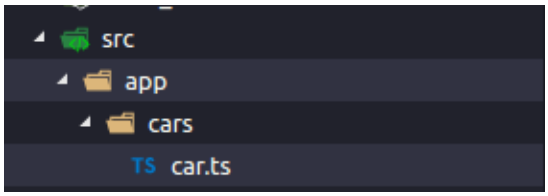
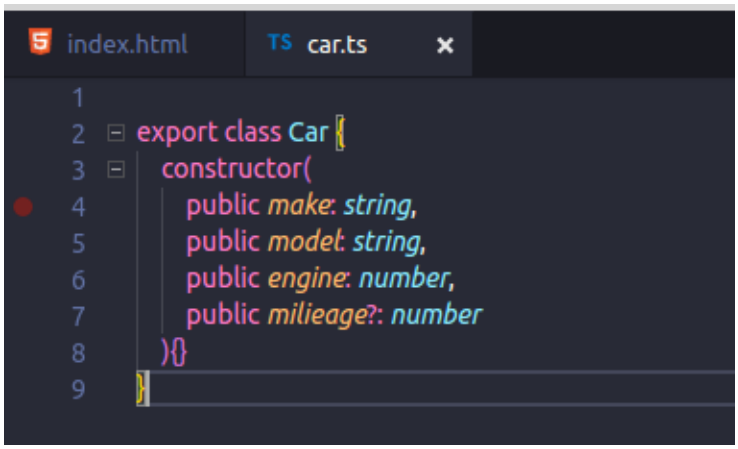
петък, 28.07.2017

Directives & Forms

- 1. Процедурата за създаването на форма е следната:

> - ***1.1 - Създаваме си някакъв базов модел клас***





> - ***1.2 - След това създаваме компонент за формата***

> ***Както и html темплейта за компонента:***

```
<form>
  <div class="form-group">
    <label for="make">Make:</label>
    <input type="text" class="form-control" id="make" required="required" />
  </div>

  <div class="form-group">
    <label for="model">Model:</label>
    <input type="text" class="form-control" id="model" required="required" />
  </div>

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

  <div class="form-group">
    <label for="milieage">Milieage:</label>
    <input type="text" class="form-control" id="milieage" required="required" />
  </div>


  <button type="submit" class="btn btn-success">Submit</button>
```

***Д**А 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

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

import { AppComponent } from './app.component';
import { CarFormComponent } from './cars/car-form.component'

@NgModule({
  declarations: [
    AppComponent,
    CarFormComponent
  ],
  imports: [
    BrowserModule,
    FormsModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

- ***1.3 - Трябва да bind-нем пропъртита 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 проняна 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.*

Ангулар слага следните класове:

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

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

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

Когато 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>
```

Тук className все едно означава document.getElementById()

car.ts | styles.css | car-form.component.ts | car-form.component

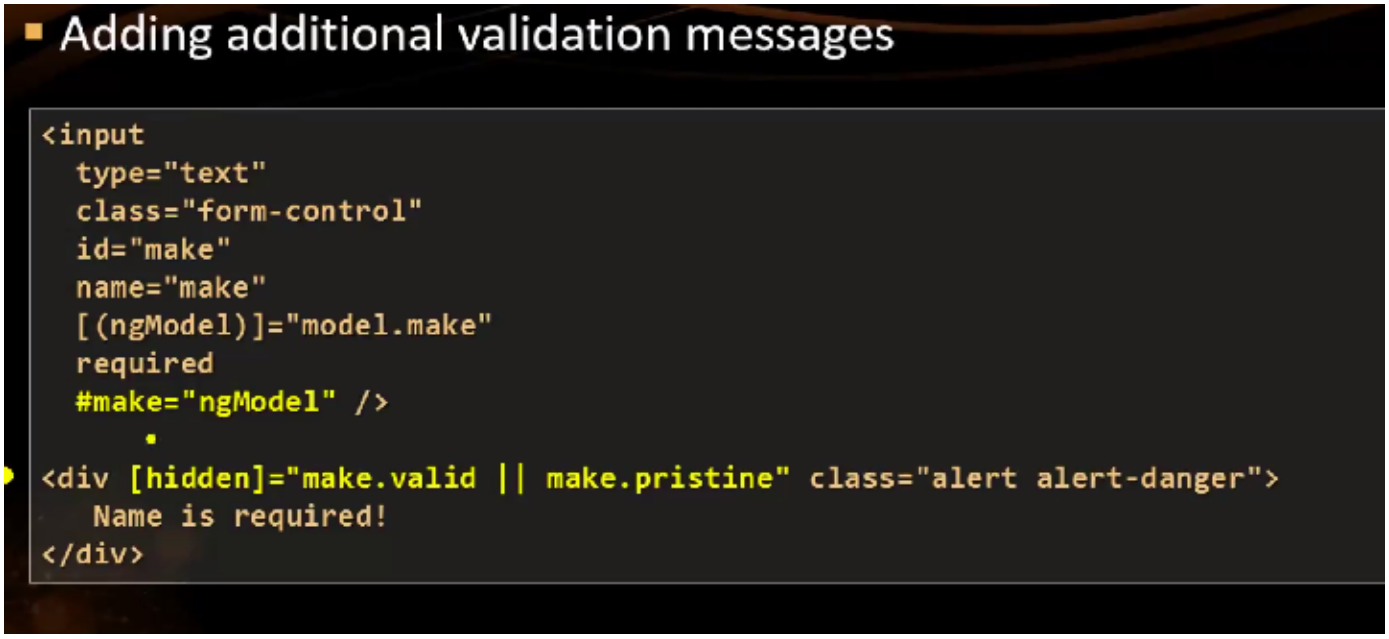
```
/* You can add global styles to this file, and also import other style files */
input.ng-invalid.ng-dirty{
  background-color: pink;
}

