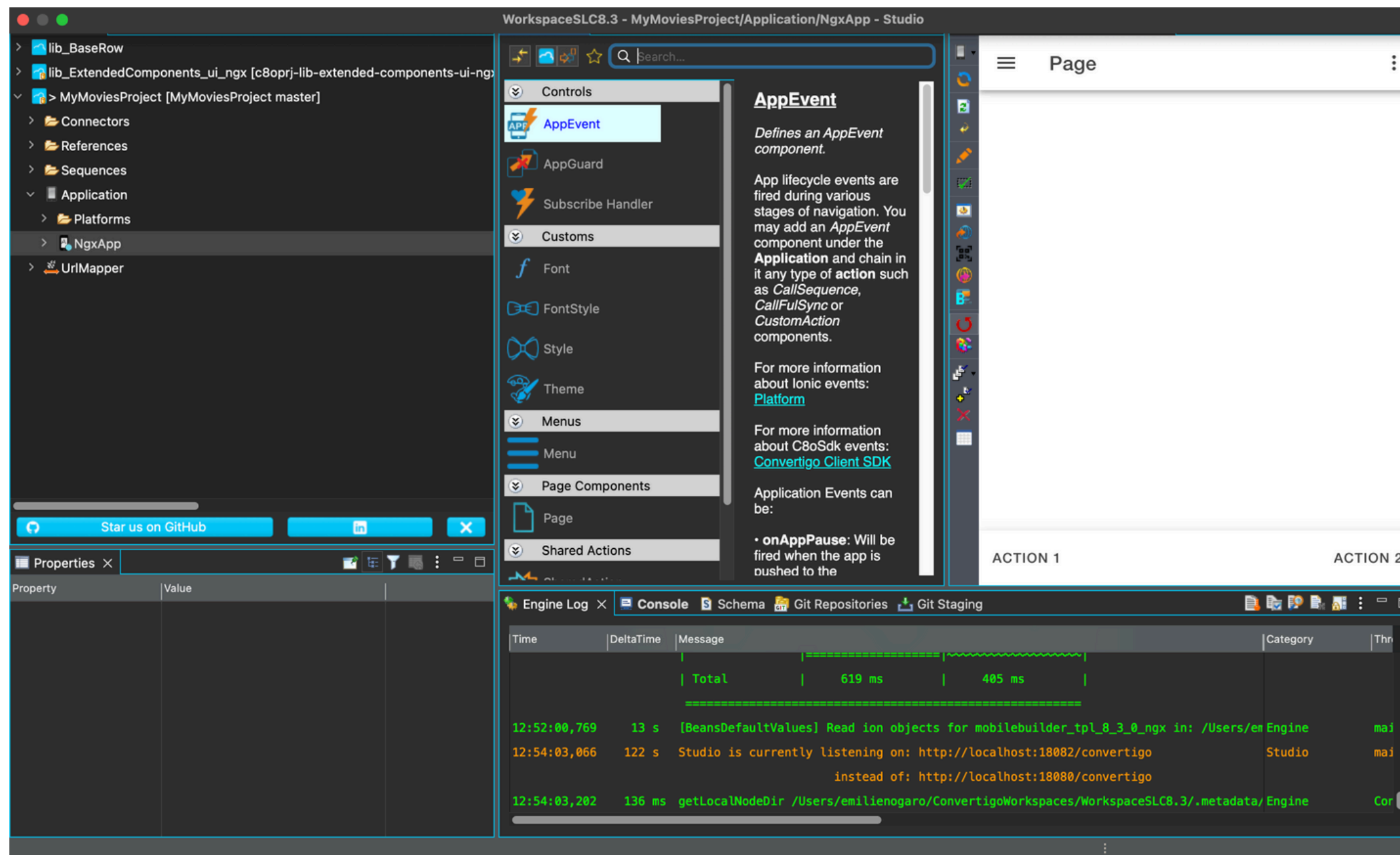
BACKEND & PACKAGE MANAGING

- Node.js & NPM
- Java
(for advanced users)

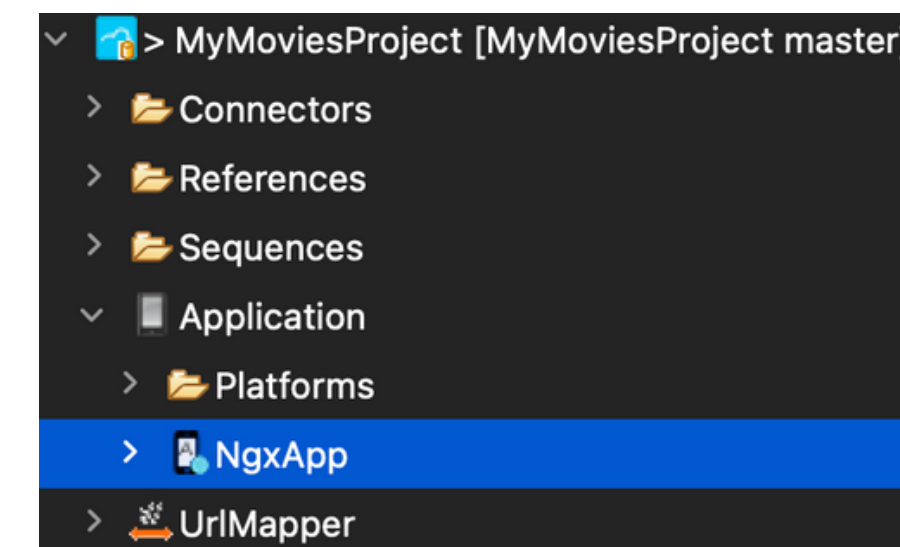


1.2 NgxApp & Application Viewer

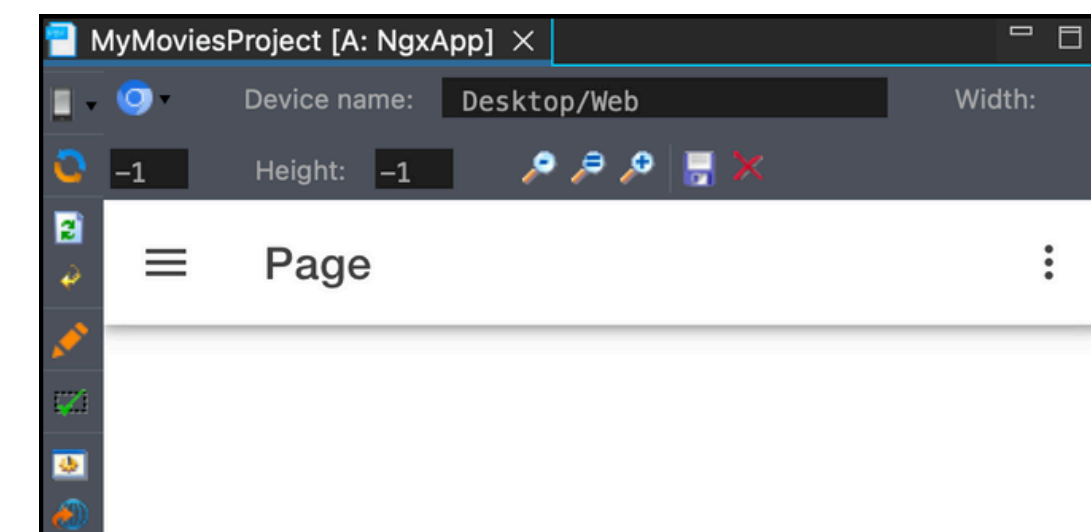
In the project treeview, under the **Application** object, the **NgxApp** object represents the **graphical User Interface** displayed in the **Application Viewer** in the right panel.



NgxApp object in treeview



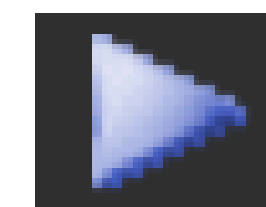
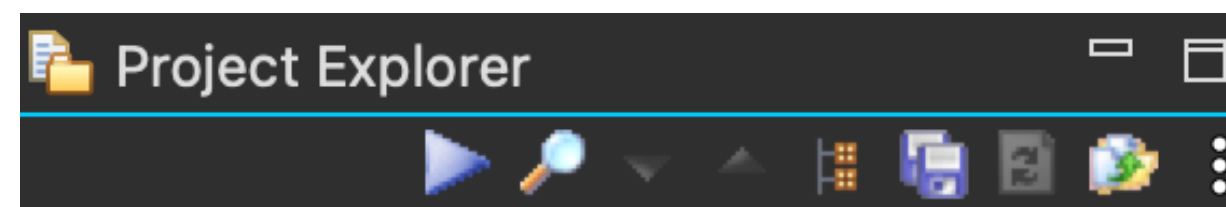
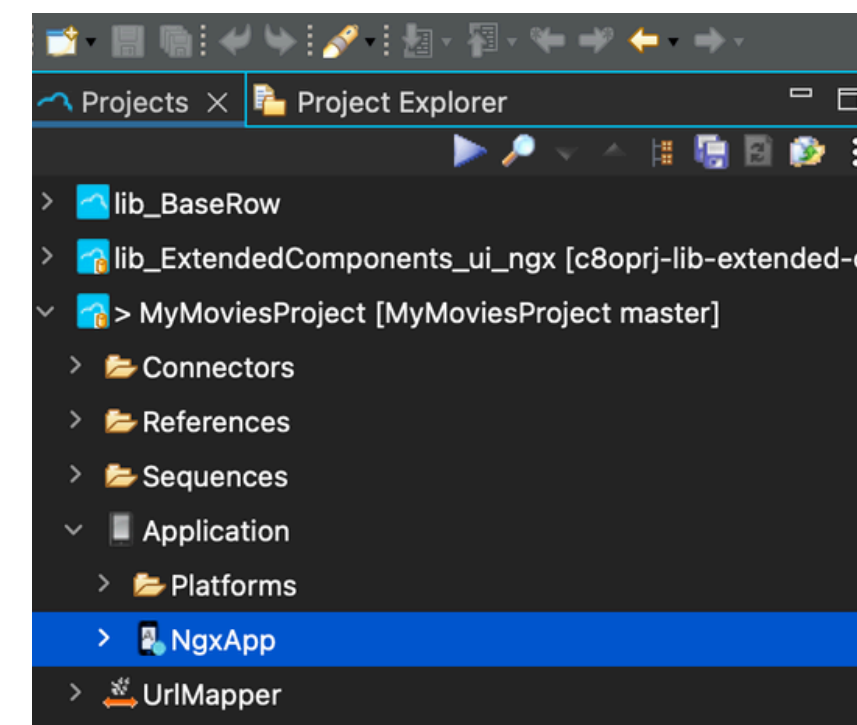
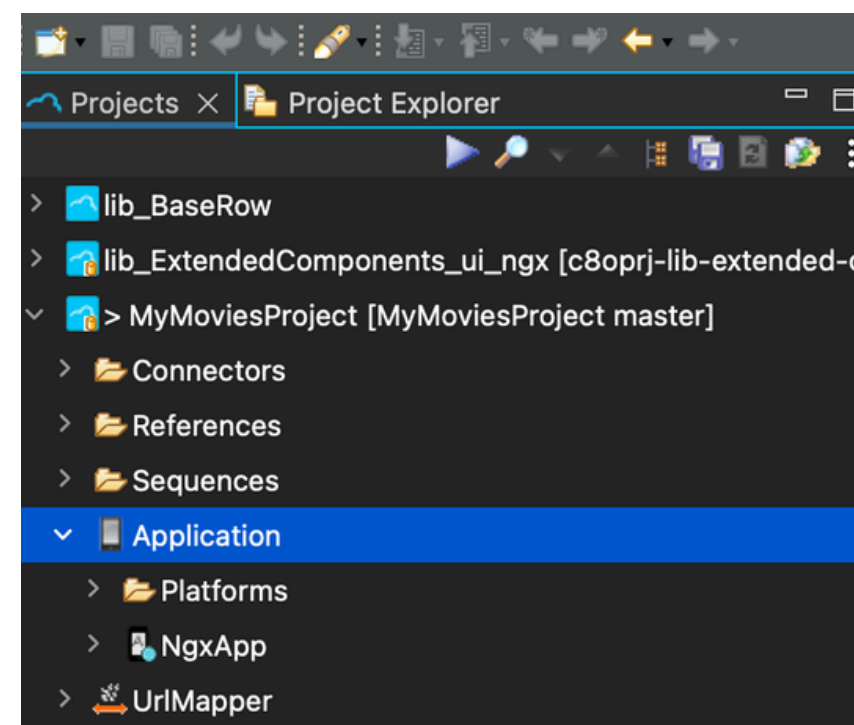
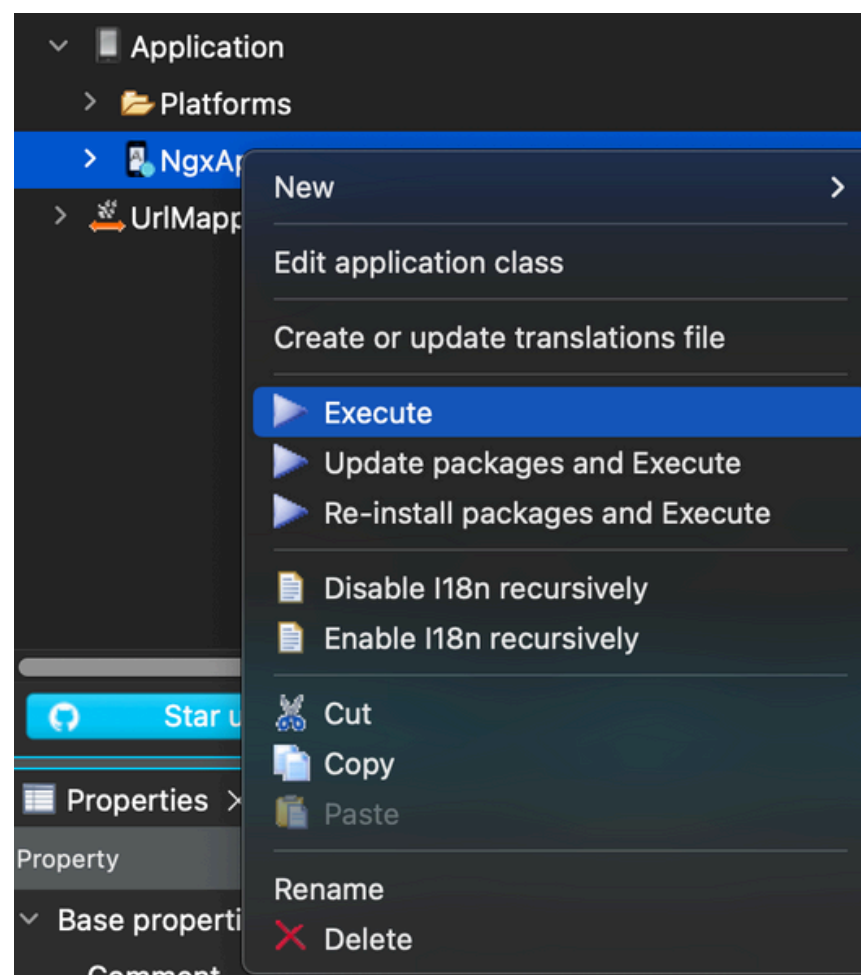
Application Viewer



1.3 Launching the Application Viewer

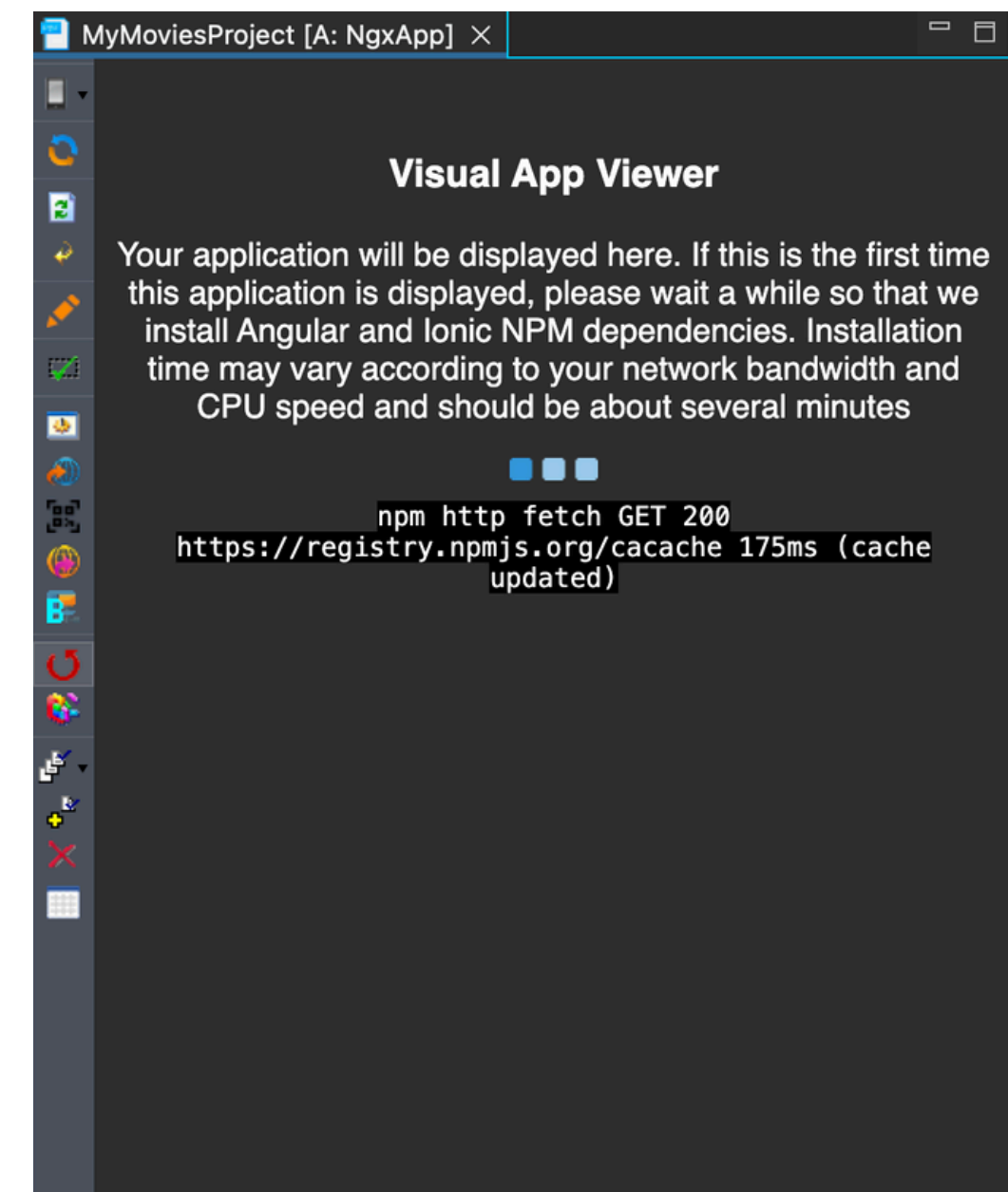
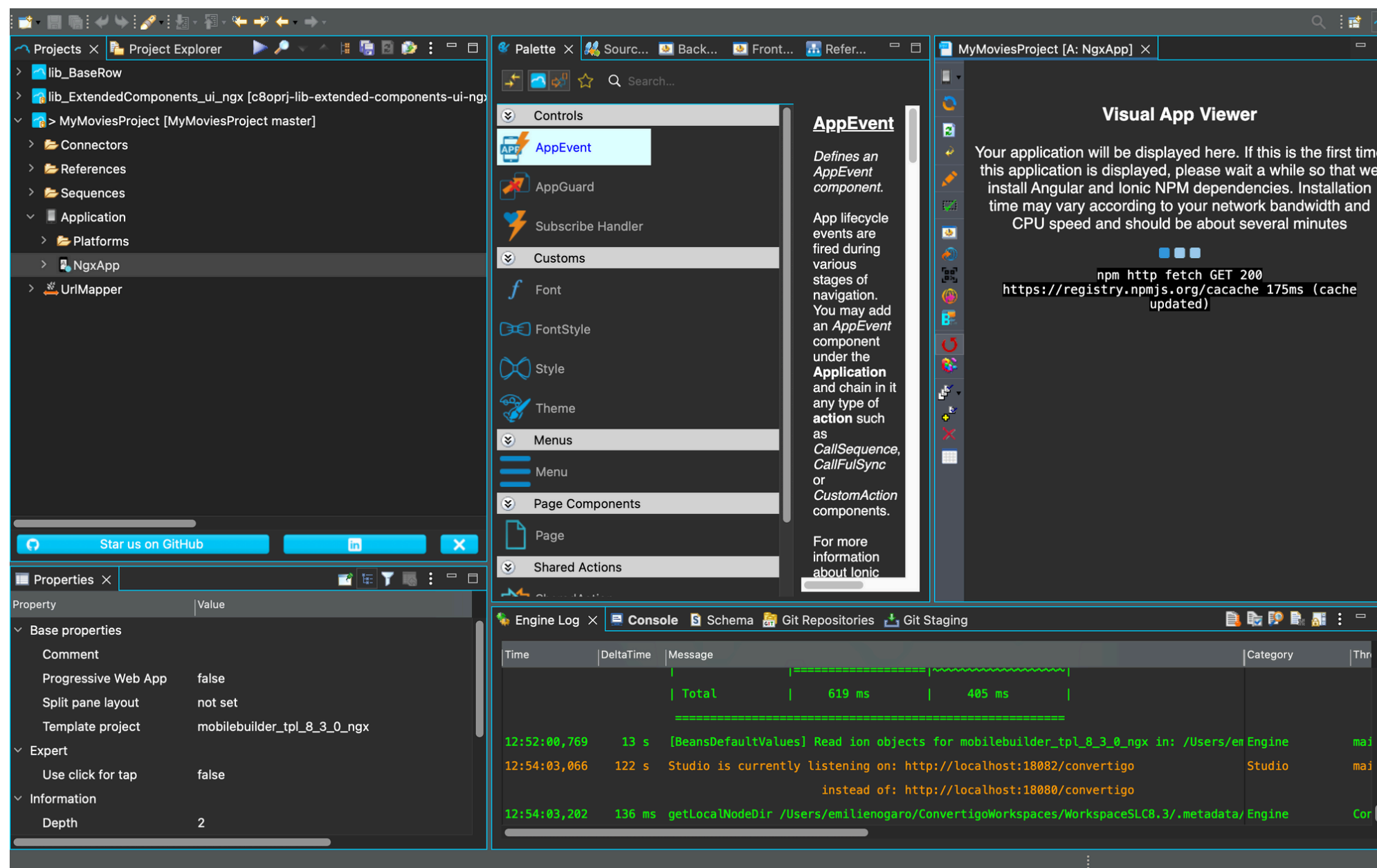
To launch the Application Viewer, you can

- click once on **NgxApp** in the treeview then click on **Execute** in the panel
- double-click on **NgxApp** in the treeview
- clic on the arrow at the top of the **Projects panel**
(available only when the cursor is on **Application** or **NgxApp** in the treeview)



1.3 Launching the Application Viewer

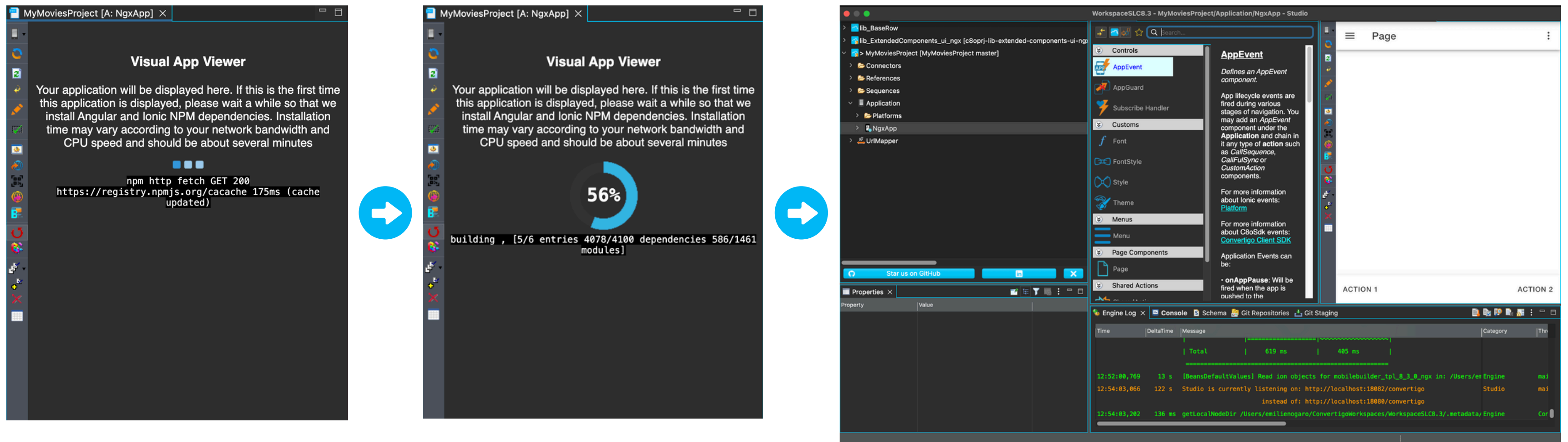
When launched, the Application Viewer appears in the right panel and starts to build the application. First, it uses **NPM** to download and install the application packages dependencies.



1.3 Launching the Application Viewer

Once the packages dependencies have been downloaded and installed, the viewer starts to build your application to be displayed.

While the viewer is displayed, any modification in the Builder project tree is reflected in real time in the viewer.



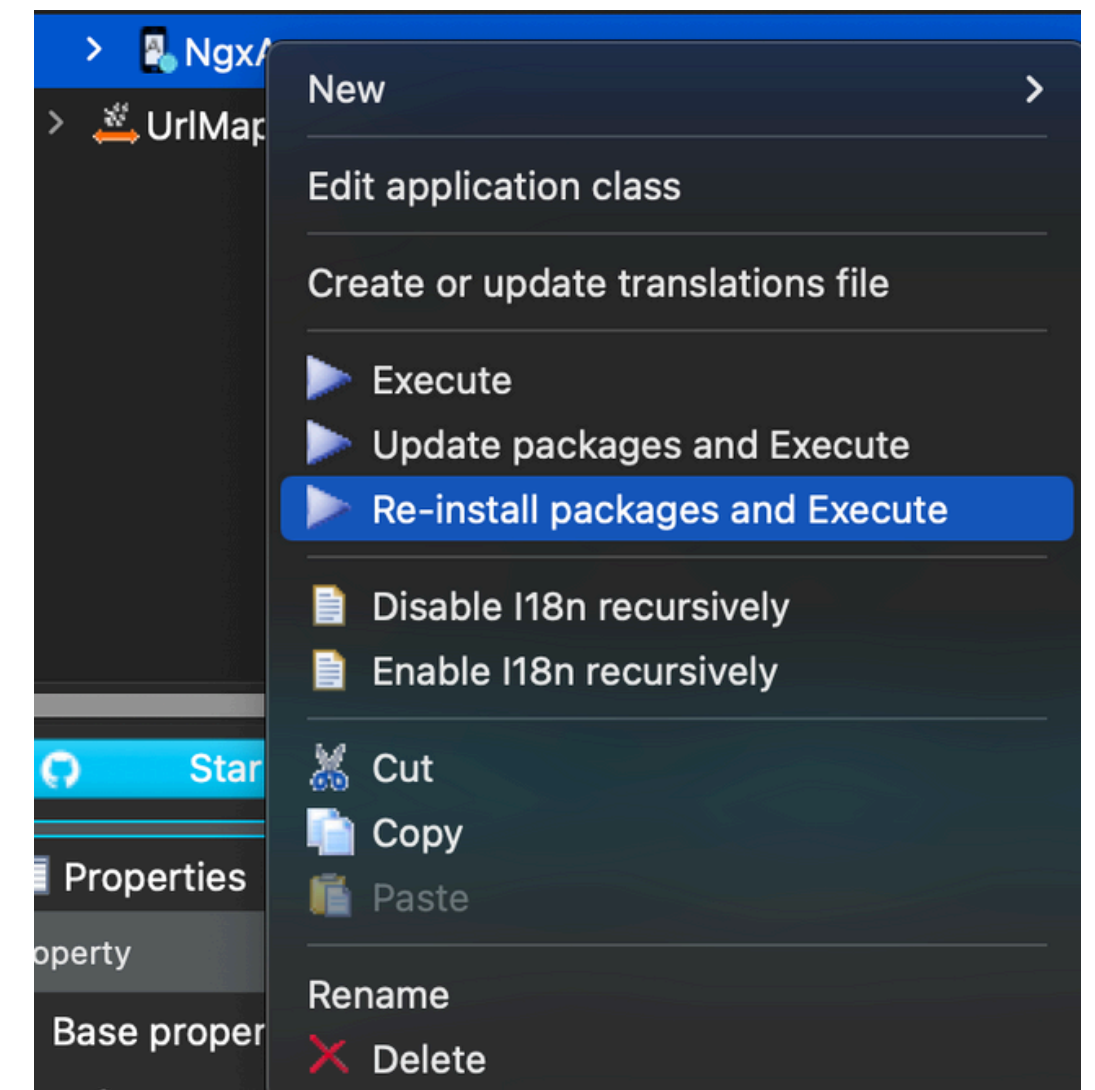
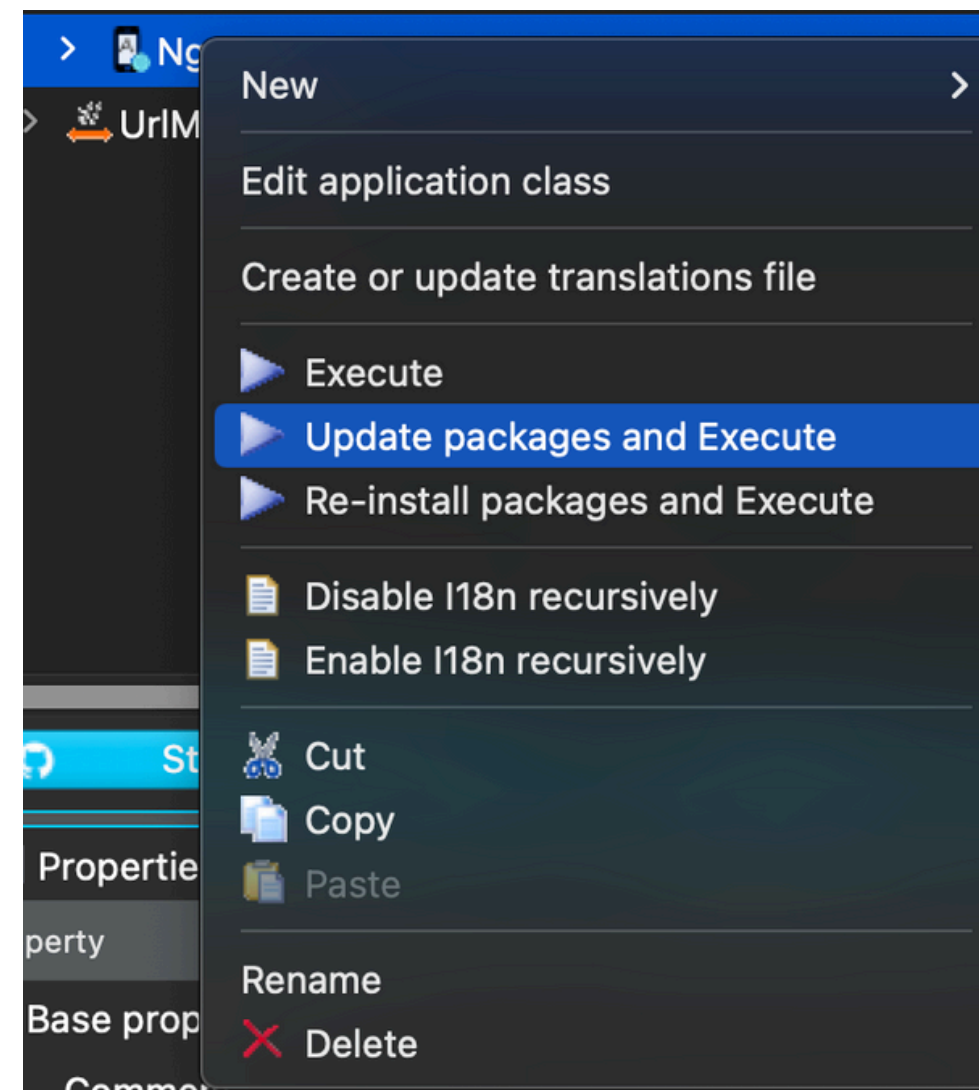
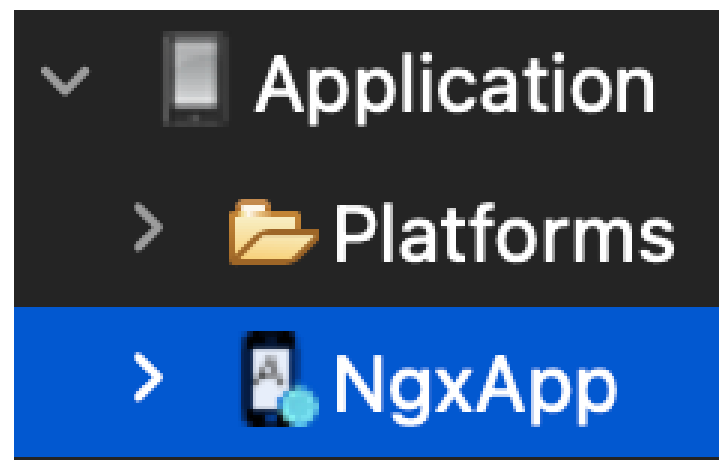
1.3 Launching the Application Viewer

This NPM install process can last several minutes and is usually done only once.

However, if you need to **update NPM dependencies**, or **re-install** them, you can right-click on **NgxApp**

then click on

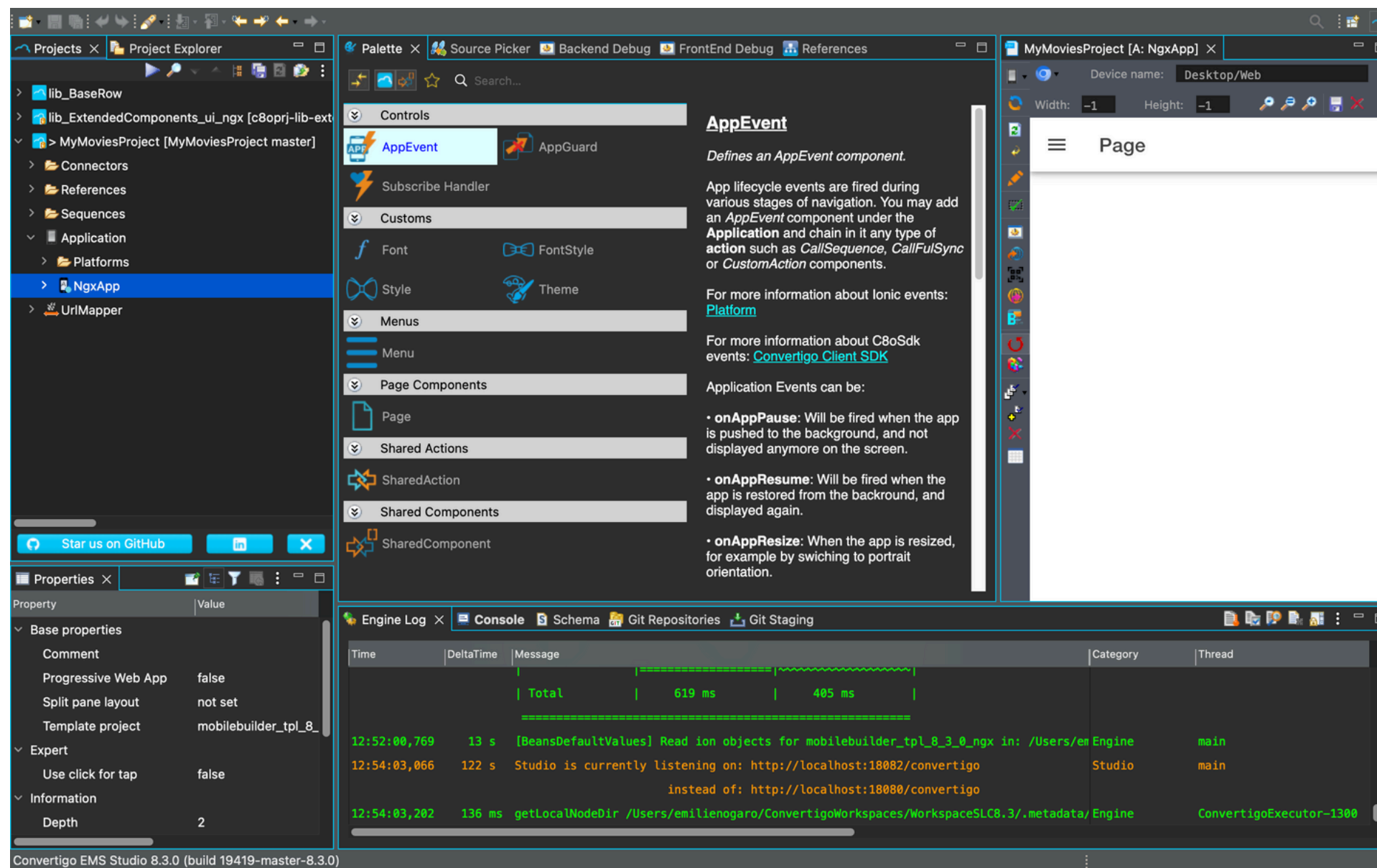
- **Update packages and Execute**
- **Re-install packages and Execute**



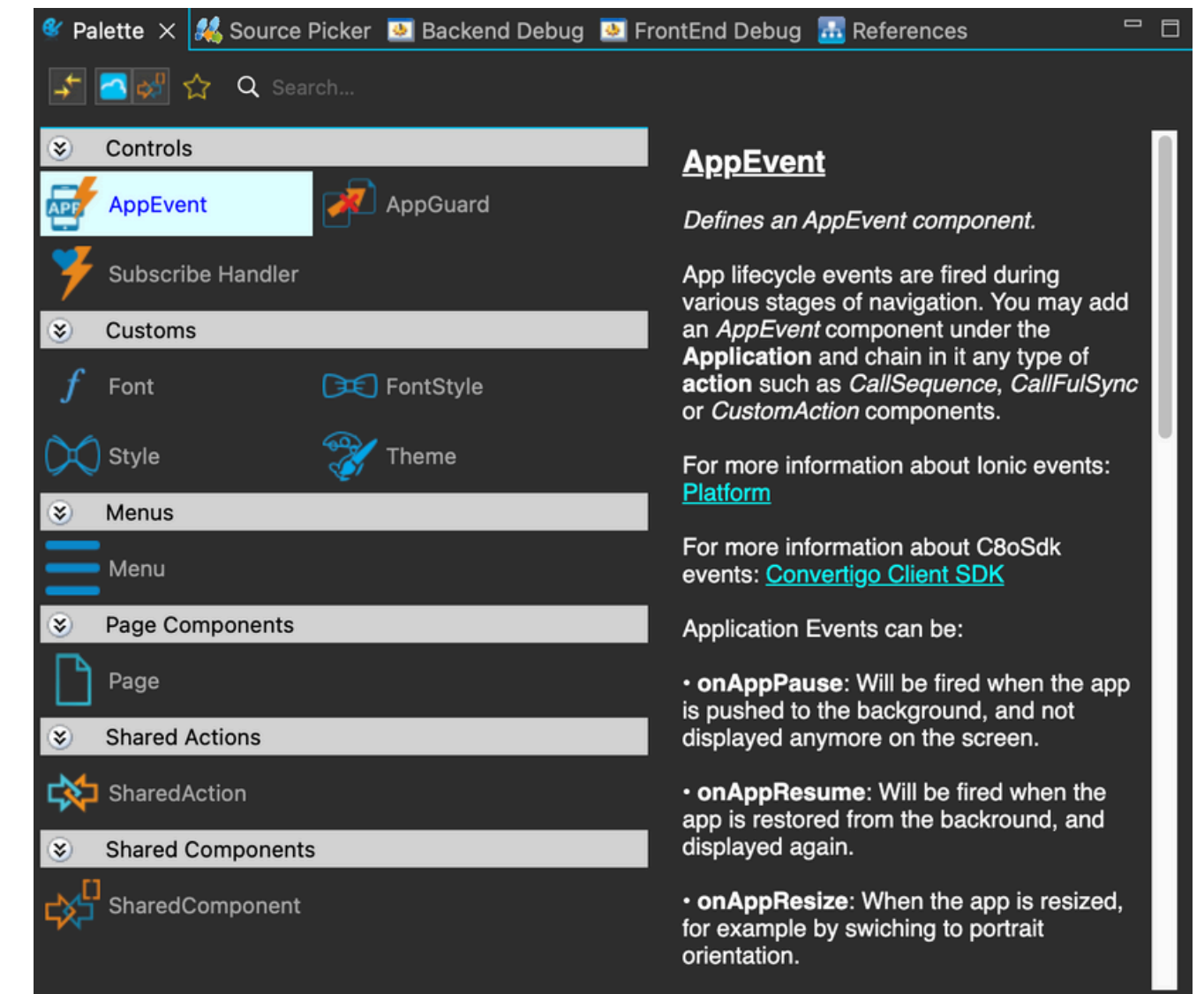
1.4 NGX palette & Components

To design and build an application, **front-end objects** or **components** are dragged & dropped from the **NGX palette** in the **NgxApp** treeview.

The **Application Viewer** automatically **reflects the changes** to display what the application will look like.



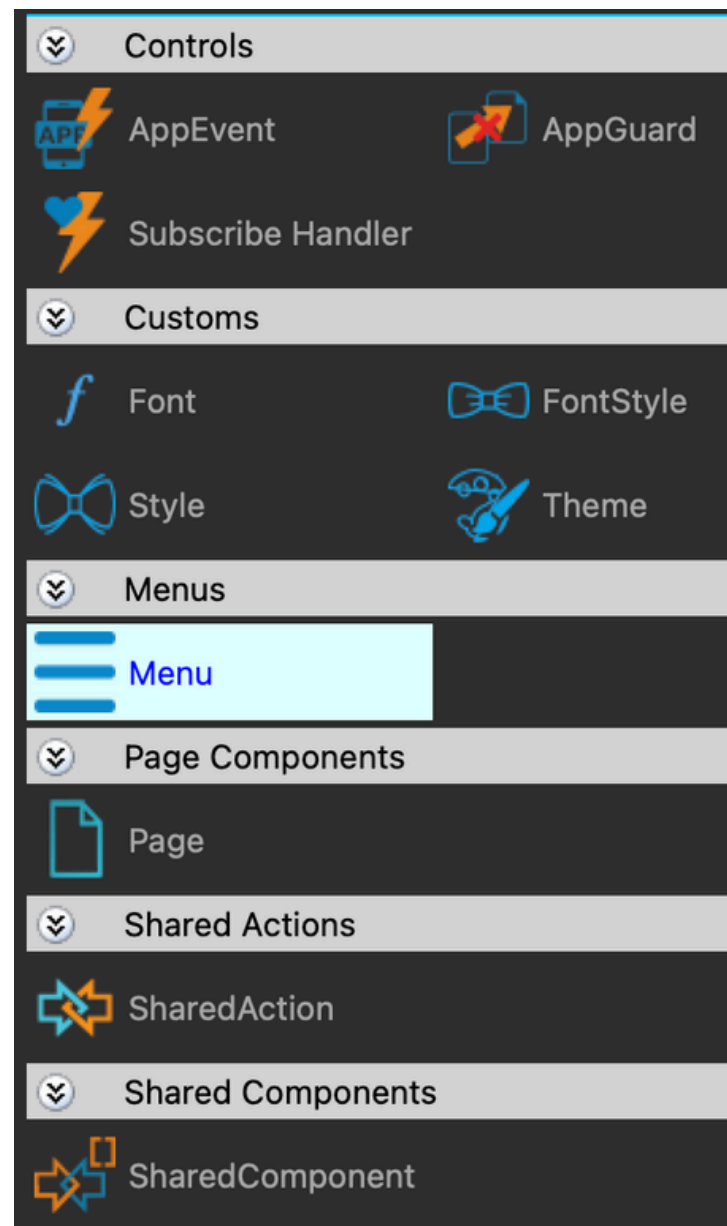
NGX palette



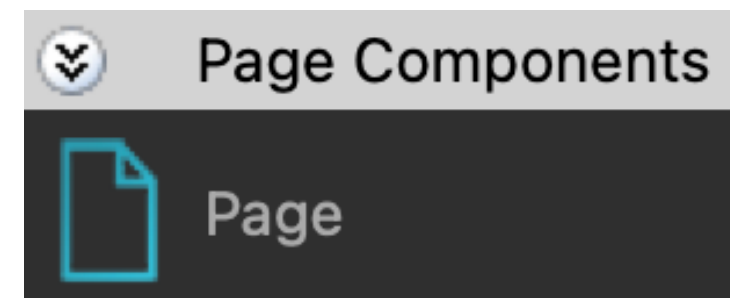
1.4 NGX palette & Components

The NGX palette groups different types of components as categories (HTML, Controls, Customs, Pages, Menus...)

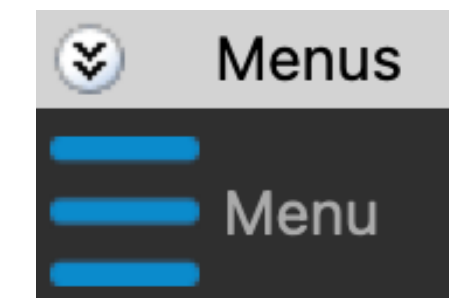
Components available
for the **NgxApp** object



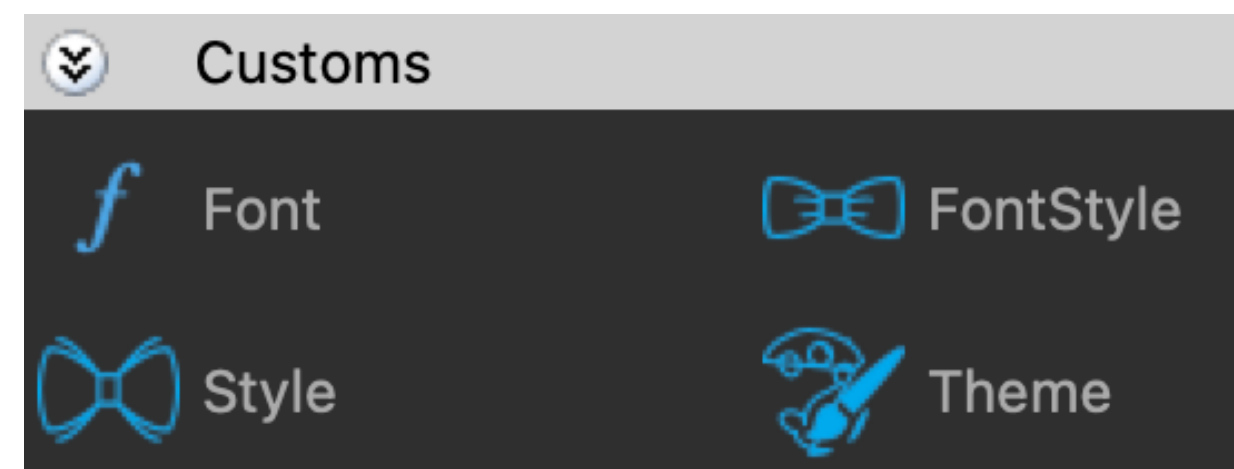
The **Page component** is used to
create new pages for the application.



The **Menu component** is used to
add menus to navigate through pages.



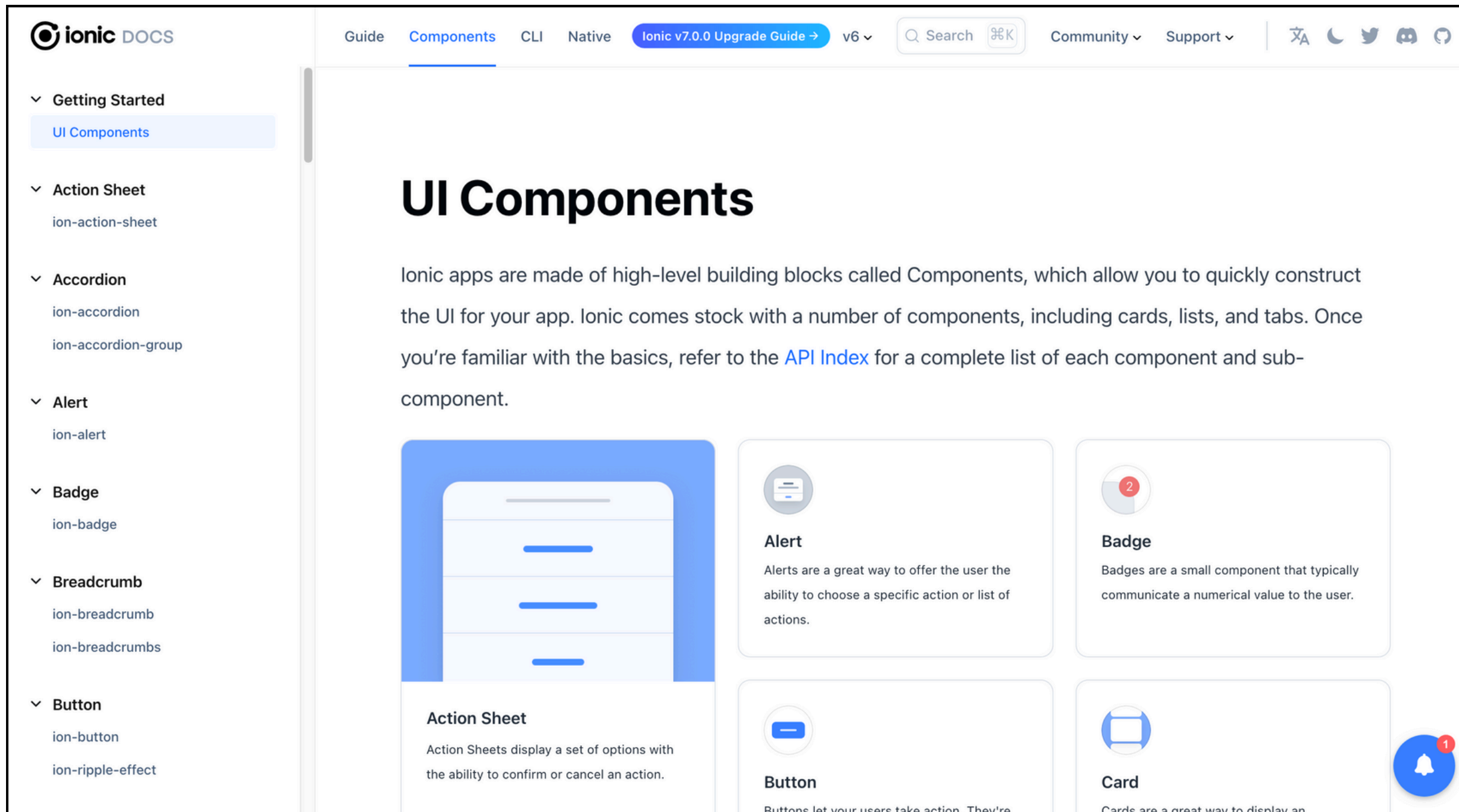
These four components (**Font, FontStyle, Style, Theme**)
are used to **style** the application.



1.5 UI Components

Among **NGX palette components**, **UI Components** are based on **Ionic Web Components**.

<https://ionic-docs-o31kiyk8l-ionic1.vercel.app/docs/v6>



The screenshot shows the Ionic v6 documentation page for UI Components. The page has a sidebar on the left with a list of components: Getting Started, UI Components (highlighted), Action Sheet, Accordion, Alert, Badge, Breadcrumb, and Button. The main content area is titled "UI Components" and contains a paragraph explaining that Ionic apps are made of high-level building blocks called Components. Below the text is a grid of component cards: Action Sheet, Alert, Badge, Button, and Card. Each card features an icon, a title, and a brief description. The Action Sheet card shows a blue modal with three options. The Alert card shows a gray alert box. The Badge card shows a gray circle with a red '2'. The Button card shows a blue button. The Card card shows a white card with a blue header. A blue notification bell icon with a red '1' is in the bottom right corner.

ionic DOCS

Guide **Components** CLI Native [Ionic v7.0.0 Upgrade Guide →](#) v6 ▾

Search 🔍

Community ▾ Support ▾

🌐 🌙 🐦 🗨️ 🔄

▾ **Getting Started**

UI Components

▾ **Action Sheet**

ion-action-sheet

▾ **Accordion**

ion-accordion

ion-accordion-group

▾ **Alert**

ion-alert

▾ **Badge**

ion-badge

▾ **Breadcrumb**

ion-breadcrumb

ion-breadcrumbs

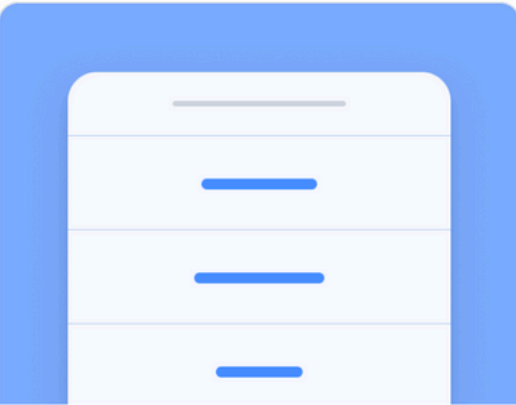
▾ **Button**

ion-button

ion-ripple-effect

UI Components

Ionic apps are made of high-level building blocks called Components, which allow you to quickly construct the UI for your app. Ionic comes stock with a number of components, including cards, lists, and tabs. Once you're familiar with the basics, refer to the [API Index](#) for a complete list of each component and sub-component.



Action Sheet

Action Sheets display a set of options with the ability to confirm or cancel an action.



Alert

Alerts are a great way to offer the user the ability to choose a specific action or list of actions.



Badge

Badges are a small component that typically communicate a numerical value to the user.



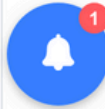
Button

Buttons let your users take action. They're



Card

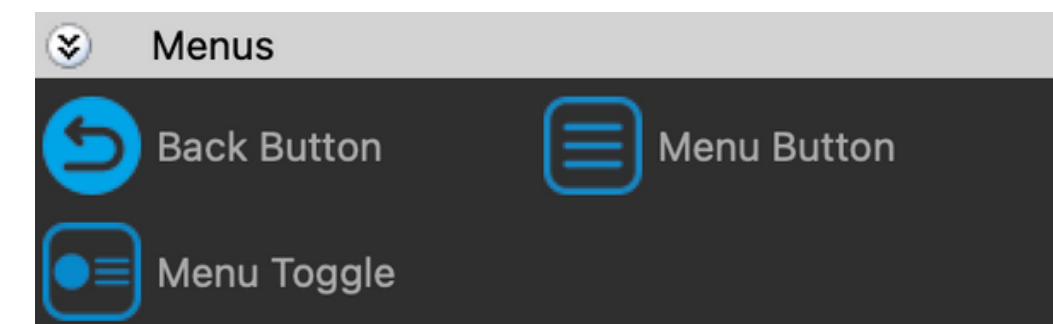
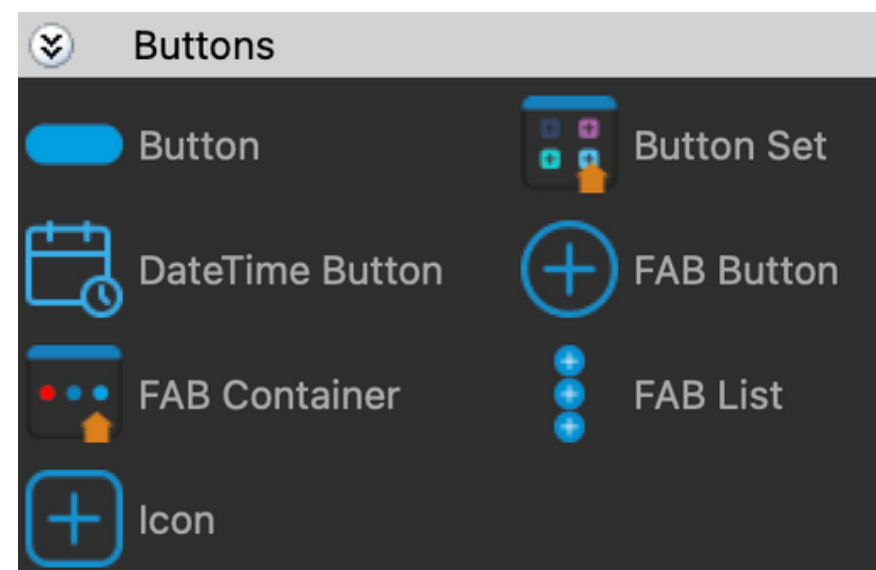
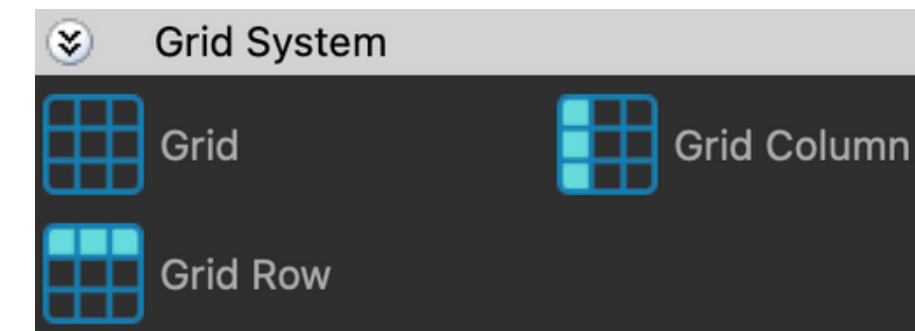
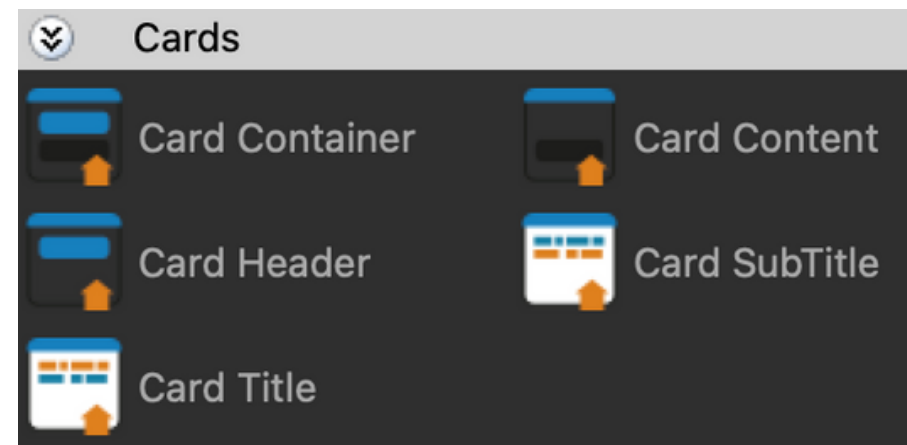
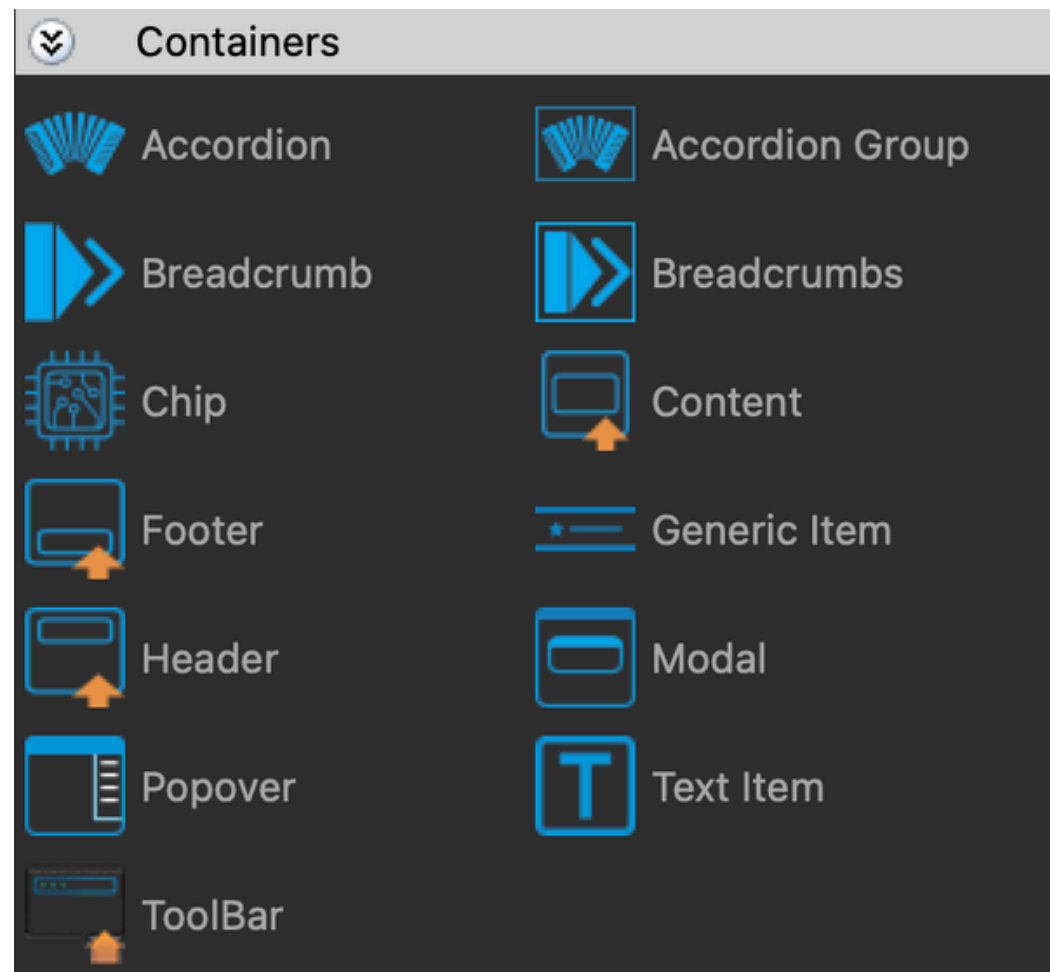
Cards are a great way to display an





1.5 UI Components

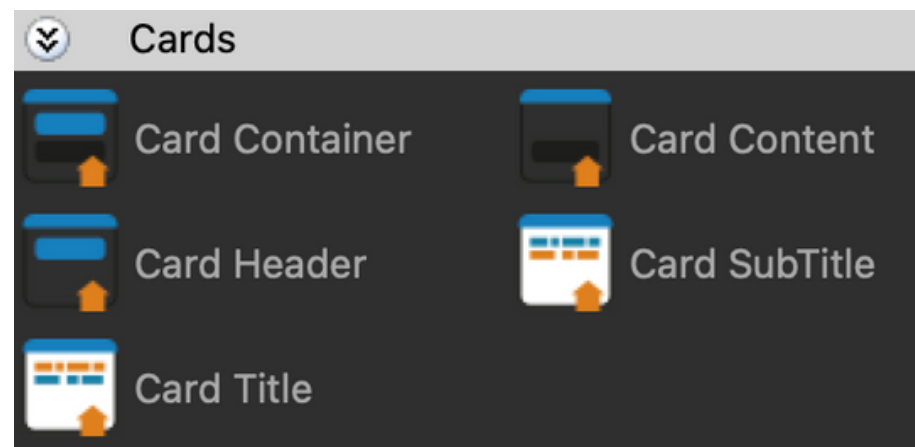
In the **NGX palette**, these categories group **UI Components** based on **Ionic Web Components**.



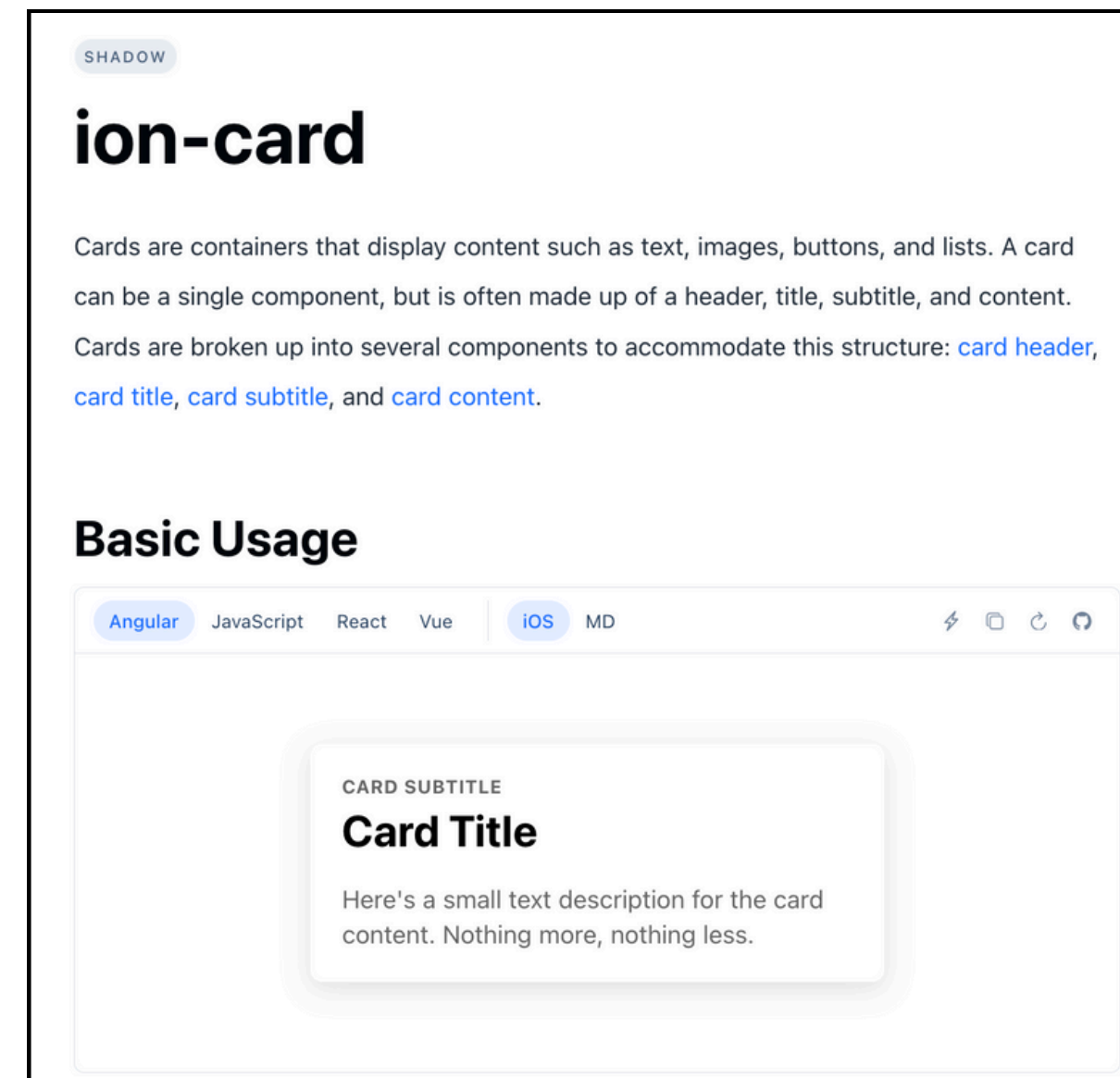
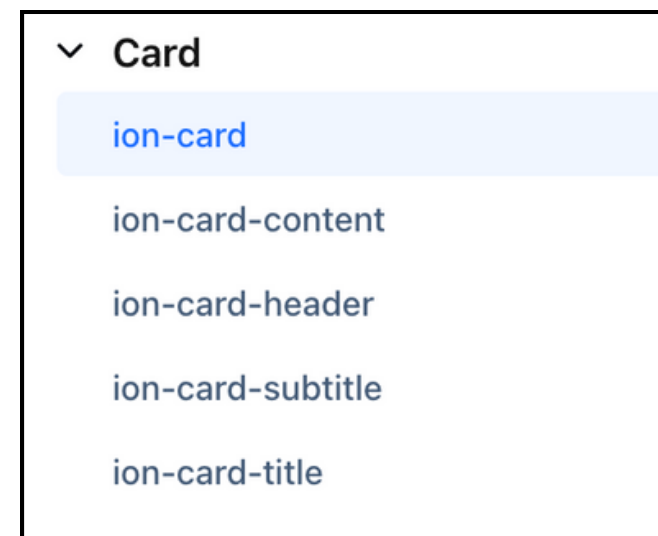
1.5 UI Components

For example, in the **Cards category**, we can find the same components than in the **Ionic documentation**.

NGX palette



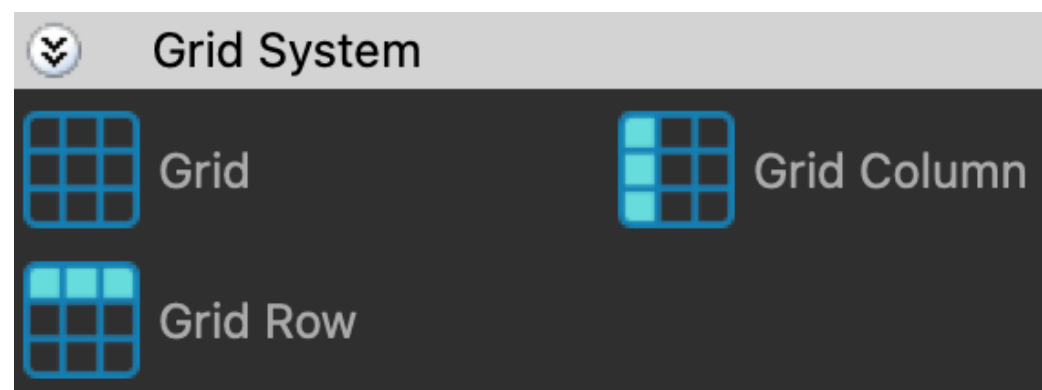
Ionic documentation



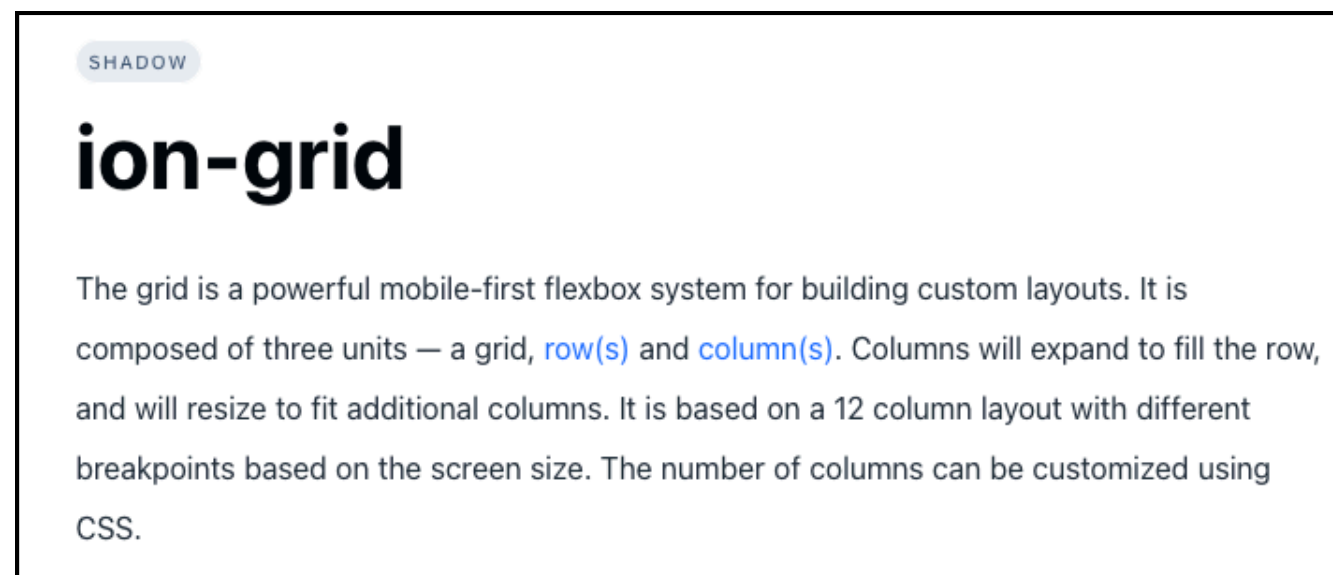
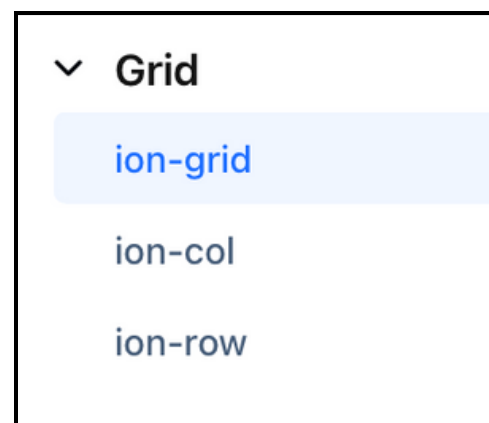
1.5 UI Components

For example, in the **Grid System category**, we can find the same components than in the **Ionic documentation**.

NGX palette



Ionic documentation



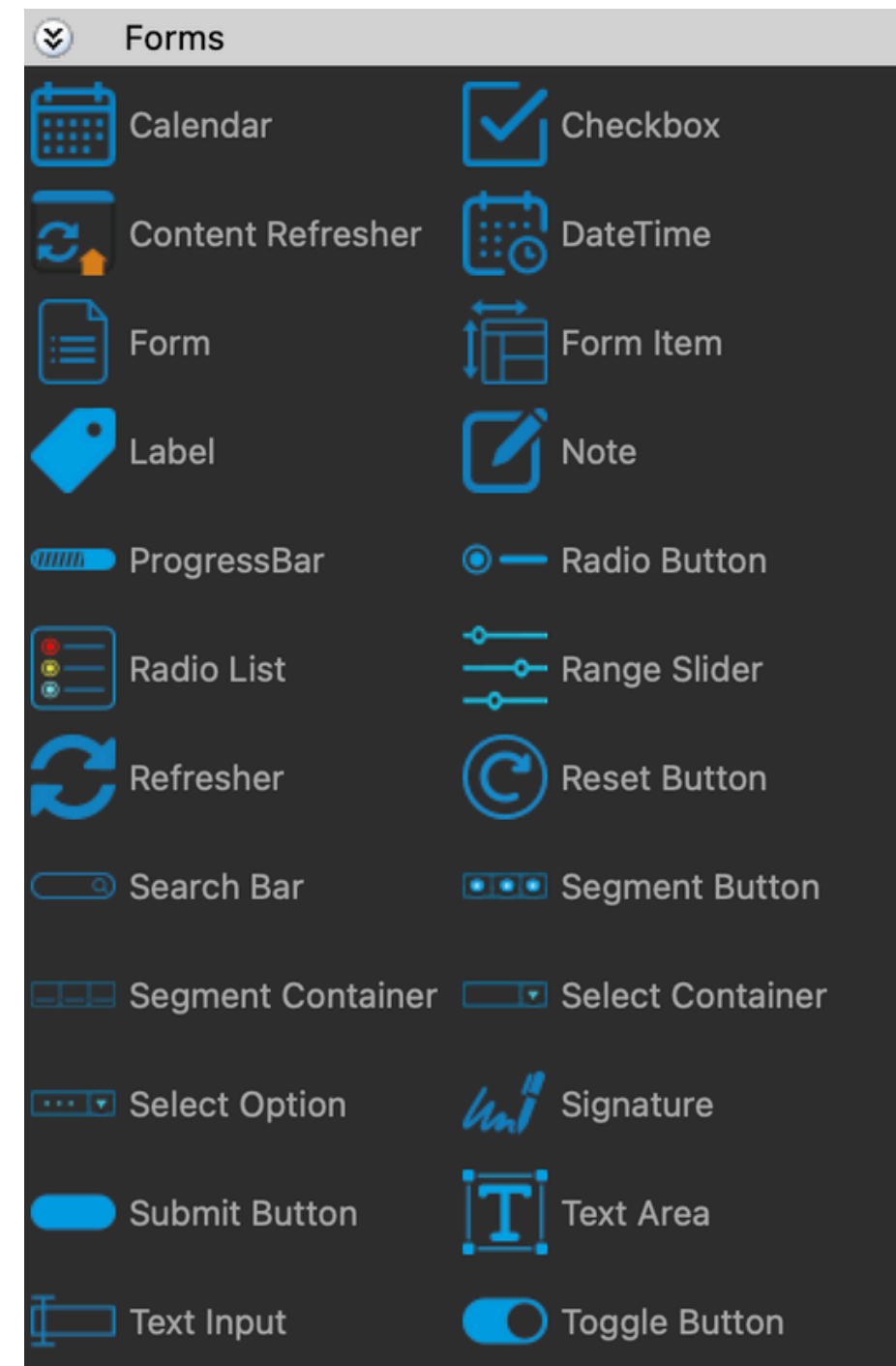
Overview

- Grids act as a container for all rows and columns. Grids take up the full width of their container, but adding the `fixed` property will set the width based on the screen size, see [Fixed Grid](#) below.
- Rows are horizontal groups of columns that line the columns up properly.
- Content should be placed within columns, and only columns may be immediate children of rows.
- The `size` property indicates the number of columns to use out of the default 12 per row. So, `size="4"` can be added to a column in order to take up 1/3 of the grid, or 4 of the 12 columns.
- Columns without a value for size will automatically have equal widths. For example, four columns will each automatically be 25% wide.
- Column widths are set as a percentage, so they're always fluid and sized relative to their parent element.
- There is padding between individual columns. However, the padding can be removed from the grid and columns by adding the `ion-no-padding` class to the grid. See the [CSS Utilities](#) for more styles that can be applied to the grid.
- There are five grid tiers, one for each responsive breakpoint: all breakpoints (extra small), small, medium, large, and extra large.
- Grid tiers are based on minimum widths, meaning they apply to their tier and all those larger than them (e.g., `size-sm="4"` applies to small, medium, large, and extra large devices).
- Grids can be customized via CSS variables. See [Customizing the Grid](#).

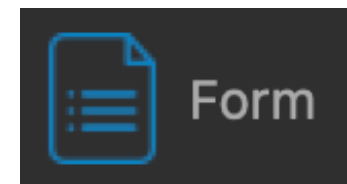


1.5 UI Components

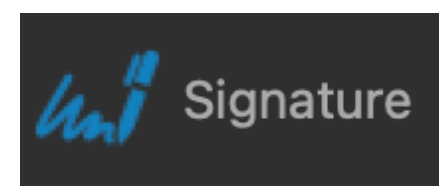
Some categories mix **UI Components** with other types of components.



For example, in the **Forms category**, all components are **UI components**, except for **Form** (HTML Component)

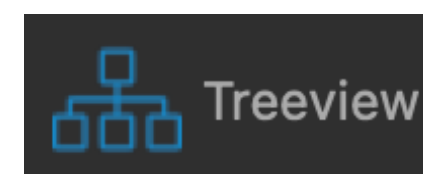
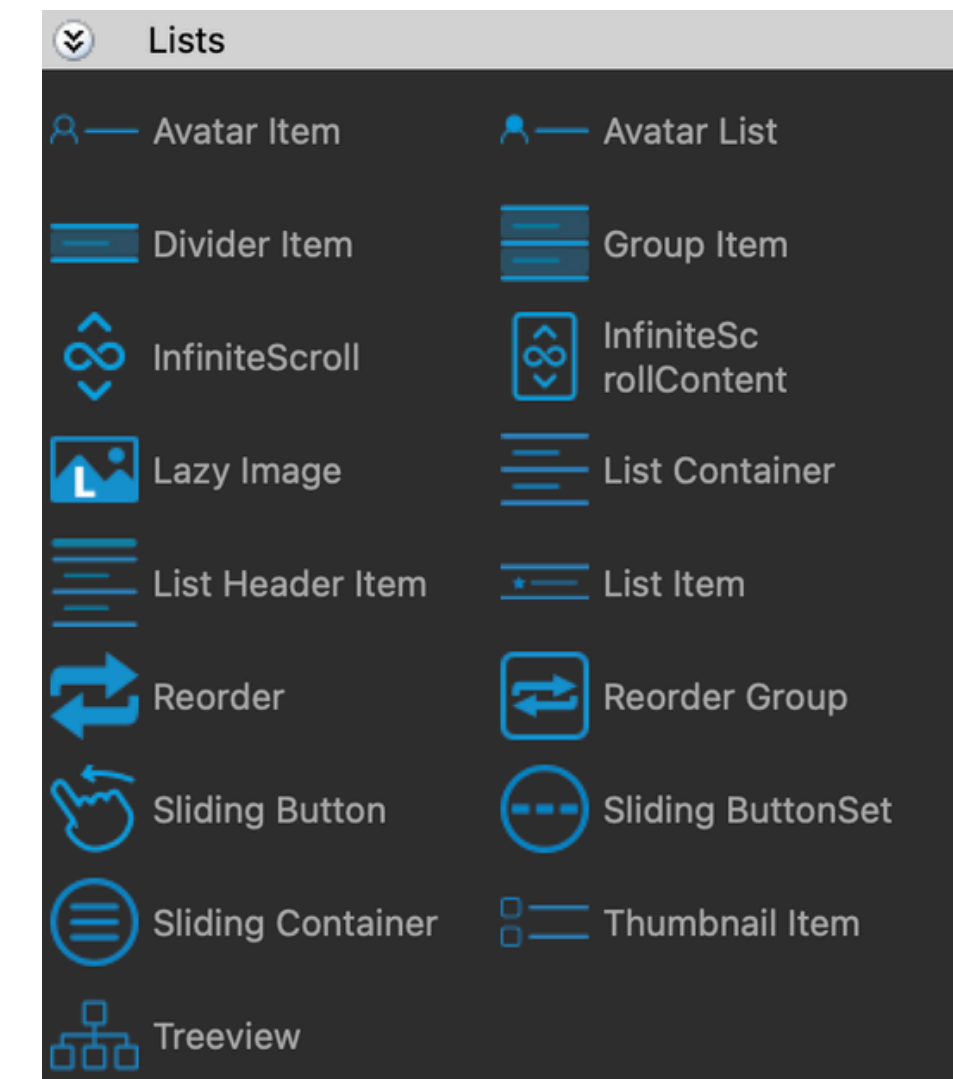


and **Signature** (uses an Angular2 component based on the **Signature Pad JavaScript library**)



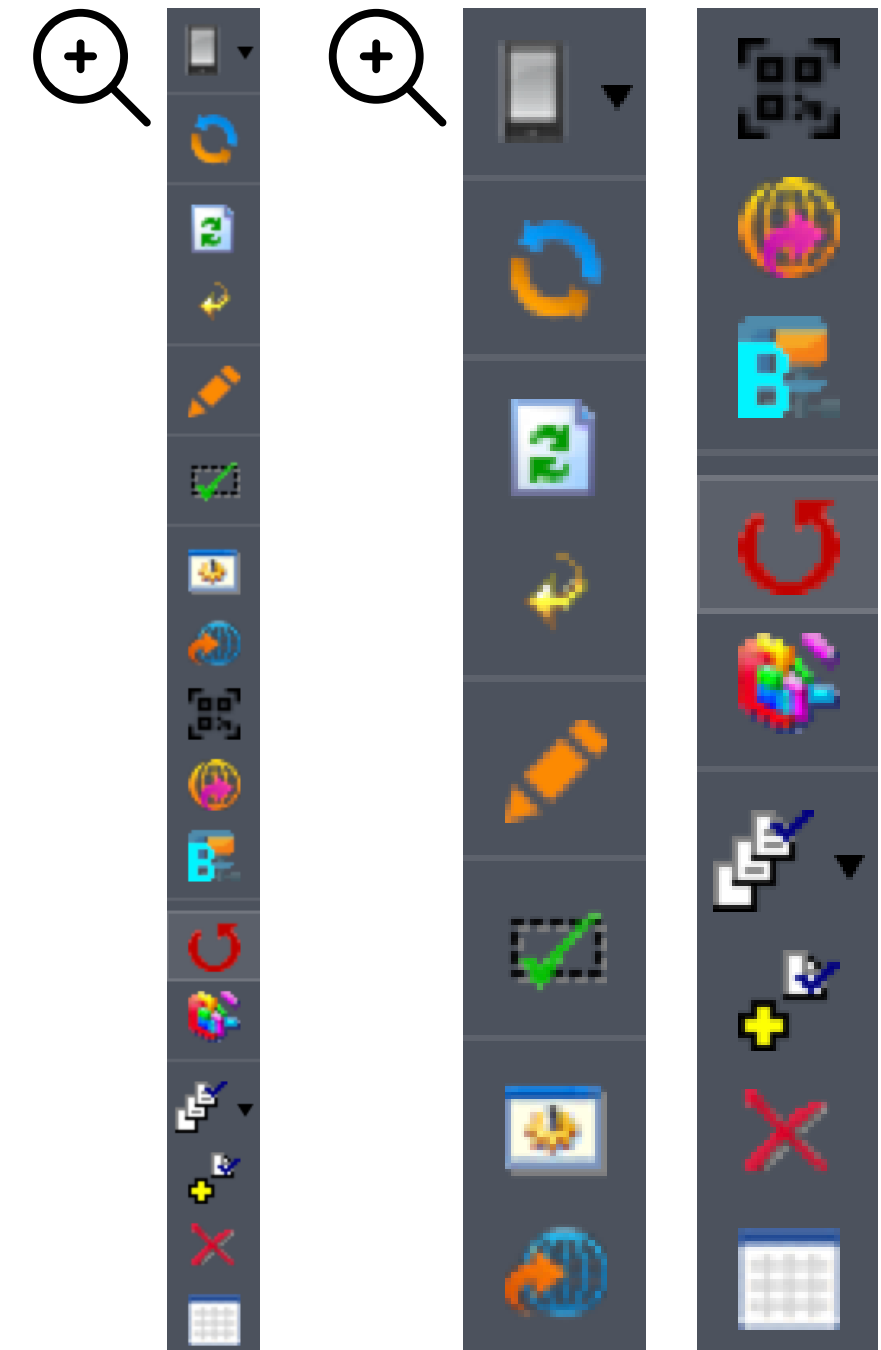
Same thing for the **Forms category**.

All components are **UI components**.



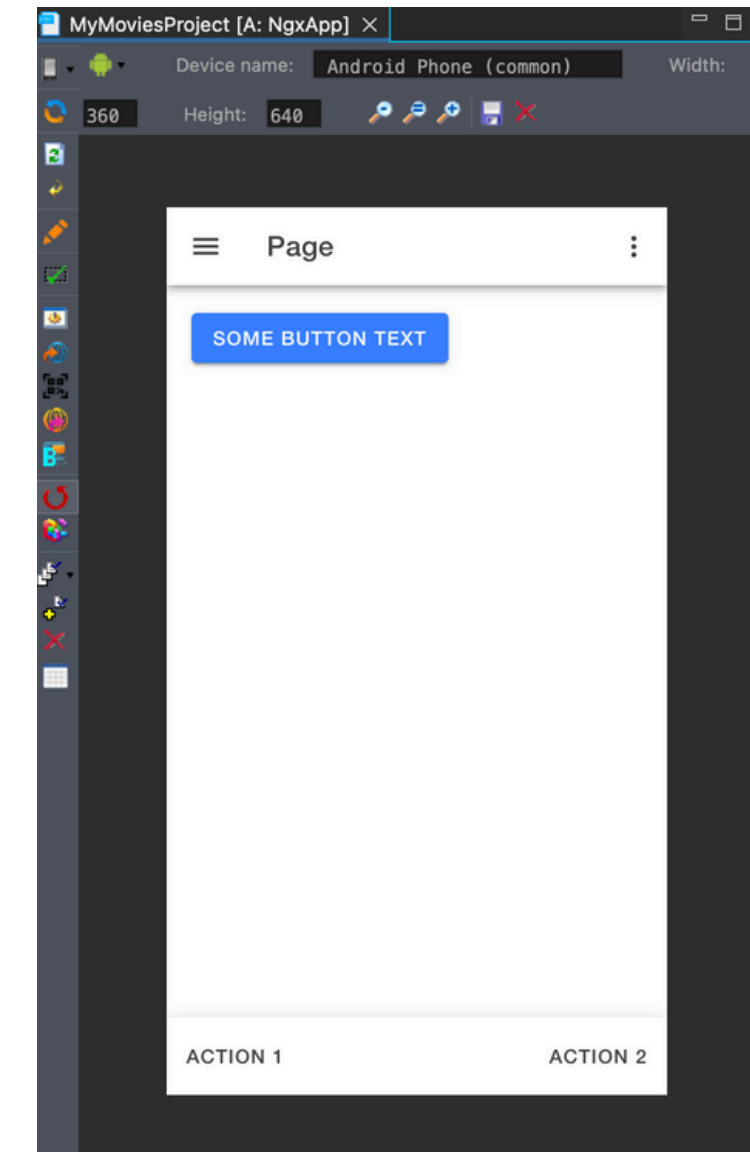
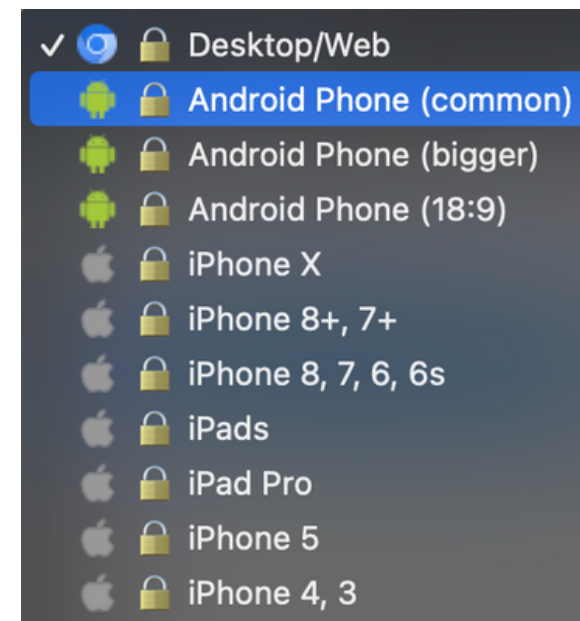
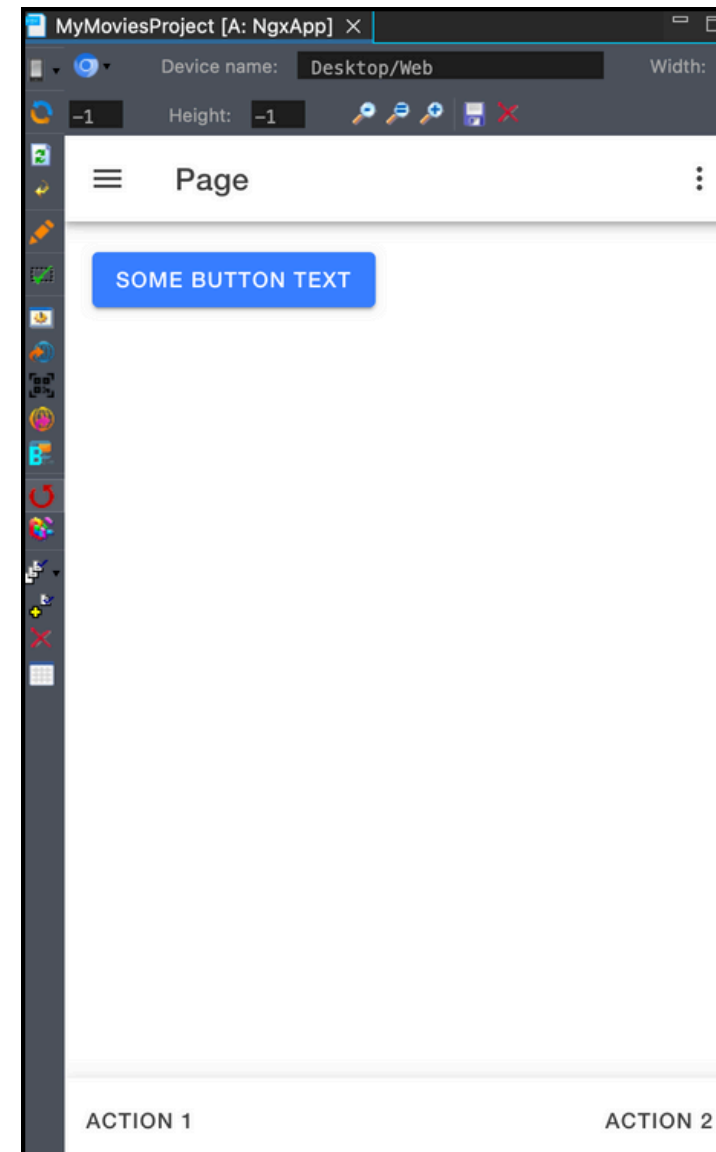
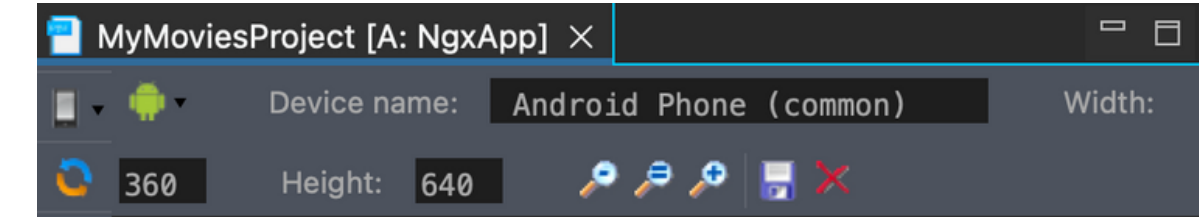
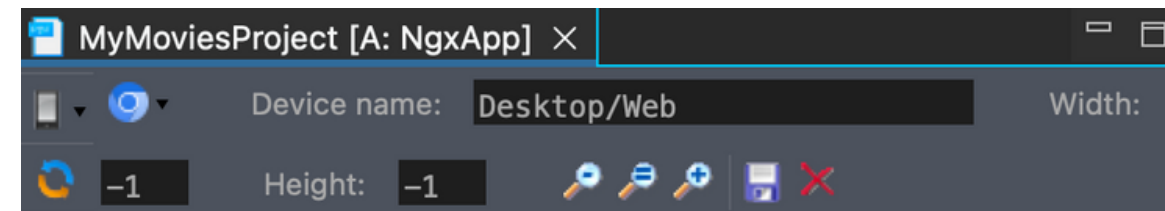
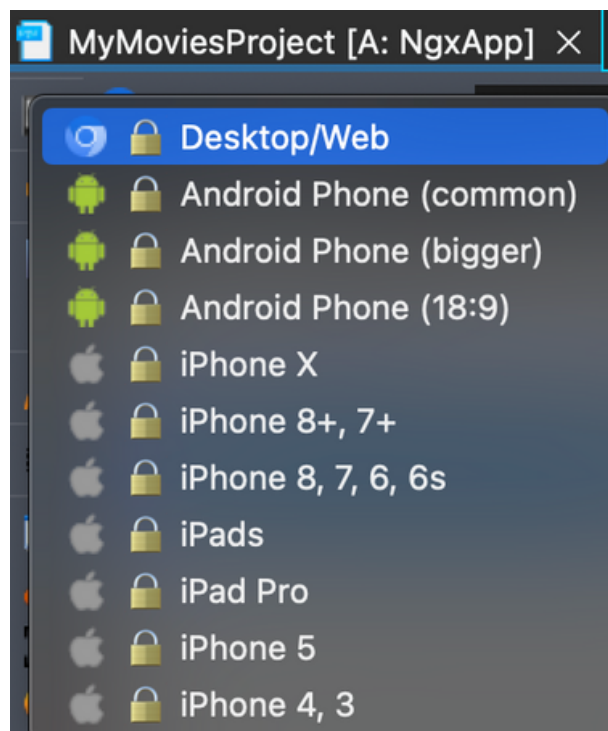
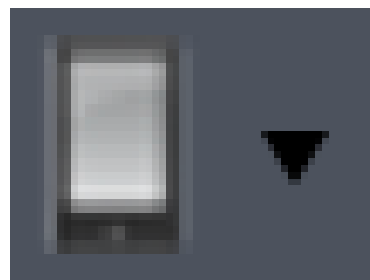
Except for **Treeview** (based on the **angular-tree-component library**).





1.6 Application Viewer Controls

Devices : Selects device viewport. Click to toggle the custom device bar.

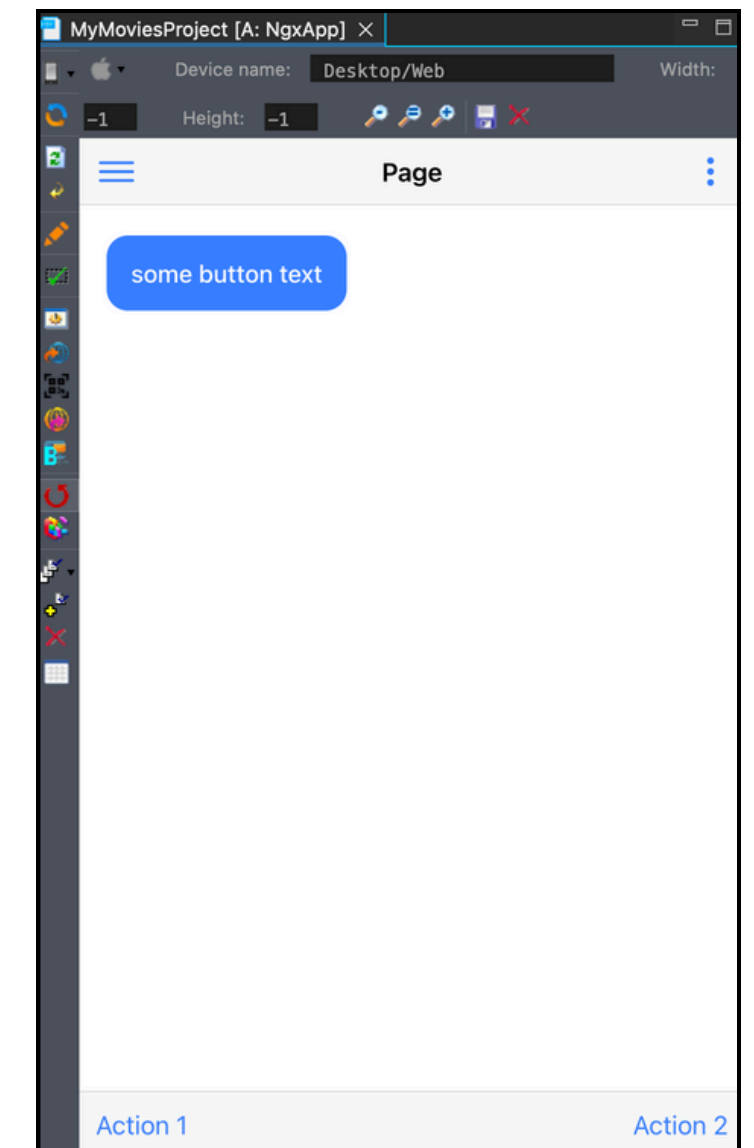
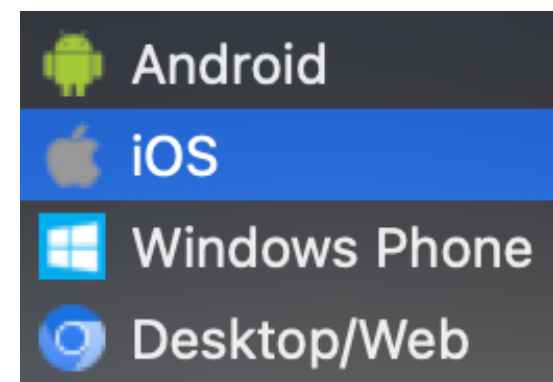
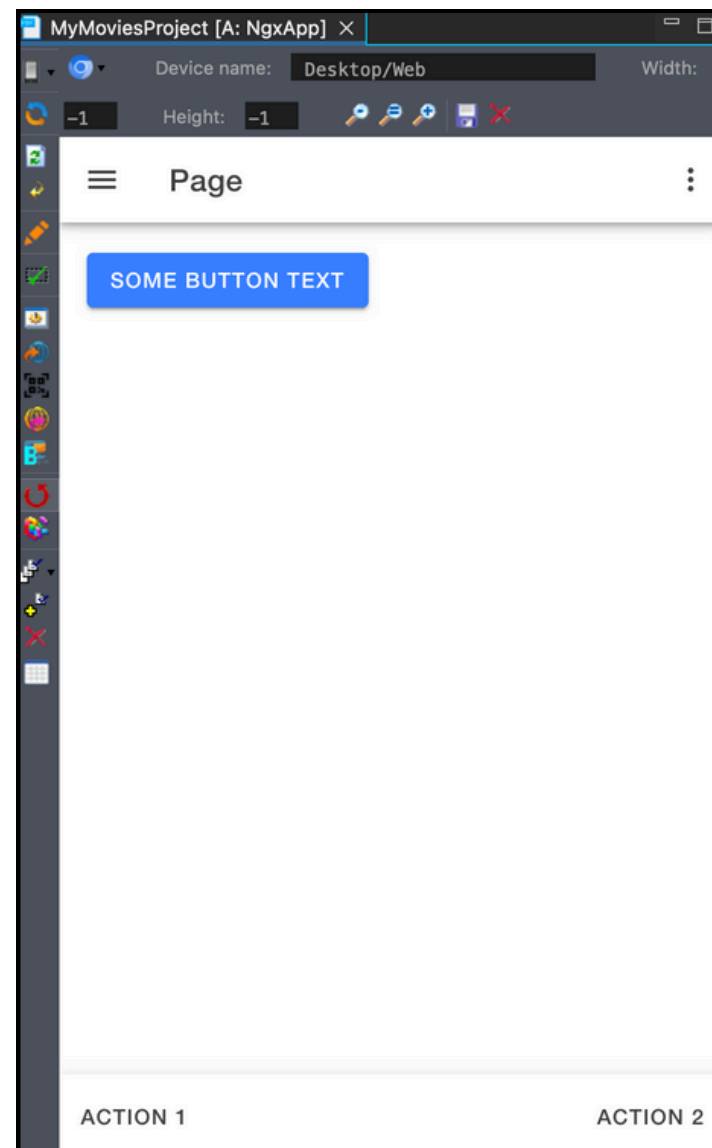
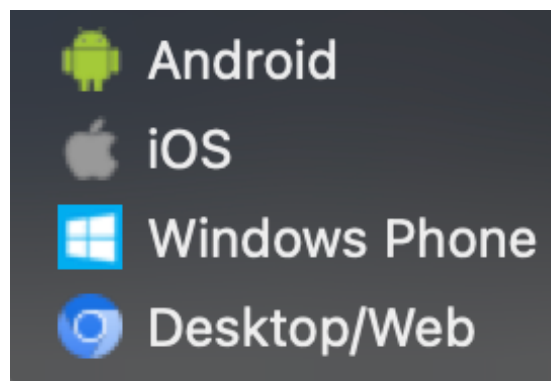
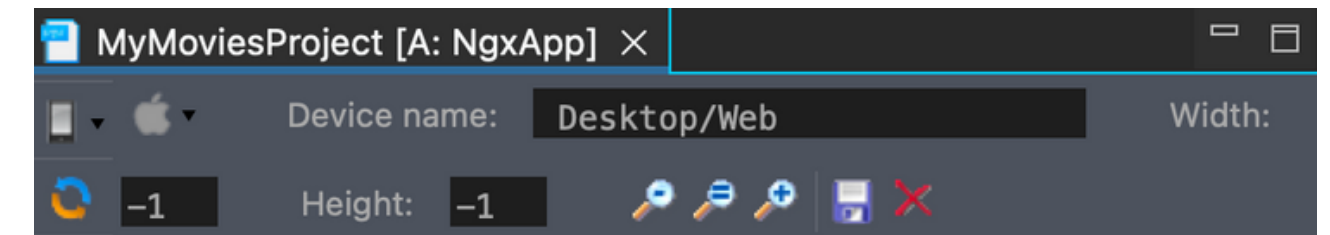
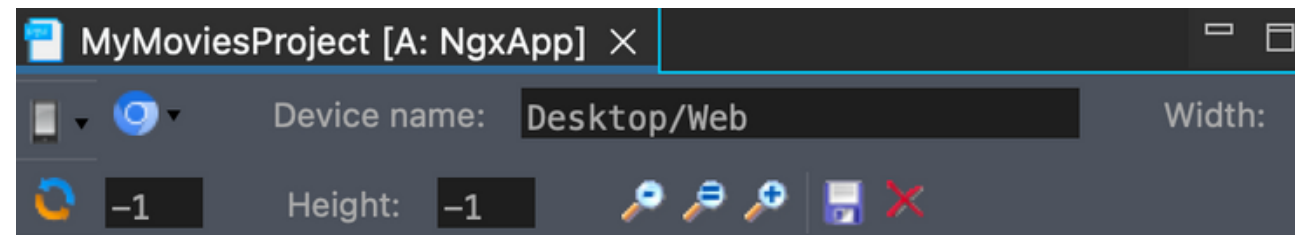
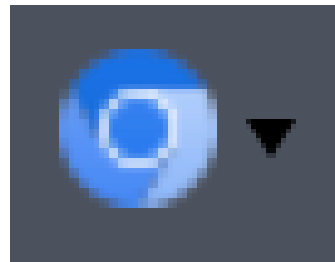


If you select **Android phone**, the application will appear as such in the viewer.



1.6 Application Viewer Controls

Selects device OS. Click to toggle the custom device OS bar.



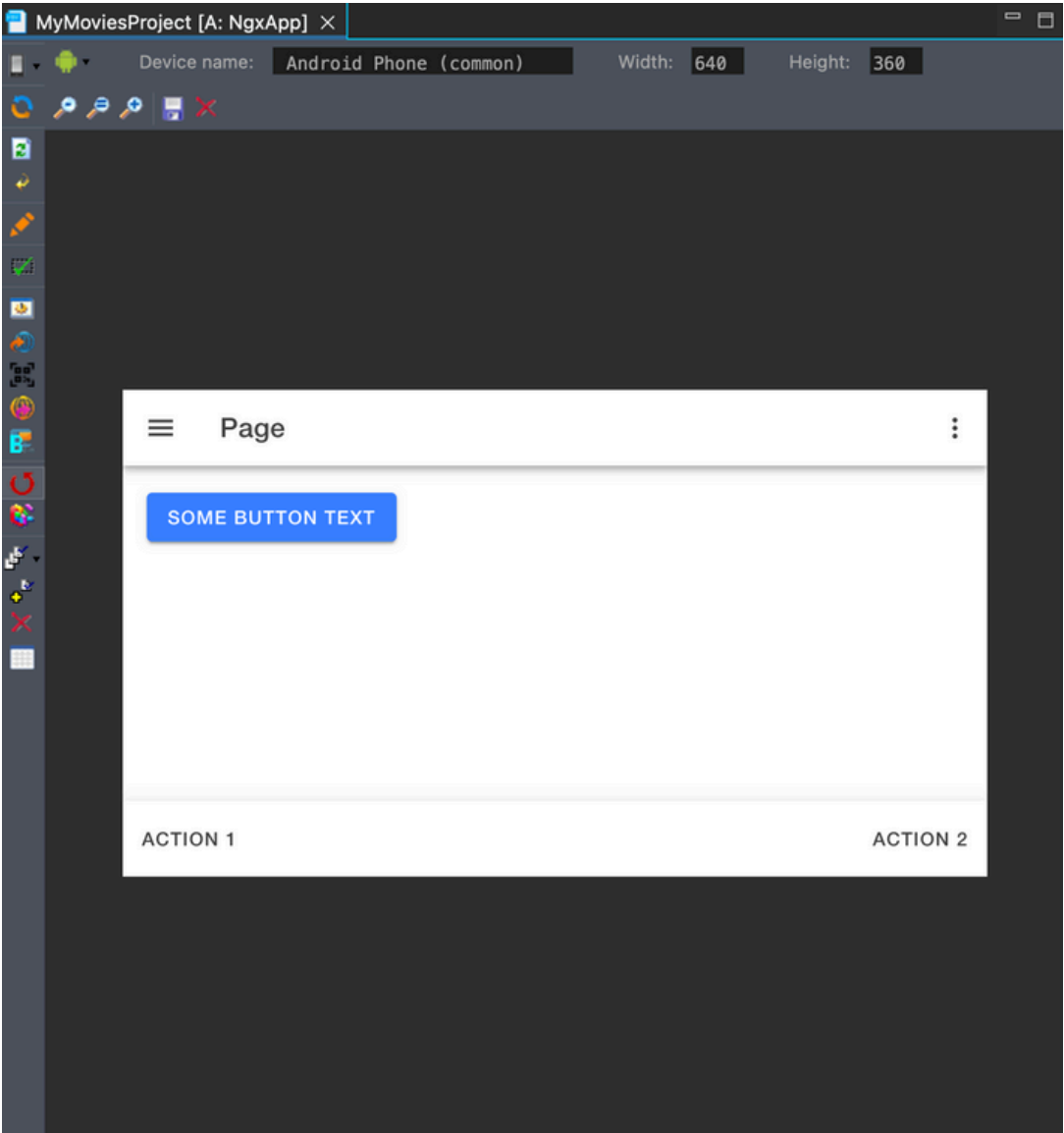
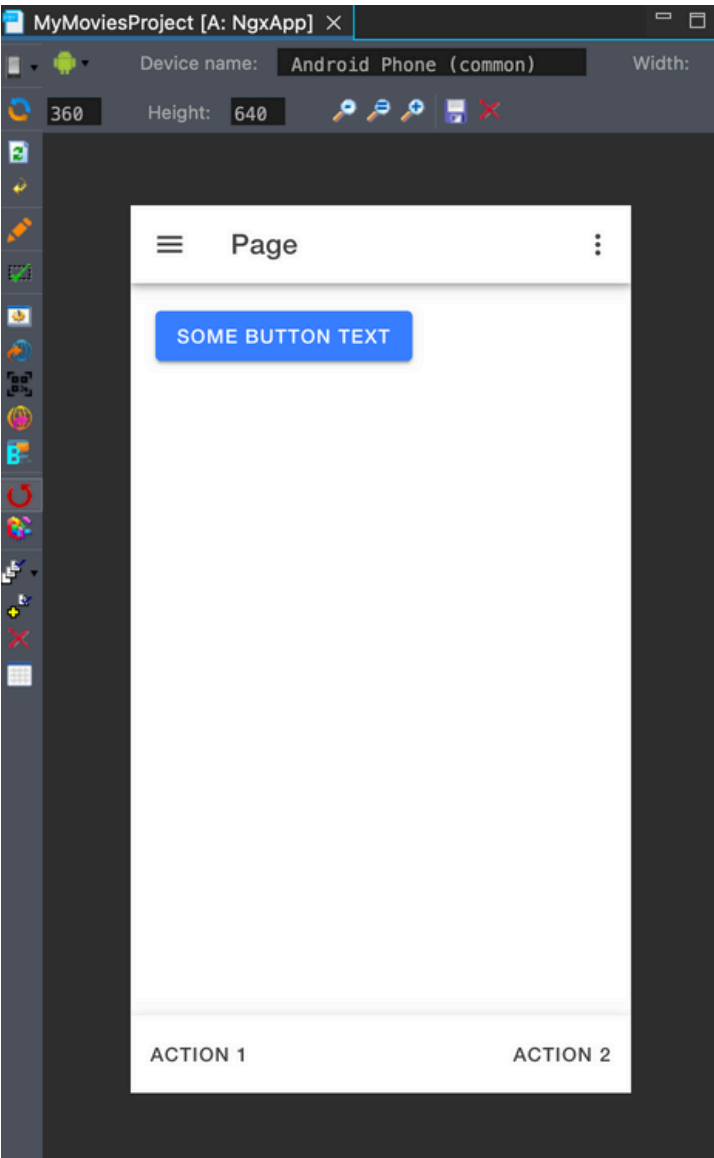
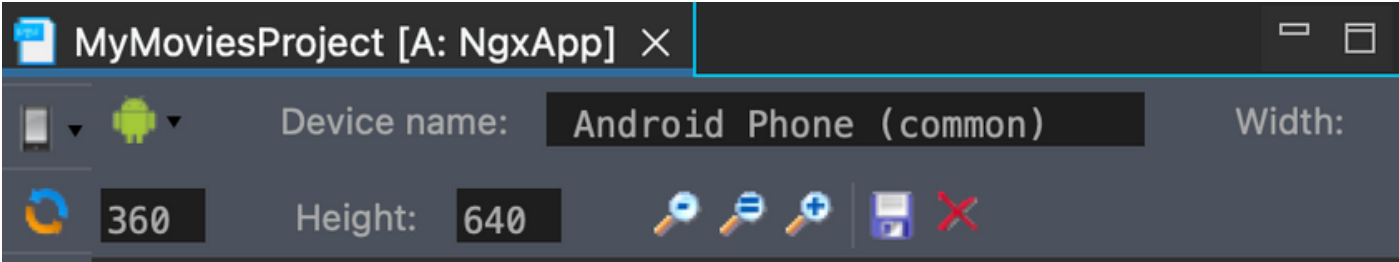
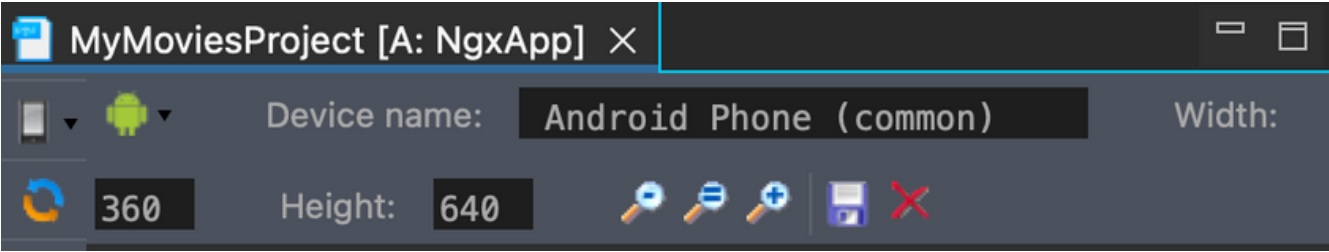
If you select **iOS**,
the application will appear as
such in the viewer.



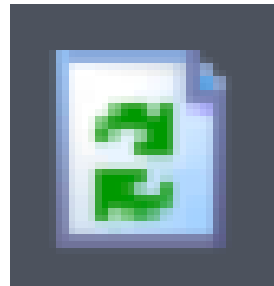
1.6 Application Viewer Controls



Orientation : Changes device orientation (Landscape / portrait)

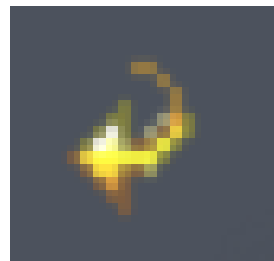


1.6 Application Viewer Controls



Refresh

Relaunches the app on the current page



Back

Goes back on page (Like the back button on a device)

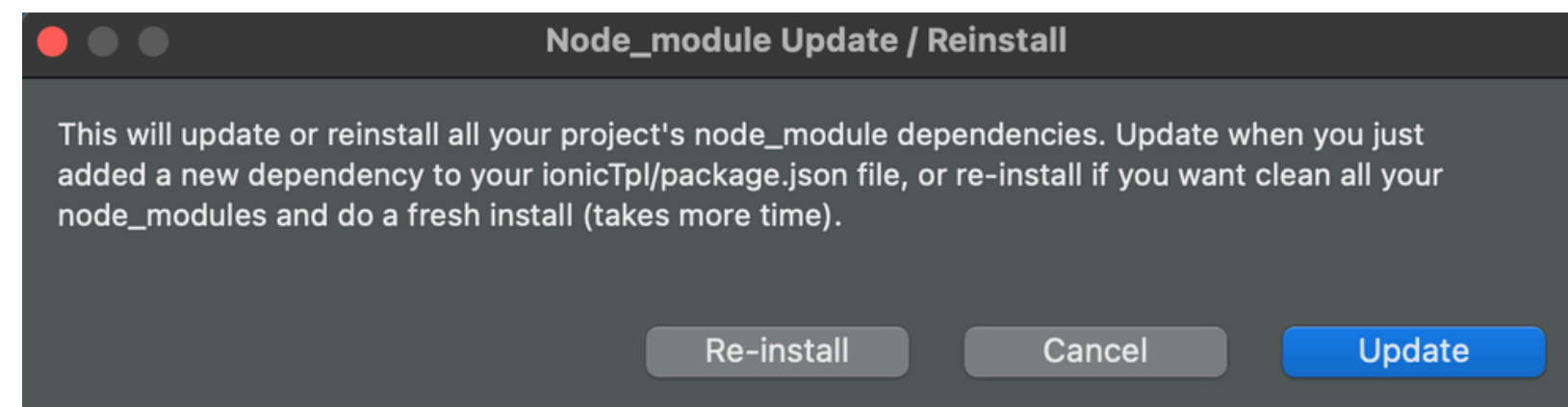


Manage Modules

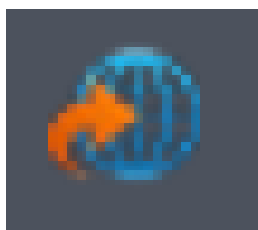
To update or re-install NPM node_modules for this app.

Can be useful for corrupted NPM node_modules installations.

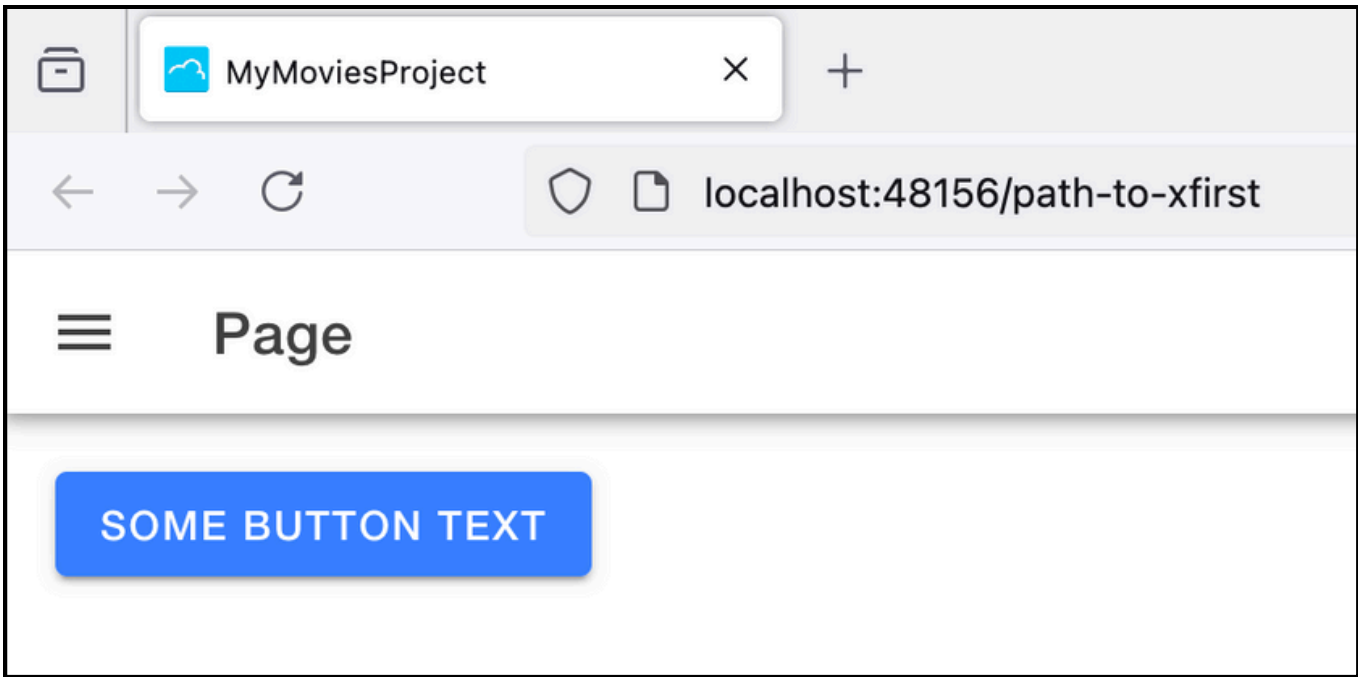
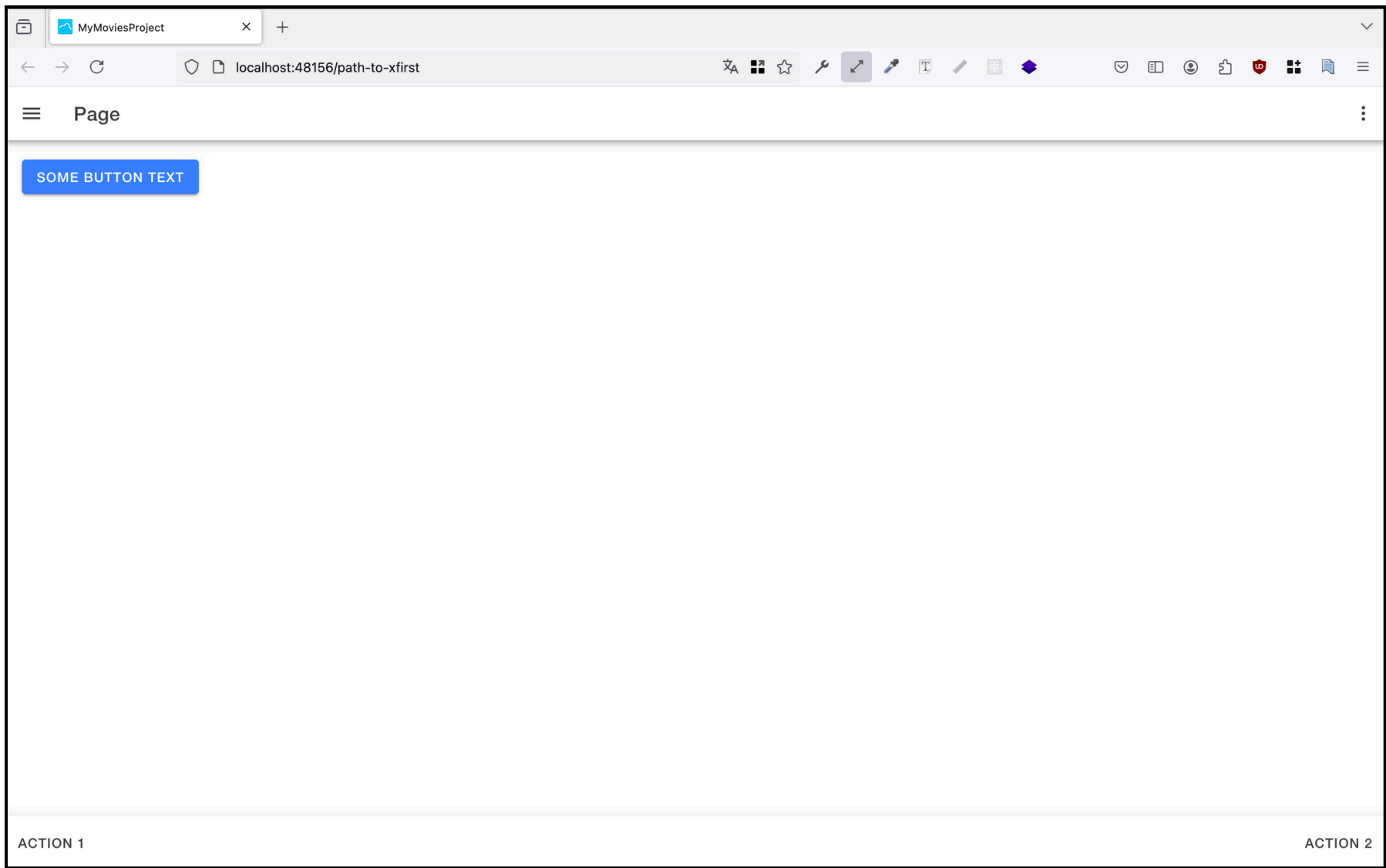
Click to toggle the panel.



1.6 Application Viewer Controls



Open in default browser
Launches application in your default external browser



1.6 Application Viewer Controls



Headless build

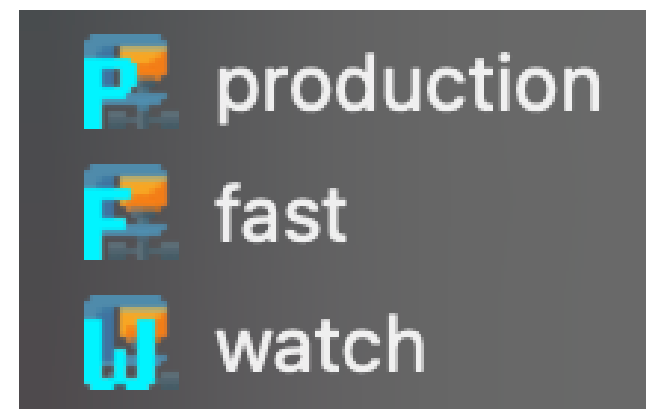
Launches the application in your default external browser (like **Open in default browser**) and deactivates live build in the **Application Viewer**.

Live build is only performed in the external browser, saving memory and CPU consumption.



Build mode

Click to toggle the build menu.



3 modes are available :

Production, Fast or Watch

(more details later in the tutorial).



Autobuild

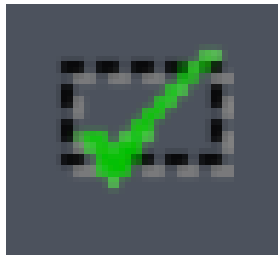
Toggle the autobuild.

If selected, the app will be built and refreshed automatically each time you modify a component in the application.

If not, the autobuild is disabled.

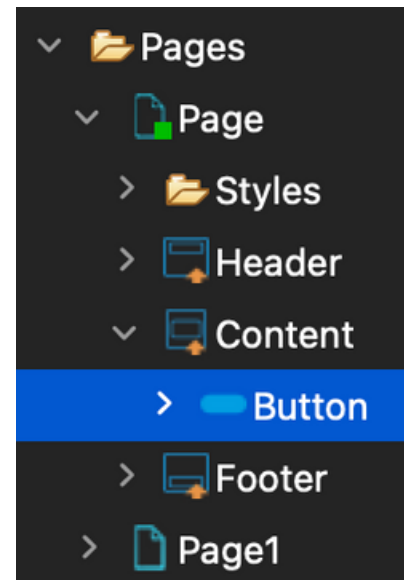
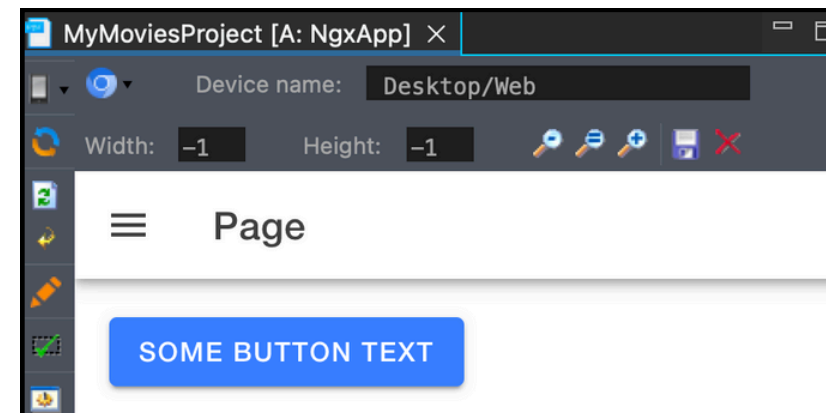
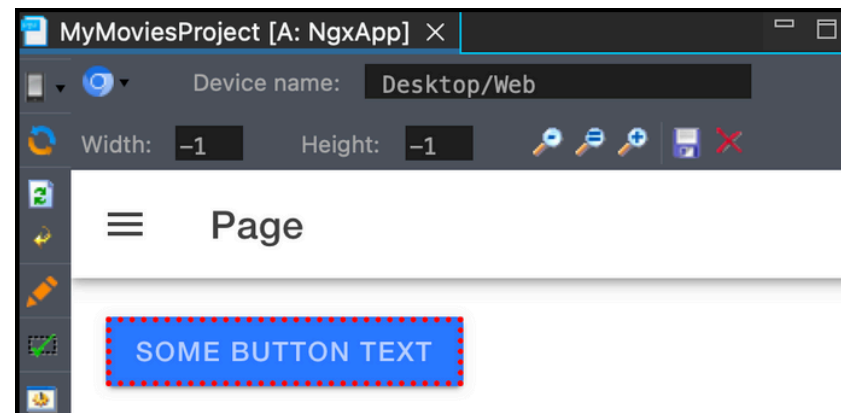


1.6 Application Viewer Controls



Remove highlight :

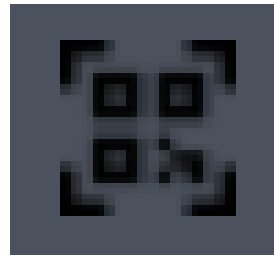
Removes any highlighted object



Objects are highlighted
when you right click on them in the treeview.



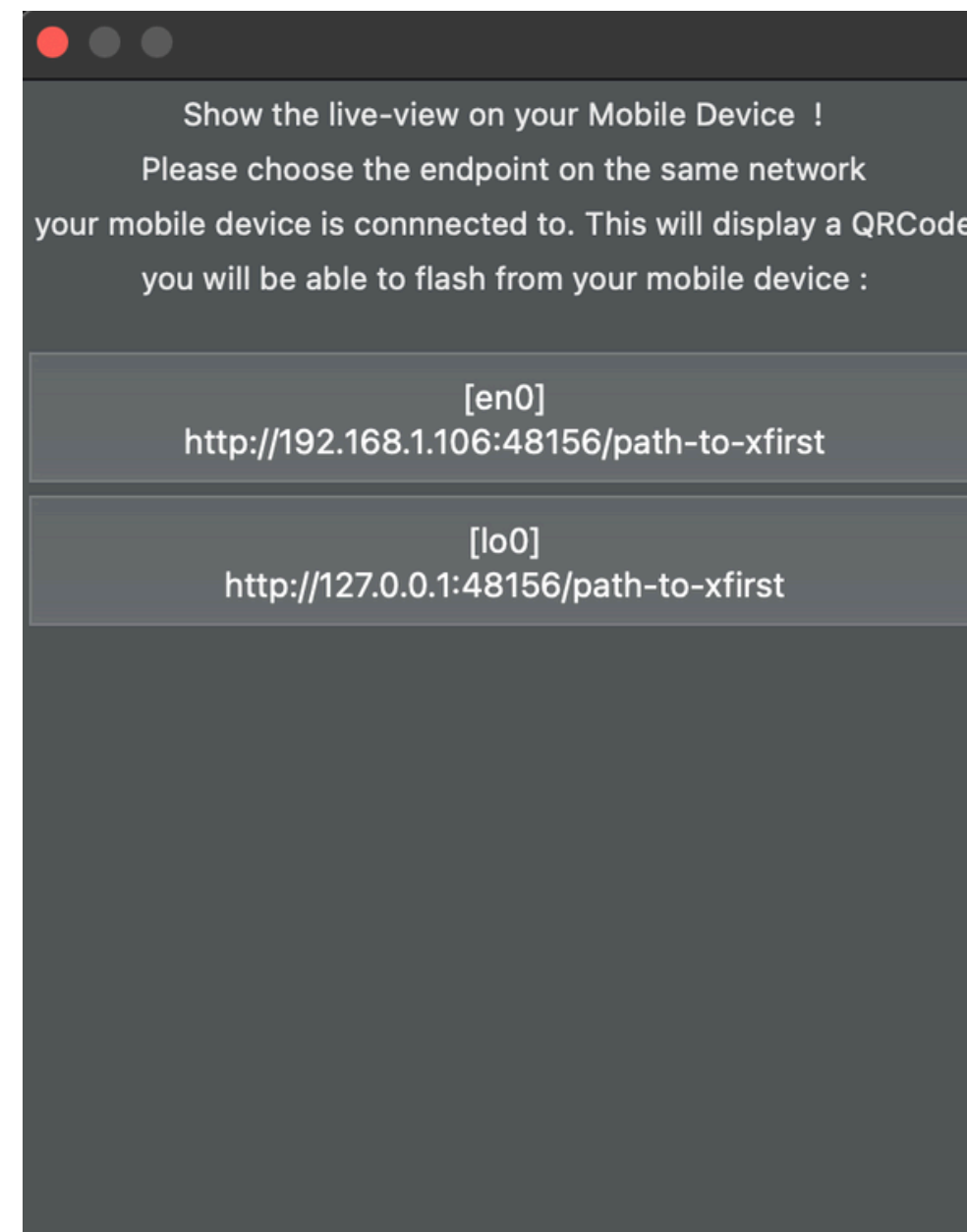
1.6 Application Viewer Controls



Show QR Code : Click to toggle the panel.

Displays a QR code to show the live view on a mobile device.

Click to toggle the panel.



1.6 Application Viewer Controls



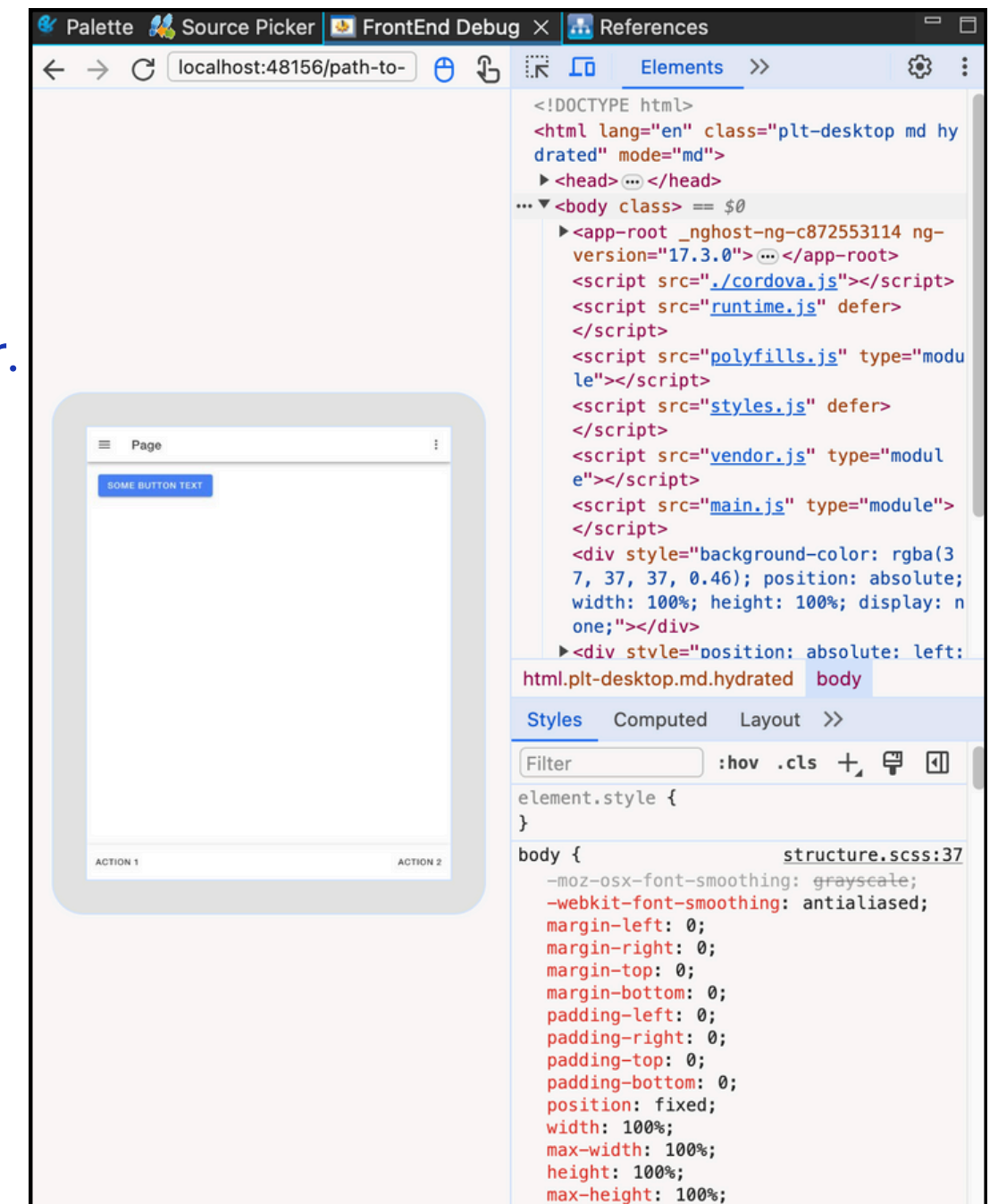
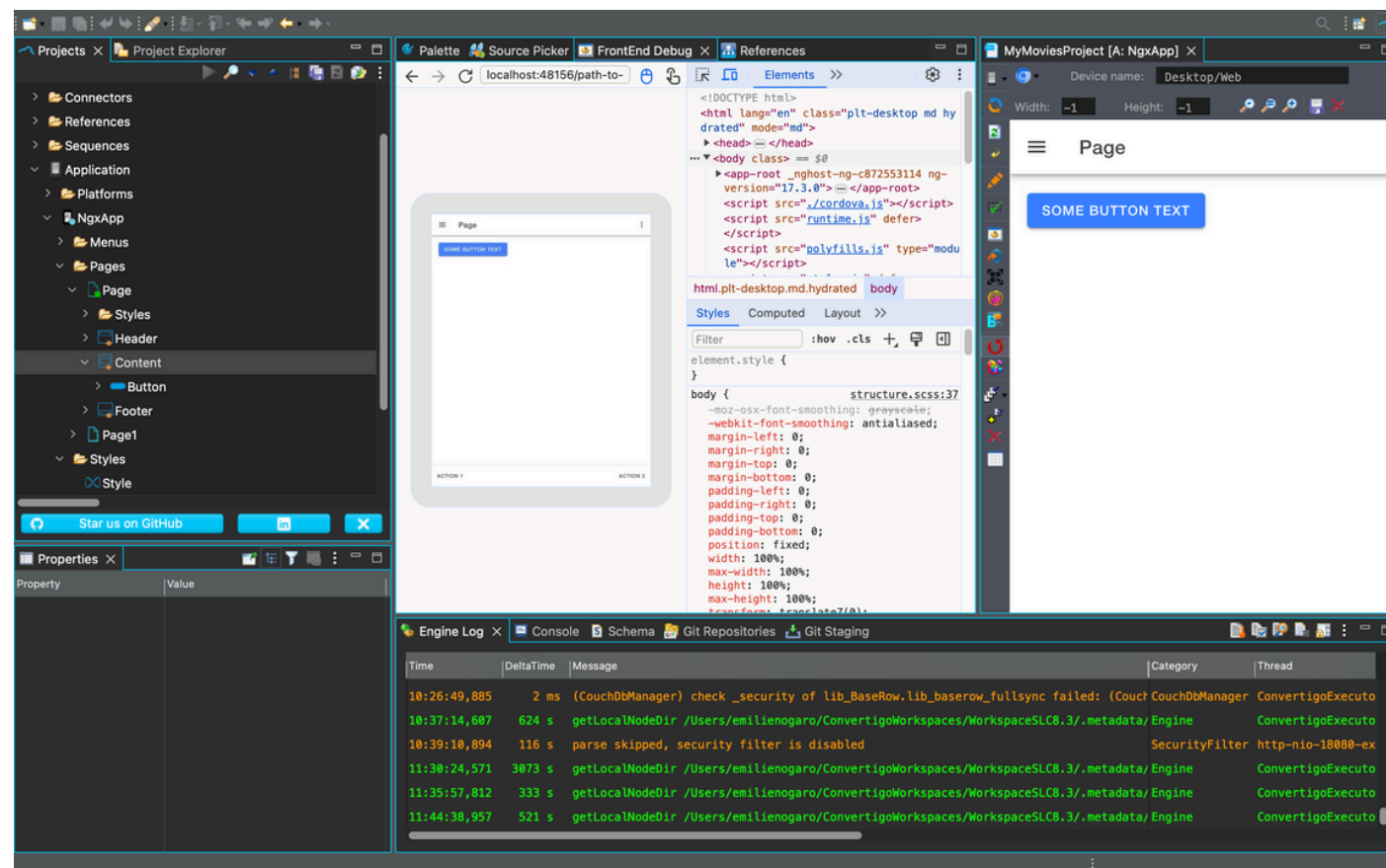
Debug

Shows Debugger view in the Studio.

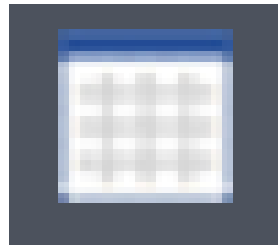
The **FrontEnd debugger** is used to **monitor and debug** your application.

With roughly the same features and usage than the **Google Chrome debugger**, it can be used to adjust styles, margins, and components placements precisely.

It also shows **network interactions** between the device and a Convertigo Server.



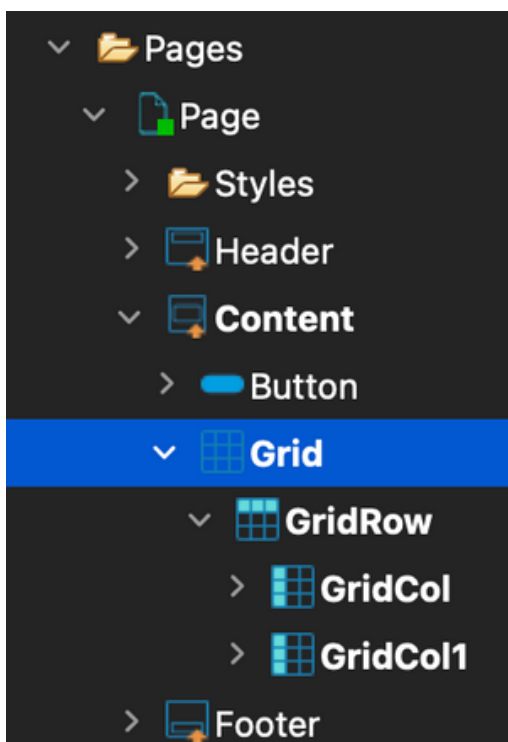
1.6 Application Viewer Controls



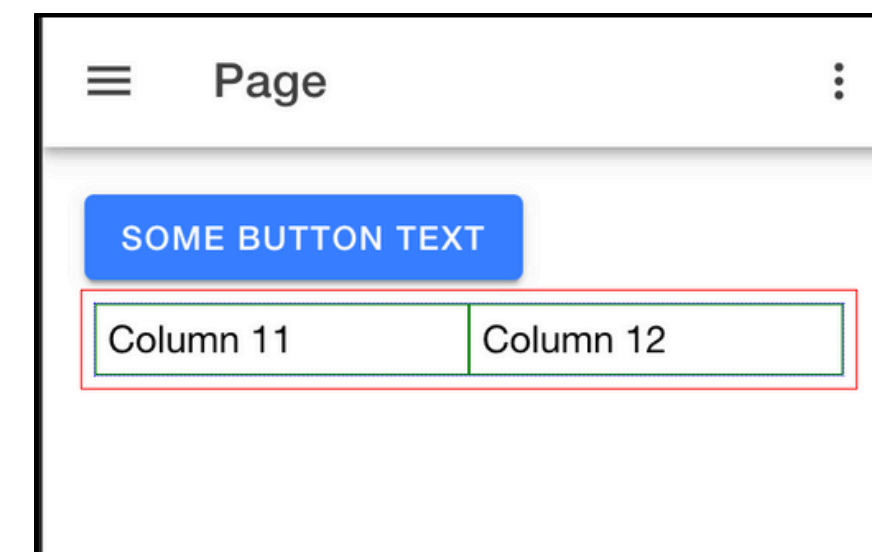
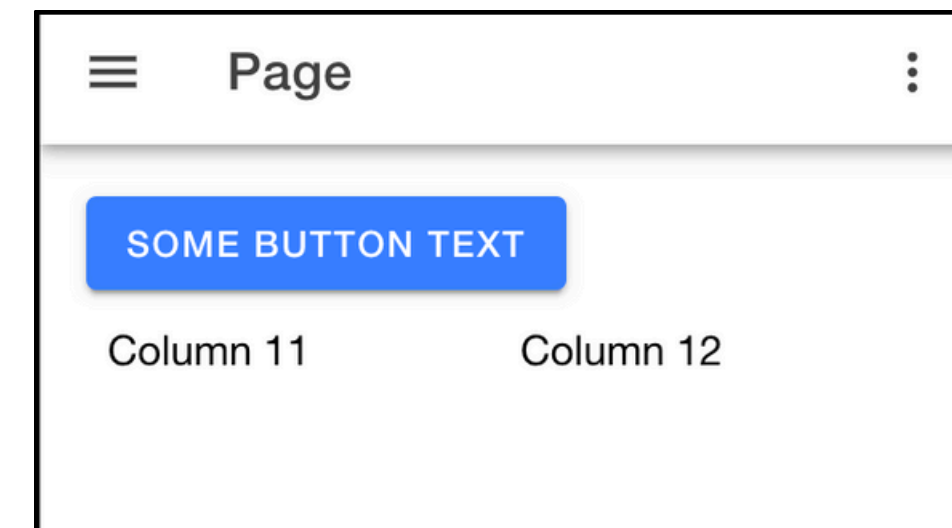
Shows all grids or current selected



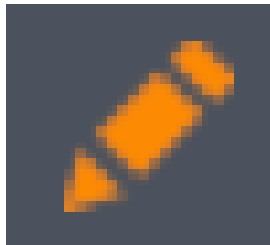
The **Grid component**, based on flexbox, is composed of three units :
grid, rows and columns.
(more details later in the tutorial)



When a grid is present in a page, this control adds a **red border** to the **grid** and a **blue border** to the **columns** to show the **padding** of grid elements.

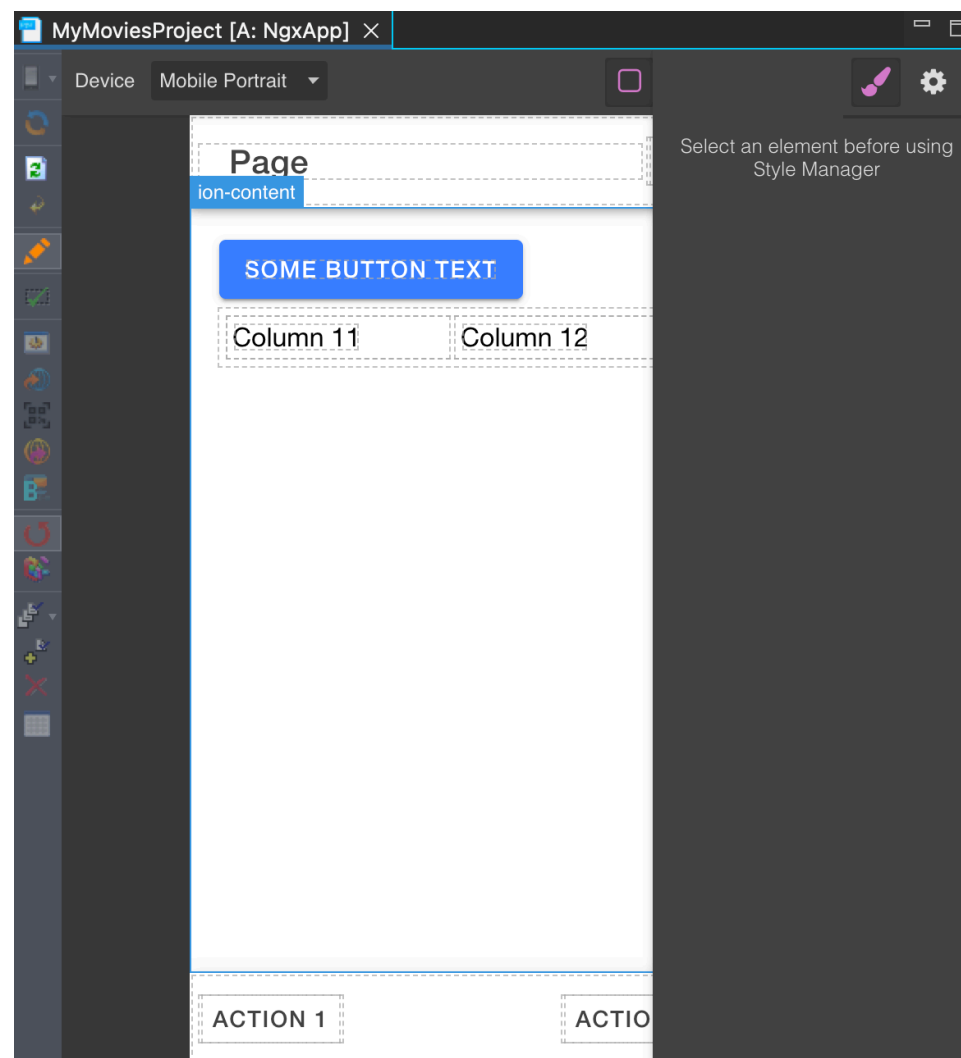


1.6 Application Viewer Controls



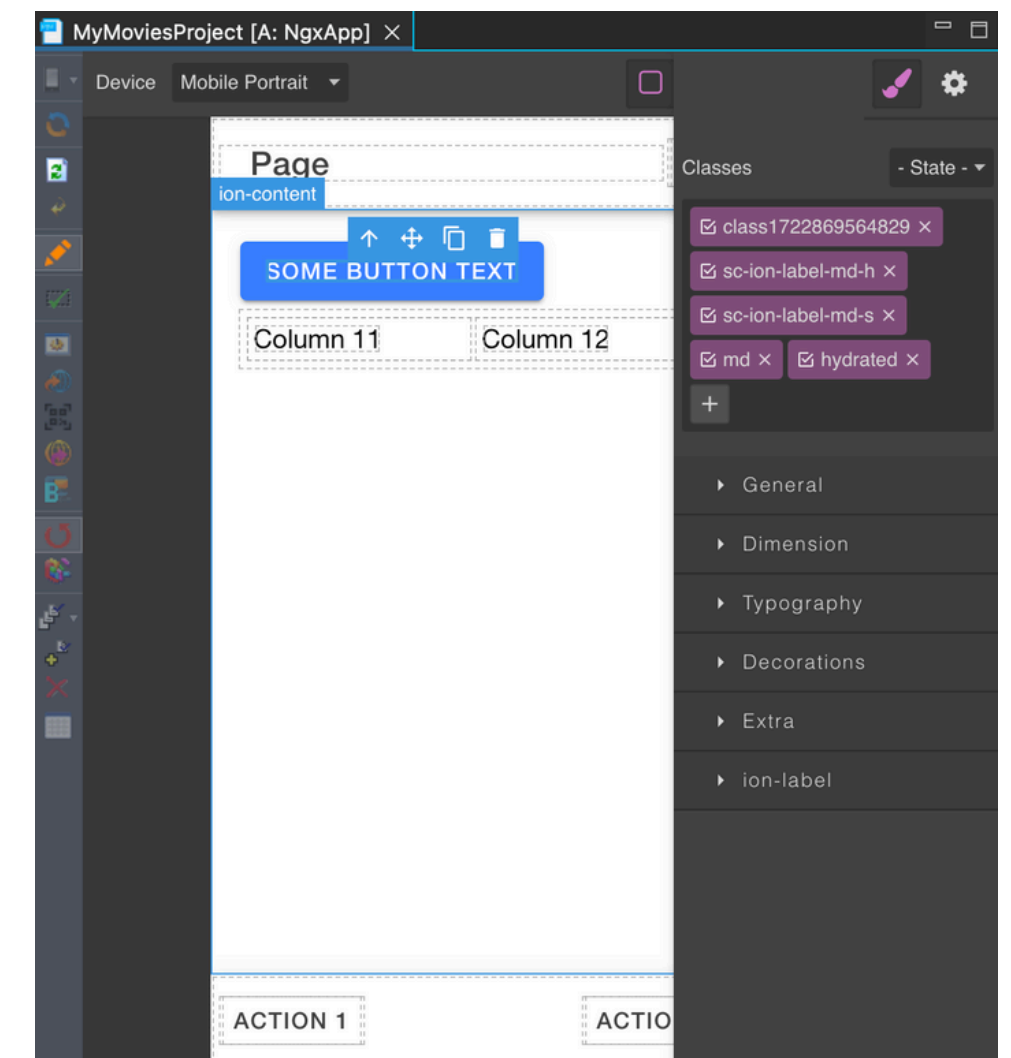
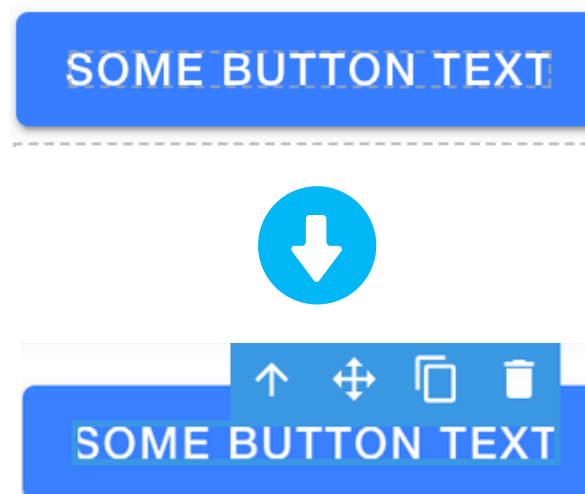
Style editor

When clicked, opens the Style editor.



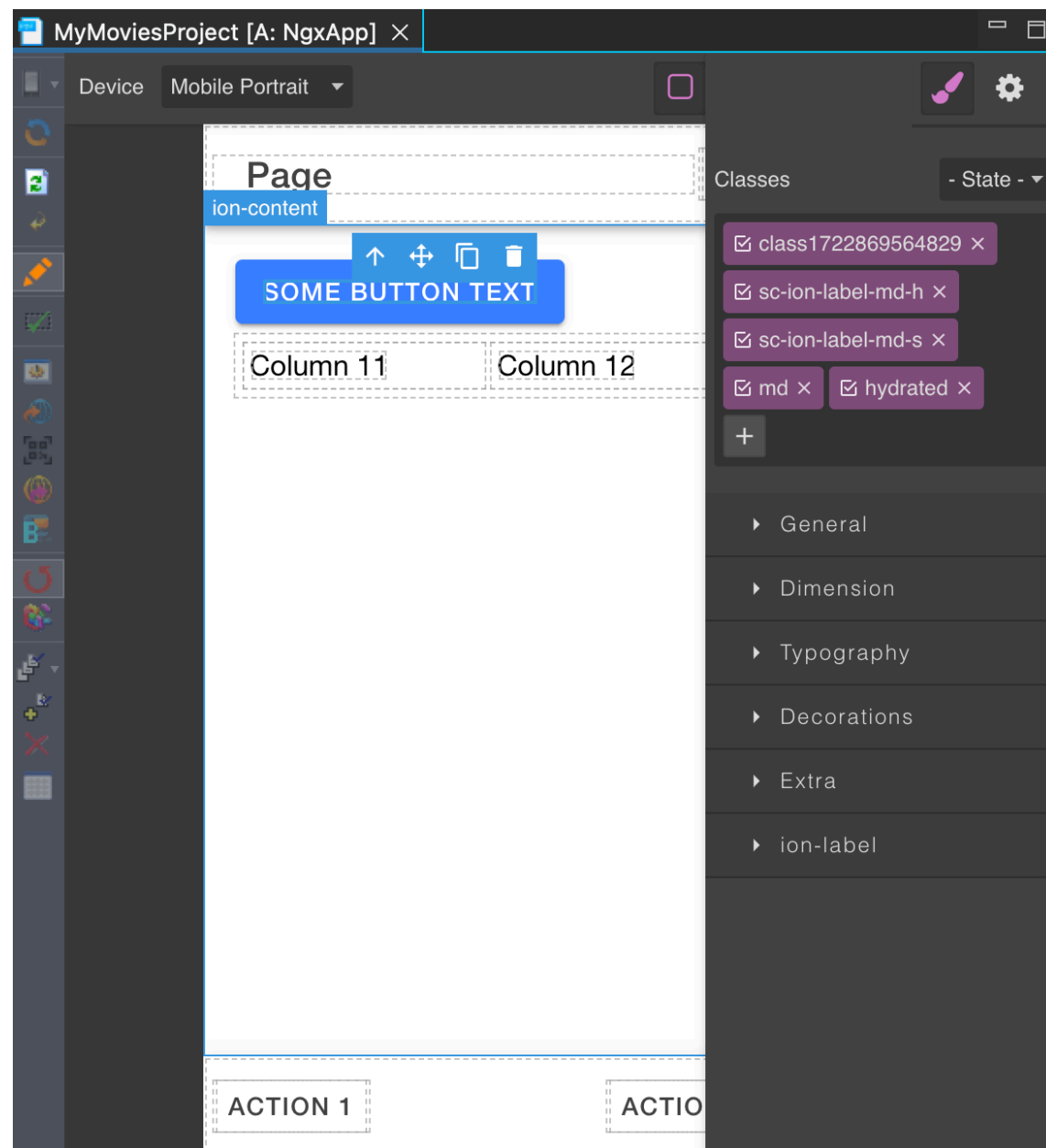
Select an element before using
Style Manager

In the **Application viewer**,
click on the **Button** to select it
and display the **Style Manager**.

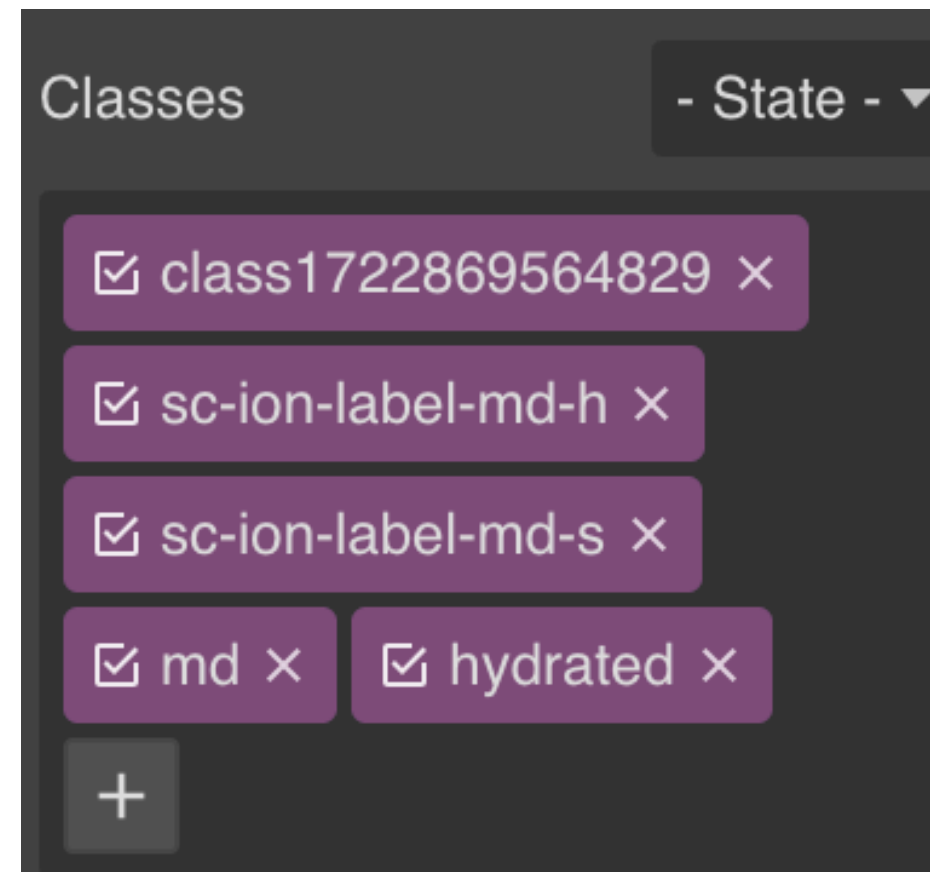


1.6 Application Viewer Controls

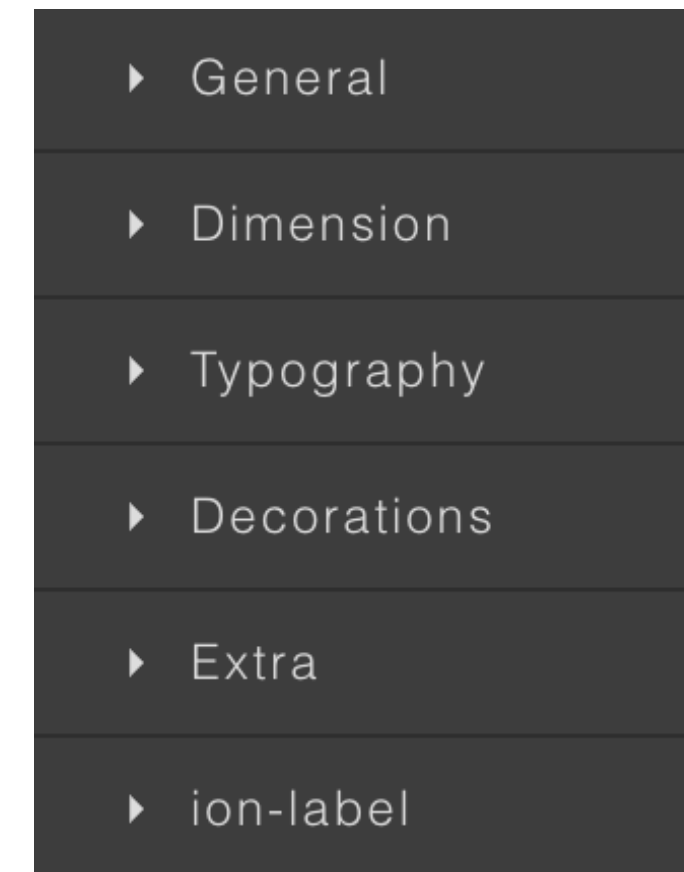
In the Style Manager, on the right of the Application Viewer, are displayed the different classes applied to the Button and a menu with controls to style it.



Classes
applied to the Button



Styling options
for the Button



1.6 Application Viewer Controls

In the Style Manager, different styling options are used to change the CSS rules applied to the Button component.

General

Display

block

Float

none

left

right

Position

static

relative

absolute

fixed

Top

auto

Right

auto

Left

auto

Bottom

auto

Dimension

Typography

Decorations

Extra

ion-label

Dimension

Width

auto

Height

auto

Max width

auto

Min height

auto

Margin

Top

0

Right

0

Bottom

0

Left

0

Padding

Top

0

Right

0

Bottom

0

Left

0

Typography

Font family

Arial

Font size

medium

Font weight

Normal

Letter spacing

normal

Color

black

Line height

normal

Text align

left

center

right

justify

Text shadow

Decorations

Background color

none

Border radius

Top Left

0

Top Right

0

Bottom Right

0

Bottom Left

0

Border

Width

0

Style

solid

Color

black

Box shadow

Background

Extra

Opacity

1

Transition

Transform

ion-label

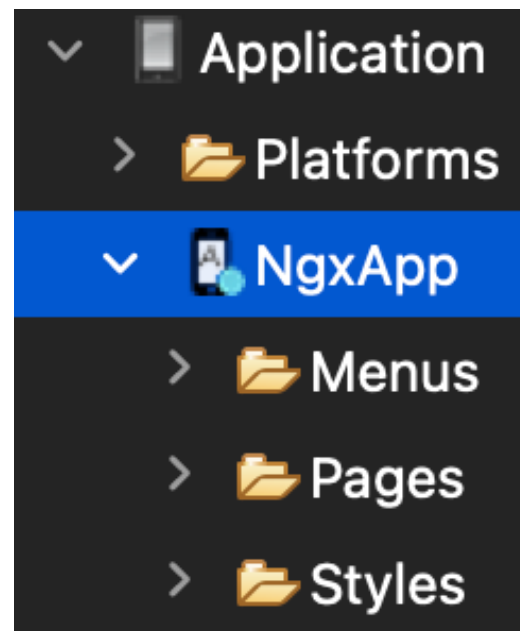
color



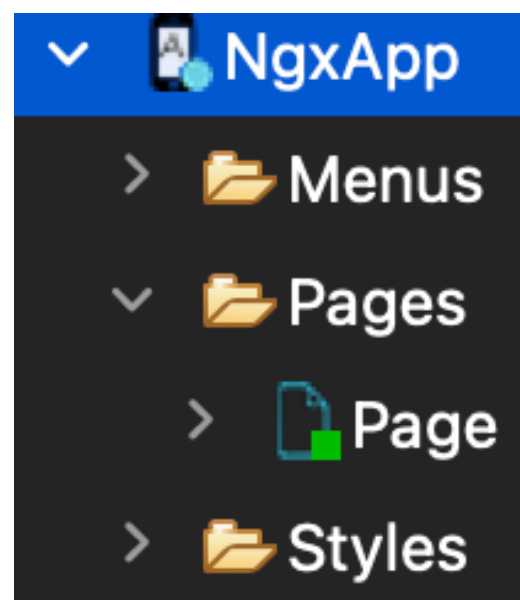
1.7 Pages & Menus

Let's have a look on the treeview of the Application.

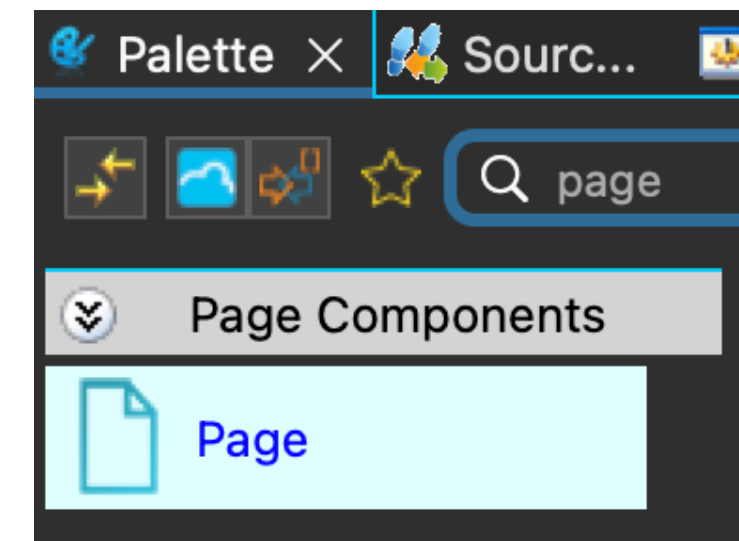
By default, 3 folders are created in the NgxApp object :



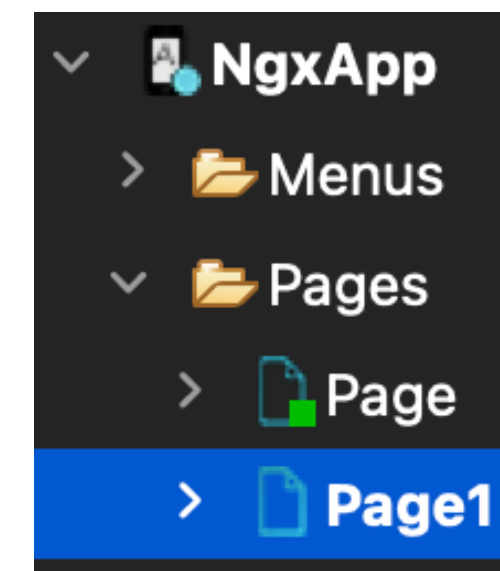
- Menus
- Pages
- Styles



Pages are the **screens** that form the application.
The **root page** is indicated by a green square.



To add a new page, drag the **Page component** from the **Palette**.

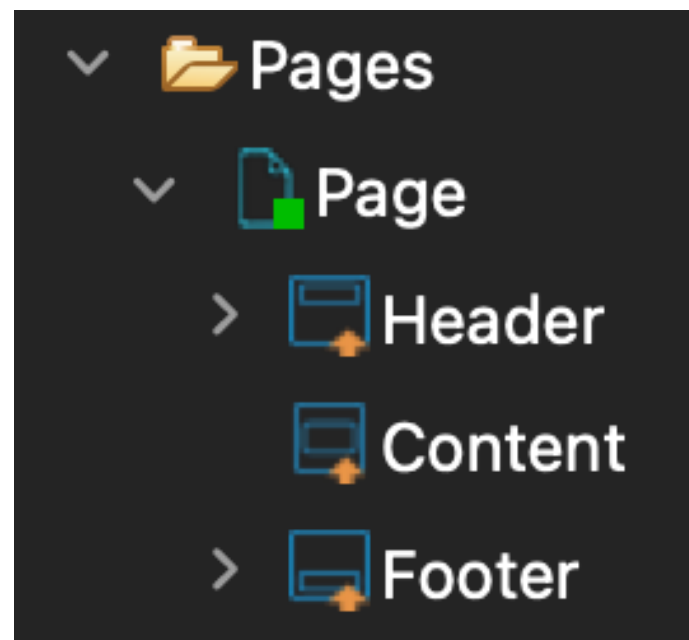


And drop it in the **NgxApp object** or in the **Pages folder**.



1.7 Pages & Menus

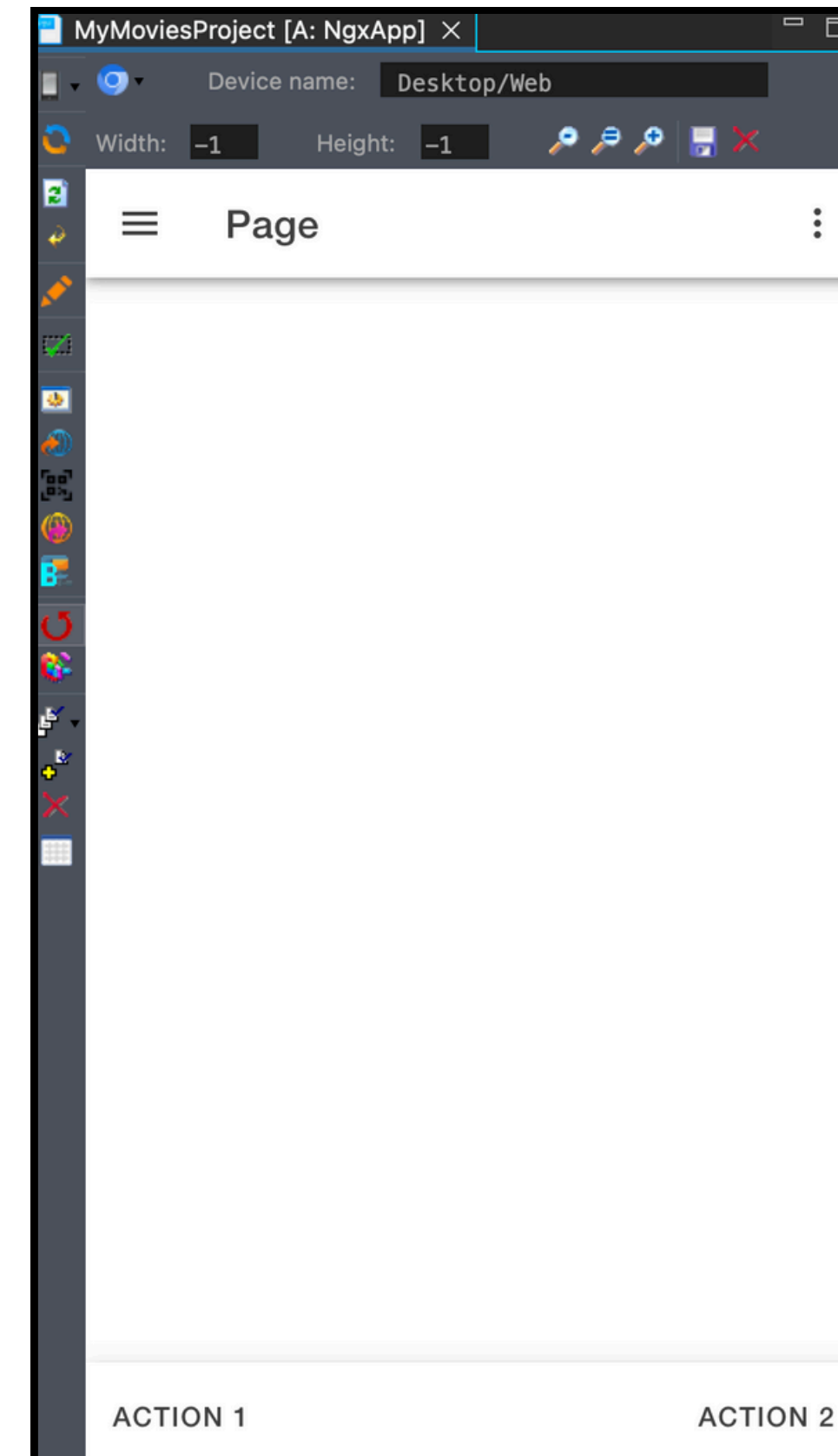
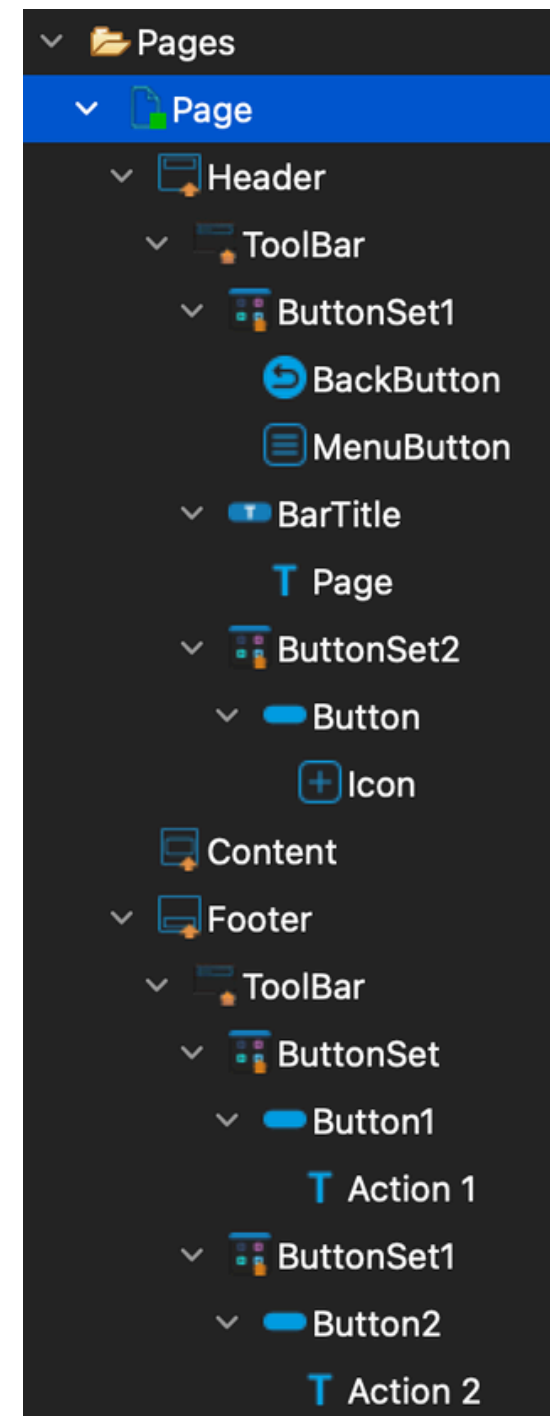
By default, 3 main components are created in the Page Component, as seen in the Application Viewer :



- Header
- Content
- Footer

The Header and Footer components include other components by default :

- Toolbar
- ButtonSet
- BarTitle
- etc.



Header

Content

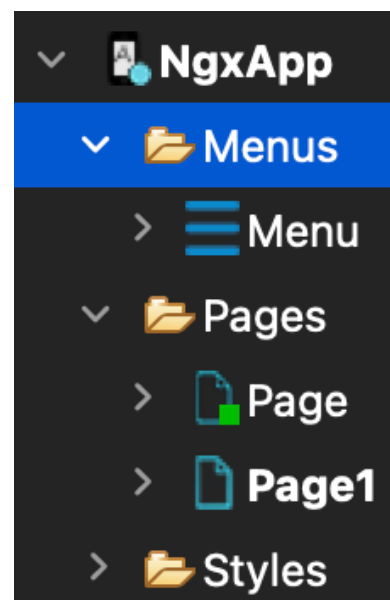
Footer



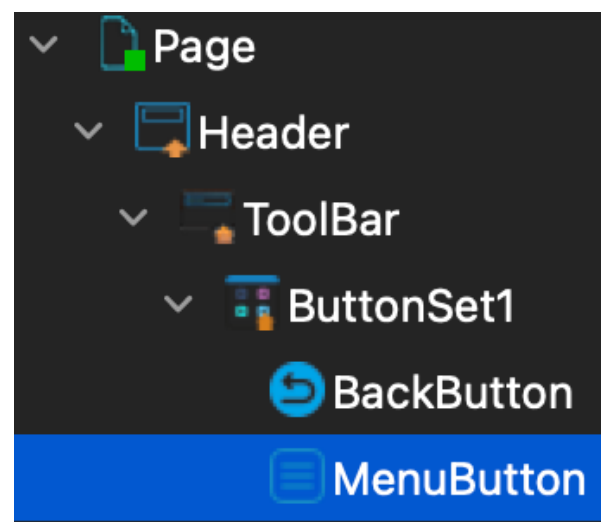
1.7 Pages & Menus

In the **Menus folder**, the **Menu component** is created by default.

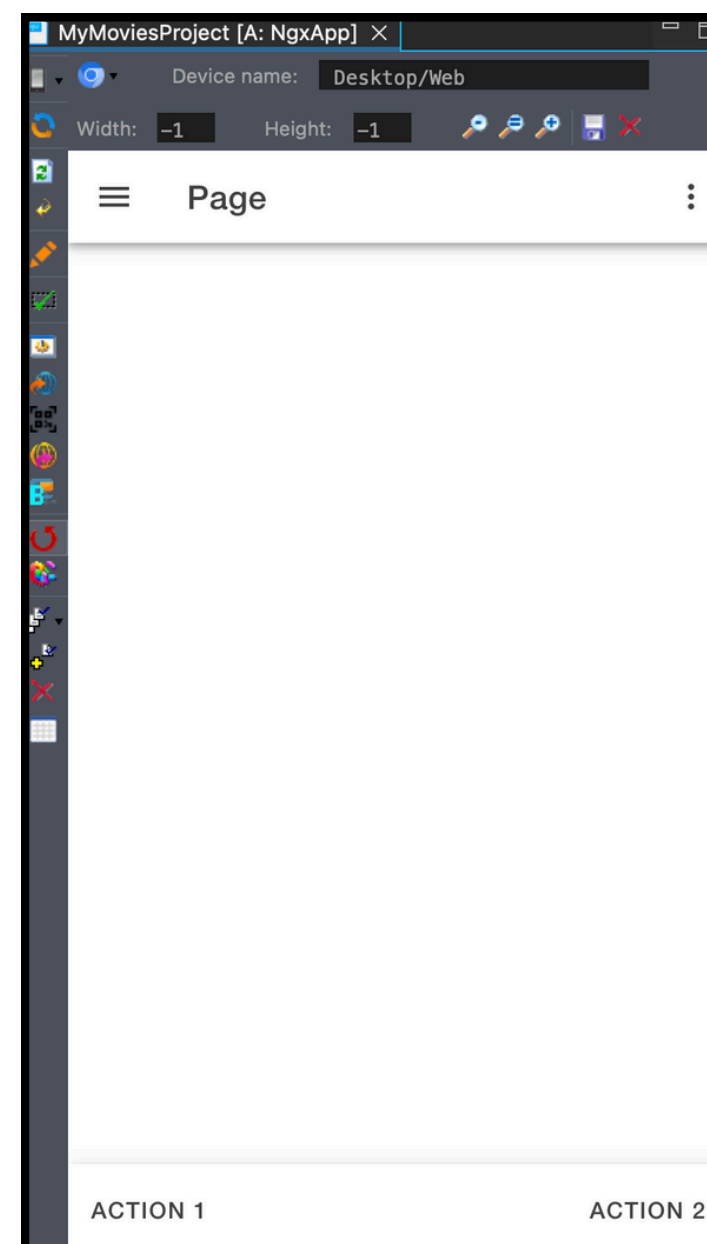
It is linked to the **Menu Button** component displayed in the **Page Header Toolbar** as seen in the top left corner of the Application Viewer



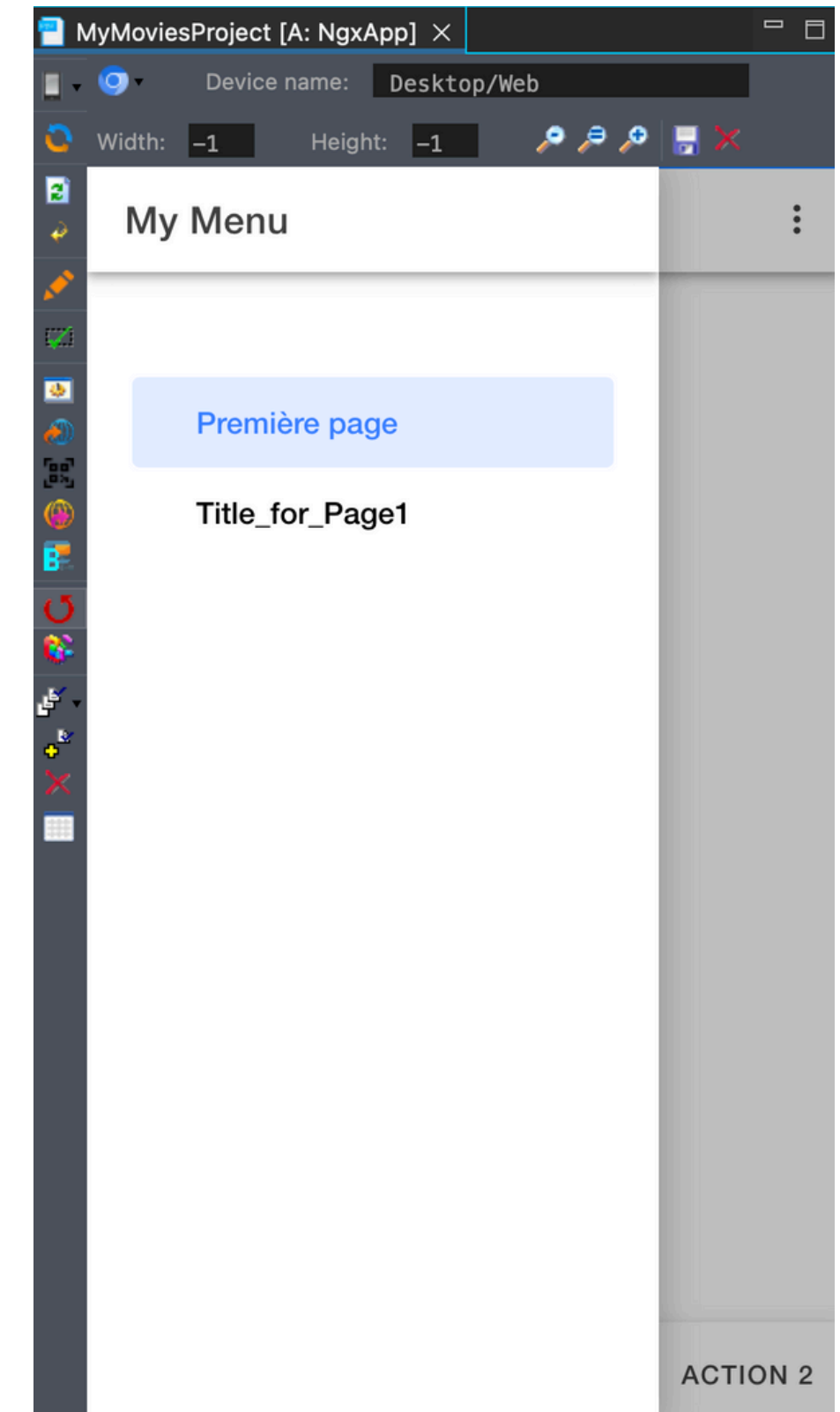
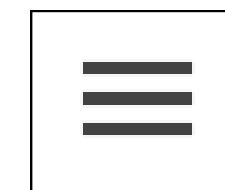
Menu Component
in **Menus folder**



Menu Button
in **Page Header Toolbar**

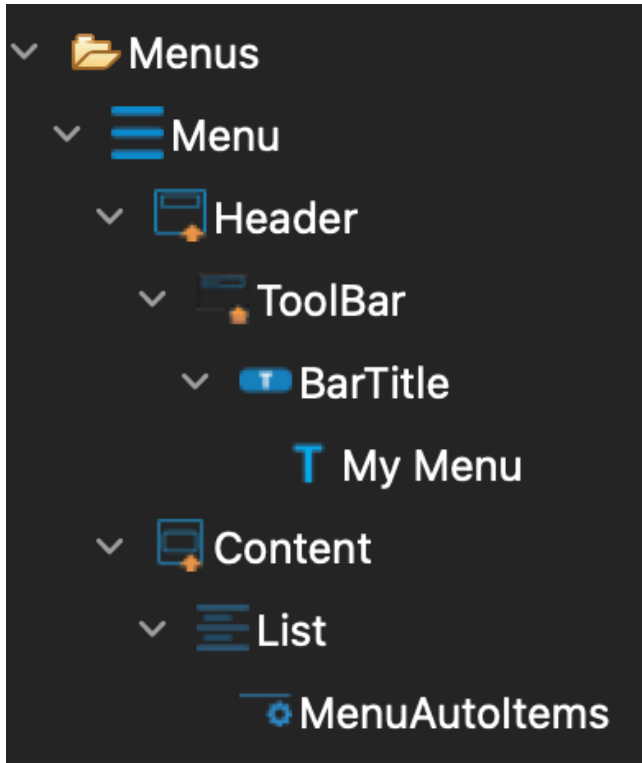
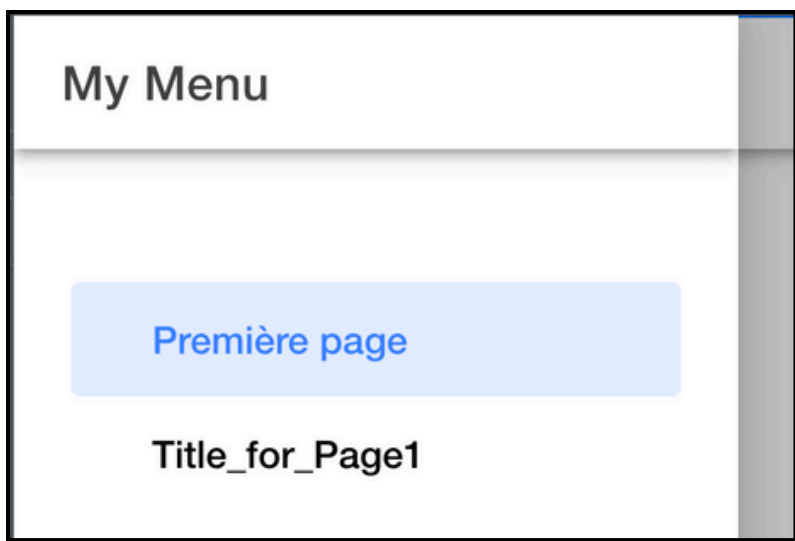


Click on the
MenuButton
to open the **Menu**

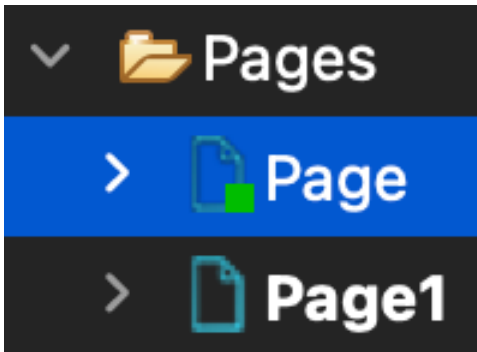


1.7 Pages & Menus

In the **Menu component**, the **Menu Auto Items** component generates a menu with links to the pages whose property **Include in auto menu** is set to true.

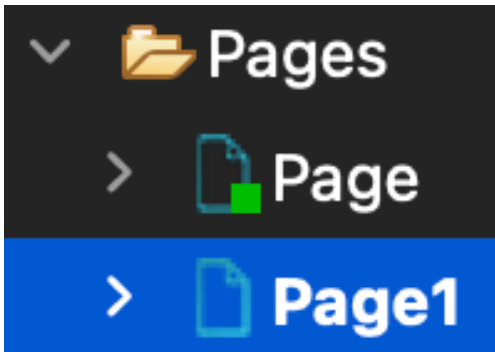


Menu Auto Items
in Menu Content List



(Root) Page
properties

Properties ×	
Property	Value
Base properties	
Comment	My First Page as root page
Include in auto menu	true
Is active	true
Menu icon	
Menu icon position	
Menu title	First page
Page end menu	
Page start menu	
Expert	
Changes detection strategy	Default
Preloading	high
Segment	path-to-xfirst

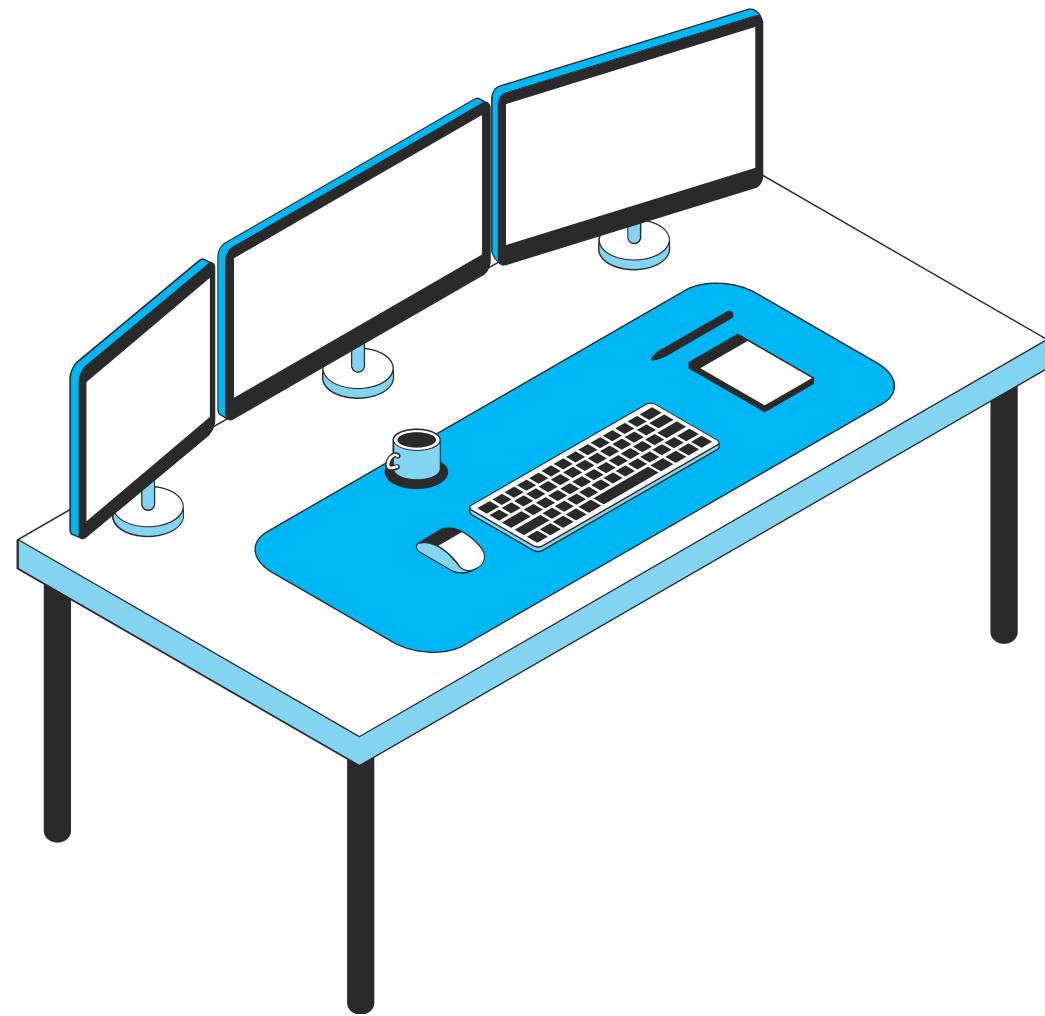


Page 1
properties

Properties ×	
Property	Value
Base properties	
Comment	
Include in auto menu	true
Is active	true
Menu icon	
Menu icon position	
Menu title	Title for Page1
Page end menu	
Page start menu	
Expert	
Changes detection strategy	Default
Preloading	low
Segment	path-to-page1



2 – Theming & Styling



2.1 Theming Basics

2.2 LightTheme & AutoDarkTheme

2.3 Theme Component

2.4 Style components & Global styles

2.5 Style components & Specific styles

2.6 CSS Custom Properties

2.7 Theme generator

2.8 Add a new font

2.1 Theming Basics

In Convertigo Low Code studio, the front end is based on **Angular** 13+ and uses **Ionic 6+** for the UI interface.

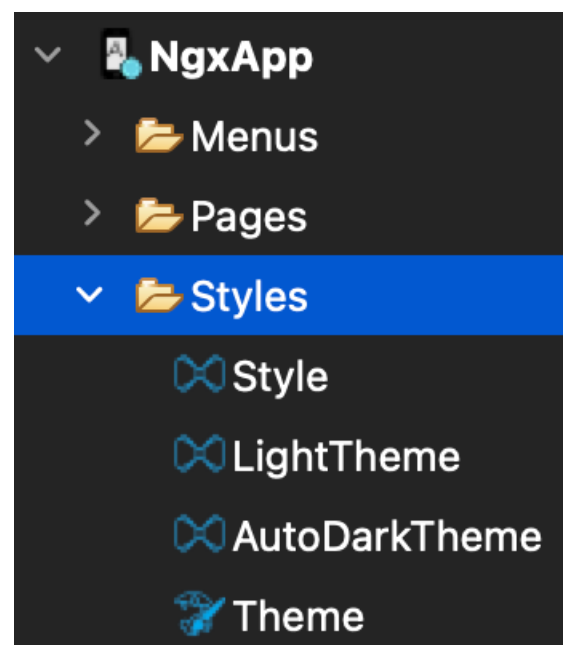
Ionic 6+ is built with **CSS variables**.

It comes with pre-baked **default styles** and **themes** easily **customized** by changing the value of these variables.

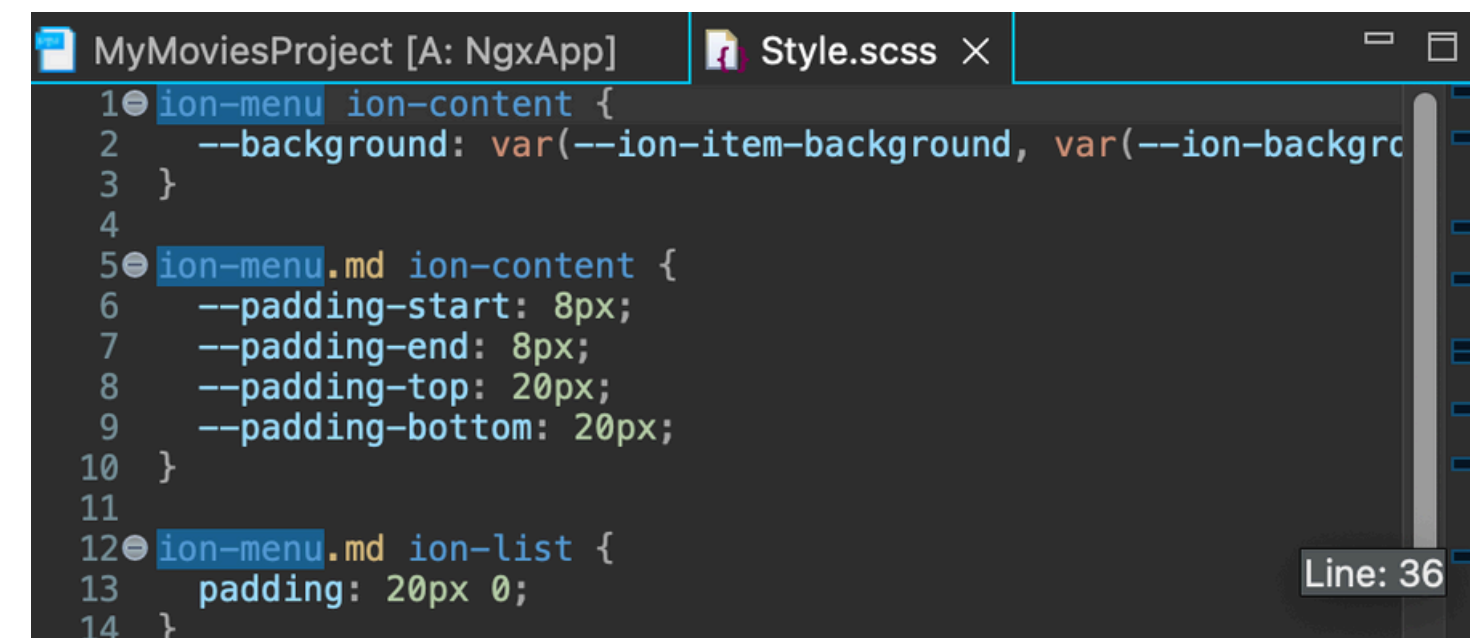
Theming basics are available in Ionic 6 documentation :

<https://ionic-docs-o31kiyk8l-ionic1.vercel.app/docs/v6/theming/basics>

In the Styles folder, a **Theme component** and 3 **Style components** appear by default.

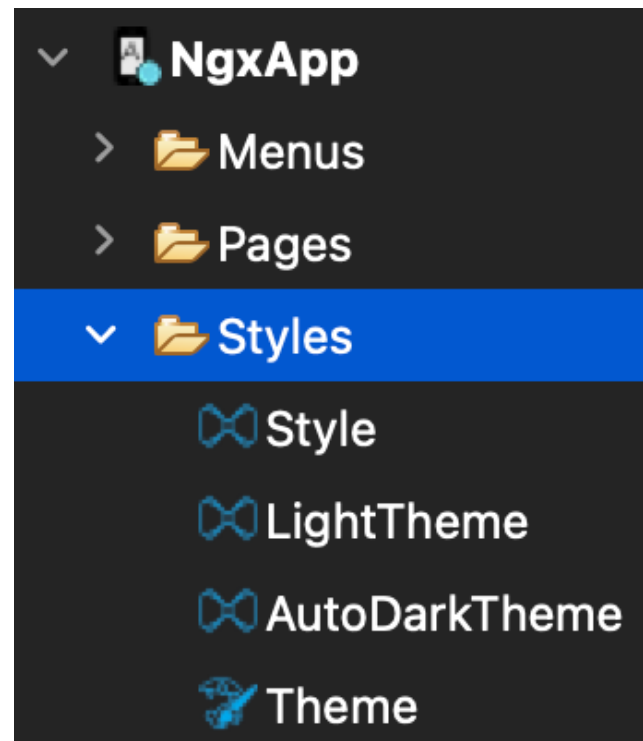


They represent editable scss files.
Double-click on these components
in the treeview
to open the corresponding scss files
in the text editor (in the right panel)
and modify CSS rules.



2.2 LightTheme & AutoDarkTheme

LightTheme and AutoDarkTheme define the light and dark themes by default of the application.



You can disable the **AutoDarkTheme**, if you don't need it.

```
MyMoviesProject [A: NgxApp] | LightTheme.scss
1 root {
2   /** primary **/
3   --ion-color-primary: #3880ff;
4   --ion-color-primary-rgb: 56, 128, 255;
5   --ion-color-primary-contrast: #ffffff;
6   --ion-color-primary-contrast-rgb: 255, 255, 255;
7   --ion-color-primary-shade: #3171e0;
8   --ion-color-primary-tint: #4c8dff;
9
10  /** secondary **/
11  --ion-color-secondary: #3dc2ff;
12  --ion-color-secondary-rgb: 61, 194, 255;
13  --ion-color-secondary-contrast: #ffffff;
14  --ion-color-secondary-contrast-rgb: 255, 255, 255;
15  --ion-color-secondary-shade: #36abe0;
16  --ion-color-secondary-tint: #50c8ff;
17
18  /** tertiary **/
19  --ion-color-tertiary: #5260ff;
20  --ion-color-tertiary-rgb: 82, 96, 255;
21  --ion-color-tertiary-contrast: #ffffff;
22  --ion-color-tertiary-contrast-rgb: 255, 255, 255;
23  --ion-color-tertiary-shade: #4854e0;
24  --ion-color-tertiary-tint: #6370ff;
25
26  /** success **/
27  --ion-color-success: #2dd36f;
28  --ion-color-success-rgb: 45, 211, 111;
29  --ion-color-success-contrast: #ffffff;
30  --ion-color-success-contrast-rgb: 255, 255, 255;
31  --ion-color-success-shade: #28ba62;
32  --ion-color-success-tint: #42d77d;
33
34  /** warning **/
35  --ion-color-warning: #ffc409;
36  --ion-color-warning-rgb: 255, 196, 9;
37  --ion-color-warning-contrast: #000000;
38  --ion-color-warning-contrast-rgb: 0, 0, 0;
39  --ion-color-warning-shade: #e0ac08;
```

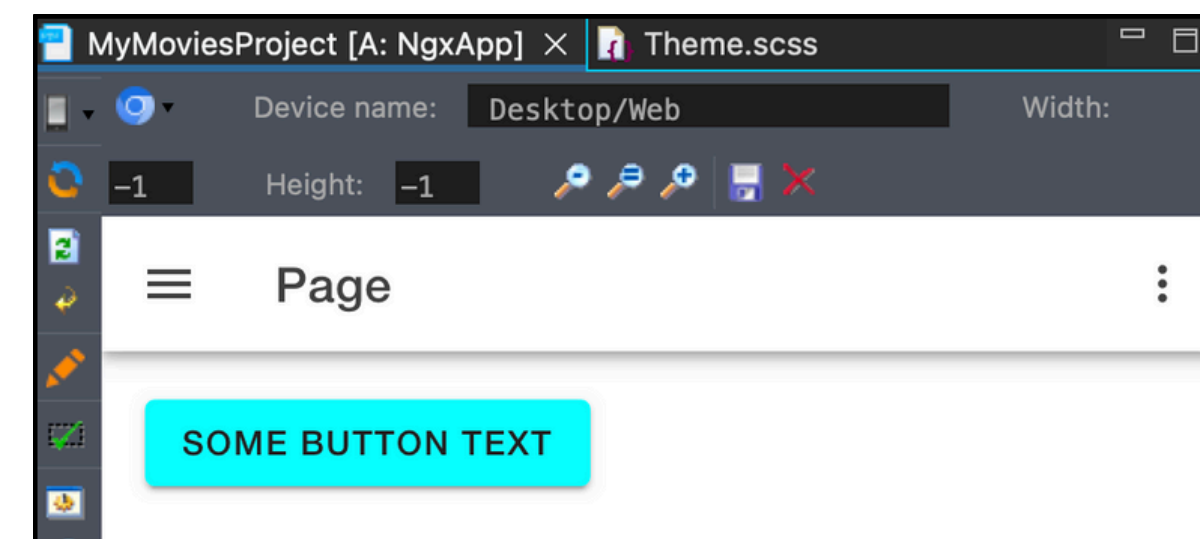
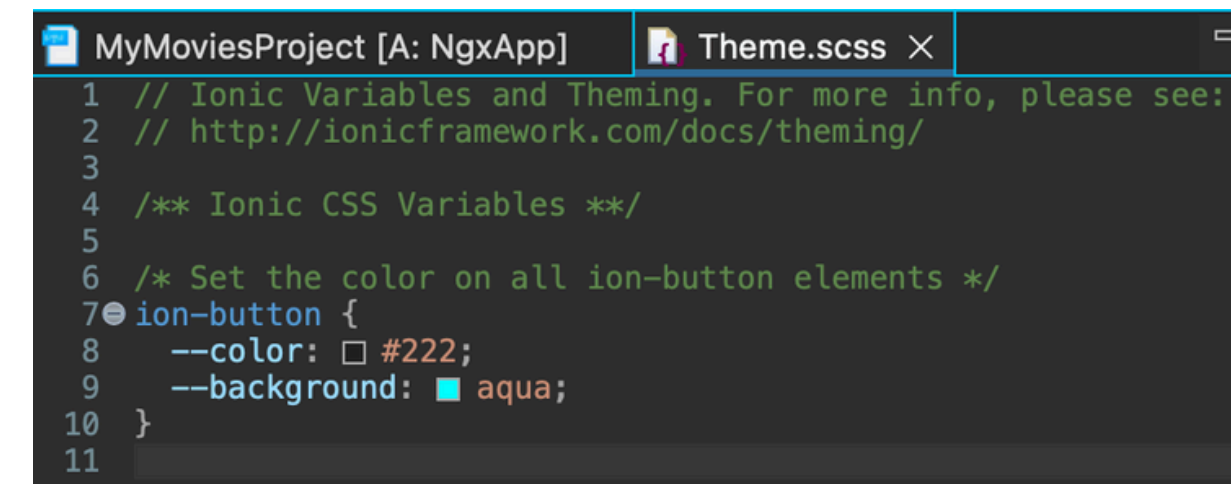
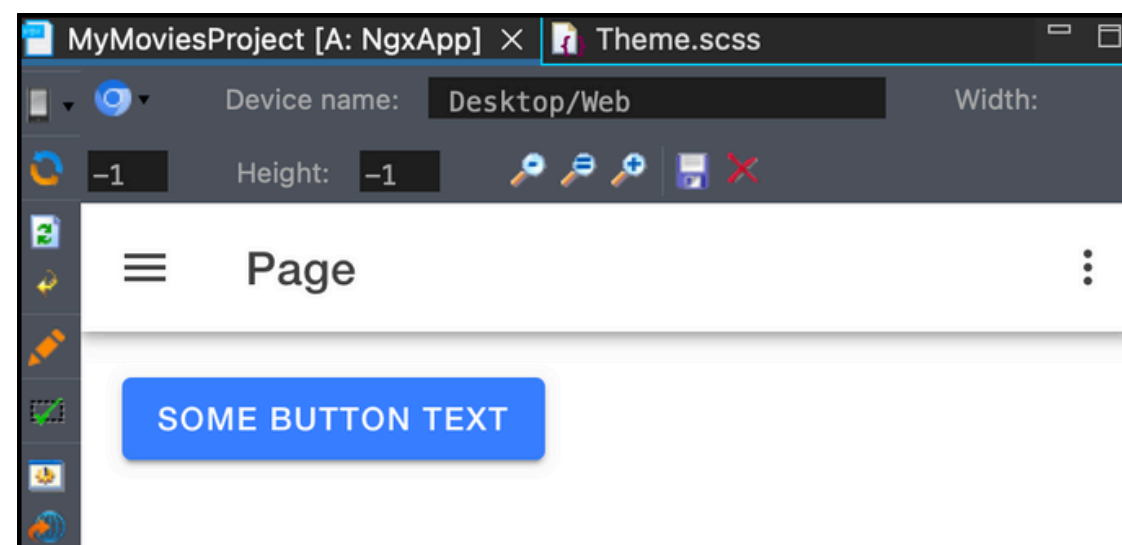
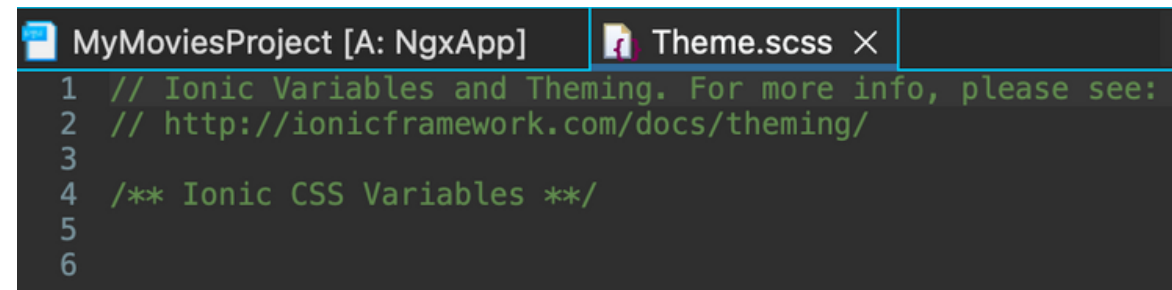
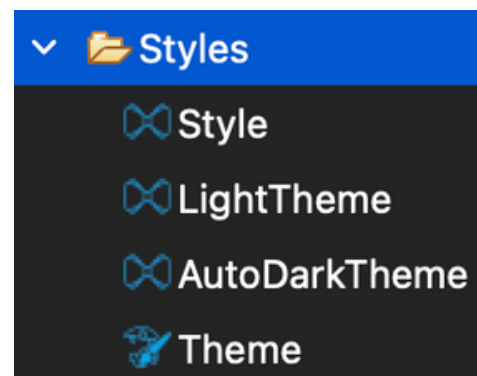
```
MyMoviesProject [A: NgxApp] | AutoDarkTheme.scss
1 @media (prefers-color-scheme: dark) {
2   /**
3    * Dark Colors
4    * -----
5    */
6
7   body {
8     --ion-color-primary: #428cff;
9     --ion-color-primary-rgb: 66, 140, 255;
10    --ion-color-primary-contrast: #ffffff;
11    --ion-color-primary-contrast-rgb: 255, 255, 255;
12    --ion-color-primary-shade: #3a7be0;
13    --ion-color-primary-tint: #5598ff;
14
15    --ion-color-secondary: #50c8ff;
16    --ion-color-secondary-rgb: 80, 200, 255;
17    --ion-color-secondary-contrast: #ffffff;
18    --ion-color-secondary-contrast-rgb: 255, 255, 255;
19    --ion-color-secondary-shade: #46b0e0;
20    --ion-color-secondary-tint: #62ceff;
21
22    --ion-color-tertiary: #6a64ff;
23    --ion-color-tertiary-rgb: 106, 100, 255;
24    --ion-color-tertiary-contrast: #ffffff;
25    --ion-color-tertiary-contrast-rgb: 255, 255, 255;
26    --ion-color-tertiary-shade: #5d58e0;
27    --ion-color-tertiary-tint: #7974ff;
28
29    --ion-color-success: #2fdf75;
30    --ion-color-success-rgb: 47, 223, 117;
31    --ion-color-success-contrast: #000000;
32    --ion-color-success-contrast-rgb: 0, 0, 0;
33    --ion-color-success-shade: #29c467;
34    --ion-color-success-tint: #44e283;
35
36    --ion-color-warning: #ffd534;
37    --ion-color-warning-rgb: 255, 213, 52;
38    --ion-color-warning-contrast: #000000;
39    --ion-color-warning-contrast-rgb: 0, 0, 0;
```



2.3 Theme Component

In the Styles folder, the **Theme component** is used for additional theme customization.

