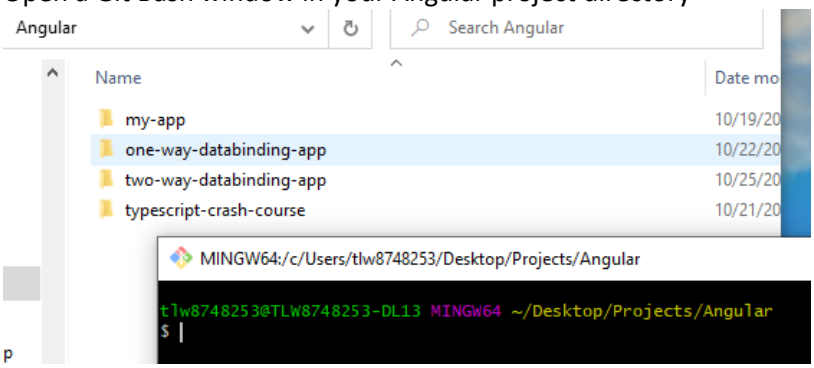
Table of Contents

Angular project structural-directives

This document is related to the Aug 25th 2021 recording related to the [Appendix: angular-directives.md]() notes, around timestamp 1:44:18.

Open a Git Bash window in your Angular project directory

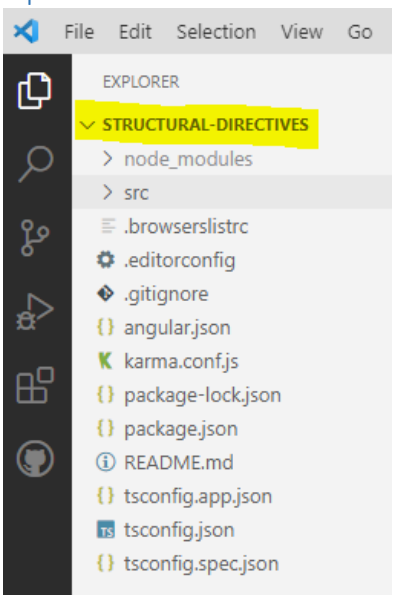


Create the structural-directives project

```
ng new structural-directives
```

Continue once ng new completes.

Open the folder in VS Code



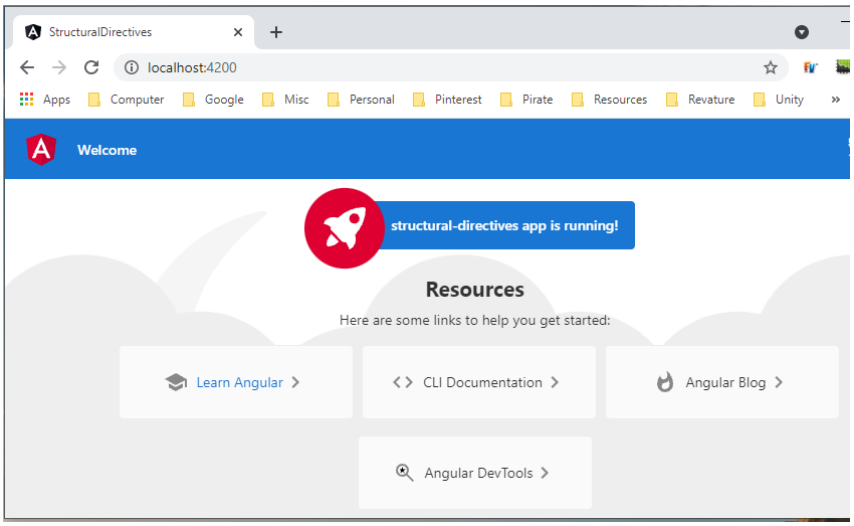
Run the application

```
cd structural-directives
npm run start
```

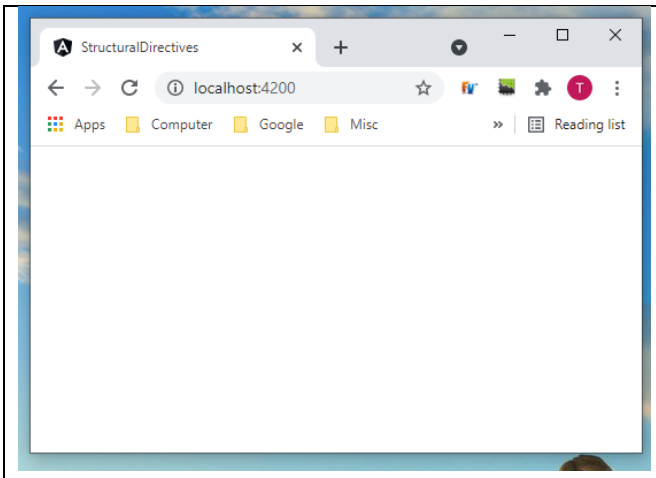
```
$ npm run start

> structural-directives@0.0.0 start C:\Users\tlw8748253\Desktop\Projects\Angular\structural-directives
> ng serve

- Generating browser application bundles (phase: setup)...
Compiling @angular/core : es2015 as esm2015
Compiling @angular/common : es2015 as esm2015
Compiling @angular/platform-browser : es2015 as esm2015
Compiling @angular/platform-browser-dynamic : es2015 as esm2015
√ Browser application bundle generation complete.

Initial Chunk Files   | Names         |      Size
vendor.js             | vendor        |   2.09 MB
polyfills.js          | polyfills     | 510.60 kB
styles.css, styles.js | styles        | 383.39 kB
main.js               | main          |  55.07 kB
runtime.js            | runtime       |   6.64 kB

| Initial Total |   3.02 MB

Build at: 2021-10-25T19:08:40.992Z - Hash: 936d93dab80585b783d2 - Time: 31918ms

** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost
4200/ **


√ Compiled successfully.
√ Browser application bundle generation complete.

5 unchanged chunks

Build at: 2021-10-25T19:08:42.907Z - Hash: 5b483ad0f0ad4db76d0e - Time: 1135ms

√ Compiled successfully.
```

