MySpecialGrid

- Custom Angular Component -

Overview

MySpecialGrid is a custom Angular component which enables displaying elements paginated using a grid. The second custom Angular component which is incorporated in the first is MySpecialCard. This card presents information about the elements being displayed in the gird. Upon clicking a certain element displayed, the user will be redirected to a predefined link.

Usage

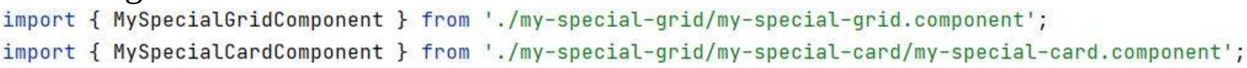
The grid component can be useful in many different scenarios. One can find it easy to integrate it into a news website, a shopping site or an online course provider.

Start

For starting the development server of the project maintained in Github you should run the “*ng serve*” command then navigate to http://localhost:4200/. If something is already running on that port you can opt to change the port when prompted. Upon changing any of the source files the application will automatically perform a reload operation.

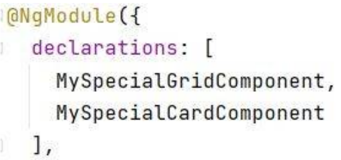
Integrate

In order to add this component into your Angular application you should copy the directory found at /src/app/my-special-grid and import in your app.module.ts the following:



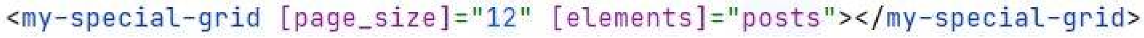
```
import { MySpecialGridComponent } from './my-special-grid/my-special-grid.component';
import { MySpecialCardComponent } from './my-special-grid/my-special-card/my-special-card.component';
```

and declare:



```
@NgModule({
  declarations: [
    MySpecialGridComponent,
    MySpecialCardComponent
  ],
```

To add it to your page use the name my-special-grid for the custom tag:



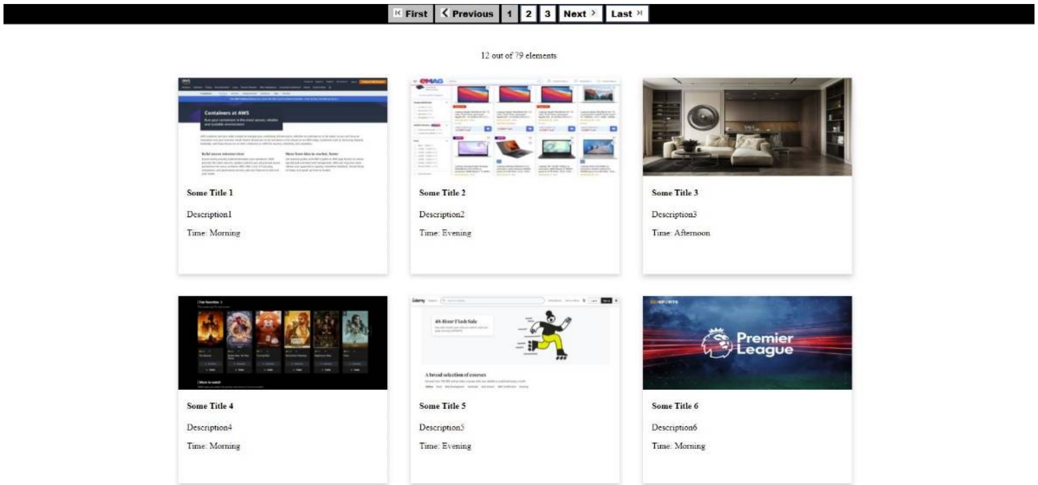
```
<my-special-grid [page_size]="12" [elements]="posts"></my-special-grid>
```

Data format

Presumably the frontend Angular application receives a list of elements to be displayed as a grid paginated. For a correct behavior the list should not be empty and the elements contained should have the following fields: “title”, “imagePath”, “description”, “time” and “onClickHref”. Also, the page size set from Angular should be higher than zero and no bigger than the number of the elements provided in the list.

Description

Visually the elements displayed on three columns but they can rearrange automatically on smaller screens on two columns. The cards will rearrange too accordingly. The card is formed a upper image part followed by a bottom info part.



12 out of 79 elements

Some Title 1

Description1

Time: Morning

Some Title 2

Description2

Time: Evening

Some Title 3

Description3

Time: Afternoon

Some Title 4

Description4

Time: Morning

Some Title 5

Description5

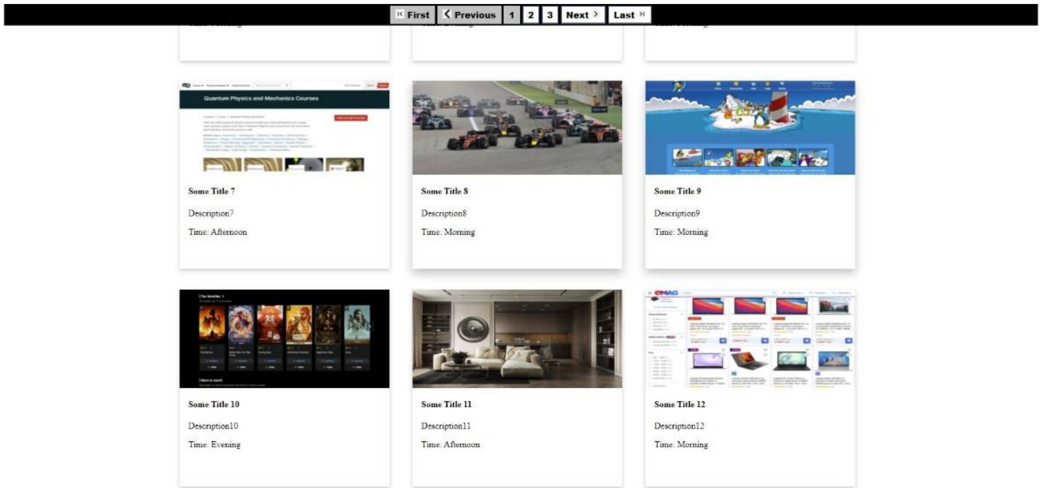
Time: Evening

Some Title 6

Description6

Time: Morning

The navigation bar is formed by the buttons necessary for pagination. Upon scroll the bar stops at the very top of the page enabling the user the possibility to use it directly whenever it is needed.