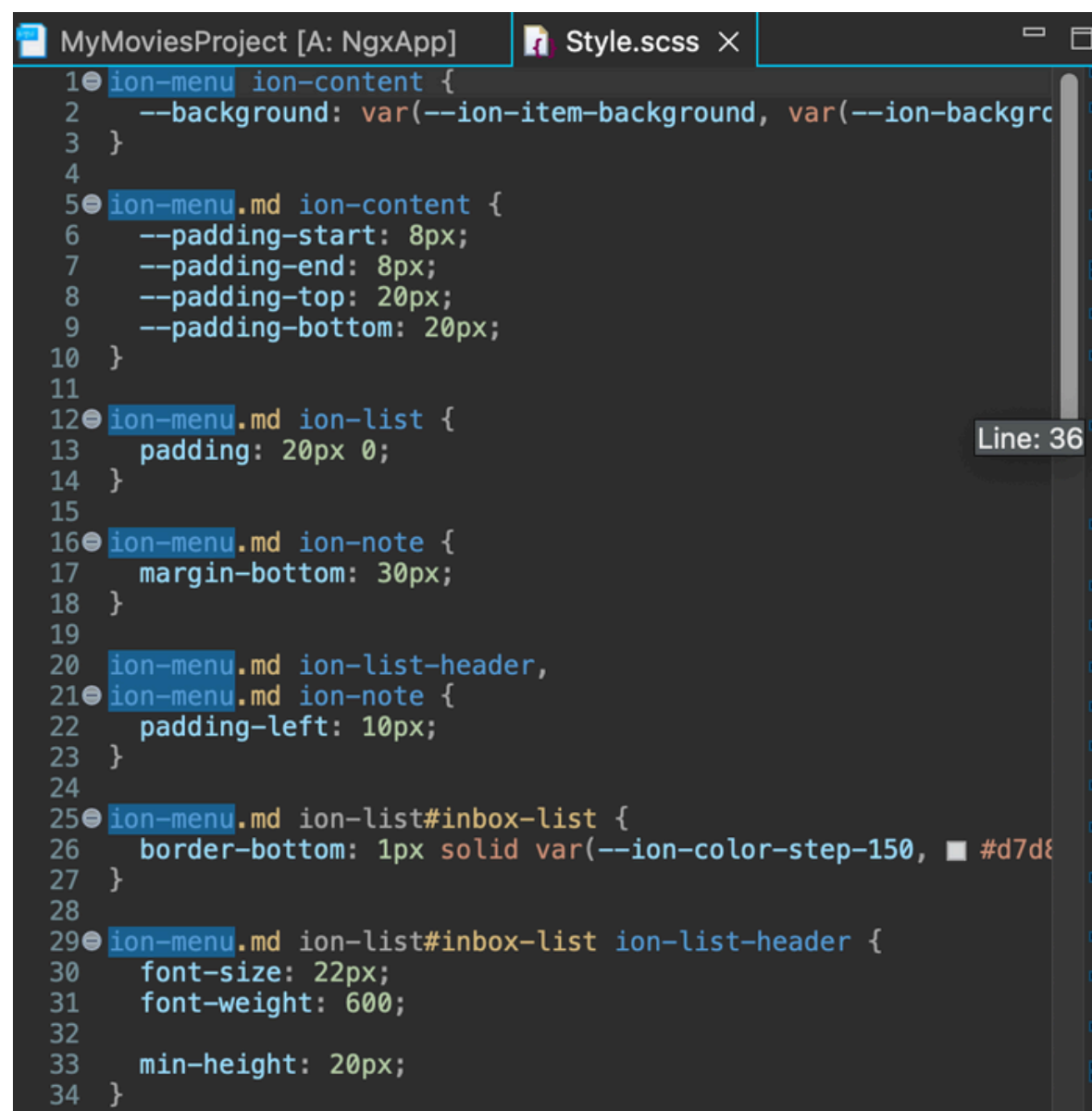
Double-click on the Theme component to open **Theme.scss** in the text editor and modify Ionic defaults (color variables, fonts, icons...)



2.4 Style components & Global styles

In the **Styles folder**, double-click on the **Style component** to open **Style.scss** in the text editor and **edit or delete the styles by default**.

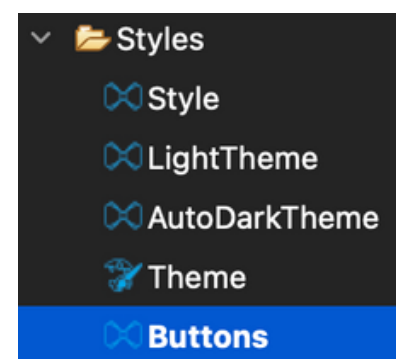
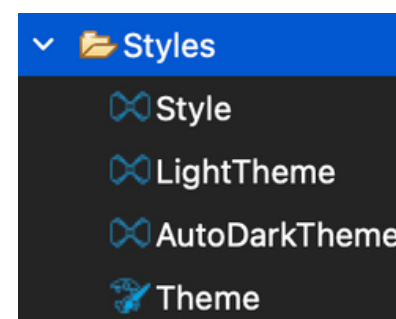
Styles present in the **Styles folder** are **global styles** applied to the **entire application**.



```

1 ion-menu ion-content {
2   --background: var(--ion-item-background, var(--ion-background));
3 }
4
5 ion-menu.md ion-content {
6   --padding-start: 8px;
7   --padding-end: 8px;
8   --padding-top: 20px;
9   --padding-bottom: 20px;
10 }
11
12 ion-menu.md ion-list {
13   padding: 20px 0;
14 }
15
16 ion-menu.md ion-note {
17   margin-bottom: 30px;
18 }
19
20 ion-menu.md ion-list-header,
21 ion-menu.md ion-note {
22   padding-left: 10px;
23 }
24
25 ion-menu.md ion-list#inbox-list {
26   border-bottom: 1px solid var(--ion-color-step-150, #d7d8db);
27 }
28
29 ion-menu.md ion-list#inbox-list ion-list-header {
30   font-size: 22px;
31   font-weight: 600;
32   min-height: 20px;
33 }
34

```



You can **add as many Styles components** as necessary, by dragging them from the **NGX palette** and dropping them in the **Styles folder**.

As good practice, **add a new stylesheet for each global element** you want to style. For example, **create a new stylesheet** named **Buttons.scss** for styling buttons with the same style anywhere in your application.

All stylesheets you create will be **compiled in one file**.

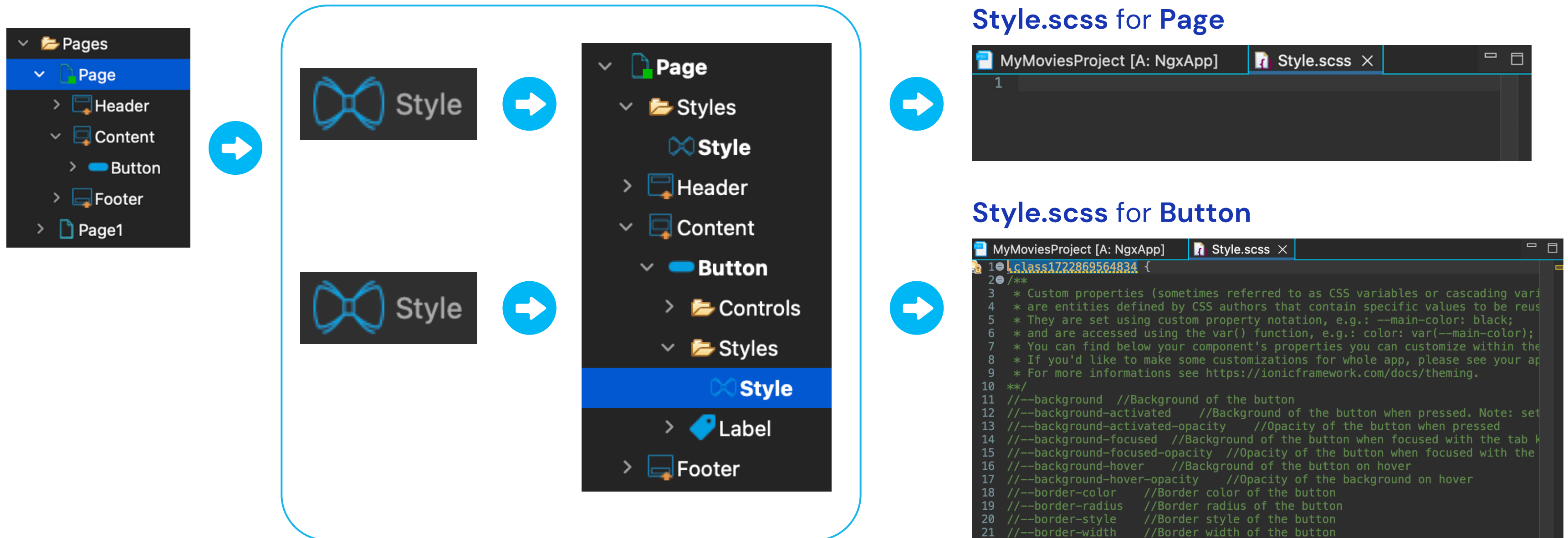


2.5 Style components & Specific styles

You can also add a **Style component** to a **Page** or any **Component**.

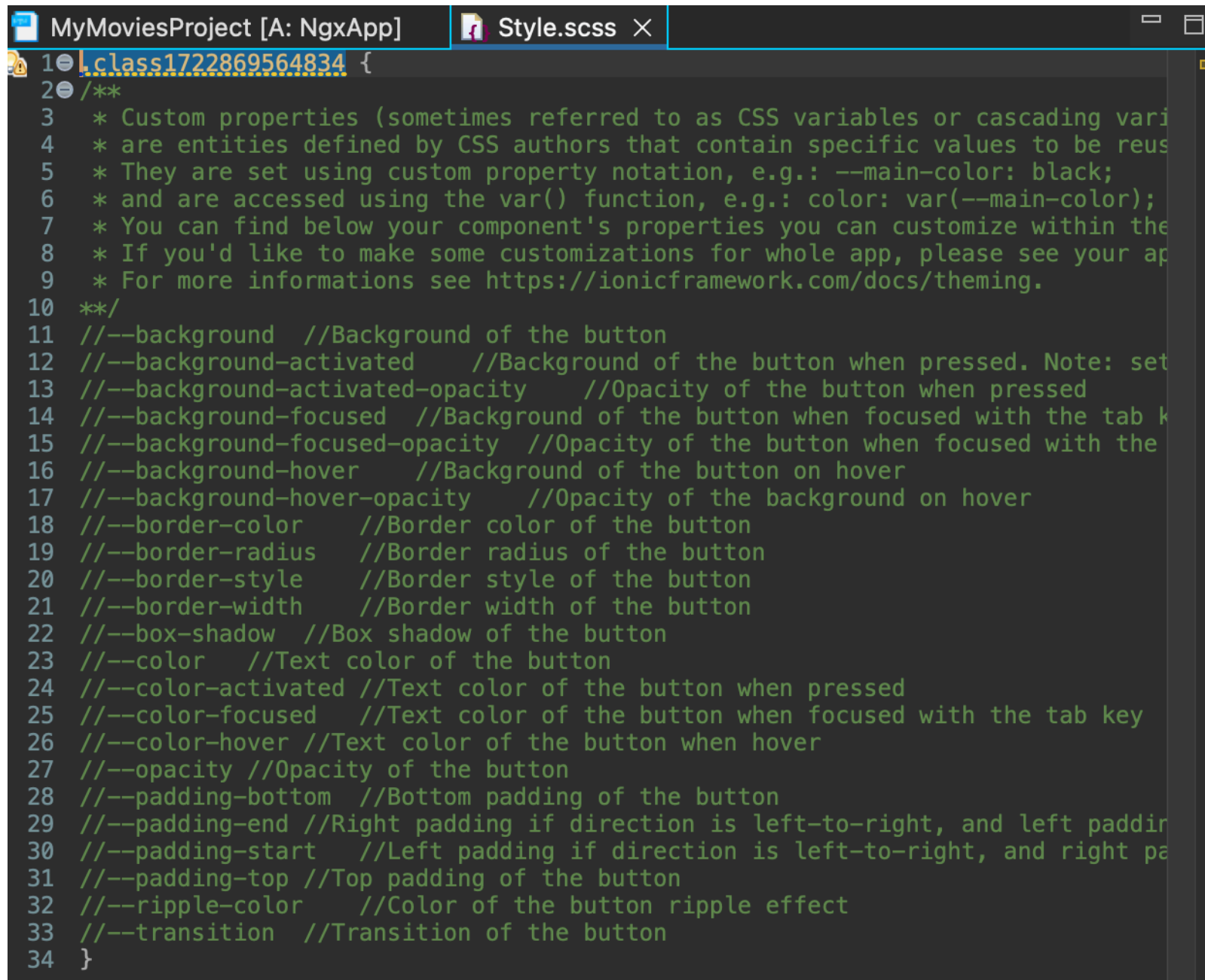
Drag it from the **NGX palette** & drop it under a page or component in the treeview.

Then double-click on the Style component to edit the CSS rules.



2.6 CSS Custom Properties

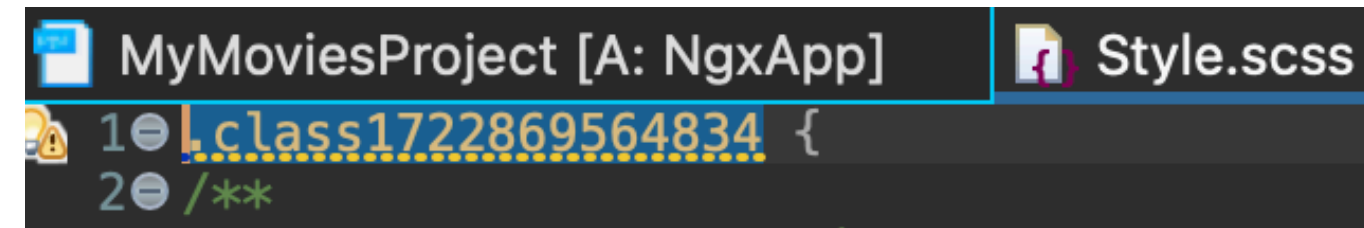
When a **Style component** is added to a **component**, the corresponding **Style.scss** file comes as a class with commented instructions to edit CSS rules.



```

1  @class1722869564834 {
2  /**
3   * Custom properties (sometimes referred to as CSS variables or cascading variables)
4   * are entities defined by CSS authors that contain specific values to be reused.
5   * They are set using custom property notation, e.g.: --main-color: black;
6   * and are accessed using the var() function, e.g.: color: var(--main-color);
7   * You can find below your component's properties you can customize within the app.
8   * If you'd like to make some customizations for whole app, please see your app's global
9   * For more informations see https://ionicframework.com/docs/theming.
10  */
11  //--background //Background of the button
12  //--background-activated //Background of the button when pressed. Note: set
13  //--background-activated-opacity //Opacity of the button when pressed
14  //--background-focused //Background of the button when focused with the tab key
15  //--background-focused-opacity //Opacity of the button when focused with the tab key
16  //--background-hover //Background of the button on hover
17  //--background-hover-opacity //Opacity of the background on hover
18  //--border-color //Border color of the button
19  //--border-radius //Border radius of the button
20  //--border-style //Border style of the button
21  //--border-width //Border width of the button
22  //--box-shadow //Box shadow of the button
23  //--color //Text color of the button
24  //--color-activated //Text color of the button when pressed
25  //--color-focused //Text color of the button when focused with the tab key
26  //--color-hover //Text color of the button when hover
27  //--opacity //Opacity of the button
28  //--padding-bottom //Bottom padding of the button
29  //--padding-end //Right padding if direction is left-to-right, and left padding if right-to-left
30  //--padding-start //Left padding if direction is left-to-right, and right padding if right-to-left
31  //--padding-top //Top padding of the button
32  //--ripple-color //Color of the button ripple effect
33  //--transition //Transition of the button
34  }

```

```

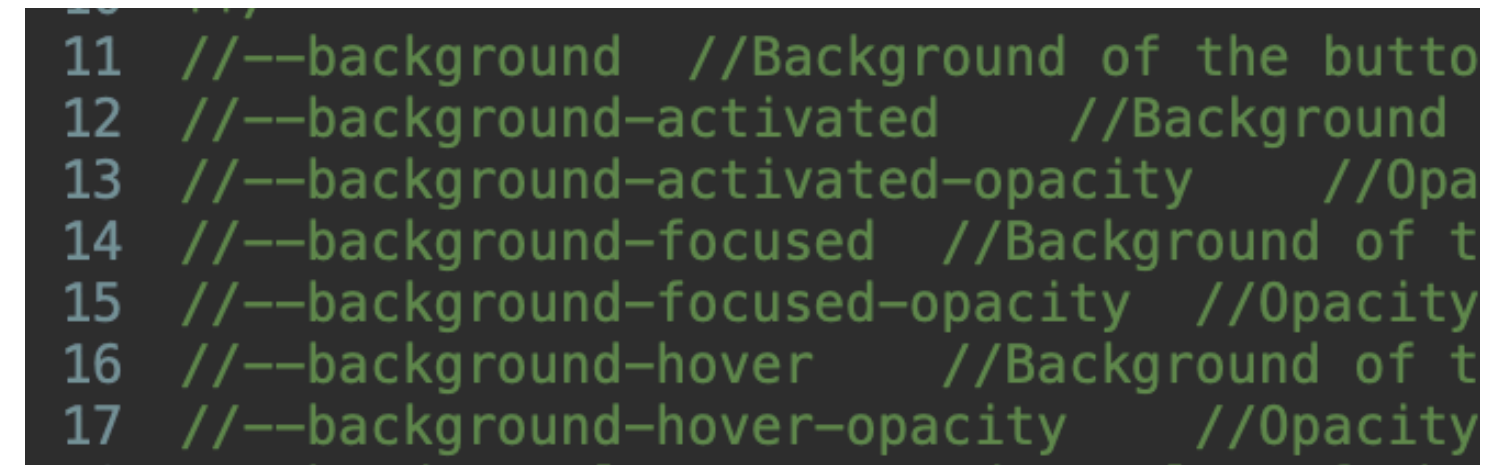
1  @class1722869564834 {
2  /**

```

Each Ionic component has a set of **CSS Custom Properties**, which are listed here in **Style.scss**.

They always start by **--**.

For example, here are the **CSS Custom Properties** used to style the **background** of a **Button**.

```

11  //--background //Background of the button
12  //--background-activated //Background of the button when pressed
13  //--background-activated-opacity //Opacity of the button when pressed
14  //--background-focused //Background of the button when focused with the tab key
15  //--background-focused-opacity //Opacity of the button when focused with the tab key
16  //--background-hover //Background of the button on hover
17  //--background-hover-opacity //Opacity of the background on hover

```



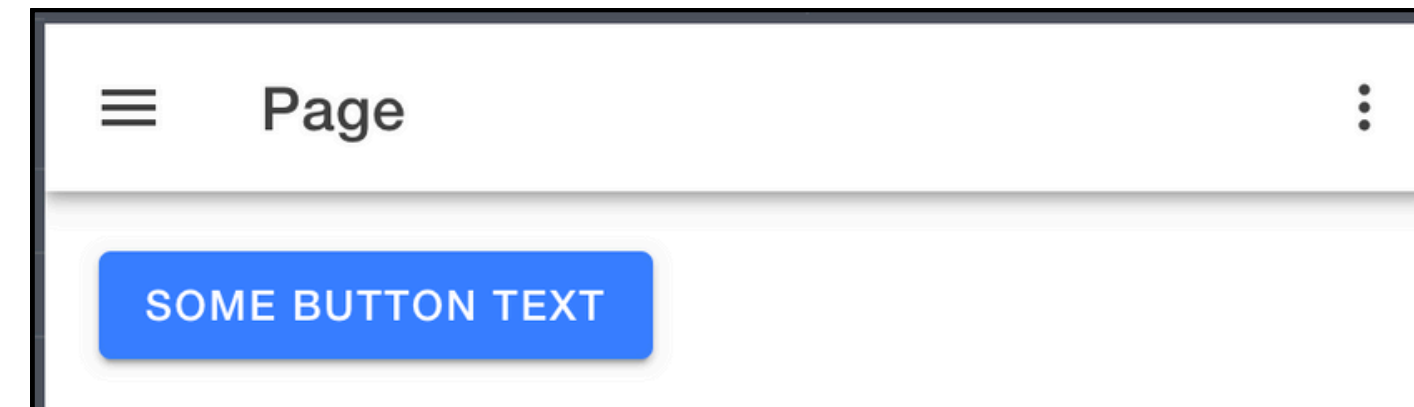
2.6 CSS Custom Properties

With these **CSS Custom Properties**, let's change the style of the Button.

```
MyMoviesProject [A: NgxApp] | Style.scss x
1 .class1722869564834 {
2 /**
3  * Custom properties (sometimes referred to as CSS variables or cascading vari
4  * are entities defined by CSS authors that contain specific values to be reus
5  * They are set using custom property notation, e.g.: --main-color: black;
6  * and are accessed using the var() function, e.g.: color: var(--main-color);
7  * You can find below your component's properties you can customize within the
8  * If you'd like to make some customizations for whole app, please see your ap
9  * For more informations see https://ionicframework.com/docs/theming.
10 */
11 //--background //Background of the button
12 //--background-activated //Background of the button when pressed. Note: set
13 //--background-activated-opacity //Opacity of the button when pressed
14 //--background-focused //Background of the button when focused with the tab k
15 //--background-focused-opacity //Opacity of the button when focused with the
16 //--background-hover //Background of the button on hover
17 //--background-hover-opacity //Opacity of the background on hover
18 //--border-color //Border color of the button
19 //--border-opacity //Border opacity of the button
```



Originally, the **Button** appears as such in the page.

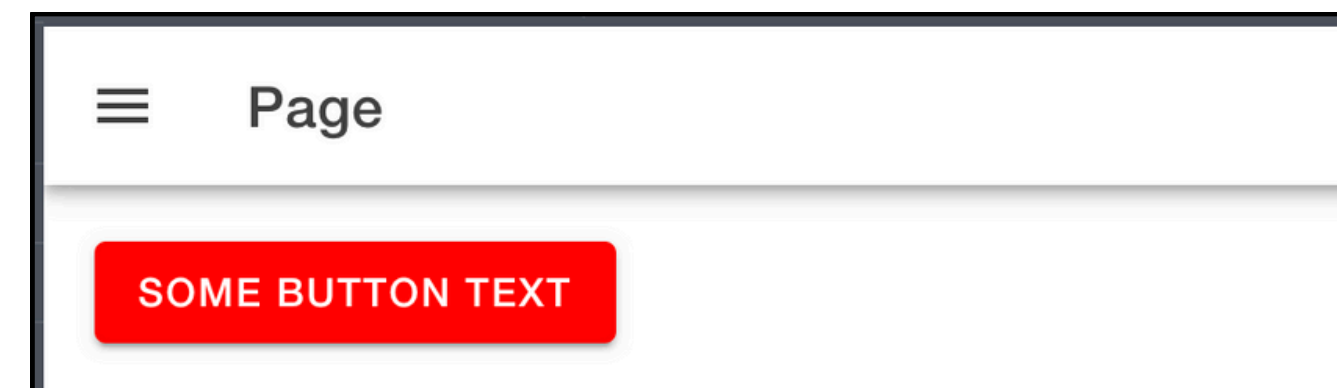


Let's edit the class
and add the **--background property**
with red as background color.

```
MyMoviesProject [A: NgxApp] | Style.scss x
1 .class1722869564834 {
2
3   --background: red;
4 }
```



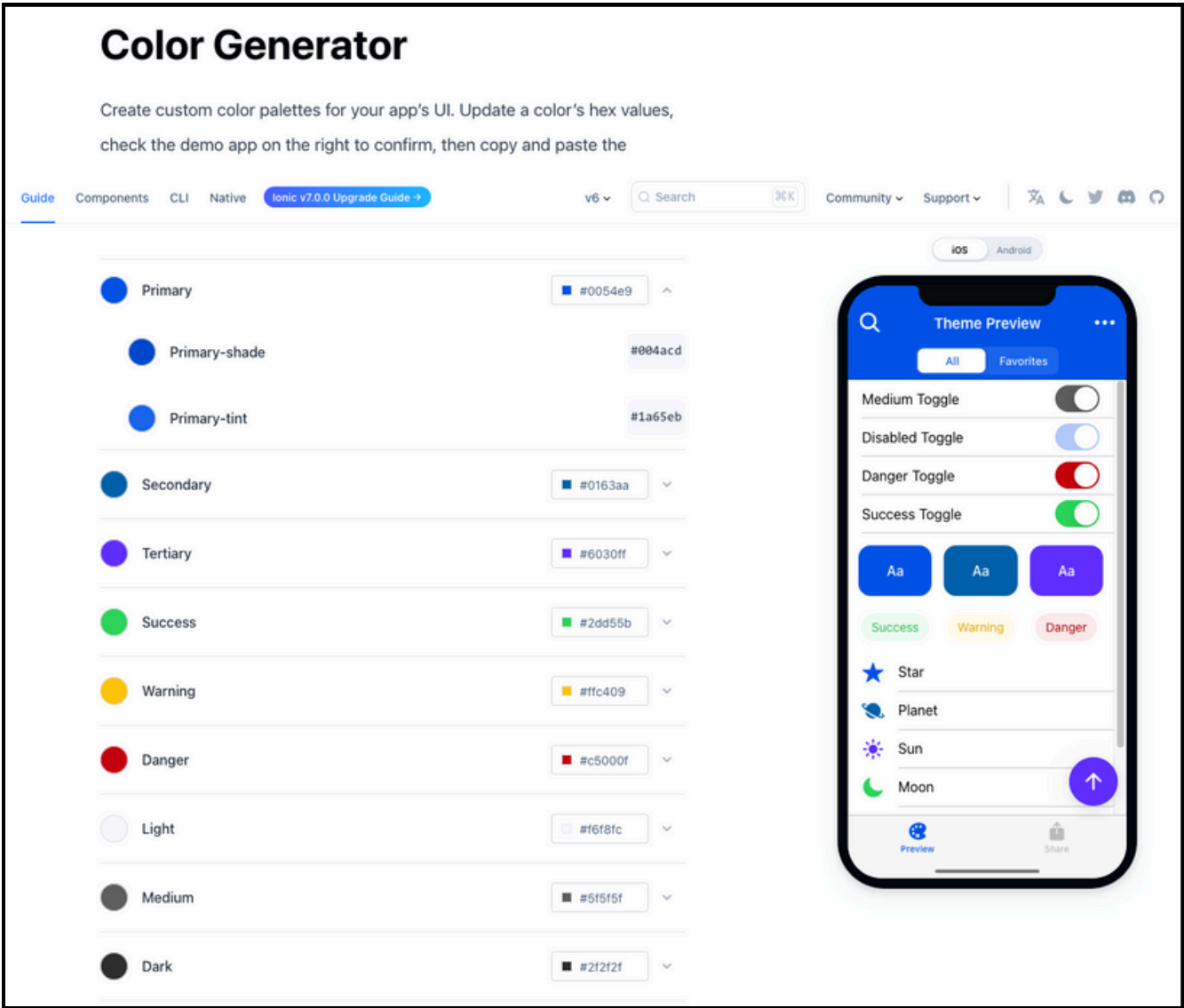
The background of the **Button** is now red.



2.7 Theme generator

To create a **new theme** for your application, you can use the **Color Generator** in Ionic Docs.
<https://ionicframework.com/docs/theming/color-generator>

You can easily generate a **custom color palette** by replacing the original colors by any color you want.



```
:root {
  --ion-color-primary: #0054e9;
  --ion-color-primary-rgb: 0,84,233;
  --ion-color-primary-contrast: #ffffff;
  --ion-color-primary-contrast-rgb: 255,255,255;
  --ion-color-primary-shade: #004acd;
  --ion-color-primary-tint: #1a65eb;

  --ion-color-secondary: #0163aa;
  --ion-color-secondary-rgb: 1,99,170;
  --ion-color-secondary-contrast: #ffffff;
  --ion-color-secondary-contrast-rgb: 255,255,255;
  --ion-color-secondary-shade: #015796;
  --ion-color-secondary-tint: #1a73b3;

  --ion-color-tertiary: #6030ff;
  --ion-color-tertiary-rgb: 96,48,255;
  --ion-color-tertiary-contrast: #ffffff;
  --ion-color-tertiary-contrast-rgb: 255,255,255;
  --ion-color-tertiary-shade: #542ae0;
  --ion-color-tertiary-tint: #7045ff;

  --ion-color-success: #2dd55b;
  --ion-color-success-rgb: 45,213,91;
  --ion-color-success-contrast: #000000;
  --ion-color-success-contrast-rgb: 0,0,0;
  --ion-color-success-shade: #28bb50;
  --ion-color-success-tint: #42d96b;

  --ion-color-warning: #ffc409;
  --ion-color-warning-rgb: 255,196,9;
  --ion-color-warning-contrast: #000000;
  --ion-color-warning-contrast-rgb: 0,0,0;
  --ion-color-warning-shade: #e0ac08;
  --ion-color-warning-tint: #ffca22;

  --ion-color-danger: #c5000f;
  --ion-color-danger-rgb: 197,0,15;
  --ion-color-danger-contrast: #ffffff;
  --ion-color-danger-contrast-rgb: 255,255,255;
  --ion-color-danger-shade: #ad000d;
  --ion-color-danger-tint: #cb1a27;

  --ion-color-light: #f6f8fc;
  --ion-color-light-rgb: 246,248,252;
  --ion-color-light-contrast: #000000;
  --ion-color-light-contrast-rgb: 0,0,0;
  --ion-color-light-shade: #d8dade;
  --ion-color-light-tint: #f7f9fc;

  --ion-color-medium: #5f5f5f;
  --ion-color-medium-rgb: 95,95,95;
  --ion-color-medium-contrast: #ffffff;
  --ion-color-medium-contrast-rgb: 255,255,255;
  --ion-color-medium-shade: #545454;
  --ion-color-medium-tint: #666666;

  --ion-color-dark: #2f2f2f;
  --ion-color-dark-rgb: 47,47,47;
  --ion-color-dark-contrast: #ffffff;
  --ion-color-dark-contrast-rgb: 255,255,255;
  --ion-color-dark-shade: #292929;
  --ion-color-dark-tint: #444444;
}
```



```
:root {
  --ion-color-primary: #0054e9;
  --ion-color-primary-rgb: 0,84,233;
  --ion-color-primary-contrast: #ffffff;
  --ion-color-primary-contrast-rgb: 255,255,255;
  --ion-color-primary-shade: #004acd;
  --ion-color-primary-tint: #1a65eb;

  --ion-color-secondary: #0163aa;
  --ion-color-secondary-rgb: 1,99,170;
  --ion-color-secondary-contrast: #ffffff;
  --ion-color-secondary-contrast-rgb: 255,255,255;
  --ion-color-secondary-shade: #015796;
  --ion-color-secondary-tint: #1a73b3;

  --ion-color-tertiary: #6030ff;
  --ion-color-tertiary-rgb: 96,48,255;
  --ion-color-tertiary-contrast: #ffffff;
  --ion-color-tertiary-contrast-rgb: 255,255,255;
  --ion-color-tertiary-shade: #542ae0;
  --ion-color-tertiary-tint: #7045ff;

  --ion-color-success: #2dd55b;
  --ion-color-success-rgb: 45,213,91;
  --ion-color-success-contrast: #000000;
  --ion-color-success-contrast-rgb: 0,0,0;
  --ion-color-success-shade: #28bb50;
  --ion-color-success-tint: #42d96b;

  --ion-color-warning: #ffc409;
  --ion-color-warning-rgb: 255,196,9;
  --ion-color-warning-contrast: #000000;
  --ion-color-warning-contrast-rgb: 0,0,0;
  --ion-color-warning-shade: #e0ac08;
  --ion-color-warning-tint: #ffca22;

  --ion-color-danger: #c5000f;
  --ion-color-danger-rgb: 197,0,15;
  --ion-color-danger-contrast: #ffffff;
  --ion-color-danger-contrast-rgb: 255,255,255;
  --ion-color-danger-shade: #ad000d;
  --ion-color-danger-tint: #cb1a27;

  --ion-color-light: #f6f8fc;
  --ion-color-light-rgb: 246,248,252;
  --ion-color-light-contrast: #000000;
  --ion-color-light-contrast-rgb: 0,0,0;
  --ion-color-light-shade: #d8dade;
  --ion-color-light-tint: #f7f9fc;

  --ion-color-medium: #5f5f5f;
  --ion-color-medium-rgb: 95,95,95;
  --ion-color-medium-contrast: #ffffff;
  --ion-color-medium-contrast-rgb: 255,255,255;
  --ion-color-medium-shade: #545454;
  --ion-color-medium-tint: #666666;

  --ion-color-dark: #2f2f2f;
  --ion-color-dark-rgb: 47,47,47;
  --ion-color-dark-contrast: #ffffff;
  --ion-color-dark-contrast-rgb: 255,255,255;
  --ion-color-dark-shade: #292929;
  --ion-color-dark-tint: #444444;
}
```



2.7 Theme generator



Here, we use the logo colors for the primary, secondary, and tertiary colors.



Primary

#425EA5

Secondary

#73C3F9

Tertiary

#1B2D79

Tertiary-shade

#18286a

Tertiary-tint

#324286

Success

#2dd55b

Warning

#ffc409

Danger

#c5000f

Light

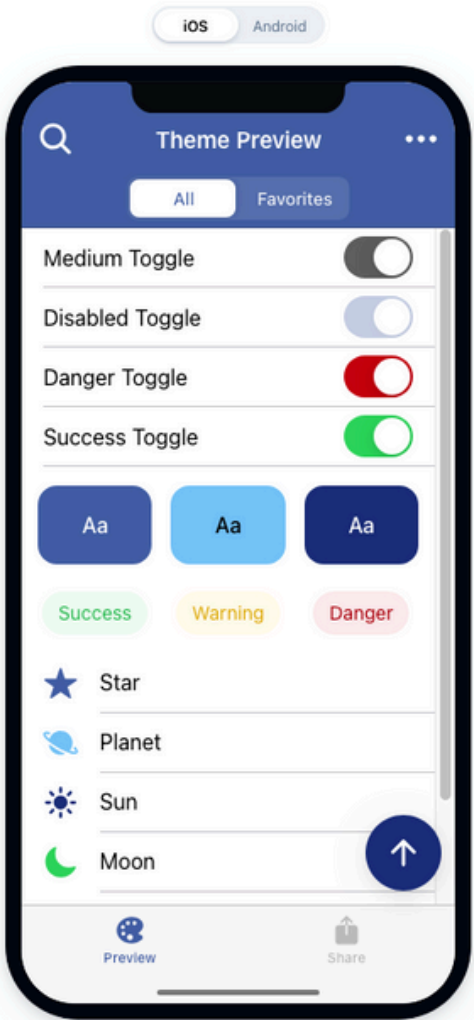
#f6f8fc

Medium

#5f5f5f

Dark

#2f2f2f



```
:root {
  --ion-color-primary: #425EA5;
  --ion-color-primary-rgb: 66,94,165;
  --ion-color-primary-contrast: #ffffff;
  --ion-color-primary-contrast-rgb: 255,255,255;
  --ion-color-primary-shade: #3a5391;
  --ion-color-primary-tint: #556eae;

  --ion-color-secondary: #73C3F9;
  --ion-color-secondary-rgb: 115,195,249;
  --ion-color-secondary-contrast: #000000;
  --ion-color-secondary-contrast-rgb: 0,0,0;
  --ion-color-secondary-shade: #65acdb;
  --ion-color-secondary-tint: #81c9fa;

  --ion-color-tertiary: #1B2D79;
  --ion-color-tertiary-rgb: 27,45,121;
  --ion-color-tertiary-contrast: #ffffff;
  --ion-color-tertiary-contrast-rgb: 255,255,255;
  --ion-color-tertiary-shade: #18286a;
  --ion-color-tertiary-tint: #324286;

  --ion-color-success: #2dd55b;
  --ion-color-success-rgb: 45,213,91;
  --ion-color-success-contrast: #000000;
  --ion-color-success-contrast-rgb: 0,0,0;
  --ion-color-success-shade: #28bb50;
  --ion-color-success-tint: #42d96b;

  --ion-color-warning: #ffc409;
  --ion-color-warning-rgb: 255,196,9;
  --ion-color-warning-contrast: #000000;
  --ion-color-warning-contrast-rgb: 0,0,0;
  --ion-color-warning-shade: #e0ac08;
  --ion-color-warning-tint: #ffca22;

  --ion-color-danger: #c5000f;
  --ion-color-danger-rgb: 197,0,15;
  --ion-color-danger-contrast: #ffffff;
  --ion-color-danger-contrast-rgb: 255,255,255;
  --ion-color-danger-shade: #ad000d;
  --ion-color-danger-tint: #cb1a27;

  --ion-color-light: #f6f8fc;
  --ion-color-light-rgb: 246,248,252;
  --ion-color-light-contrast: #000000;
  --ion-color-light-contrast-rgb: 0,0,0;
  --ion-color-light-shade: #d8dade;
  --ion-color-light-tint: #f7f9fc;

  --ion-color-medium: #5f5f5f;
  --ion-color-medium-rgb: 95,95,95;
  --ion-color-medium-contrast: #ffffff;
  --ion-color-medium-contrast-rgb: 255,255,255;
  --ion-color-medium-shade: #545454;
  --ion-color-medium-tint: #6f6f6f;

  --ion-color-dark: #2f2f2f;
  --ion-color-dark-rgb: 47,47,47;
  --ion-color-dark-contrast: #ffffff;
  --ion-color-dark-contrast-rgb: 255,255,255;
  --ion-color-dark-shade: #292929;
  --ion-color-dark-tint: #444444;
}
```

Primary

#425EA5

Secondary

#73C3F9

Tertiary

#1B2D79

```
:root {
  --ion-color-primary: #425EA5;
  --ion-color-primary-rgb: 66,94,165;
  --ion-color-primary-contrast: #ffffff;
  --ion-color-primary-contrast-rgb: 255,255,255;
  --ion-color-primary-shade: #3a5391;
  --ion-color-primary-tint: #556eae;

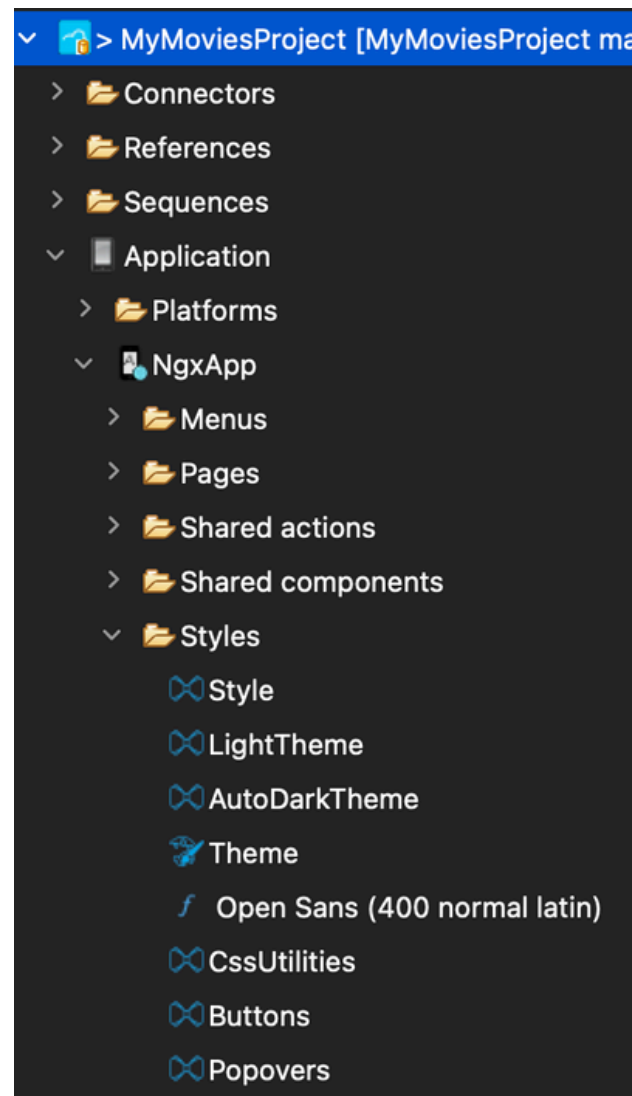
  --ion-color-secondary: #73C3F9;
  --ion-color-secondary-rgb: 115,195,249;
  --ion-color-secondary-contrast: #000000;
  --ion-color-secondary-contrast-rgb: 0,0,0;
  --ion-color-secondary-shade: #65acdb;
  --ion-color-secondary-tint: #81c9fa;

  --ion-color-tertiary: #1B2D79;
  --ion-color-tertiary-rgb: 27,45,121;
  --ion-color-tertiary-contrast: #ffffff;
  --ion-color-tertiary-contrast-rgb: 255,255,255;
  --ion-color-tertiary-shade: #18286a;
  --ion-color-tertiary-tint: #324286;
}
```



2.7 Theme generator

Then, open the file **LightTheme.scss** and replace the default color palette by the one you generated.



```
LightTheme.scss X
1 :root {
2   /** primary **/
3   --ion-color-primary: #3880ff;
4   --ion-color-primary-rgb: 56, 128, 255;
5   --ion-color-primary-contrast: #ffffff;
6   --ion-color-primary-contrast-rgb: 255, 255, 255;
7   --ion-color-primary-shade: #3171e0;
8   --ion-color-primary-tint: #4c8dff;
9
10  /** secondary **/
11  --ion-color-secondary: #3dc2ff;
12  --ion-color-secondary-rgb: 61, 194, 255;
13  --ion-color-secondary-contrast: #ffffff;
14  --ion-color-secondary-contrast-rgb: 255, 255, 255;
15  --ion-color-secondary-shade: #36abe0;
16  --ion-color-secondary-tint: #50c8ff;
17
18  /** tertiary **/
19  --ion-color-tertiary: #5260ff;
20  --ion-color-tertiary-rgb: 82, 96, 255;
21  --ion-color-tertiary-contrast: #ffffff;
22  --ion-color-tertiary-contrast-rgb: 255, 255, 255;
23  --ion-color-tertiary-shade: #4854e0;
24  --ion-color-tertiary-tint: #6370ff;
25
26  /** success **/
27  --ion-color-success: #2dd36f;
28  --ion-color-success-rgb: 45, 211, 111;
29  --ion-color-success-contrast: #ffffff;
30  --ion-color-success-contrast-rgb: 255, 255, 255;
31  --ion-color-success-shade: #28ba62;
32  --ion-color-success-tint: #42d77d;
33
34  /** warning **/
35  --ion-color-warning: #ffc409;
36  --ion-color-warning-rgb: 255, 196, 9;
37  --ion-color-warning-contrast: #000000;
38  --ion-color-warning-contrast-rgb: 0, 0, 0;
39  --ion-color-warning-shade: #e0ac08;
40  --ion-color-warning-tint: #ffc422;
41
42  /** danger **/
43  --ion-color-danger: #eb445a;
44  --ion-color-danger-rgb: 235, 68, 90;
45  --ion-color-danger-contrast: #ffffff;
46  --ion-color-danger-contrast-rgb: 255, 255, 255;
47  --ion-color-danger-shade: #cf3c4f;
48  --ion-color-danger-tint: #ed576b;
49
50  /** dark **/
51  --ion-color-dark: #222428;
```



```
:root {
  --ion-color-primary: #425EA5;
  --ion-color-primary-rgb: 66,94,165;
  --ion-color-primary-contrast: #ffffff;
  --ion-color-primary-contrast-rgb: 255,255,255;
  --ion-color-primary-shade: #3a5391;
  --ion-color-primary-tint: #556eae;

  --ion-color-secondary: #73C3F9;
  --ion-color-secondary-rgb: 115,195,249;
  --ion-color-secondary-contrast: #000000;
  --ion-color-secondary-contrast-rgb: 0,0,0;
  --ion-color-secondary-shade: #65acdb;
  --ion-color-secondary-tint: #81c9fa;

  --ion-color-tertiary: #182D79;
  --ion-color-tertiary-rgb: 27,45,121;
  --ion-color-tertiary-contrast: #ffffff;
  --ion-color-tertiary-contrast-rgb: 255,255,255;
  --ion-color-tertiary-shade: #18286a;
  --ion-color-tertiary-tint: #324286;

  --ion-color-success: #2dd55b;
  --ion-color-success-rgb: 45,213,91;
  --ion-color-success-contrast: #000000;
  --ion-color-success-contrast-rgb: 0,0,0;
  --ion-color-success-shade: #28bb50;
  --ion-color-success-tint: #42d96b;

  --ion-color-warning: #ffc409;
  --ion-color-warning-rgb: 255,196,9;
  --ion-color-warning-contrast: #000000;
  --ion-color-warning-contrast-rgb: 0,0,0;
  --ion-color-warning-shade: #e0ac08;
  --ion-color-warning-tint: #ffca22;

  --ion-color-danger: #c5000f;
  --ion-color-danger-rgb: 197,0,15;
  --ion-color-danger-contrast: #ffffff;
  --ion-color-danger-contrast-rgb: 255,255,255;
  --ion-color-danger-shade: #ad000d;
  --ion-color-danger-tint: #cb1a27;

  --ion-color-light: #f6f8fc;
  --ion-color-light-rgb: 246,248,252;
  --ion-color-light-contrast: #000000;
  --ion-color-light-contrast-rgb: 0,0,0;
  --ion-color-light-shade: #d8dade;
  --ion-color-light-tint: #f7f9fc;

  --ion-color-medium: #5f5f5f;
  --ion-color-medium-rgb: 95,95,95;
  --ion-color-medium-contrast: #ffffff;
  --ion-color-medium-contrast-rgb: 255,255,255;
  --ion-color-medium-shade: #545454;
  --ion-color-medium-tint: #6f6f6f;

  --ion-color-dark: #2f2f2f;
  --ion-color-dark-rgb: 47,47,47;
  --ion-color-dark-contrast: #ffffff;
  --ion-color-dark-contrast-rgb: 255,255,255;
  --ion-color-dark-shade: #292929;
  --ion-color-dark-tint: #444444;
}
```

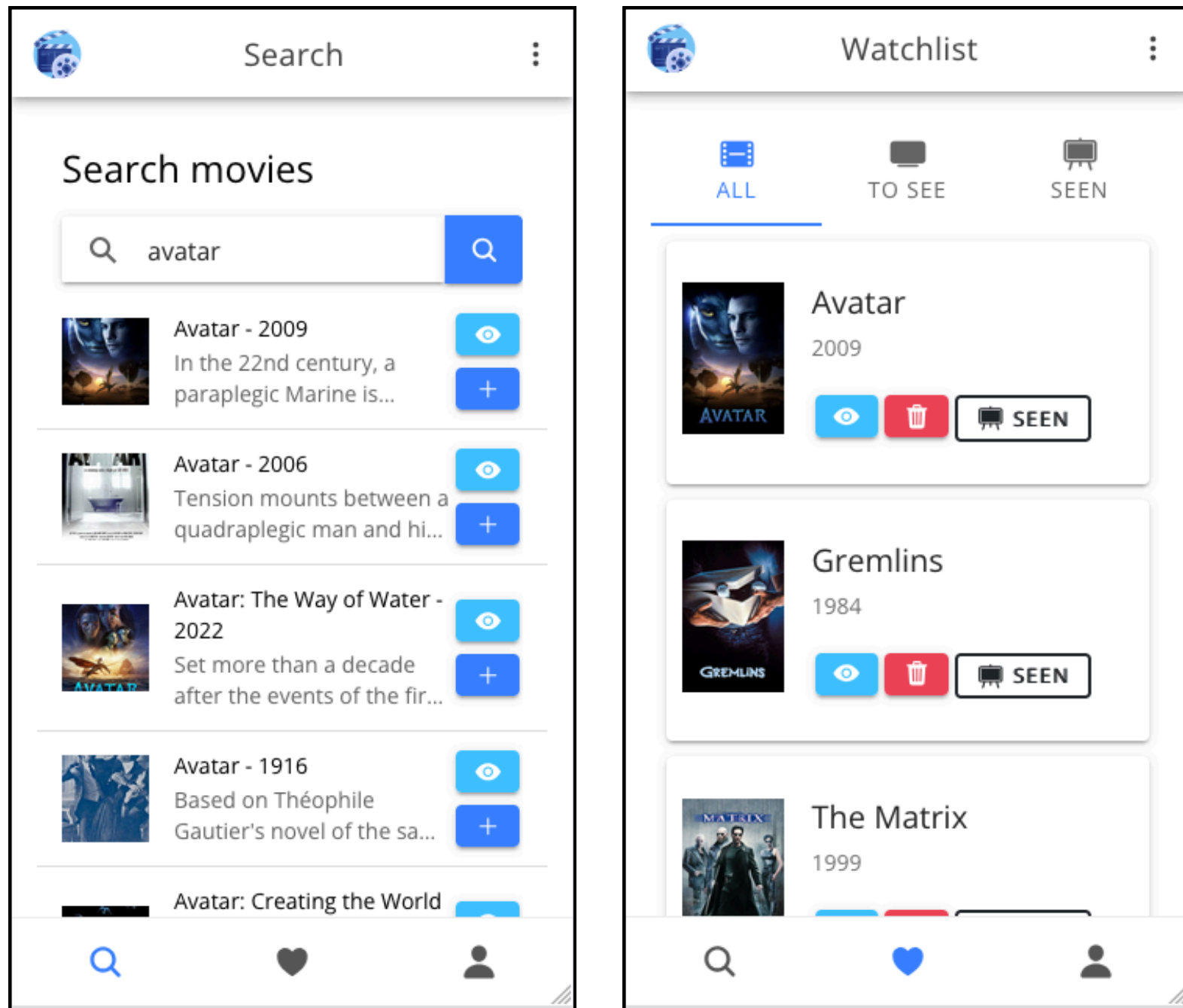


```
LightTheme.scss X
1 :root {
2   --ion-color-primary: #425EA5;
3   --ion-color-primary-rgb: 66,94,165;
4   --ion-color-primary-contrast: #ffffff;
5   --ion-color-primary-contrast-rgb: 255,255,255;
6   --ion-color-primary-shade: #3a5391;
7   --ion-color-primary-tint: #556eae;
8
9   --ion-color-secondary: #73C3F9;
10  --ion-color-secondary-rgb: 115,195,249;
11  --ion-color-secondary-contrast: #000000;
12  --ion-color-secondary-contrast-rgb: 0,0,0;
13  --ion-color-secondary-shade: #65acdb;
14  --ion-color-secondary-tint: #81c9fa;
15
16  --ion-color-tertiary: #182D79;
17  --ion-color-tertiary-rgb: 27,45,121;
18  --ion-color-tertiary-contrast: #ffffff;
19  --ion-color-tertiary-contrast-rgb: 255,255,255;
20  --ion-color-tertiary-shade: #18286a;
21  --ion-color-tertiary-tint: #324286;
22
23  --ion-color-success: #2dd55b;
24  --ion-color-success-rgb: 45,213,91;
25  --ion-color-success-contrast: #000000;
26  --ion-color-success-contrast-rgb: 0,0,0;
27  --ion-color-success-shade: #28bb50;
28  --ion-color-success-tint: #42d96b;
29
30  --ion-color-warning: #ffc409;
31  --ion-color-warning-rgb: 255,196,9;
32  --ion-color-warning-contrast: #000000;
33  --ion-color-warning-contrast-rgb: 0,0,0;
34  --ion-color-warning-shade: #e0ac08;
35  --ion-color-warning-tint: #ffca22;
36
37  --ion-color-danger: #c5000f;
38  --ion-color-danger-rgb: 197,0,15;
39  --ion-color-danger-contrast: #ffffff;
40  --ion-color-danger-contrast-rgb: 255,255,255;
41  --ion-color-danger-shade: #ad000d;
42  --ion-color-danger-tint: #cb1a27;
43
44  --ion-color-light: #f6f8fc;
45  --ion-color-light-rgb: 246,248,252;
46  --ion-color-light-contrast: #000000;
47  --ion-color-light-contrast-rgb: 0,0,0;
48  --ion-color-light-shade: #d8dade;
49  --ion-color-light-tint: #f7f9fc;
```

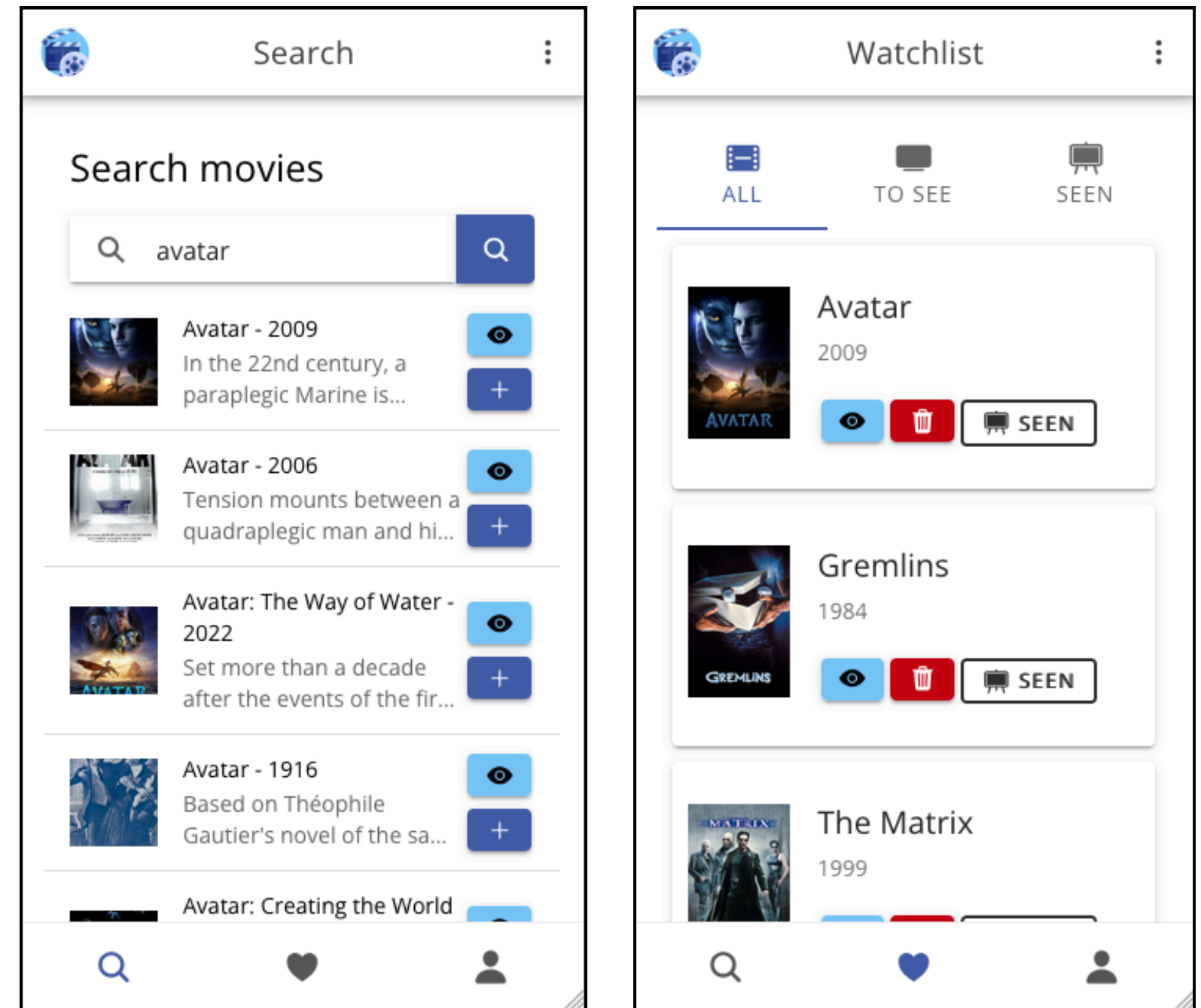


2.7 Theme generator

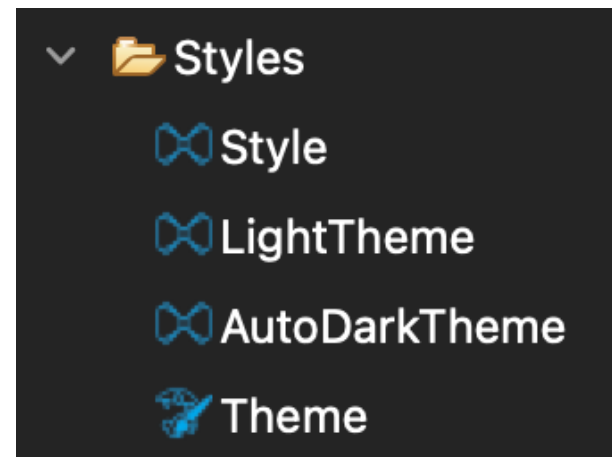
With the original theme



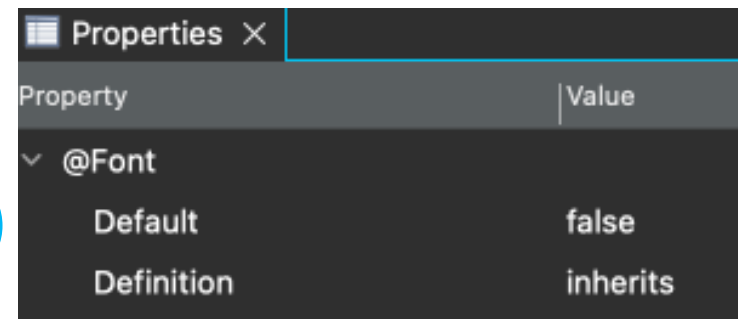
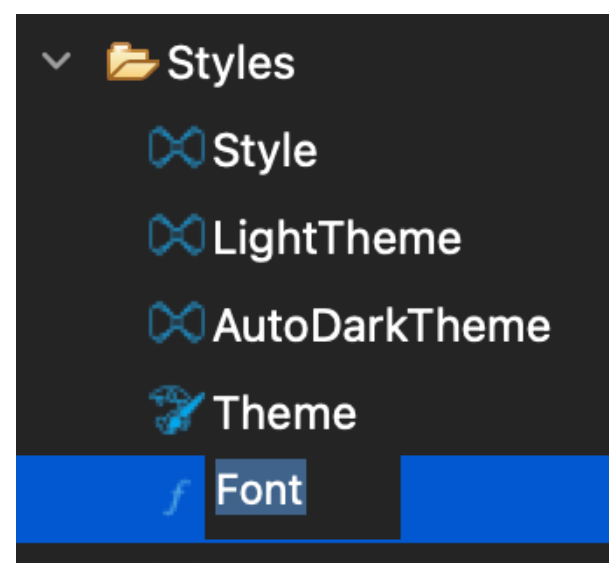
With the new theme



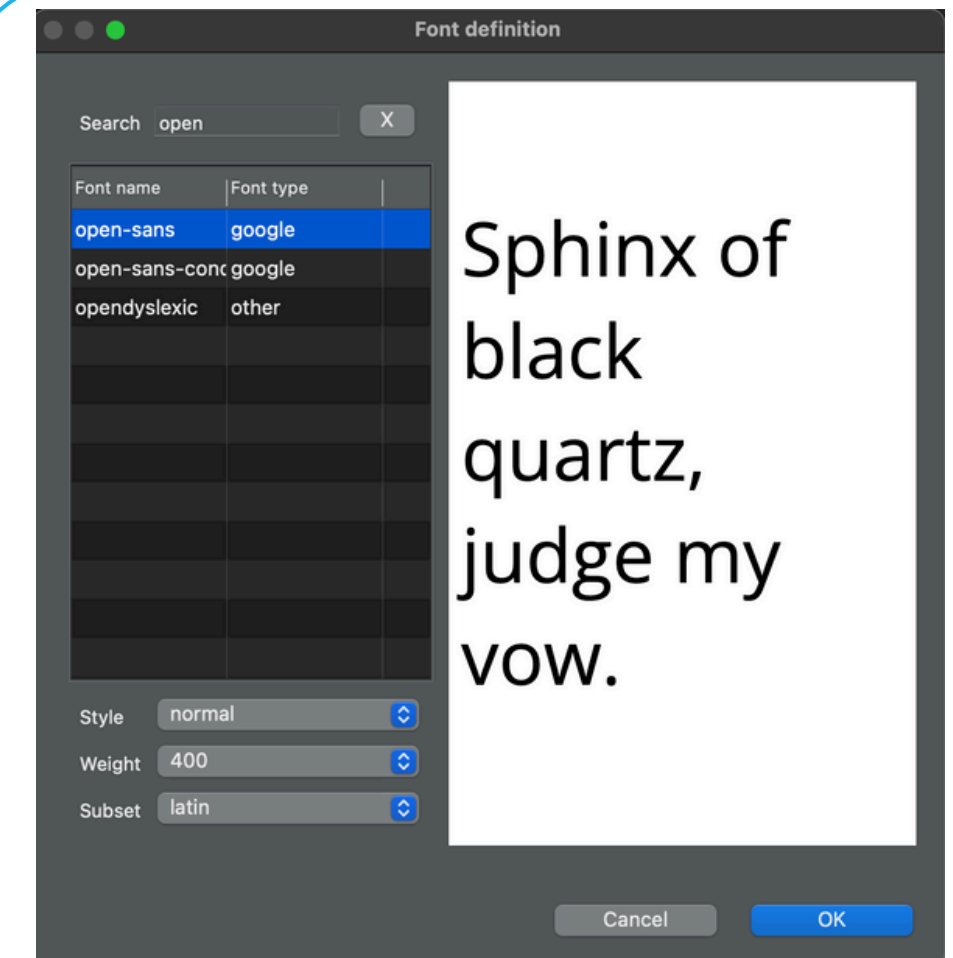
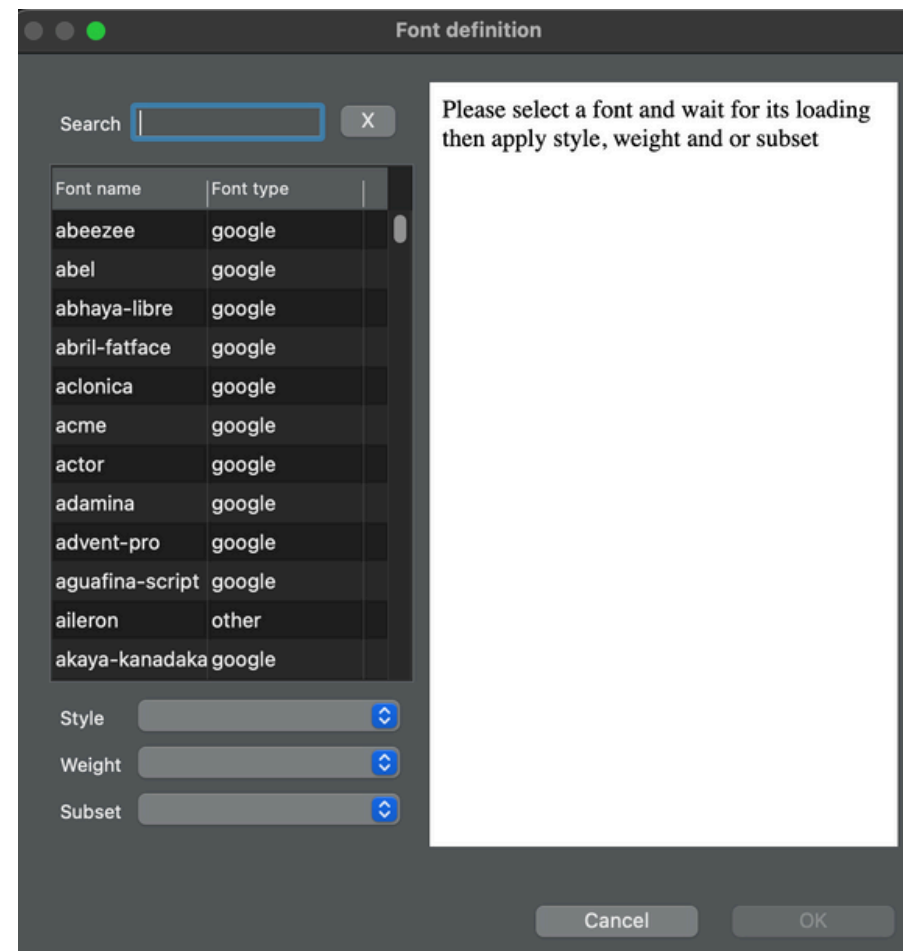
2.8 Add a new font



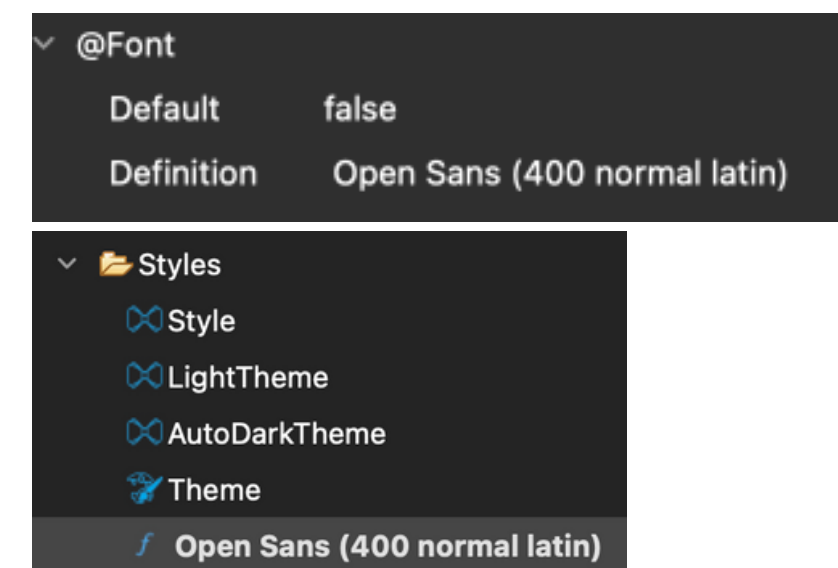
In the **Styles** folder,
add a **Font** component.



In **Font** component's properties,
click on 'inherits', the value of **Definition** property,
to open the **Font definition** panel.



Choose a font and click on OK.



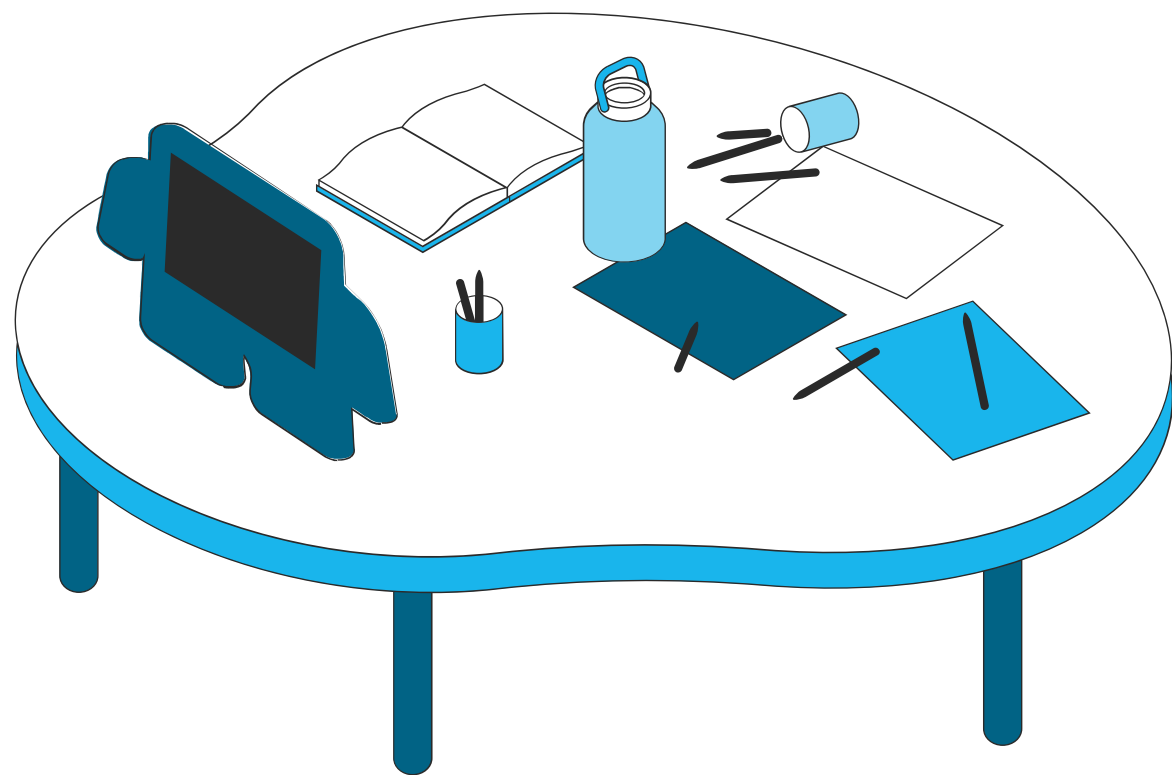
THANKS FOR THE IMAGES



[Cinema icons created by Freepik – Flaticon](https://www.flaticon.com/free-icons/cinema "cinema icons")
https://www.flaticon.com/free-icon/cinema_2828307?term=movies&page=1&position=14&origin=tag&related_id=2828307



3 – Integrate pages with UI components



3.1 Components Overview

3.2 Navigation with tabs

3.3 List & List items

3.4 Segments

3.5 Cards & Grid

3.6 Responsive with Grid

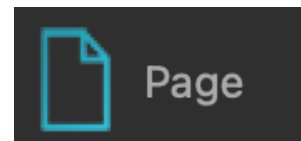
3.7 Flexbox & Alignment with Grid

3.8 Finalize the card layout

3.1 Components overview

Let's have a look on the various components we are going to use in this tutorial.

In addition to the styling components in the **Theming & Styling chapter**, we already know a few of them.



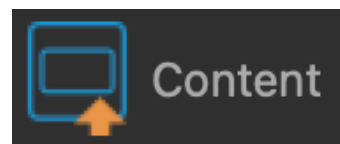
Page component

Used to create pages for the application,
it contains three root elements :
Header, Content, Footer.



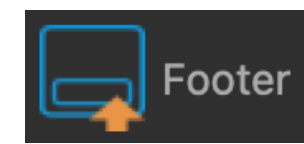
Header component

Root component at the top of the page.
It may hold a **NavBar** or a **ToolBar** component.



Content component

Root component at the middle of the page.
This is the main space
where are inserted other components.



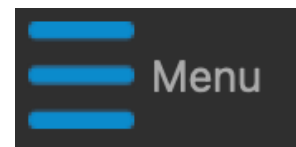
Footer component

Root component at the bottom of the page.
It can be used as a wrapper for a **ToolBar**
to make sure the content area is sized correctly.



3.1 Components overview

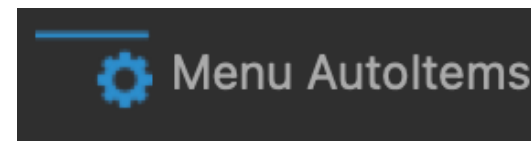
Menu Components



Menu

Menu component

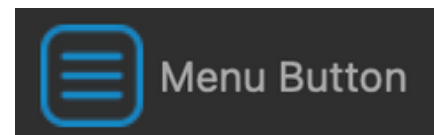
Used to navigate through pages and may contain one **MenuAutoItems** and/or several **MenuItem** components. An application may have several menus but only the first one is used by default.



Menu AutoItems

Menu AutoItems component

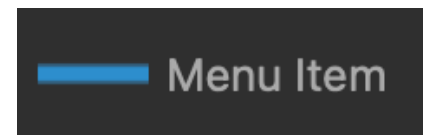
Added once inside a **Menu component**, and used to generate automatically a menu with links to visible pages. (i.e. an **enabled** page with its **Include in auto menu property** set to true.)



Menu Button

Menu Button component

Used to create automatically the menu icon and functionality to open a menu on a page.



Menu Item

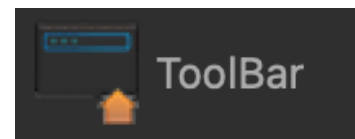
Menu Item component

Added in a **Menu component**, one per page you'd like to navigate to, and used to navigate to the associated page, when clicked.



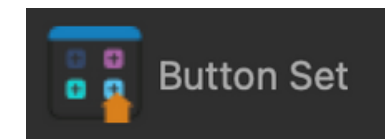
3.1 Components overview

ToolBar Components & Button



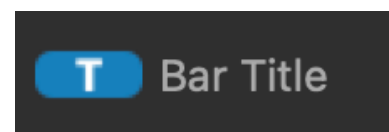
ToolBar

Generic bar used in an application as a header, sub-header, footer, or sub-footer.



Button Set

Acts as a set of buttons used in other components, like **Generic Item** or **NavBar** component.



Bar Title

Sets the title of a **ToolBar** or **NavBar** component.



Button

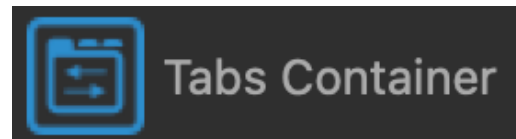
Used to navigate through the application, and communicate an action which occurs when the user taps it.

To handle button click, add an **Event component** with chained **Actions** under the **Button**.



3.1 Components overview

Tabs Components

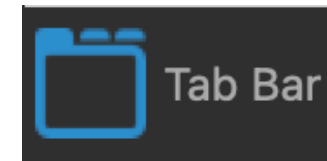


Tabs Container

Tabs Container

Used to implement a **tab-based navigation**, it works as a **router outlet**.

It contains a **Tab Bar** with a set of **Tab Buttons**. They provide styling, UI feedback and mechanism to switch between tabs.



Tab Bar

Tab Bar

Contains a set of **Tab Buttons**, and must be provided inside of a **Tabs Container** to communicate with each tab.



Tab Button

Tab Button

Inserted in a **Tab Bar**, each **Tab Button** is **associated to a page** specified in its properties.

Important: The associated page must contain a **Content component** `<ion-content>` to work properly with a **Tab Button**.



3.1 Components overview

Segments Components



Segment Container

Segment Container

Segments are used for **toggleing between different views** inside the main content or a **ToolBar**, in the **same page**.
If you **navigate between pages**, use **Tabs** instead.

The **Segment Container** displays a group of **related buttons**, or **segmented controls**, in a horizontal row.



Segment Button

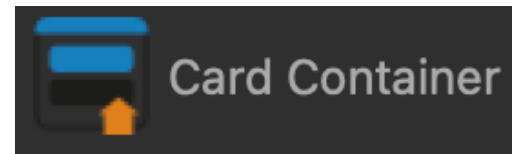
Segment Button

When added to a **Segment Container**, a **Segment Button** is assigned a value through its **Value property**.
Only one **SegmentButton** can be selected at a time.



3.1 Components overview

Cards Components

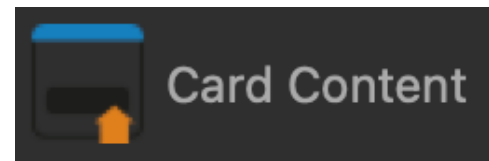


Card Container

Card Container

Cards are **containers** that display content such as text, images, buttons, and lists.

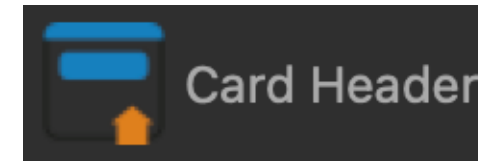
A card can be a **single component**, but is often made up of **several components**: card header, card title, card subtitle, card content.



Card Content

Card Content

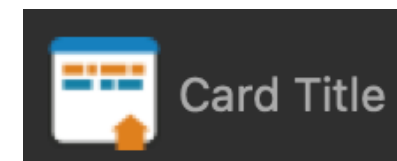
Used inside a **Card Container**, after the **Card Header**, this is the **content** part of the card.



Card Header

Card Header

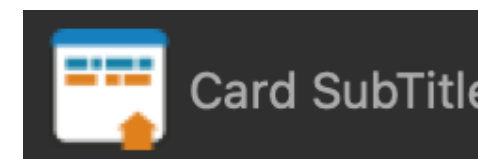
Used inside a **Card Container**, this is the **header** part of the card.



Card Title

Card Title

Used inside a **Card Header**.



Card SubTitle

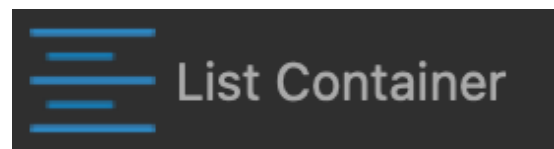
Card SubTitle

Used inside a **Card Header**.



3.1 Components overview

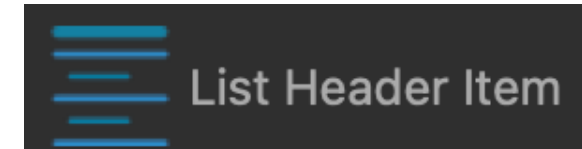
List Components



List Container

List Container

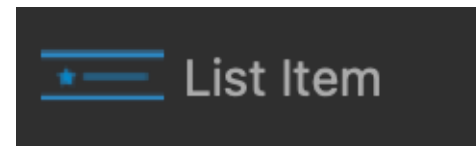
Used as a **container** for **multiple rows of List Items**.



List Header Item

List Header Item

Each **List Container** can include a **List Header** at the top of the list before **List Items**.



List Item

List Item

In a **List Item**, you can insert many other components such as **Icons**, **Avatars** or **Thumbnails**

It can be made **clickable** like a **Button**

by setting its **Button type property** to **ion-item**.

It can also contain **Sliding Options** to reveal buttons when swiped left or right.



Thumbnail Item

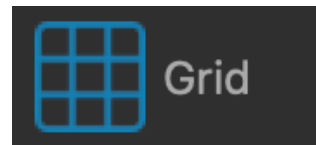
Thumbnail Item

Square component usually wrapping an image or icon, it can be used to display a group of larger images or provide a preview of the full-size image, by itself or inside an element.



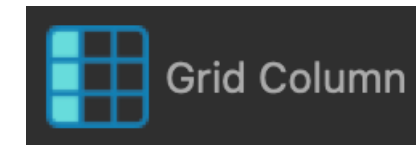
3.1 Components overview

Grid Components



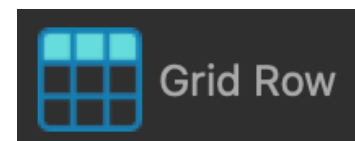
Grid

Based on flexbox, the **Grid component** is composed of three units :
grid, rows and columns.



Grid Col

The **Grid Column** component must be inserted in a **Grid Row** and contains other components or HTML.



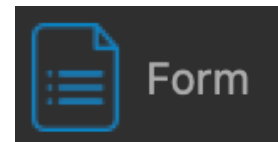
Grid Row

The **Grid Row component** must be inserted in a **Grid** and contains **Grid Column components**.
It is composed of 12 slots that can be filled with **Grid Columns**.



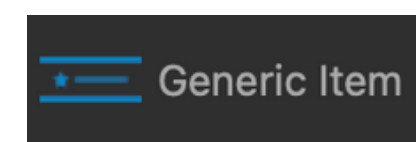
3.1 Components overview

Form Components



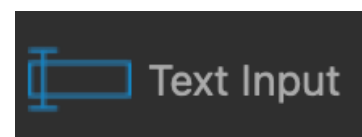
Form

Used to collect user inputs, like an HTML `<form>`.
Based on the **Angular 2's form library**,
it acts as a **reactive form**.
Every form's inputs should be added
under a **Generic Item** to work properly in **Form**.



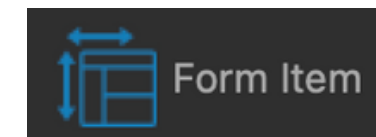
Generic Item

Container for text, icons, avatars, images, inputs.
It is also used as rows in a **List**
It can be made clickable
with its **button type property** set to **ion-item**,
or contain **Sliding Options**.



Text Input

Used for text type inputs:
text, password, email, number, search, tel, and url.



Form Item

Used to contain form's input components,
such as **Label** or **Text Input**.



3.1 Components overview

Form Components



Select Container

Select Container

Based on Ionic's select component, it is used to sort through and select the preferred option or options. When it is tapped, a dialog will appear, displaying all the **Select Options** added in the **Select Container**.



Select Option

Select Option

Each **Select Option component** must be placed under a **Select Container** component to give a list of selectable options.



Text Area

Text Area

Used for multiline text type inputs: text, password, email, number, search, tel, and url.



Search Bar

Search Bar

Used to search or filter items.



3.1 Components overview

Text Components



Simple component that can be used to style the text color of an inner **Text component**.



As the base component to display text, it acts as a standard **HTML Text node** which is mostly the **inner content of an HTML element**.

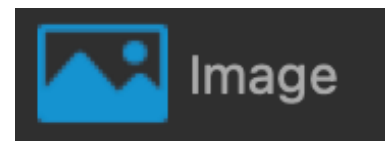
You can add a **Text component** to any component, like **Button**, or **HTML components**, like **P** or **H1**.

If you add several **Text components** to the same component, they will be concatenated.



3.1 Components overview

Image & Icon Components



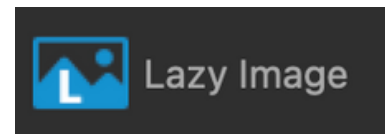
Image

It acts as the standard HTML img tag.
An **Image** can be used alone or within
a **Thumbnail Item** or **Avatar Item component**.



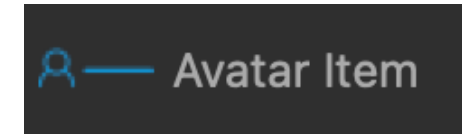
Icon

Used to display icons from the **Ionicons library**,
or a custom SVG. **Icons** can be used on their own,
or inside of a number of components.



Lazy Image

Used to lazily load an image whenever it is
in the viewport.
This is extremely useful when generating a large list
as images are only loaded when they're visible.



Avatar Item

Circular components that usually wrap an image
or icon, Avatars can be used to represent a person
or an object.
They can be used by themselves or inside an element.



3.1 Components overview

HTML Components

These components act as standard HTML tags.

`</> Div`

Div

`</> Break`

Break

`</> Span`

Span

`</> Heading`

Heading

`</> H1`

`</> H2`

`</> H3`

`</> H4`

`</> H5`

`</> H6`

`</> Paragraph`

Paragraph



3.1 Components overview

Custom Components



Tag component

Tag acts as a standard HTML element defined with a starting and an ending tag.

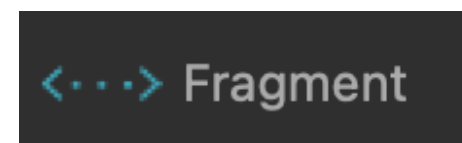
All mobile components in the palette are already defined with a tag name.

You can create your own mobile elements by using the **Tag component** specifying its **Tag name property**.



Attr component

Acting as a standard HTML attribute, it can be added to any component to create a custom attribute to customize a component behavior.



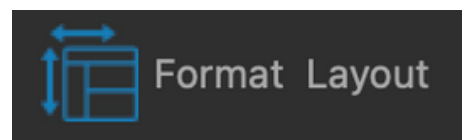
Fragment component

Acting as a standard HTML fragment inside the HTML document, Fragment is used to write your own custom HTML code in a page.



3.1 Components overview

Misc Components



Format Layout

This component can be used on any component in order

- to adjust the padding and margin
- to modify the text or element placement with **Alignment** (text-align in CSS), **Float** , **Hidden**
- To set the alignment of flex elements with
 - **Container** : for a Flex container.
 - **Item** : for a Flex item.

Properties X	
Property	Value
▼ @Border Display	
Hidden	not set
▼ @Component	
Identifier	
▼ @Component Display	
Hidden	not set
Hidden 0-576 px (xs)	not set
Hidden 1200-over px (xl)	not set
Hidden 576-768 px (sm)	not set
Hidden 768-992 px (md)	not set
Hidden 992-1200 px (lg)	not set
▼ @Component Margin	
Margin	not set
▼ @Component Padding	
Padding	not set
▼ @Flex Alignment	
Container	not set
Item	not set

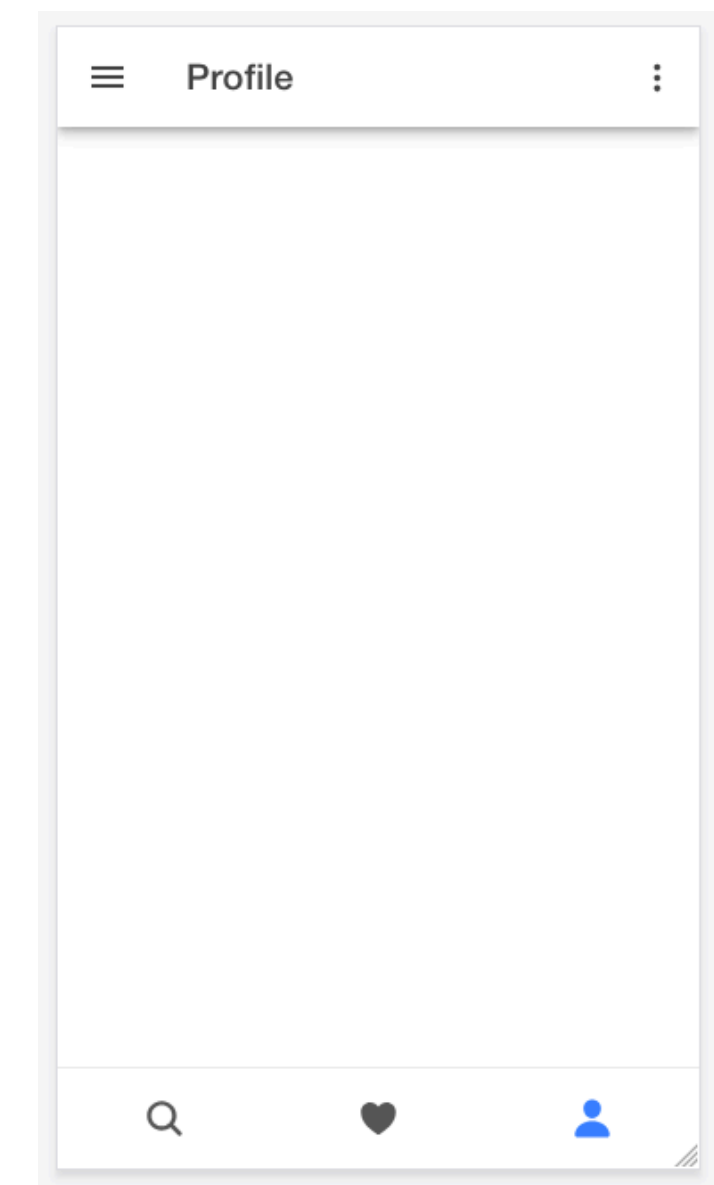
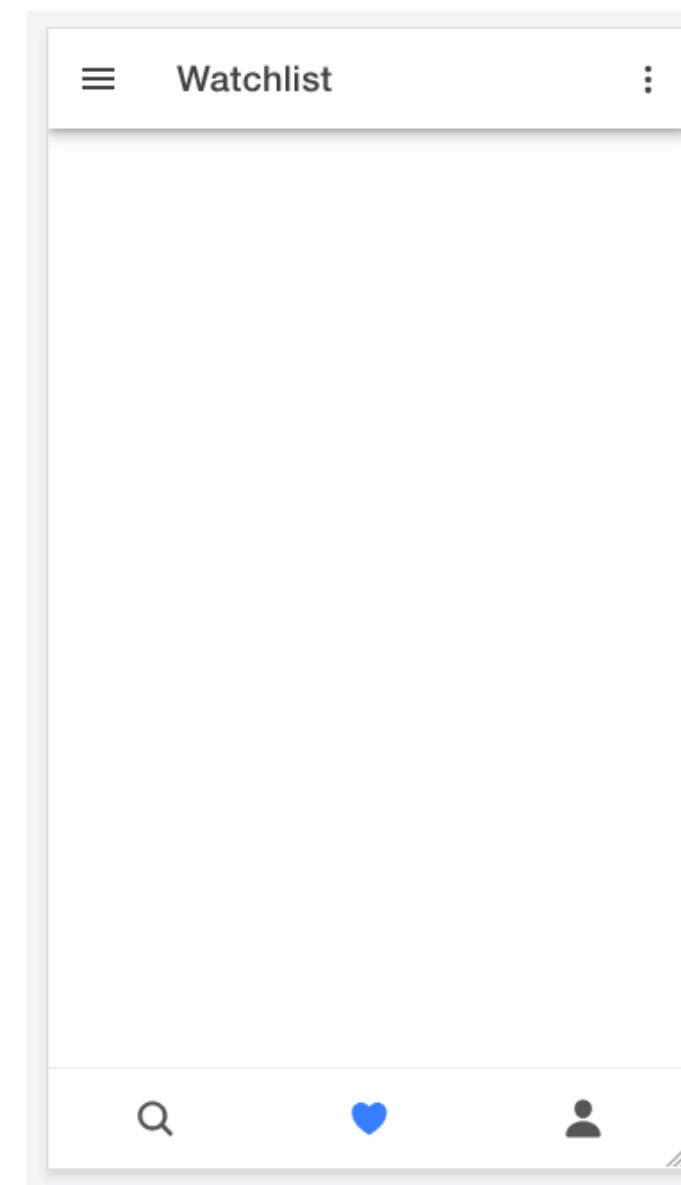
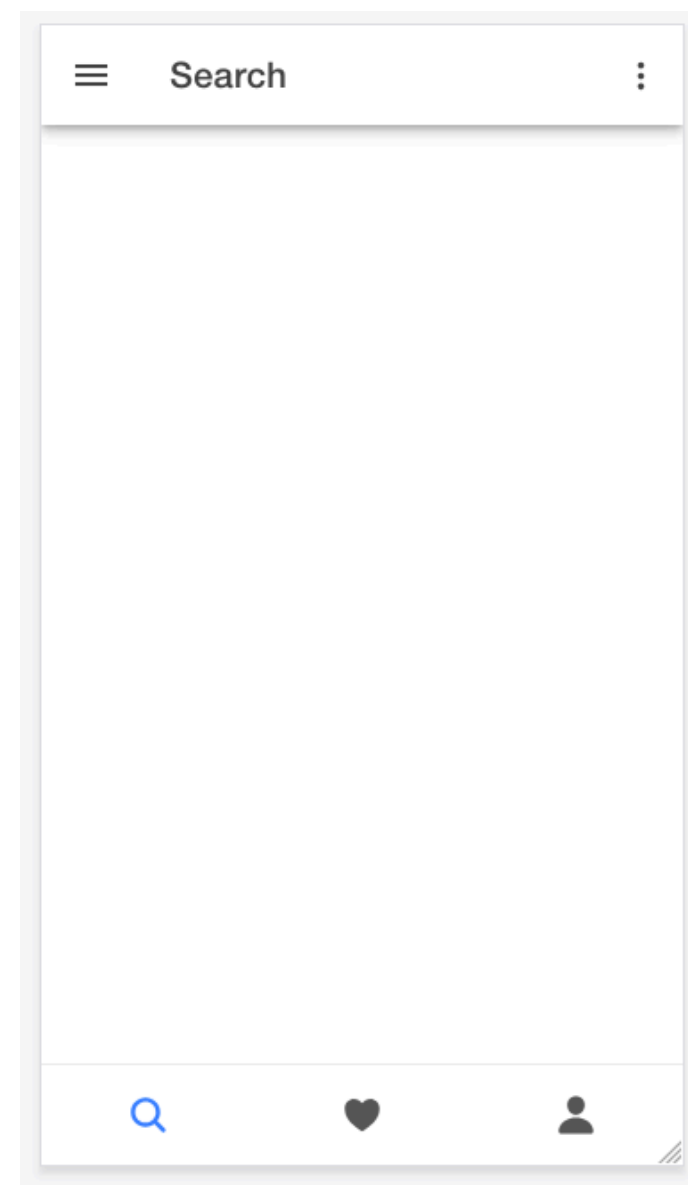
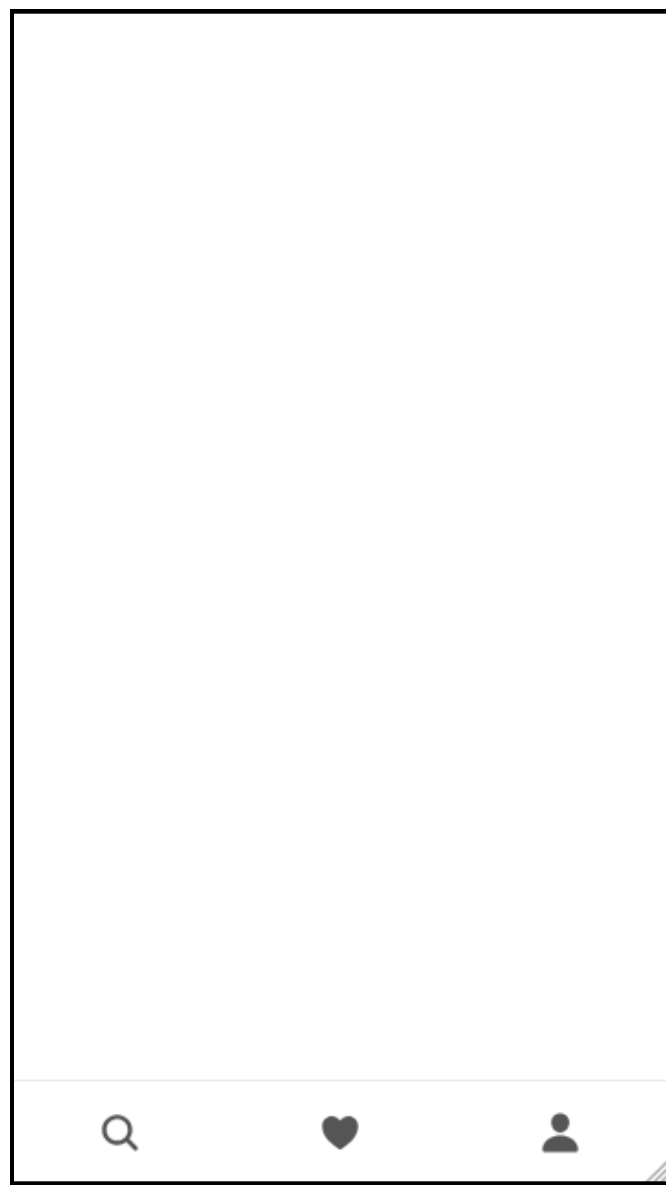
▼ @Float Placement	
Float	not set
Float 0-576 px (xs)	not set
Float 1200-over px (xl)	not set
Float 576-768 px (sm)	not set
Float 768-992 px (md)	not set
Float 992-1200 px (lg)	not set
▼ @Text Alignment	
Alignment	not set
Alignment 0-576 px (xs)	not set
Alignment 1200-over px (xl)	not set
Alignment 576-768 px (sm)	not set
Alignment 768-992 px (md)	not set
Alignment 992-1200 px (lg)	not set
▼ @Text Transformation	
Transformation	not set
Transformation 0-576 px (xs)	not set
Transformation 1200-over px (xl)	not set
Transformation 576-768 px (sm)	not set
Transformation 768-992 px (md)	not set
Transformation 992-1200 px (lg)	not set



3.2 Navigation with tabs

Now, let's start to create our frontend ! We're going to use **tabs to navigate between pages**.

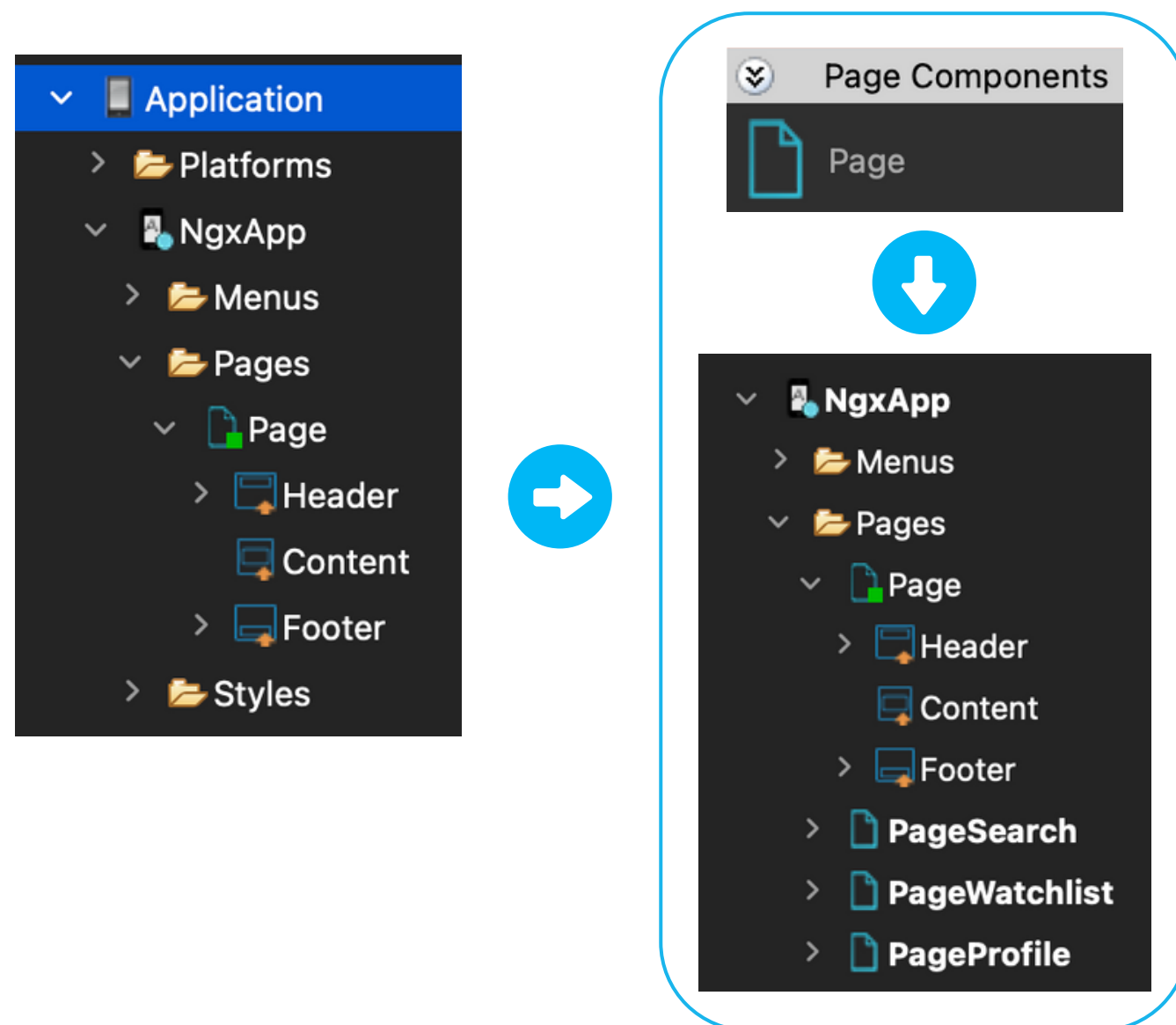
Page **Root** where we create our tabs Pages **Search, Watchlist** and **Profile** where we navigate with the **TabButtons**.



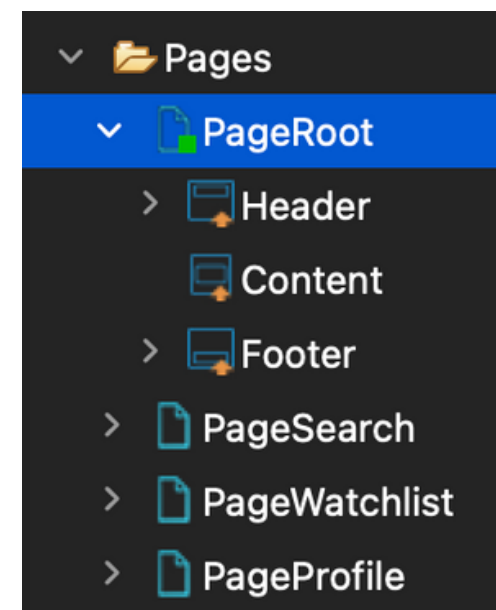
3.2 Navigation with tabs

By default, we have a **root page** called **Page** and pages are created with **Header**, **Content** and **Footer** components inside.

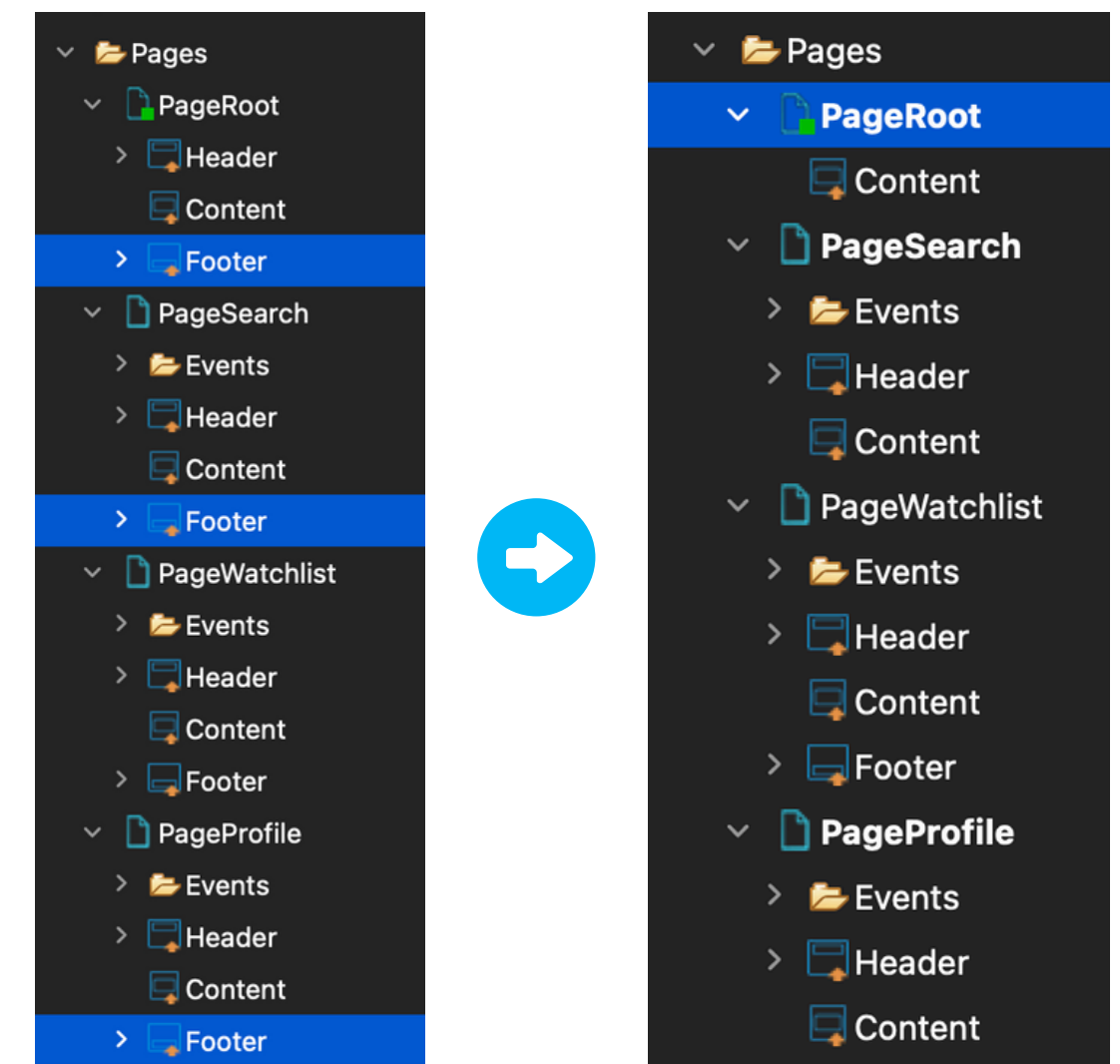
Let's add 3 pages to our app : **Search**, **Watchlist** and **Profile**.



Let's rename the **root page** as **PageRoot**.

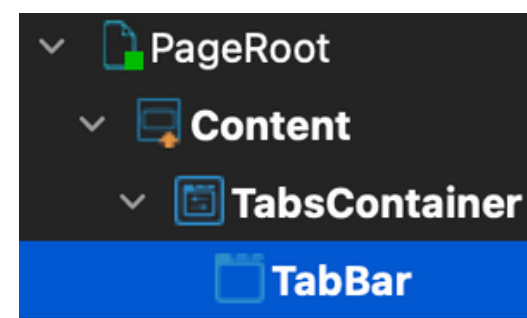
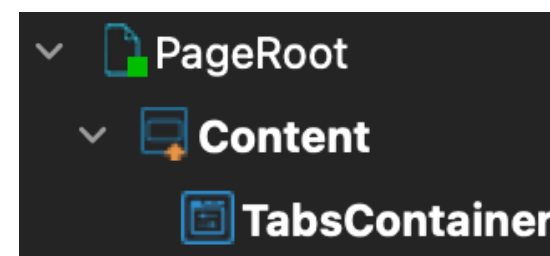
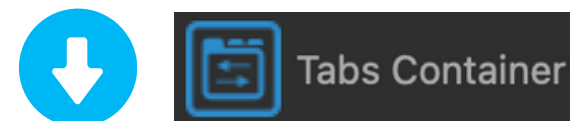
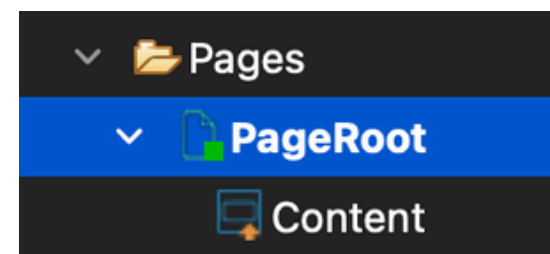


Let's remove the **footer** for all pages and **Header** for page **Page**.

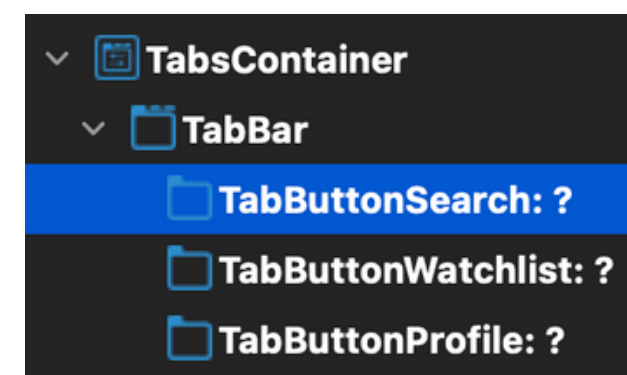
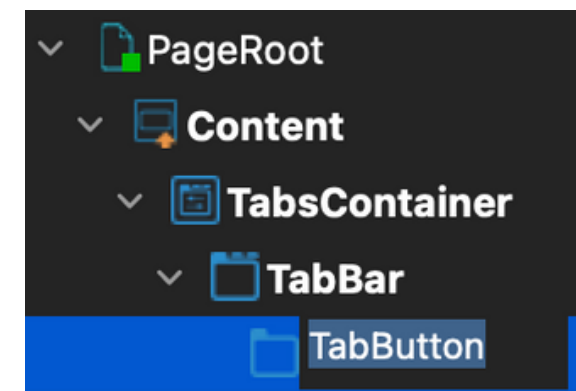


3.2 Navigation with tabs

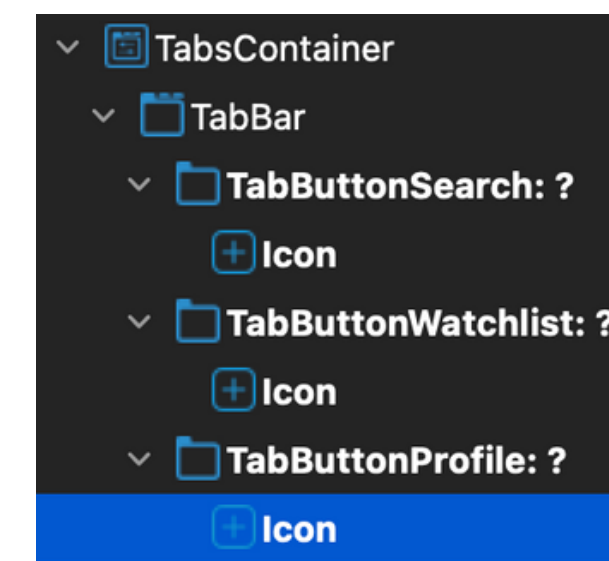
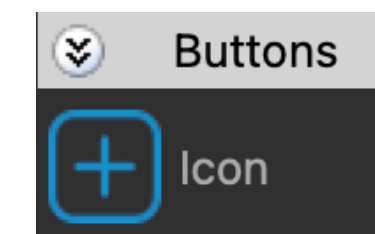
In **PageRoot's Content**,
let's add a **TabsContainer**,
and a **TabBar** in TabsContainer.



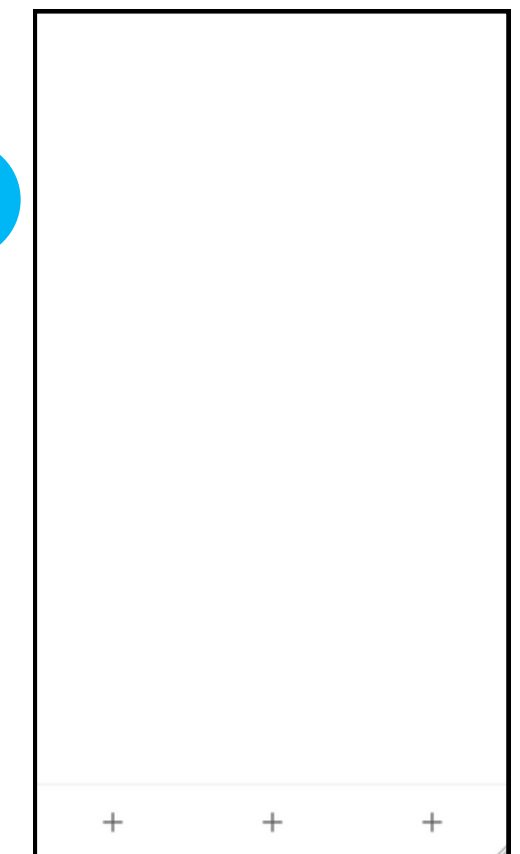
In **TabBar**,
let's add 3 **TabButtons**:
TabButtonSearch,
TabButtonWatchlist,
and **TabButtonProfile**.



Then add
an **Icon** component
in each TabButton.

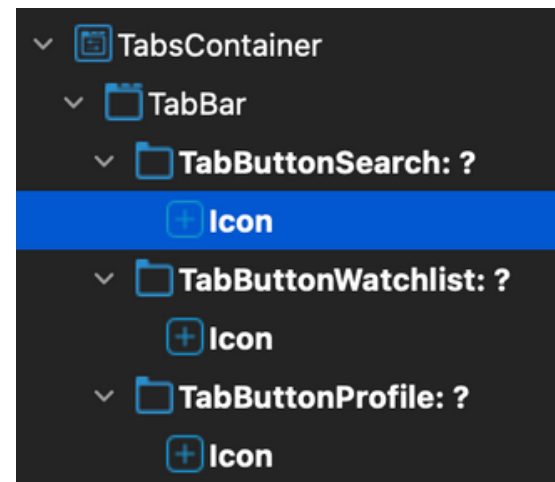


The **TabsContainer**
is displayed
at the bottom
of **PageRoot**.

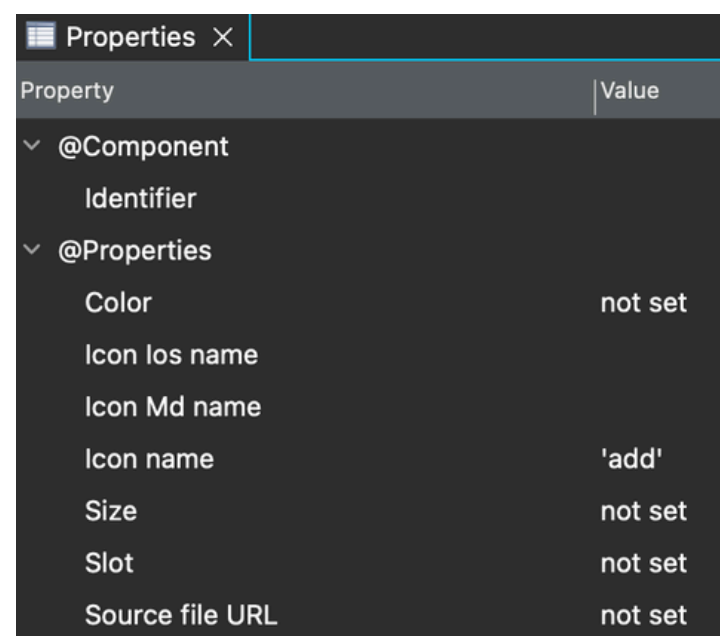


3.2 Navigation with tabs

In **TabButtonSearch**,
click on **Icon** to see its properties.



The icon by default
of the Icon component is 'add'.

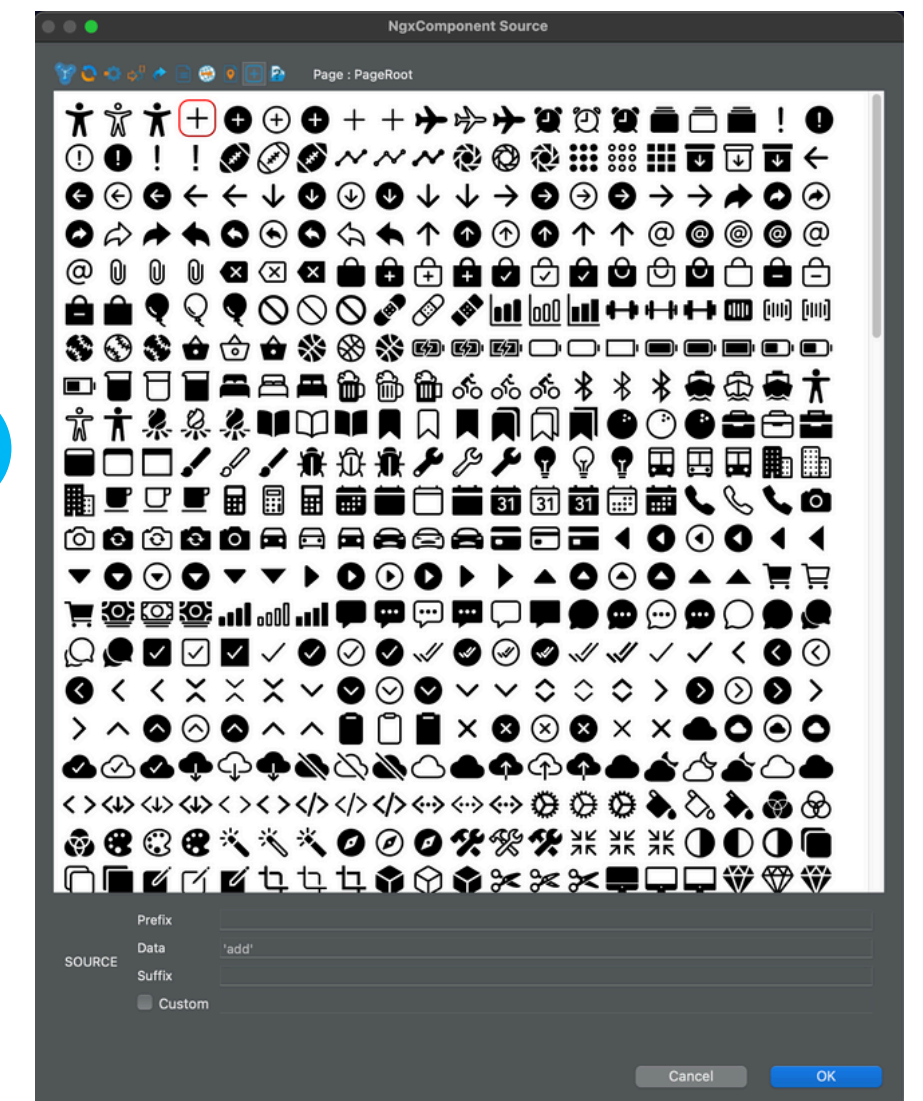


In Icon's properties,
click on **Icon name property**
to see the **buttons ...**, TX, TS, SC.



Then, click on **...** or **SC**
to open the panel
NgxComponentSource.

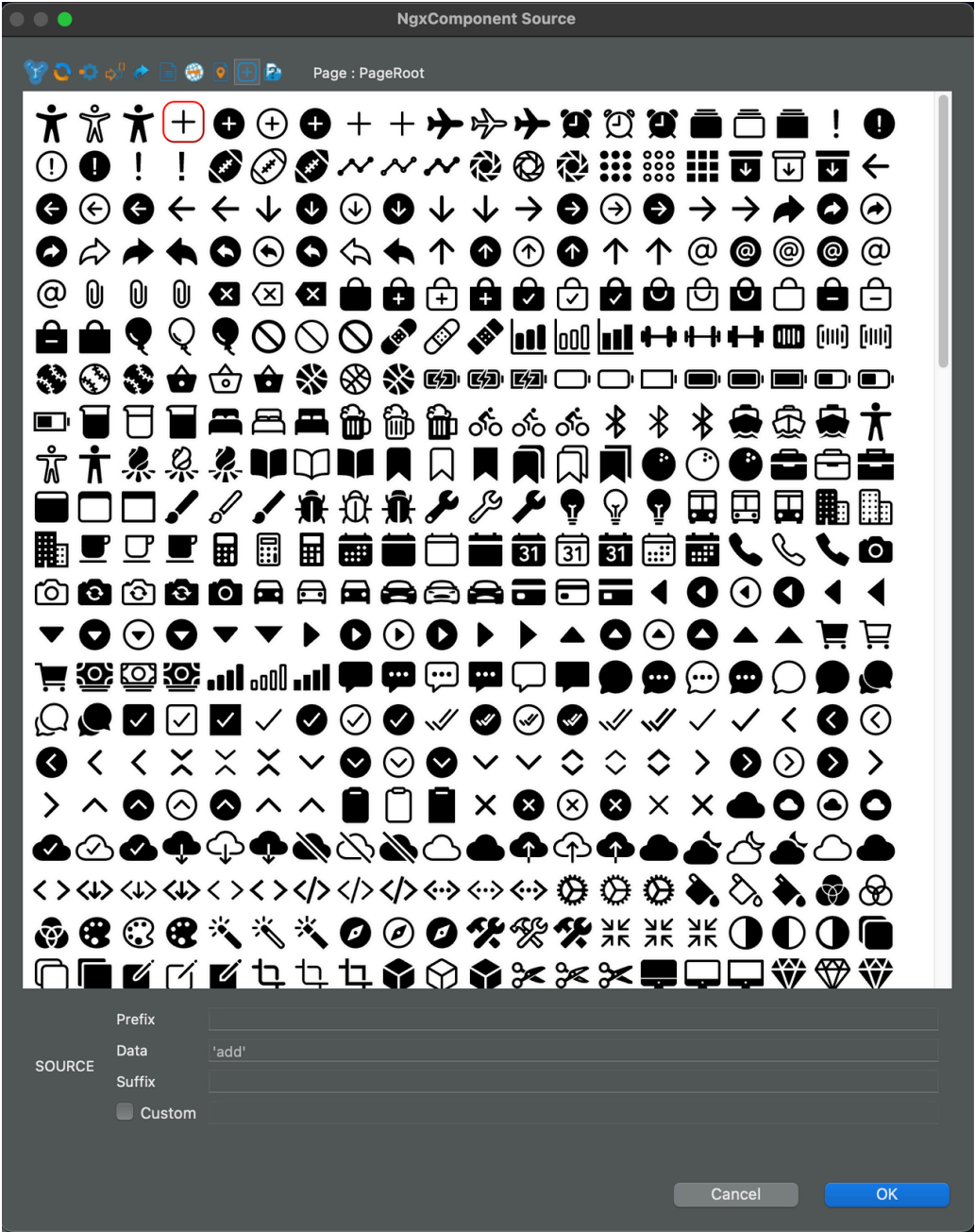
There, you can see
all icons available in Ionic.



3.2 Navigation with tabs



In the panel NgxComponentSource...



Prefix

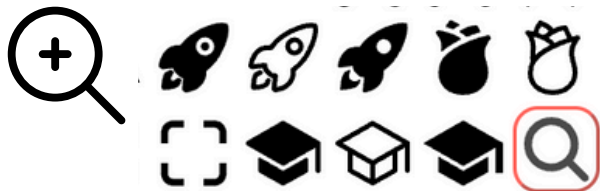
Data 'add'

Suffix

☐ Custom



Click on the 'search' icon then **OK** to choose it.



Prefix

Data 'search'

Suffix

☐ Custom

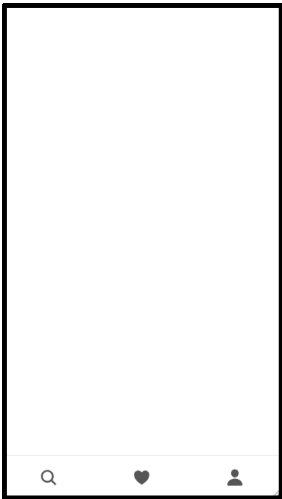


Property	Value
@Component	
Identifier	
@Properties	
Color	not set
Icon los name	
Icon Md name	
Icon name	'search'
Size	not set
Slot	not set
Source file URL	not set

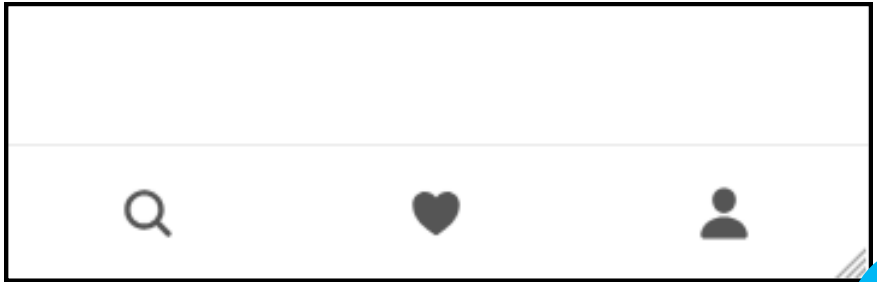


Do the same thing

- for **TabButtonWatchlist** with icon 'heart'
- for **TabButtonProfile** with icon 'person'.



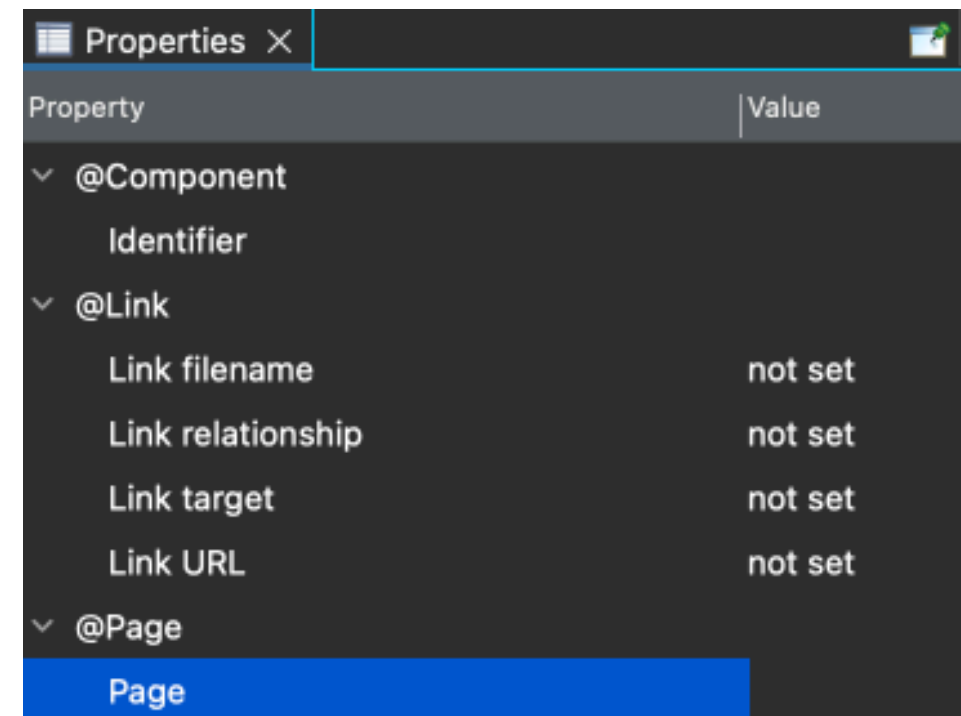
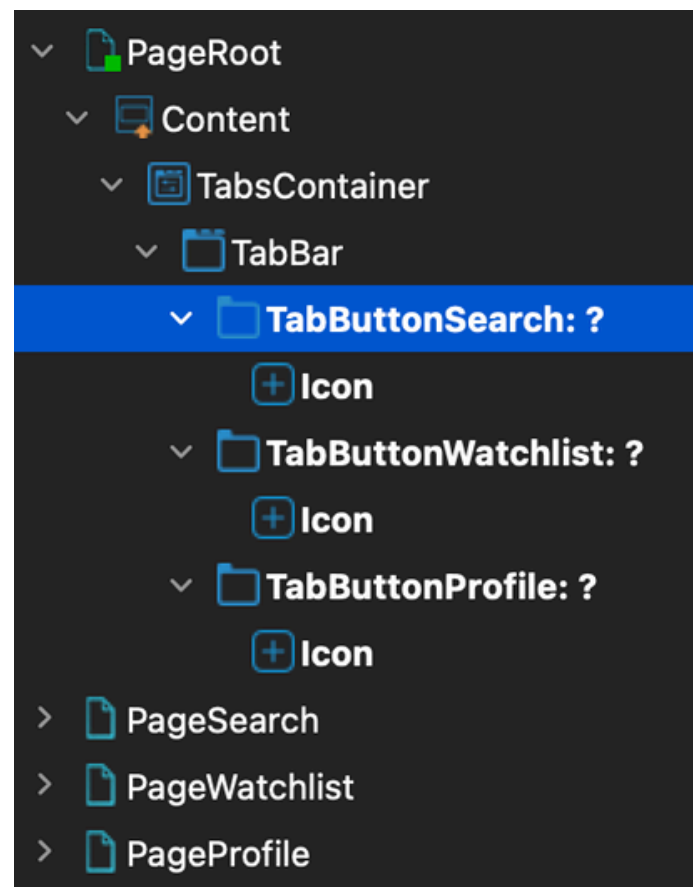
TabsContainer
in **PageRoot**.



3.2 Navigation with tabs

Now, let's link the **TabButtons** to pages.

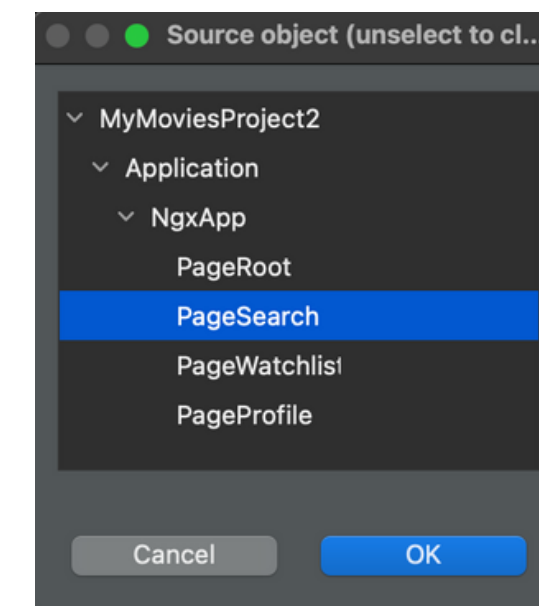
Click on **TabButtonSearch** to see its properties.



In the properties, click on **Page** to see the button ...



Click on ... to open the **Source object panel**.

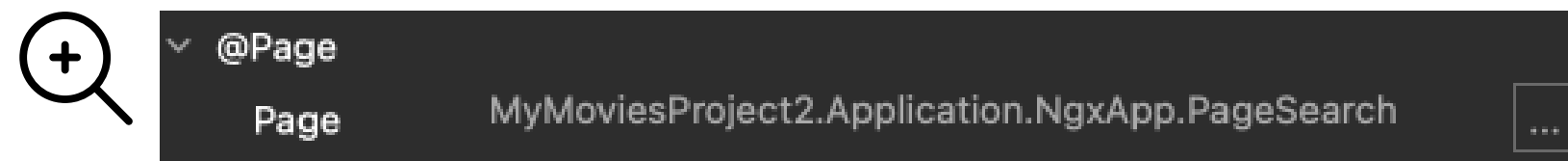
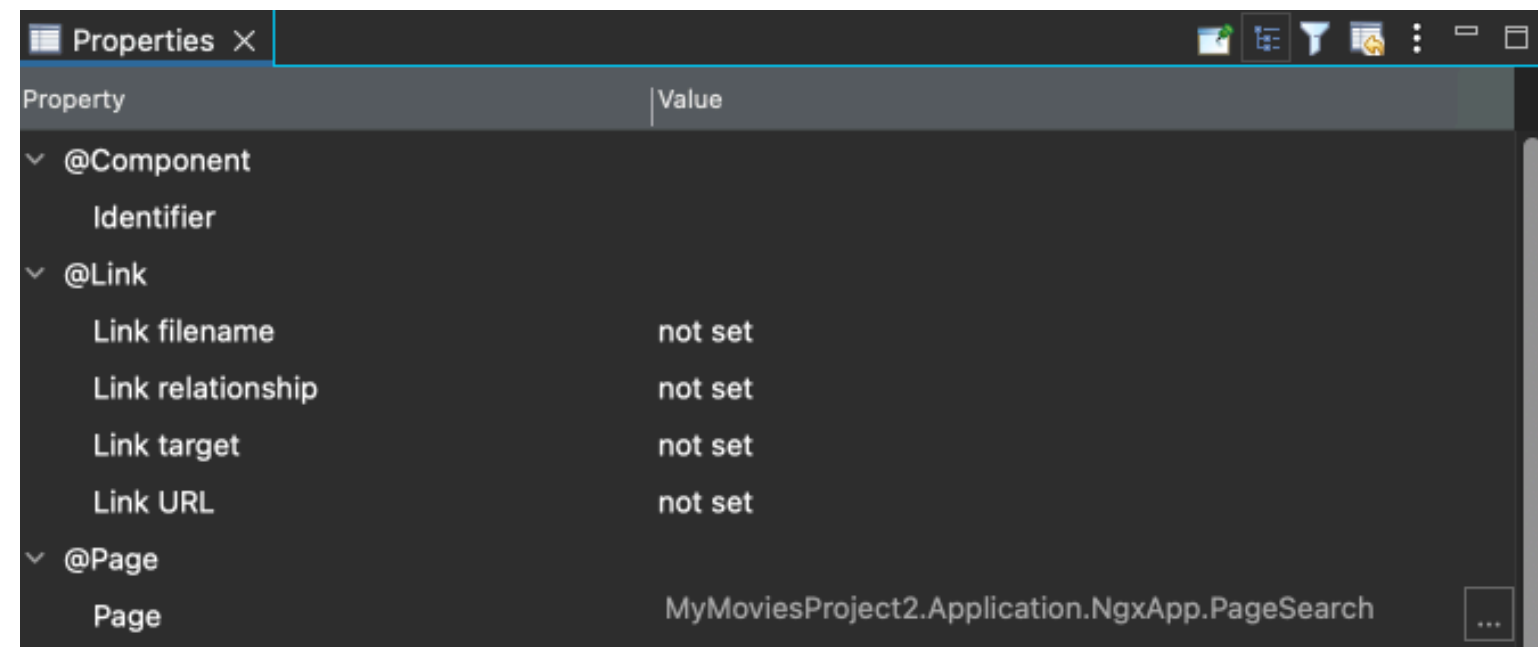


In the **Source object panel**, select **PageSearch** and click on **OK**.

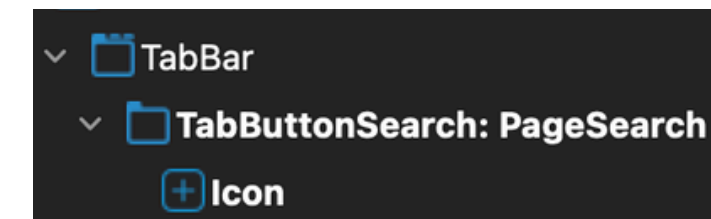


3.2 Navigation with tabs

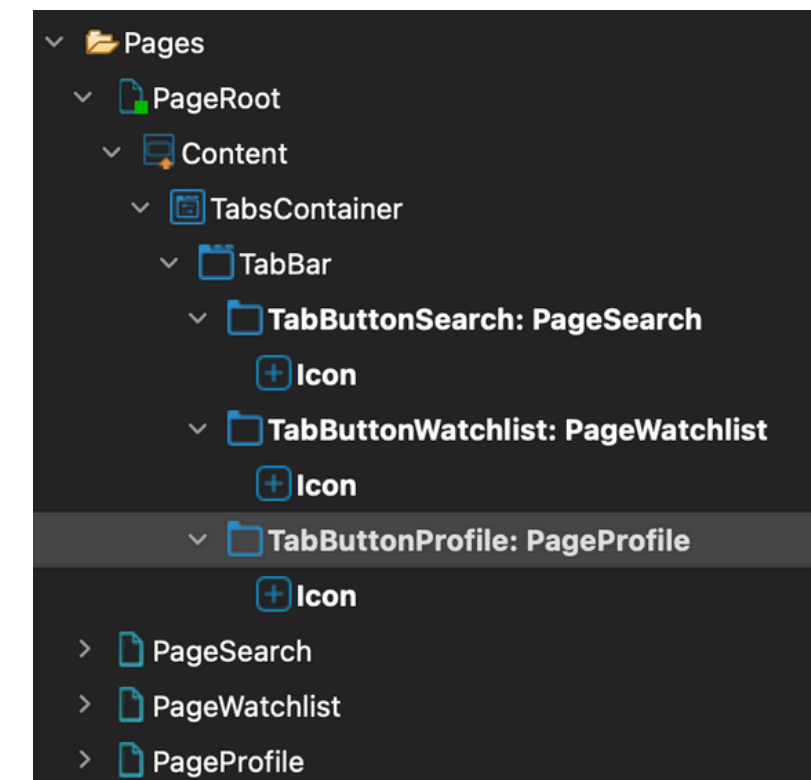
In **TabButtonSearch**'s properties, **PageSearch** is now selected as **Page**.



And appears as such in the treeview.



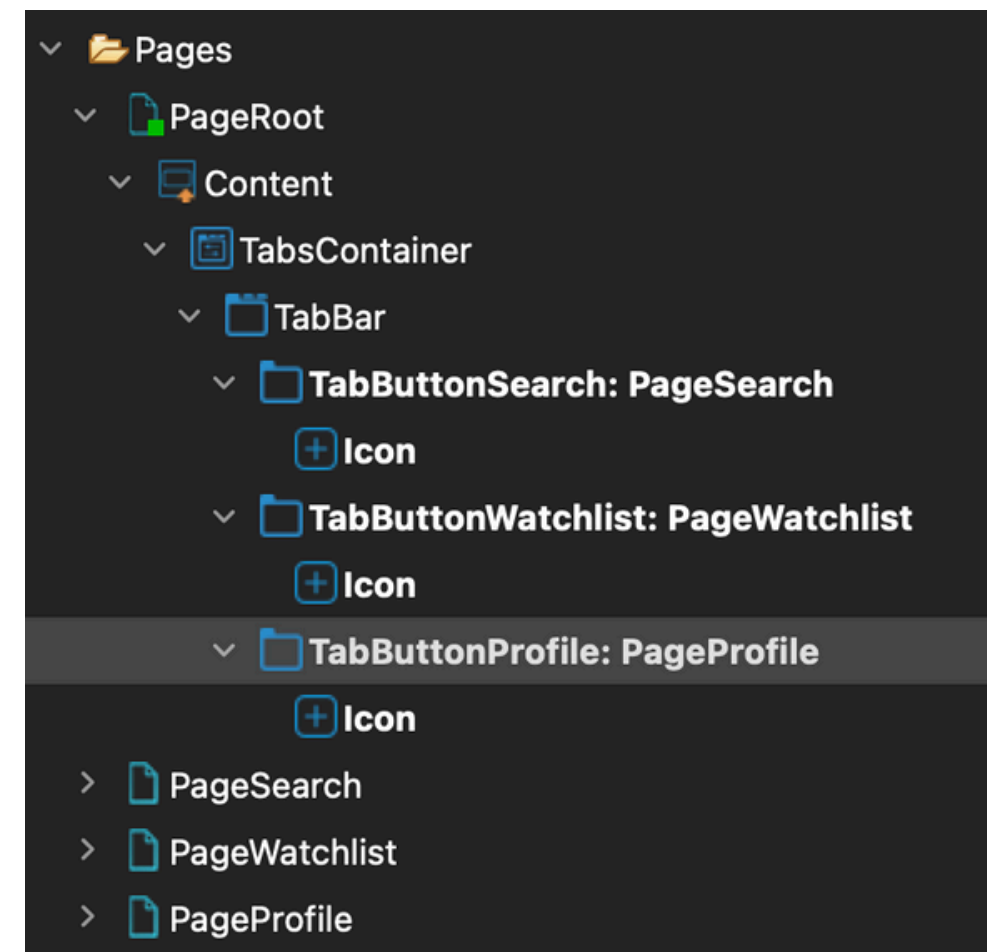
Do the same thing for **TabButtonWatchlist** and **TabButtonProfile**.



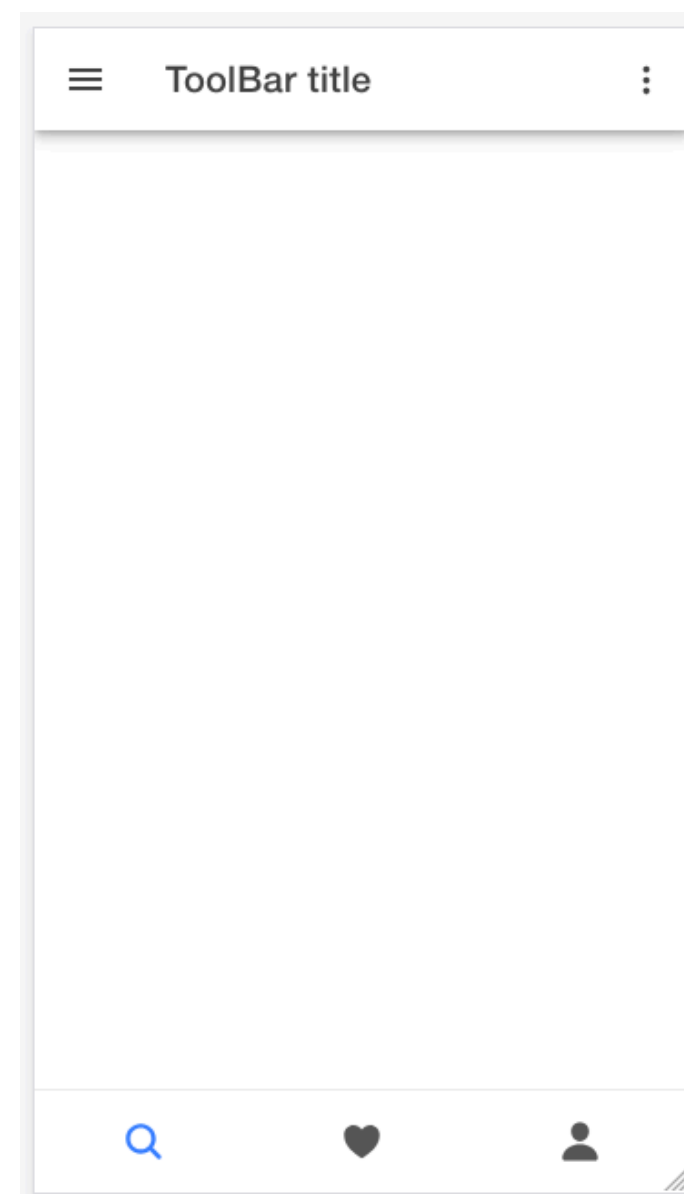
3.2 Navigation with tabs

The **TabButtons** are linked to their respective pages.

When we click on **TabButtonSearch**, we navigate to **PageSearch**.

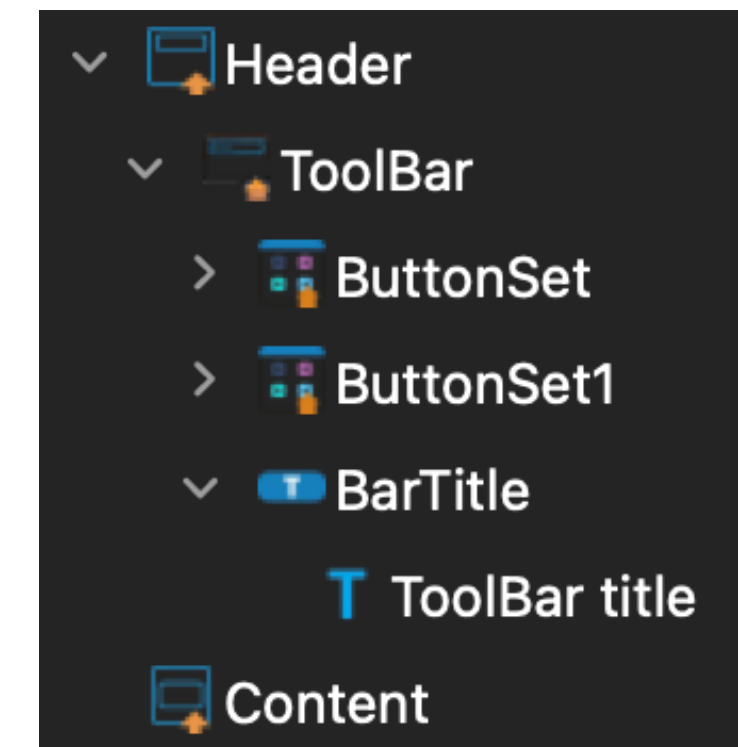


Page Search



The 3 pages have the same content :

a **Header** with a **ToolBar** and an empty **Content**.



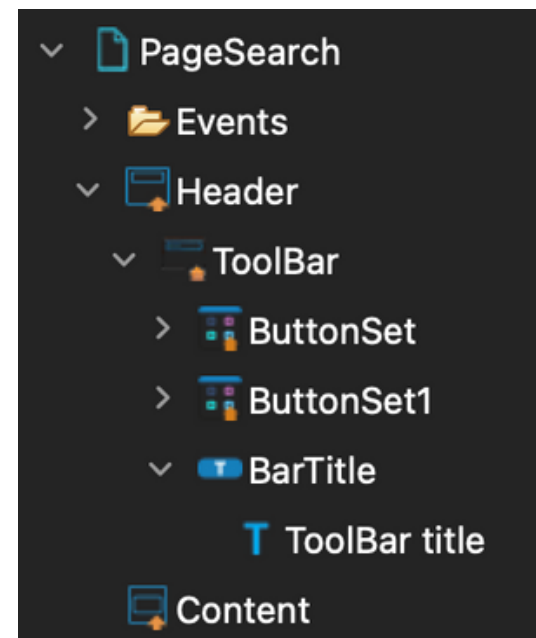
Now, we need to change the title of each page, to distinguish them, and to show that we are indeed navigating.



3.2 Navigation with tabs

Let's rename the **ToolBar title** in the **Header** of each page.

In **PageSearch**,
click on the **text element** in **BarTitle** to display its properties.



Properties X	
Property	Value
Base properties	
Comment	
I18n translate	false
Is active	true
Text value	ToolBar title

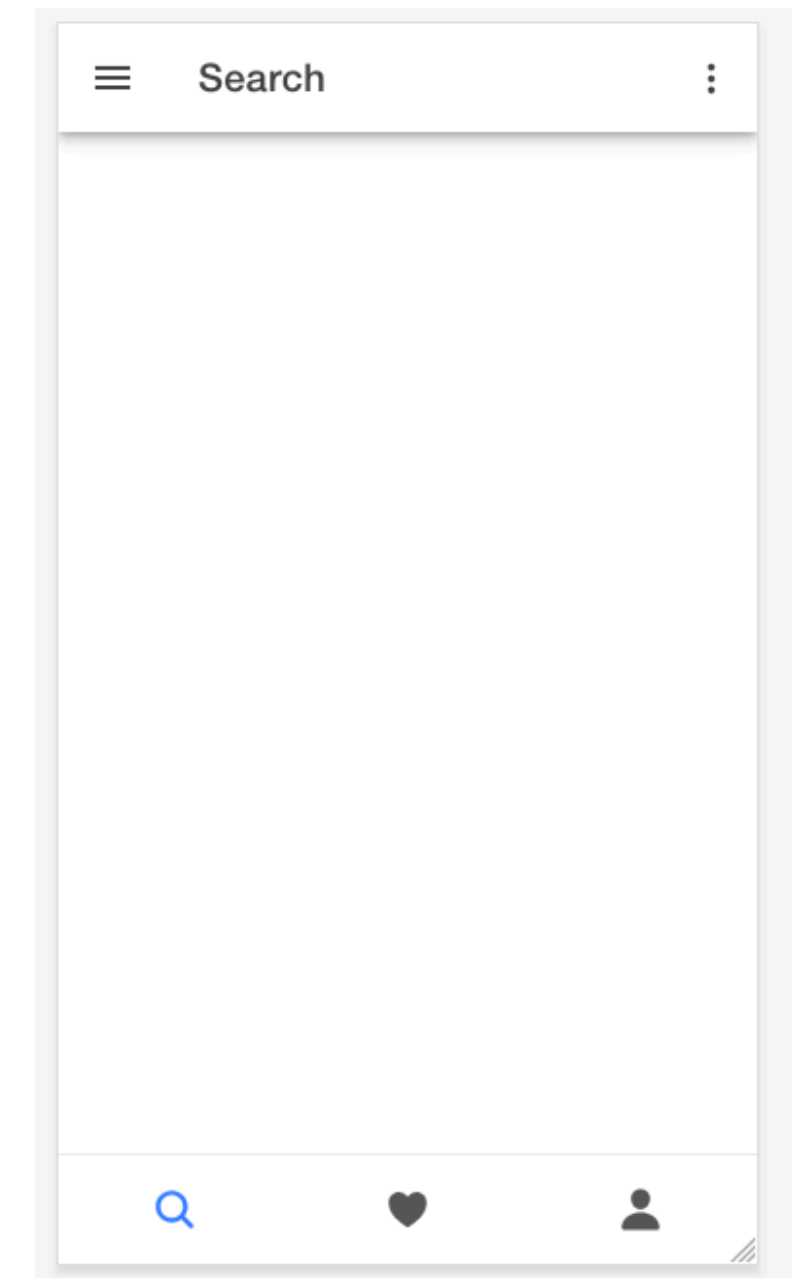
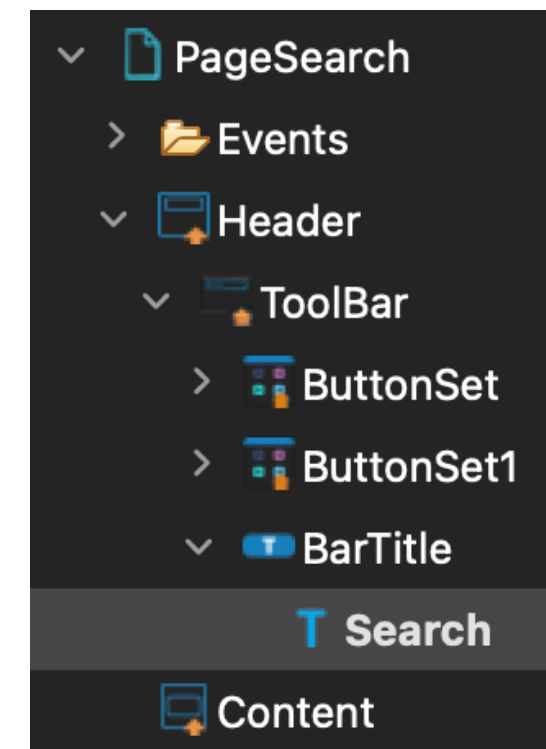


Then change its **Text value**
from **ToolBar title** to **Search**.

Properties X	
Property	Value
Base properties	
Comment	
I18n translate	false
Is active	true
Text value	Search



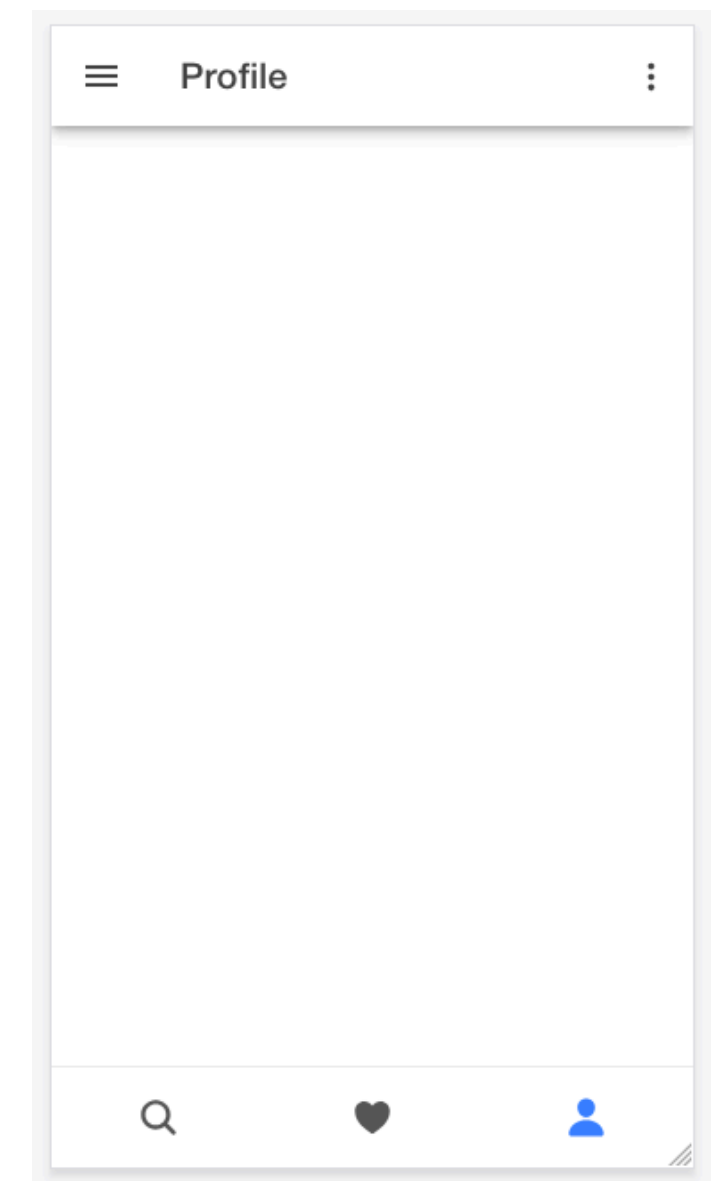
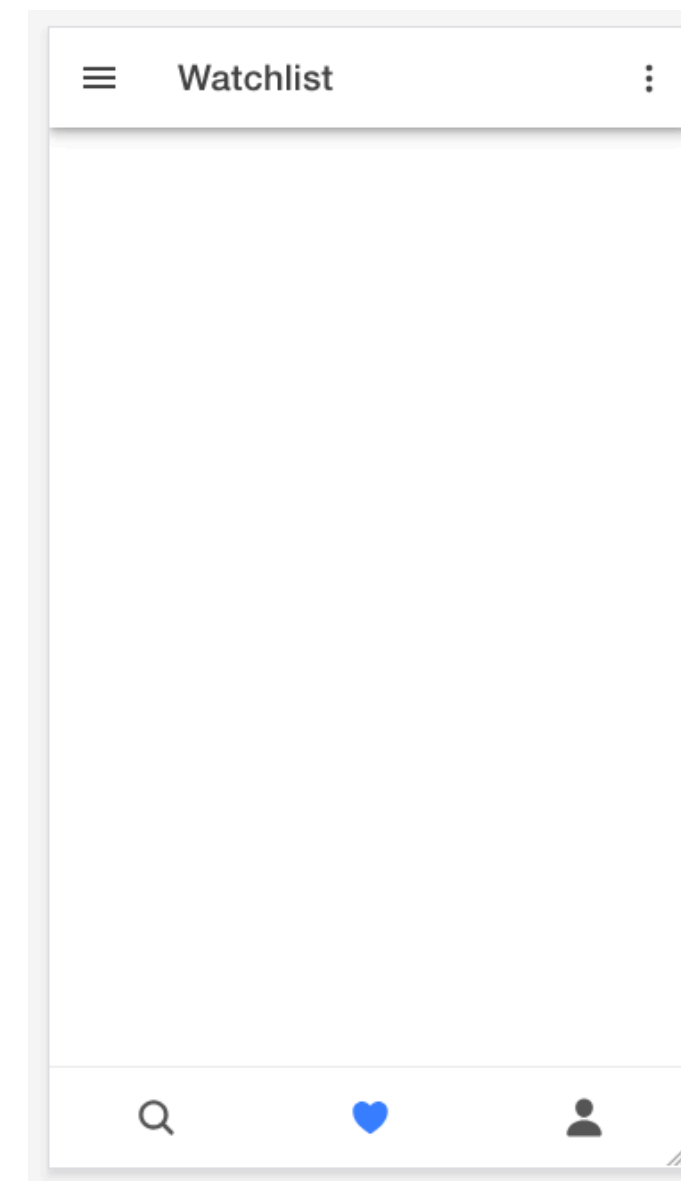
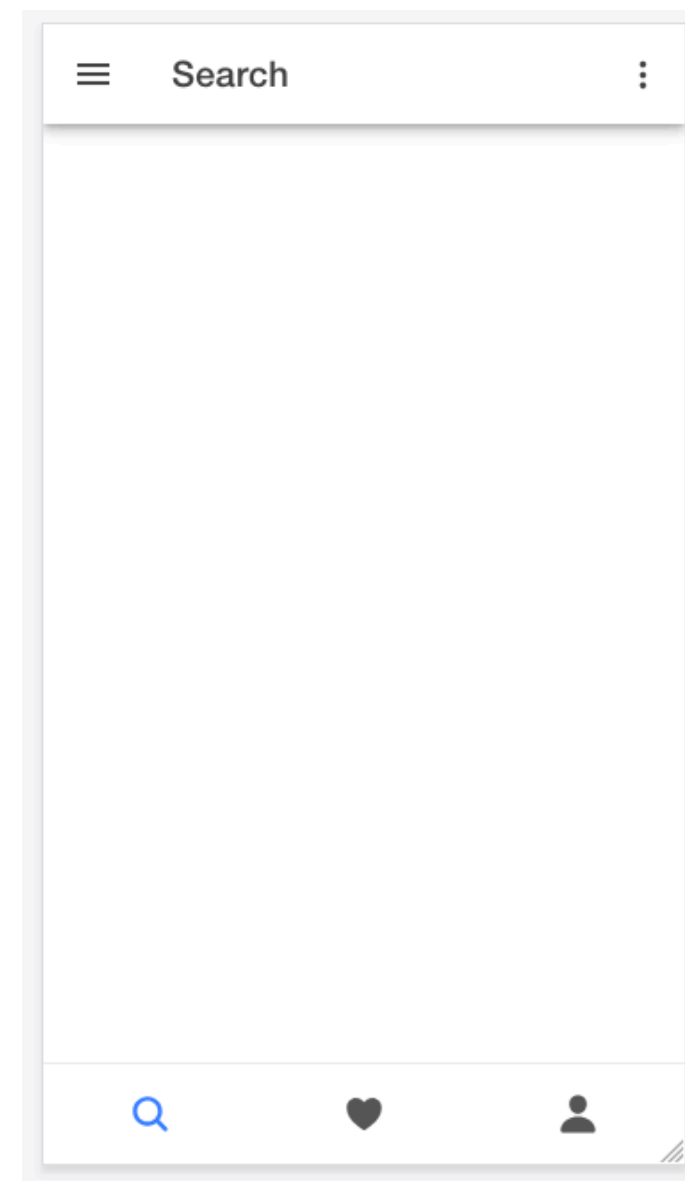
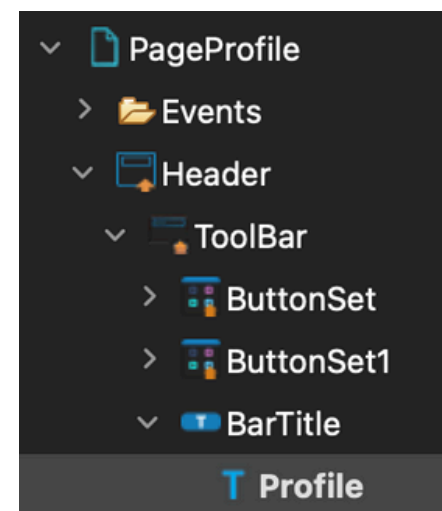
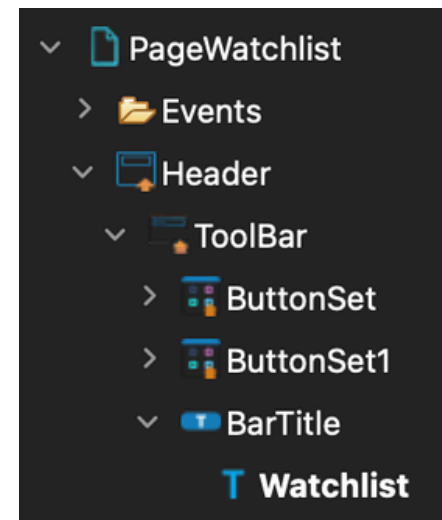
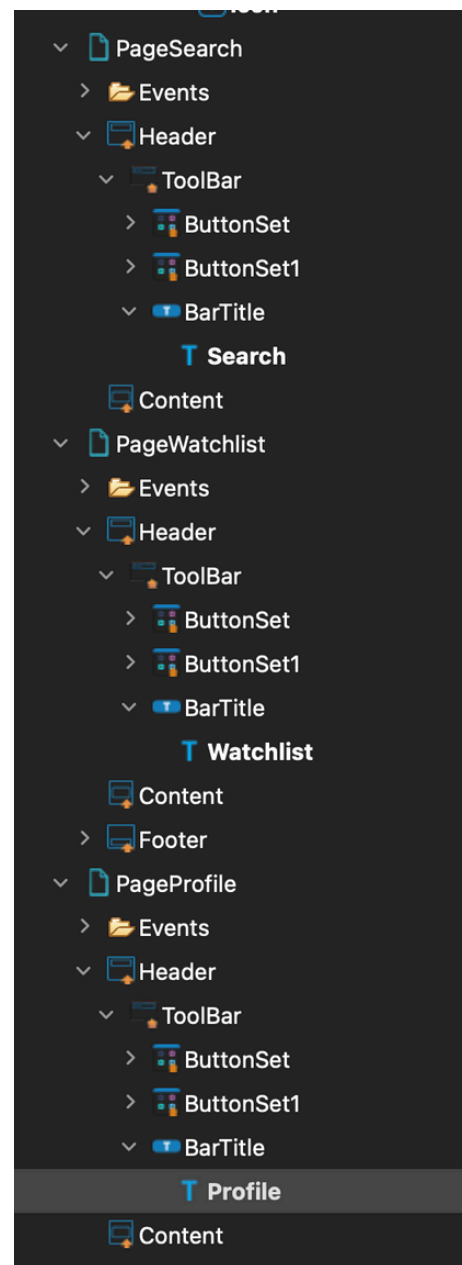
The title appears now as **Search** in the page.



3.2 Navigation with tabs

Do the same thing with **PageWatchlist** and **Watchlist**, and **PageProfile** and **Profile**.

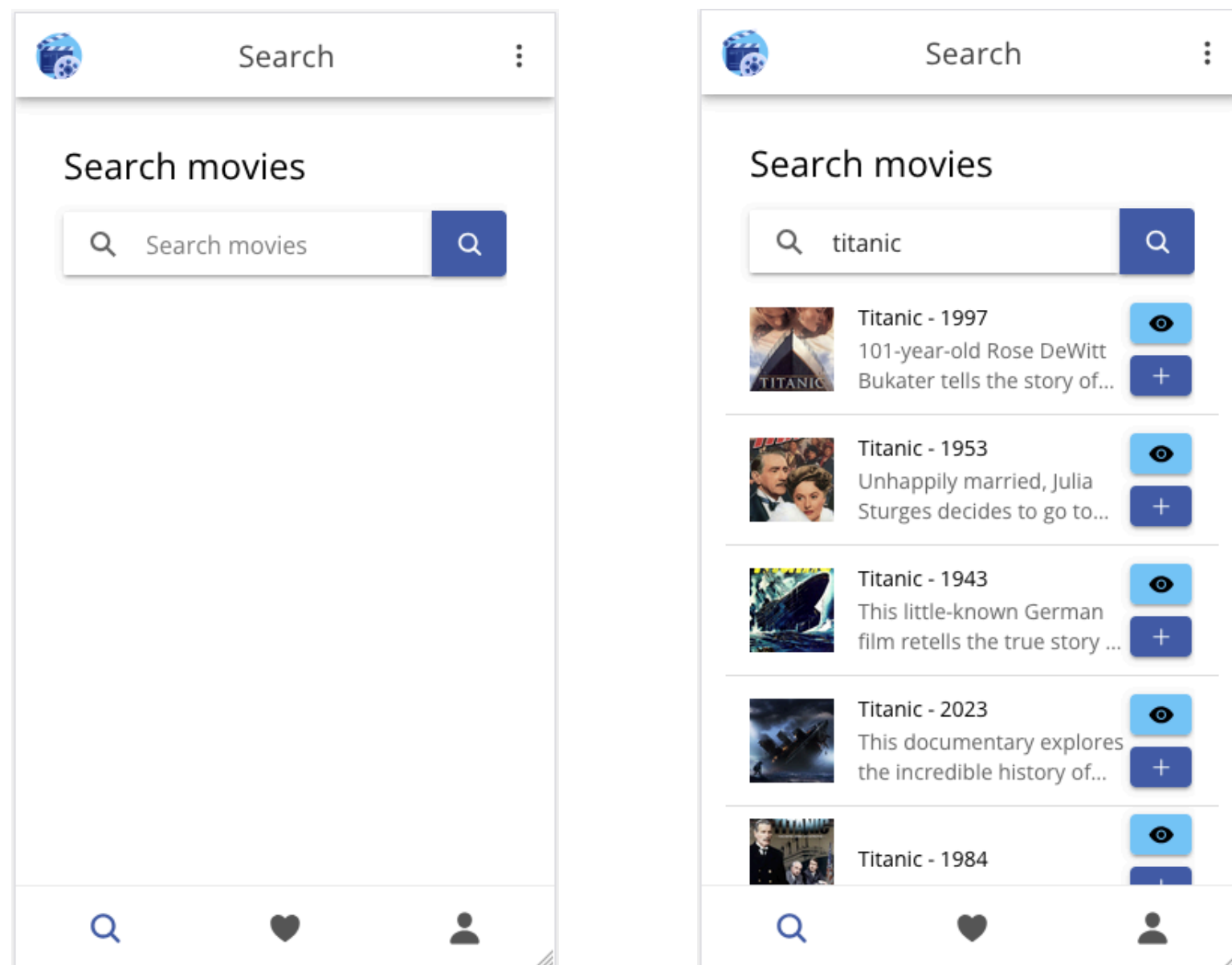
Our navigation is now ready !



3.3 List & List items

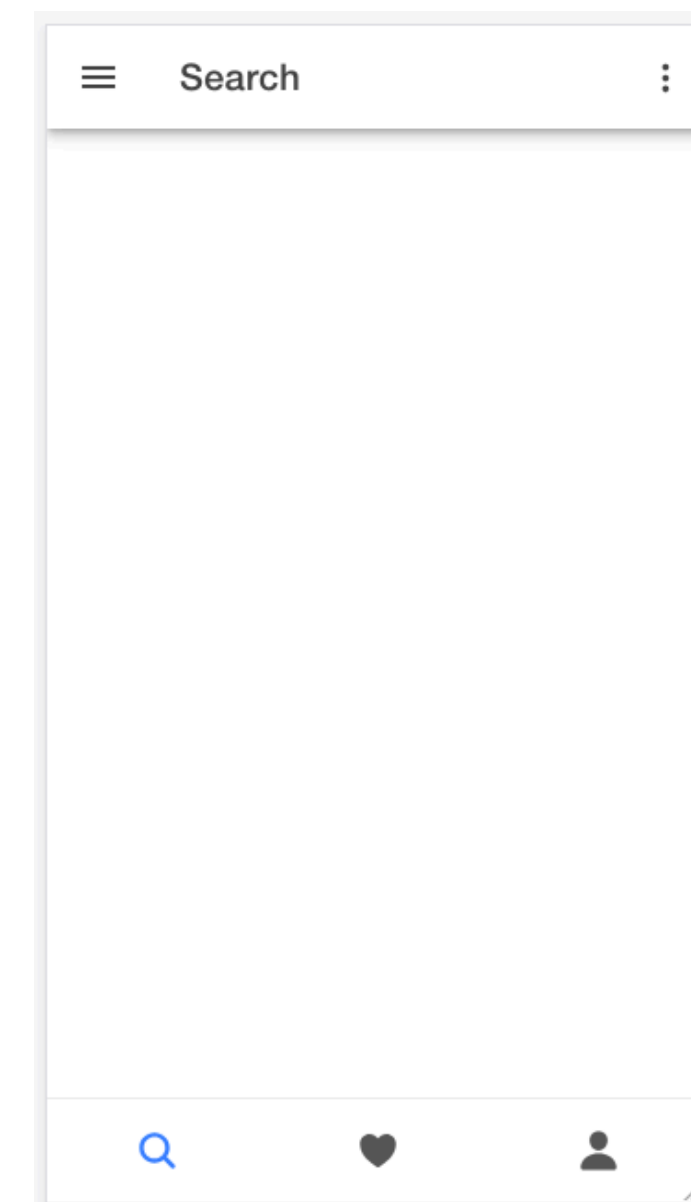
Let's create the layout of **PageSearch**.

In this page, we want to display a searchBar and a list of results, similar to the example below.

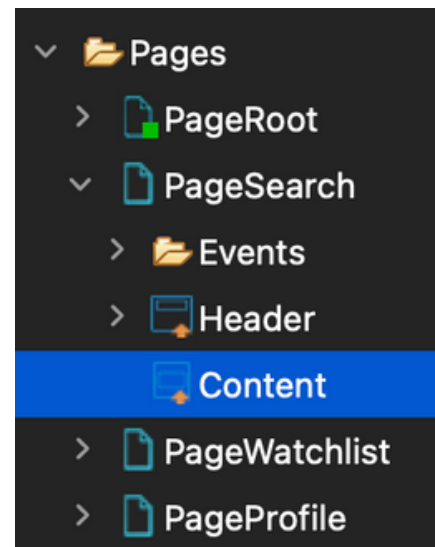


As for now,

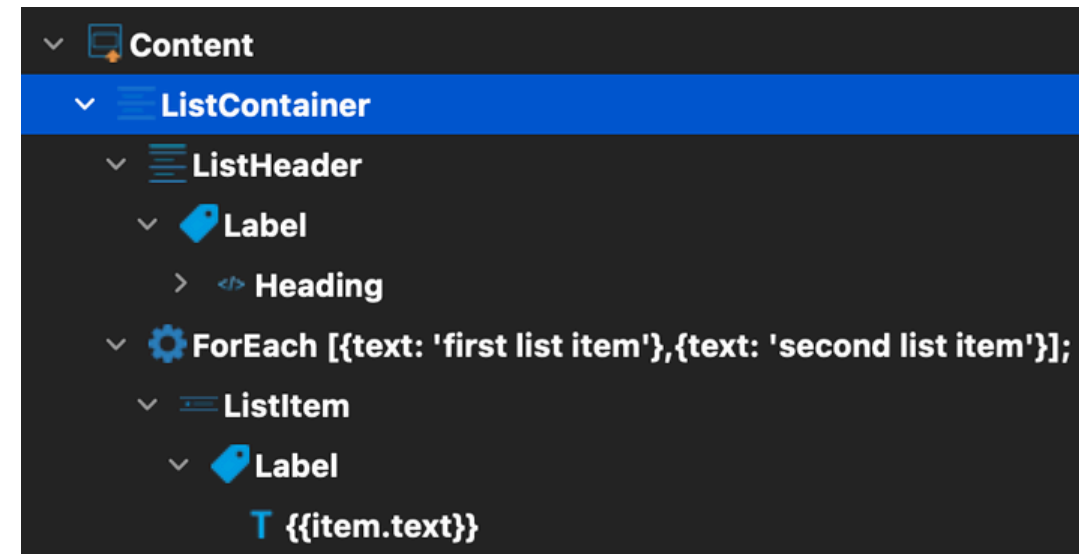
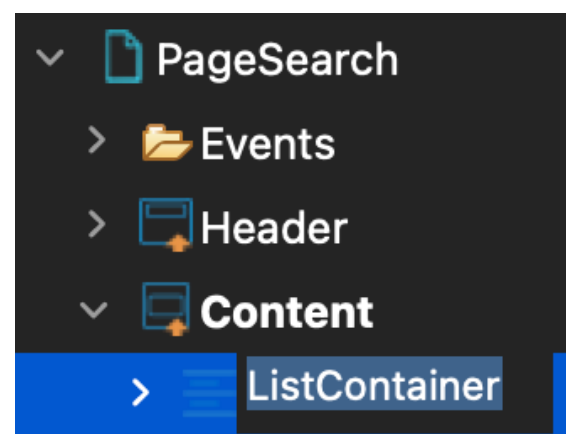
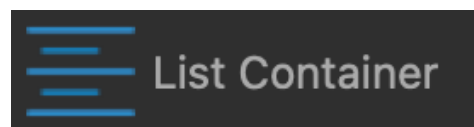
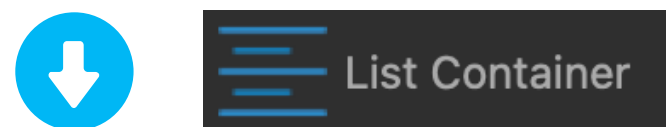
the **Content** of **PageSearch** is empty and it appears as such.



3.3 List & List items



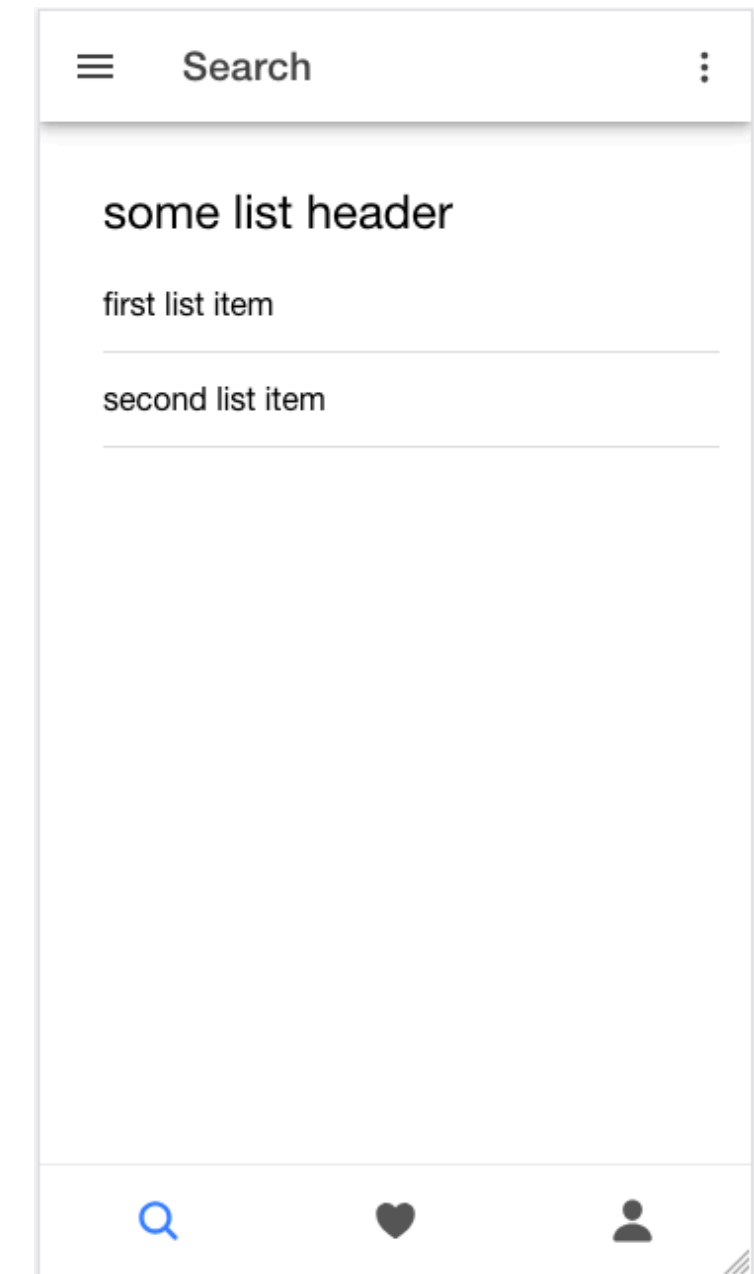
Let's add a **ListContainer** component inside the **Content** of **PageSearch**.



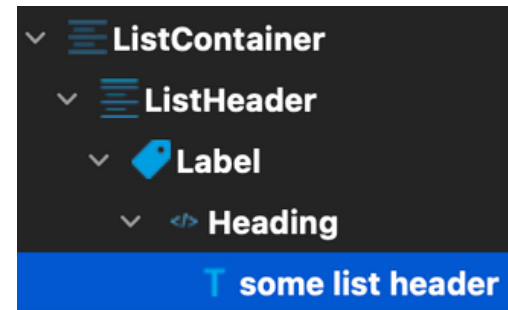
By default,
the **ListContainer** is created with
other components inside:

- a **ListHeader**
- a **ForEach directive** containing a **ListItem**.

As for now, let's focus on the layout,
not on the directive.



3.3 List & List items

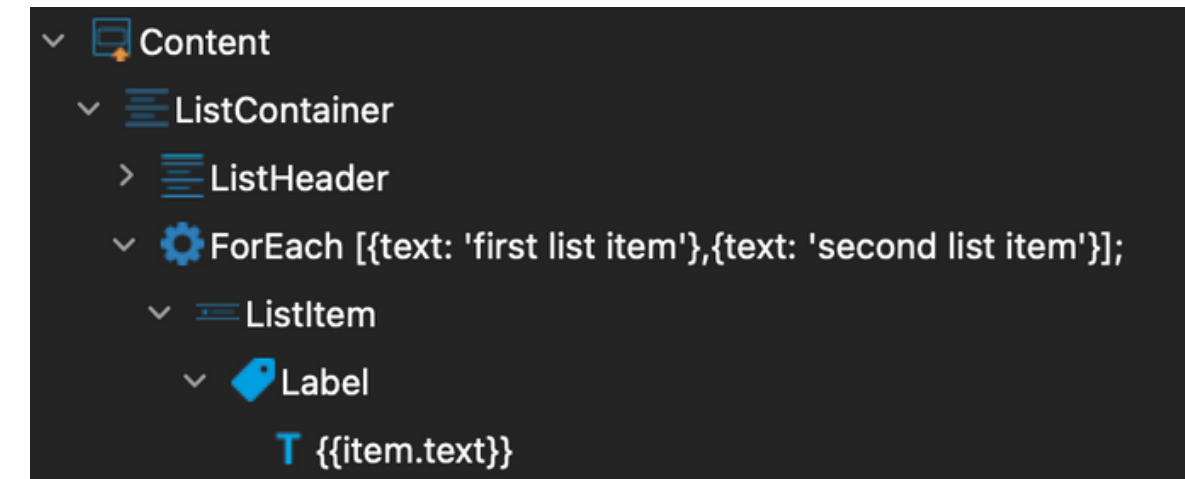
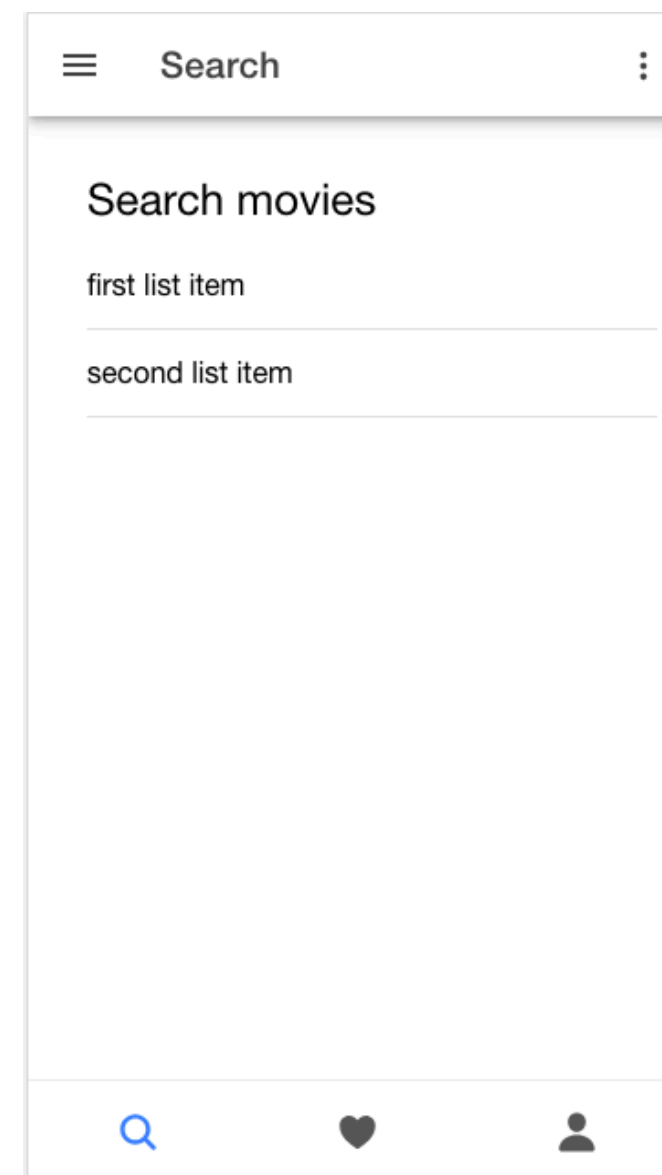
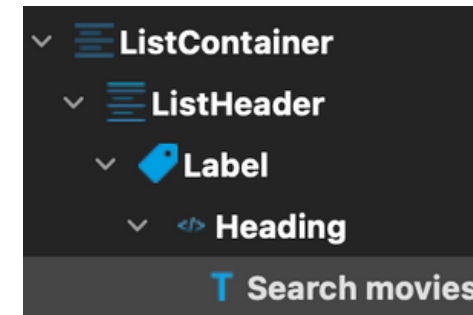


Properties	
Property	Value
Base properties	
Comment	
l18n translate	false
Is active	true
Text value	some list header

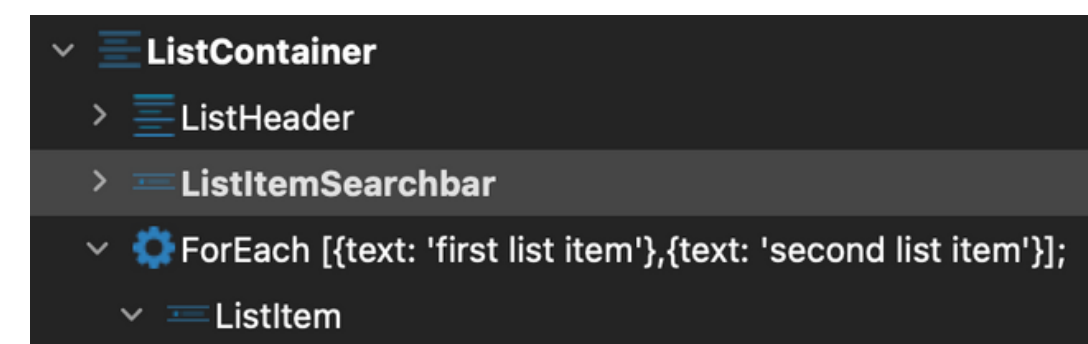
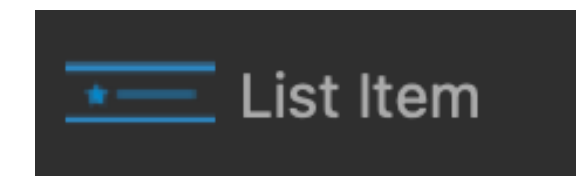


Rename the **Text** element
in the **Heading** of **ListHeader**
as **Search Movies**.

Base properties	
Comment	
l18n translate	false
Is active	true
Text value	Search movies



Add a **ListItem** below **ListHeader**,
and name it **ListItemSearchbar**.



3.3 List & List items

```
▼ ListContainer
  > ListHeader
  > ListItemSearchbar
  ▼ ⚙️ ForEach [{text: 'first list item'}, {text: 'second list item'}];
    ▼ ListItem
```



In **ListItemSearchbar**, remove the **Label**.

```
▼ ListContainer
  > ListHeader
  ▼ ListItemSearchbar
    > Label
```



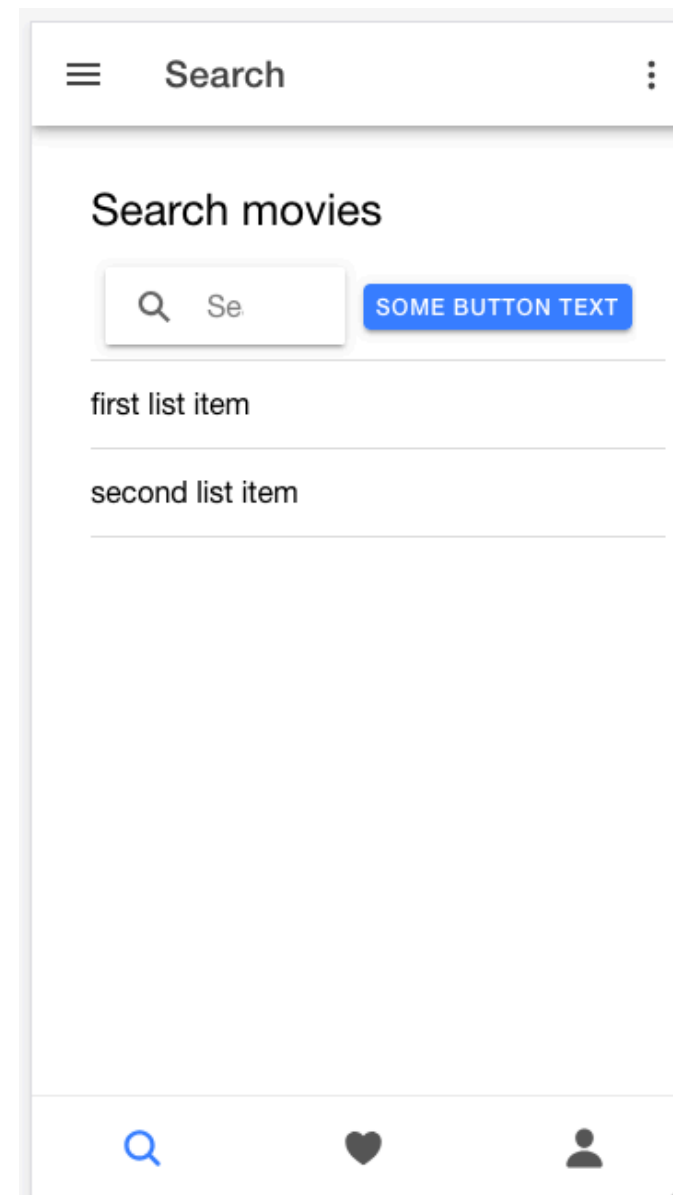
And add 2 components:
SearchBar and **Button**.

```
Search Bar
```

```
Button
```



```
▼ ListContainer
  > ListHeader
  ▼ ListItemSearchbar
    > SearchBar
    > Button
```



```
▼ ListItemSearchbar
  > SearchBar
  ▼ Button
    > Controls
    ▼ Label
      T some button text
```



Inside the **Button**,
remove the **Label**,
and add an **Icon**.

```
▼ ListItemSearchbar
  > SearchBar
  ▼ Button
    > Controls
    > Icon
```



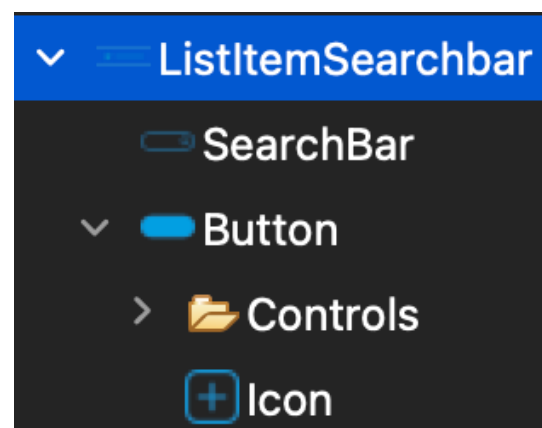
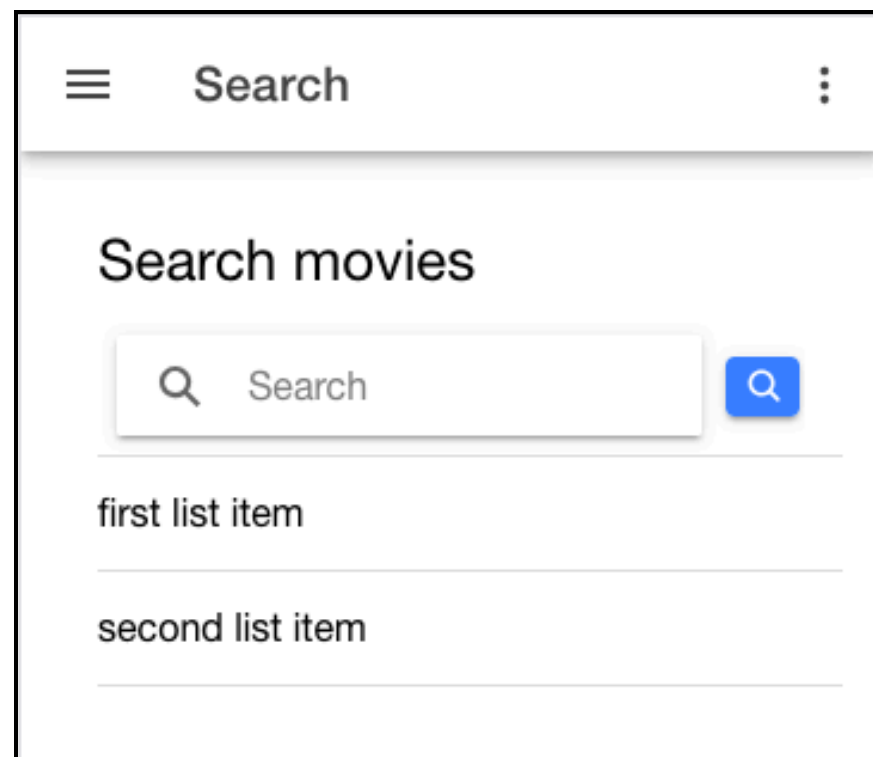
Change the Icon name
to 'search'.

```
Icon name 'search'
```

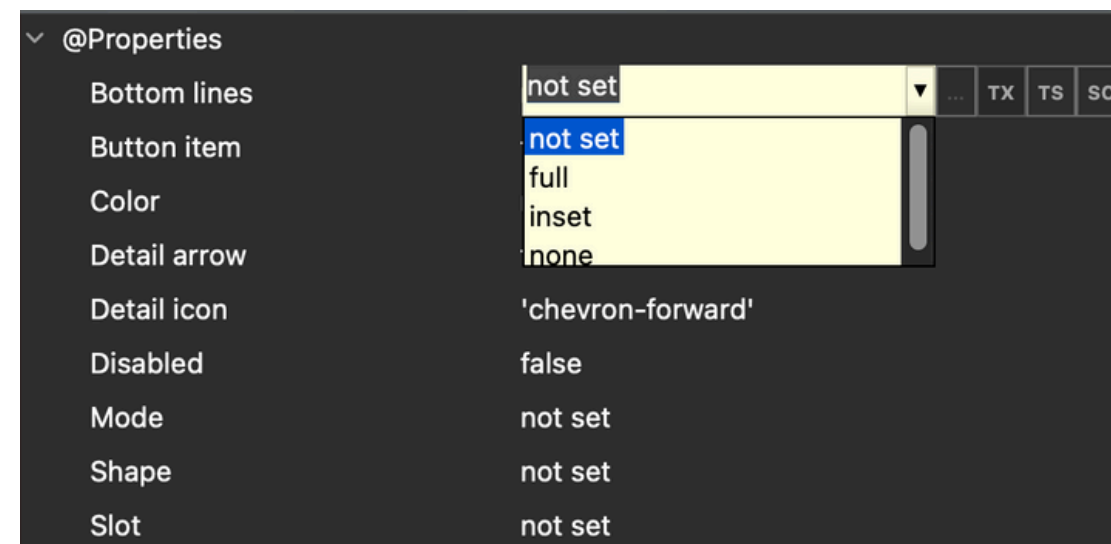


3.3 List & List items

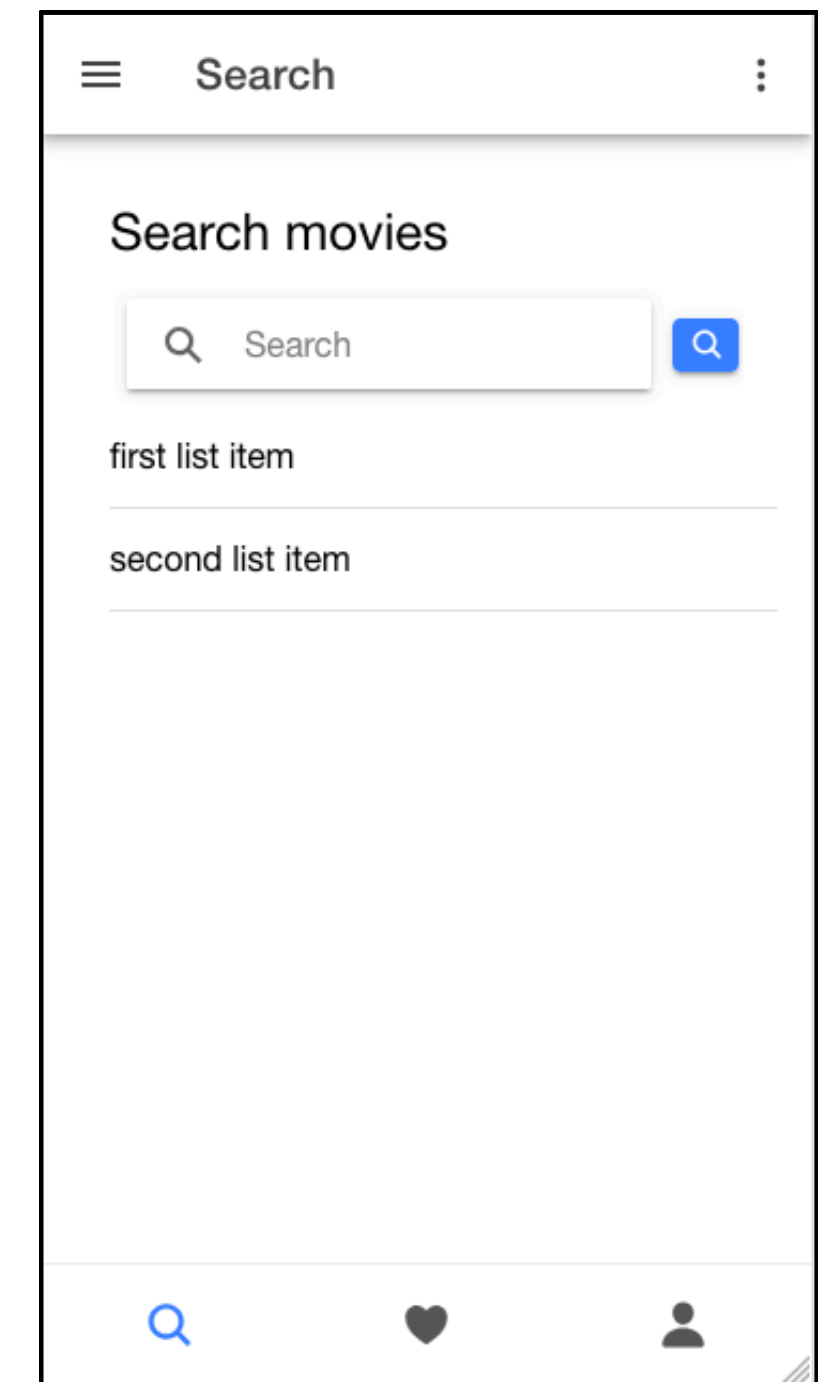
Let's remove the bottom line
at the bottom of **ListItemSearchbar**.



In **ListItemSearchbar**'s properties,
click on the select of the value
of the **Bottom Line** property.

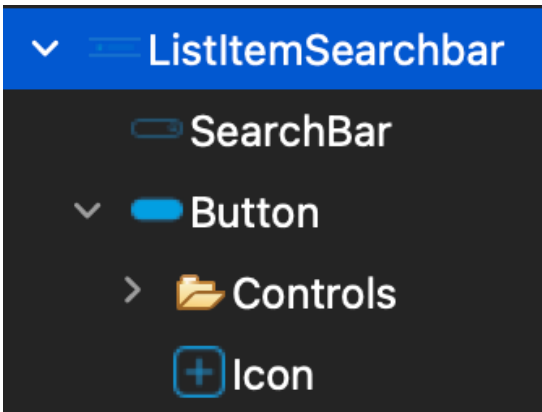
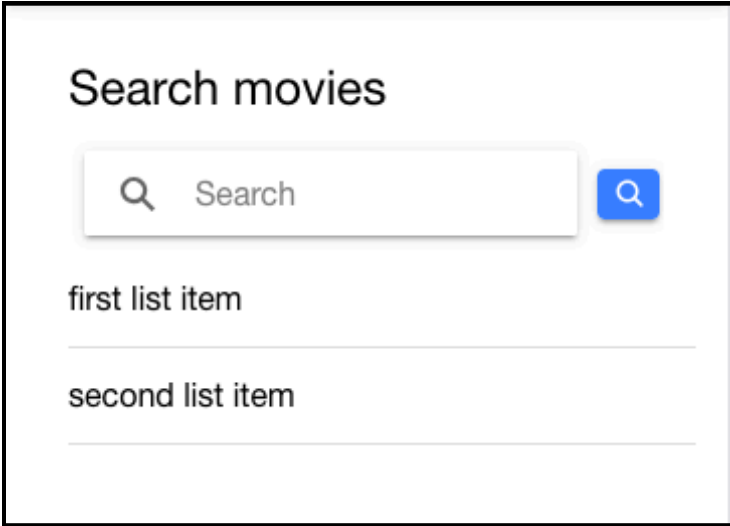


Select **none** to remove the bottom line.

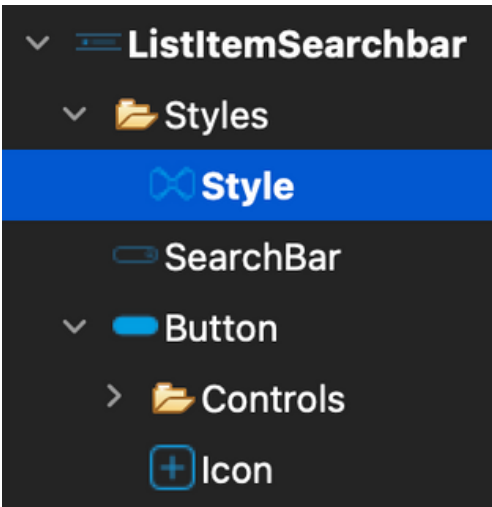


3.3 List & List items

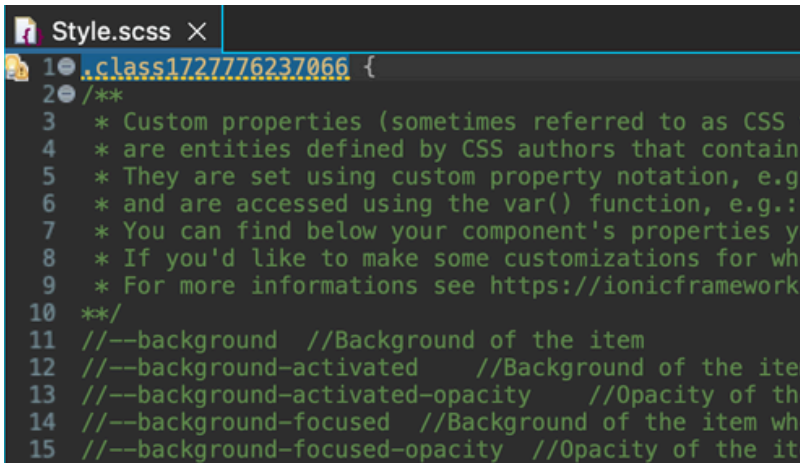
Let's add some styles to align the searchbar with the title and remove the space between the searchbar and the button.



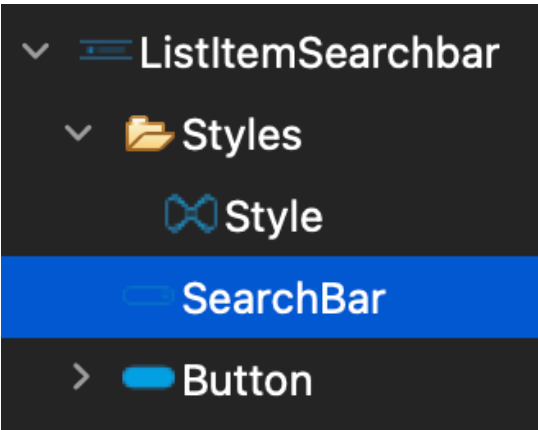
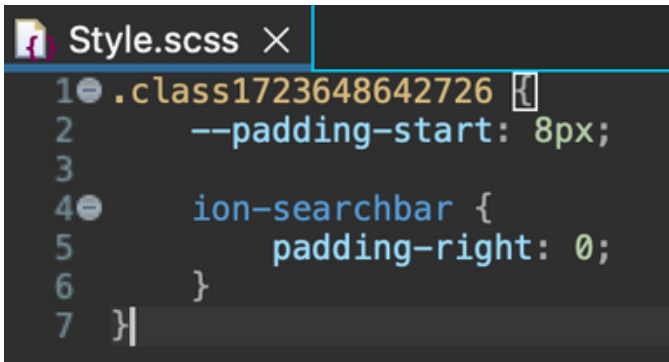
In **ListItemSearchbar**, add a **Style** component.



Click on the style component to open the corresponding **Style.scss** file.



In **Style.scss**, inside the **.class**, replace the commented styles by this css/scss.