Open the application in the browser.

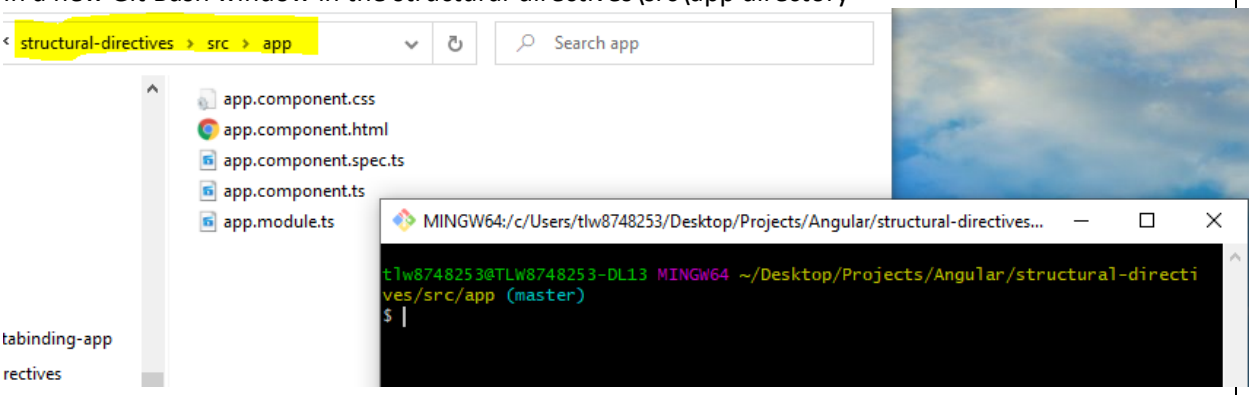
```
http://localhost:4200/
```



Remove the default html code from app.component.html
Save the changes automatically updates the browser.



In a new Git Bash window in the structural-directives\src\app directory



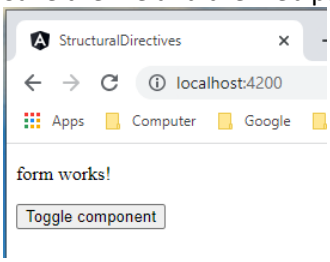
Create a new component called form

```
ng generate component form
```

Add the following code to app.component.html

```
<app-form></app-form>


<button>Toggle component</button>
```

Save the file and the web page updates



Install bootstrap in the Git Bash window:

```
npm install bootstrap
```

```
MINGW64:/c/Users/tlw8748253/Desktop/Proje

tlw8748253@TLW8748253-DL13 MIN
tives/src/app (master)
$ npm install bootstrap
```

Add global styling in styles.css

```
@import 'bootstrap/dist/css/bootstrap.min.css';
```

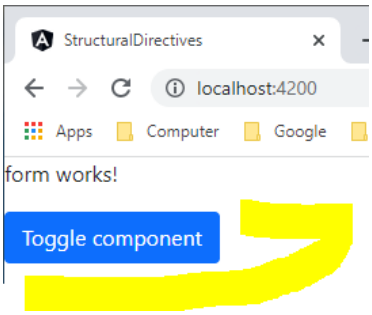
.

Update the following line of code in app.component.html

```
From

        <button>Toggle component</button>
To

        <button class="btn btn-primary">Toggle component</button>
```

```
app.component.html M ×   # styles.css M
src > app > app.component.html > button.btn.btn-primary
1  <app-form></app-form>
2
3  <button class="btn btn-primary">Toggle component</button>
```

Save the file and see the button style change:



Add ngIf structural directive to program

In the app.component.ts file add the following:

```
formComponentShouldBeDisplayed: boolean = true;
```

```
app.component.html M   TS app.component.ts M ×   # styles.css
c > app > TS app.component.ts > AppComponent > formComponei
1   import { Component } from '@angular/core';
2
3   @Component({
4     selector: 'app-root',
5     templateUrl: './app.component.html',
6     styleUrls: ['./app.component.css']
7   })
8   export class AppComponent {
9     title = 'structural-directives';
10
11    formComponentShouldBeDisplayed: boolean = true;
12  }
```

Update the following line of code in file app.component.html

```
From:

        <app-form></app-form>
To:
```

```
<app-form *ngIf="formComponentShouldBeDisplayed"></app-form>
```

app.component.html M × | app.component.ts M | styles.css M

src > app > app.component.html > app-form

```
1 <app-form *ngIf="formComponentShouldBeDisplayed"></app-form>
2
3 <button class="btn btn-primary">Toggle component</button>
```