Some Title 7

Description7

Time: Afternoon

Some Title 8

Description8

Time: Morning

Some Title 9

Description9

Time: Morning

Some Title 10

Description10

Time: Evening

Some Title 11

Description11

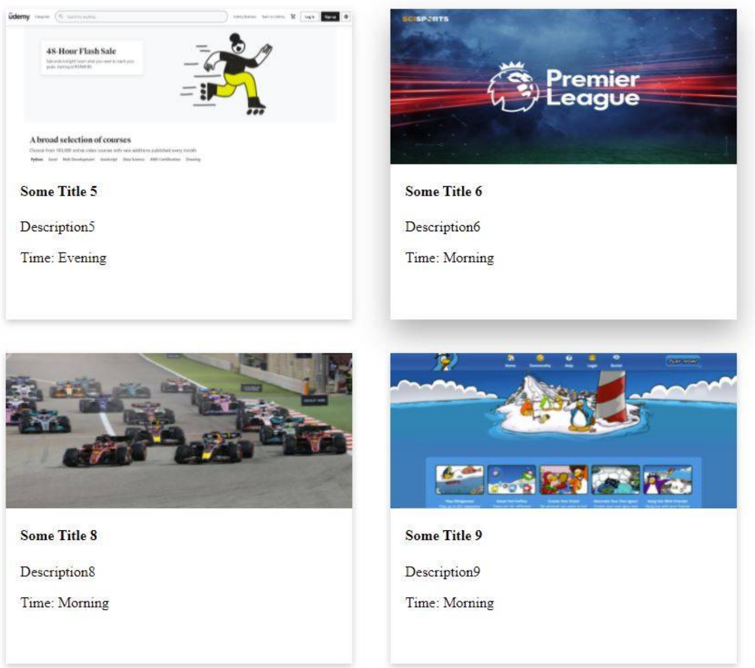
Time: Afternoon

Some Title 12

Description12

Time: Morning

When hovering over an element the user can see it receiving a shadow as the element seems to be lifted from an imaginary table. The elements can be clicked and will redirect the user to the link associated to the corresponding element.



Some Title 5

Description5

Time: Evening

Some Title 6

Description6

Time: Morning

Some Title 8

Description8

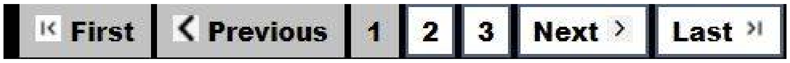
Time: Morning

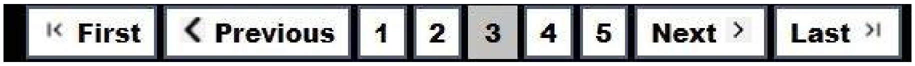
Some Title 9

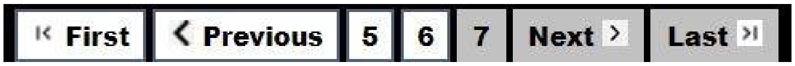
Description9

Time: Morning

A button with the current page number will become inactive when the user finds himself on that specific page. Button inactivation is triggered also at the end of the catalogue of pages as well as at the end of it. In these cases the buttons which serve no purpose for navigation will be inactive.







Code

- MySpecialGrid -

The “@Input” annotation is a decorator which is linked to the variables which will hold external data. Several variables are needed for the element to correctly display the elements on the current page and create the navigation bar buttons.

These variables are used by the functions that decide what index should have the elements displayed on the current page, the number of pages shown in the navigation bar, and if a button should become inactive or active.

The “ngOnInit()” method is called at the initialization of the component. Prior to the call of initialization, the component already received the data needed to be displayed so the initialization method could proceed further. It reaches the point of data validation. Error messages are provided if data is invalid. Then a series of computations are needed to determine the number of pages and establish the current page.

The rest of the methods are utility methods called whenever data needs to change. Upon calling these functions the view changes automatically according to the results provided. There are a series of functions that simply set the current page to the first page, previous page, next page, or last page when their corresponding buttons are pressed.

The “moveToPage(page)” method is called by pressing the page numbers on the navigation bar and moving the user on the selected page.

The functions which orchestrate the elements with which ids are to be shown and the disabling and enabling of the buttons are: “computeIdsAndButtons()”.
The “arrayOfIndexes()” and “arrayOfPages()” are all about reaching all the ids of the elements in the list that will be displayed on the current page also the numbers of the pages that would have buttons in the navigation bar in the current page.

The last function: ”isPageCurrentPage()“ decides if a page button should or should not be disabled. The page on which the user is currently is always disabled.

- MySpecialCard -

The card receives one element with the help of @Input decorator and shows an image and some details. All details should be provided through the input element. If additional information needs to be added or removed this is the place to perform such operation because this component has the role of encapsulating the details of the elements shown in the grid. This provides programmers with an easy way to reuse and change code.