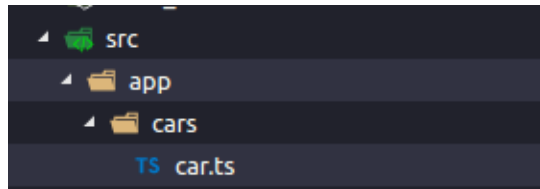


# петък, 28.07.2017

## Directives & Forms

- 1. Процедурата за създаването на форма е следната:

### - 1.1 - Създаваме си някакъв базов модел клас



```
index.html TS car.ts x
1
2 export class Car {
3   constructor(
4     public make: string,
5     public model: string,
6     public engine: number,
7     public milieage?: number
8   ) {}
9 }
```

### - 1.2 - След това създаваме компонент за формата

```
TS car.ts car-form.component.ts x ca
1 import { Component } from '@angular/core'
2 import { Car } from './car'
3
4
5 @Component({
6   selector: 'car-form',
7   templateUrl: './car-form.component.html'
8 })
9
10 export class CarFormComponent {
11   engines = [1.2, 1.6, 2.0, 4.0]
12
13   car = new Car('', '', 0)
14 }
```

**Както и html темплейта за компонента:**

```
TS car.ts x car-form.component.ts car-form.component.html x app.module.ts a
2 <h1>Car form: </h1>
3
4 <div class="row">
5   <div class="col-md-6 col-md-offset-3">
```

```

6
7     <form>
8         <div class="form-group">
9             <label for="make">Make:</label>
10            <input type="text" class="form-control" id="make" required="required" />
11        </div>
12
13        <div class="form-group">
14            <label for="model">Model:</label>
15            <input type="text" class="form-control" id="model" required="required" />
16        </div>
17
18        <div class="form-group">
19            <label for="engine">Engine:</label>
20            <select class="form-control" id="engine" required="required">
21                <option *ngFor="let engine of engines" [value]="engine">
22                    {{engine}}
23                </option>
24            </select>
25        </div>
26
27        <div class="form-group">
28            <label for="milieage">Milieage:</label>
29            <input type="text" class="form-control" id="milieage" required="required" />
30        </div>
31
32
33        <button type="submit" class="btn btn-success">Submit</button>

```

**ДА НЕ СЕ ЗАБРАВИ ДА СЕ ДОБАВИ КОМПОНЕНТА В *app.module.ts* .**  
**ВАЖНО Е ДА СЕ ИМПОРТНЕ И МОДУЛА ЗА ФОРМИТЕ, ИНАЧЕ НЯМА ДА**  
**ГРАЗПОЗНАЕ АТИБУТИТЕ.**

```

car.ts  car-form.component.ts  car-form.component.html
1  import { BrowserModule } from '@angular/platform-browser';
2  import { FormsModule } from '@angular/forms'
3  import { NgModule } from '@angular/core';
4
5  import { AppComponent } from './app.component';
6  import { CarFormComponent } from './cars/car-form.component'
7
8  @NgModule({
9      declarations: [
10         AppComponent,
11         CarFormComponent
12     ],
13     imports: [
14         BrowserModule,
15         FormsModule
16     ],
17     providers: [],
18     bootstrap: [AppComponent]
19 })
20 export class AppModule {}

```

**- 1.3 - Трябва да bind-нем пропъртите към input-тите, които потребителят попълва. ЗАДЪЛЖИТЕЛНО ТРЯБВА ДА СЛОЖИМ NAME АТРИБУТ НА ИНПУТИТЕ, ЗАЩОТО НЕ РАБОТИ БЕЗ ТЯХ.**

**Това, което ни позволява да правим със самата форма и Ангулар да прихваща стойностите се нарича [(ngModel)]** Това е двупосочен байндинг , което означава че при промяна и в кода , и в инпута двете се синхронизират и затова се пише с [], което означава че работи хем като Input, хем и като Output. И така.. в тага <input> слагаме [(ngModel)] и му казваме кое пропърти отговаря за този инпут.

```
<div class="form-group">
  <label for="model">Model:</label>
  <input
    type="text"
    class="form-control"
    id="model"
    required="required"
    name="model"
    [(ngModel)]="car.model"
  />
  {{car.model}}
</div>

<div class="form-group">
  <label for="engine">Engine:</label>
  <select
    class="form-control"
    id="engine"
    required="required"
    name="engine"
    [(ngModel)]="car.engine"
  >
    <option *ngFor="let engine of engines" [value]="engine">
      {{engine}}
    </option>
  </select>
  {{car.engine}}
</div>
```

**За следващите неща , за да работят върху формата трябва да и кажем, че е форма , тоест да се напише следното <form #carform="ngForm">, като #carform е избрано от нас име спрямо контекста, за който се използва.**

- 1.4 - По избор (опционално) могат да се добавят *css*(това са специфични ангулар неща), а не просто да си оцветим *input*-ите. Ангулар в зависимост от това дали нашите *input*-и са пипнати, променени, валидни и тн.. им слага специфични класове.

**А**нгулар слага следните класове:

- Visited - *ng-touched* / *ng-untouched*

- Changed - *ng-dirty* / *ng-pristine*

- Valid - *ng-valid* / *ng-invalid*

**К**огато искаме да реферираме някакви елементи от *html*-а и да ги използваме по надолу просто в таговете на елементите се слага **#някакво** име.. Това е показано на следващата снимка:

```
<form #carForm="ngForm">
  <div class="form-group">
    <label for="make">Make:</label>
    <input
      type="text"
      class="form-control"
      id="make" required="required"
      name="make"
      [(ngModel)]="car.make"
      #spy
    />
    <br />
    {{spy.className}}
    <!-- {{car.make}} -->
  </div>
```

**Т**ук *className* все едно означава *document.getElementById()*

```
1  /* You can add global styles to this file, and also import other style files */
2  input.ng-invalid.ng-dirty{
3    background-color: pink;
4  }
5
6  input.ng-valid.ng-dirty{
7    background-color: lightgreen;
8  }
```

### 1.5 - Да покажем някакви съобщения при валидация

**Как да сложим validation message ?** - На всеки `input` можем да му сложим референция и да му кажем, че той е `ng-Model`

-> `#make = "ngModel"`

**Вс е едно му казваме** - "Искам оттук нататък да използвам `make` като референция и да имам стойности свързани с `Model`"

#### ■ Adding additional validation messages

```
<input
  type="text"
  class="form-control"
  id="make"
  name="make"
  [(ngModel)]="model.make"
  required
  #make="ngModel" />
  .
  <div [hidden]="make.valid || make.pristine" class="alert alert-danger">
    Name is required!
  </div>
```

### 1.6 - И накрая да обработим самата форма при submit

#### Template-Driven Forms



- Finally, to submit the form

```
<form (ngSubmit)="onSubmit()" #carForm="ngForm">
```

- You may disable the button too, if you want

```
<button type="submit" [disabled]="!carForm.form.valid">Submit</button>
```

- More info

- Validation - <https://angular.io/guide/form-validation>
- Reactive forms - <https://angular.io/guide/reactive-forms>

- 2. Формите сами по себе си като логика се съдържат в отделен модул и ние трябва да си го импортваме.

## Етикети

- make
- carform
- някакво