Add event binding to the code

First add an event listener to app.component.ts

```
onToggleButtonClick() {
  this.formComponentShouldBeDisplayed = !this.formComponentShouldBeDisplayed;
}
```

app.component.html M | app.component.ts M × | styles.css M

src > app > app.component.ts > ...

```
1  import { Component } from '@angular/core';
2
3  @Component({
4    selector: 'app-root',
5    templateUrl: './app.component.html',
6    styleUrls: ['./app.component.css']
7  })
8  export class AppComponent {
9    title = 'structural-directives';
10
11   formComponentShouldBeDisplayed: boolean = true;
12
13   onToggleButtonClick() {
14     this.formComponentShouldBeDisplayed = !this.formComponentShouldBeDisplayed;
15   }
16
17 }
18
```

Next add the event to app.component.html

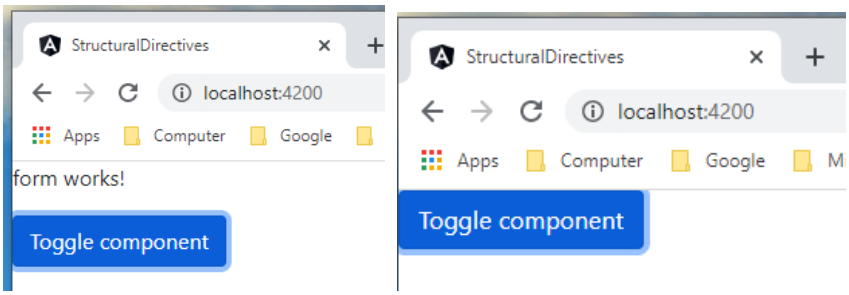
```
<button class="btn btn-primary"(click)="onToggleButtonClick()">Toggle component</button>
```

app.component.html M × | app.component.ts M | styles.css M

src > app > app.component.html > button.btn.btn-primary

```
1 <app-form *ngIf="formComponentShouldBeDisplayed"></app-form>
2
3 <button class="btn btn-primary"(click)="onToggleButtonClick()">Toggle component</button>
```

Test the button



Add ngFor structural directive to program

Update the app.component.html file to look like the following.

Note adding mt-5 adds a margin from global bootstrap styling.

```
<div class="container">
  <app-form *ngIf="formComponentShouldBeDisplayed"></app-form>

  <button class="btn btn-primary mt-5"(click)="onToggleButtonClick()">Toggle component</button>
</div>
```

app.component.html M × | form.component.html U | app.component.ts M | styles.css M

src > app > app.component.html > div.container

```
1 <div class="container">
2   <app-form *ngIf="formComponentShouldBeDisplayed"></app-form>
3
4   <button class="btn btn-primary mt-5" (click)="onToggleButtonClick()">Toggle component</button>
5 </div>
```

Modify the form.component.html file

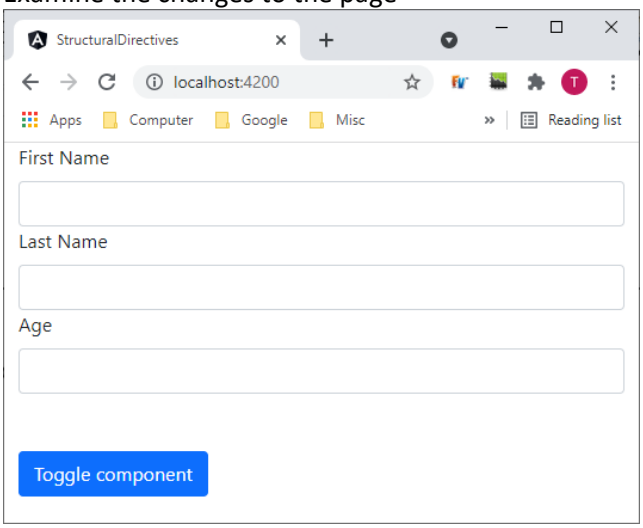
```
<div>
    <label class="form-label">First Name</label>
    <input class="form-control" type="text" />
</div>
<div>
    <label class="form-label">Last Name</label>
    <input class="form-control" type="text" />
</div>
<div>
    <label class="form-label">Age</label>
    <input class="form-control" type="number" />
</div>
```

app.component.html M | form.component.html U × | app.

src > app > form > form.component.html > div

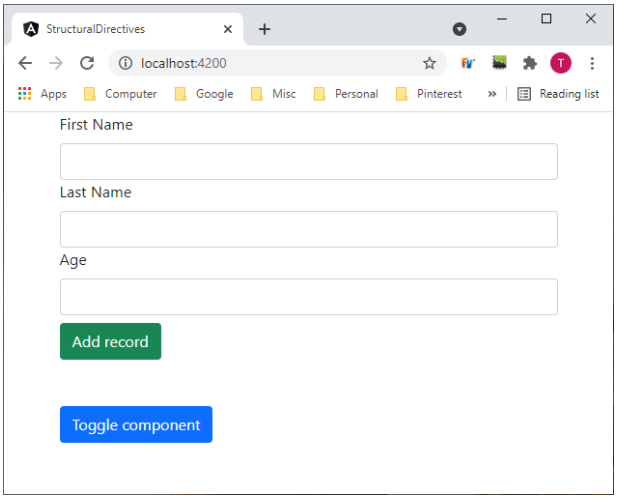
```
1  <div>
2      <label class="form-label">First Name</label>
3      <input class="form-control" type="text" />
4  </div>
5  <div>
6      <label class="form-label">Last Name</label>
7      <input class="form-control" type="text" />
8  </div>
9  <div>
10     <label class="form-label">Age</label>
11     <input class="form-control" type="number" />
12 </div>
```

