Angular Components Lab - Articles

1. Create Angular App

Generate a new angular app using the command "**ng new articles-app**". After that change the title of your site to "**Article Site**" in "**app.component.ts**", replace the html in "**app.component.html**" with the following:

```
<div style="text-align:center">
  <h1>
    Welcome to {{title}}!
  </h1>
</div>
```

And change the header css in "**app.component.css**":

```
@import
url('https://fonts.googleapis.com/css?family=Raleway');
h1 {
  margin: -10px -8px;
  padding: 15px 10px 15px 10px;
  background: #616192;
  color: white;
  font-family: 'Raleway',
  sans-serif;
}
```

Start the app with the command "**ng serve –open**". This will build the angular application and open a new window in your default browser. Your app should look like this:

2. Create Article Model

We need an article model **class** to hold the information. Each article has **title** (string), **description** (string), **author** (string), **imageUrl** (string). Create a folder **models** and in **src/app** and inside create a file "**article.model.ts**". We also have to **export** the class in order to use it in whichever file we want.

```
export class Article {
  constructor(
    public title: string,
    public description: string,
    public author: string,
    public imageUrl: string) { }
}
```

3. Create Article Data

We have to create a seed file to hold our **dummy data** that we will display later. Create a folder called **data** in **src/app** and in it generate a new file called **seed.ts** which will hold an array of objects. After that **export** the array. You can change the information inside the objects however you like just don"t change the property names.

```
const data = [{
  title: 'Article 1',
  description: 'LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1v',
  author: 'Author 1',
  imageUrl: 'http://ichef.bbci.co.uk/news/976/cpsprodpb/10434/production/_90121666_agreementcartoon.jpg'
},
{
  title: 'Article 2',
  description: 'LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1v',
  author: 'Author 2',
  imageUrl: 'http://www.digitalmeetsculture.net/wp-content/uploads/2015/04/article.jpg'
},
{
```

```
  title: 'Article 3',
  description: 'LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc
1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc
1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc
1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc
1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc
1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1v',
  author: 'Author 3',
  imageUrl: 'http://i.dailymail.co.uk/i/pix/2010/04/17/article-1266852-092DE58A000005DC-
112_634x411.jpg'
},
{
  title: 'Article 4',
  description: 'LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc
1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc
1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc
1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc
1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc
1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1v',
  author: 'Author 4',
  imageUrl: 'https://www.seoclerk.com/pics/518476-1iDFBR1489198900.png'
},
{
  title: 'Article 5',
  description: 'LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc
1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc
1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc
1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc
1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc
1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1LongDesc 1v',
  author: 'Author 5',
  imageUrl: 'http://www.stat.columbia.edu/~gelman/teaching/literacy/images/IV.gif'
}]

export { data };
```

We also need a **fake service** class. Inside we will map each object to an article **instance** and return a new array of articles. Create a new file in data called **data.ts**, import the **article model** and the data from "**seed.ts**" and write the following class:

```
export class ArticleData {
  getData() : Article[] {
    let articles : Article[] = [];

    for (let i = 0; i < data.length; i++) {
      articles[i] = new Article(data[i].title, data[i].description,
        data[i].author, data[i].imageUrl);
    }

    return articles;
  }
}
```

The data has a **getData()** function which iterates through all data objects and creates **instances** of articles which are saved in an array and returned from the function. Later on we will learn more about real **services** and **injectables** and how to implement this the **correct** way.

4. Create Article Component

Generate an article component using the command "**ng generate component article**". You can also create it **yourself without** using the **CLI,** but note that you have the create a **typescript file** to hold the component logic, an **html file** to hold the markup and a **css file** that is **strictly** for this component. Also, if you create the component yourself, you have to remember to import it in the **declarations** array inside the main **app module**. The CLI does all that work for us.

Copy the following css inside "**article.component.css**":

```
@import url('https://fonts.googleapis.com/css?family=Raleway');
* {
  font-family: 'Raleway',
  sans-serif;
  font-weight: bold;
}
li {
  list-style-type: none;
  width: 240px;
  text-align: center;
  border-radius: 5px;
  padding-top: 20px;
  padding-bottom: 20px;
  transition: all 0.2s linear;
  text-align: center;
  margin-bottom: 20px;
  margin-right: 15px;
  background: #ADADE8;
}
li:hover {
```

```
  transform: scale(1.04);
  box-shadow:  -1px 2px 27px 3px rgba(0,0,0,0.09)
}
img {
  width: 150px;
  height: 150px;
  margin-top: 20px;
  border-radius: 4px;
  border: 1px solid rgb(248, 248, 248);
}
#title {
  margin-bottom: 10px;
  color: white;
}
.desc {
  padding: 10px;
}
button {
  outline: none;
  border: none;
  background: white;
  color:rgb(7, 7, 7);
  padding: 6px;
  border-radius: 3px;
  margin: 5px;
}
button:hover {
  background: #616192;
  color: white;
}
```

We have to establish a **relationship** between the **article** component and the future **articles** component. The one we are implementing right now is the **child component**. It will receive **two properties** that need to be **transferred** from the parent to the child component.