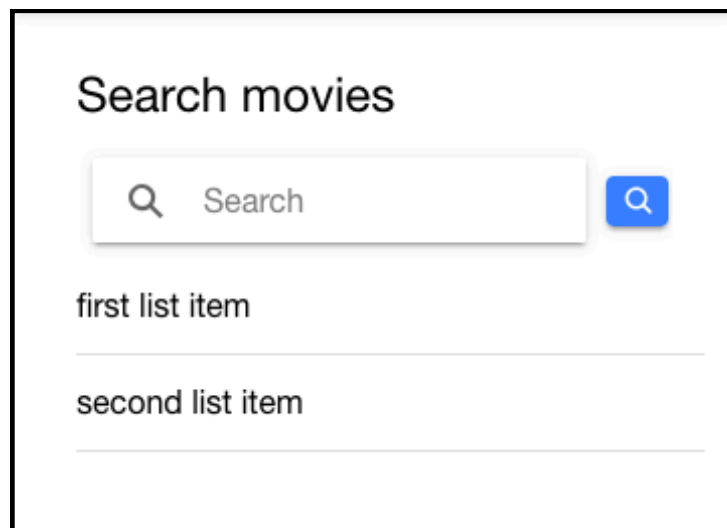


ion-searchbar is the **selector** for the SearchBar in **scss**. In **SearchBar's** **Base** properties, it is the **value** of **Tag name**.

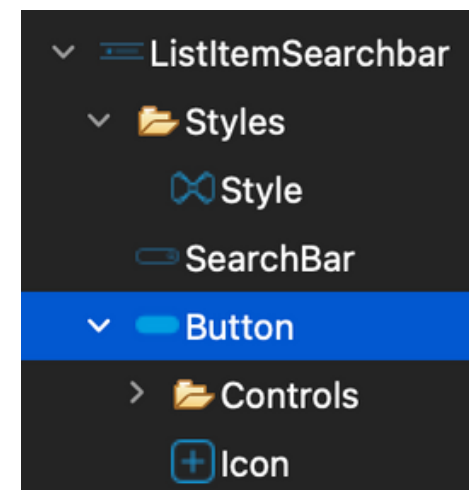
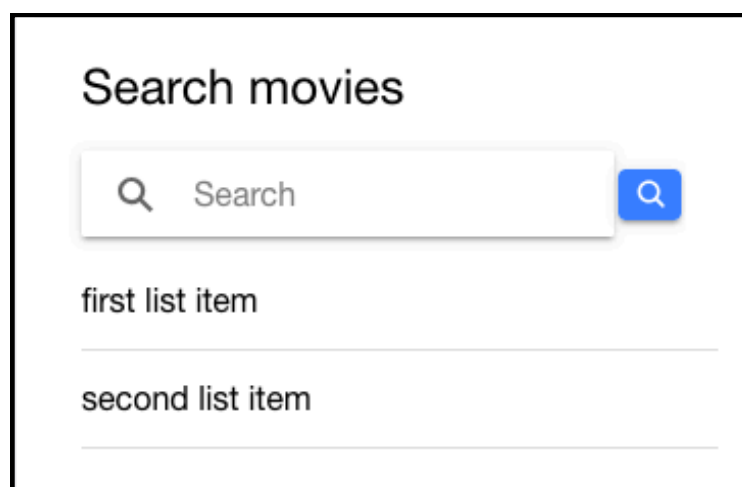
Base properties	
Comment	
Is active	true
Tag name	ion-searchbar



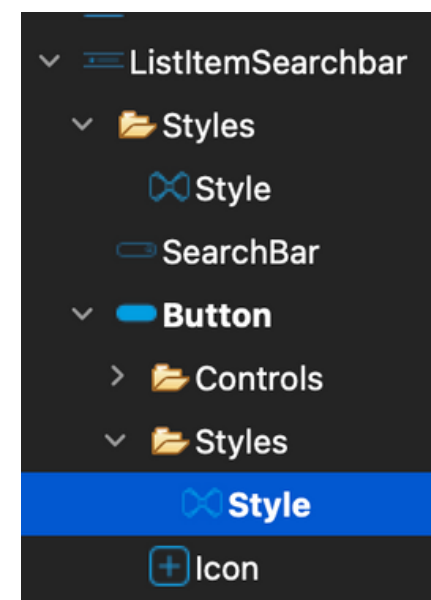
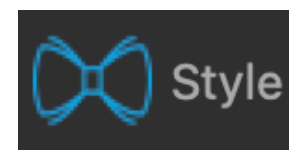
3.3 List & List items



Now that the **SearchBar** is aligned with the title and the space removed, let's style the **Button**.



In **Button**, add a **Style** component.



Click on the style component to open the corresponding **Style.scss** file.

```
Style.scss x
1 .class1727776328566
2 /*
3  * Custom properties (sometimes referred to as CSS v
4  * are entities defined by CSS authors that contain
5  * They are set using custom property notation, e.g.
6  * and are accessed using the var() function, e.g.:
7  * You can find below your component's properties yo
8  * If you'd like to make some customizations for who
9  * For more informations see https://ionicframework.
10 */
11 //--background //Background of the button
12 //--background-activated //Background of the butt
13 //--background-activated-opacity //Opacity of the
```

In **Style.scss**, inside the **.class**, replace the commented styles by this css/scss.

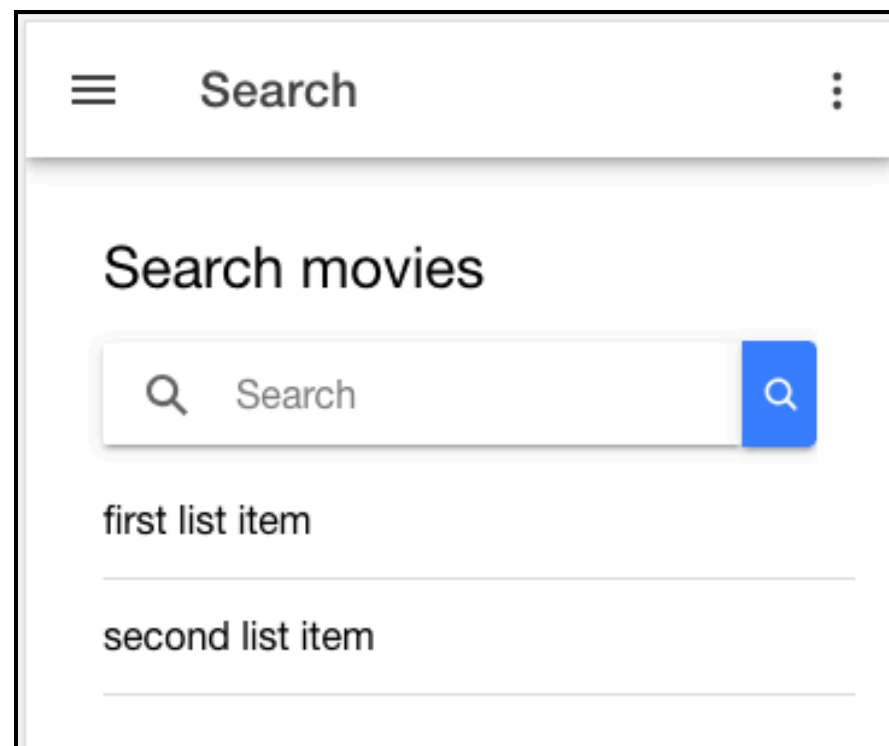


```
Style.scss x
1 .class1727776328566
2   margin: 0;
3   height: 44px;
4   --border-radius: 0 4px 4px 0;
5 }
```

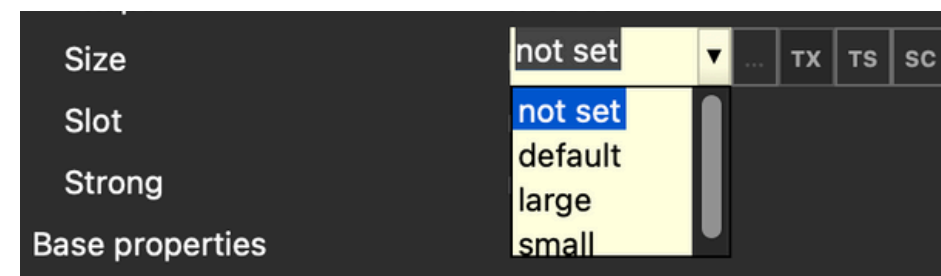
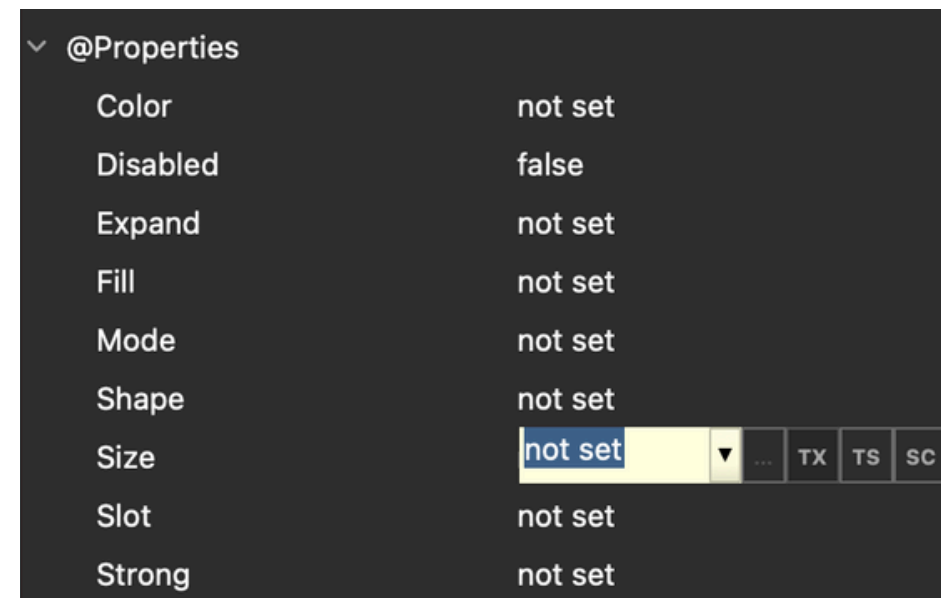


3.3 List & List items

Now, the **Button** has the same height as the **SearchBar**.



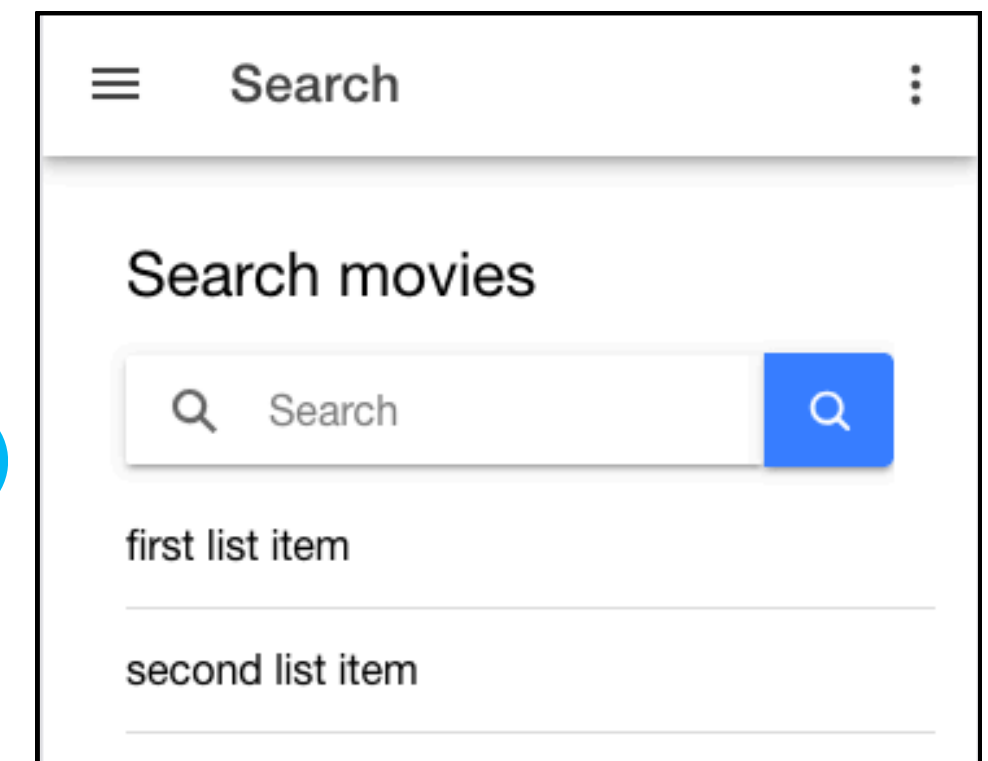
But the **Button**'s width is a little small.



Let's change the **Button's Size** property to **default**.

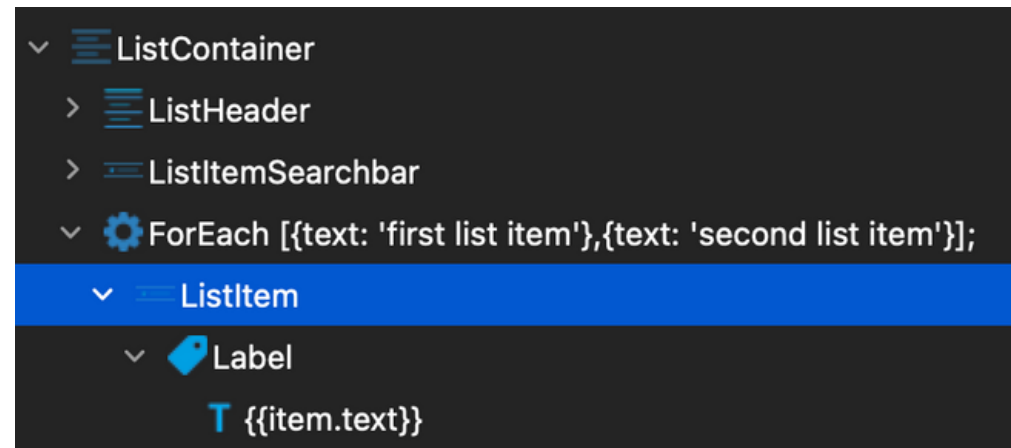


Our **SearchBar**'s style is ready.



3.3 List & List items

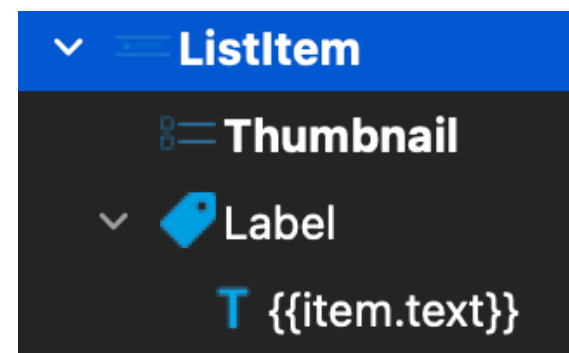
Let's make some changes in the **ListItem**.



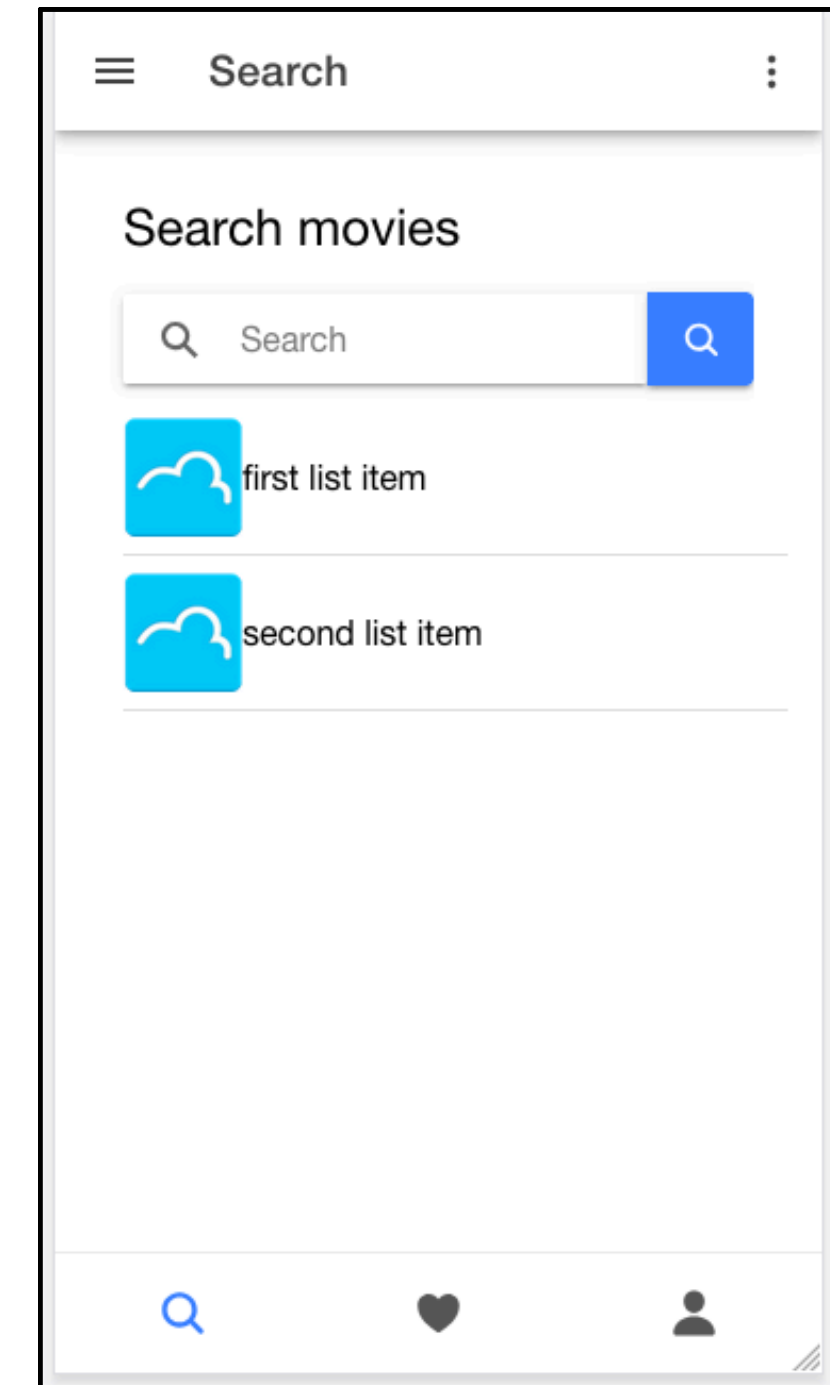
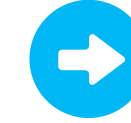
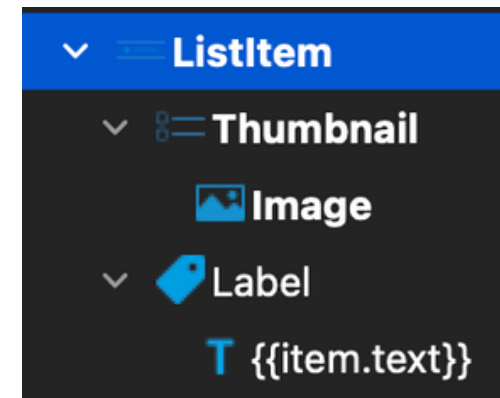
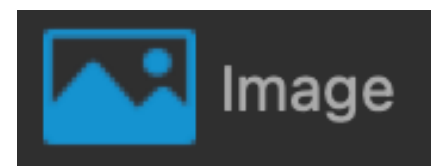
Add a **Thumbnail** item
in the **ListItem**.



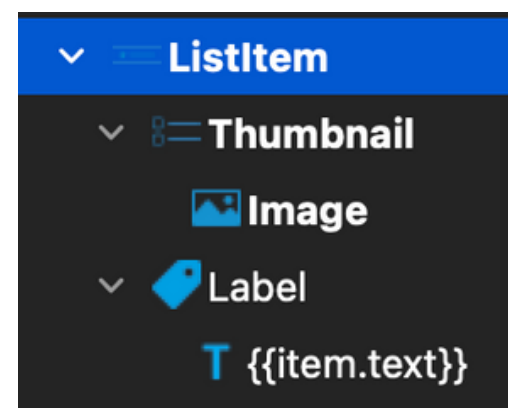
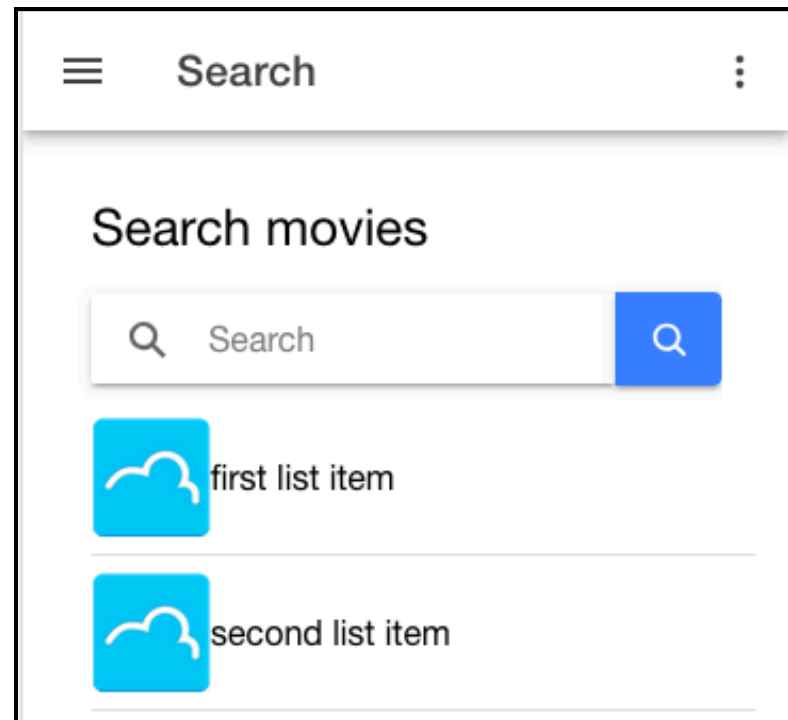
Thumbnail Item



And add an **Image**
in the **Thumbnail** item.

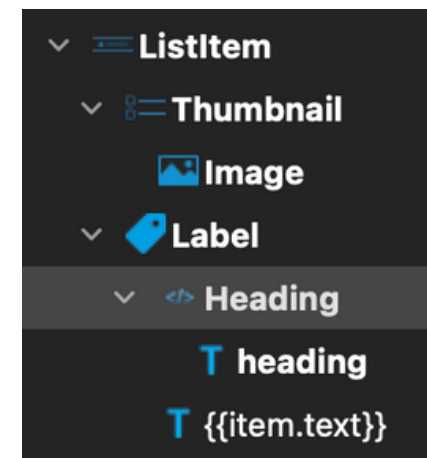


3.3 List & List items

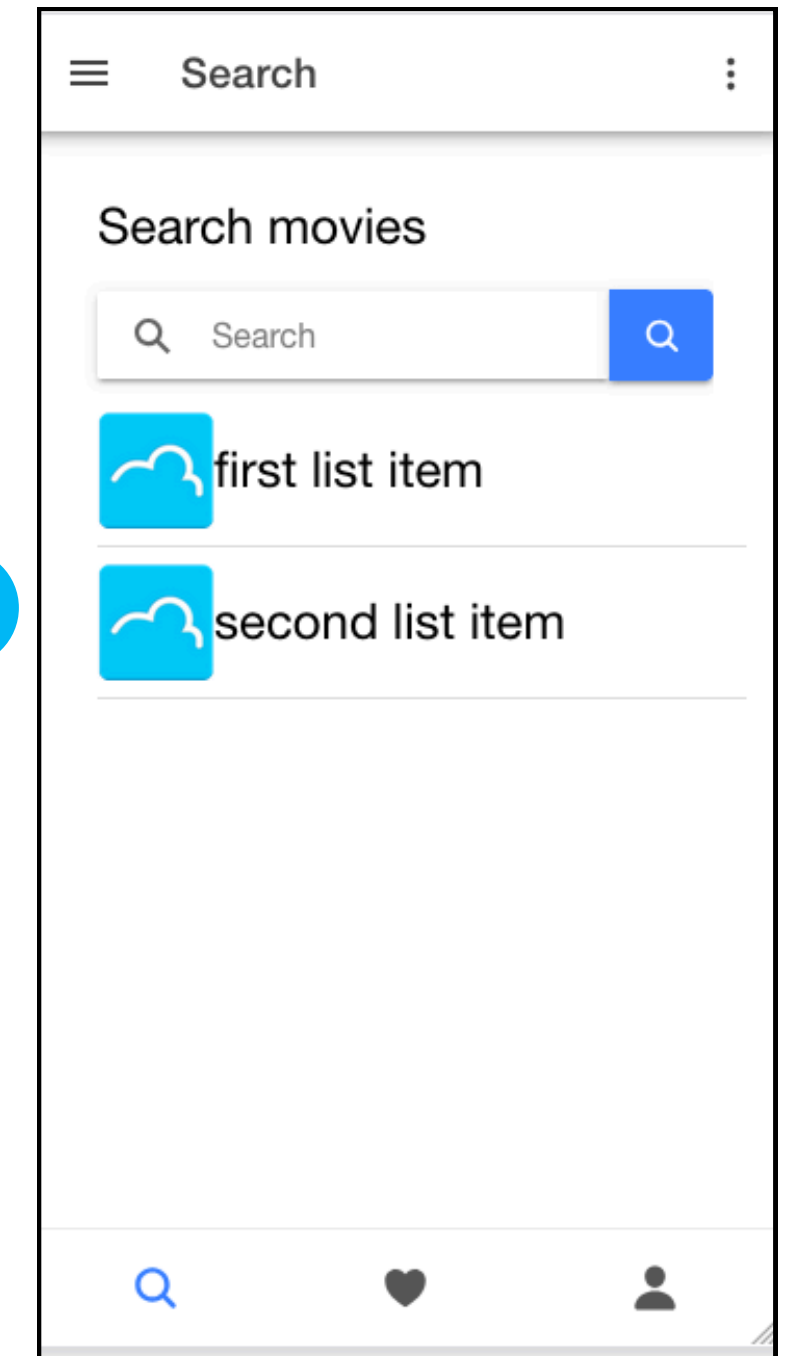
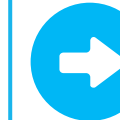
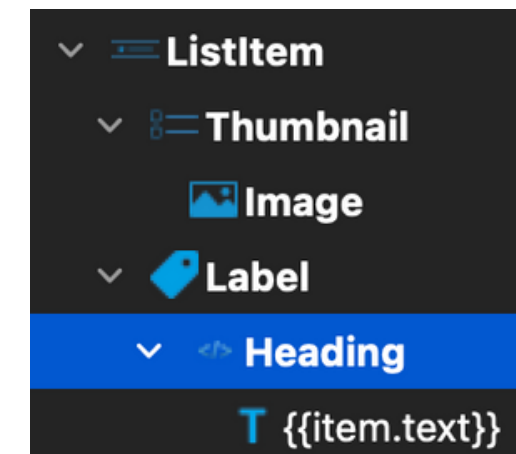


Add a **Heading**
in the **Label**
of the **ListItem**.

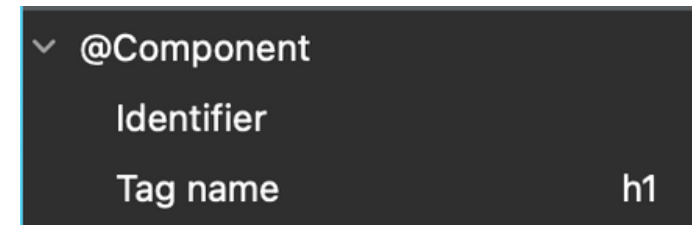
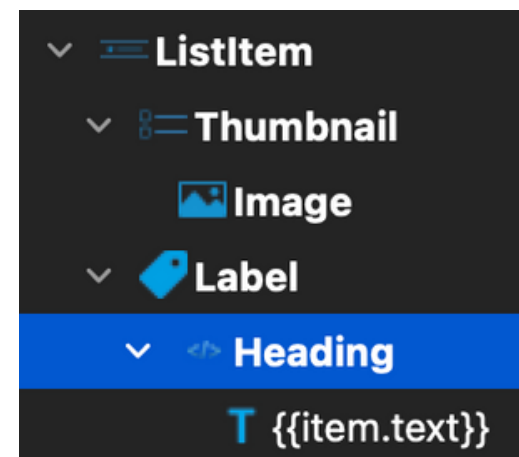
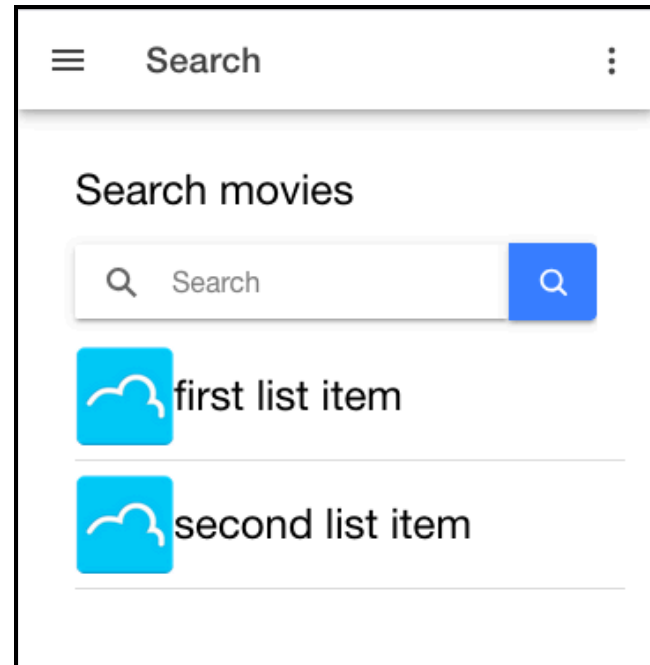
```
</> Heading
```



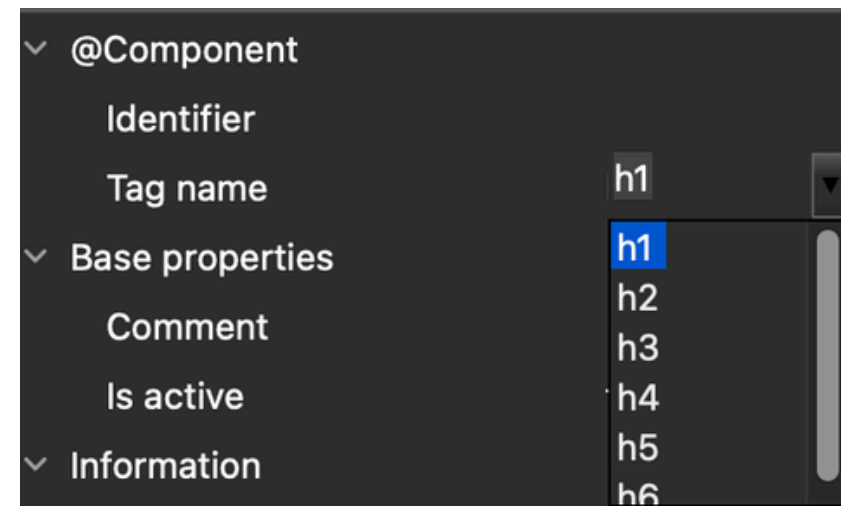
And replace
the **Text element**
of the **Heading**
by the **Text element**
of the **Label**.



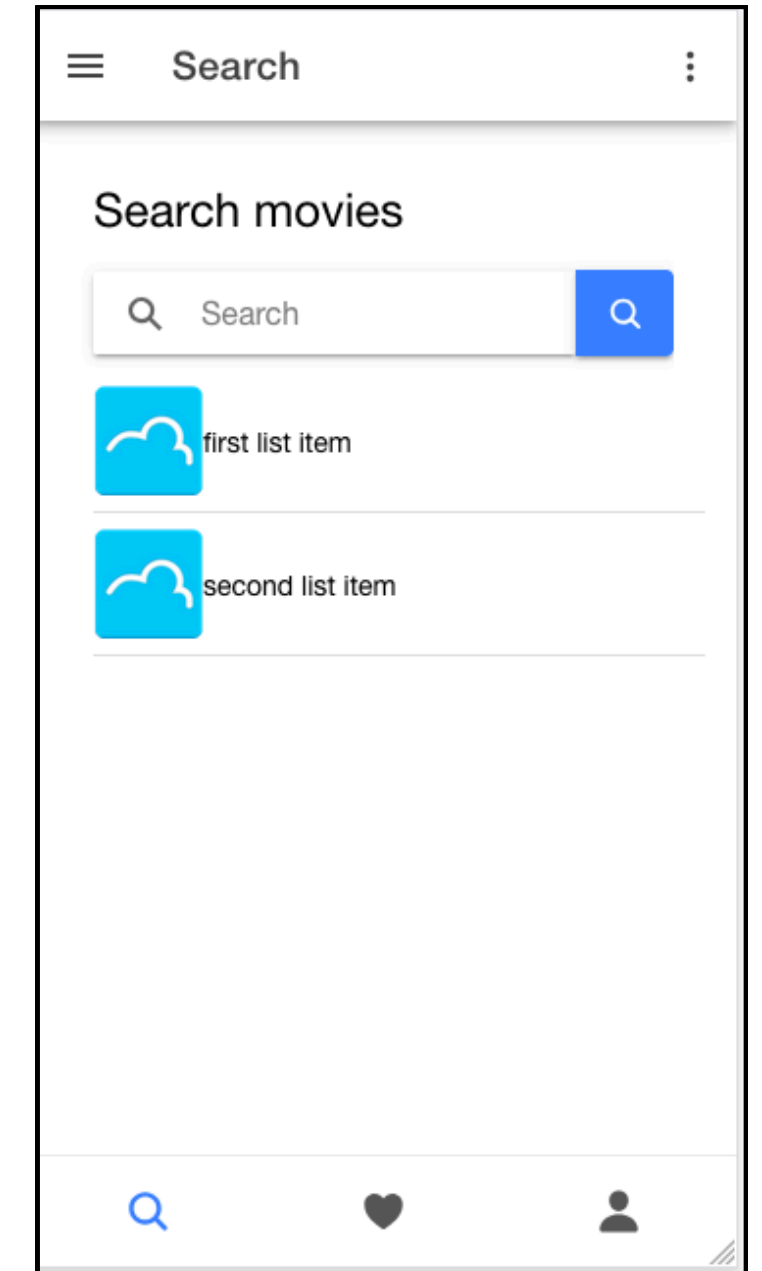
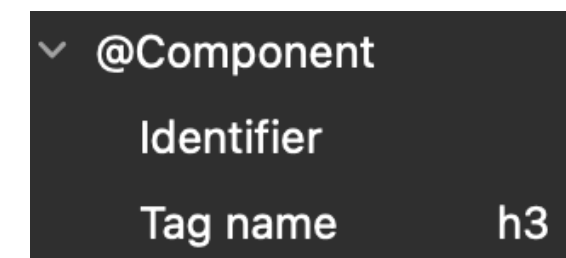
3.3 List & List items



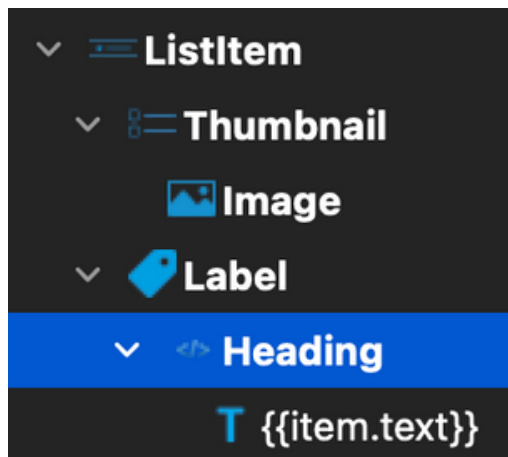
In the **Heading's** properties, click on **h1** to display the select.



And change the **Tag name** from **h1** to **h3**.



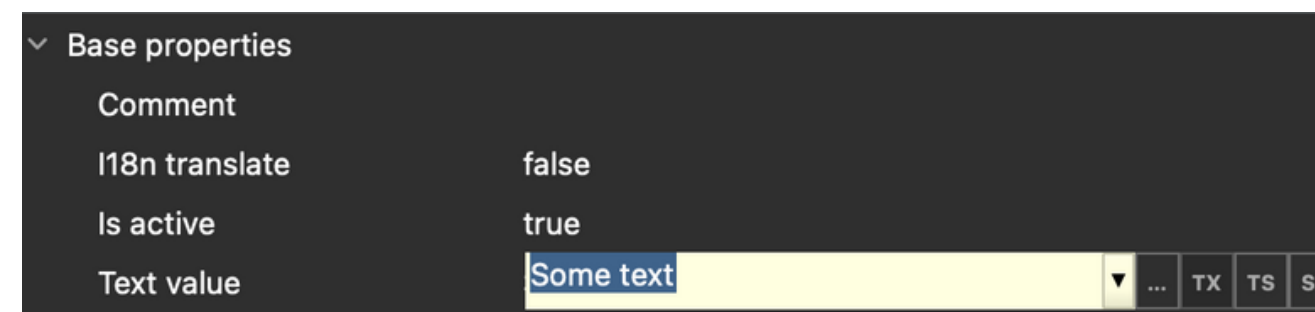
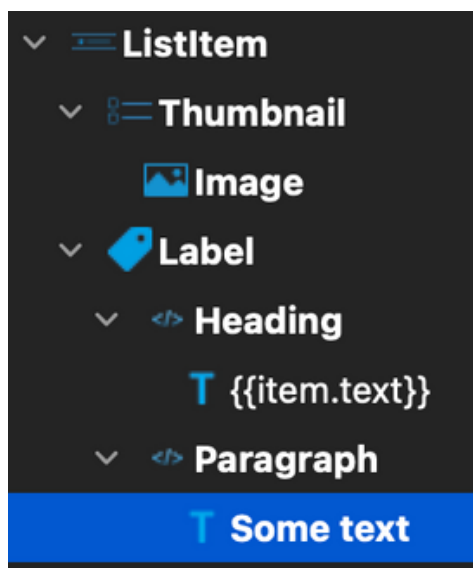
3.3 List & List items



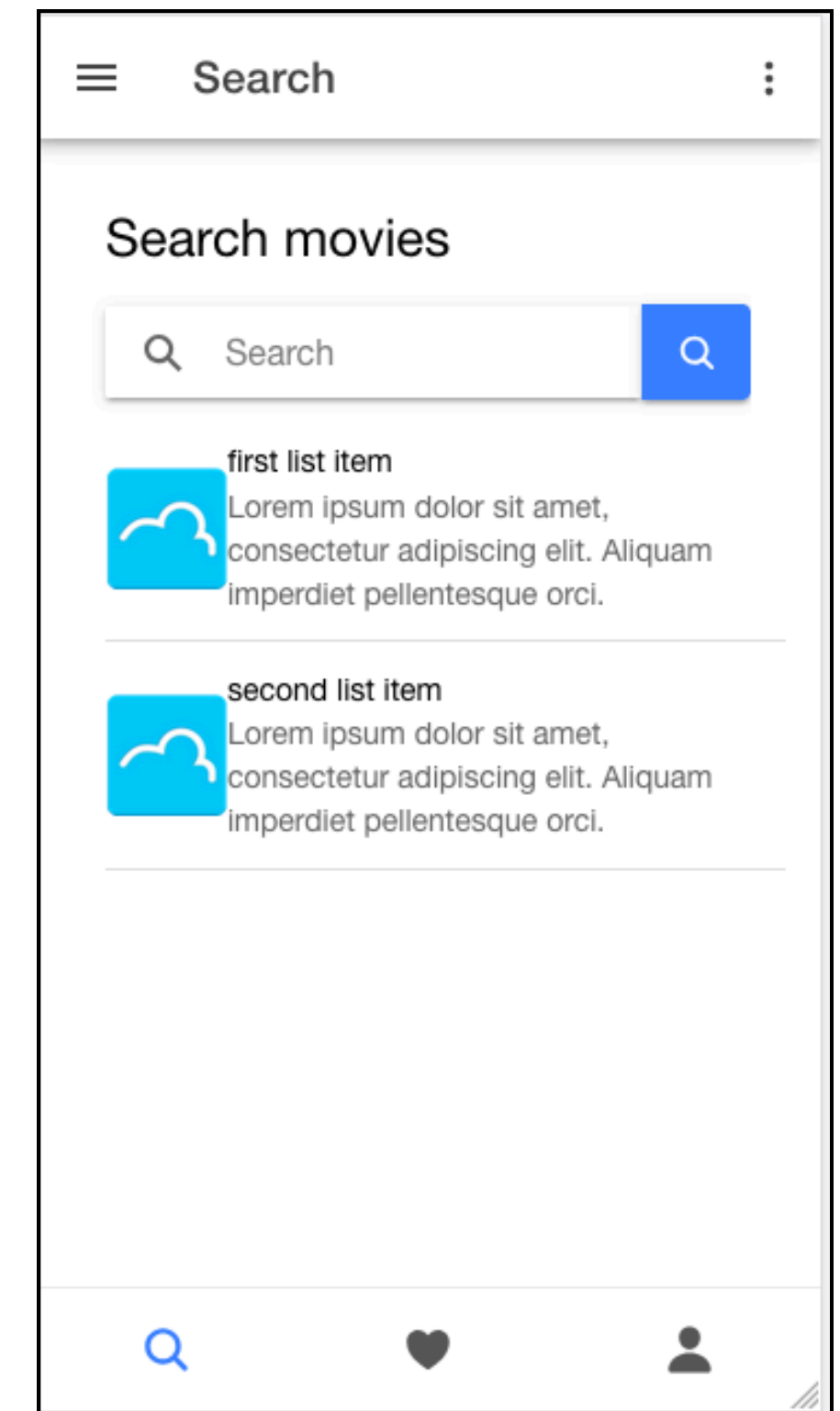
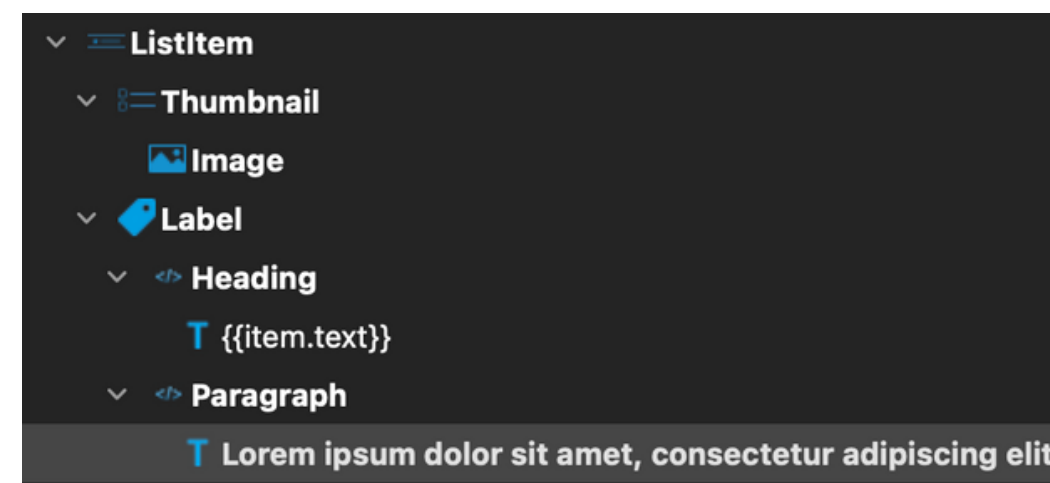
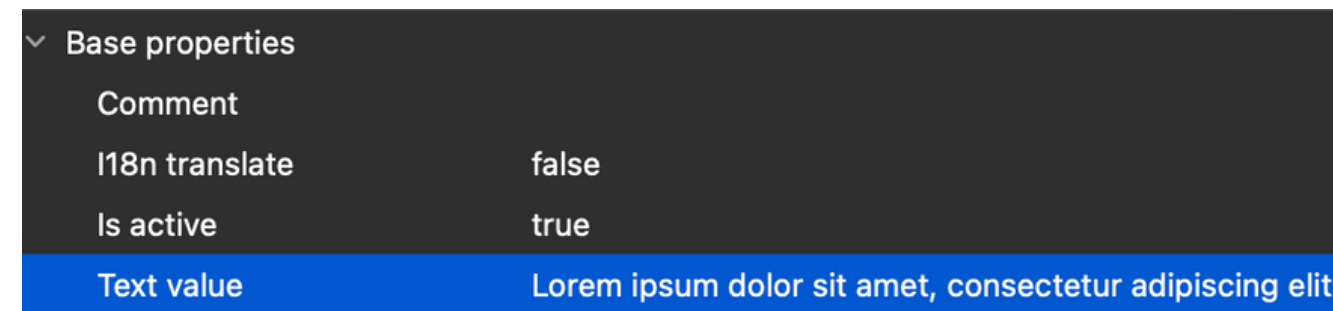
In the **Label**, after the **Heading**, add a **Paragraph**.



</> Paragraph

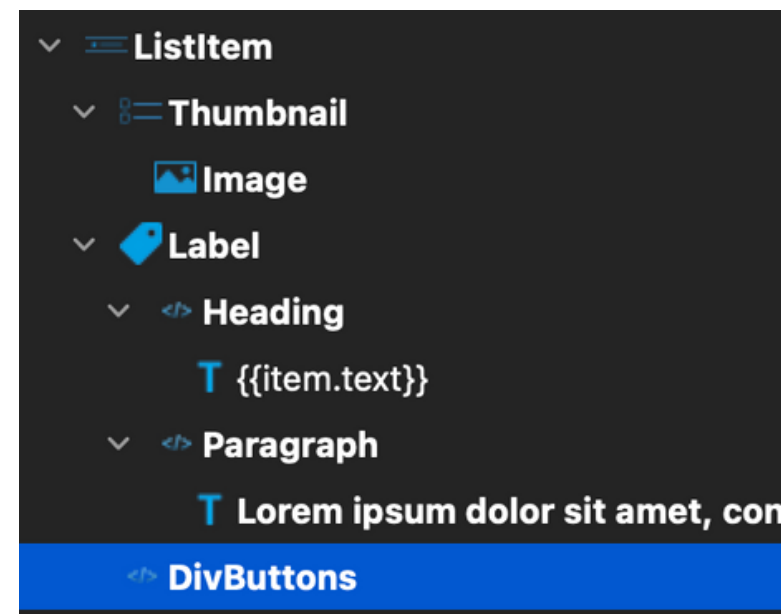


Replace the **Paragraph's Text value** by a few lines in Lorem ipsum.

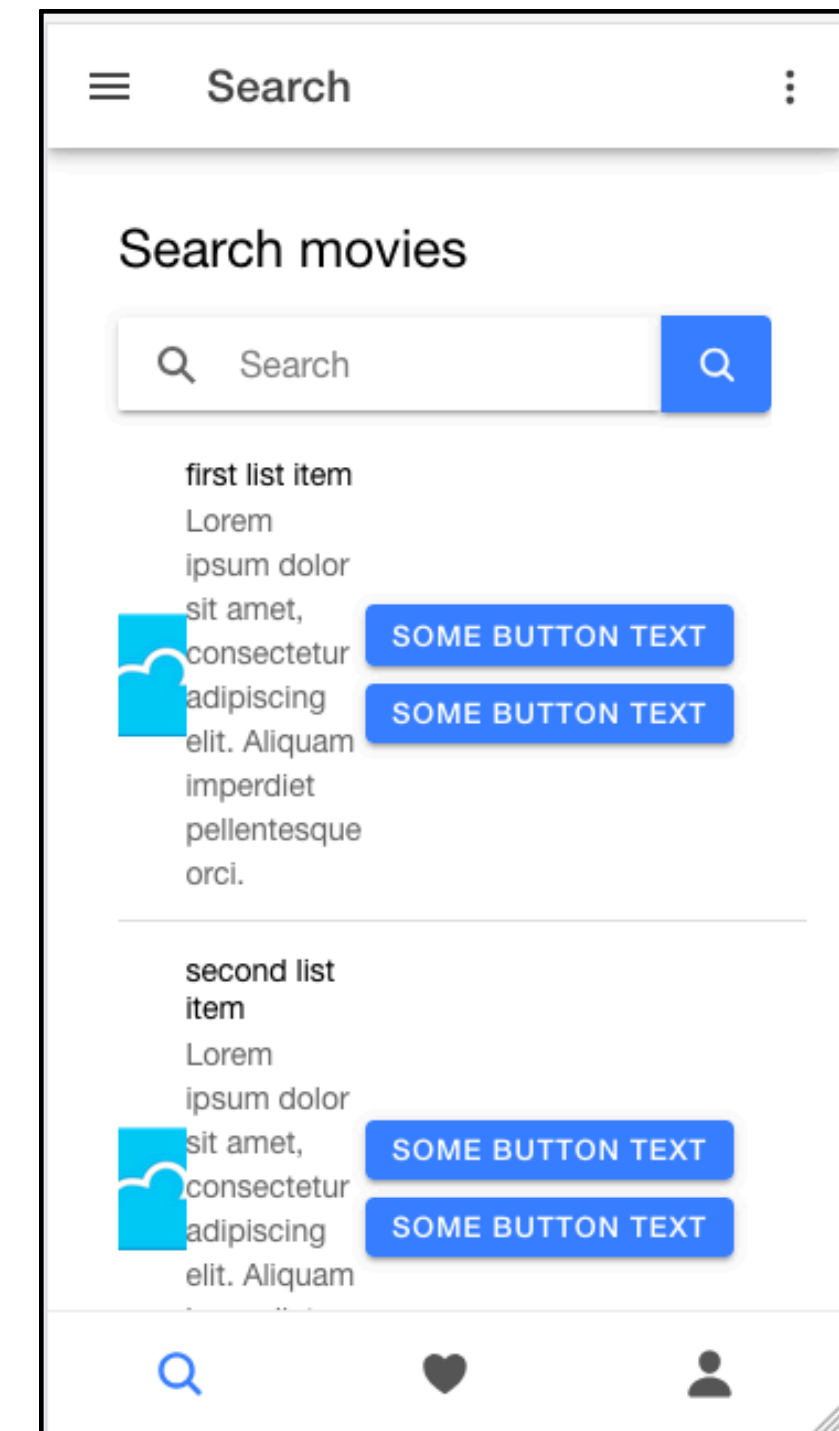
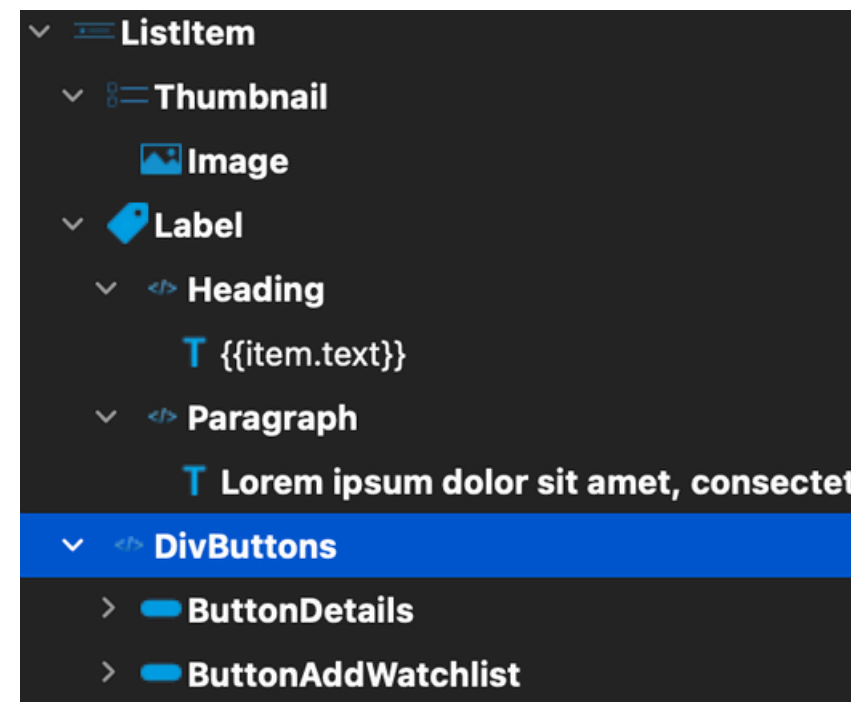


3.3 List & List items

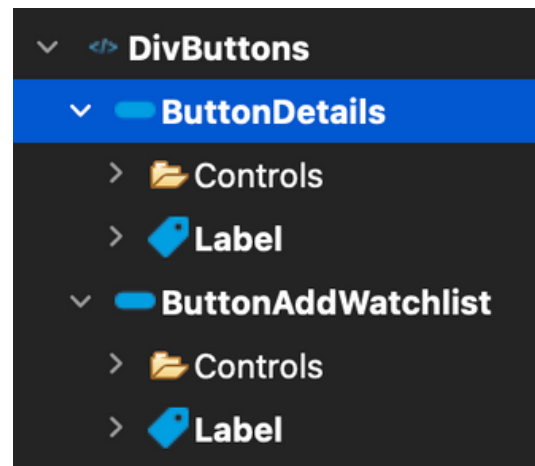
In the **Label**,
after the **Paragraph**,
add a **Div**
and rename it **DivButtons**.



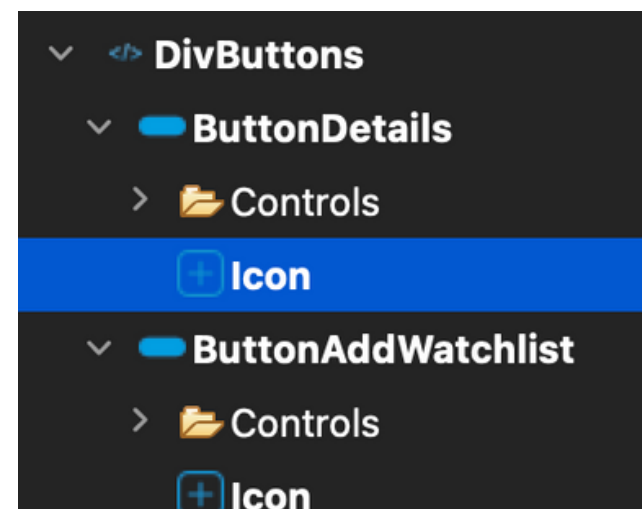
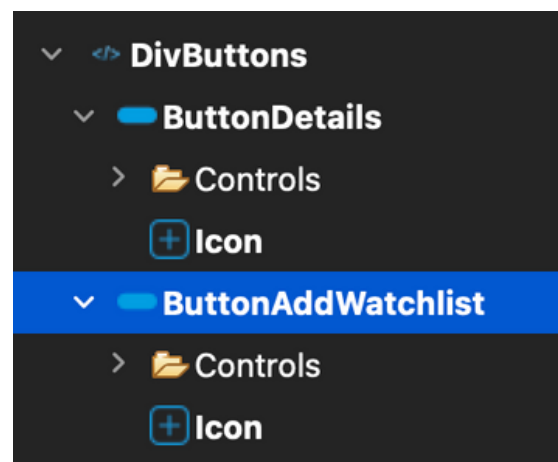
Add 2 buttons in **DivButtons**
and rename it **ButtonDetails**
and **ButtonAddWatchlist**.



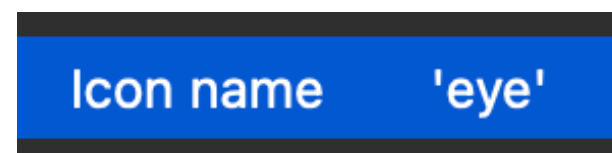
3.3 List & List items



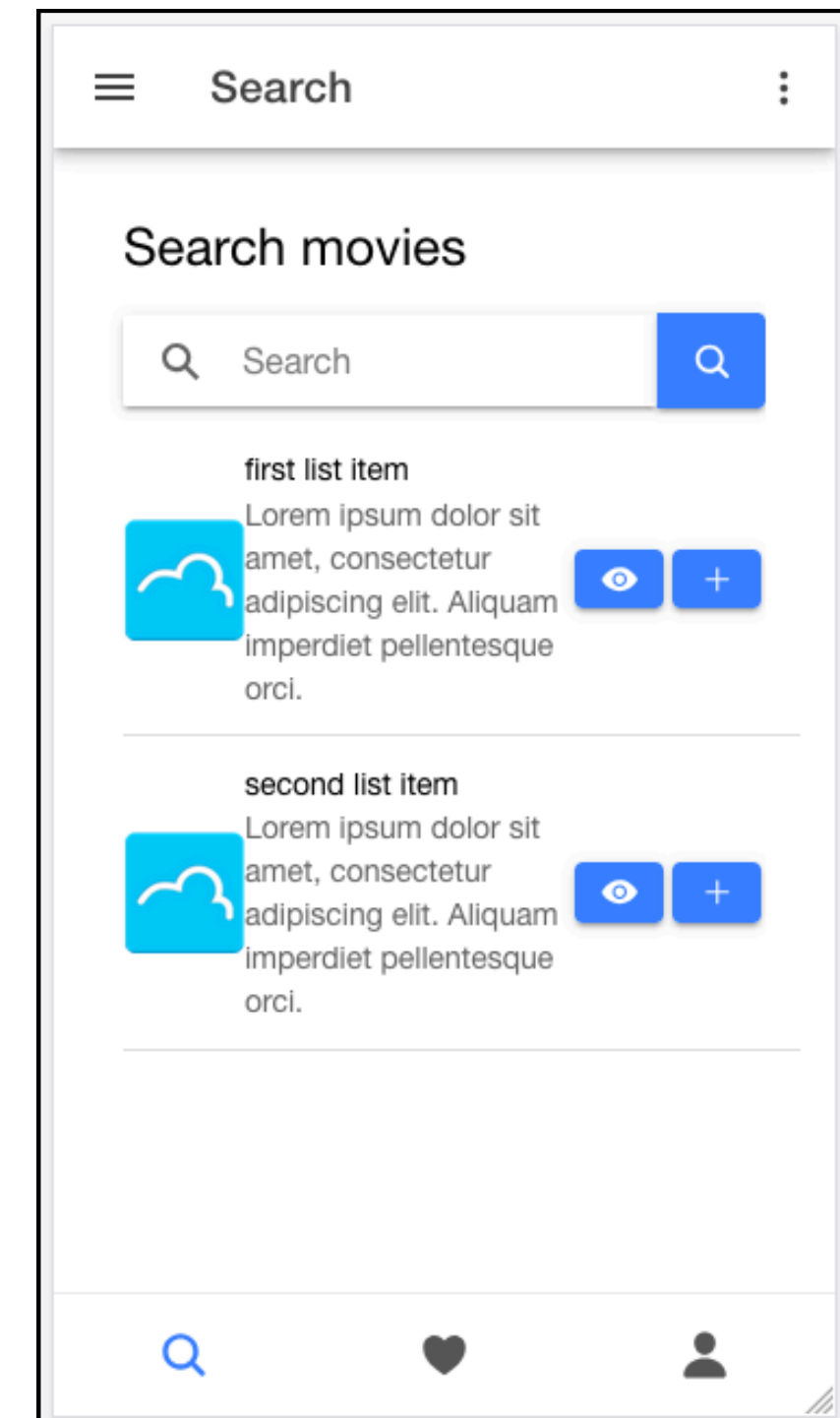
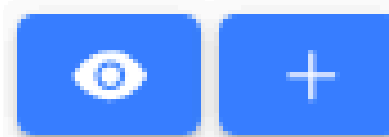
In each button,
replace the **Label**
by an **Icon**.



In **ButtonDetails**,
replace the **Icon** by '**eye**'.

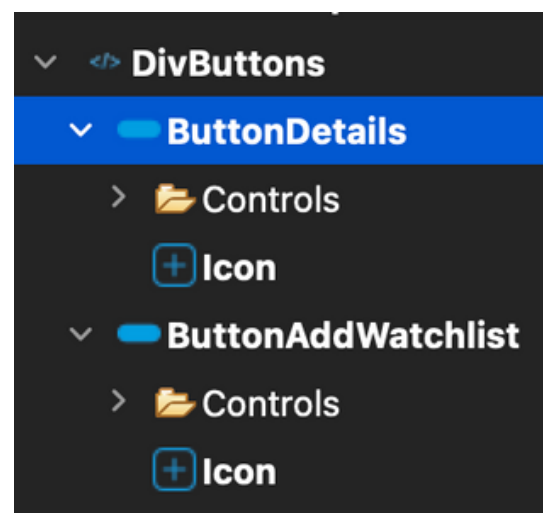
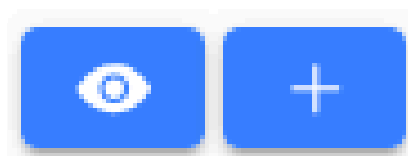


Keep the **Icon 'add'**
for **ButtonAddWatchlist**.



3.3 List & List items

To change the color of **ButtonDetails** by the **Secondary color** of the application's theme.



@Properties	
Color	not set
Disabled	false
Expand	not set
Fill	not set
Mode	not set

In **ButtonDetails's** properties,



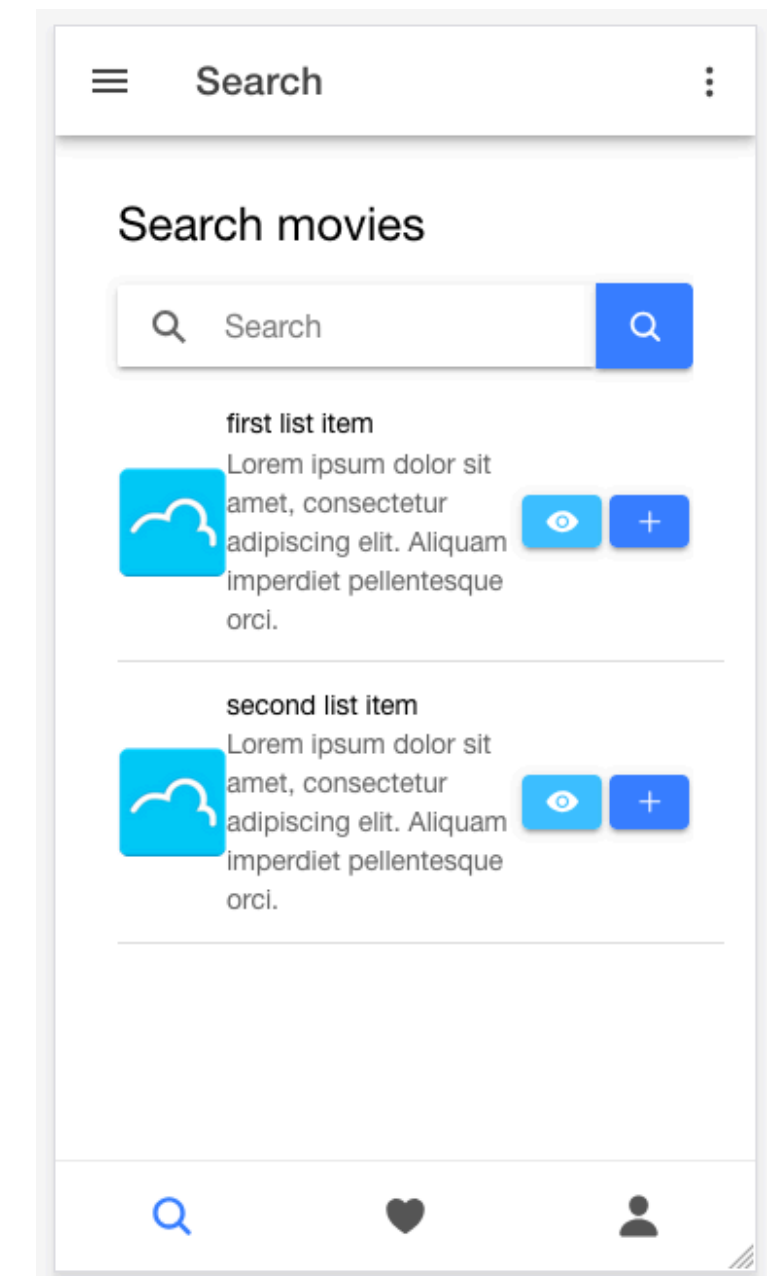
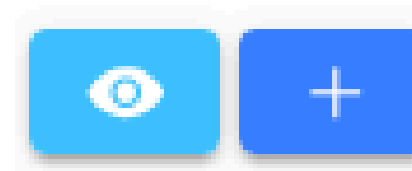
click on

Color property's Text value
to open the select.

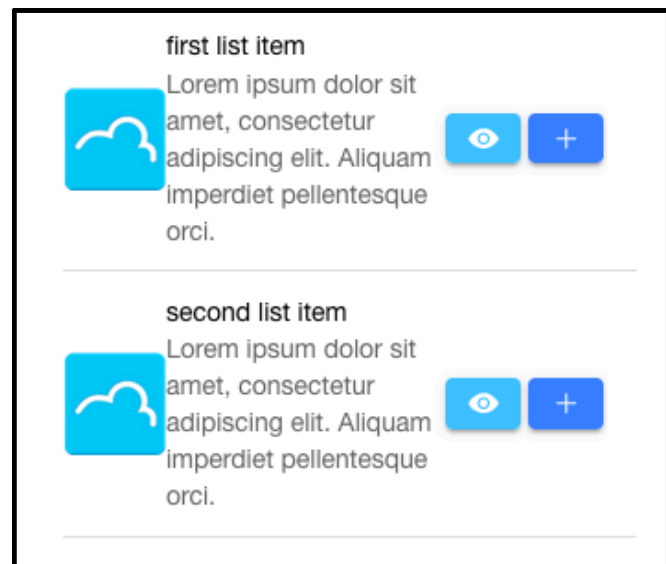
@Properties	
Color	not set
Disabled	not set
Expand	danger
Fill	dark
Mode	light
Shape	medium
Size	primary
Slot	secondary
	success
	tertiary

And select 'secondary'.

@Properties	
Color	secondary
Disabled	false



3.3 List & List items

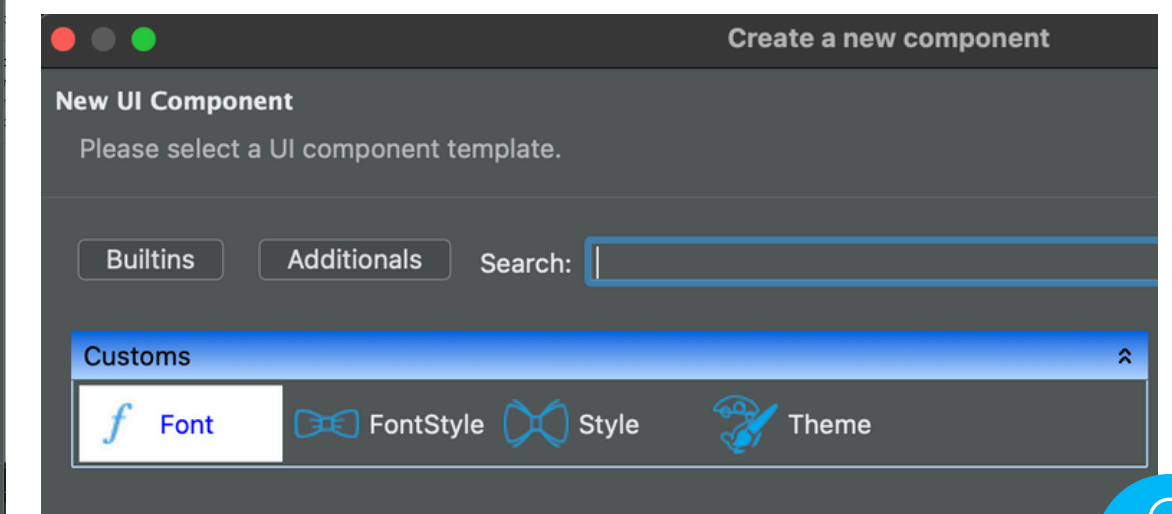
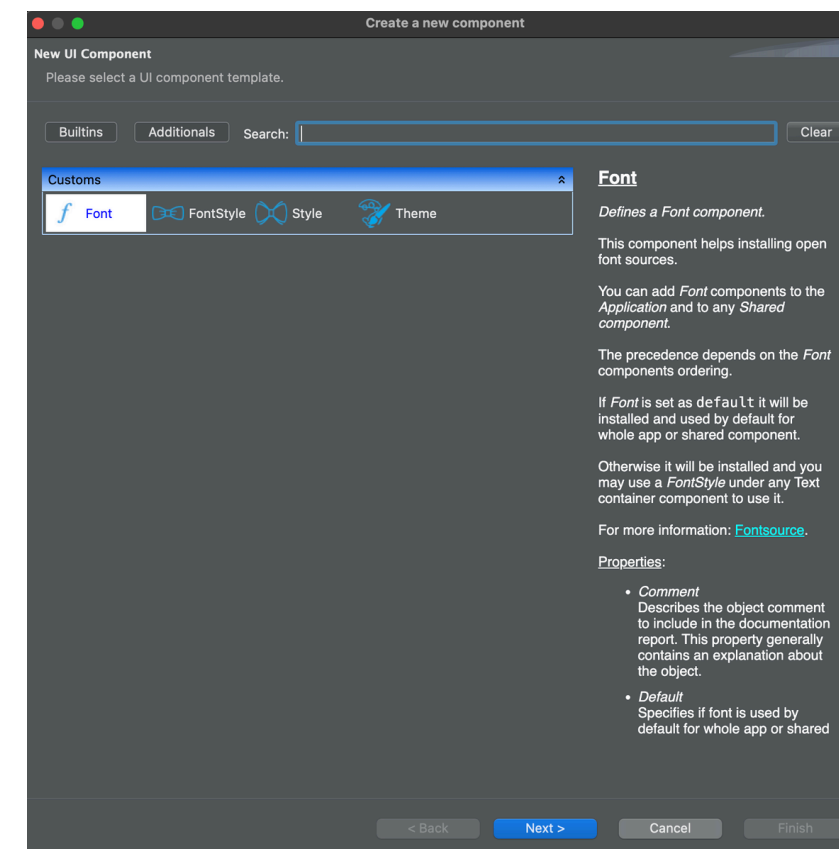
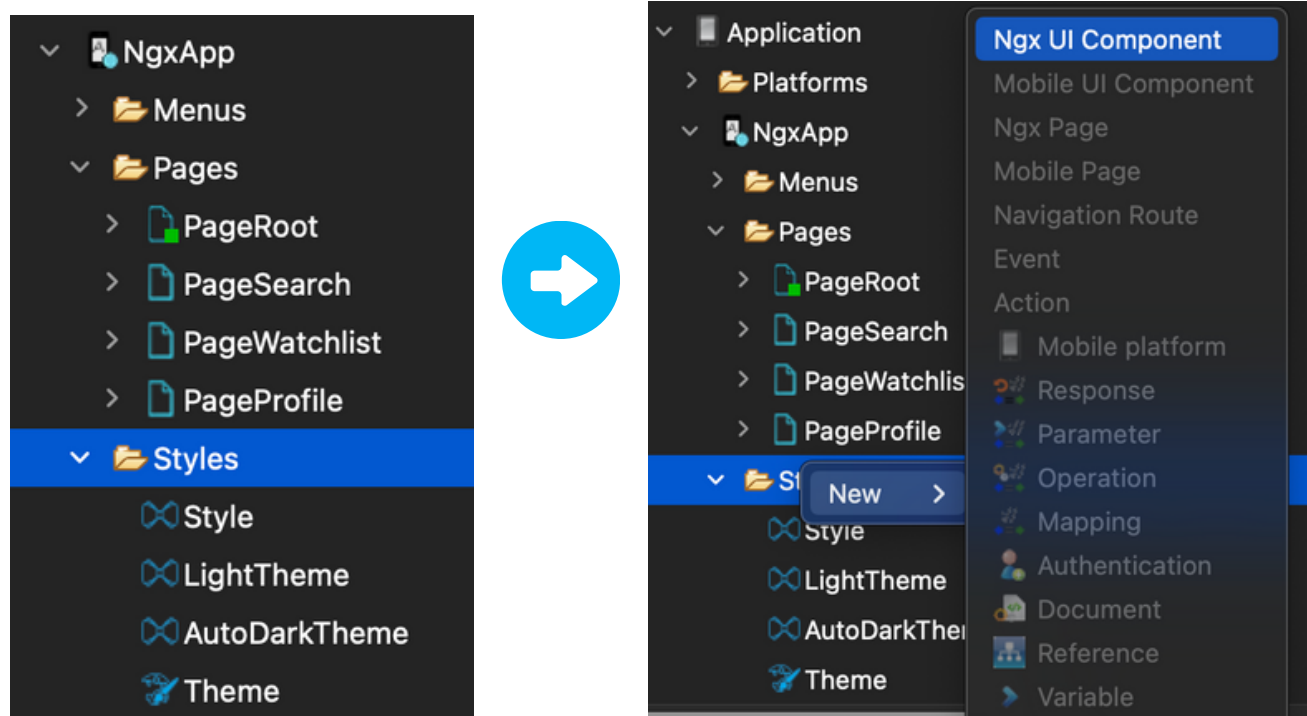


Let's add some styles to shorten the text of the **Paragraph**, and arrange the buttons in column.

To create a **new global Style**, you can either drag a **Style component** from the **Palette** and drop it in the **Styles folder**.

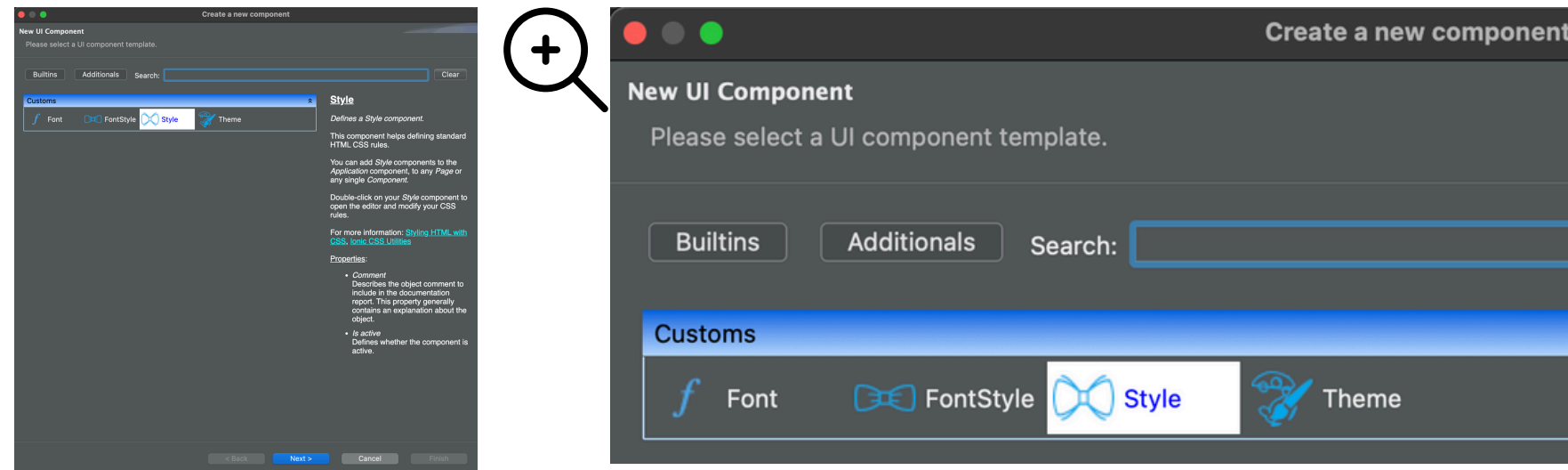
Or right-click on the **Styles folder** and select **New**, then **Ngx UI Component**...

...to open the **Create a new component panel**.

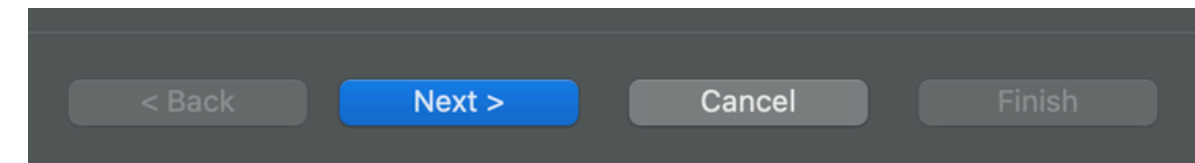


3.3 List & List items

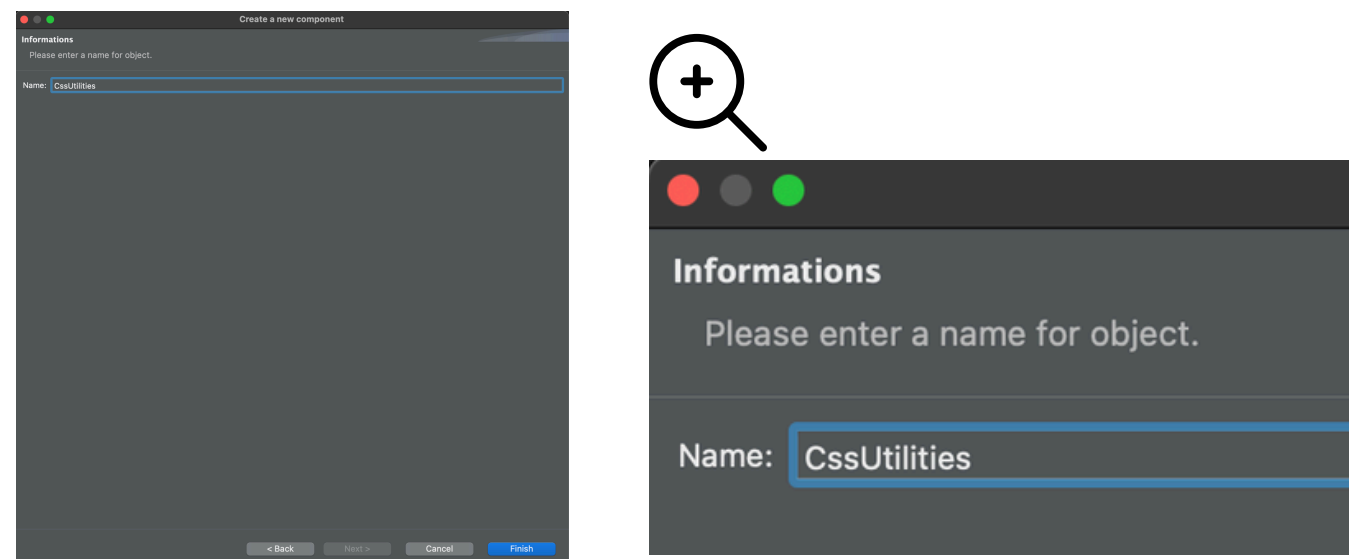
In the **Create a new component** panel, select **Style**.



And click on **Next**.



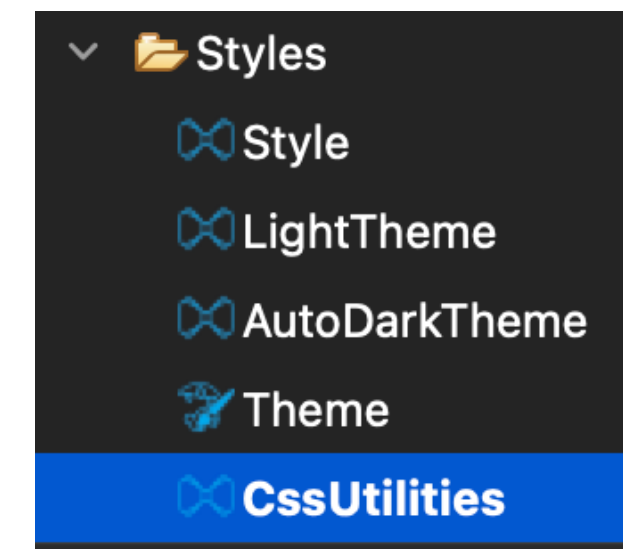
Enter **CssUtilities** as name for the **Style Component**.



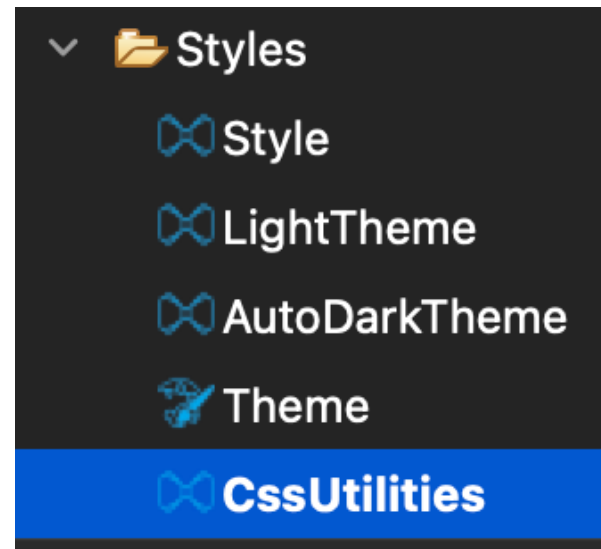
And click on **Finish**.



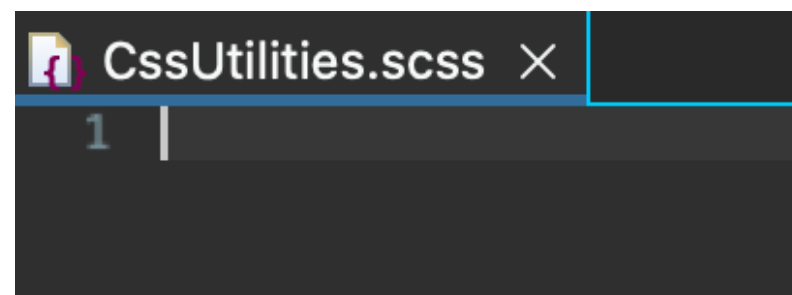
The **Style CssUtilities** is added in the **Style** folder.



3.3 List & List items



Click on **Style** **CssUtilities** to open **CssUtilities.scss**.



And copy these 3 css classes in **CssUtilities.scss**.

```
// Limit text's length to 2 lines
.clamp-2-lines {
  display: -webkit-box;
  -webkit-line-clamp: 2;
  -webkit-box-orient: vertical;
}

.overflow-hidden {
  overflow: hidden !important;
}

.flex-column {
  display: flex;
  flex-direction: column;
}
```



```
CssUtilities.scss x
1 // Limit text's length to 2 lines
2 .clamp-2-lines {
3   display: -webkit-box;
4   -webkit-line-clamp: 2;
5   -webkit-box-orient: vertical;
6 }
7
8 .overflow-hidden {
9   overflow: hidden !important;
10 }
11
12 .flex-column {
13   display: flex;
14   flex-direction: column;
15 }
16
```

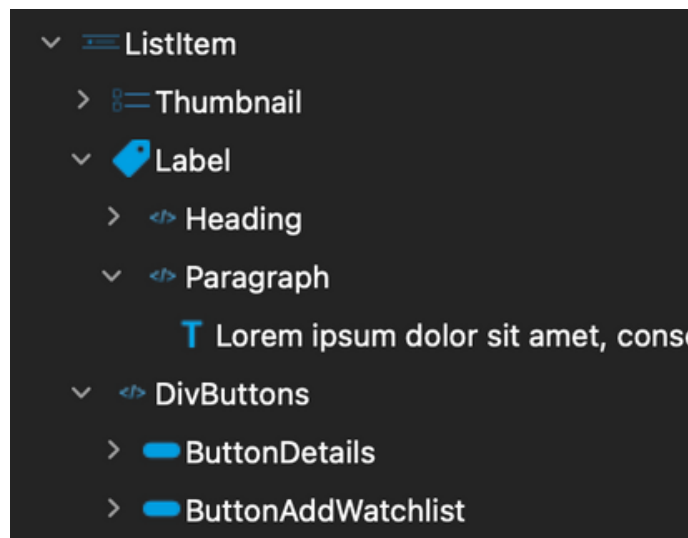
The classes declared in **CssUtilities.scss** are global classes and can be used everywhere in the application.



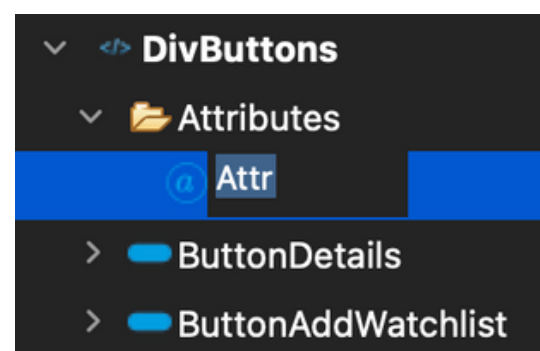
3.3 List & List items

To arrange the buttons in column, let's add the **class flex-column** to the **DivButtons**

And change the **Attr value** to **flex-column**.



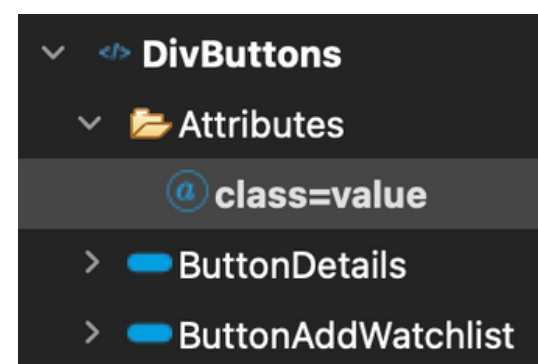
In **DivButtons**,
let's add an **Attr** component.



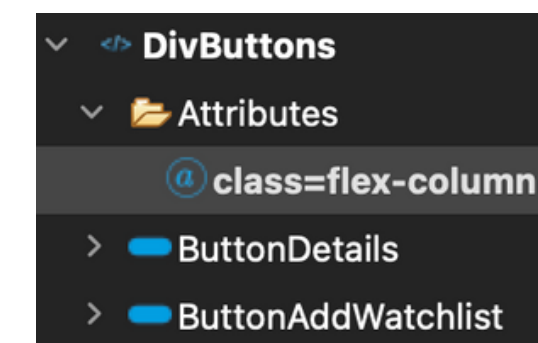
Properties X	
Property	Value
Base properties	
Attr name	attr
Attr value	value
Comment	
Is active	true

In the **Attr** component's properties,
let's change the **Attr** name to **class**.

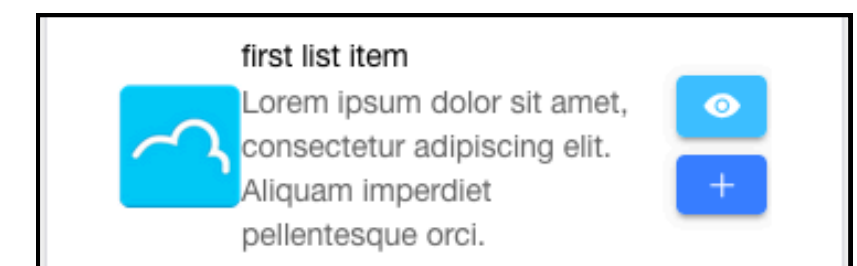
Base properties	
Attr name	class
Attr value	value
Comment	
Is active	true



Base properties	
Attr name	class
Attr value	flex-column
Comment	
Is active	true



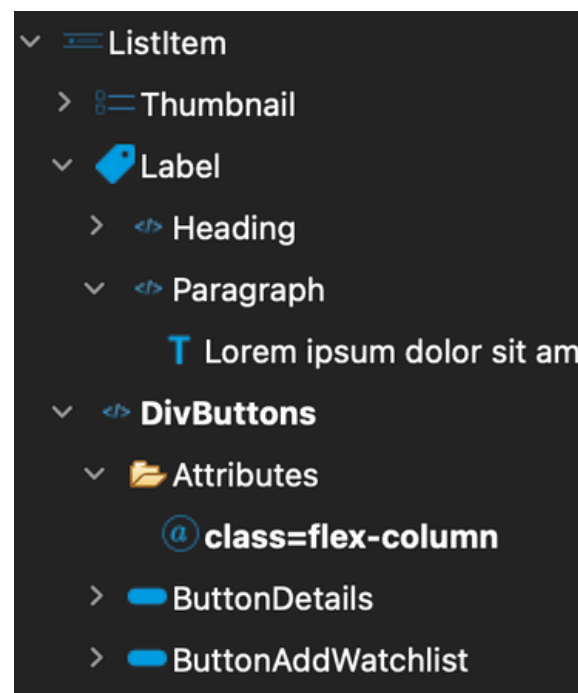
The class **flex-column**
is now applied to **DivButtons**.



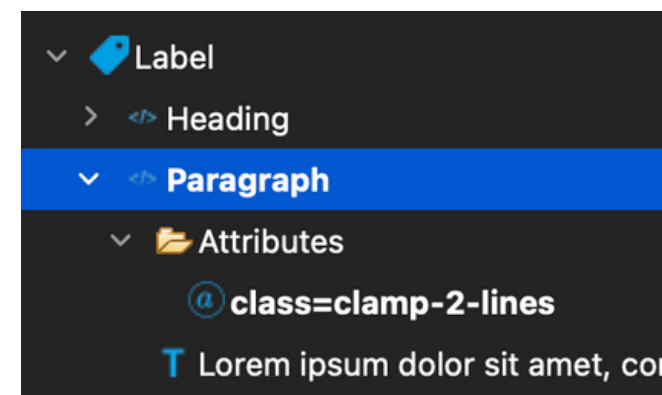
3.3 List & List items

We need to shorten the text of the **Paragraph**.

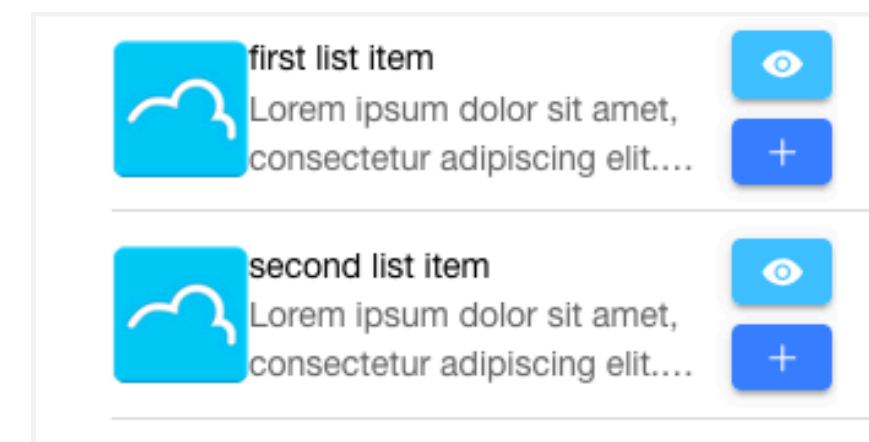
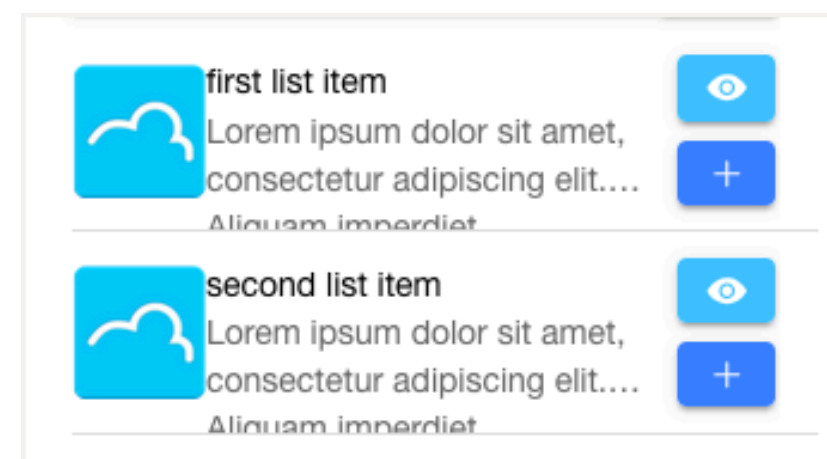
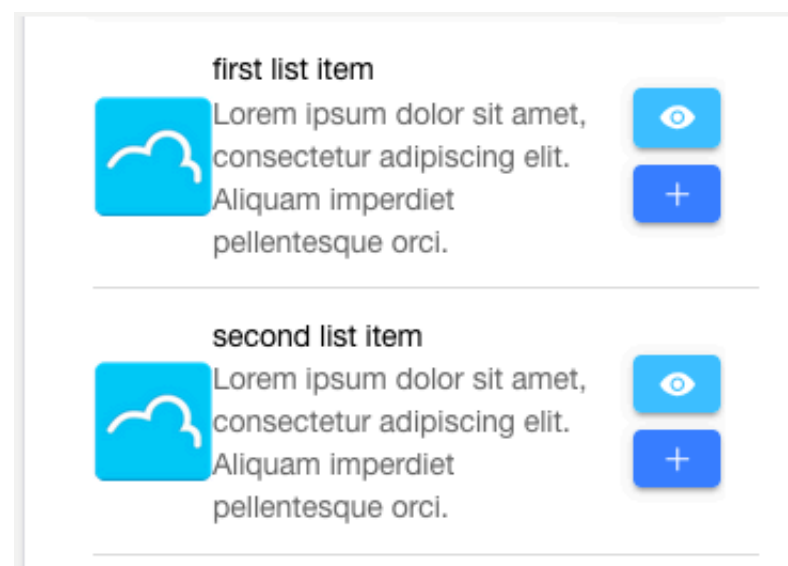
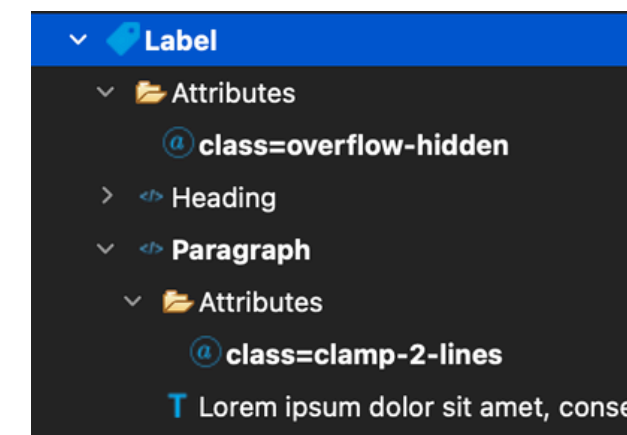
As shown before, let's use Attr components to use the classes declared in CssUtilities.



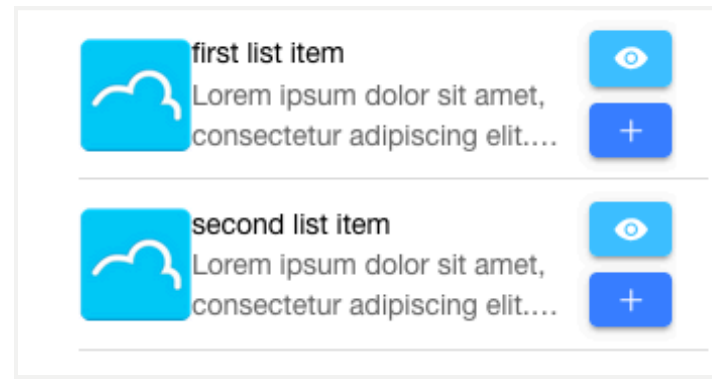
Add the **class clamp-2-lines** to the **Paragraph**.



Add the **class overflow-hidden** to the **Label**.



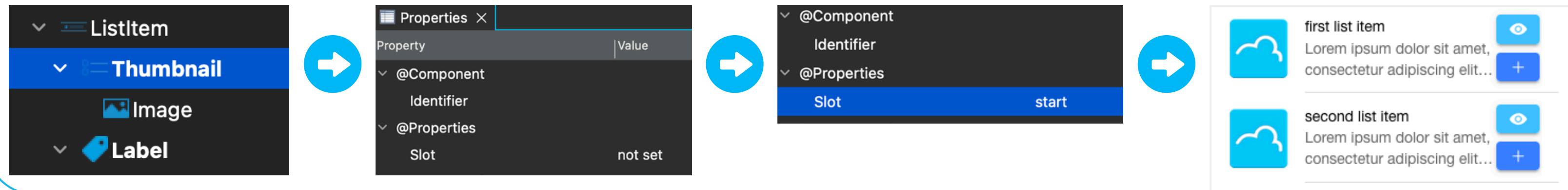
3.3 List & List items



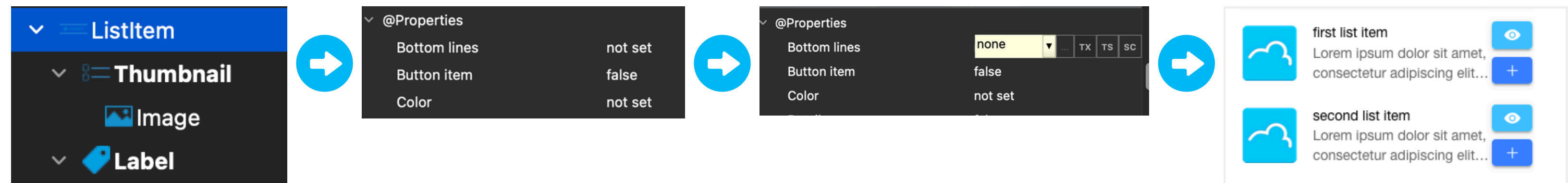
We need

- to add some space between the **Thumbnail** and the **Label**
- to remove the bottom lines under each **List item**.

In **Thumbnail's** properties, let's change the **value** of the **Slot** property from **not set** to **start**.



In **ListItem's** properties, let's change the **value** of the **Bottom lines** property from **not set** to **none**.



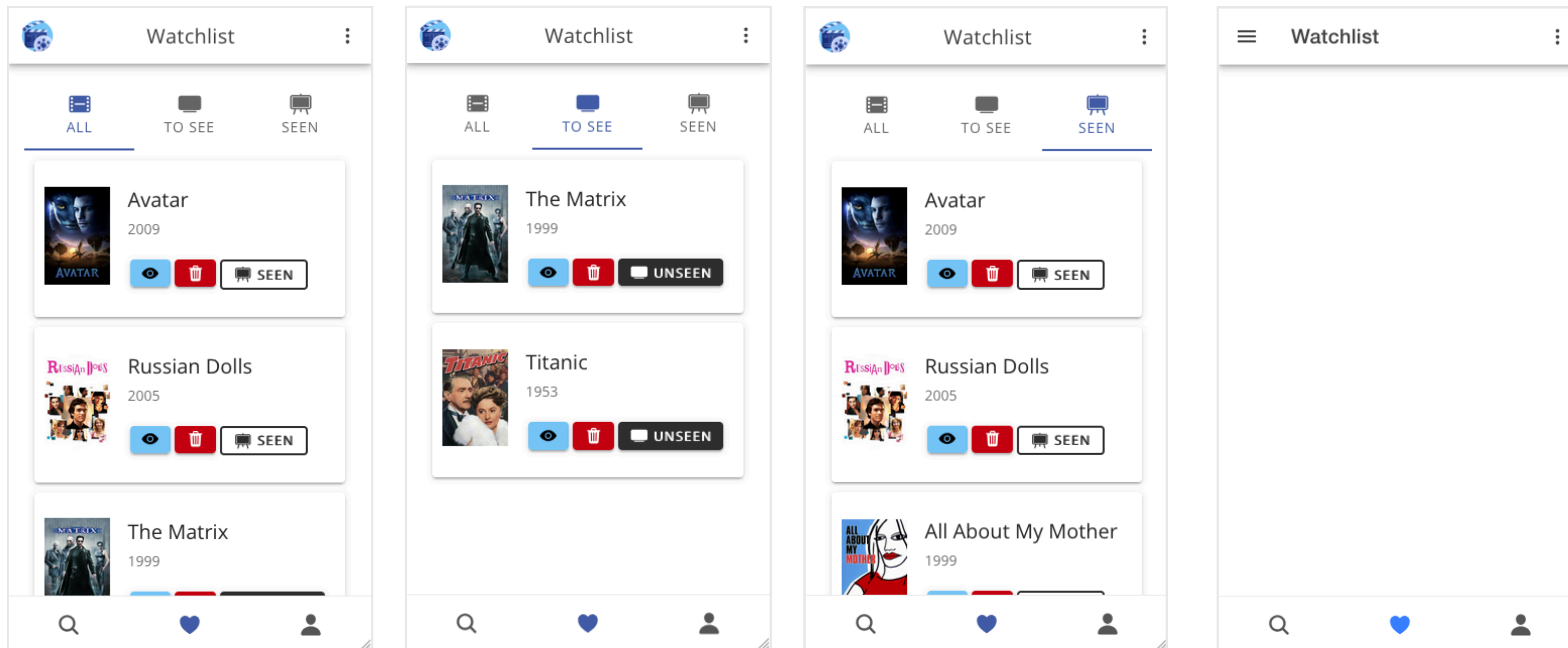
3.4 Segments

Let's create the layout of **PageWatchlist**.

In this page, we want to display 3 segments, similar to the example below.

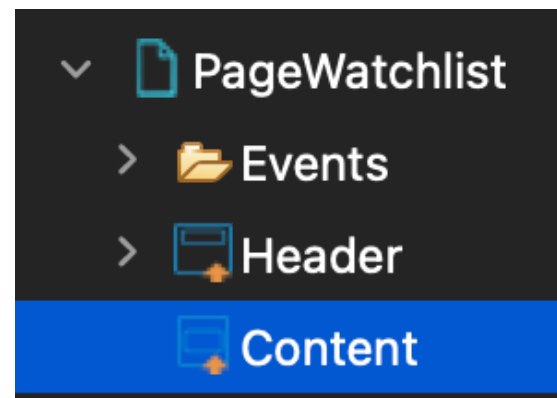
As for now,

the **Content** of **PageWatchlist** is empty and it appears as such.

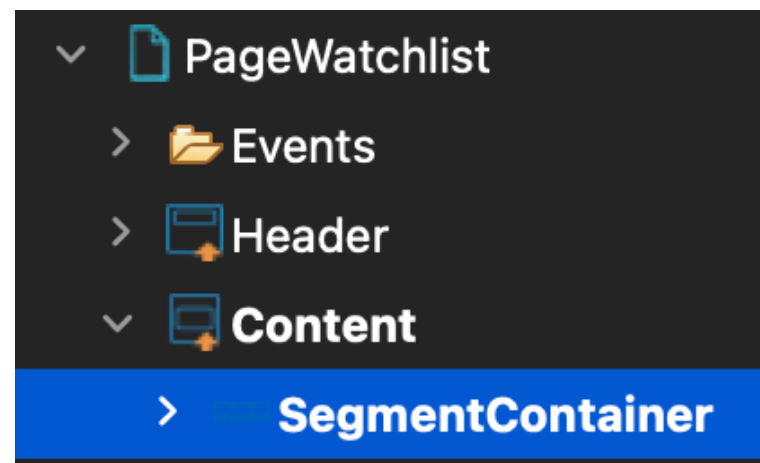


3.4 Segments

Add a **Segment Container** component in **PageWatchlist**'s **Content**.



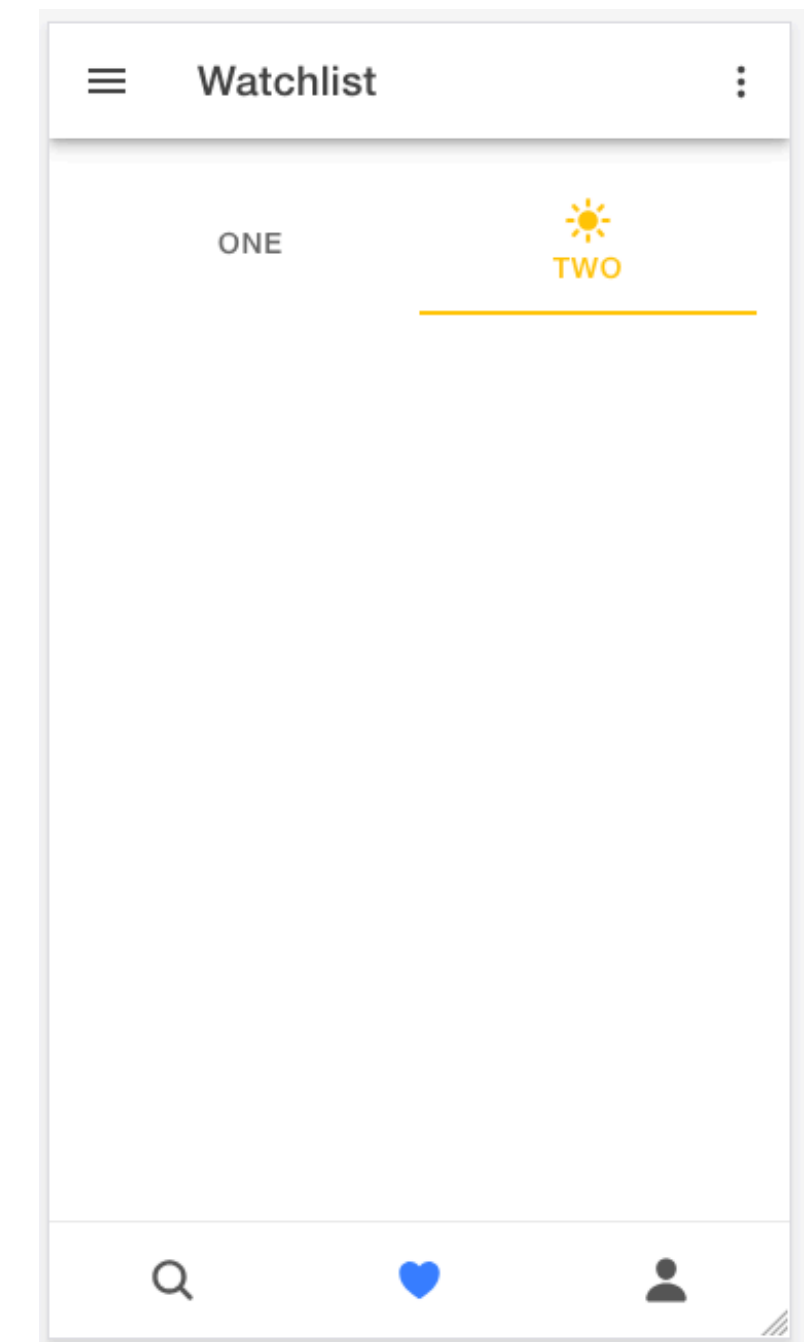
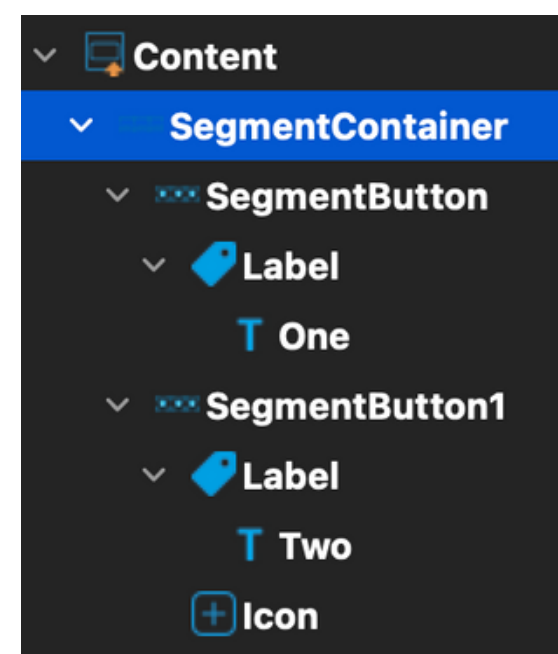
Segment Container



By default, the **Segment Container** is created with 2 **Segment Buttons** inside.



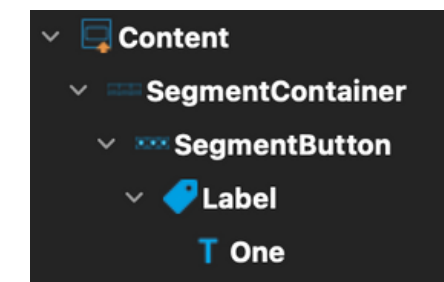
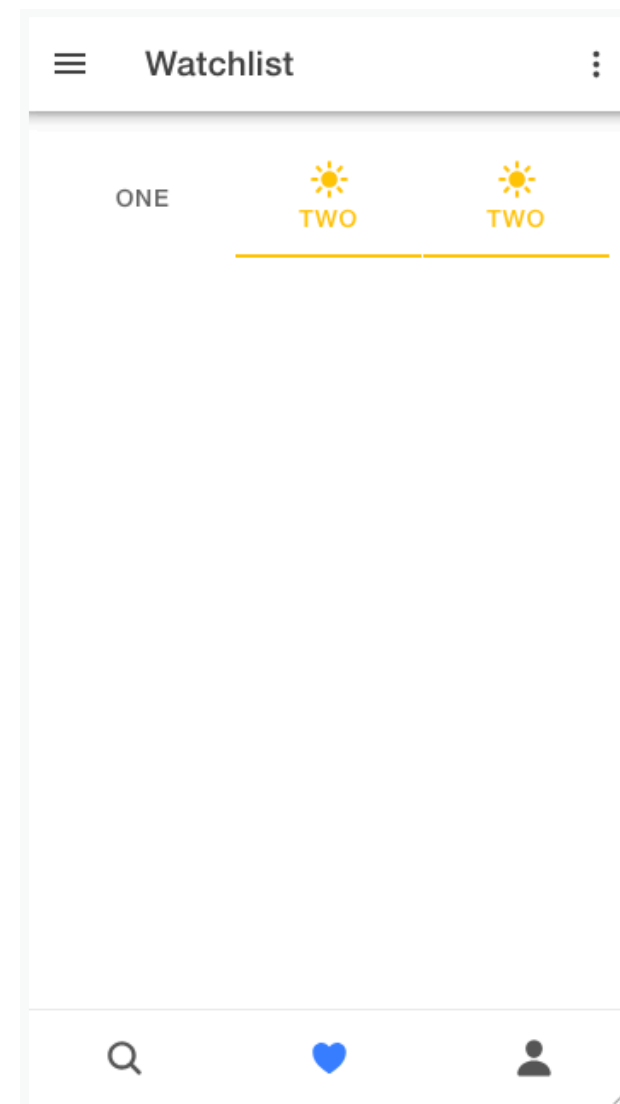
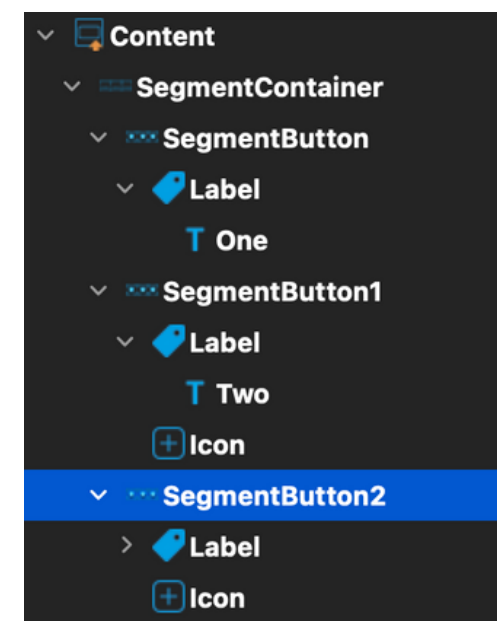
Each of these **Segment Buttons** is created with a **Label** and an **Icon** inside.



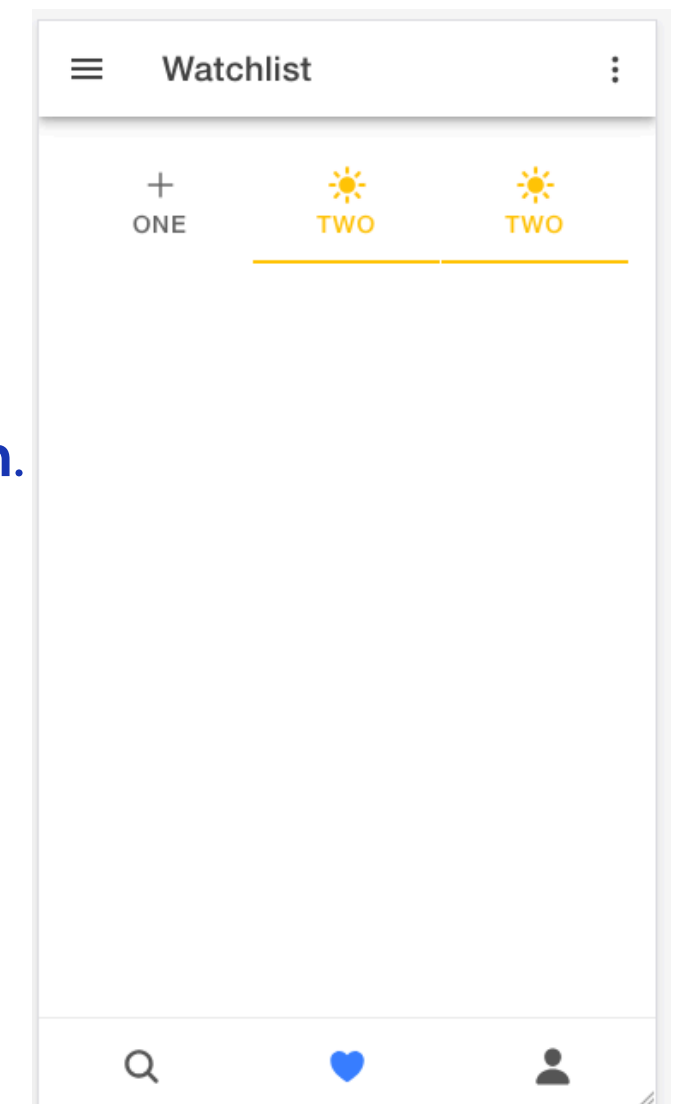
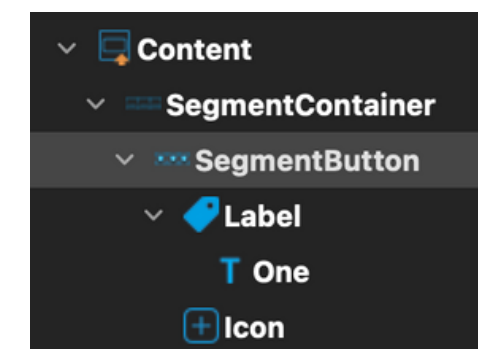
3.4 Segments



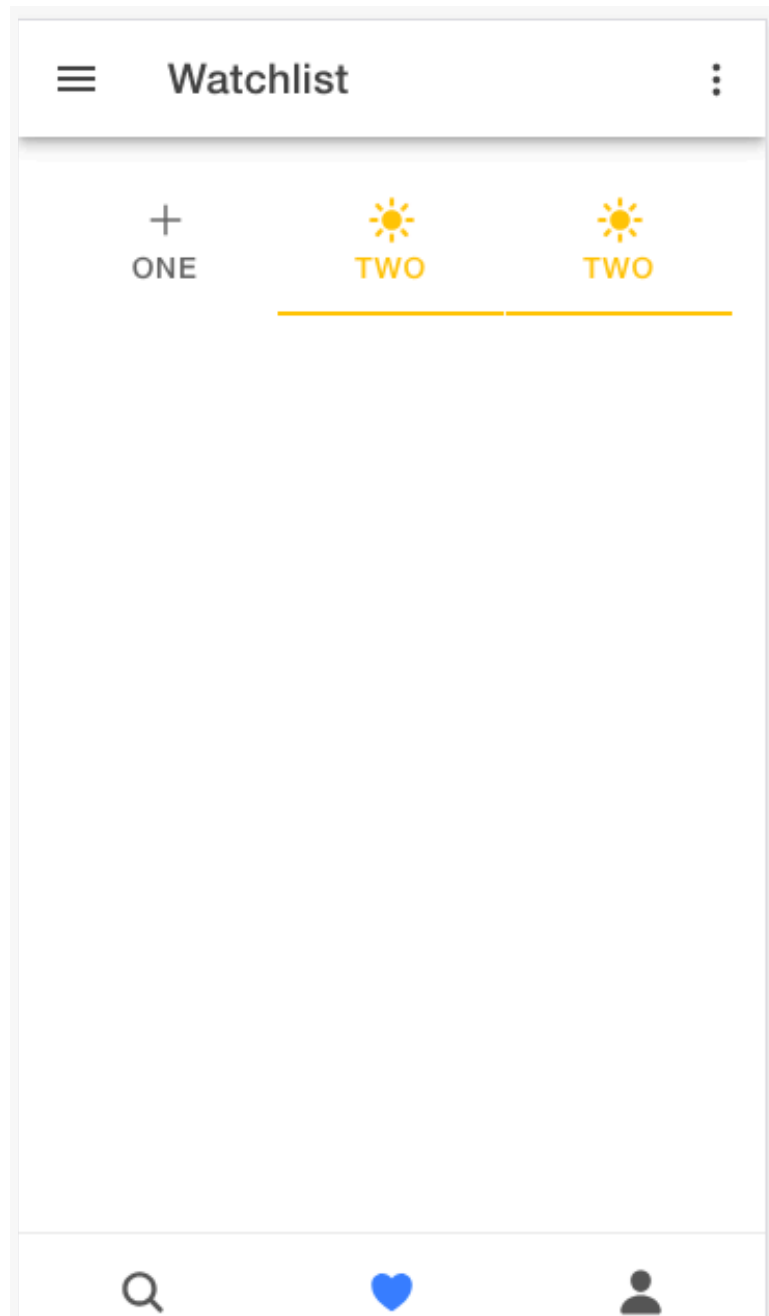
Duplicate
the second **Segment Button**
to add a third one.



Add an **Icon**
in the first **Segment Button**.



3.4 Segments



For each **Segment Button**

- rename it
- change the **value** of the **Label's Text** element
- change the **Icon name**



SegmentButtonAll

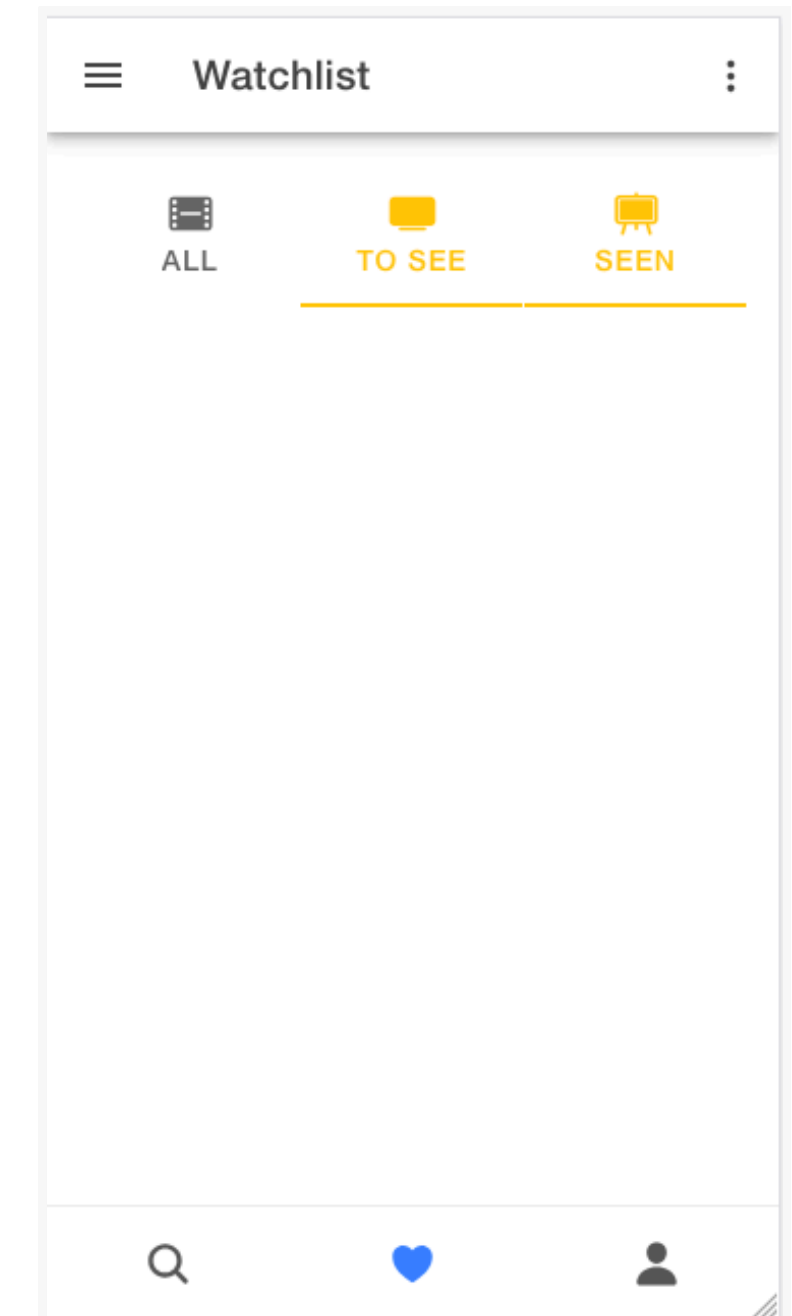
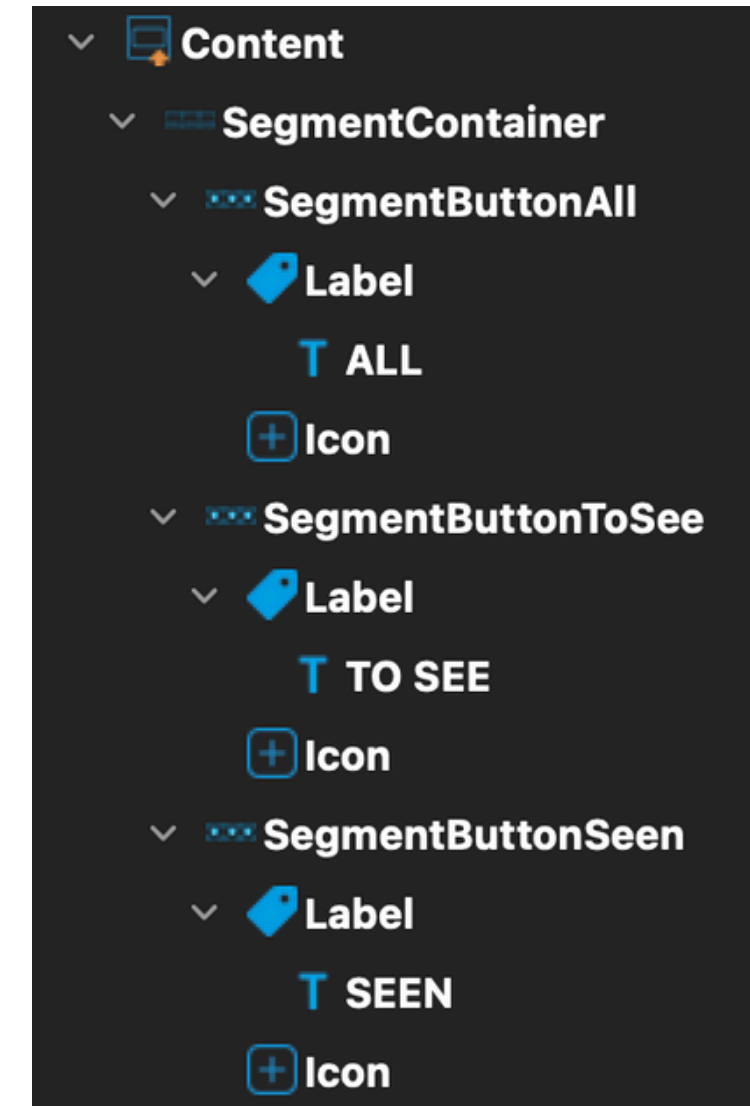
- Label : ALL
- Icon : 'film'

SegmentButtonToSee

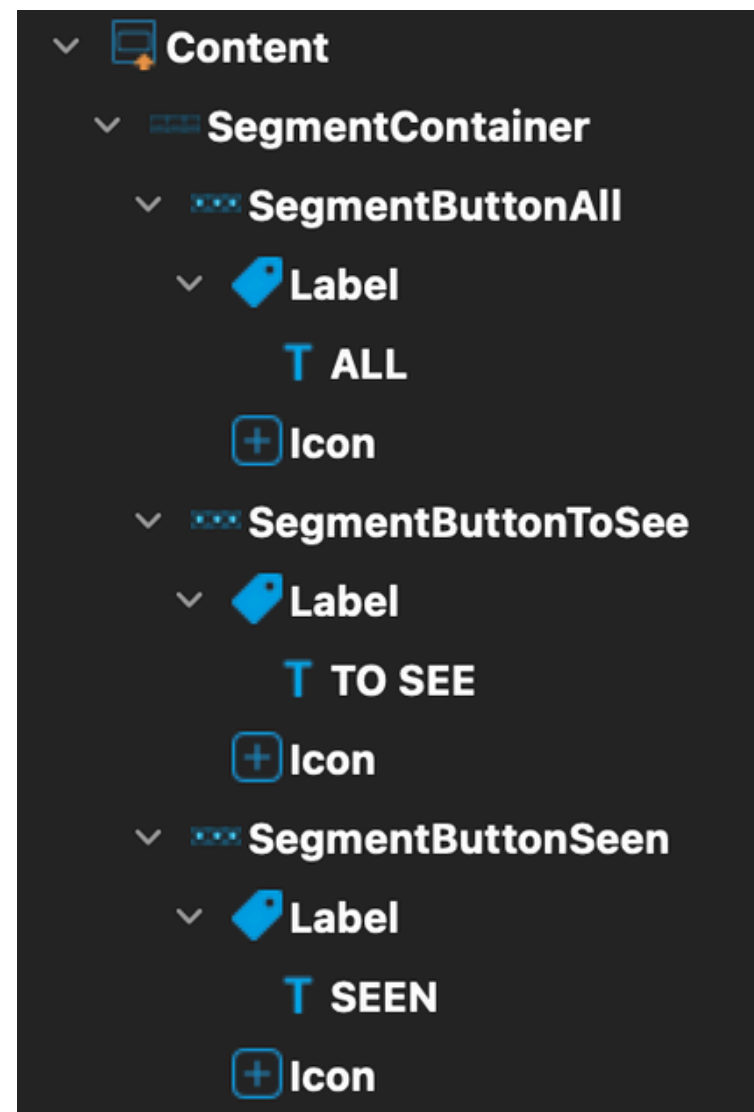
- Label : TO SEE
- Icon : 'tv'

SegmentButtonSeen

- Label : SEEN
- Icon : 'easel'



3.4 Segments

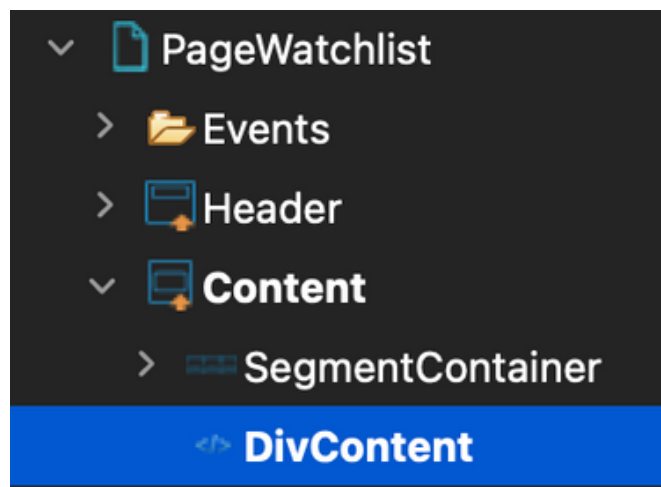


In PageWatchlist's Content, after SegmentContainer, add a Div and rename it DivContent.

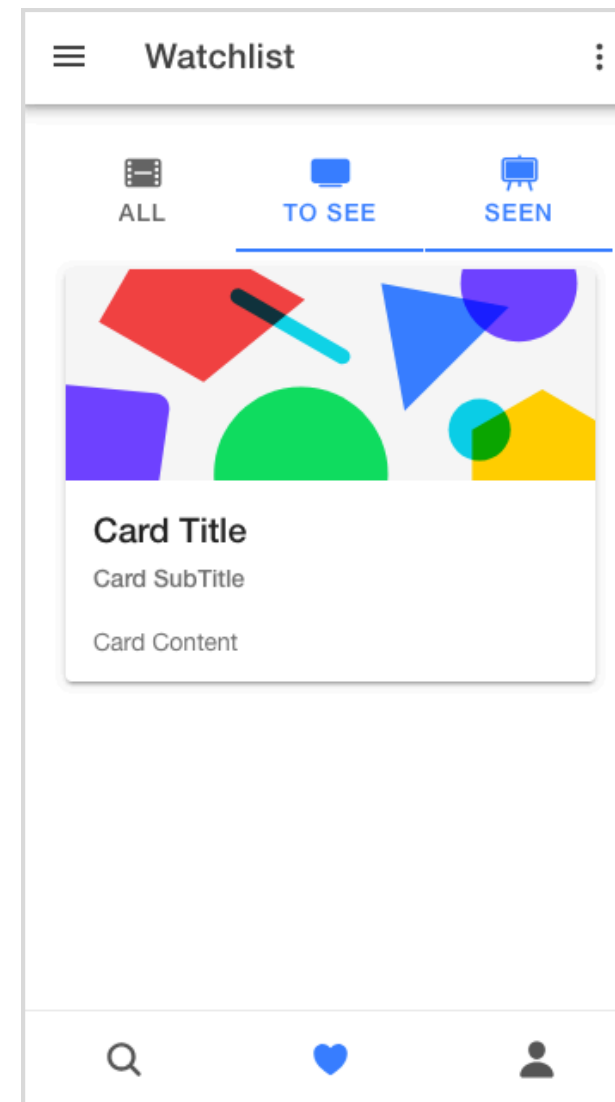
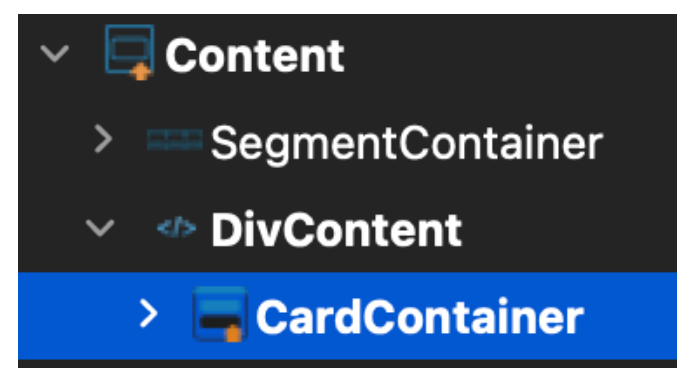
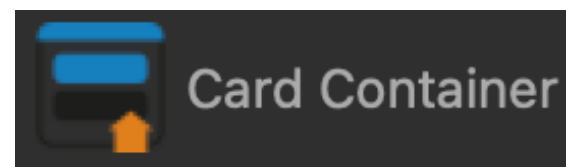
We will use it as container for the Segments' content.



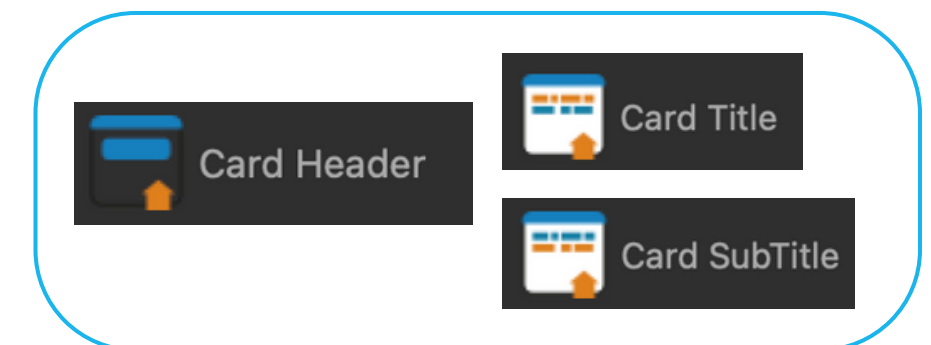
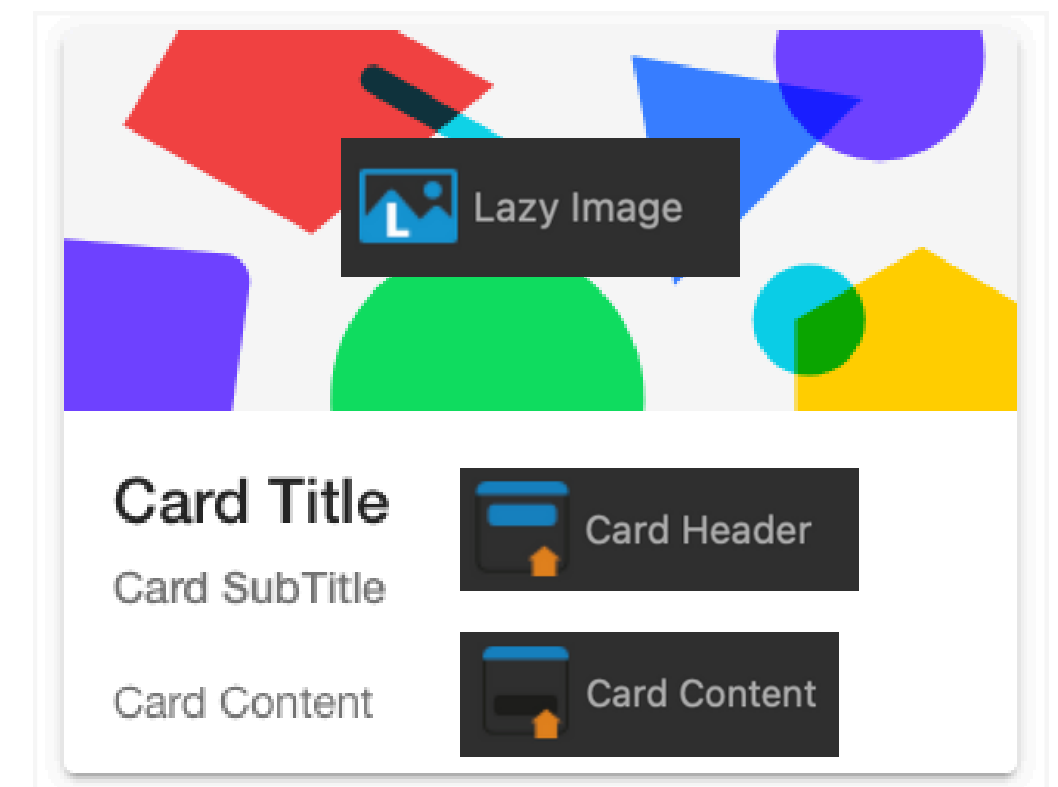
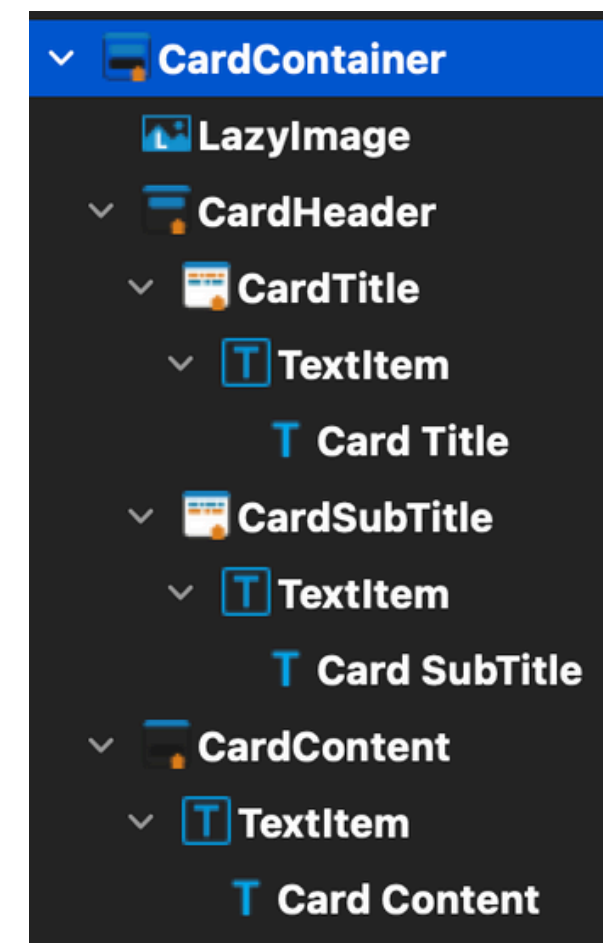
3.5 Cards & Grid



Add a **Card Container** component in **DivContent**.

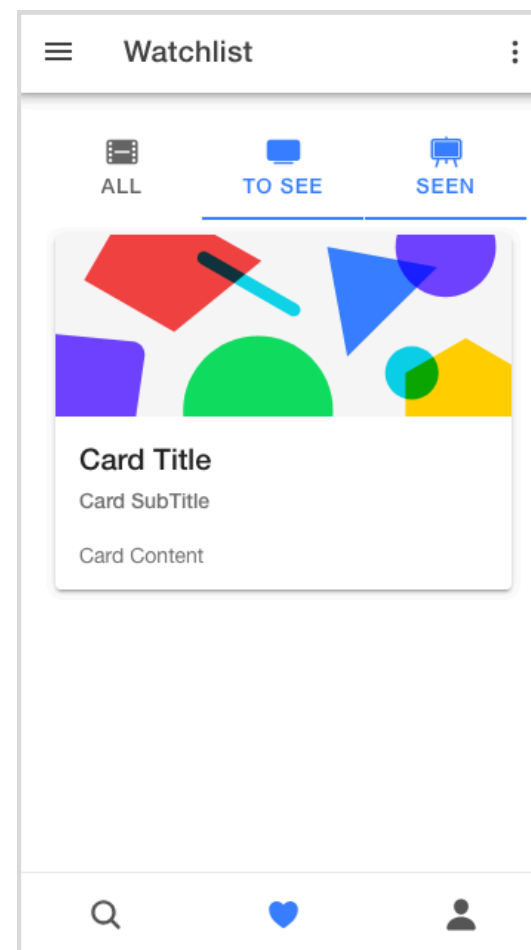
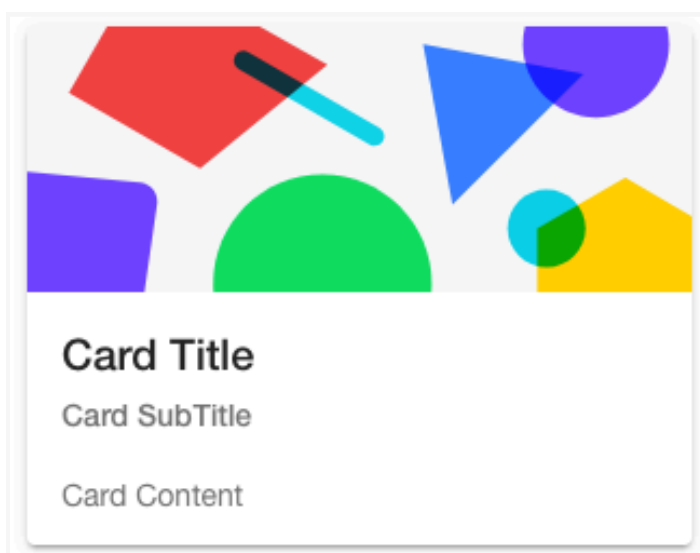
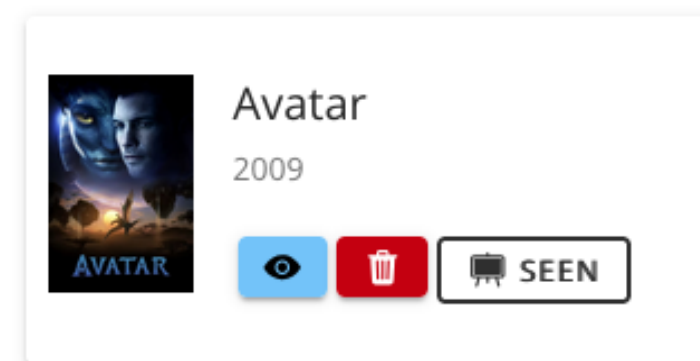


By default, a **Card Container** is created with various components inside.

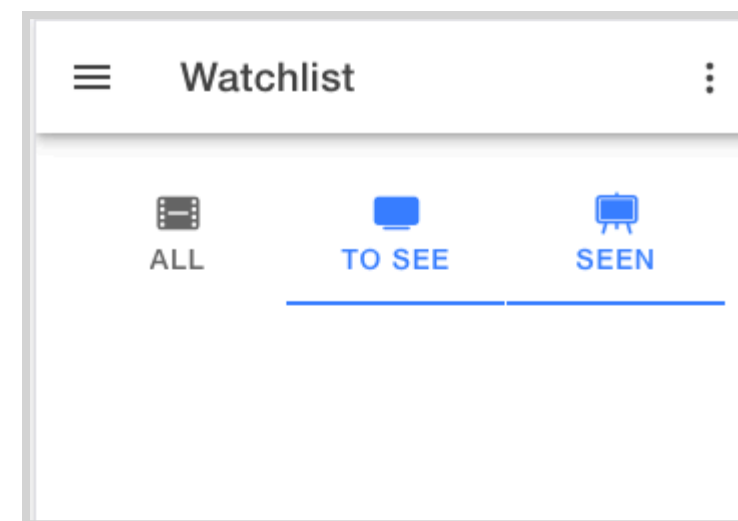
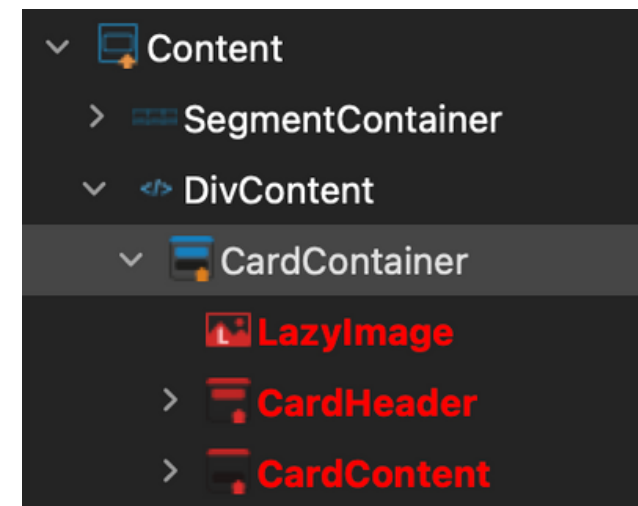


3.5 Cards & Grid

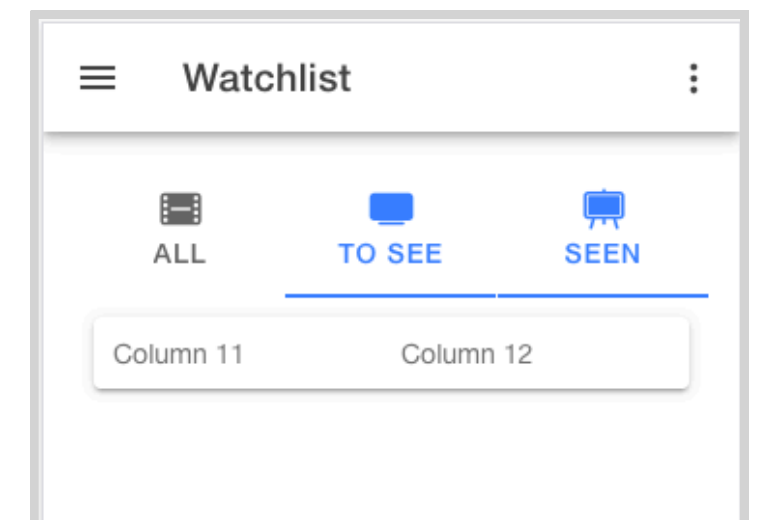
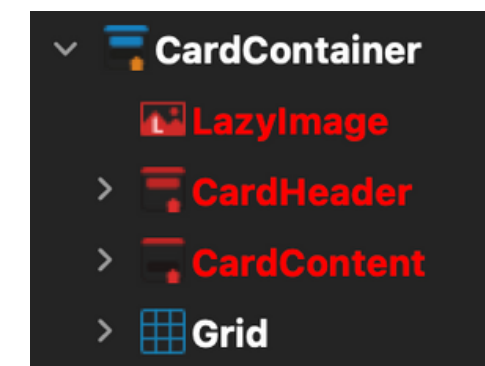
In the example,
the layout of the **Card Container** is different,
and the image is on the left.
Let's make a few changes
in our default **Card Container**.



First, disable the **LazyImage**,
CardHeader and **CardContent**
in the **CardContainer**.



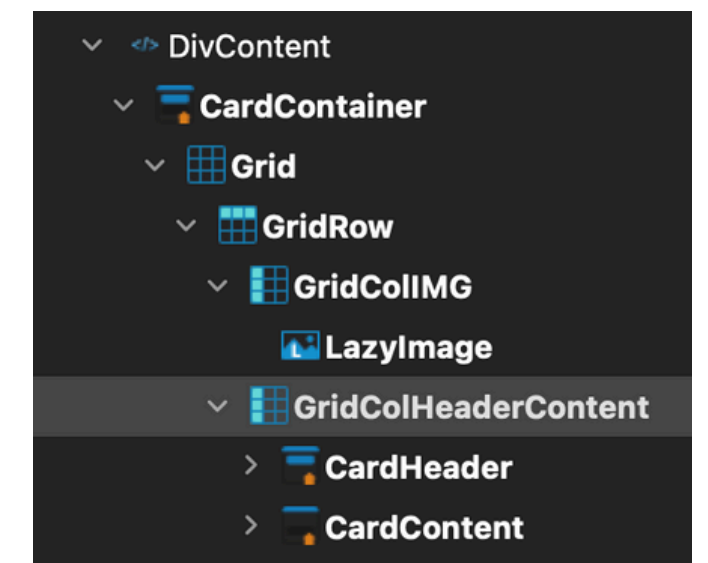
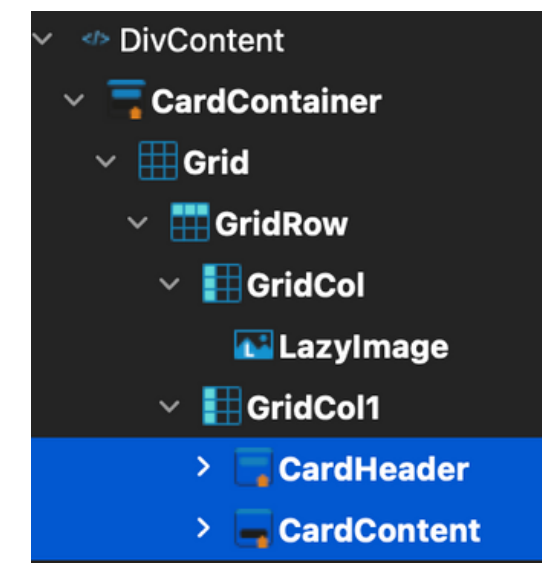
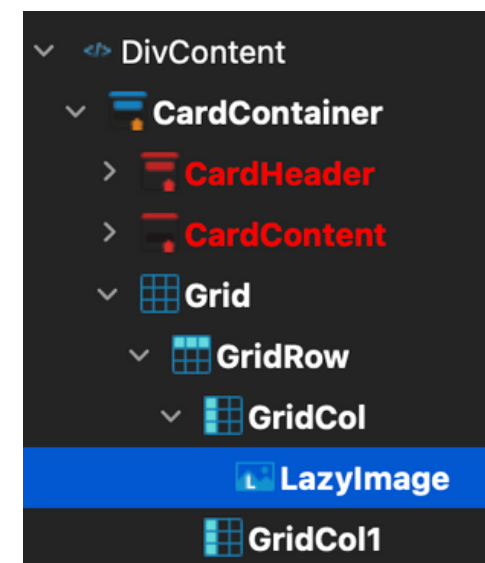
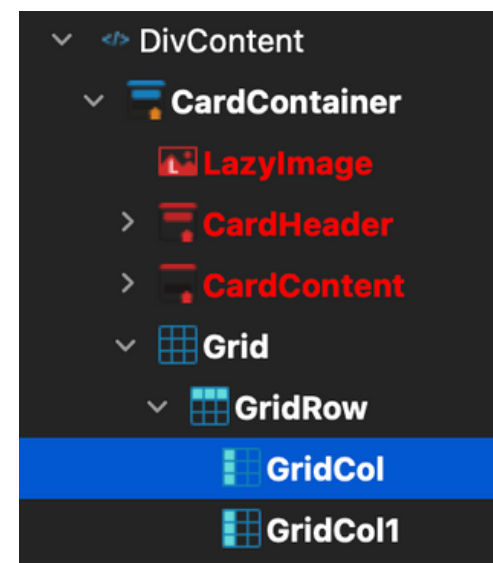
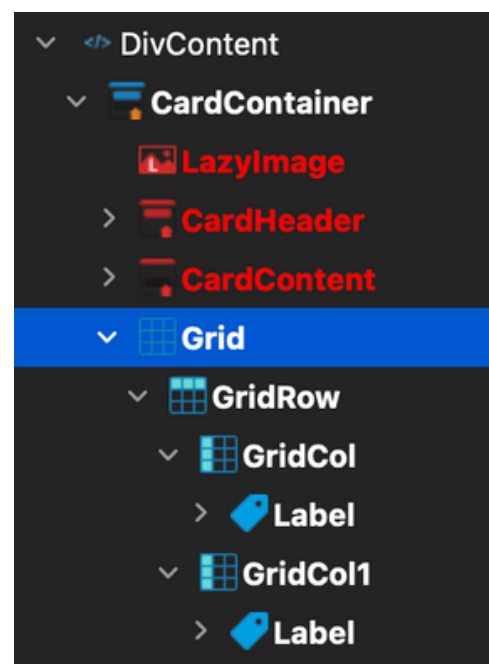
Then add a **Grid**
in the **CardContainer**.



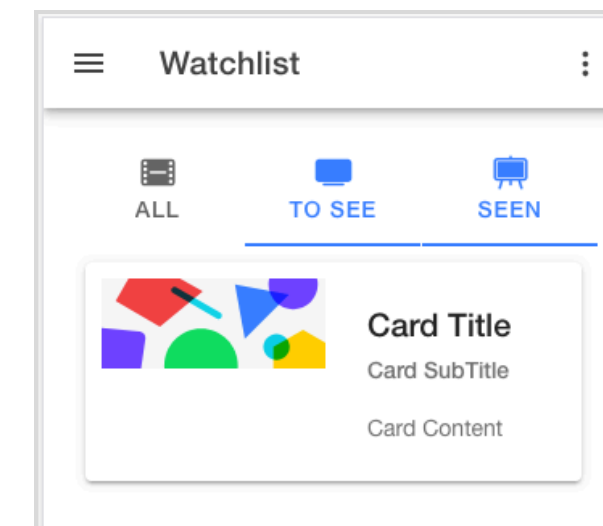
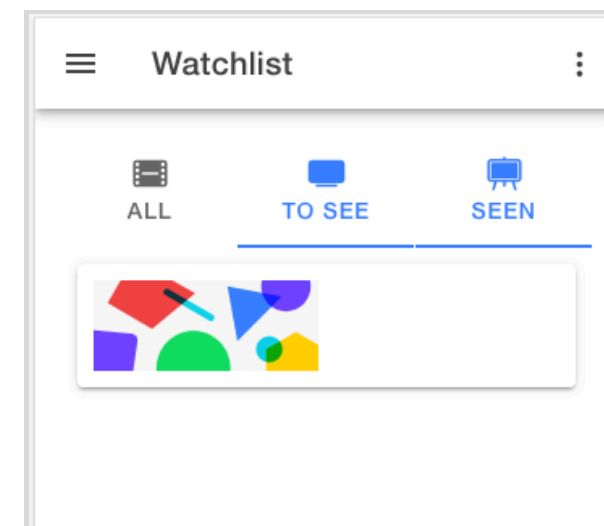
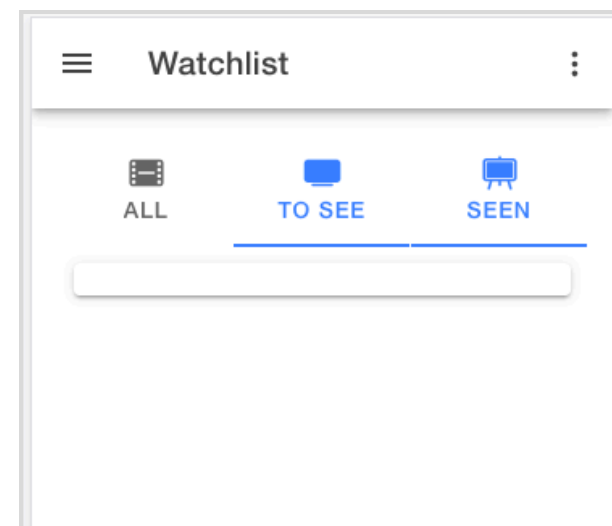
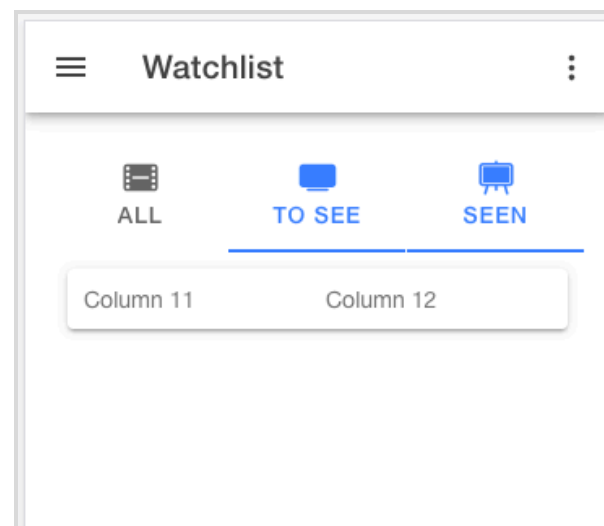
3.5 Cards & Grid



In the **GridCols** of the **Grid**,
remove the **Labels**.



Rename the first **GridCol**
as **GridColIMG**,
and the second
as **CardHeaderContent**.



3.6 Responsive with Grid

As seen before, the **Grid** is based on a 12 column layout
where columns expand to fill the row,
and resize to fit additional columns.

3 cols	1 2 3
6 cols	1 2 3 4 5 6
12 cols	1 2 3 4 5 6 7 8 9 10 11 12

More info on

<https://ionic-docs-o31kiyk8l-ionic1.vercel.app/docs/api/grid>

This layout has different breakpoints
based on the screen size.

	Set columns when min-width =
xs	0
sm	576px
md	768px
lg	992px
xl	1200px

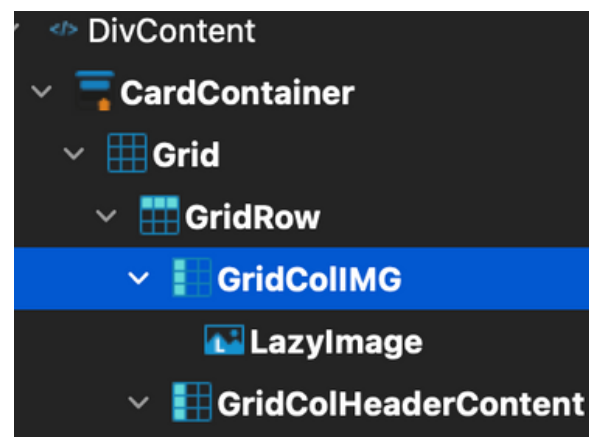


3.6 Responsive with Grid

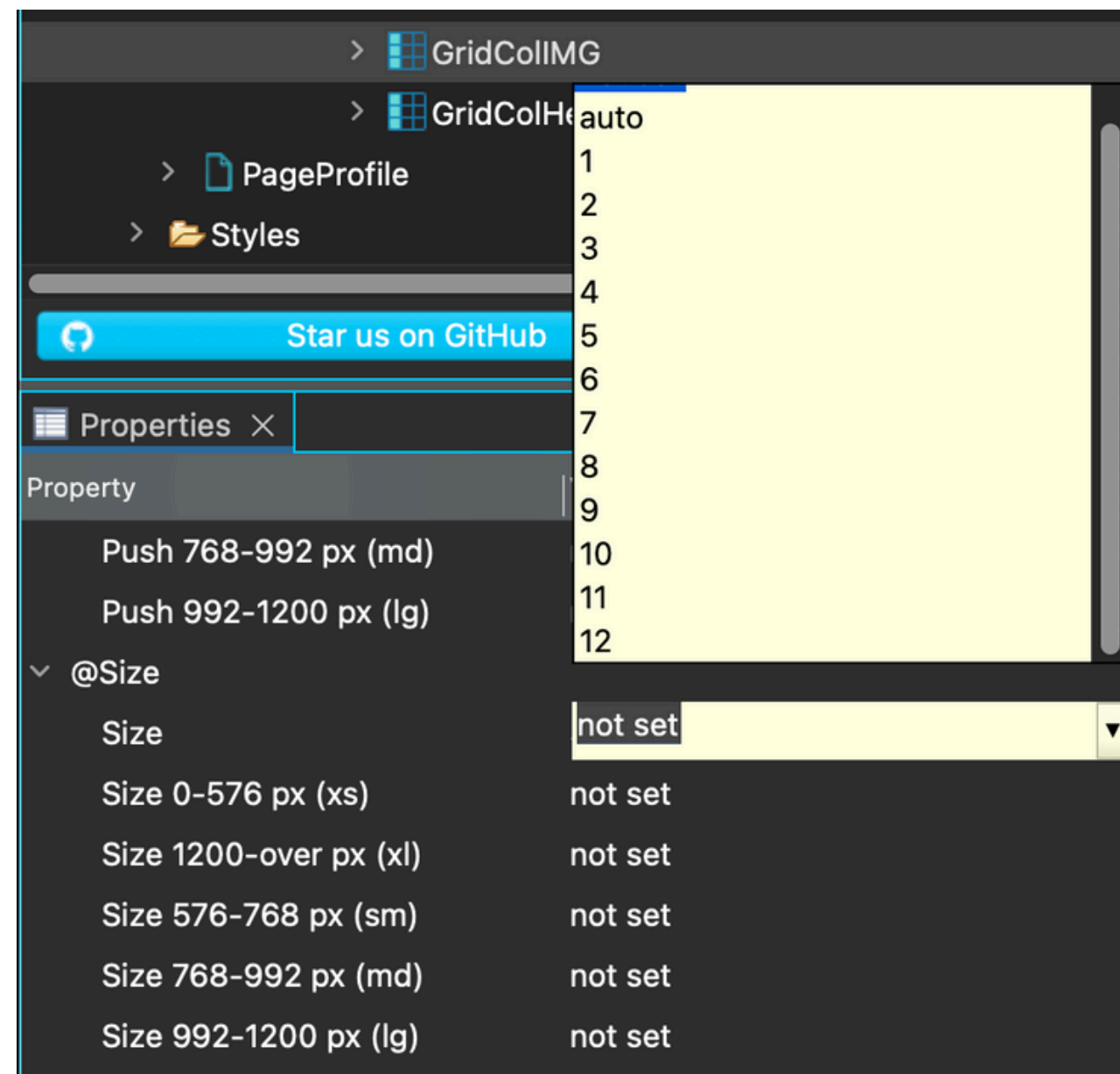
In **GridColIMG's** properties, the **Size property** sets the size of the column, (how many columns it should take up out of 12).

If **auto** is set, the column will be the size of its content.

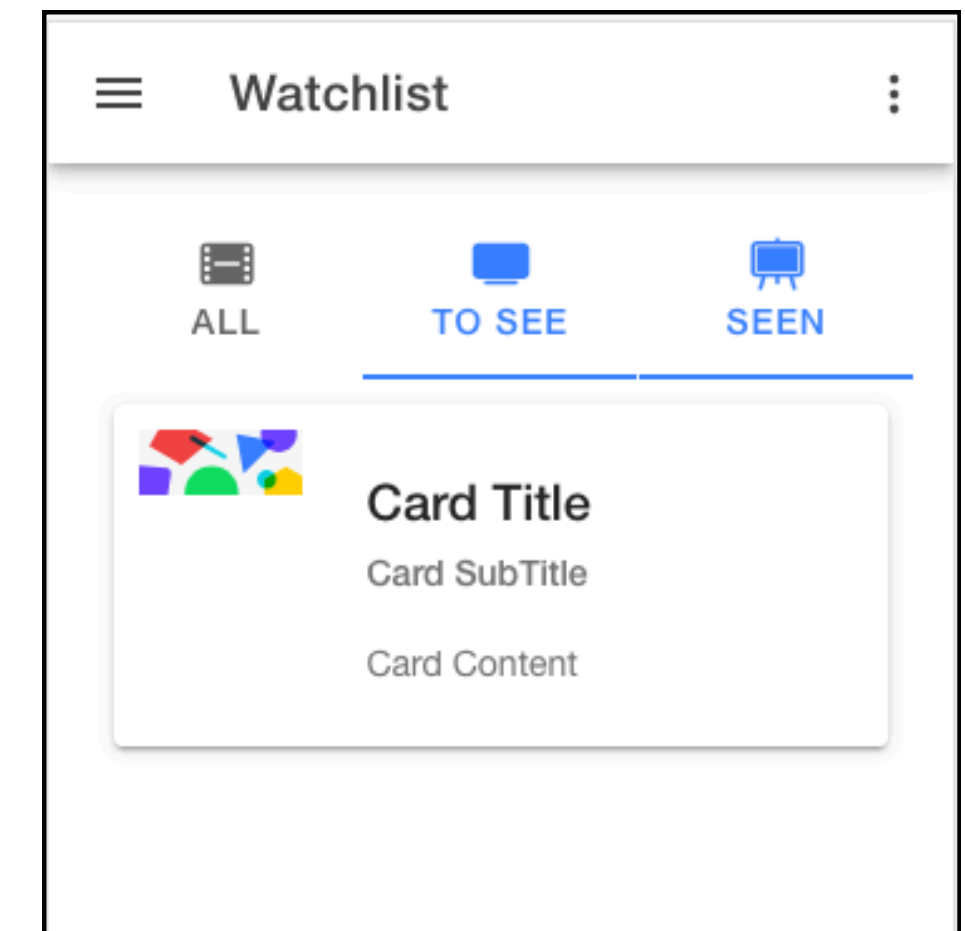
To have the **ColIMG** to take 25% of the space in the **Card**, set the **Size property** to 3.



Properties	
Property	Value
@Size	
Size	not set
Size 0-576 px (xs)	not set
Size 1200-over px (xl)	not set
Size 576-768 px (sm)	not set
Size 768-992 px (md)	not set
Size 992-1200 px (lg)	not set

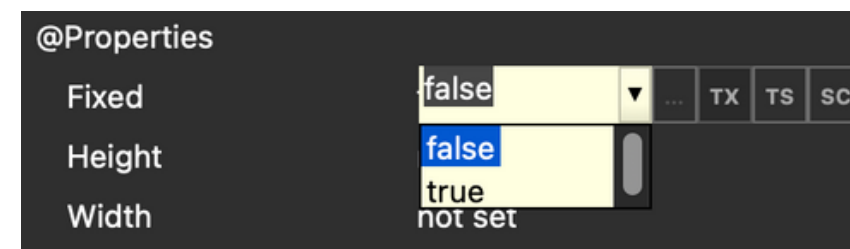
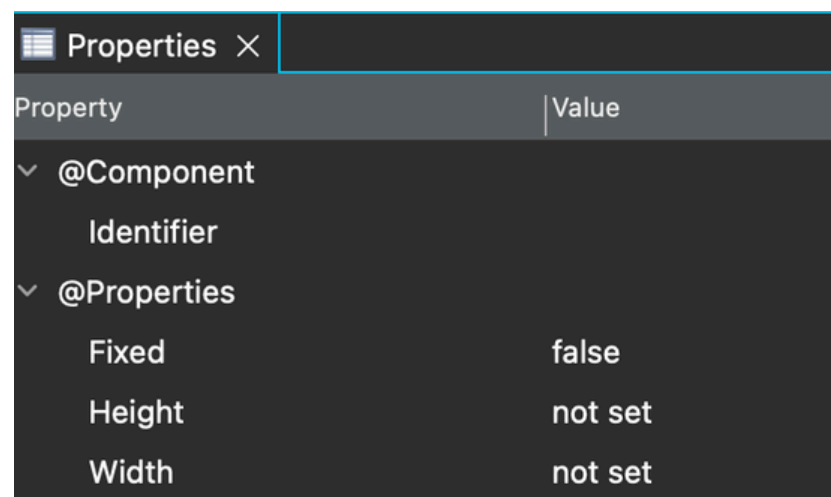
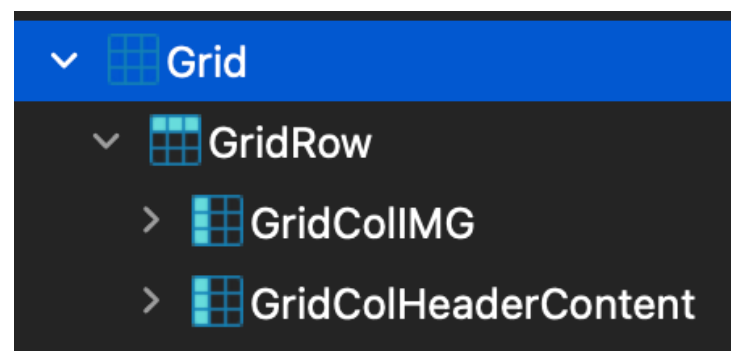


@Size	
Size	3
Size 0-576 px (xs)	not set



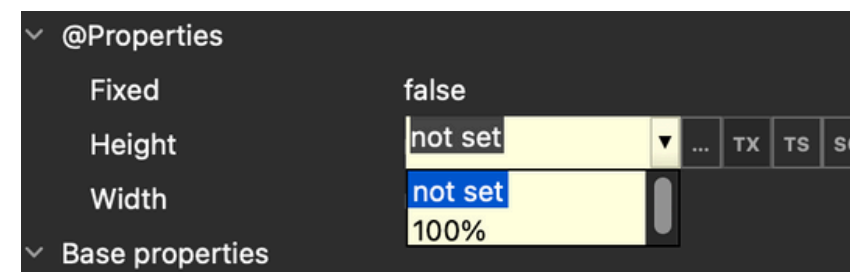
3.7 Flexbox & Alignment with Grid

The **Grid component** is based on **flexbox**, and acts as a flex-container for **GridRows**.
In its properties, you can customize its height and width.



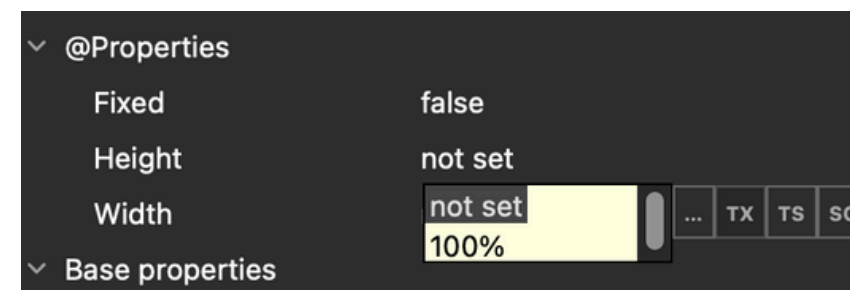
Fixed

Set a max width based on the screen size.



Height

Sets the height of the **Grid**, pixel (px) or percentage (%).
Use 100% to set a grid full screen,
usefull for vertical centering.



Width

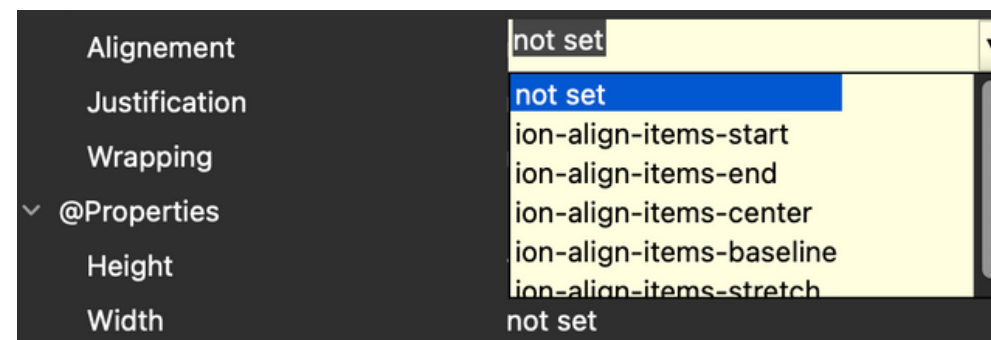
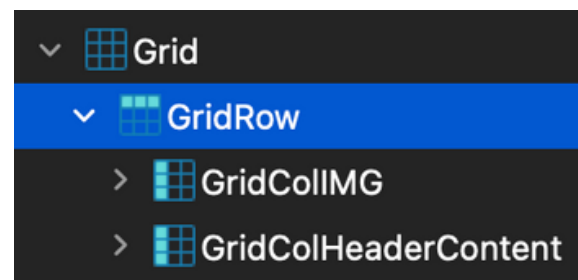
Sets the width of the **Grid**, pixel (px) or percentage (%).



3.7 Flexbox & Alignment with Grid

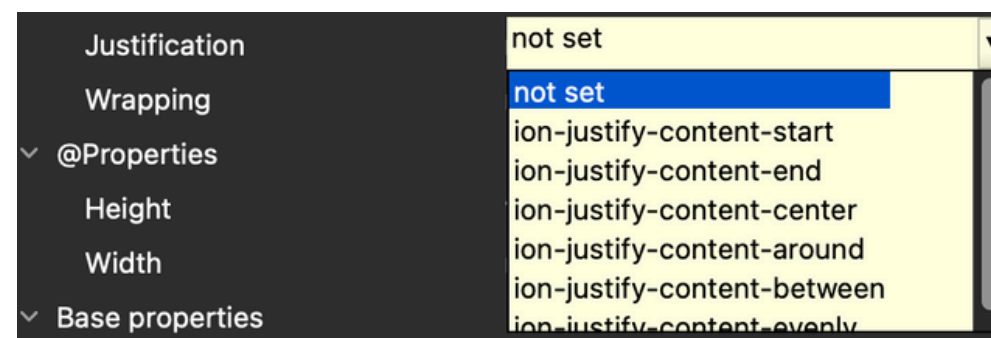
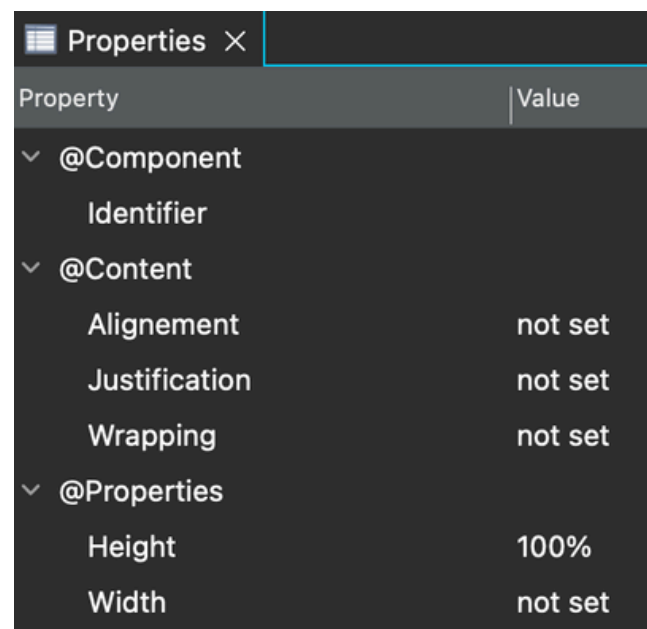
The **GridRow** component is a flex container.

In its properties, you can customize how the items inside it are positioned.



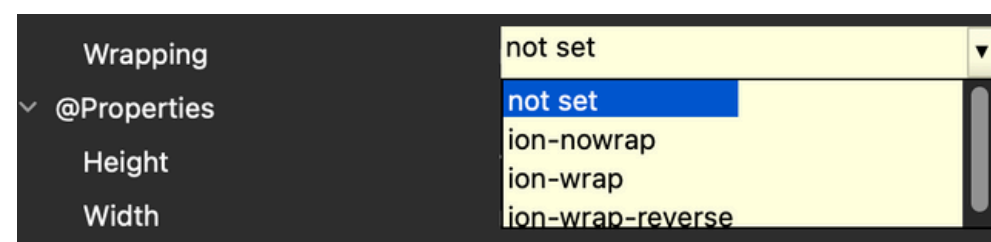
Alignement

Sets how items are aligned on the cross axis.



Justification

Sets how items are packed on the main axis.



Wrapping

Sets how items are wrapped on line(s).

Height: Sets the height of the row, pixel (px) or percentage (%).

Use 100% to set a row taking the full grid height, usefull for vertical centering.

Width: Sets the width of the row, pixel (px) or percentage (%).

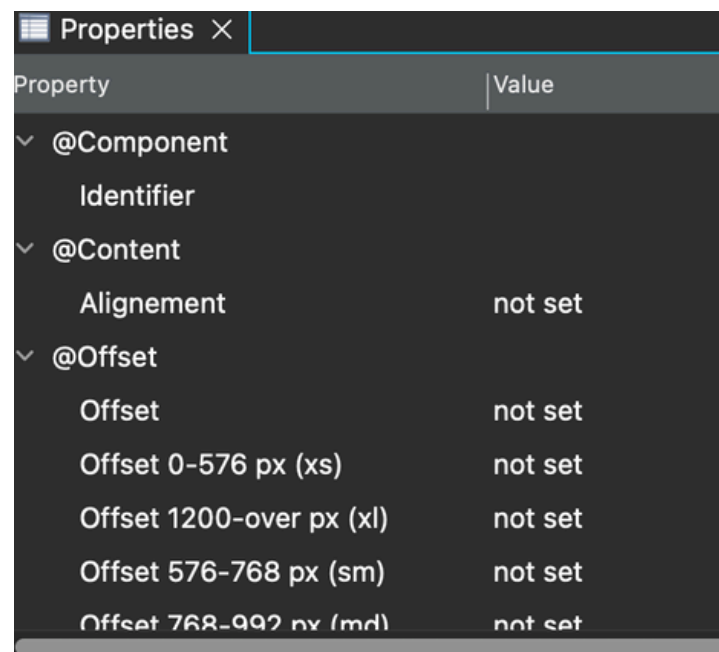
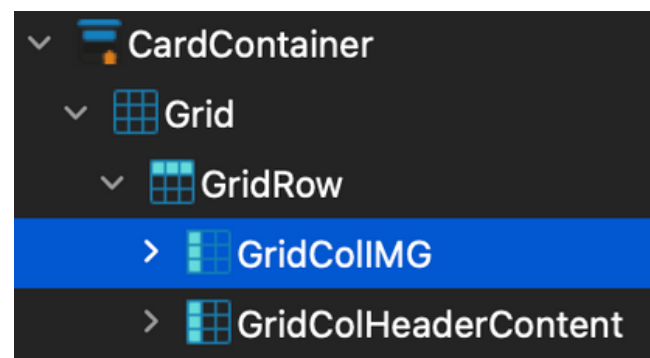


3.7 Flexbox & Alignment with Grid

By default, columns will stretch to fill the entire height of the row.

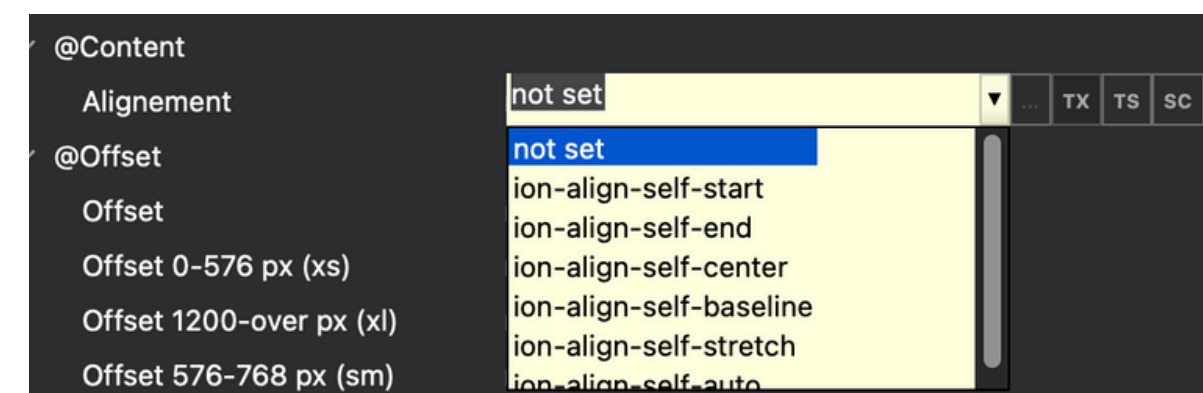
The **GridColumn** is a flex item.

In its properties, you can customize how the items inside it are positioned.



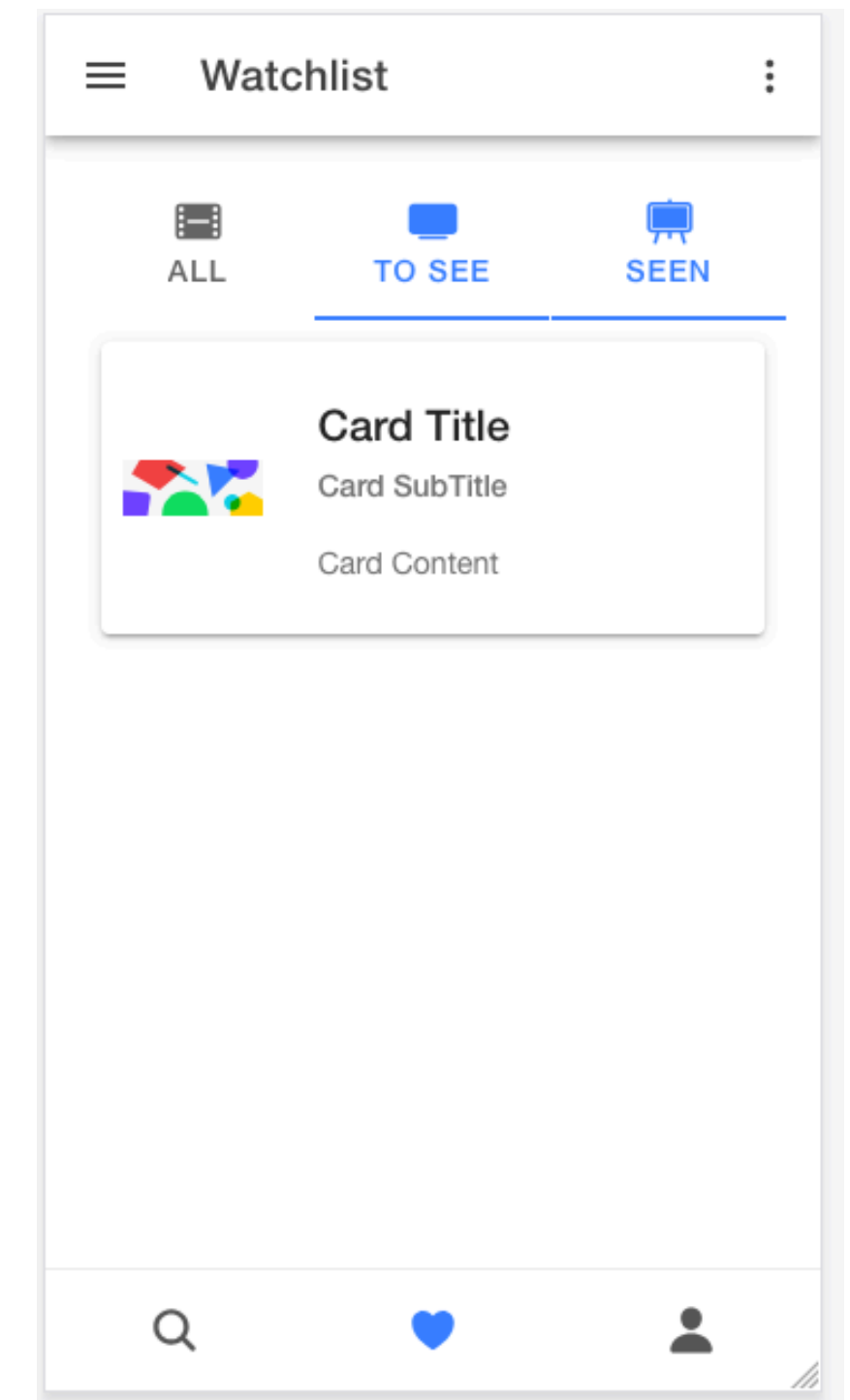
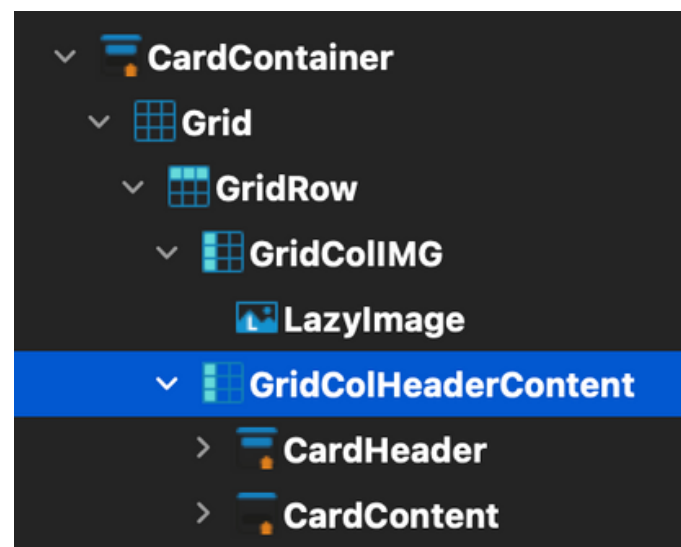
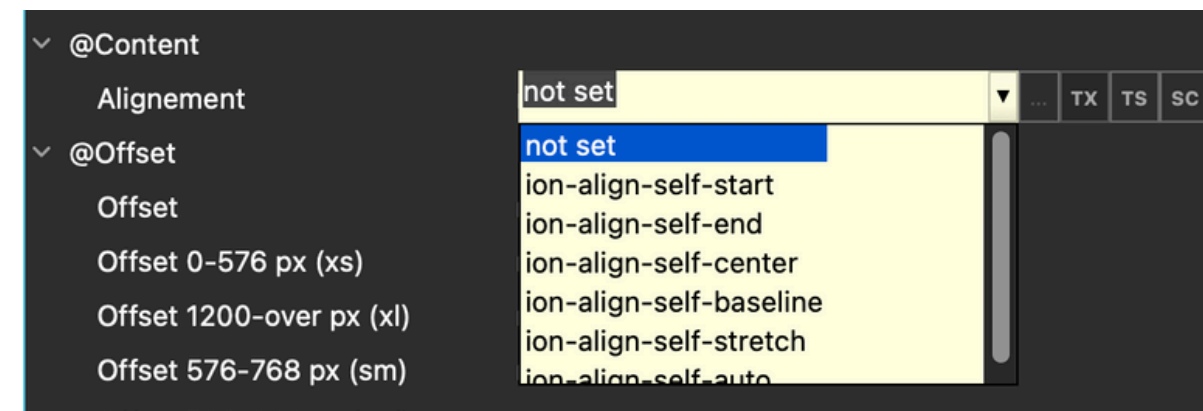
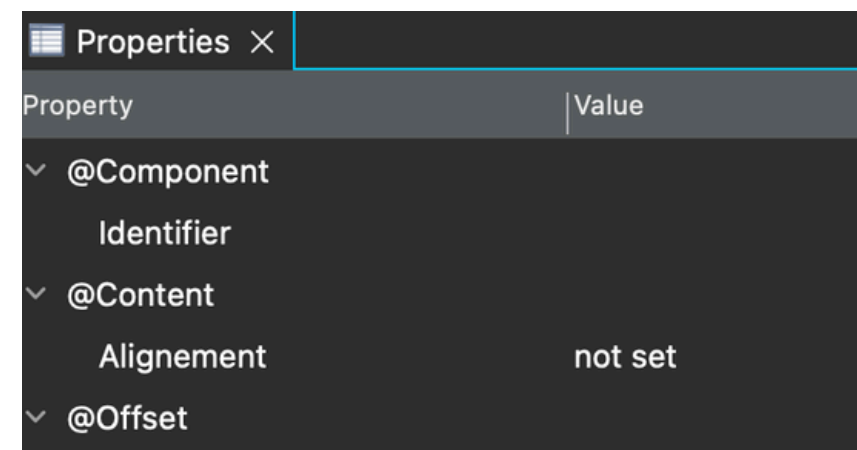
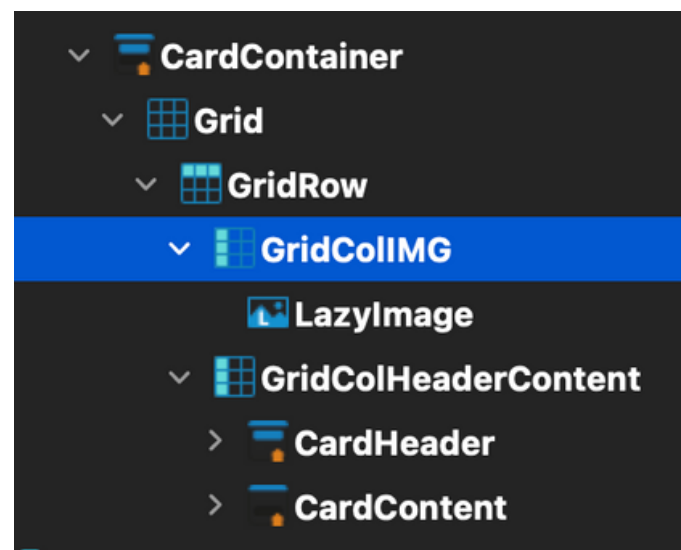
Alignement

Sets how the items are aligned on the cross axis.

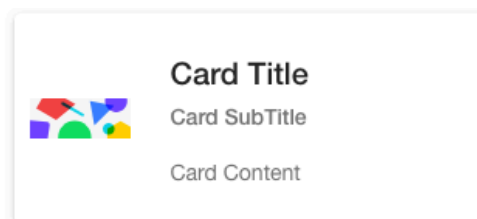
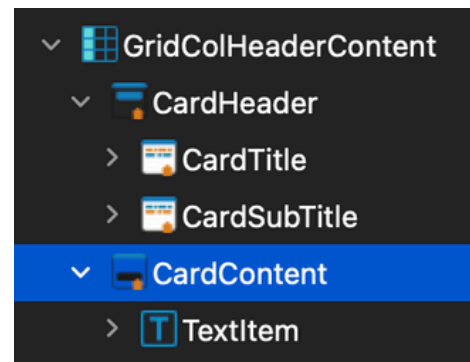


3.7 Flexbox & Alignment with Grid

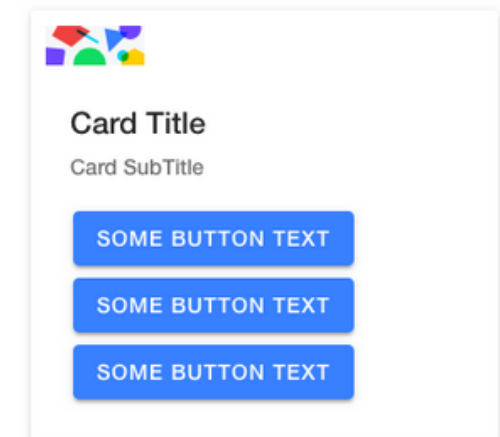
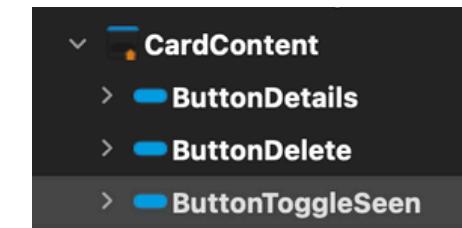
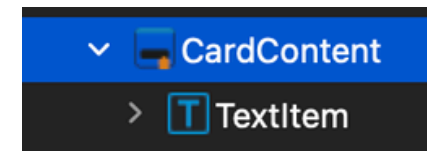
To have its content vertically centered in the column,
in **GridColIMG**'s and **CardHeaderContent**'s properties,
set the **Alignement** property to **ion-align-self-center**.



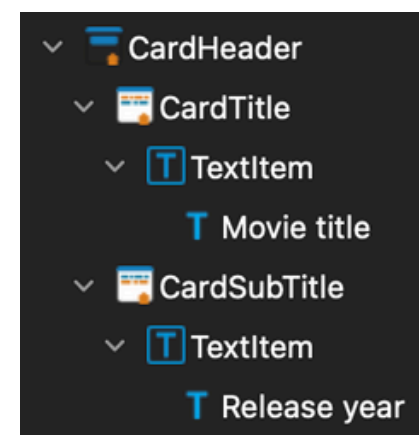
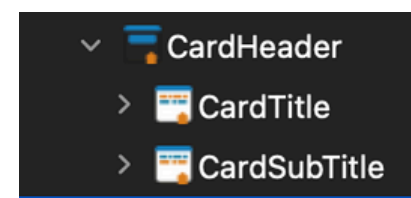
3.8 Finalize the card layout



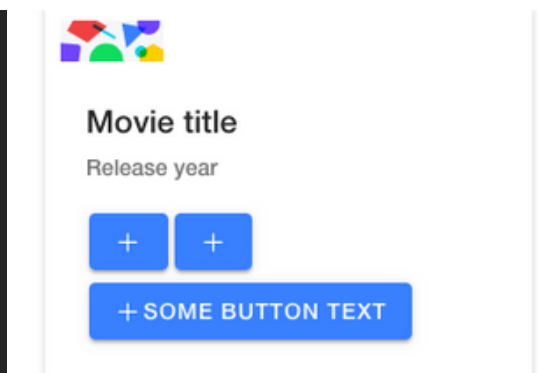
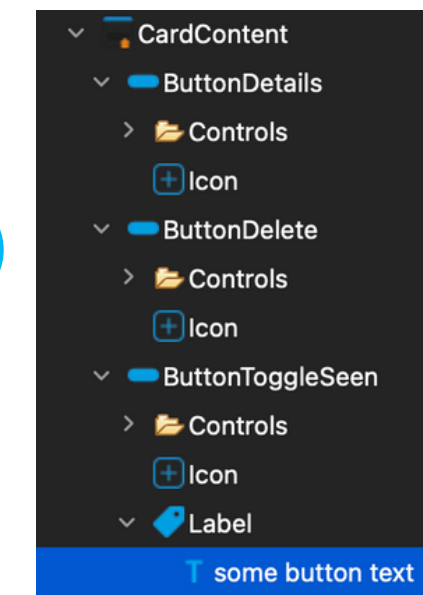
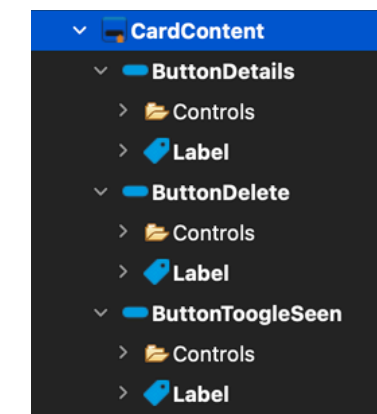
In **Card Content**, replace **TextItem** by 3 buttons :
ButtonDetails, **ButtonDelete**, **ButtonToggleSeen**



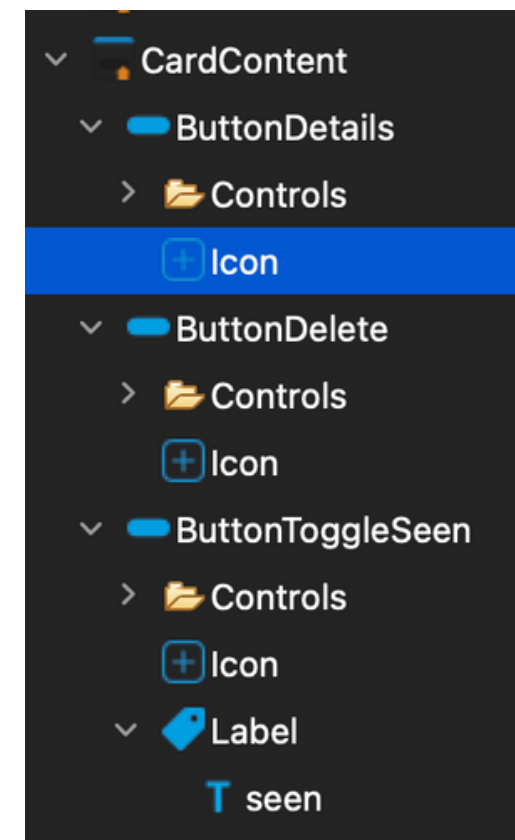
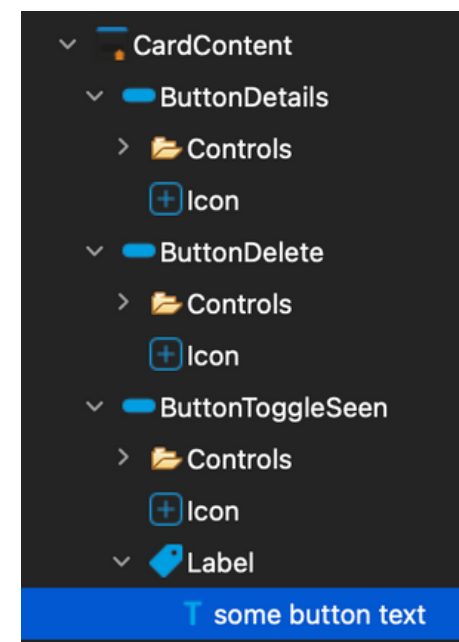
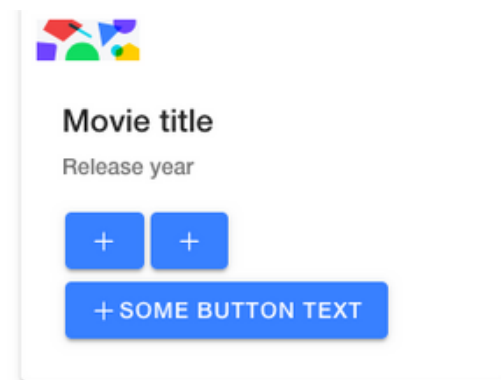
In **CardHeader**, in the **TextItem**,
replace the **value** of the **Text** element
by **Movie title** in **CardTitle**
by **Release year** in **CardSubTitle**



In each button , add an **Icon** and delete the **Label**
in **ButtonDetails** and **ButtonDelete**



3.8 Finalize the card layout

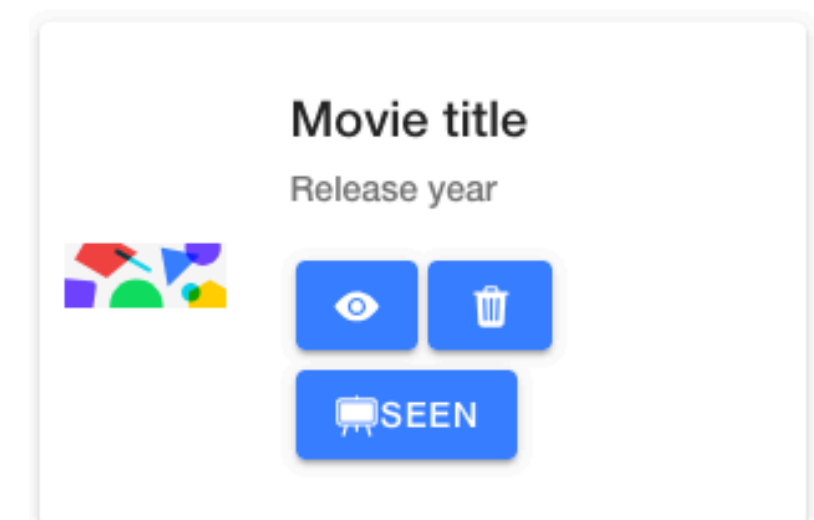
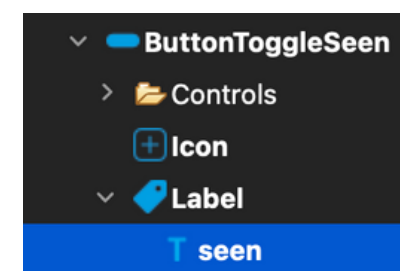
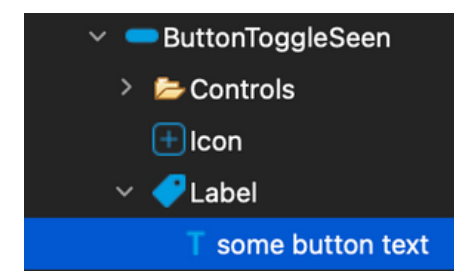


In **Card Content**, for each button, in **Properties** replace the **Icon name** by :

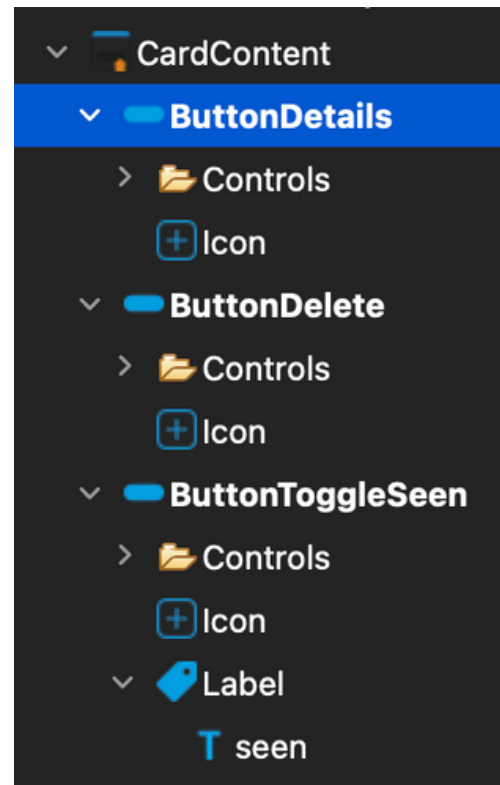
- 'eye' for **ButtonDetails**,
- 'trash' for **ButtonDelete**,
- 'easel' for **ButtonToggleSeen**



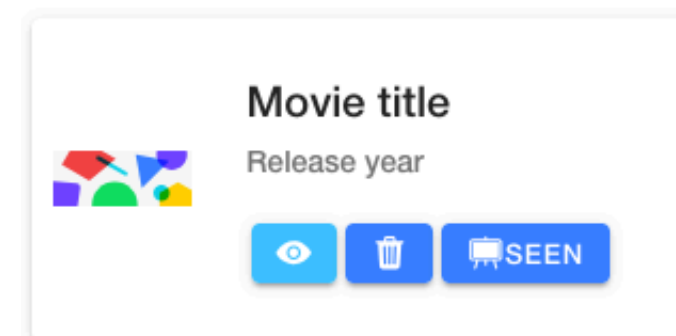
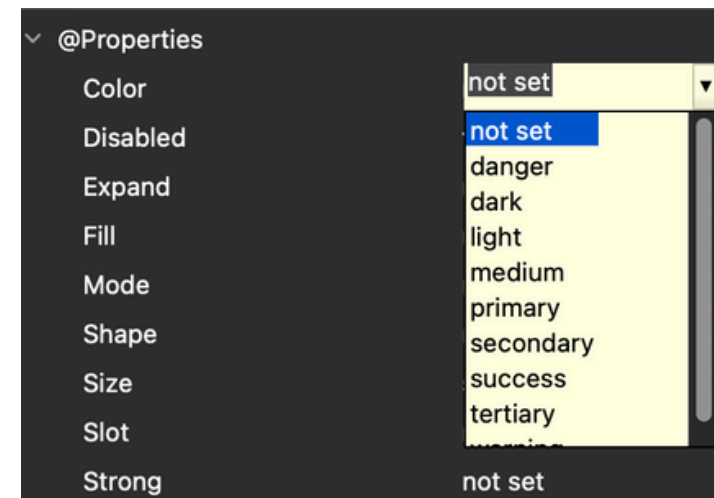
In **ButtonToggleSeen**, in the **Label**, replace the **value** of the **Text** element by **seen**.



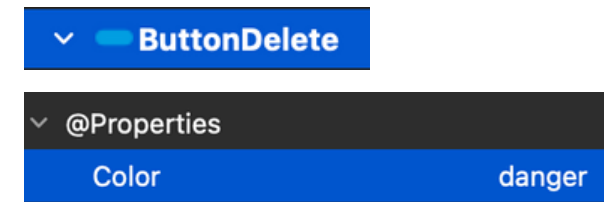
3.8 Finalize the card layout



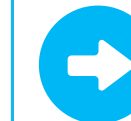
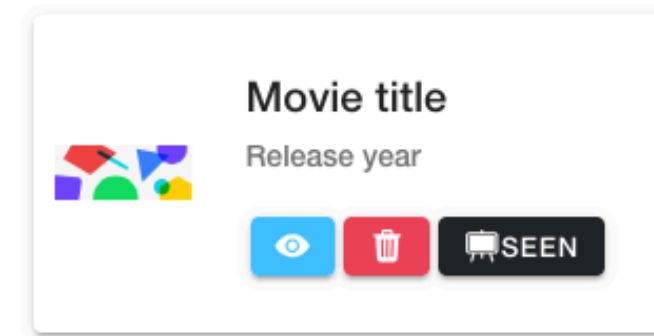
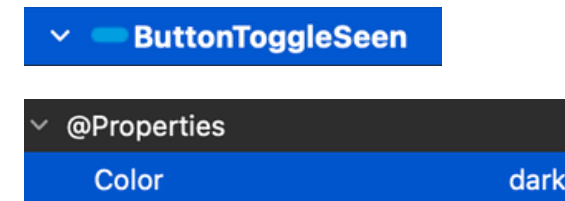
In **ButtonDetail**,
set the **Color** property
to **secondary**.



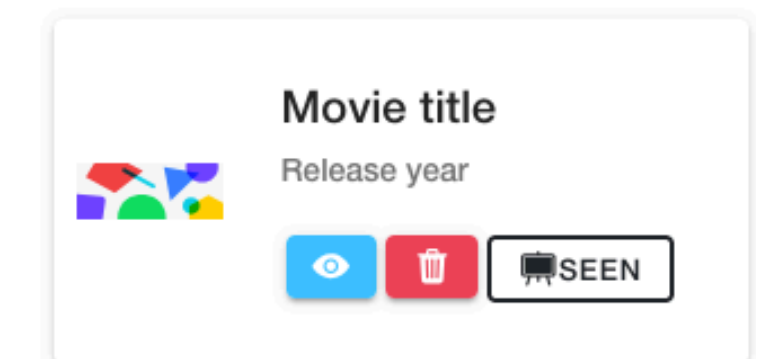
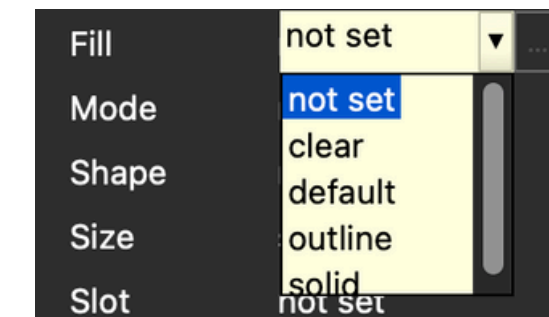
In **ButtonDelete**,
set the **Color** property
to **danger**.



In **ButtonToggleSeen**,
set the **Color** property
to 'dark'.



In **ButtonToggleSeen**,
set the **Fill** property
to **outline**.