Examine the changes to the page



Update form.component.html with an Add record button

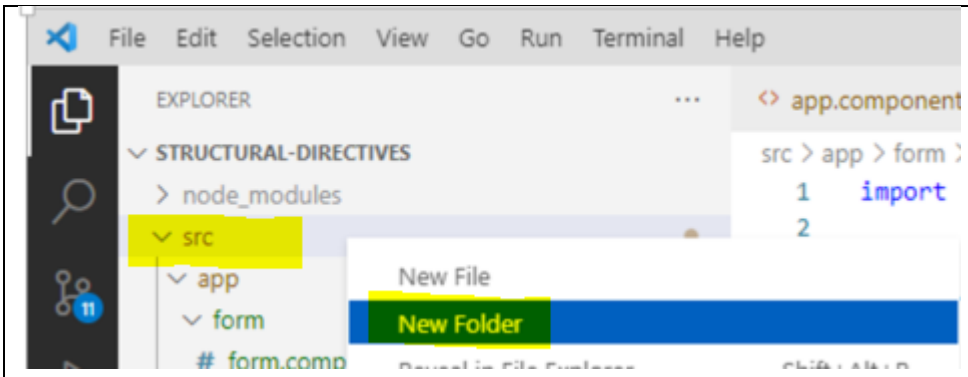
```
<div>
    <button class="btn btn-success mt-2">Add record</button>
</div>
```



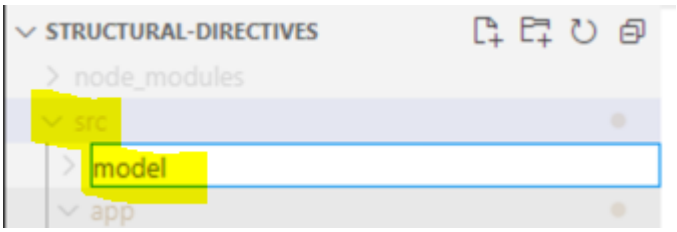
Update form.component.ts to store the add record components

First create an interface to store record information

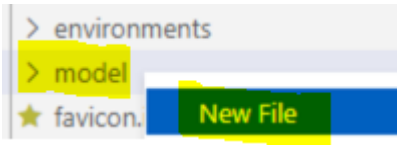
Create a new folder (model) within the “src” folder
Right mouse click over the “src” folder
Select New Folder



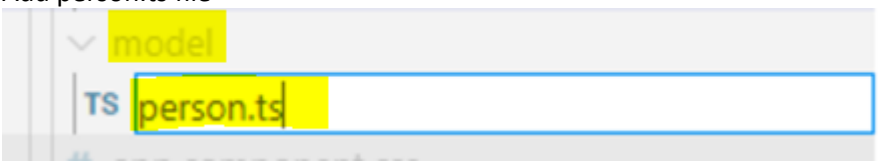
Add the folder model



Create new file in the model folder
Right mouse click on model
Select New File



Add person.ts file



Add code to the person.ts file
With TypeScript you can have attribute definitions in the interface.
Keyword export makes the interface visible for use.

```
export interface Person {
    firstName: string,
    lastName: string,
    age: number
  }
```

src > app > model > TS person.ts > ...

```
1  export interface Person {
2      firstName: string,
3      lastName: string,
4      age: number
5  }
```

Import interface into form.component.ts and add data structure

```
…
import { Person } from '../../model/person';
…
…
  firstNameInputValue: string = "";
  lastNameInputValue: string = "";
  ageInputvalue: number = 0;

  people: Person[] = [];
```

src > app > form > TS form.component.ts > FormComponent

```
1   import { Component, OnInit } from '@angular/core';
2
3   import { Person } from '../../model/person';
4
5   @Component({
6     selector: 'app-form',
7     templateUrl: './form.component.html',
8     styleUrls: ['./form.component.css']
9   })
10  export class FormComponent implements OnInit {
11
12    firstNameInputValue: string = "";
13    lastNameInputValue: string = "";
14    ageInputvalue: number = 0;
15
16    people: Person[] = [];
17
18    constructor() { }
```

Use two-data-binding to connect our record to the html form

Need to import the FormsModule in file app.module.ts

```
…
import { FormsModule } from '@angular/forms';
…
…
    BrowserModule,
    FormsModule
…
```

src > app > TS app.module.ts > ...

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { FormsModule } from '@angular/forms';

import { AppComponent } from './app.component';
import { FormComponent } from './form/form.component';

@NgModule({
  declarations: [
    AppComponent,
    FormComponent
  ],
  imports: [
    BrowserModule,
    FormsModule
  ],
```

Update form.component.html with data binding directive [(ngModel)].
This will bind the form data to our attributes in form.component.ts file

```
...
<input [(ngModel)]="firstNameInputValue" class="form-control" type="text" />
...
<input [(ngModel)]="lastNameInputValue" class="form-control" type="text" />
...
<input [(ngModel)]="ageInputvalue" class="form-control" type="number" />
...
```

src > app > form > form.component.html > ...