Problem 4.1 Create the Html Markup

Copy the following html inside "**article.component.html**":

```
<li>
    <div id="title">{{article.title}}</div>
```

```
    <button *ngIf="showReadMoreBtn" (click)="readMore()">Read More</button>
    <button *ngIf="showHideBtn" (click)="hideDesc()">Hide Desc</button>
    <button (click)="toggleImage()">{{imageButtonTitle}}</button>
    <div class="desc" *ngIf="articleDescLen > 0">{{descToShow}}</div>
    <img *ngIf="imageIsShown" src="{{article.imageUrl}}" alt="article image" />
</li>
```

Inside the html markup we will show a detailed description of each article (**title**, **description** and **image**). We have a total of **3 buttons,** and each has an **onclick** handler function. Two of these buttons will be **toggled**. Each time we click on "**Read More**" 250 more symbols of the article description will be shown **until** there is no more. The moment our description ends we have to show "**Hide Desc**". Clicking "**Hide Desc**" should **remove** all the description inside and **reset** the counter. Both buttons have an attached function to them – **readMore()** and **hideDesc()**. We also have to show/hide the image, this is why we attach a **toggleImage()** function that we will implement later. All the properties inside the ***ngIf** directive are controlled in our article component.

Problem 4.2 Create the Needed Properties

Inside "**article.component.ts**" we have to declare the following properties which we showcased in the previous section:

```
export class ArticleComponent {
  private symbols: number = 250;
  @Input() article: Article;
  @Input() articleDesc: string;
  descToShow: string;
  articleDescLen: number;
  showReadMoreBtn : boolean = true;
  showHideBtn : boolean = false;
  imageIsShown: boolean = false;
  imageButtonTitle: string = "Show Image";
```

Both **article** and **aricleDesc** will come from **above** in the **articles component** which we will implement later. This is why we use the **@Input** decorator that we need to import. We need a property to show the current description which can vary from **0 symbols** until the **end**. The article description length is a **counter** which controls how many symbols to show. The **3 boolean** properties are used inside the ***ngIf** directive to determine whether or not to **show** or **hide** a button. Also, the content of the image button is switched between "**Show Image**" and "**Hide Image**".

Inside the constructor of the component define the article description length to **zero** and the description to show to an **empty string** (we can also do this in the **ngOnInit** lifecycle hook):

```
  constructor() {
    this.articleDescLen = 0;
    this.descToShow = "";
  }
```

Problem 4.3 Implement the Read More Function

Each time we click on **readMore()** we have to **increase** the article description length with **250 symbols** and if the **new** length is **more** than the **actual** description length, we have to change **both boolean** properties accordingly. If the new length is **less** than the actual, we have to **change** the description to show. The following image is blurred, try to implement the logic yourself:

Problem 4.4 Implement the ToggleImage() Function

Each time we click on "**Show Image**" we have to switch the boolean property to **true** and change the **image title** to "**Hide Image**". If the content is "**Hide Image**" we have to do the **opposite**.

Problem 4.5 Implement the HideDesc() Function

This function should **reset** the description to show and the description length counter and also **switch** both buttons.

5. Create Articles Component

After we have created the child component the only thing left is to create the articles component and **pass** the array of data from **parent** to **child**.

Create articles component using "**ng generate component articles**" and copy the following **css** inside "**articles.component.css**":

```
ul {
  display: flex;
  flex-flow: row wrap;
```

```
  justify-content: center;
  margin: 40px auto;
}
```

Problem 5.1 Implement Logic Inside the Component

Inside the articles component we need an array property which will hold our articles and we need to initialize the array inside the **ngOnInit** hook. In this case we create an **instance** of our article data.

*Note: This is not recommended since Angular has integrated **dependency injection** which is always a good practice to use, but since we don't know how to implement it yet this will do for now.*

```
export class ArticlesComponent implements OnInit {
  articles : Article[];

  constructor() { }

  ngOnInit() {
    this.articles = new ArticleData().getData();
  }
}
```

Problem 5.2 Create the Html Markup

Now we have to create the html markup for the articles component. We have to **iterate** through all articles using the ***ngFor** directive and **pass** each article and article description to the **child** article component using the square brackets. Also use the article component **selector** inside.

```
<ul>
  <app-article *ngFor="let a of articles"
   [article]="a"
   [articleDesc]="a.description">
  </app-article>
</ul>
```

Only thing left is to place the articles selector inside "**app.component.html**" using both tags:

<app-articles></app-articles>

6. Test the Application

If everything went smoothly you can test your application and try out all of the functionality. Show/Hide the description and Show/Hide the image. It should look like this: