## Fix all the bugs before trying to implement any other features

***ERROR 1***

zone.js:388 Unhandled Promise rejection:

Component AppComponent is not part of any NgModule or the module has not been imported into your module.

SUGGESTION:

- Declare ***AppComponent*** in @NgModule --> see: **app.module.ts**

***ERROR 2***

EXCEPTION: Error in ./AppComponent class AppComponent\_Host - inline template:0:0 caused by: The selector "bookstore" did not match any elements

SUGGESTION:

The selector "bookstore" did not match any elements in the parent-template.

De parent-template of the AppComponent is**: index.html**

Change the tag: <**your-app*>*** in the index.html to 🡪 <bookstore>

***ERROR 3***

EXCEPTION: Error in ../app/app.template.html:25:12 caused by: No provider for TemplateRef!

SOLUTION:

- Change **app.template.html** --> **ngIf** should be **\*ngFor**

***ERROR 4***

EXCEPTION: Error in ../app/app.template.html:16:46 caused by: Cannot read property 'name' of undefined

SUGGESTION:

Define **categories** in app.component.ts

HOW?

Use mocks data in**app.component.ts**.

Add the following In **app.component.ts**:

**import** myCategories, {Category} **from './mocks/categories'**;

…..

Assign myCategories to the array Category[]

**Finally: ngFor** should be **\*ngFor in app.template.html**

***ERROR 5***

Unhandled Promise rejection: Template parse errors:

Can't bind to 'ngForIn' since it isn't a known property of

'li'. (" <a data-type="all" href="#0">All</a>

</li>

<li [ERROR ->]\*ngFor="let category in categories">

<a class="selected" href="#0">{{category.name}}<"): AppComponent@15:16

SUGGESTION:

Use:  **\*ngFor="let category of categories"**

Instead of:  **\*ngFor="let category in categories"**

## TODO: Features to implement in your app

**Display all books on the main area**

1. Add this *import* in **app.components.ts:**

import { mockBooks, Book } from './mocks/books';

2. Assign the mockBooks in the **app.components.ts:**

books: Book[] = mockBooks;

3. SOLVE EXCEPTION

book.cover: GET http://localhost:3000/book.cover 404 (Not Found)

SUGGESTION:

Go to the **app.template.html:**

<li \*ngFor="let book of books"><img src="book.cover" click="clicked"/></li>

Use a property bindings here!

**On click on a category, update the list of books**

SUGGESTION:

A filter should be applied to the selected category of the book!

Change the following in **app.template.html:**

|  |
| --- |
| <li \*ngFor="let category of categories">  <a class="selected" href="#0" **(**click**)**="clicked(category)">{{category.name}}</a>  </li> |

Add the following in your **app.components.ts:**

|  |
| --- |
| clicked(cat : Category) {  **if** (cat.**name** == **'All'**) {  **this**.**books** = ***mockBooks***;  } **else** {  **this**.**books** = ***mockBooks***.filter(b => b.**category** === cat.**name**);  } } |

**On click on a category, update the list of books and change the class to selected**

SUGGESTION

The css class should be applied to the anchor: use **[class.selected] or [ngClass]**

Change the following in **app.template.html:**

|  |
| --- |
| <li \*ngFor="let category of categories">  <a **[ngClass]="selCat==category.name ? 'selected' : ''"** href="#0" (click)="clicked(category)">{{category.name}}</a>  </li> |

Change the following in your **app.components.ts:**

|  |
| --- |
| clicked(cat : Category) {  **if** (cat.**name** == **'All'**) {  **this.selCat = cat.name;**  **this**.**books** = ***mockBooks***;  } **else** {  **this**.**books** = ***mockBooks***.filter(b => b.**category** === cat.**name**);  **this**.**selCat** = cat.**name**;  } } |

**On click on the filter button, open the sidebar filter component**

SUGGESTION:

Change the following in **app.template.html** in the section: *Sidebar navigation*

<div **[ngClass]**="!navClosed ? 'filter filter-is-visible': 'filter'">

Define an onclick in your **app.components.ts**:

**(click)**="toggleSideBar**()**"

**Learn about Forms and NgModel**

**Implement the search**

Add the following in **app.module.ts:**

|  |
| --- |
| **import { FormsModule } from '@angular/forms';**  @NgModule({  imports: [ BrowserModule, **FormsModule** ], // Inject built-in modules  declarations: [ AppComponent ], // Inject your own modules  bootstrap: [ AppComponent ] // Module you need to bootstrap  }) |

Change the following in **app.template.html** in the section: *Sidebar navigation*

|  |
| --- |
| <form (submit)="search()">  <input class="form-control" type="text" id="search"  required  **[(ngModel)]**="searchString" name="searchString" placeholder="Title or category">  </form> |

Implement this method in **app.components.ts**:

|  |
| --- |
| search(){  ***console***.log(**this**.**searchString**);  **this**.**books** = ***mockBooks***.filter(b => b.**category**.toLowerCase().includes(**this**.**searchString**.toLowerCase()) || b.**title**.toLowerCase().includes(**this**.**searchString**.toLowerCase()));  } |