```
<div>
    <label class="form-label">First Name</label>
    <input [(ngModel)]="firstNameInputValue" class="form-control" type="text" />
  </div>
  <div>
    <label class="form-label">Last Name</label>
    <input [(ngModel)]="lastNameInputValue" class="form-control" type="text" />
  </div>
  <div>
    <label class="form-label">Age</label>
    <input [(ngModel)]="ageInputvalue" class="form-control" type="number" />
  </div>
```

Create Add Record event and listener

Update form.component.html with event

```
<button (click)="addRecord()" class="btn btn-success mt-2">Add record</button>
```

src > app > form > form.component.html > div

```
<div>
    <label class="form-label">First Name</label>
    <input [(ngModel)]="firstNameInputValue" class="form-control" type="text" />
</div>
<div>
    <label class="form-label">Last Name</label>
    <input [(ngModel)]="lastNameInputValue" class="form-control" type="text" />
</div>
<div>
    <label class="form-label">Age</label>
    <input [(ngModel)]="ageInputvalue" class="form-control" type="number" />
</div>
<div>
    <button (click)="addRecord()" class="btn btn-success mt-2">Add record</button>
</div>
```

Update form.componet.ts with a listener

Retrieve values from the form and push the values to the Person data structure.

```
addRecord() {
  let person: Person = {
    'firstName': this.firstNameInputValue,
    'lastName': this.lastNameInputValue,
    'age': this.ageInputvalue
  }

  this.people.push(person);
}
```

src > app > form > TS form.component.ts > FormComponent

```
1  import { Component, OnInit } from '@angular/core';
2
3  import { Person } from '../../model/person';
4
5  @Component({
6    selector: 'app-form',
7    templateUrl: './form.component.html',
8    styleUrls: ['./form.component.css']
9  })
10 export class FormComponent implements OnInit {
11
12   firstNameInputValue: string = "";
13   lastNameInputValue: string = "";
14   ageInputvalue: number = 0;
15
16   people: Person[] = [];
17
18   constructor() { }
19
20   ngOnInit(): void {
21   }
22
23   addRecord() {
24     let person: Person = {
25       'firstName': this.firstNameInputValue,
26       'lastName': this.lastNameInputValue,
27       'age': this.ageInputvalue
28     }
29
30     this.people.push(person);
31   }
32
33 }
```

Add record display elements using ngFor

Add display elements to for.component.html

The `<tbody *ngFor="let person of people">` code with the <td> definitions are what controls the mapping and display of records from the people data structure to the table in the html.

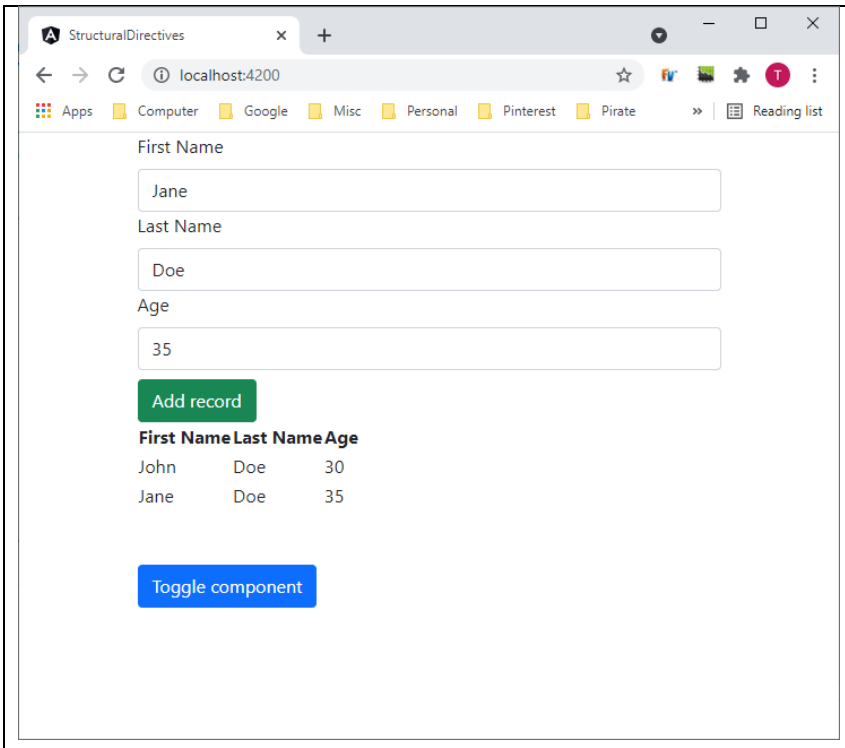
```
<table>
    <thead>
      <tr>
        <th>First Name</th>
        <th>Last Name</th>
        <th>Age</th>
```

```
      </tr>
    </thead>
    <tbody *ngFor="let person of people">
      <tr>
        <td>{{ person.firstName }}</td>
        <td>{{ person.lastName }}</td>
        <td>{{ person.age }}</td>
      </tr>
    </tbody>
  </table>
```

src > app > form > form.component.html > table

