**1. Course Overview and Development Environment Setup**

5. Installing The Lessons Code - Learn About Git Remote vs Local Branches

**NodeJS**სასურველია **Current**ვერსია ჩავწეროთ ფრონტისთვის, რამდენადაც უფრო მეტად უჭერს მხარს.

**1. Course Overview and Development Environment Setup**

5. Installing The Lessons Code - Learn About Git Remote vs Local Branches

პირველ რიგში ვკლონავთ გიტს გიტ ბაშში:

git clone https://github.com/angular-university/angular-advanced-course.git

შემდეგ **vscode**-ში ვხსნით დაკლონილ პროექტს და შემდეგ ბრძანებით: git branchვამოწმებთ რომელ ბრენჩში ვიმყოფებით გლობალურად. წესით **master**-ში უნდა ვიყოთ.

შემდეგ ვამოწმებთ ჩვენთვის ხელმისაწვდომ რემოუტ ბრენჩებს: git branch -r

შემდეგ უნდა ავირჩიოთ ბრენჩი, რომლის გამოყენებაც გვსურს ლოკალურად:

git checkout -b au-input origin/au-input

ამის შემდეგ კი შეგვიძლია შევამოწმოთ git branch -ის ქომანდით რომ უკვე **au-input** ბრენჩში ვიმყოფებით უკვე.

**1. Course Overview and Development Environment Setup**

6. Installing Yarn and CLI, Setting Up a Development Server

შემდეგ ვაინსტალირებთ **npm**პაკეტებს: npm install -g @angular/cli

**cli**-ის ვერსიას შემდეგი ბრძანებით ვამოწმებთ: ng --version

შემდეგ ვაინსტალირებთ **yarn**-ის პაკეტის მენეჯერს: npm install -g yarn

შემდეგ: yarn -g (ლოკალური ფაილის დასაგენერირებლად)

**yarn npm**-ზე სწრაფი ბიბლიოთეკაა, იგი **fb**-ს მიერაა განვითარებული, იგი ასევე თავსებადია **npm**პაკეჯებთან.

მის ვერსიას ასე ვამოწმებთ: yarn -v

შემდეგ შევდივარ თავად პროექტის ფოლდერში და ვუშვებ ბრძანებას (იგი **npm install**-ის ალტერნატივაა): yarn

შემდეგ კი ვუშვებთ ბრძანებას: **npm start** , რომელიც დეველოპმენტ მოუდში გადაგვიყვანს.

**2. Our First Angular Library - The Font Awesome And Material Design Input Boxes**

9. Beginning The Implementation Of The Font Awesome Input Box

კლასების მინიჭების სტილი [**ngClass**]-ში:

**ts:**

1. @Input() icon: string;
2. get classes(){
3. const cssClasses = {
4. 'fa': true
5. }
6. if(this.icon){
7. cssClasses['fa-'+this.icon] = true
8. }
9. return cssClasses
10. }

**html:**

1. <i [ngClass]="classes"></i>

**hrml2:**

1. <div class="form-row">
2. <label>Email:</label>
3. <au-fa-input icon="envelope"></au-fa-input>
4. </div>

**2. Our First Angular Library - The Font Awesome And Material Design Input Boxes**

11. Learn An Angular CSS Extension Feature - The Host Pseudo Selector

ანგულარი გვთავაზობს **css**-ში**:host**სელექტორს, რომელიც თავად ჰოსტს დაასტაილებს.

**2. Our First Angular Library - The Font Awesome And Material Design Input Boxes**

12. Component Styling Best Practices - Ensure Solid Styles For Multiple Widget Sizes

**index.html** ფაილში შეგვიძლია ცალკეული კომპონენტების სტილიც კი გავწეროთ:

au-fa-input { height: 30px; }

**2. Our First Angular Library - The Font Awesome And Material Design Input Boxes**

13. Component API Design - Simpler and More Reusable Components With ng-content

ბესტ ფრაქთისი იქნებოდა, თუ ცალკეულ კომპონენტში შევქმნიდი მაგალითად ინფუთებს, ოღონდ არა თავად კომპონენტის თემფლეითში, არამედ სადაც მის სელექტორს ვიძახებ, შემდეგ კი **ng-content**-ის საშუალებით ანგულარს ვამცნობდი, რომ თაგებს შორის მოქცეული კონტენტი თავს კომპონენტში გადაიტანოს, თუმცა ამ შემთხვევაში სტილი ცოტა განსხვავებულად გაიწერებოდა. მაგალითი:

1. <au-fa-input icon="cc-stripe">
2. <input type="text" placeholder="Stripe">
3. </au-fa-input>

**2. Our First Angular Library - The Font Awesome And Material Design Input Boxes**

14. Understanding ng-content and Style Isolation - Learn The Deep Style Modifier

როდესაც კონტენტის პროჯექშენი ხდება**ng-content**-ით, ხოლო სტილი გაწერილი გვაქვს თავად იმ კომპონენტში, რომელშიც ვაპროჯექტებთ, აუცილებელია დიპ და ჰოსტ დონეზე მივწვდეთ ჩაპროჯექტებულ კონტენტს, რათა სტილი შევუცვალოთ, მაგალითად:

**au-fa-input.component.css:**

1. :host /deep/ input {
2. border: none;
3. outline: none;
4. height: 100%;
5. margin: 1px 0;
6. box-sizing: border-box;
7. }

**2. Our First Angular Library - The Font Awesome And Material Design Input Boxes**

16. The ContentChild Decorator, How Does It Work ? Component Design Best Practices

იმისათვის, რომ პროჯექტირებულ კონტენტს მივწვდეთ, უნდა გამოვიყენოთ ანგულარის დეკორატორი **@ContentChild** (ან **@ContentChildren**), რომელიც არგუმენტად მიიღებს კონკრეტულ ჩაპროჯექტებულ კონტენტზე არსებულ ლოქალ რეფერენსს, ხოლო მისი დაჭერა შეგვიძლია **AfterContentInit**ჰუკში:

**ts:**

1. @ContentChild('input', {static: false}) input: HTMLInputElement;
2. ngAfterContentInit(){
3. console.log('input', this.input)
4. }

**html:**

1. <au-fa-input icon="envelope">
2. <input #inputtype="email" placeholder="Email">
3. </au-fa-input>

თუმცა, რამდენადაც ლოქალ რეფერენსი არაა საუკეთესო გამოსავალი, შემდეგში ვნახავთ დაპროჯექტებული კონტენტის დანახვის სხვა გზას.