input.ng-valid.ng-dirty{
  background-color: lightgreen;
}
```

> **- *1.5 - Да покажем някакви съобщения при валидация***

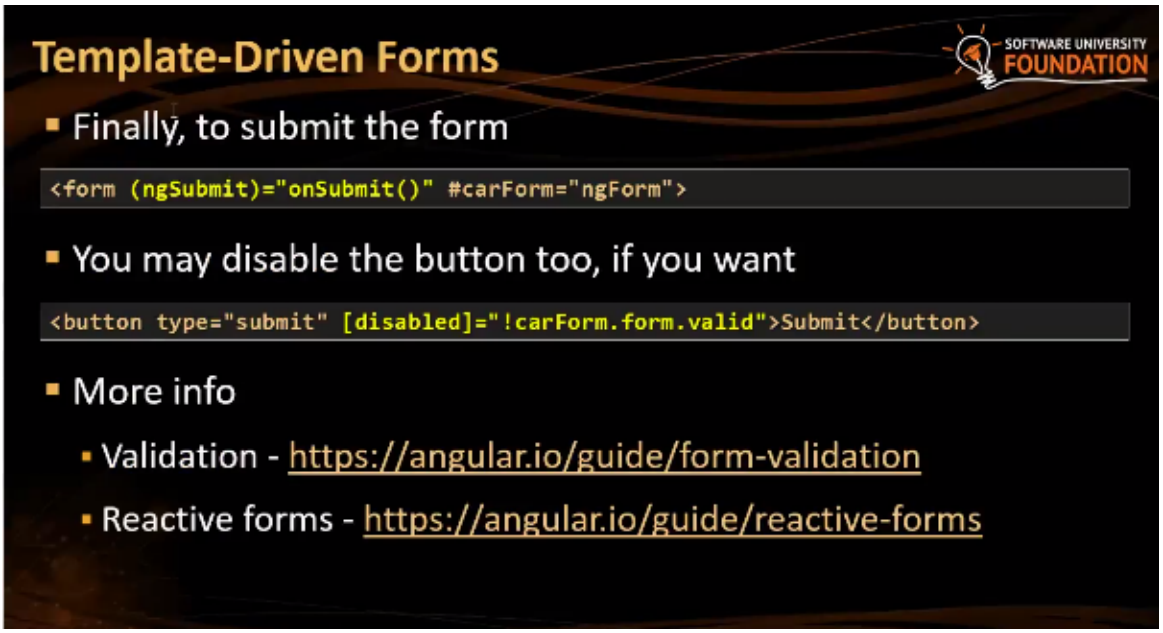
> ***Как да сложим validation message ? - На всеки input можем да му сложим*** *референция и да му кажем, че той е ng-Model*

-> **#make = "ngModel"**

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



> **- *1.6 - И накрая да обработим самата форма при submit***



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

Етикети

- make
- carform
- някакво