```
<div>
    <label class="form-label">First Name</label>
    <input [(ngModel)]="firstNameInputValue" class="form-control" type="text" />
  </div>
<div>
    <label class="form-label">Last Name</label>
    <input [(ngModel)]="lastNameInputValue" class="form-control" type="text" />
  </div>
<div>
    <label class="form-label">Age</label>
    <input [(ngModel)]="ageInputvalue" class="form-control" type="number" />
  </div>
<div>
    <button (click)="addRecord()" class="btn btn-success mt-2">Add record</button>
  </div>

<table>
    <thead>
      <tr>
        <th>First Name</th>
        <th>Last Name</th>
        <th>Age</th>
      </tr>
    </thead>
    <tbody *ngFor="let person of people">
      <tr>
        <td>{{ person.firstName }}</td>
        <td>{{ person.lastName }}</td>
        <td>{{ person.age }}</td>
      </tr>
    </tbody>
  </table>
```

As you add records in the web page, they will be added to the display table



Update program with attribute directive ngClass

Change code in app.component.html. The new code will switch between red and green colored button for the "Toggle component" button.

From:

```
<button class="btn btn-primary mt-5" (click)="onToggleButtonClick()">Toggle component</button>
```

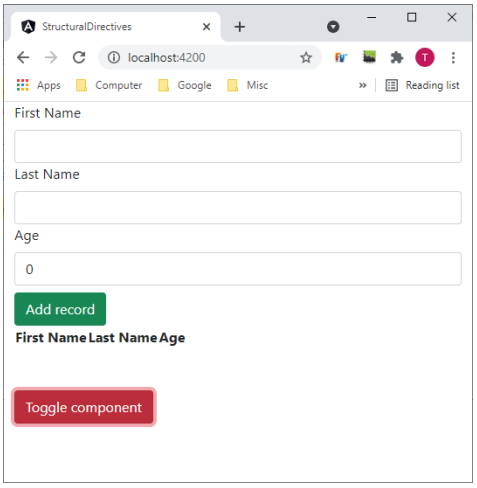
To:

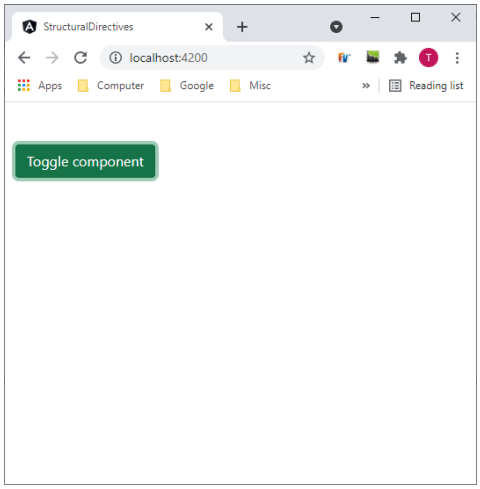
```
  <button
  [ngClass]="{
    'btn': true,
    'btn-danger': formComponentShouldBeDisplayed,
    'btn-success': !formComponentShouldBeDisplayed,
    'mt-5': true
  }"
  (click)="onToggleButtonClick()">Toggle component</button>
```

src > app > app.component.html > div.container

```
<div class="container">
  <app-form *ngIf="formComponentShouldBeDisplayed"></app-form>

  <button
  [ngClass]="{
    'btn': true,
    'btn-danger': formComponentShouldBeDisplayed,
    'btn-success': !formComponentShouldBeDisplayed,
    'mt-5': true
  }"
  (click)="onToggleButtonClick()">Toggle component</button>

</div>
```

Results:





Update program with attribute directive ngStyle

Add <h1> header to form app.component.html.

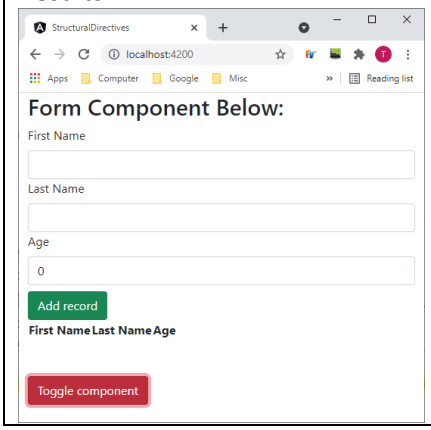
The [ngStyle] directive will flip the header between visible and hidden but not remove the element.

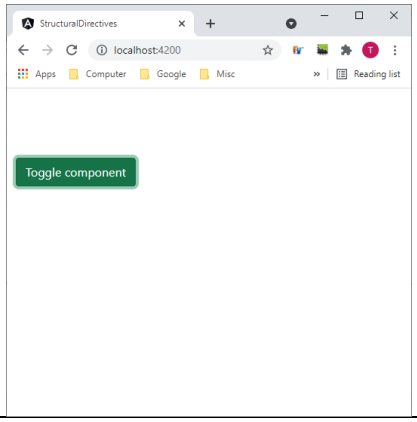
```
<h1 [ngStyle]="{ 'visibility': formComponentShouldBeDisplayed ? 'visible' : 'hidden' }">Form
Component Below:</h1>
```

src > app > app.component.html > div.container