**Define a service with promises (for retrieving the mockBooks and categories):**

Create a new folder: **app/services**

Add a service in this folder: **app.service.ts**

|  |
| --- |
| **import** { Injectable } **from '@angular/core'**; **import** { Book, mockBooks } **from '../mocks/books'**; **import** { categories, Category } **from '../mocks/categories'**;  **@Injectable() export class** AppService {  getBooks(): Promise<Book[]> {  **return *Promise***.resolve(***mockBooks***);  }   getCategories(): Promise<Category[]> {  **return *Promise***.resolve(***categories***);  } } |

Import this new service in **app.component.ts:**

|  |
| --- |
| **import** { AppService } **from './services/app.service'**; |

Define the service as a provider in app.component.ts

To make this AppService injectable, we need to create a provider for it first.

|  |
| --- |
| @Component({  **selector**: **'bookstore'**,  *// template: ``,* **templateUrl**: **'../app/app.template.html'**,  **providers: [ AppService ]** }) |

Redefine books and categories in app.component.ts:

|  |
| --- |
| **export class** AppComponent {  **books**: Book [] ;  **categories**: Category[]; |

Inject an instance of the Service via the constructor of the AppComponent:

|  |
| --- |
| **constructor**(**private** appService: AppService){  *//empty constructor* } |

Get the mockdata via the lifecycle hook **ngOnInit()** :

Make use of the AppService:

|  |
| --- |
| **public** ngOnInit(){  **this**.appService.getBooks().then((books) => {  **this**.**books** = books;   });   **this**.appService.getCategories().then(categories => {  **this**.**categories** = categories;   }); } |

Change this in categories.ts:

|  |
| --- |
| **export const** categories |

**Architecture: Split your app into small components**

**Which components do you recognize?**

* HeaderComponent
* BookListComponent
* MenuComponent
* SideBarComponent

Steps to create a BookListComponent:

* Create a new folder: app/components/book
* Add new template for the booklist in this folder:

booklist.template.html

* + Copy the html for showing a booklist to: Booklist.template.html
  + Create a new BookListComponent in this folder

booklist.component.ts

* + Replace the html in In app.template.html

**<book-list [books]=“books”></book-list>**

* + Import(!) **and** Declare(!) the BookListComponent in app.module.ts

The BookListComponent:

|  |
| --- |
| **import {Component, Input} from *'angular/core'*;**  ***@Component*({**  **moduleId: module.id,**  **selector: ‘book-list'**  **templateUrl: ‘booklist.template.html'**  **})**  **export class BookListComponent {**  **@Input books;**  **}** |

**Steps to create a *MenuComponent*:**

* Create a new folder: app/components/menu
* Add new template for the menu in this folder:

menu.template.html

* + Copy the html for showing the menu to: menu.template.html
  + Create a new MenuComponent in this folder

menu.component.ts

* + Replace the html in In app.template.html
* <**menu [categories]="categories" (categoryChanged)="clicked($event)"**></**menu**>
  + Import(!) **and** Declare(!) the MenuComponent in app.module.ts

The MenuComponent:

|  |
| --- |
| **import** { Component, Output, Input, EventEmitter } **from '@angular/core'**; @Component({  **moduleId**: ***module***.**id**,  **selector**: **'bs-menu'**,  **templateUrl**: **'menu.template.html'** })  **export class** MenuComponent {   @Input() **categories**;  @Output() **categoryChanged** = **new** EventEmitter();   changeCategory(category) : **void** {  **this**.**categoryChanged**.emit(category);  category.**selected** = **true**;  } } |

**Menu.template.html**

|  |
| --- |
| <**div class="tab-filter-wrapper"**>  <**div class="tab-filter"**>  <**ul**>  <**li class="placeholder"**>  <**a data-type="all" href="#0"**>All</**a**>  </**li**>  <**li \*ngFor="let category of categories"** >  <**a [class.selected] ="category.selected" href="#0" (click)="changeCategory(category)"**>{{**category**.**name**}}</**a**>  </**li**>  </**ul**>  </**div**> </**div**> |

**Replace the html in the app.template.html**

<**menu [categories]="categories" (categoryChanged)="clicked($event)"**></**menu**>

Problem:

* The css class should be applied to the anchor

SUGGESTION:

In the app.component.ts ‘loop’ through the categories and set the category.selected on true or false.

|  |
| --- |
| changeCategory(selectedCategory: Category): **void** {  **this**.**categories** = **this**.**categories**.map(category => {  **if**(category === selectedCategory)  category.selected = **true**;  **else** category.selected = **false**;  **return** category;  });   **this**.filterBooks(selectedCategory); }  filterBooks(category: Category): **void** {  **if**(category.**name** === **"All"**) {  **this**.**books** = ***mockBooks***;  **return**;  }  **this**.**books** = ***mockBooks***.filter(book => book.**category** === category.**name**); } |