3.8 Finalize the card layout

In **CssUtilities.scss**,
add the following classes:

```
.padding-left-8 {
padding-left: 8px;
}

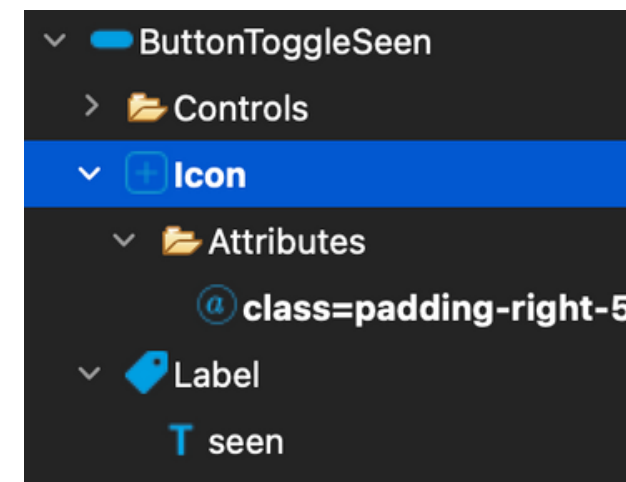
.padding-right-5 {
padding-right: 5px;
}

.padding-right-0 {
padding-right: 0;
}

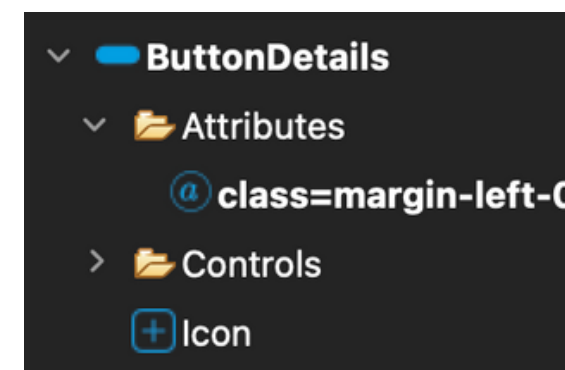
.margin-left-0 {
margin-left: 0;
}
```

```
CssUtilities.scss x
1 // Limit text's length to 2 lines
2 .clamp-2-lines {
3   display: -webkit-box;
4   -webkit-line-clamp: 2;
5   -webkit-box-orient: vertical;
6 }
7
8 .overflow-hidden {
9   overflow: hidden !important;
10 }
11
12 .flex-column {
13   display: flex;
14   flex-direction: column;
15 }
16
17 .padding-left-8 {
18   padding-left: 8px;
19 }
20
21 .padding-right-5 {
22   padding-right: 5px;
23 }
24
25 .padding-right-0 {
26   padding-right: 0;
27 }
28
29 .margin-left-0 {
30   margin-left: 0;
31 }
```

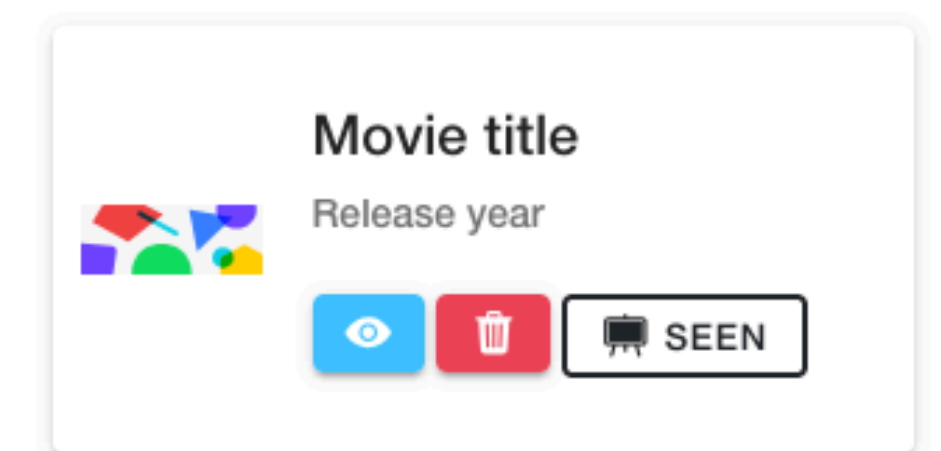
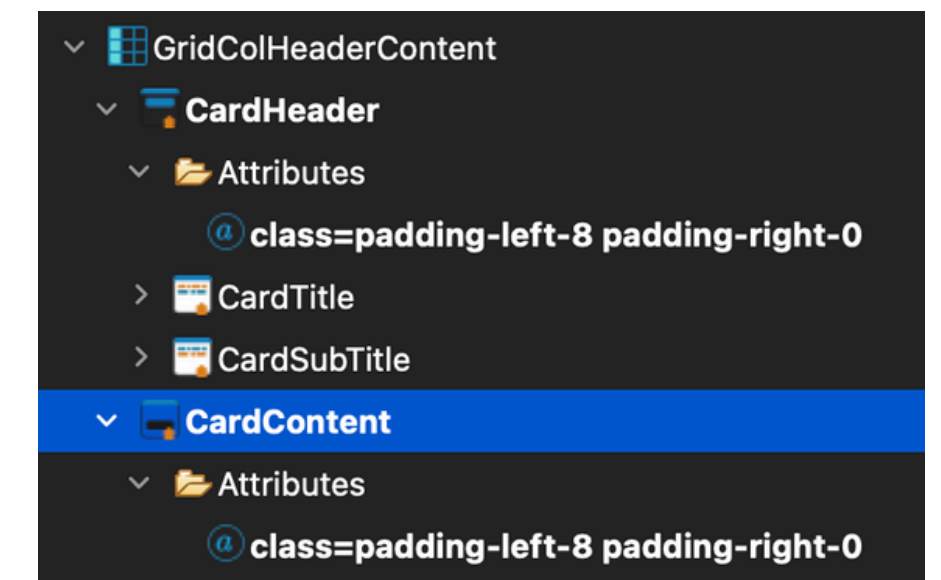
Add class **padding-right-5**
to the **Icon** in **ButtonToggleSeen**.



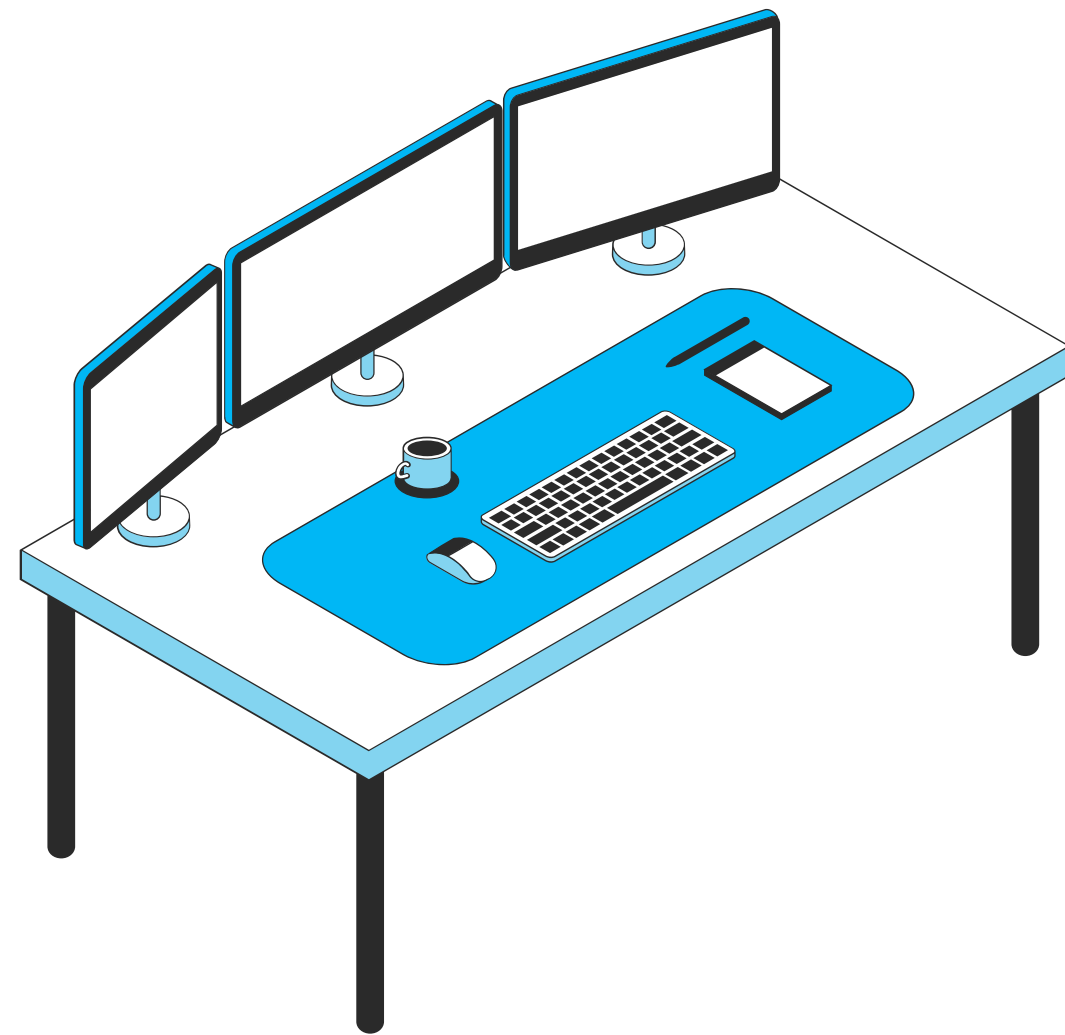
Add class **margin-left-0**
to **ButtonDetails**.



Add classes **padding-left-8**
and **padding-right-0**
to **CardHeader** and **CardContent**.



4 – Shared Components



4.1 Add a shared component

4.2 Use a shared component

4.3 Add variables

4.4 Bind the variables

4.5 Import variables in UseSharedComponent

4.6 Add a value to UseVariable

4.7 Create a header as sharedComponent

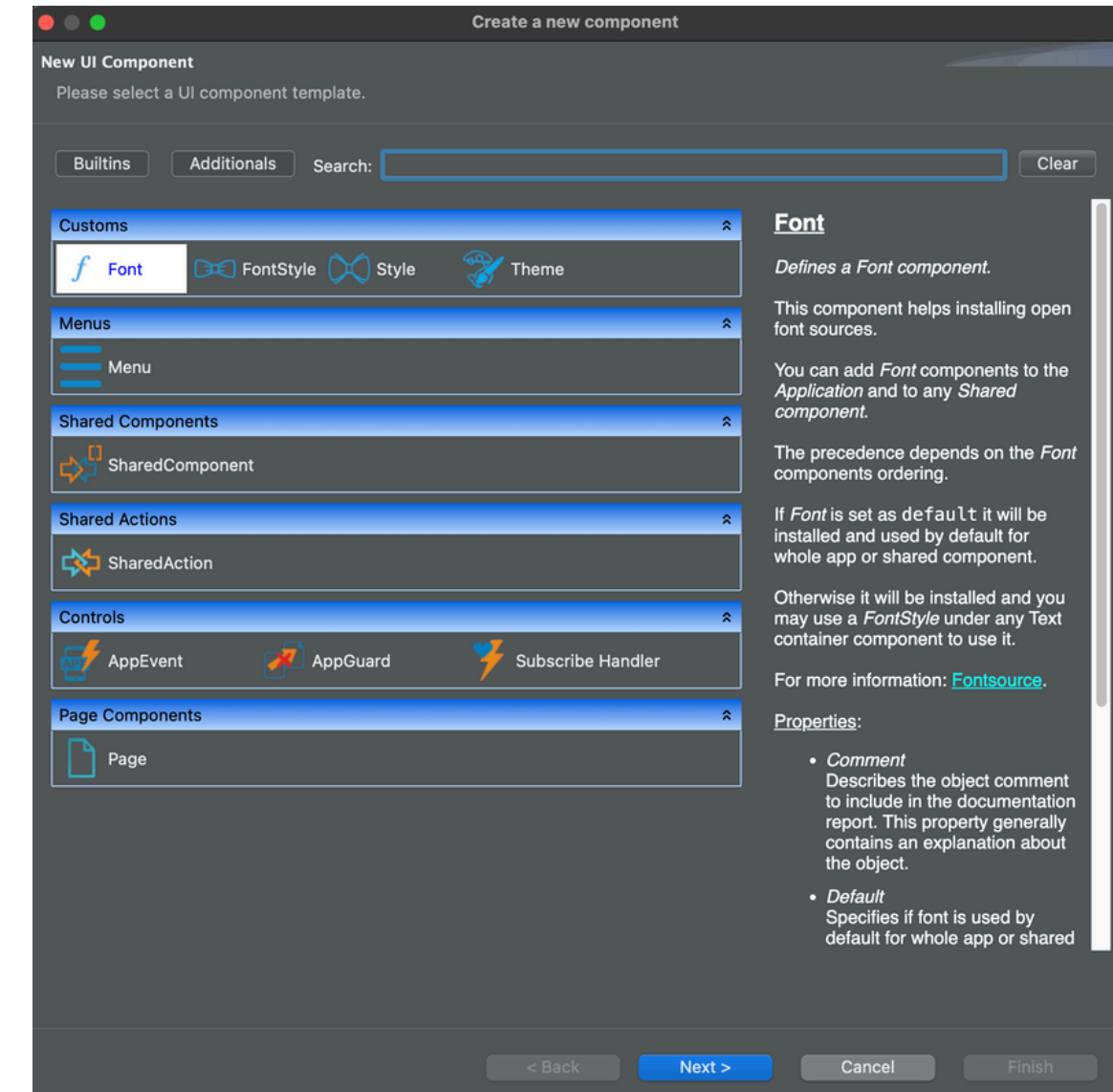
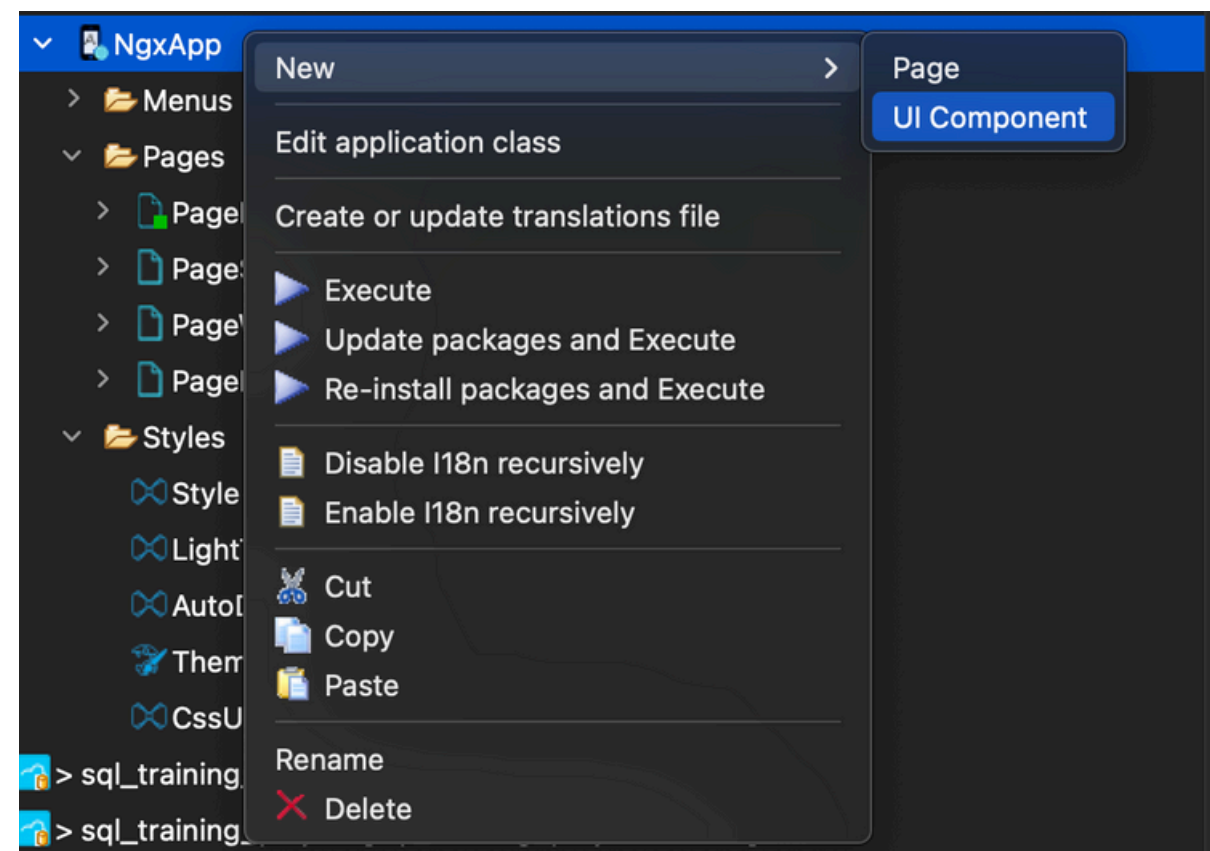
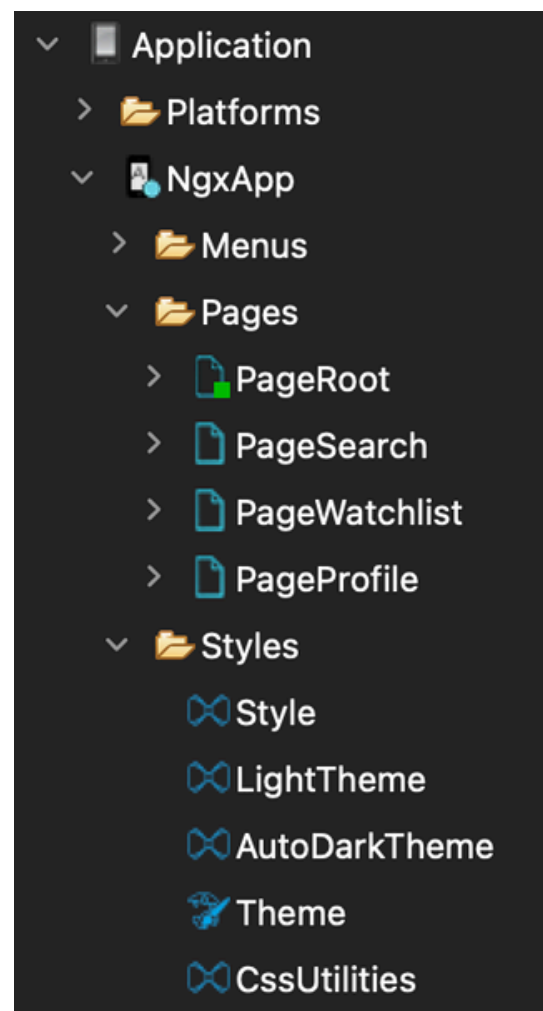
4.1 Add a shared component

In **PageWatchlist**, we have created a **card component**.

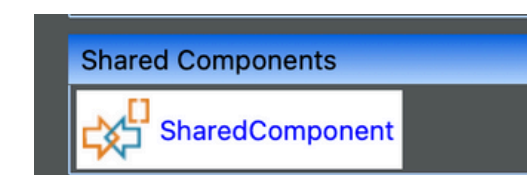
We want to use it in the 3 **segments** without copying it in each segment.

To that end, we are going to create a **shared component**.

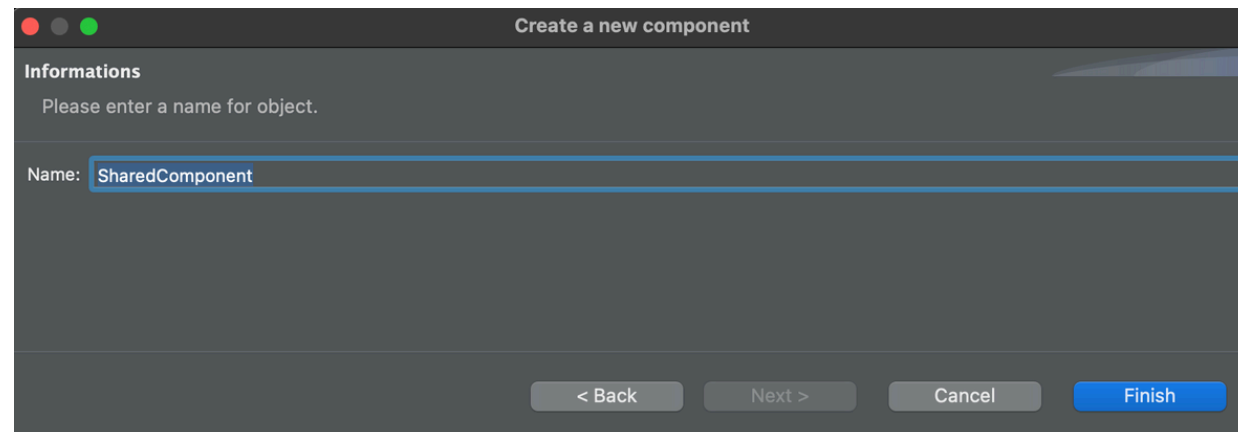
Click on **NgxApp**
then select **New** and **UI Component**
to open the **Create a new component panel**.



In the **Create a new component panel**,
click on **SharedComponent**, then on **Next**.



4.1 Add a shared component



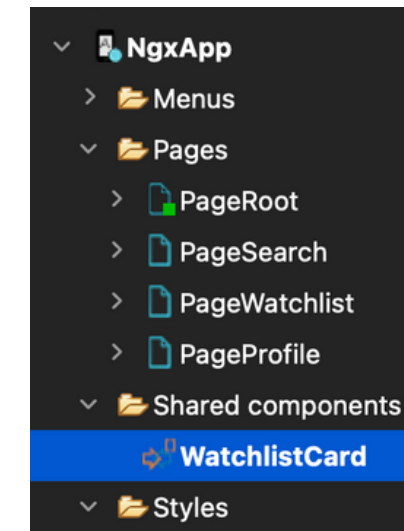
Create a new component

Informations

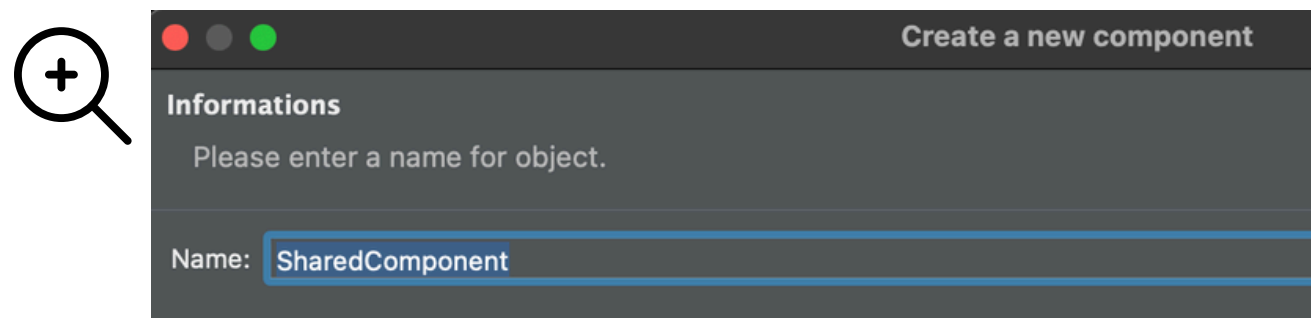
Please enter a name for object.

Name: SharedComponent

< Back Next > Cancel Finish



A new folder, **Shared components**, appears in the treeview with **WatchlistCard**, the shared component inside.



Create a new component

Informations

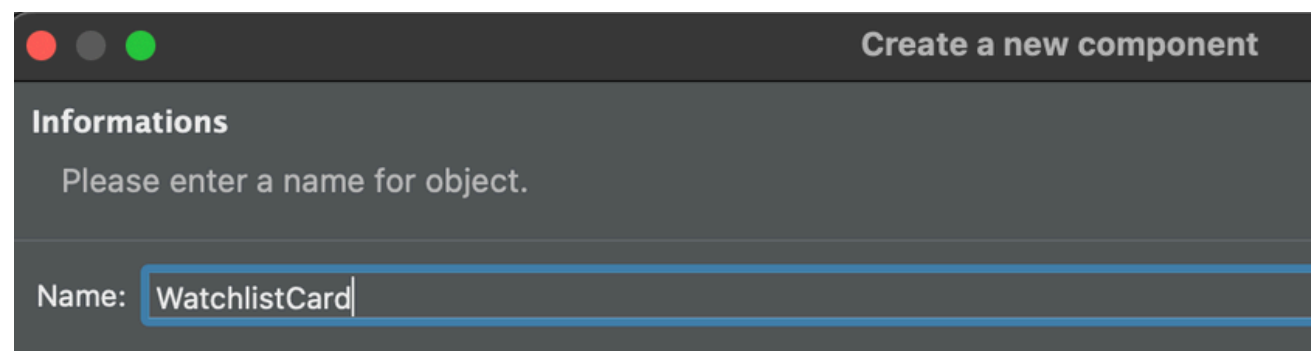
Please enter a name for object.

Name: WatchlistCard

< Back Next > Cancel Finish



Enter **WatchlistCard** as name for the component, then click on **Finish**.



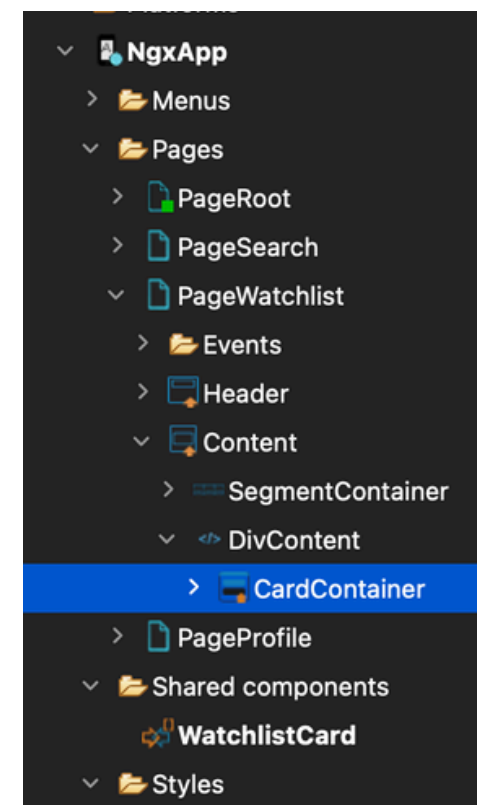
Create a new component

Informations

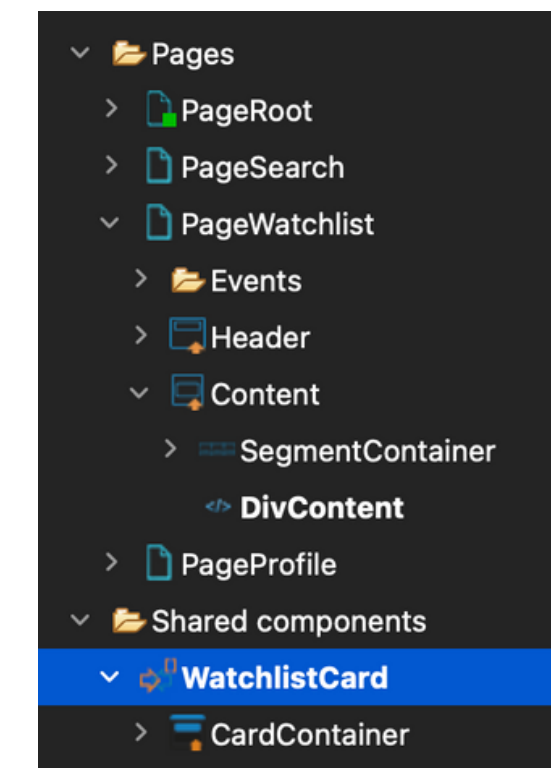
Please enter a name for object.

Name: WatchlistCard

< Back Next > Cancel Finish

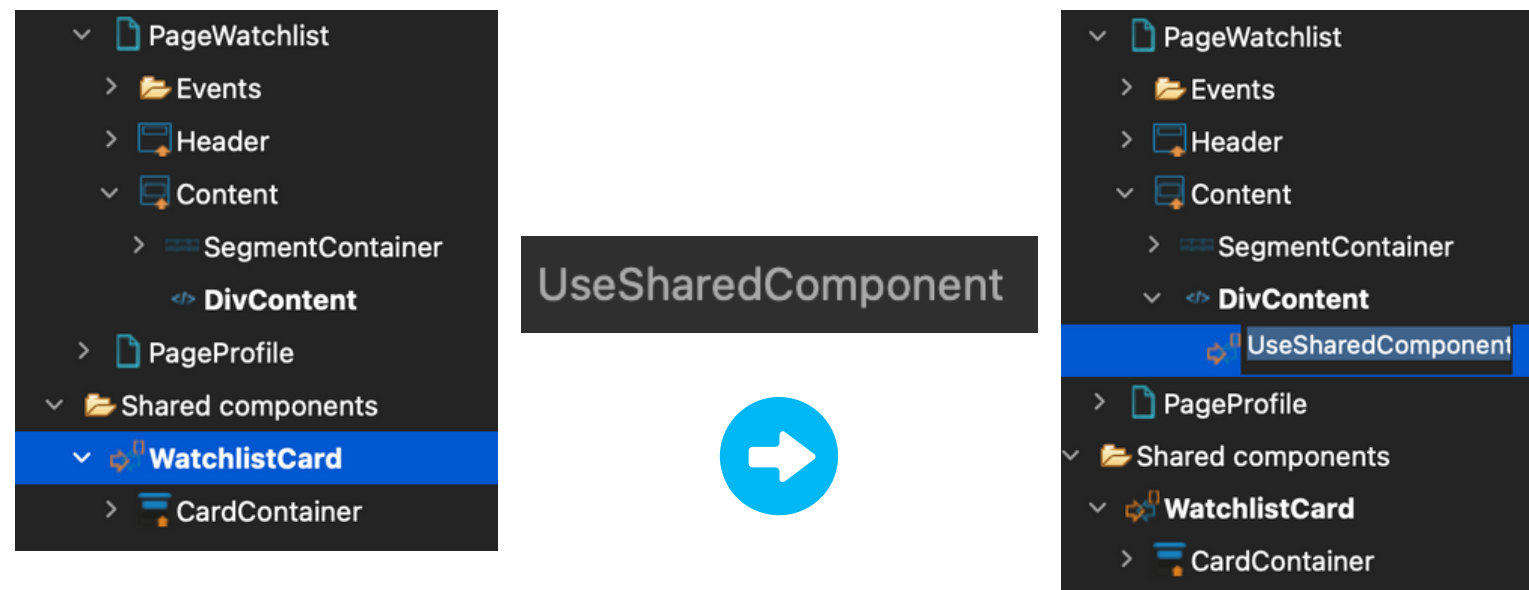


Now let's drag the **Card container** from the **DivContent** of **PageWatchlist** and drop it inside **WatchlistCard**.

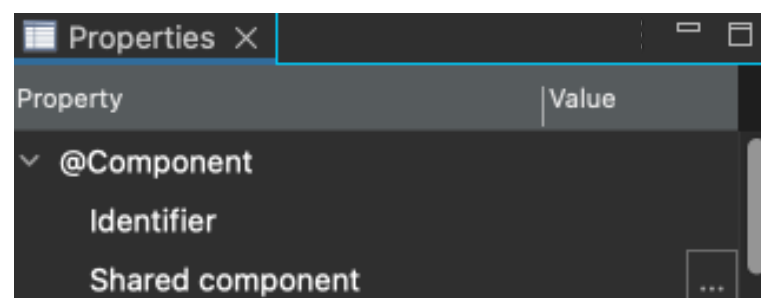


4.2 Use a shared component

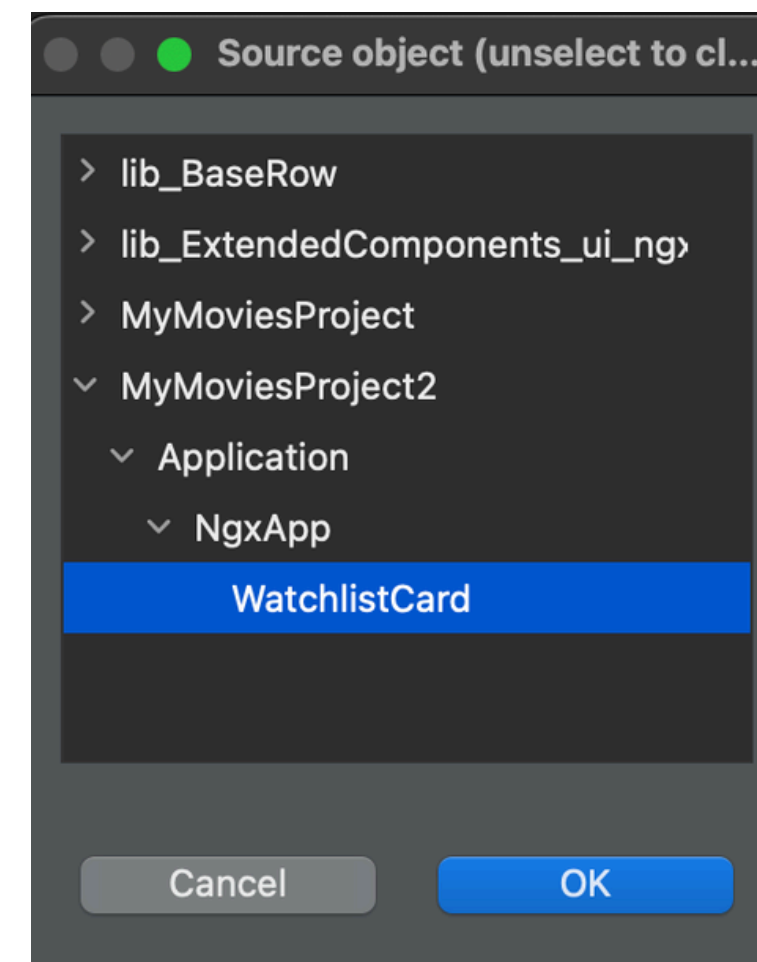
To use **WatchlistCard** in **PageWatchlist**, as it was before drag the component **UseSharedComponent** from the palette and drop it inside **DivContent**.



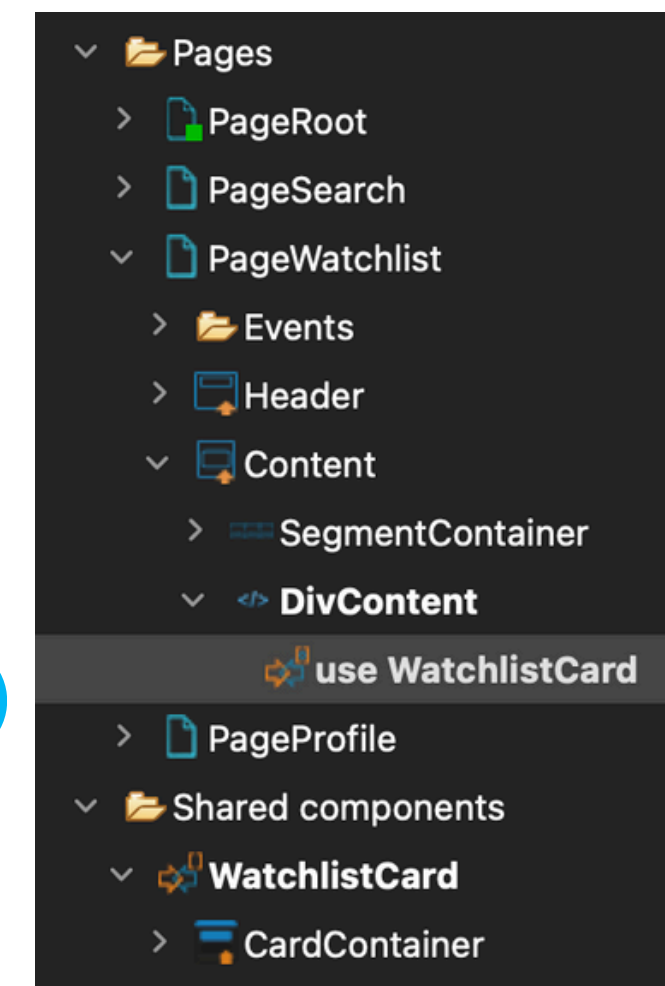
In **UseSharedComponent**'s properties click on ... to open the **Source object** panel.



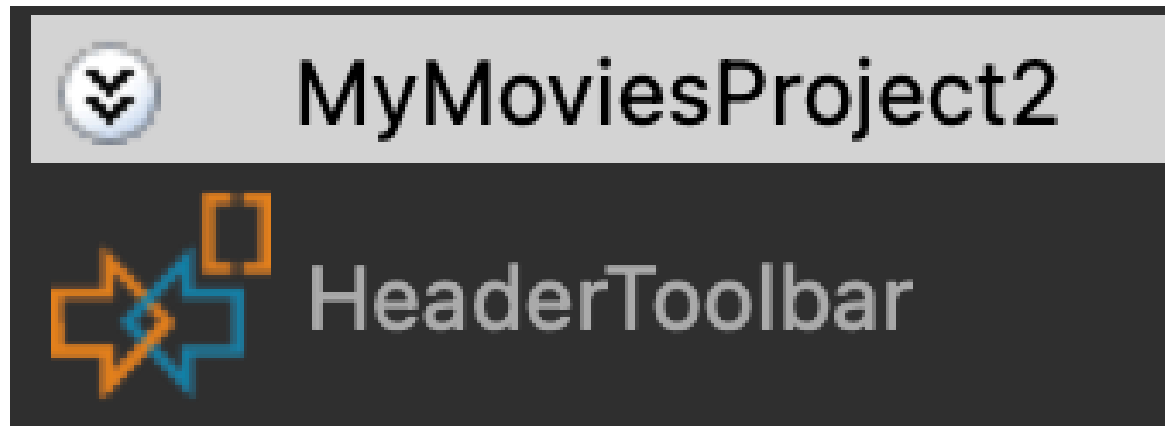
In the **Source object** panel, select **WatchlistCard**.



In the treeview, the **UseSharedComponent** appears as **useWatchlistCard**.



4.2 Use a shared component

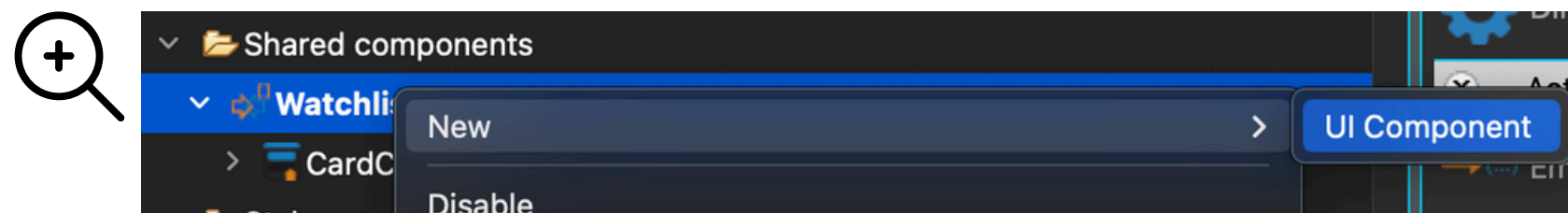
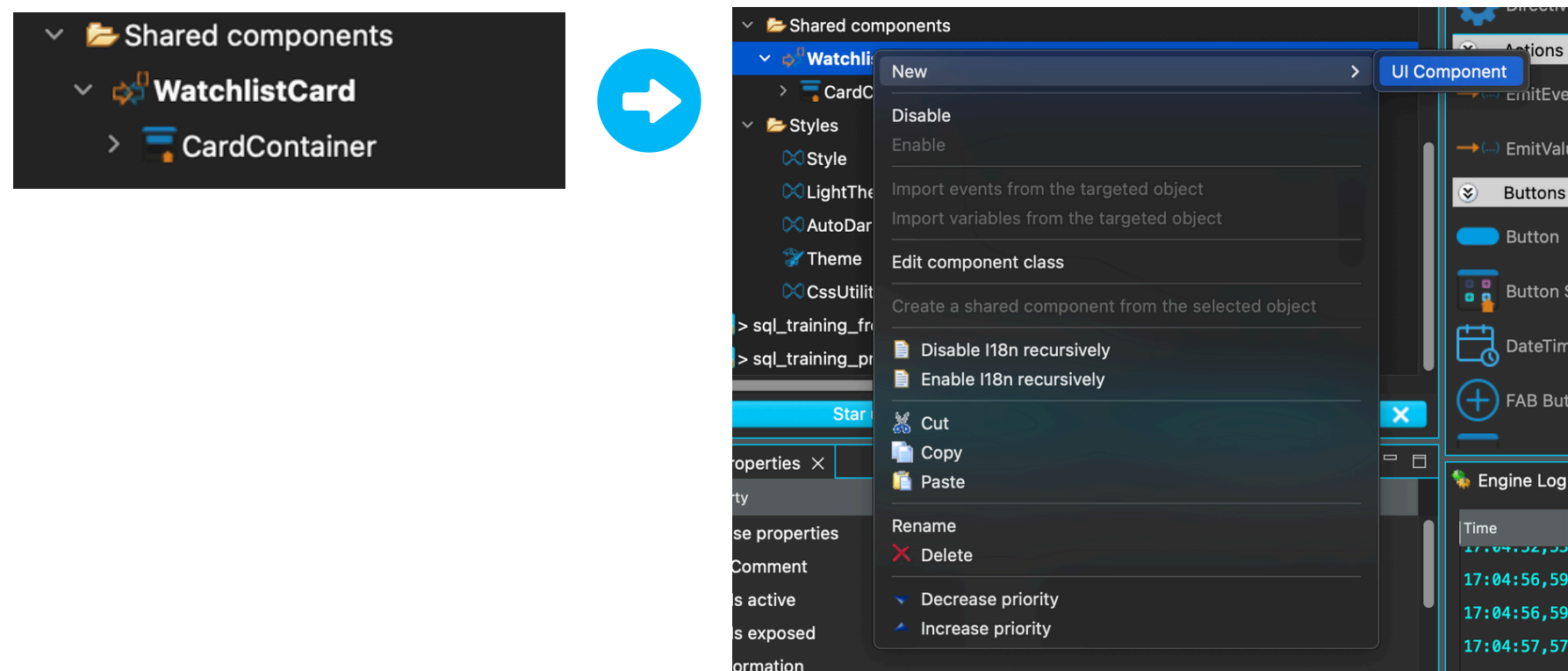


When a **shared component** is created,
it becomes **available** in the **palette**
for **every project** in the **workspace**.

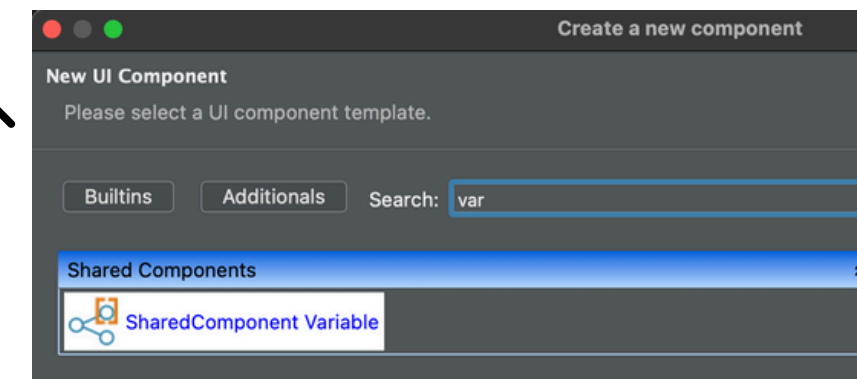


4.3 Add variables

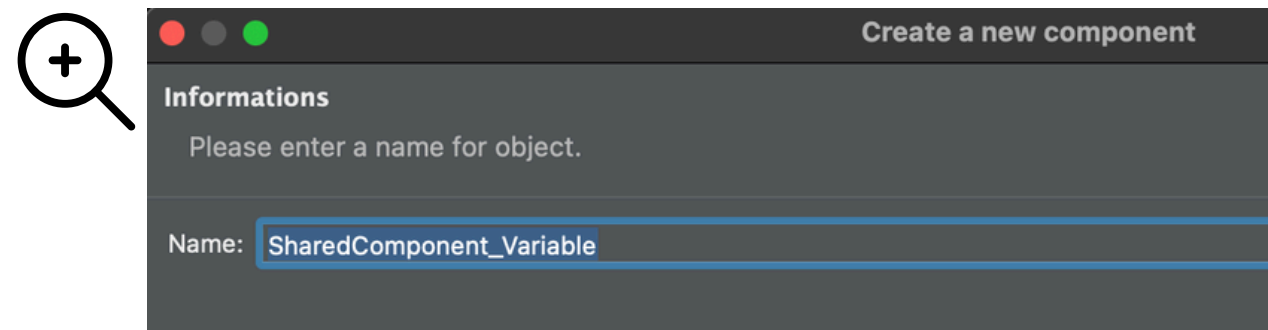
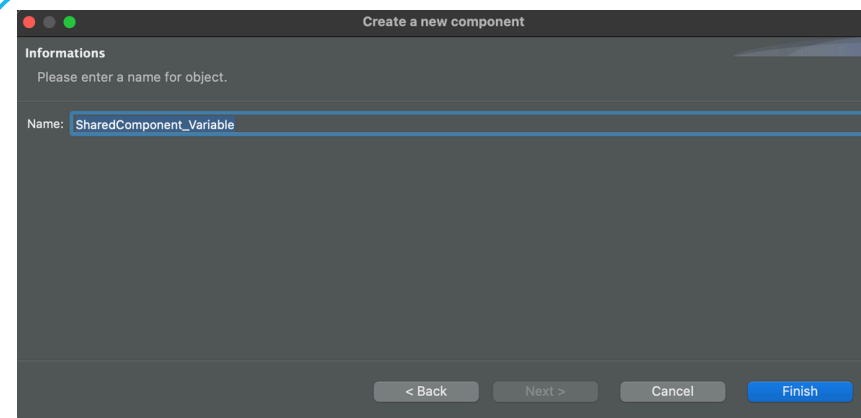
To add **variables** to **WatchlistCard**, right-click on the **shared component**, select **New**, then **UI Component** to open the **Create a new component panel**.



In the **Create a new component panel**, click on **SharedComponentVariable**, then on **Next**.

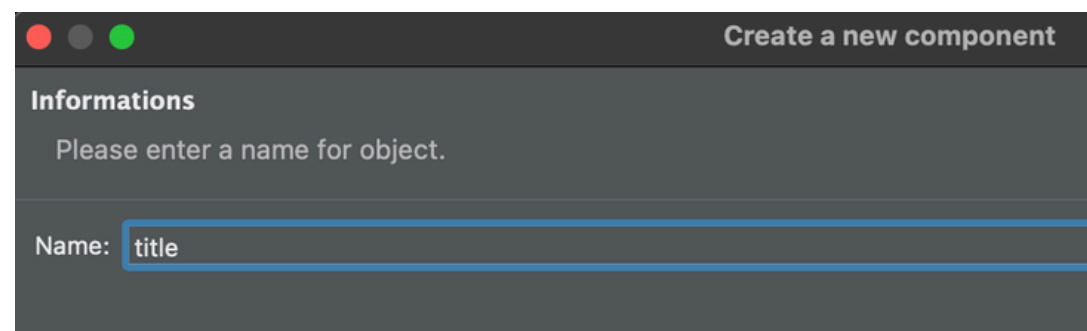


4.3 Add variables

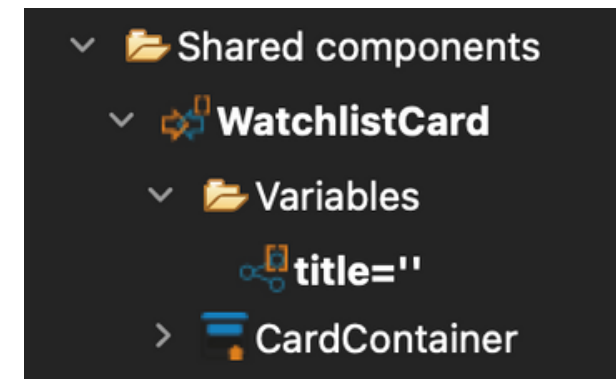


Enter **title** as name

↓ for the **SharedComponentVariable**, then click on **Finish**.

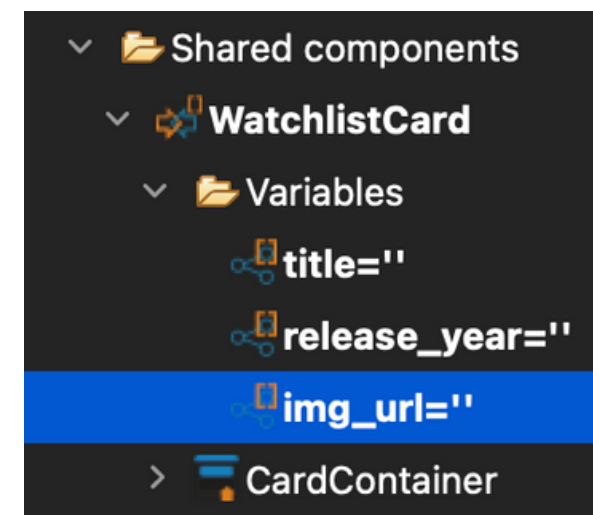


Inside **WatchlistCard**, appears a new folder, **Variables**, with the **variable title**.



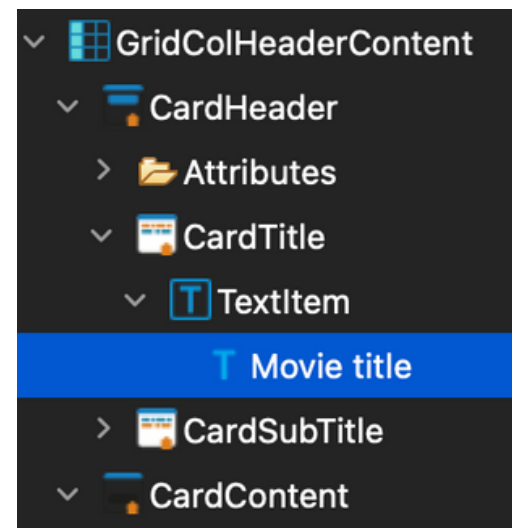
Repeat the same process

↓ to create 2 other **SharedComponentVariables**: **release_year** and **img_url**.

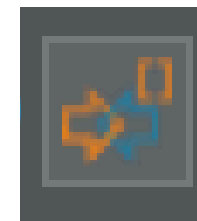
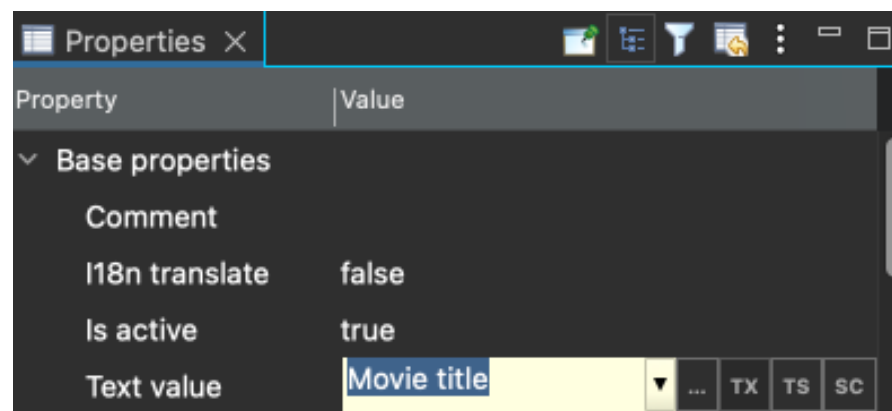


4.4 Bind the variables

We want to bind the **SharedComponentVariable title** to the **Text element** in the **TextItem** of **CardTitle**.

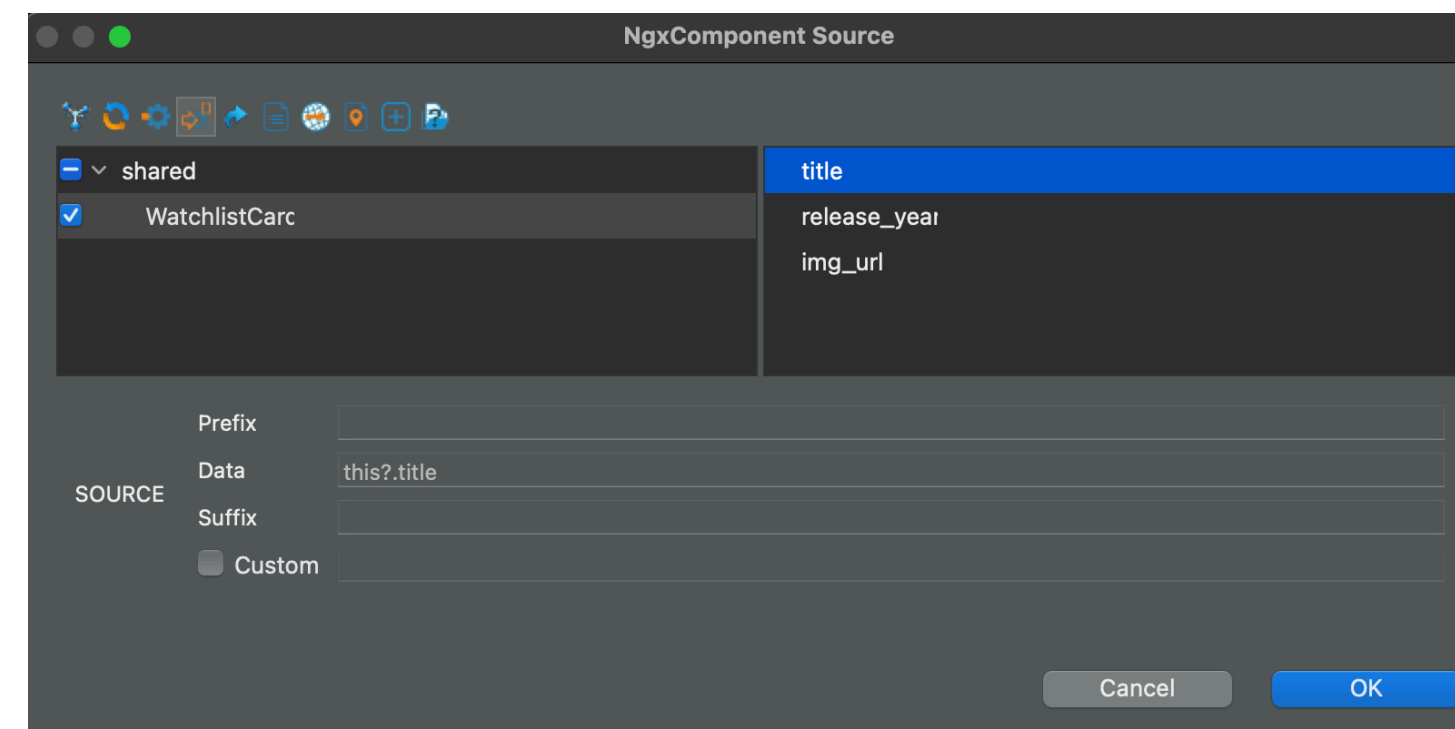


In the **Text element's** properties, click on ... to open the **NgxComponent Source** panel.



In the **NgxComponent Source** panel, click on this icon to choose **SharedComponents** as **source**.

Then click on **WatchlistCard** to choose this component, and on **title** to choose it as source.



4.4 Bind the variables

In the **Text** element's properties, the **Text value** is now **this.title**.

Property	Value
Base properties	
Comment	
I18n translate	false
Is active	true
Text value	this.title

In the treeview, the **Text element** appears now as **{{this.title}}**.

GridColHeaderContent
CardHeader
Attributes
CardTitle
TextItem
T {{this.title}}
CardSubTitle
CardContent



Do the same thing with **CardSubTitle** and the variable **release_year**.

Base properties	
Comment	
I18n translate	false
Is active	true
Text value	this.release_year

And again with **LazyImage** and the variable **img_url**.

Property	Value
@Component	
Identifier	
@Properties	
Alt	not set
Src	this.img_url



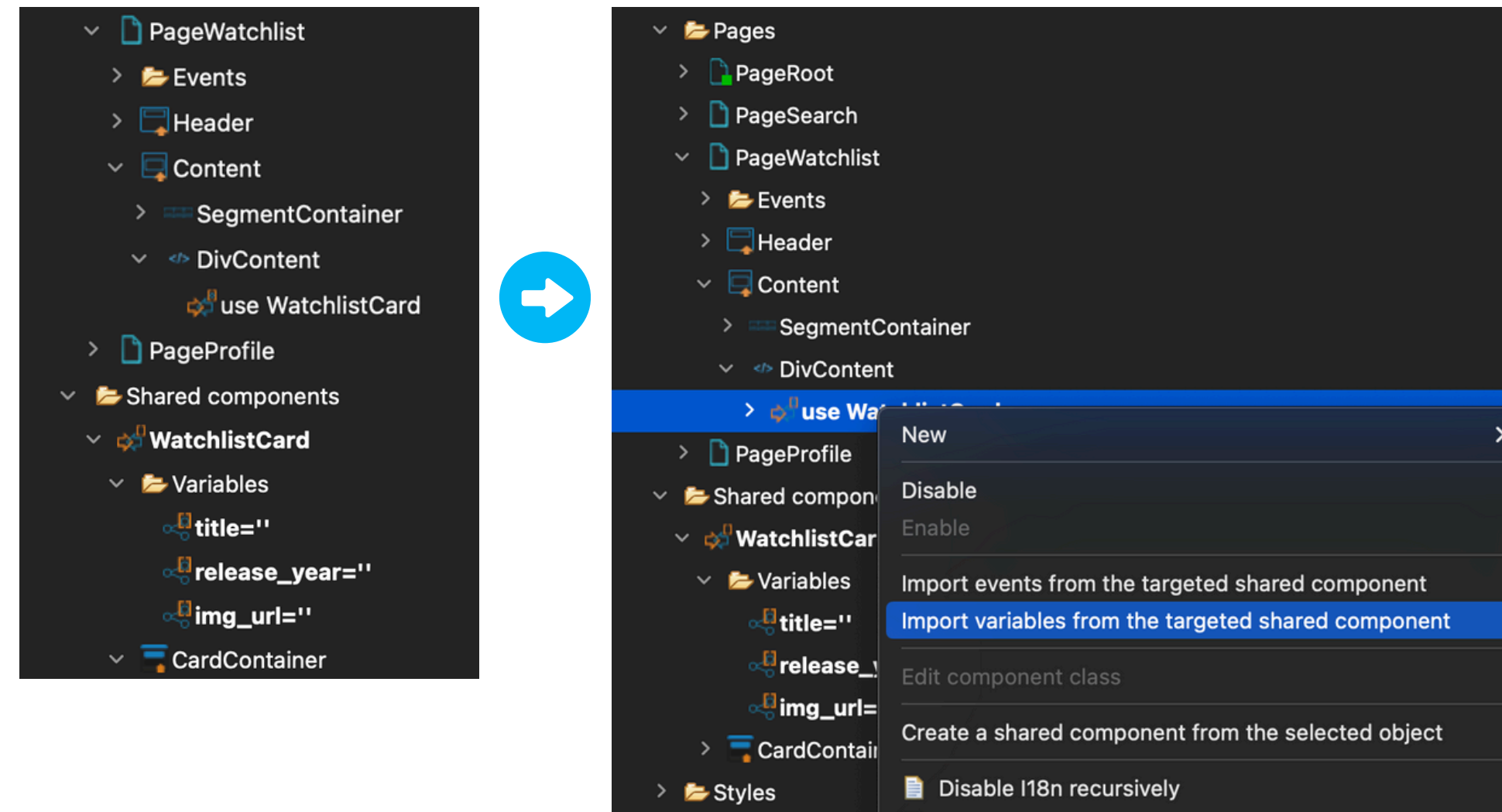
The variables of **WatchlistCard** are now binded.

WatchlistCard
Variables
title=""
release_year=""
img_url=""
CardContainer
Grid
GridRow
GridColIMG
LazyImage
GridColHeaderContent
CardHeader
Attributes
CardTitle
TextItem
T {{this.title}}
CardSubTitle
TextItem
T {{this.release_year}}
CardContent

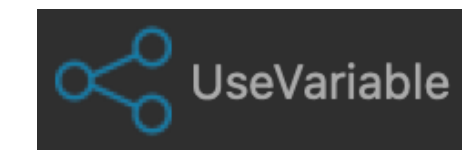
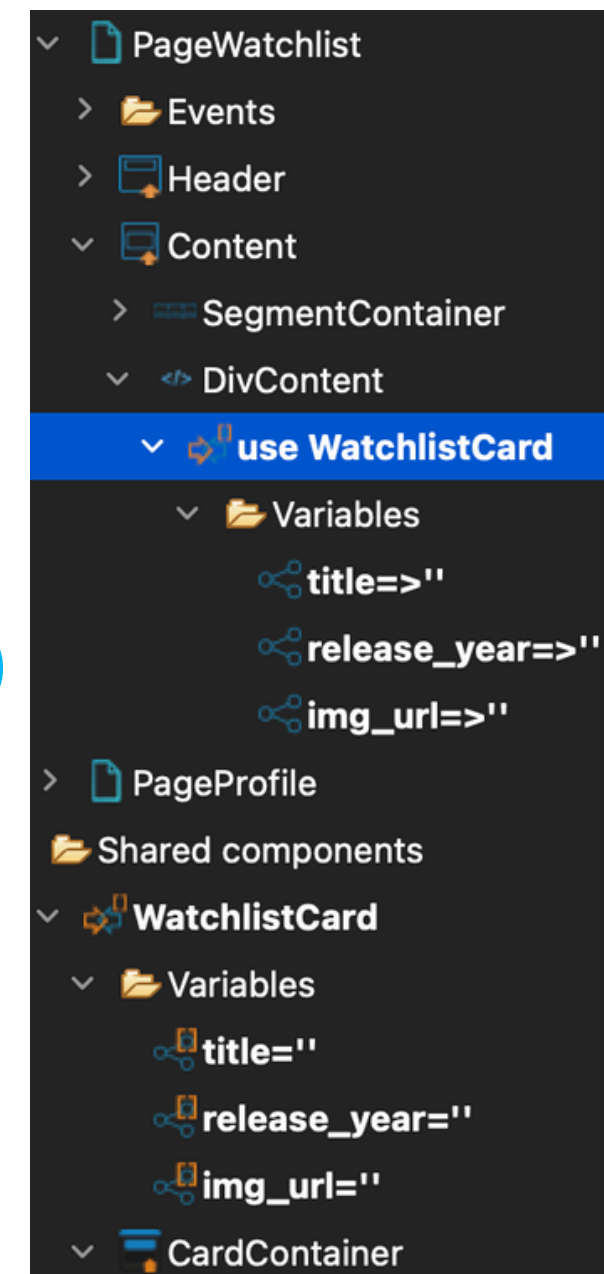


4.5 Import variables in UseSharedComponent

To import the variables of **WatchlistCard** in **useWatchlistCard**,
right-click on **useWatchlistCard**,
and select **Import variables from the targeted shared component**.



The variables are now imported.

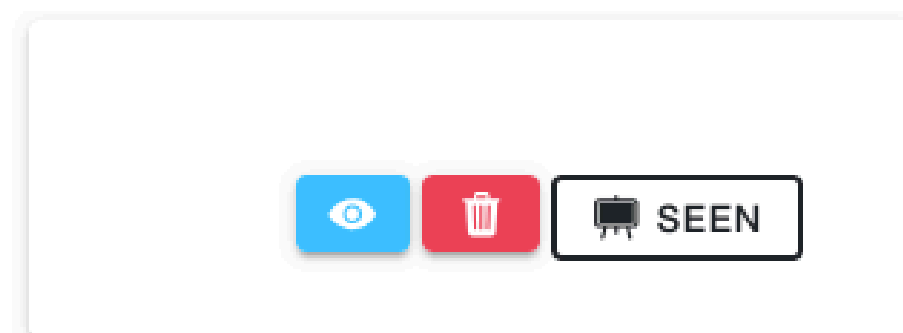
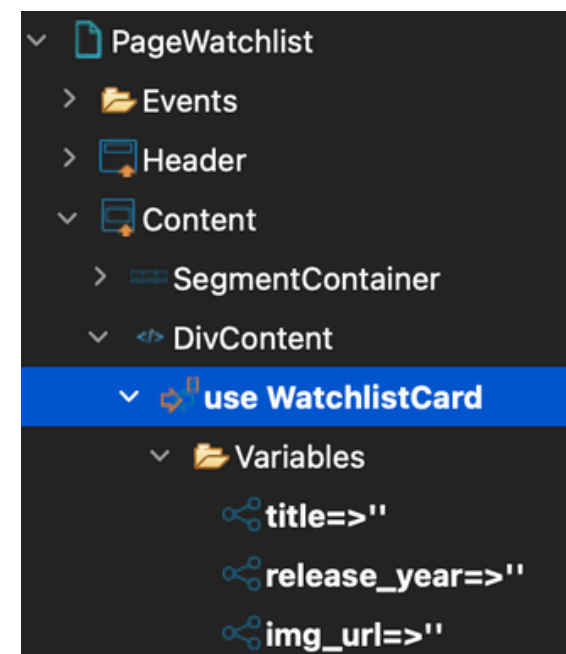


In the **NGXPalette**,
the component
corresponding
to the variables of a
usedSharedComponent
is **UseVariable**.

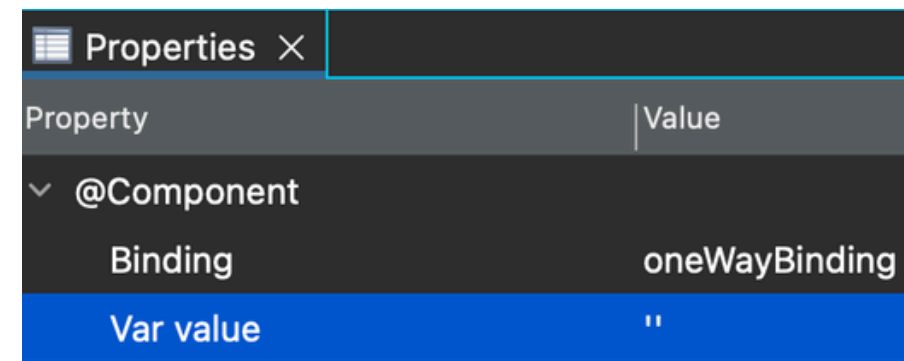


4.6 Add a value to UseVariable

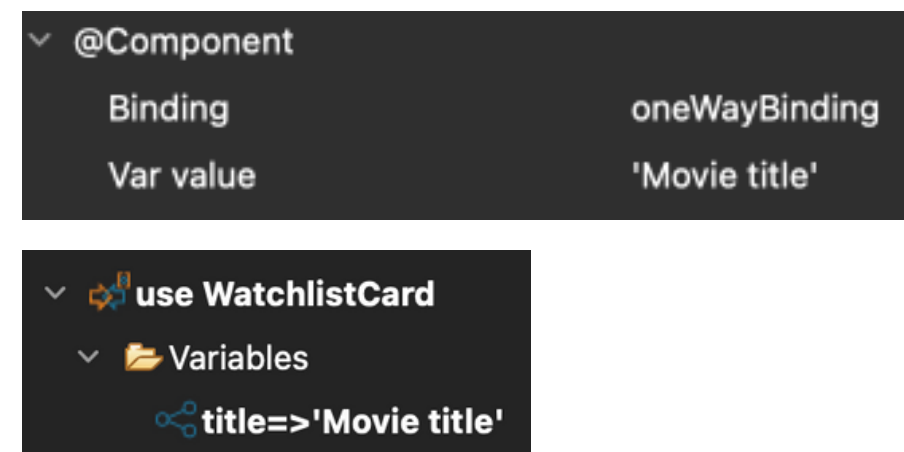
The variables in **UseWatchlistCard** have "", an empty string, as value, and the **title**, **release_year** and **image** of the component are empty.



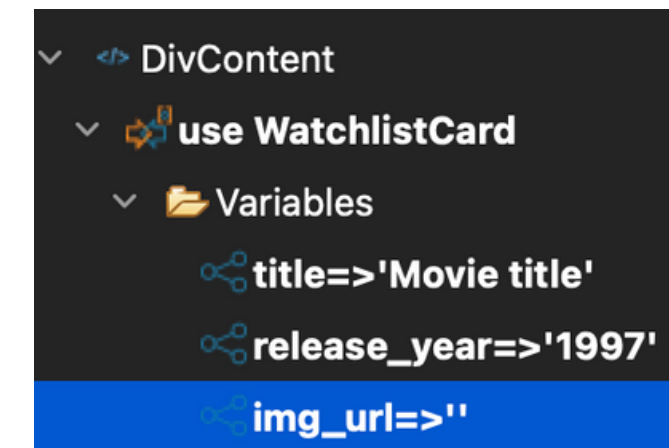
Click on the variable 'title' in **useWatchlistCard** to display its properties.



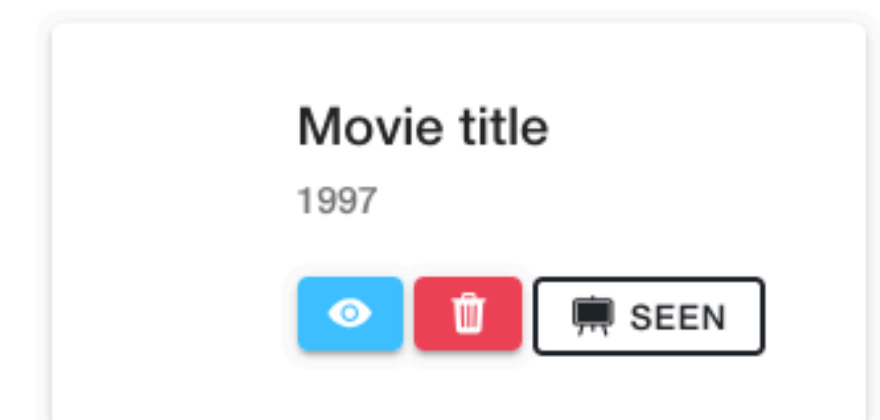
In **Var value** property, replace "" by '**MovieTitle**'.



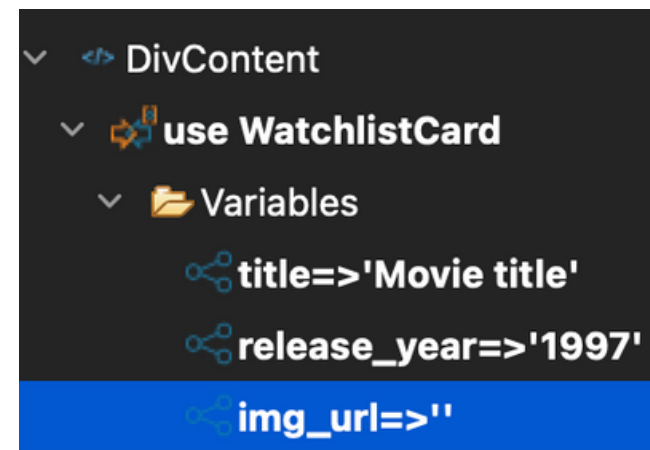
Do the same thing with **release_year** and '**1997**'.



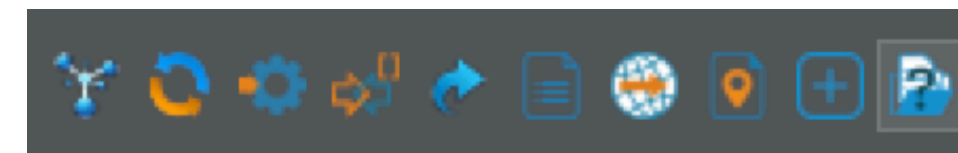
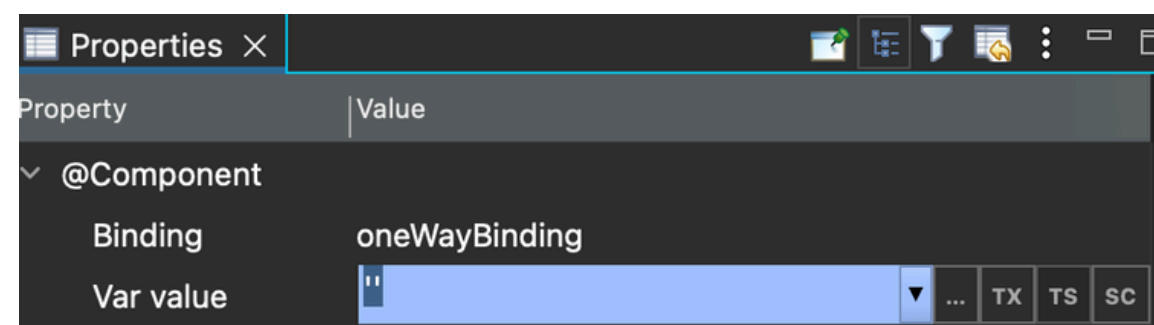
MovieTitle and **1997** appear now in the component.



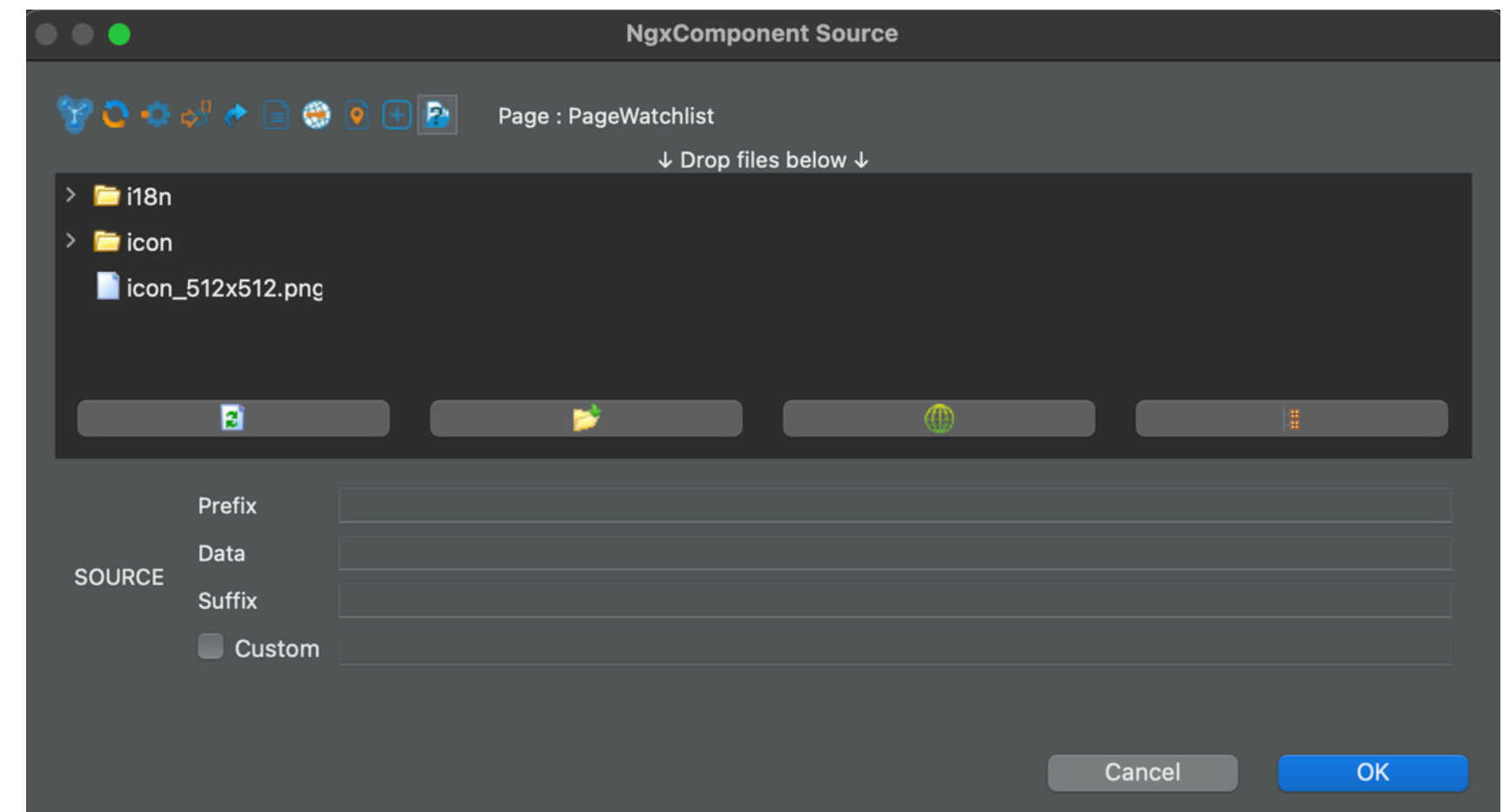
4.6 Add a value to UseVariable



In the `img_url`'s properties,
click on **SC** in **Var value**
to open
the **NgxComponent Source** panel.

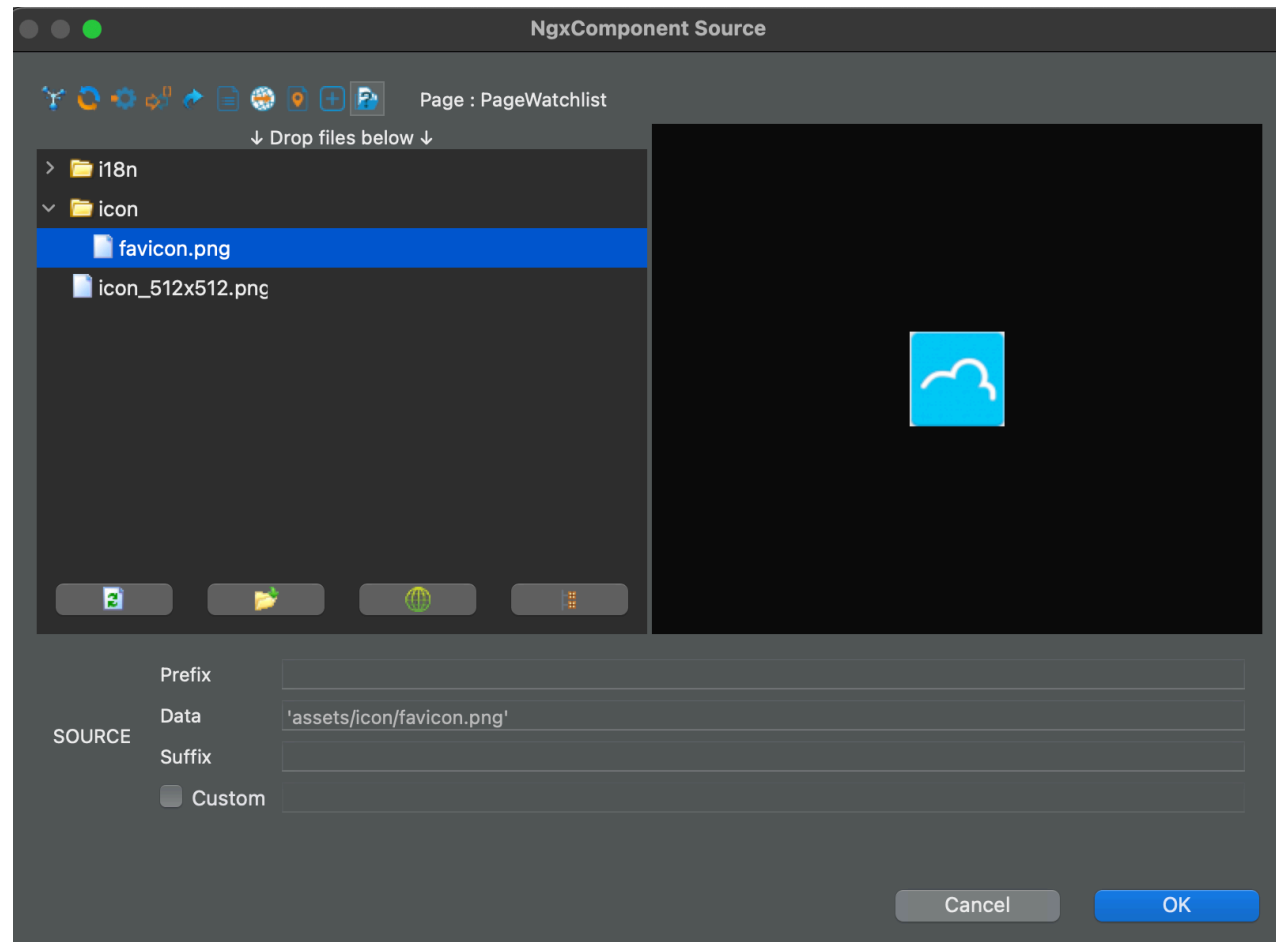


In the **NgxComponent Source** panel,
click on this icon
to select the **assets** folder as **source**.

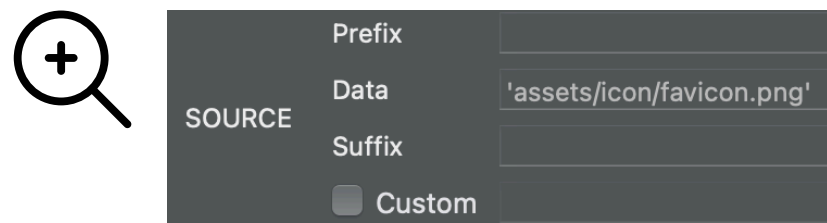


4.6 Add a value to UseVariable

In the **Icon** folder of the **assets** folder, click on **favicon.png** to select it as **source**.



The **path** for **favicon.png** appears in **SOURCE Data**.



➡ Then click on **OK**.



The **path** for **favicon.png** appears in **Var value** property, and in the **value** in the treeview of the **useVariable** **img_url**.

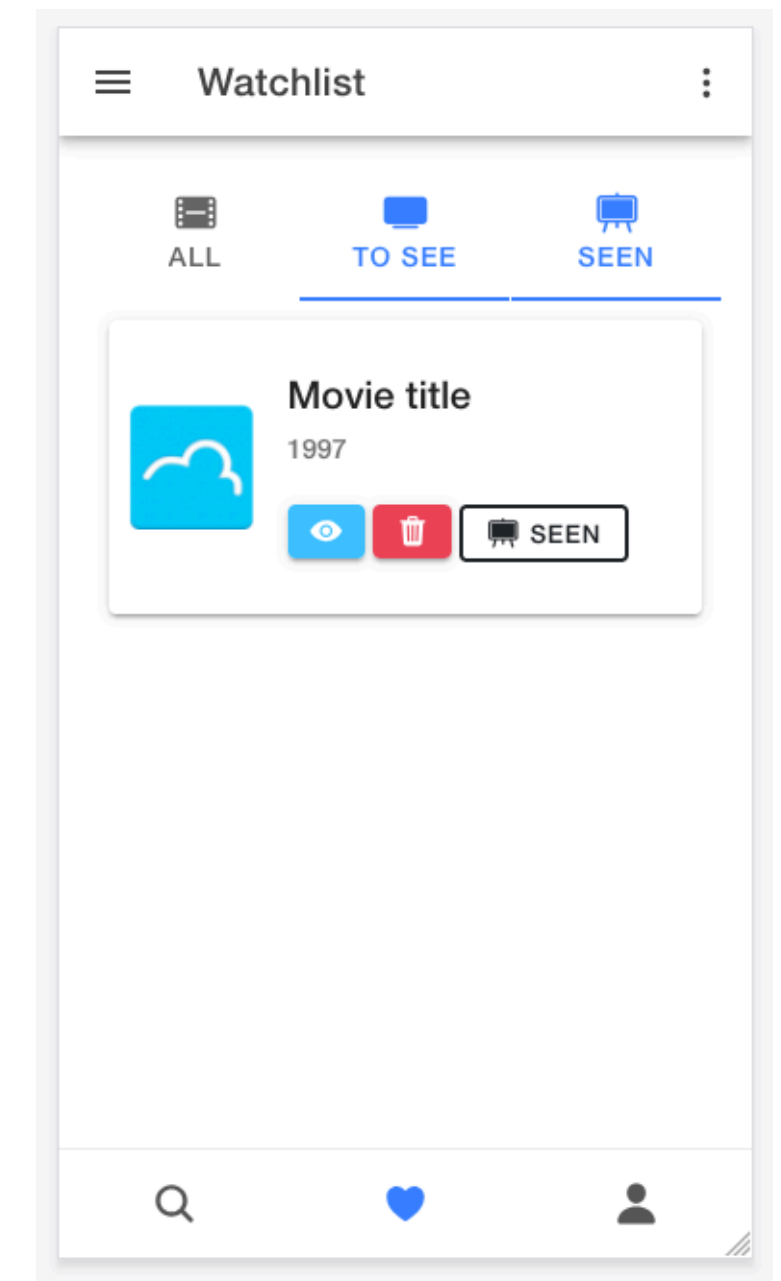
```

@Component
  Binding      oneWayBinding
  Var value    'assets/icon/favicon.png'
  
```

```

DivContent
  use WatchlistCard
    Variables
      title=>'Movie title'
      release_year=>'1997'
      img_url=>'assets/icon/favicon.png'
  
```

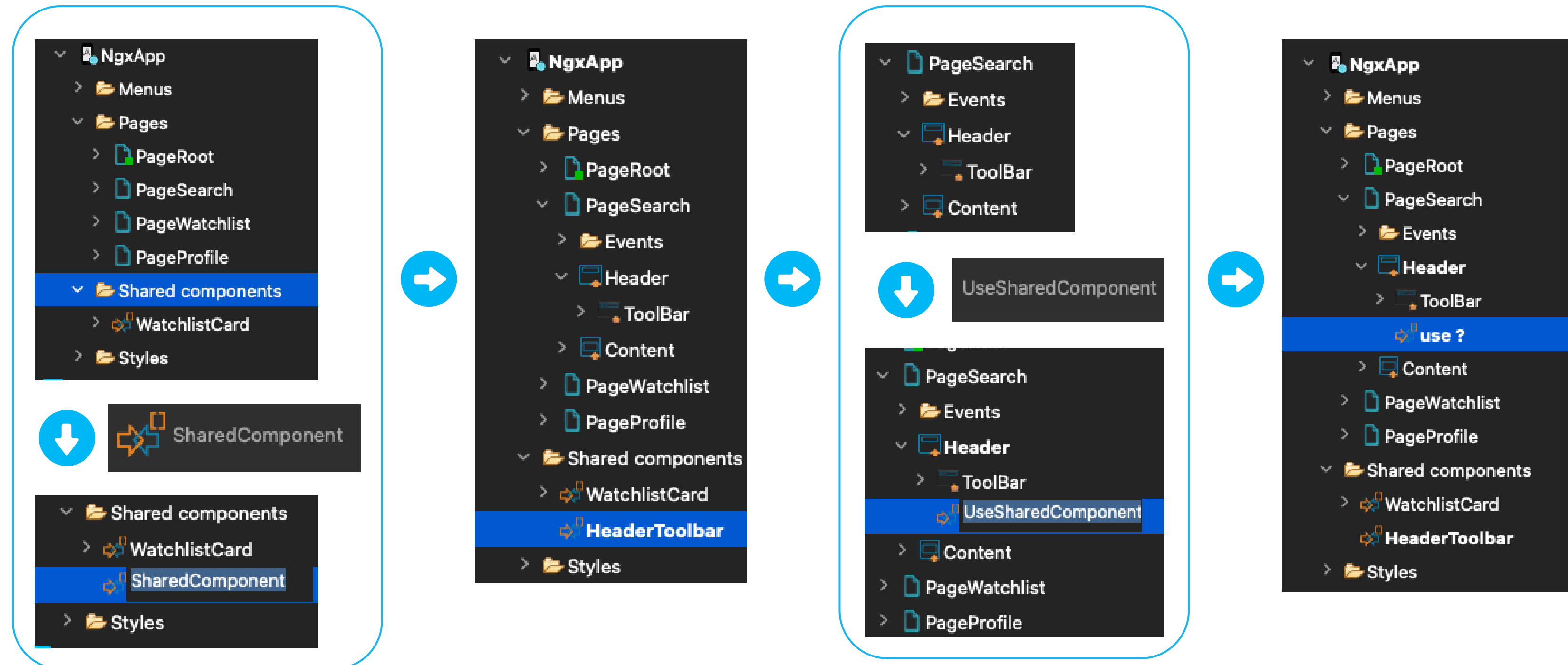
The image **favicon.png** appears now in the component.



4.7 Create a header as sharedComponent

Now, we want a header to use in all pages.

So let's create a **sharedComponent**, name it **HeaderToolBar** and use it in **Header**.

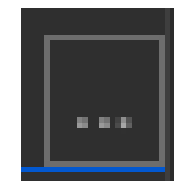
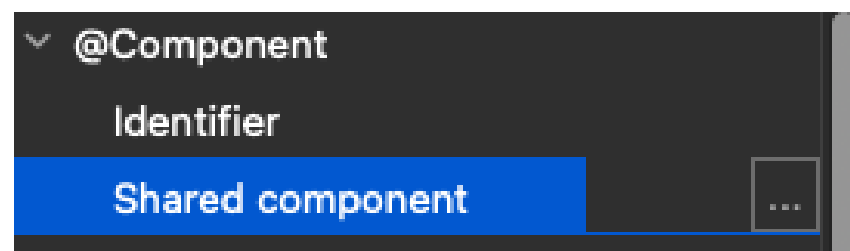
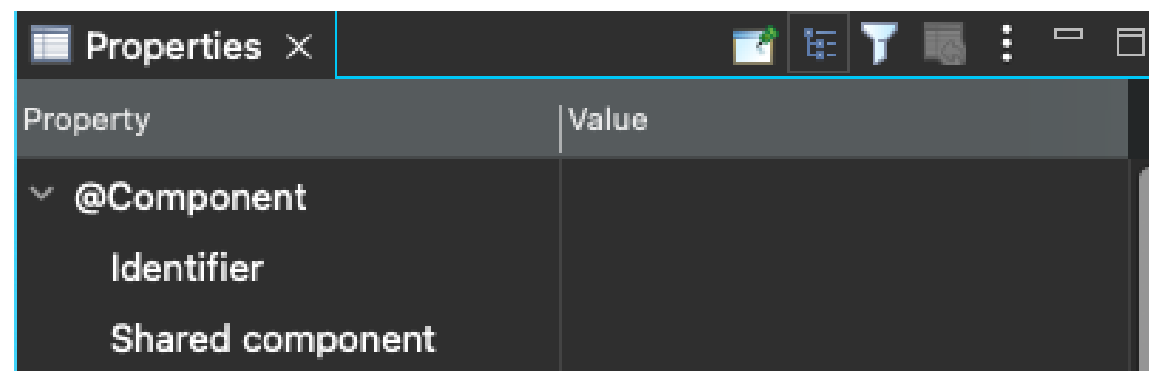


4.7 Create a header as sharedComponent

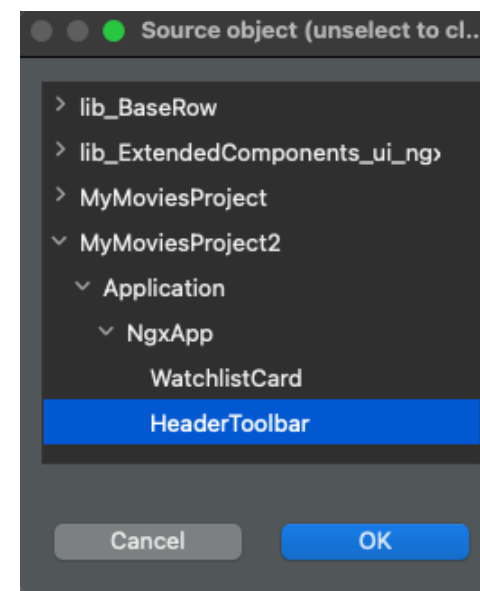
In **useSharedComponent**'s properties,



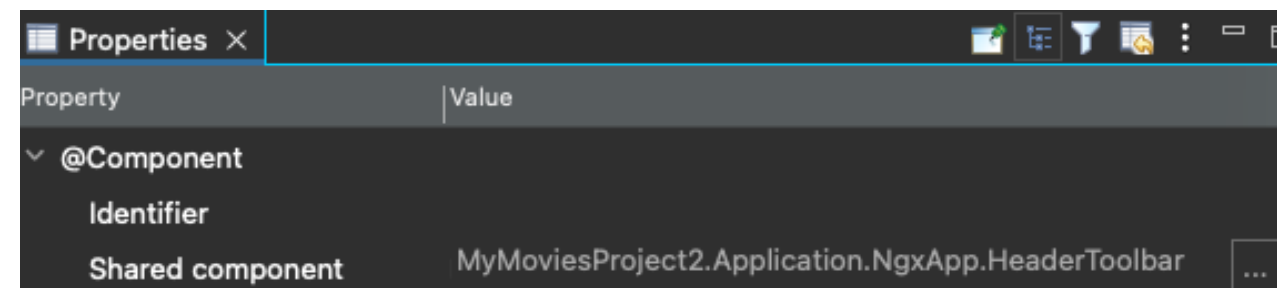
click on the property **Shared component** to display the ... and click on ... to open the **Source object panel**.



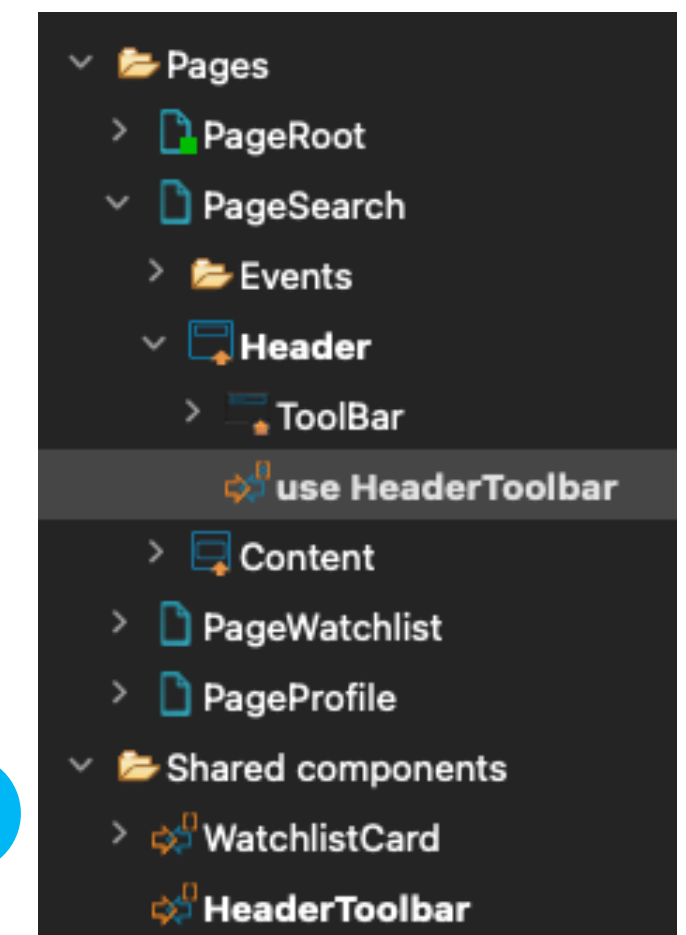
In the **Source object panel**, select **HeaderToolBar**.



HeaderToolBar appears as **value** for the **Shared component** property.



use HeaderToolBar appears in the treeview



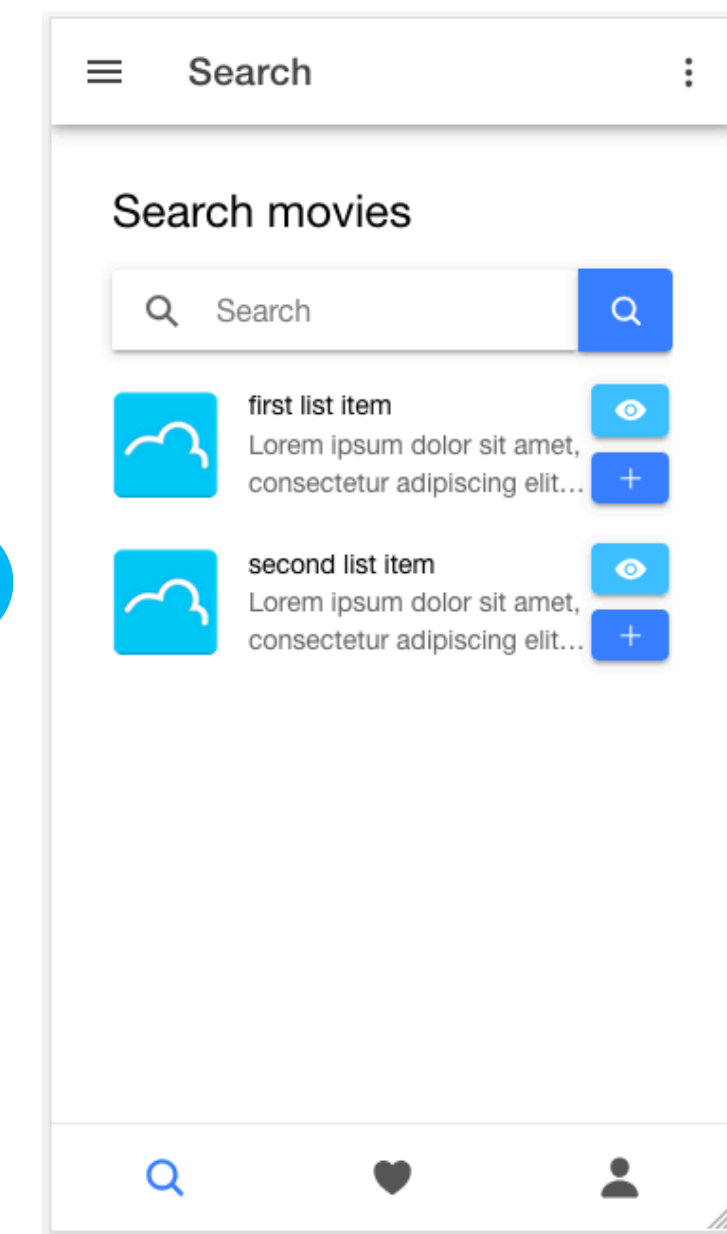
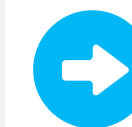
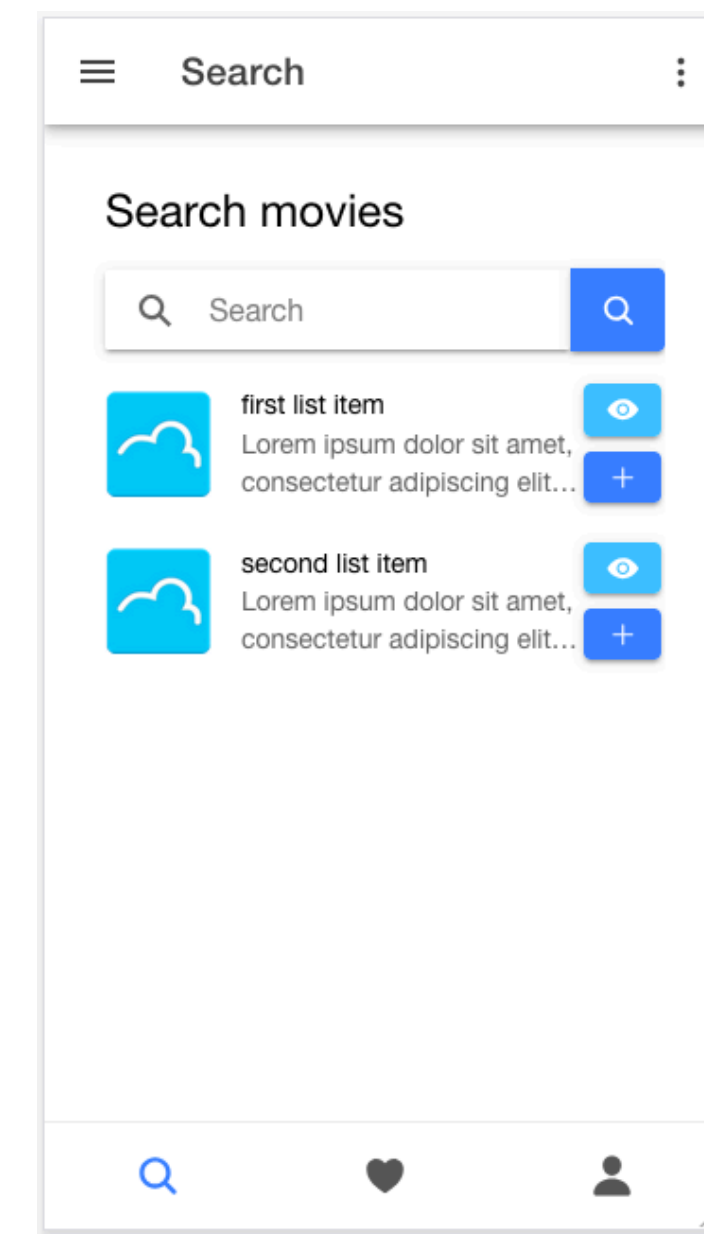
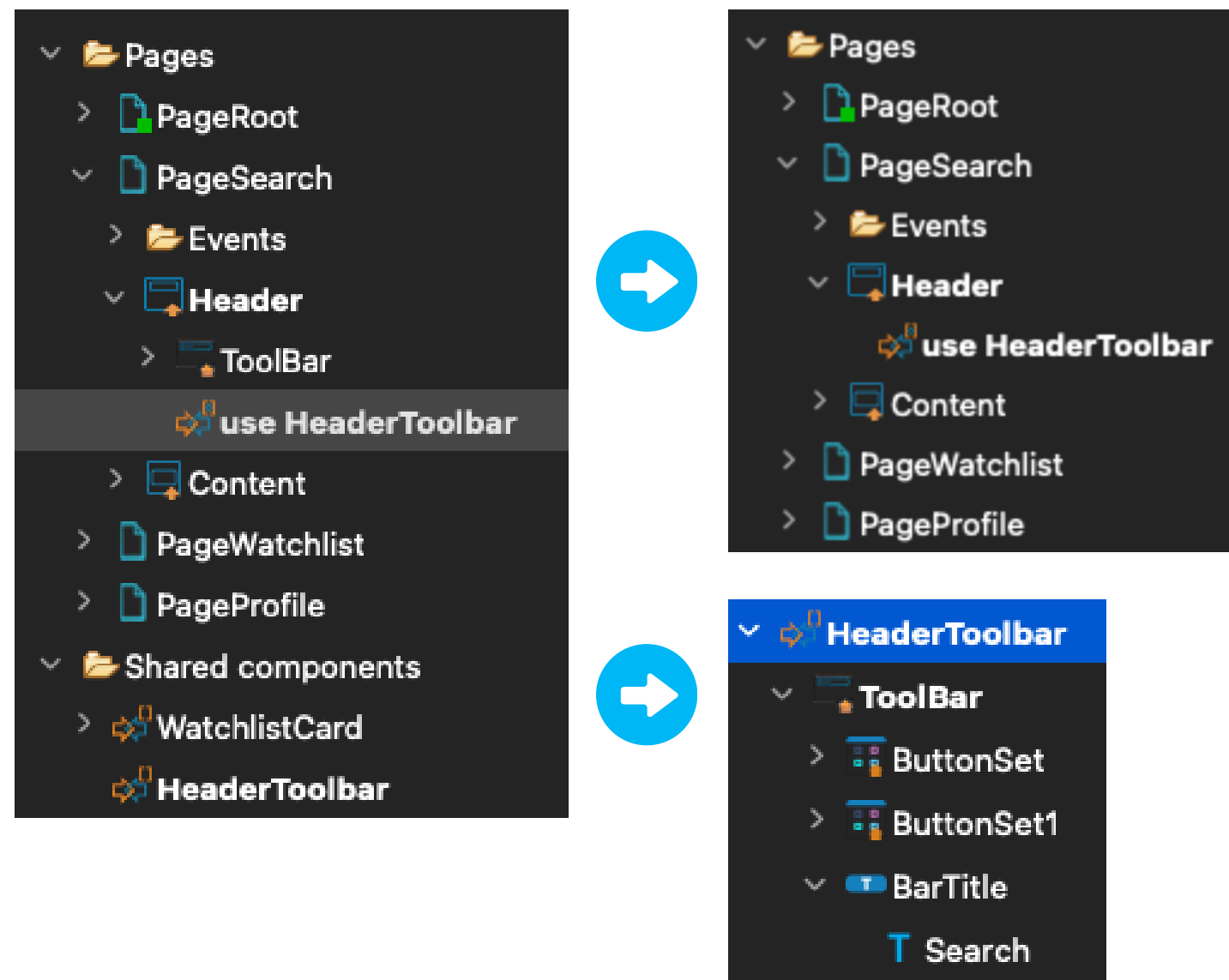
4.7 Create a header as sharedComponent

Drag the **ToolBar** Component from the **Header** and drop it into the **shared component HeaderToolbar**.

Visually, the result is identical.

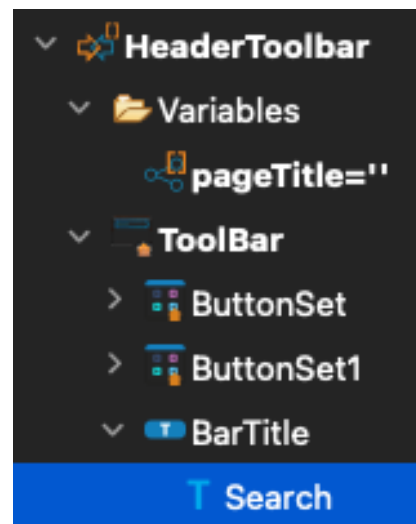
The **ToolBar** is in **Header**.

The **ToolBar** is in **HeaderToolbar**.

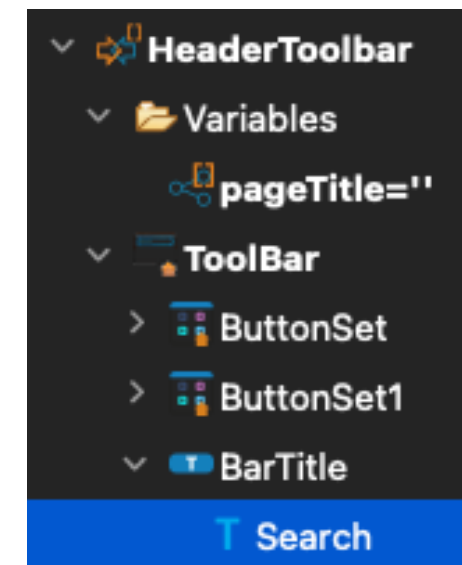
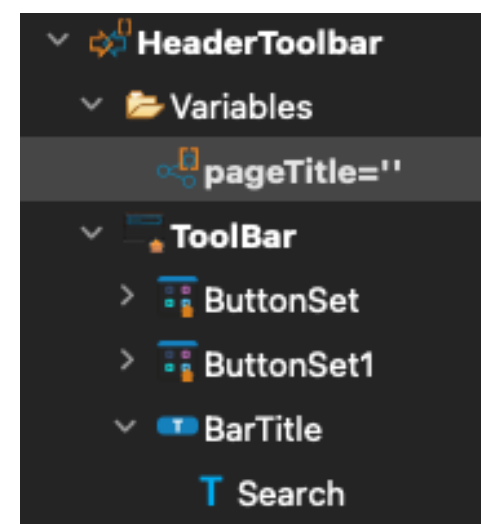
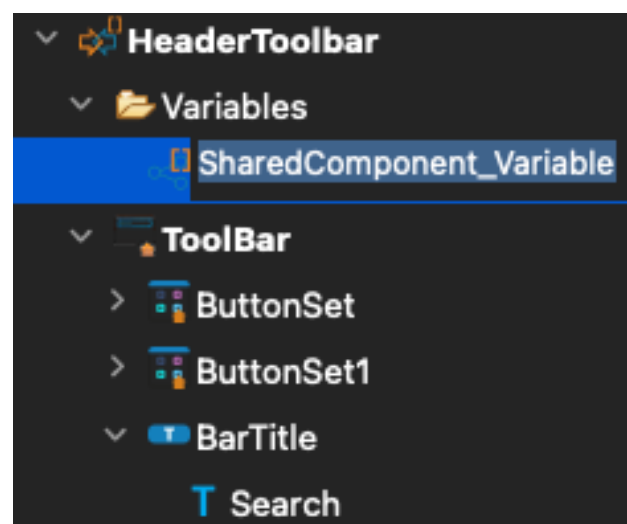


4.7 Create a header as sharedComponent

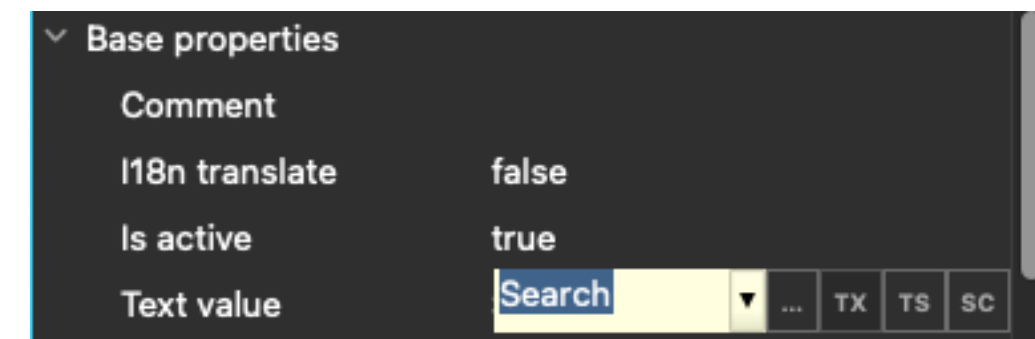
To use the header in all pages, let's add a variable for the title.



Let's add a **SharedComponent variable** in **HeaderToolbar**, and name it **pageTitle**.



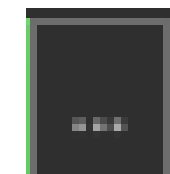
Click on the **text element** of **BarTitle** in the **ToolBar** of **HeaderToolbar** to display its properties.



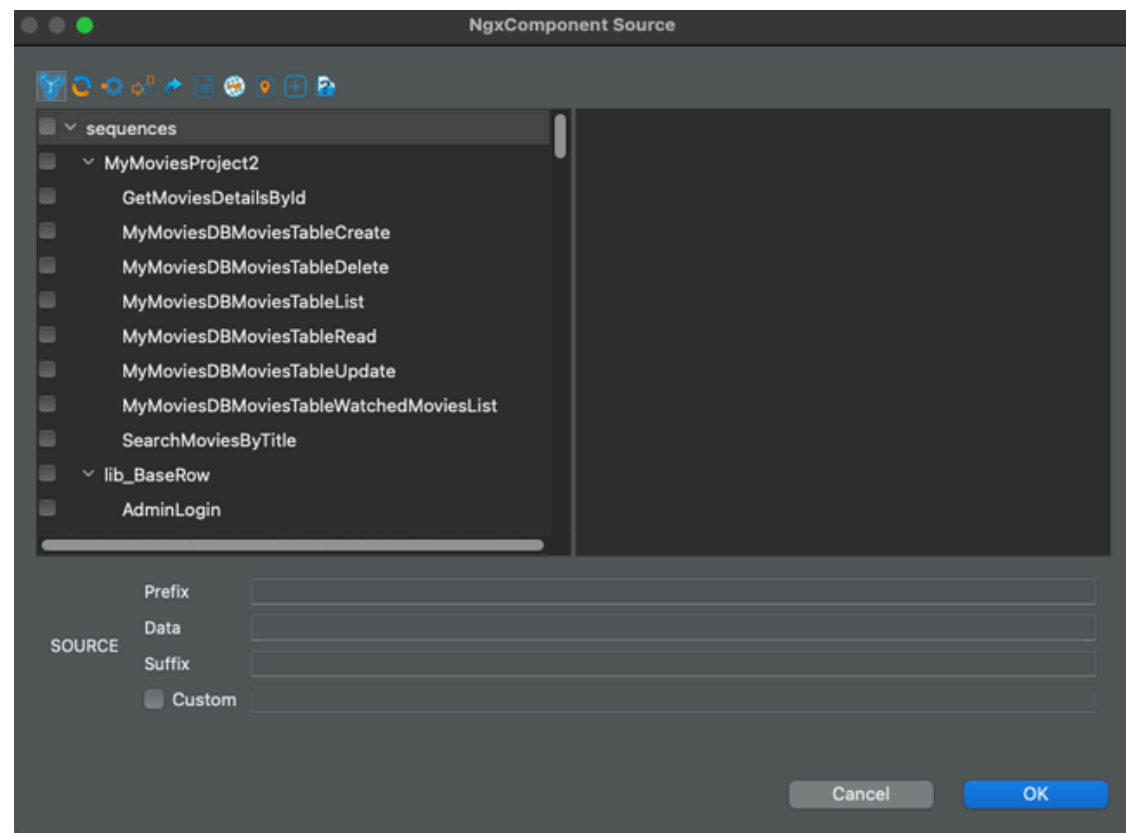
Click on the property **Text value** to display the buttons and click on ... or on **SC** to open the **NgxComponent Source** panel.



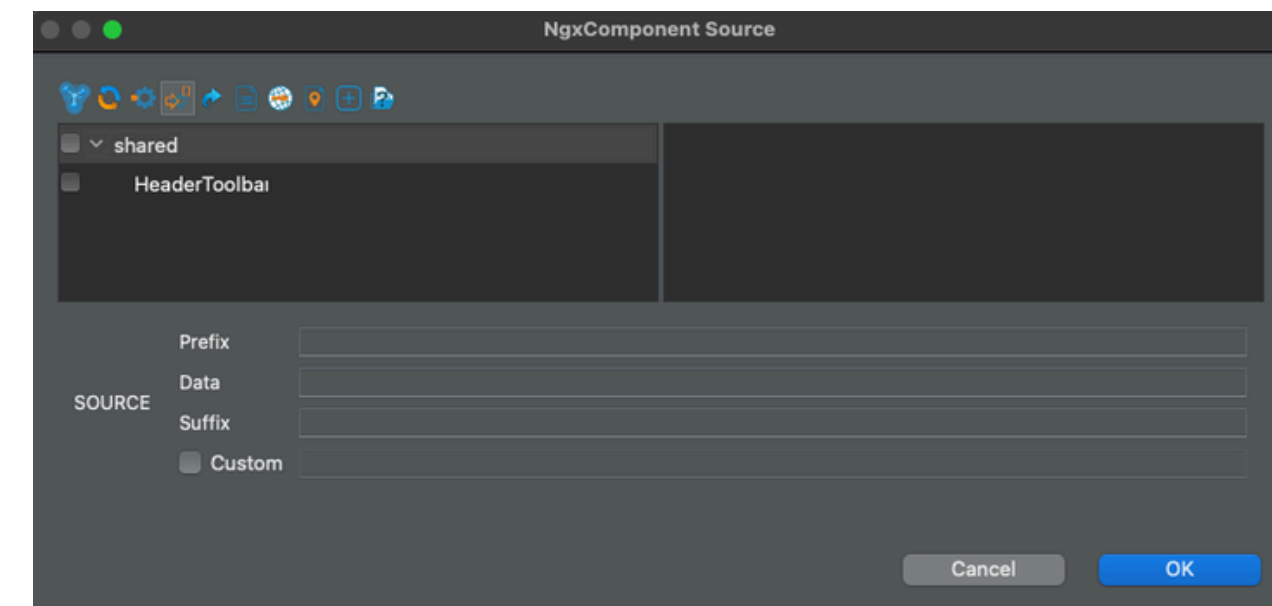
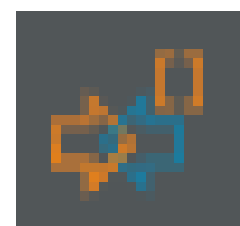
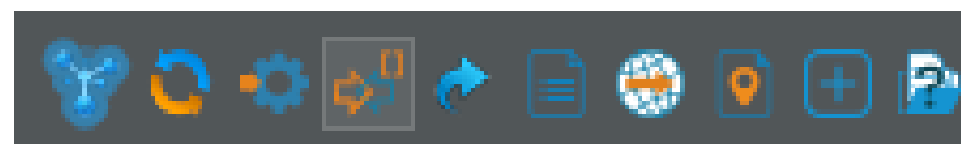
Or



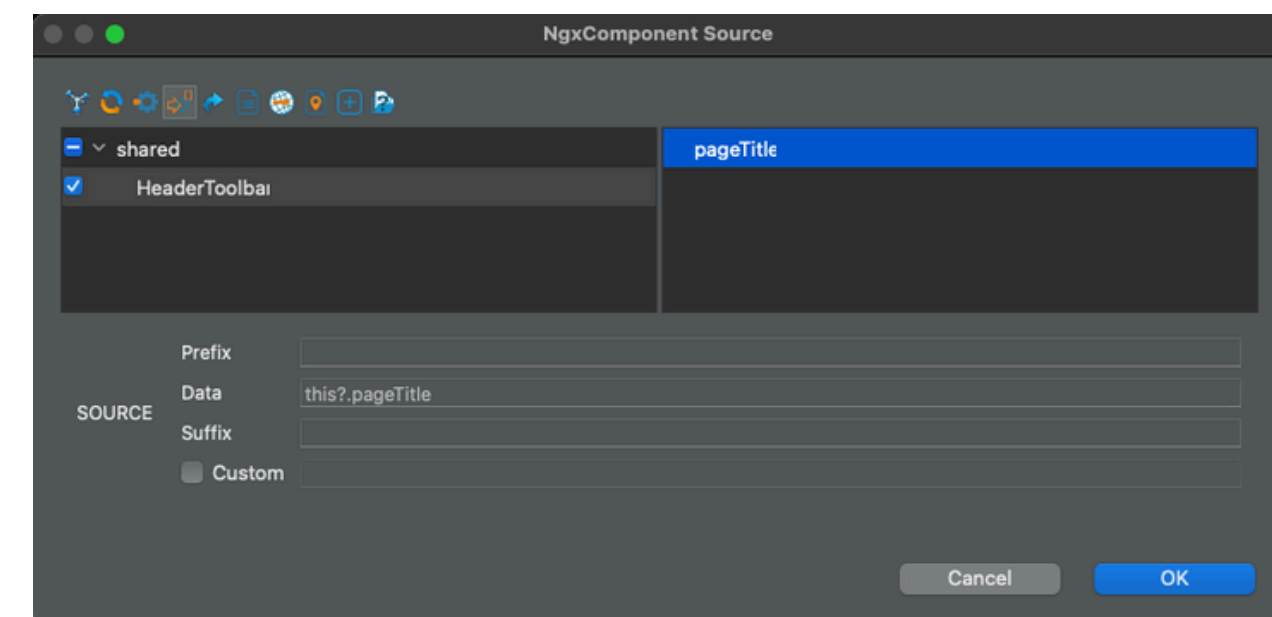
4.7 Create a header as sharedComponent



In the **NgxComponent Source** panel, click on the **shared component icon** to display the **shared component sources**.



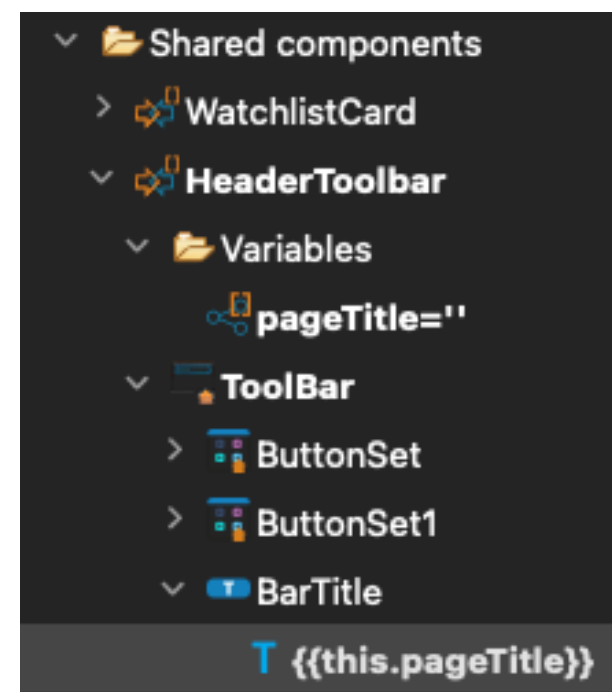
In the **shared component sources**, click on **HeaderToolbar** then on **pageTitle**.



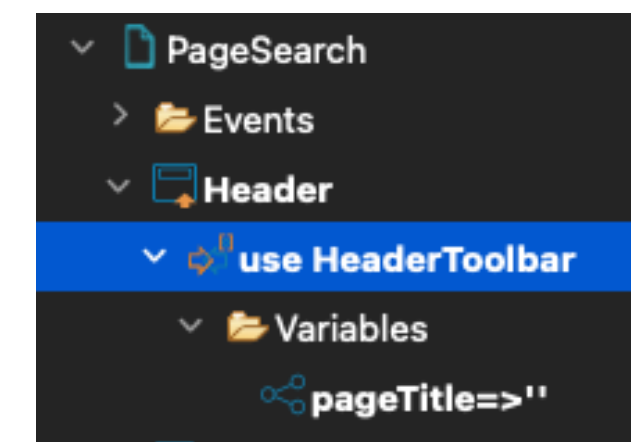
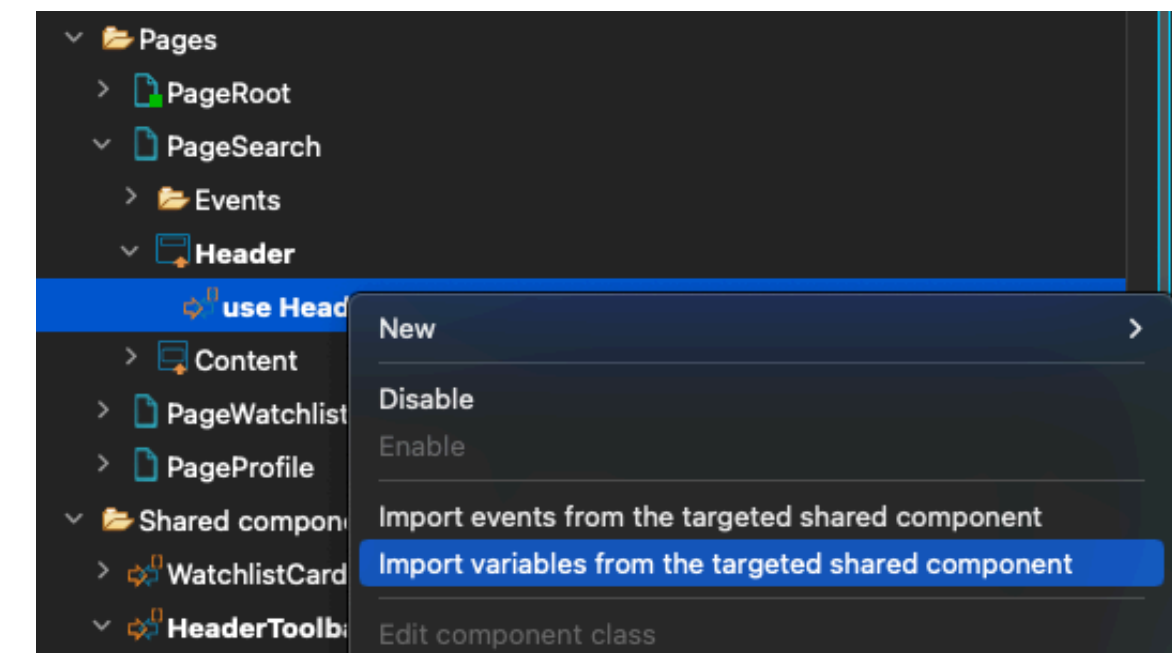
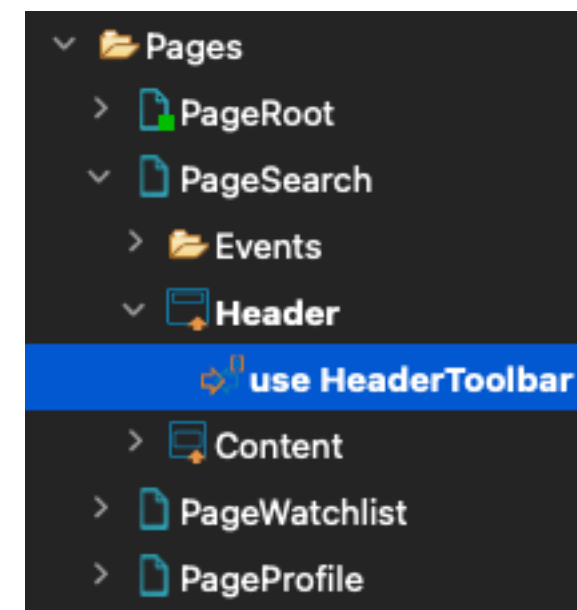
4.7 Create a header as sharedComponent

this.pageTitle appears as **value** in the **Text value property** of the **Text** element of **BarTitle**.

Base properties	
Comment	
l18n translate	false
Is active	true
Text value	this.pageTitle

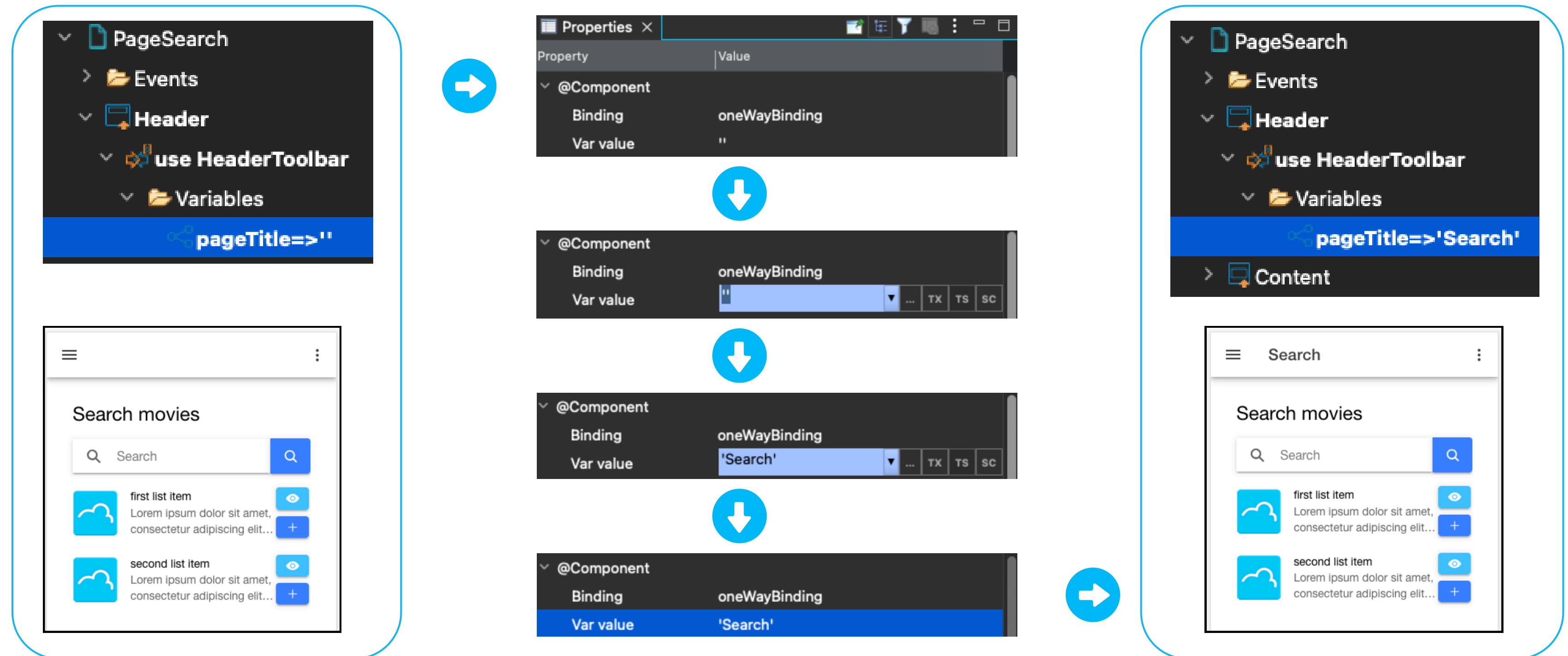


Now, let's import the **pageTitle** variable in use **HeaderToolbar**.



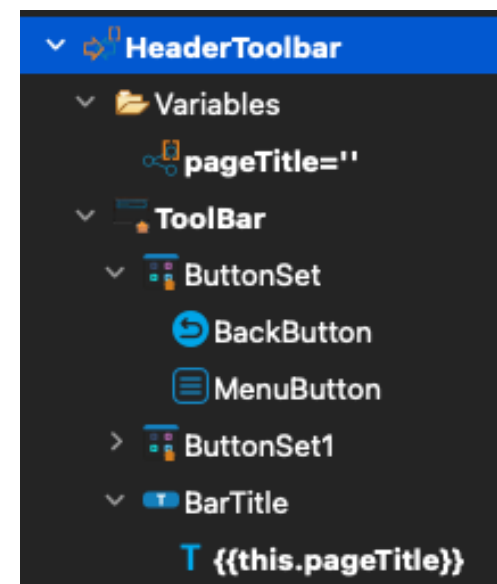
4.7 Create a header as sharedComponent

Now, let's add the value 'Search' to the variable in use HeaderToolbar.

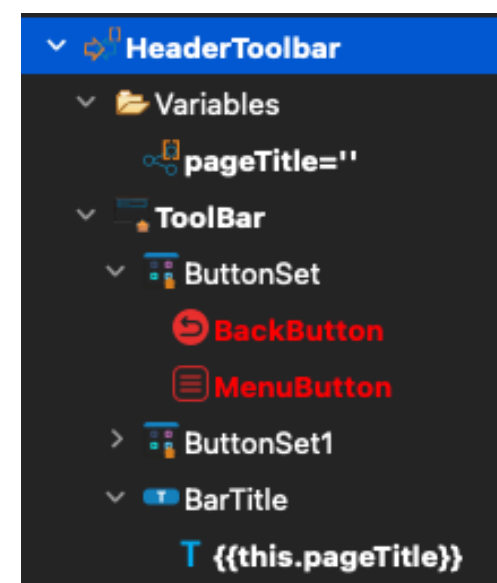
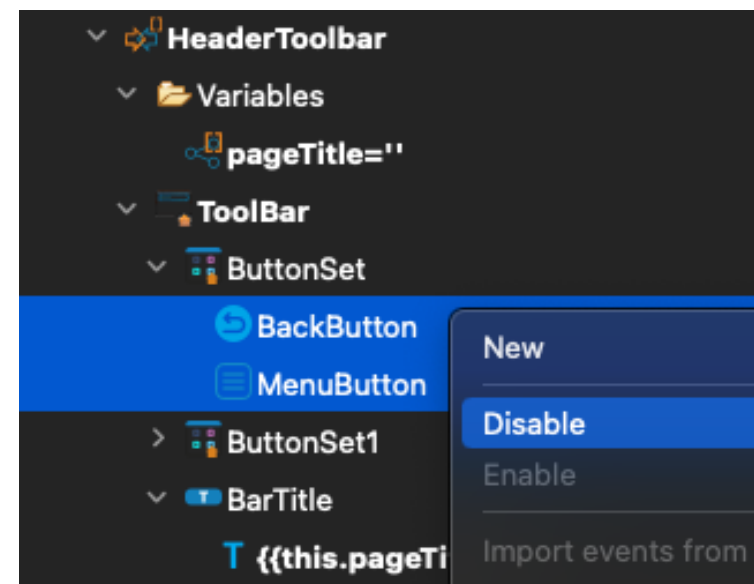


4.7 Create a header as sharedComponent

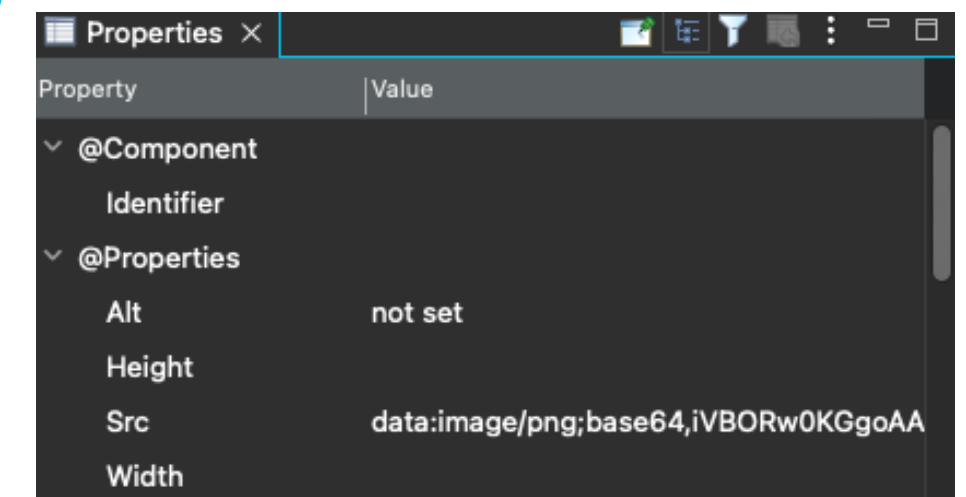
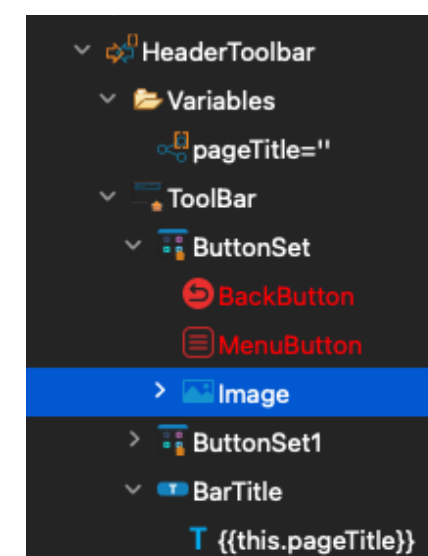
Now, let's change the layout of our HeaderToolBar.



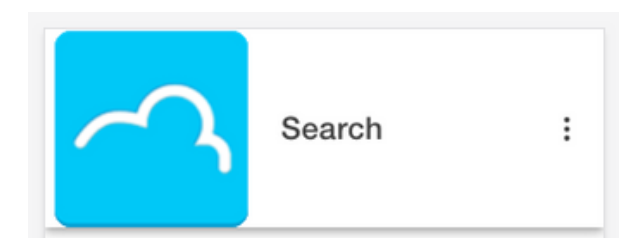
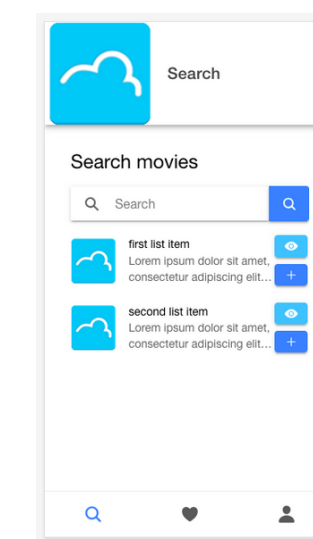
First, disable the **BackButton** and the **MenuButton** on the first **ButtonSet** in **ToolBar**.



Then, add an **Image** component in the same **ButtonSet**.

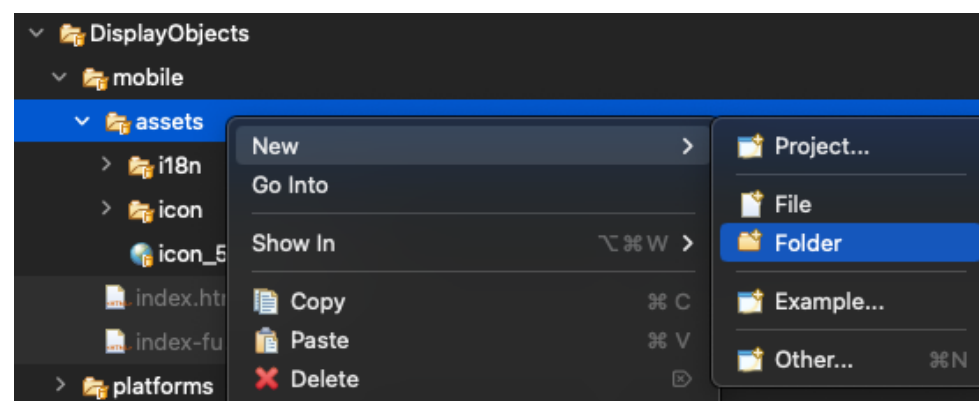
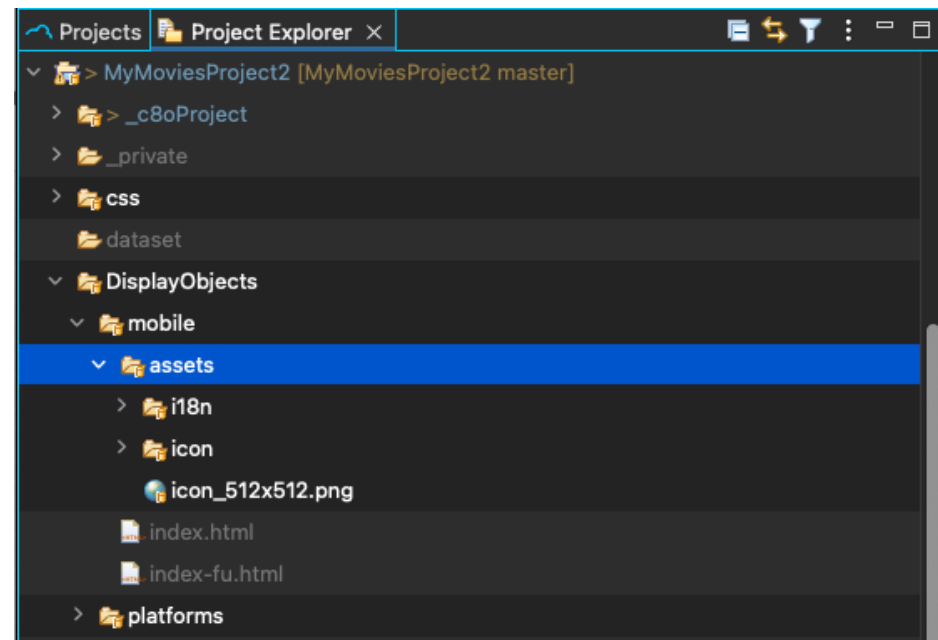


The **Image** component appears with an image by default.

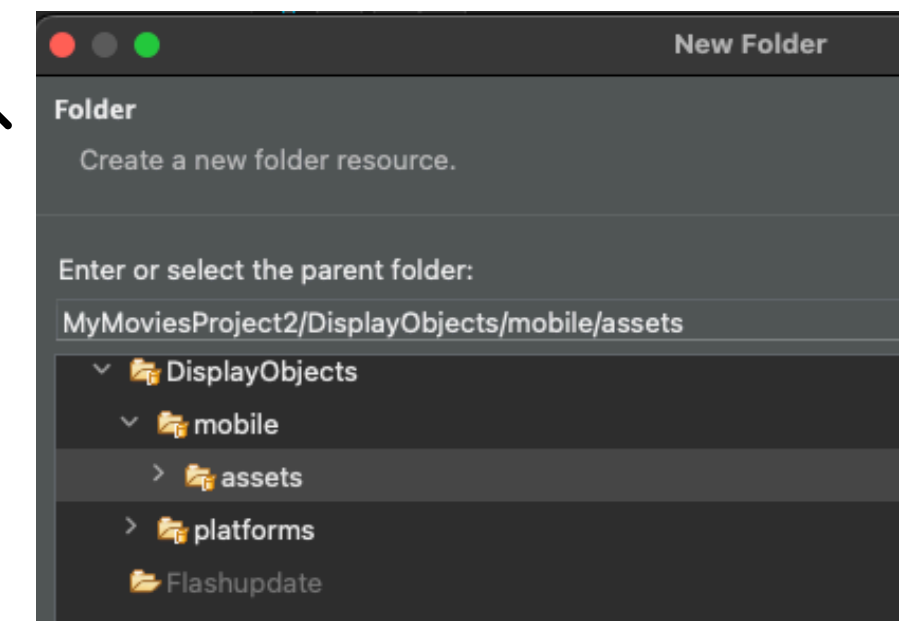
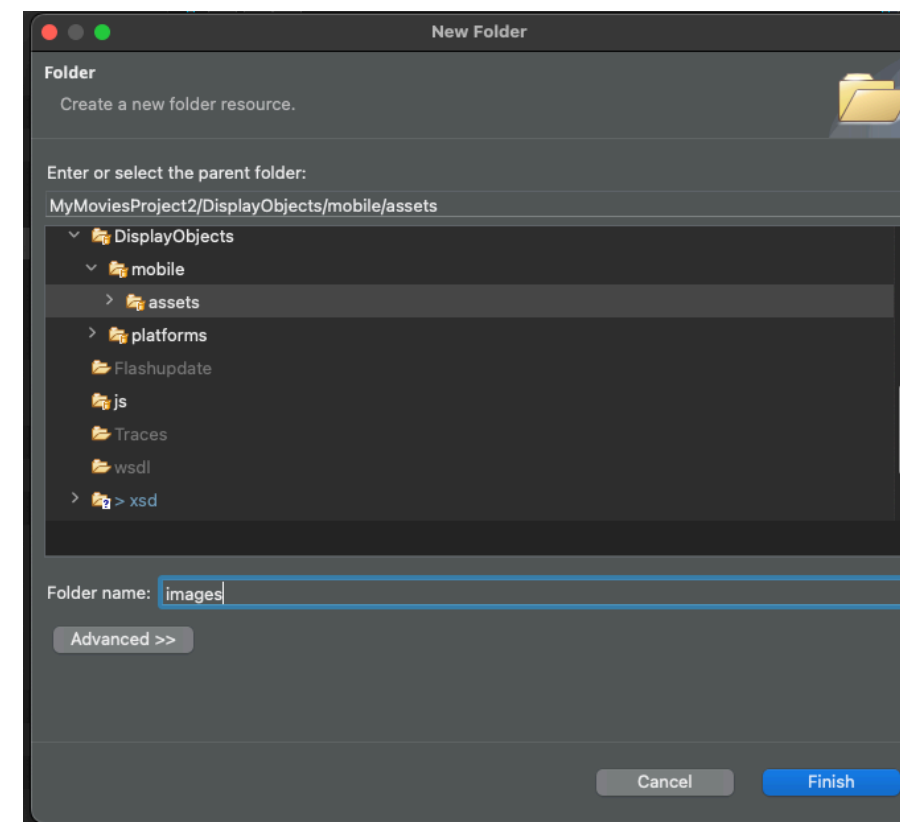


4.7 Create a header as sharedComponent

To replace the image by default,
first, we need to add the new image in the **assets folder** of the project in **Project Explorer**.
To find the **assets folder** : in **DisplayObjects > mobile > assets**.



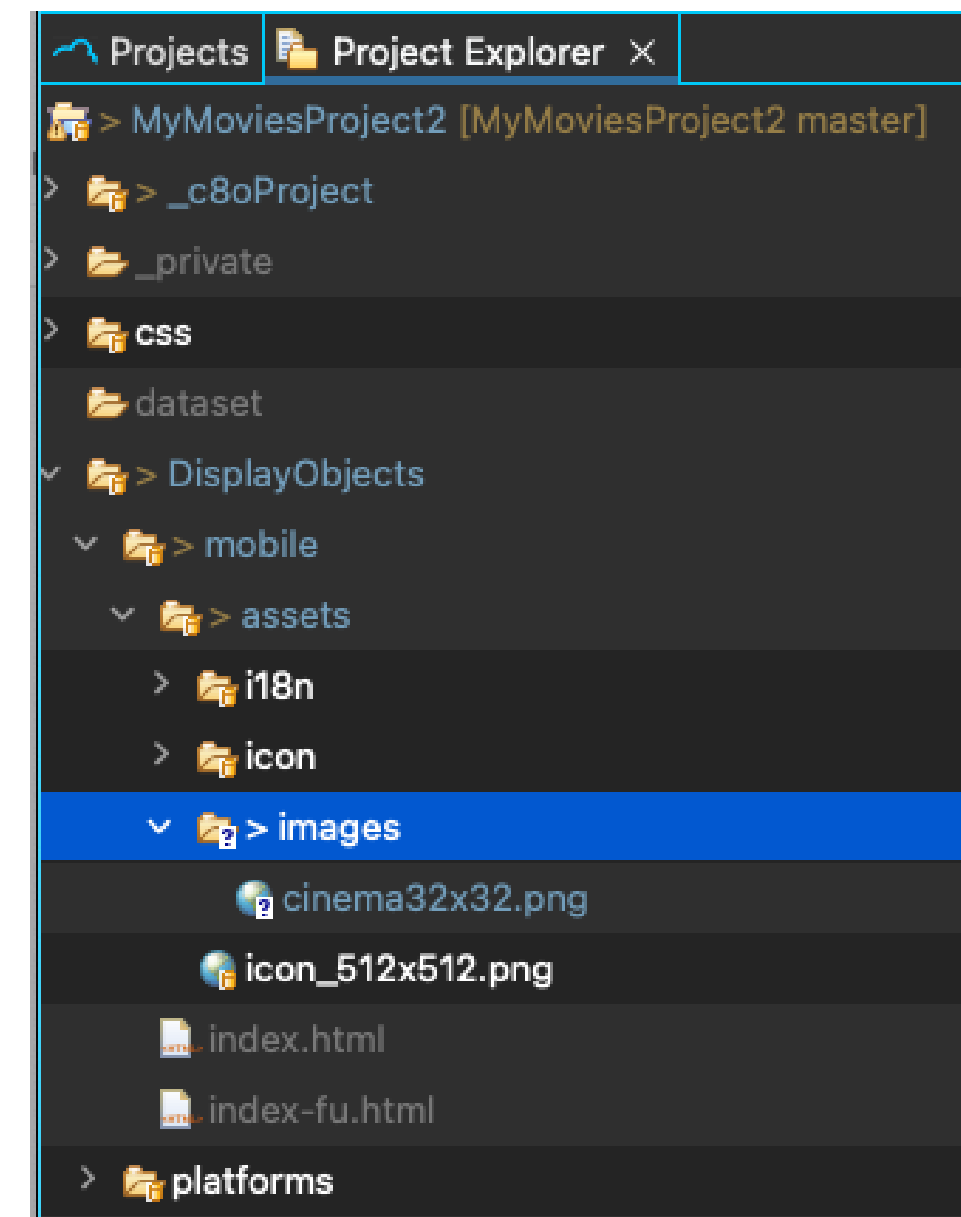
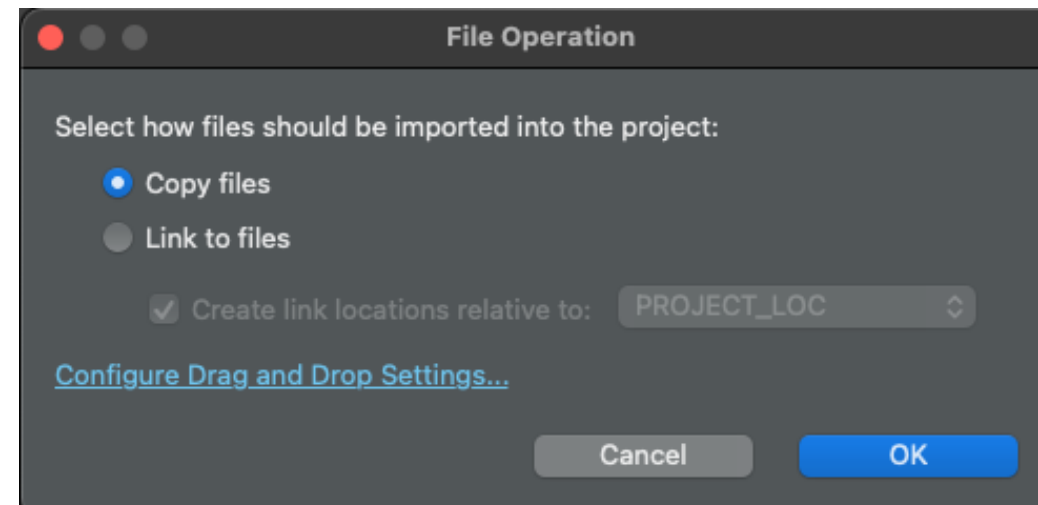
In the **assets folder**, create a new folder and name it **Images**.



4.7 Create a header as sharedComponent

Then, drag and drop the new image in the images folder.

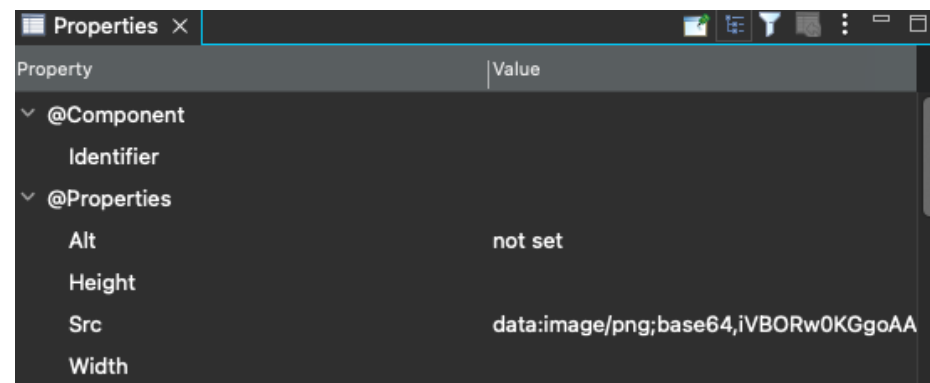
When asked, select **Copy files** to copy the image in the folder



To avoid resizing the image, add an image with the final size you need.
Here a 32x32 pixels image is better.



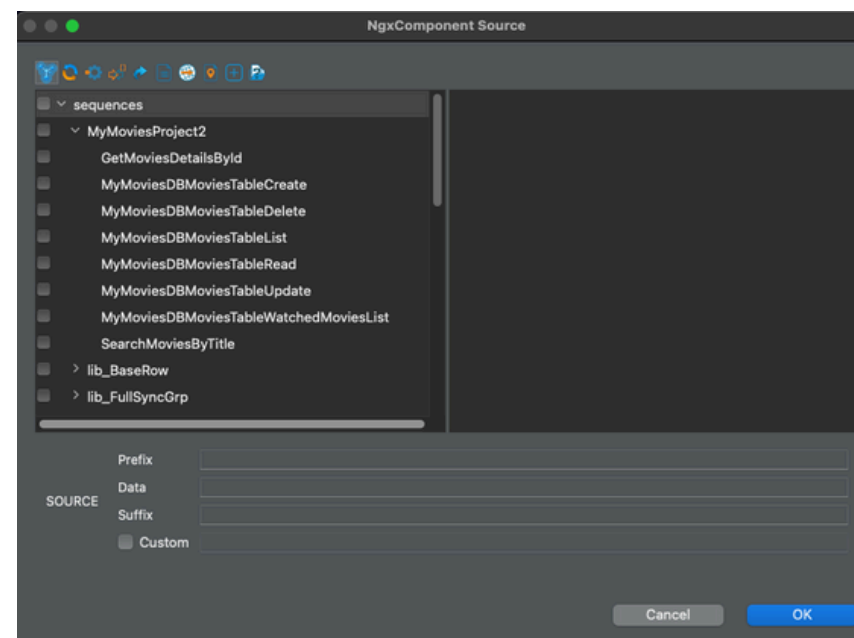
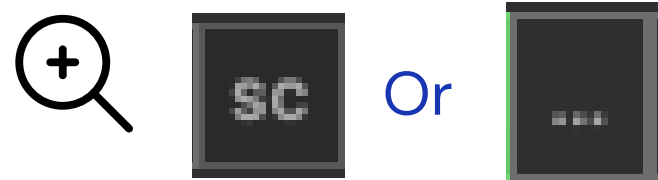
4.7 Create a header as sharedComponent



In the **Image's** properties, click on the **Src** property to display the buttons.



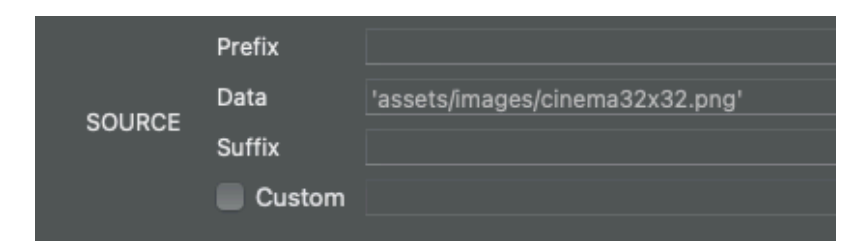
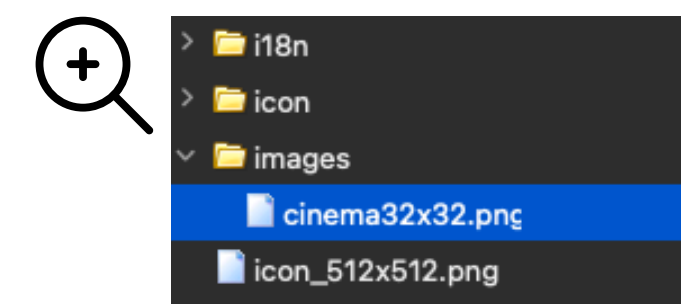
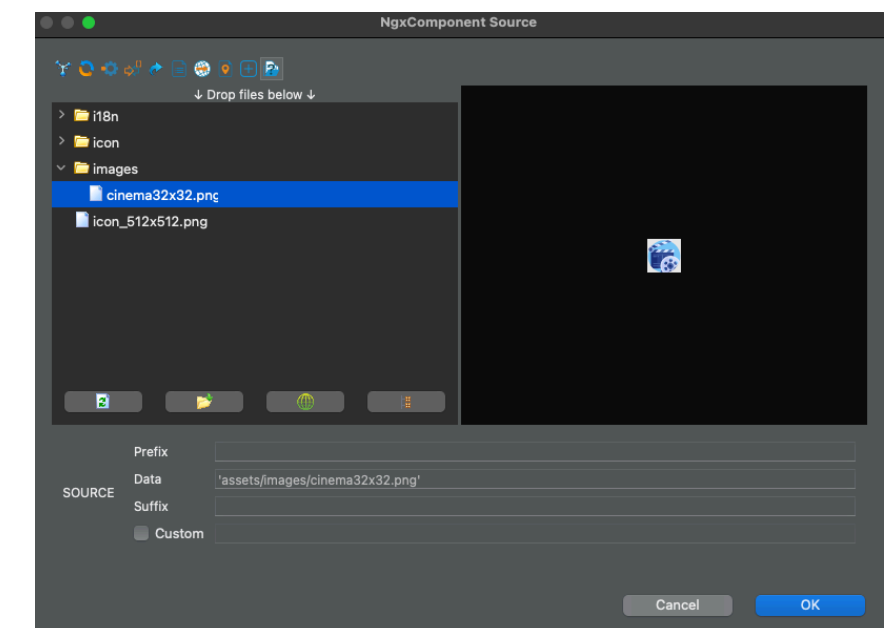
Click on ... or on **SC** to open the **NgxComponent Source** panel.



In the **NgxComponent Source** panel, click on the **File?** icon to display the **sources** in **assets** folder.

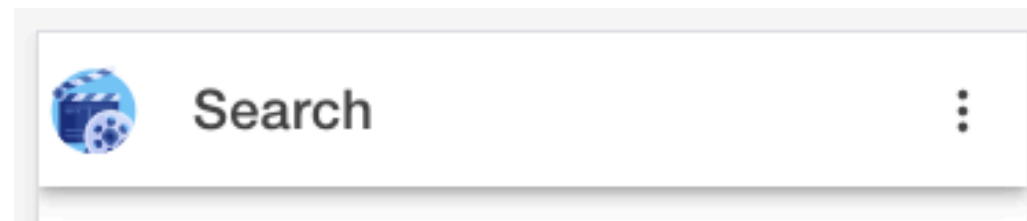


Select the image in the **images** folder.

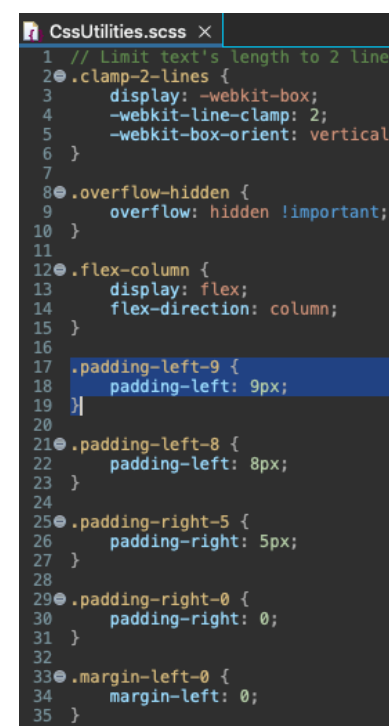
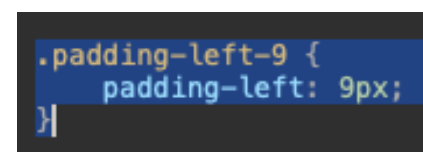
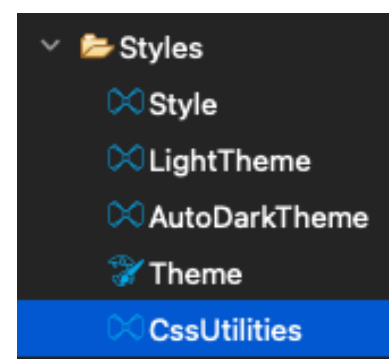


4.7 Create a header as sharedComponent

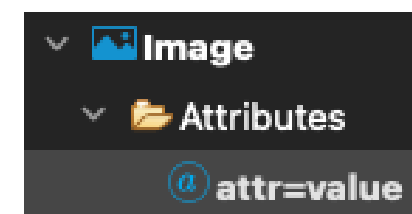
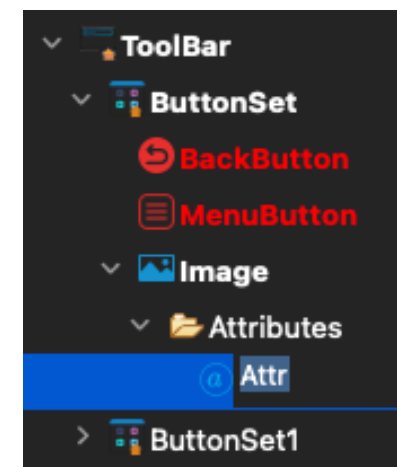
The image appears in the **Header**, a little too much on the left.



In **CssUtilities**,
add a class named **.padding-left-9**
with a padding-left of 9px.



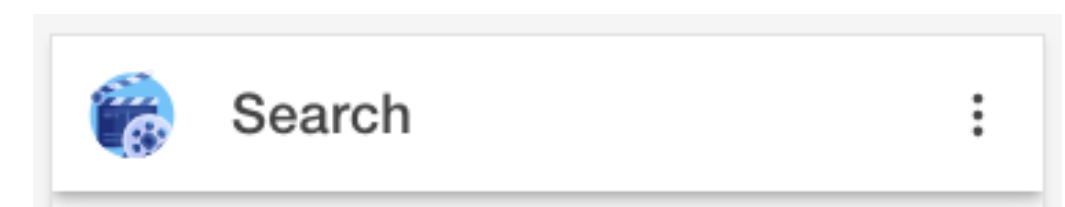
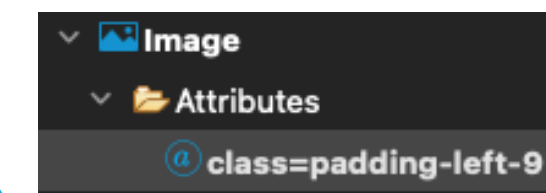
In the treeview,
add an **Attr** component
to the **Image**.



Property	Value
Base properties	
Attr name	attr
Attr value	value

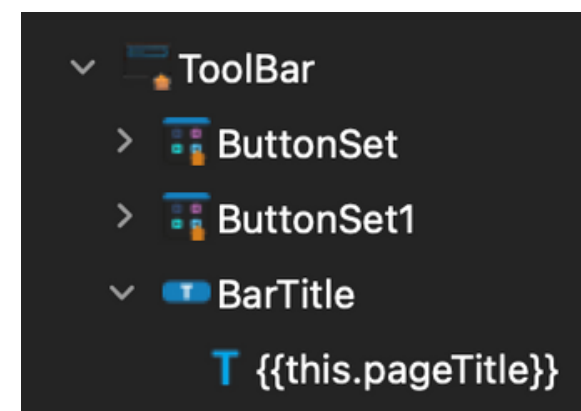
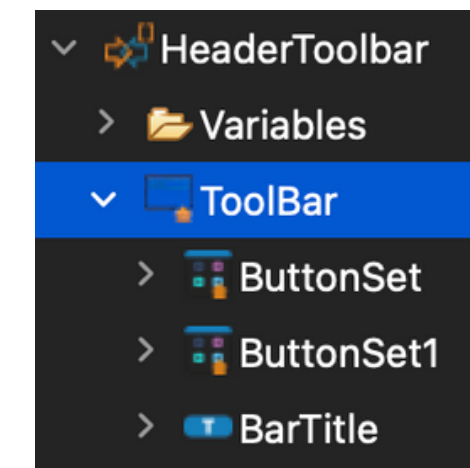
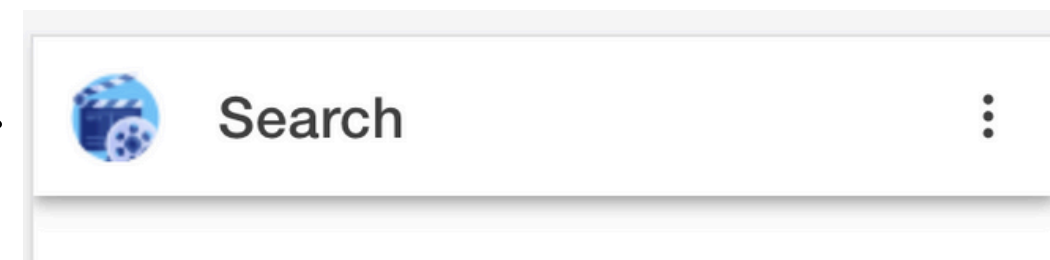
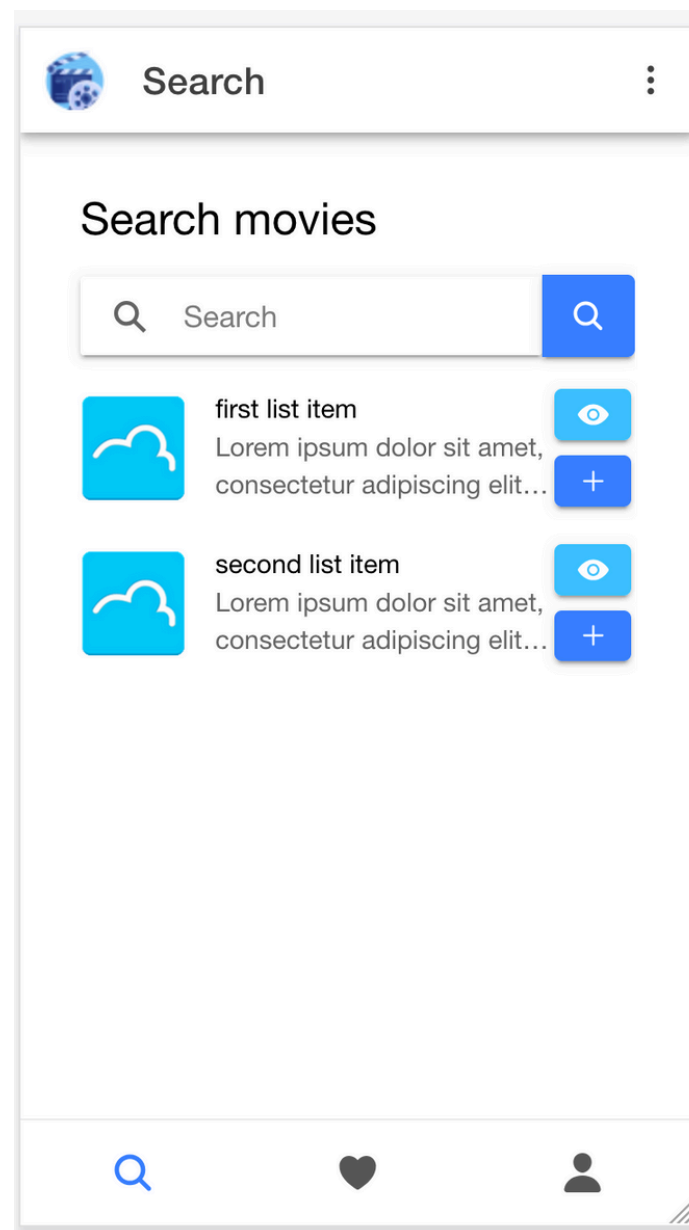
In the **Attr** component's properties,
change the value
of **Attr name** to **class**,
of **Attr value** to **padding-left-9**.

Property	Value
Base properties	
Attr name	class
Attr value	padding-left-9

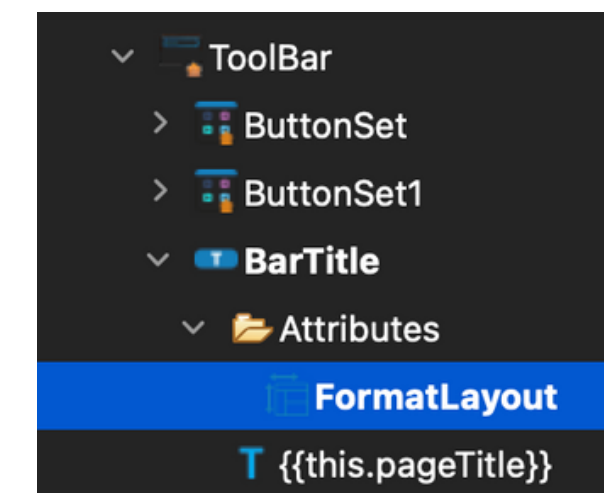
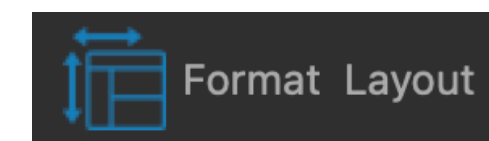


4.7 Create a header as sharedComponent

To center the **page title** in the **ToolBar** of **HeaderToolBar**, we need to center the **Text element** inside **BarTitle** with a **FormatLayout** component.



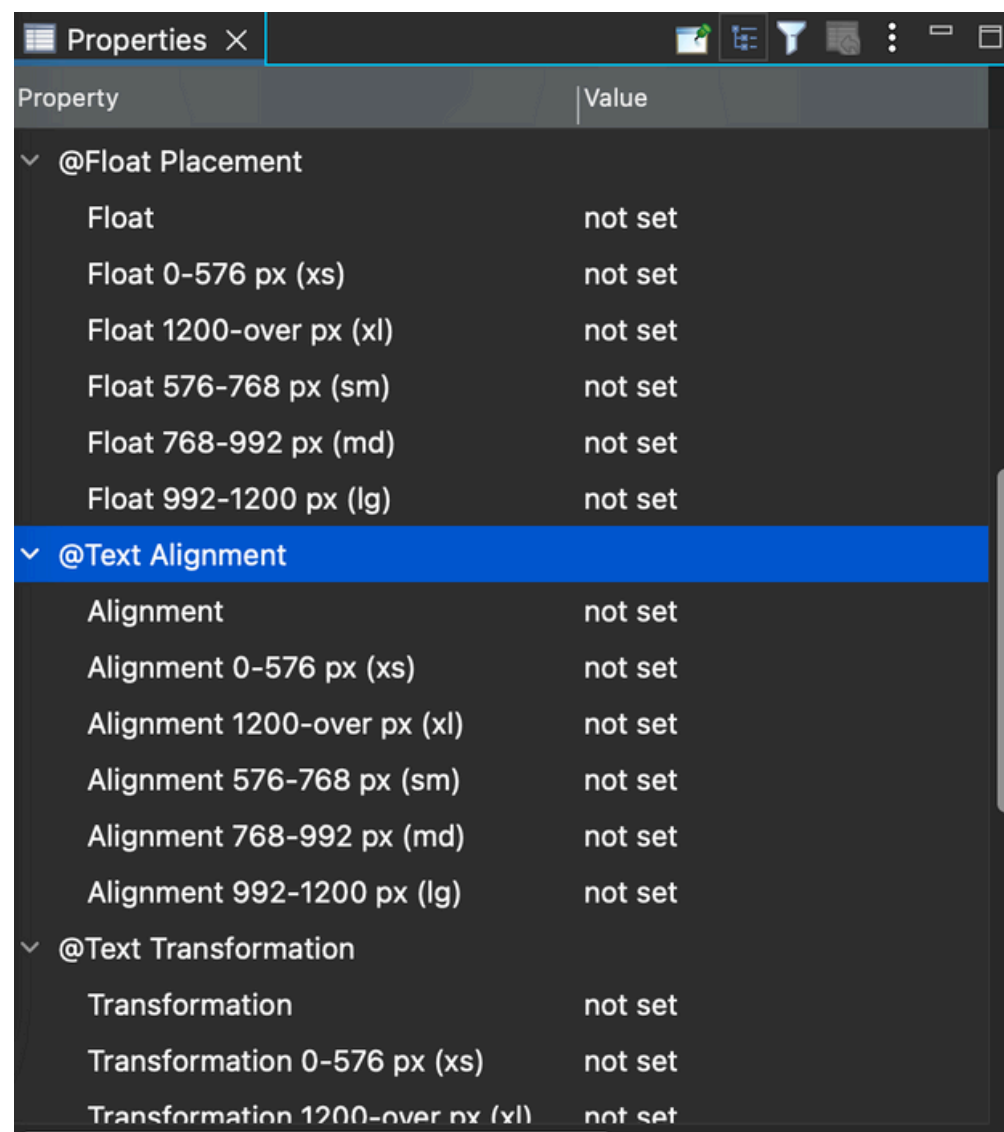
In **BarTitle** inside the **ToolBar**, add a **FormatLayout** component.



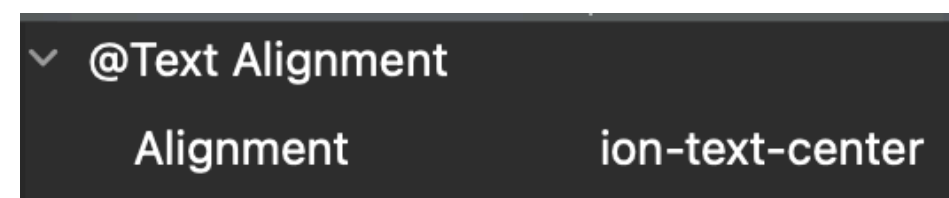
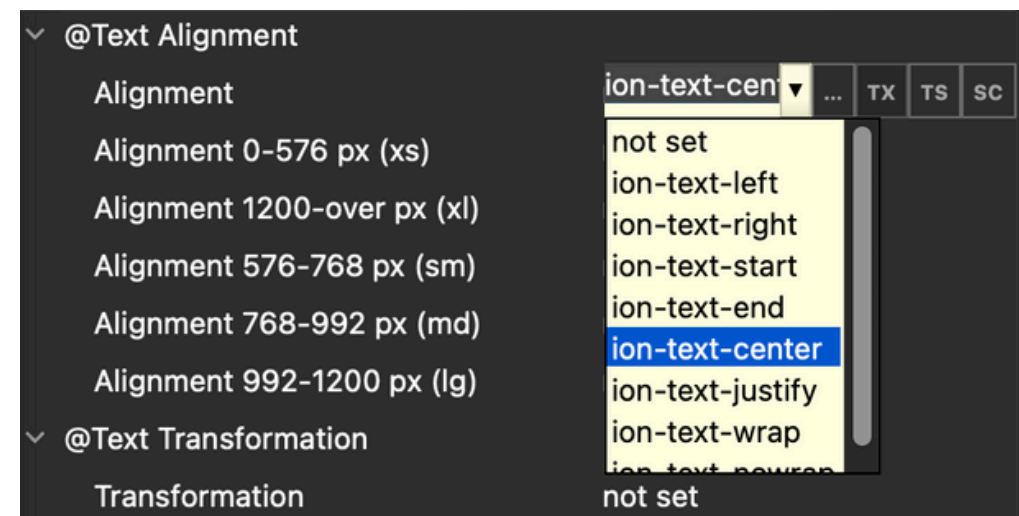
4.7 Create a header as sharedComponent

In FormatLayout's

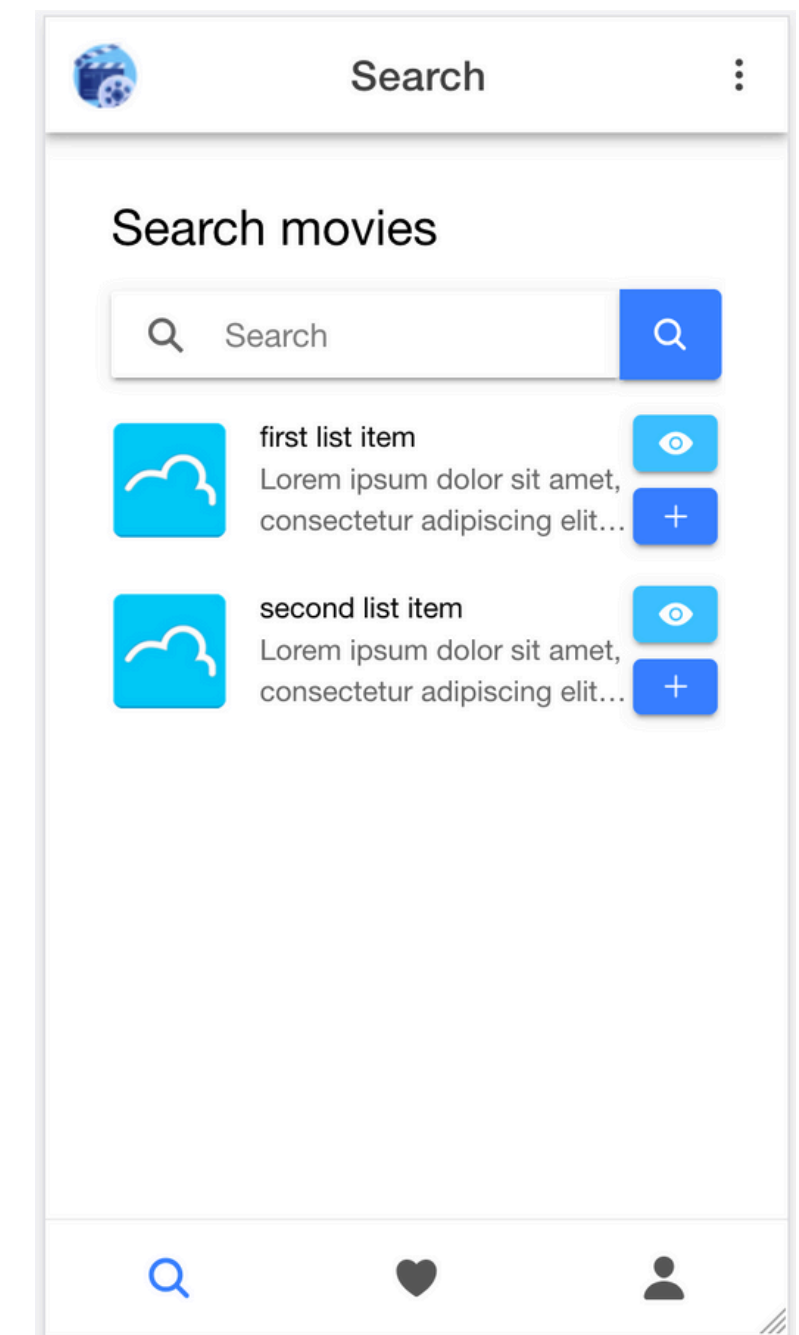
@Text Alignment properties



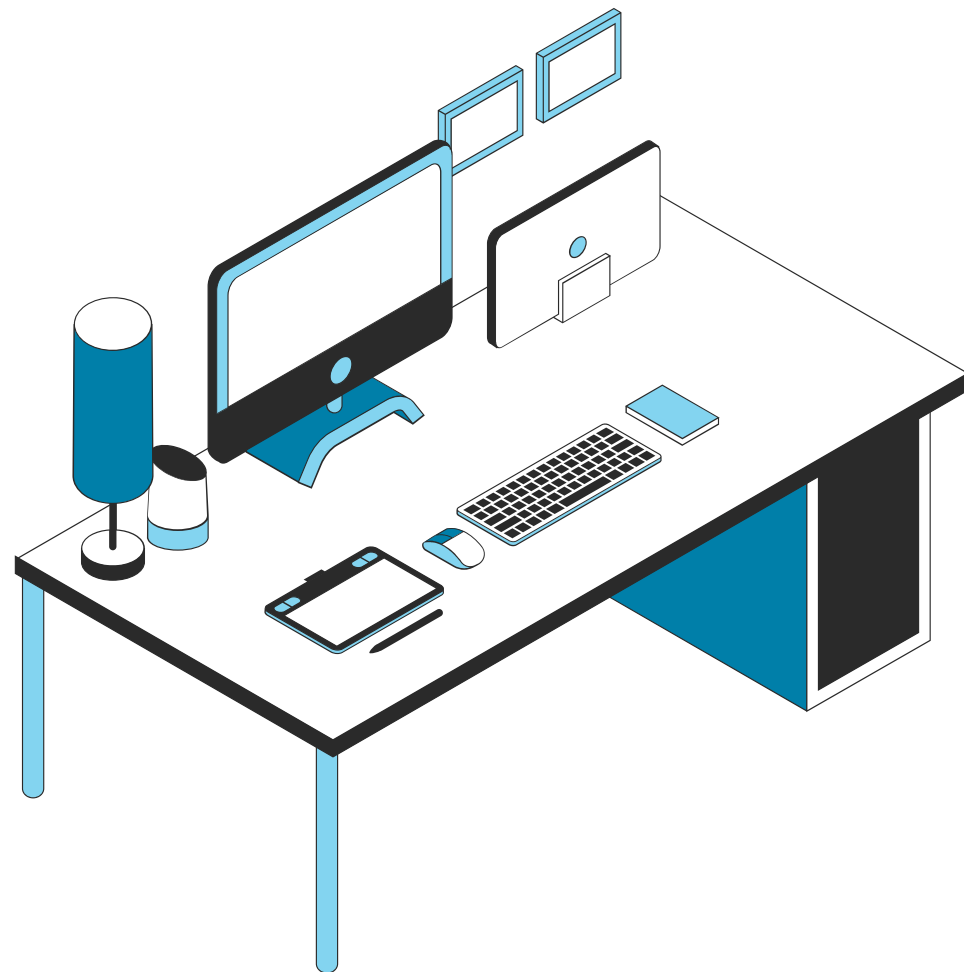
Set the **Alignment** property to **ion-text-center**.



The page title is centered.



5 – Directives & Binding Data



5.1 Directives & Directive component

5.2 ForEach Directive

5.3 Binding the ForEach Directive to a sequence

5.4 Binding segments with the Switch Directive

5.1 Directives & Directive component



Defines a **Directive component**.

Angular's **structural directives** are responsible for HTML layout.

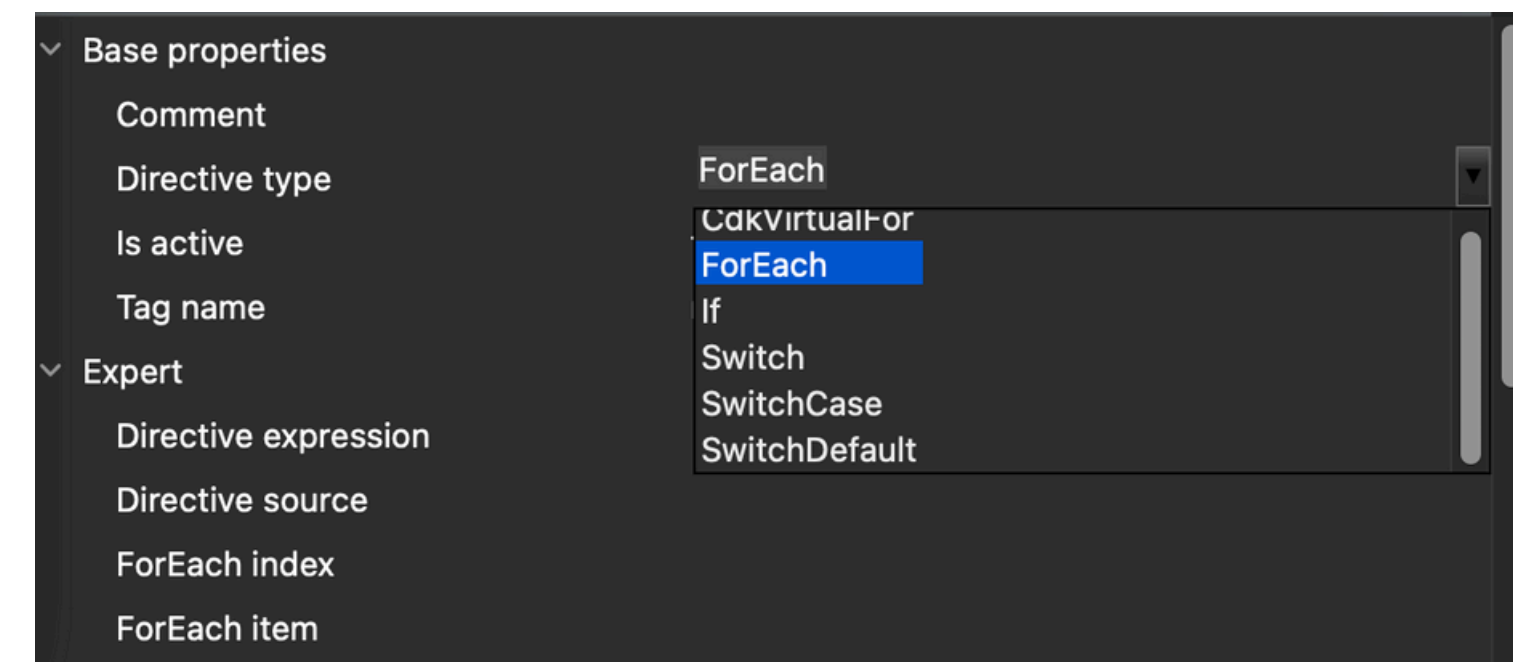
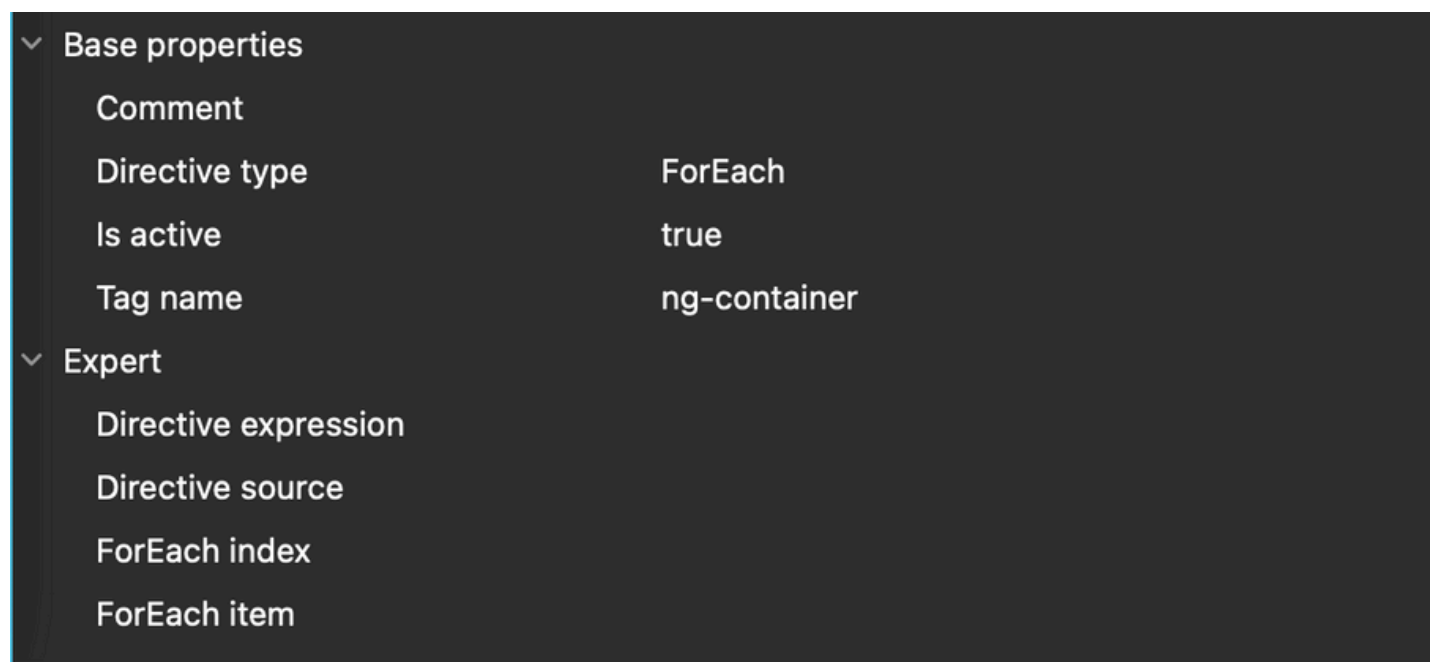
They shape or reshape the DOM's structure by adding, removing, or manipulating elements.

Directives can be:

- **ForEach** : **each occurrence** of a source collection generates an **UI Component**.
- **If** : the UI Component is displayed **If** the source is **true**. (will be showed later in the tutorial)
- **Switch, SwitchCase, SwichtDefault** : the **UI Component** is displayed with the **Switch options**.

By default, a **Directive component** is created as **ForEach**.

You can change it with **Directive Type property**.



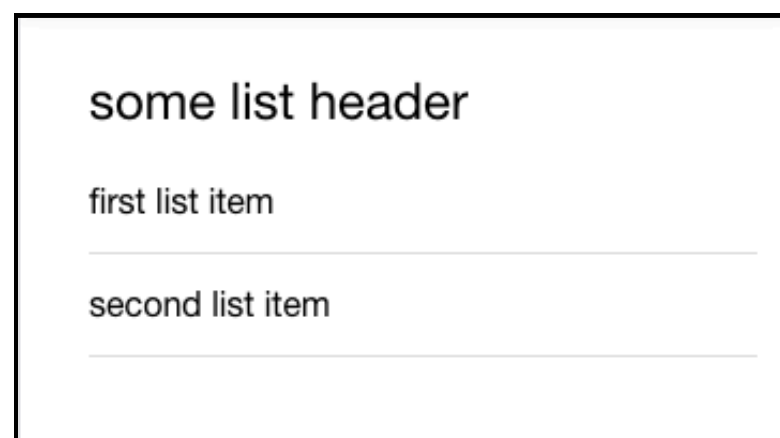
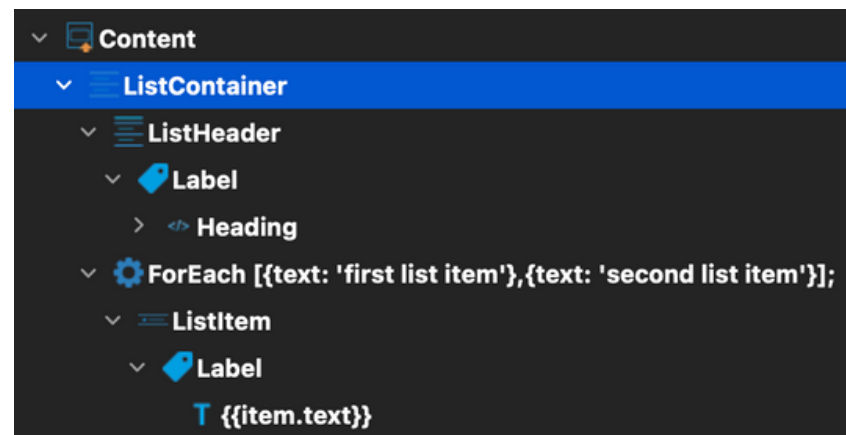
5.2 ForEach Directive

Previously, in **PageSearch**, we have added a **ListContainer** in the **Content**.

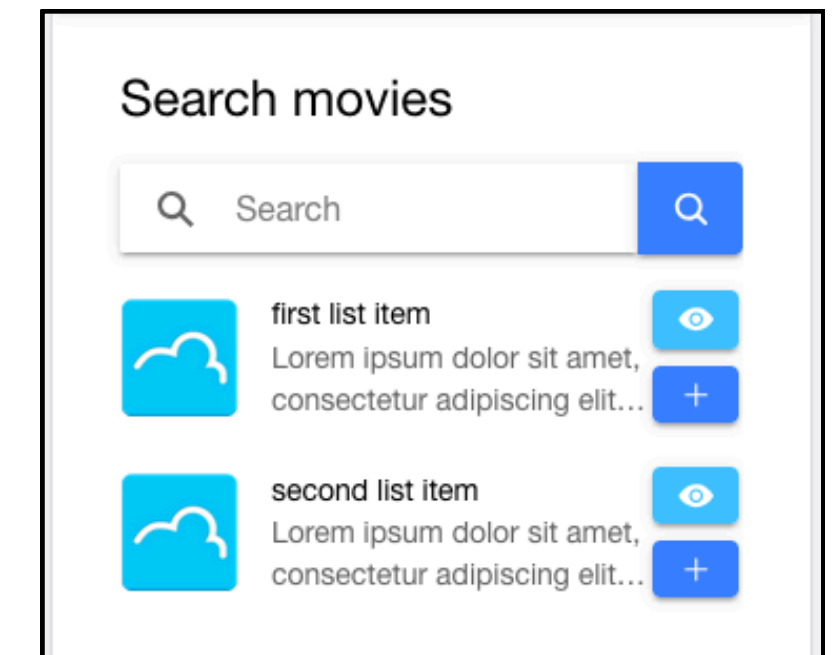
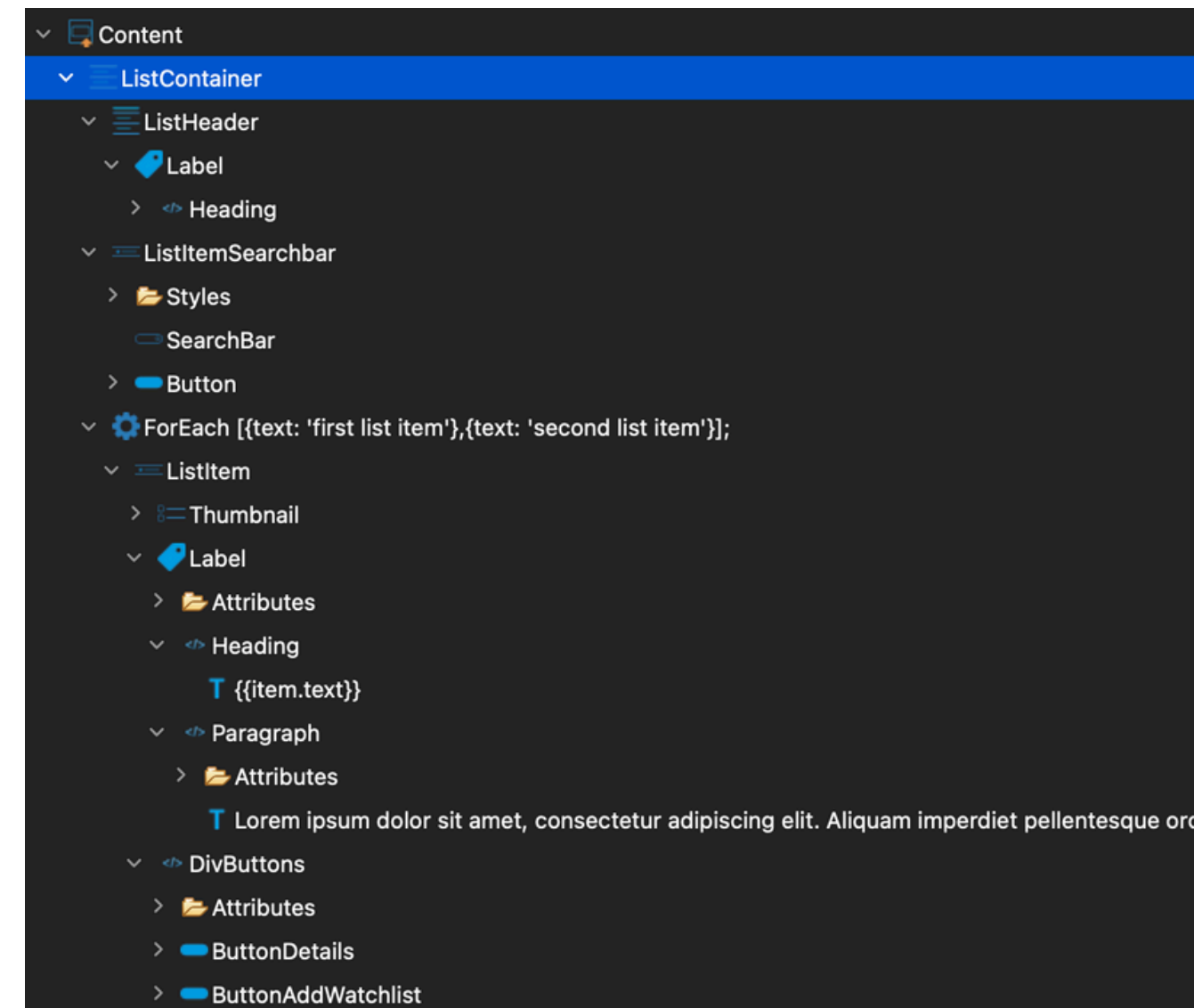
By default, the **ListContainer** is created with a **ForEach directive** containing a **ListItem**.

We have previously modified its original layout, by adding some content.

Original ListContainer

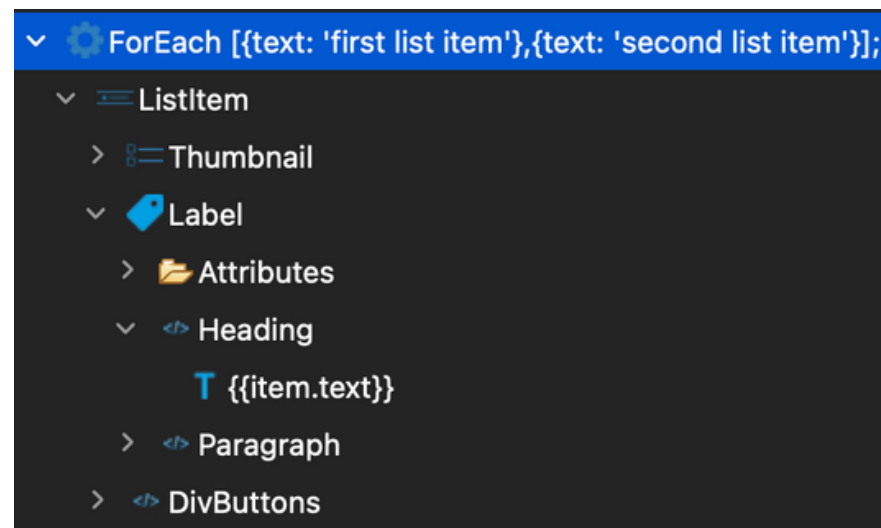


ListContainer



5.2 ForEach Directive

Let's look at the **properties by default** of the **ForEach** directive.



Base properties	
Comment	
Directive type	ForEach
Is active	true
Tag name	ng-container
Expert	
Directive expression	
Directive source	[[{text: 'first list item'}, {text: 'second list item'}]]
ForEach index	idx
ForEach item	item

- **Directive source** is the source of data for the directive.

By default, it's an array of objects in TS : `[[{text: 'first list item'}, {text: 'second list item'}]]`



- **ForEach index** and **ForEach item** are used to define the index and item names.

These names are used to access their values in other components:

ForEach index **idx**

T {{idx}}

- **ForEach index** name by default is **idx**.

Its value is used in **TS scope** as **scope.idx** and in components as `{{ idx }}`.

ForEach item **item**

T {{item.text}}

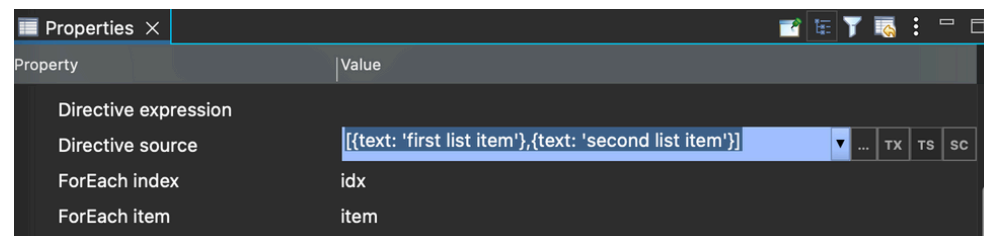
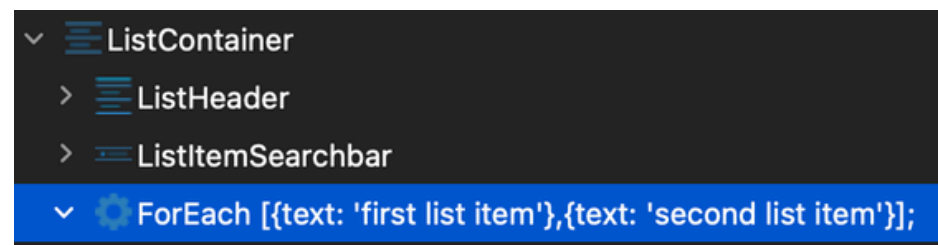
- **ForEach item** name by default is **item**.

Its value is used in **TS scope** as **scope.item** and in components as `{{ item.objectKey }}` (ex: `item.text` in the example by default)

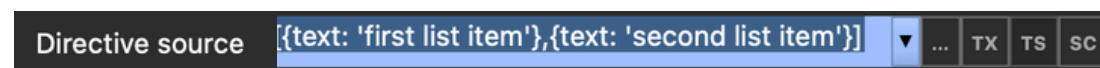


5.3 Binding the ForEach Directive to a sequence

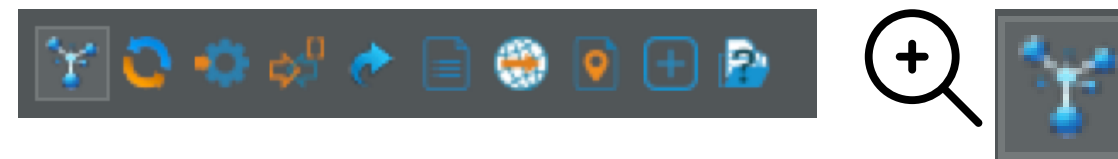
Let's change the directive source to display a list of movies with the sequence **SearchMoviesByTitle**.



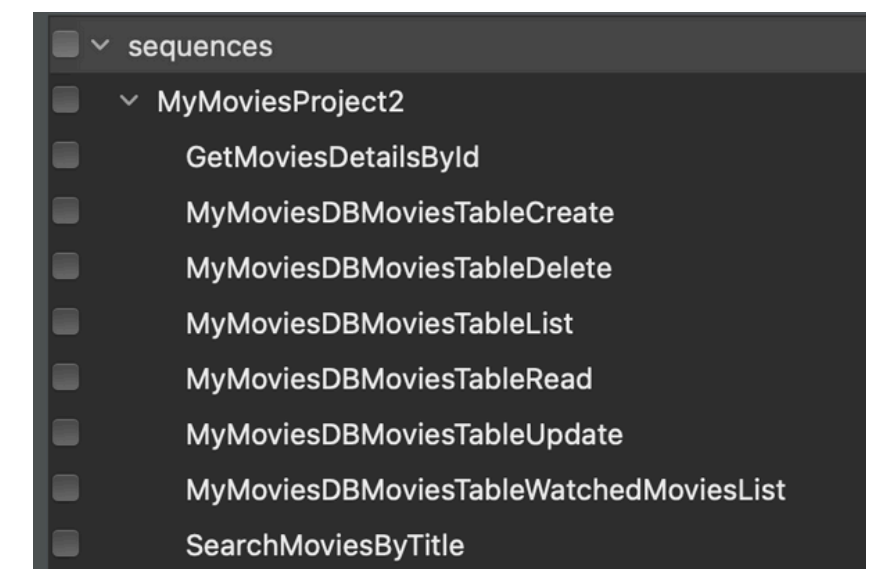
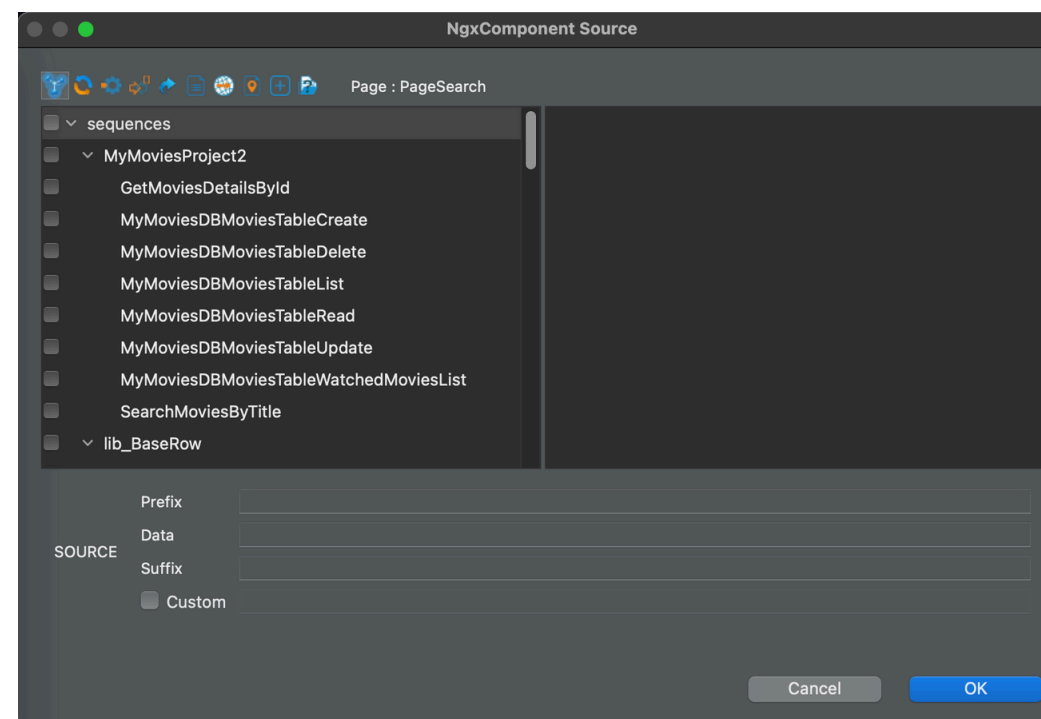
In the **Directive source** property, click on ... or on **SC** to open the **NgxComponent Source** panel.



In the **NgxComponent Source** panel, the sequences are the source selected by default.

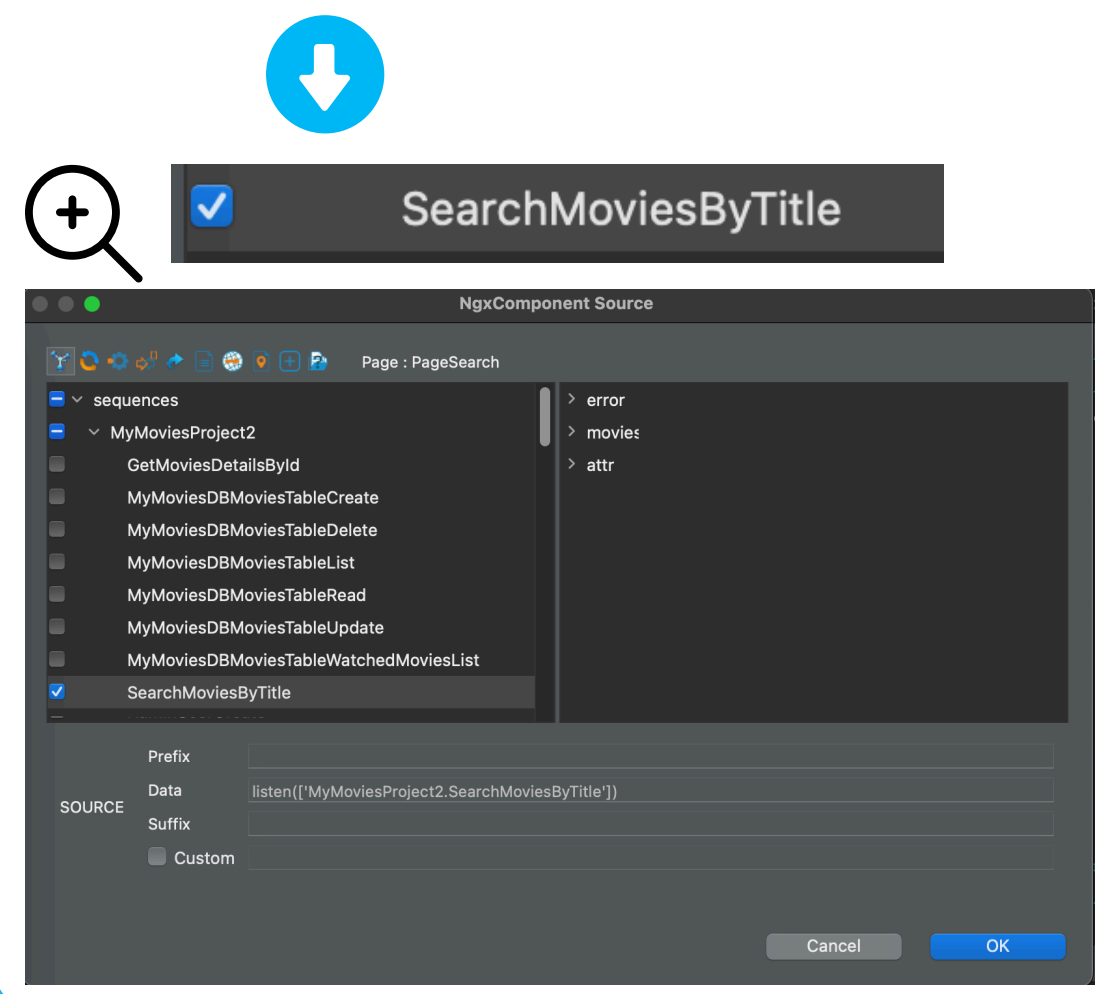


The sequences we created in our project (and all the sequences present in the workspace in other projects) are available to be used as sources.