```
<div class="container">
  <h1 [ngStyle]="{ 'visibility': formComponentShouldBeDisplayed ? 'visible' : 'hidden' }">Form Component Below:</h1>
  <app-form *ngIf="formComponentShouldBeDisplayed"></app-form>

```

Results



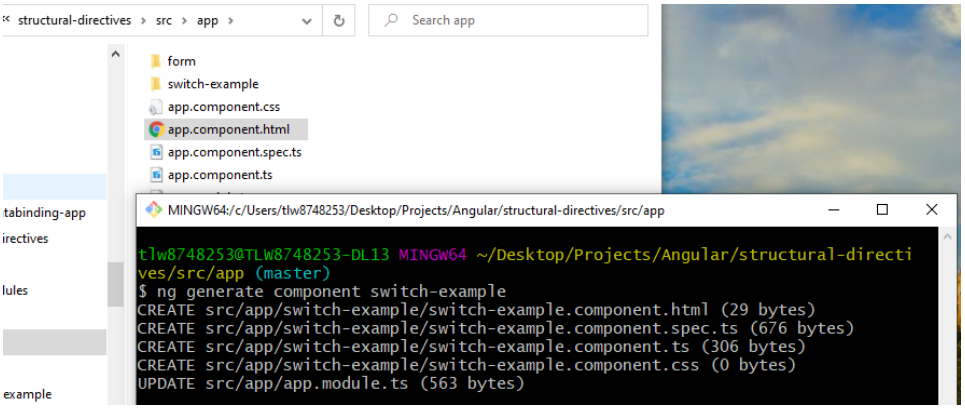


Update program with structural directive ngSwitch

Create a new application component.

In the Git Hub window in the ../src/app folder, enter:

```
ng generate component switch-example
```



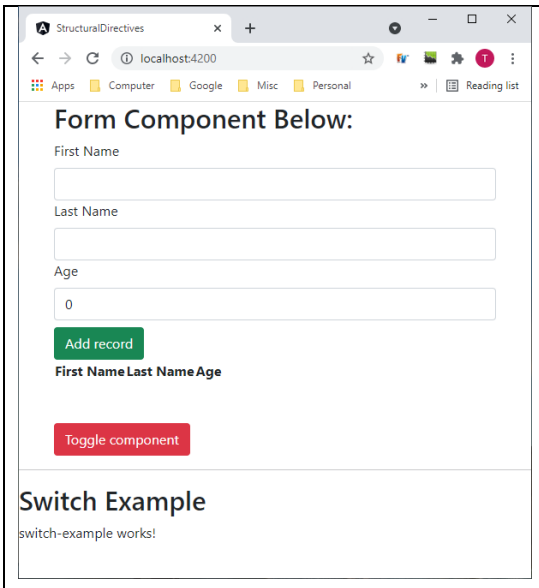
Add new component to app.component.html

```
<hr>
<h1>Switch Example</h1>
<app-switch-example></app-switch-example>
```

src > app > app.component.html > app-switch-example

```
<div class="container">
  <h1 [ngStyle]="{ 'visibility': formComponentShouldBeDisplay
  <app-form *ngIf="formComponentShouldBeDisplayed"></app-form

  <button
  [ngClass]="{
    'btn': true,
    'btn-danger': formComponentShouldBeDisplayed,
    'btn-success': !formComponentShouldBeDisplayed,
    'mt-5': true
  }"
  (click)="onToggleButtonClick()">Toggle component</button>
</div>

<hr>
<h1>Switch Example</h1>
<app-switch-example></app-switch-example>
```

Results



Update the switch-example.component.html file. Replace default code with a ngSwitch example:

```
<div [ngSwitch]="someVariable">
    <p *ngSwitchCase="'first-case'">This is the first case</p>
    <p *ngSwitchCase="'second-case'">This is the second case</p>
    <p *ngSwitchCase="'third-case'">This is the third case</p>
    <p *ngSwitchDefault>This is the default case</p>
</div>

<div>
    <button (click)="onButtonOneClick()">Button 1</button>
    <button (click)="onButtonTwoClick()">Button 2</button>
    <button (click)="onButtonThreeClick()">Button 3</button>
</div>
```

src > app > switch-example > switch-example.component.html > div

```
1   <div [ngSwitch]="someVariable">
2       <p *ngSwitchCase="'first-case'">This is the first case</p>
3       <p *ngSwitchCase="'second-case'">This is the second case</p>
4       <p *ngSwitchCase="'third-case'">This is the third case</p>
5       <p *ngSwitchDefault>This is the default case</p>
6   </div>
7
8   <div>
9       <button (click)="onButtonOneClick()">Button 1</button>
10      <button (click)="onButtonTwoClick()">Button 2</button>
11      <button (click)="onButtonThreeClick()">Button 3</button>
12  </div>
```

Update the switch-example.component.ts file as follows:

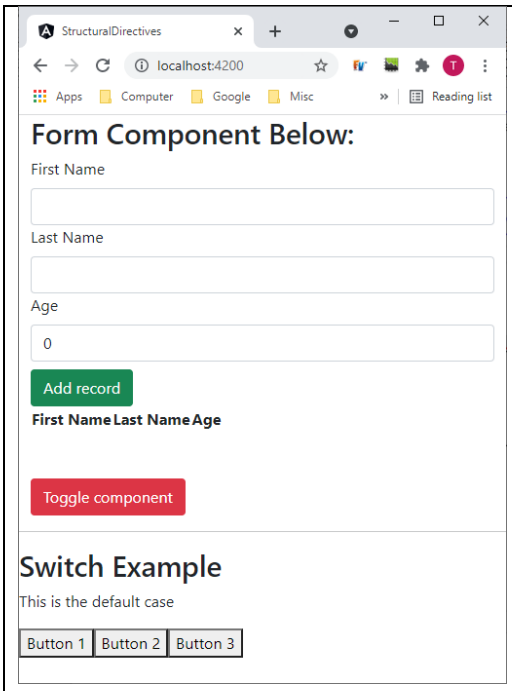
```
…
someVariable: string = "";
…
  onButtonOneClick() {
```

