Seneca
SCHOOL OF INFORMATION AND COMMUNICATIONS TECHNOLOGY

BTI425

Web Programming for Apps and Services

Notes	Weekly
Resources	Graded work
Policies	Standards
Professors & addendum	Code examples

Angular Components Example

In this document, you will create an app with several components.

The goal is to become comfortable with the idea of components, and the process of creating and displaying them.

Getting started, new project

Your professors believe that it is important to plan your work *before writing code*. How?

Visualize what you are trying to do, and then quickly draw a simple diagram. (Diagramming skills are important for a programmer. These skills help you plan and later code more effectively. They also help you communicate with others, whether understanding someone else's plan, or selling your plan to someone else.)

A simple diagram will help you identify the information and functionality that your app needs. It will also naturally suggest what components it will need (and later services and so on).

> *Do this BEFORE WRITING CODE.*

For our example app, here's what we're trying to do. Each rectangle will (or could) have information and/or functionality inside.

Generate a new project

At this point, we will assume that you are comfortable using the Angular CLI to create a new project. For example, this command will create a new “animals” project:

```
ng new animals --routing -S -g
```

Recall:

The `--routing` option adds the code we need for “routing”, which is a topic that will be covered in detail next week. Adding routing now (when the new project is created) is a *best practice*.

The `-S` option does not add “testing” code. One of the effects is that it reduces the size of the project, and makes it slightly faster in the change detection and build processes.

The `-g` option does not create a Git repository for the project. That simplifies the configuration for us.

Adding the Bootstrap styles

As we have seen, a quick way to kickstart the style / structure of your project is to use the Bootstrap framework. (At the time of writing, Bootstrap version 3.3.7 is a good choice, as is version 3.4. Also, Bootstrap 4 is new, with many new things, and some breaking changes.)

We can add this framework by adding this now-familiar code to the <head> element in the `src/index.html` file:

```
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstra
```

Create the structural components

Create the app's basic structural components:

- Header
- Navigation (named "NavBar")
- Main content (named "Content")
- Secondary content (named "SideBar")
- Footer

This is done with Angular CLI commands:

```
ng g c header --flat -S
ng g c navbar --flat -S
ng g c content --flat -S
ng g c sidebar --flat -S
ng g c footer --flat -S

# If your ng version is 9 or later, omit -S...
ng g c header --flat
```

To review, the `ng g c` command creates the component's source code files, and updates the app module (by adding import-related code).

A few notes about the source code files and generated code, using the first command above, "header":

In `header.component.ts`:

- A class named `HeaderComponent` is created

- Its selector is `app-header`

In `header.component.html`, some getting-started text is generated. We always replace that generated text with our own. For example (using Bootstrap classes):

```
<div class="header">
  <div class="row">
    <h3>Animals are awesome!</h3>
  </div>
</div>
```

Create components to support routing (discussed next week)

Create components to support routing:

- Home (a home or start page)
- Page not found (named “PageNotFound”)

This is done with Angular CLI commands:

```
ng g c home --flat -S
ng g c PageNotFound --flat -S
```

Notice the use of PascalCase on the “PageNotFound” name. The Angular CLI will parse that, and make the component’s file name assets use lower case, with dash separators between the words. Nice.

Create components to hold content

Finally, create components to hold content; we’ll use an *animals* theme:

- Horse
- Lizard
- Bear
- Eagle
- Dolphin

This is done with Angular CLI commands:

```
ng g c horse --flat -S
```

```
ng g c lizard --flat -S
ng g c bear --flat -S
ng g c eagle --flat -S
ng g c dolphin --flat -S
```

At this point, you can edit the HTML template content to meet your needs. For example, the "horse" component has this HTML template to render a Bootstrap 3 "Panel" (the text content is from Wikipedia, and the image is from a Google search):

```
<div class="horse panel panel-default" id="horse">
  <div class="panel-heading">
    <h3 class="panel-title">Horse</h3>
  </div>
  <div class="panel-body">
    <p>The horse (Equus ferus caballus)[2][3] is one of two extant su
      belonging to the taxonomic family Equidae. </p>
    <p>The horse has evolved over the past 45 to 55 million years fro
      single-toed animal of today. Humans began to domesticate horses
      to have been widespread by 3000 BC. Horses in the subspecies ca
      populations live in the wild as feral horses. These feral popul
      to describe horses that have never been domesticated, such as t
      and the only remaining true wild horse. There is an extensive,
      concepts, covering everything from anatomy to life stages, size
    <img src="https://www.telegraph.co.uk/content/dam/news/2016/09/03
  </div>
</div>
```

Building the user interface

In the root of the project's file system, the `index.html` source code file can be left as-is, with no changes required.

The first substantial change will be in the app component's HTML template. (Why? Because that is used - bootstrapped - when the app loads for the first time.)

Remove the existing content, except for the `<router-outlet></router-outlet>` element.

Then, add elements for each of the structural components:

```
<app-header></app-header>
<app-navbar></app-navbar>
<div class="container"> <!-- All main "content" will be in a bootstra
```

```
    <app-content></app-content>
  </div>
  <app-footer></app-footer>
```

Notice the `app-content` selector. Let's look at the correspond "content" component next. Only its HTML template needs to be updated at the moment:

```
<div class="content row">
  <div class="col-md-12">
      <app-guide></app-guide>
      <app-home></app-home>
      <app-page-not-found></app-page-not-found>
      <app-horse></app-horse>
      <app-lizard></app-lizard>
      <app-bear></app-bear>
      <app-eagle></app-eagle>
      <app-dolphin></app-dolphin>
    </div>
</div>
```

Notice how the "content" component contains all of our other "content" and "routing" components. This corresponds directly with our initial design from the "Getting Started" section above! Once our components are defined, it's simple to place them within other components to create complex views.

Updating our Navigation component

For now, the navigation component - i.e. the menu bar - will simply be used to jump to a specific "content" component (ie: "horse" using it's "id" element, ie

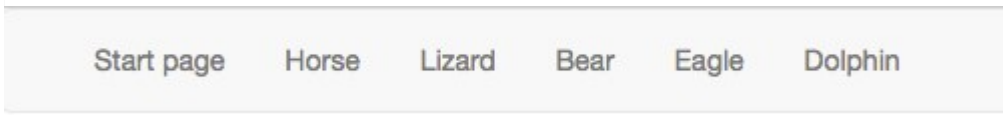
```
<a href="#horse">Horse</a>
```

Open the HTML template for the "navbar" component. Create a standard, Bootstrap 3 Navbar:

```
<nav class="navbar navbar-default">
  <div class="container-fluid">
    <div class="navbar-header"> <!-- home page link -->
      <a href="#start" class="navbar-brand">Start page</a>
    </div>
    <div> <!-- links to content -->
      <ul class="nav navbar-nav">
```

```
          <li>
            <a href="#horse">Horse</a>
          </li>
          <li>
            <a href="#lizard">Lizard</a>
          </li>
          <li>
            <a href="#bear">Bear</a>
          </li>
          <li>
            <a href="#eagle">Eagle</a>
          </li>
          <li>
            <a href="#dolphin">Dolphin</a>
          </li>
        </ul>
      </div>
    </div>
  </nav>
```

The result will look something like the following:



Study the code example

See the "animals" code example in the Week 7 folder of our BTI425 GitHub repository.

It implements the instructions in this document.

For best learning results, you should try to reproduce it.