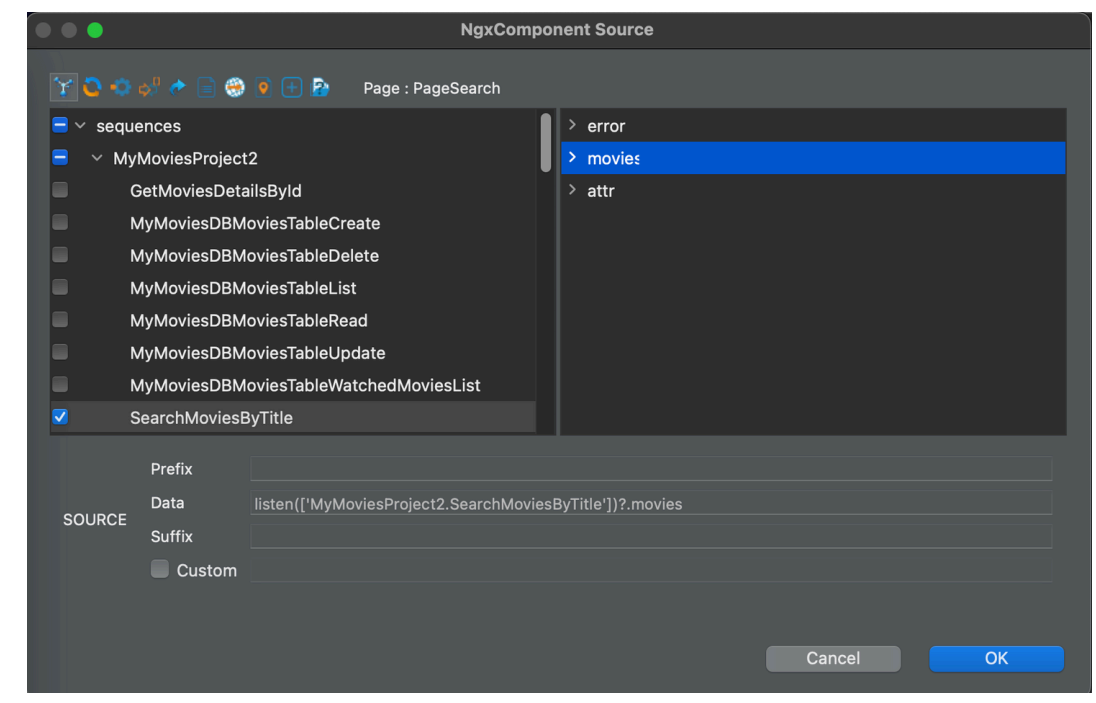
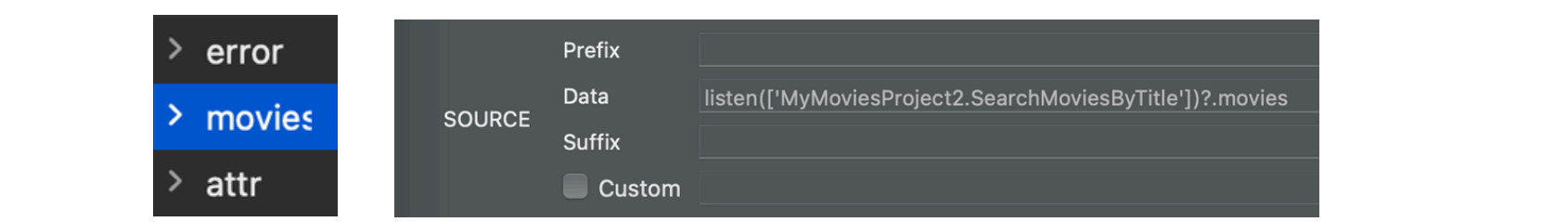


5.3 Binding the ForEach Directive to a sequence

In the **NgxComponent Source** panel, click on **SearchMoviesByTitle** to select the sequence as source.



Then click on **movies** to select it as source.

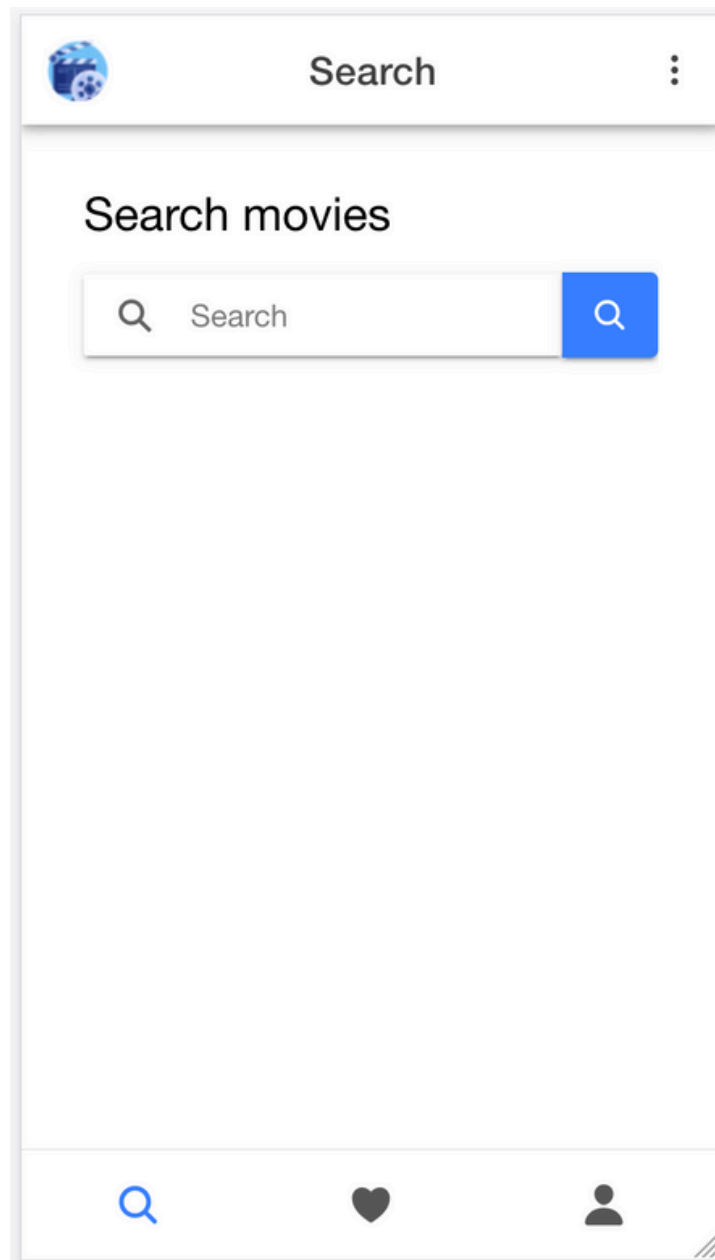


Directive expression	
Directive source	MyMoviesProject2.SearchMoviesByTitle->movies
ForEach index	idx
ForEach item	item

The **movies Array** of the sequence **SearchMoviesByTitle** appear as **Directive source** in properties.



5.3 Binding the ForEach Directive to a sequence

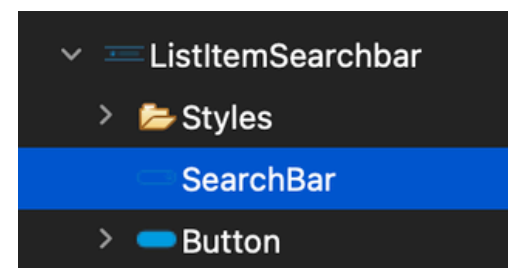


As for now, there are no **ListItems** in the **SearchPage** because

- the **movies Array** is **empty**
- our components are **no longer binded** to the **Directive source**

First we need to **get data** in the **movies Array**

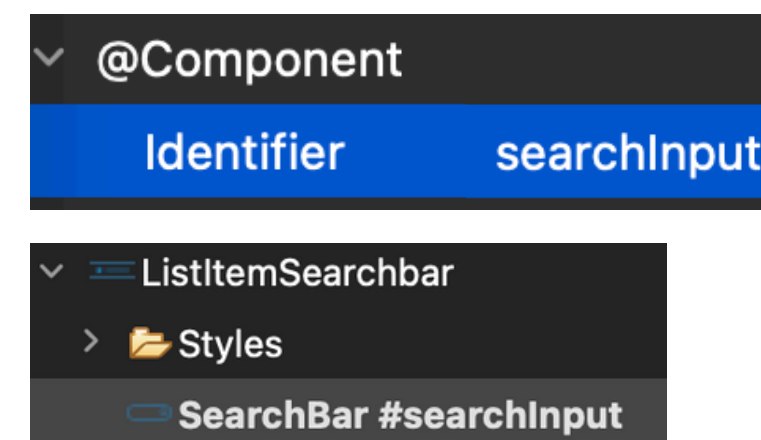
- by entering a title as **variable** in the **searchbar**
- and calling the sequence **SearchMoviesByTitle** by clicking on the **SearchBar button**.



Property	Value
@Component	
Identifier	
@Control	
Binding	not set
Control name	name1727776307091



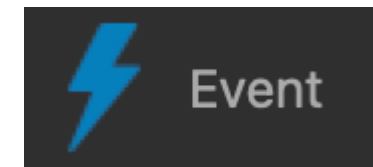
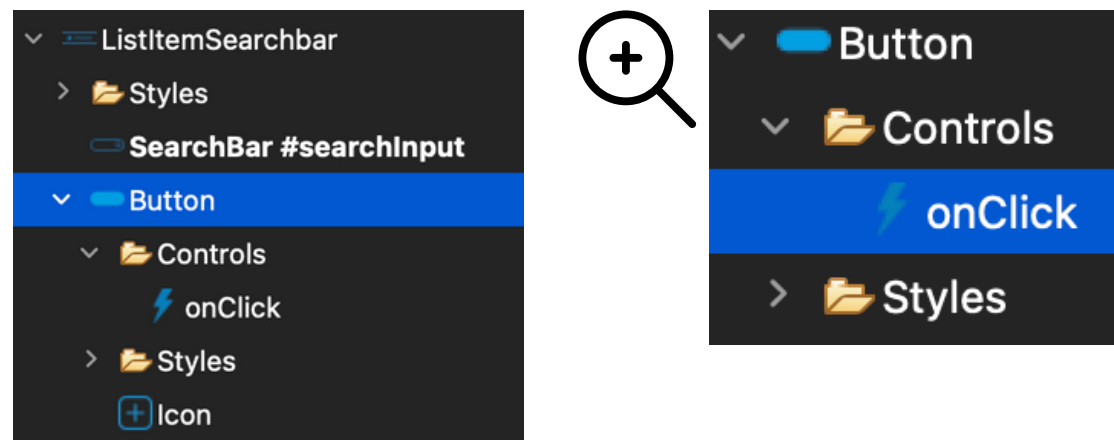
In the **SearchBar Identifier** property, enter **searchInput** as value.



5.3 Binding the ForEach Directive to a sequence

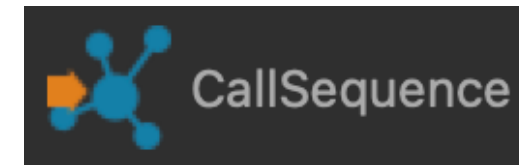
Now we need to call the sequence **SearchMoviesByTitle** by clicking on the **SearchBar** button.

Inside the **Button**, in the **Controls** folder, there is an **onClick** event by default.



Event

onClick is an **Event** component..



CallSequence

CallSequence is an **action** component.

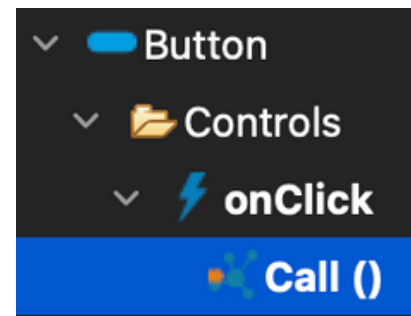
We will give more details about these components later in the tutorial



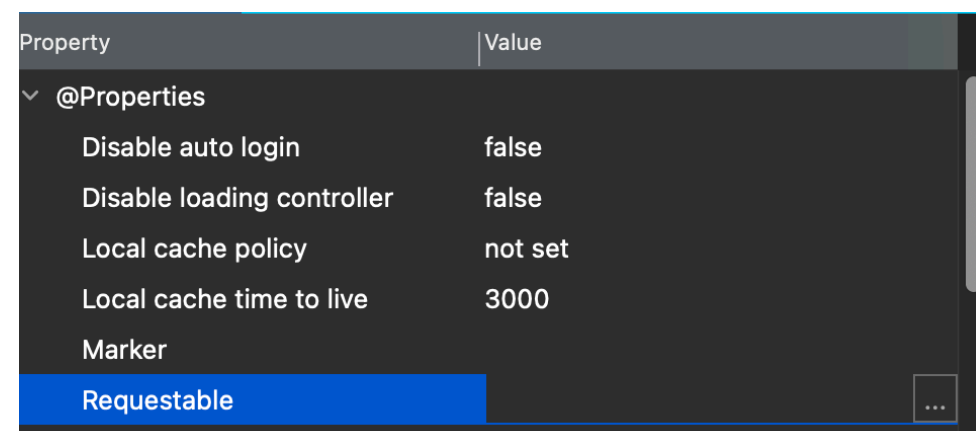
In the **onClick** event, add a **CallSequence** component.



5.3 Binding the ForEach Directive to a sequence



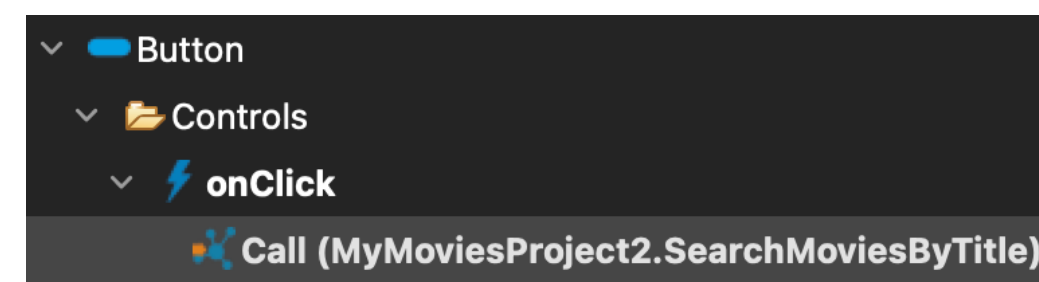
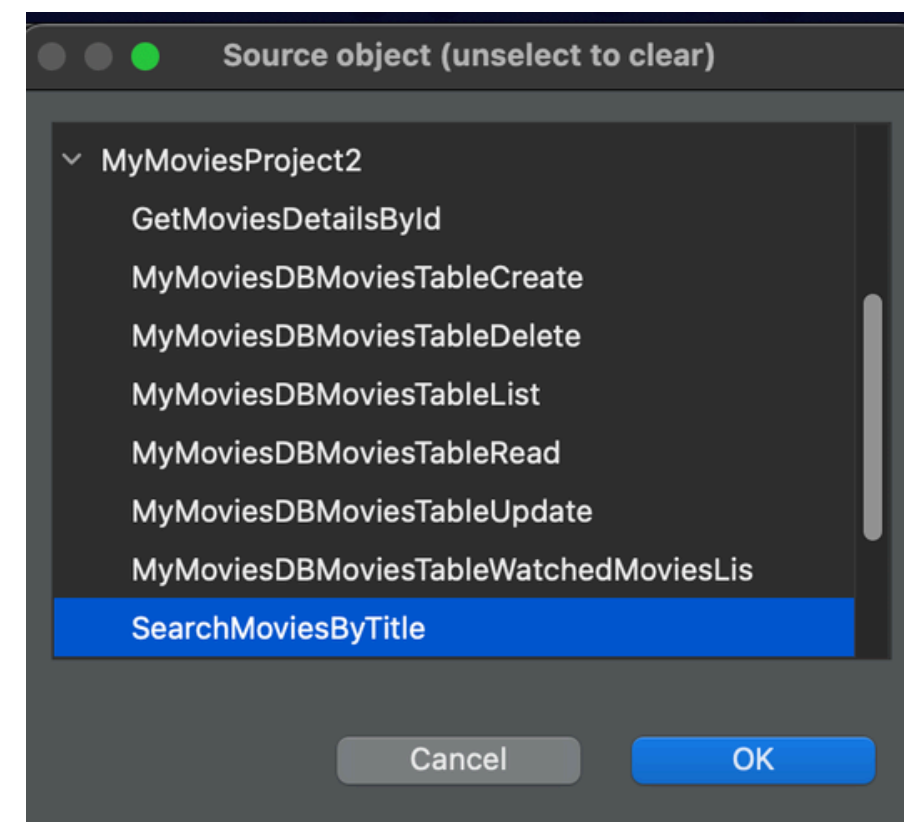
In the **CallSequence** component's properties, in the **Requestable** property, click on ... to open the **Source Object** panel.



Requestable



In the **Source Object** panel, select **SearchMoviesByTitle**.

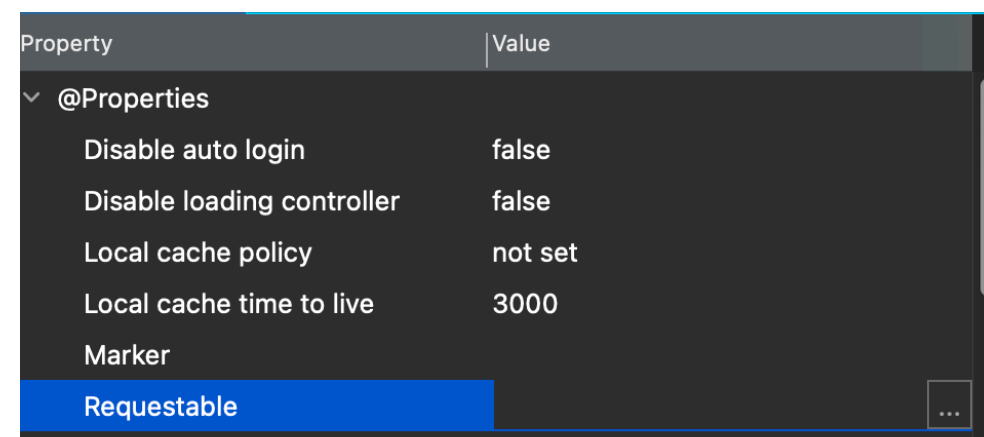


5.3 Binding the ForEach Directive to a sequence

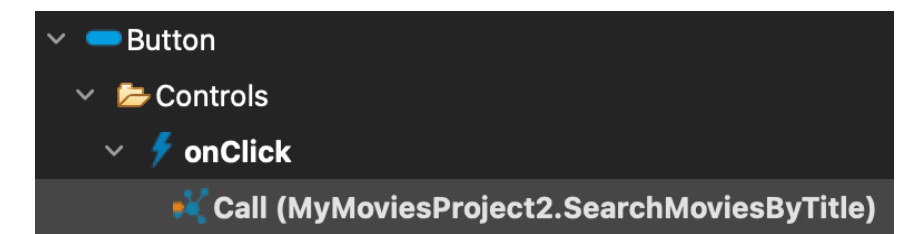
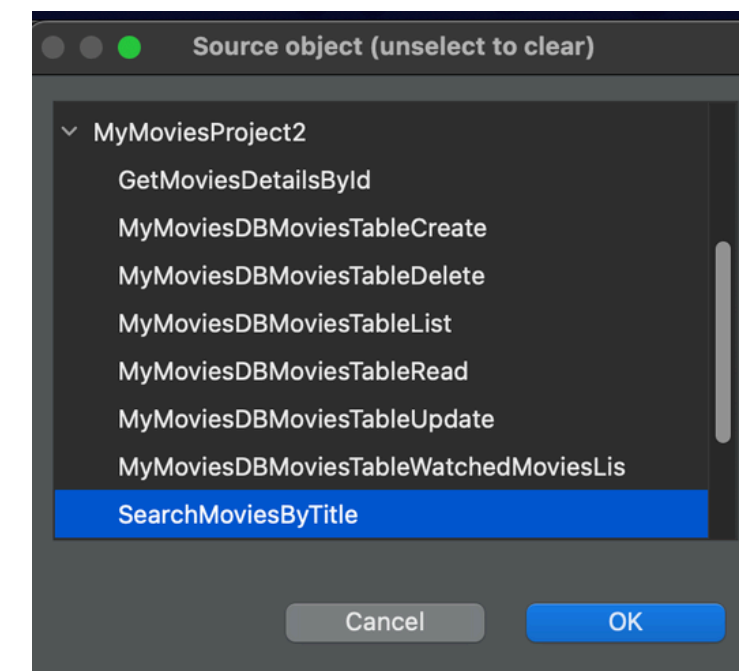
In the **onClick** event, add a **CallSequence** action component.



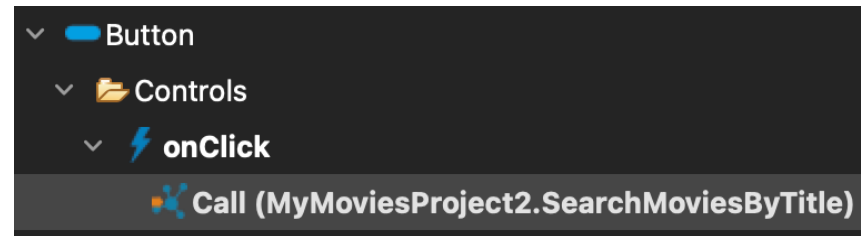
In the **CallSequence** component's properties, in the **Requestable** property, click on ... to open the **Source Object** panel.



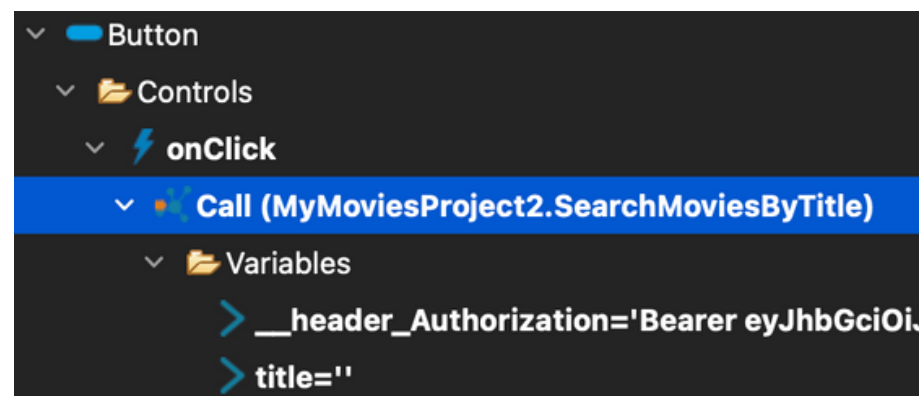
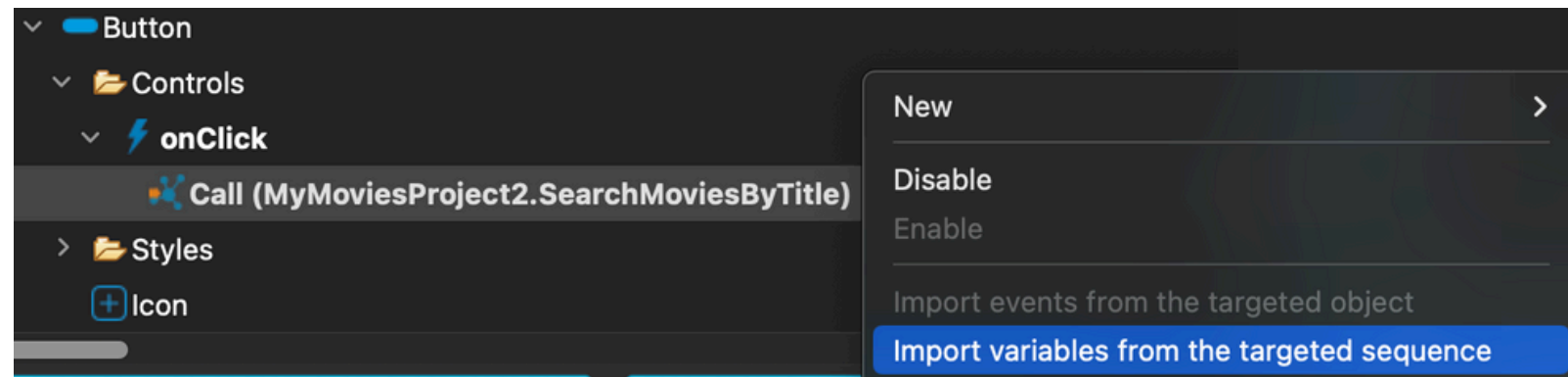
In the **Source Object** panel, select **SearchMoviesByTitle**.



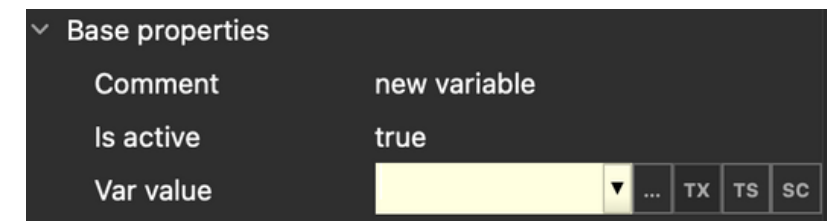
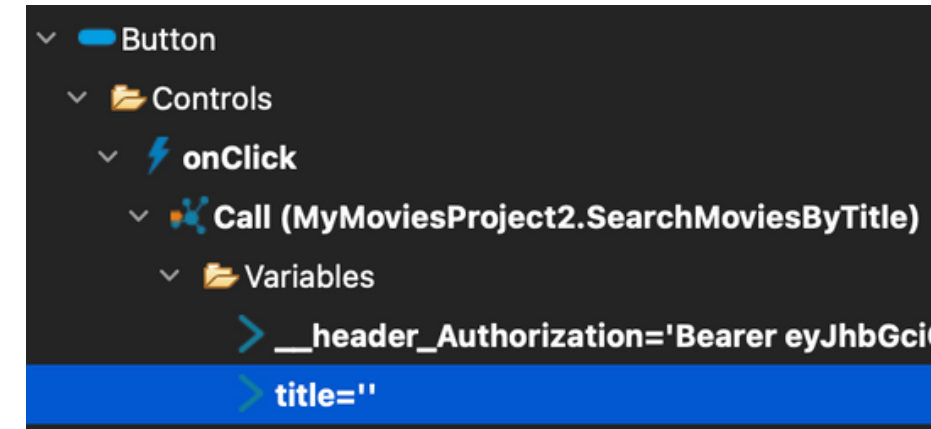
5.3 Binding the ForEach Directive to a sequence



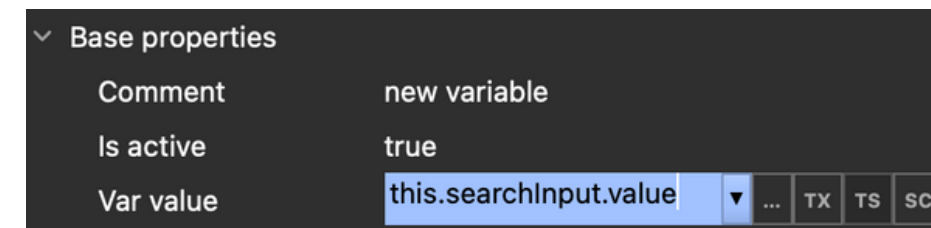
Right-click on the sequence call,
and select **Import variables**
from the targeted sequence.



The variables
are now imported
in the sequence call.

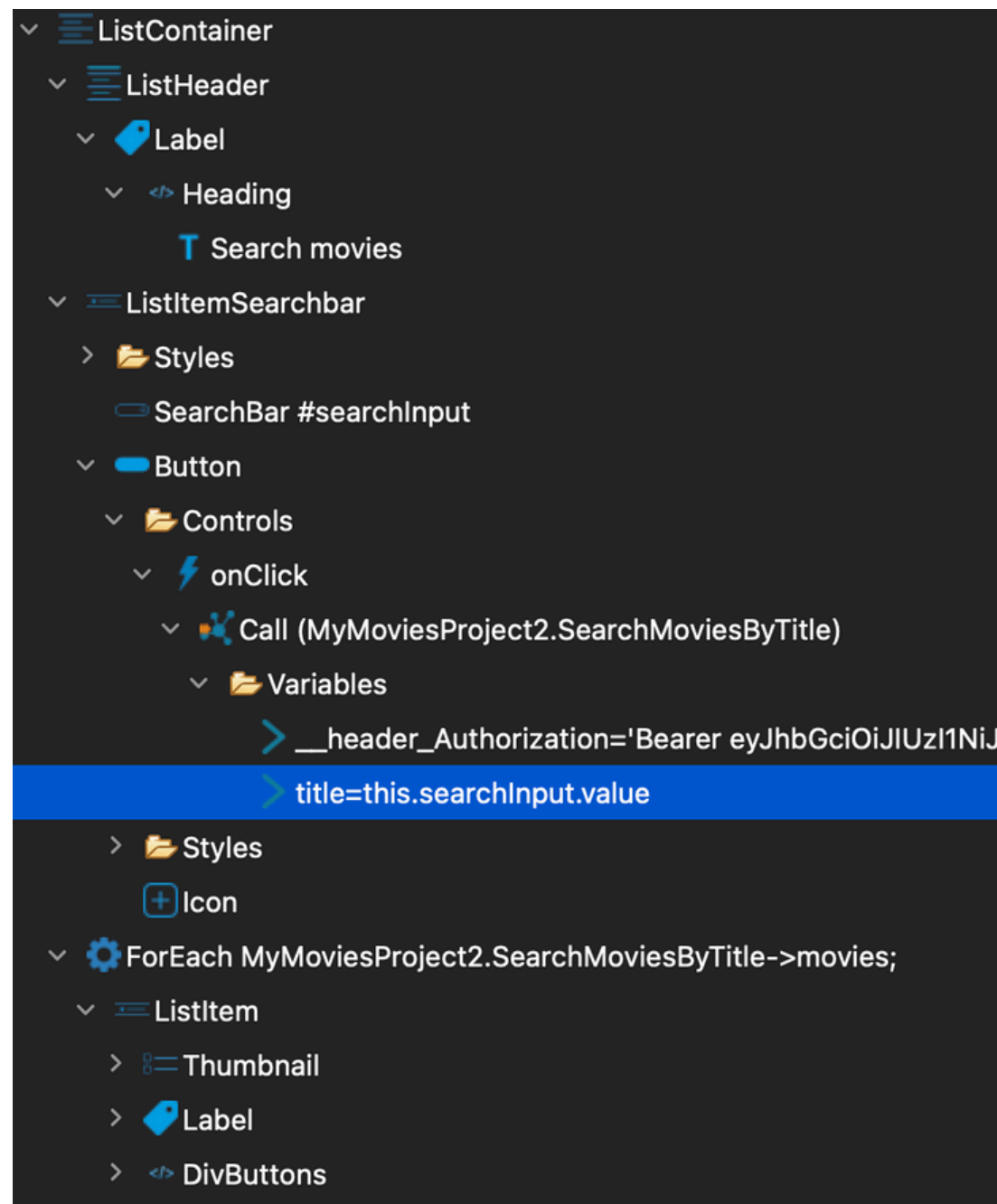


In the **Var value** property of the **title** variable,
enter **this.searchInput.value** as value.

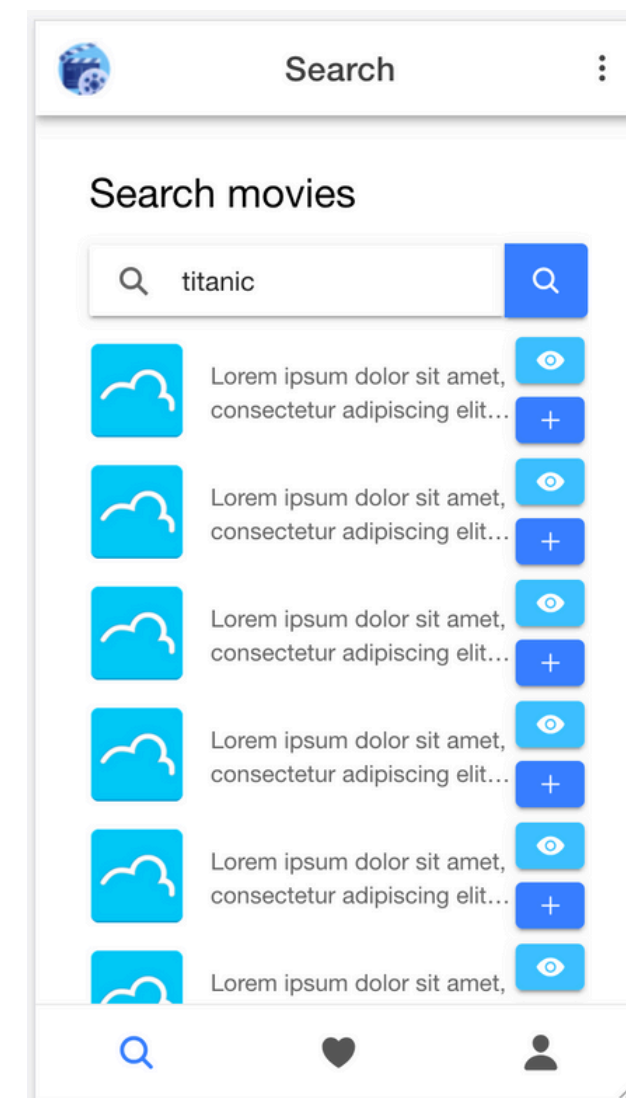
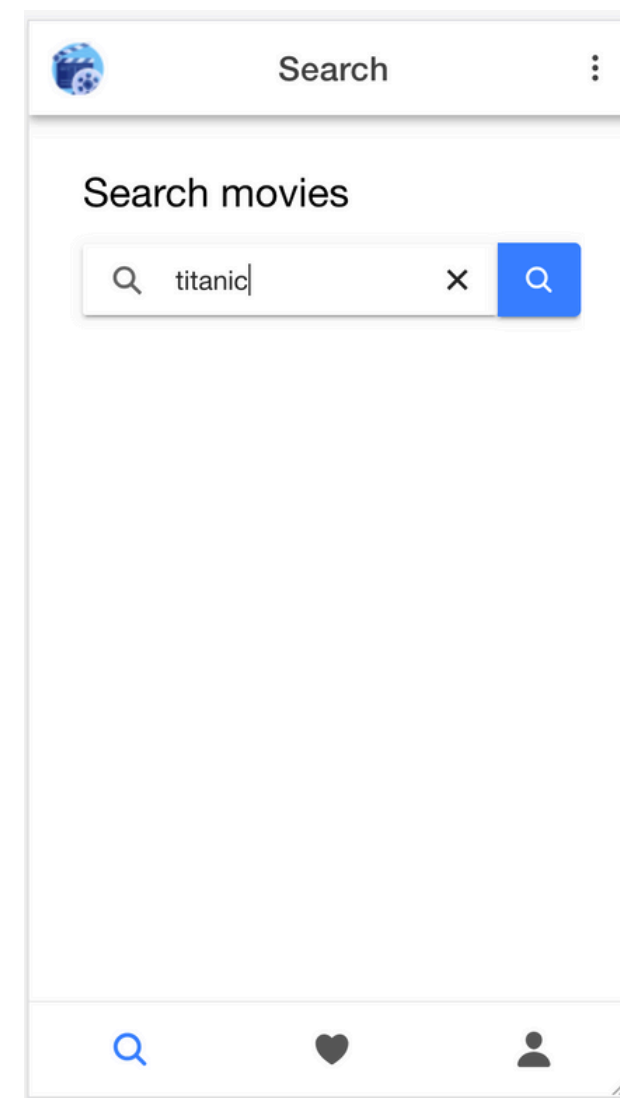


5.3 Binding the ForEach Directive to a sequence

The sequence call is now binded to the **SearchBar** input.

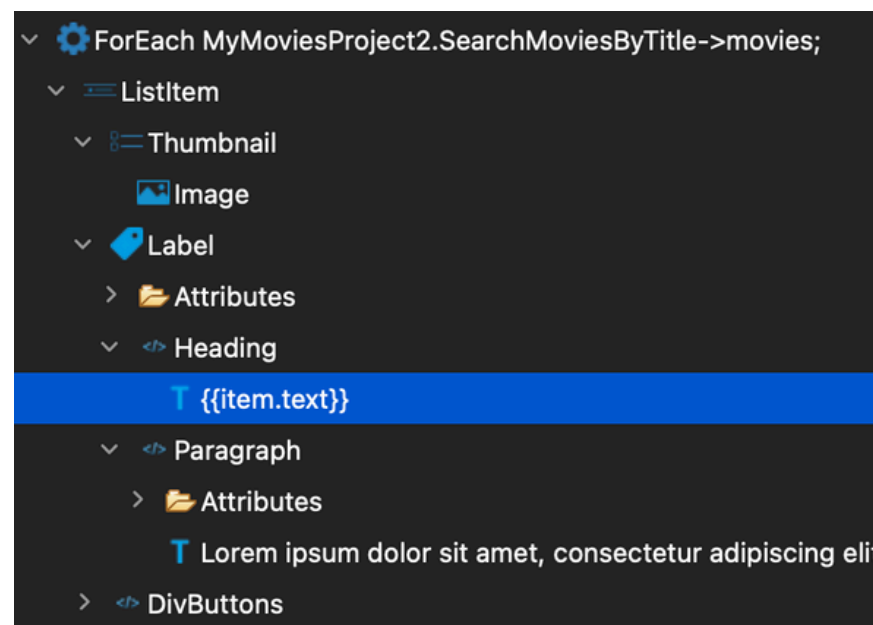


When we enter a value in the **SearchBar**, the results are called and a series of **ListItems** is displayed but not the values of each result in each **ListItem**.

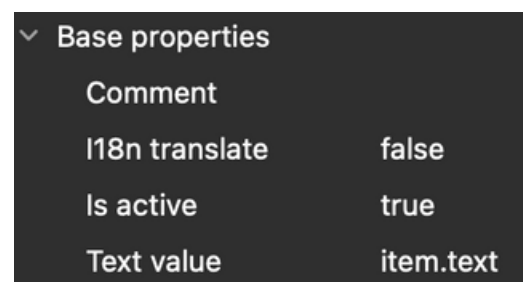


5.3 Binding the ForEach Directive to a sequence

We need to bind the **ListItem components** (Heading, Paragraph, Image) to the **Directive source** to display the title, description and image of each movie.



Click on the **text element** of **Heading** to display its properties.

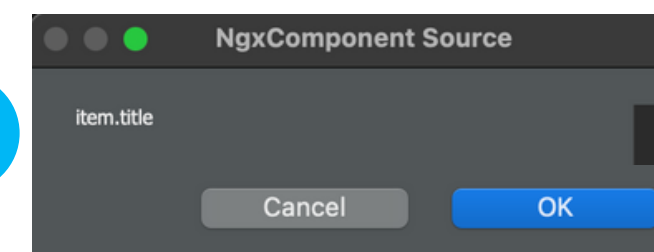
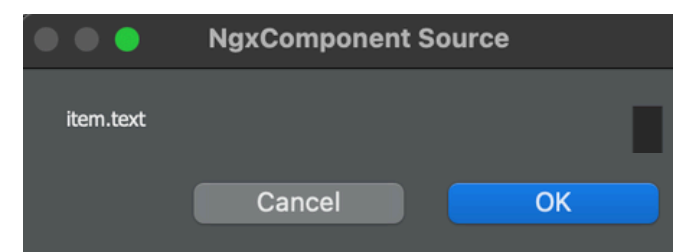


In the **Text value property**, the value is in **TS scope** by default with the **ForEach item name** and an **object key**.



To replace this value, you can either

- type directly the **ForEach item name** and an **new object key** in this field in **TS scope**.
- click on ... to open the **NgxComponent Source panel** in TS mode to type it.



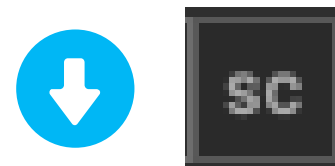
5.3 Binding the ForEach Directive to a sequence

There is another, and easiest way, to bind our components to the **Directive source**.

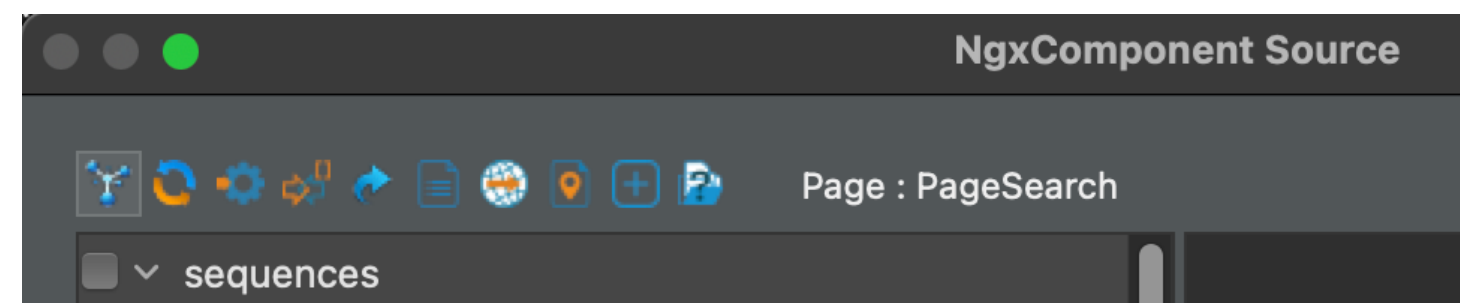
It is very useful when you don't know the object keys names.



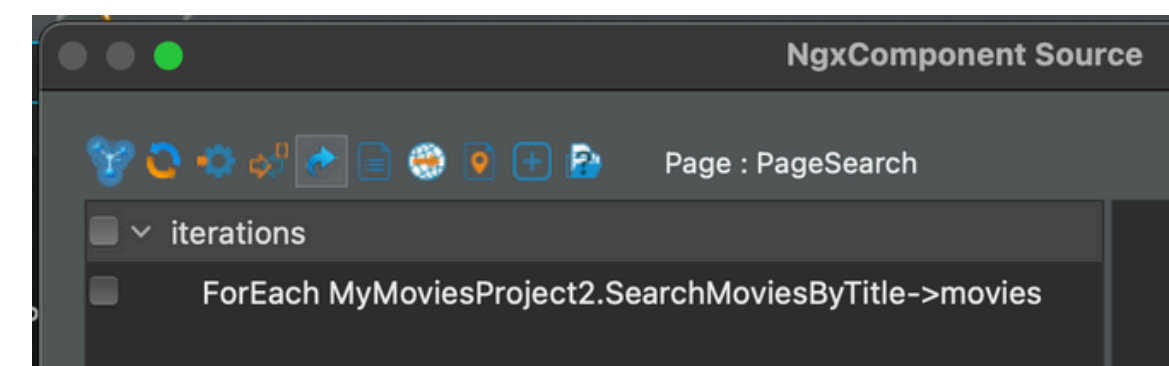
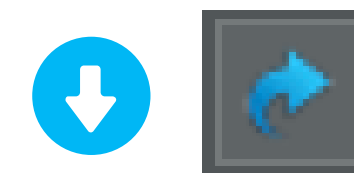
Click on **SC** to open the **NgxComponent Source** panel.



This time, the **NgxComponent Source** panel opens in **source mode**.



In the **NgxComponent Source** panel, click on the **Iterator** icon to display the **ForEach directive sources** available.



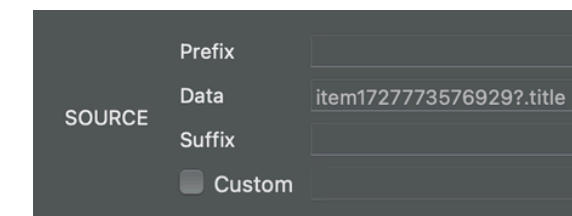
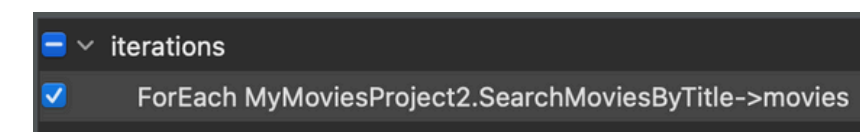
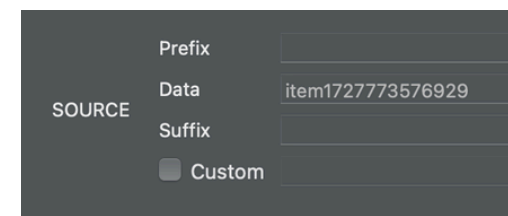
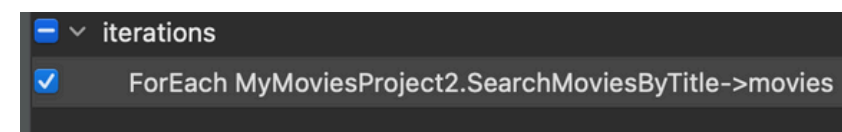
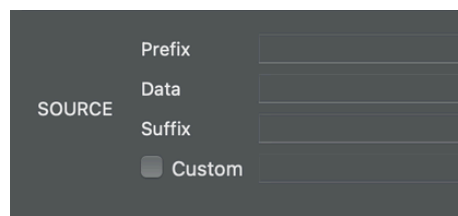
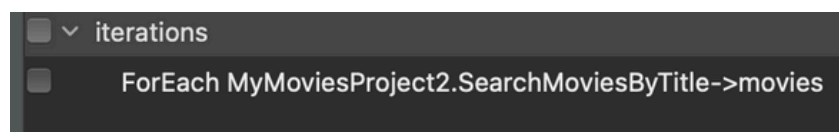
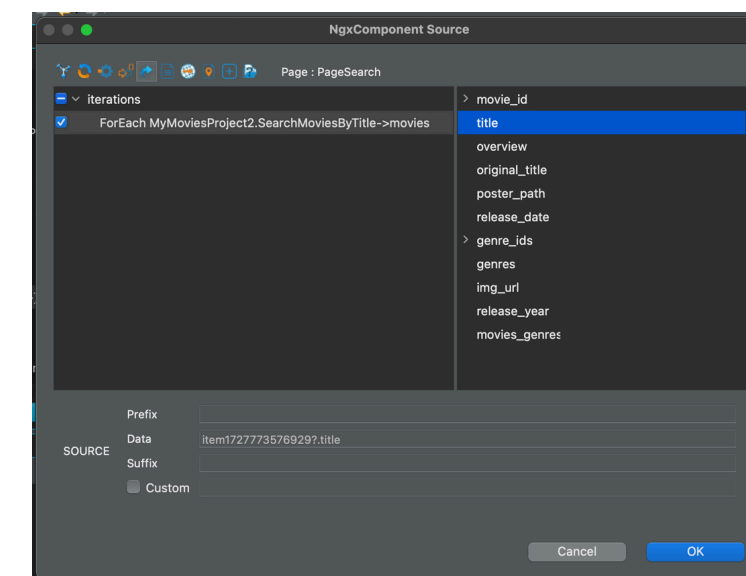
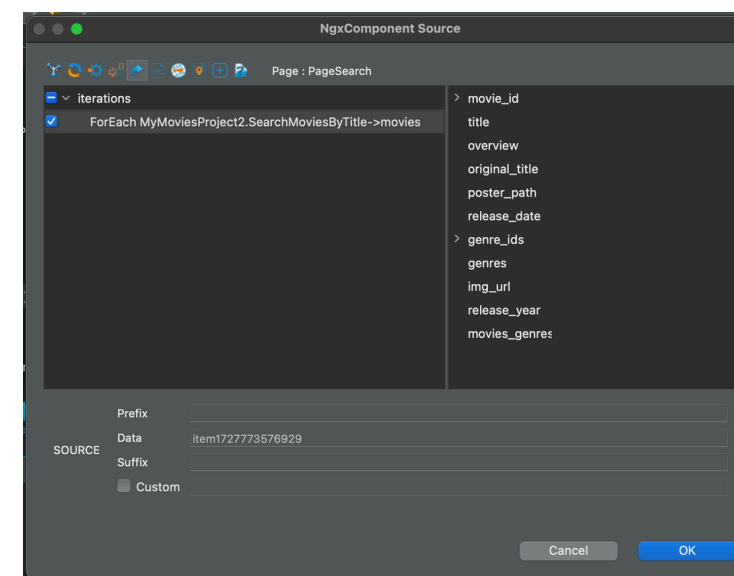
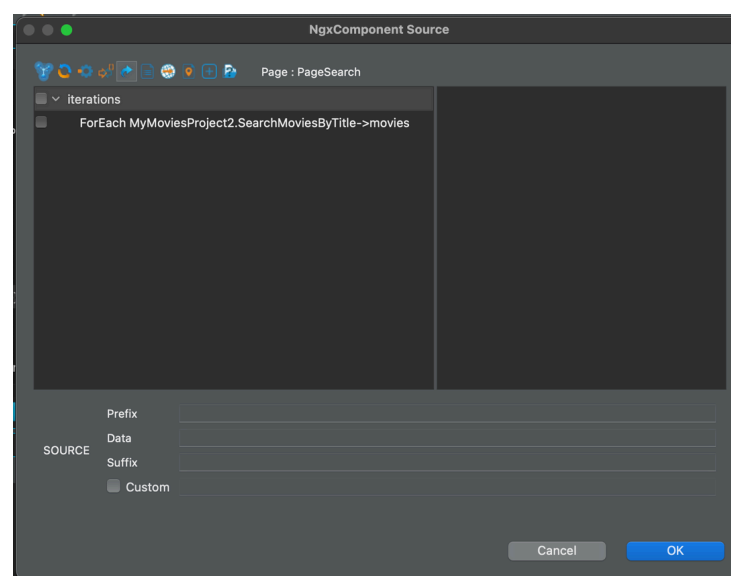
5.3 Binding the ForEach Directive to a sequence

There is another, and easiest way, to bind our components to the **Directive source**.

It is very useful when you don't know the **object keys names**.

Click on the directive to select it and display the object keys.

Click on the **object key title** to select it.



5.3 Binding the ForEach Directive to a sequence

The title is now binded to the **Directive source**.

Base properties	
Comment	
I18n translate	false
Is active	true
Text value	item1727773576929?.title



Base properties	
Comment	
I18n translate	false
Is active	true
Text value	->title



And appears as such in the treeview.

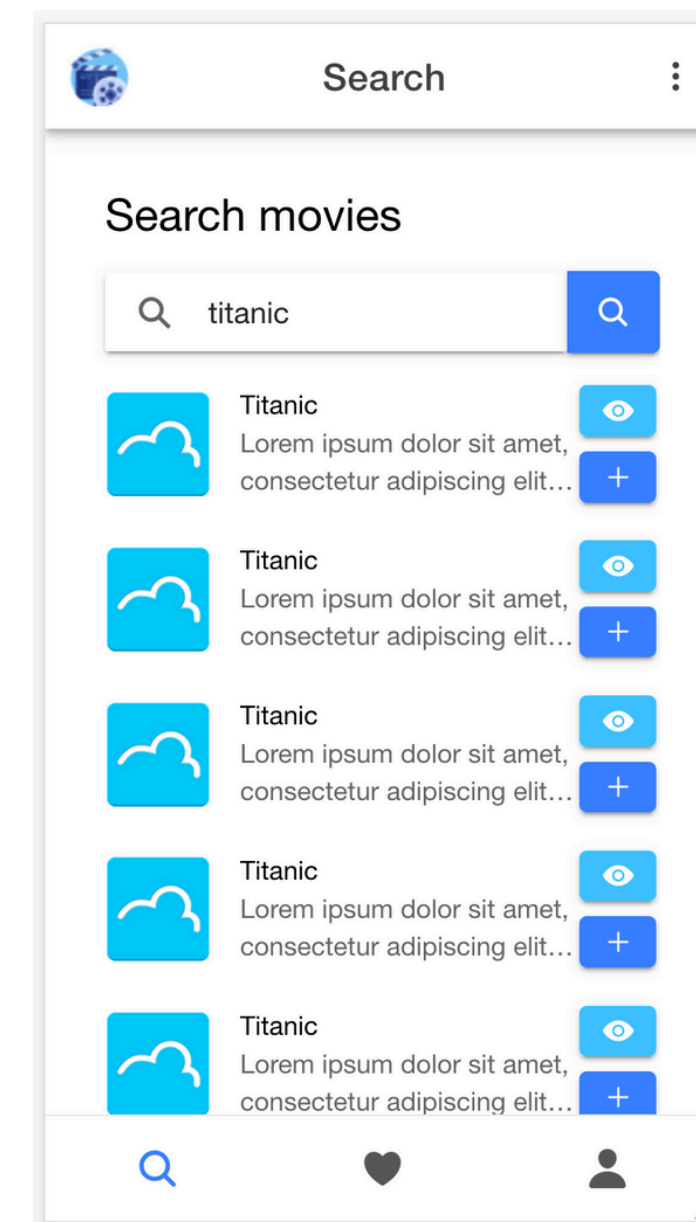
ForEach MyMoviesProject2.SearchMoviesByTitle->movies;
ListItem
Thumbnail
Image
Label
Attributes
Heading
T {{->title}}
Paragraph
Attributes
T Lorem ipsum dolor sit amet, consectetur adipiscing elit.
DivButtons



Heading
T {{->title}}



And the title appears in the search results.



Titanic
Lorem ipsum
consectetur a

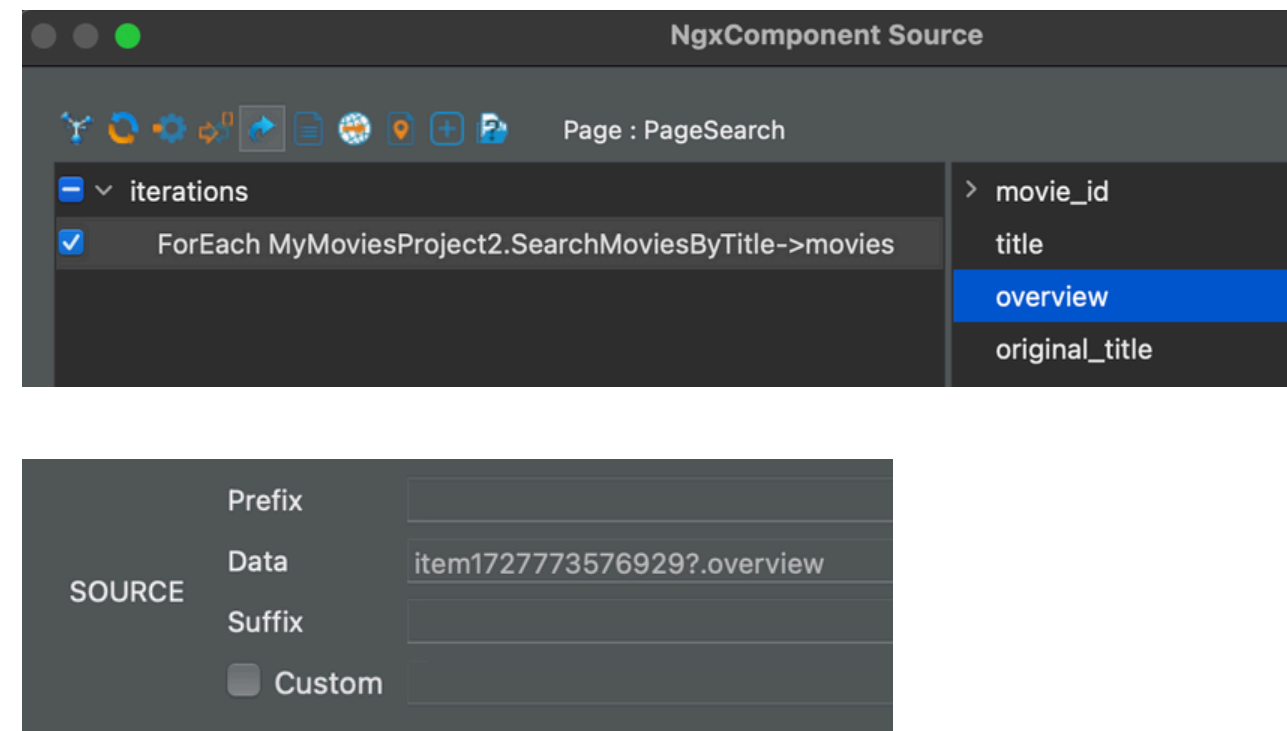
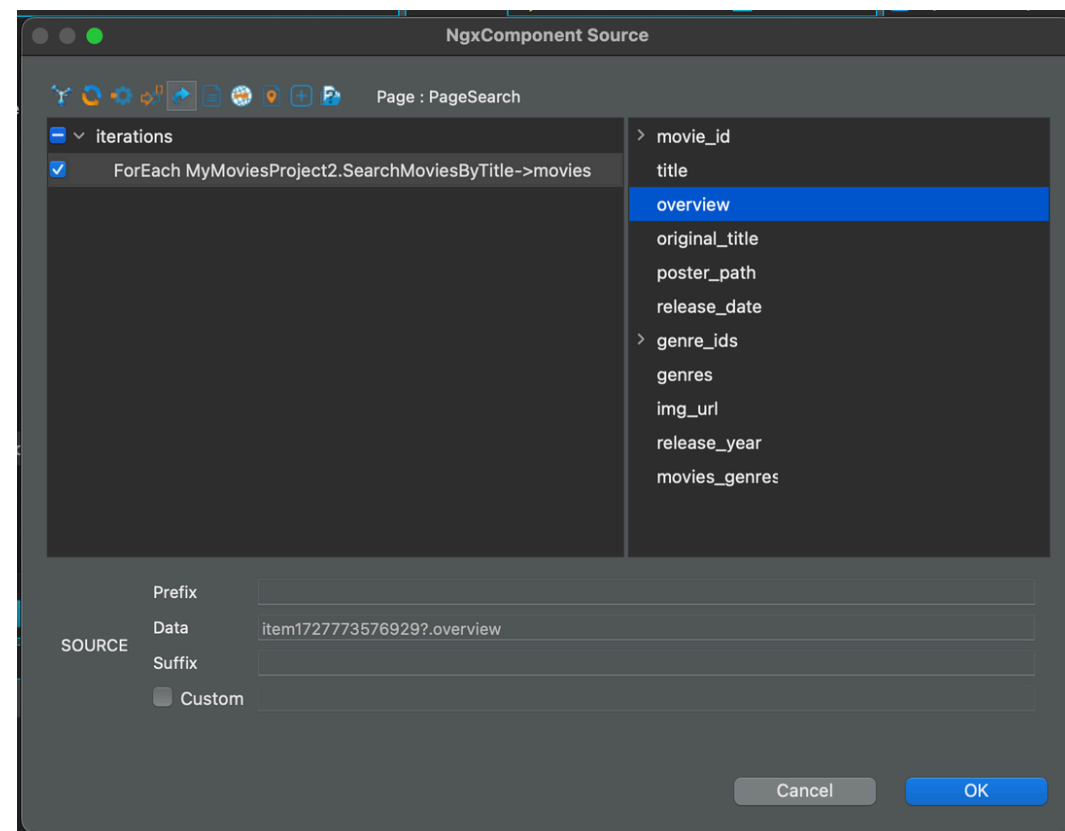
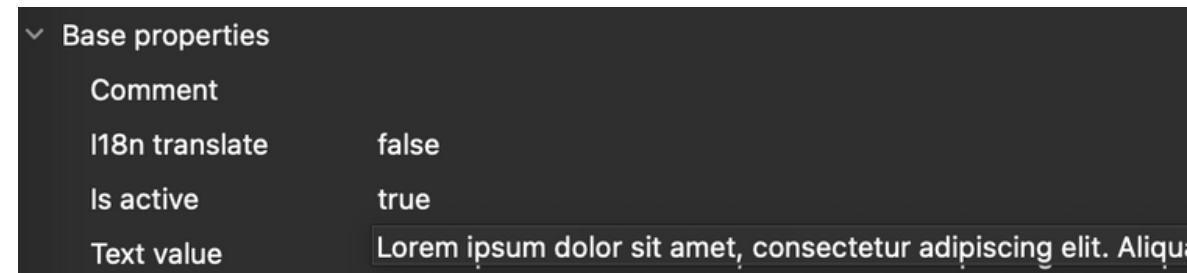
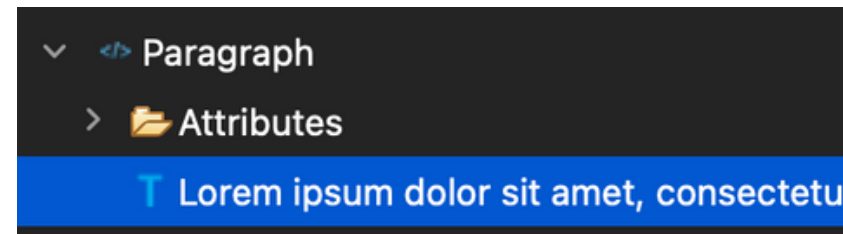


Titanic
Lorem ipsum
consectetur a



5.3 Binding the ForEach Directive to a sequence

Let's bind the paragraph and overview in the same way.





5.3 Binding the ForEach Directive to a sequence

Let's bind the image in the same way.

ForEach MyMoviesProject2.SearchMoviesByTitle->movies;

ListItem

Thumbnail

Image

Label

Attributes

Heading

Paragraph

Attributes

DivButtons

Alt

Height

Src

Width

not set

not set

->img_url

not set

Property

Value

@Component

Identifier

@Properties

Alt

Height

Src

Width

not set

not set

data:image/png;base64,iVBORw0KGgoA

not set



NgxComponent Source

Page : PageSearch

iterations

ForEach MyMoviesProject2.SearchMoviesByTitle->movies

movie_id

title

overview

original_title

poster_path

release_date

genre_ids

genres

img_url

release_year

movies_genres

Prefix

Data

Suffix

Custom

item1727773576929?.img_url

Cancel

OK

+

Prefix

Data

Suffix

genres

img_url

release_year

@Properties

Alt

Height

Src

Width

not set

not set

item1727773576929?.img_url

not set



@Properties

Alt

Height

Src

Width

not set

not set

->img_url

not set

Search

Search movies

titanic

Titanic

101-year-old Rose DeWitt

Bukater tells the story of h...

Titanic

Unhappily married, Julia

Sturges decides to go to...

Titanic

This little-known German

film retells the true story o...

Titanic

This documentary explores

the incredible history of th...

Titanic

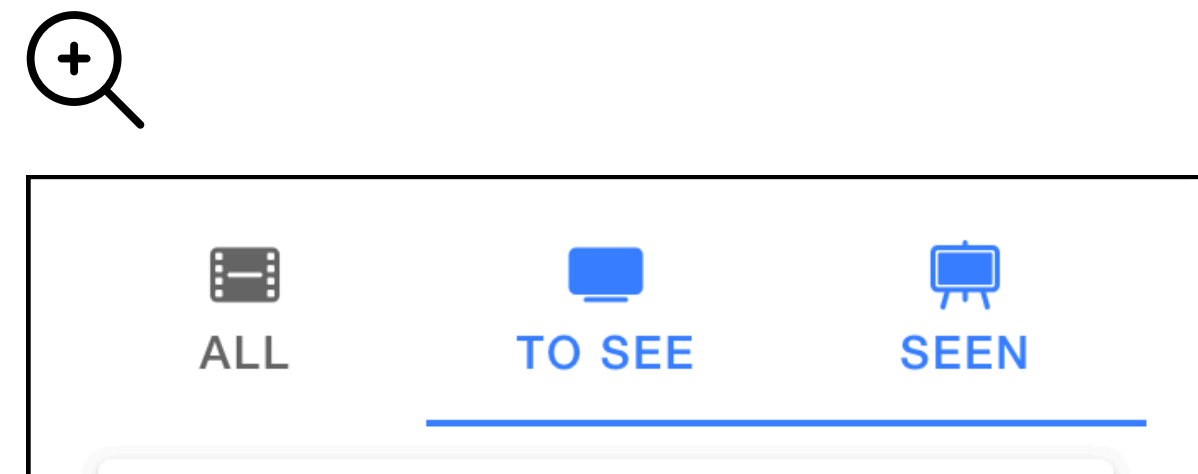
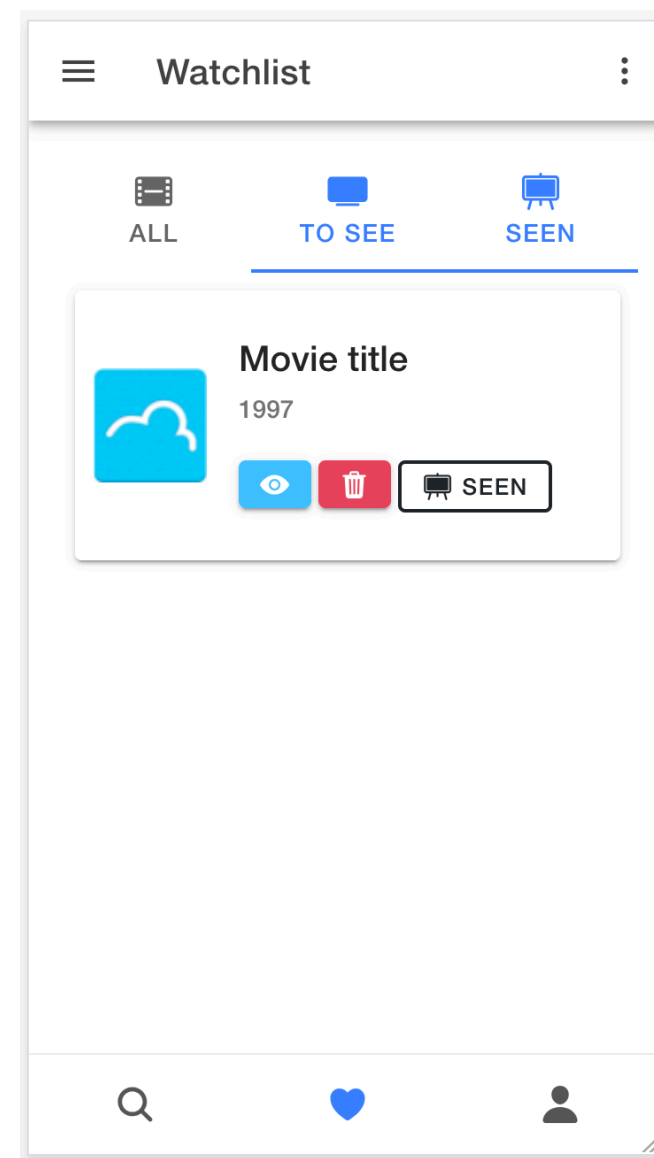
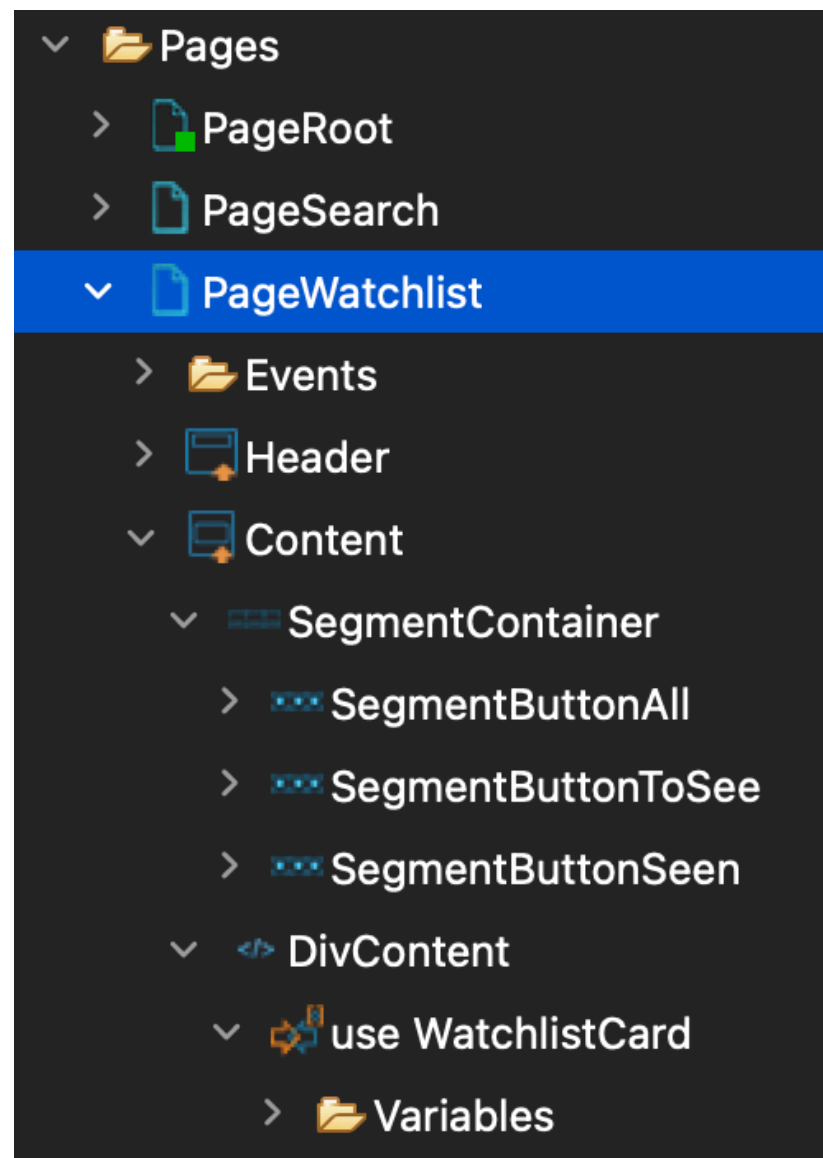


5.4 Binding segments with the Switch Directive

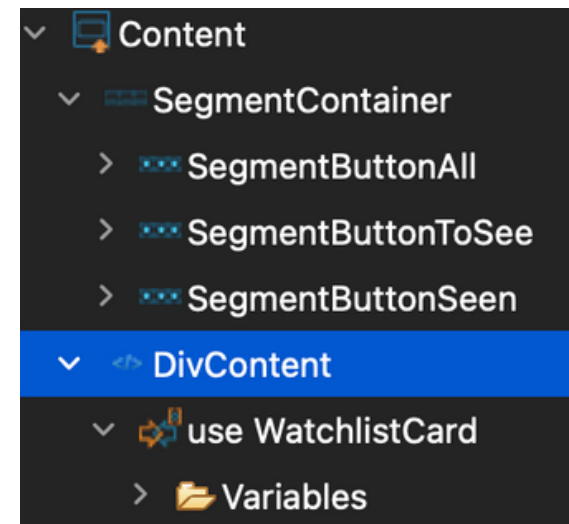
In **PageWatchlist**, we added a **SegmentContainer** and 3 **SegmentButtons** to display different contents in the same page.

As for now, we display the same content.

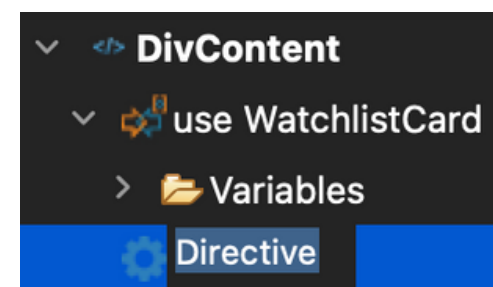
To switch between contents, we are going to use the **Switch** and **SwitchCases** directives.



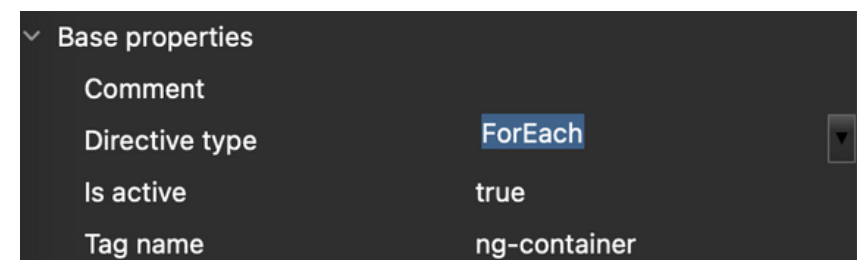
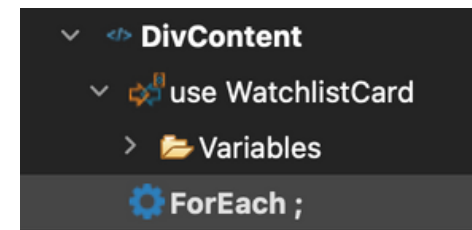
5.4 Binding segments with the Switch Directive



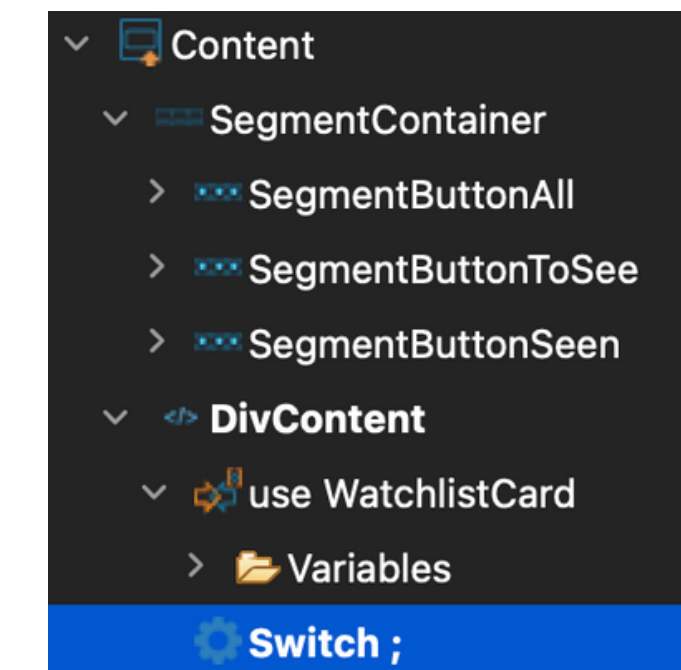
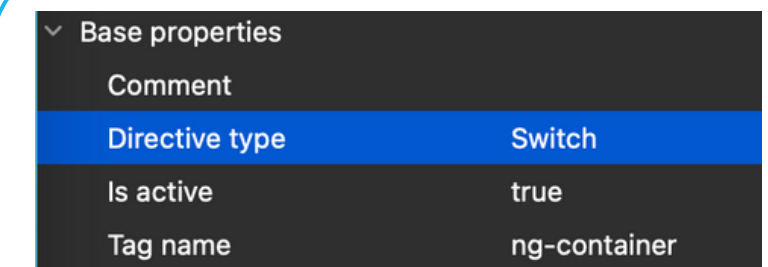
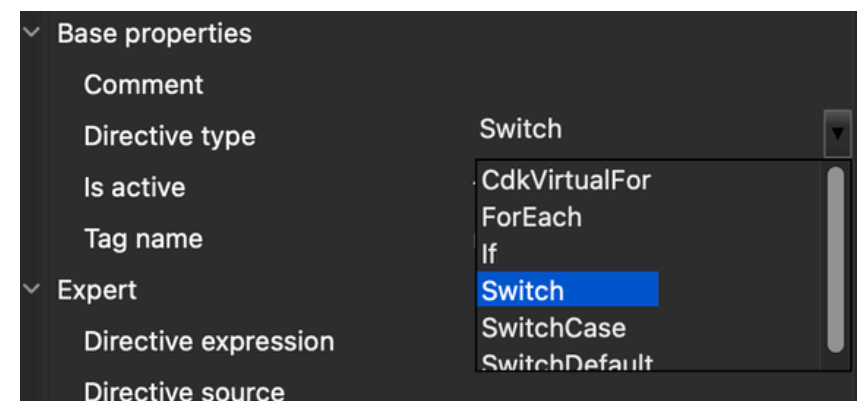
In **DivContent**,
let's add a **Directive** component.



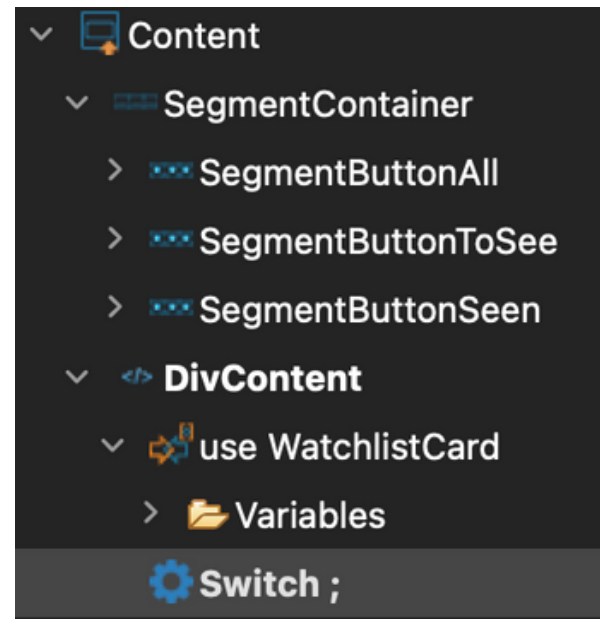
By default, a **Directive** component
is a **ForEach**.



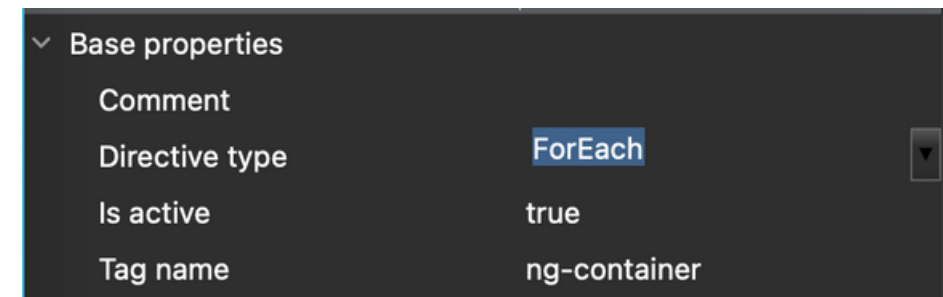
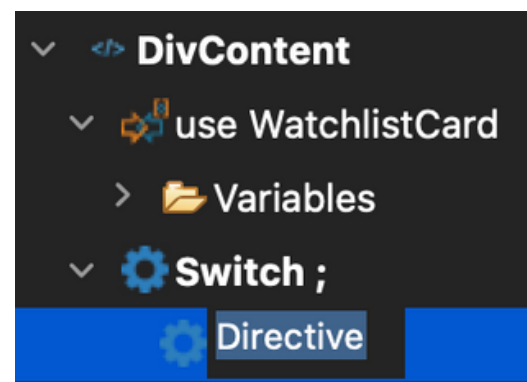
Let's change the directive
from **ForEach** to **Switch**.



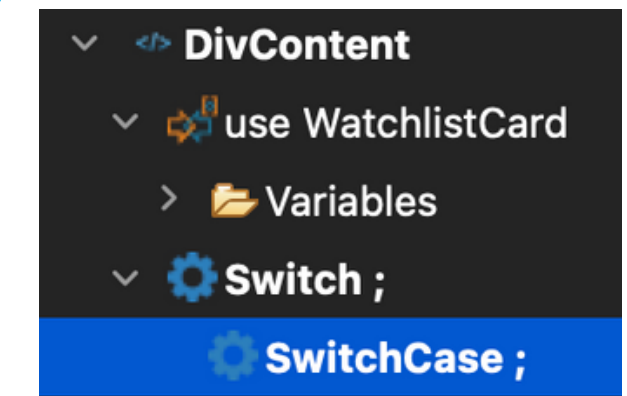
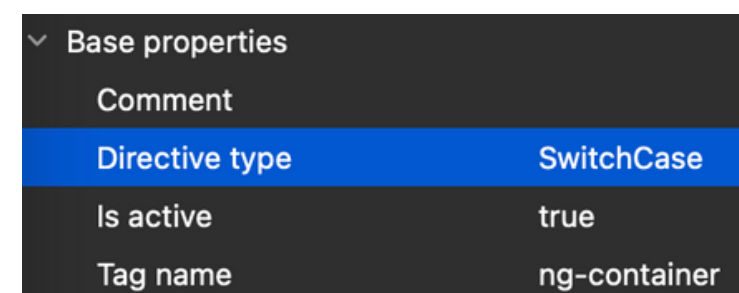
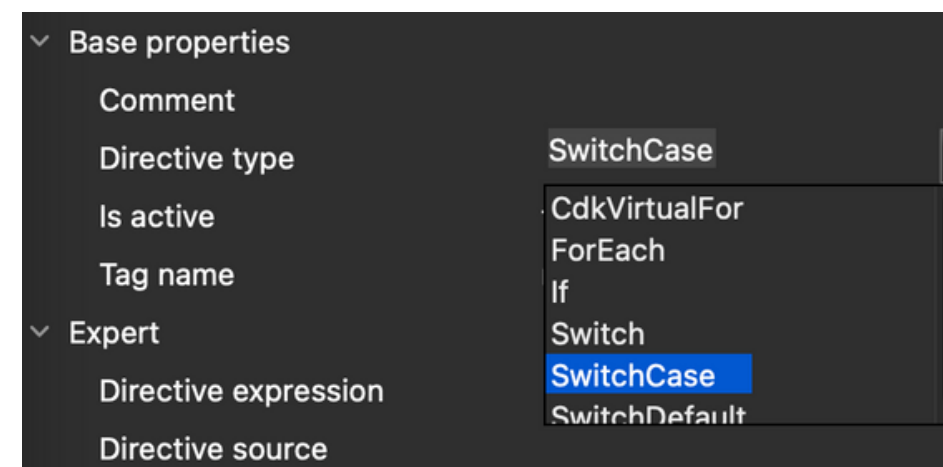
5.4 Binding segments with the Switch Directive



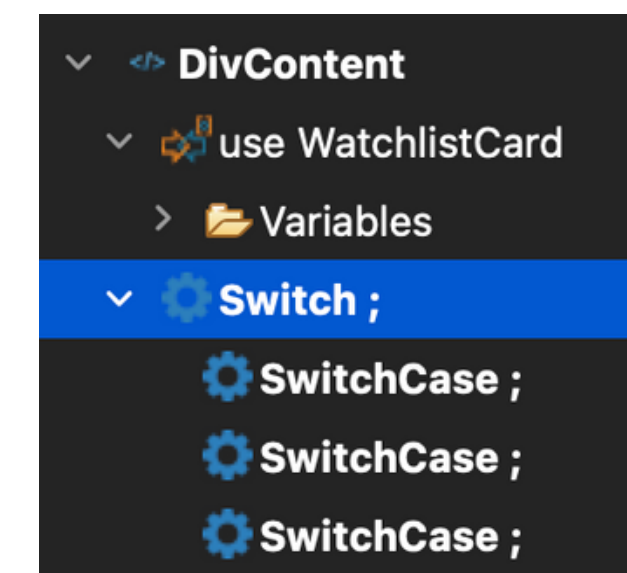
In the **Switch Directive**,
add another **Directive component**.



Let's change the directive
from **ForEach** to **SwitchCase**.



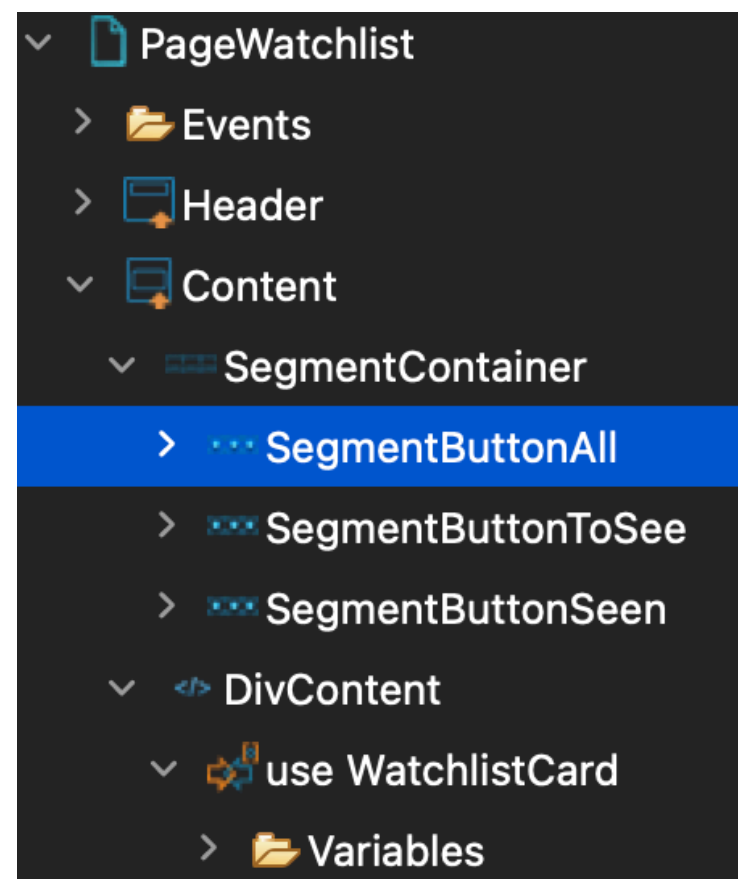
Repeat this 2 times
to have 3 **SwitchCases**.



5.4 Binding segments with the Switch Directive

Let's have a look on the **SegmentContainer** and the **SegmentButtons**.

First, in **SegmentContainer**, let's have a look on **SegmentButtonAll**'s properties.



In the **Identifier** property, add **segment_all_movies** as value.

Property	Value
@Component Identifier	
@Forms Button form type	button
@Properties Disabled	false
Layout	icon-top
Mode	not set
Value	one

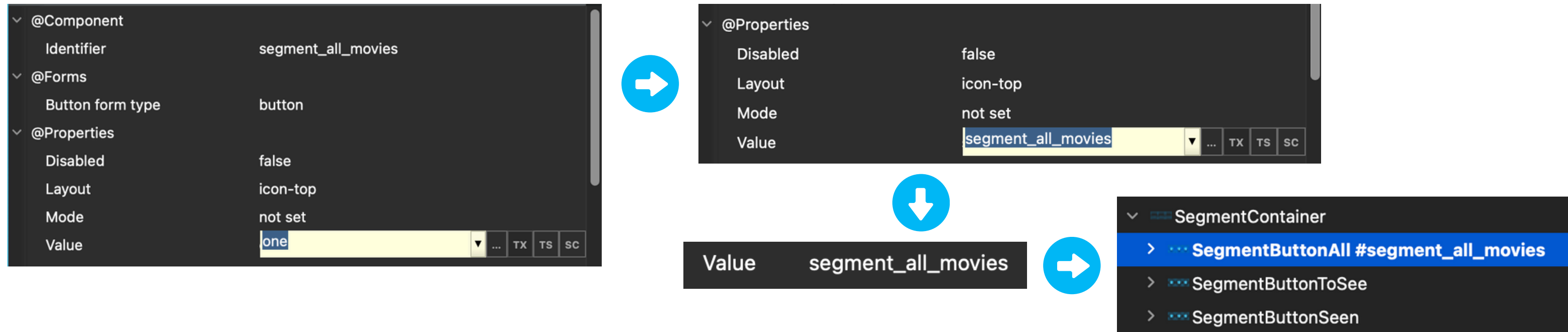


@Component Identifier	segment_all_movies
@Forms Button form type	button
@Properties Disabled	false
Layout	icon-top
Mode	not set
Value	one

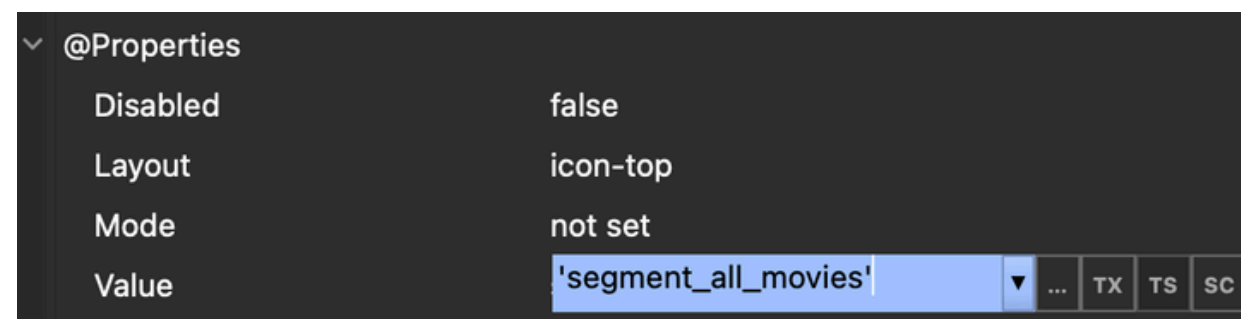
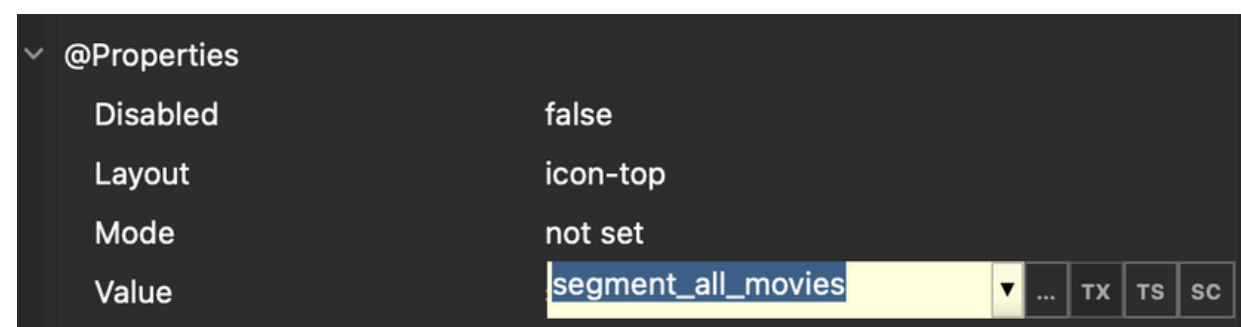


5.4 Binding segments with the Switch Directive

In the **Value** property, replace one by `segment_all_movies` as value.



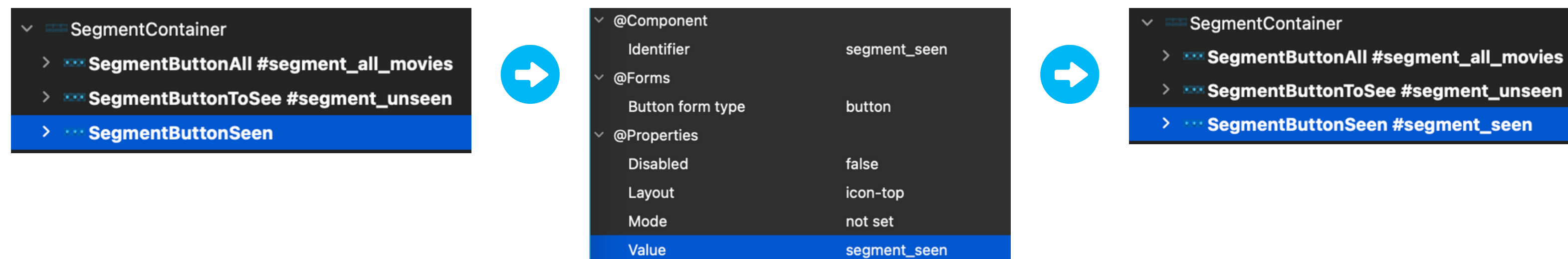
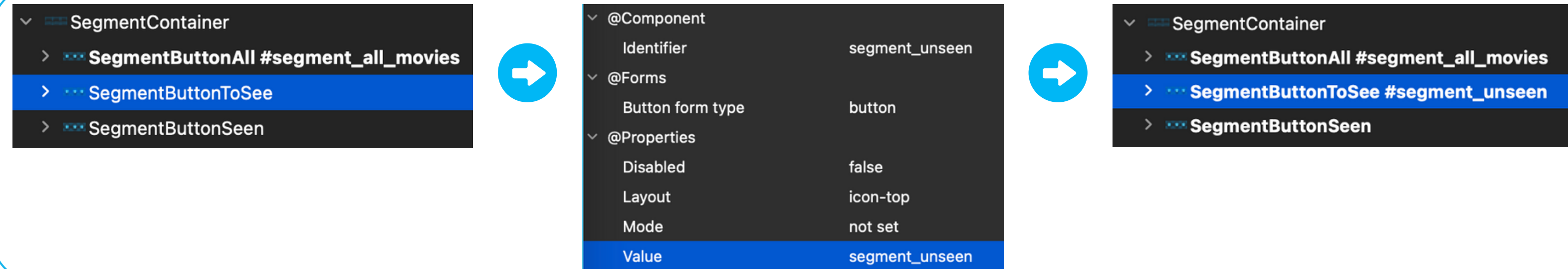
The **Value** property is in text or TX by default
 If you enter the value in TS,
 add quotes to the value as '`segment_all_movies`'.



5.4 Binding segments with the Switch Directive

In the same way, in the **Identifier** and **Value** properties,

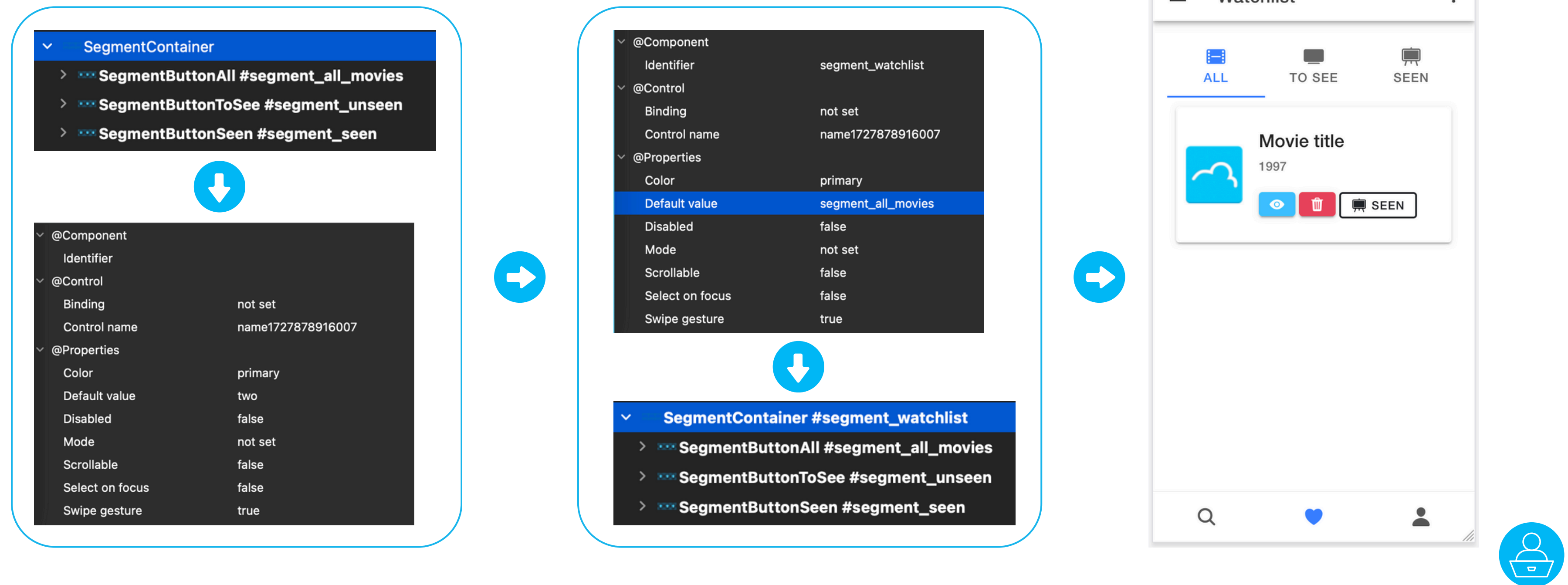
- enter **segment_to_see** as **value** for **SegmentButtonToSee**
- enter **segment_seen** as **value** for **SegmentButtonSeen**



5.4 Binding segments with the Switch Directive

In **SegmentContainer**'s properties

- in the **Identifier** property, enter **segment_watchlist** as value
- in the **Default value** property, enter **segment_all_movies** as value



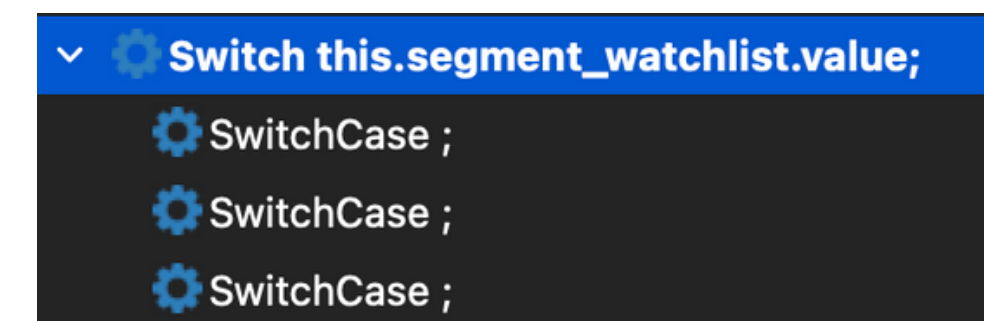
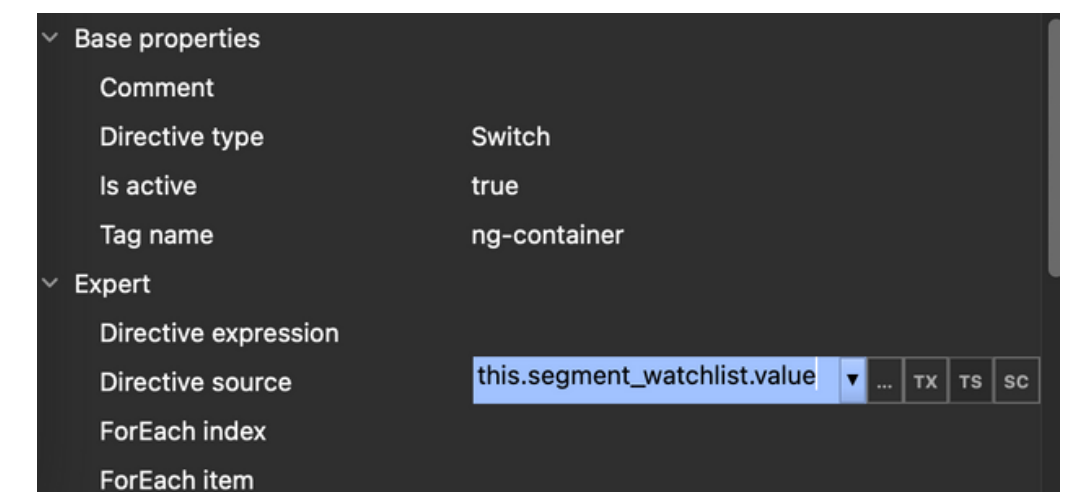
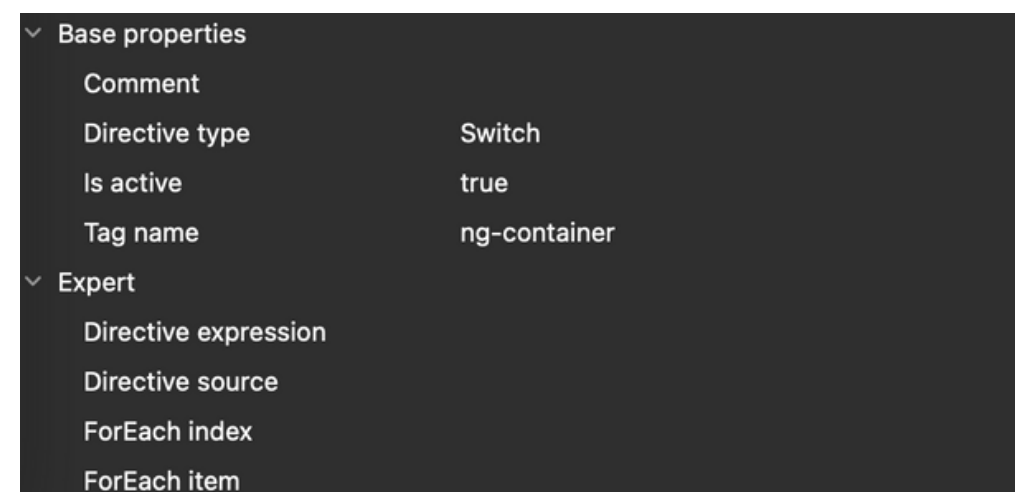
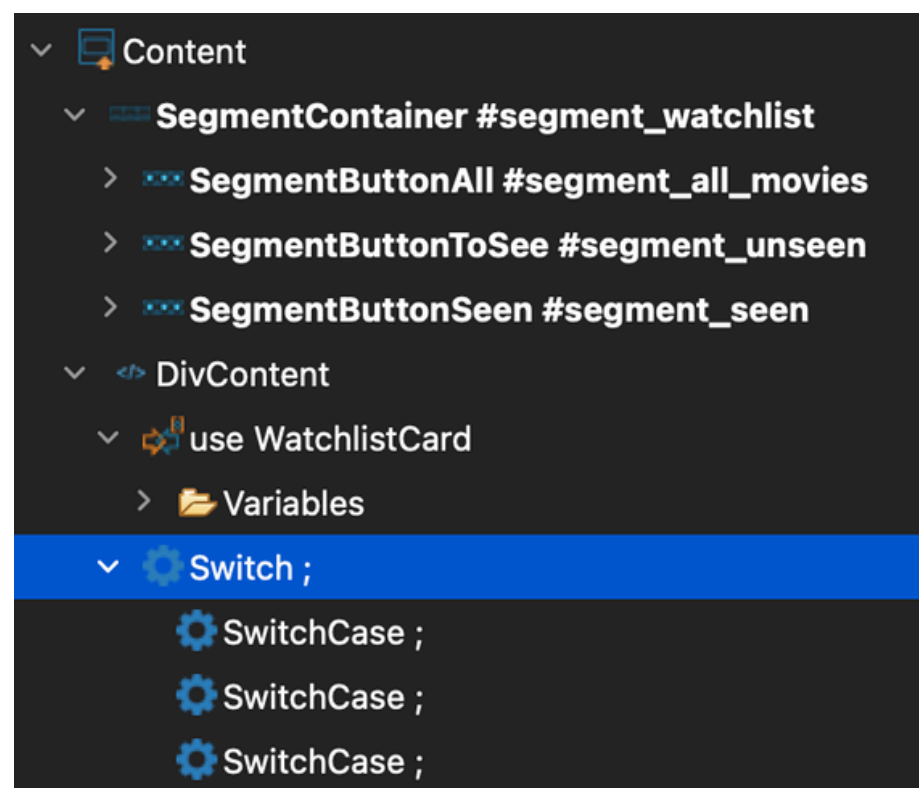
5.4 Binding segments with the Switch Directive

Now, we need to bind the segments to the **Switch** and **SwitchCase** Directives.

First, let's bind the **Switch** directive to the **SegmentContainer**.

The **SegmentContainer**'s Identifier is **#segment_watchlist**.

In the **Switch** Directive's properties, in the **Directive source** property, add **this.segment_watchlist.value** as value.



5.4 Binding segments with the Switch Directive

Now, let's bind the **SwitchCase Directives** to the **SegmentButtons**.

```

Switch this.segment_watchlist.value;
  SwitchCase ;
  SwitchCase ;
  SwitchCase ;
  
```

First, let's add a **Label** with a different text in each **SwitchCase directive**.

```

Switch this.segment_watchlist.value;
  SwitchCase ;
  Label
    ALL MOVIES
  SwitchCase ;
  Label
    TO SEE
  SwitchCase ;
  Label
    SEEN
  
```

Now, let's bind the first **SwitchCase Directive** to **SegmentButtonAll** whose identifier is **#segment_all_movies**.

```

Switch this.segment_watchlist.value;
  > SwitchCase ;
  > SwitchCase ;
  > SwitchCase ;
  
```

```

Content
  SegmentContainer #segment_watchlist
    > SegmentButtonAll #segment_all_movies
    > SegmentButtonToSee #segment_unseen
    > SegmentButtonSeen #segment_seen
  
```

In the **SwitchCase Directive's** properties, in the **Directive source property**, add **'segment_all_movies'** as value in TS.

Base properties	
Comment	
Directive type	SwitchCase
Is active	true
Tag name	ng-container
Expert	
Directive expression	
Directive source	'segment_all_movies' TX TS SC
ForEach index	
ForEach item	

```

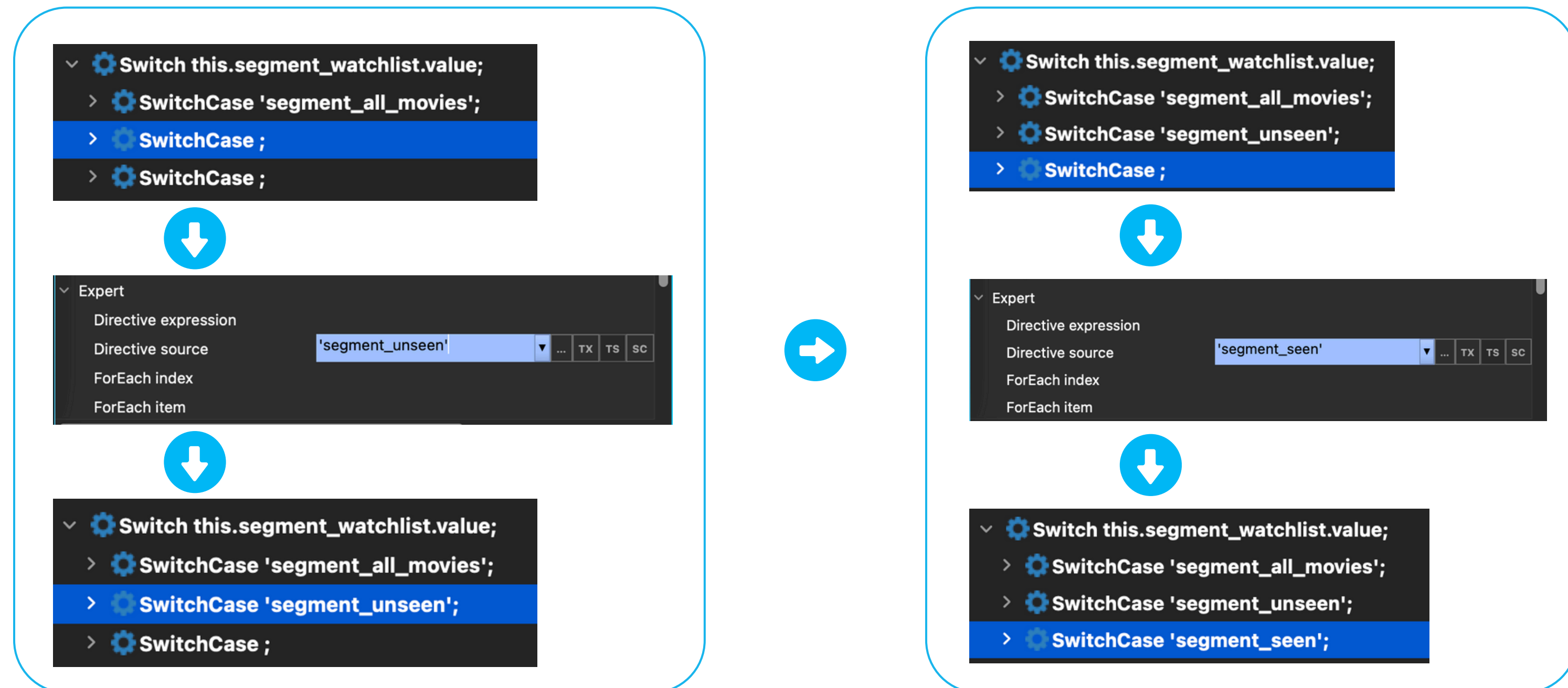
Switch this.segment_watchlist.value;
  > SwitchCase 'segment_all_movies';
  > SwitchCase ;
  > SwitchCase ;
  
```



5.4 Binding segments with the Switch Directive

Bind the 2 other **SwitchCase Directives** to the **SegmentButtons**.

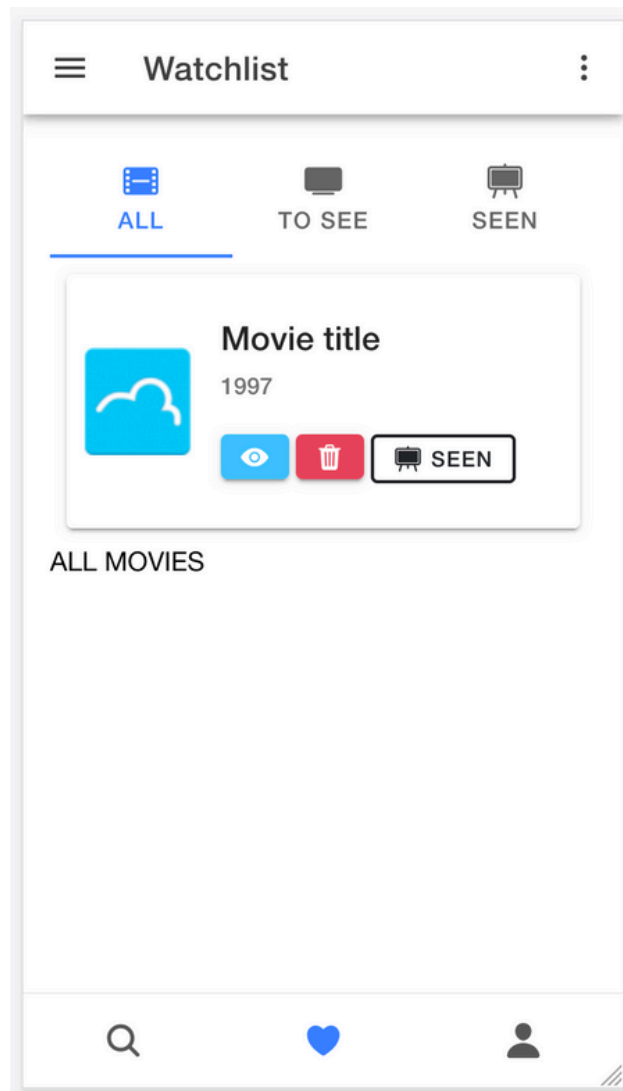
- For the second **SwitchCase Directive**, in the **Directive source property**, add 'segment_unseen' as **value** in **TS**.
- For the third **SwitchCase Directive**, in the **Directive source property**, add 'segment_seen' as **value** in **TS**.



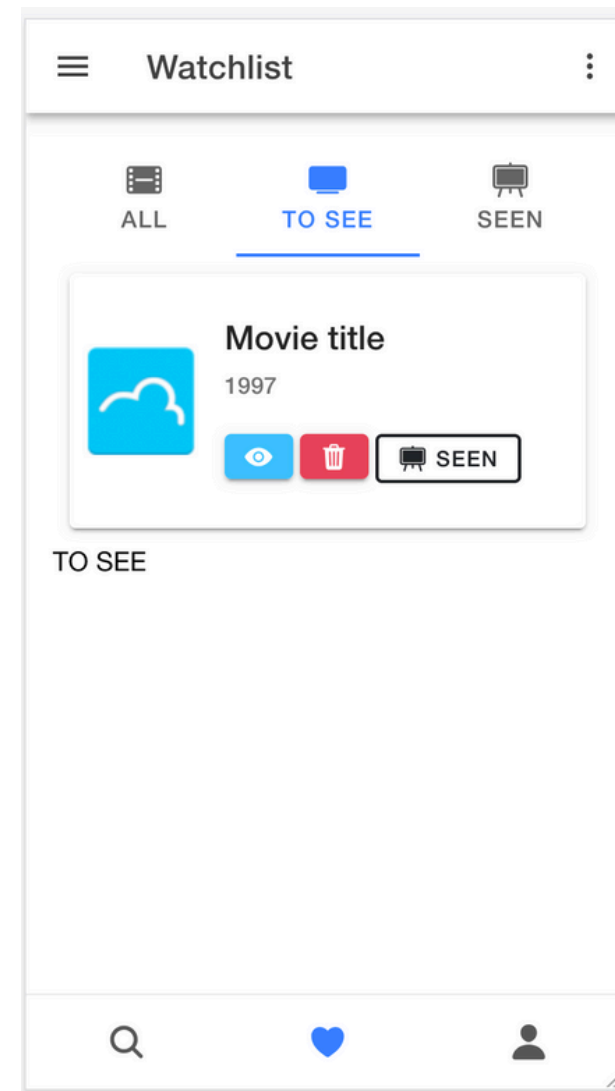
5.4 Binding segments with the Switch Directive

The **SegmentButtons** are now binded to the **Switch** and **SwitchCase Directives** and a different **Label** is displayed when clicking a different button.

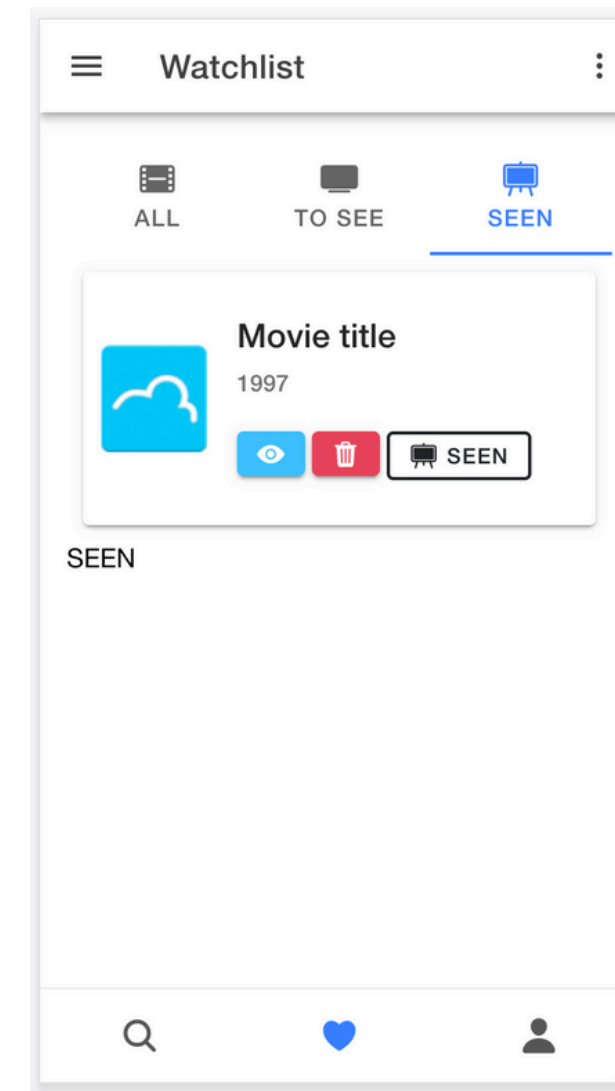
SegmentButtonAll



SegmentButtonToSee



SegmentButtonSeen



5.4 Binding segments with the Switch Directive

These **segments** will be used to display a **watchlist** recorded in our database.

As for now, we'll just use **UseSharedComponents UseWatchlistCard** with static data.

Copy **UseWatchlistCard** in each **SwitchCase**,
and remove it from **DivContent**

Then remove the Labels from each SwitchCase.

```

  <div>
    <PageWatchlist>
      <Events>
      <Header>
      <Content>
        <SegmentContainer #segment_watchlist>
          <DivContent>
            <use WatchlistCard>
            <Switch this.segment_watchlist.value>
              <SwitchCase 'segment_all_movies'>
                <Label>
                <use WatchlistCard>
              <SwitchCase 'segment_unseen'>
                <Label>
                <use WatchlistCard>
              <SwitchCase 'segment_seen'>
                <Label>
  
```



```

  <div>
    <DivContent>
      <Switch this.segment_watchlist.value>
        <SwitchCase 'segment_all_movies'>
          <Label>
          <use WatchlistCard>
        <SwitchCase 'segment_unseen'>
          <Label>
          <use WatchlistCard>
        <SwitchCase 'segment_seen'>
          <Label>
          <use WatchlistCard>
  
```



```

  <div>
    <DivContent>
      <Switch this.segment_watchlist.value>
        <SwitchCase 'segment_all_movies'>
          <Label>
          <use WatchlistCard>
        <SwitchCase 'segment_unseen'>
          <Label>
          <use WatchlistCard>
        <SwitchCase 'segment_seen'>
          <Label>
          <use WatchlistCard>
  
```

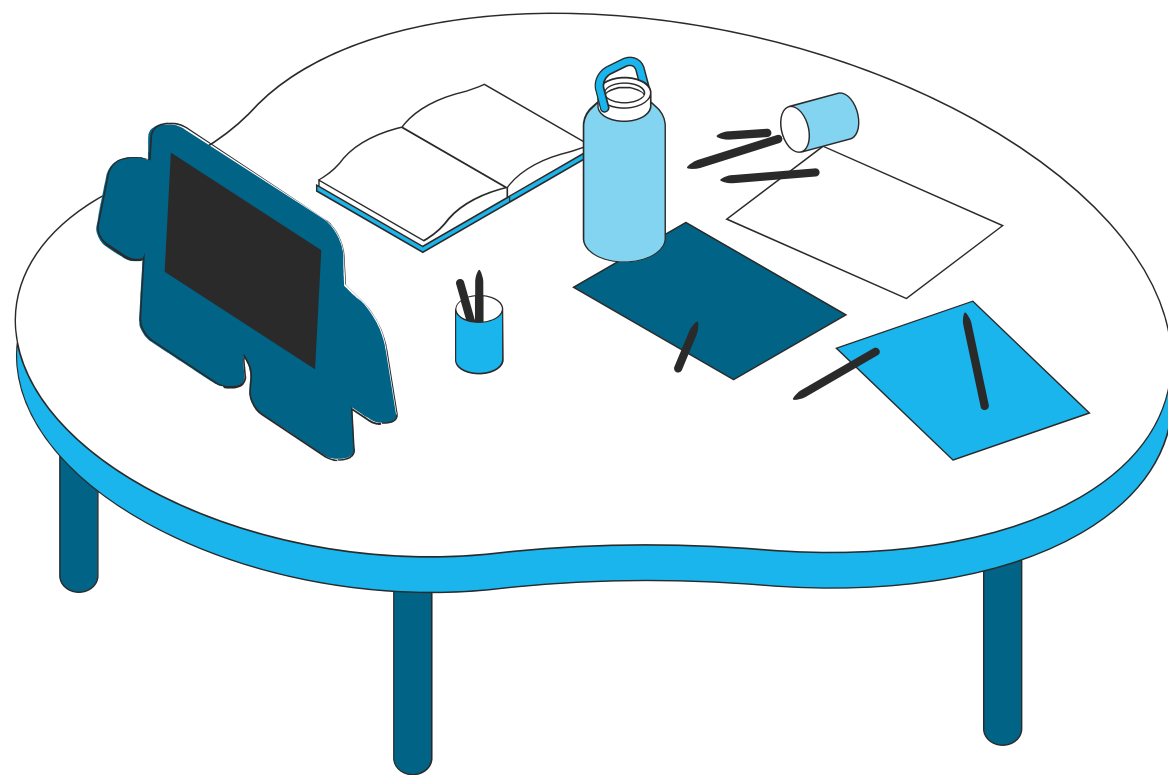


```

  <div>
    <DivContent>
      <Switch this.segment_watchlist.value>
        <SwitchCase 'segment_all_movies'>
          <use WatchlistCard>
        <SwitchCase 'segment_unseen'>
          <use WatchlistCard>
        <SwitchCase 'segment_seen'>
          <use WatchlistCard>
  
```



6 – Events, Actions & Variables



6.1 Event component

6.2 PageEvent component

6.3 CallSequence Component

6.4 Variable component

6.5 CustomAction Component

6.6 SetLocal & SetGlobal action components

6.7 Calling a sequence with PageEvent

6.8 Use a CustomAction to process data

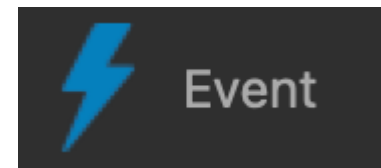
6.9 Set data as a local variable

6.10 Display data from a local variable

6.11 Display filtered data in segments

6.1 Event component

We have seen that components can be associated with an event with the **Event component**.

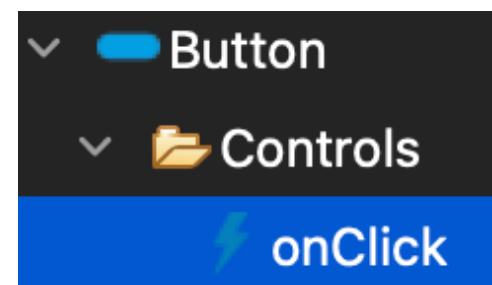


The **Event component**

is used for Angular's event binding.

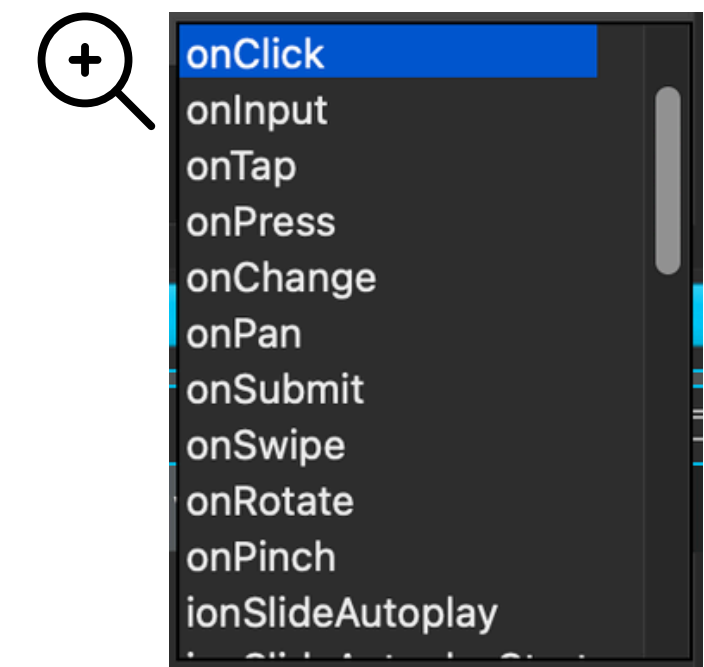
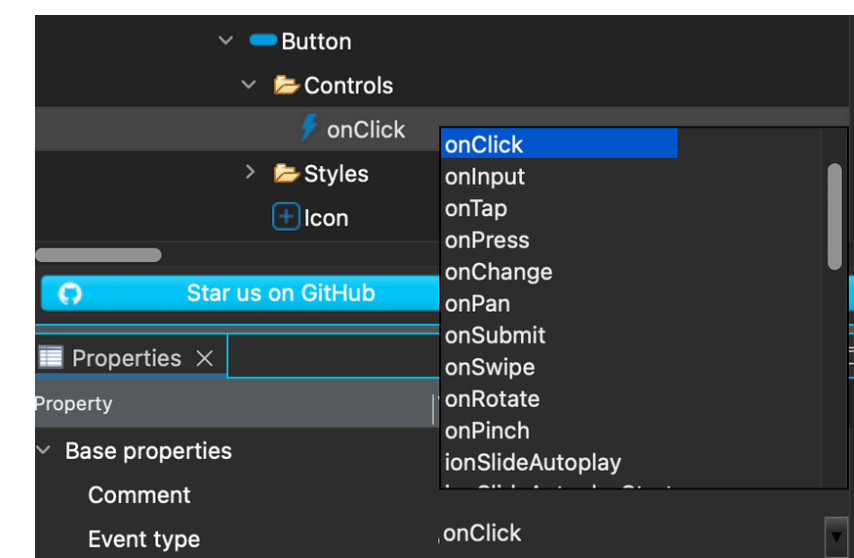
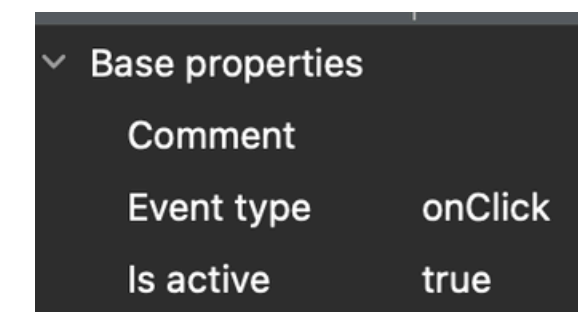
It is added to components which can emit events such as the Button component.

When added to a component, the event appears in a **Controls folder**, and is an **onClick event** by default.



The event's type is chosen with the **Event Type property**

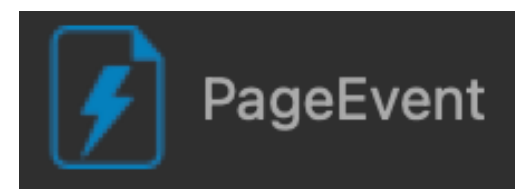
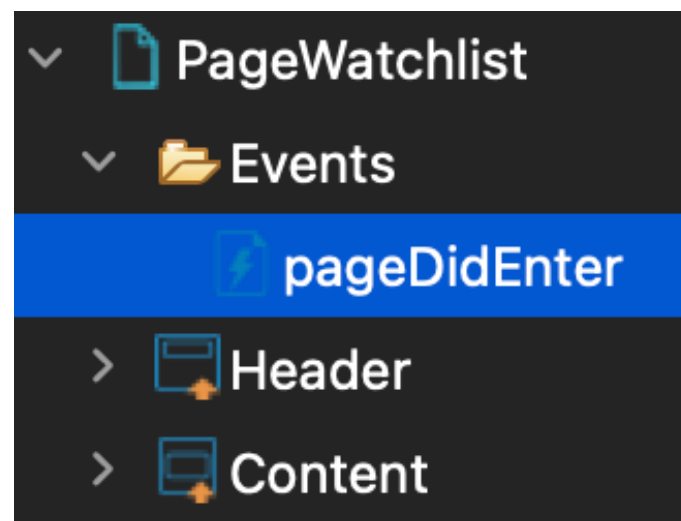
- by selecting it in a list of events
- or directly entered as value using the syntax (click)



6.2 PageEvent component

There are different event components in the Low Code studio.

Pages are created with a **PageEvent** by default.



The **PageEvent** component is based on **Angular** and **Ionic** life cycle events.

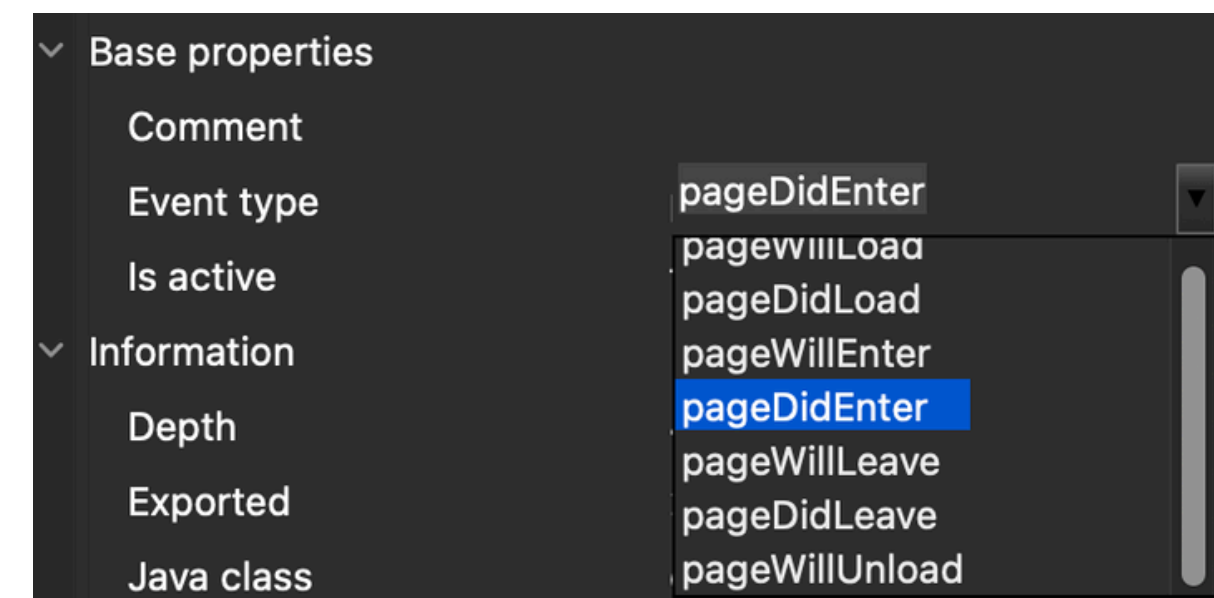
Page lifecycle events are fired during various stages of **navigation**:

pageWillLoad, pageDidLoad, pageWillEnter, pageDidEnter and so on.

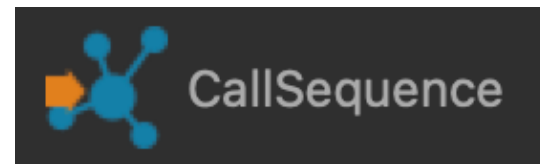
Its type by default is **pageDidEnter**.

You can select its type with the **Event type** property.

Property	Value
Base properties	
Comment	
Event type	pageDidEnter
Is active	true



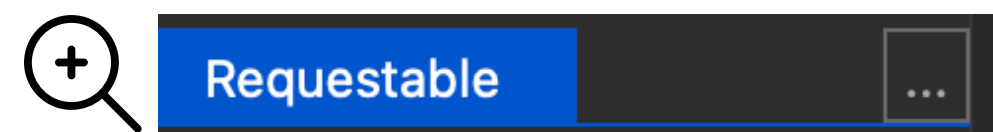
6.3 CallSequence Component



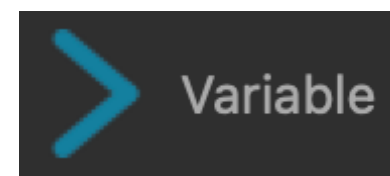
The **CallSequence Component** is used to **invoke a sequence** and is added under a **PageEvent**, an **Event** or another **Action**.

The sequence is specified through the **Requestable property** in the **CallSequence's properties**.

Property	Value
▼ @Properties	
Disable auto login	false
Disable loading controller	false
Local cache policy	not set
Local cache time to live	3000
Marker	
Requestable	

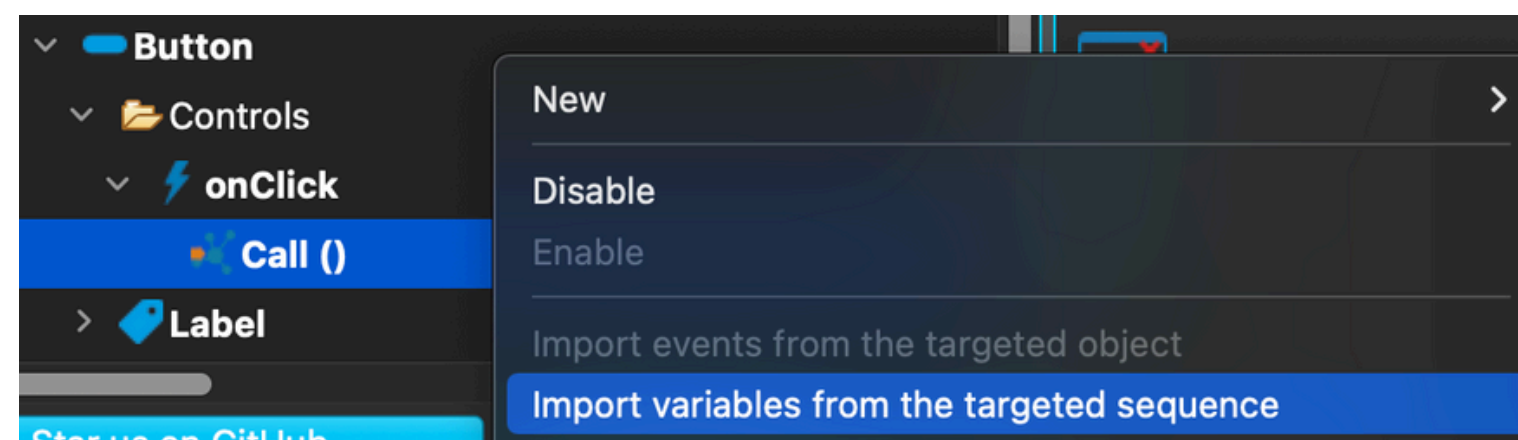


If the targeted sequence requires input variables,

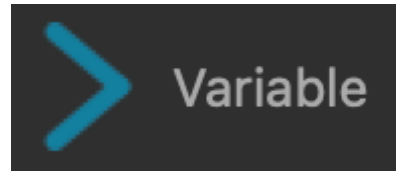


add them to the **CallSequence** with **Variable components**

or by selecting **Import variables from the targeted sequence** in the right-click menu.

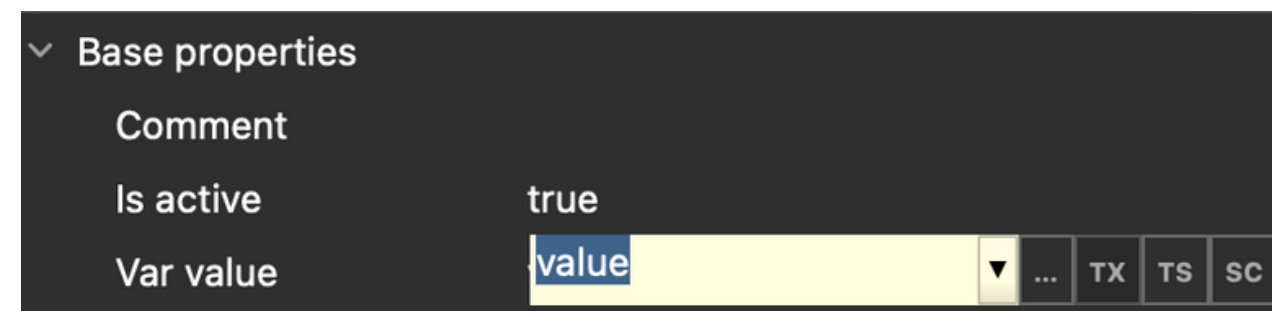
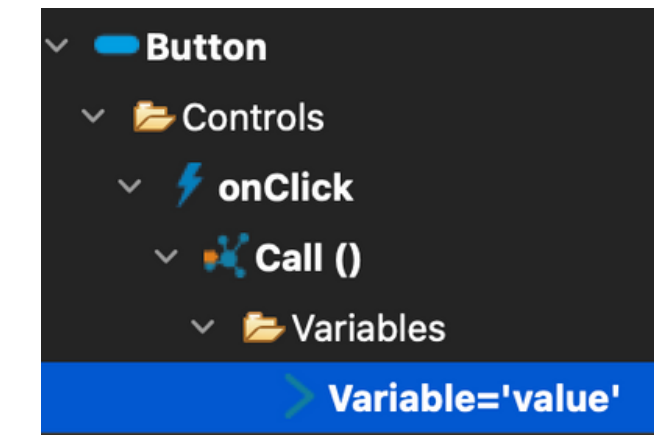


6.4 Variable component



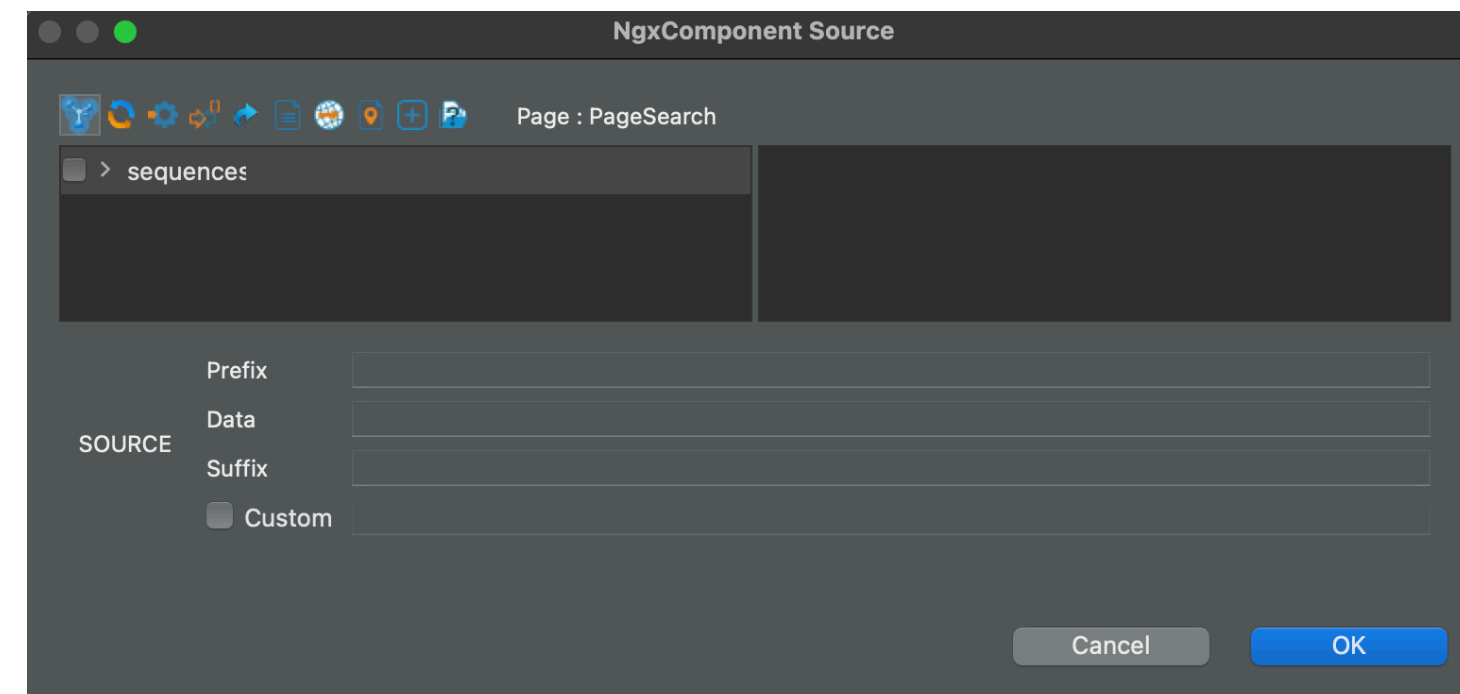
The **Variable Component** is used to pass additional parameters to the executing function of an action.

It can be added to any action, like **CallSequence**, or **CustomAction**

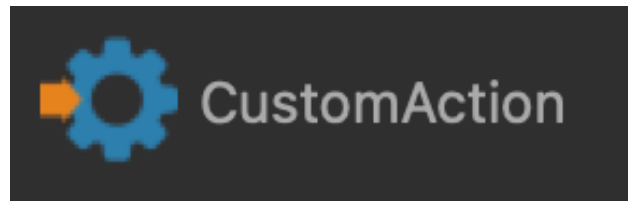


The Var value property can be set to

- a fixed value in **text mode (TX)**
- a typescript expression in **TS mode**
- any available source shown in the **NgxComponent Source panel**, in **SC mode**



6.5 CustomAction Component

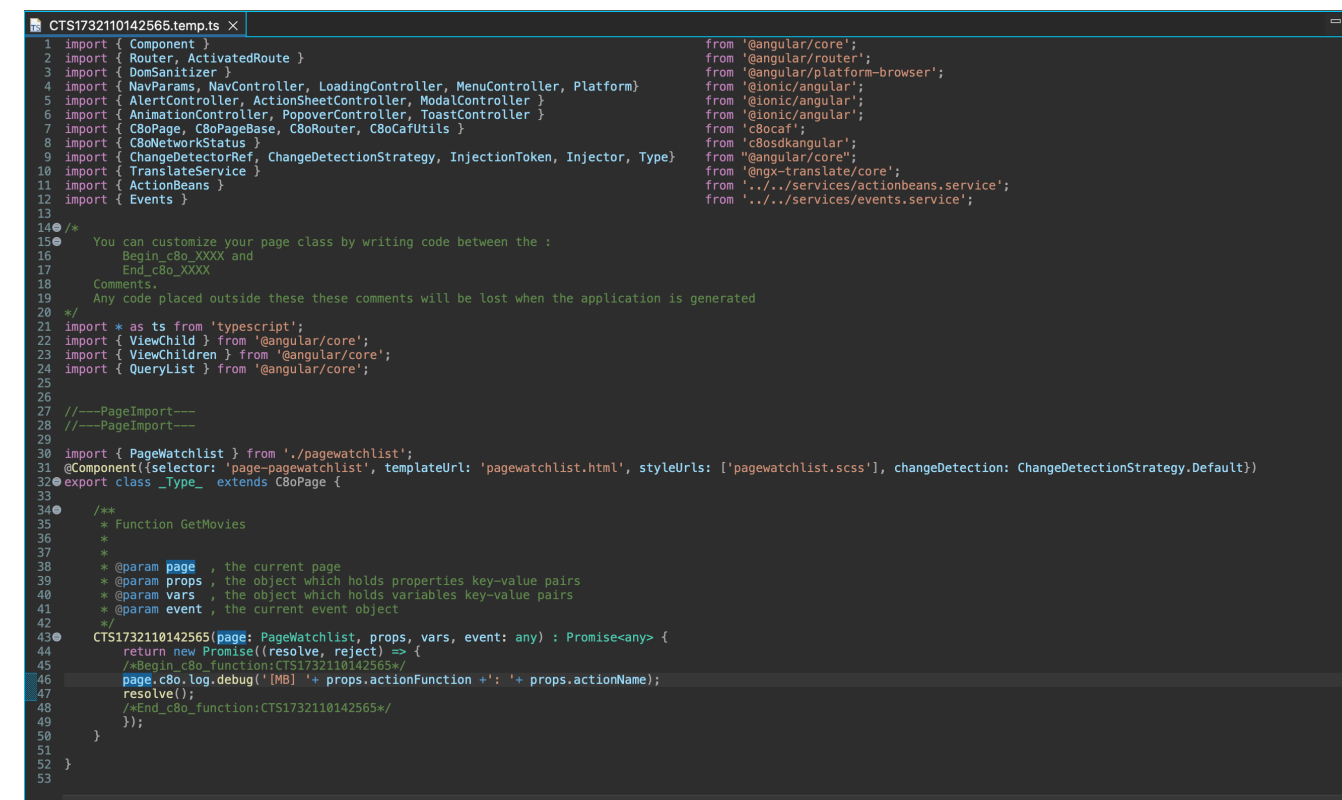
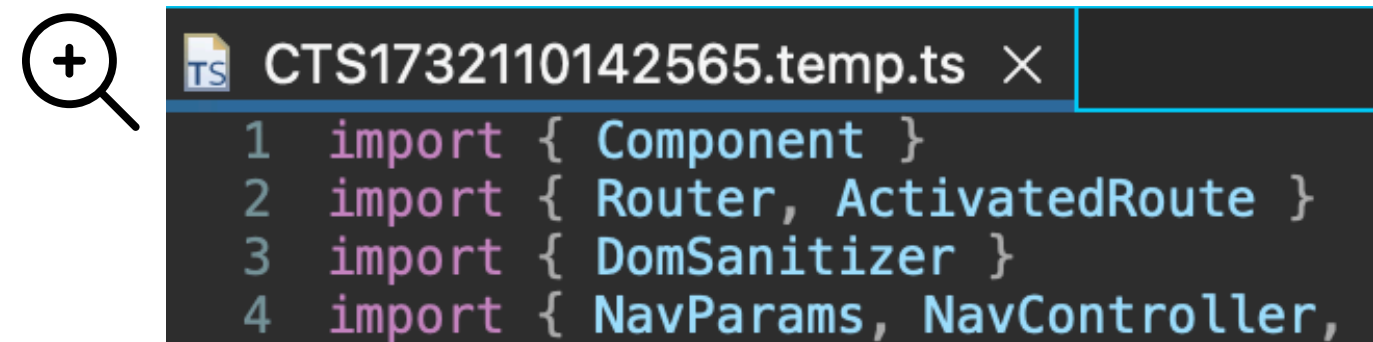


This component helps you writing your own processing code.

Custom actions can be used

- to process data, in an event processing chain, combined with other actions.
- to access **Variables** or pass them to actions.
- to add and call additional **NPM packages** in your projects.
- to add any **modules, imports, providers, plugins, scripts, styles** needed by this action.

To edit code,
double click on the **CustomAction** in the treeview
to open it in the text editor
as **CTSXXXXXXXXXXXXX.temps.ts** file.



6.5 CustomAction Component

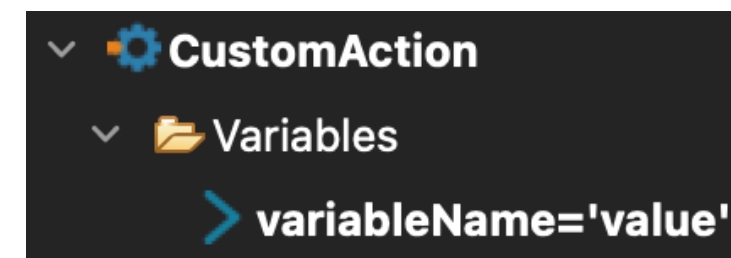
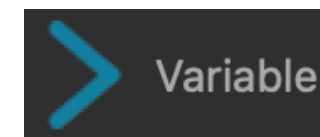
Write your custom code in TypeScript between the `/*Begin_c8o_function */` and `/*End_c8o_function */` delimiters.
Any code placed outside these comments will be lost when the application is generated.



```
26
27 //---PageImport---
28 //---PageImport---
29
30 import { PageWatchlist } from './pagewatchlist';
31 @Component({selector: 'page-pagewatchlist', templateUrl: 'pagewatchlist.html', styleUrls:
32 export class _Type_ extends C8oPage {
33
34 /**
35  * Function GetMovies
36  *
37  *
38  * @param page , the current page
39  * @param props , the object which holds properties key-value pairs
40  * @param vars , the object which holds variables key-value pairs
41  * @param event , the current event object
42  */
43 CTS1732110142565(page: PageWatchlist, props, vars, event: any) : Promise<any> {
44     return new Promise((resolve, reject) => {
45         /*Begin_c8o_function:CTS1732110142565*/
46         page.c8o.log.debug(' [MB] ' + props.actionFunction + ': ' + props.actionName);
47         resolve();
48         /*End_c8o_function:CTS1732110142565*/
49     });
50 }
51
```

These **@params** are used to access their values in the CTS file.

```
/**
 * Function GetMovies
 *
 *
 * @param page , the current page
 * @param props , the object which holds properties key-value pairs
 * @param vars , the object which holds variables key-value pairs
 * @param event , the current event object
 */
```

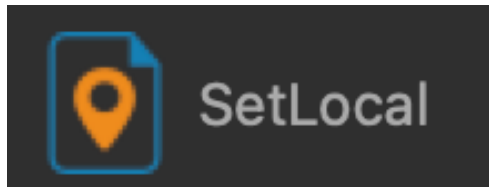


```
*/
CTS1732110142565(page: PageWatchlist, props, vars, event: any) : Promise<any> {
    return new Promise((resolve, reject) => {
        /*Begin_c8o_function:CTS1732110142565*/
        page.c8o.log.debug(' [MB] ' + props.actionFunction + ': ' + props.actionName);
        resolve();
        /*End_c8o_function:CTS1732110142565*/
    });
}
```

To access a value in a **CTS file**,
add a **Variable component**
in his **CustomAction** in the treeview
and use **vars.variableName**



6.6 SetLocal & SetGlobal action components

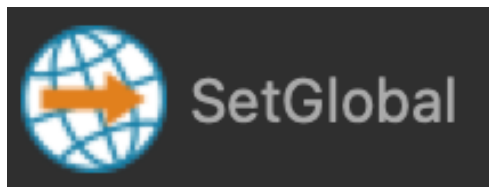


The **SetLocal** action component

is used to **store data in the current page or component.**

To retrieve and use the value of a **stored property** (for example **myProperty**) to be displayed on a page,

source a **Text component** in **TS mode** to **this.local?.myProperty**.



The **SetGlobal** action component

is used to **store data in the global application wide space.**

to access it from **any page in the application.**

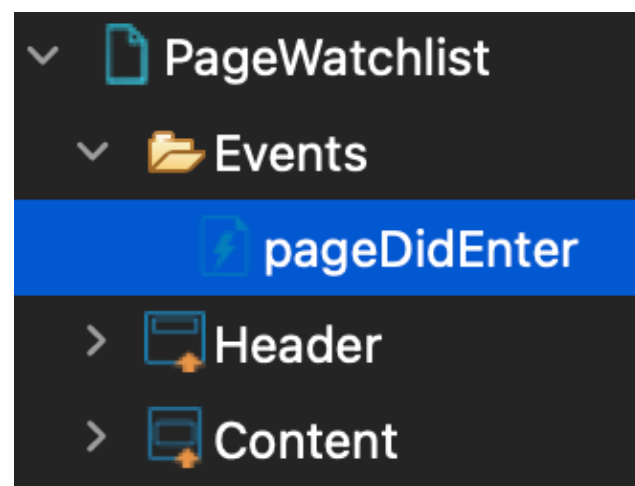
To retrieve and use the value of a **stored property** (for example **myProperty**) to be displayed on a page,

source a **Text component** in **TS mode** to **this.global?.myProperty**.



6.7 Calling a sequence with PageEvent

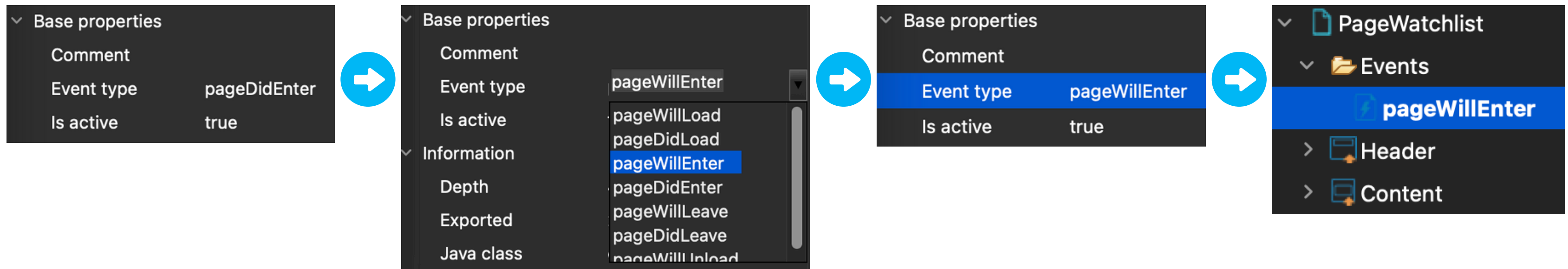
In **PageWatchlist**, we are going to use the **PageEvent** to get movies data by calling the sequence **MyMoviesDBMoviesTableList**, which retrieves all movies present in the database.



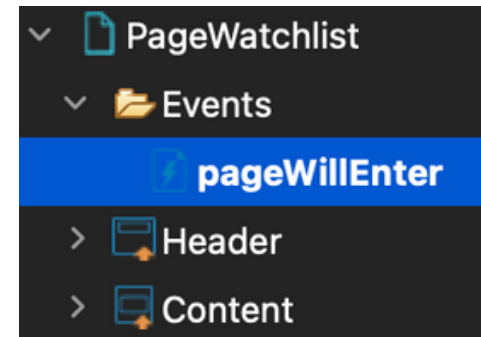
Let's change the **PageEvent type** to **pageWillEnter**.

The **pageWillEnter event** runs when the page is about to enter and become the active page.

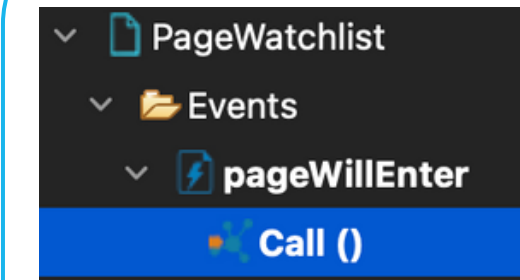
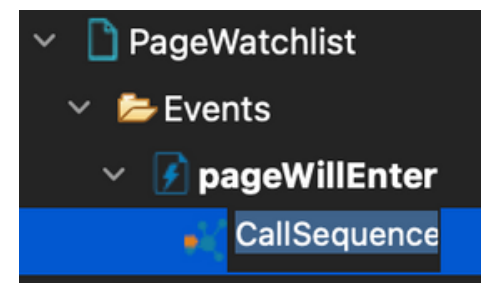
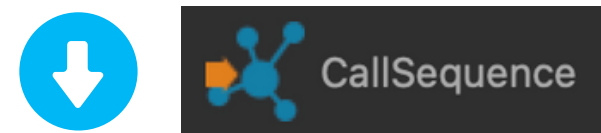
In the **PageEvent properties**, change the **Event type** property value to **pageWillEnter**.



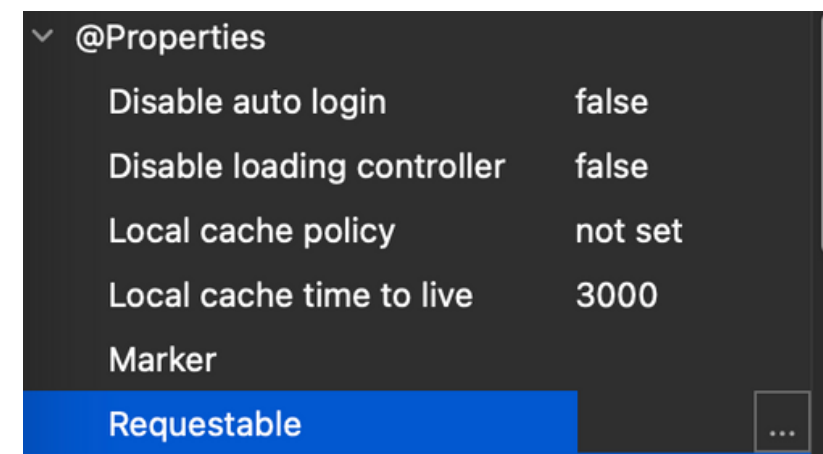
6.7 Calling a sequence with PageEvent



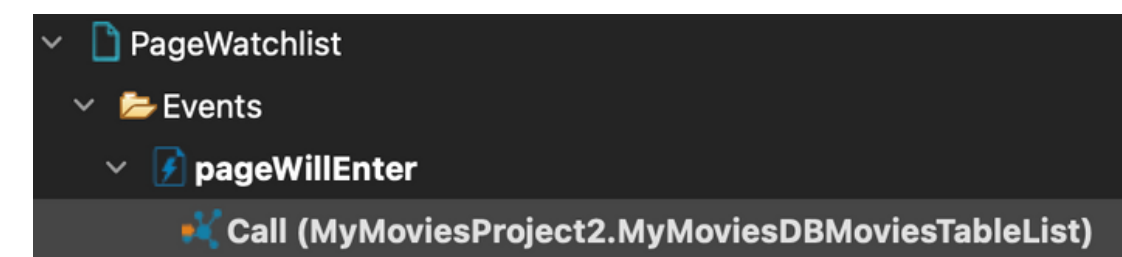
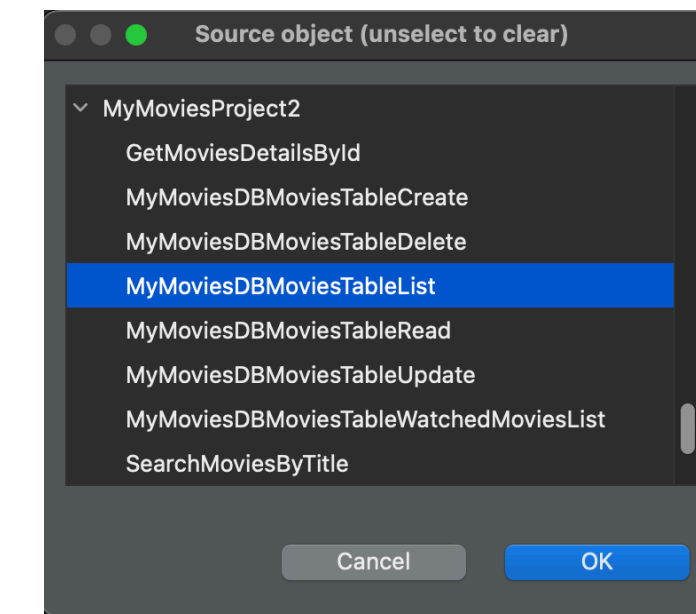
In pageWillEnter,
add a CallSequence
component.



In the **CallSequence's** properties,
in the **Requestable** property,
click on ...
to open the **Source object panel**.



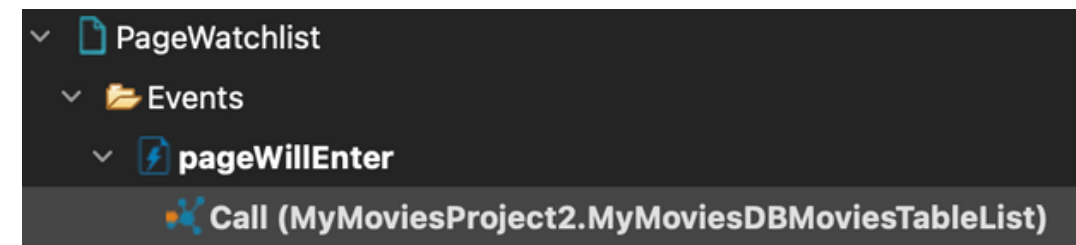
In the **Source object panel**,
select the **sequence**
MyMoviesDBMoviesTableList



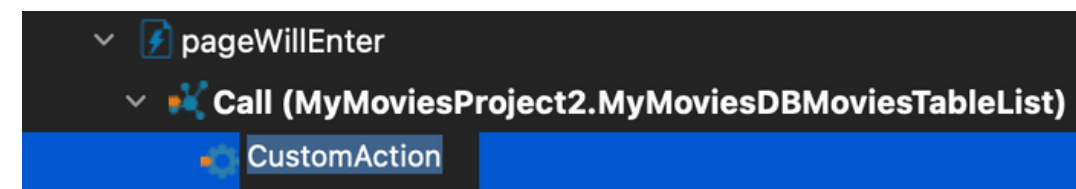
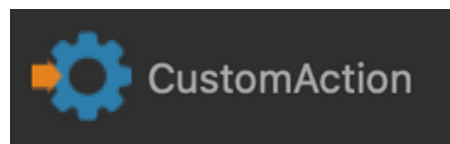
6.8 Use a CustomAction to process data

The sequence is now called when **PageWatchlist** is about to enter.

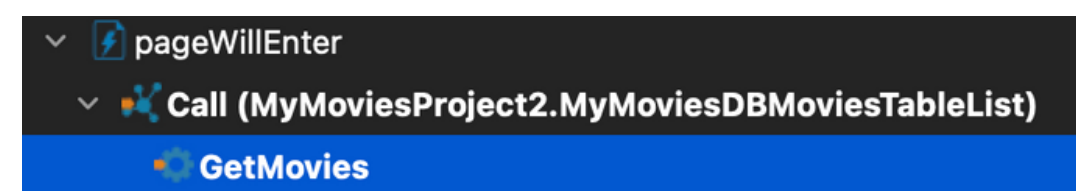
We are going to use a **CustomAction** component to get and process the data.



Add a **CustomAction** component to the **CallSequence**.



Rename it **GetMovies**.



Click twice on **GetMovies** to open the **CTS** file.

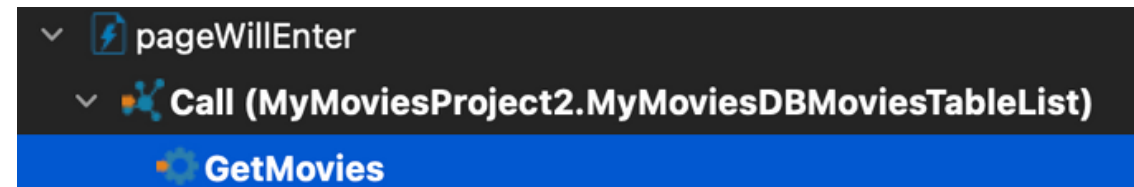
```

1+ import { Component }
31 @Component({selector: 'page-pagewatchlist', templateUrl: 'pagewatchlist.html', styleUrls
32 export class _Type_ extends C8oPage {
33
34 /**
35  * Function GetMovies
36  *
37  *
38  * @param page , the current page
39  * @param props , the object which holds properties key-value pairs
40  * @param vars , the object which holds variables key-value pairs
41  * @param event , the current event object
42  */
43 CTS1732110142565(page: PageWatchlist, props, vars, event: any) : Promise<any> {
44 return new Promise((resolve, reject) => {
45 /*Begin_c8o_function:CTS1732110142565*/
46 page.c8o.log.debug(' [MB] ' + props.actionFunction + ': ' + props.actionName);
47 resolve();
48 /*End_c8o_function:CTS1732110142565*/
49 });
50 }

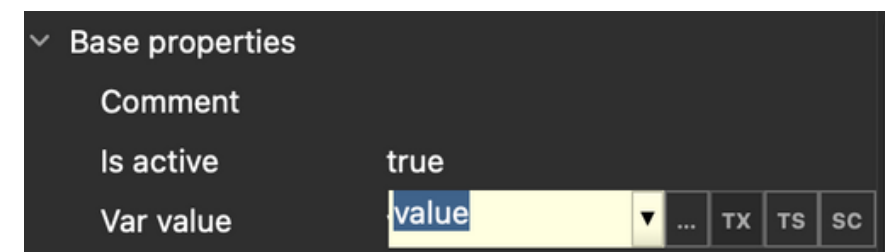
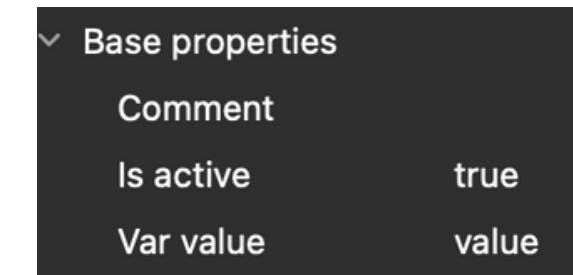
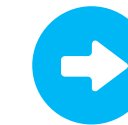
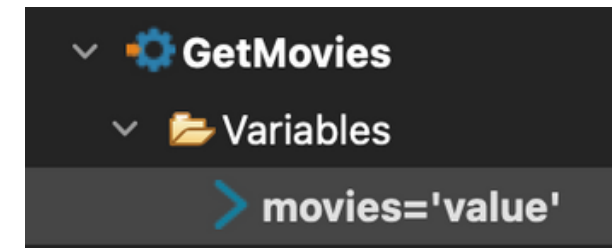
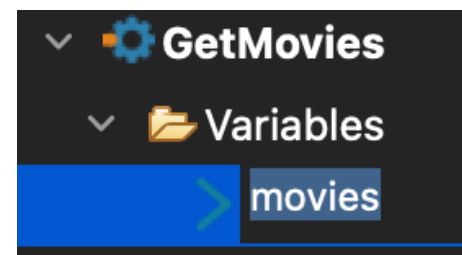
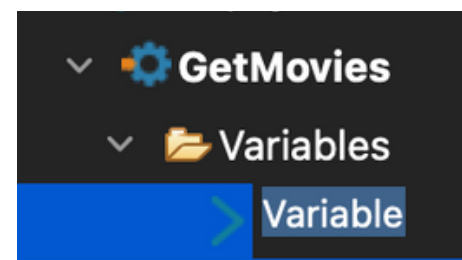
```



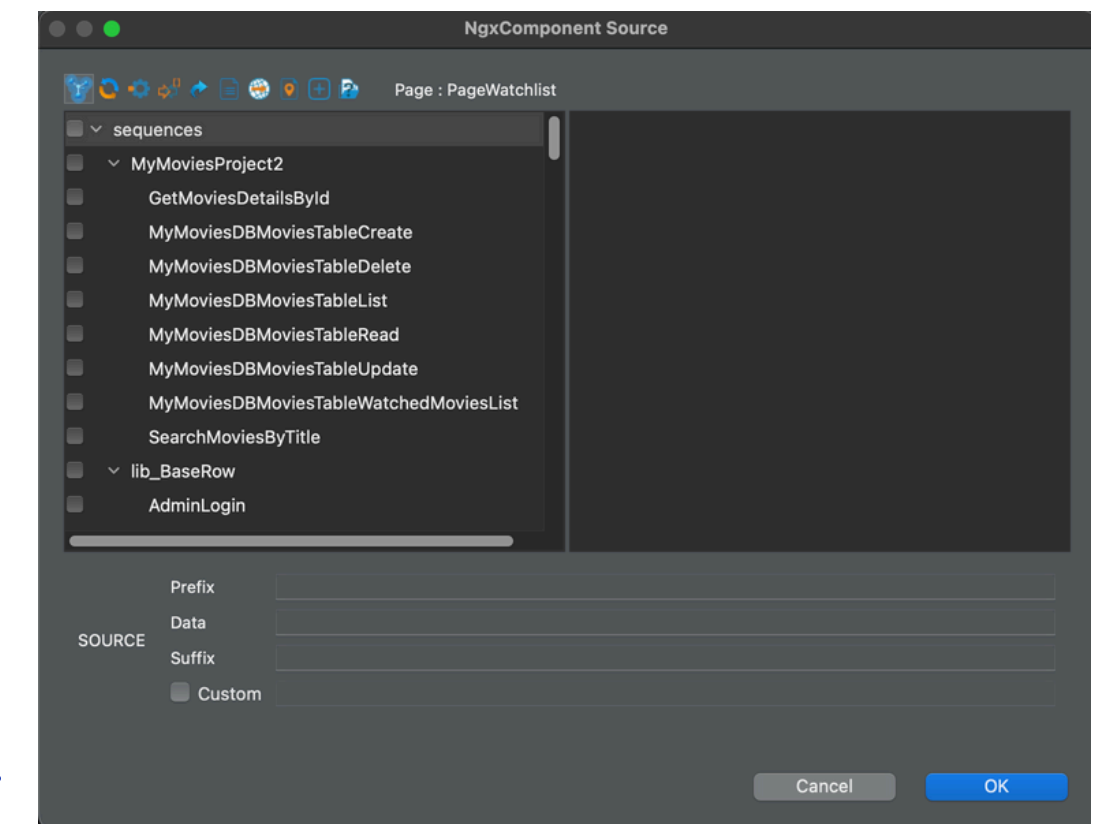
6.8 Use a CustomAction to process data



Add a **Variable** component to **GetMovies**.

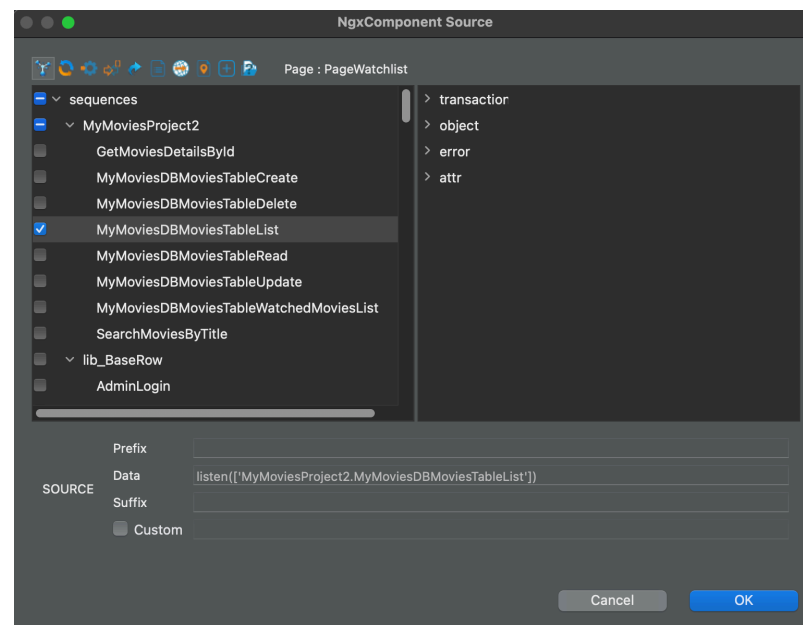


In the **movies Variable component's** properties, in the Var value property, click on ... or **SC** to open the **NgxComponent source panel**.

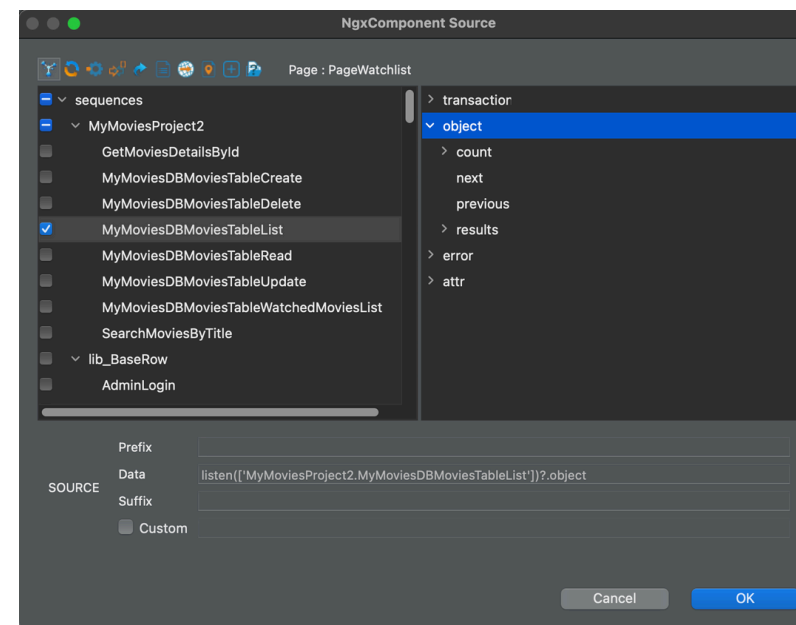


6.8 Use a CustomAction to process data

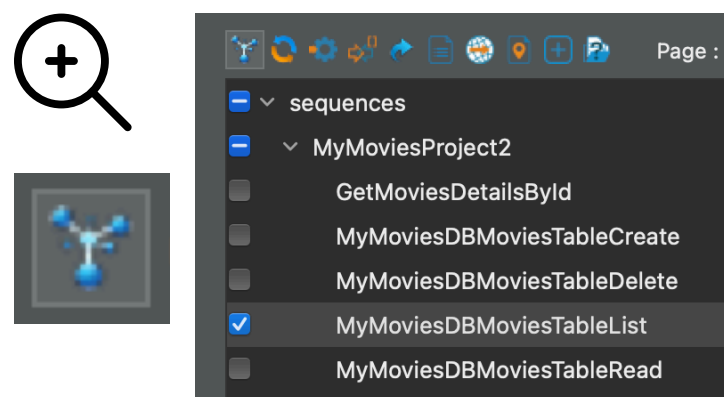
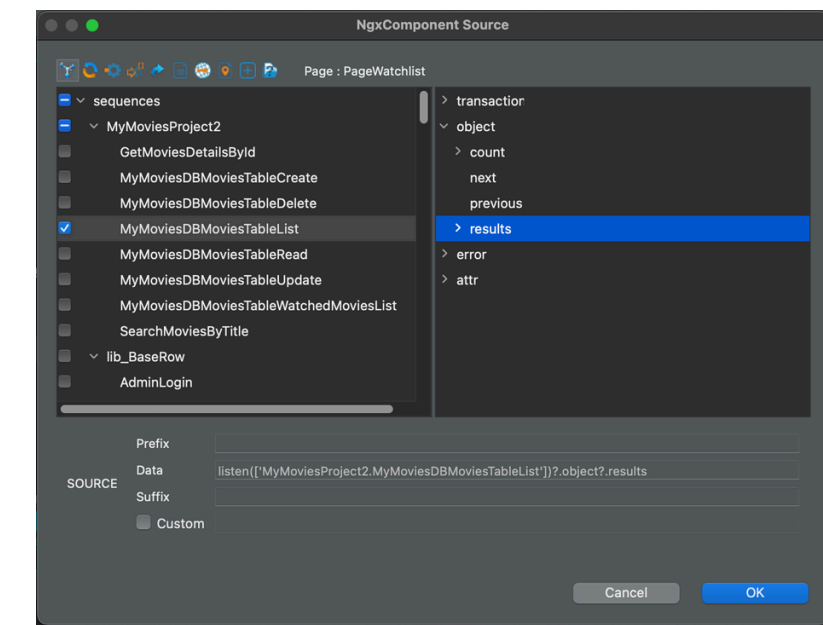
In the **NgxComponent** source panel,
Select the **sequence**
MyMoviesDBMoviesTableList.



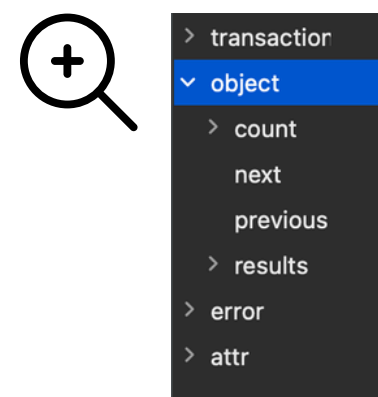
In the right part of the panel,
select **object**.



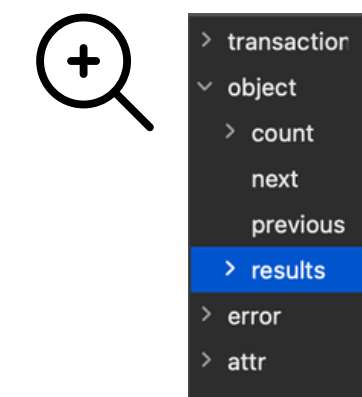
In object, select **results**
and click **OK**.



Prefix	
Data	listen(['MyMoviesProject2.MyMoviesDBMoviesTableList'])
Suffix	



Prefix	
Data	listen(['MyMoviesProject2.MyMoviesDBMoviesTableList'])?object
Suffix	



Prefix	
Data	listen(['MyMoviesProject2.MyMoviesDBMoviesTableList'])?object?.results
Suffix	



6.8 Use a CustomAction to process data

The **movies** Variable is now linked to the **results** array in the response of the sequence **MyMoviesDBMoviesTableList**.

Base properties	
Comment	
Is active	true
Var value	MyMoviesProject2.MyMoviesDBMoviesTableList->object.results



Call (MyMoviesProject2.MyMoviesDBMoviesTableList)
GetMovies
Variables
<code>> movies=MyMoviesProject2.MyMoviesDBMoviesTableList->object.results ..</code>



In the **CTS** file of **GetMovies**, add the following code:
console.log('vars.movies', vars.movies);
as shown below.

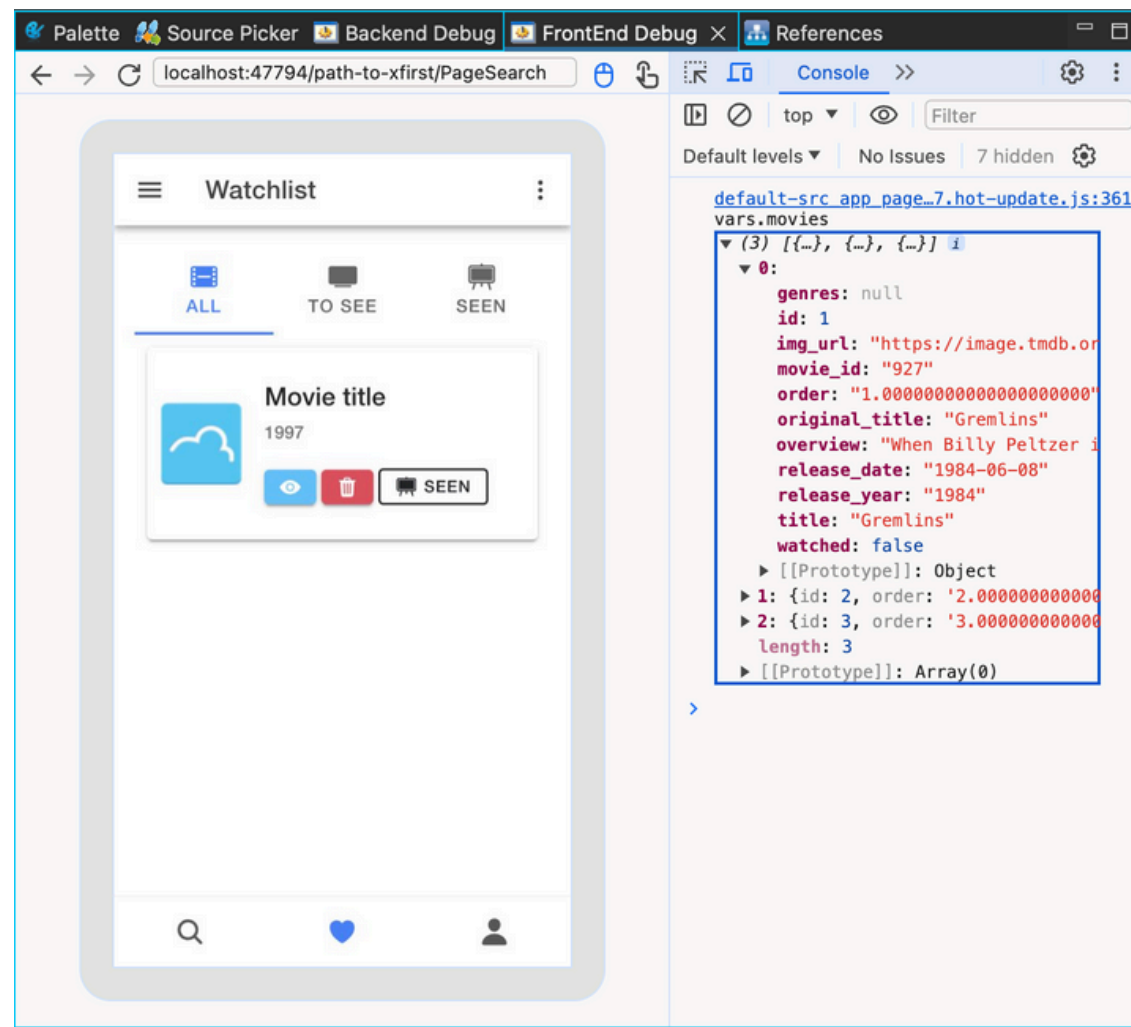
```
CTS1732110142565.temp.ts X
1 import { Component }
31 @Component({selector: 'page-pagewatchlist', templateUrl: 'pagewatchlist.html', styleUrls
32 export class _Type_ extends C8oPage {
33
34 /**
35  * Function GetMovies
36  *
37  *
38  * @param page , the current page
39  * @param props , the object which holds properties key-value pairs
40  * @param vars , the object which holds variables key-value pairs
41  * @param event , the current event object
42  */
43 CTS1732110142565(page: PageWatchlist, props, vars, event: any) : Promise<any> {
44 return new Promise((resolve, reject) => {
45 /*Begin_c8o_function:CTS1732110142565*/
46 page.c8o.log.debug('[MB] ' + props.actionFunction + ': ' + props.actionName);
47
48 console.log('vars.movies', vars.movies);
49
50 resolve();
51 /*End_c8o_function:CTS1732110142565*/
52 });
53 }
54
55 }
56
```



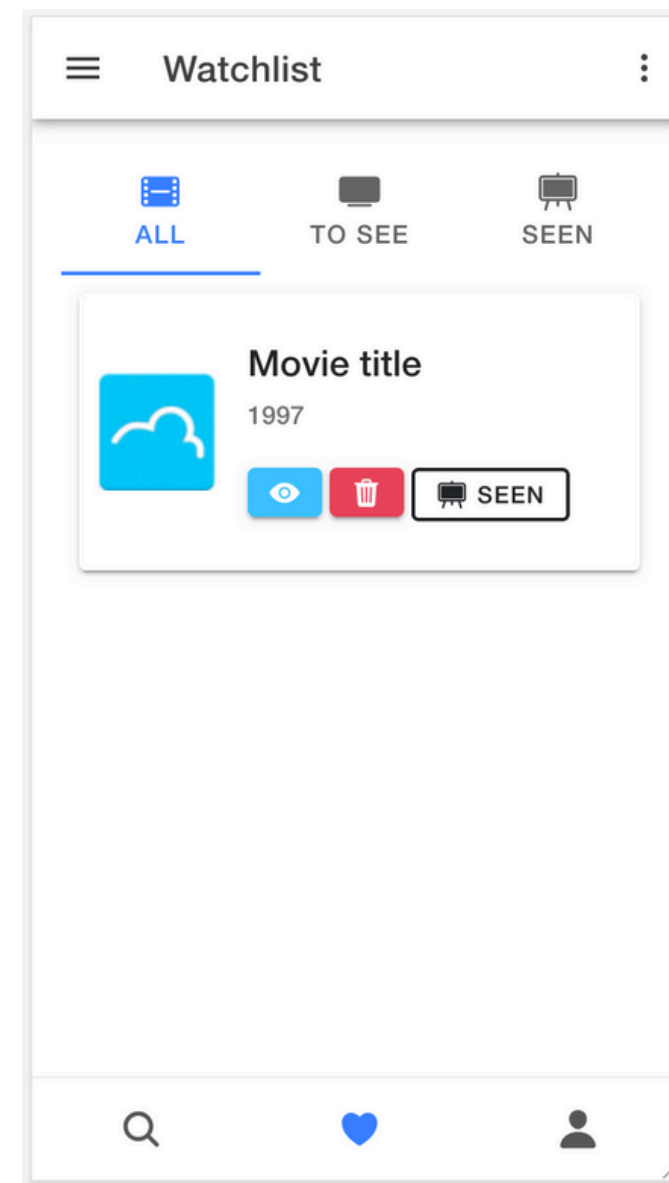
6.8 Use a CustomAction to process data

In the Frontend Debug panel or the browser, vars.movies is displayed in the console.

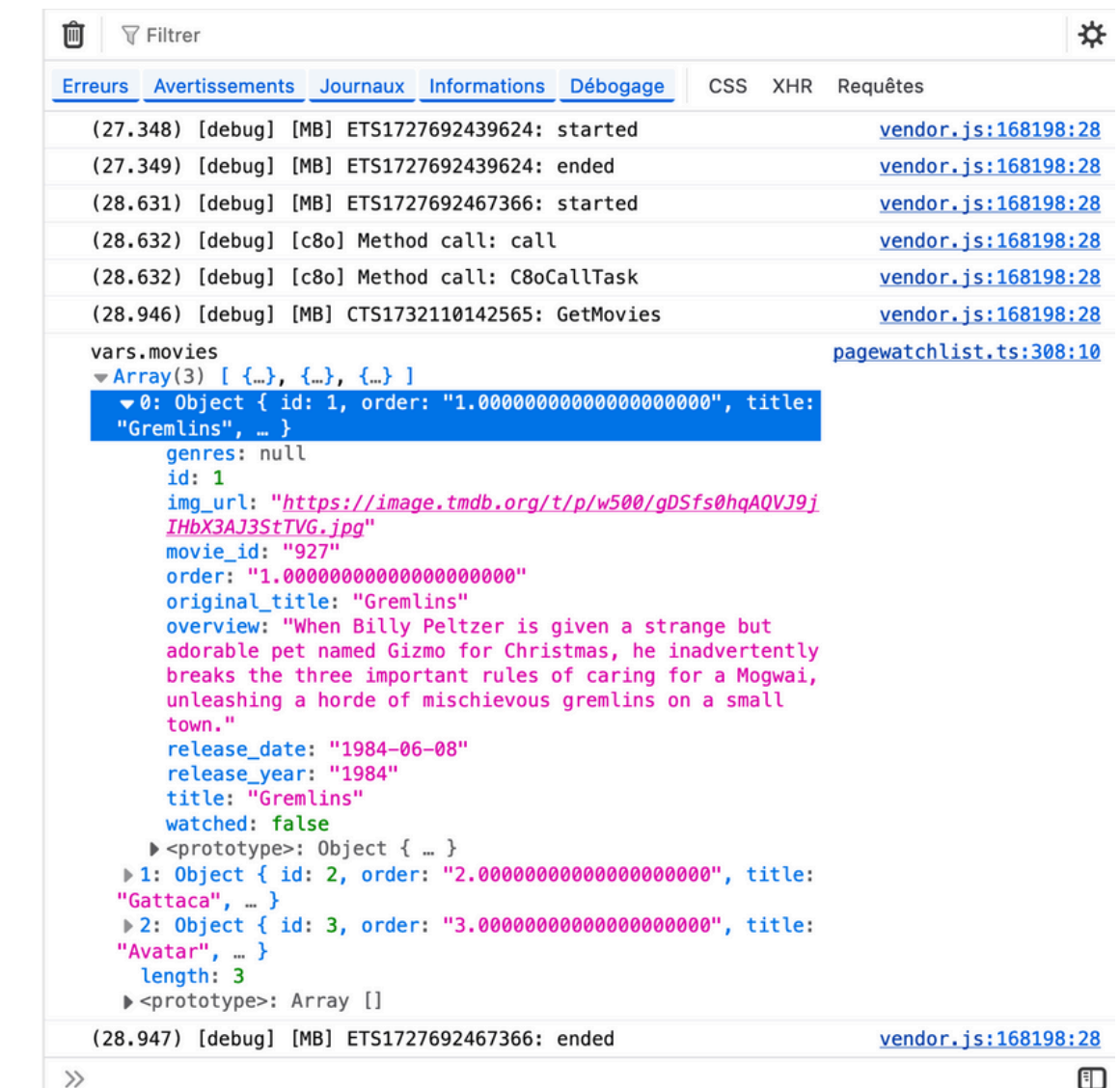
Frontend Debug panel



PageWatchlist in browser



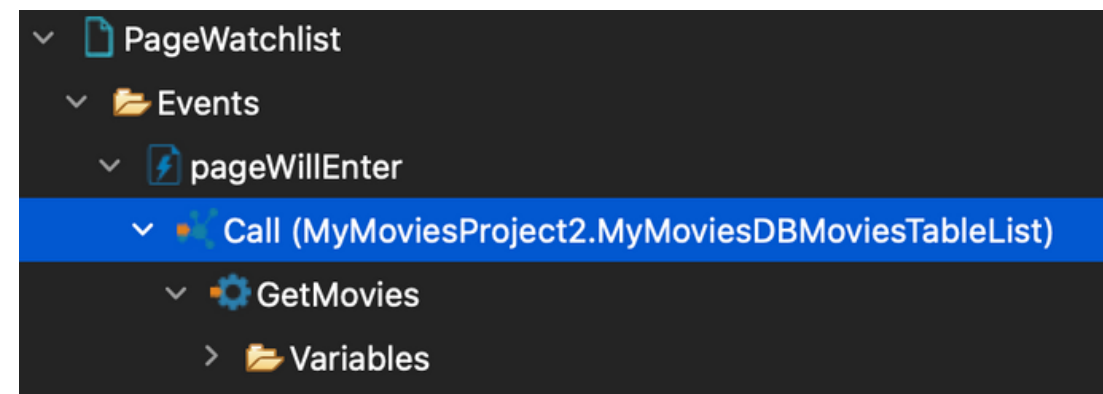
Browser console



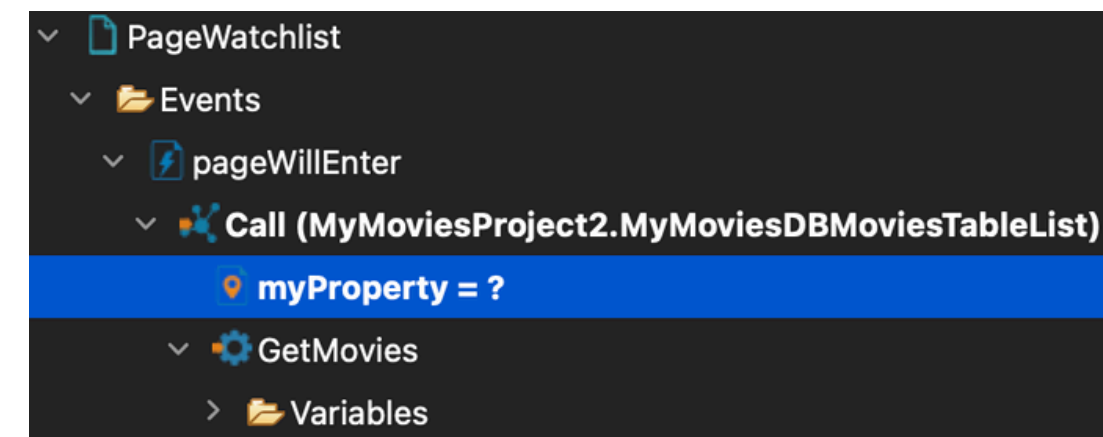
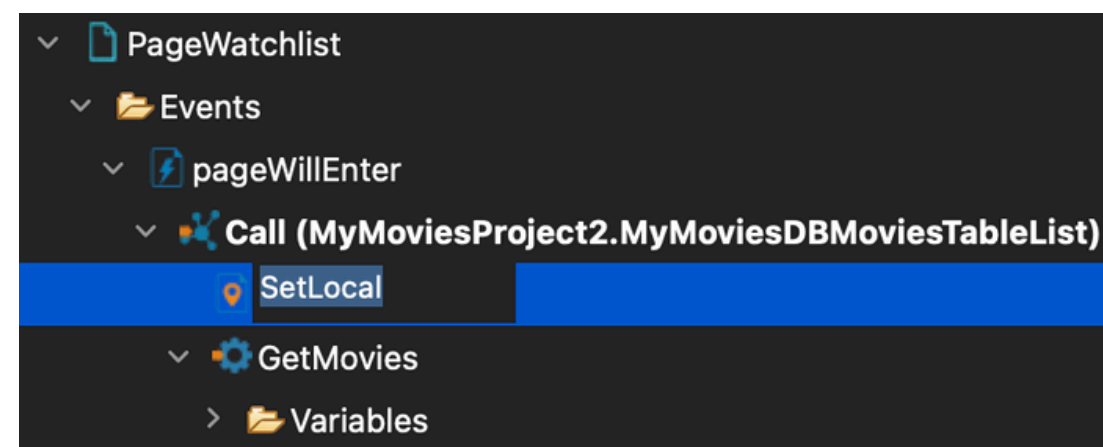
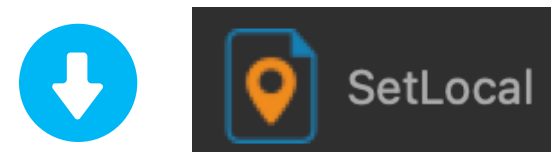
6.9 Set data as a local variable

Let's add local variables to the page to access data in components.

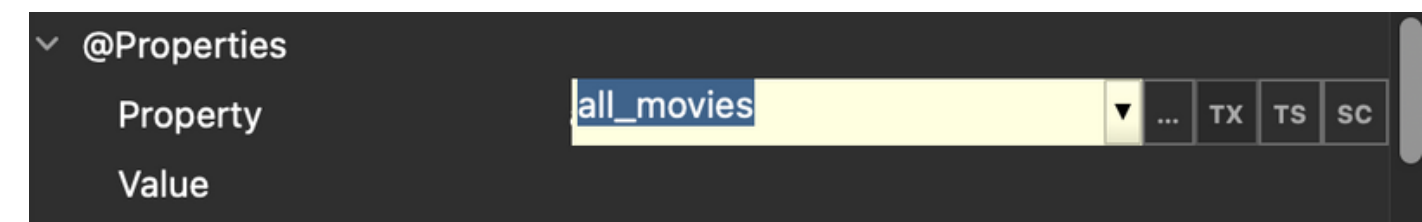
In **pageWatchlist**, in the **pageWillEnter PageEvent**



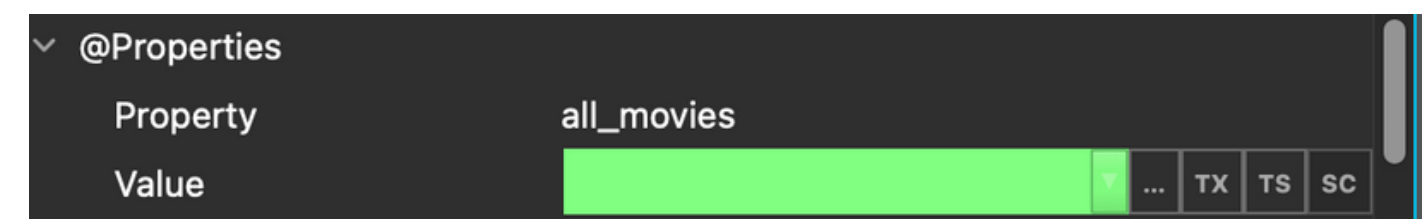
Add a **SetLocal** component
inside the **CallSequence**, before **GetMovies**.



↓ Rename it **all_movies**.

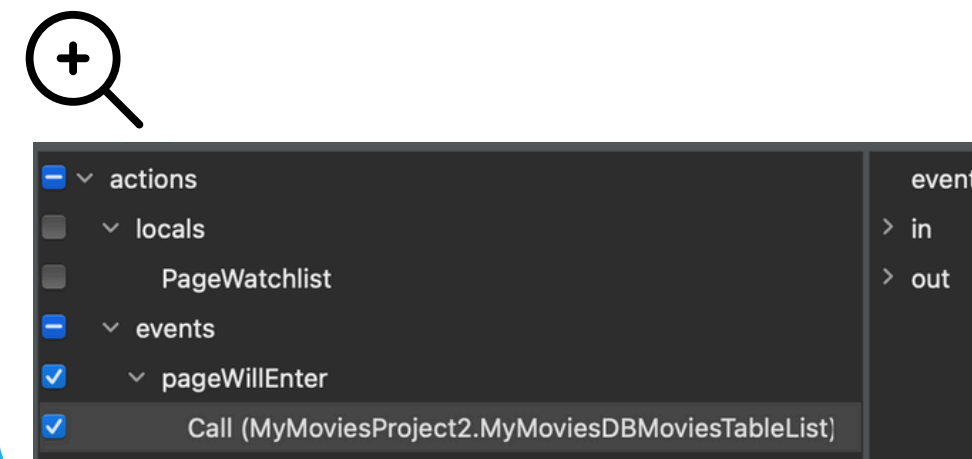
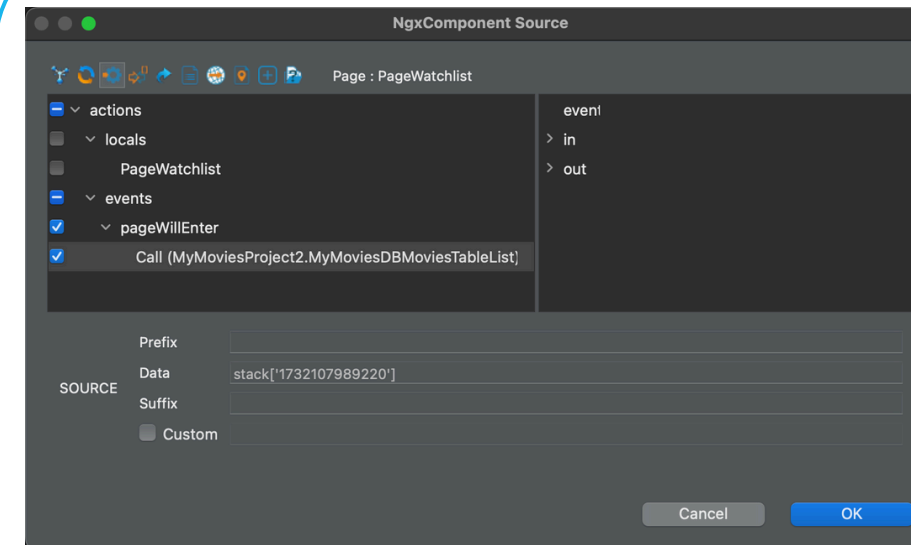
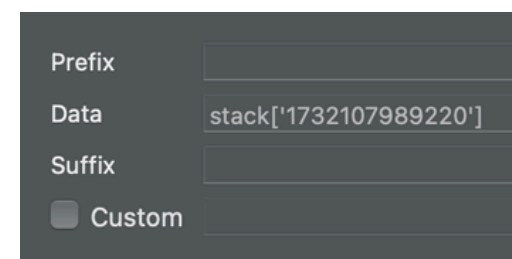
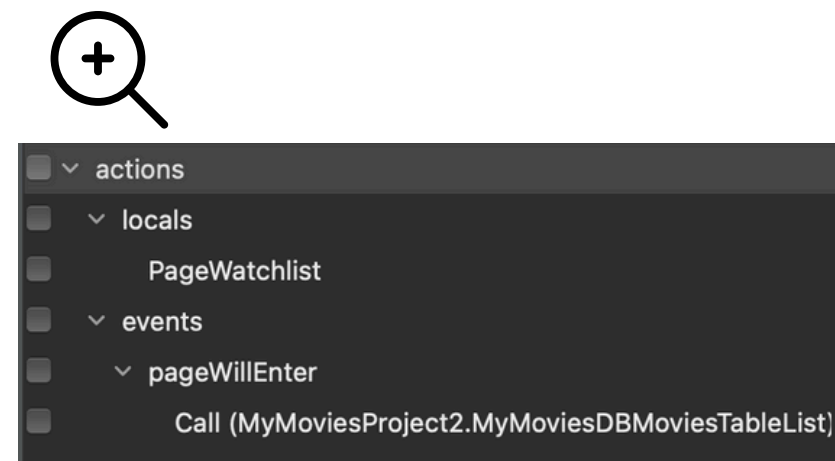
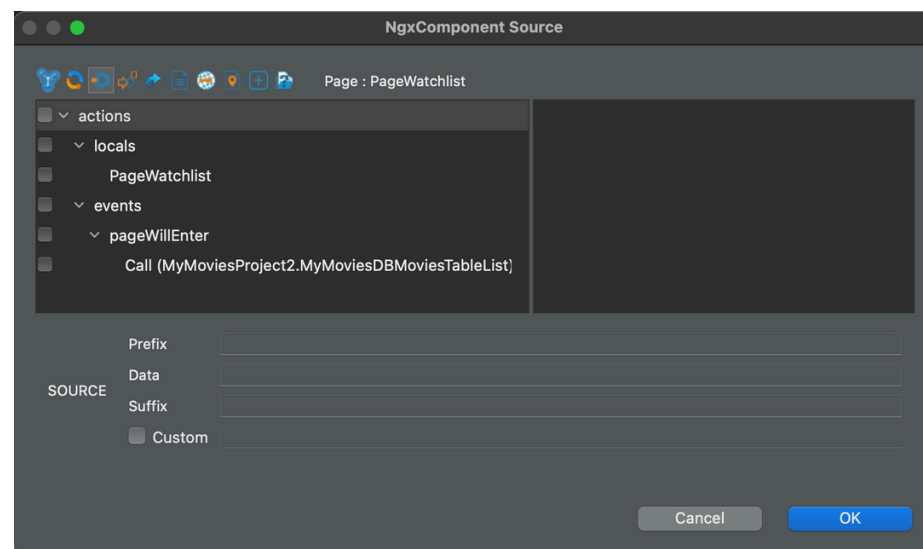
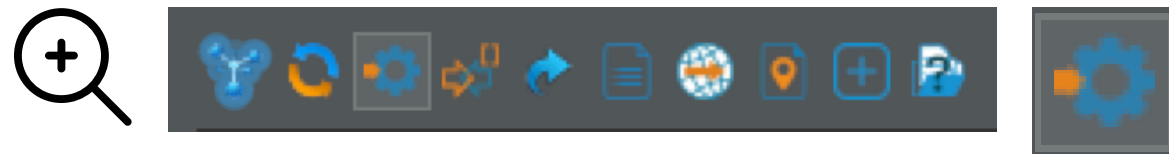


↓ Click on ... or **SC**
to open the **NgxComponent source panel**.

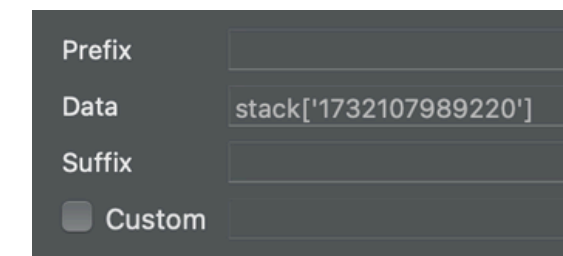


6.9 Set data as a local variable

In the **NgxComponent** source panel,
click on the **CustomAction** icon to display the actions as sources.

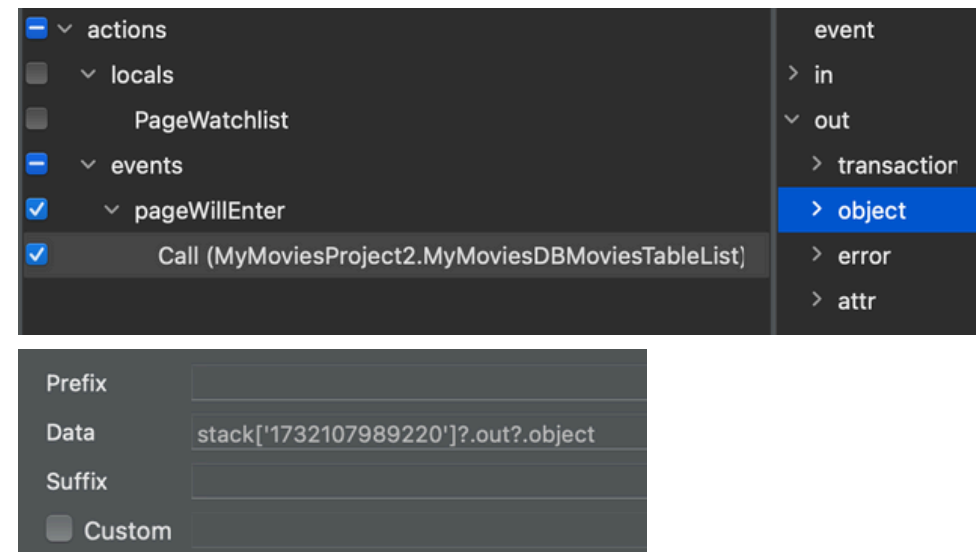
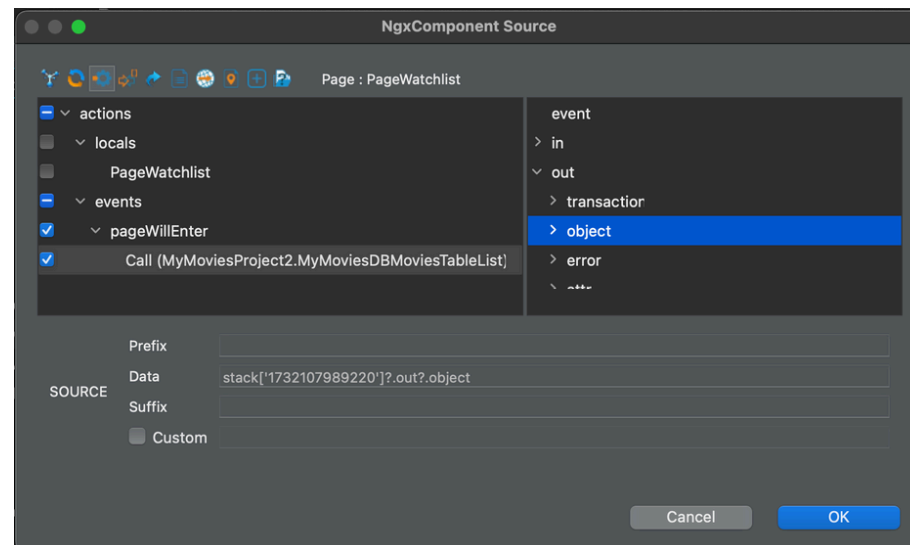


Then, click on the **CallSequence**
MyMoviesDBMoviesTableList
to select it as source.

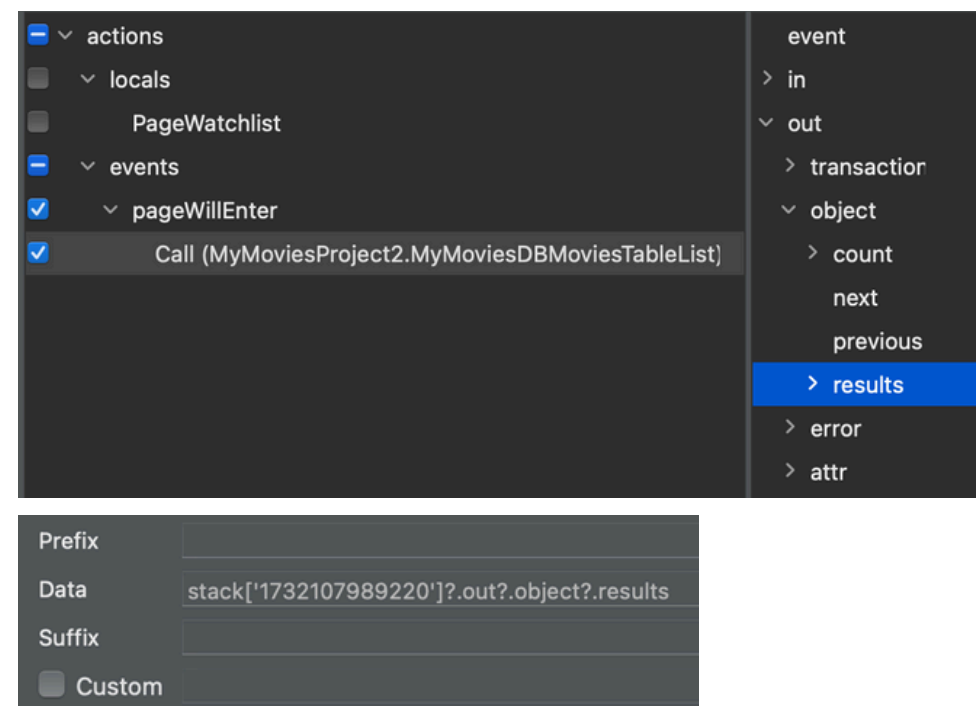
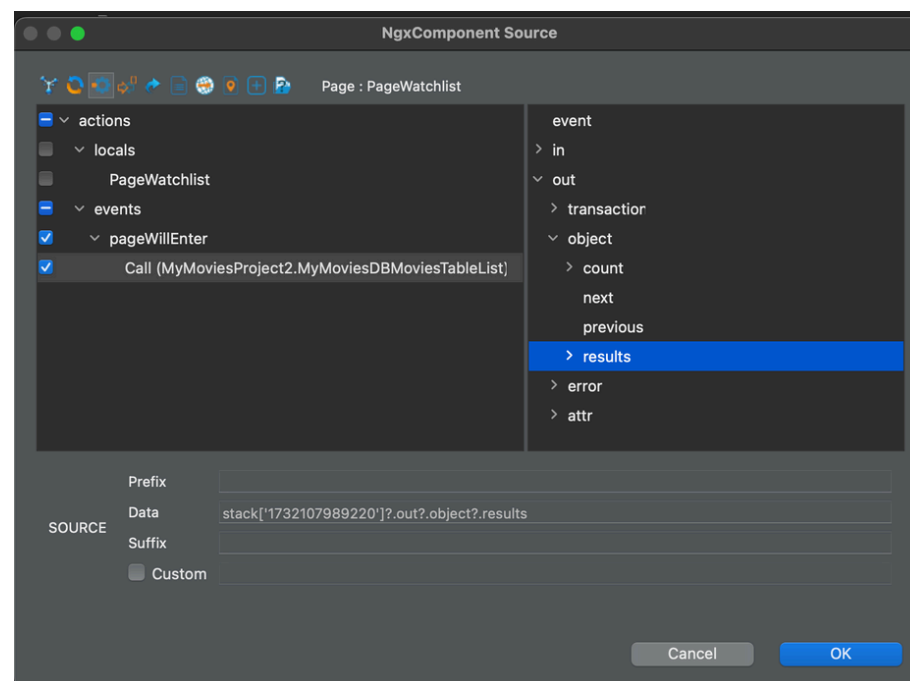


6.9 Set data as a local variable

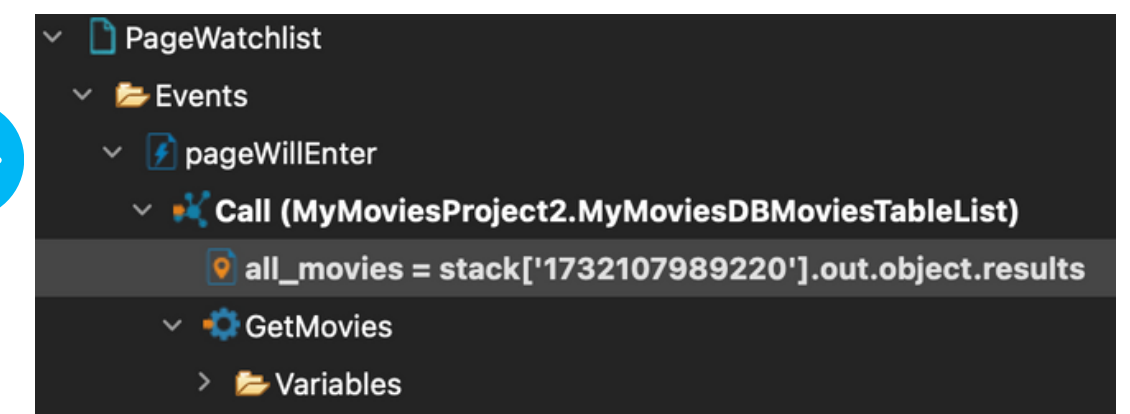
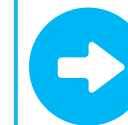
In the right panel, click on **object** to select it as source.



Then, click on **results** to select it as source.

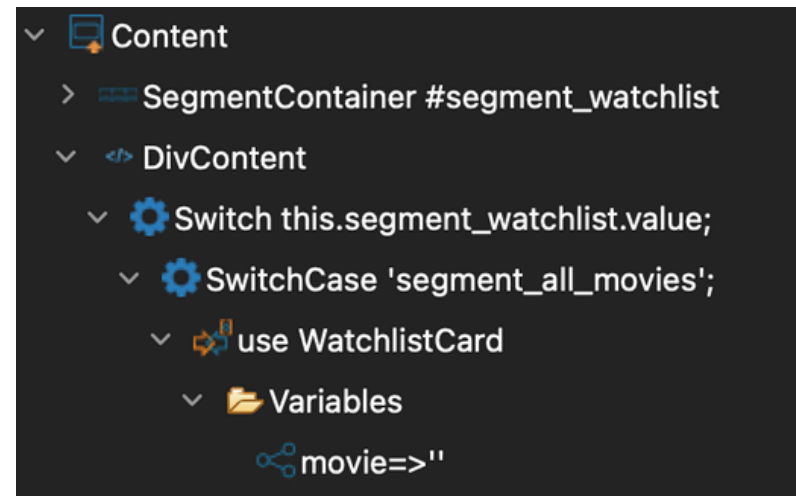


The **all_movies SetLocal**
is now sourced
on the **CallSequence**
MyMoviesDBMoviesTableList.

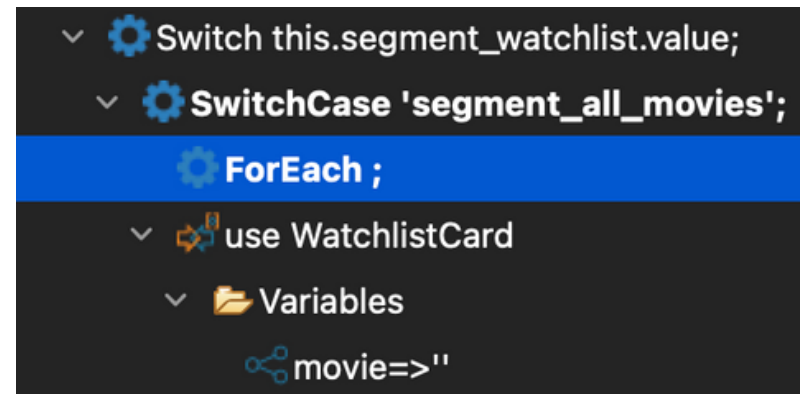
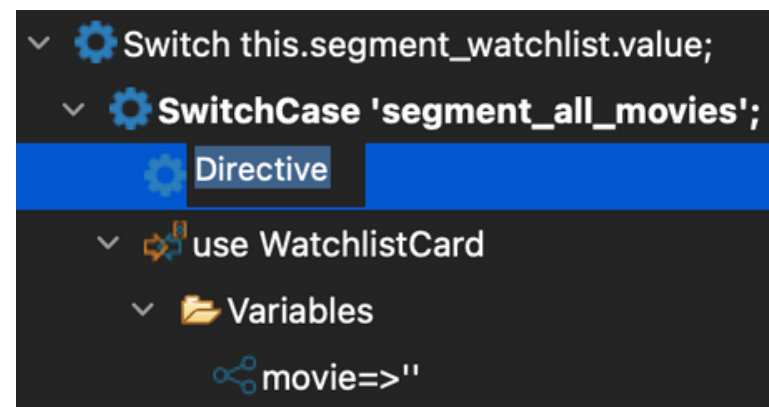
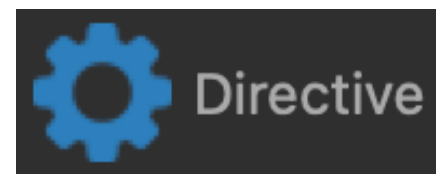


6.10 Display data from a local variable

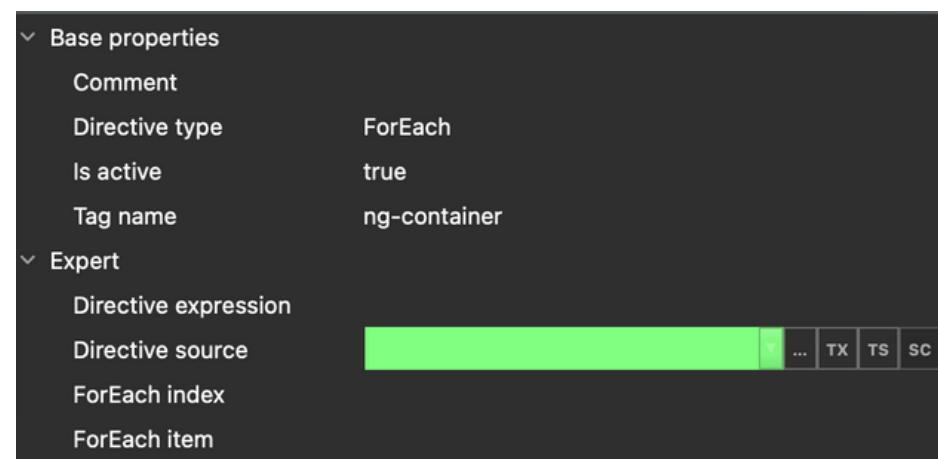
Now, let's display our data in **PageWatchlist** segments.



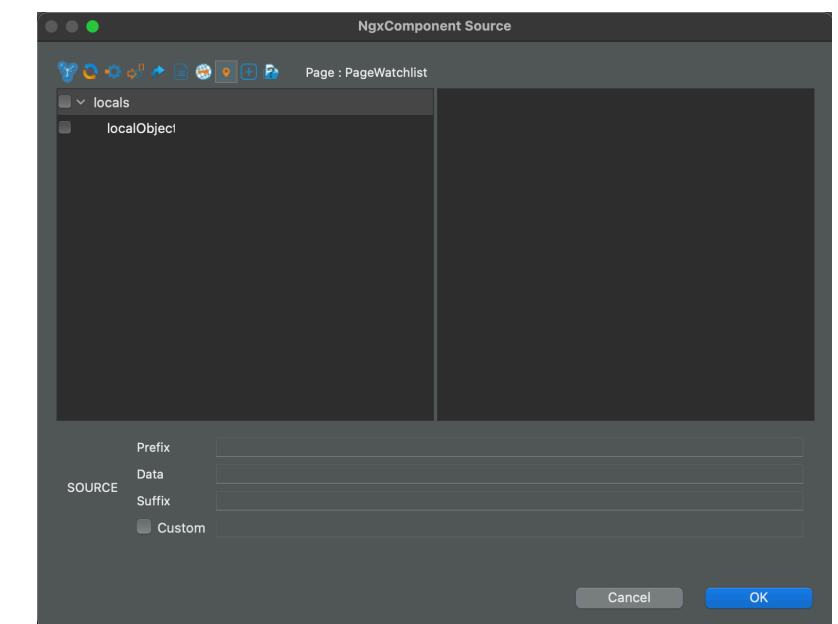
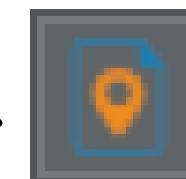
Add a **ForEach Directive** in the first **SwitchCase segment_all_movies**.



In the **ForEach Directive's** properties, in the **Directive Source** property, click on ... or **SC** to open the **NgxComponent Source** panel.

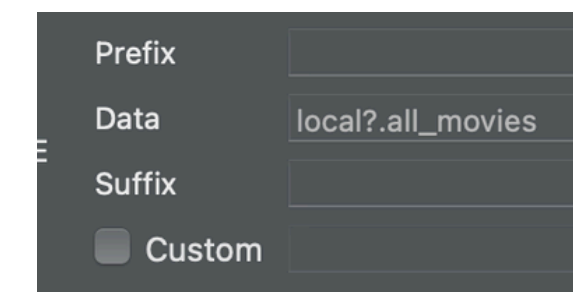
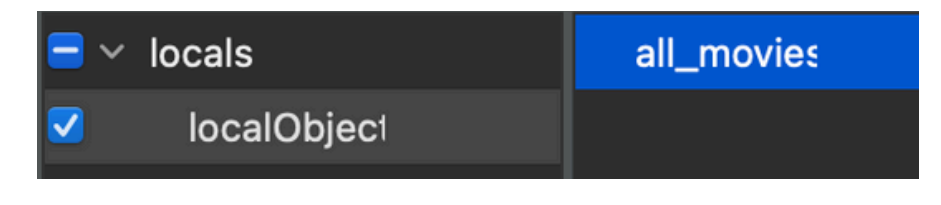
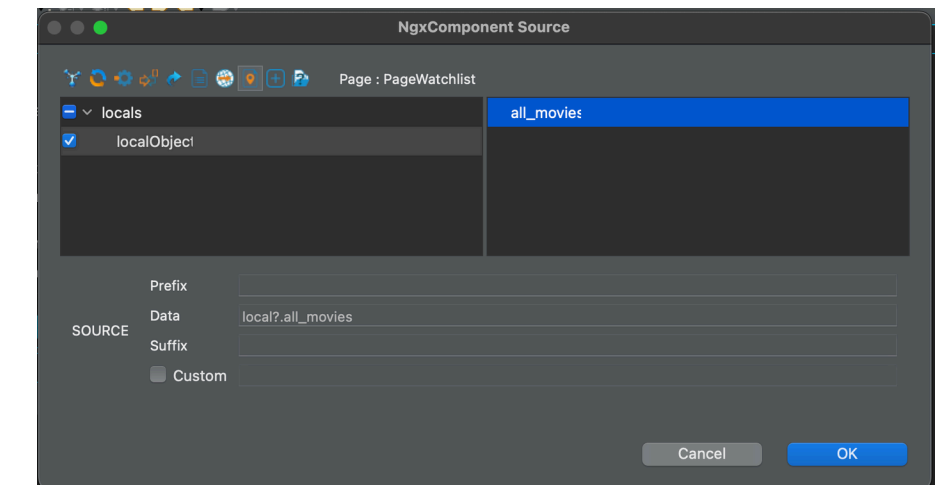
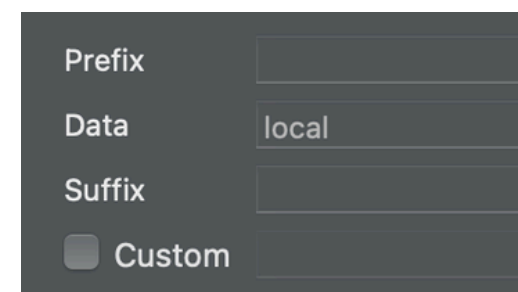
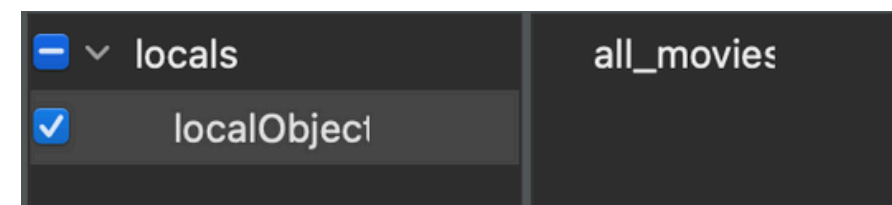
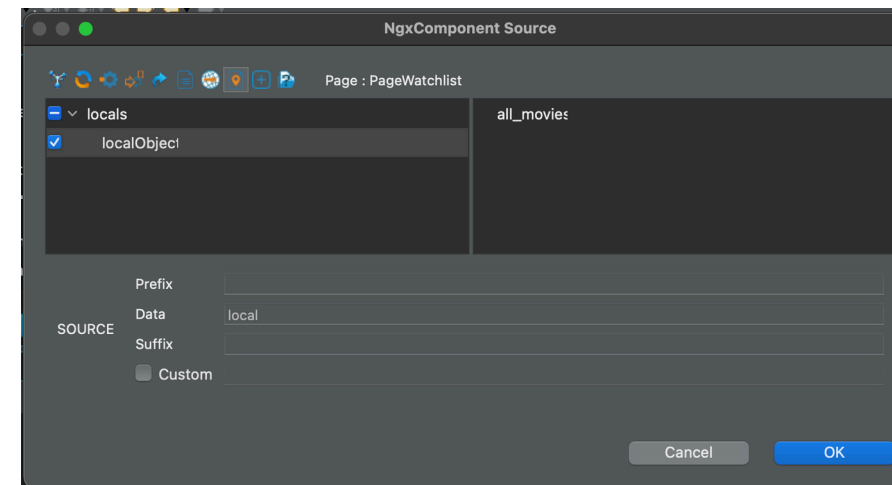
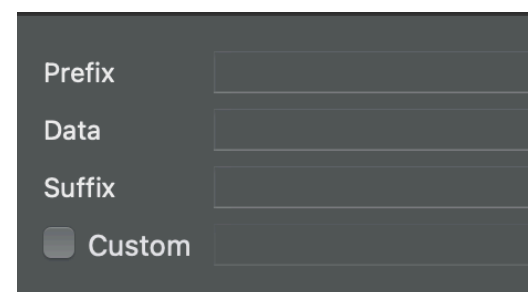
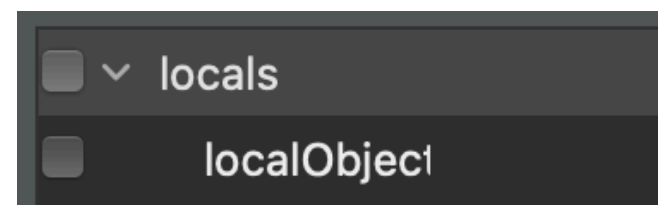
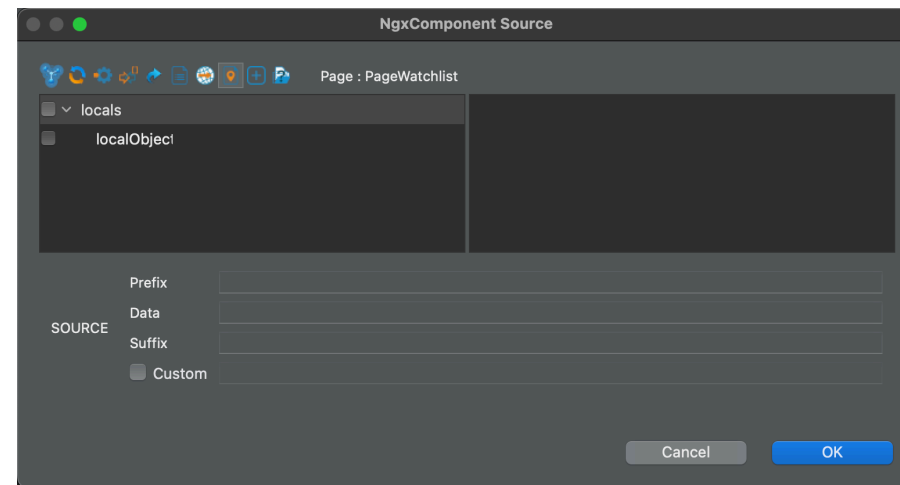


In the **NgxComponent source** panel, click on the **SetLocal** icon to display it as sources.



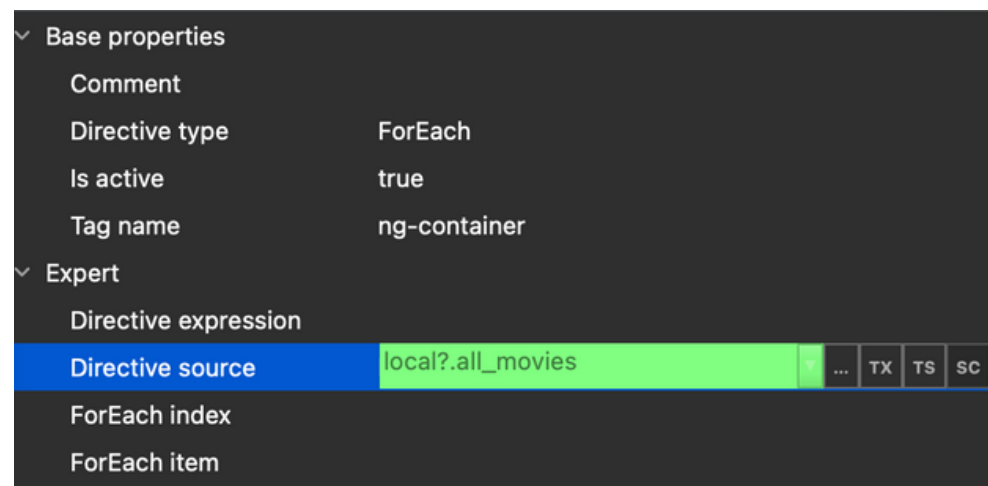
6.10 Display data from a local variable

Click on **localObject** to display the local sources. Then, click on **all_movies** to select the **all_movies SetLocal** as source.

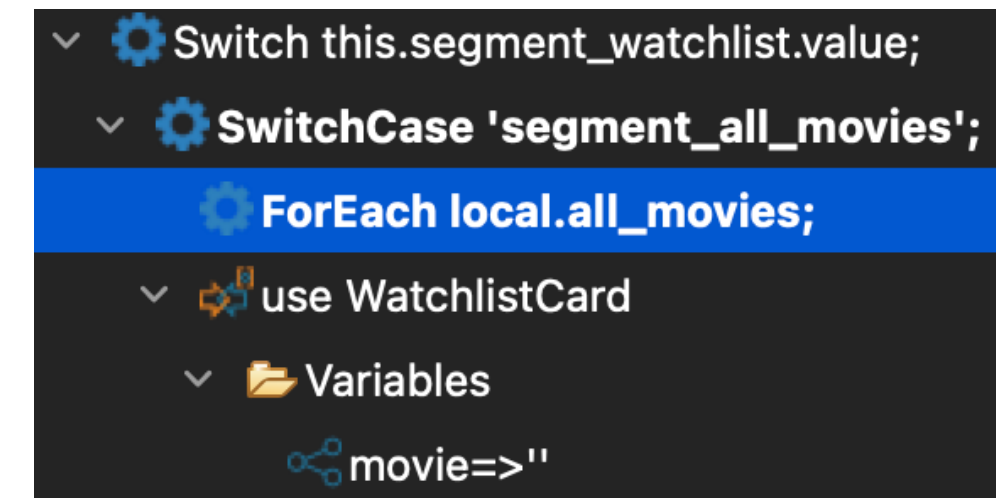
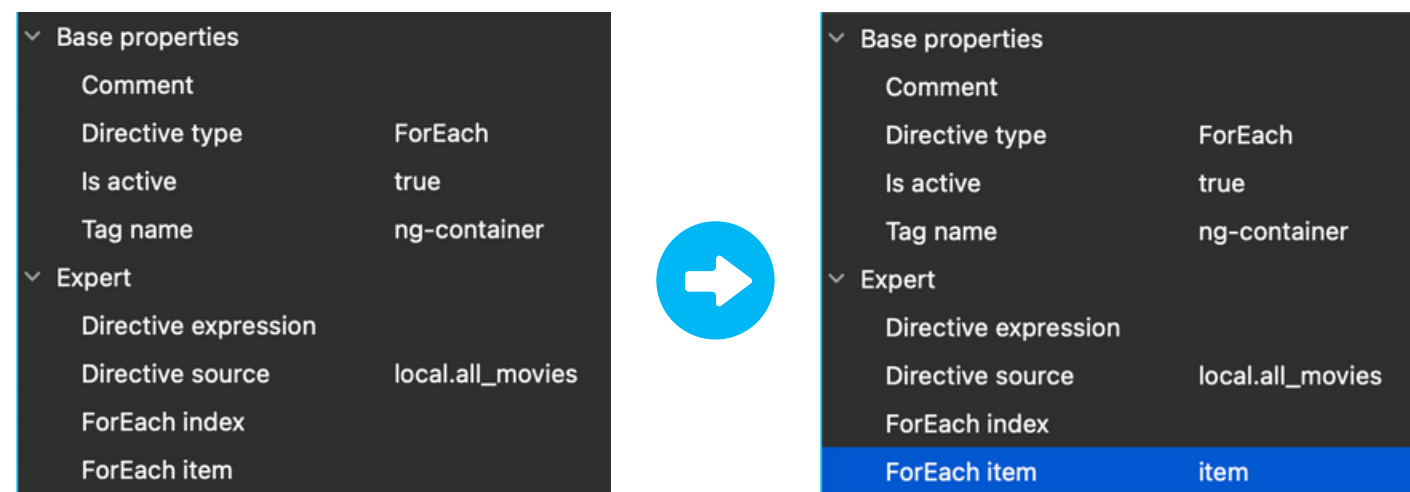


6.10 Display data from a local variable

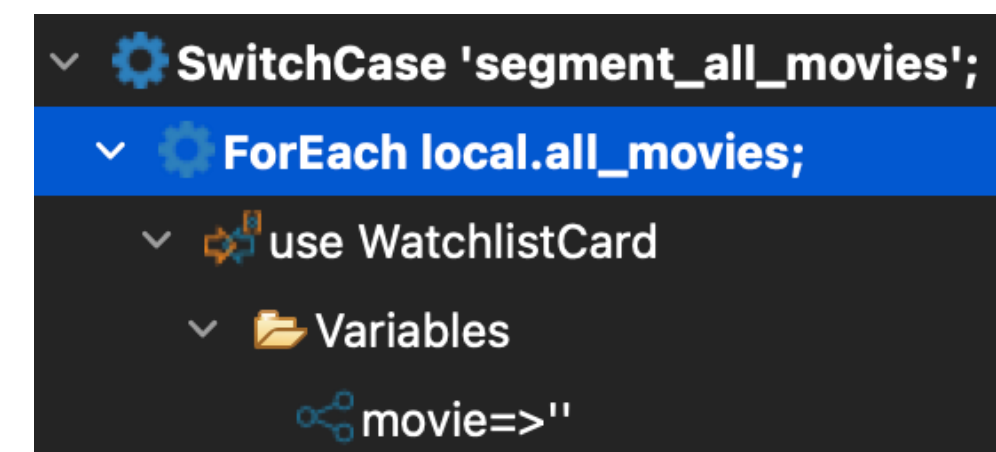
In the **ForEach Directive's** properties,
in the **Directive Source** property,
the **all_movies SetLocal** is now selected as source



In the **ForEach item** property, enter **item** as value.



In the **SwitchCase** **segment_all_movies**,
move the **use WatchlistCard**
inside the **ForEach Directive**.



6.10 Display data from a local variable

In the **SwitchCase** **segment_all_movies**, click on the **movie useVariable** in the **use WatchlistCard** to display its properties.

```
SwitchCase 'segment_all_movies';
  ForEach local.all_movies;
    use WatchlistCard
      Variables
        movie=>"
```



Enter **item** as **Var value**.

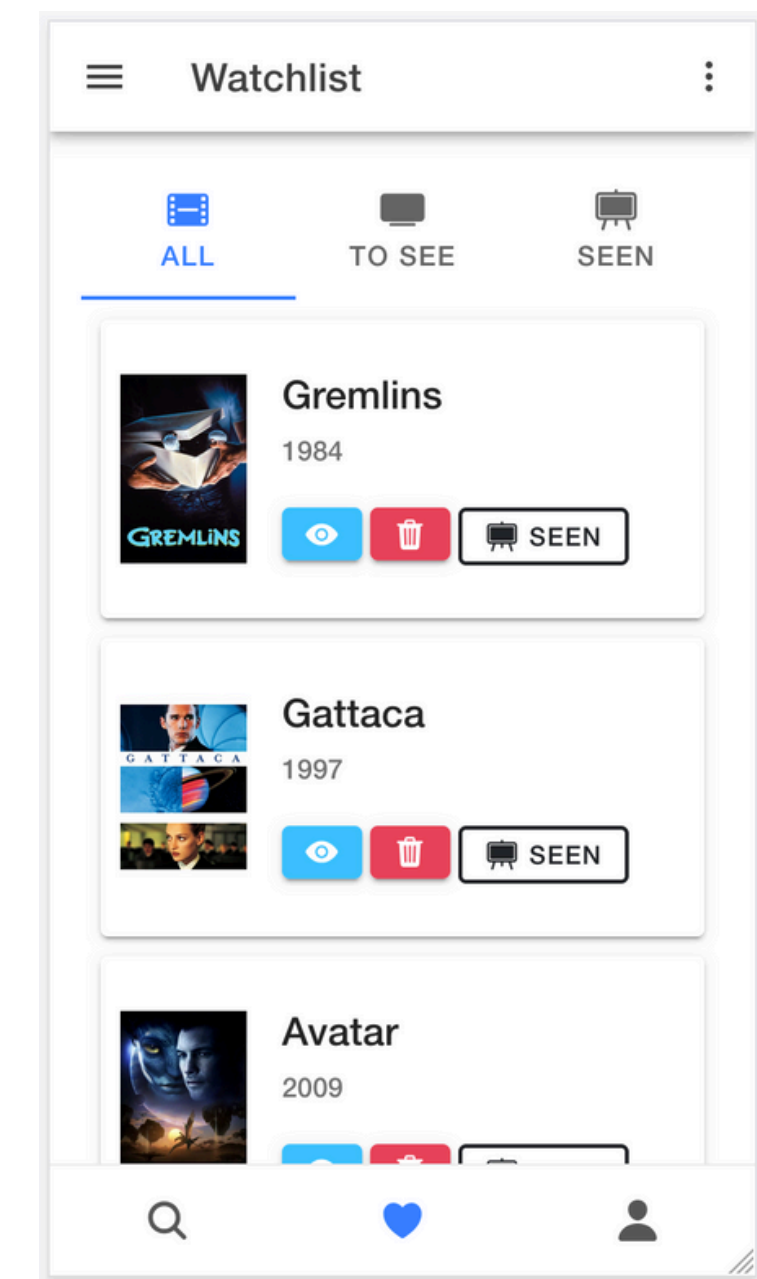
```
@Component
  Binding oneWayBinding
  Var value item
```



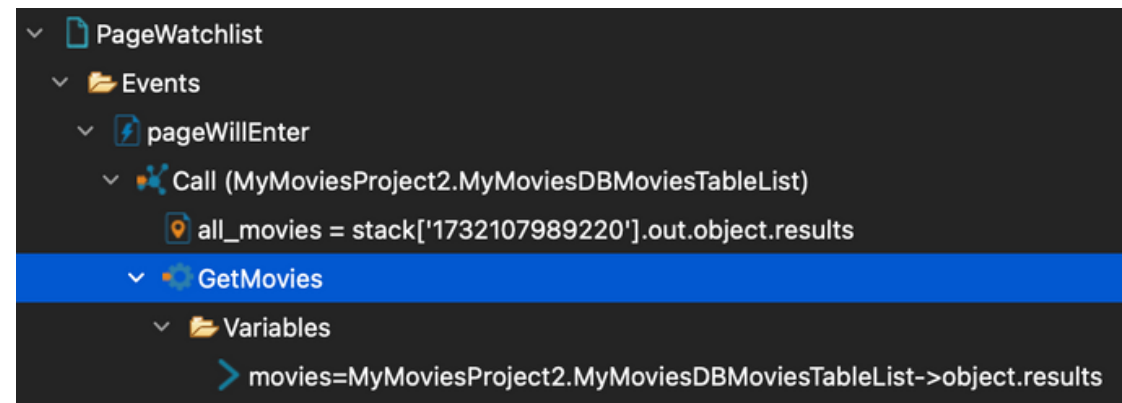
```
Switch this.segment_watchlist.value;
  SwitchCase 'segment_all_movies';
    ForEach local.all_movies;
      use WatchlistCard
        Variables
          movie=>item
```



The data appears in the segment.



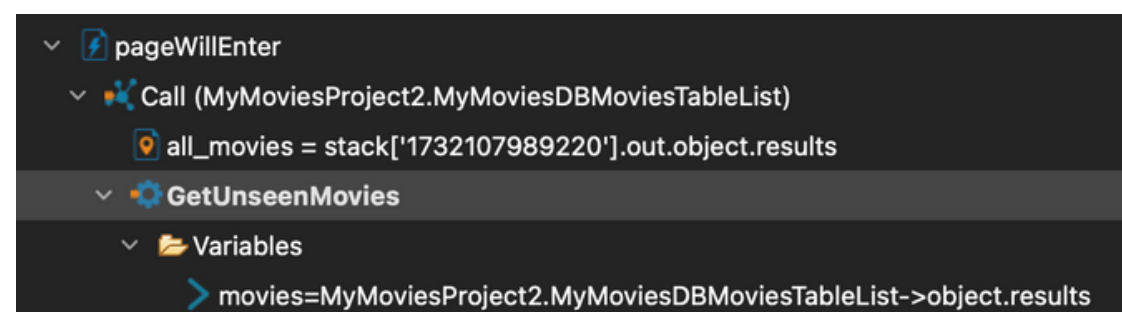
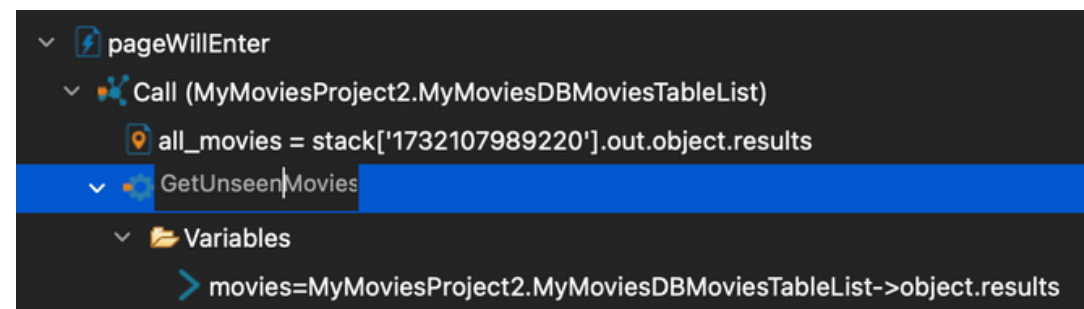
6.11 Display filtered data in segments



In the **pageWillEnter** PageEvent,



rename the **GetMovies** CustomAction to **GetUnseenMovies**.



```
30 import { PageWatchlist } from './pagewatchlist';
31 @Component({selector: 'page-pagewatchlist', templateUrl: 'pagewatchlist.html', stylel
32 export class _Type_ extends C8oPage {
33
34   /**
35    * Function GetUnseenMovies
36    *
37    *
38    * @param page , the current page
39    * @param props , the object which holds properties key-value pairs
40    * @param vars , the object which holds variables key-value pairs
41    * @param event , the current event object
42    */
43   CTS1732110142565(page: PageWatchlist, props, vars, event: any) : Promise<any> {
44     return new Promise((resolve, reject) => {
45       /*Begin_c8o_function:CTS1732110142565*/
46
47       console.log('vars.movies', vars.movies);|
48
49       resolve();
50       /*End_c8o_function:CTS1732110142565*/
51     });
52   }
```

In **GetUnseenMovies** CTS file, add the following code:



```
let unseenMovies = vars.movies.filter(movie => movie.watched == false);
console.log(unseenMovies);
resolve(unseenMovies);
```

```
42   /**
43   CTS1732110142565(page: PageWatchlist, props, vars, event: any) : Promise<any> {
44     return new Promise((resolve, reject) => {
45       /*Begin_c8o_function:CTS1732110142565*/
46
47       console.log('vars.movies', vars.movies);
48
49       let unseenMovies = vars.movies.filter(movie => movie.watched == false);
50       console.log(unseenMovies);
51
52       resolve(unseenMovies);
53       /*End_c8o_function:CTS1732110142565*/
54     });
55   }
56
57 }
```



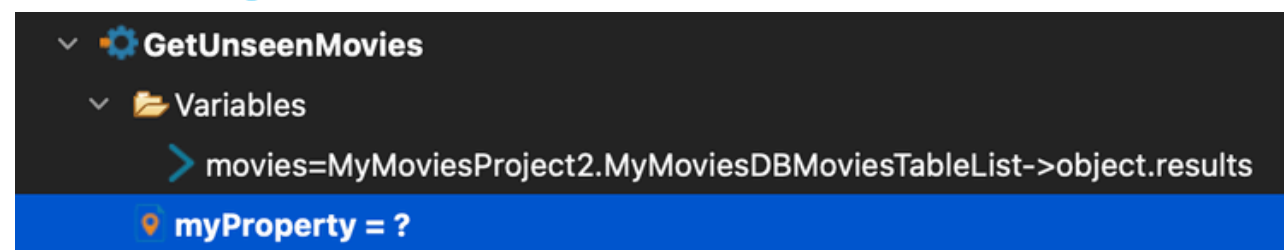
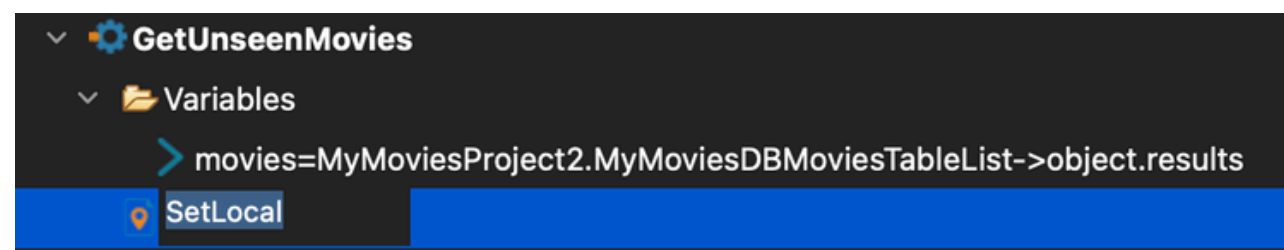
6.11 Display filtered data in segments

In the browser console,
we can see the data from `console.log(unseenMovies);`

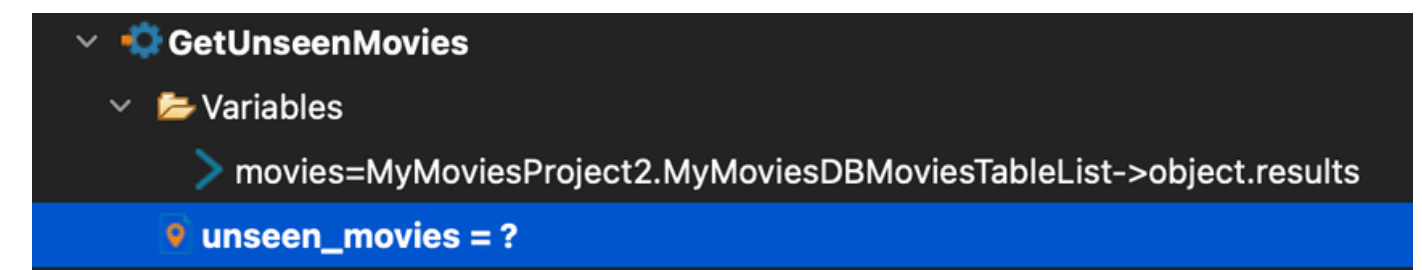
```
vars.movies ▶ Array(3) [ {...}, {...}, {...} ] pagewatchlist.ts:345:10
▼ Array [ {...}, {...} ] pagewatchlist.ts:348:10
  ▶ 0: Object { id: 2, order: "2.00000000000000000000",
    title: "Gattaca", ... }
  ▶ 1: Object { id: 3, order: "3.00000000000000000000",
    title: "Avatar", ... }
    length: 2
    <prototype>: Array []
```

Now, let's use the data from `resolve(unseenMovies);`
in a **local variable**.

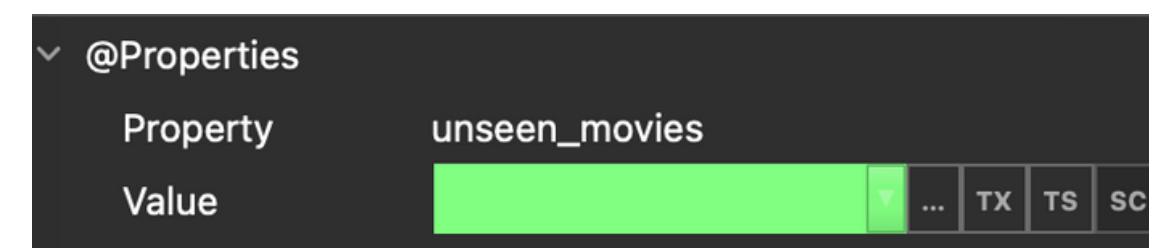
Add a **SetLocal** component in **GetUnseenMovies**.



Rename it **unseen_movies**.

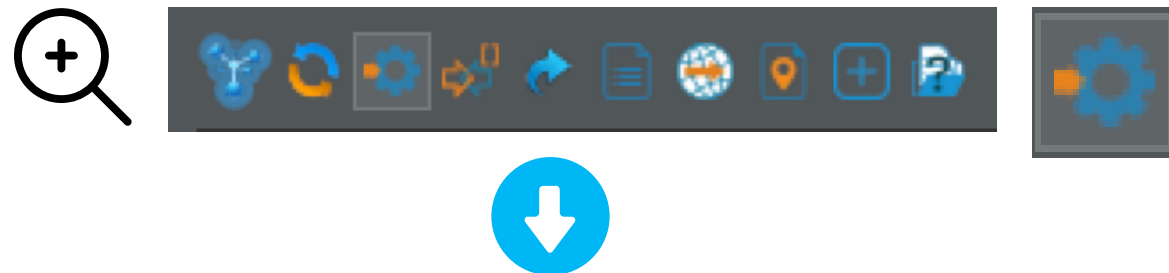


Click on ... or **SC** to open
the **NgxComponent Source** panel.

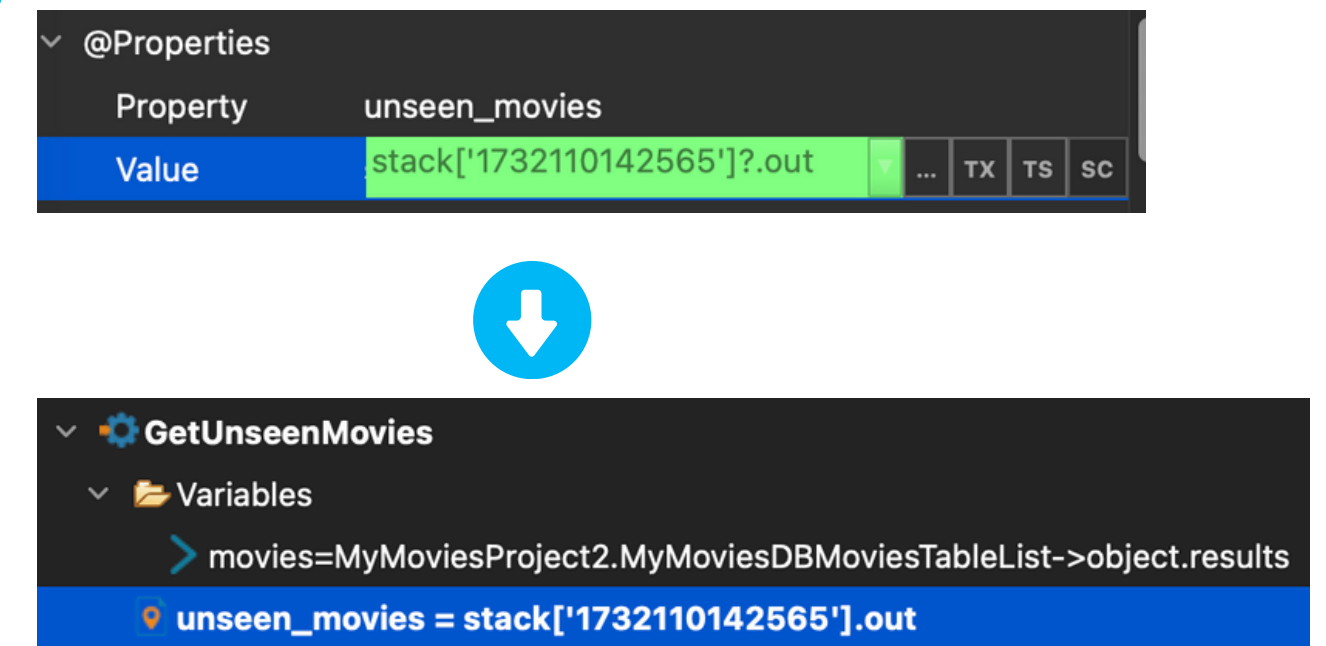
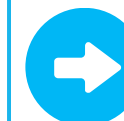
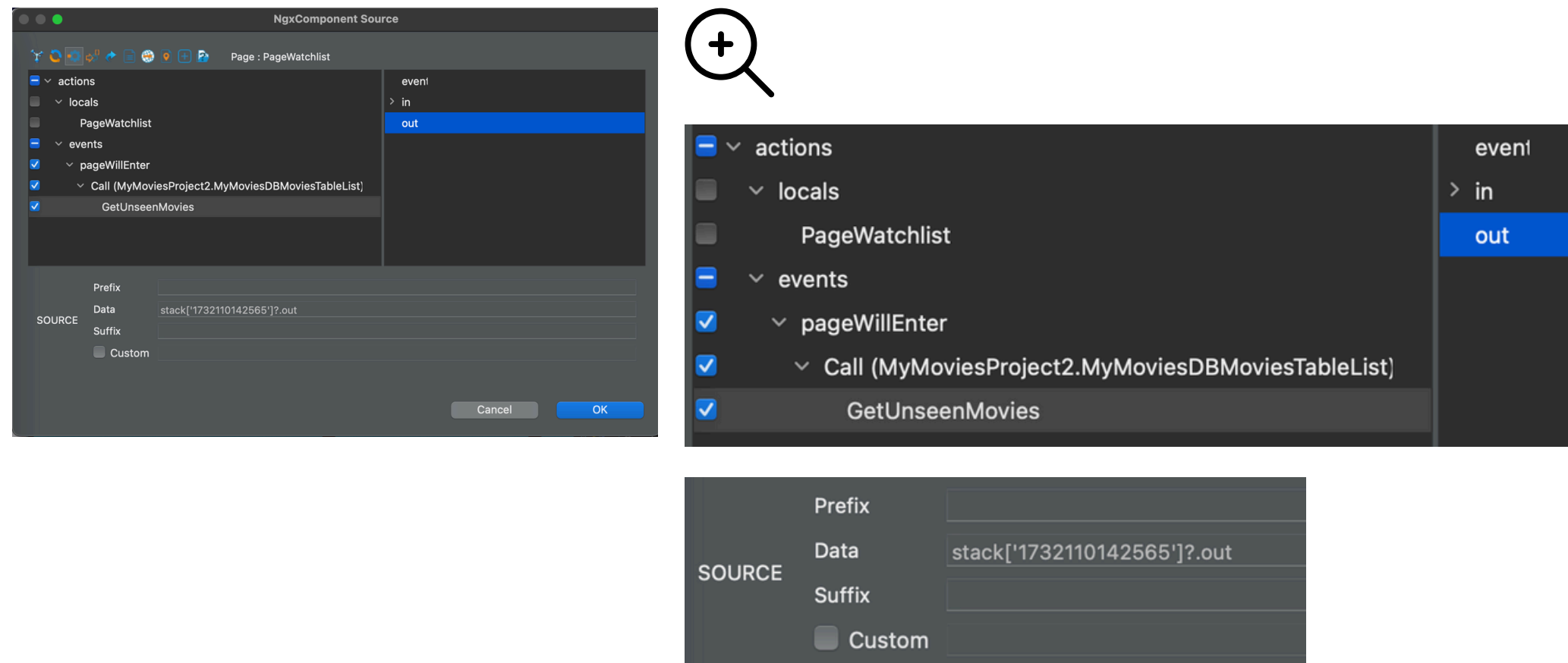


6.11 Display filtered data in segments

In the **NgxComponent** source panel,
click on the **CustomAction** icon to display the actions as sources.

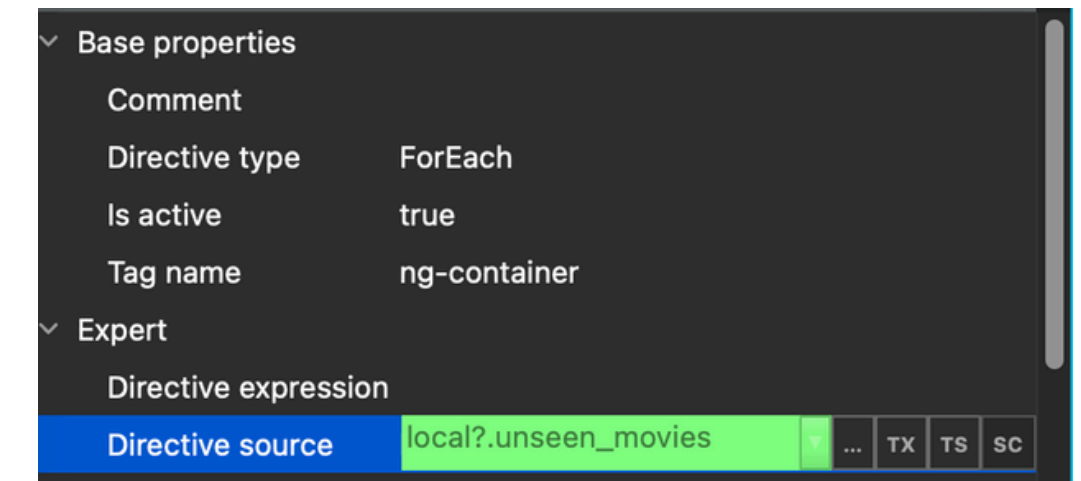
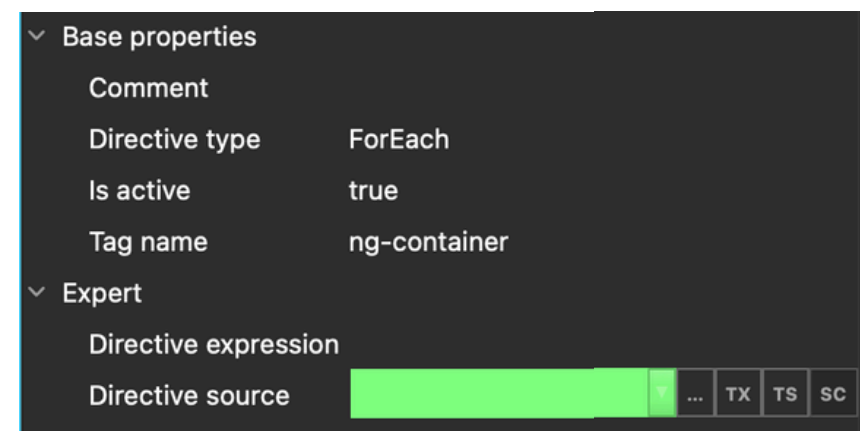
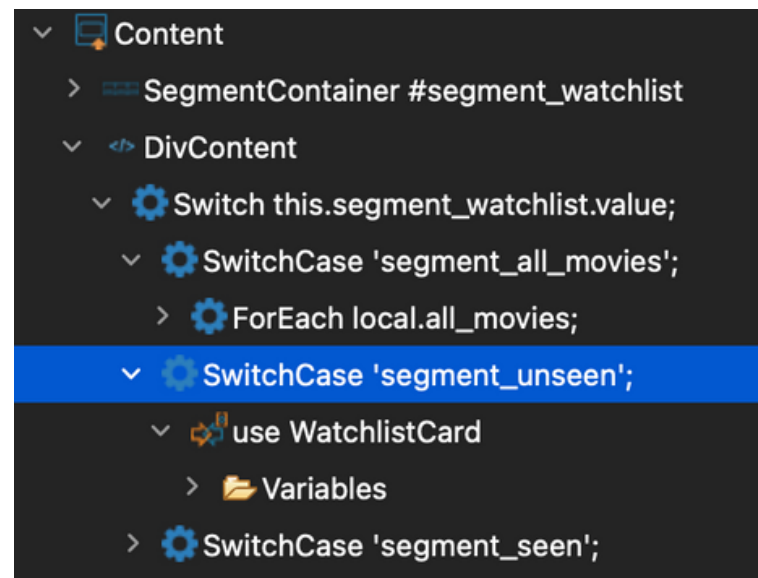


Click on the **CustomAction GetUnseenMovies** to select it as source.

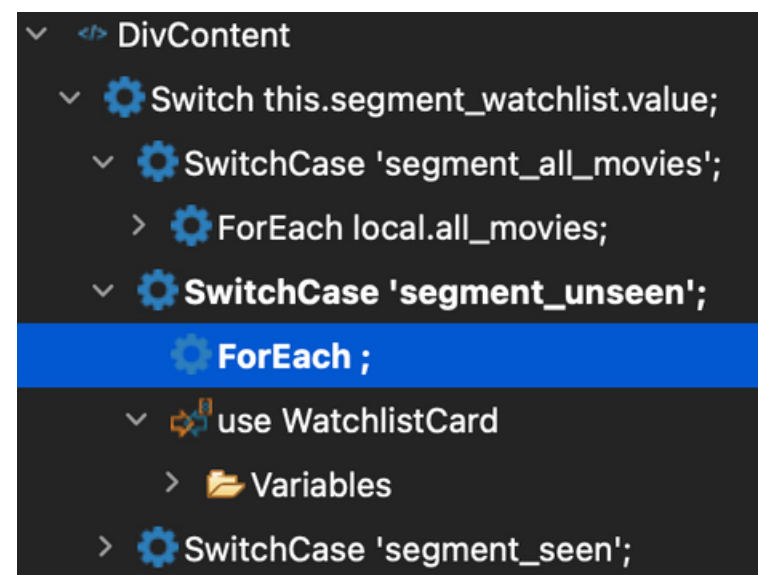


6.11 Display filtered data in segments

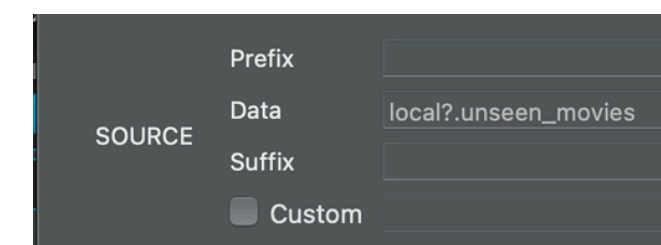
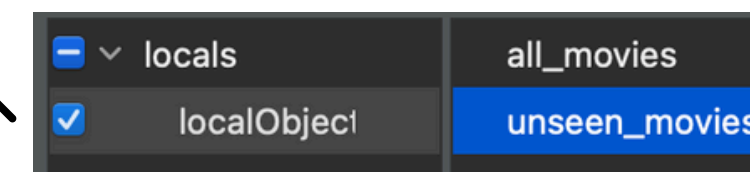
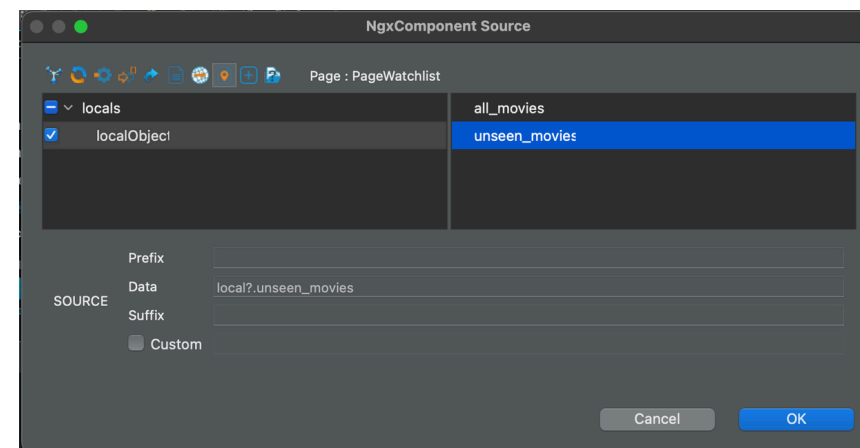
In the **ForEach Directive** 's properties, in the **Directive Source** property,
Click on ... or **SC** to open the
NgxComponent Source panel.



In the second **SwitchCase**
↓ **segment_unseen**,
add a **ForEach Directive**.



Click on **localObject** to display the local sources.
Then, click on **unseen_movies** to select the **unseen_movies SetLocal** as source.



6.11 Display filtered data in segments

In the **ForEach Directive** 's properties,
in the **ForEach item** property,
add **item** as value.

Base properties	
Comment	
Directive type	ForEach
Is active	true
Tag name	ng-container
Expert	
Directive expression	
Directive source	local.unseen_movies
ForEach index	
ForEach item	item



In the **SwitchCase**
segment_unseen,
move the **use WatchlistCard**
inside the **ForEach Directive**.

SwitchCase 'segment_unseen';
ForEach local.unseen_movies;
use WatchlistCard
Variables
movie=>"



Let's look at the **use WatchlistCard**
of the **SwitchCase** **segment_unseen**.
In the **movie useVariable**'s properties,
enter **item** as **value** of the **Var value** property.

@Component	
Binding	oneWayBinding
Var value	item



@Component	
Binding	oneWayBinding
Var value	item

SwitchCase 'segment_unseen';
ForEach local.unseen_movies;
use WatchlistCard
Variables
movie=>item

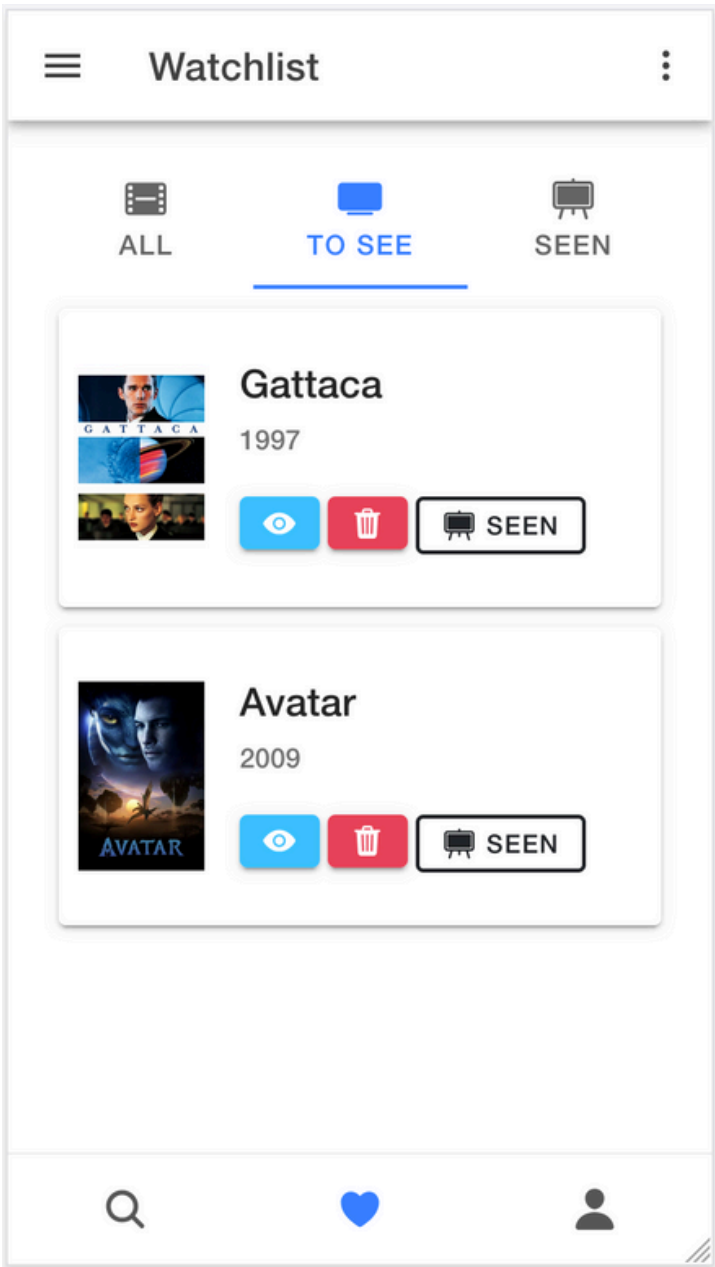


6.11 Display filtered data in segments

convertigo

The data appears in the **segment segment_unseen**.

For the third **segment segment_seen**, we could use the same way to retrieve data. But let's try something else.



In the database, we created a **filter** keeping only the **movies** where the **watched** field is **true**

WatchedMovies

1 Filter

Sort

Group

Share view

Color

Hide Fields

Height

	T title	overview	watched	# movie_id	T original_title	T img_url	T release_year	T genres	T release_date
1	Gremlins	When Billy Peltzer is given ...	✓	927	Gremlins	https://image.tmbd...	1984		1984-06-08
+									

+

WatchedMovies

1 Filter

Sort

Group

SI

	T title	overview	watched
1	Gremlins	When Billy Peltzer is given ...	✓
+			

Where watched is ☒ Selected

+ Add filter

+ Add filter group

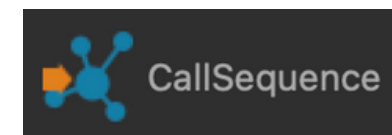
☐ all disabled



6.11 Display filtered data in segments

In PageWatchlist, in the **pageWillEnter** PageEvent

```
▼ pageWillEnter
  ▼ Call (MyMoviesProject2.MyMoviesDBMoviesTableList)
    all_movies = stack['1732107989220'].out.object.results
  ▼ GetUnseenMovies
    ▼ Variables
      > movies=MyMoviesProject2.MyMoviesDBMoviesTableList->object.results
    unseen_movies = stack['1732110142565'].out
```



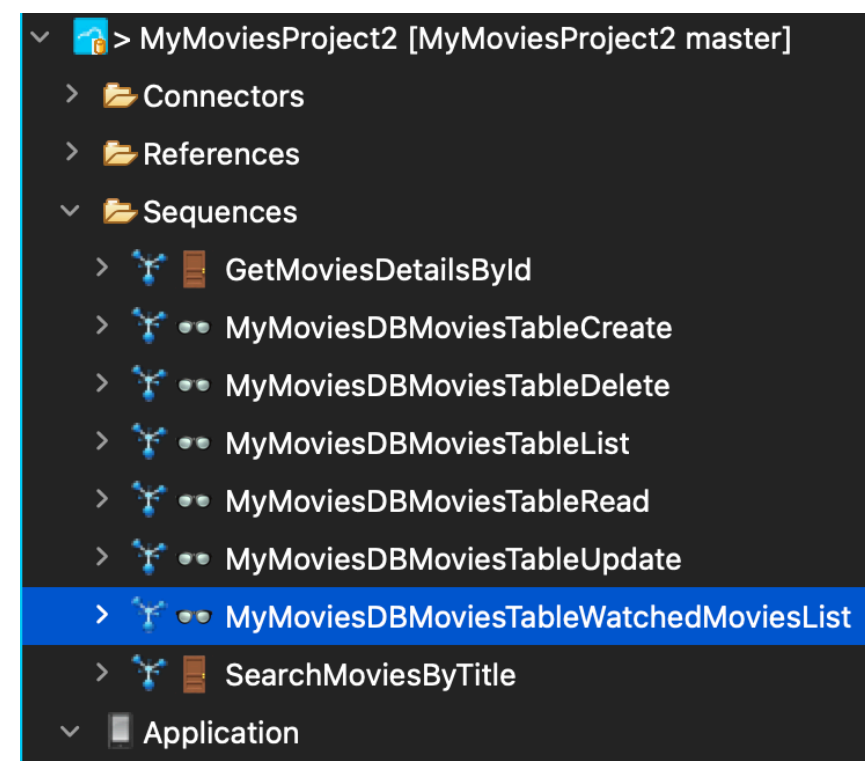
Add a **CallSequence** component after the first **CallSequence**.

```
▼ pageWillEnter
  ▼ Call (MyMoviesProject2.MyMoviesDBMoviesTableList)
    all_movies = stack['1732107989220'].out.object.results
  ▼ GetUnseenMovies
    ▼ Variables
      > movies=MyMoviesProject2.MyMoviesDBMoviesTableList->object.results
    unseen_movies = stack['1732110142565'].out
  CallSequence1
```

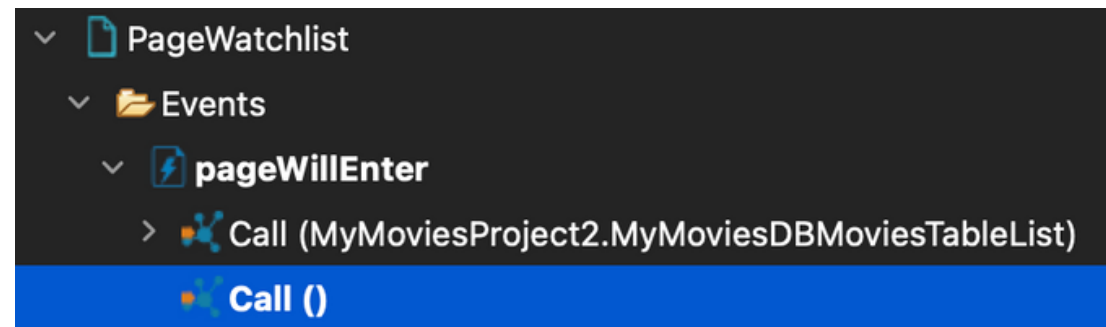


```
▼ pageWillEnter
  ▼ Call (MyMoviesProject2.MyMoviesDBMoviesTableList)
    all_movies = stack['1732107989220'].out.object.results
  ▼ GetUnseenMovies
    ▼ Variables
      > movies=MyMoviesProject2.MyMoviesDBMoviesTableList->object.results
    unseen_movies = stack['1732110142565'].out
  Call ()
```

This filtered data can be retrieved with the sequence **MyMoviesDBMoviesTableWatchedMoviesList**.



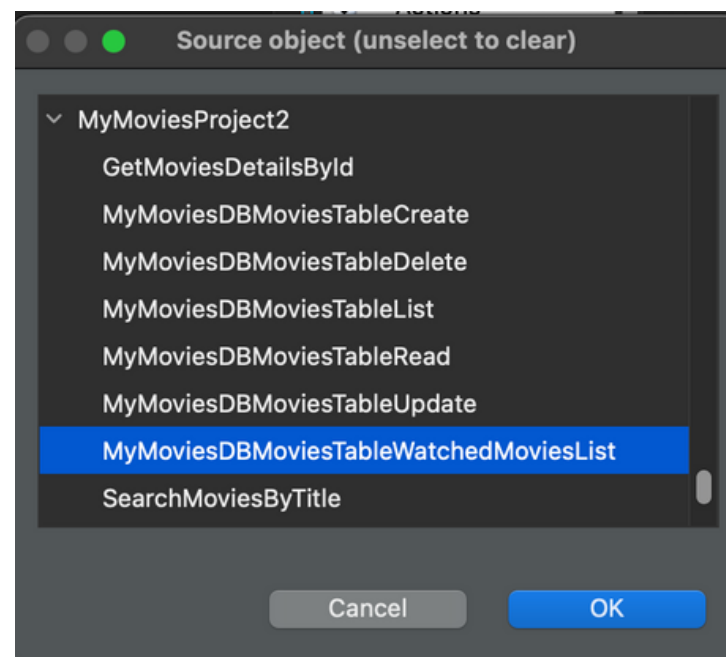
6.11 Display filtered data in segments



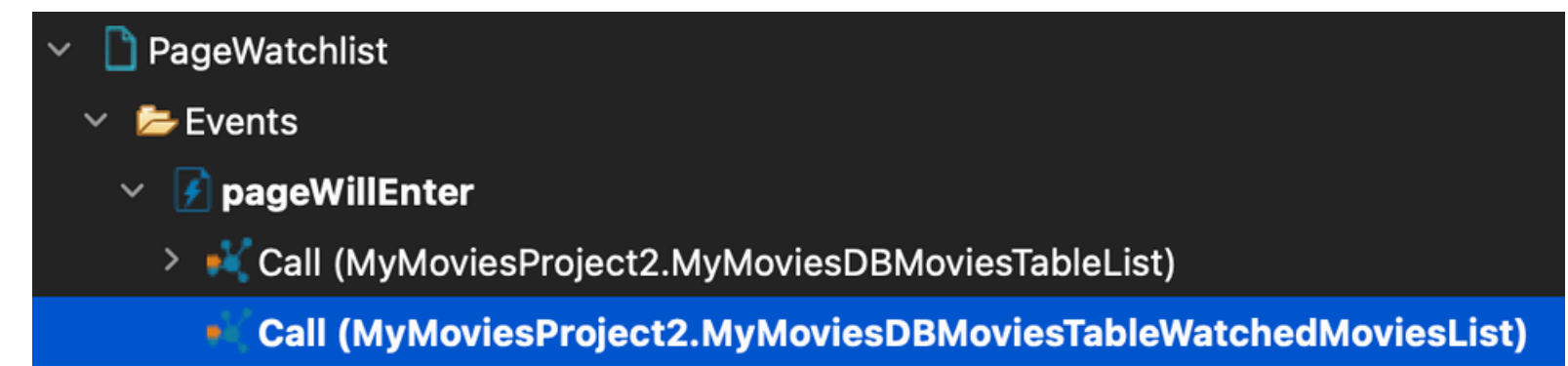
In the **CallSequence** component's properties in the **Requestable** property,



open the **NgxComponent Source** panel and select as source the sequence **MyMoviesDBMoviesTableWatchedMoviesList**.

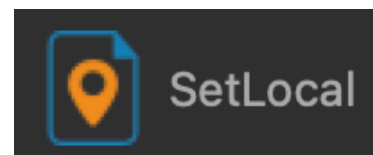


Property	Value
▼ @Properties	
Disable auto login	false
Disable loading controller	false
Local cache policy	not set
Local cache time to live	3000
Marker	
Requestable	MyMoviesProject2.MyMoviesDBMoviesTableWatchedMoviesList ...
Threshold	not set
Timeout	not set



6.11 Display filtered data in segments

```
▼ ⚡ pageWillEnter
  > 🔗 Call (MyMoviesProject2.MyMoviesDBMoviesTableList)
  🔗 Call (MyMoviesProject2.MyMoviesDBMoviesTableWatchedMoviesList)
```



Add a **SetLocal** component inside the **CallSequence**.

```
▼ ⚡ pageWillEnter
  > 🔗 Call (MyMoviesProject2.MyMoviesDBMoviesTableList)
  ▼ 🔗 Call (MyMoviesProject2.MyMoviesDBMoviesTableWatchedMoviesList)
    📍 myProperty = ?
```



```
▼ @Properties
  Property      myProperty
  Value
```

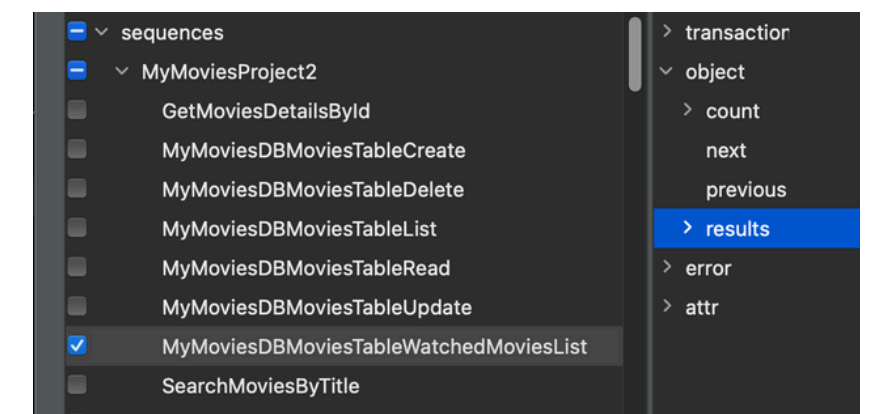
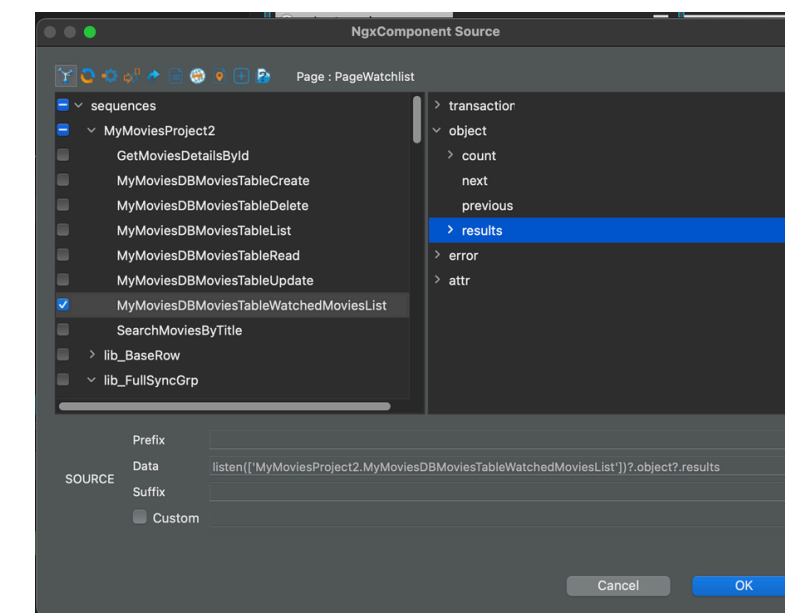


Rename it **seen_movies**.

```
▼ @Properties
  Property      seen_movies ▼ ... TX TS SC
  Value
```



Open the **NgxComponent** source panel, click on the **CallSequence MyMoviesDBMoviesTableList** and select **results** in **object** as source.



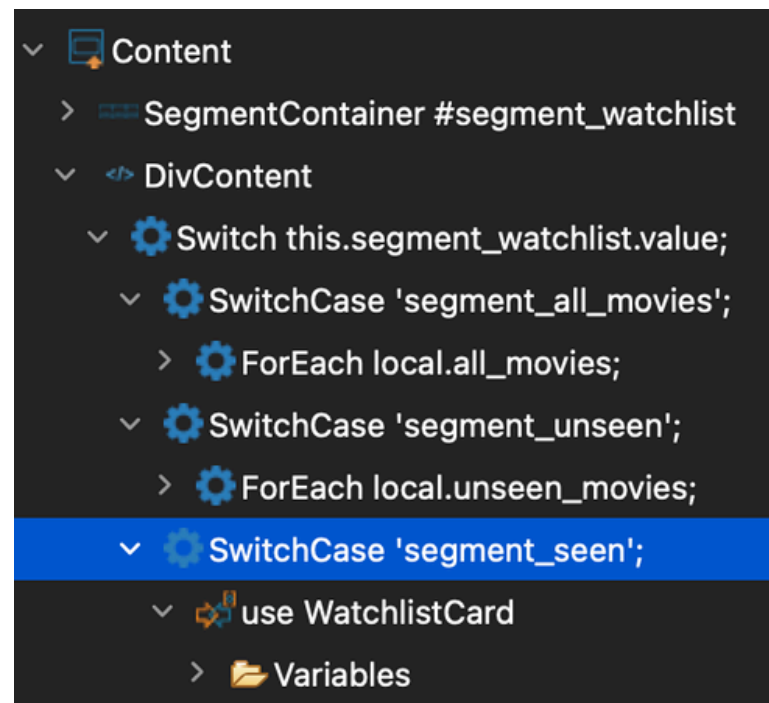
```
Prefix
Data    listen(['MyMoviesProject2.MyMoviesDBMoviesTableWatchedMoviesList'])?object?.results
Suffix
```



```
▼ ⚡ pageWillEnter
  > 🔗 Call (MyMoviesProject2.MyMoviesDBMoviesTableList)
  ▼ 🔗 Call (MyMoviesProject2.MyMoviesDBMoviesTableWatchedMoviesList)
    📍 seen_movies = MyMoviesProject2.MyMoviesDBMoviesTableWatchedMoviesList->object.results ...
```



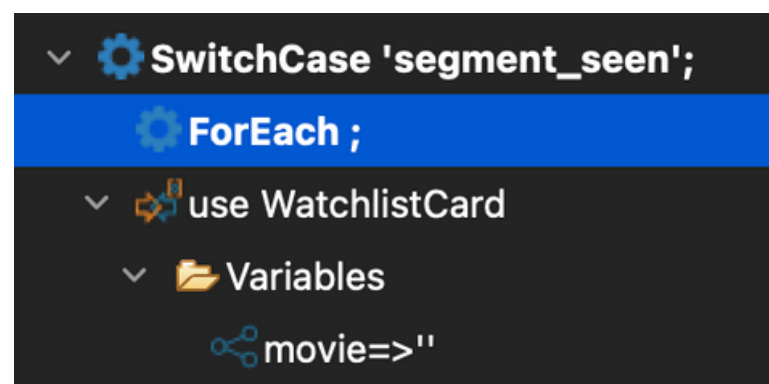
6.11 Display filtered data in segments



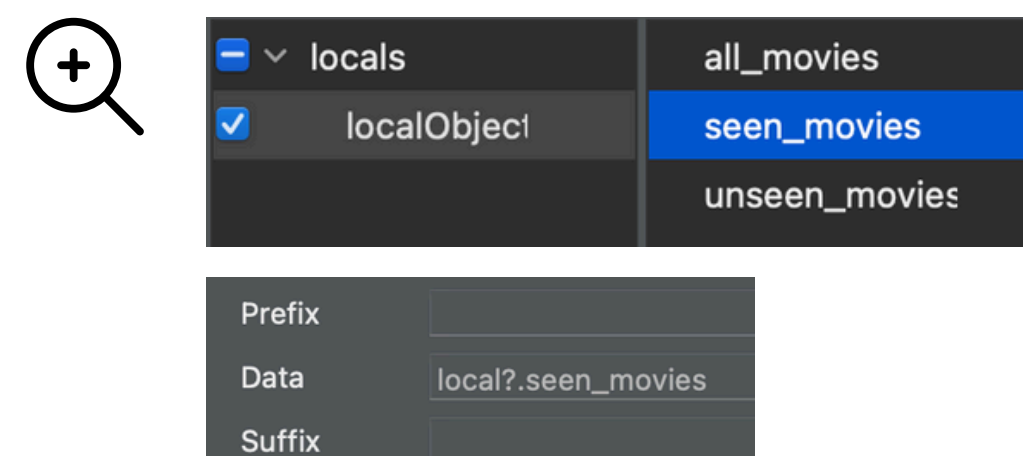
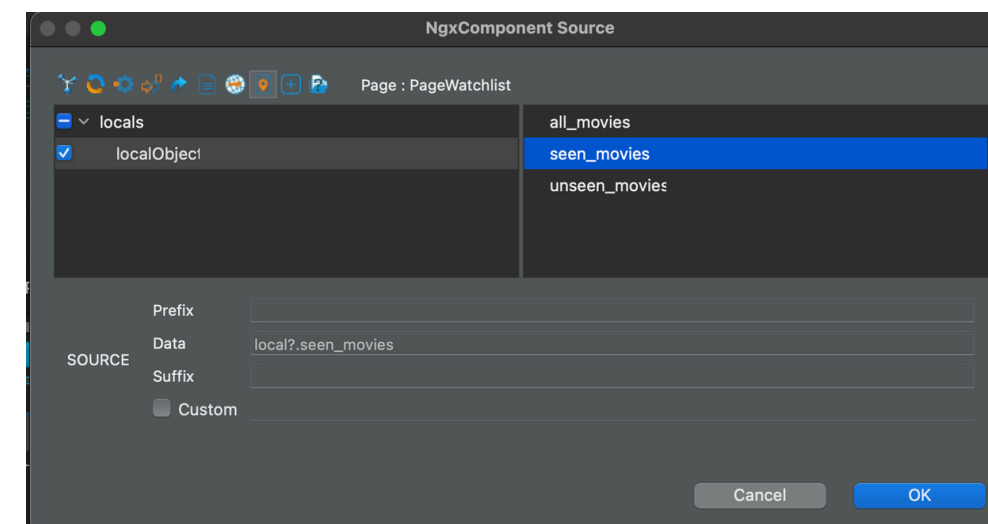
In the third **SwitchCase**



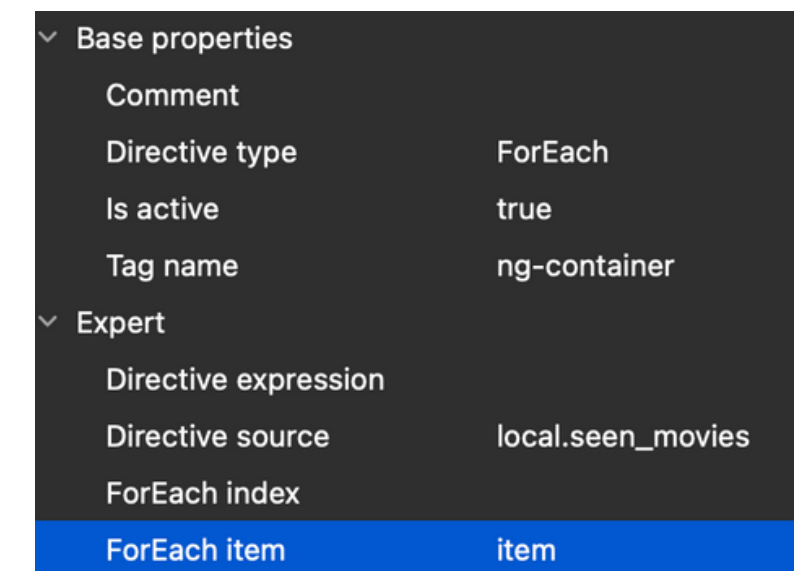
segment_seen,
add a **ForEach Directive**.



In the **ForEach Directive** 's properties,
in the **Directive Source** property,
open the **NgxComponent Source** panel
and select the **seen_movies SetLocal**
as source.

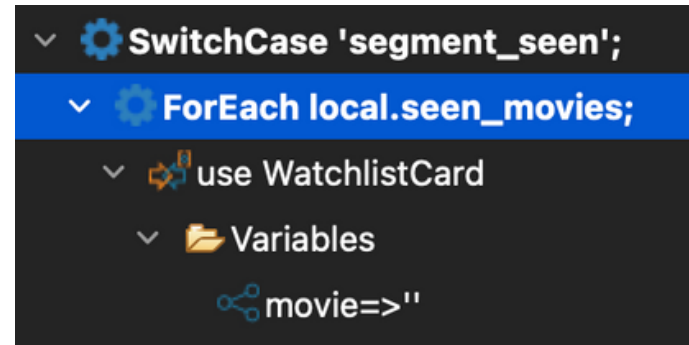


In the directive 's properties,
in the **ForEach item** property,
add **item** as value.



6.11 Display filtered data in segments

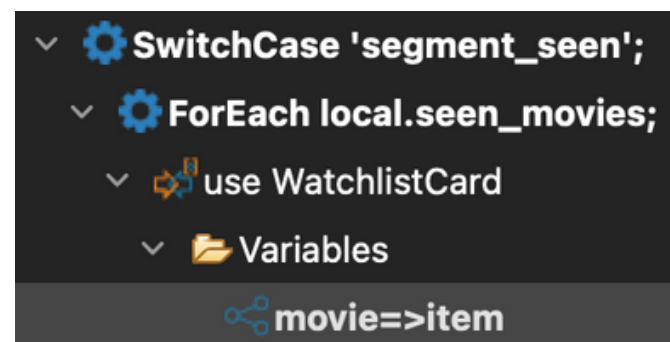
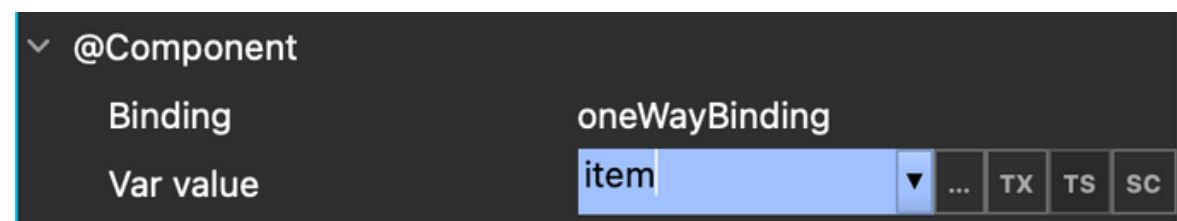
In the **SwitchCase** `segment_unseen`,
move the **use WatchlistCard** inside the **ForEach Directive**.



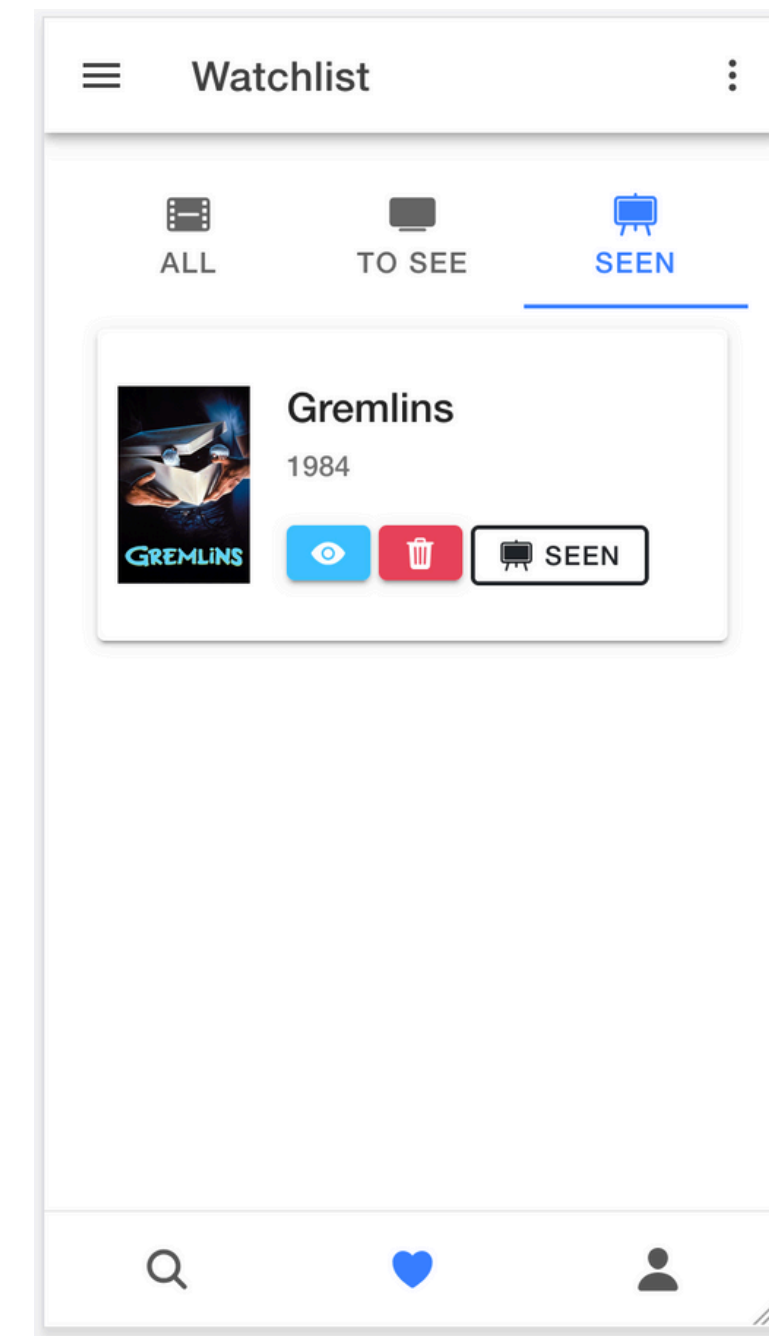
In the **use WatchlistCard** of the **SwitchCase** `segment_seen`,



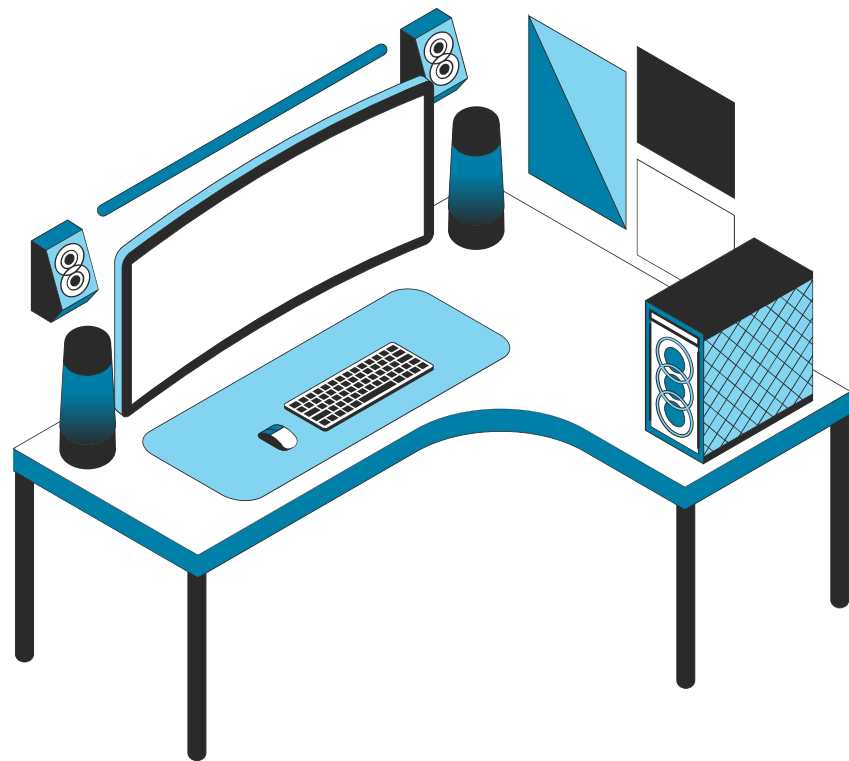
in the **movie useVariable**'s properties,
enter **item** as **value** of the **Var value** property.



The data appears
in the **segment** `segment_seen`.



7 – Shared components Events



7.1 SharedComponent Event components

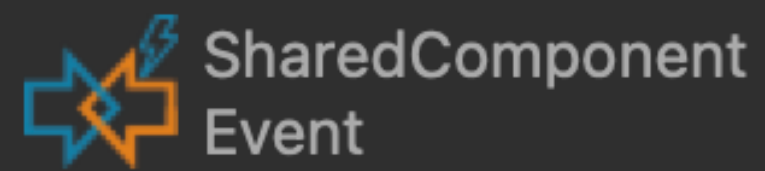
7.2 EmitEvent & EmitValue components

7.3 Add a button as SharedComponent

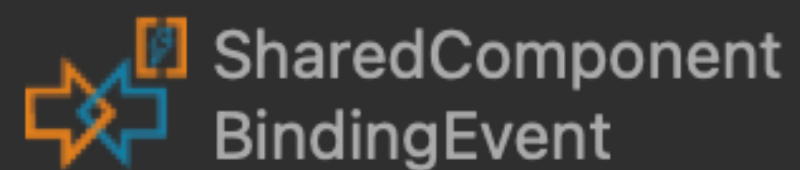
7.4 Use variables in SharedComponents

7.5 Use events in SharedComponents

7.1 SharedComponent Event components



The **SharedComponent Event** component is added under a **SharedComponent**, and is used to add and chain any type of **action** such as **CallSequence**, or **CustomAction**.



The **SharedComponent BindingEvent** component is used to define an **event** a shared component may emit.

This component stands for Angular's event binding.



7.2 EmitEvent & EmitValue components

→ (...) EmitEvent

The **EmitEvent** action component is used to **emit an event** from a shared component.

The event may be handled by an existing **Event component** added to a **Use Shared component**.

→ (...) EmitValue

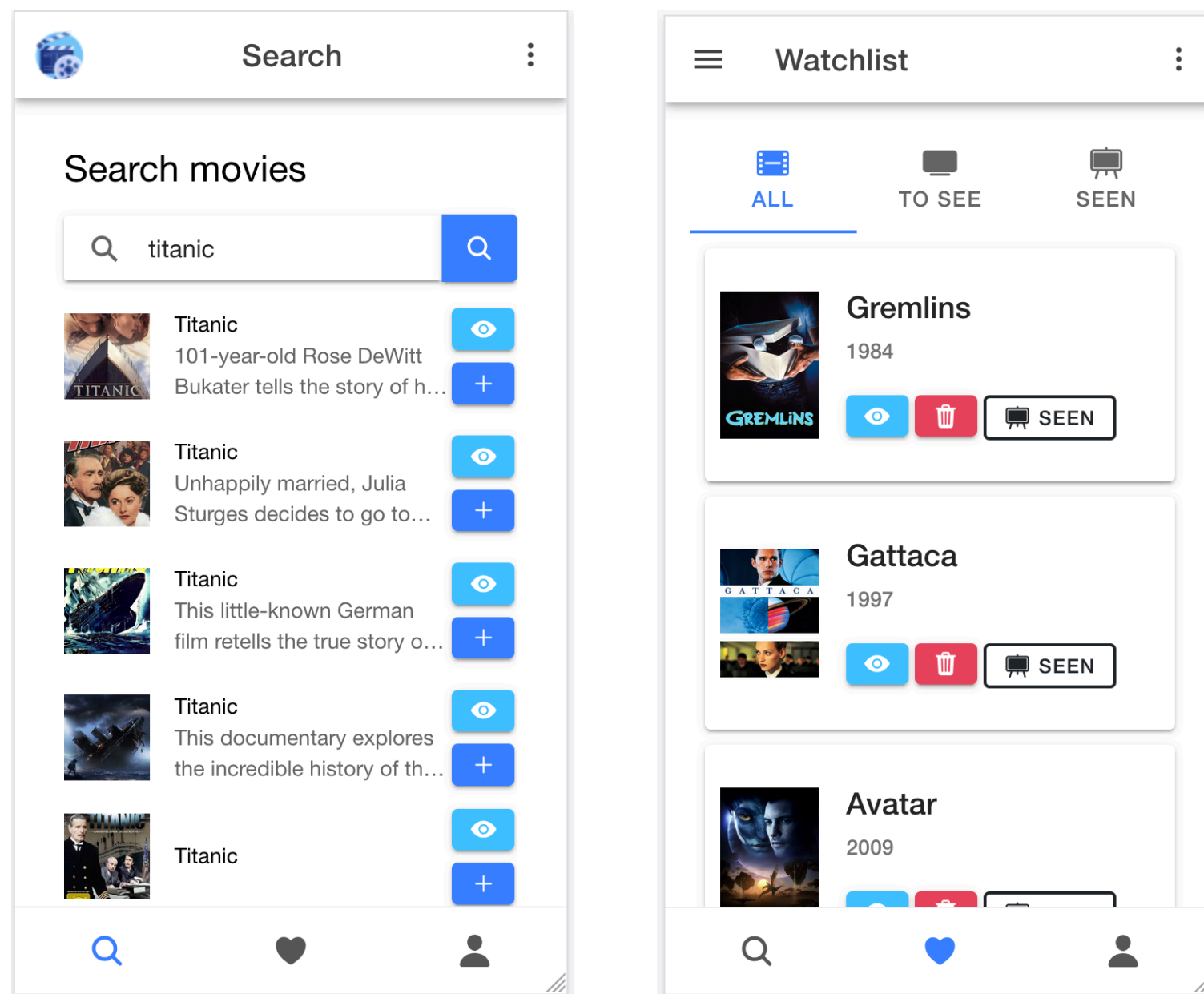
The **EmitValue** action component is used to **emit an input variable value** from a shared component.



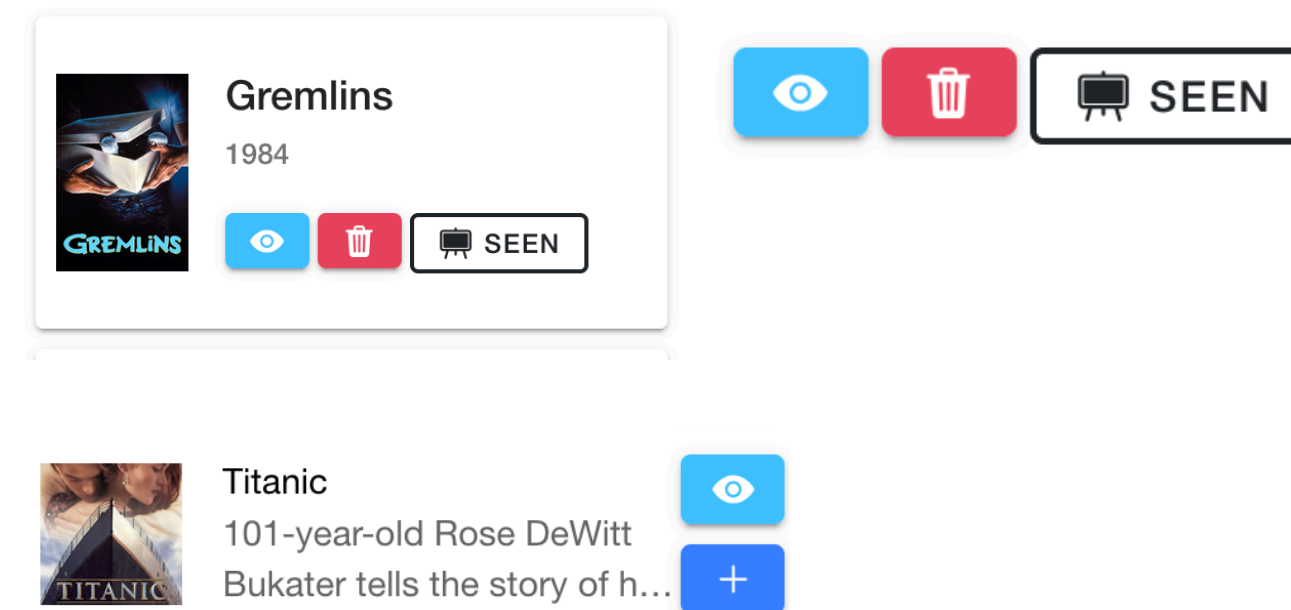
7.3 Add a button as SharedComponent

As for now, we display data on **pages Search** and **Watchlist**.

We still need to add actions to the different buttons in our pages layouts.



We need to create events and actions for the buttons in **WatchlistCard** shared component.

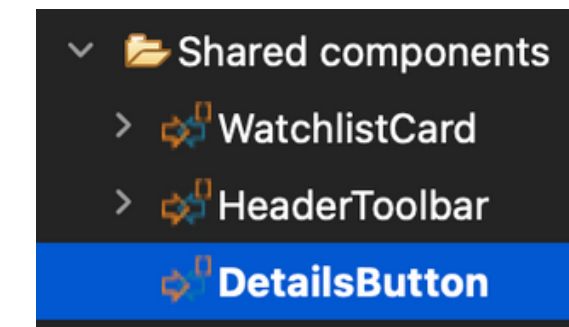
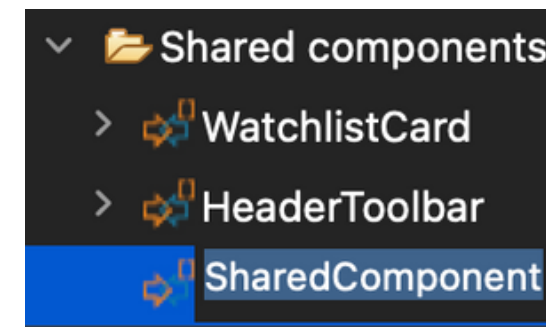
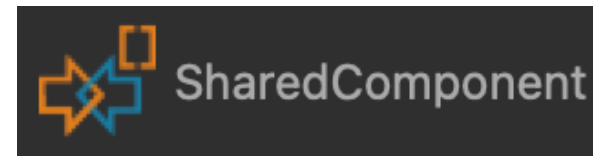
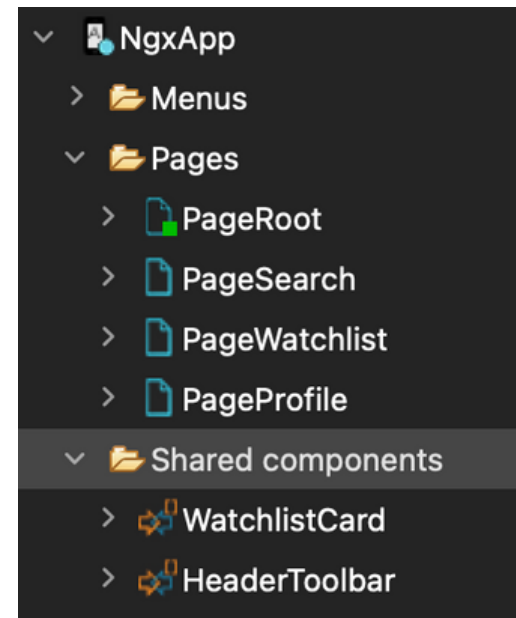


The same button appears in two different places.
We are now going to create a shared component to demonstrate how to use events in shared components.

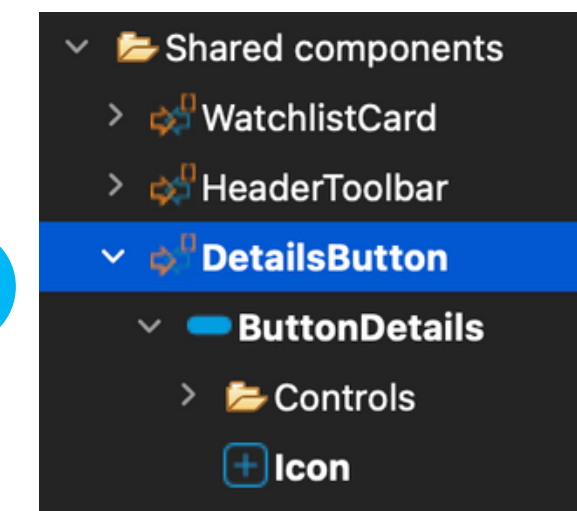
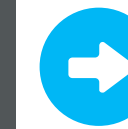
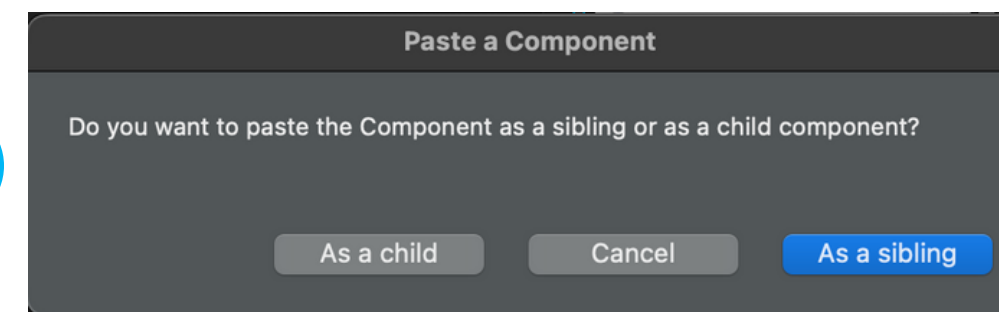
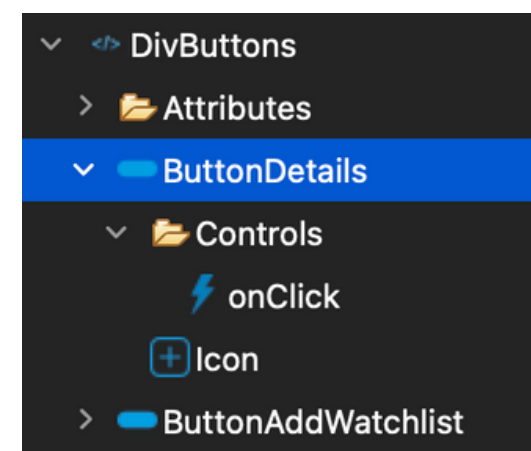
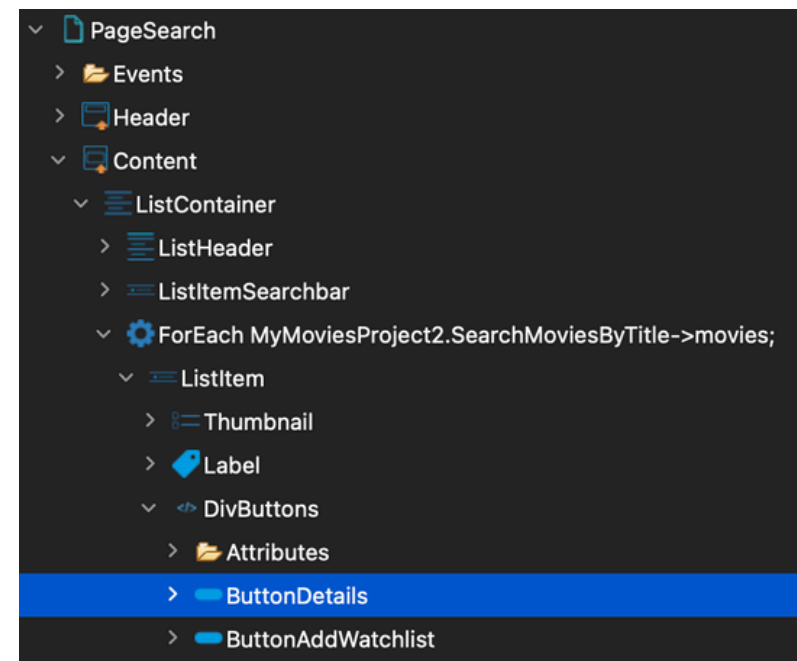


7.3 Add a button as SharedComponent

In the **Shared components** folder, add a **SharedComponent** and rename it **DetailsButton**.

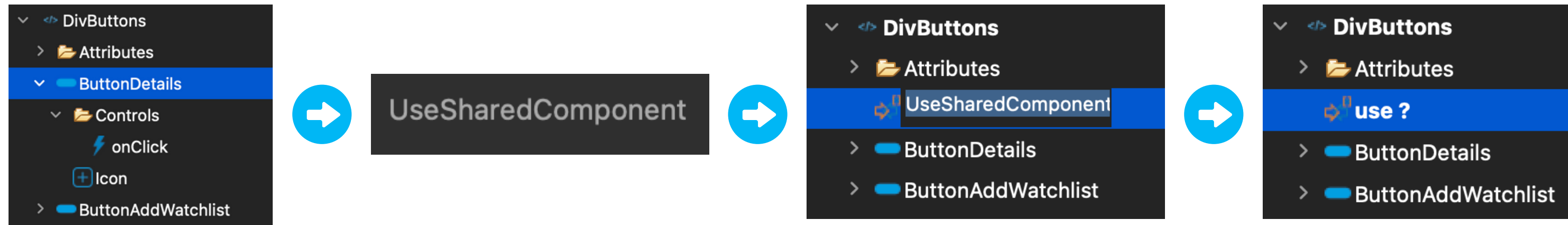


In the **ListItem** of **PageSearch**, copy the **ButtonDetails** and paste it in **DetailsButton SharedComponent**.

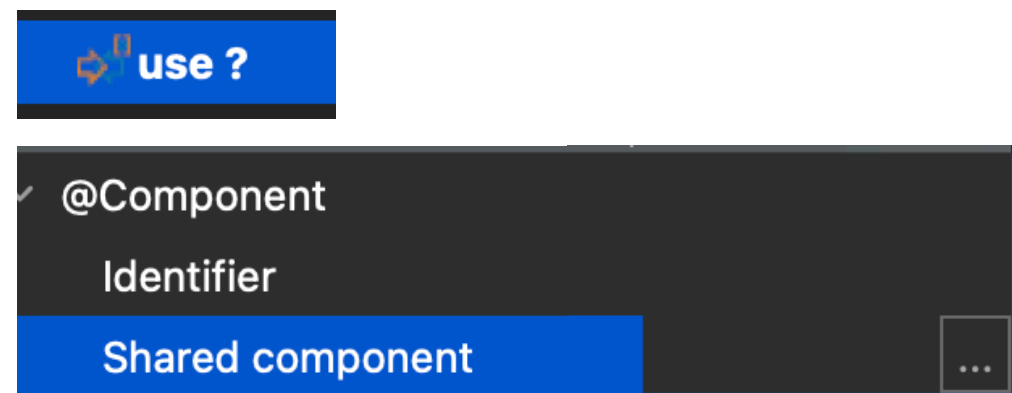


7.3 Add a button as SharedComponent

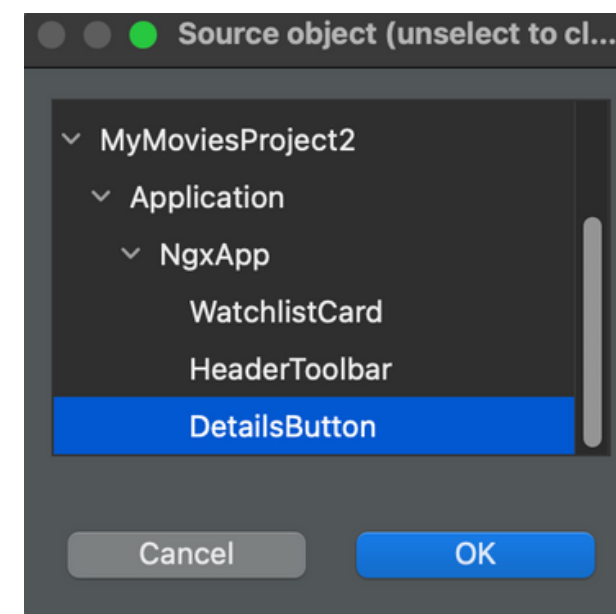
In the **ListItem** of **PageSearch**, before the **ButtonDetails** in **DivButtons**, add a **UseSharedComponent**.



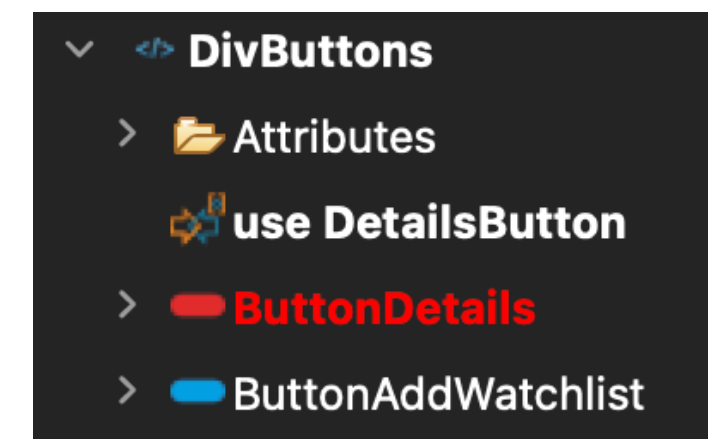
In the **UseSharedComponent**'s properties, in the **Shared component property**, click on ... to open the **Source object panel**



In the **Source object panel**, click on **DetailsButton** to select the **DetailsButton SharedComponent** as Source.

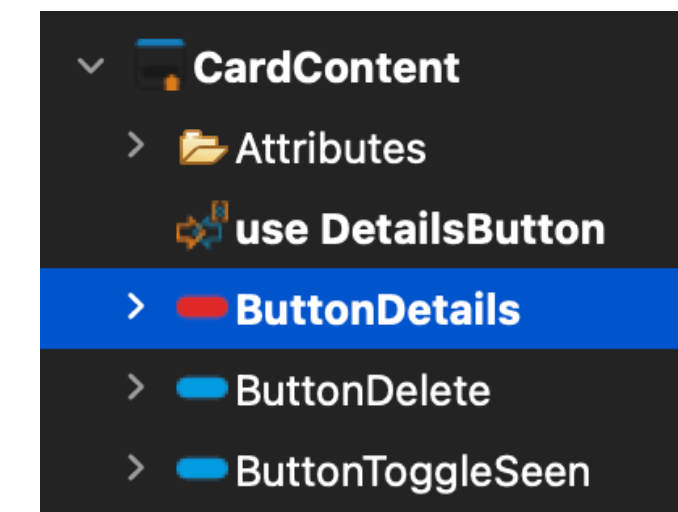
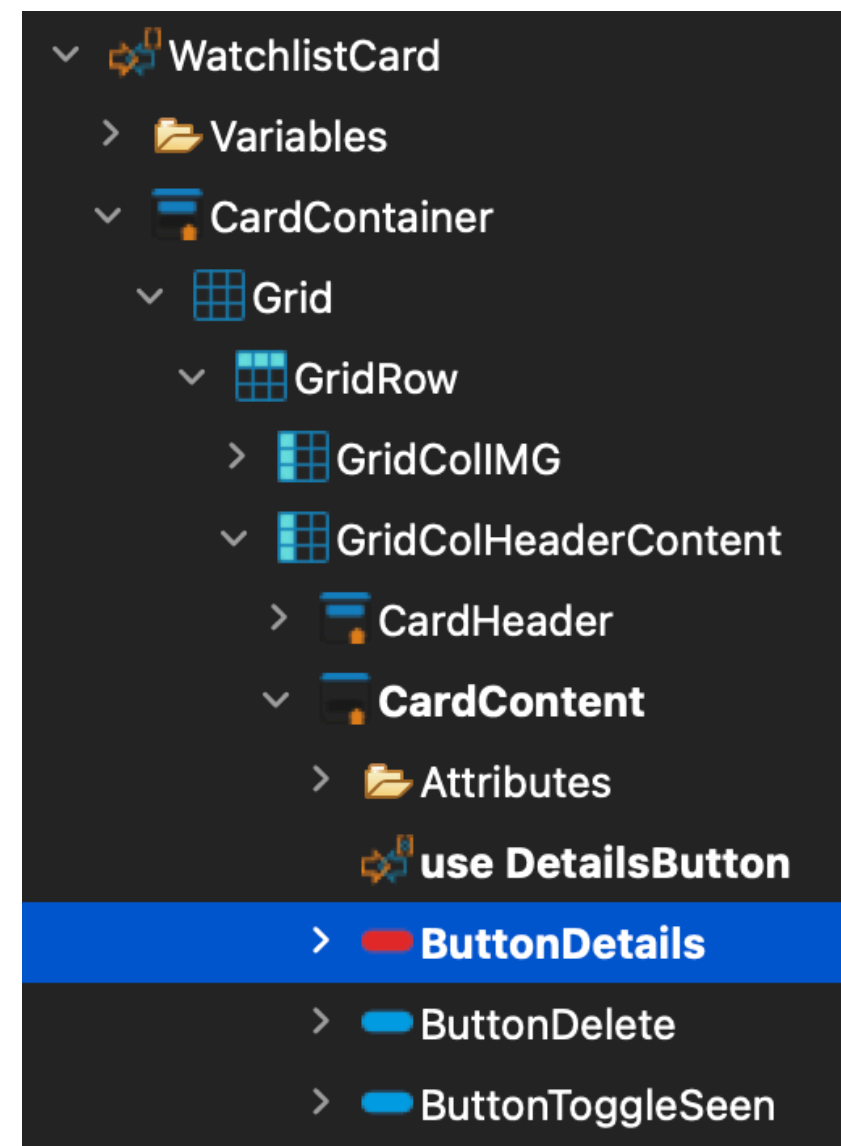
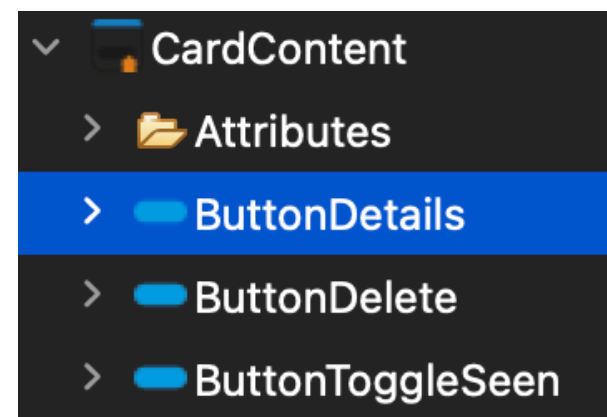
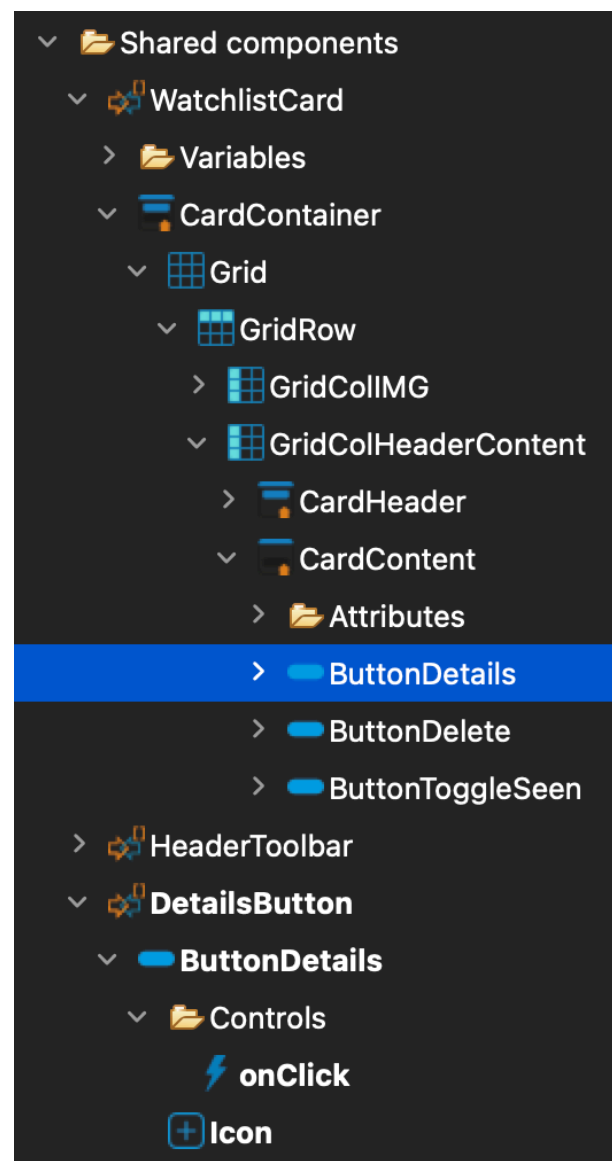


Then, disable **ButtonDetails**.



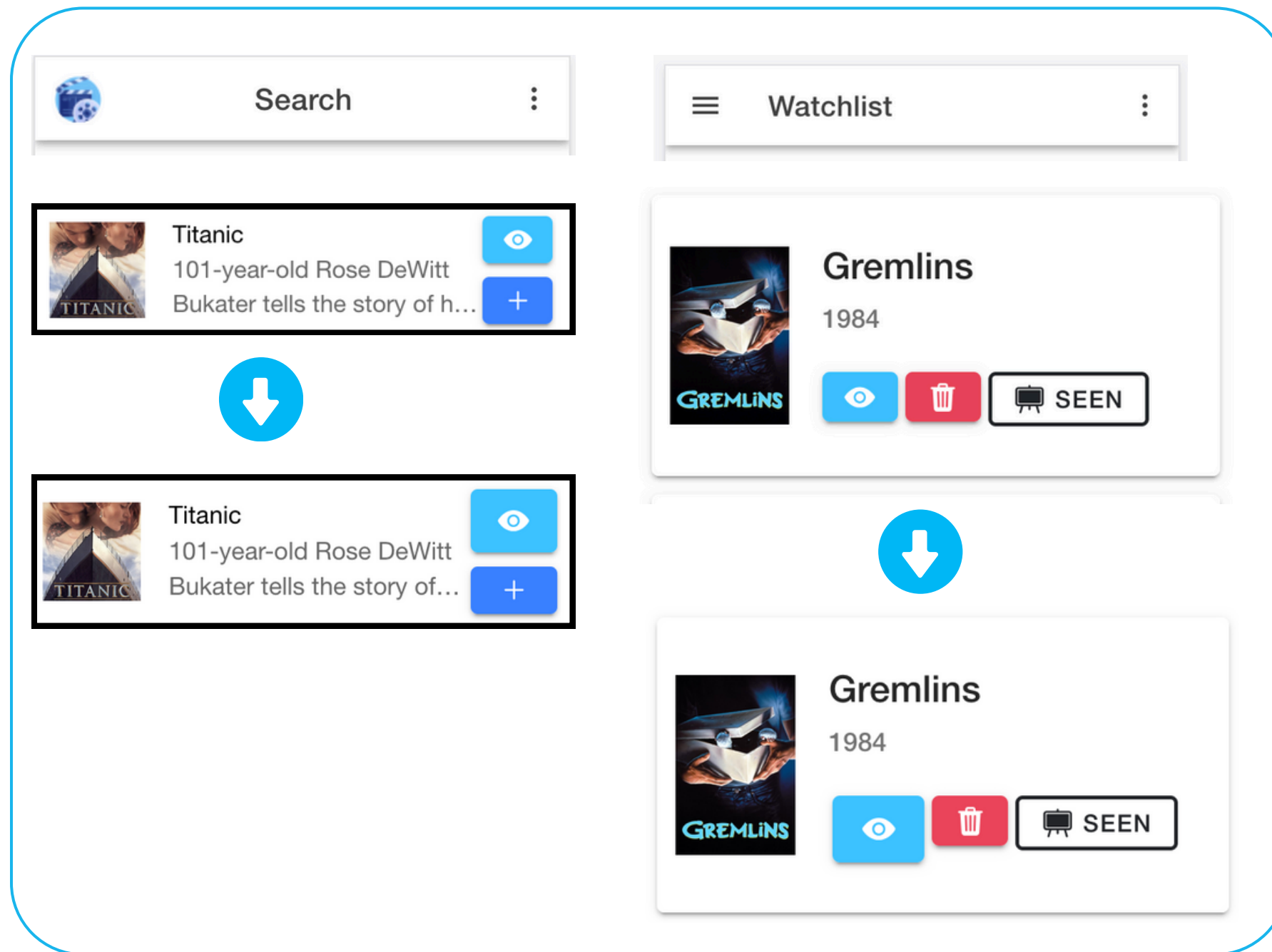
7.3 Add a button as SharedComponent

In the **Shared components** folder, in the **WatchlistCard** Shared component, proceed the same way to add **Use DetailsButton** and disable **ButtonDetails**.



7.4 Use variables in SharedComponents

In the process, the button size has changed in both pages.

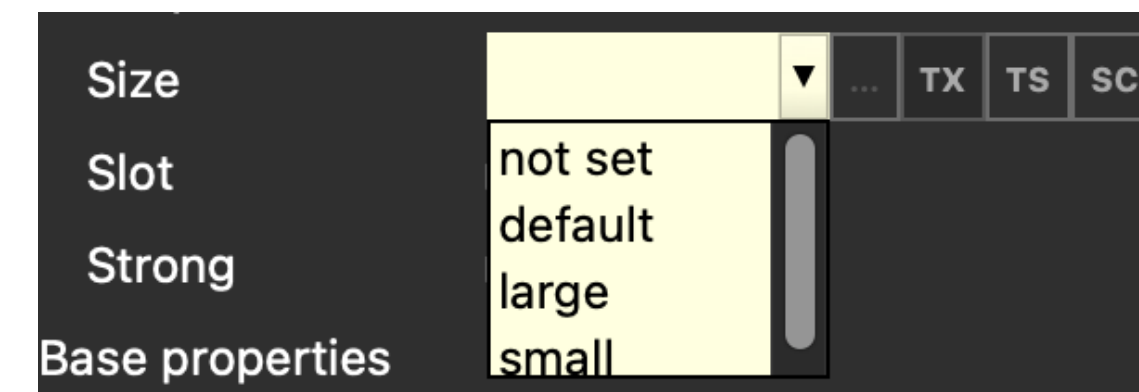


In a button, the **button Size property** can be :

not set, default, large, small.

When **not set** is selected,

the button size adapts to the container.



When encapsulated in a shared component, the button size reacts differently.

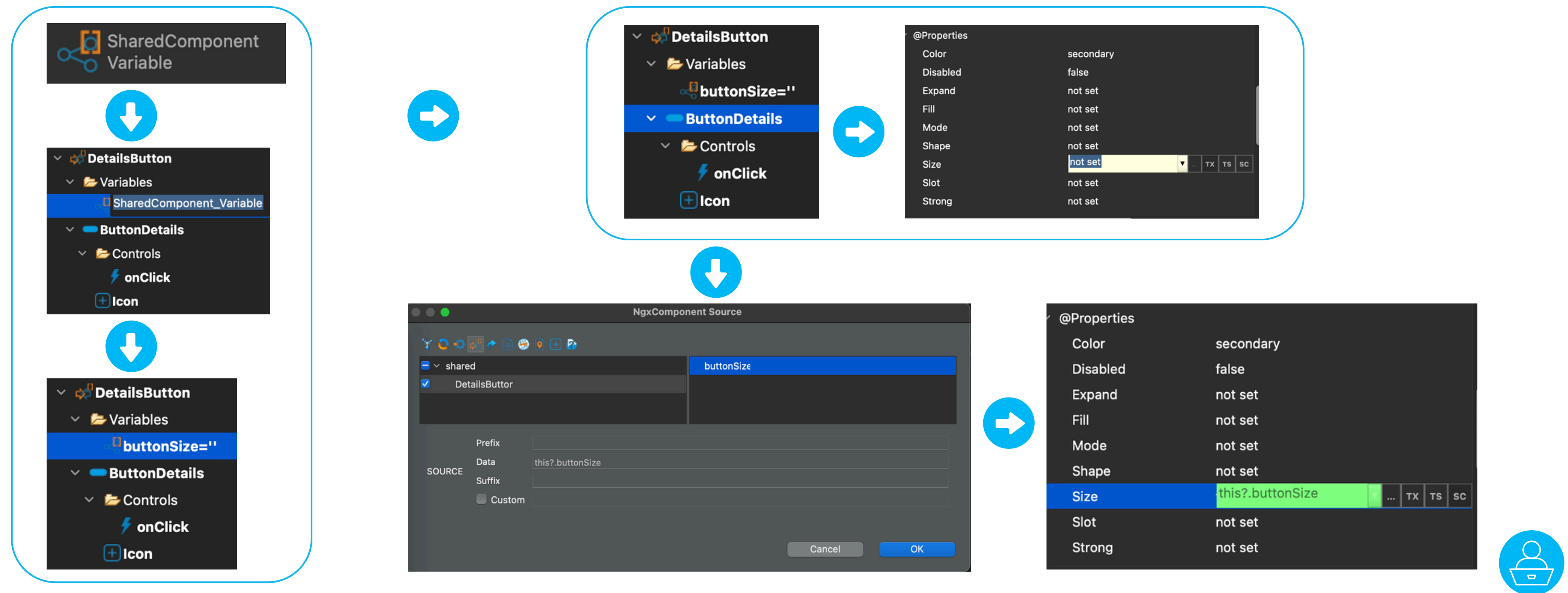
To adapt the size of our button shared component, we need to add a **variable linked to the size property**.



7.4 Use variables in SharedComponents

In the **DetailsButton** SharedComponent, add a **SharedComponent** variable, and rename it **buttonSize**.

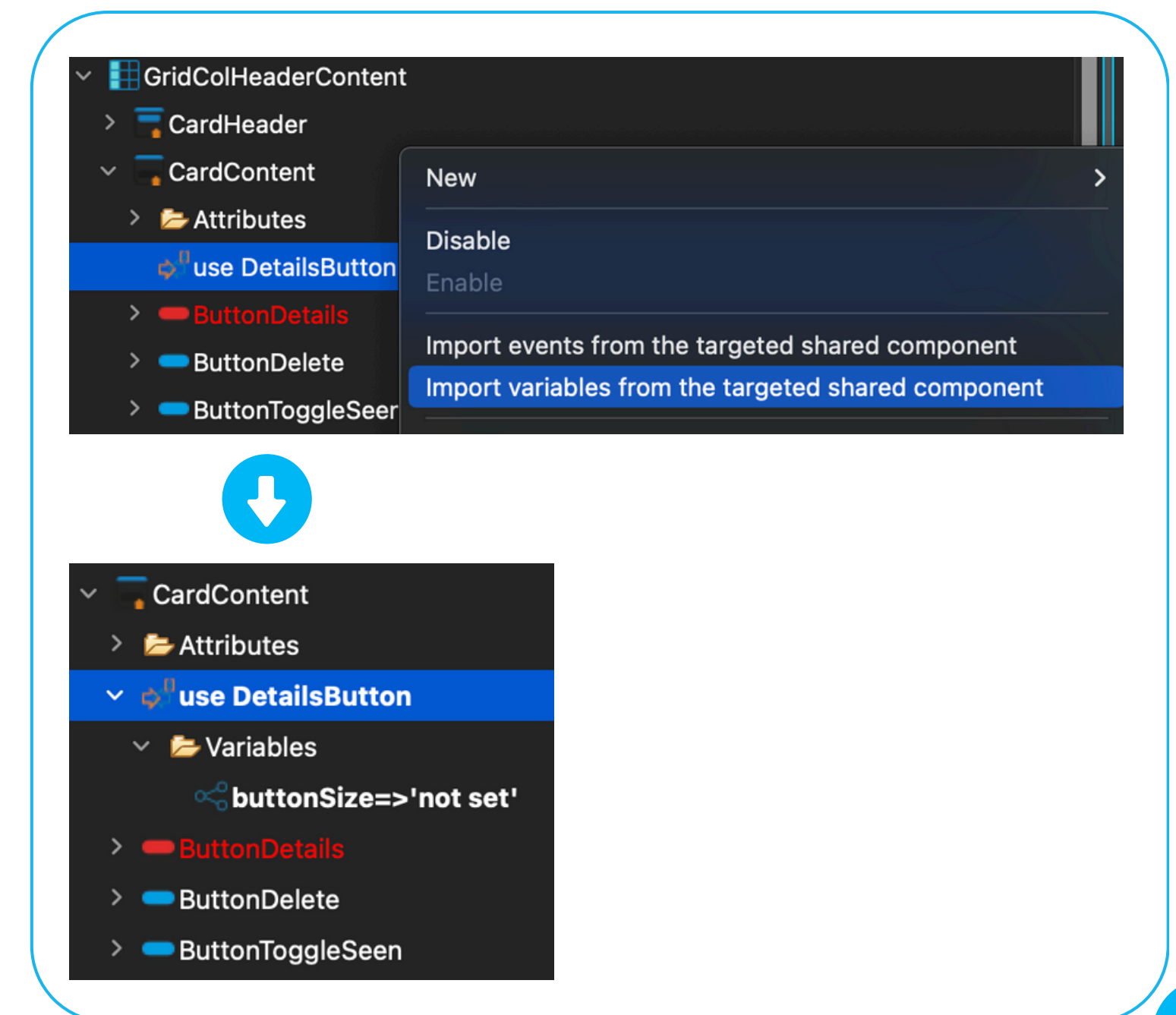
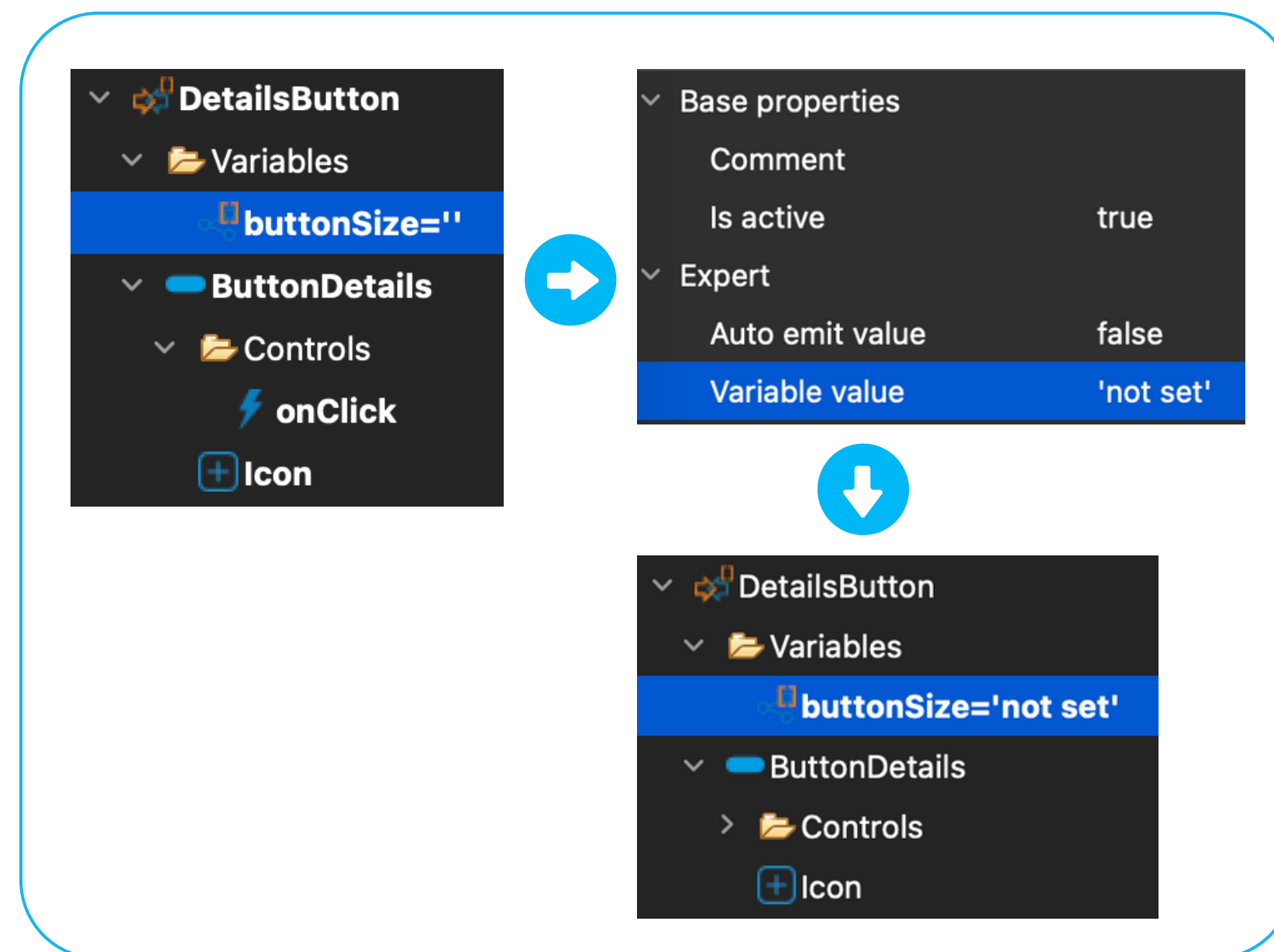
Inside the **SharedComponent**, in **ButtonDetails**'s properties, select the **SharedComponent** variable **buttonSize** as **source** for the **Size** property.



7.4 Use variables in SharedComponents

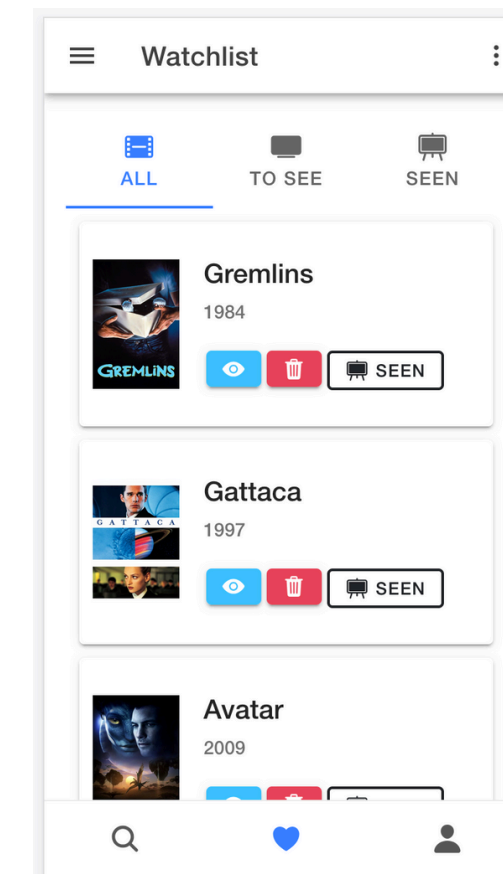
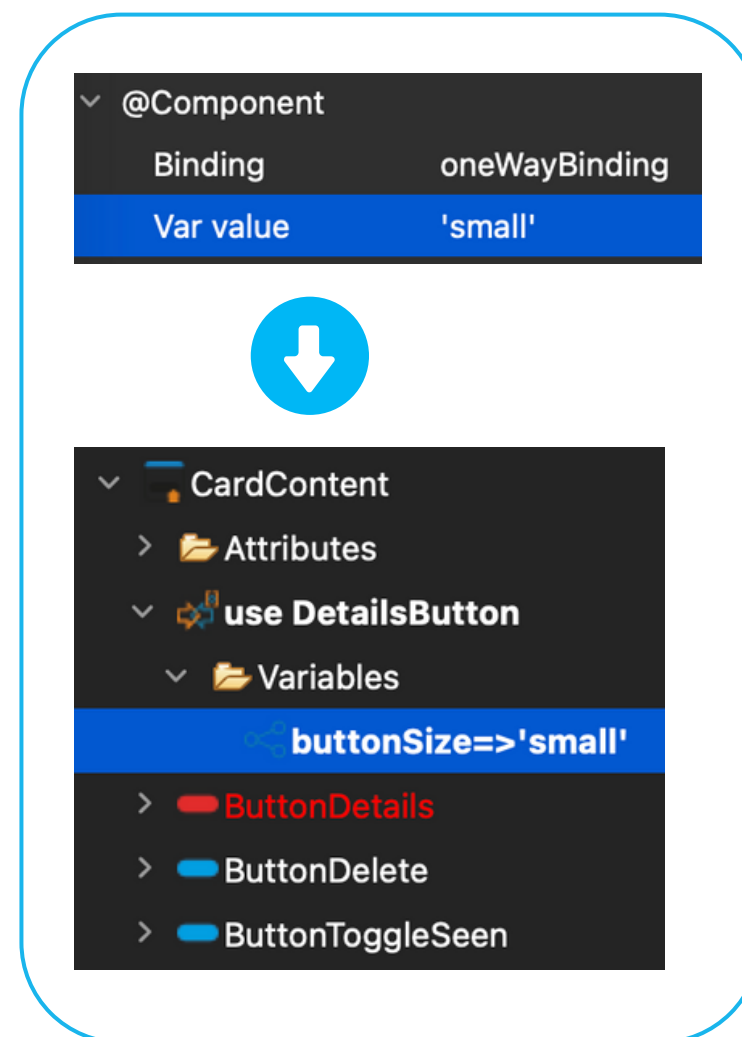
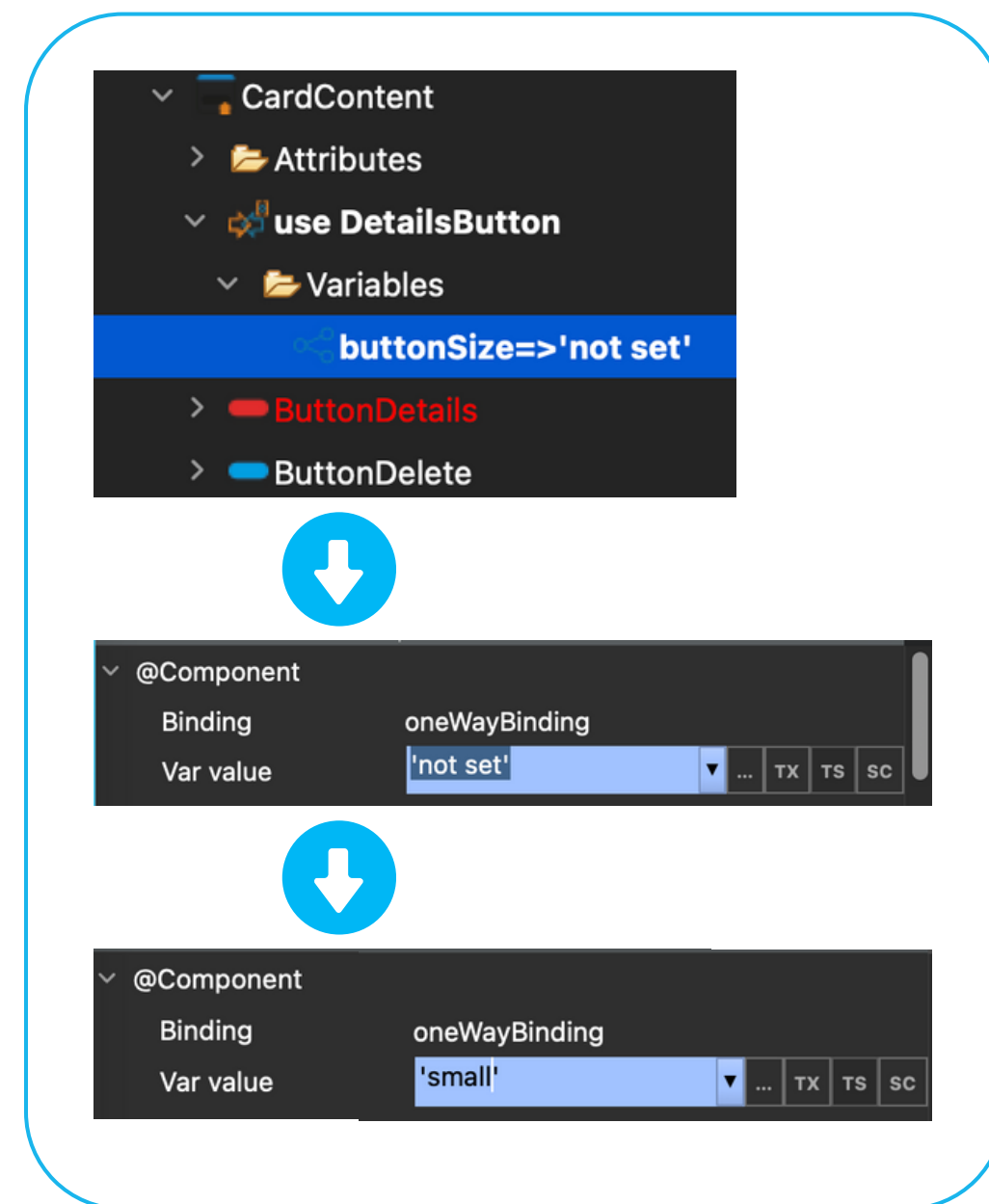
In the **SharedComponent** variable **buttonSize**'s properties, select **'not set'** as **default value** for the **Variable value** property.

Inside **WatchlistCard** **Shared** component, in the **use DetailsButton**, import the **variable**.

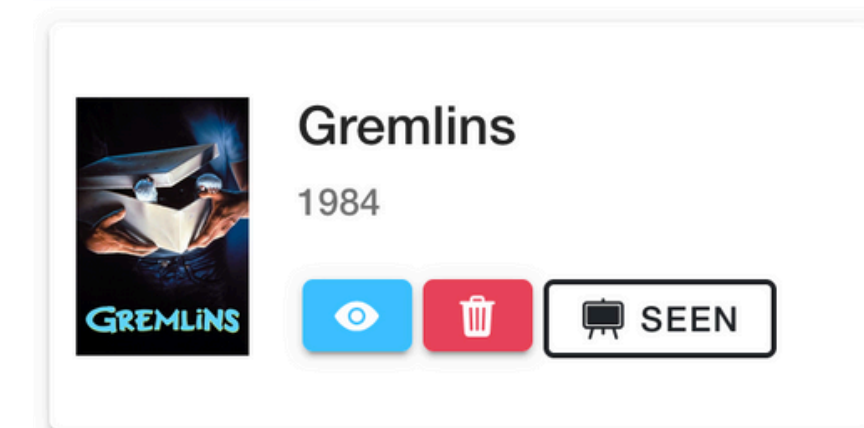


7.4 Use variables in SharedComponents

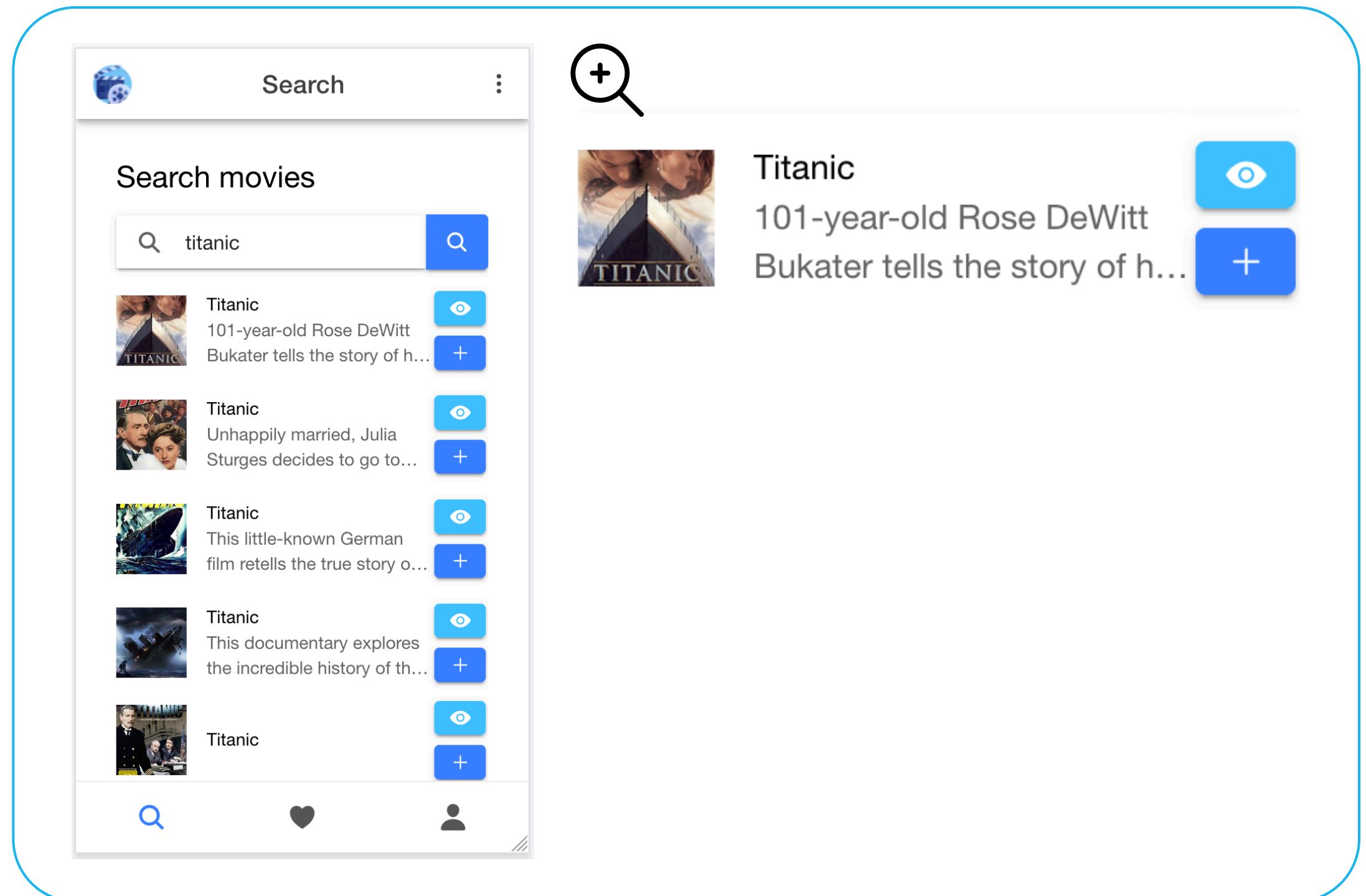
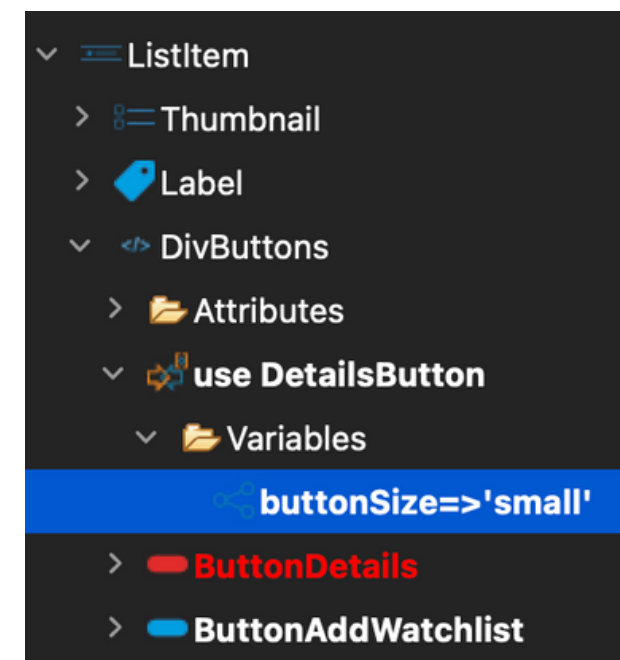
Inside **WatchlistCard** Shared component, in the **use DetailsButton**, click on the **variable buttonSize** and change its **Var value** to **'small'** in **TS scope**.



 The button has now the correct size.

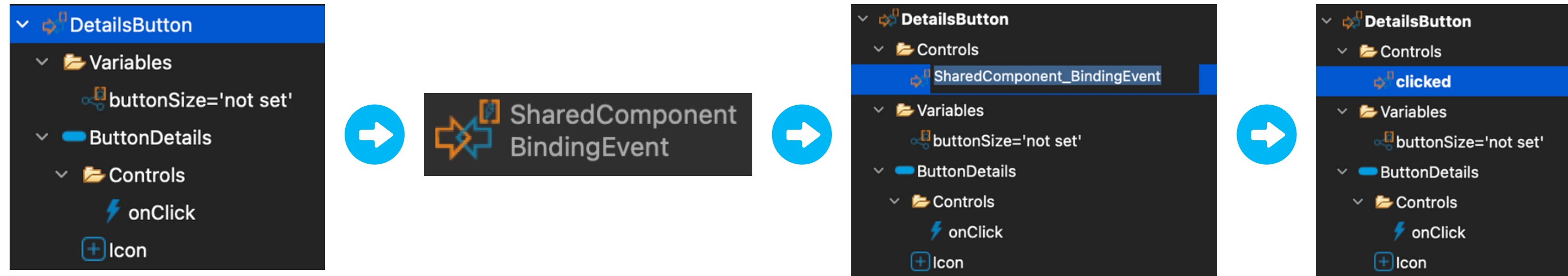


- ▼  ForEach MyMoviesProject2.SearchMoviesByTitle->movies;
- ▼  ListItem
 - >  Thumbnail
 - >  Label
 - ▼  DivButtons
 - >  Attributes
 -  **use DetailsButton**
 - >  **ButtonDetails**
 - >  **ButtonAddWatchlist**

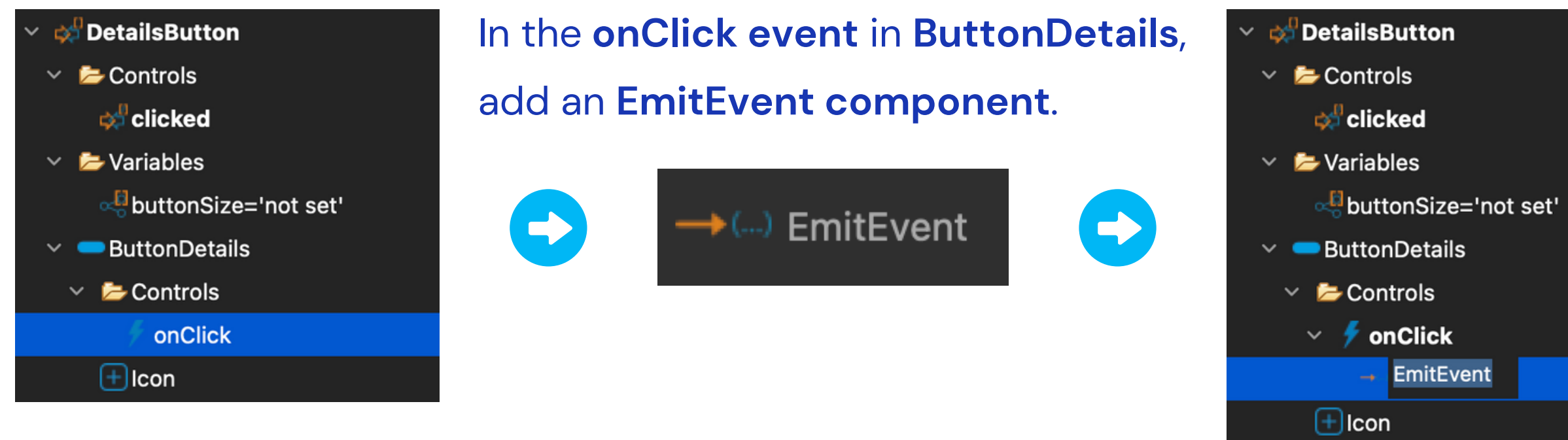


7.5 Use events in SharedComponents

In **DetailsButton** sharedComponent, add a **SharedComponent BindingEvent** and rename it **clicked**.

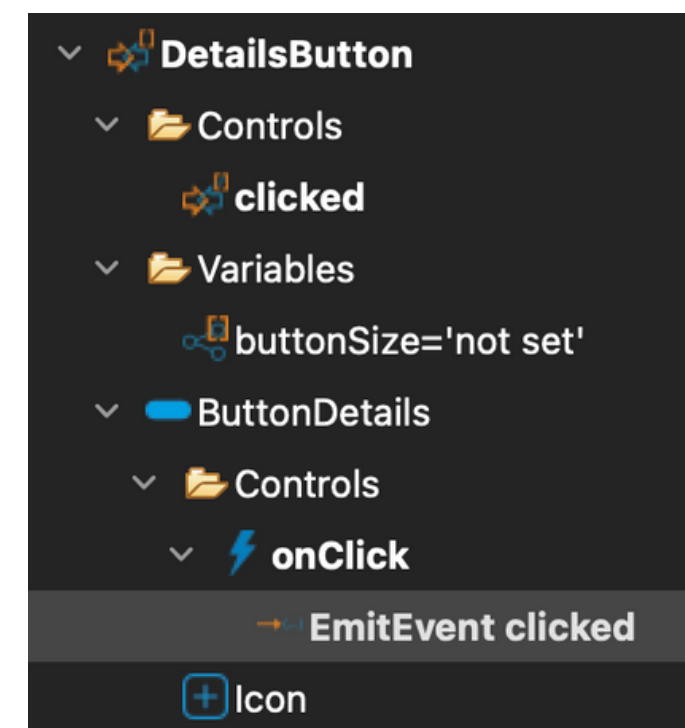
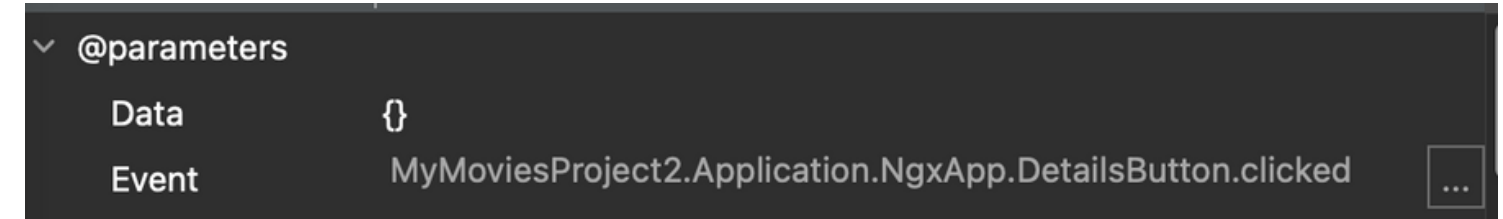
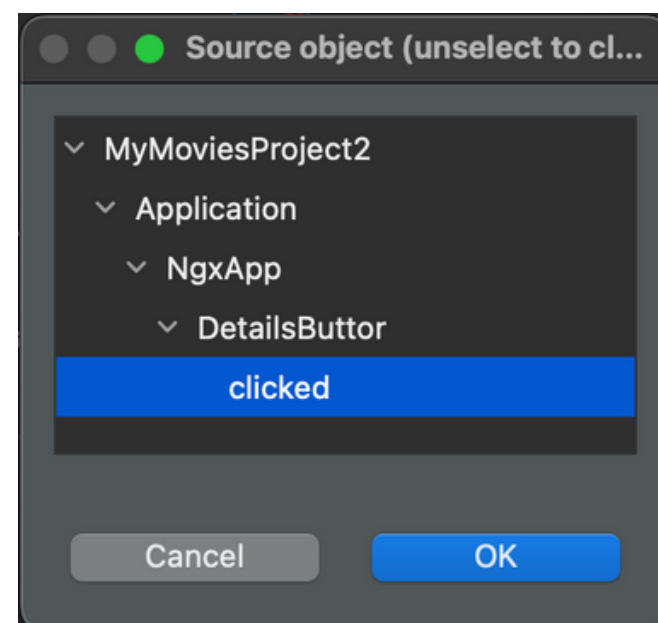
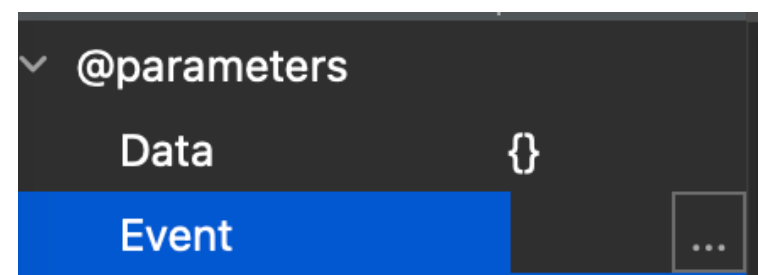
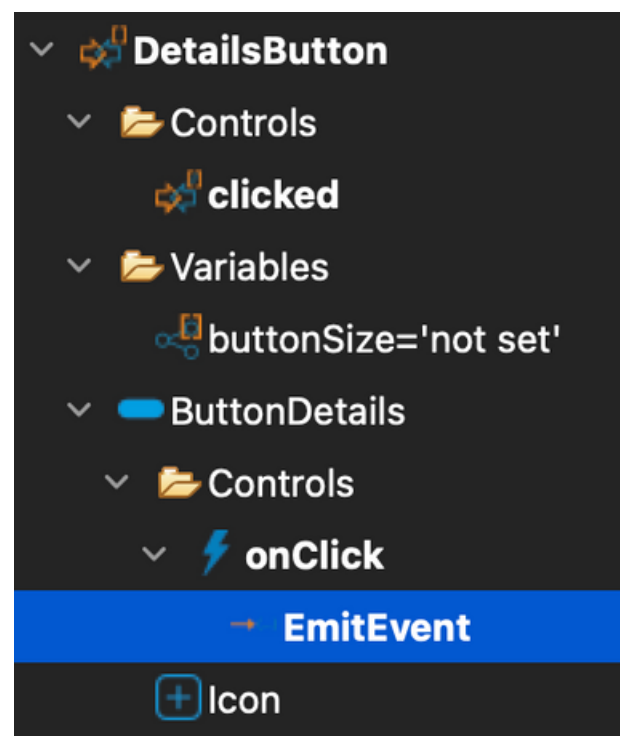


In the **onClick** event in **ButtonDetails**, add an **EmitEvent** component.



7.5 Use events in SharedComponents

In the **EmitEvent**'s properties,
in the **Event** property, click on ... to open the **Source object panel** and select **clicked** as value.

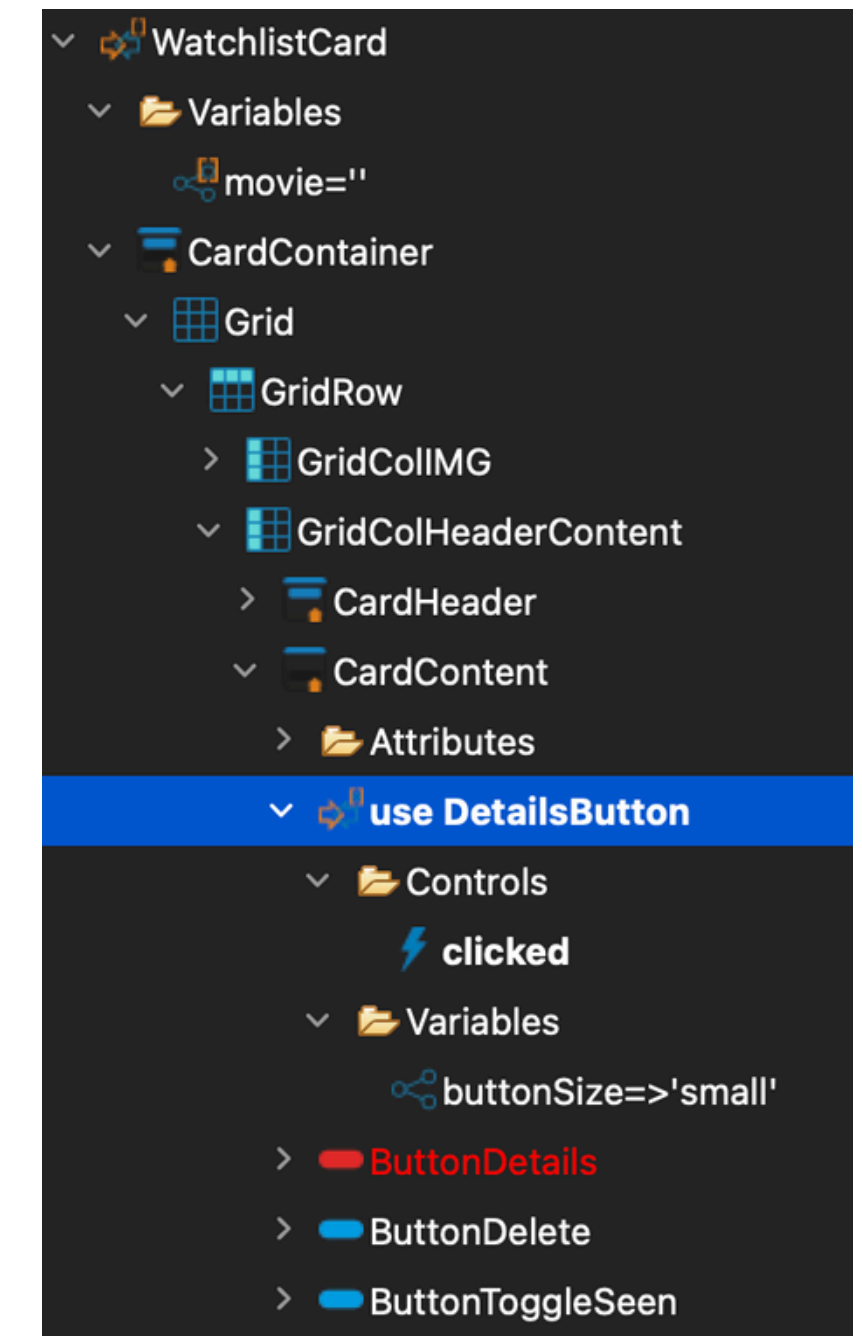
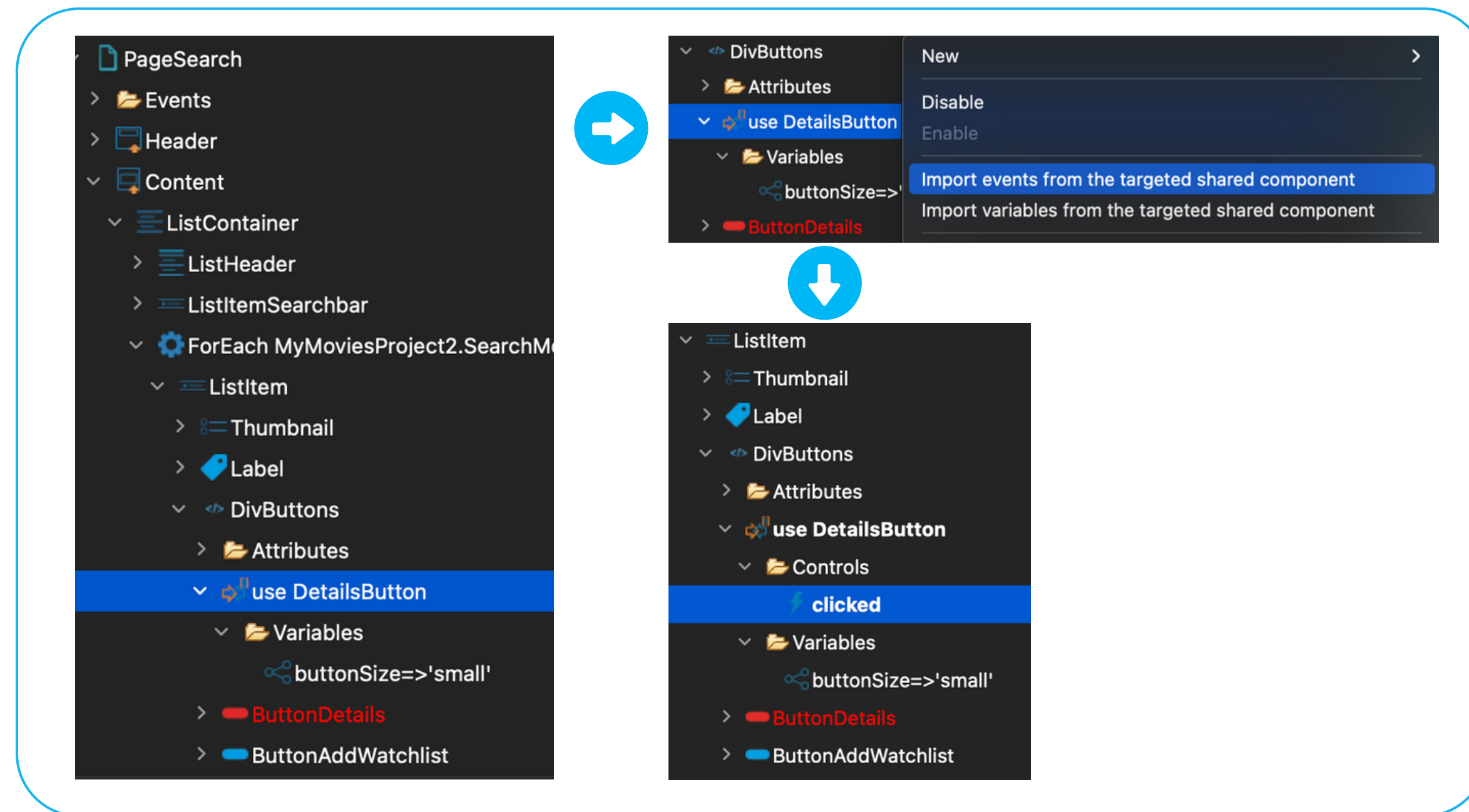


7.5 Use events in SharedComponents

In **PageSearch**, right-click on **use DetailsButton**, and select **Import event** from the targeted shared component to import the **SharedComponent BindingEvent clicked**.

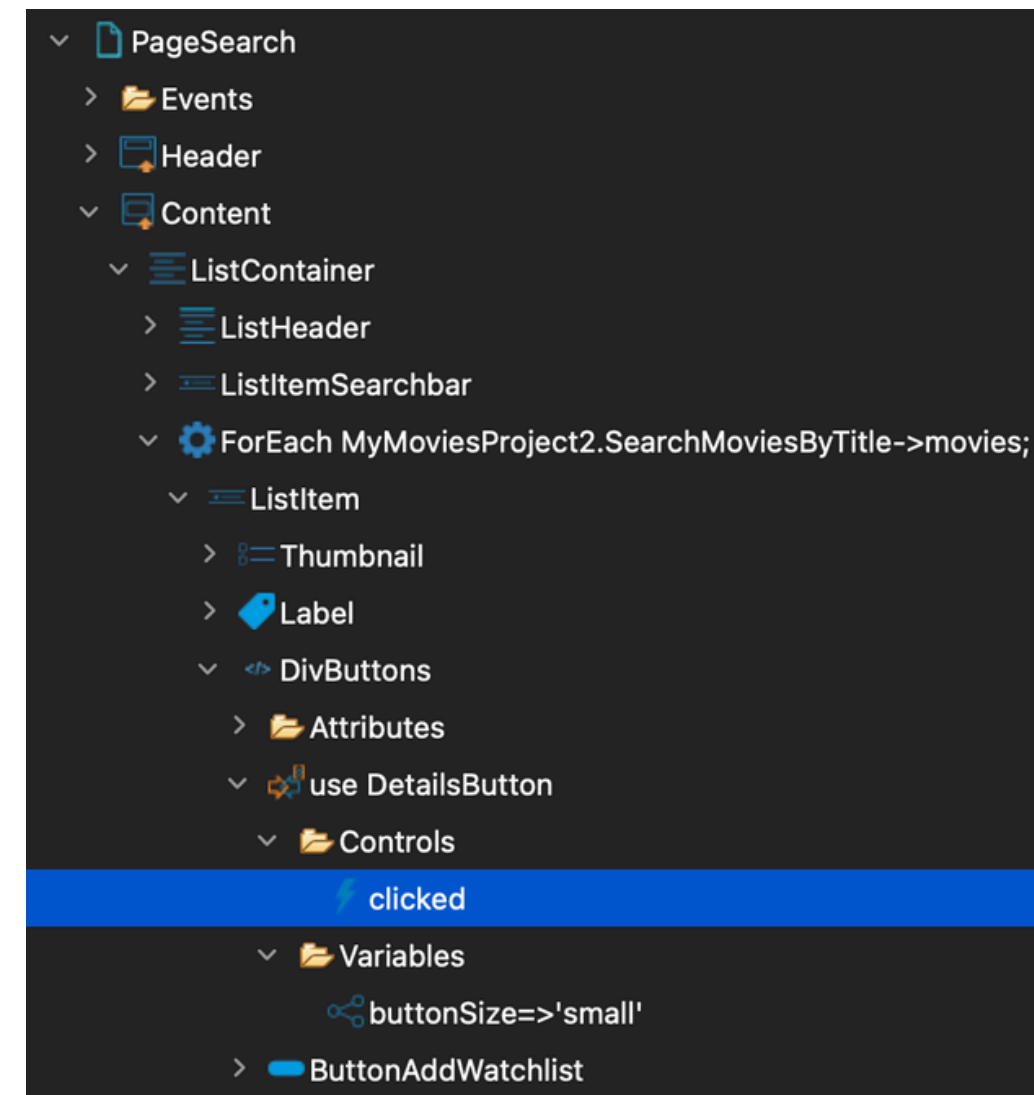
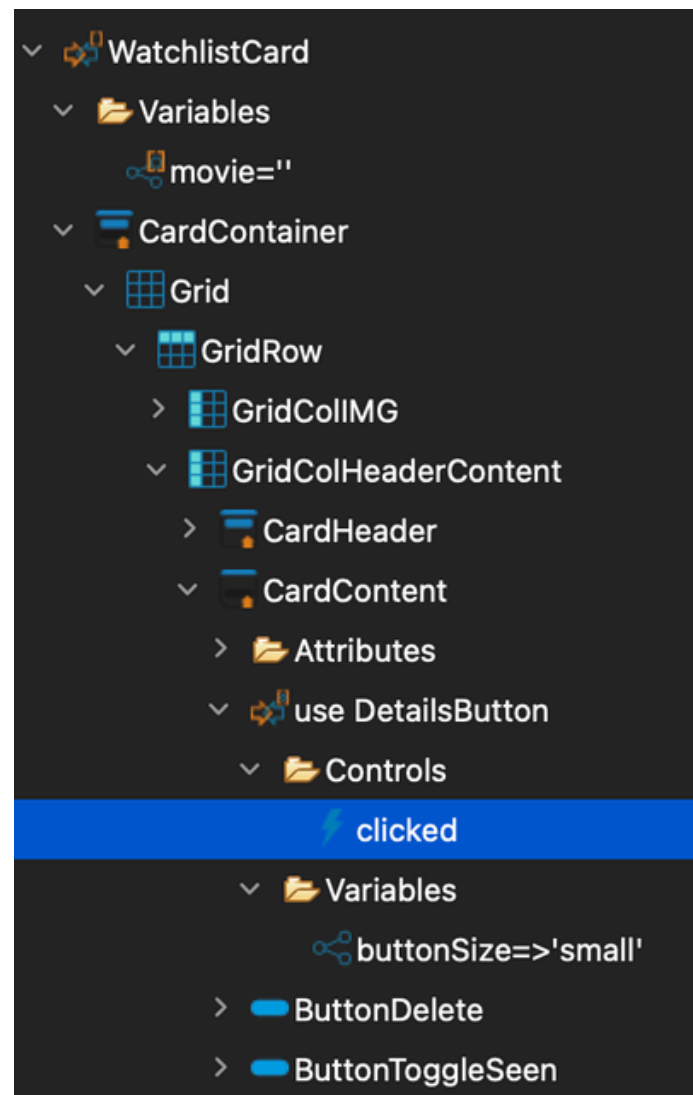


Do the same thing for **use DetailsButton** in **WatchlistCard**.



7.5 Use events in SharedComponents

Now, in both use **DetailsButton**, in **PageSearch** and **WatchlistCard**, we want to use the **SharedComponent BindingEvent clicked** to do the same thing : go to a new page, **PageMovieDetails**.

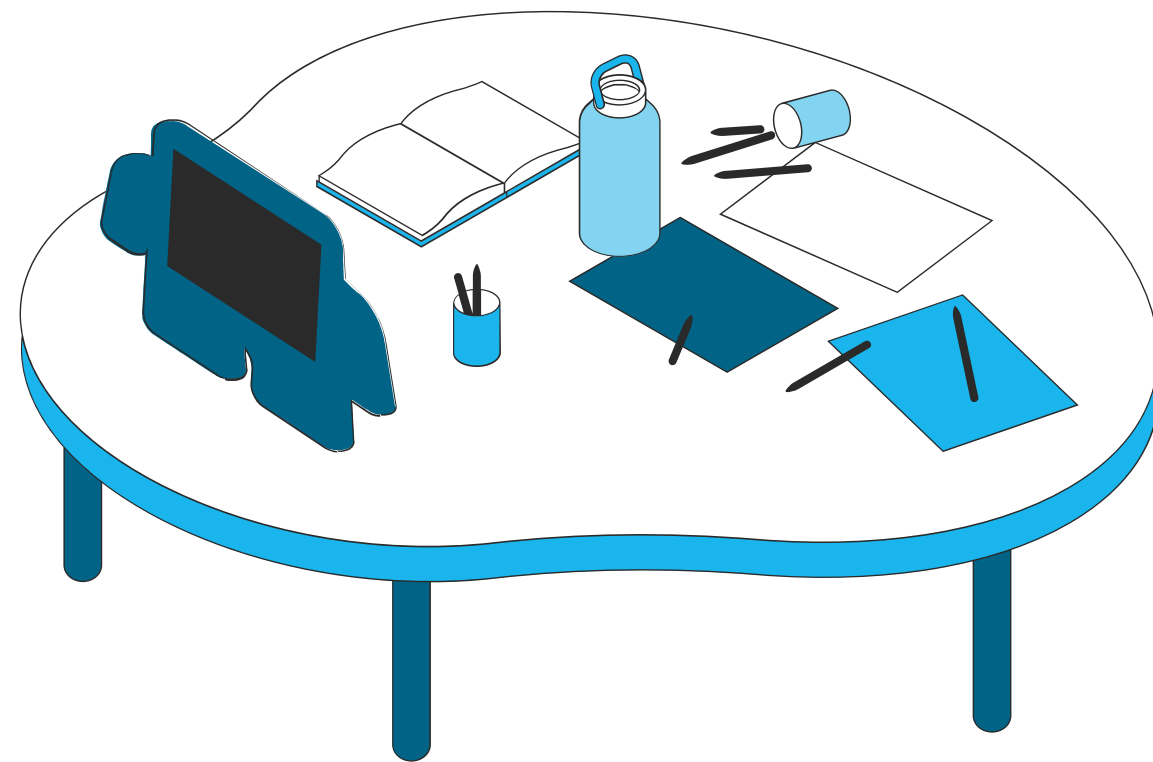


To do that,
we need to add the same new **action**
to **push the page** in each use **DetailsButton**.

To avoid repeating the action,
we are going to create a **SharedAction**.



8 – Shared Actions



8.1 SharedAction components

8.2 Create PageMovieDetails

8.3 Create a Shared action

8.4 Invoke a Shared action

8.5 Add the PageMovieDetails layout

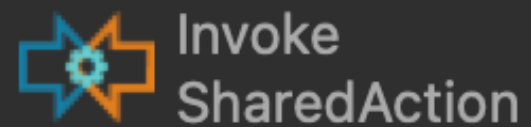
8.6 Add a Shared action variable

8.7 Use a Shared action variable

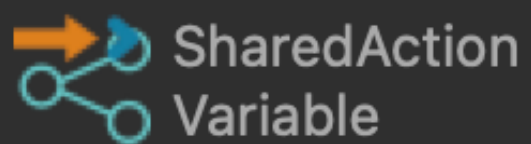
8.1 SharedAction components



The **SharedAction component** is used to define an entire action logic. It is shared between all the applications in the workspace.



You can add a **SharedAction** under an application and use it through an **Invoke SharedAction** from any application page.



You may add **SharedAction Variables** to pass additional parameters to the executing function of the action.

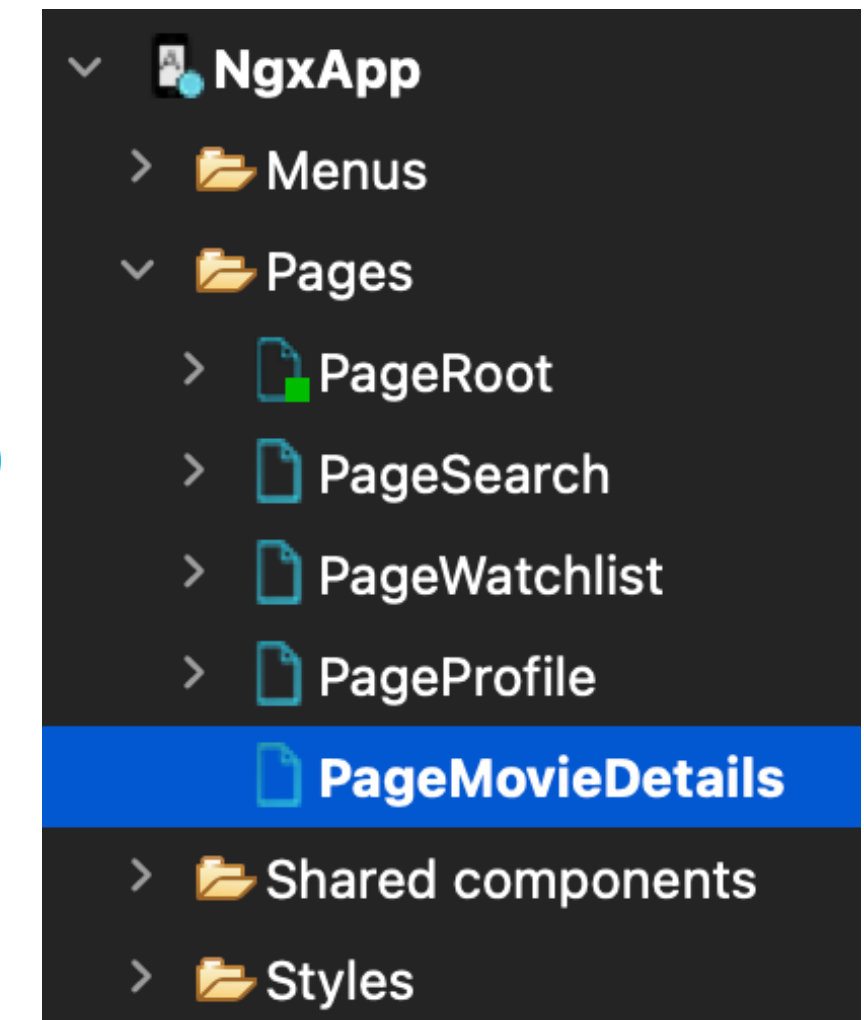
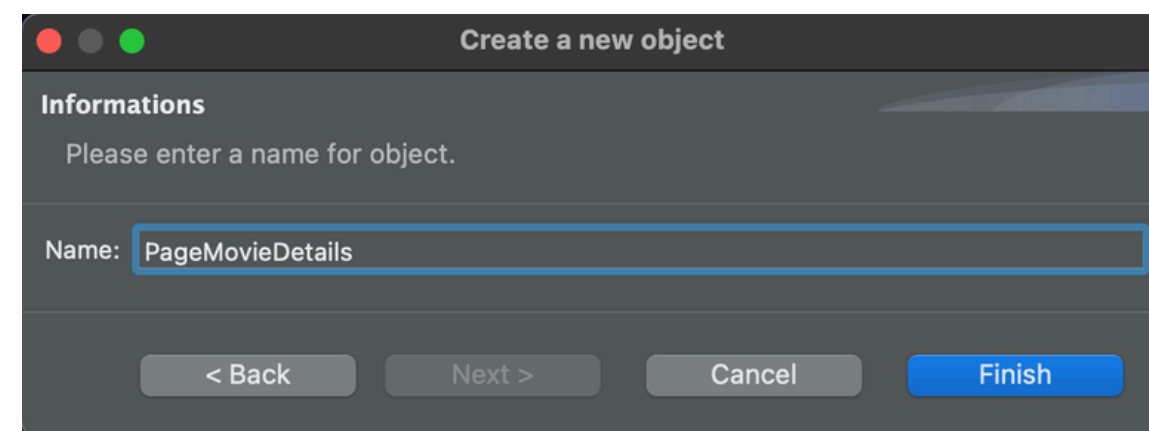
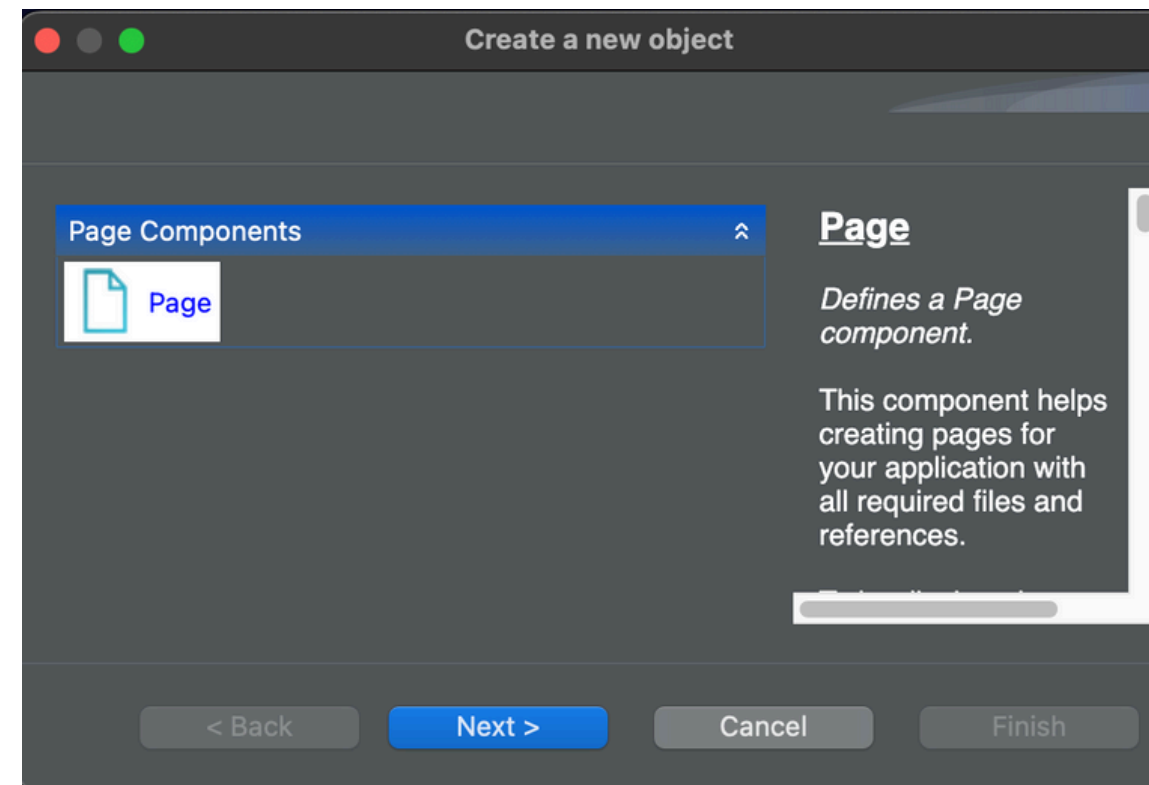
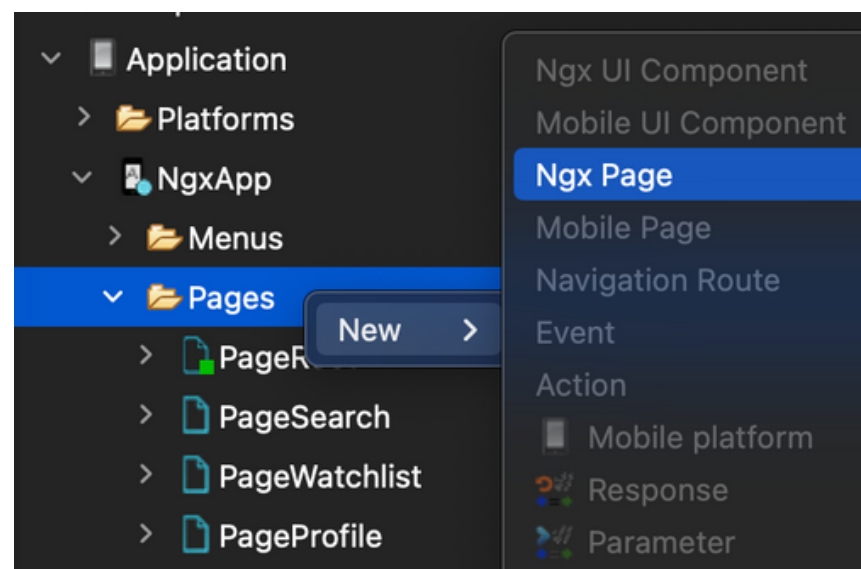
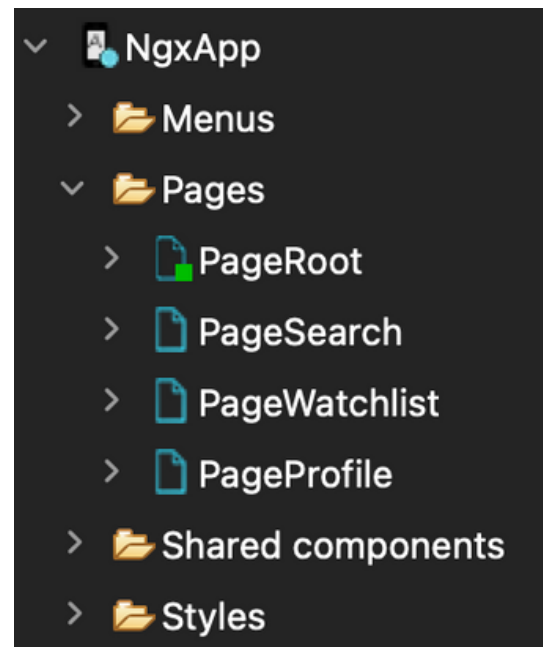
To access a shared action **SharedAction Variable** from any child Action property or variable, use in script mode (TS):

- `params.<variable-name>`
- `stack['<shared-action-name>'].vars.in.<variable-name>`

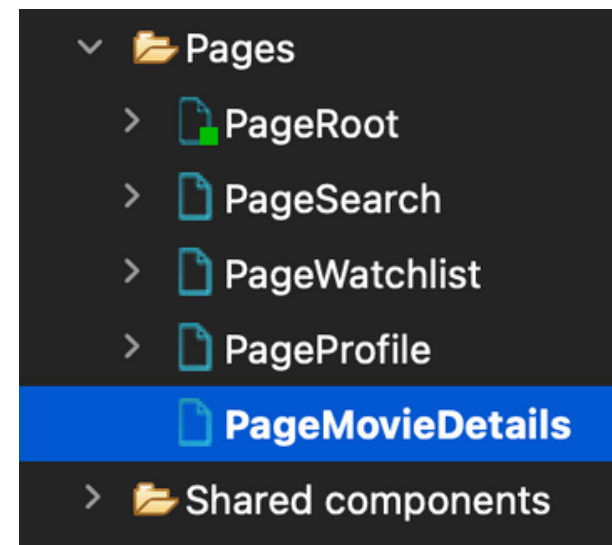


8.2 Create PageMovieDetails

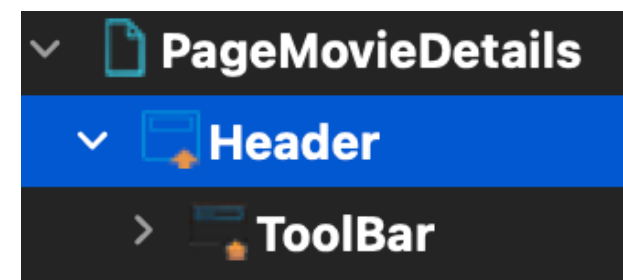
Create a new page and rename it **MovieDetails**.



8.2 Create PageMovieDetails



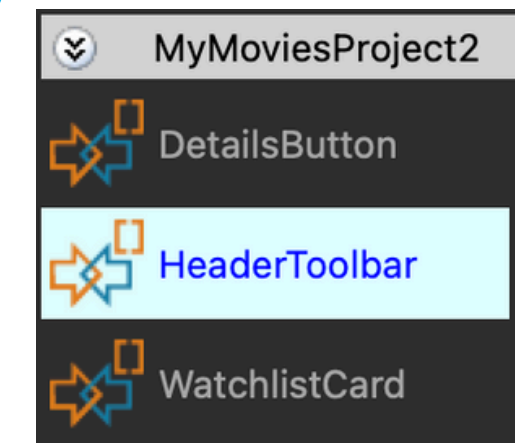
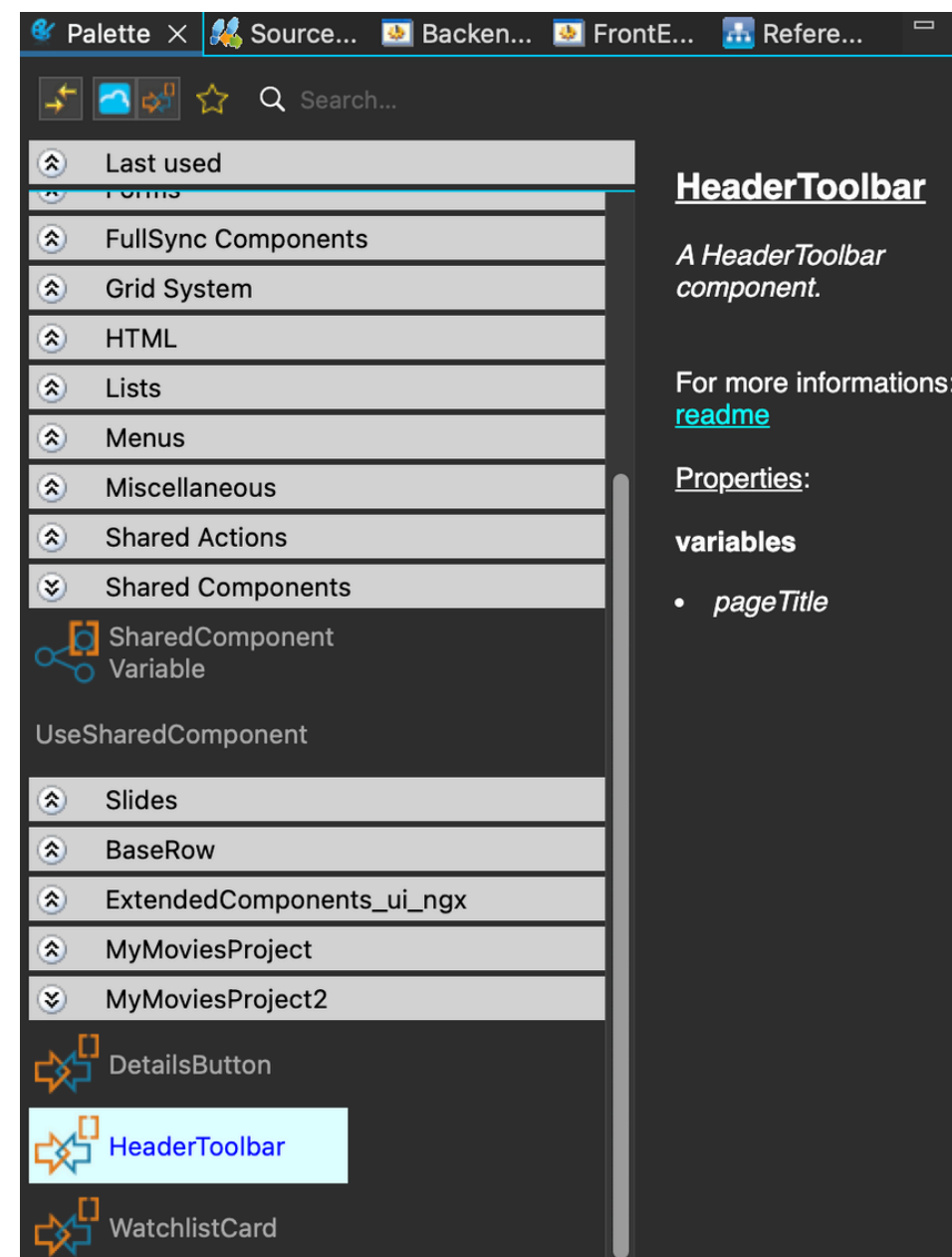
Add a **Header** in **PageMovieDetails**.



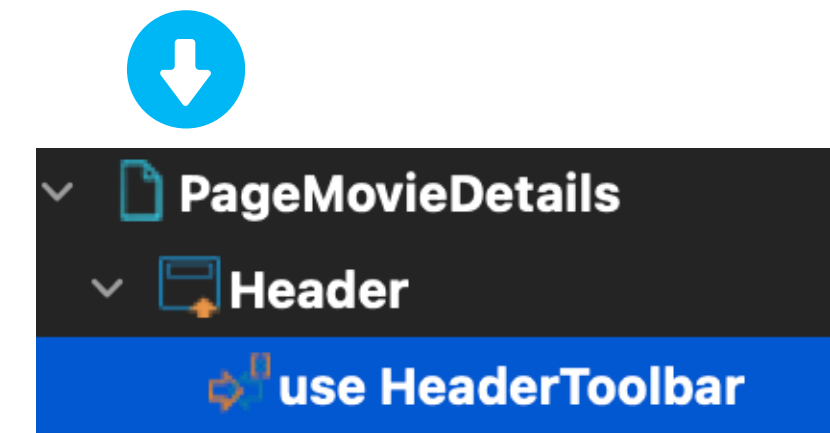
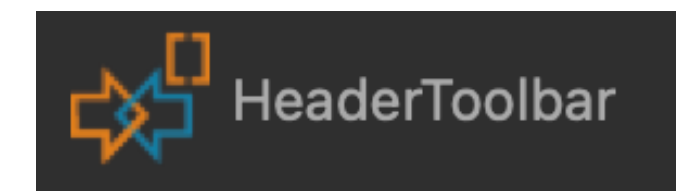
Remove the **ToolBar** inside.



All the sharedComponents created in the workspace are available in the palette.

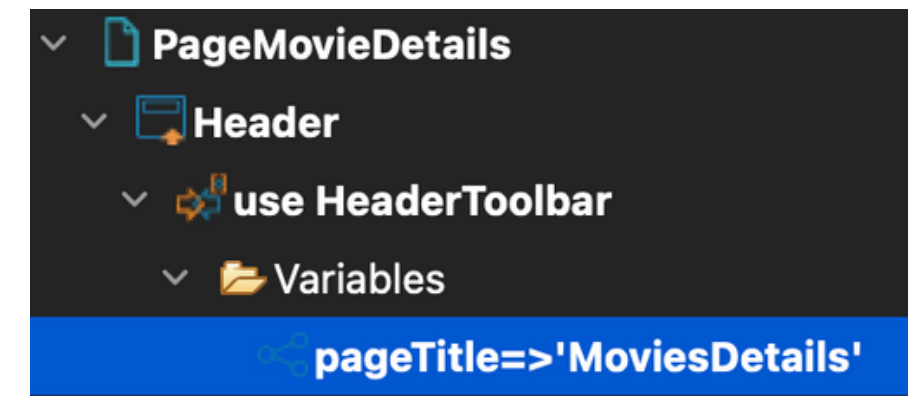
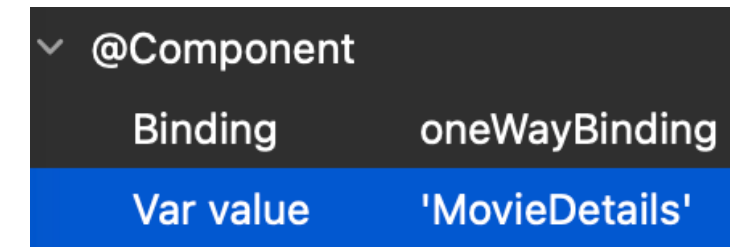
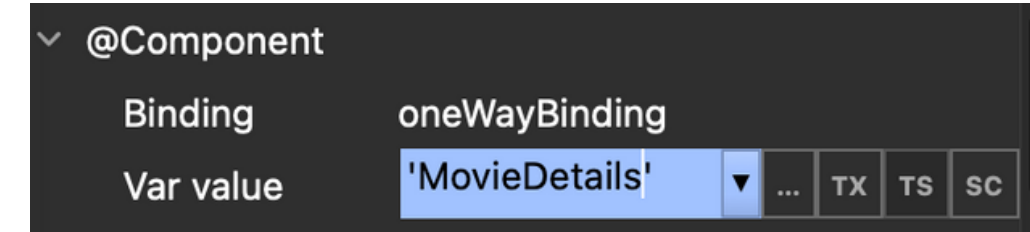
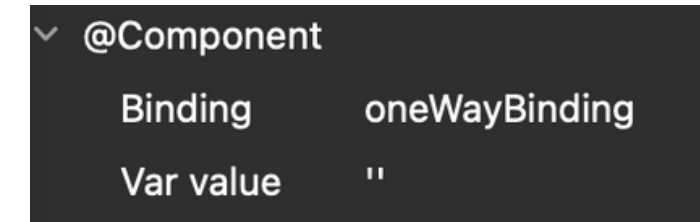
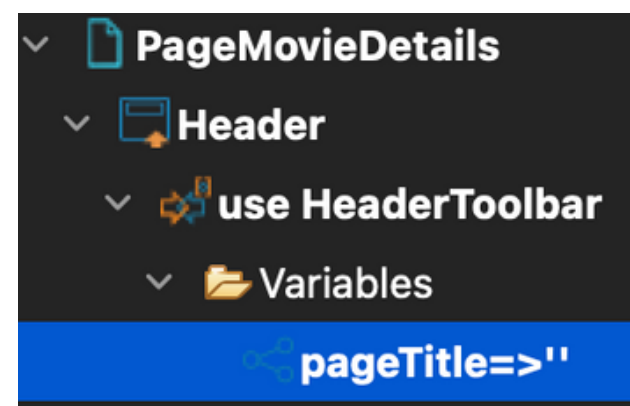
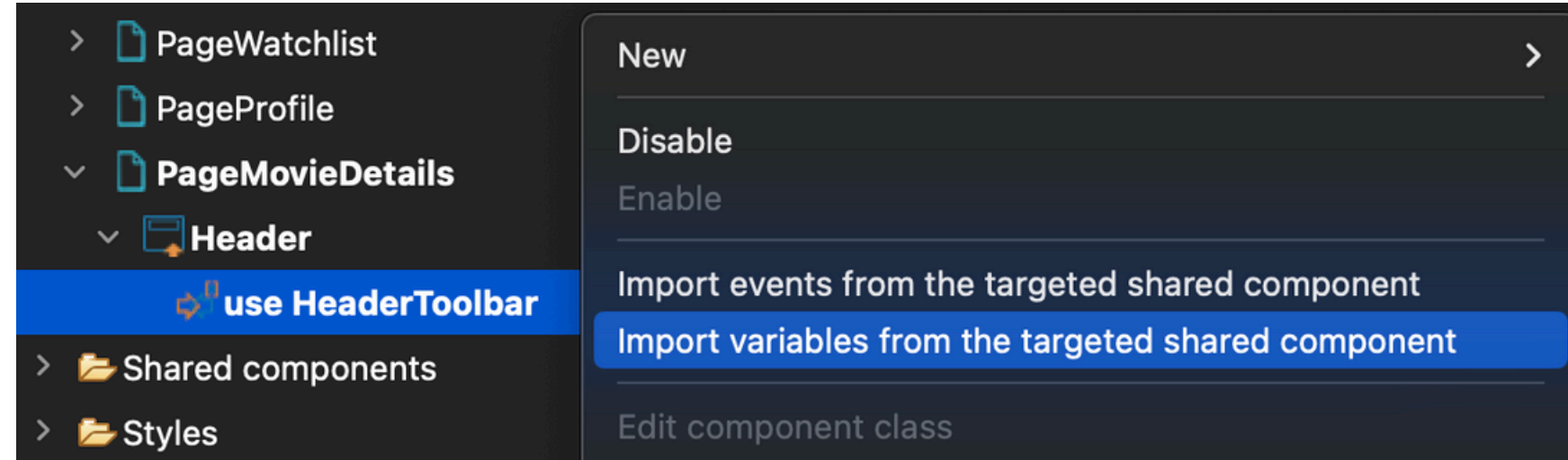


Drag an **HeaderToolBar** from the palette and drop it in the **Header**.

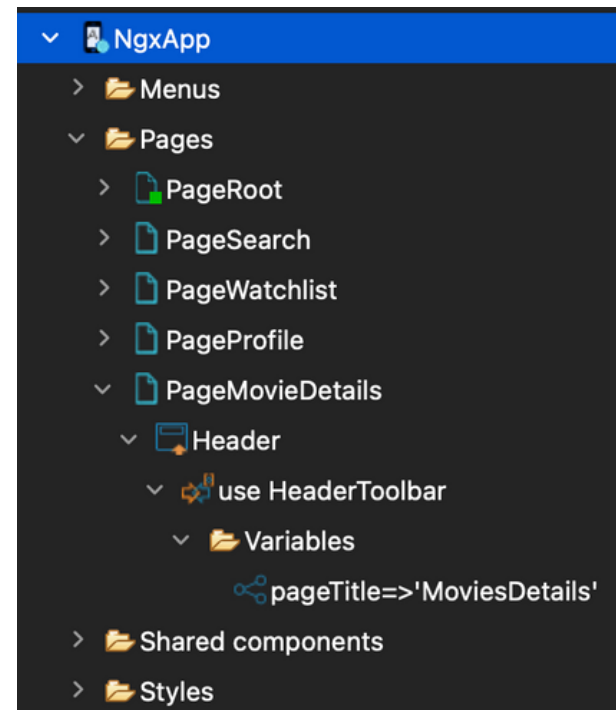


8.2 Create PageMovieDetails

Import the **pageTitle** variable,
and enter '**MovieDetails**' as **Var value** in TS scope.



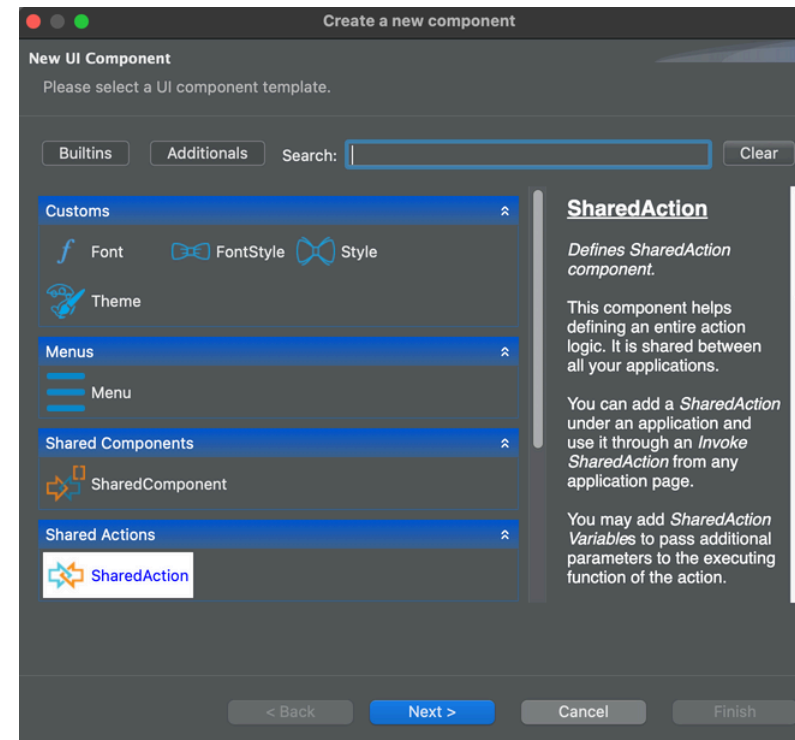
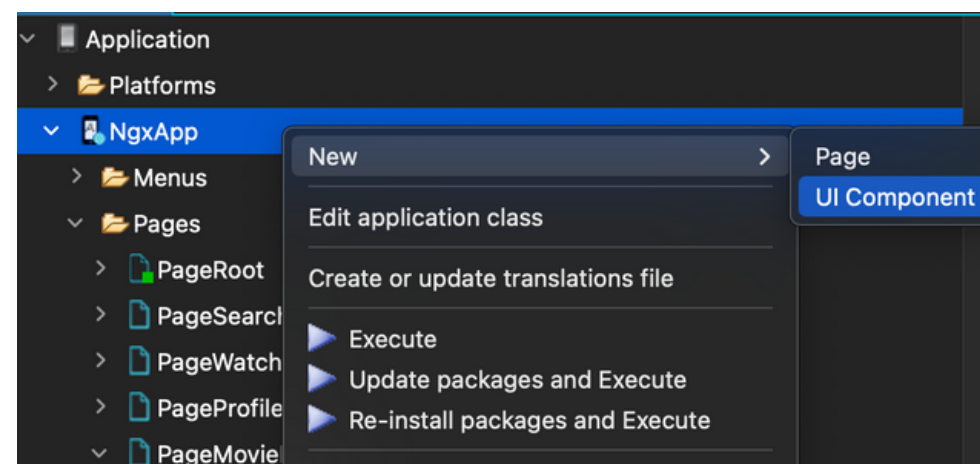
8.3 Create a Shared action



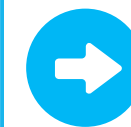
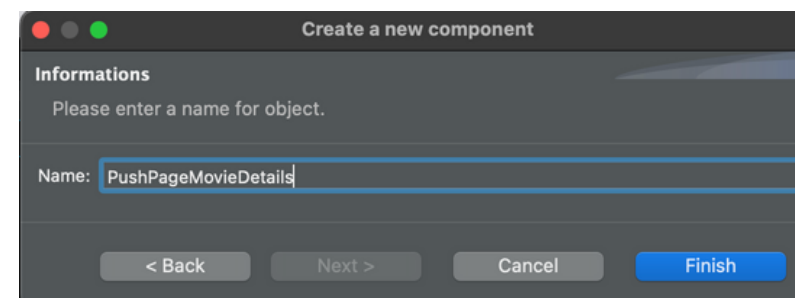
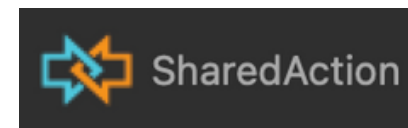
Right-click on **NgxApp**



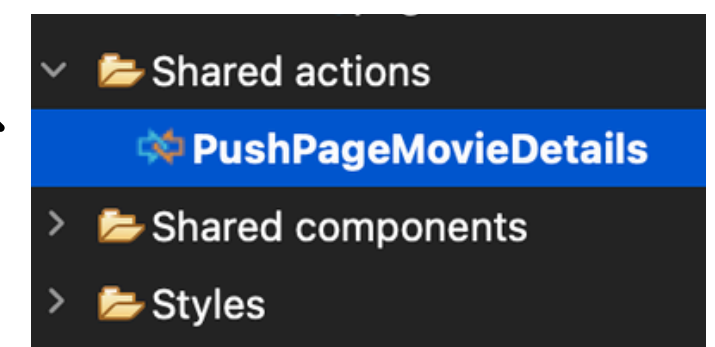
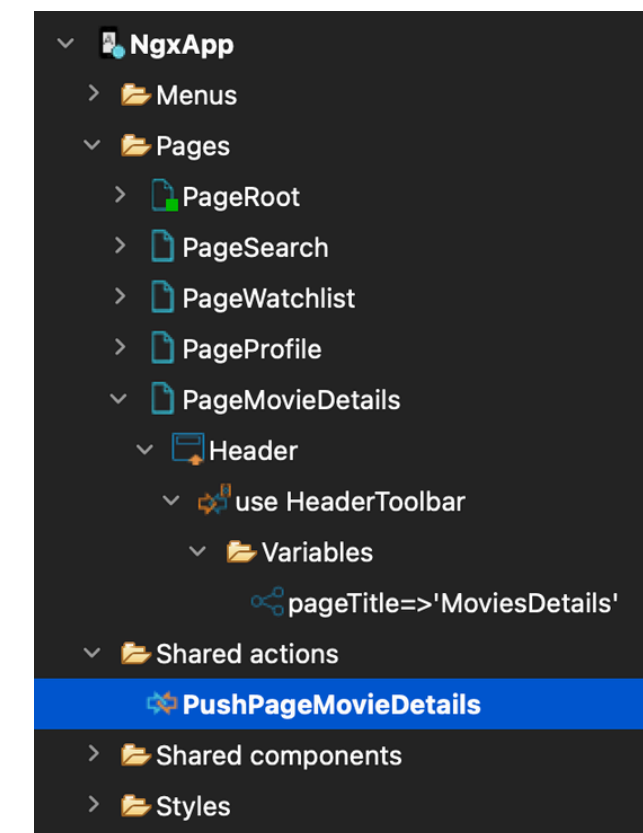
and select **New**,
then **UI Component**.



In the **Create a new component** panel,
click on **SharedAction**, then enter
PushPageMovieDetails as name.

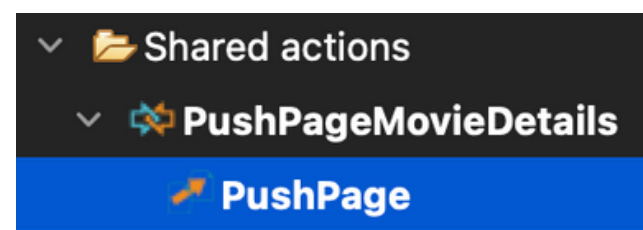
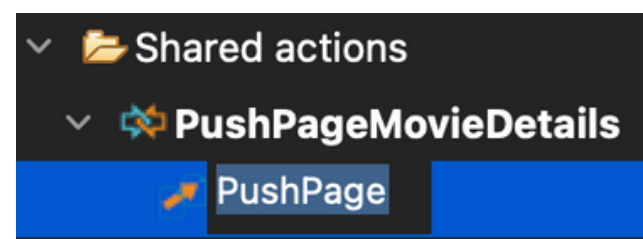
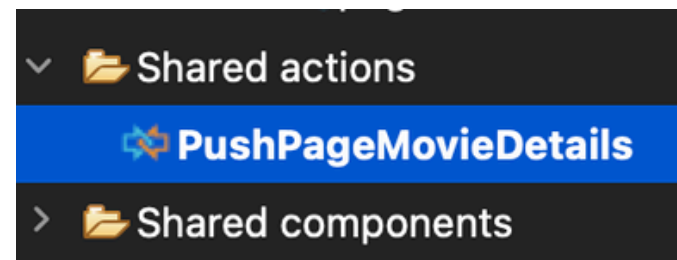


The **PushPageMovieDetails**
SharedAction has been created
in a **Shared actions** folder.



8.3 Create a Shared action

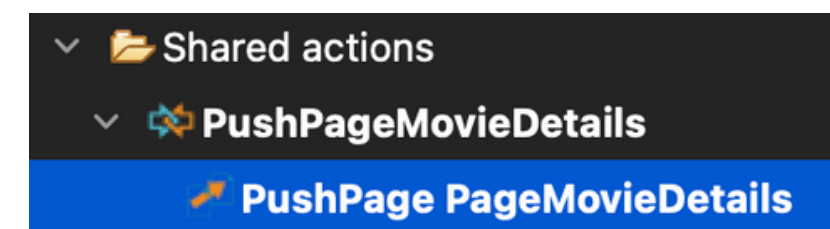
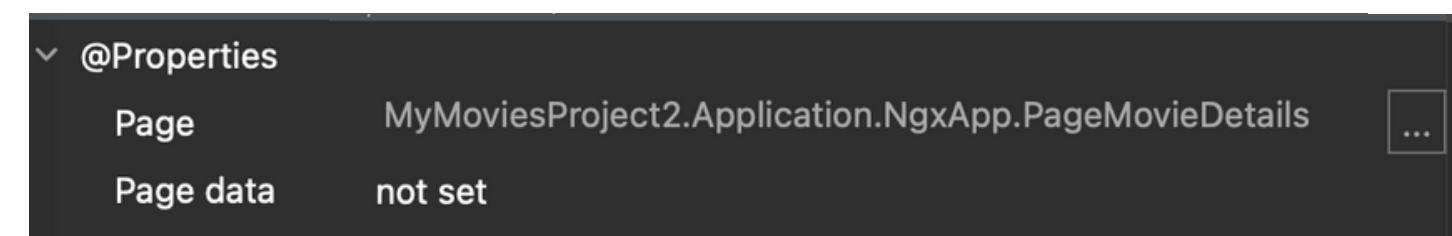
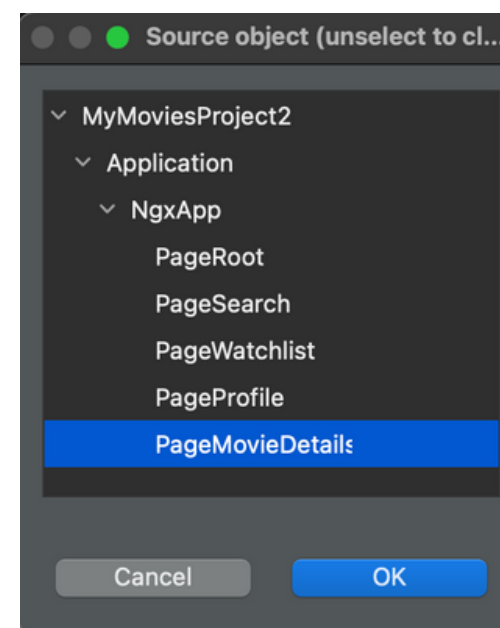
In the **PushPageMovieDetails** **SharedAction** add a **PushPage** component.



In the **PushPage** component's properties, in the **Page** property, click on ... to open the **Source object panel**.

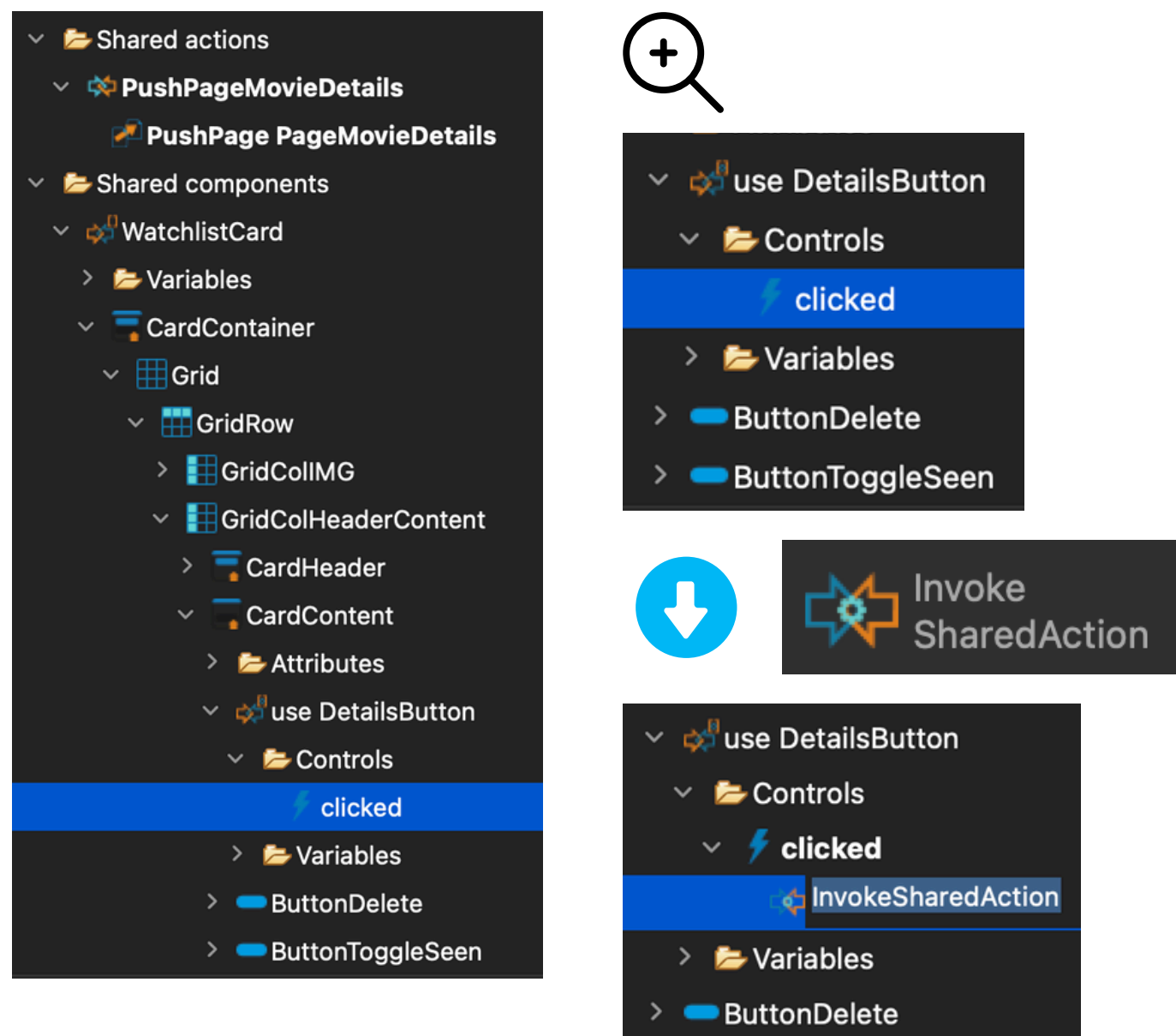


In the **Source object panel**, select **PageMovieDetail** as source.



8.4 Invoke a Shared action

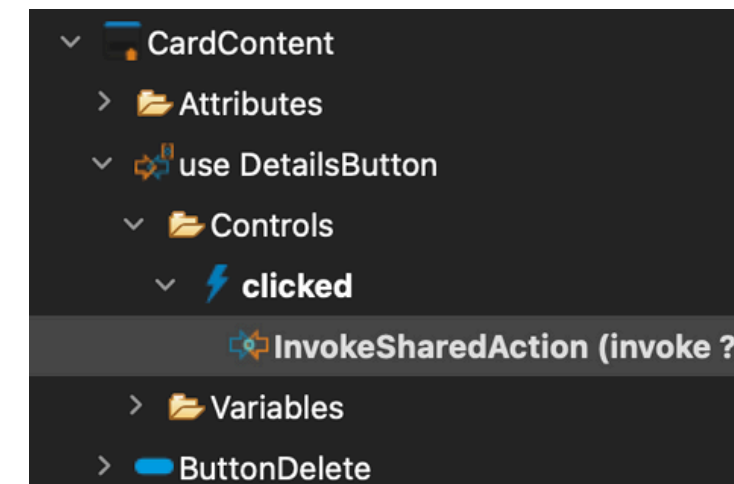
In **WatchlistCard**, in the **use DetailsButton**, add an **Invoke SharedAction** component in the **SharedComponent BindingEvent** clicked.



The diagram illustrates the process of adding an **Invoke SharedAction** component to the **use DetailsButton** BindingEvent. It shows a tree view of the **WatchlistCard** component, where the **use DetailsButton** component is selected. The **clicked** event is highlighted, and the **Invoke SharedAction** component is added to the **SharedComponent BindingEvent** list.



In the **Invoke SharedAction** component's properties



The screenshot shows the **Invoke SharedAction** component's properties panel. The **Clicked** property is selected, and the **Invoke SharedAction** component is highlighted in the list of shared actions.

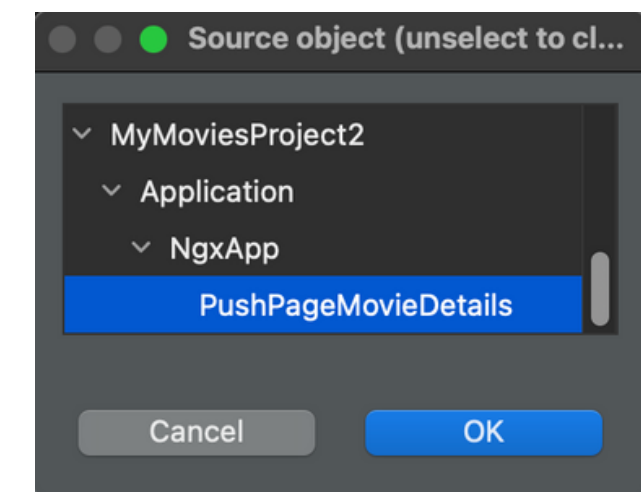
In the **Shared action** property, click on ... to open the **Source object** panel.



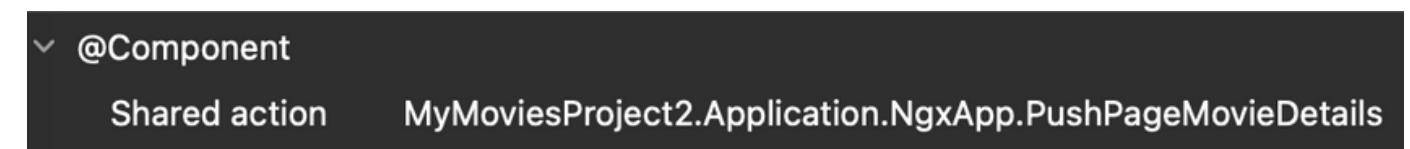

The screenshot shows the **Shared action** property panel. The **Shared action** dropdown menu is open, and the **PushPageMovieDetails** action is selected.



In the **Source object** panel, select **PushPageMovieDetails**.



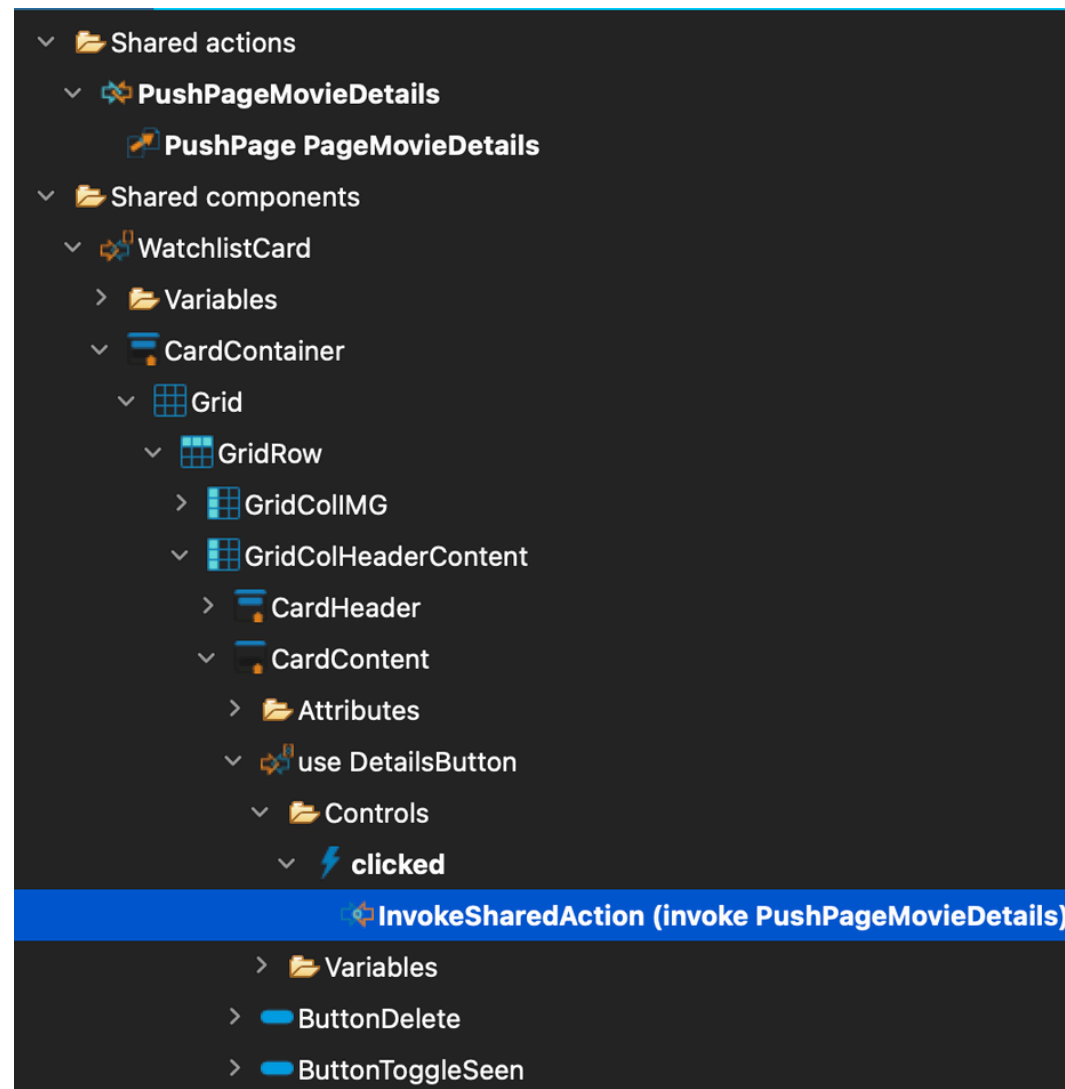
The screenshot shows the **Source object** panel. The **PushPageMovieDetails** action is selected in the list of source objects.

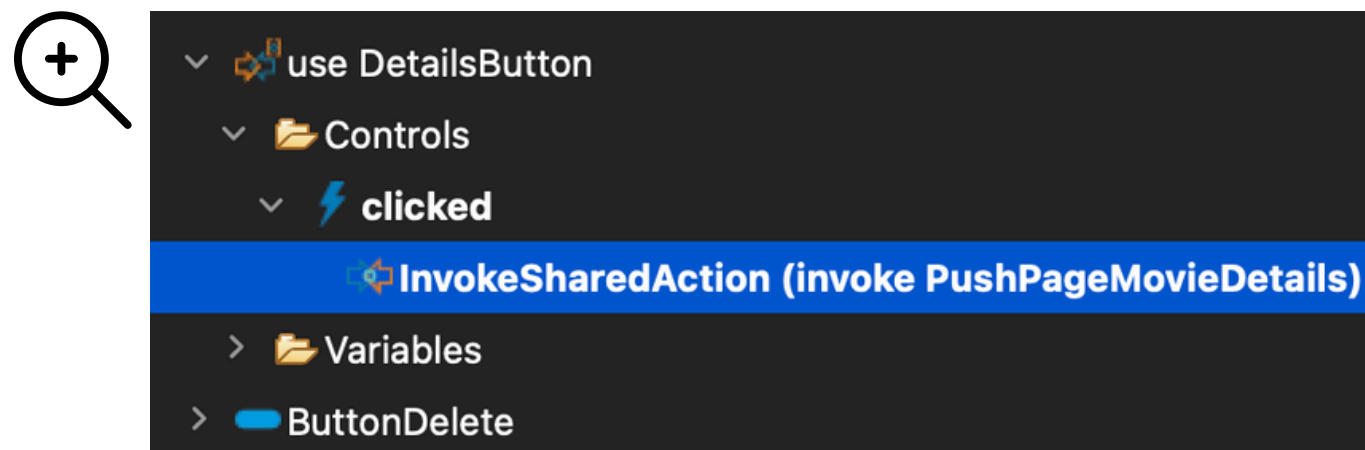
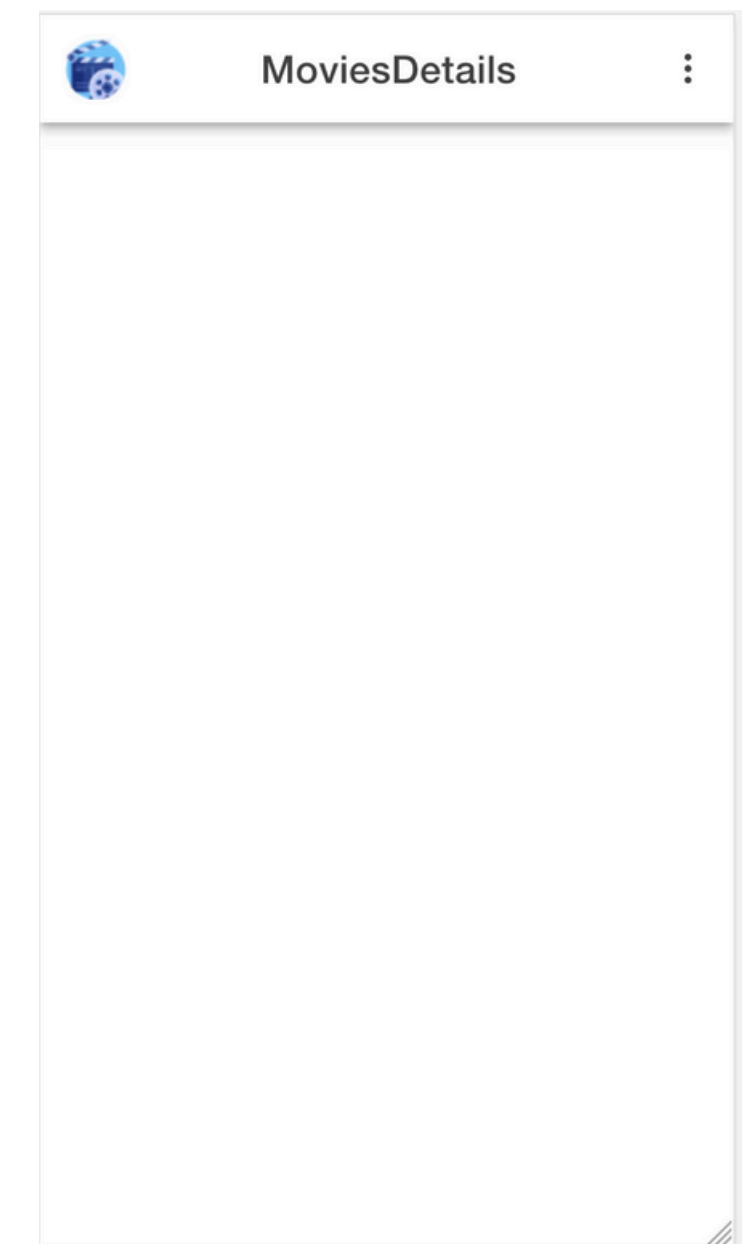
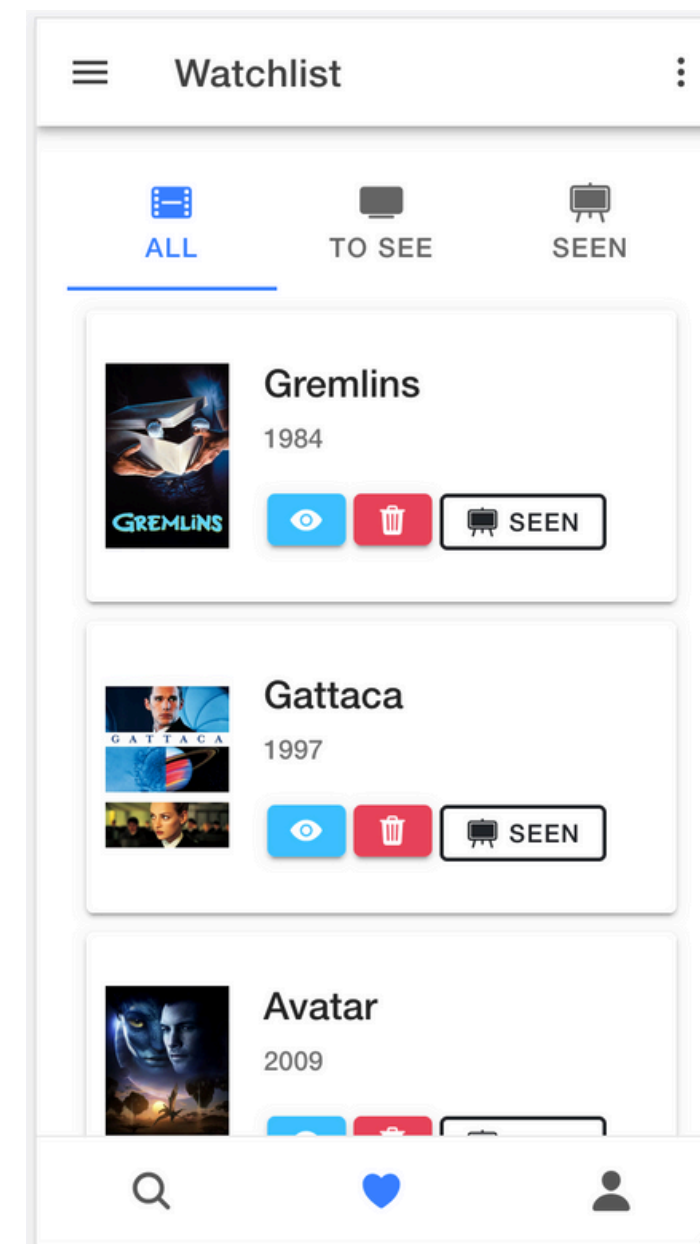
The screenshot shows the **Source object** panel. The **PushPageMovieDetails** action is selected in the list of source objects.



8.4 Invoke a Shared action

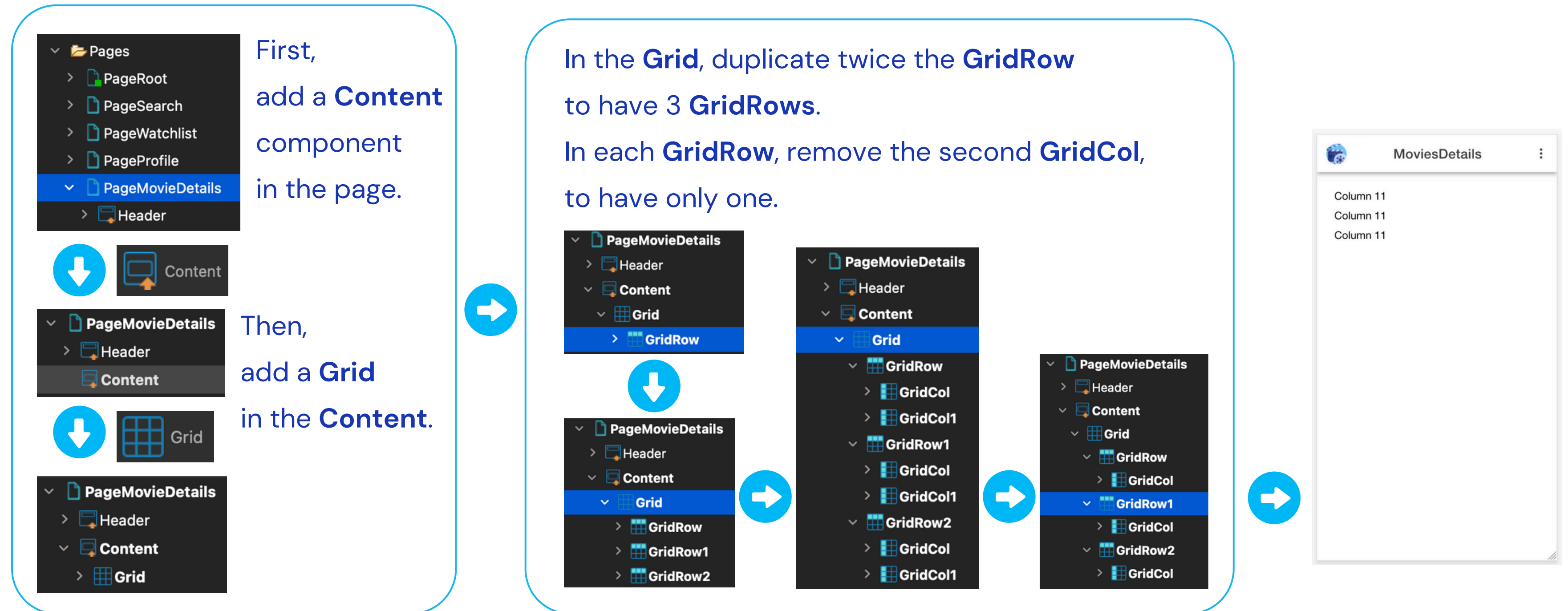


The **SharedAction PushPageMovieDetails** is now invoked when the button is clicked. and **PageMovieDetails** is pushed.



8.5 Add the PageMovieDetails layout

Now, we are going to create the layout to display movie details in our page.



8.5 Add the PageMovieDetails layout

In the first **GridCol**, remove the **Label**.
Add an **h1 Heading** and 2 **Buttons**.
Rename the **Buttons** : **ButtonAddToWatchlist**
and **ButtonDeleteFromWatchlist**.

</> Heading

@Component

Identifier

Tag name

h1

Button

↓

PageMovieDetails

Header

Content

Grid

GridRow

GridCol

Heading

ButtonAddToWatchlist

ButtonDeleteFromWatchlist

GridRow1

GridCol

GridRow2

GridCol

MoviesDetails

heading

SOME BUTTON TEXT

SOME BUTTON TEXT

Column 11

Column 11



In the first **GridCol**, add a **FormatLayout**.
In its **@Text Alignment** properties,
set the **Alignment** property to **ion-text-center**.

↓

Format Layout

PageMovieDetails

Header

Content

Grid

GridRow

GridCol

Attributes

FormatLayout

Heading

ButtonAddToWatchlist

ButtonDeleteFromWatchlist

GridRow1

GridCol

GridRow2

GridCol

@Text Alignment

Alignment

ion-text-center

Alignment 0-576 px (xs)

not set

Alignment 1200-over px (xl)

not set

Alignment 576-768 px (sm)

not set

Alignment 768-992 px (md)

not set

Alignment 992-1200 px (lg)

not set

MoviesDetails

heading

SOME BUTTON TEXT

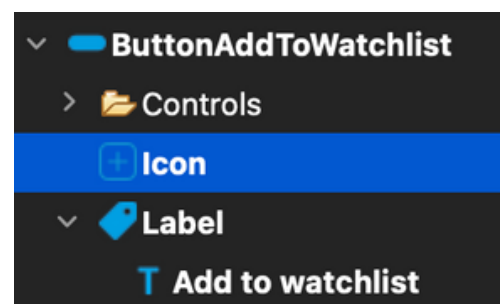
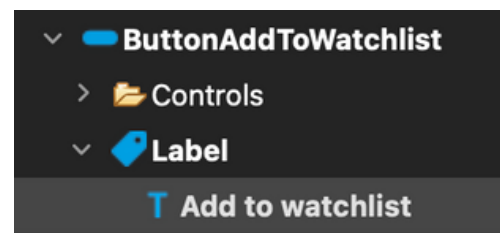
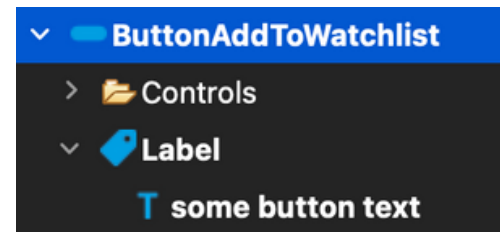
SOME BUTTON TEXT

Column 11

Column 11



8.5 Add the PageMovieDetails layout

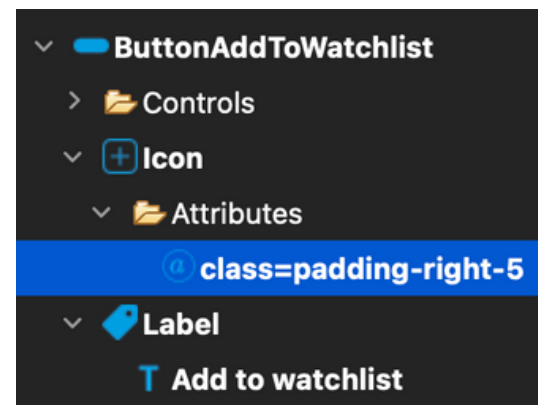


In **ButtonAddToWatchlist**,
change the text
inside the **Label**
to **Add to watchlist**.

Add an **Icon** before the **Label**.
Set its **Icon name** property
to 'add-circle'.



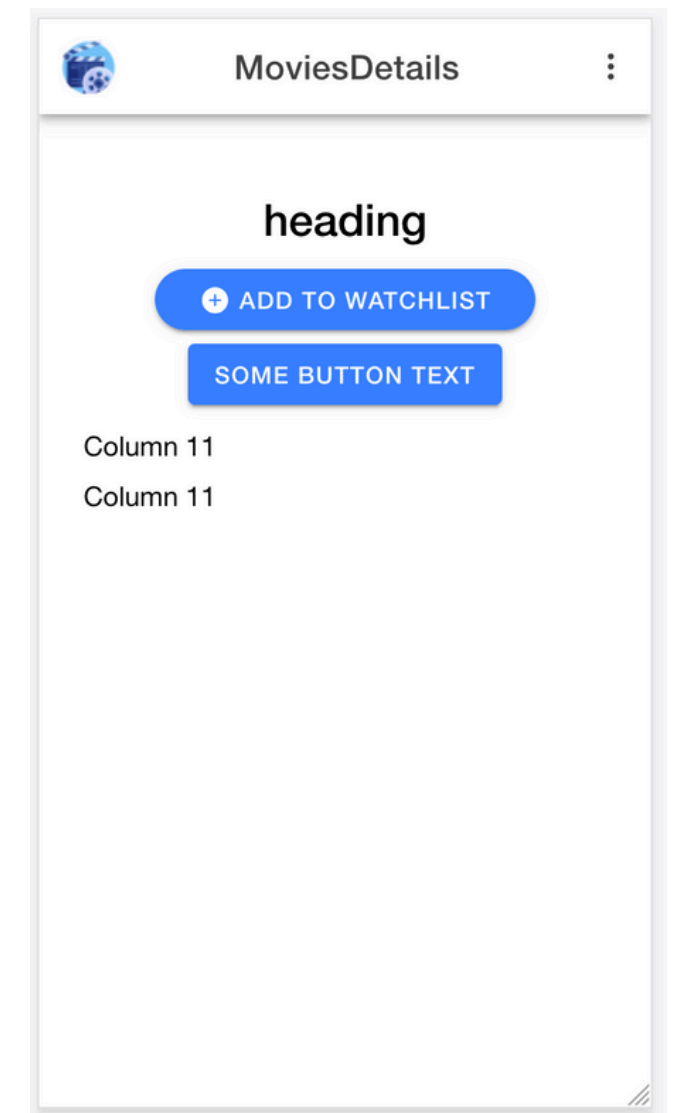
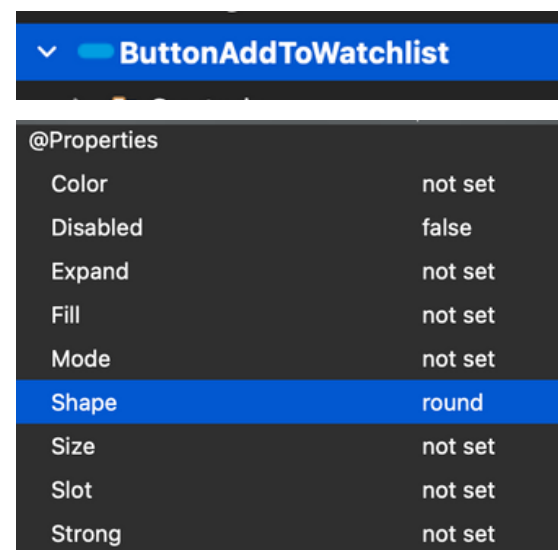
@Properties	
Color	not set
Icon los name	
Icon Md name	
Icon name	'add-circle'
Size	not set
Slot	not set
Source file URL	not set



In the **Icon**,
add an **Attr** component.
Set it to **class=padding-right-5**.

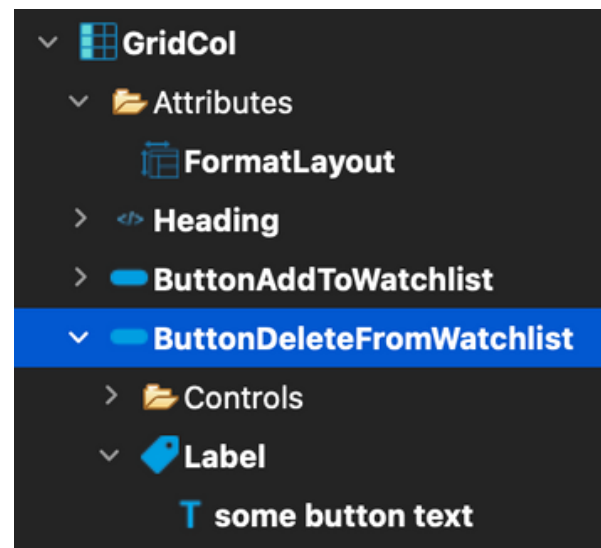


In **ButtonAddToWatchlist's** properties
set the **Shape** property to **round**.

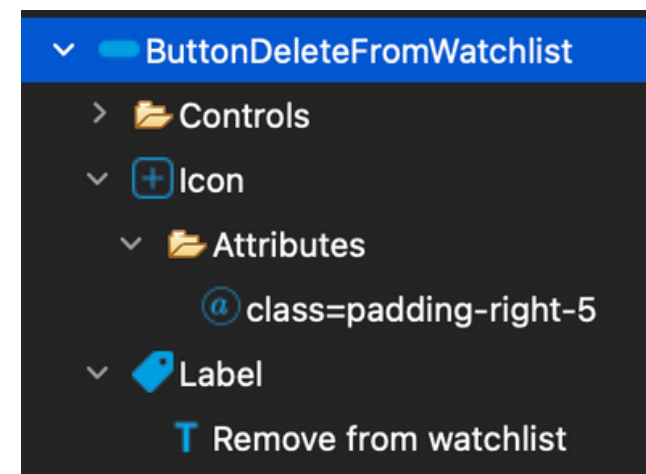
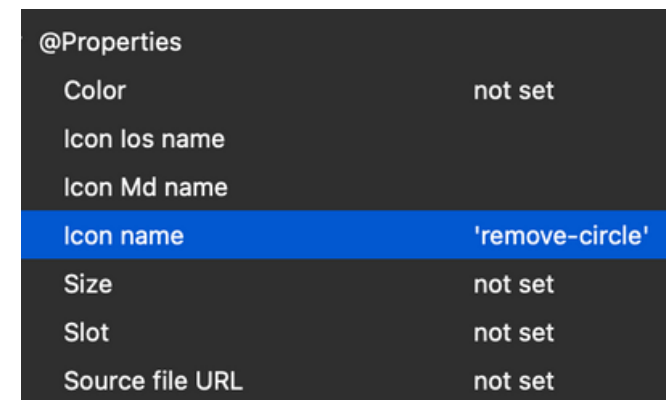
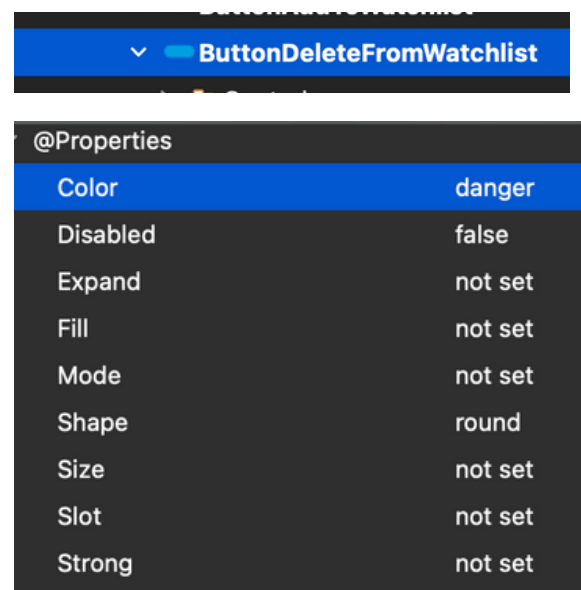


8.5 Add the PageMovieDetails layout

Let's have a look on **ButtonDeleteFromWatchlist**.

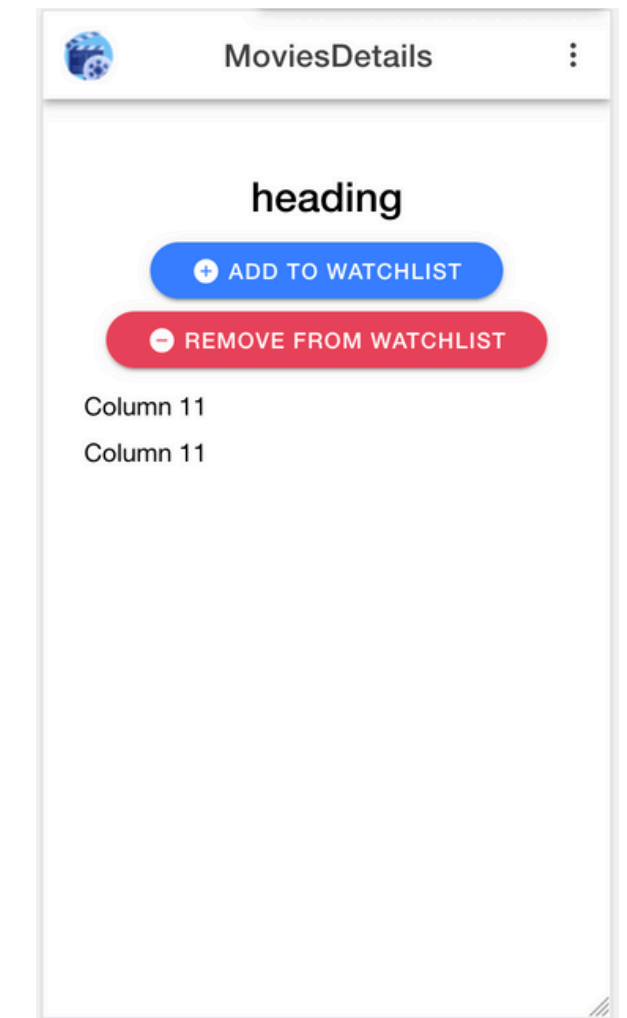


In this **Button's** properties set the **Color** property to **danger**, set the **Shape** property to **round**.



Add an **Icon** before the **Label**. Set its **Icon name** property to **'remove-circle'**.

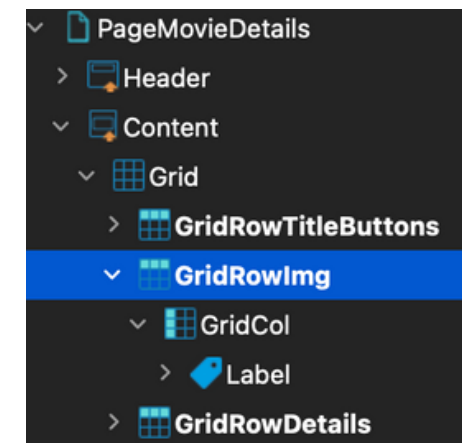
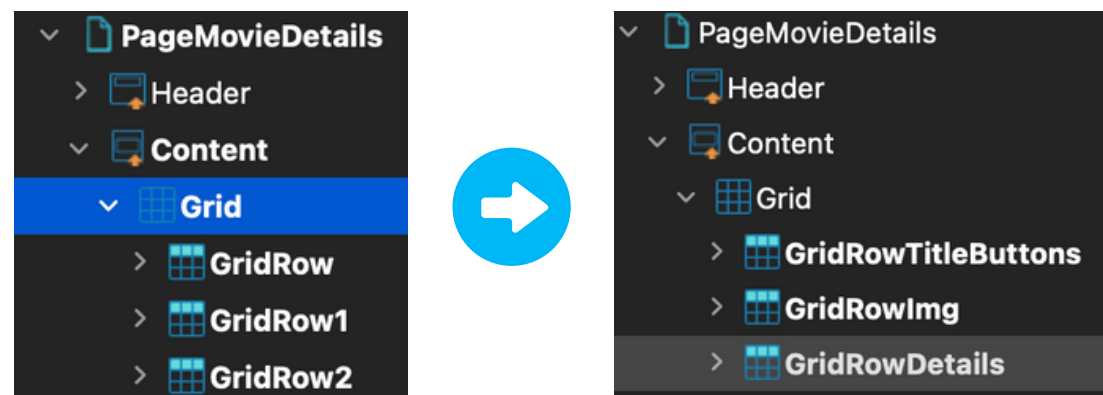
In the **Icon**, add an **Attr** component. Set it to **class=padding-right-5**. In **ButtonDeleteFromWatchlist**, change the text inside the **Label** to **Remove from watchlist**.



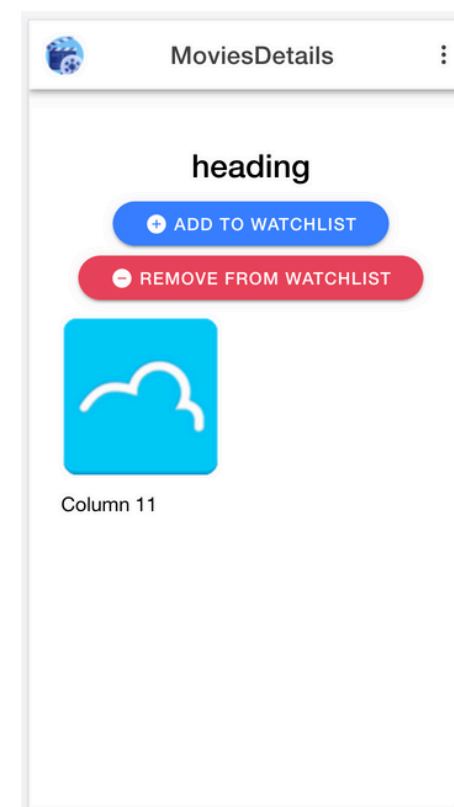
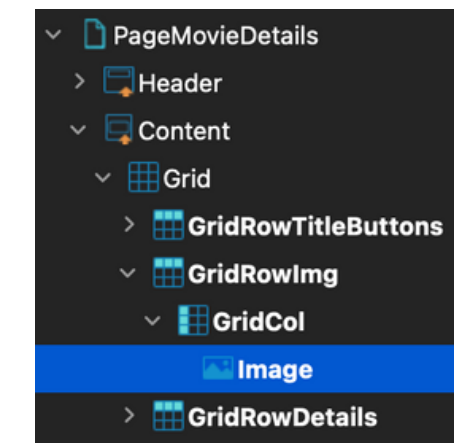
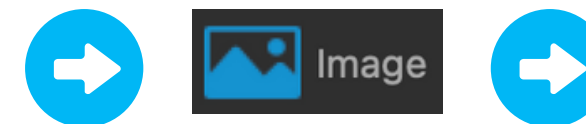
8.5 Add the PageMovieDetails layout

In **PageMovieDetails**,
rename the **GridRows** to:

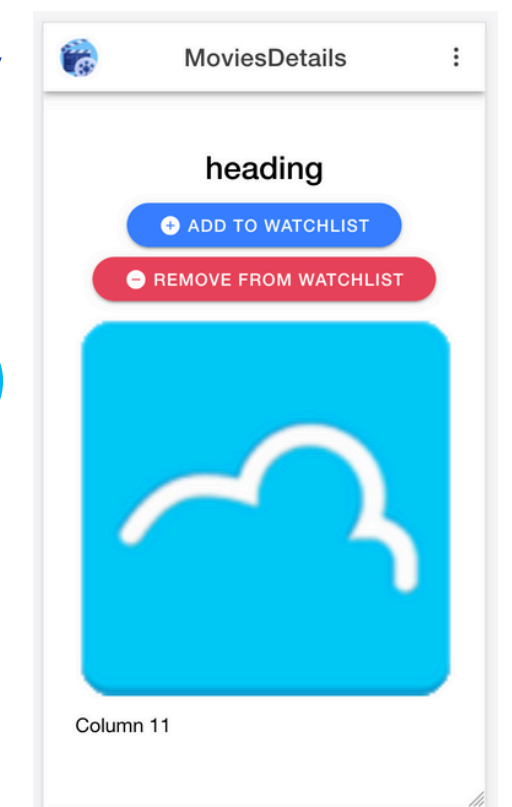
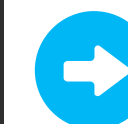
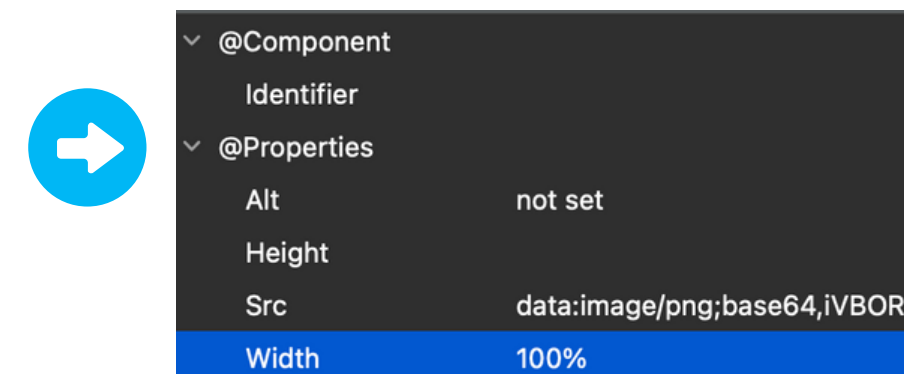
- **GridRowTitleButtons** for the first
- **GridRowImg** for the second
- **GridRowDetails** for the third



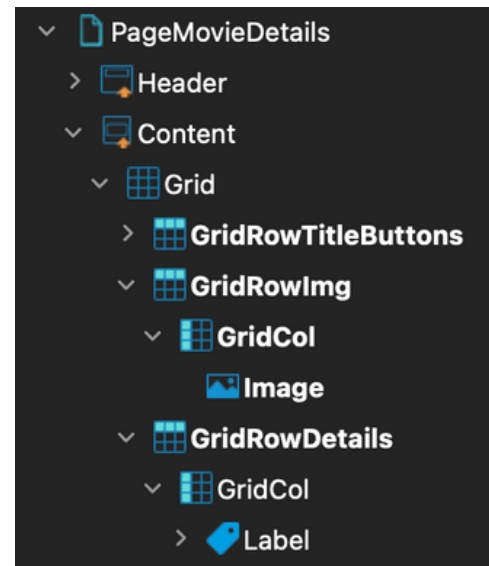
In **GridRowImg**,
remove the **Label** and
add an **Image** component.



In the **Image** component's properties,
set the **width** property to **100%**.



8.5 Add the PageMovieDetails layout



In **GridRowDetails**,
remove the **Label**.



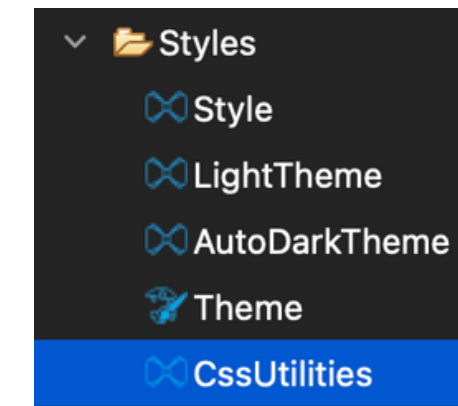
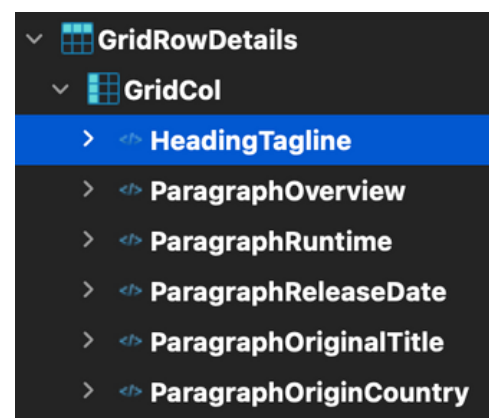
`</>` Heading

`</>` Paragraph

Add an **Heading**
and rename it **HeadingTagline**.

Add 5 **Paragraphs** and rename them

- ParagraphOverview
- ParagraphRuntime
- ParagraphReleaseDate
- ParagraphOriginalTitle
- ParagraphOriginCountry



In **CssUtilities**,
add the following classes.

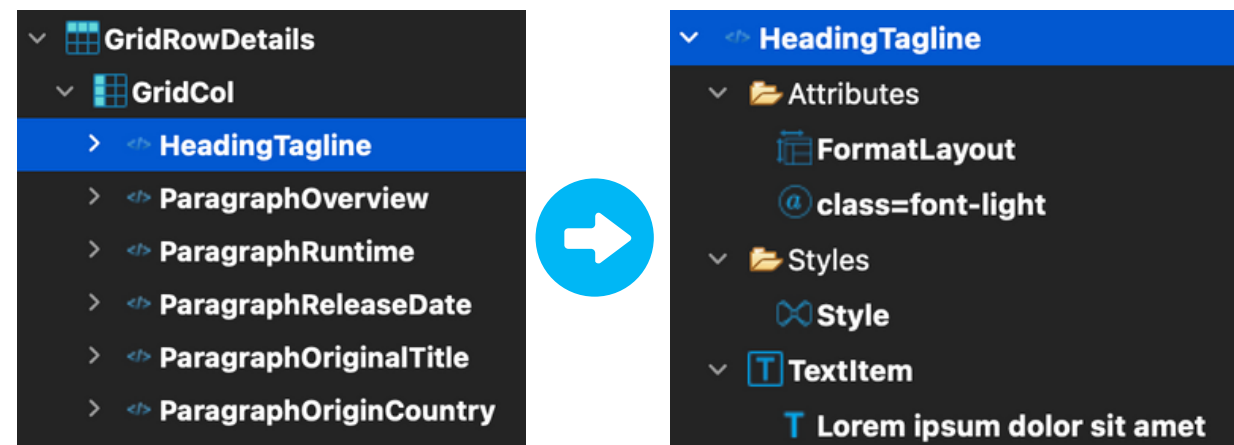
```
.flex {  
  display: flex;  
}
```

```
.font-semibold {  
  font-weight: 600;  
}  
  
.font-light {  
  font-weight: 300;  
}
```



8.5 Add the PageMovieDetails layout

Let's have a look on HeadingTagline.



To get this result...

Lorem ipsum dolor sit amet

In **HeadingTagline's** properties, set the **Tag name** property to **h3**.

> < HeadingTagline

> @Component

Identifier

Tag name h3

In **HeadingTagline**

Add a **Style** component.

In **Style.scss**, add **font-style: italic;**



Add an **Attr** component.

Set it to **class=font-light**.

We could have used an **Attr** for both styles.

The **Style component** is used only for demonstration purposes .

Add a **FormatLayout**.

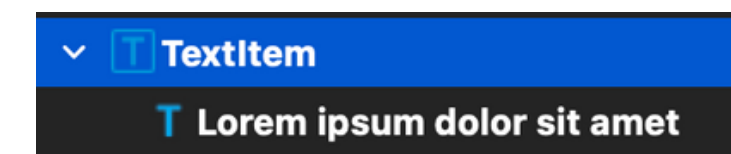
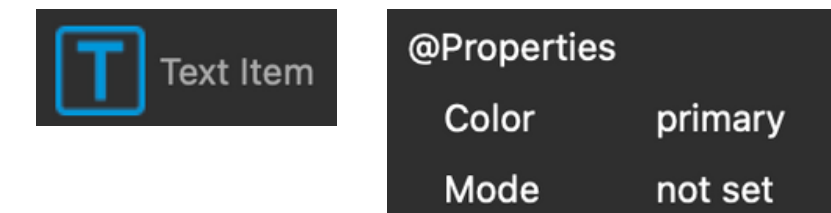
In its **@Text Alignment** properties, set the **Alignment** property to **ion-text-center**.

FormatLayout

> @Text Alignment

Alignment ion-text-center

Alignment 0-576 px (x not set)



Add a **Text** item.

Set its **Color** property to **primary**.

Set its text to a short sentence.



8.5 Add the PageMovieDetails layout

Lorem ipsum dolor sit amet

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis sagittis consequat molestie. Nunc scelerisque massa non nulla aliquam dignissim.

To get this result...

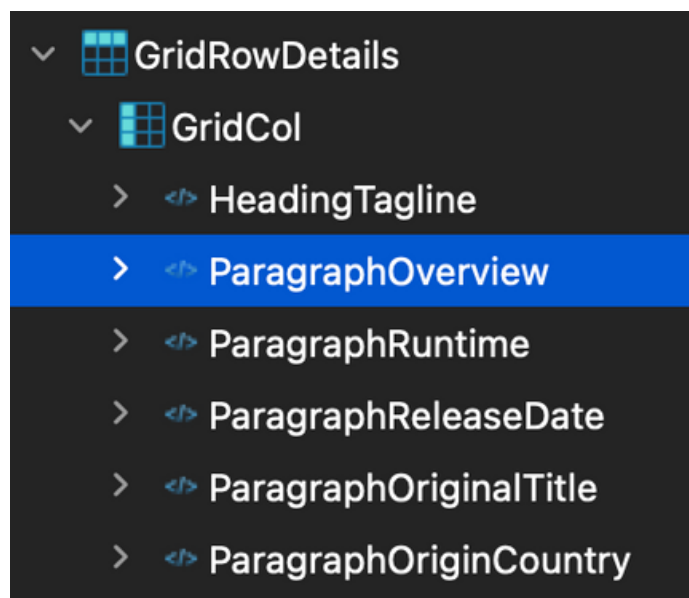
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis sagittis consequat molestie. Nunc scelerisque massa non nulla aliquam dignissim.

Let's have a look on **ParagraphOverview**.



In **ParagraphOverview**,
replace the text by some Lorem Ipsum

▼ **ParagraphOverview**
T Lorem ipsum dolor sit amet, consectetur adipiscing elit.



8.5 Add the PageMovieDetails layout

Lorem ipsum dolor sit amet

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis sagittis consequat molestie. Nunc scelerisque massa non nulla aliquam dignissim.

Runtime:

120 min

To get this result...

Runtime:

120 min

ParagraphRuntime

Attributes

FormatLayout

class=flex

TextItem

Runtime:

TextItem1

Attributes

class=font-semibold

120 min

In ParagraphRuntime

ParagraphRuntime



Add a FormatLayout.

In its @Flex Alignment properties, set the Container property to **ion-justify-content-between**.

FormatLayout

@Flex Alignment	
Container	ion-justify-content-between
Item	not set



Add an Attr component.

Set it to **class=flex**.

ParagraphRuntime

Attributes

FormatLayout

class=flex

Add 2 Text items.

TextItem

For the first one, set its **Color** property to **primary**. Set its text to **Runtime**.

TextItem

Runtime:

@Properties

Color	primary
Mode	not set

For the second one, Add an **Attr** component and set it to **class=font-semibold**. Set its text to 120 min.

TextItem1

Attributes

class=font-semibold

120 min



convertigo

8.5 Add the PageMovieDetails layout

Lorem ipsum dolor sit amet

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis sagittis consequat molestie. Nunc scelerisque massa non nulla aliquam dignissim.

Runtime:

120 min

Release date:

2009-12-15

Original Title:

Title

Origin Country:

US, FR

Runtime:

120 min

Release date:

2009-12-15

Original Title:

Title

Origin Country:

US, FR

- Now, let's apply the same layout as ParagraphRuntime in
- ParagraphReleaseDate
 - ParagraphOriginalTitle
 - ParagraphOriginCountry

GridRowDetails

GridCol

HeadingTagline

ParagraphOverview

ParagraphRuntime

ParagraphReleaseDate

ParagraphOriginalTitle

ParagraphOriginCountry

ParagraphOriginCountry

Attributes

FormatLayout

class=flex

TextItem

Origin Country:

TextItem1

Attributes

class=font-semibold

US, FR

ParagraphOriginalTitle

Attributes

FormatLayout

class=flex

TextItem

Original Title:

TextItem1

Attributes

class=font-semibold

Title

ParagraphReleaseDate

Attributes

FormatLayout

class=flex

TextItem

Release date:

TextItem1

Attributes

class=font-semibold

2009-12-15



MoviesDetails

heading

ADD TO WATCHLIST

REMOVE FROM WATCHLIST

Lorem ipsum dolor sit amet

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis sagittis consequat molestie. Nunc scelerisque massa non nulla aliquam dignissim.

Runtime:

120 min

Release date:

2009-12-15

Original Title:

Title

Origin Country:

US, FR



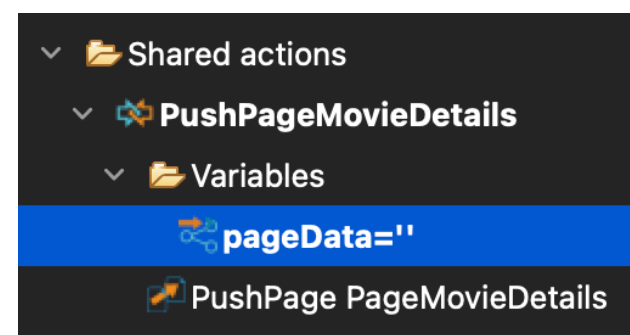
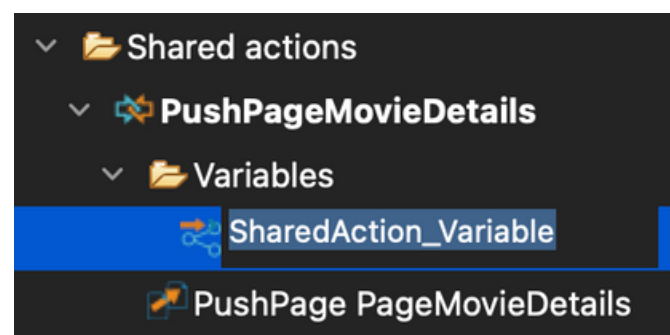
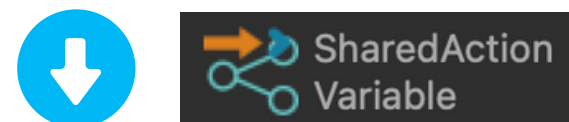
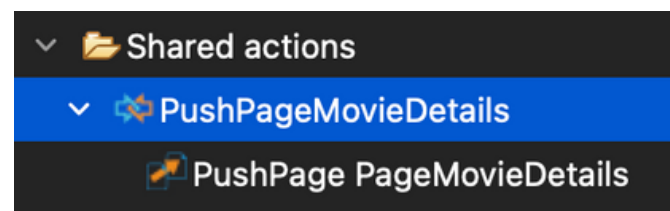
8.6 Add a Shared action variable

As for now, In **PageMovieDetails**, we only display static data.

So when we push the page, we need to pass data, like the movie Id, from the previous page to **PageMovieDetails**

For that, we are going to use a **SharedAction Variable**.

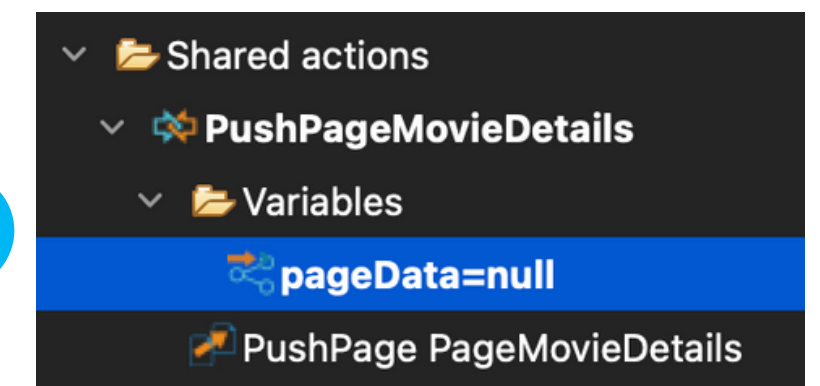
In the **PushPageMovieDetails SharedAction**, add a **SharedAction Variable** and rename it **pageData**.



Base properties	
Comment	
Is active	true
Variable value	"



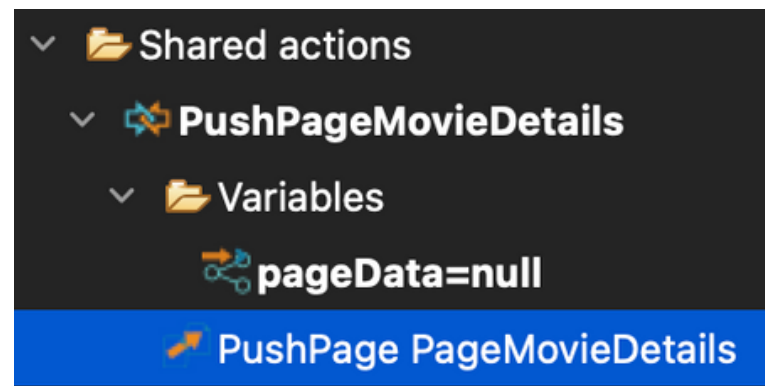
Base properties	
Comment	
Is active	true
Variable value	null



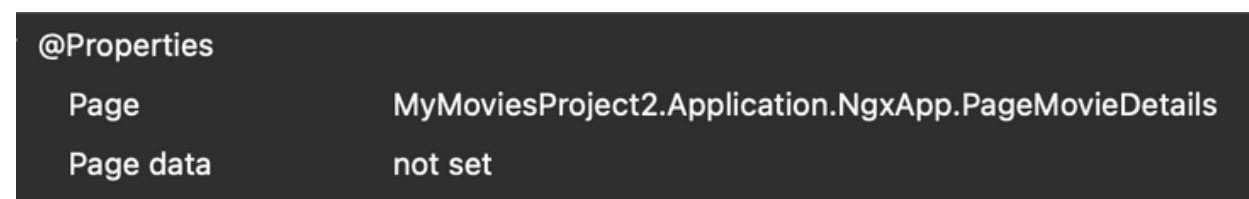
In **pageData's** properties, set the **Variable value** to **null**.



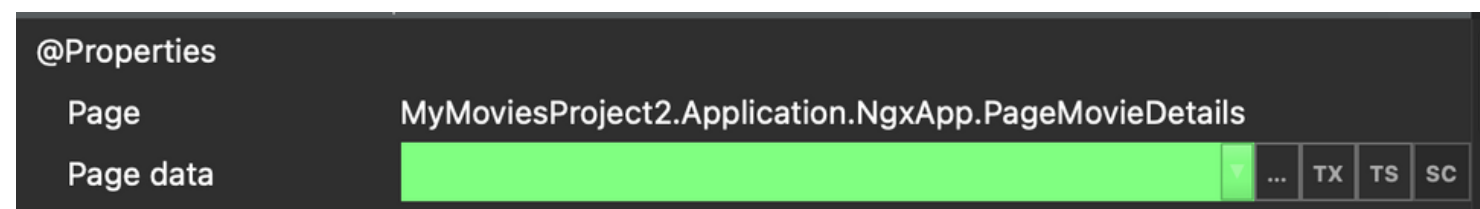
8.6 Add a Shared action variable



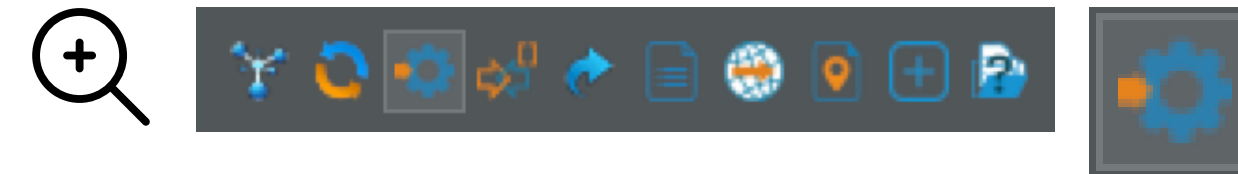
In the PushPage PageMovieDetails's properties



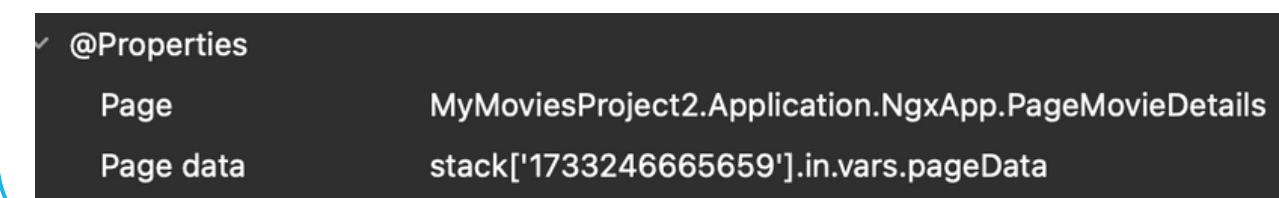
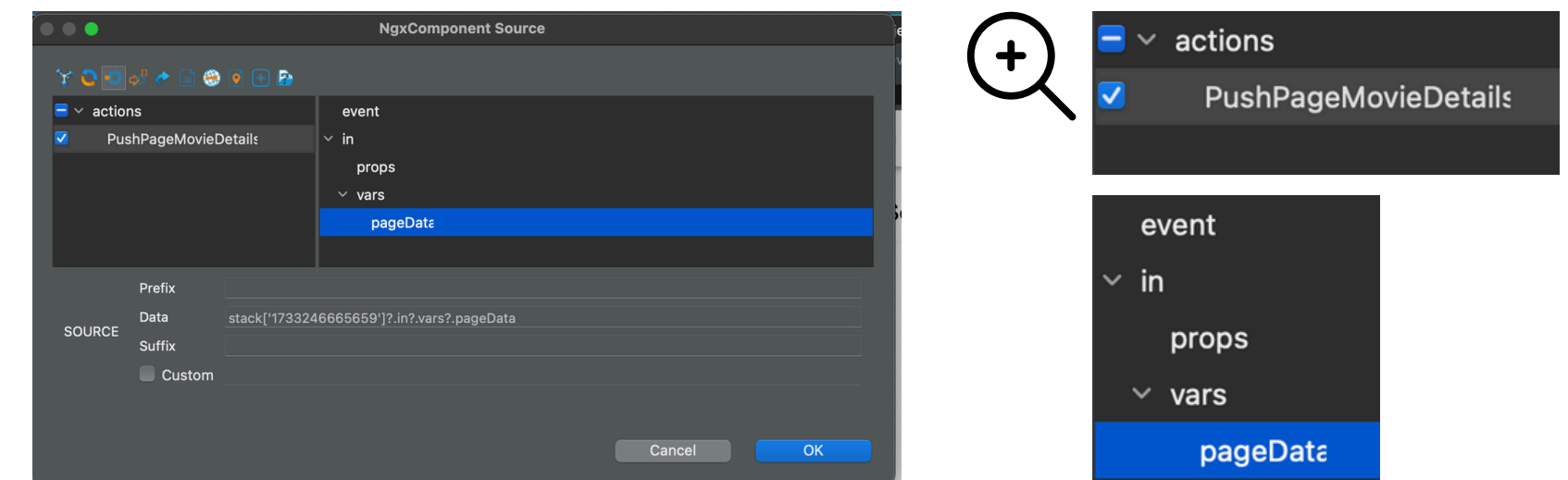
In the Page data property, click on ... to open the NgxComponent Source panel.



In the NgxComponent Source panel, click on the CustomAction Icon to display actions as sources.

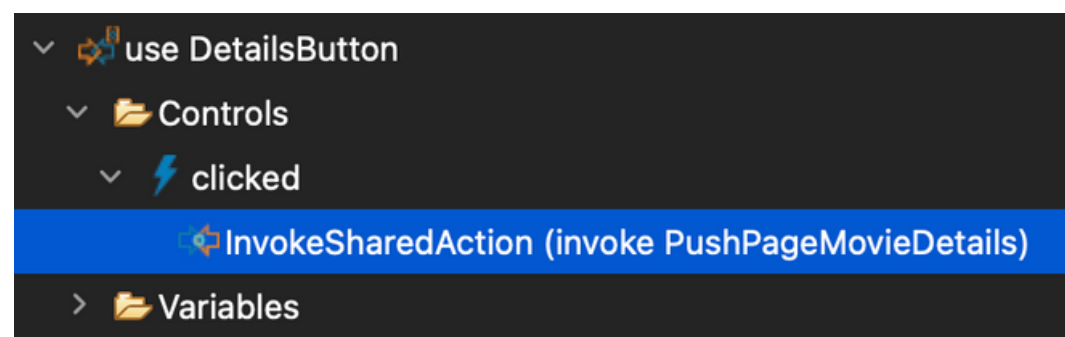
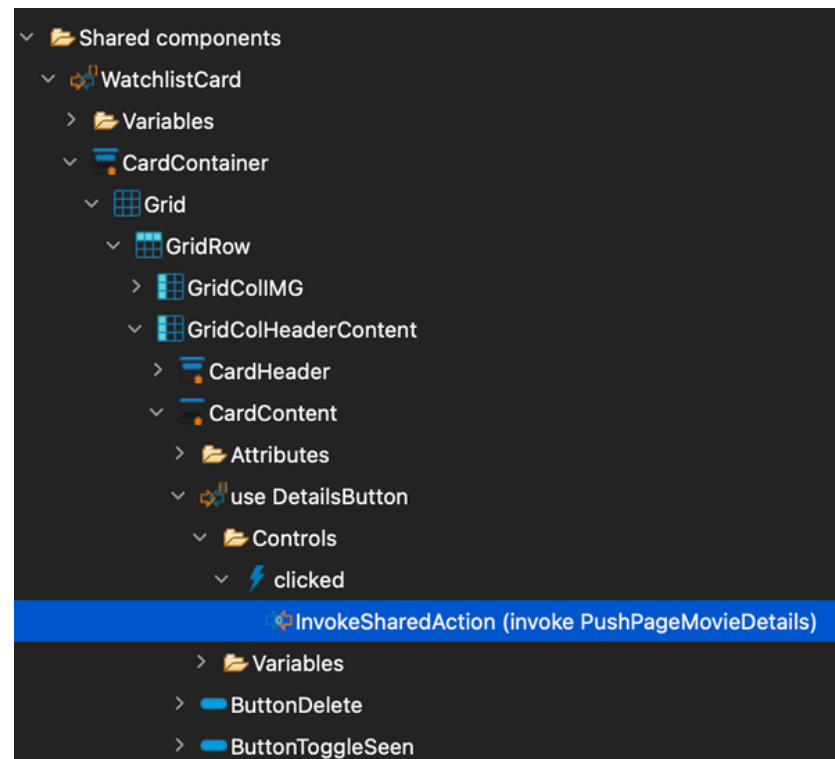


Click on PushPageMovieDetails, then on in, then on vars, then select pageData.

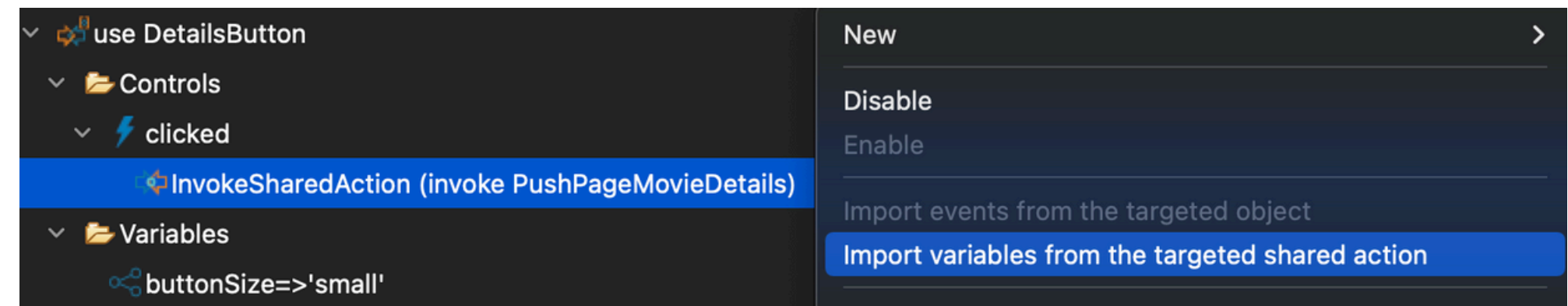


8.7 Use a Shared action variable

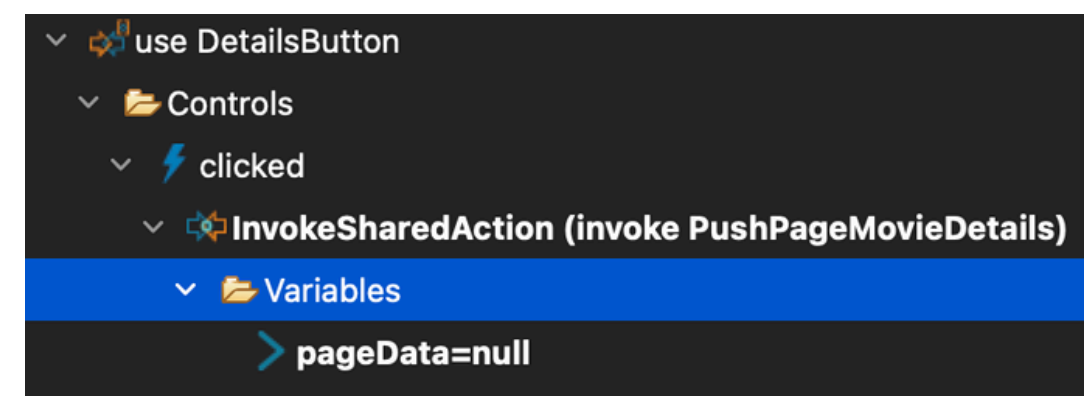
In the **WatchlistCard** shared component,
in use **DetailsButton**



Right-click on the **invoke PushPageMovieDetails InvokeSharedAction**,
and select **Import variables from the targeted shared action**.



The **variable pageData** has been imported.



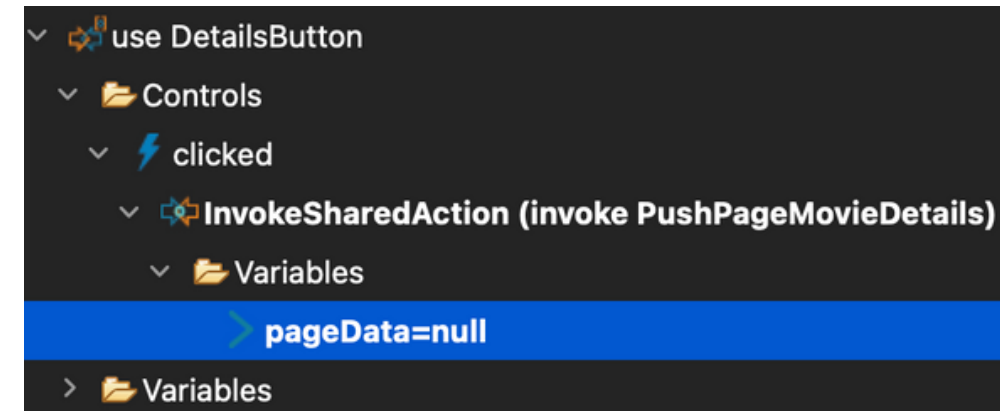
8.7 Use a Shared action variable

To set the **pageData** Var value,
let's have a look at the data structure
of the **movies** object in **pageWatchlist**.

```
vars.movies
▼ Array(3) [ {...}, {...}, {...} ]
  ▼ 0: Object { id: 1, order:
    "1.000000000000000000", title: "Gremlins", ... }
    genres: null
    id: 1
    img_url: "https://image.tmdb.org/t/p/w500/gDSfs0
    hqAOVJ9jIHbX3AJ3StTVG.jpg"
    movie_id: "927"
    order: "1.000000000000000000"
    original_title: "Gremlins"
    overview: "When Billy Peltzer is given a strange
    but adorable pet named Gizmo for Christmas, he
    inadvertently breaks the three important rules
    of caring for a Mogwai, unleashing a horde of
    mischievous gremlins on a small town."
    release_date: "1984-06-08"
    release_year: "1984"
    title: "Gremlins"
    watched: true
    ▶ <prototype>: Object { ... }
  ▶ 1: Object { id: 2, order:
    "2.000000000000000000", title: "Gattaca", ... }
  ▶ 2: Object { id: 3, order:
    "3.000000000000000000", title: "Avatar", ... }
    length: 3
  ▶ <prototype>: Array []
```



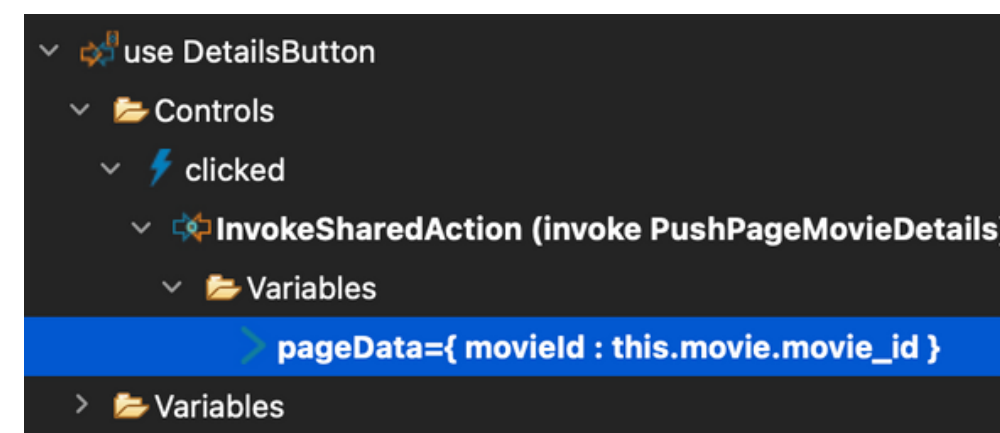
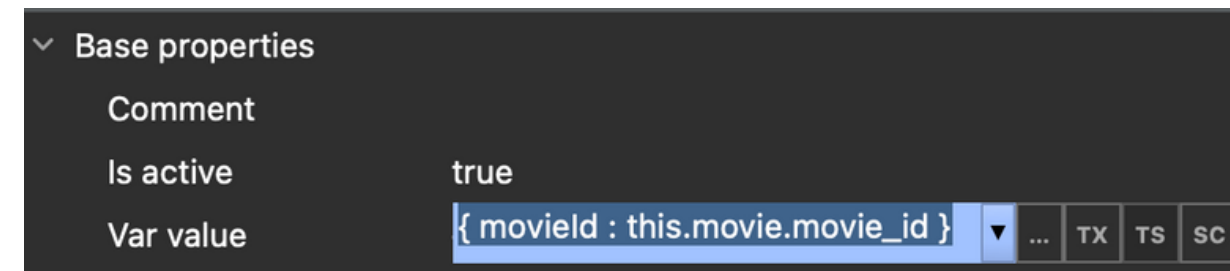
 **movie_id: "927"**



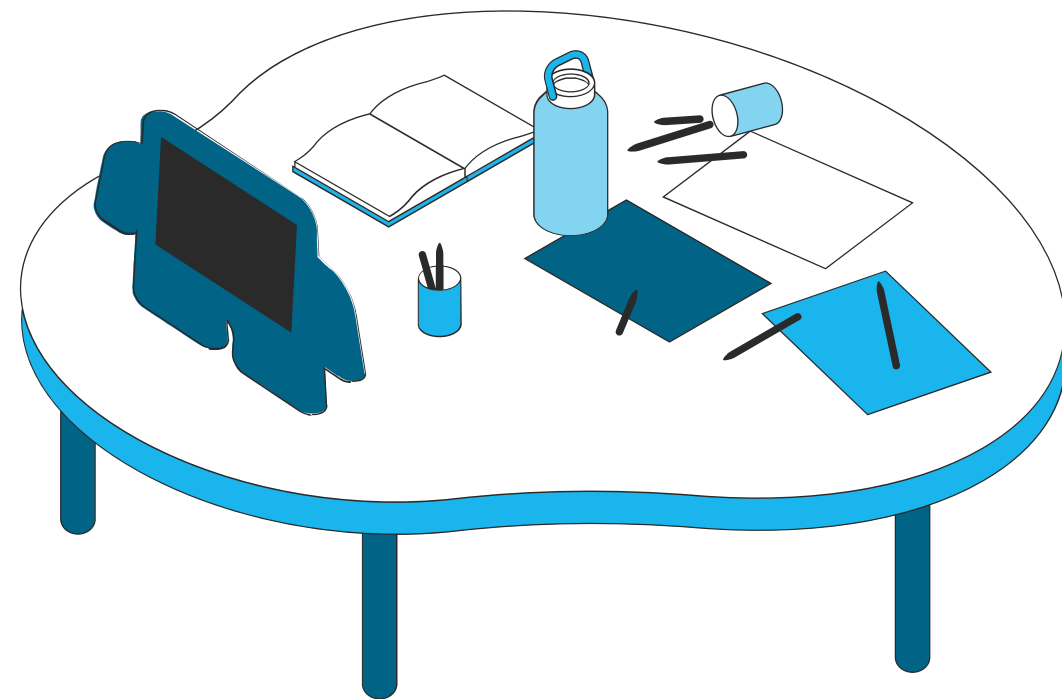
Base properties	
Comment	
Is active	true
Var value	null



Enter **{ movieId : this.movie.movie_id }** in TS scope
as **pageData** Var value,



9 – Finalize data binding



9.1 Retrieve data in a page

9.2 Display dynamic data in a page

9.3 Use a NgIf directive to display elements

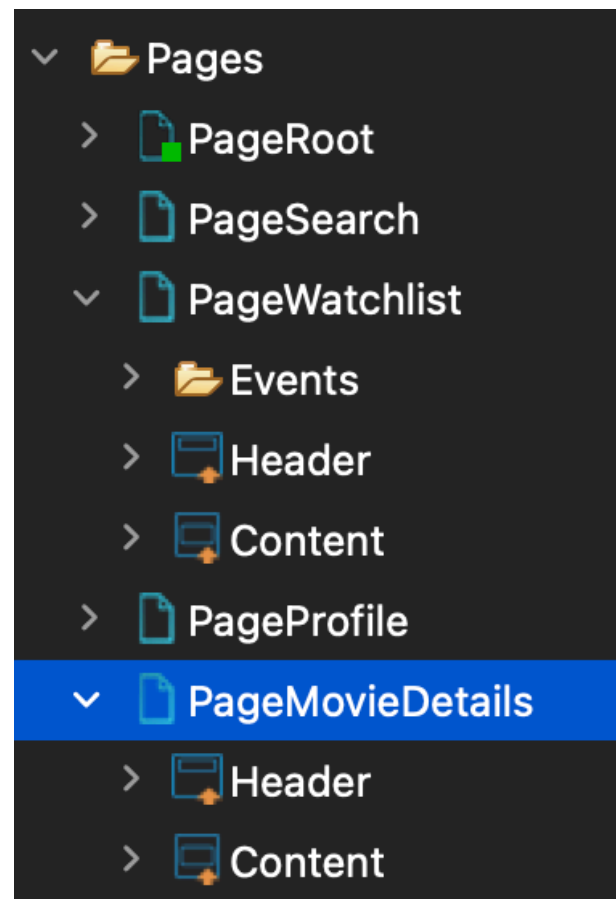
9.4 Implement ButtonAddToWatchlist

9.5 Implement ButtonDeleteFromWatchlist

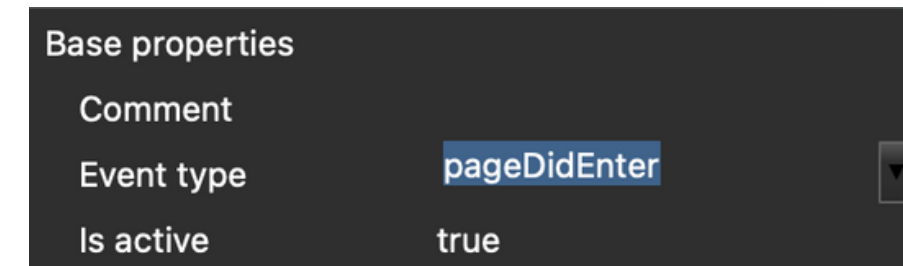
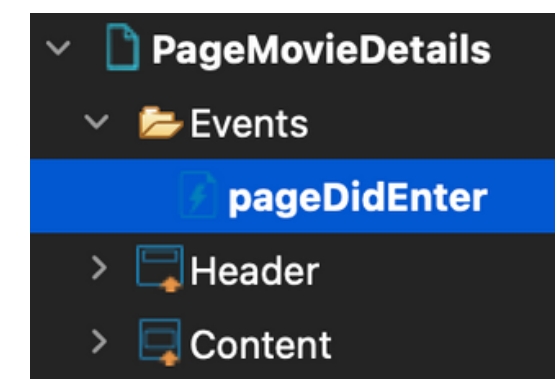
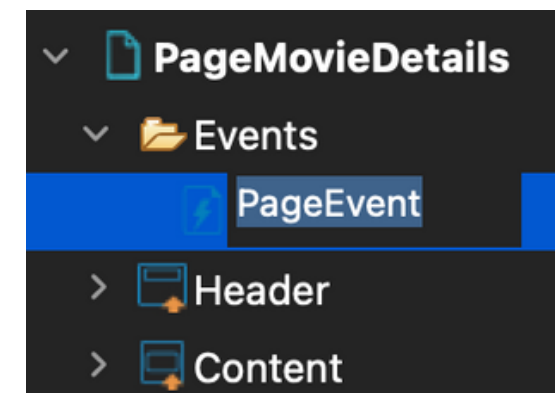
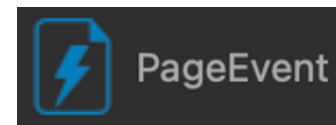
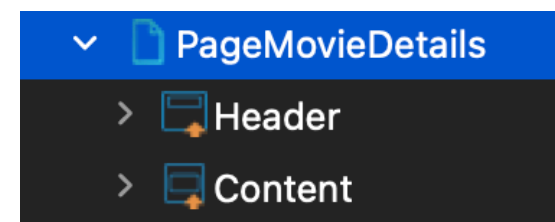
9.6 Implement ButtonToggleSeen

9.7 xxx

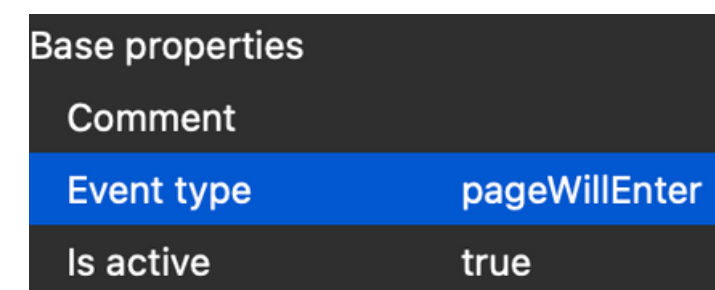
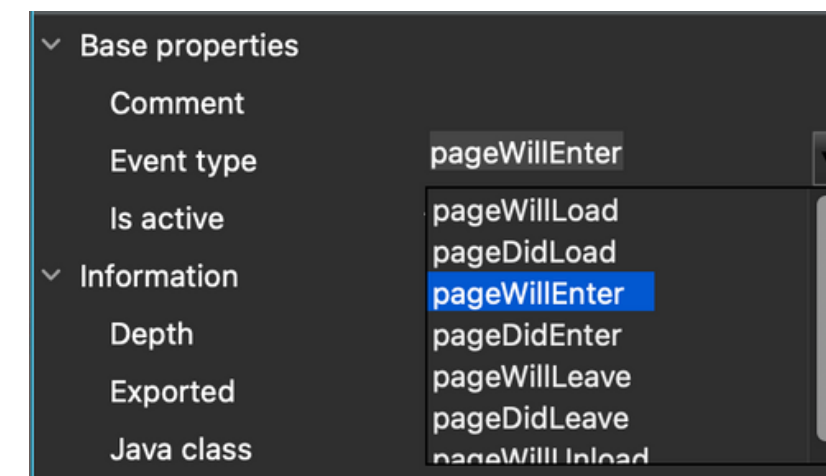
9.1 Retrieve data in a page



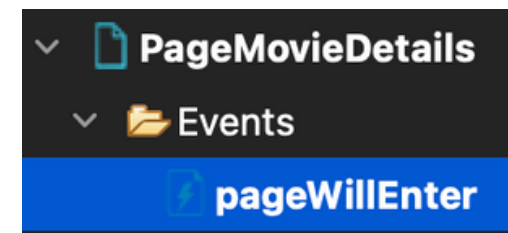
In PageMovieDetails,
add a PageEvent.



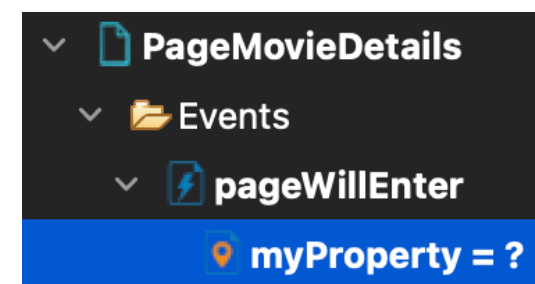
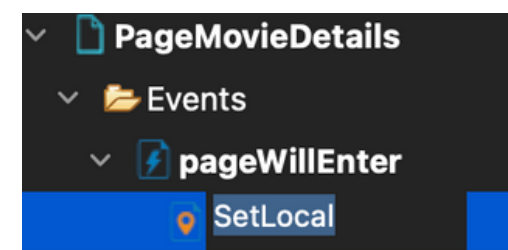
In the PageEvent's properties,
set the Event type property value
to pageWillEnter.



9.1 Retrieve data in a page



In the **pageWillEnter** pageEvent, add a **SetLocal** component.



myProperty = ?

In the **SetLocal's** properties,

@Properties	
Property	myProperty
Value	

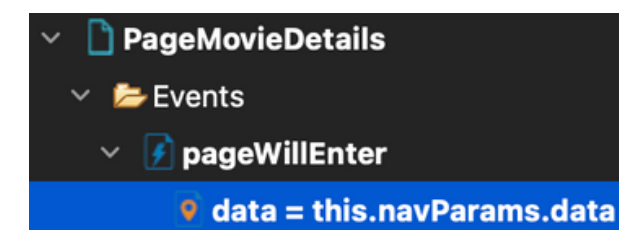
Set **data** as value



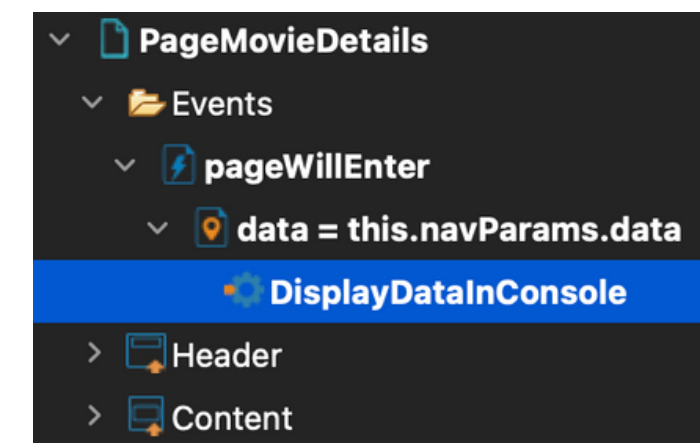
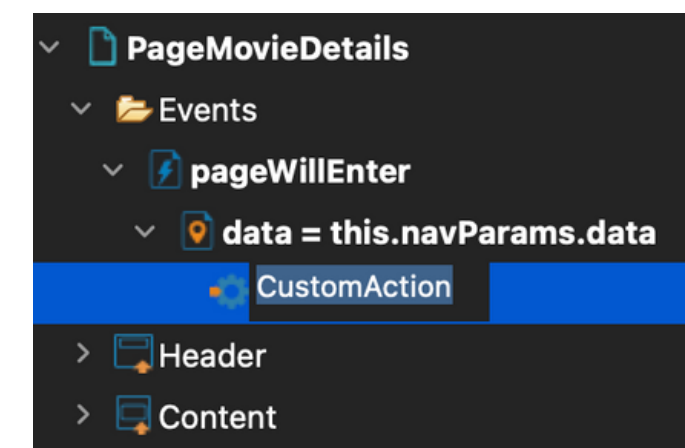
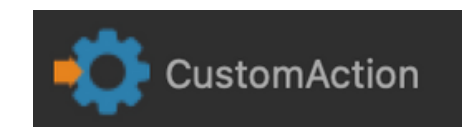
for the **Property** property, set **this.navParams.data**.

as **value** for the **Value** property.

@Properties	
Property	data
Value	this.navParams.data

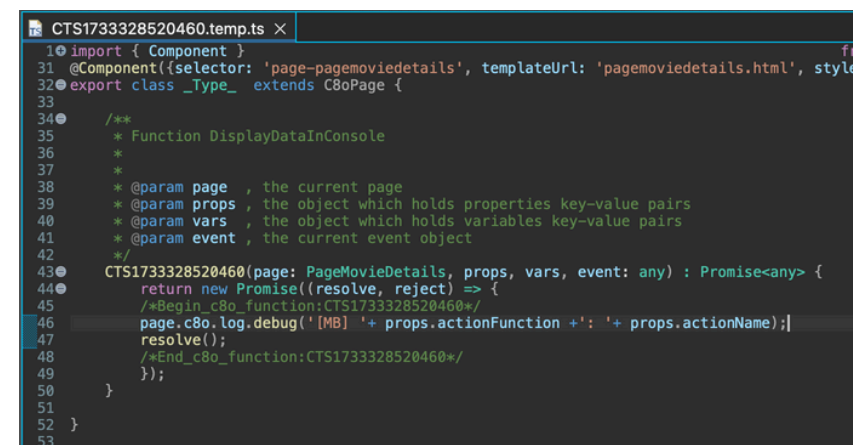
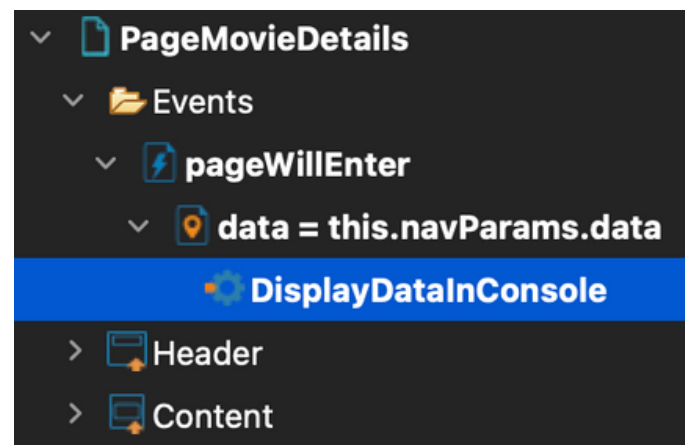


In the **data SetLocal**, add a **CustomAction** component, and rename it **DisplayDataInConsole**.

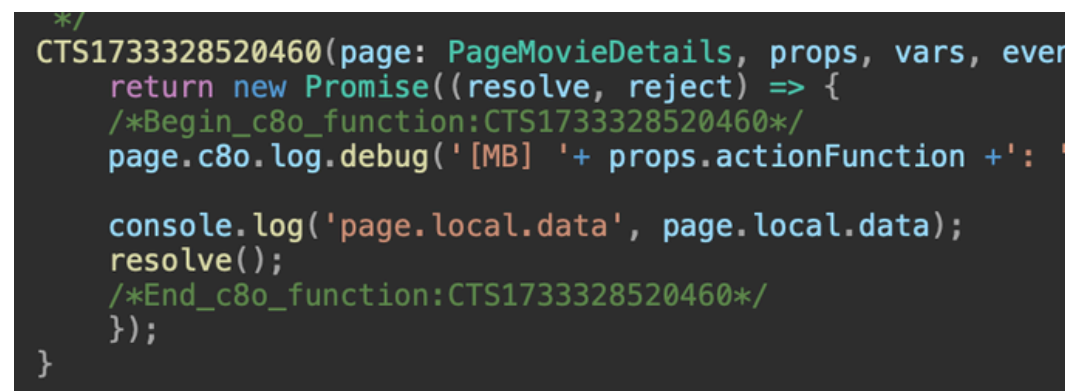


9.1 Retrieve data in a page

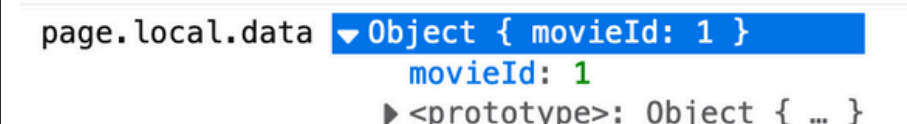
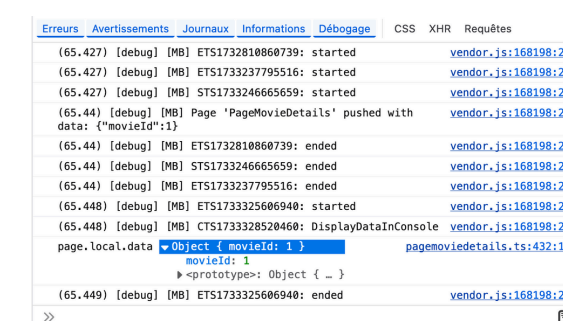
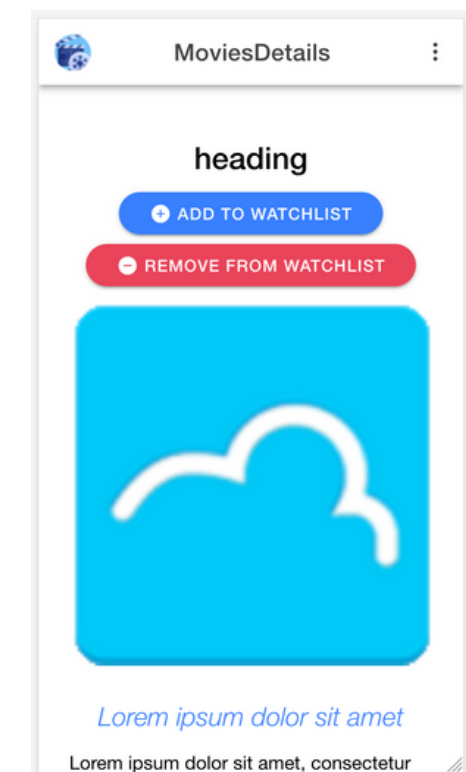
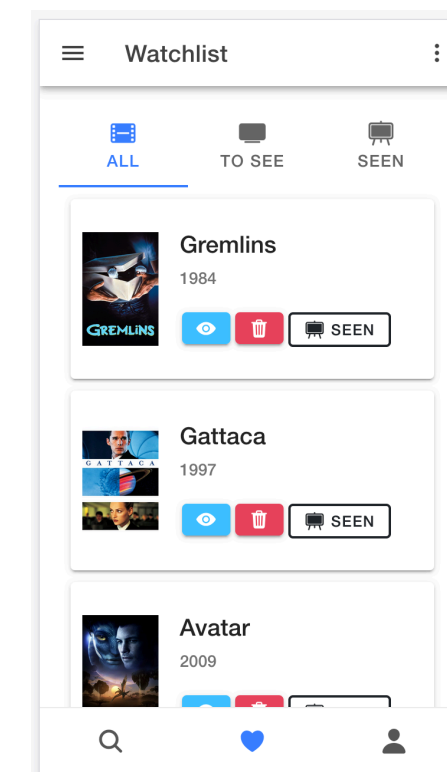
Double-click on **DisplayDataInConsole** to open the CTS file.



In the CTS file,
add `console.log('page.local.data', page.local.data);`



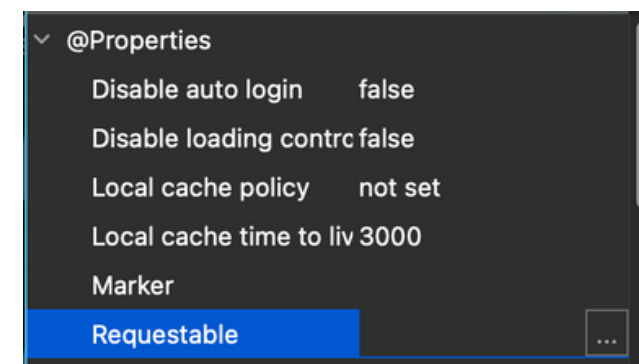
In **pageWatchlist**,
when we click on the **DetailButton** in a **WatchlistCard**,
the **MovieDetailsPage** is pushed
and **page.local.data** appears in the **browser console**.



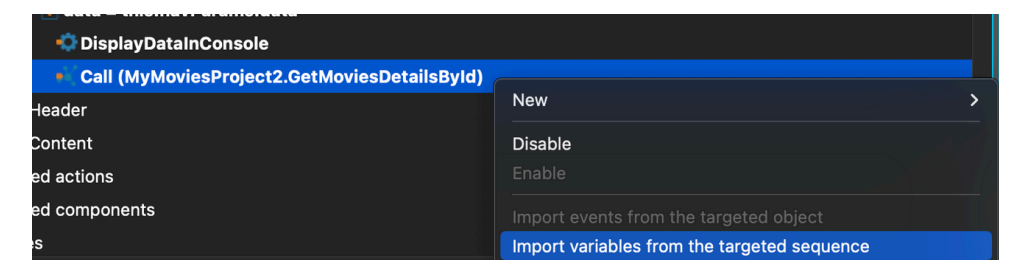
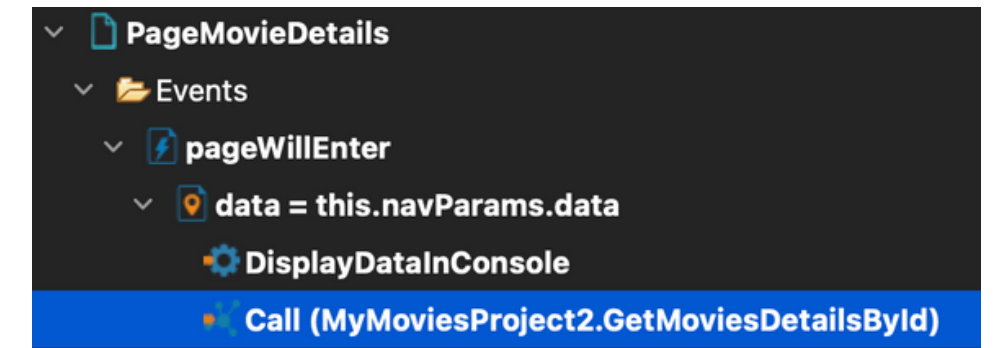
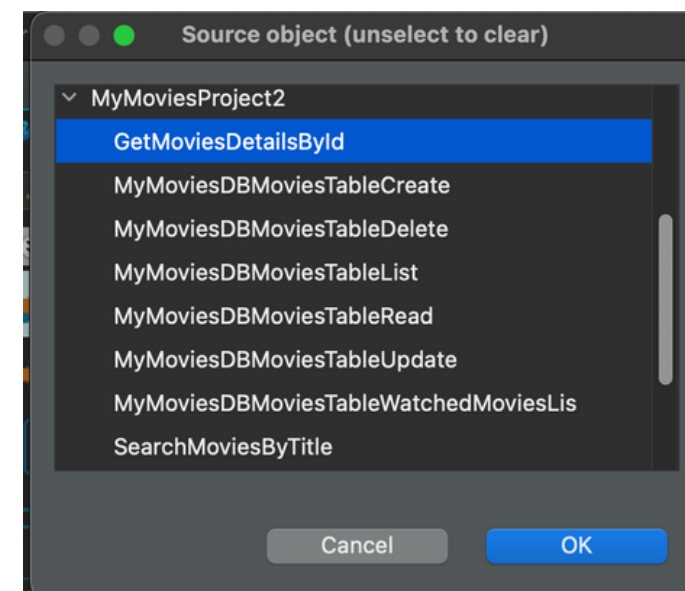
9.1 Retrieve data in a page

In the **DisplayDataInConsole CustomAction**,
add a **CallSequence** component.

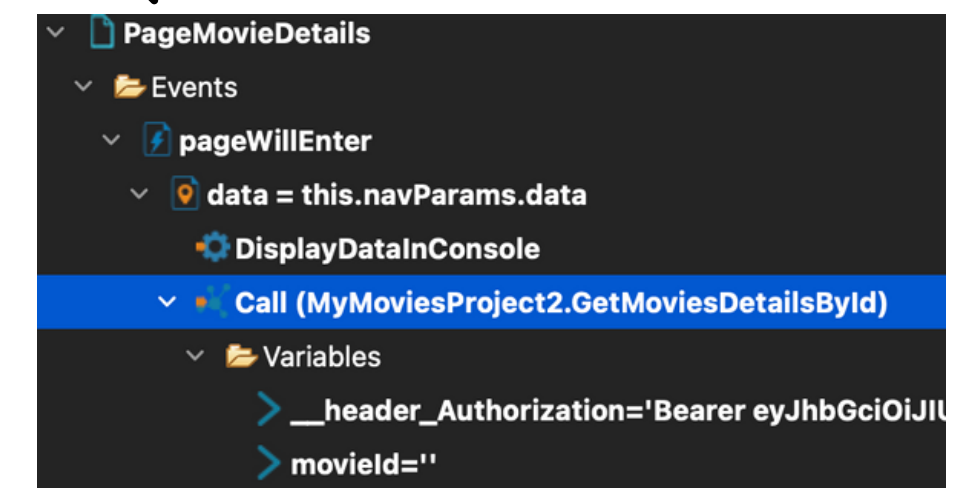
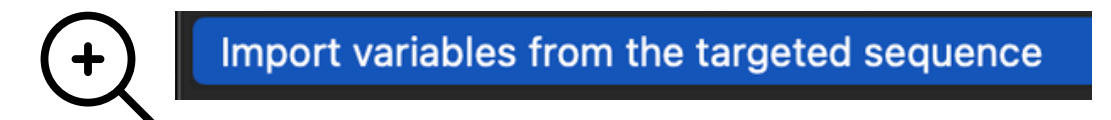
In the **CallSequence** component,
click on ... in the **Requestable** property
to open the **Source** object panel.



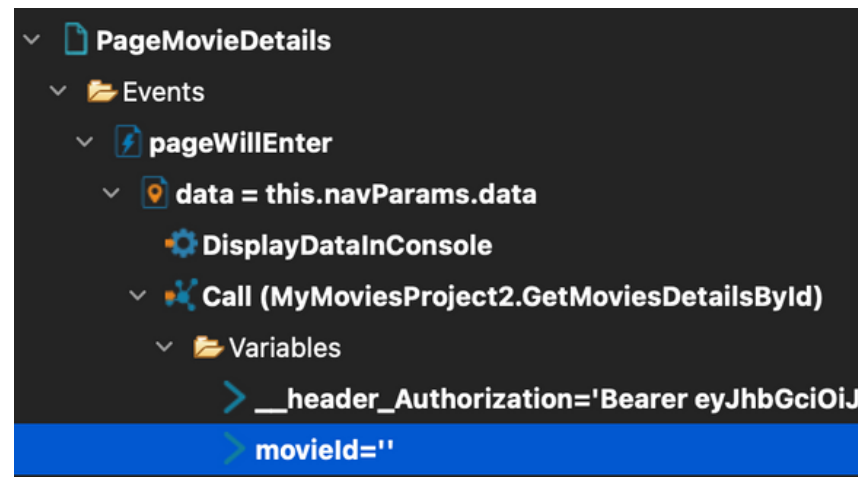
In the **Source** object panel,
select **GetMoviesDetailsById**
as **source**.



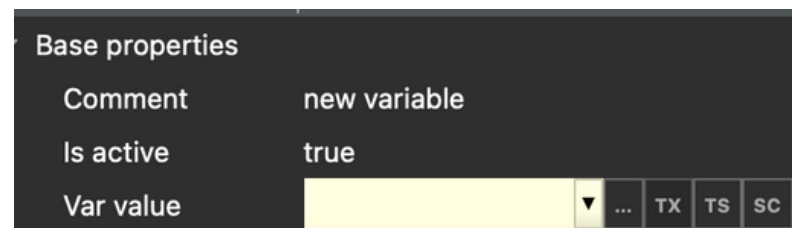
Right-click on the **CallSequence**,
and select **Import variables**
from the targeted sequence.



9.1 Retrieve data in a page



In the **CallSequence** component,
in the **movied** variable,



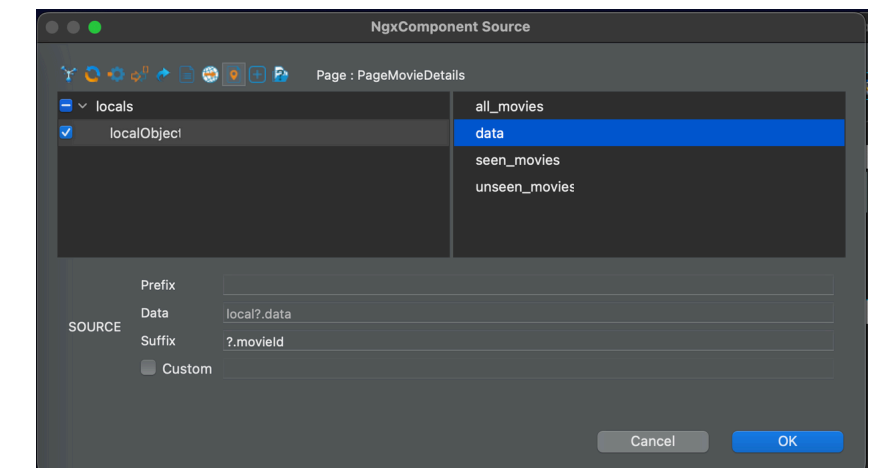
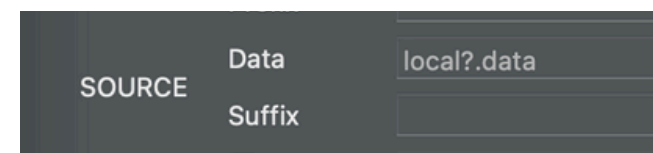
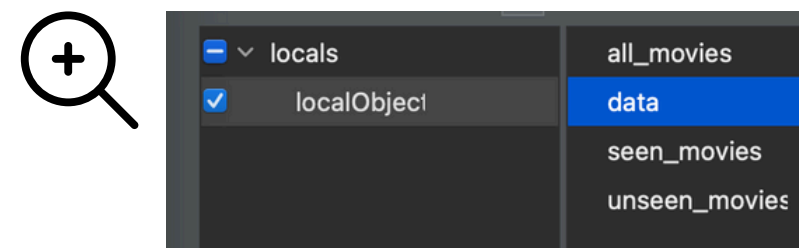
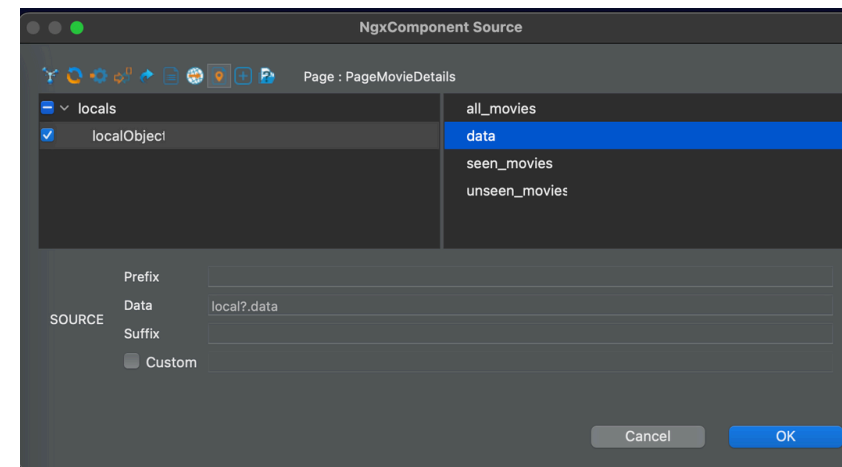
click on ... in the **Var value** property
to open
the **NgxComponent Source** panel.



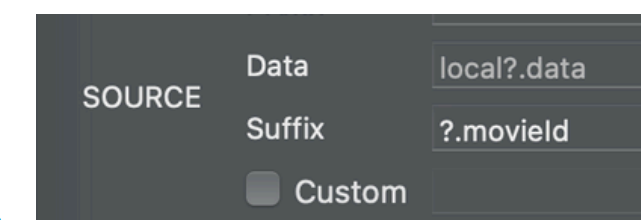
In the **NgxComponent Source** panel,
click on the **SetLocal** Icon to display the **data SetLocal** in the **sources**.



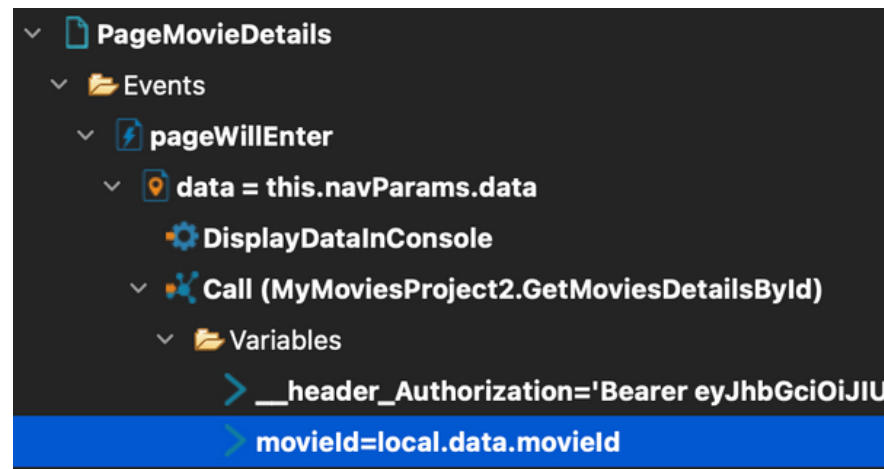
Select the **localObject** data as source.



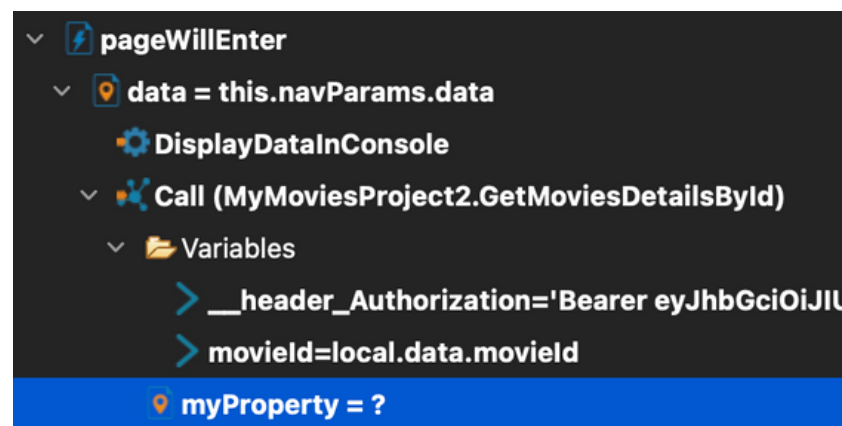
⊕ Add **?..movied** as **suffix**.



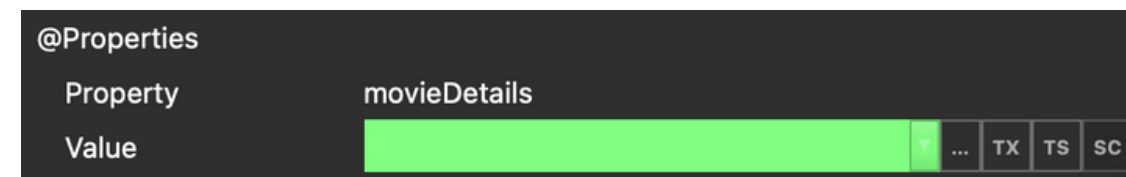
9.1 Retrieve data in a page



In the callSequence, add a SetLocal component.

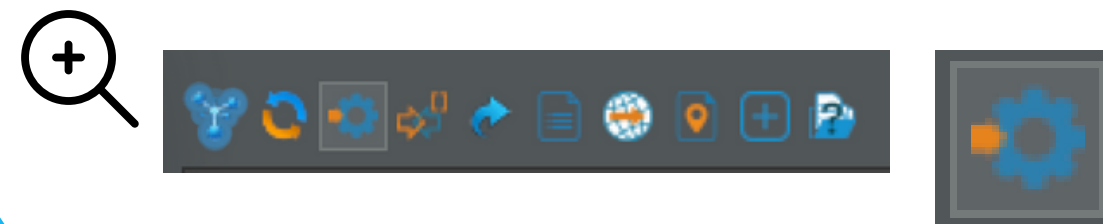


In the SetLocal's properties, Set movieDetails as **value** for the **Property** property

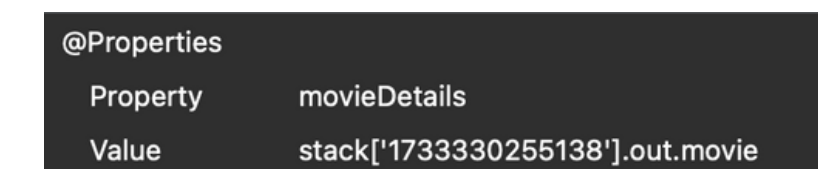
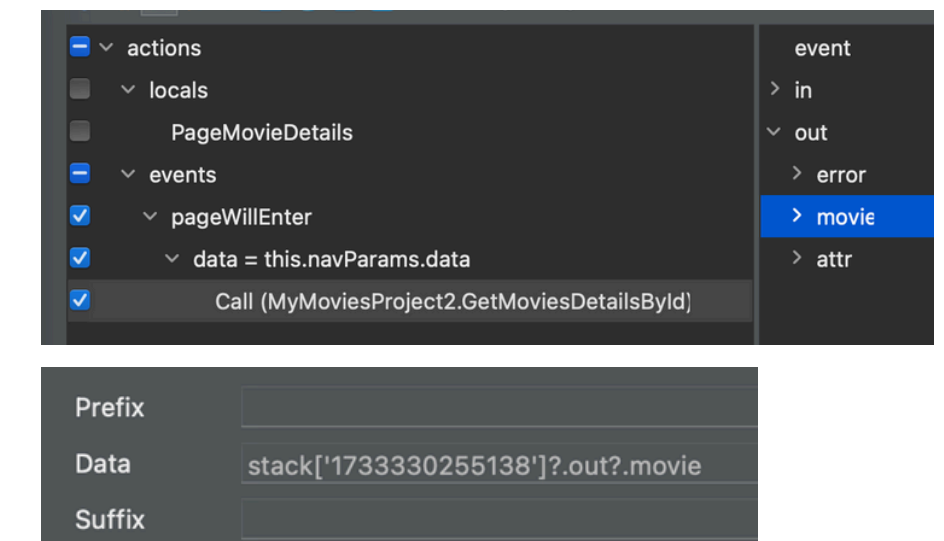
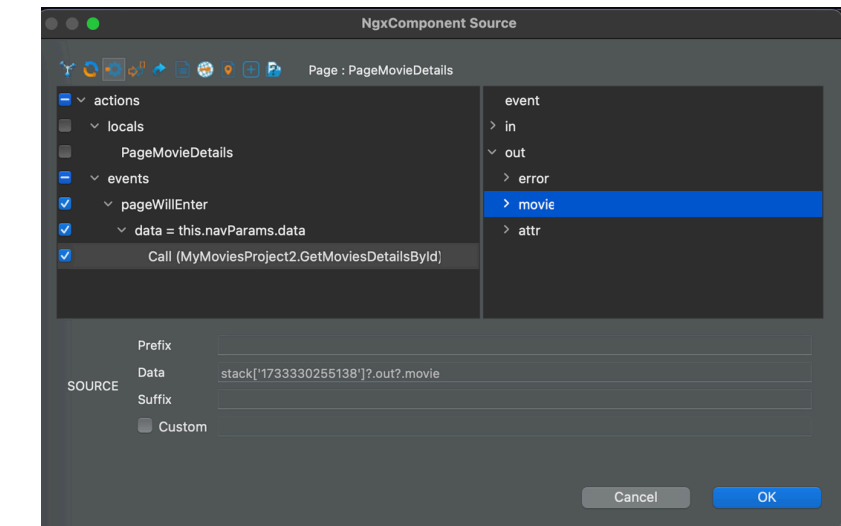


In the **Value** property, click on ... to open the **NgxComponent Source** panel.

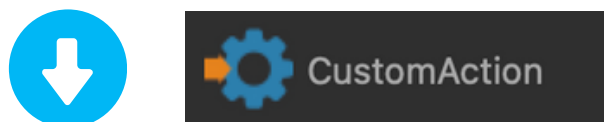
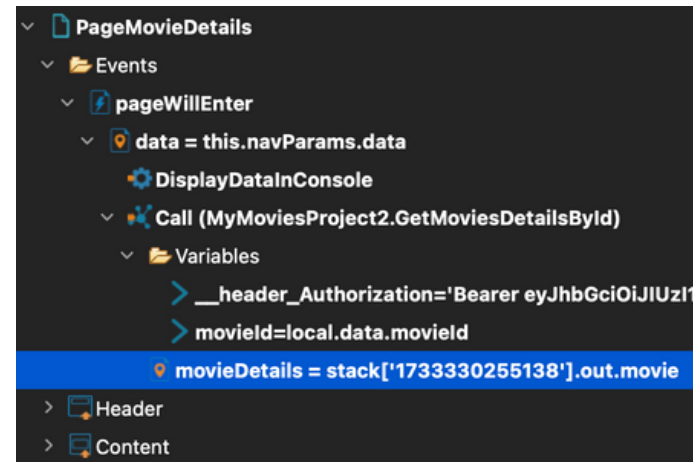
In the **NgxComponent Source** panel, click on the **CustomAction Icon** to display the CallSequence as **source**.



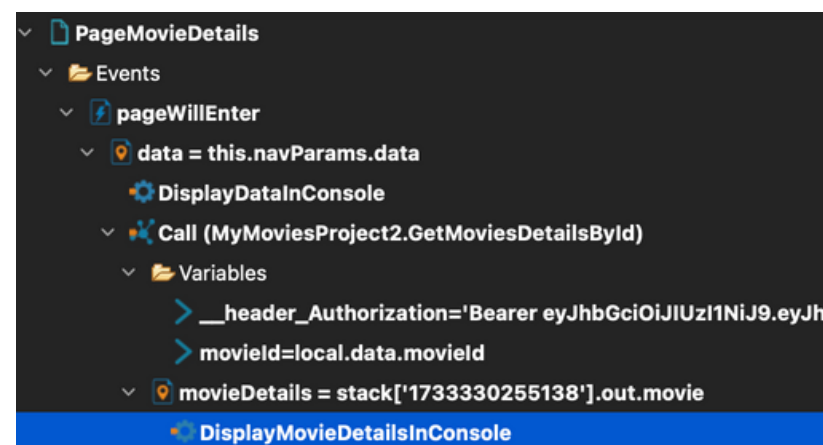
Select the CallSequence as **source**.



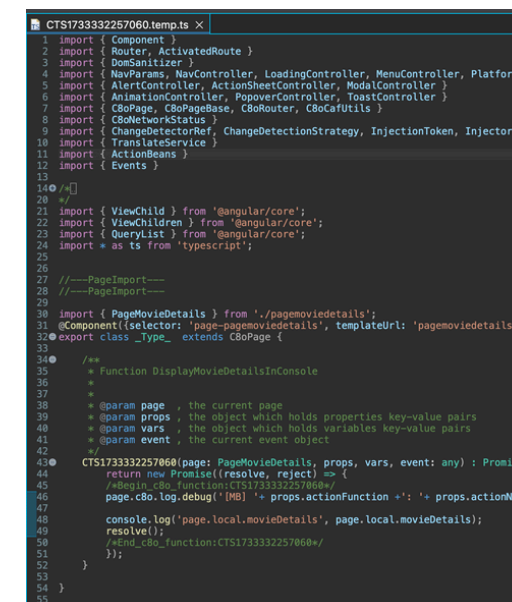
9.1 Retrieve data in a page



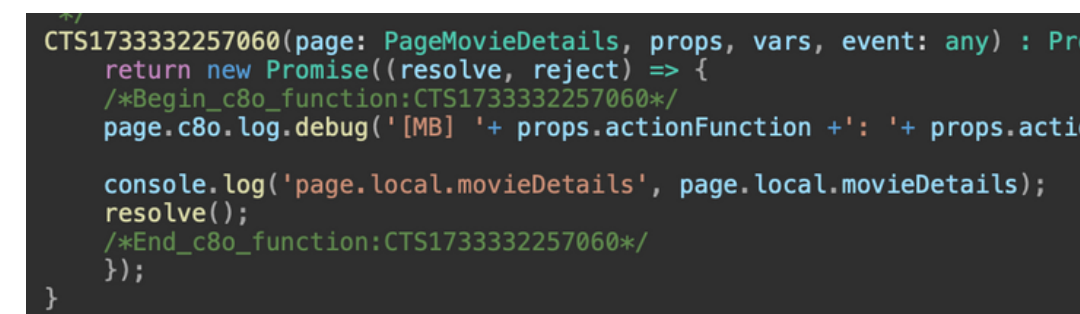
In the movieDetails SetLocal, add a CustomAction component, and rename it DisplayMovieDetailsInConsole.



Double-click on DisplayMovieDetailsInConsole to open the CTS file.



In the CTS file, add console.log('page.local.movieDetails', page.local.movieDetails);



Now, when the MovieDetailsPage is pushed page.local.movieDetails appears in the browser console.

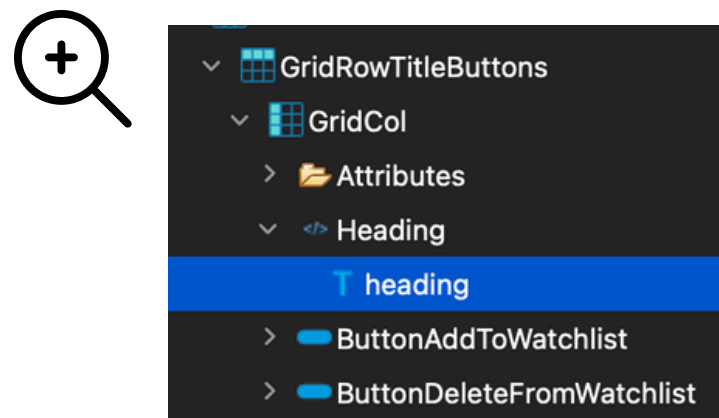
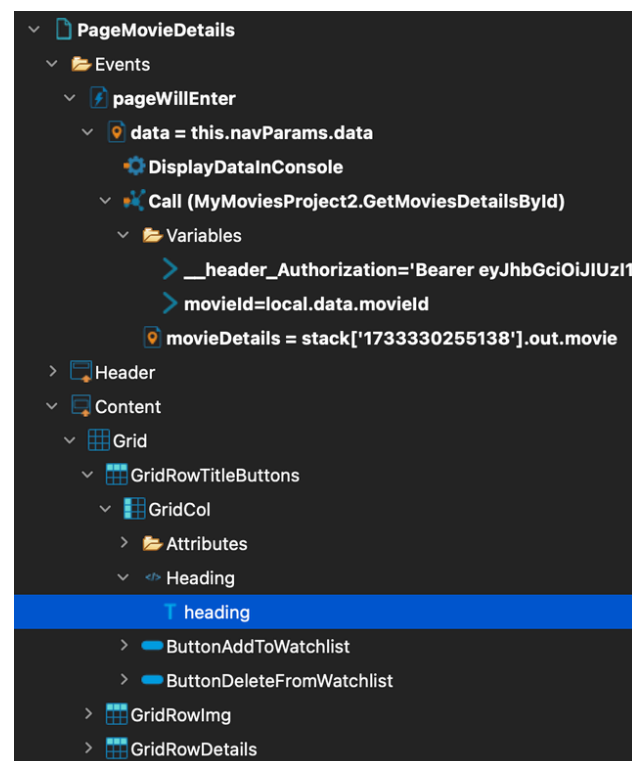
```
page.local.movieDetails
▼ Object { title: "Gremlins", overview: "When Billy Peltzer is given a strange but adorable pet named Gizmo for Christmas, he inadvertently breaks the three important rules of caring for a Mogwai, unleashing a horde of mischievous gremlins on a small town.", tagline: "Cute. Clever. Mischievous. Intelligent. Dangerous.", runtime: "106", original_title: "Gremlins", release_date: "1984-06-08", movie_id: 927, origin_country: (1) [...], img_url: "https://image.tmdb.org/t/p/w500/3iUgvvtB01BuTntPOBcG64kYk6y.jpg" }
img_url: "https://image.tmdb.org/t/p/w500/3iUgvvtB01BuTntPOBcG64kYk6y.jpg"
movie_id: 927
origin_country: Array [ "US" ]
original_title: "Gremlins"
overview: "When Billy Peltzer is given a strange but adorable pet named Gizmo for Christmas, he inadvertently breaks the three important rules of caring for a Mogwai, unleashing a horde of mischievous gremlins on a small town."
release_date: "1984-06-08"
runtime: "106"
tagline: "Cute. Clever. Mischievous. Intelligent. Dangerous."
title: "Gremlins"
<prototype>: Object { ... }
```



9.2 Display dynamic data in a page

Now, let's display the data retrieved in `page.local.movieDetails` in PageMovieDetails.

In the **Heading** of **GridRowTitleButtons**, in the **Text** element's properties, select the **setLocal MovieDetails** as **value** for the **Text value** property.



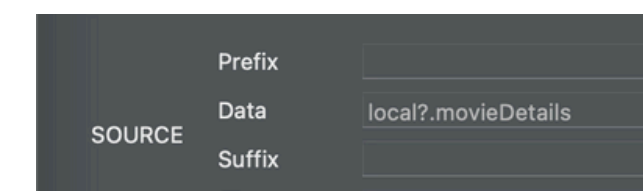
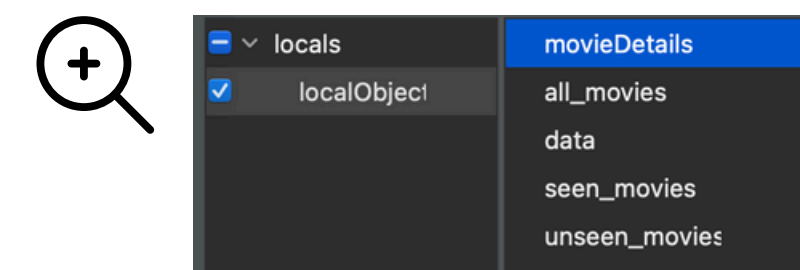
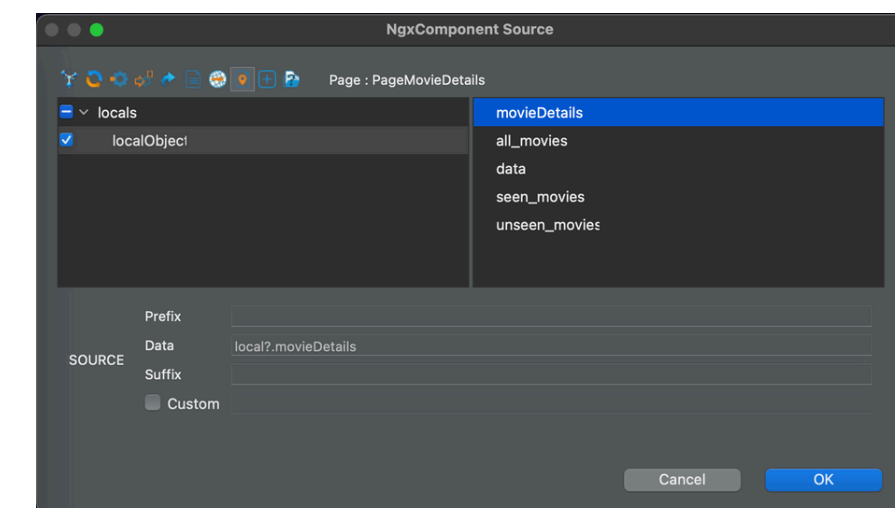
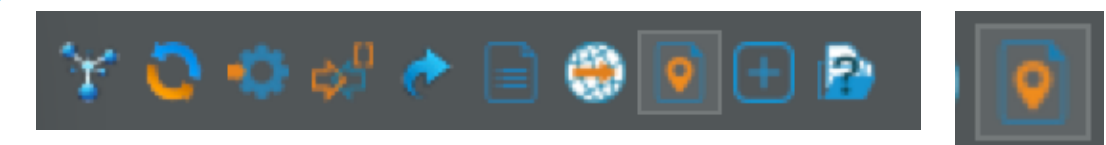
Base properties	
Comment	
118n translate	false
Is active	true
Text value	heading



Text value

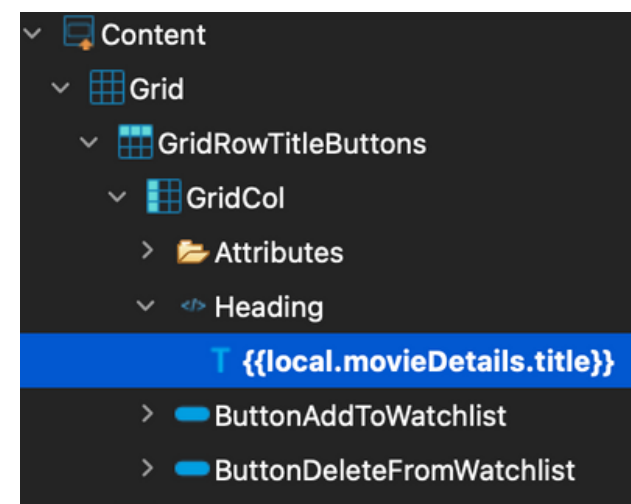
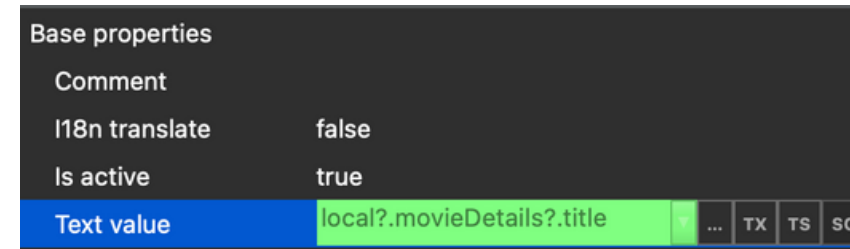
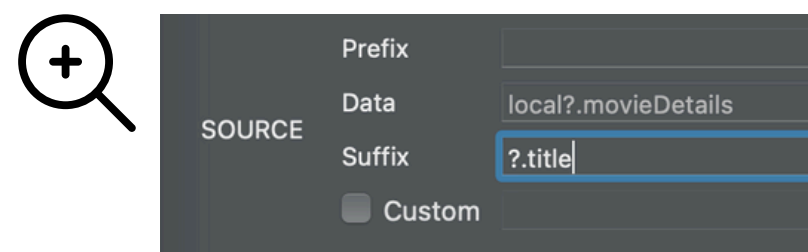
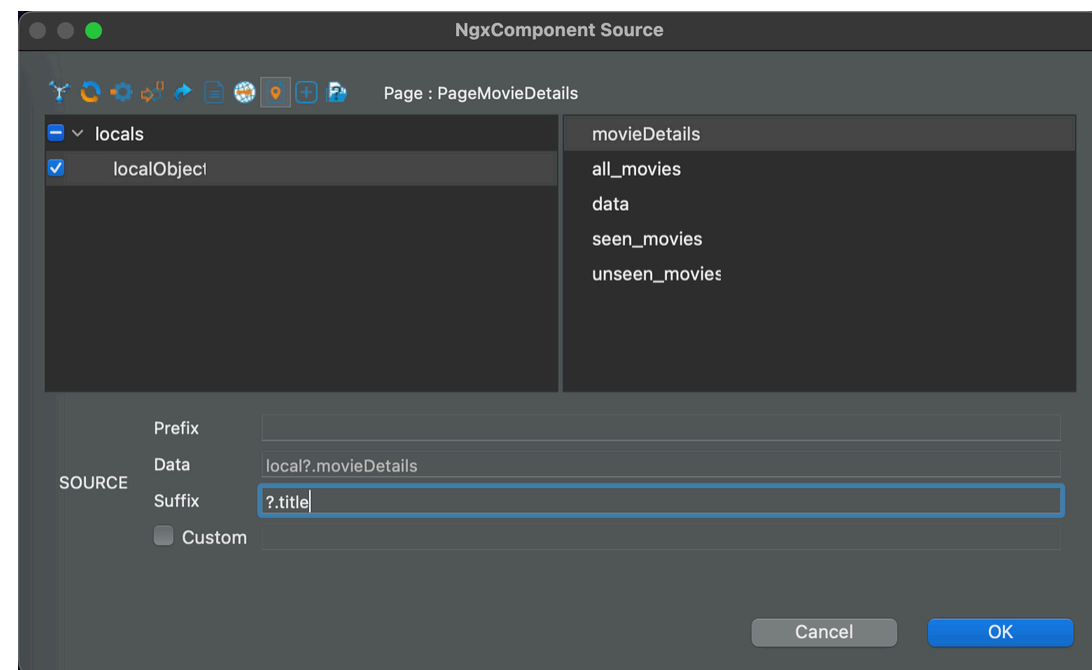


Text value

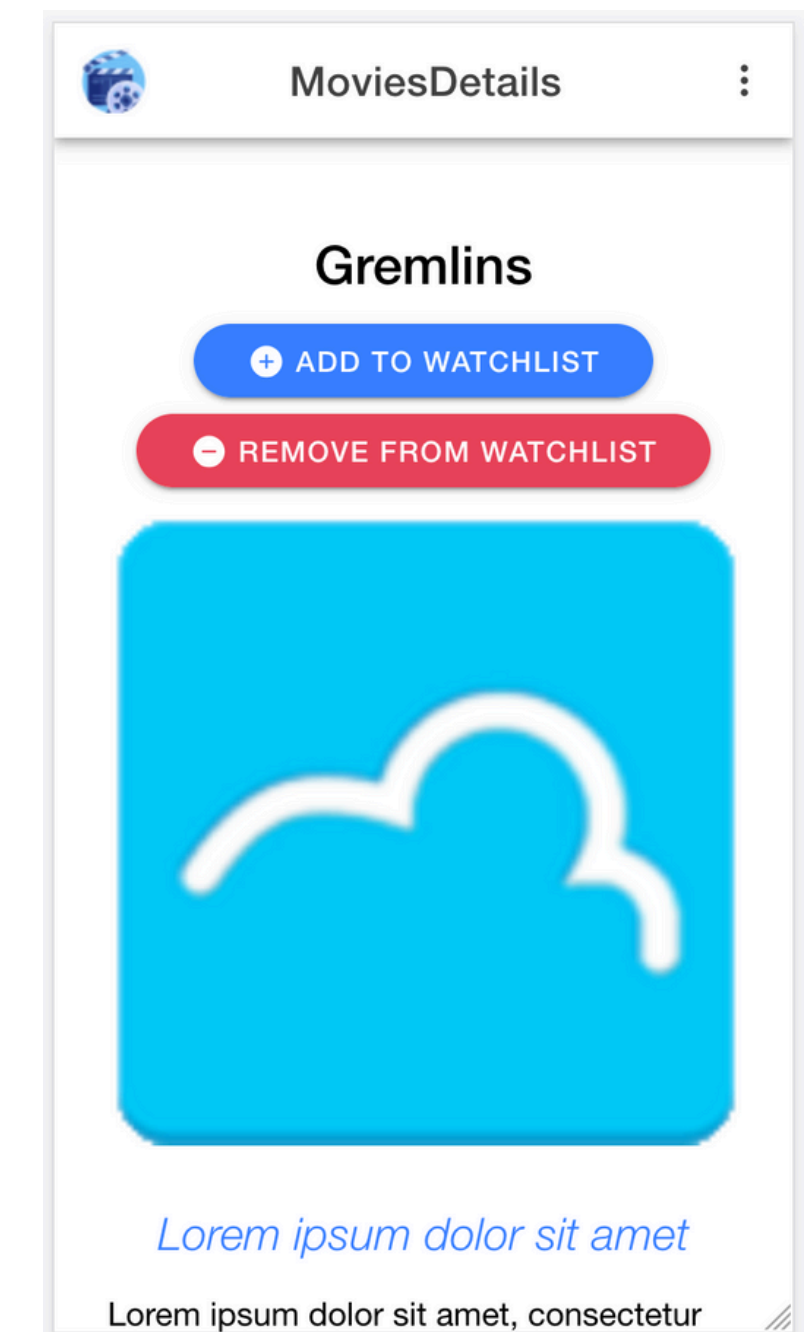


9.2 Display dynamic data in a page

In the **NgxComponent Source** panel, in **Source**, add **?.title** as suffix.



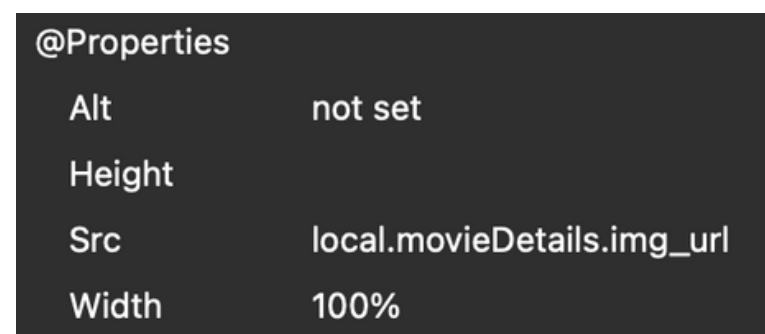
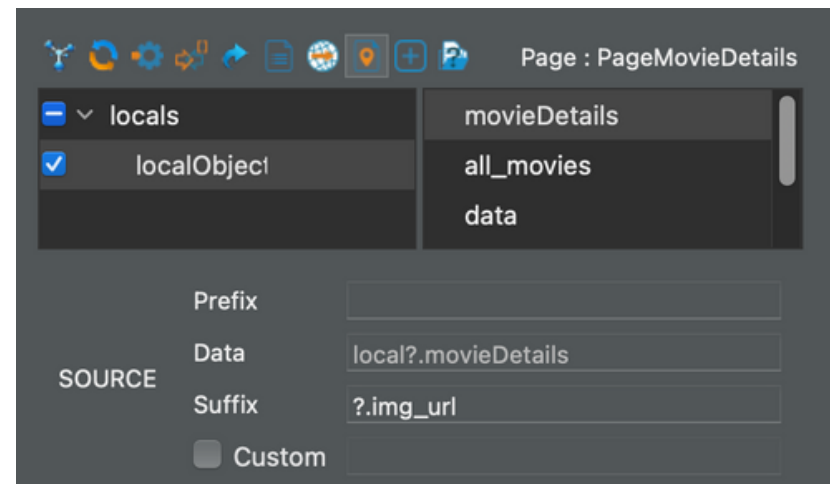
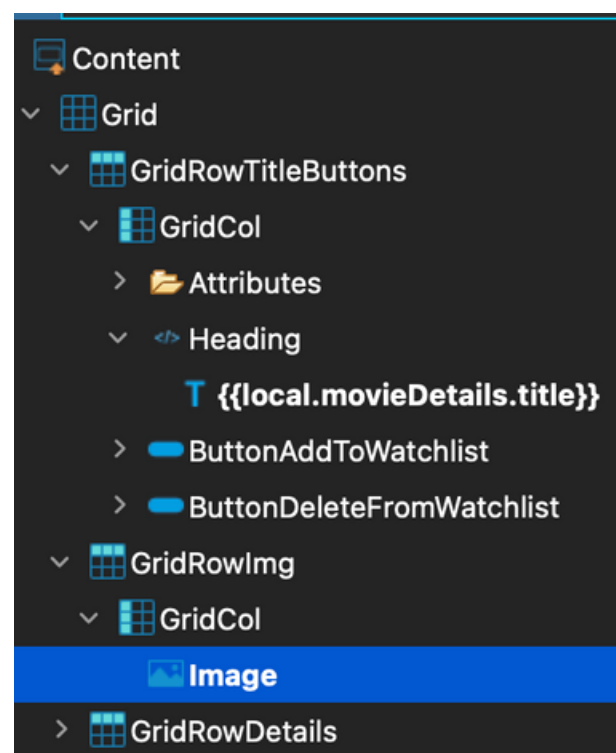
The title appears in the page.



9.2 Display dynamic data in a page

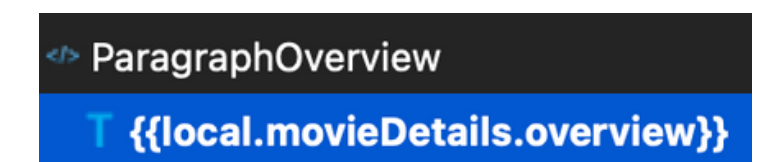
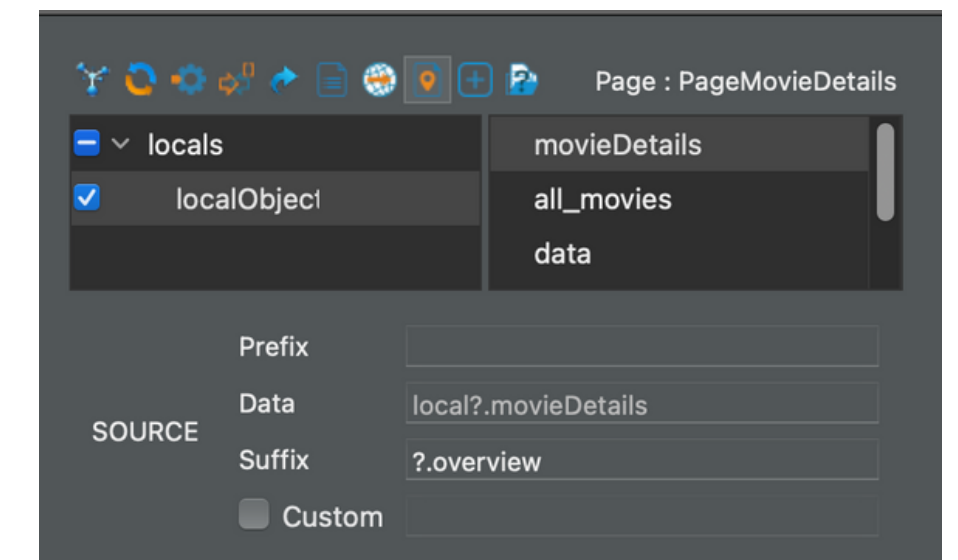
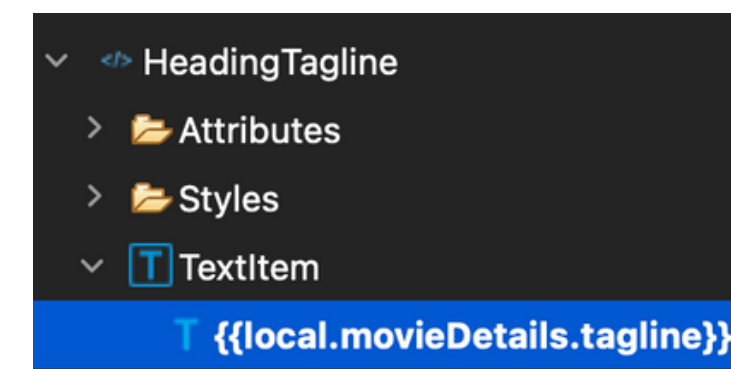
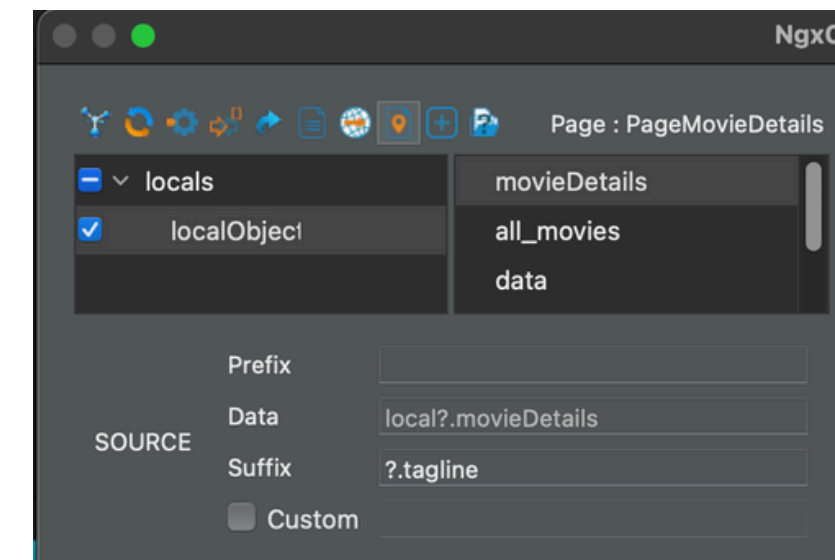
In the **Image** or **GridRowImg**, select the **setLocal MovieDetails** as value for the **Src** property.

In the **NgxComponent Source** panel, in **Source**, add **?img_url** as suffix.



Do the same thing for the **Text** element's **Text** value property

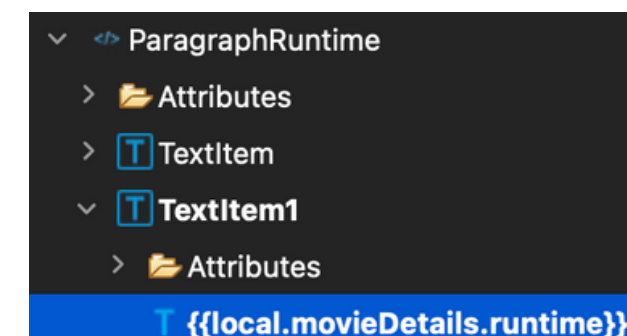
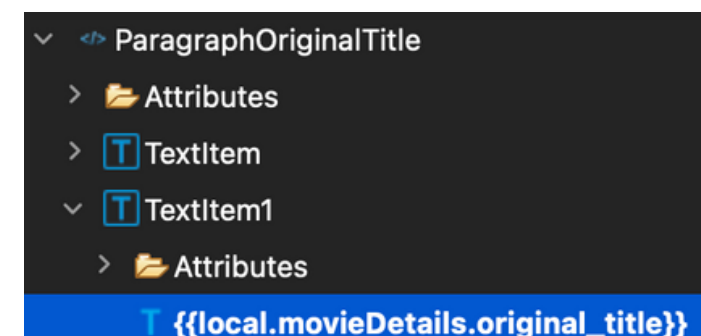
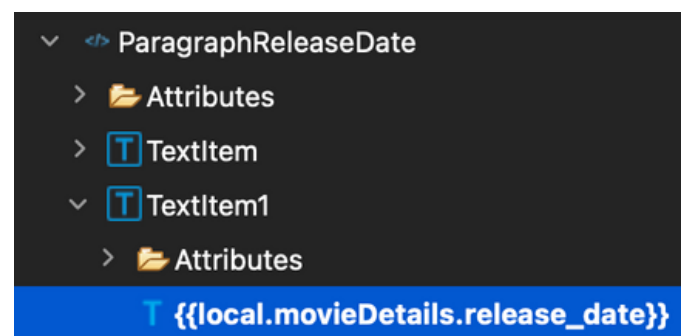
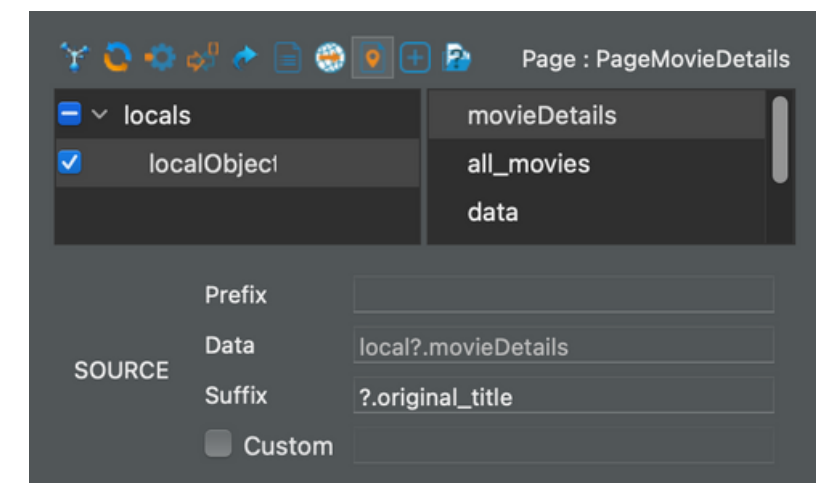
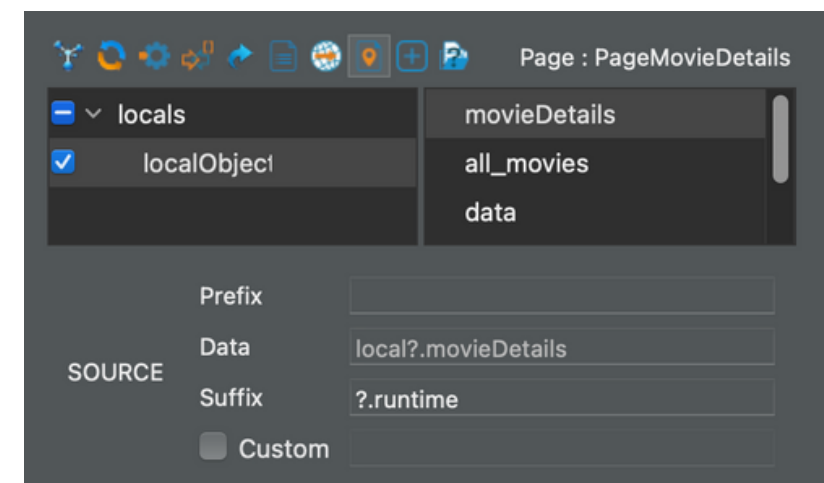
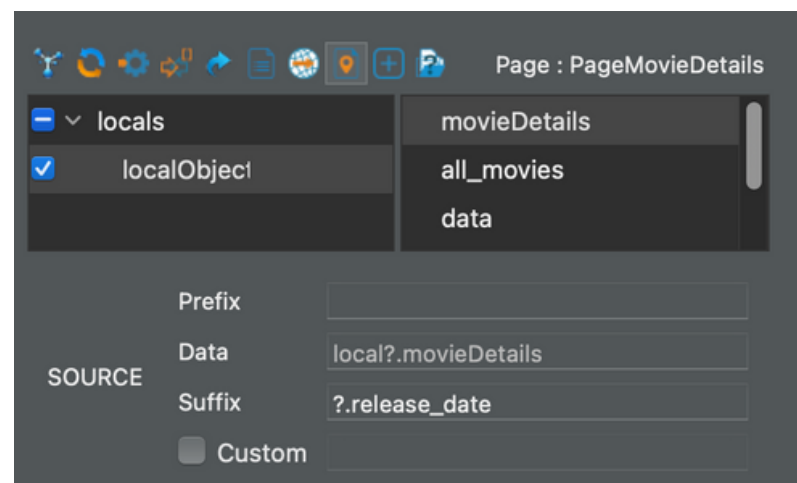
- in the **TextItem** in **HeadingTagline** with suffix **?tagline**
- in the **Paragraph** **ParagraphOverview** with suffix **?overview**



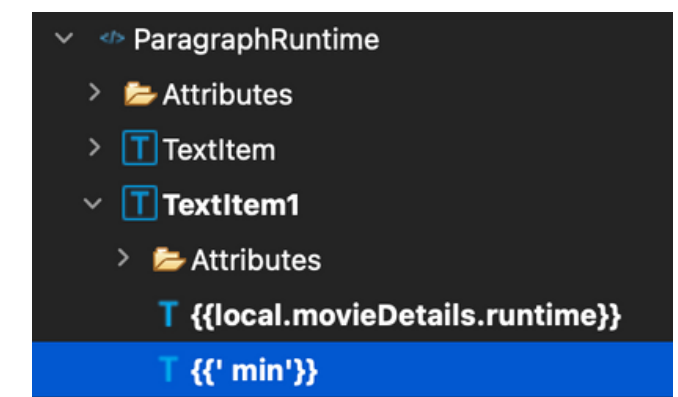
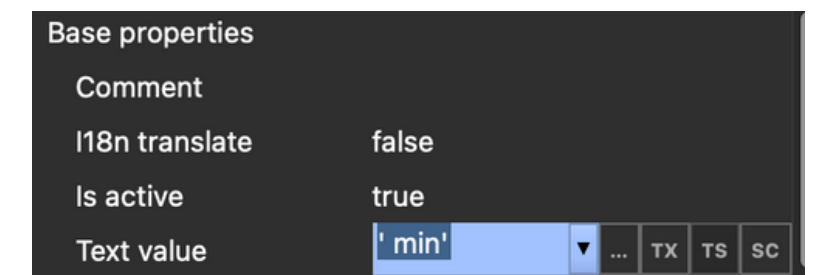
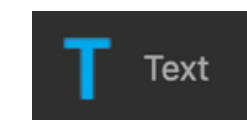
9.2 Display dynamic data in a page

Do the same thing for the **Text** element's **Text value** property in the second **TextItem**

- in the Paragraph **ParagraphReleaseDate** with suffix **?release_date**
- in the Paragraph **ParagraphOriginalTitle** with suffix **?original_title**
- in the Paragraph **ParagraphRuntime** with suffix **?runtime**



In **ParagraphRuntime**,
add a second text element
in the second **TextItem**
and set its **Text value** property
to **'min'** in **TS** scope.



9.2 Display dynamic data in a page

For ParagraphOriginCountry,

- add a **ForEach Directive** after the **first TextItem** and move the **second TextItem** inside
- select the **setLocal MovieDetails** with **suffix ?.origin_country** as **value** for the **Directive source**
- Add **item** as **value** for the **ForEach item property**
and set **' ' + item** in **TS scope** as **value** for the **Text value property** in the **Text element's properties**

Directive type	ForEach
Is active	true
Tag name	ng-container
Expert	
Directive expression	
Directive source	local.movieDetails.origin_country
ForEach index	
ForEach item	item



Base properties	
Comment	
I18n translate	false
Is active	true
Text value	' ' + item



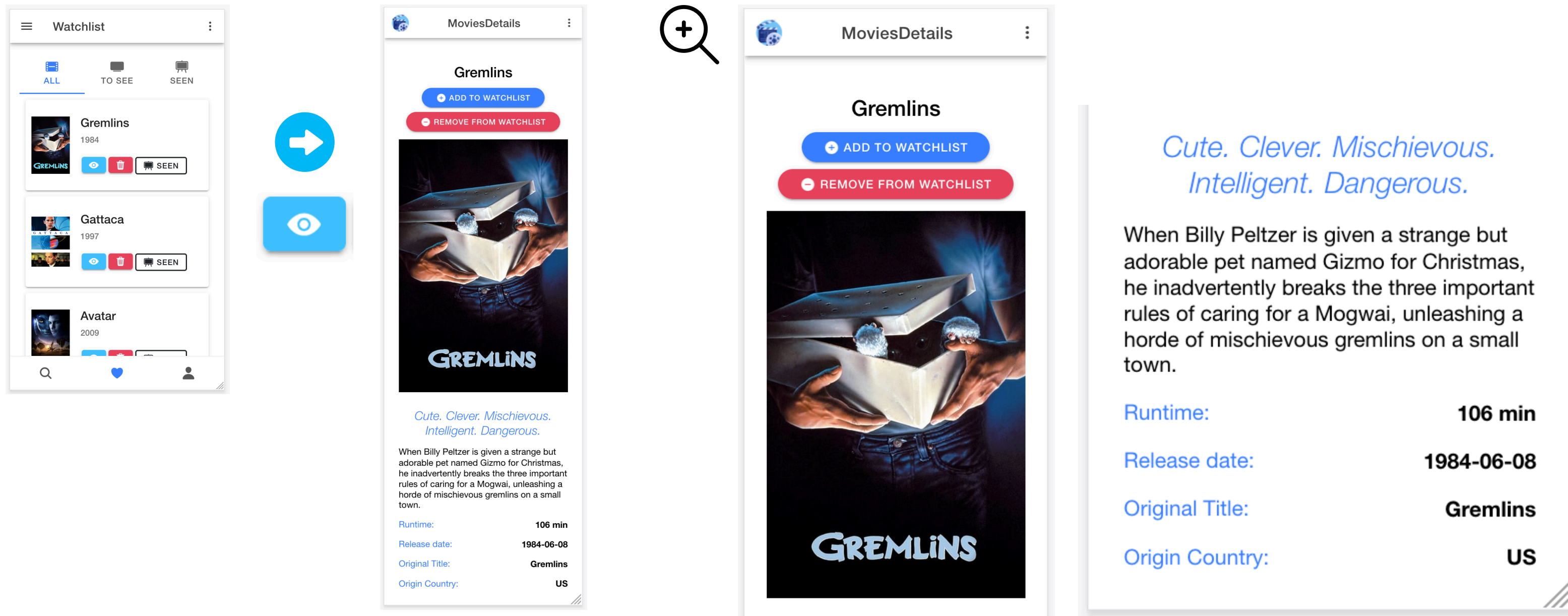
```

<? ParagraphOriginCountry
  > Attributes
  > TextItem
  > ForEach local.movieDetails.origin_country;
    > TextItem1
      > Attributes
        @ class=font-semibold
      T {{' ' + item}}
  
```



9.2 Display dynamic data in a page

Now, in `pageWatchlist`, when we click on the `DetailButton` in a `WatchlistCard`, the `pageMovieDetails` is pushed and the data appears on the page.

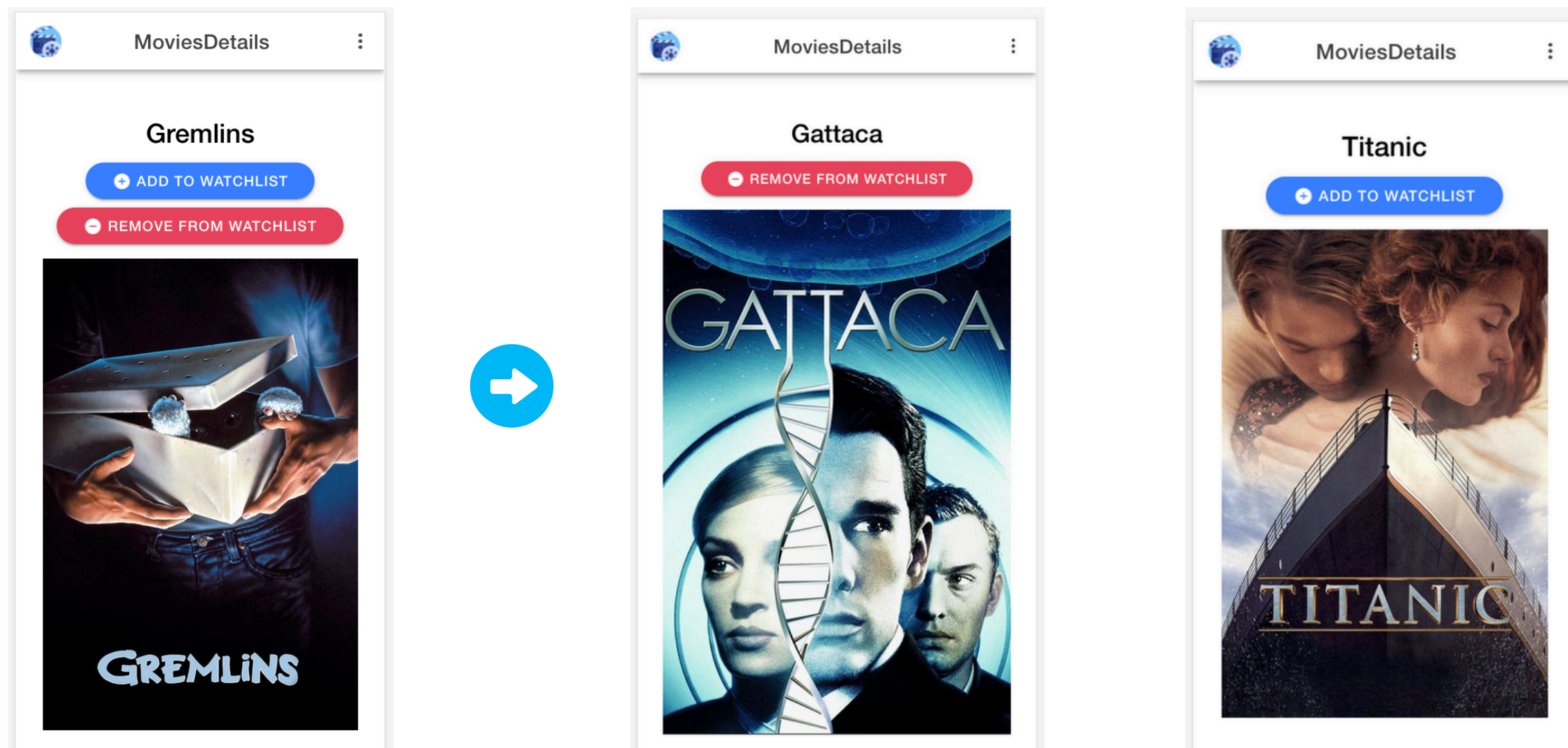


9.3 Use a NgIf directive to display elements

As for now, in **pageMovieDetails**, both buttons appear on the page.

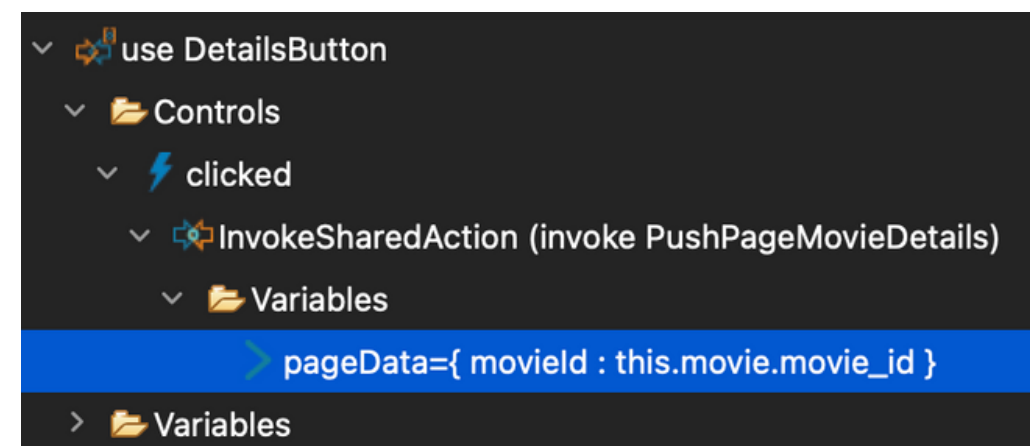
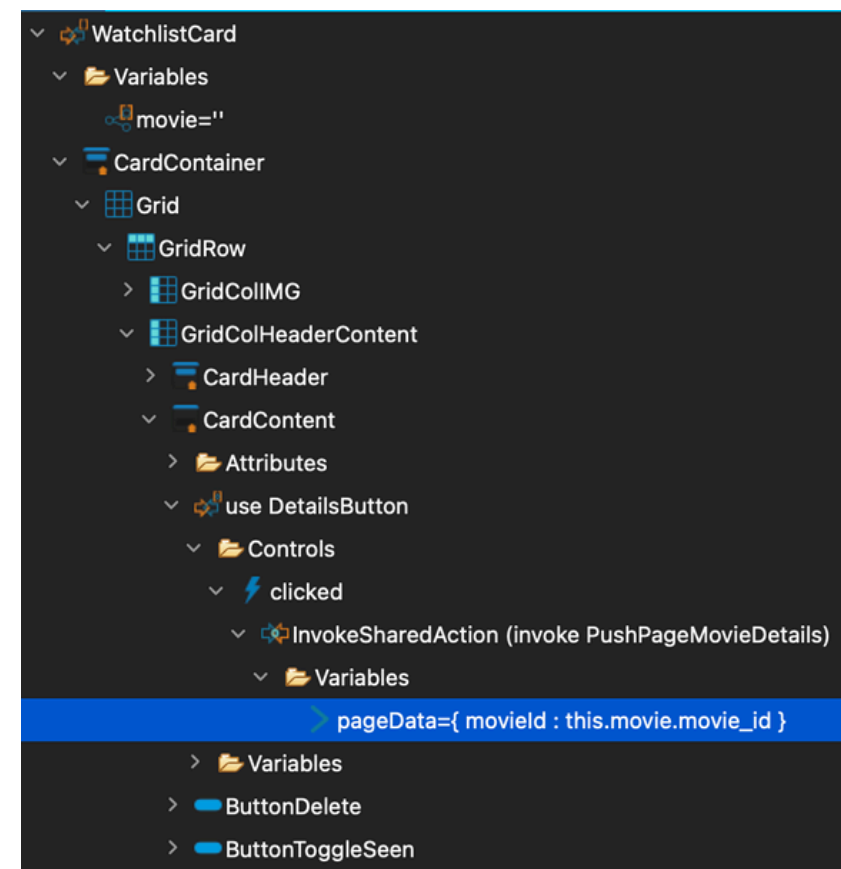
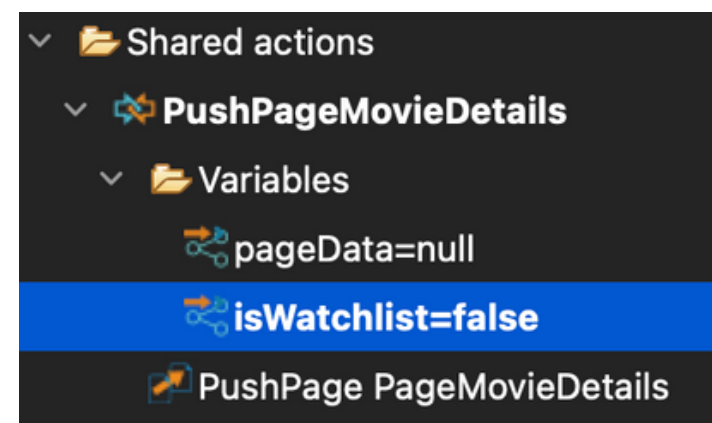
We are going to use a variable **isWatchlist** and a **NgIf directive** to display

- **DeleteFromWatchlistButton** when the page is pushed from **PageWatchlist**
- **AddToWatchlistButton** when the page is pushed from **PageSearch**

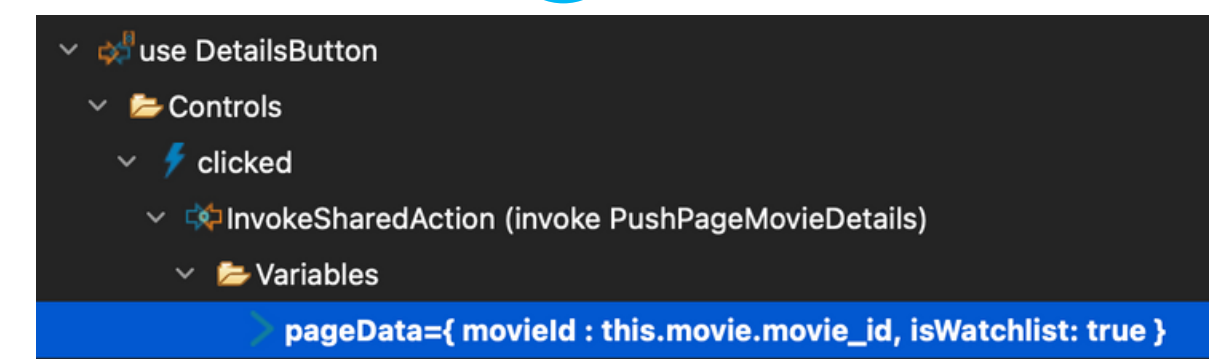
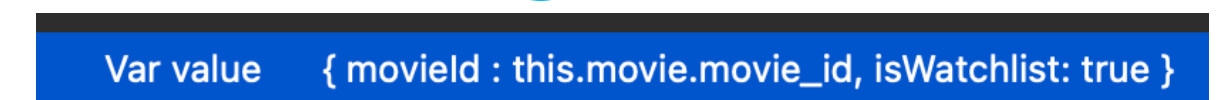
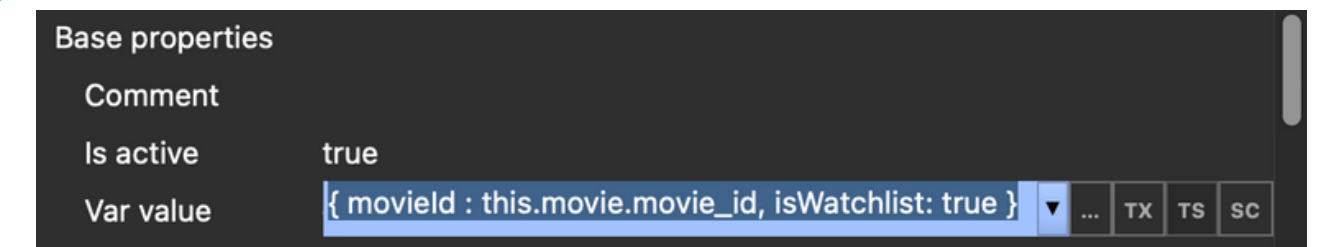


9.3 Use a NgIf directive to display elements

First, in the **SharedAction** **PushPageMovieDetails**, add a **SharedAction** variable, rename it **isWatchlist** and set its **value** to **false**.

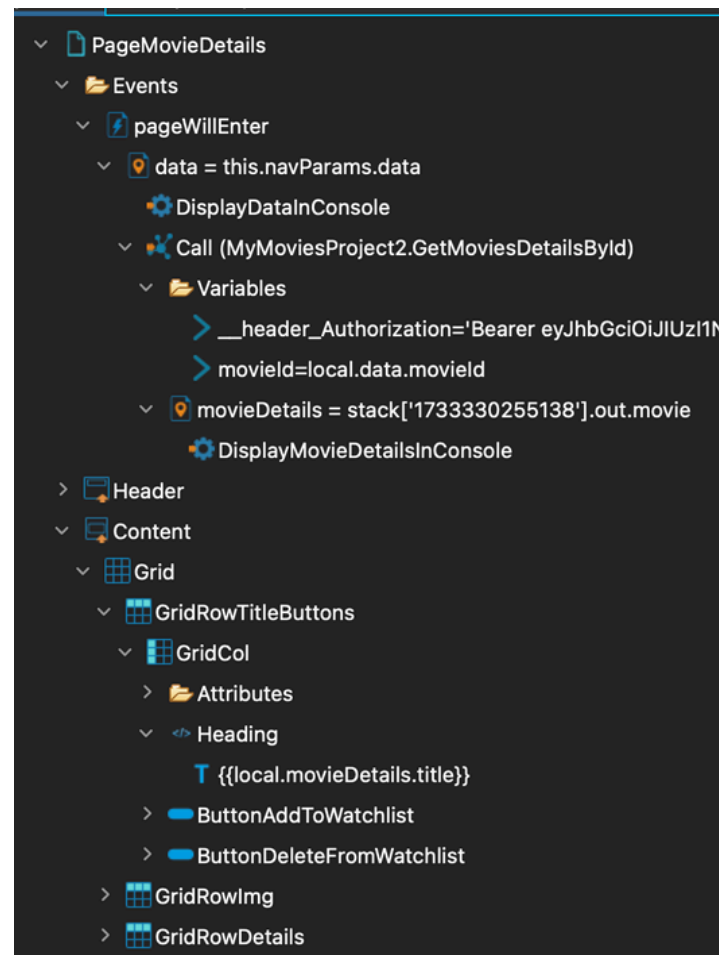


In the **sharedComponent** **WatchlistCard** in the **InvokeSharedAction** **PushPageMovieDetails**, click on the variable **pageData** to display its properties. Set **{ movieId: this.movie.movie_id, isWatchlist: true }** in **TS** scope as **value** for **Var value** property.

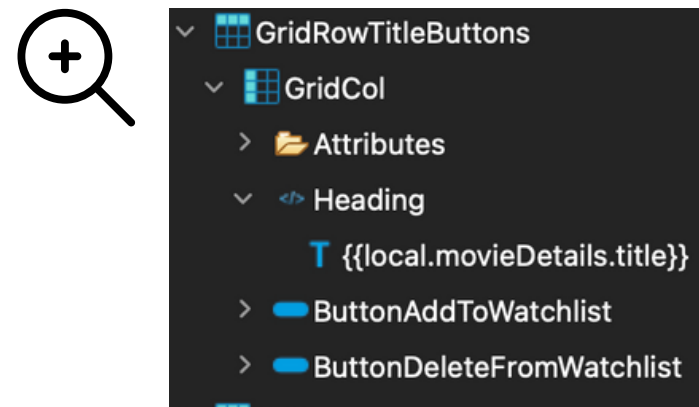


9.3 Use a NgIf directive to display elements

In PageMovieDetails



In the GridCol of GridRowTitleButtons



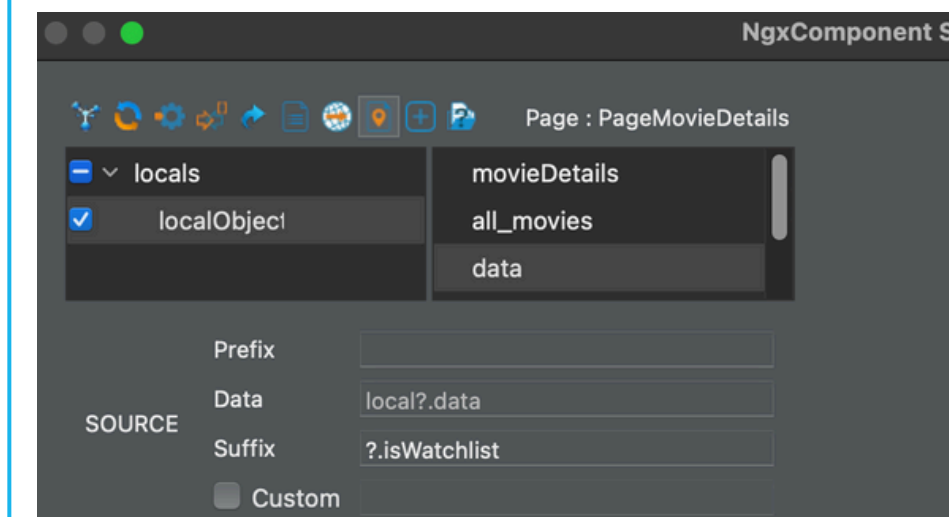
Before **ButtonAddToWatchlist**,
add a **directive** and set **If** as **value**
for the **Directive Type** property.



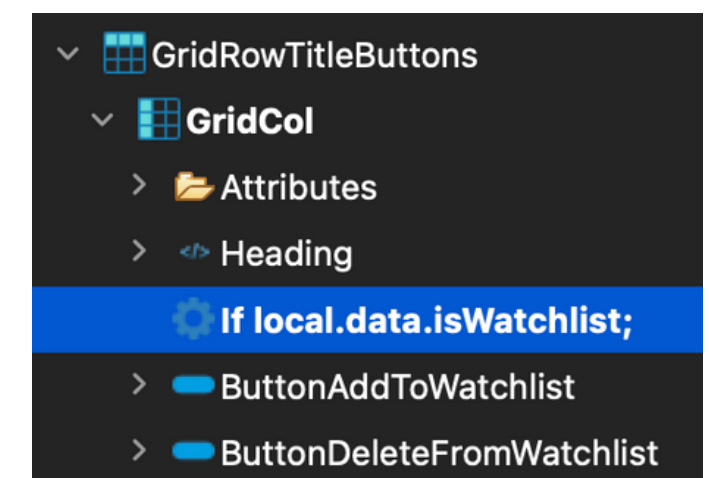
Base properties	
Comment	
Directive type	If
Is active	true
Tag name	ng-container
Expert	
Directive expression	
Directive source	
ForEach index	
ForEach item	



For the **value** of the **Directive Source** property,
in the **NgxComponent Source** panel,
set the **setLocal data** with the **suffix ?.isWatchlist**.



Base properties	
Comment	
Directive type	If
Is active	true
Tag name	ng-container
Expert	
Directive expression	
Directive source	local.data.isWatchlist
ForEach index	
ForEach item	

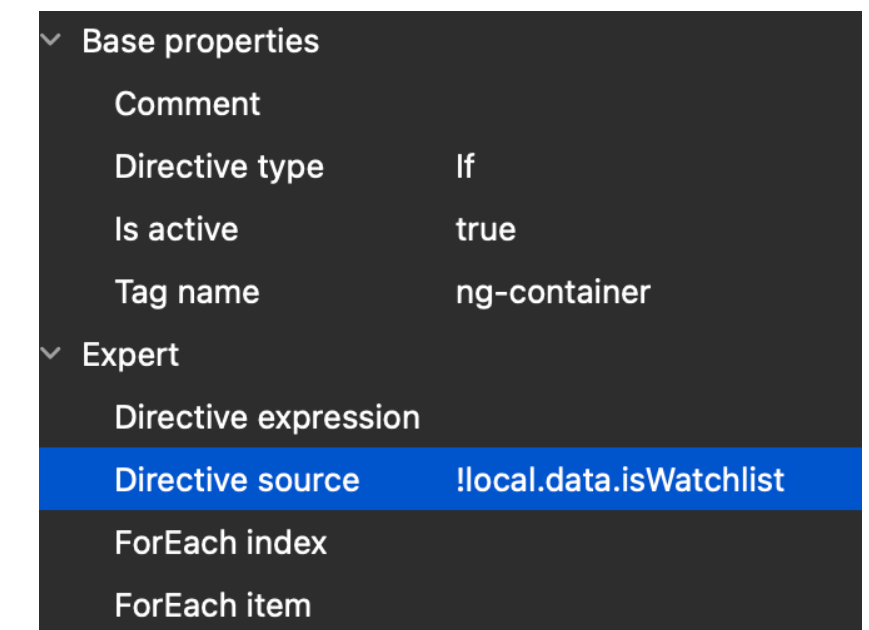
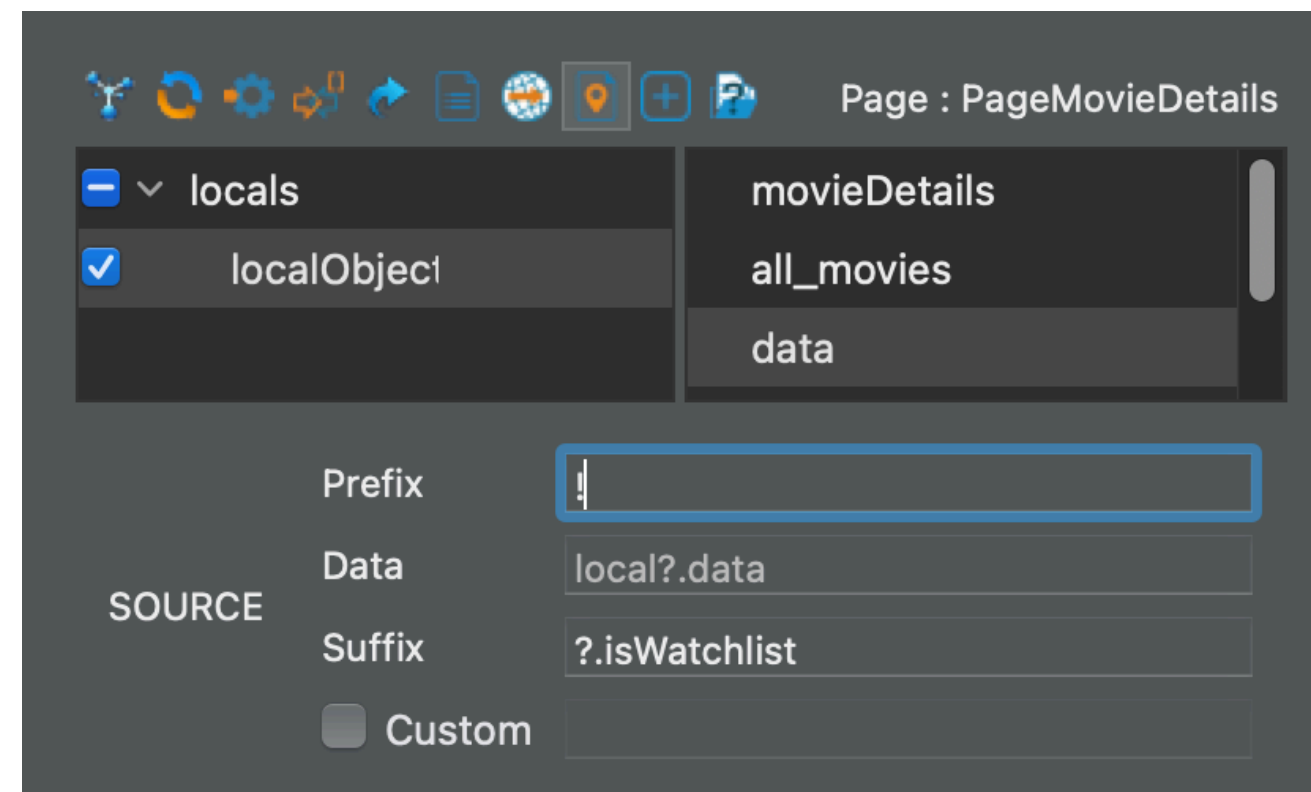
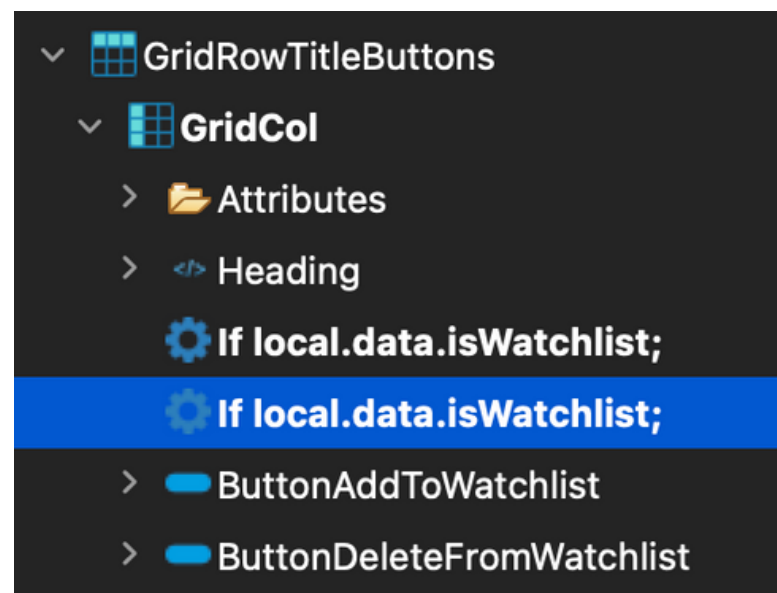


9.3 Use a NgIf directive to display elements

For the second If directive

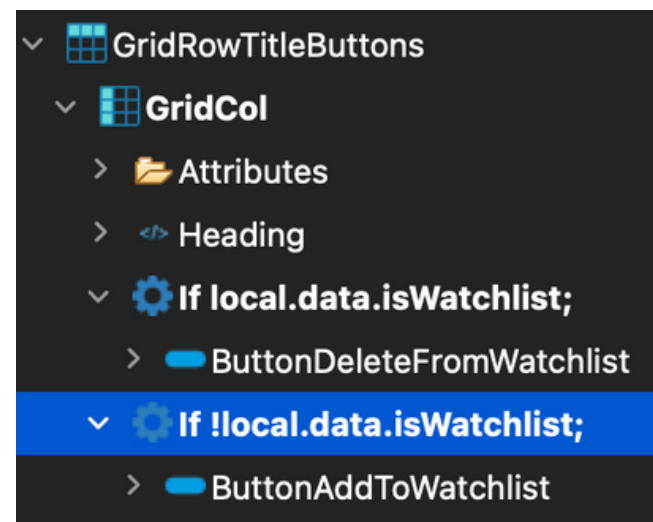
for the **value** of the **Directive Source** property,
in the **NgxComponent Source** panel,
add the **prefix !**.

Duplicate the directive.



9.3 Use a NgIf directive to display elements

Move **DeleteFromWatchlistButton**
in the first directive **If local.data.isWatchlist**.
Move **AddToWatchlistButton**
in the second directive **If !local.data.isWatchlist**.



Now, when we push **PageMovieDetails** from **PageWatchlist**,
the value of **local.data.isWatchlist** is **true**.

```
Var value { movieId : this.movie.movie_id, isWatchlist: true }
```

And only **DeleteFromWatchlistButton**
is displayed in **PageMovieDetails**.



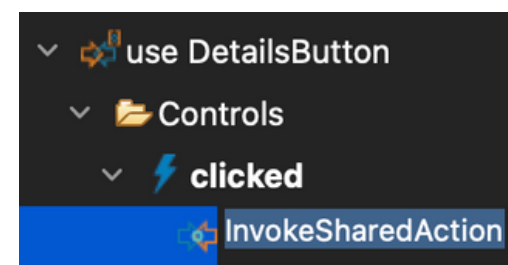
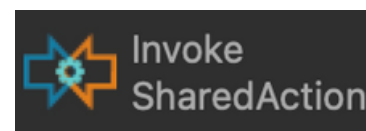
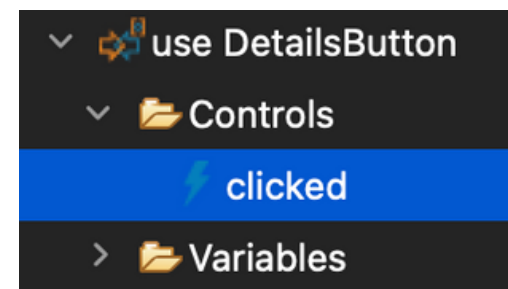
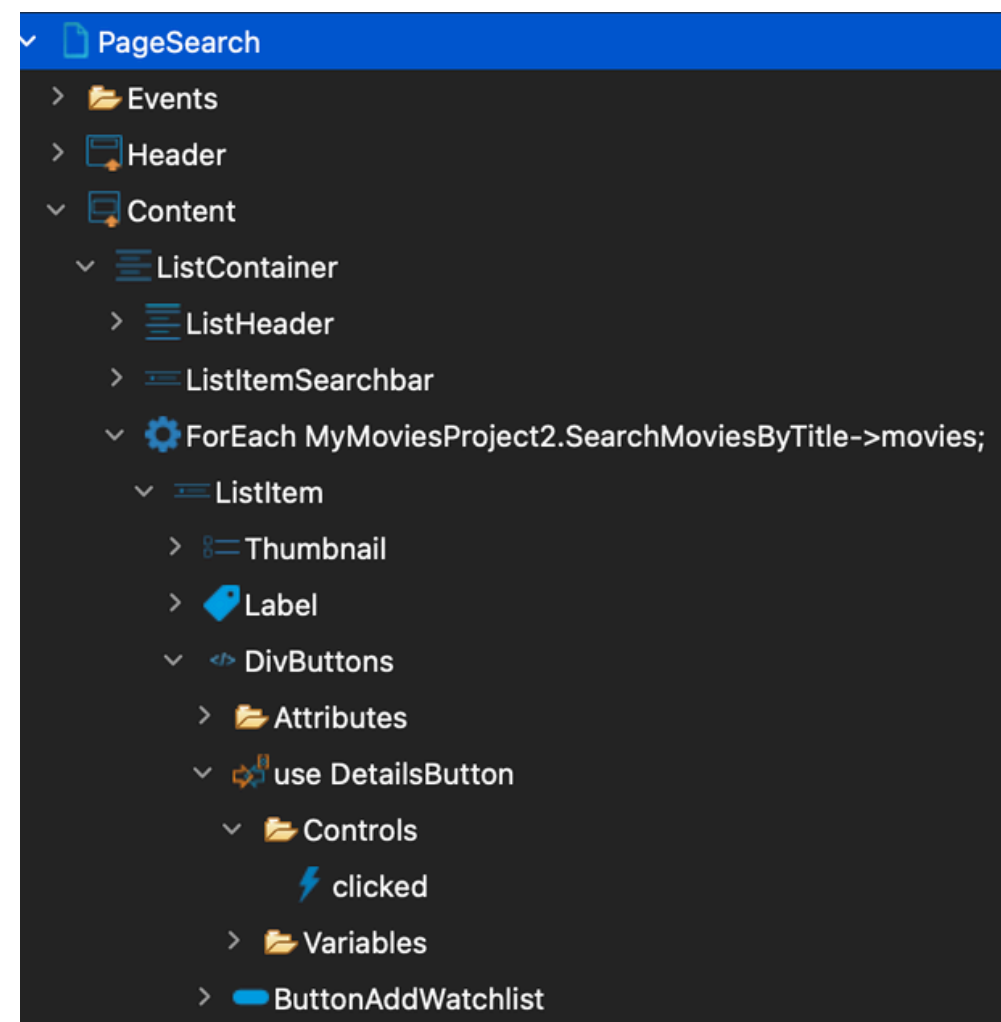
REMOVE FROM WATCHLIST



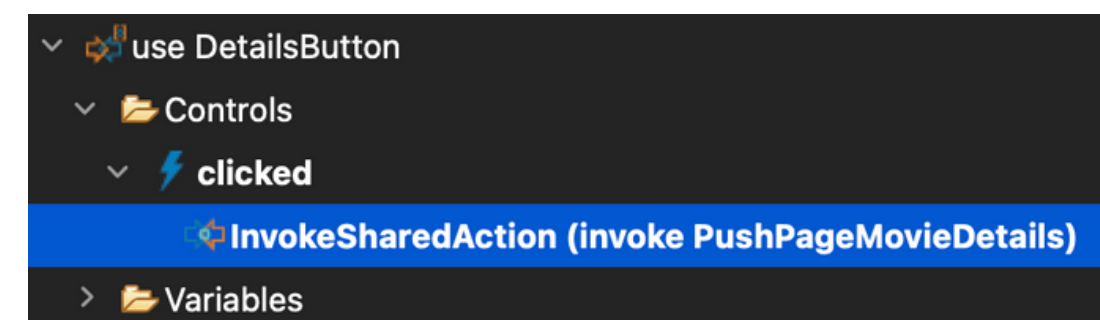
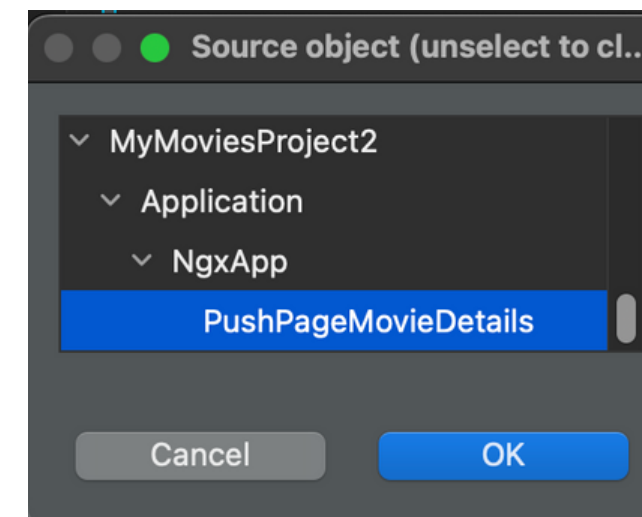
9.3 Use a NgIf directive to display elements

Now, let's do the same thing in **PageSearch**.

In the **SharedComponent** use **DetailsButton**,
add an **Invoke SharedAction** component
in the **SharedComponent BindingEvent** clicked.



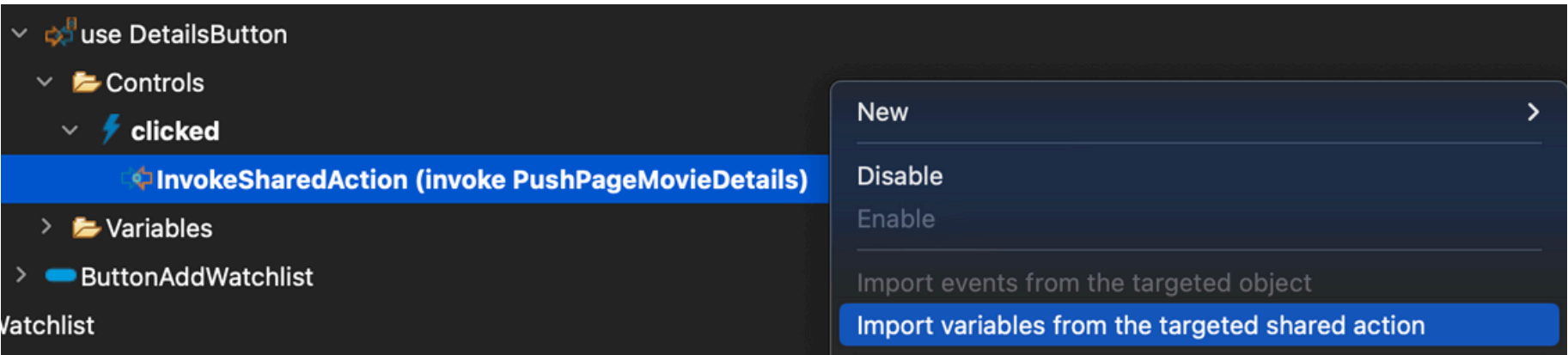
In the **Invoke SharedAction** component's properties
For the **Shared action** property,
in the **Source object** panel,
select **PushPageMovieDetails**.