```
    this.someVariable = 'first-case';
  }

  onButtonTwoClick() {
    this.someVariable = 'second-case';
  }

  onButtonThreeClick() {
    this.someVariable = 'third-case';
  }
…
```

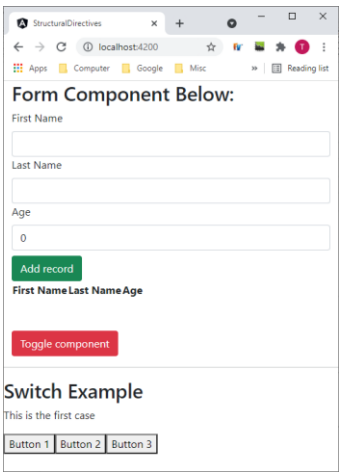
src > app > switch-example > TS switch-example.component.ts > SwitchExampleComponent > onButtonThreeClick

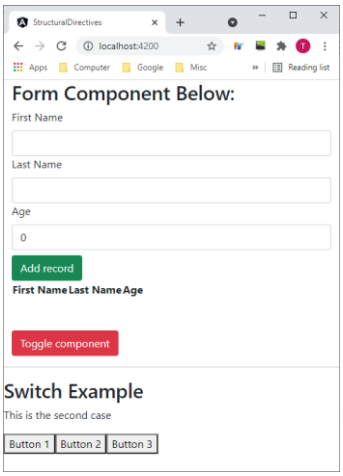
```
1  import { Component, OnInit } from '@angular/core';
2
3  @Component({
4    selector: 'app-switch-example',
5    templateUrl: './switch-example.component.html',
6    styleUrls: ['./switch-example.component.css']
7  })
8  export class SwitchExampleComponent implements OnInit {
9
10   someVariable: string = "";
11
12   constructor() { }
13
14   ngOnInit(): void {
15   }
16
17   onButtonOneClick() {
18     this.someVariable = 'first-case';
19   }
20
21   onButtonTwoClick() {
22     this.someVariable = 'second-case';
23   }
24
25   onButtonThreeClick() {
26     this.someVariable = 'third-case';
27   }
28
29 }
```

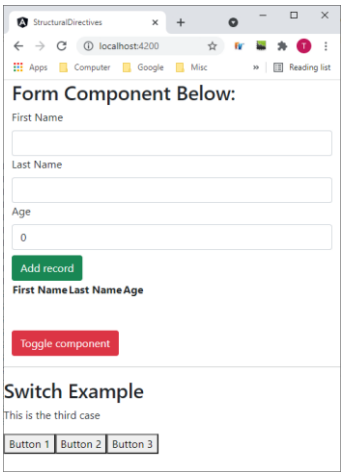
Page updates:



Click the buttons:







End of lecture and program.

Appendix: angular-directives.md

Directives

Directives are a construct of Angular. In particular, they "direct" elements within our component templates on what to do. There are 3 different types of directives:
1. Component: components are technically directives themselves, because they do indeed "direct" what should be rendered on the DOM
2. Structural Directives: used to manipulate and change the **structure** of the DOM
3. Attribute Directives: used to change the look of elements

Structural Directives
As stated previously structural directives manipulate the actual structure of the DOM. We can control when elements get displayed, how many of them get displayed, and switch to what gets displayed based on different conditions.

- *ngIf: used for conditional rendering. If an element with this directive evaluates to false, the element will not be displayed
- *ngFor: used to render a certain block of HTML multiple times. We can iterate over different data structures and populate the data according to what is contained in each iteration.
- ngSwitch
  - [ngSwitch]: attribute directive which controls
    - *ngSwitchCase: structural
    - *ngSwitchDefault: structural

Attribute Directives
Attribute directives are used to change the attributes of the DOM elements. There are two built-in attribute directives
- ngClass
- ngStyle

ngClass
ngClass is used for adding or removing the CSS classes from an HTML element. This allows us to apply classes dynamically based on a certain expression

```html
<div [ngClass]="<value>"></div>
```
- The value that can go inside of the double quotes `" "` can be
  - A string: `<div [ngClass]="'class-one class-two class-three'"></div>`
  - An array: `<div [ngClass]='['class-one', 'class-two', 'class-three']'></div>`
  - Object: `<div [ngClass]='{ 'class-one': aVariable === 'someStringValue', 'class-two': true, 'class-three': true }'></div>`

ngStyle
ngStyle is used when we want to dynamically change the style of an HTML element based on a certain expression