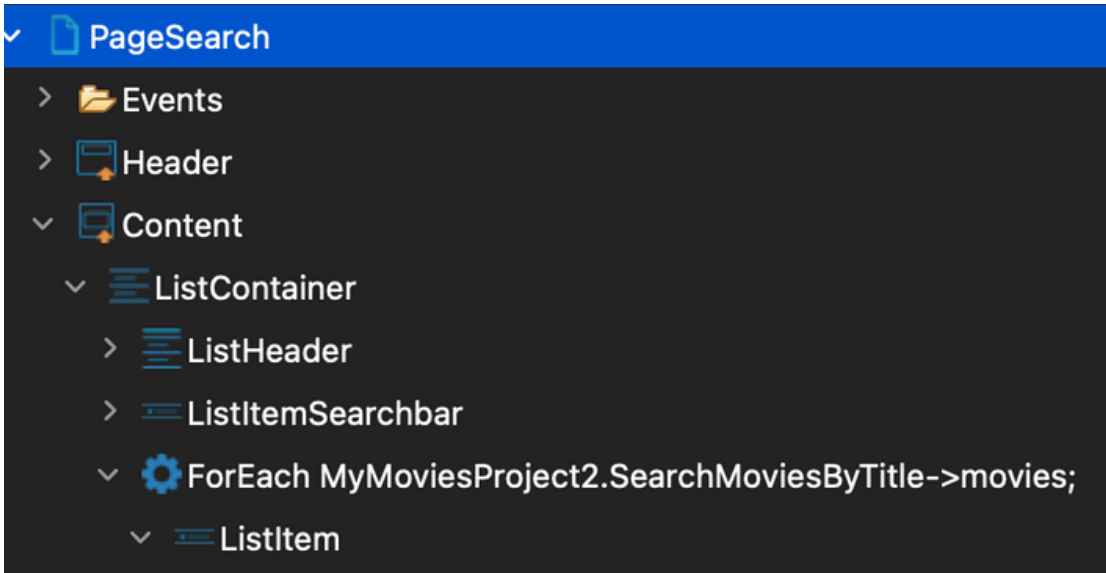
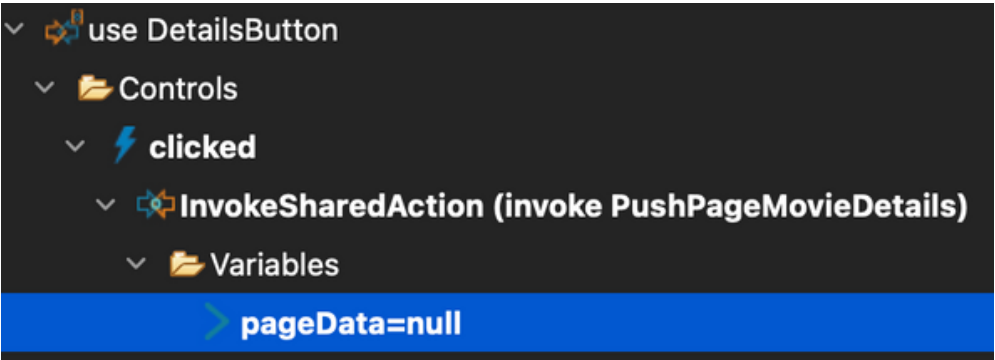
9.3 Use a NgIf directive to display elements



In use **DetailsButton**, right-click on the **InvokeSharedAction** invoke **PushPageMovieDetails** , and select **Import variables from the targeted shared action**.



The **variable pageData** has been imported.



ForEach MyMoviesProject2.SearchMoviesByTitle->movies;

In **ListContainer**, in the **ForEach** directive, the **ForEach item property** has item as **value**.

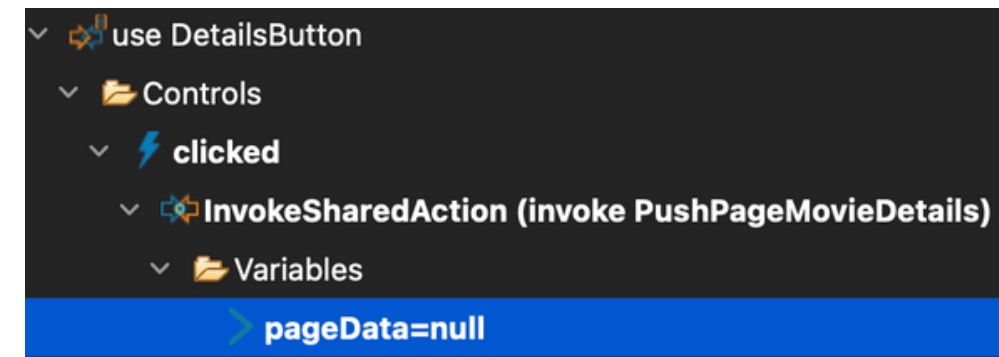
Directive expression	
Directive source	MyMoviesProject2.SearchMoviesByTitle->movies
ForEach index	idx
ForEach item	item



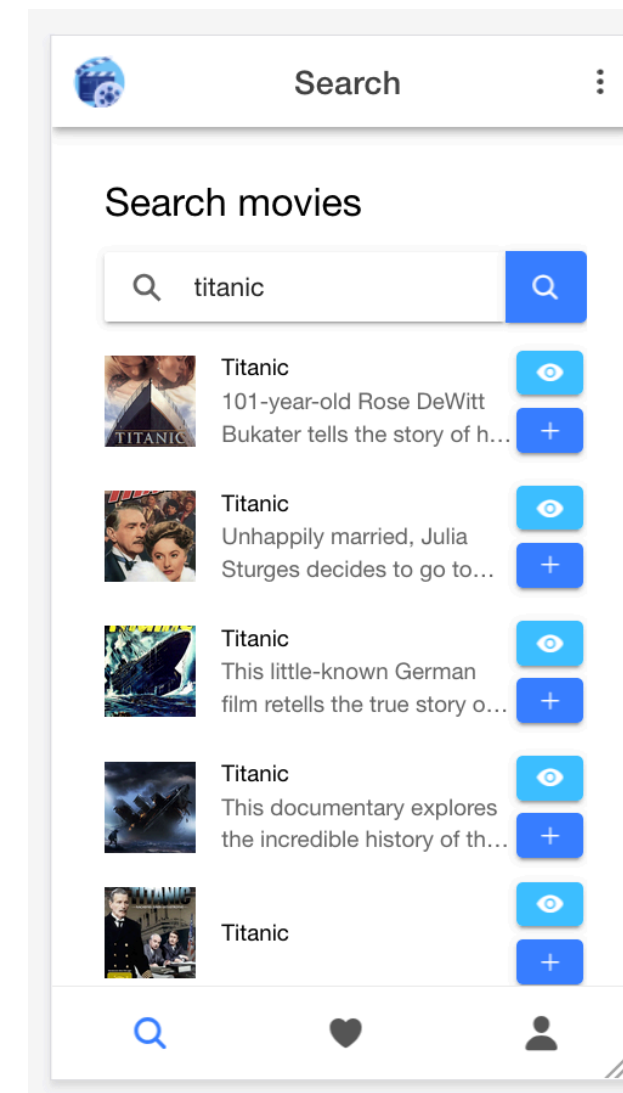
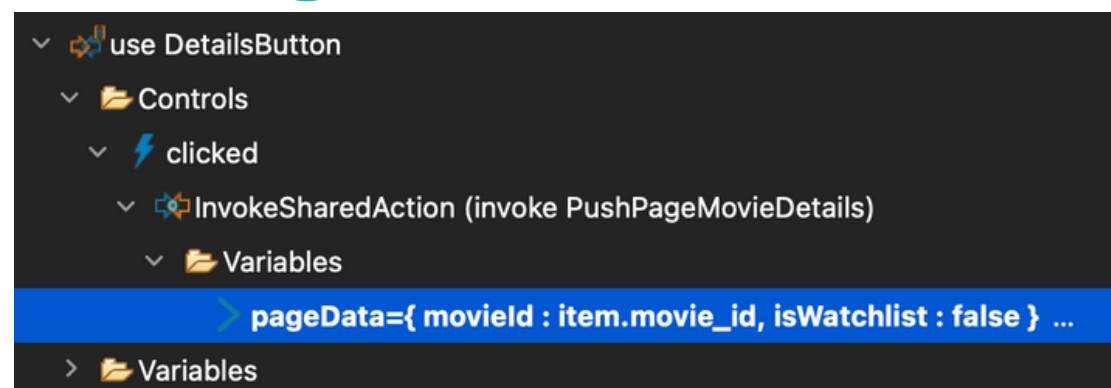
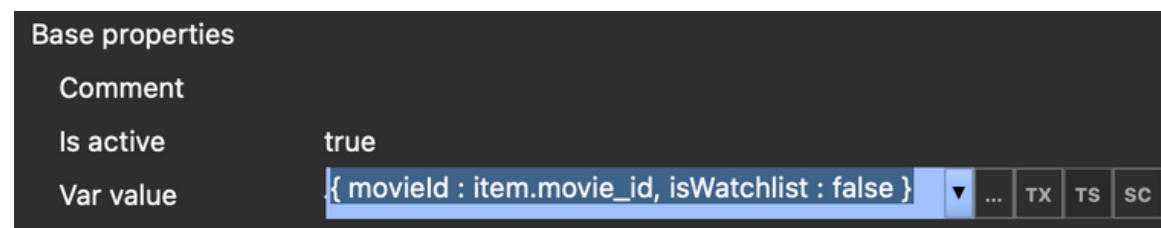
9.3 Use a NgIf directive to display elements

Now, when we push **PageMovieDetails** from **PageSearch**, the value of **local.data.isWatchlist** is **false**.

And only **AddToWatchlistButton** is displayed in **PageMovieDetails**.



In the variable **pageData**'s properties, enter **{ movieId: this.movie.movie_id, isWatchlist: false }** in **TS scope** as value for the **Var value** property.

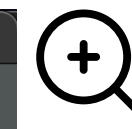
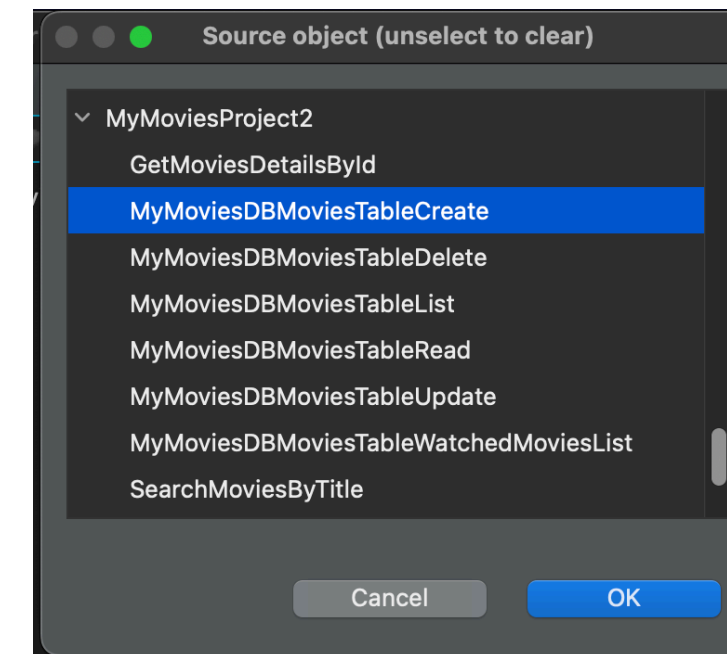


9.4 Implement ButtonAddToWatchlist

Now, in **pageSearch**,
let's implement **ButtonAddToWatchlist**.

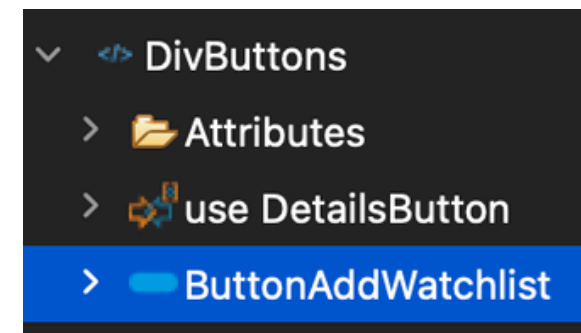
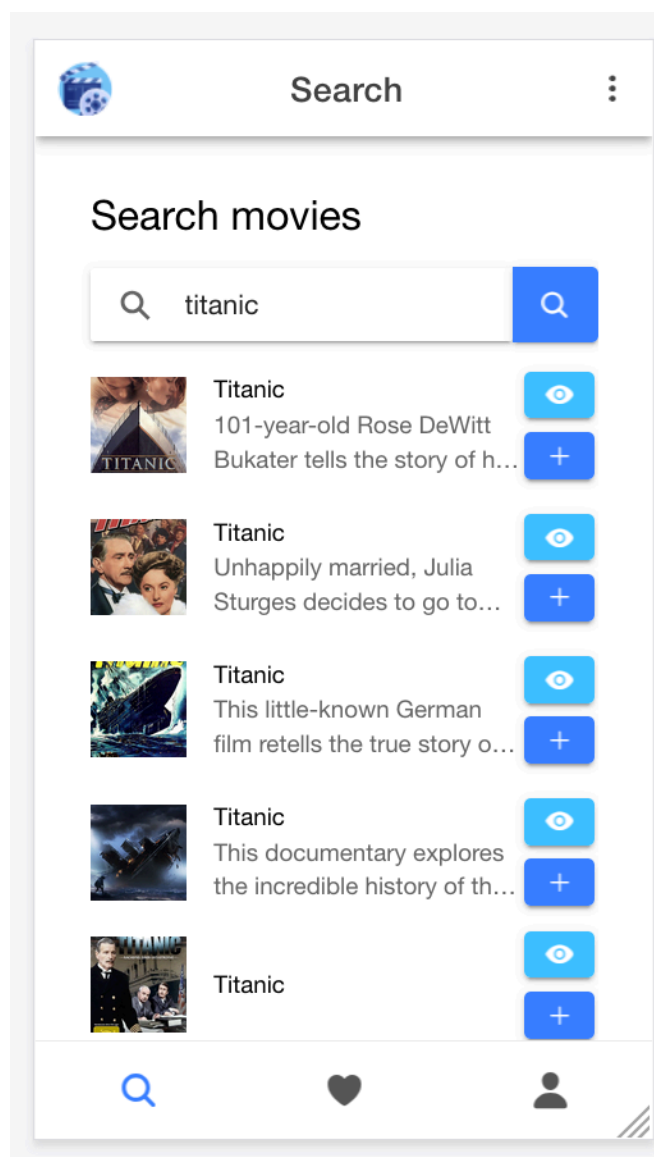


In the **CallSequence**,
choose the sequence **MyMoviesDBMoviesTableCreate**
as **value** for the **Requestable** property.

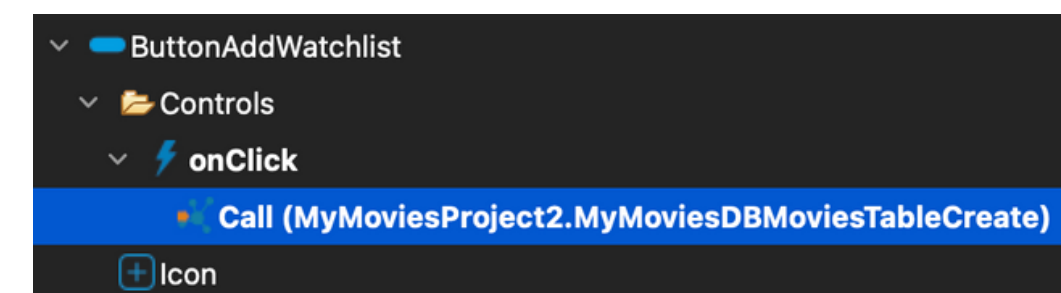
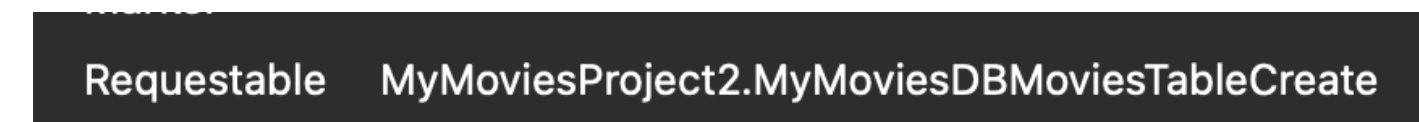
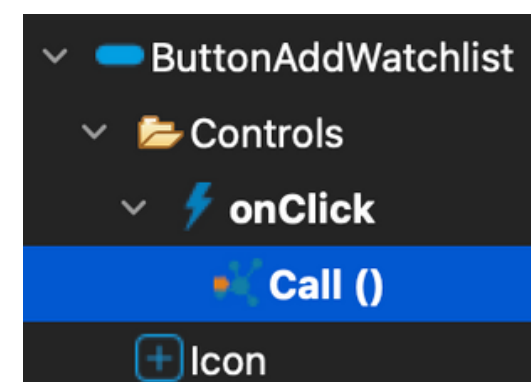


MyMoviesDBMoviesTableCreate

This **sequence**
create an entry in the database.

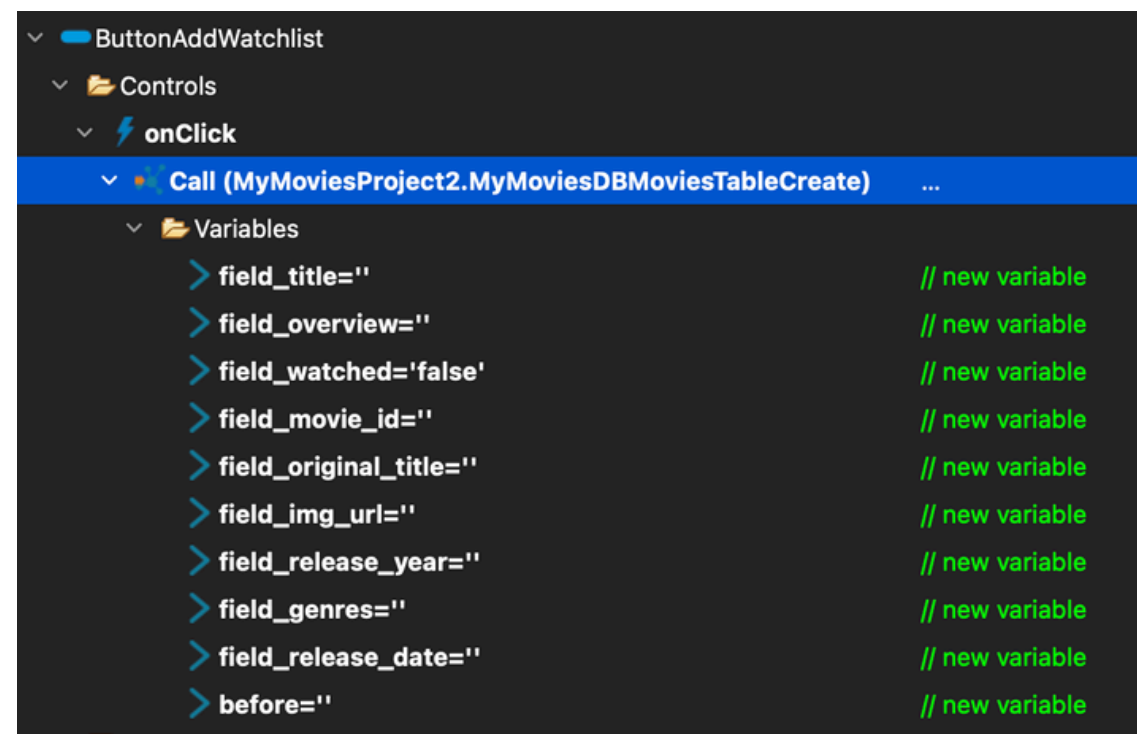
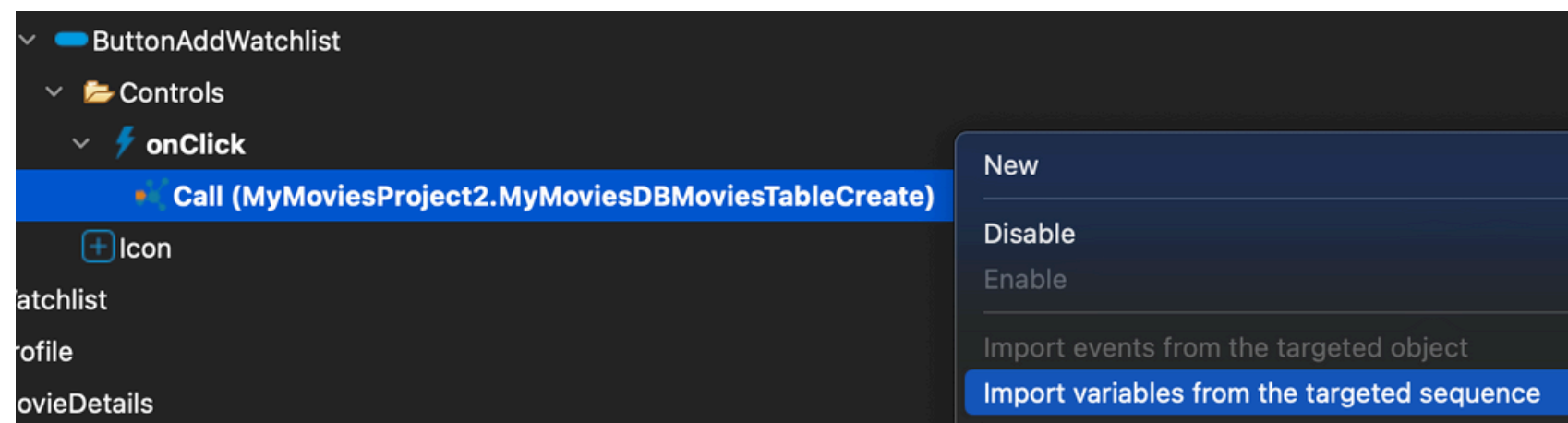


In the **onClick** event of
ButtonAddToWatchlist,
add a **CallSequence**.



9.4 Implement ButtonAddToWatchlist

Right-click on the **CallSequence** **MyMoviesDBMoviesTableCreate** and select **Import variables from the the targeted sequence**.



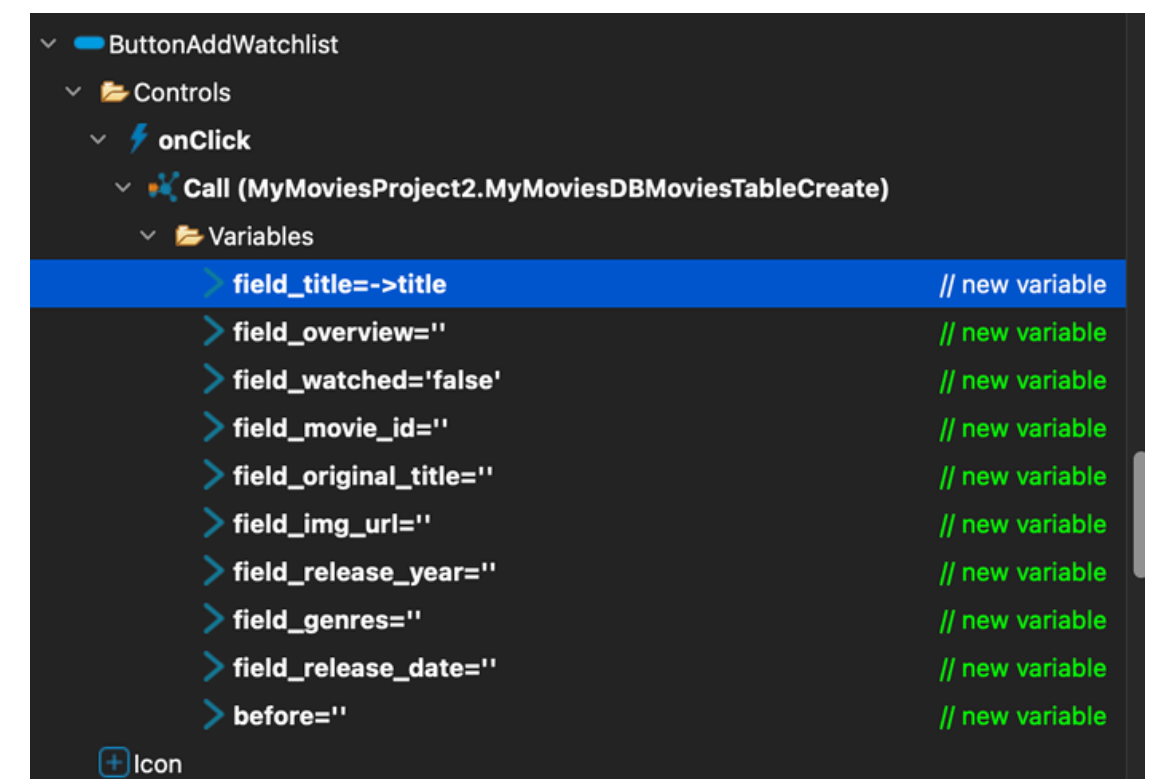
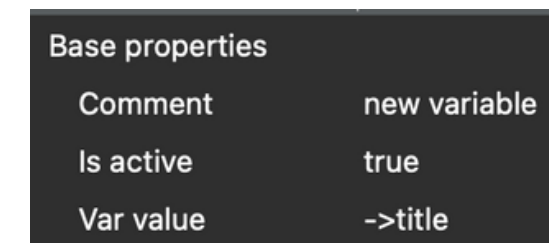
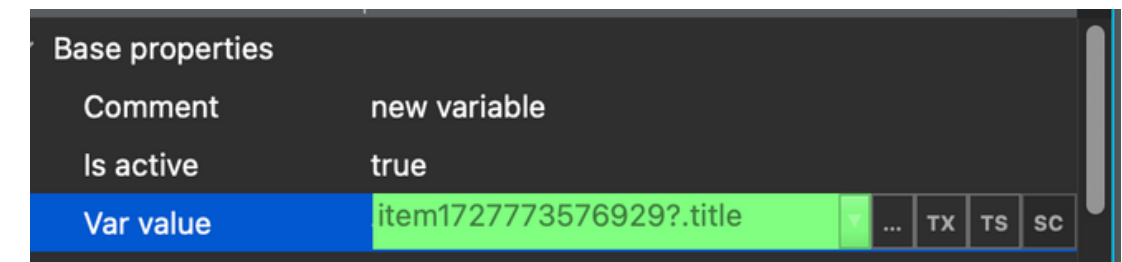
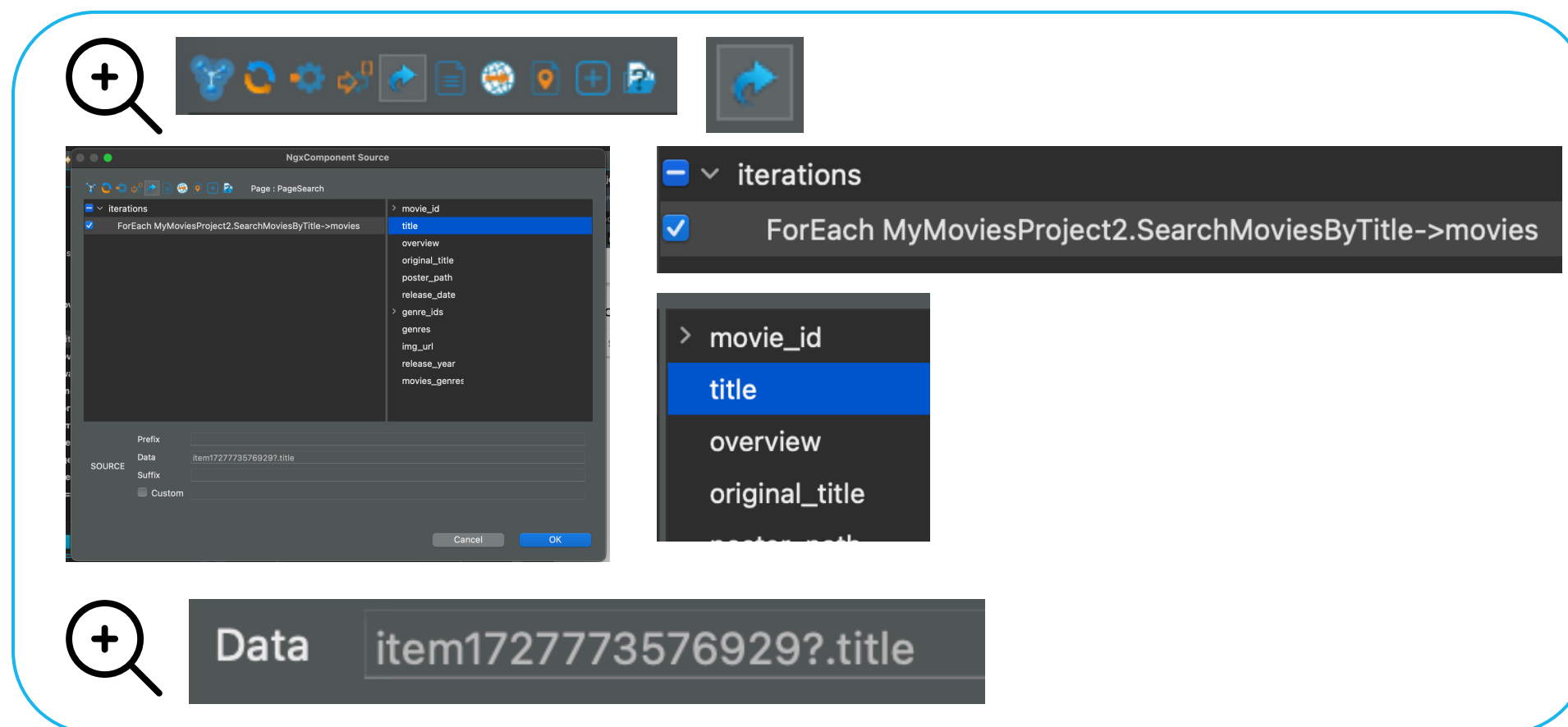
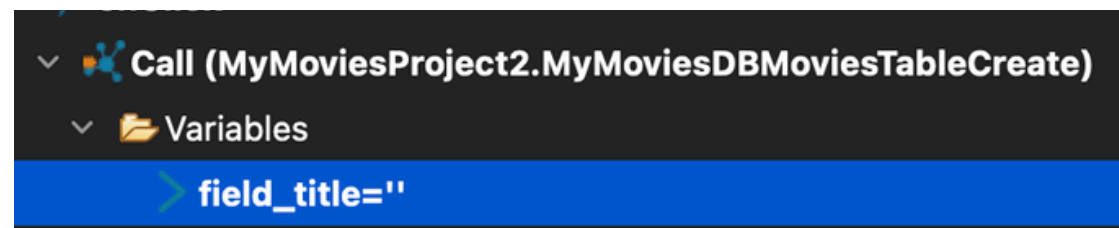
The variables corresponding to the fields to fill in the database are imported.

	T title	≡ overv...	🕒 watc...	# movi...	T original_t...	T img...	T release_y...	T gen...	T release_d...
1	Gremlins	When Billy Pe...	✓	927	Gremlins	https://imag...	1984		1984-06-08
2	Gattaca	In a future so...		782	Gattaca	https://imag...	1997		1997-09-07
3	Avatar	In the 22nd c...		19995	Avatar	https://imag...	2009		2009-12-15
+									



9.4 Implement ButtonAddToWatchlist

In the **field_title** variable,
choose the **title** field in the **ForEach** directive
as **value** to the **Var value** property.



9.4 Implement ButtonAddToWatchlist

Do the same thing for the other fields.

```
ButtonAddWatchlist
├── Controls
│   └── onClick
│       └── Call (MyMoviesProject2.MyMoviesDBMoviesTableCreate)
│           └── Variables
│               ├── field_title-->title
│               ├── field_overview-->overview
│               ├── field_watched='false'
│               ├── field_movie_id-->movie_id
│               ├── field_original_title-->original_title
│               ├── field_img_url-->img_url
│               ├── field_release_year-->release_year
│               ├── field_genres-->genres
│               ├── field_release_date-->release_date
│               └── before=0
```

Be careful and add the **number 0** in **TS scope** as **value** for the **before field**.

```
Call (MyMoviesProject2.MyMoviesDBMoviesTableCreate)
└── Variables
    ├── field_title-->title
    ├── field_overview-->overview
    ├── field_watched='false'
    ├── field_movie_id-->movie_id
    ├── field_original_title-->original_title
    ├── field_img_url-->img_url
    ├── field_release_year-->release_year
    ├── field_genres-->genres
    ├── field_release_date-->release_date
    └── before=0
```

Without a **value** for the **before field**, the request won't work

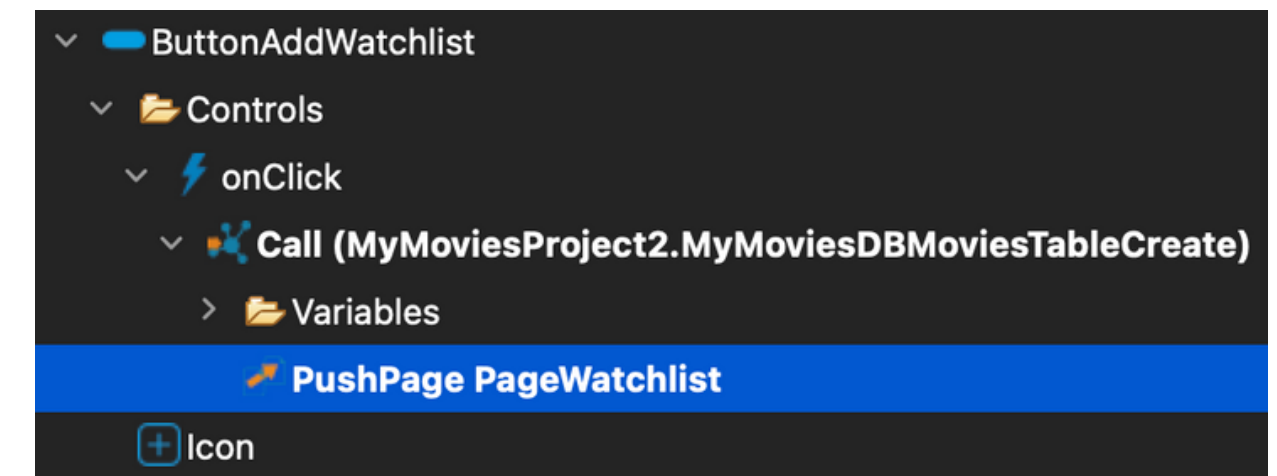
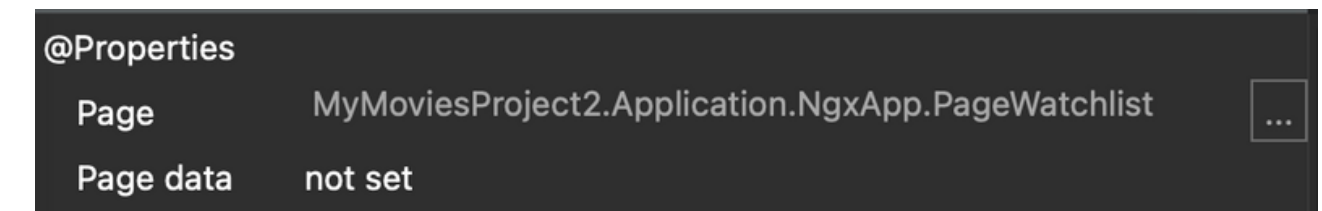
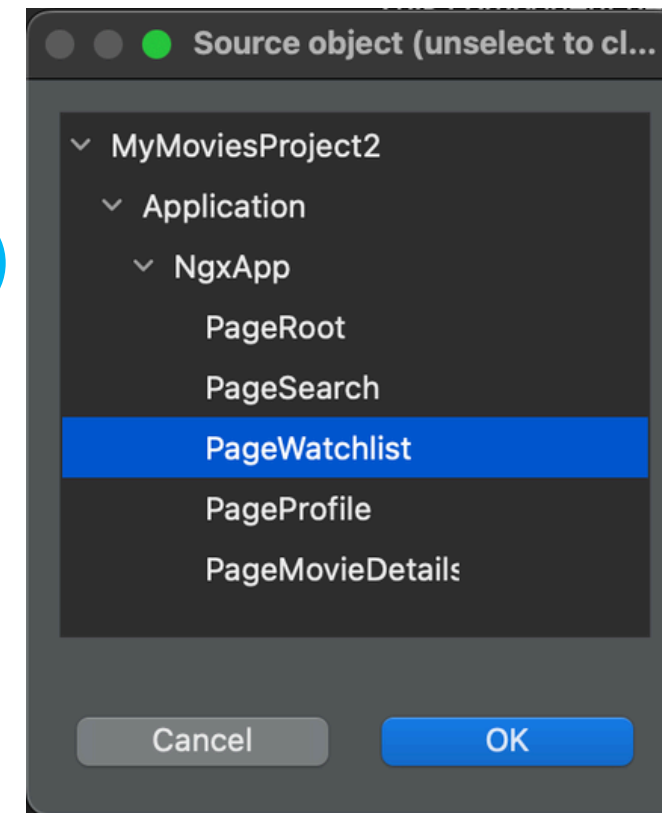
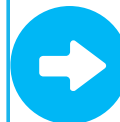
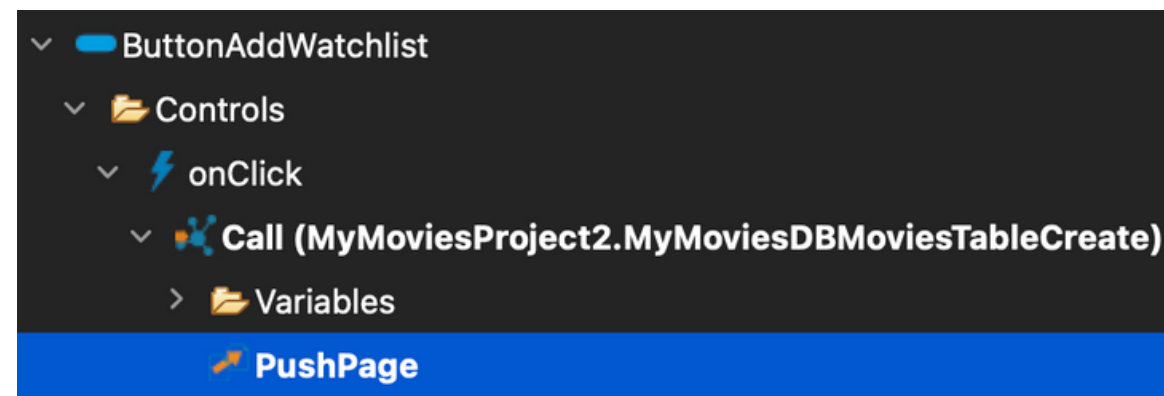
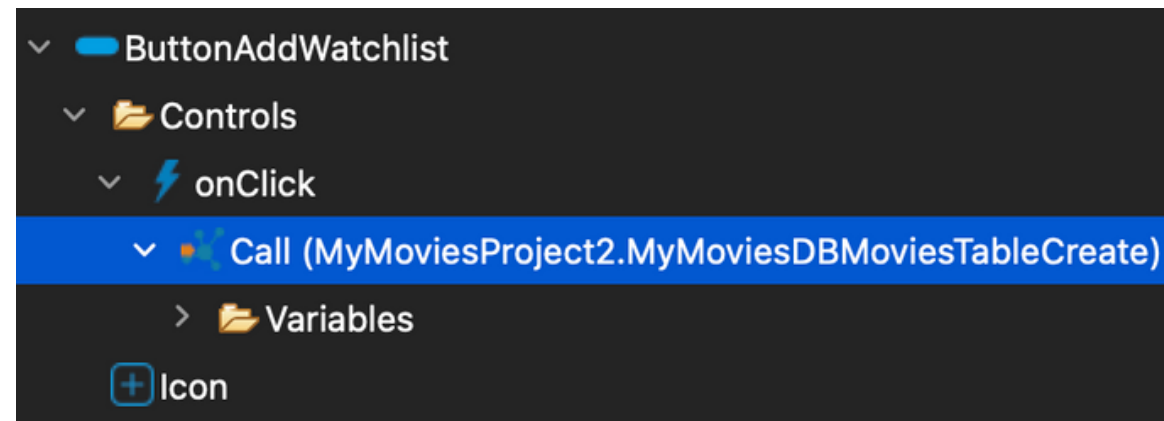


Base properties	
Comment	new variable
Is active	true
Var value	0



9.4 Implement ButtonAddToWatchlist

In the **CallSequence**, add a **PushPage** with **PageWatchlist** as value for the **Requestable** property.



PageWatchlist







Search movies





Titanic
 101-year-old Rose DeWitt Bukater tells the story of her li...






Titanic
 Unhappily married, Julia Sturges decides to go to...






Titanic
 This little-known German film retells the true story of the...






Titanic
 This documentary explores the incredible history of the Titani...






Titanic






Titanic
 "Titanic" is a Punjabi feature film. It is family movie. "Titani...






Titanic






Titanic: The Legend Goes On...
 An animated retelling of the worst passenger ship disaster...






The Chambermaid on the





Watchlist

ALL

TO SEE

SEEN

Gremlins

1984

SEEN

Gattaca

1997

SEEN

Avatar

2009

SEEN

Titanic

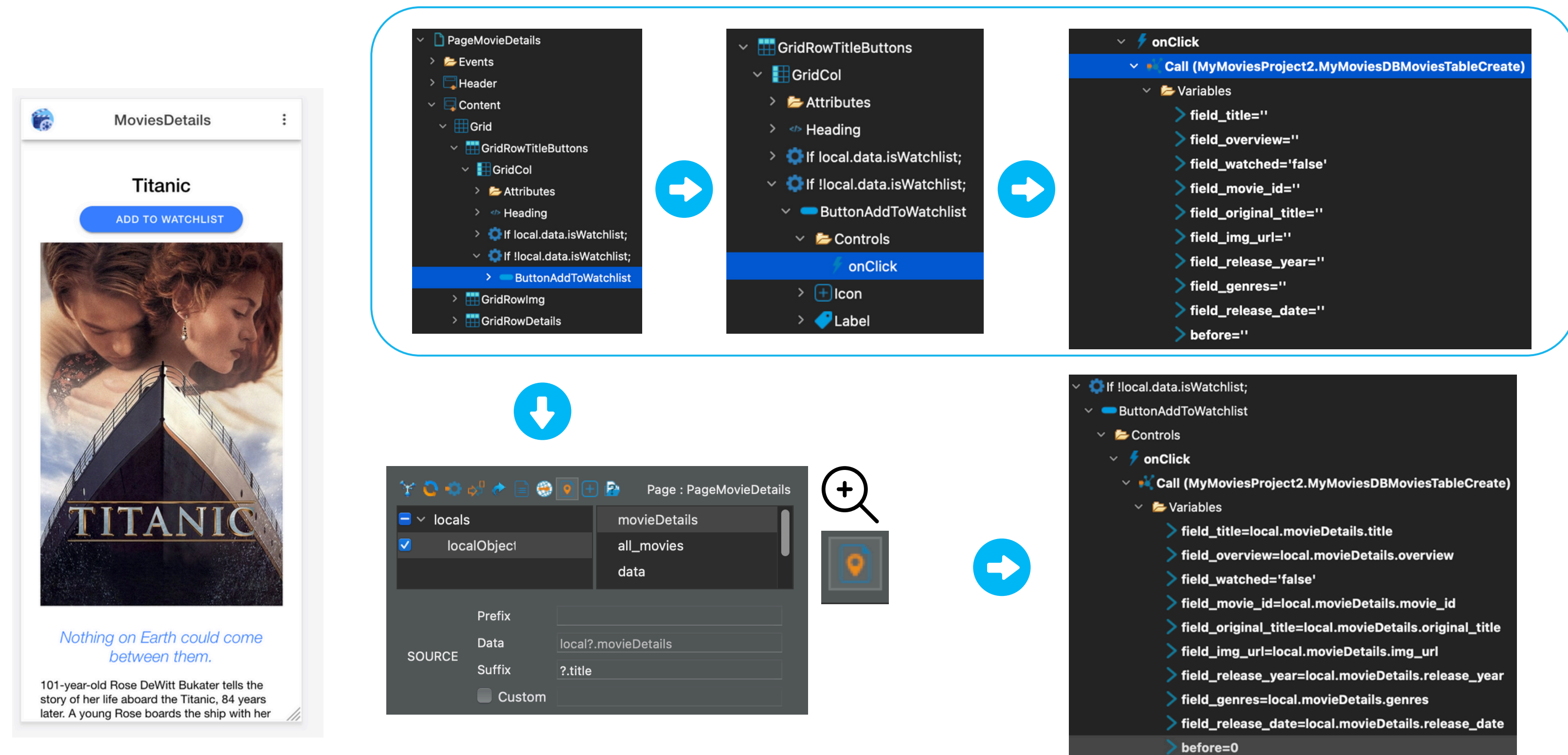
1997

SEEN



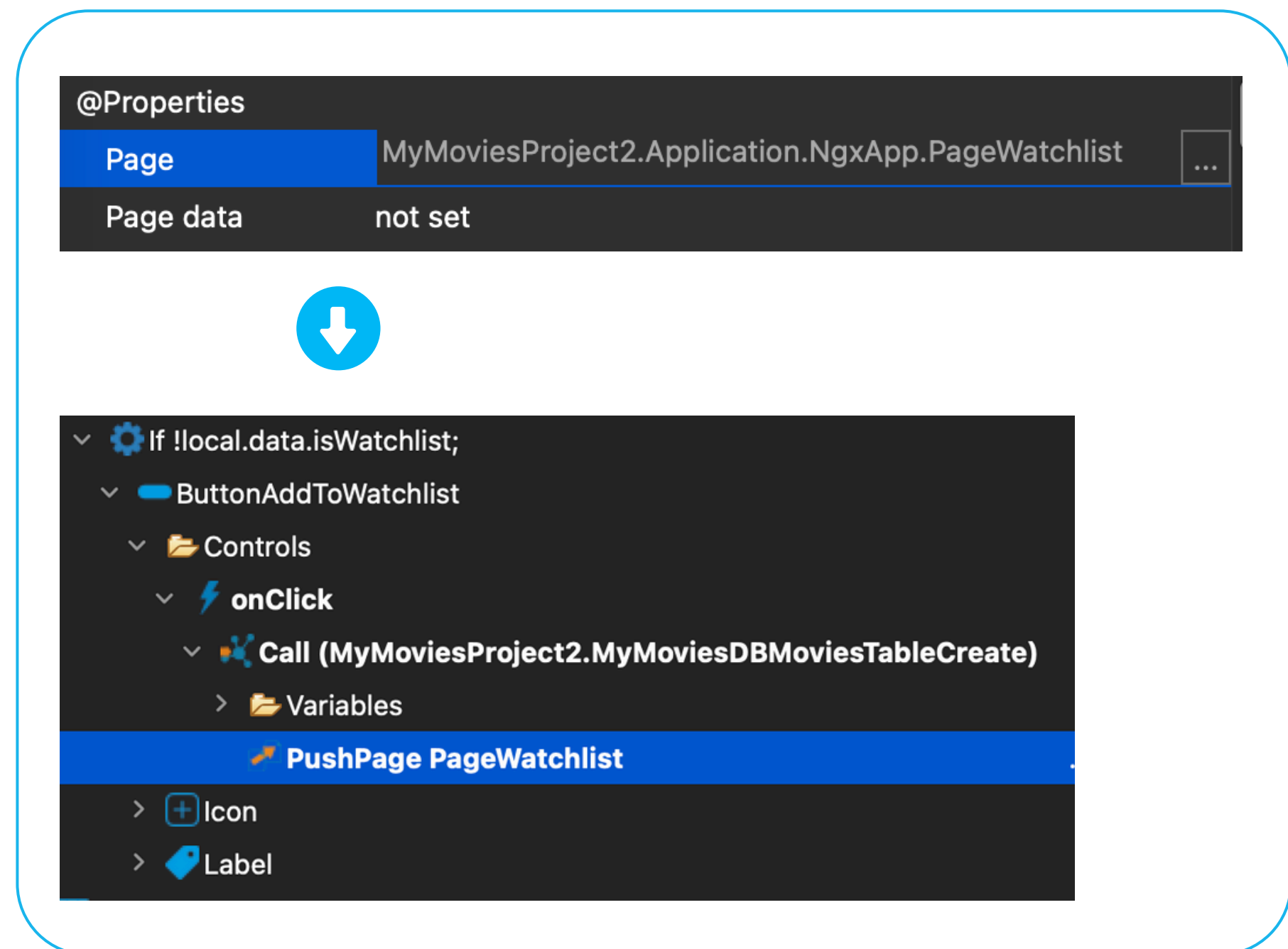
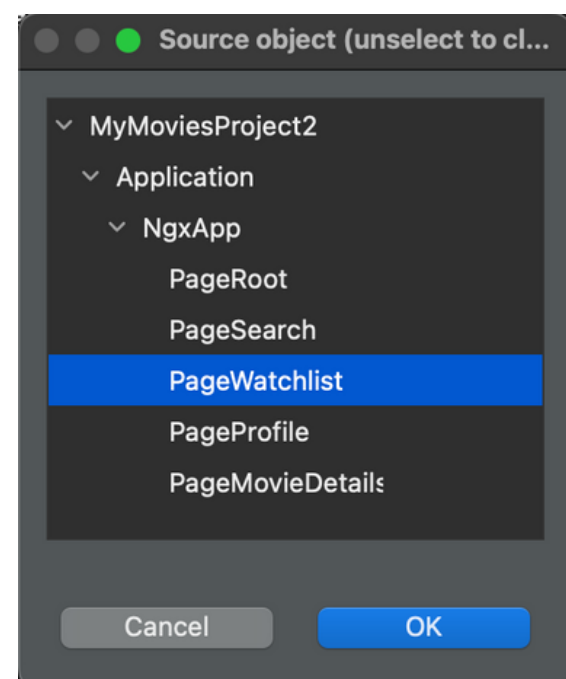
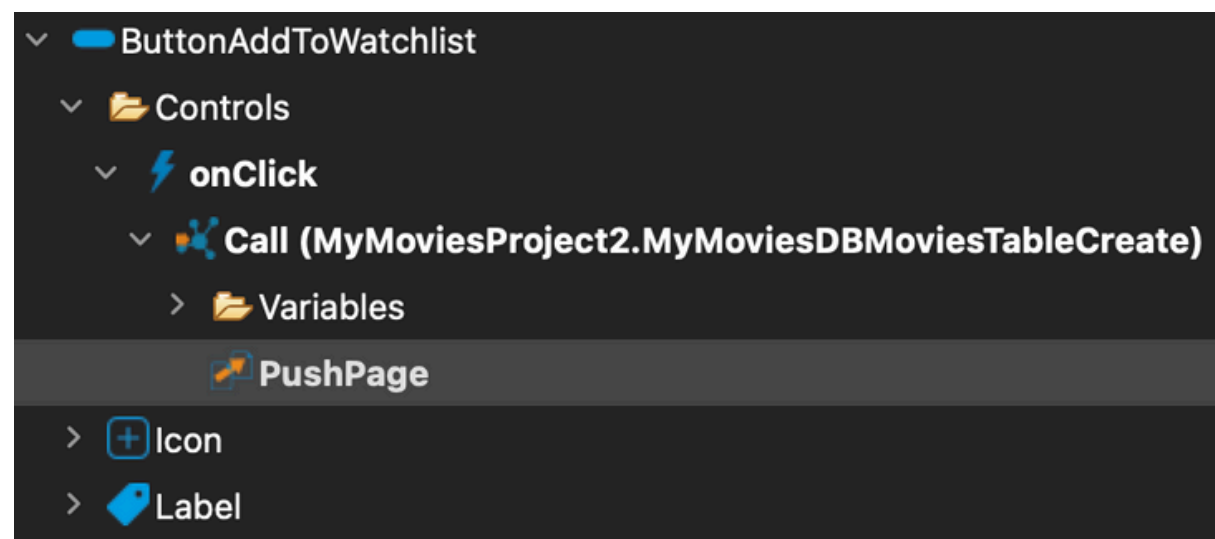
9.4 Implement ButtonAddToWatchlist

Let's do the same thing for the **ButtonAddToWatchlist** in **PageMovieDetails**

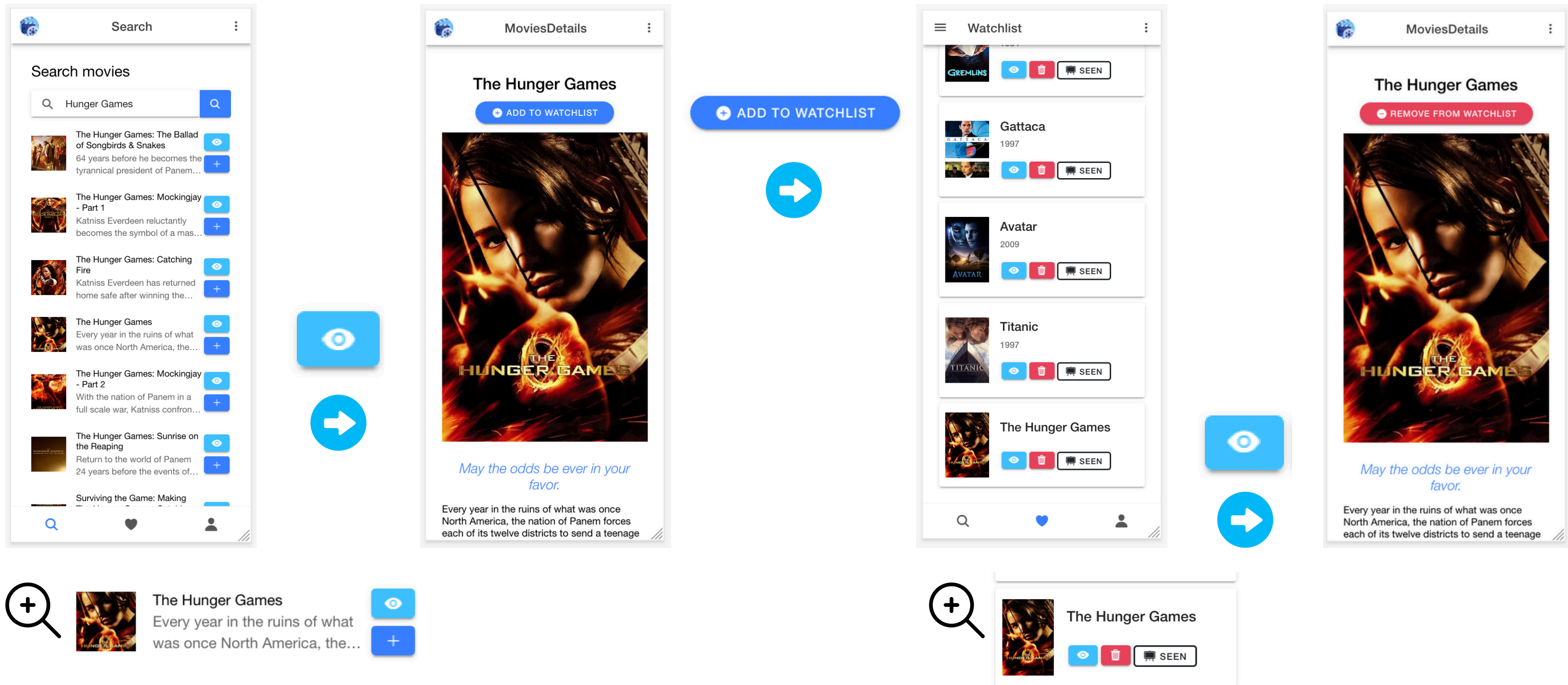


9.4 Implement ButtonAddToWatchlist

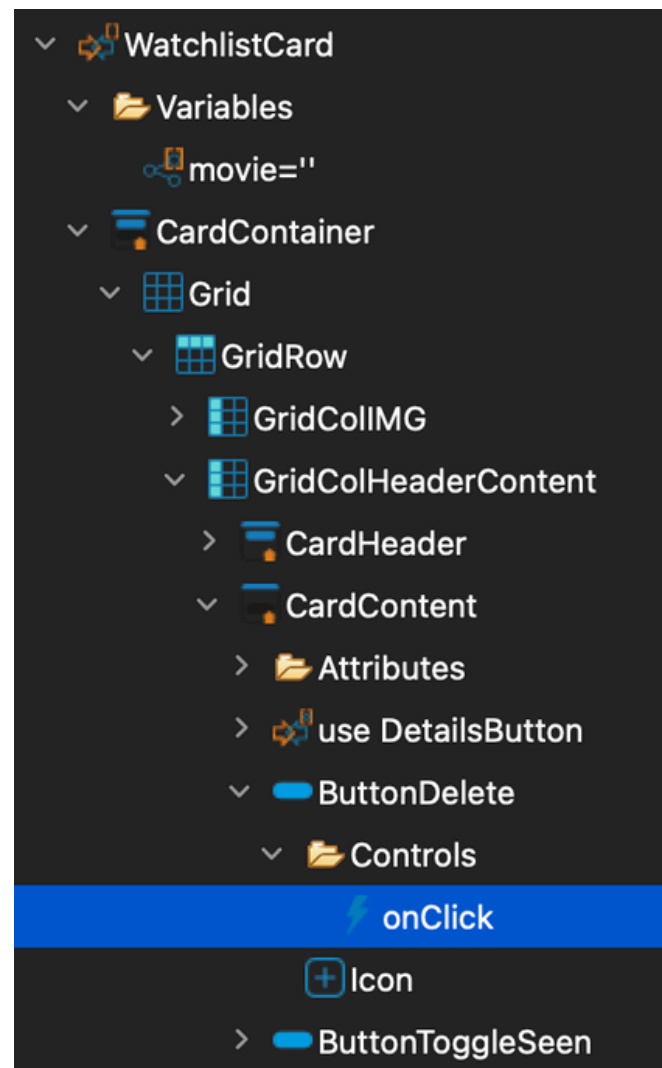
In the **CallSequence**, add a **PushPage** with **PageWatchlist** as value for the **Requestable** property.



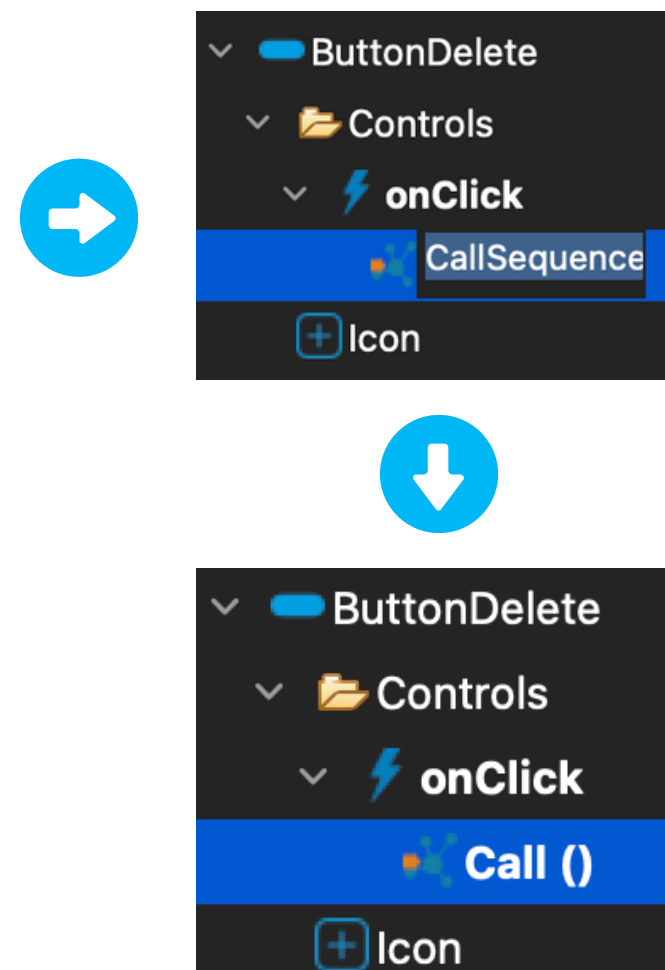
9.4 Implement ButtonAddToWatchlist



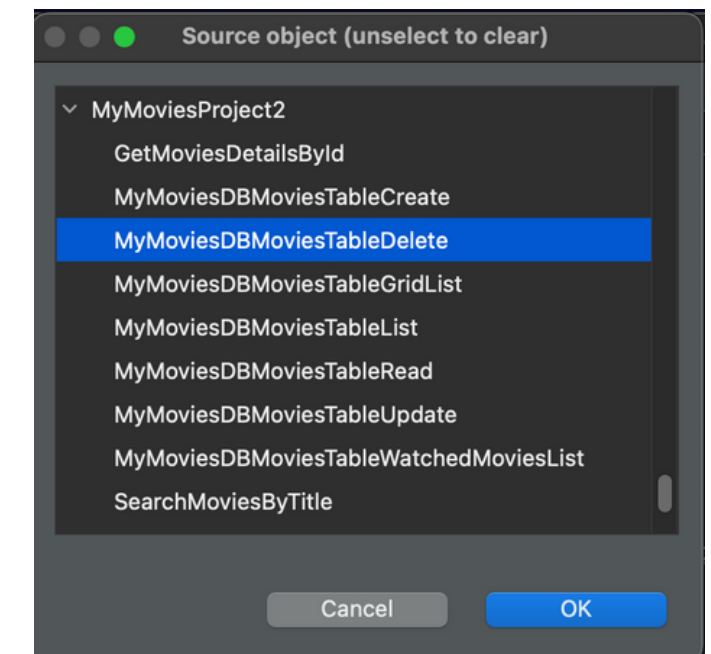
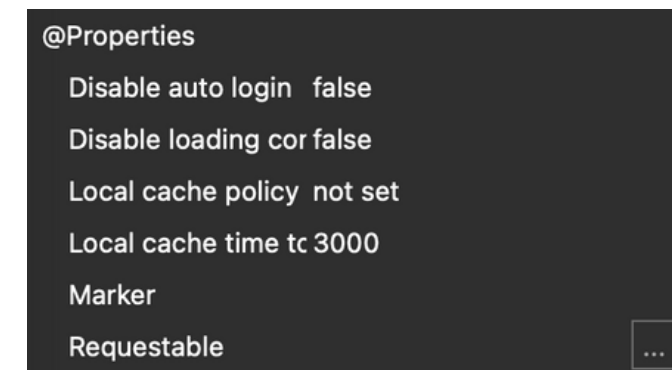
9.5 Implement ButtonDeleteFromWatchlist



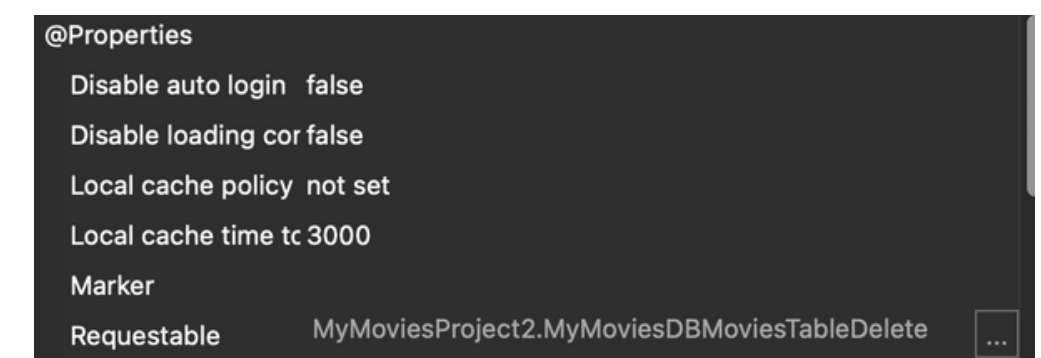
In the **onClick** event of **ButtonDeleteFromWatchlist**, add a **CallSequence**.



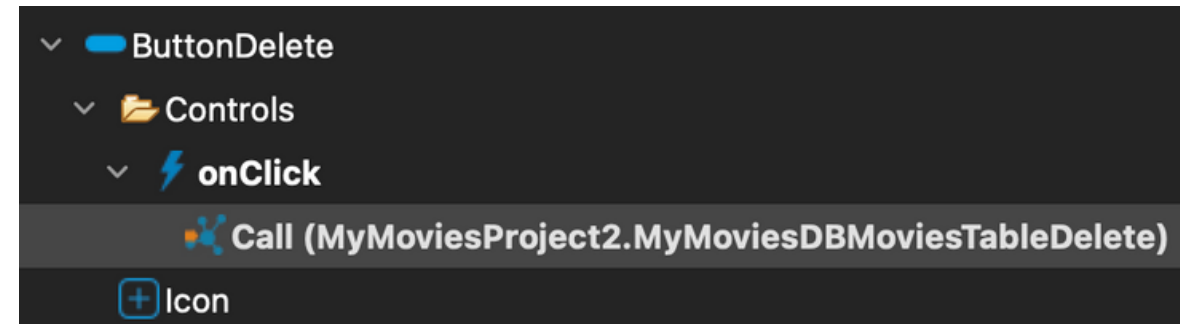
In the **CallSequence**, choose the sequence **MyMoviesDBMoviesTableDelete** as **value** for the **Requestable** property.



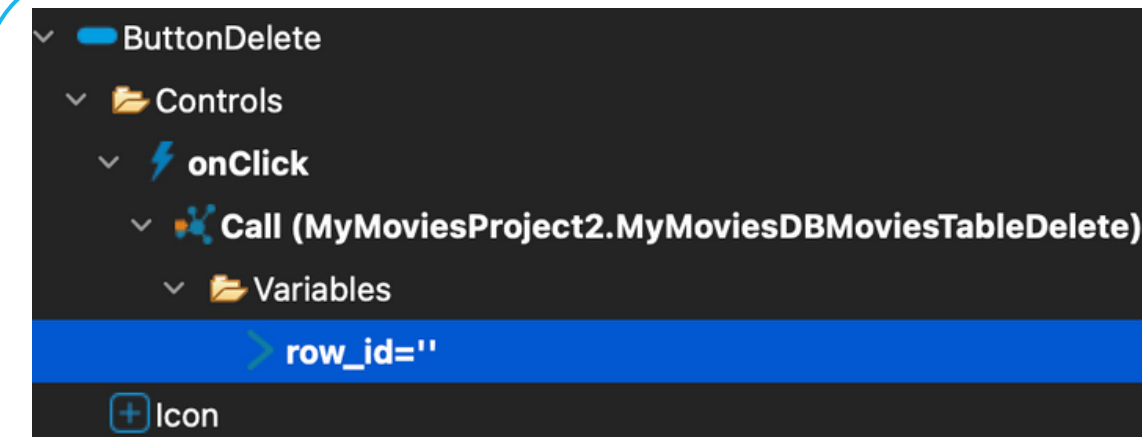
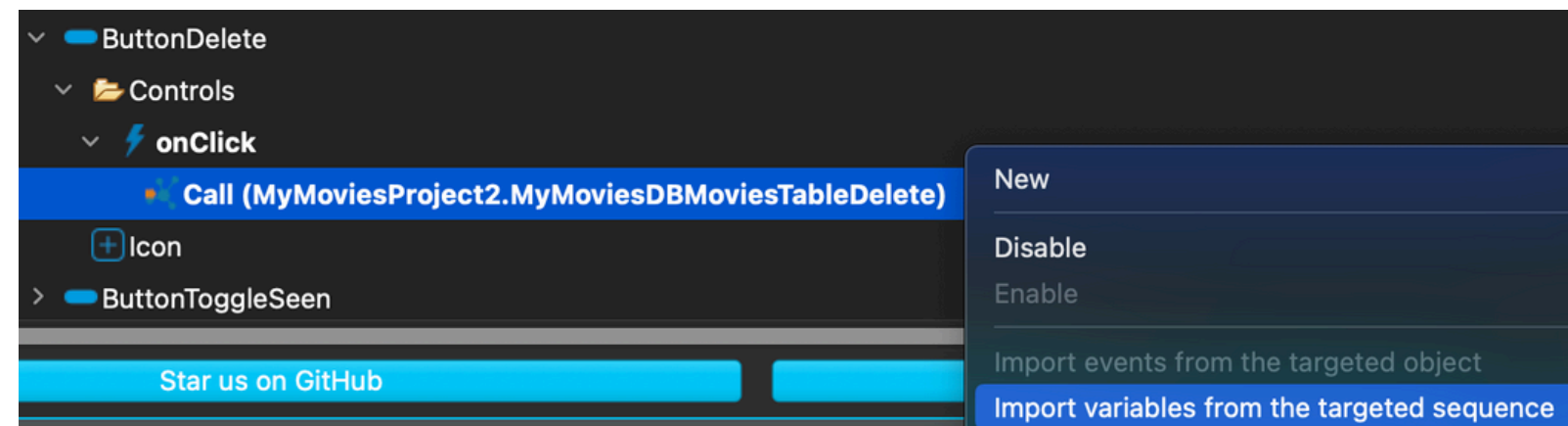
This **sequence** delete a row in the database.



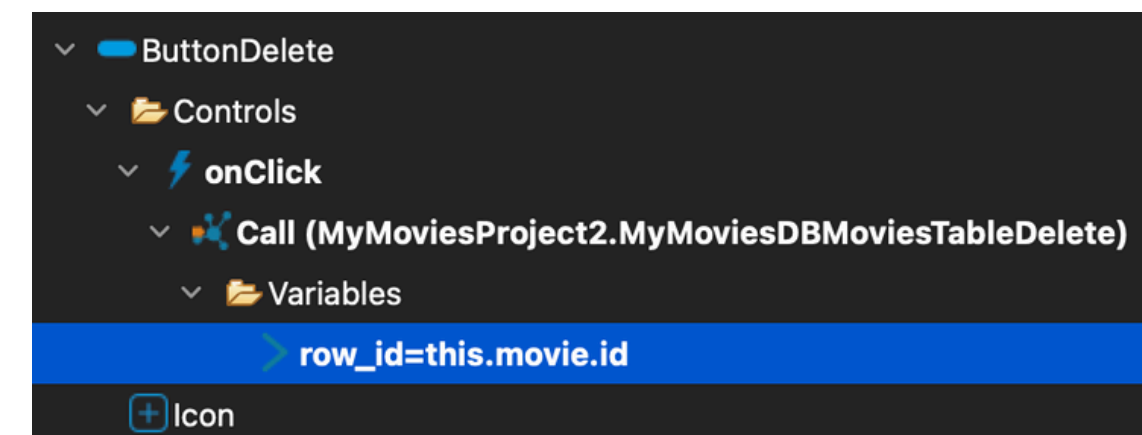
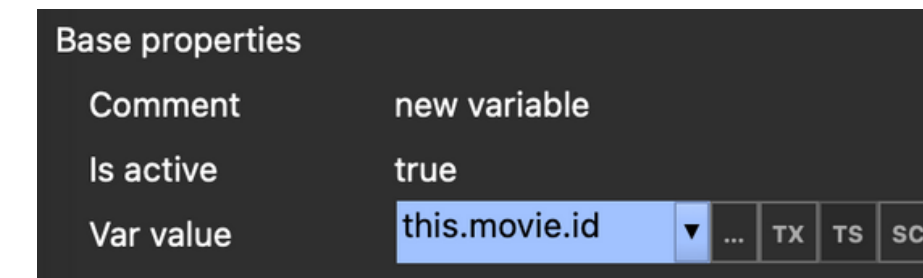
9.5 Implement ButtonDeleteFromWatchlist



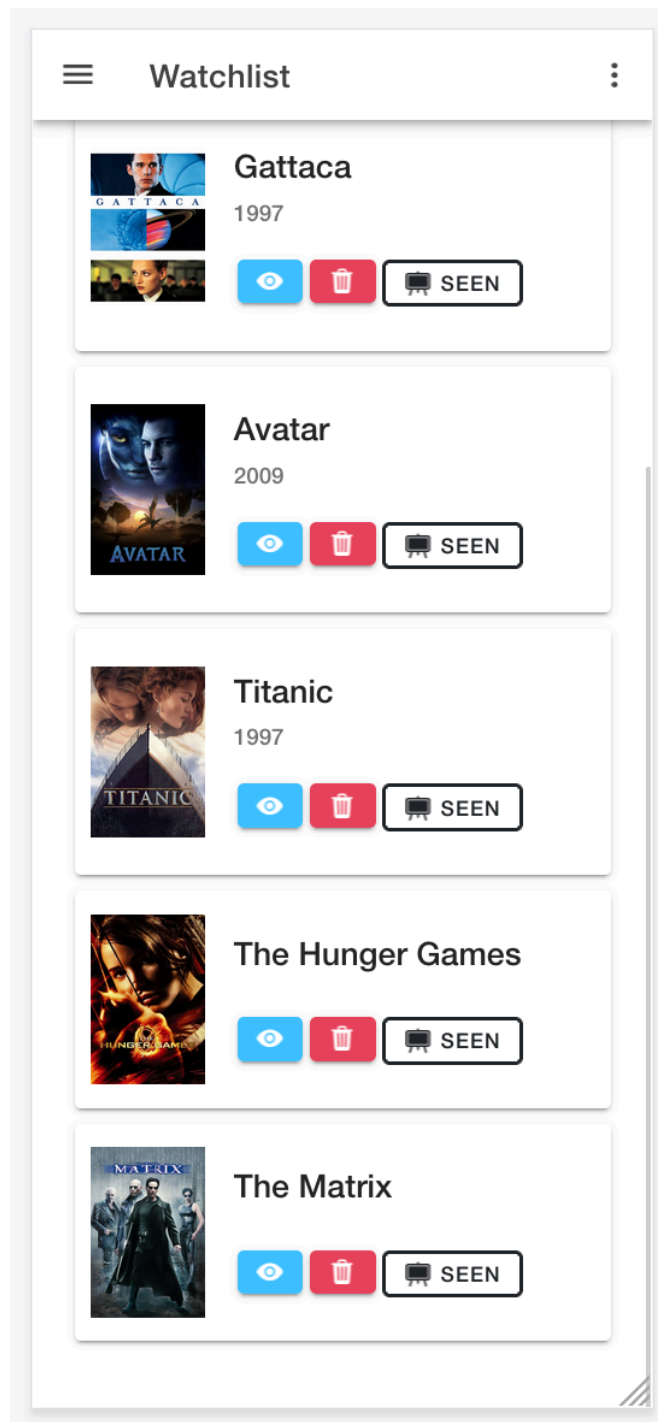
Right-click on the **CallSequence**
MyMoviesDBMoviesTableDelete
and select
Import variables from the the targeted sequence.



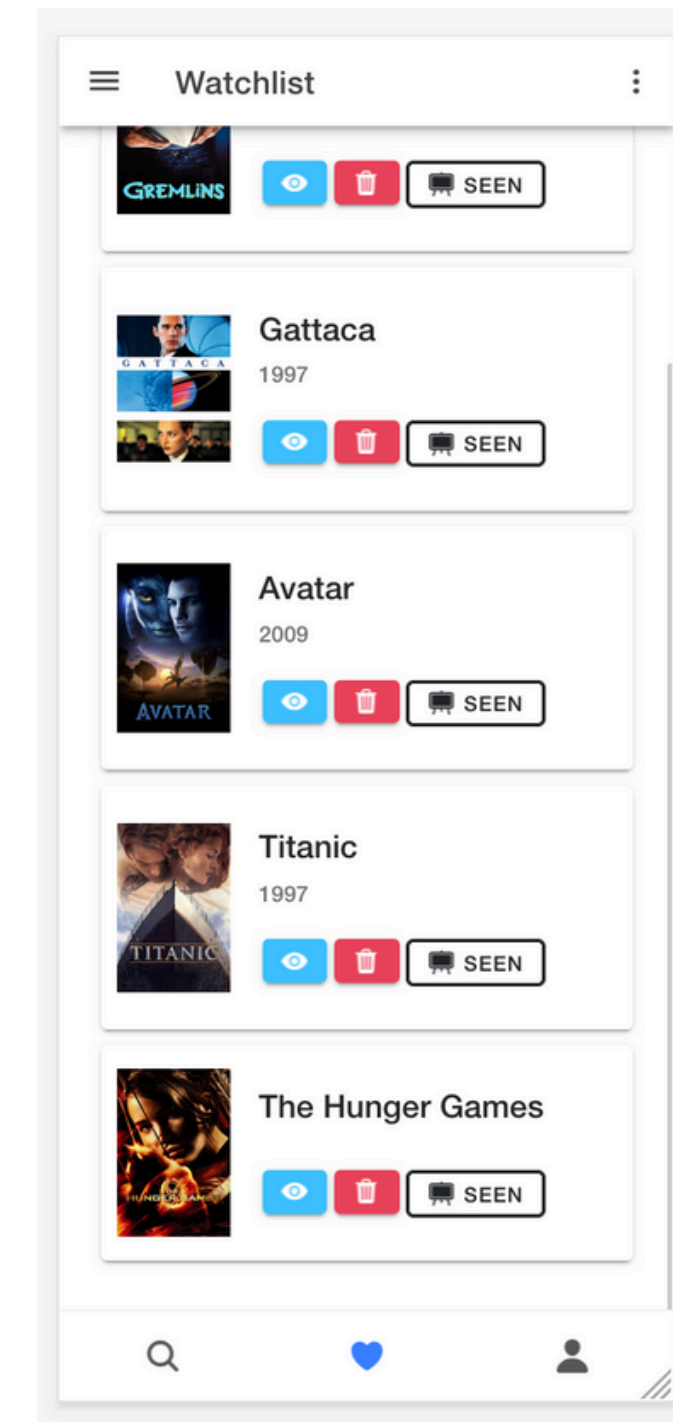
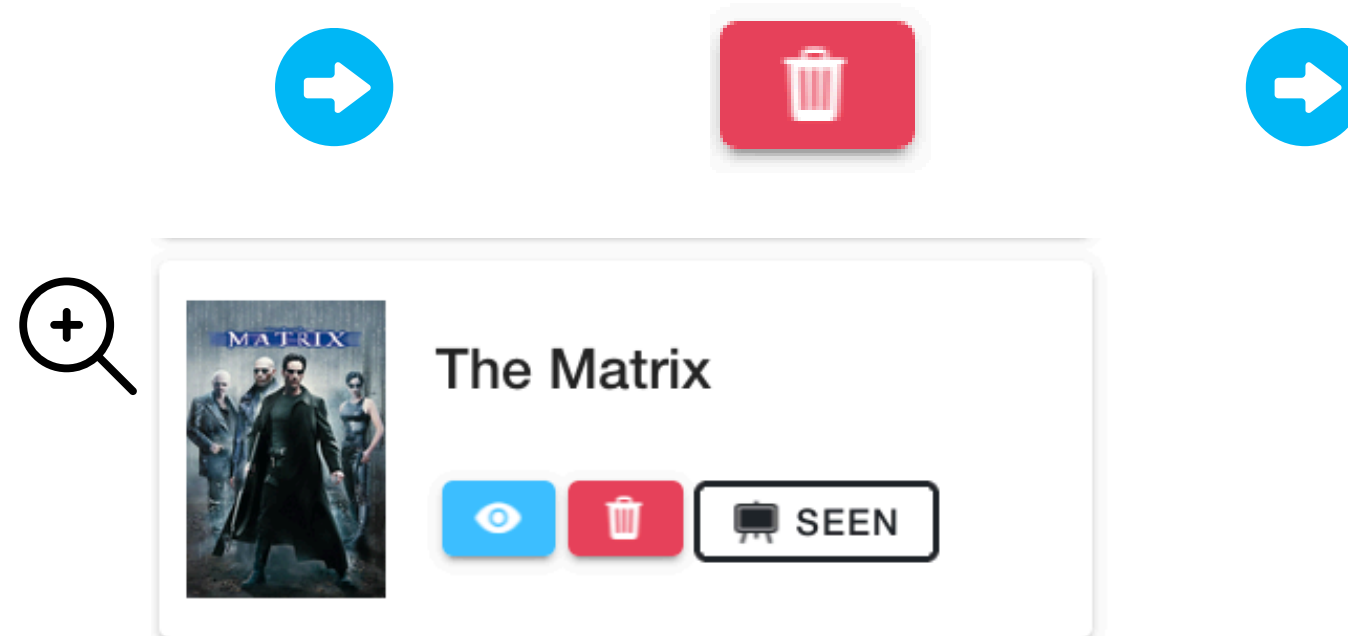
In the **row_id** variable's properties,
enter **this.movie.id** as value for the **Var value** property.



9.5 Implement ButtonDeleteFromWatchlist

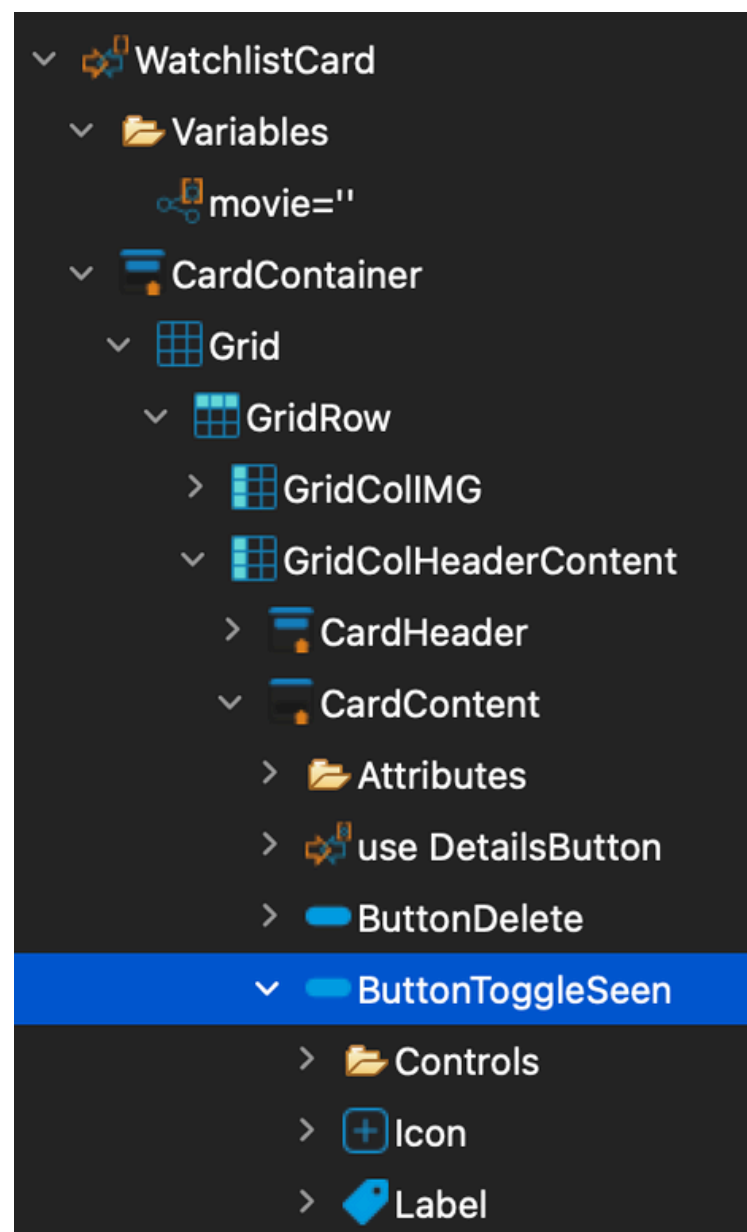


When we click on the **ButtonDeleteFromWatchlist**,
in an item of **PageWatchlist**,
the item is deleted from the database,
and disappears after the page is refreshed.

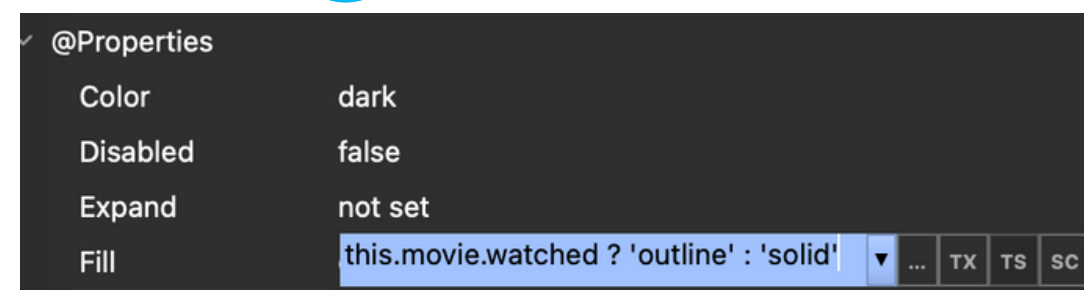
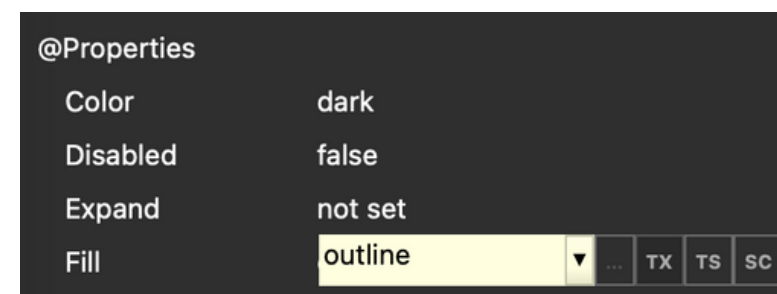
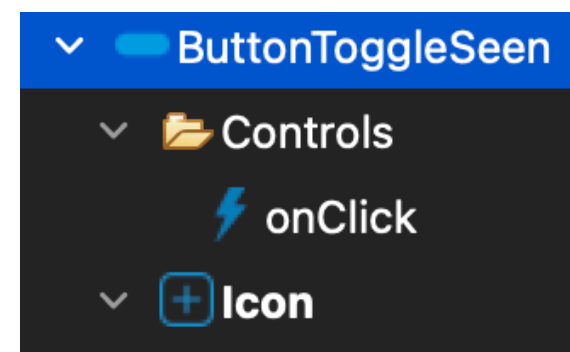


9.6 Implement ButtonToggleSeen

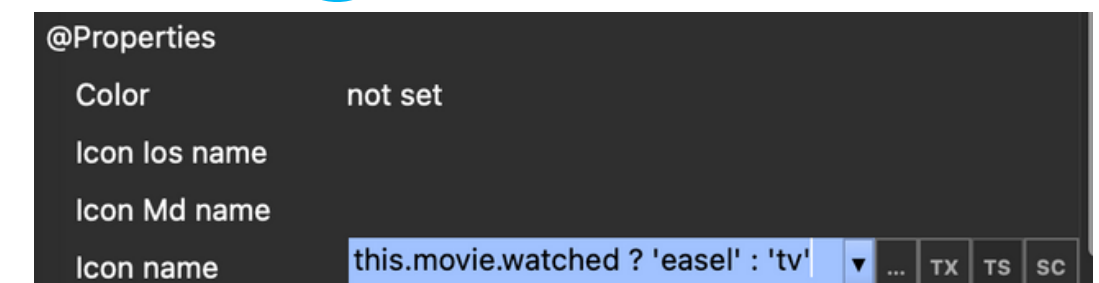
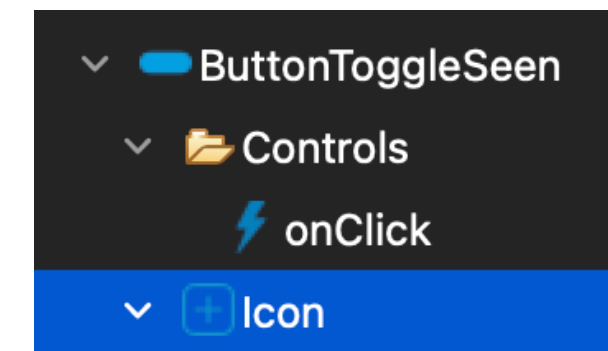
In ButtonToggleSeen,
in the shared component
WatchlistCard



In ButtonToggleSeen's properties
in the **Fill** property,
replace the value **outline** in Text mode
by **this.movie.watched ? 'outline' : 'solid'**.



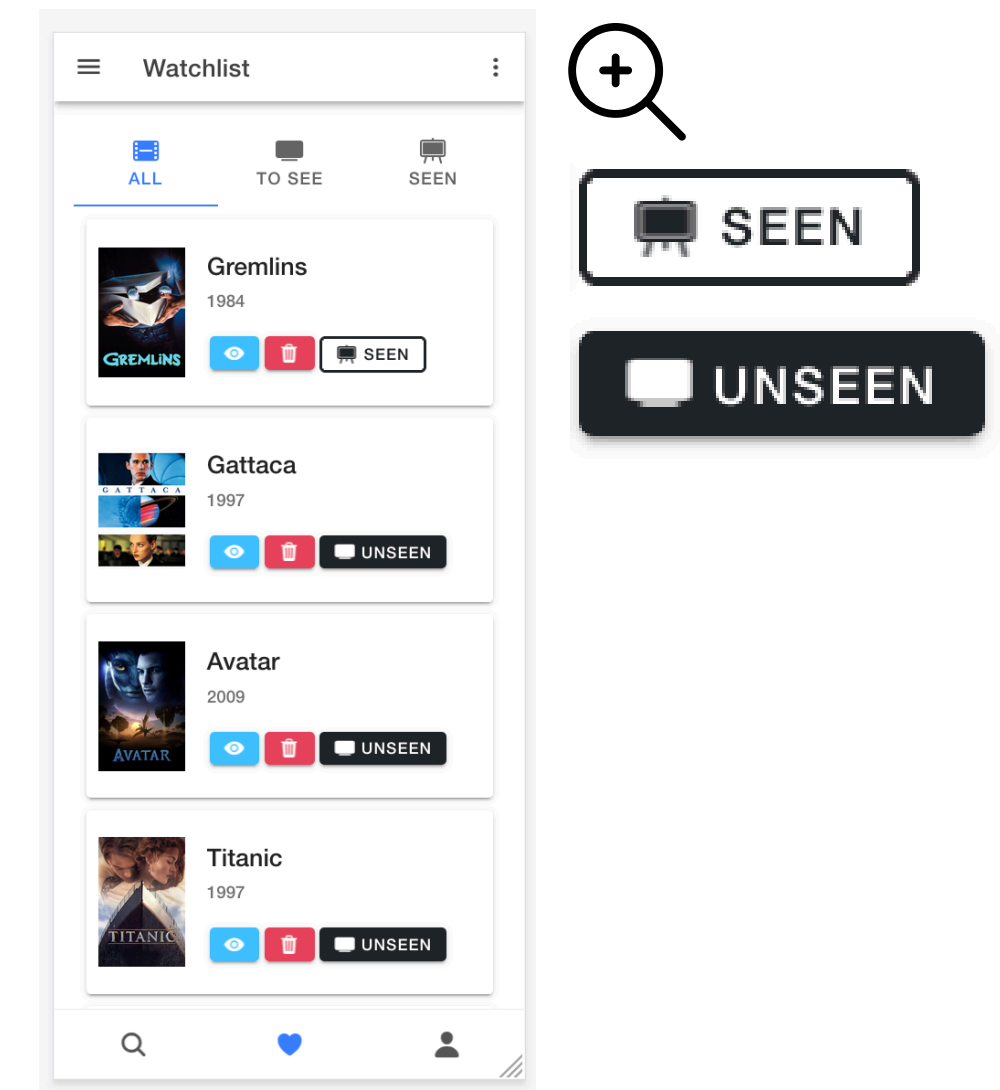
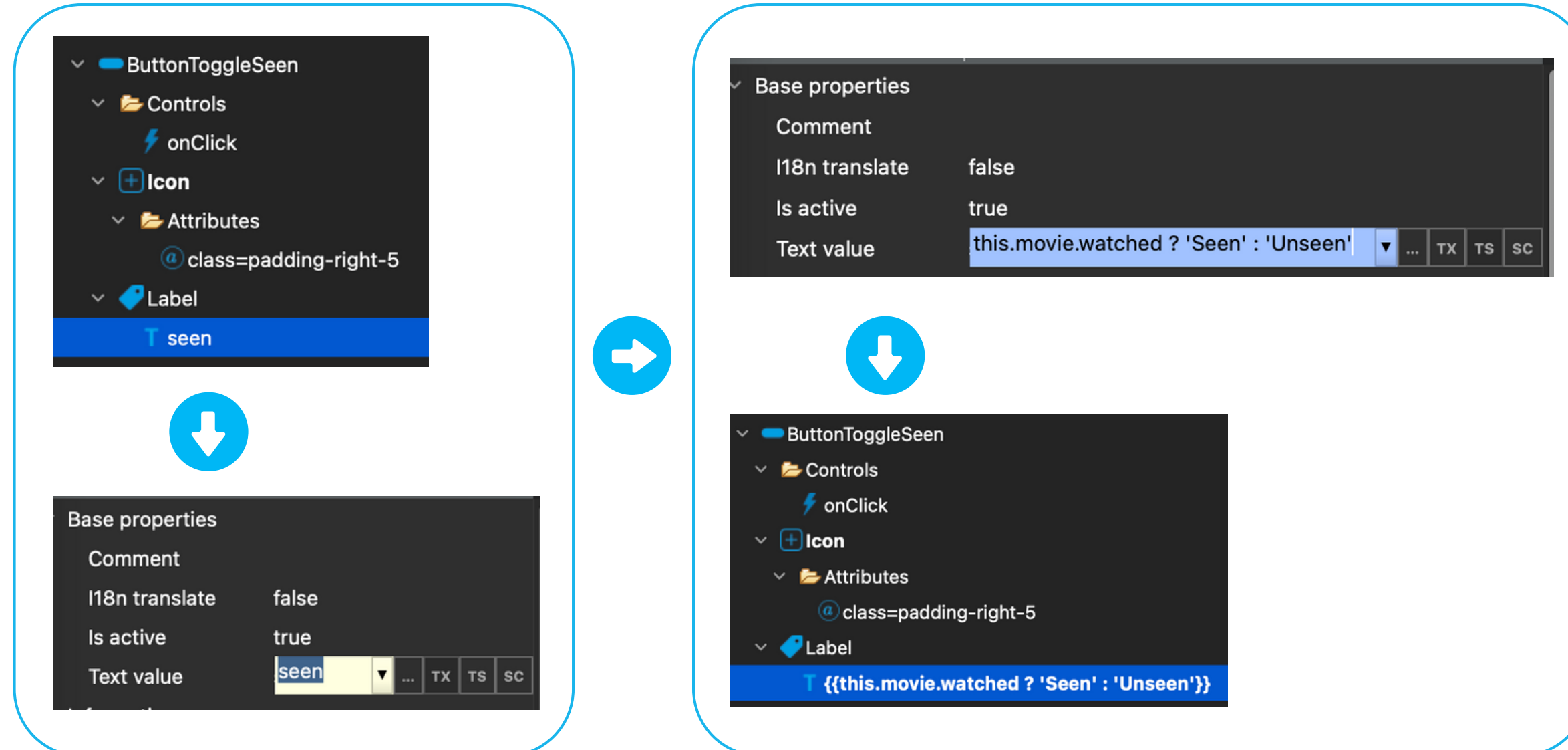
In the Icon's properties
in the **Icon name** property,
replace the value **'easel'**
by **this.movie.watched ? 'easel' : 'tv'**.



9.6 Implement ButtonToggleSeen

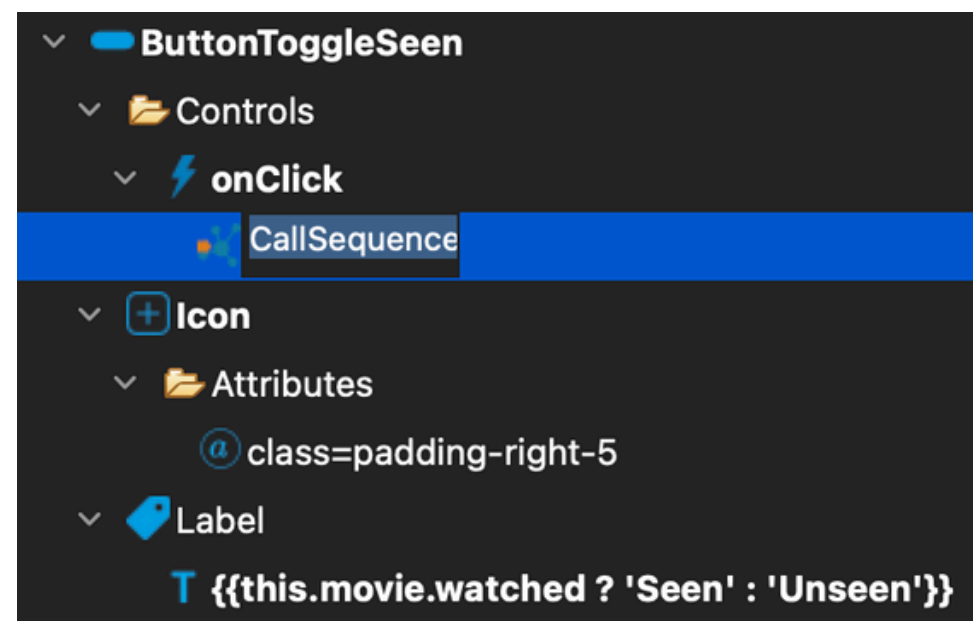
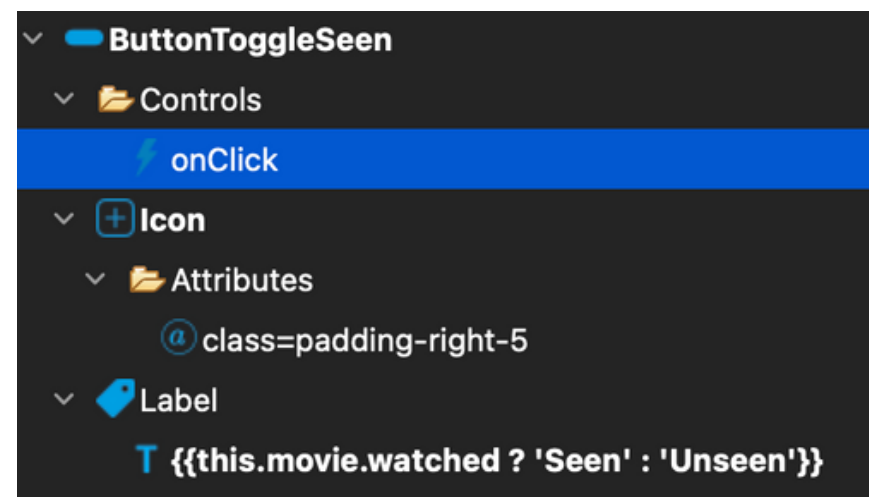
Inside the **Label**, in the **Text** element's properties
in the **Text value** property, replace the **value seen** in **Text mode**
by **this.movie.watched ? 'Seen' : 'Unseen'** in **TS**.

If the **watched** field is **true**
in the **database**,
the button appears as **SEEN**,
if **false**, it appears as **UNSEEN**.

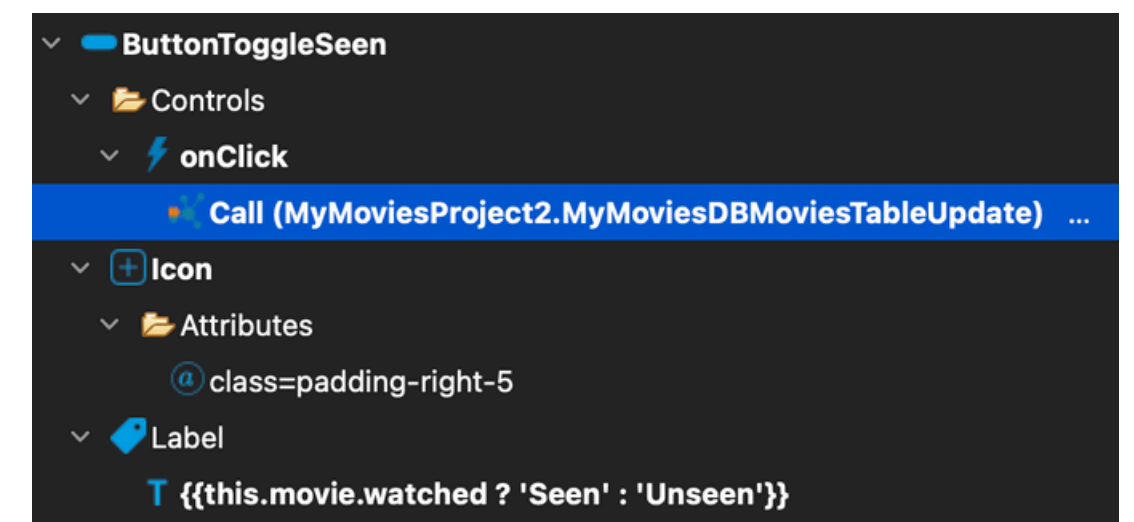
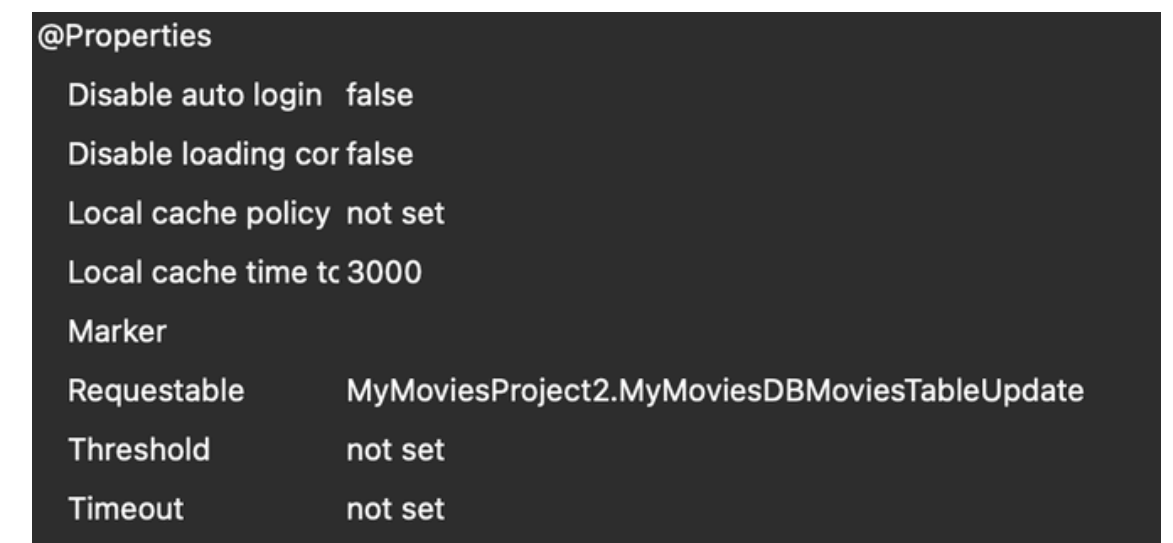
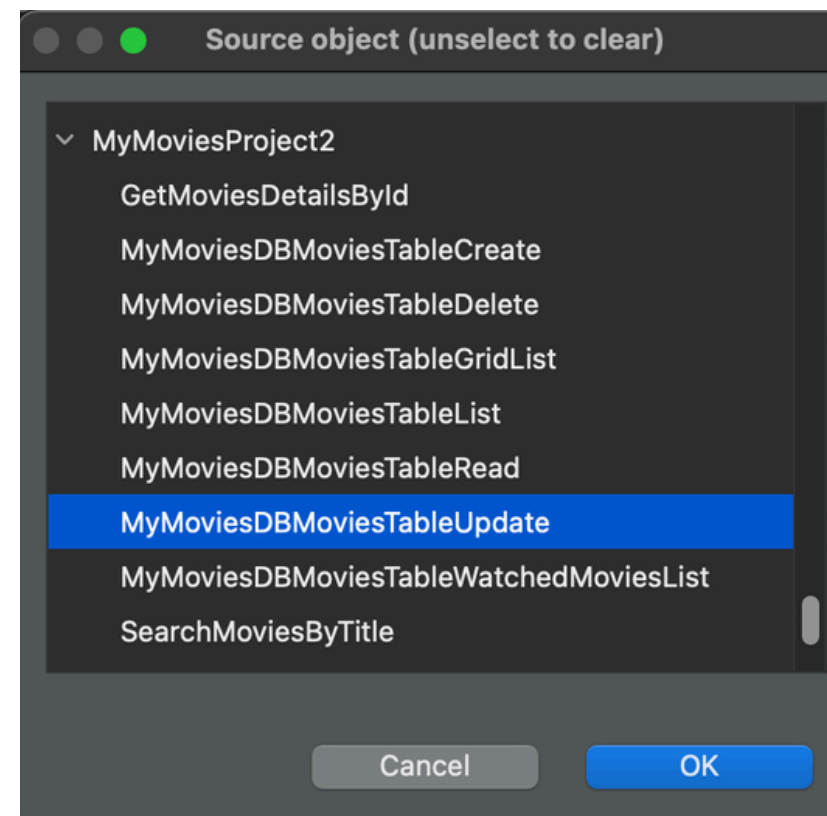


9.6 Implement ButtonToggleSeen

In the **onClick** event of **ButtonToggleSeen**, add a **CallSequence**.



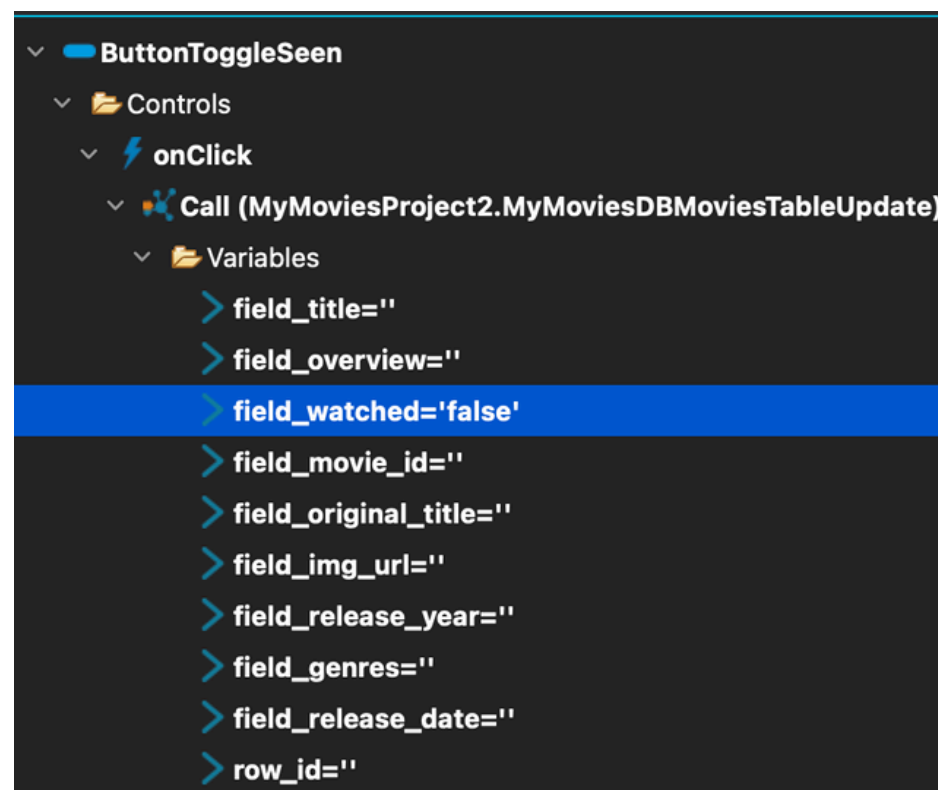
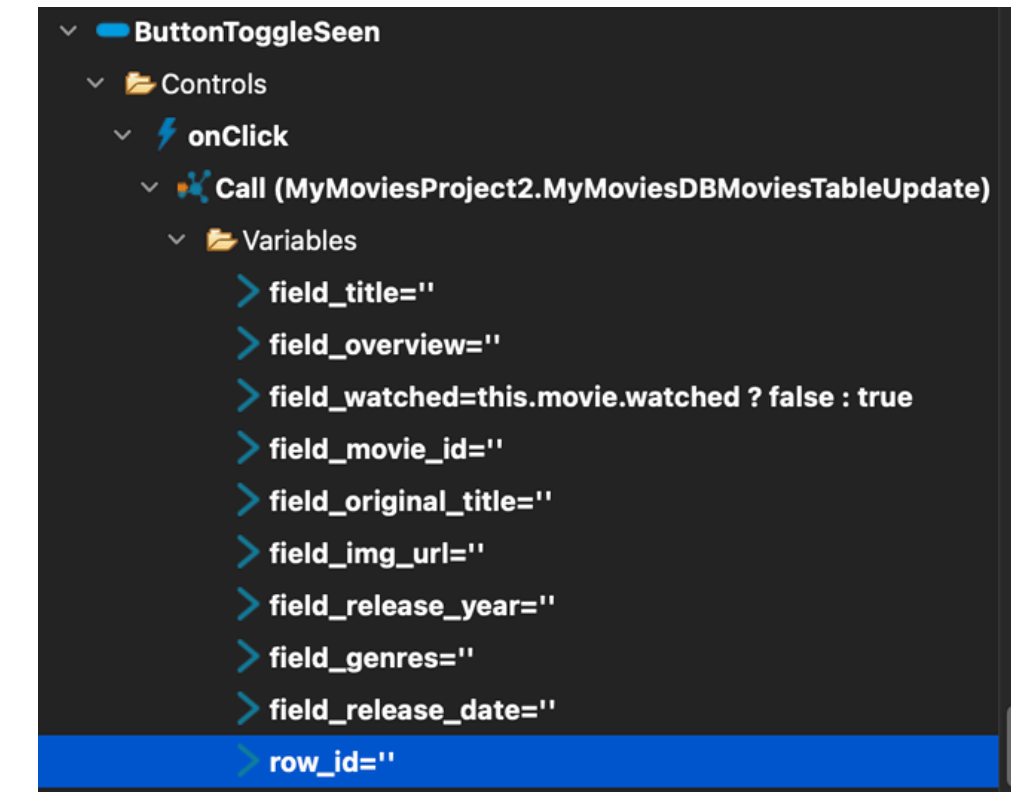
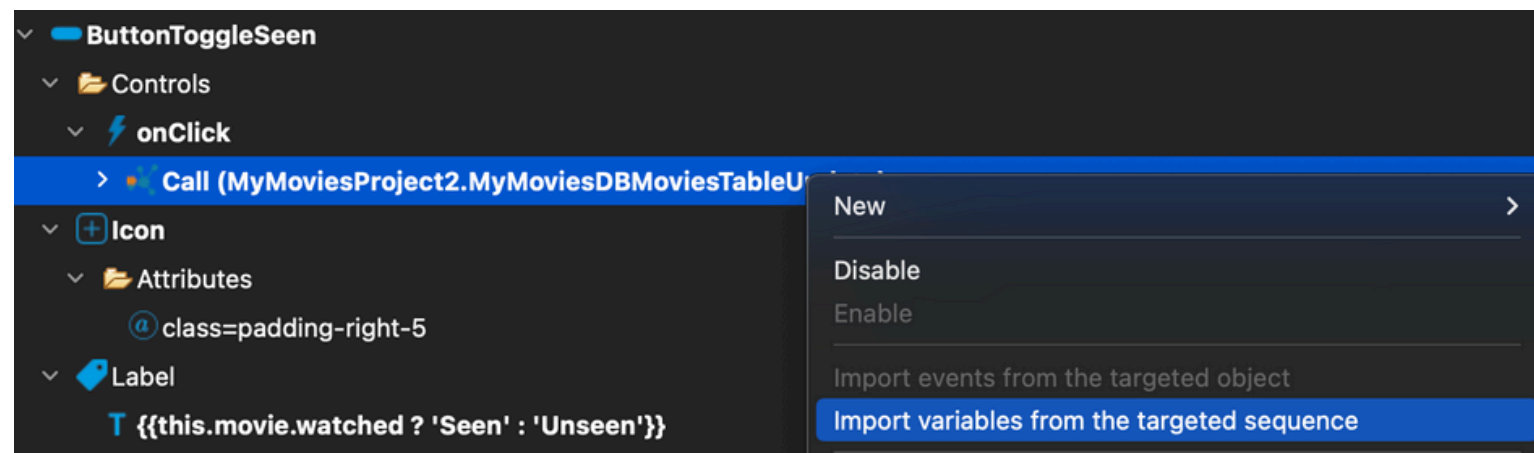
In the **CallSequence**, choose the sequence **MyMoviesDBMoviesTableUpdate** as **value** for the **Requestable** property.



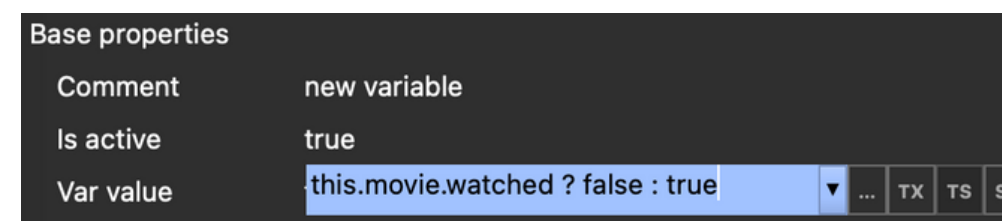
This **sequence** **updates** a row in the database.

9.6 Implement ButtonToggleSeen

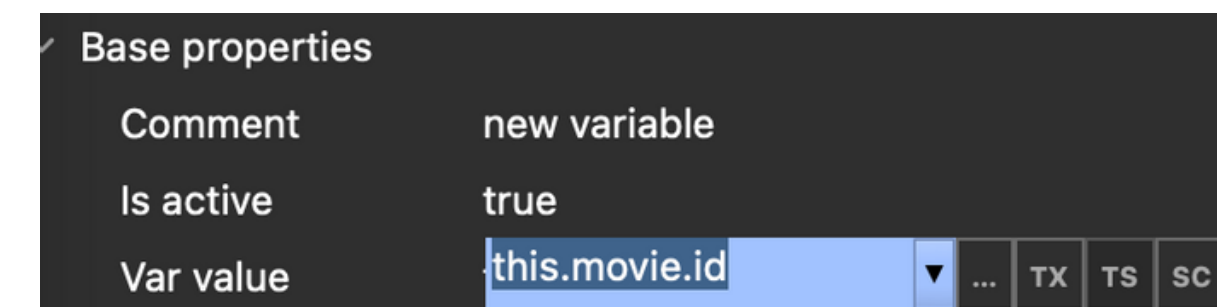
Right-click on the **CallSequence** **MyMoviesDBMoviesTableUpdate** and select **Import variables from the the targeted sequence**.



In the **field's** watched properties, enter in TS **this.movie.watched ? false : true** as **value** for the **Var value** property.

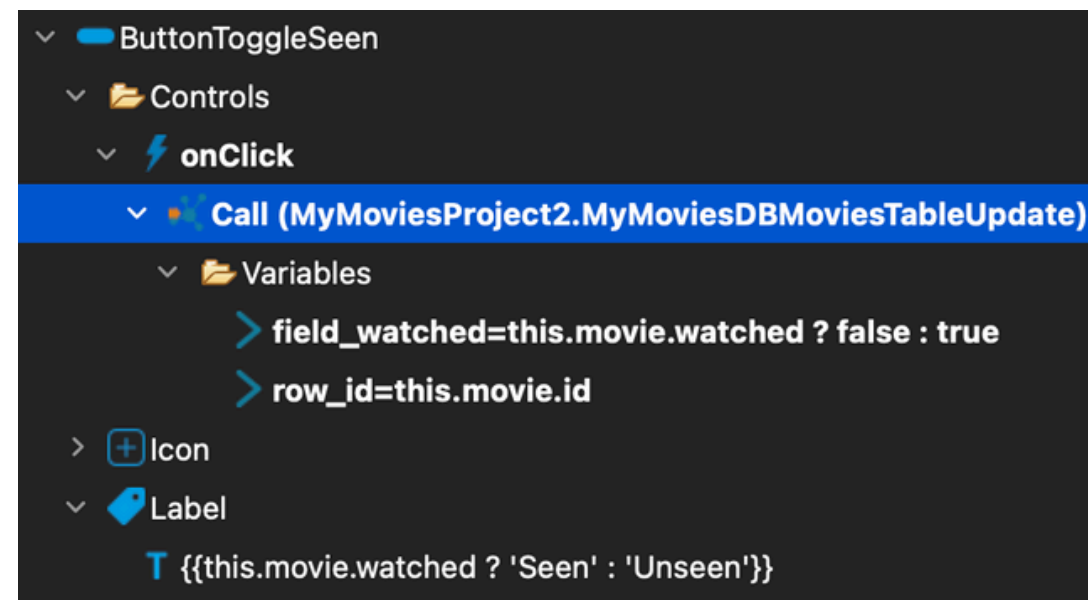


In the **row_id** variable's properties, enter **this.movie.id** as **value** for the **Var value** property.

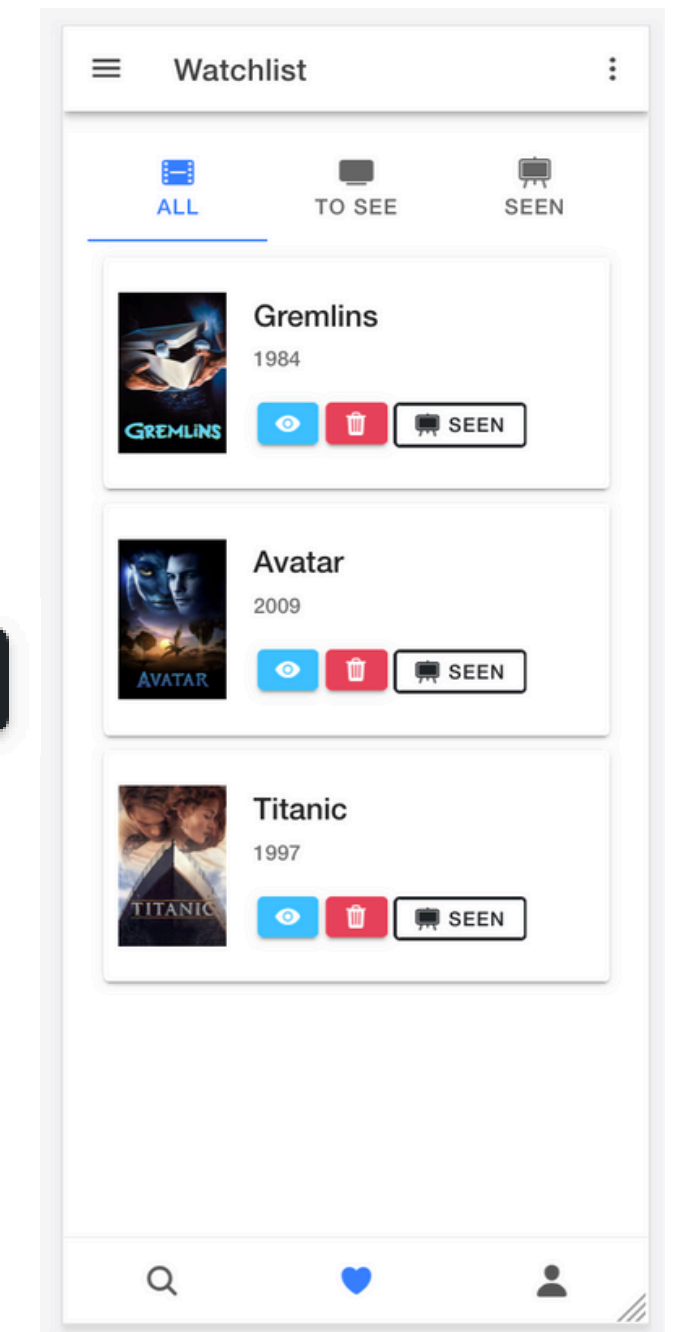
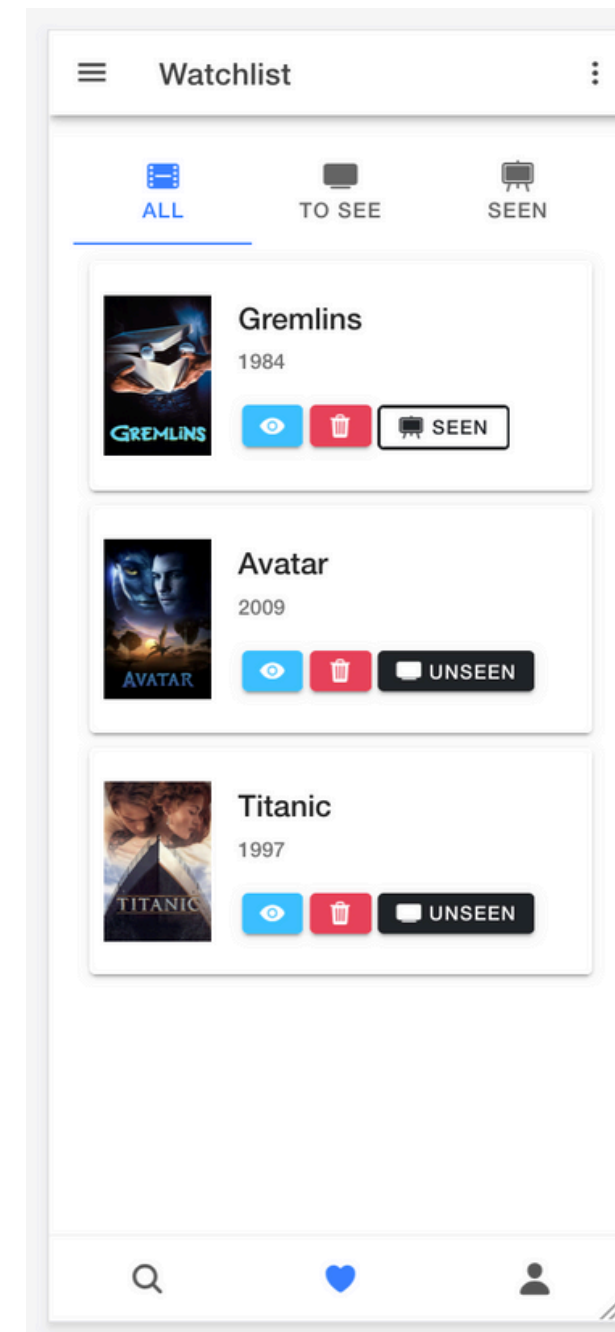


9.6 Implement ButtonToggleSeen

Remove the other variables.



When we click on the **ButtonToggleSeen**,
in an item of **PageWatchlist**,
the item is updated in from the database,
and changes after the page is refreshed.



EXERCISES

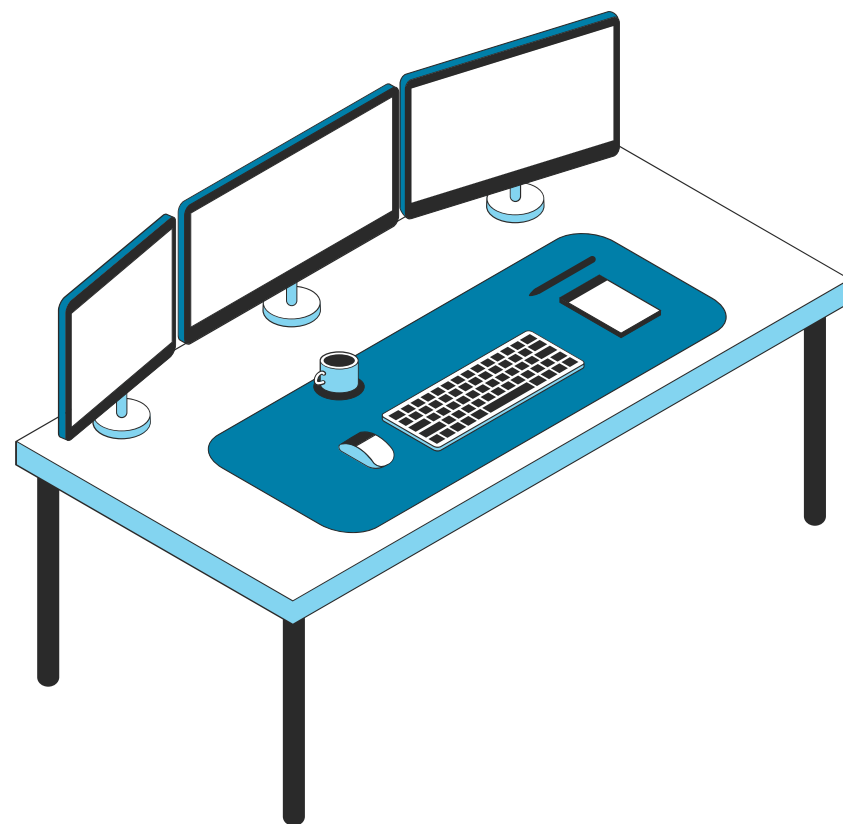
1 – Implement the Header shared component in all pages

2 – Create a shared action to refresh the data after

- ButtonAddToWatchlist
- ButtonDeleteFromWatchlist
- ButtonToggleSeen

3 – Implement the ButtonAddToWtachlist in PageMovieDetails

10 – AppEvents & Global variables



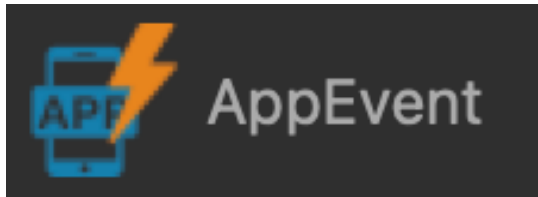
10.1 AppEvent component

10.2 Add an AppEvent component

10.3 Add a Global variable to an AppEvent

10.4 Display the Global variable in a page

10.1 AppEvent component



The **AppEvent component** is added under the **Application (NgxApp)** in the treeview and used to manage **App lifecycle events** which are fired during various stages of navigation.

Under an **AppEvent**, you can add any type of action, like **CallSequence** or **CustomAction components**.

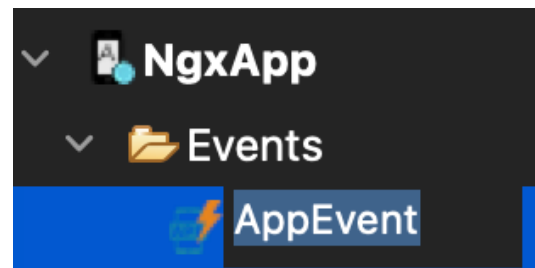
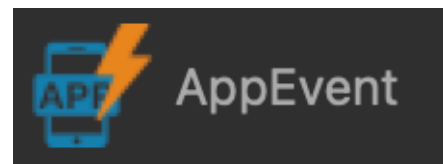
Application Events can be, for example:

- **onAutoLogin:** Will be fired when performing an automatic 'login' on the server.
- **onSessionLost:** When the app loses the session to the Convertigo server.
- **onNetworkReachable:** Will be fired as soon as the app starts and that network connectivity has been tested against Convertigo server endpoint.
- **onNetworkUnreachable:** Will be fired if you do have network connectivity but that the Convertigo server endpoint is unreachable.
- **onNetworkOffline:** Will be fired as soon as the device loses network connectivity

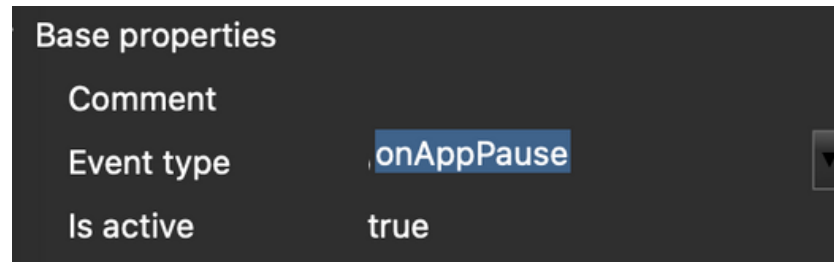
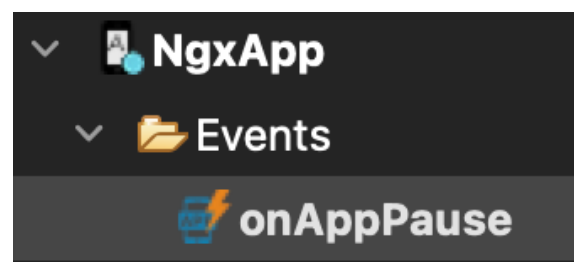


10.2 Add an AppEvent component

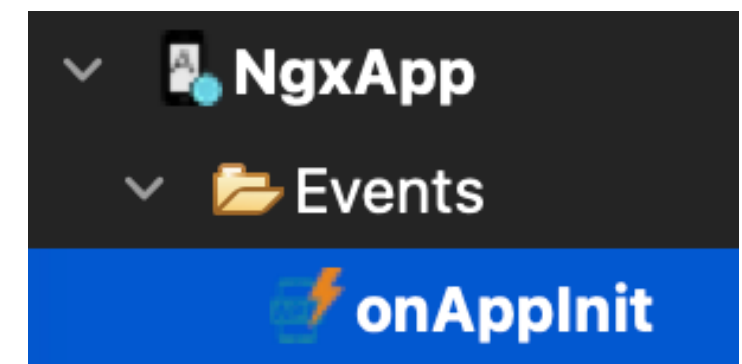
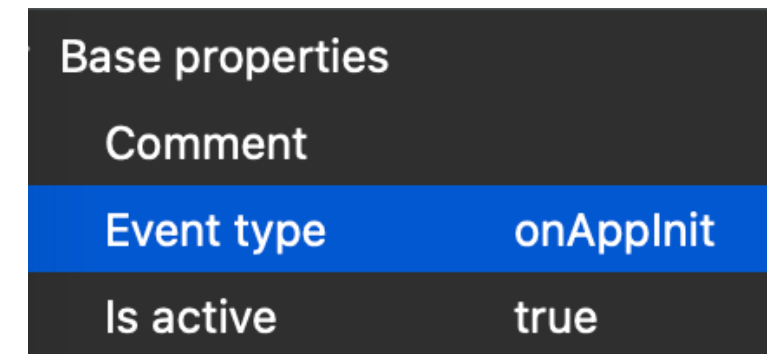
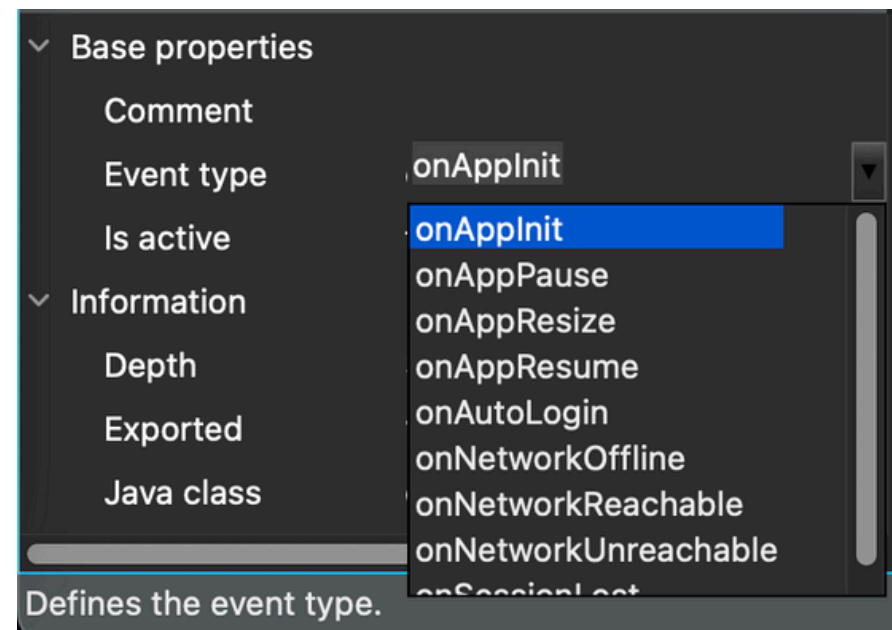
Add the **AppEvent** component under **NgxApp** in the treeview.



The **AppEvent** is **onAppPause** by default.

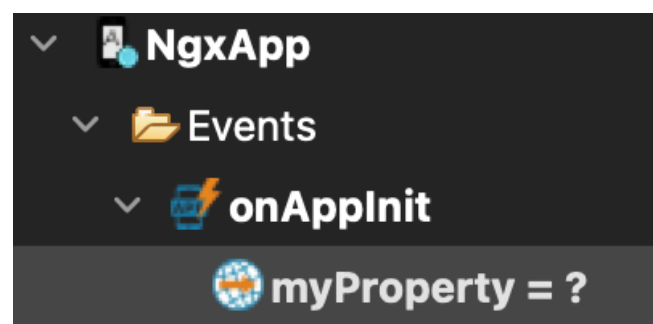
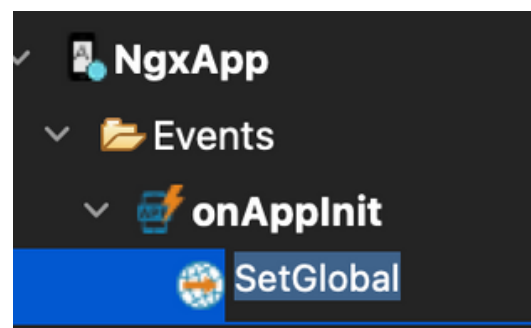
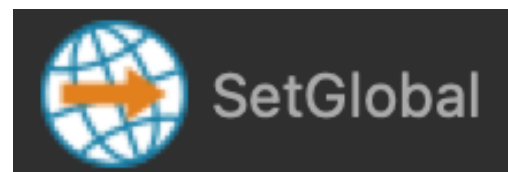


In the **AppEvent's** properties, change the **value** of the **Event type** property to **OnAppInit**.



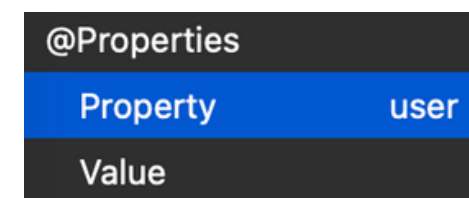
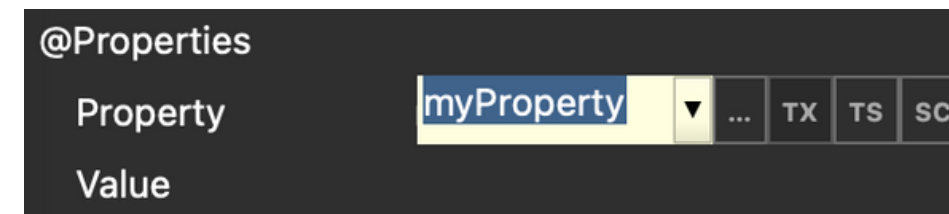
10.3 Add a Global variable to an AppEvent

Add a **SetGlobal** component under the **AppInit** AppEvent.



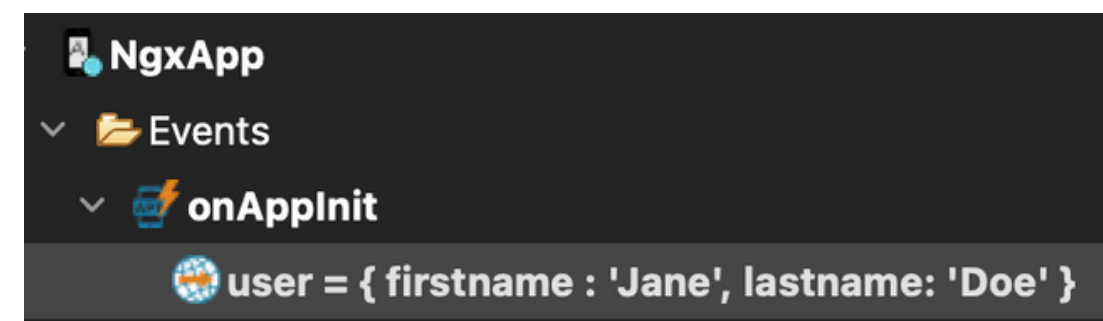
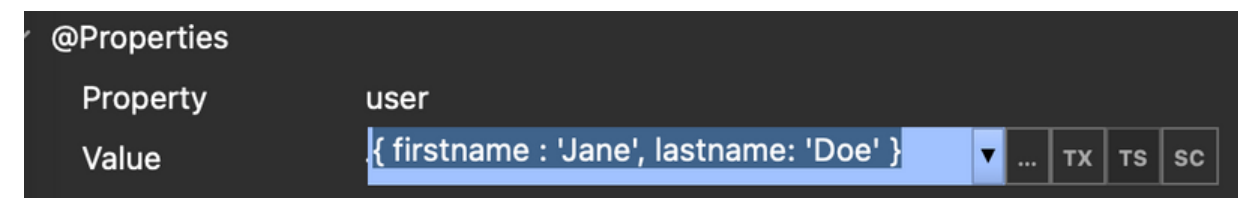
In the **SetGlobal's** properties,

set the **value** of the **Property** property to **user** in **Text mode (TX)**.



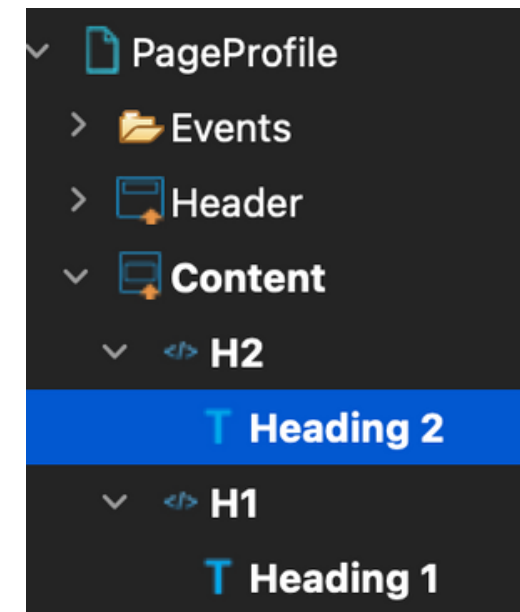
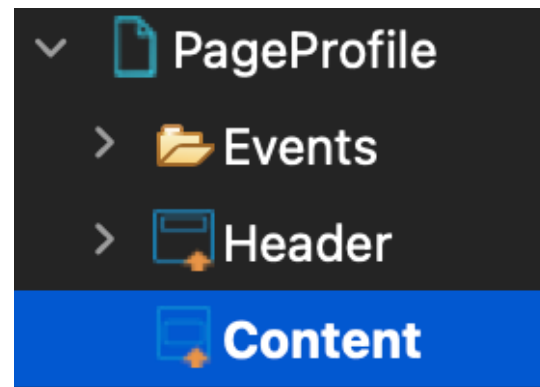
Set the **value** of the **Value** property to

{ firstname : 'Jane', lastname: 'Doe' } in **TS scope**.

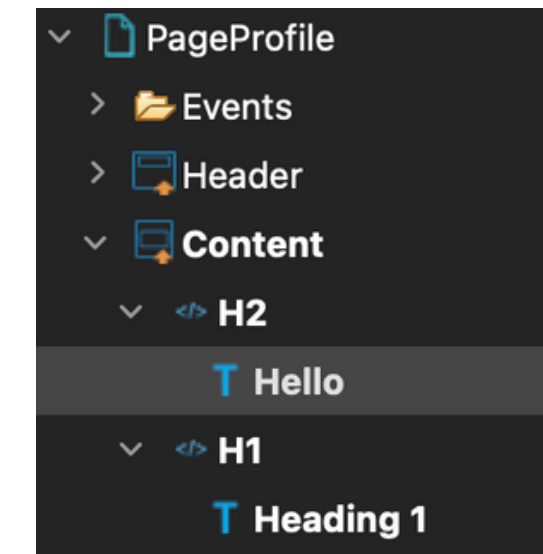


10.4 Display the Global variable in a page

In **PageProfile**,
in the **Content**,
add a **H2** and a **H1** components.

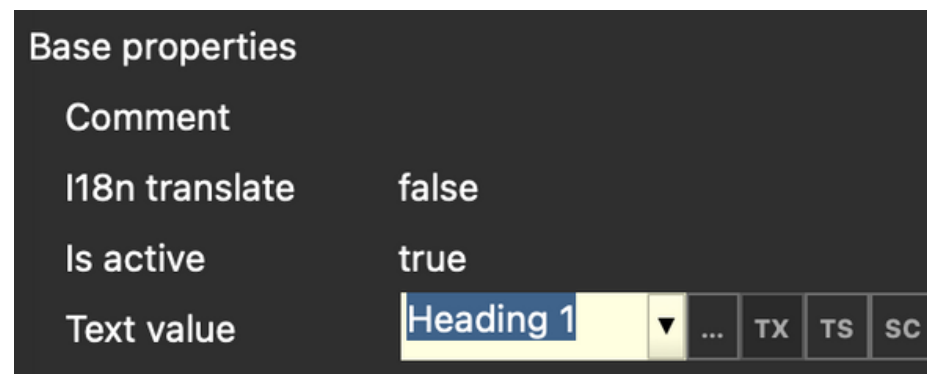


In the **Text** element of the **H2**,
change the **value** of **Text value**
to **Hello** in **Text mode**.

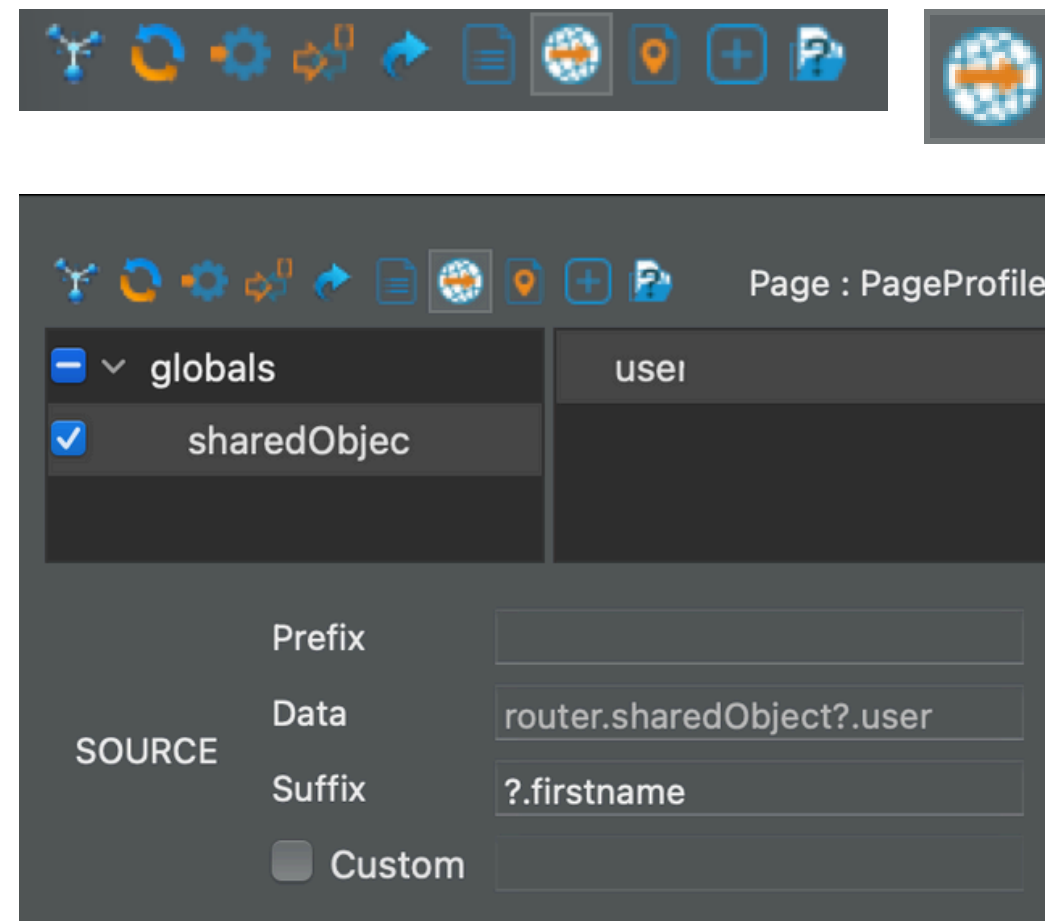


10.4 Display the Global variable in a page

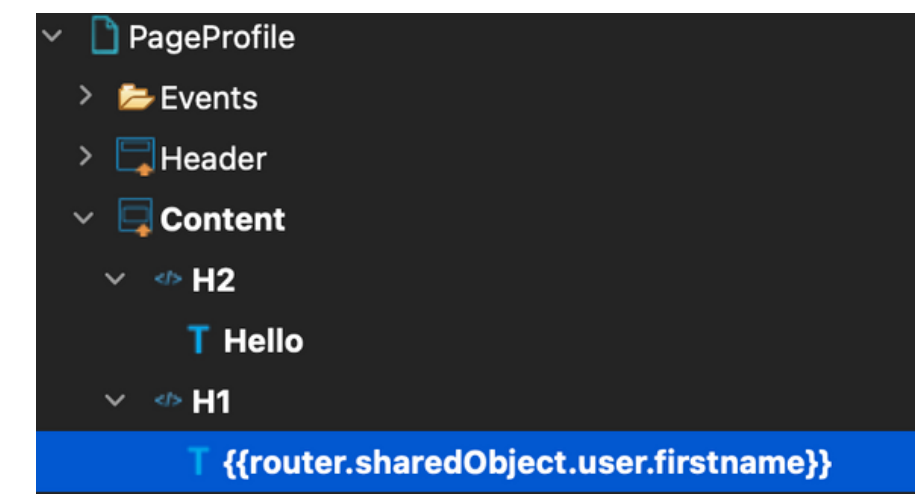
In the **Text** element of the **H1** component, to select the **value** of the **Text value** property, open the **NgxComponent Source** panel.



In the panel, click on the **SetGlobal** icon to select the global variables as source.

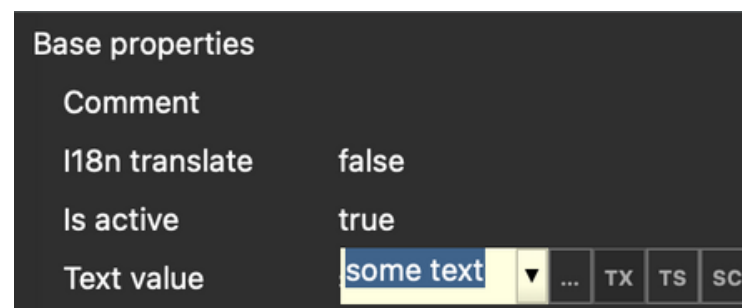
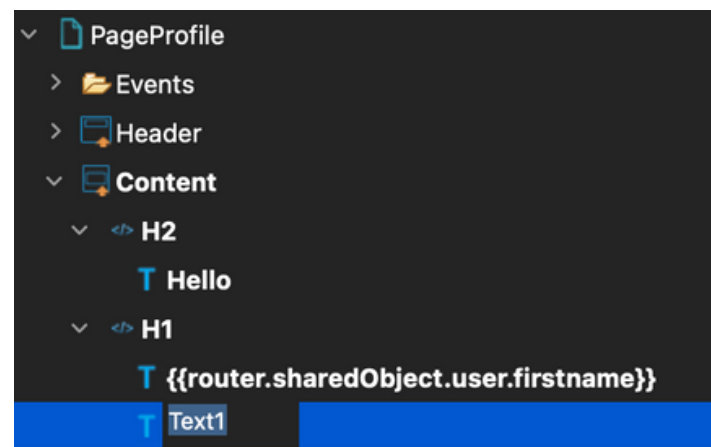


Click on **sharedObject**, then on **user**, then add **?.firstname** as **Suffix**.

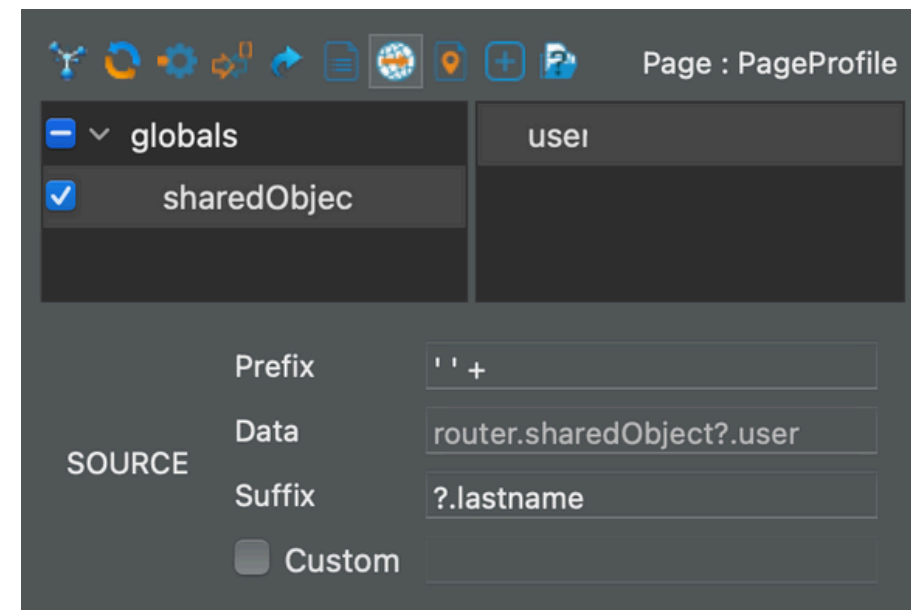


10.4 Display the Global variable in a page

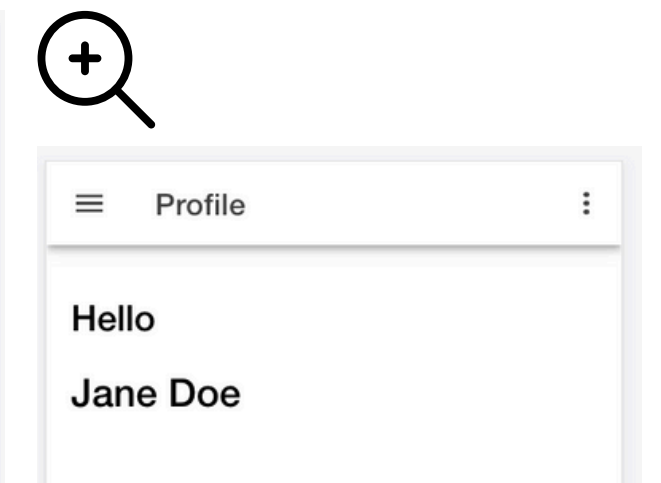
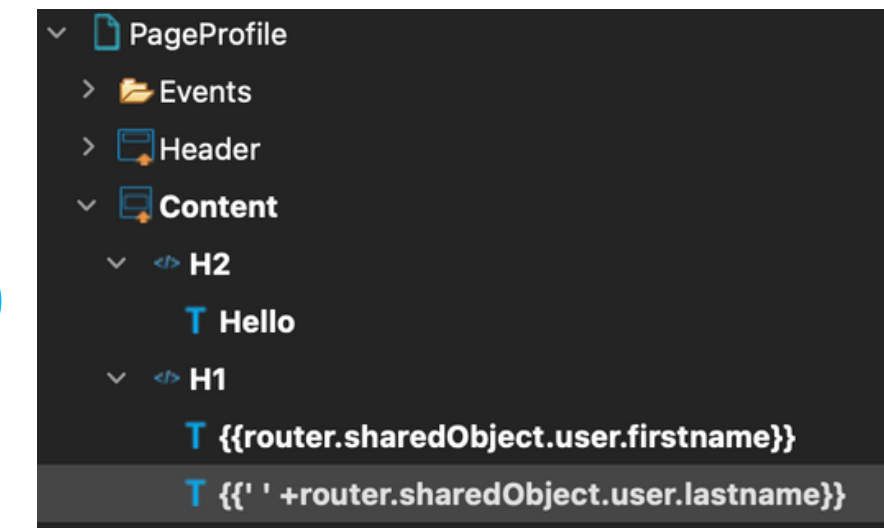
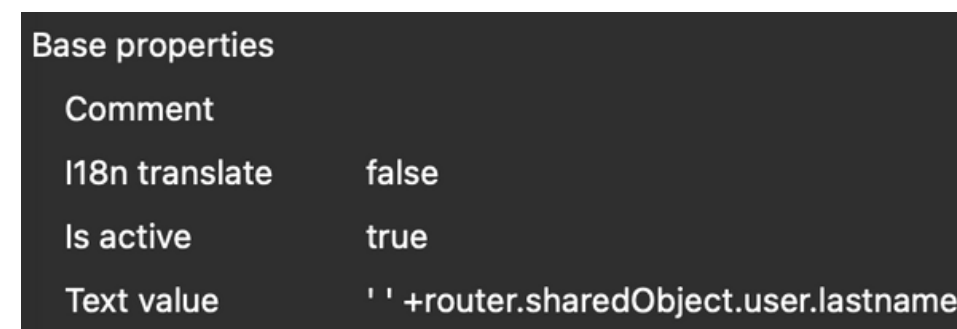
In the H1 component,
add a second Text element.



To select the **value** of its Text value property,
open the **NgxComponent Source** panel.



Click on **sharedObject**, then on **user**,
then add **?.lastname** as **Suffix**
and **'' +** as **Prefix**.



Hello, Jane Doe
appears in **PageProfile**.



Appendix



A.1 Shadow Dom & Shadow Parts

A.2 Edit AppClass & EditPageClass

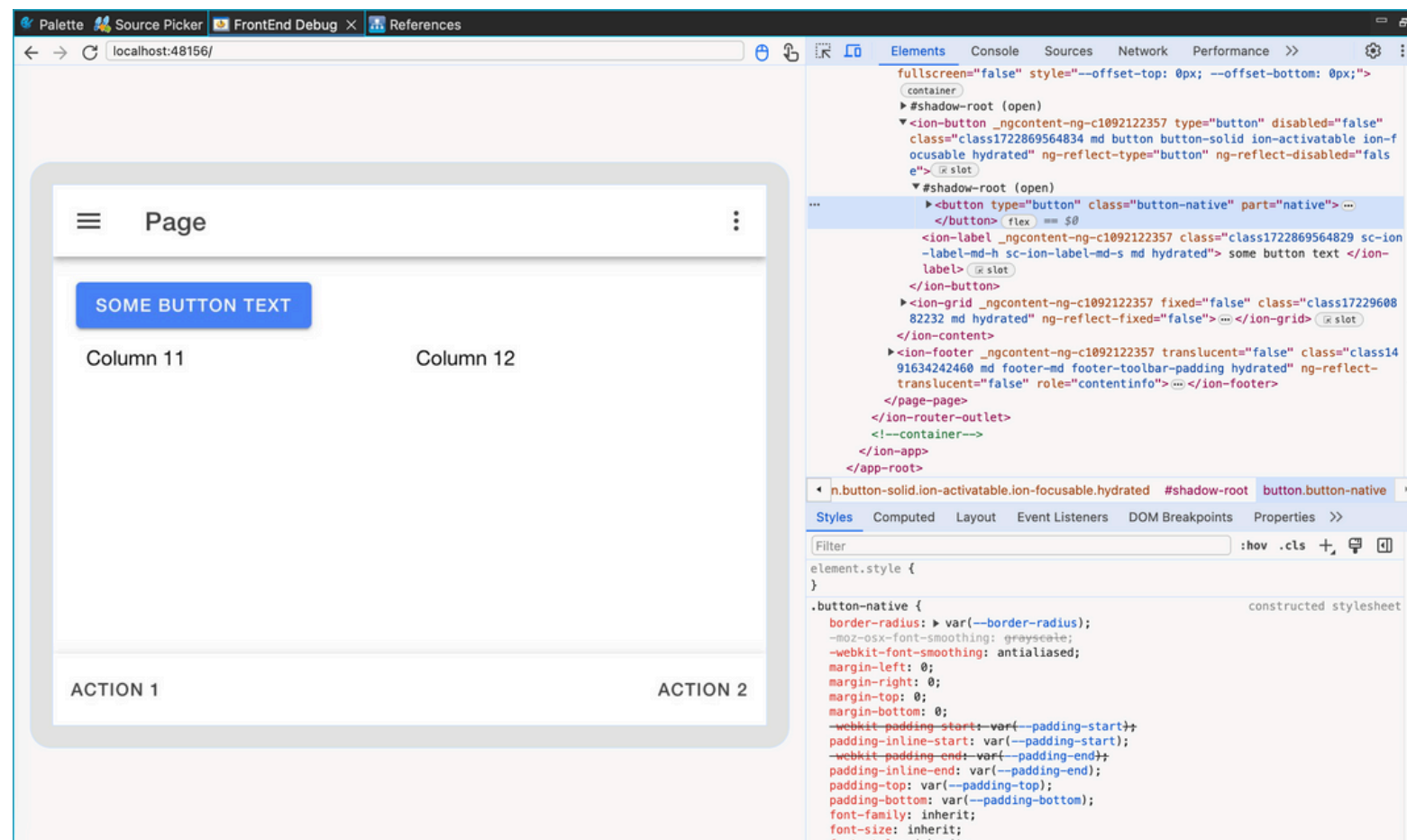
A.3 Virtual Scroll Viewport component

A.1 Shadow Dom & Shadow Parts

Ionic uses **Shadow DOM** to prevent styles from leaking out of components and unintentionally applying to other elements.

However, due to this encapsulation, **inner elements inside the shadow tree of Shadow components cannot be targeted directly with CSS either.**

Root Page in Mobile Debug view



The **Button** appears as **ion-button** with shadow dom as **#shadow-root**.

```
<ion-button _ngcontent-ng-c1092122357 type="button" disabled="false"
class="class1722869564834 md button button-solid ion-activatable ion-f
ocusable hydrated" ng-reflect-type="button" ng-reflect-disabled="fals
e">
  #shadow-root (open) == $0
  <ion-label _ngcontent-ng-c1092122357 class="class1722869564829 sc-ion
-label-md-h sc-ion-label-md-s md hydrated"> some button text </ion-
label>
</ion-button>
```

Styles won't apply on **#shadow-root** inner elements.

```
#shadow-root (open)
  <button type="button" class="button-native" part="native">
  </button>
```

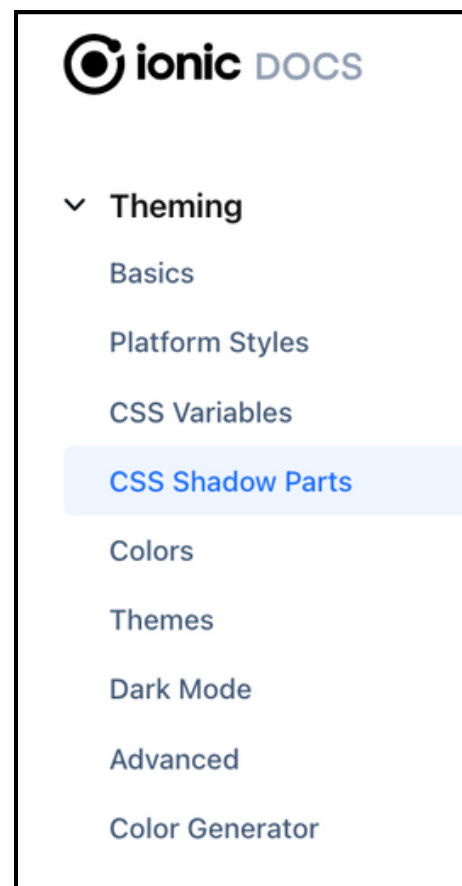


A.1 Shadow Dom & Shadow Parts

Shadow parts allow developers to style inside a **shadow tree**, from outside of that shadow tree.

In Ionic Framework, when creating a **Shadow DOM component**,
a **part** can be added to an **element inside of a shadow tree** by **assigning a part attribute on the element**.

This element is then **exposed via the part attribute**
and can be **selected and styled by using the `::part()` pseudo-element**.



You can find detailed explanations about **Shadow Parts** in **Ionic documentation** :
<https://ionic-docs-o31kiyk8l-ionic1.vercel.app/docs/v6/theming/css-shadow-parts>

CSS Shadow Parts

CSS Shadow Parts allow developers to style CSS properties on an element inside of a shadow tree. This is extremely useful in customizing Ionic Framework **Shadow DOM** components.



A badge in the top right of its documentation indicates that **ion-select** is a **Shadow DOM component**.



- icon
- placeholder
- text

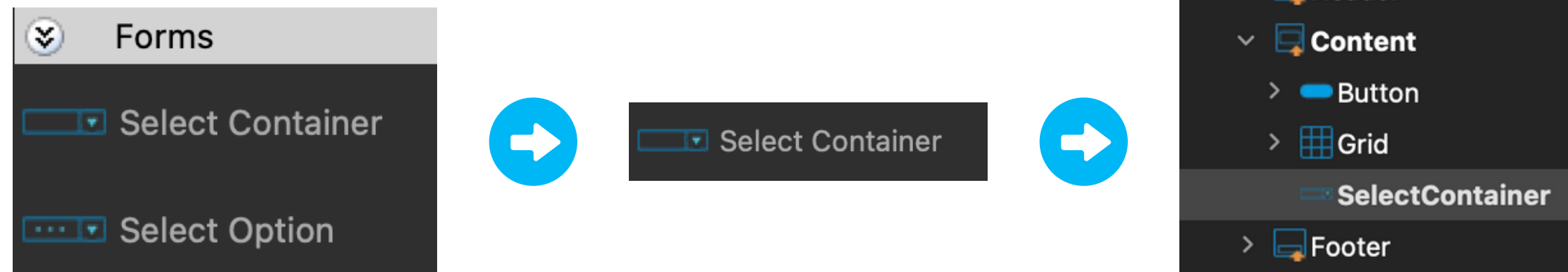
Name	Description
icon	The select icon container.
placeholder	The text displayed in the select when there is no value.
text	The displayed value of the select.



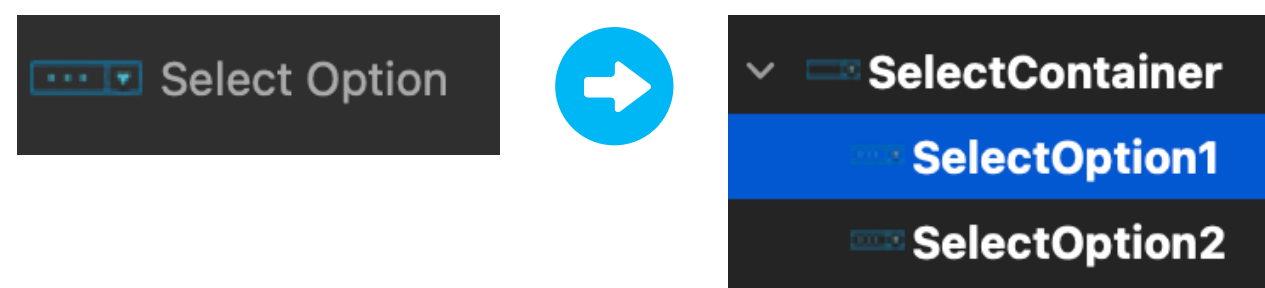
A.1 Shadow Dom & Shadow Parts

Let's add a Select component in our root page

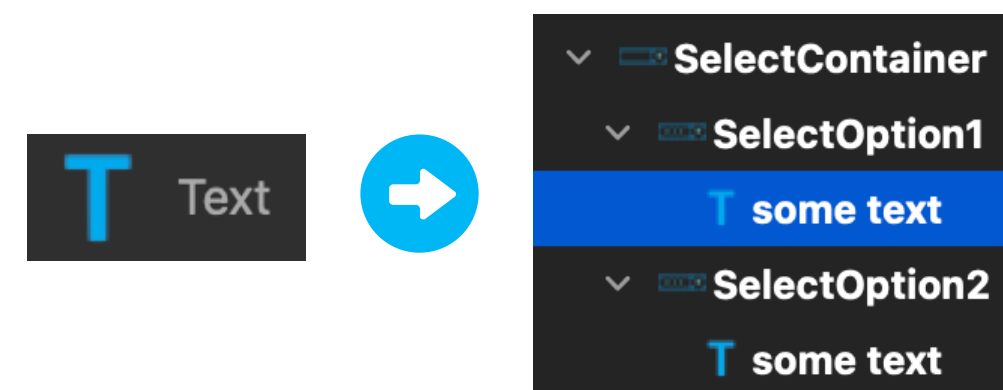
First, let's drag a **Select container** component from the **NGX Palette** and drop it in the treeview.



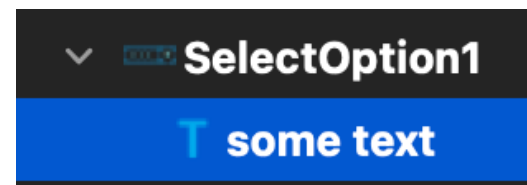
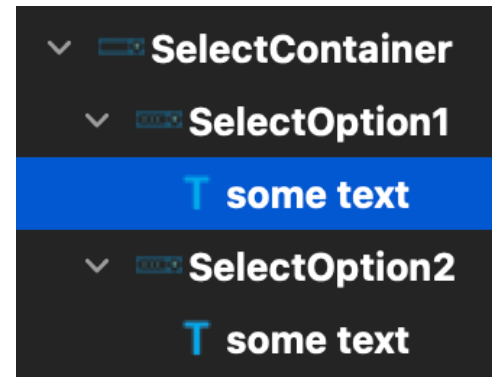
Then, let's add **Select Option** components in the **Select container** component.



And let's add **Text Components** in each **Select Option** component.



A.1 Shadow Dom & Shadow Parts

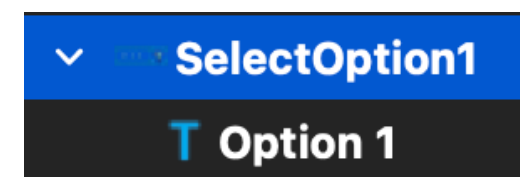


Property	Value
▼ Base properties	
Comment	
l18n translate	false
Is active	true
Text value	some text

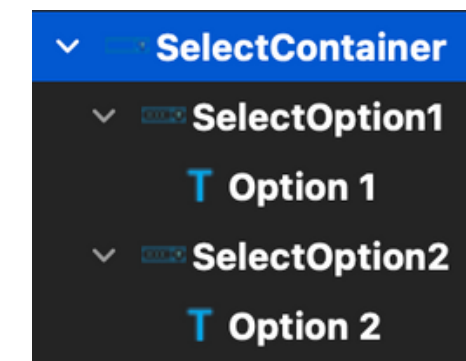


In the **properties** of the **Text** component in **SelectOption1** change the default value from **some text** to **Option 1**.

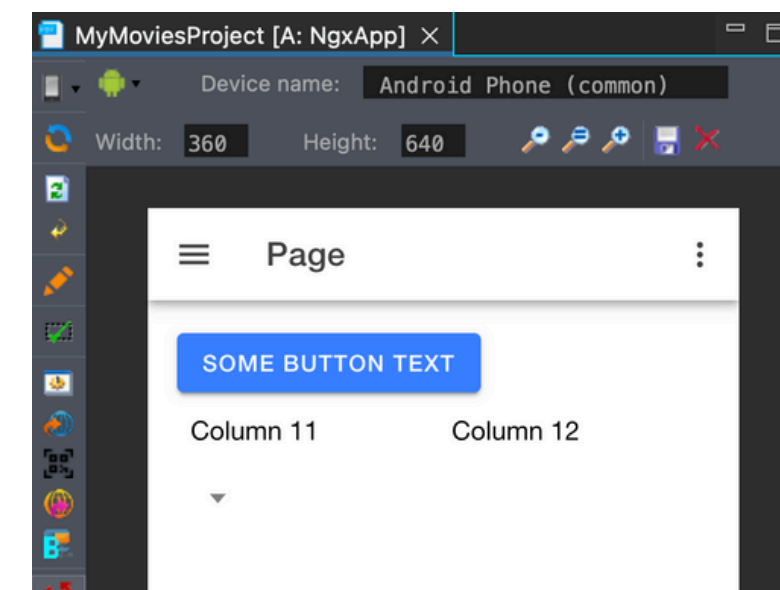
Property	Value
▼ @Component	
Identifier	
▼ @Properties	
Disabled	false
Value	Option 1



Do the same thing for the **Text** component in **SelectOption2**.



The **Select** component appears as such in the viewer



A.1 Shadow Dom & Shadow Parts

In the **Select Container** properties,
let's change the value of the **Place holder** property from **not set** to **Select an option**.

Properties

Property	Value
@Component	
Identifier	
@Control	
Binding	not set
Control default value	not set
Control name	name1723472516928
Control selected text	not set
@Properties	
Cancel Text	Cancel
Compare with	not set
Disabled	false
Expanded icon	not set
Fill	not set
Interface dialog	not set
Interface dialog options	not set
Justify	space-between
Label	not set
LabelPlacement	start
Legacy	false
Mode	not set
Multiple	false
Ok Text	Ok
Place holder	not set
Shape	not set
Toggle icon	not set

MyMoviesProject [A: NgxApp]

Device name: Android Phone (common)

Width: 360 Height: 640

Page

SOME BUTTON TEXT

Column 11 Column 12

+

Place holder not set

▼



Properties

Property	Value
@Component	
Identifier	
@Control	
Binding	not set
Control default value	not set
Control name	name1723472516928
Control selected text	not set
@Properties	
Cancel Text	Cancel
Compare with	not set
Disabled	false
Expanded icon	not set
Fill	not set
Interface dialog	not set
Interface dialog options	not set
Justify	space-between
Label	not set
LabelPlacement	start
Legacy	false
Mode	not set
Multiple	false
Ok Text	Ok
Place holder	Select an option
Shape	not set
Toggle icon	not set

MyMoviesProject [A: NgxApp]

Device name: Android Phone (common)

Width: 360 Height: 640

Page

SOME BUTTON TEXT

Column 11 Column 12

+

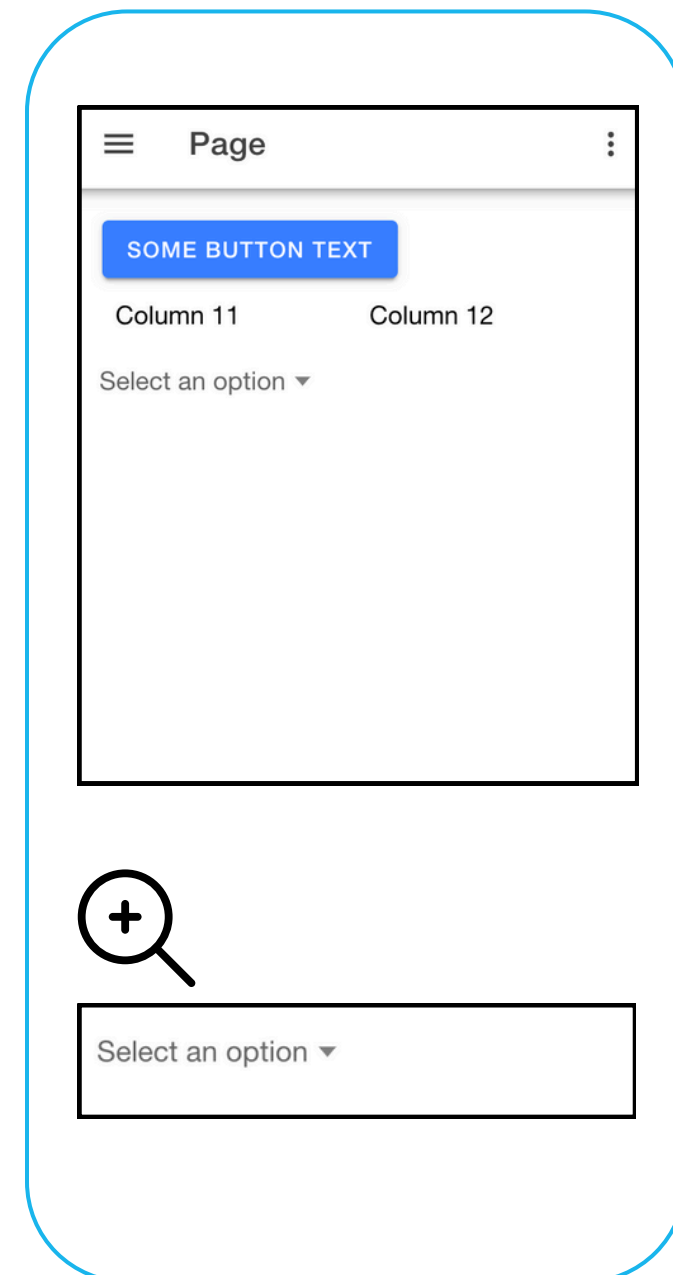
Place holder Select an option

Select an option ▼

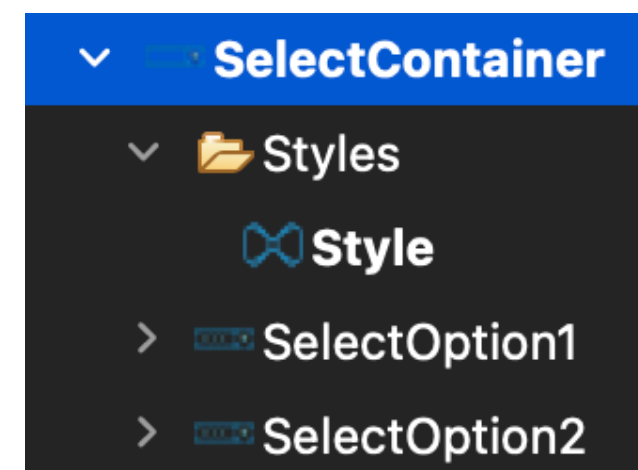
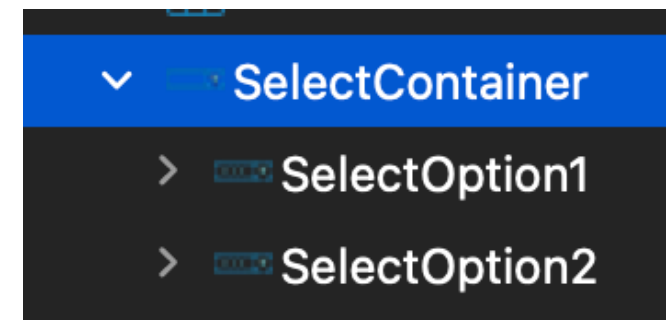


A.1 Shadow Dom & Shadow Parts

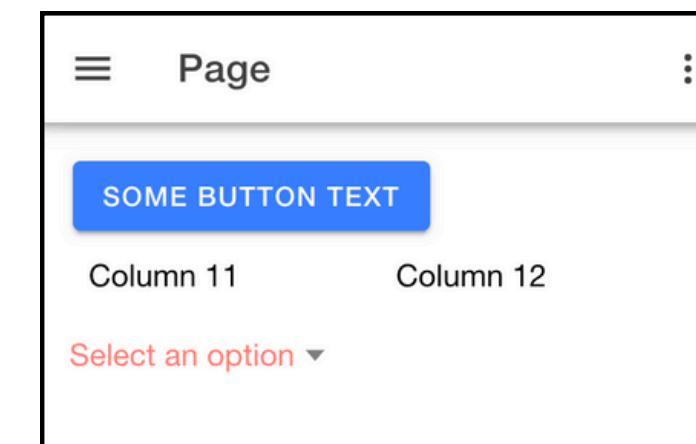
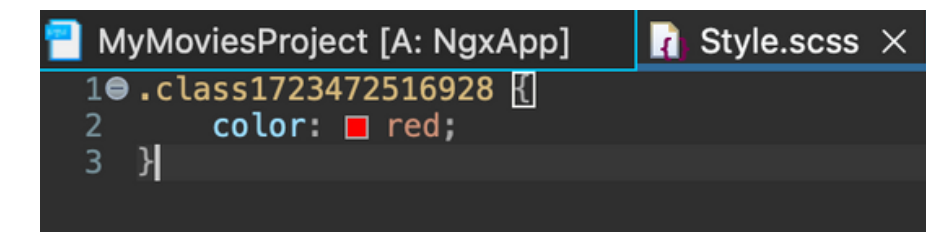
Let's change the color of the Select container component



Add a **Style** component
to the **Select container** component.



Add a **color property** with red as value
in the **Style.scss**.

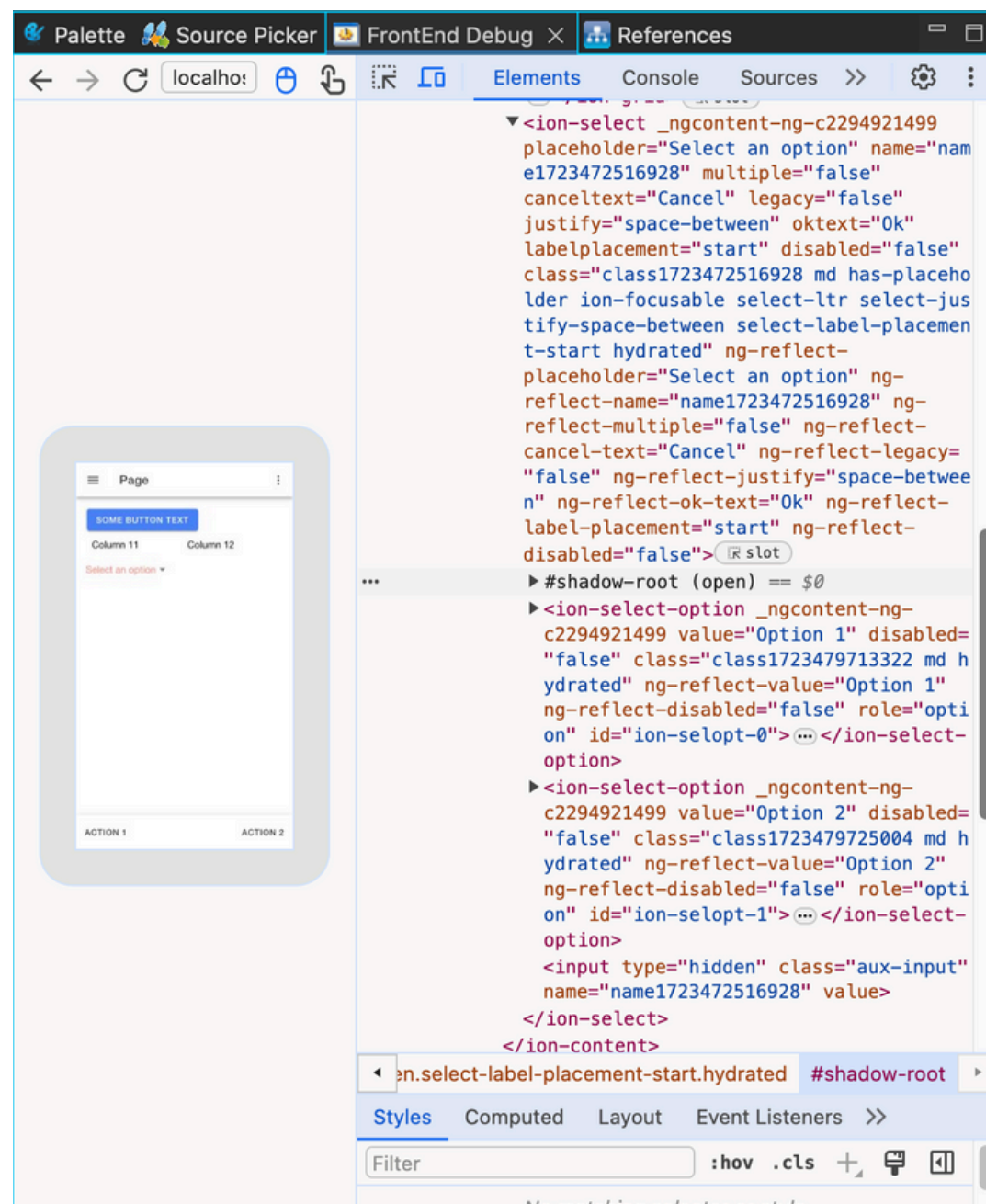


The **text color** appears as red
but not the **arrow icon**.

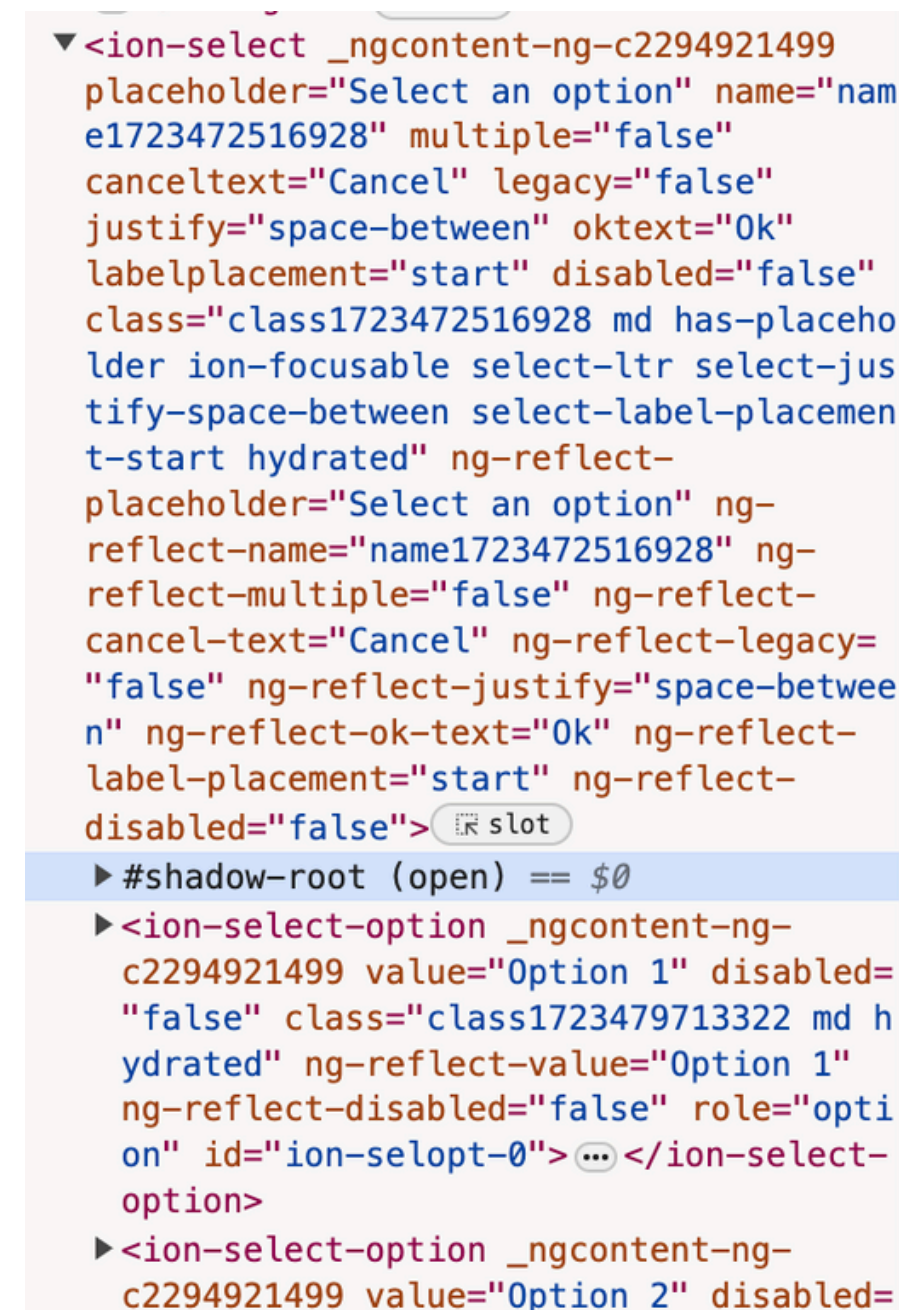


A.1 Shadow Dom & Shadow Parts

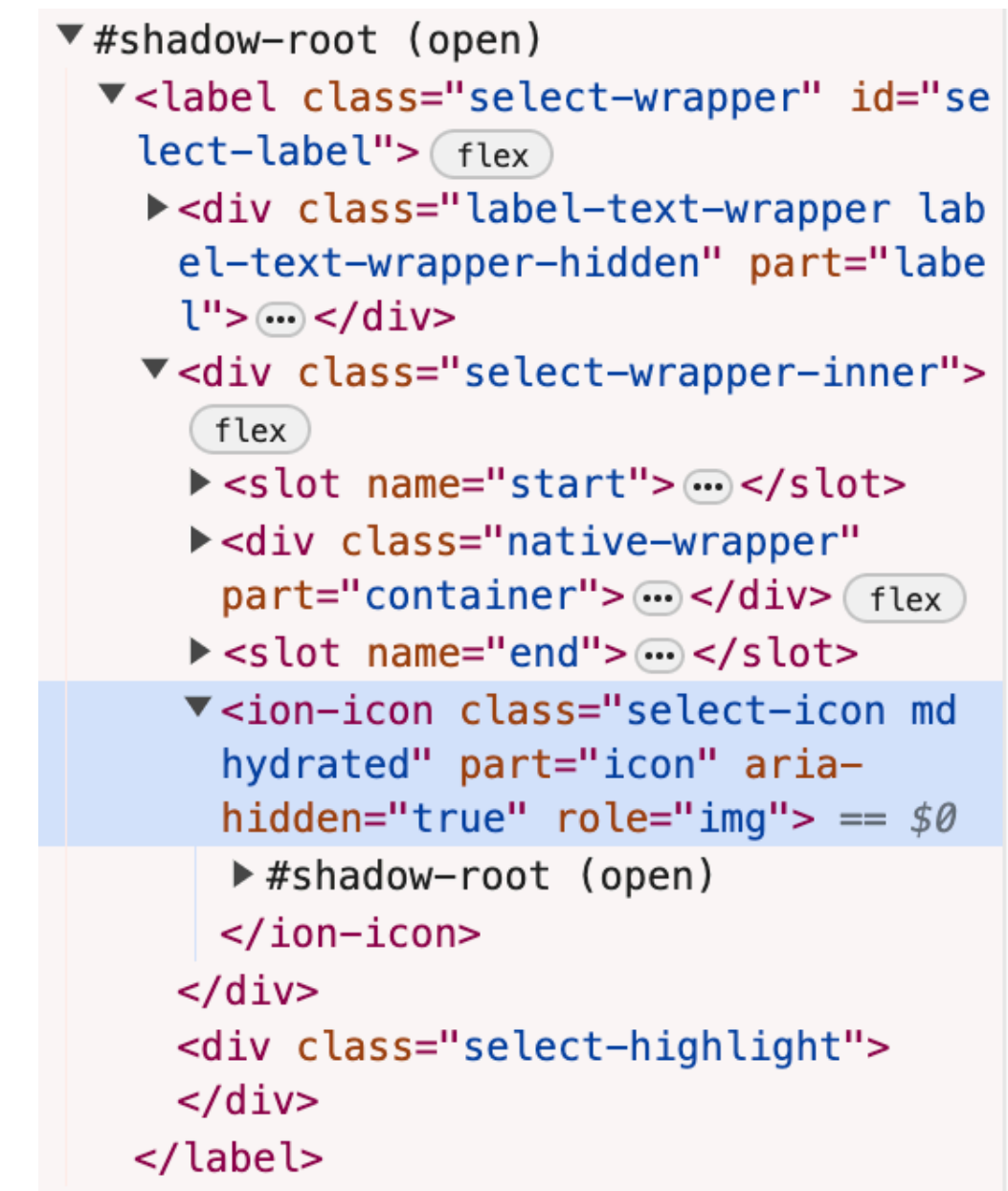
Let's have a look on the **Select** in the **FrontEnd debugger**.



🔍 We can see the **Shadow-root** of the **Select** component.

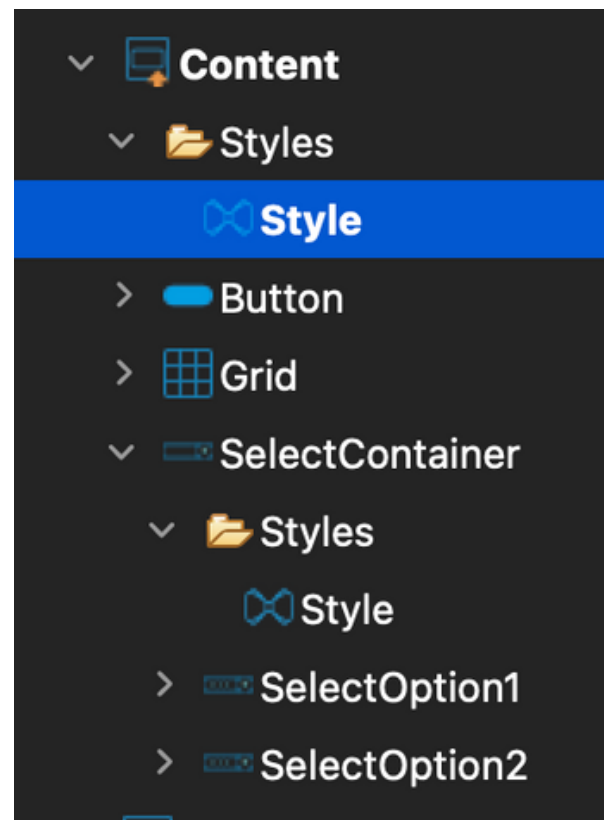


🔍 Open the **Shadow-root**.
The **icon** is inside
as **ion-icon**.



A.1 Shadow Dom & Shadow Parts

Let's add a **Style component** to the **parent component** of the **Select Container** in the treeview.

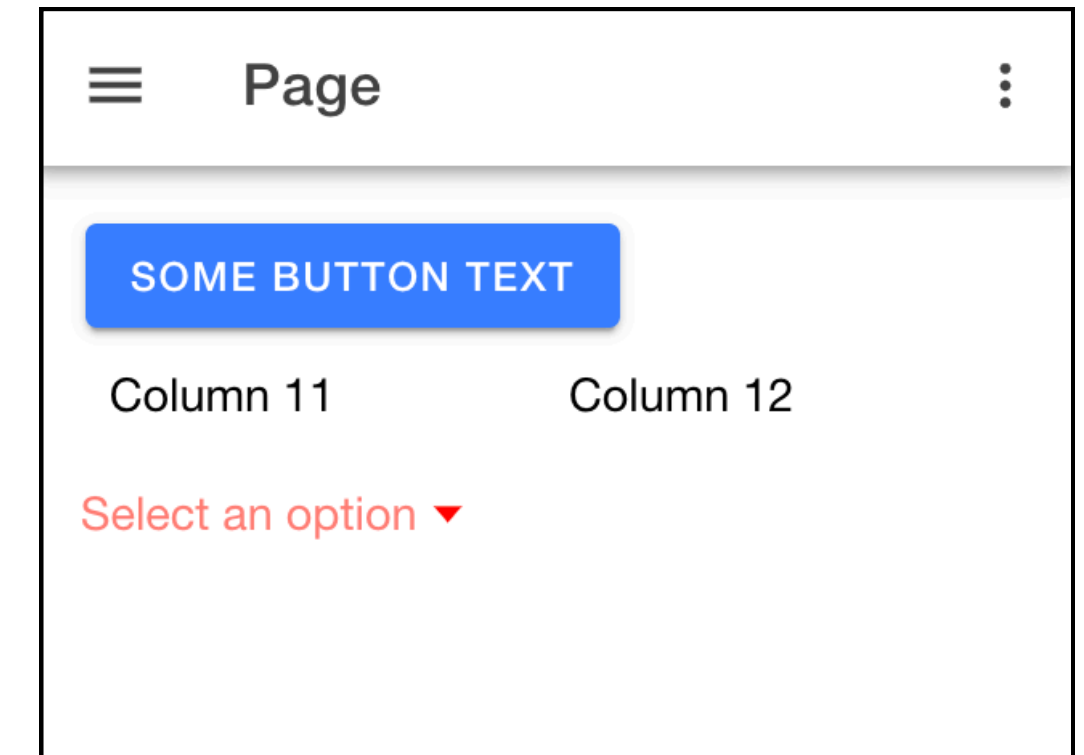


In **Style.scss**, let's add **ion-select**, the selector for **Select Container**, with the pseudo-element **::part(icon)** to select the **ion-icon** in the **shadow root**.

And add a color property with red as value.

```
MyMoviesProject [A: NgxApp] | Style.scss x
1 .class1489674805352 {
2
3   ion-select::part(icon) {
4     color: red;
5   }
6 }
```

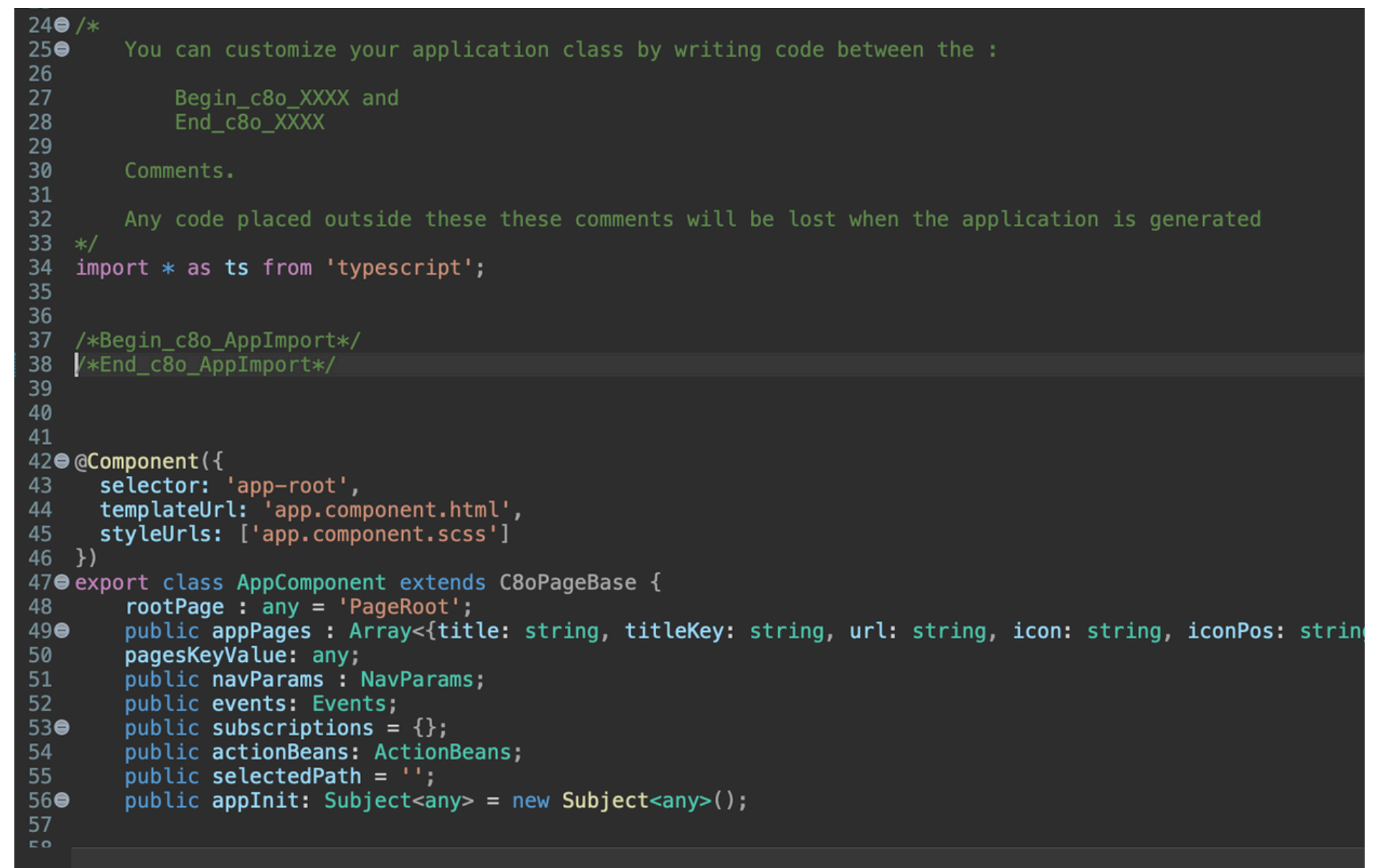
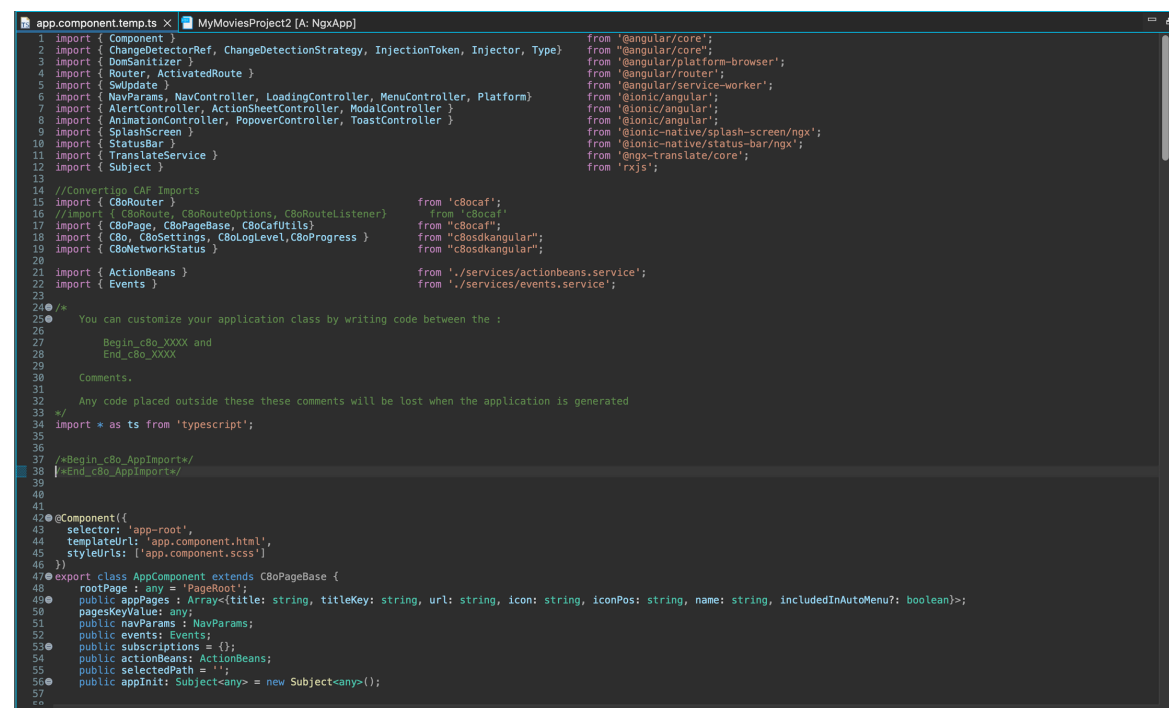
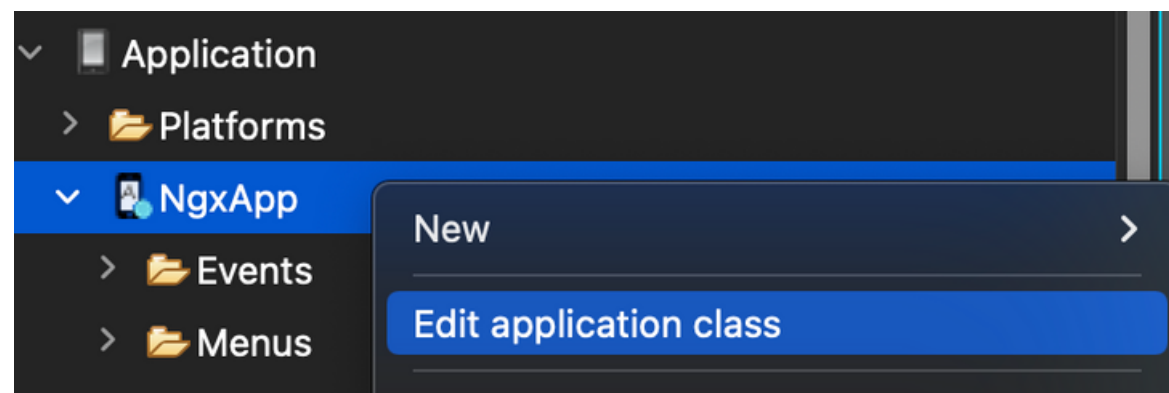
The **arrow icon** now appears as red.



A.2 Edit AppClass & EditPageClass

EditAppClass is a class used to edit the application's properties and settings.

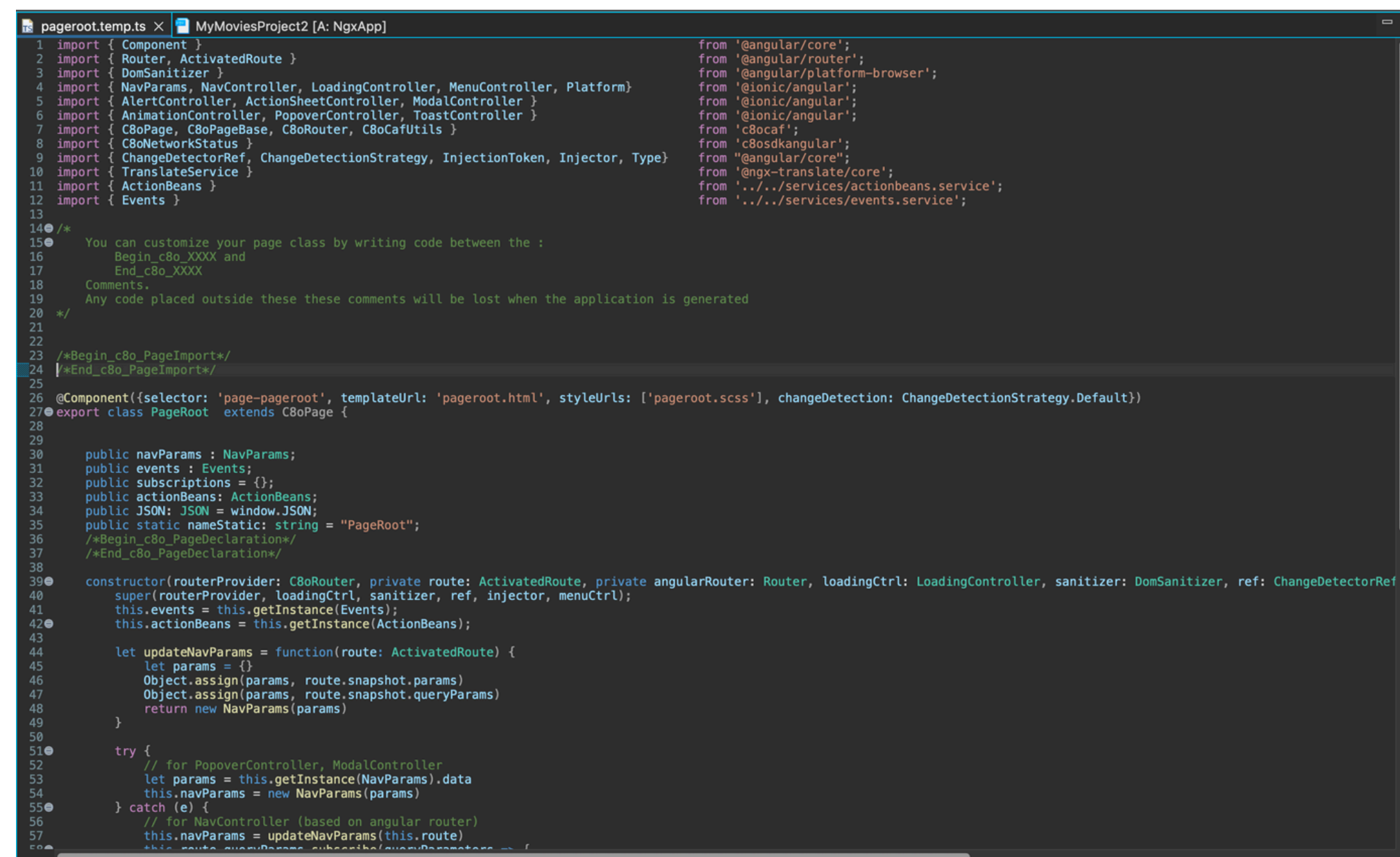
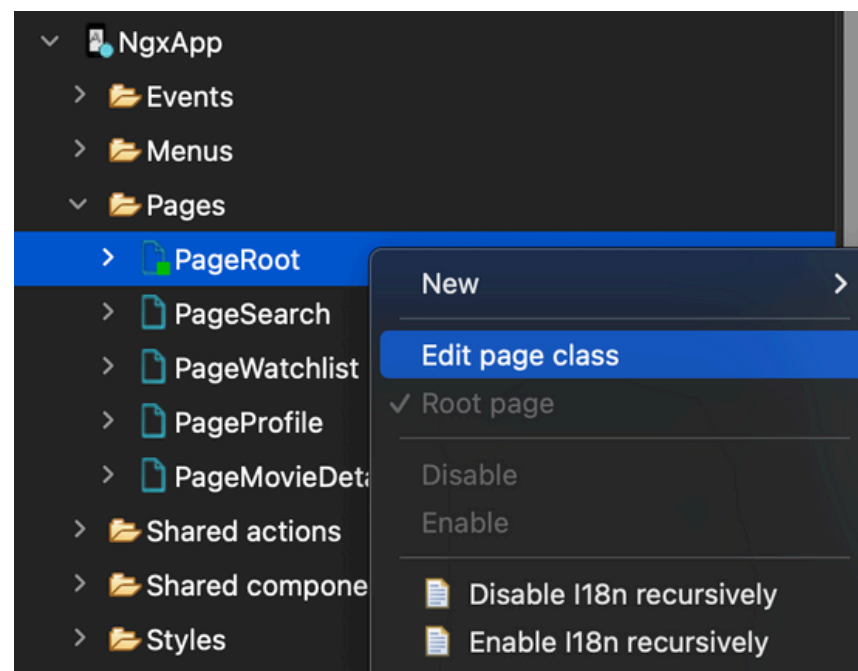
It includes methods and interfaces to manage various aspects of the application like setting application parameters, defining global variables, and configuring project-level settings.



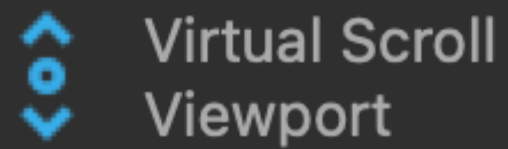
A.2 Edit AppClass & EditPageClass

EditPageClass is a class responsible for editing and managing individual pages.

It provides tools and interfaces to modify page-specific properties, manage page elements, and adjust layouts and functionalities specific to a particular page within the application.



A.3 Virtual Scroll Viewport component



The **Virtual Scroll Viewport component** is used to display large lists of elements performantly by only rendering the items that fit on-screen.

This component works with the **CdkVirtualFor Directive** to specify scrollable items to render.

<https://material.angular.io/cdk/scrolling/overview>

Loading hundreds of elements can be slow in any browser.

Virtual scrolling enables a performant way to simulate all items being rendered

by making the height of the container element the same as the height of total number of elements to be rendered, and then only rendering the items in view.

Virtual scrolling is different from strategies like infinite scroll where it renders a set amount of elements and then when you hit the end renders the rest.