**2. Our First Angular Library - The Font Awesome And Material Design Input Boxes**

17. ContentChild, Directives and HostListener - Implementing the Input Focus Feature

წინაზე ნახსენები შემდეგნაირად გაკეთდება:

უნდა შევქმნათ დირექტივი და მის სელექტორში გავწეროთ კონკრეტული კომპონენტის სელექტორის სახელი და იმ თაგის, რომელის ინჯექტიც ხდება. და შემდეგ თავად ძირითად კომპონენტში **@ContentChild**-ს არგუმენტად ჩავაწვდით ამ დირექტივს, რამდენადაც იგი წვდება ყველა დაპროჯექტებულ ინფუთ ელემენტს (ჩვენს შემთხვევაში). ასევე, რამდენადაც, გვსურს, რომ ფოკუსს და ბლურს მოვუსმინოთ, გამოიყენებთ ამ კონკრეტულ დირექტივშივე **@HostListener**-ებს. ჩვენს შემთხვევაში:

**directive:**

1. @Directive({
2. selector: 'au-fa-input input'
3. })
4. export class InputRefDirective {
5. focus = false;
6. @HostListener('focus') onFocus(){
7. this.focus = true;
8. }
9. @HostListener('blur') onBlur(){
10. this.focus = false;
11. }
12. }

**main html:** ლოქალ რეფერენსები მოვუშალე

**au-fa-input.component.ts:**

1. @ContentChild(InputRefDirective, {static: false}) input: HTMLInputElement;

**2. Our First Angular Library - The Font Awesome And Material Design Input Boxes**

18. Implementing The input Focus Functionality - The HostBinding Decorator

**au-fa-input.component.css**-ში შეგვიძლია შემდეგ ჩავამატოთ შემდეგი ფსევდო სელექტორი, რომელსაც ასევე ანგულარი გვთავაზობს -:host(.input-focus){}, რომელიც დაემატება ან წაიშლება იმაზე დამოკიდებულებით, თუ დირექტივის    focus ცვლადი true არის თუ false.

შემდეგ თავად ამ კომპონენტის .ts-ში ვამატებ @HostBinding()-ს, რომელსაც არგუმენტად ჩაეწოდება კონკრეტული კლასის დამატების სტრინგი და გაუტოლდება ფუნქციას, რომელიც წარმაოდგენს get ფუნქციას და აბრუნებს true-ს ან false-ს. კოდი:

au-fa-input.component.ts:

1. @HostBinding('class.input-focus') get isInputFocus(){
2. return this.input ? this.input.focus : false;
3. }
4. ngAfterContentInit(){
5. if(!this.input){
6. console.error('the au-fa-input needs an input insde its content');
7. }
8. }

au-fa-input.component.css:

1. :host(.input-focus){
2. outline: none;
3. border: 1px solid #4D90FE;
4. -webkit-box-shadow: 0px 0px 5px #4D90FE;
5. box-shadow: 0px 0px 5px #4D90FE;
6. }

**3. Styling Angular Components - Best Practices**

19. Separating Structural Styles From Theme Styles - Making Components Themeable

ბესტ ფრაქთისი ინქეობდა, თუ **css**-ის 2 სხვადასხვა ფაილს შევქმნიდი კიდევ და იქ დავყოფდი სტილებს, შემდეგ კი, ამ მთავარ **css**-ში უბრალოდ **@inpirt "common.css"**; და**@import "au-fa-input-default-theme.css"**;-ს დავაიმპორტდებდი.

**3. Styling Angular Components - Best Practices**

20. How To Create An Alternative Component Theme And Ship It With The Library

შემდეგ ასევე შეგვიძლია შევქმნათ ცალკე **css**ფაილი, სადაც წითელი თემის პარამეტრები გაიწერება და ისიც ასევე მთავარ **css**ფაილში (**au-fa-input.component.css**) დაიმპორტდება. აღნიშნულ ფაილში გამოვიყენებთ ანგულარის მიერ მოწოდებულ ფსევდო სელექტორს: **:host-context(){}**, რომელსაც არგუმენტად ჩაეწოდება სასურველი საძიებო კლასი, აიდი და ა.შ. რათა დადგინდეს, რა შემთხვევაში მიებას სტილები.

ანუ, თუ**:host-context(.class){}**-ში ნახსენები**.class** კლასი აქვს რომელიმე ჰოსტ ელემენტის ელემენტს, იმ კონკრეტულ ელემენტს დაედება ის კლასი, რომელსაც ფიგურულ ფრჩხილებში ჩავწერთ.

**3. Styling Angular Components - Best Practices**

21. Creating an Alternative Component Theme, See the CLI Sass Support In Action

შემდეგ აღნიშნული ფაილი უნდა გადავაკეთოთ **scss**-ად და ასევე მთავარი **css**ფაილიც: **au-fa-input.component.scss.** **scss css**-ის წერას უფრო ამარტივებს. აღნიშნულ ფაილში იქნება შემდეგი კოდი:

**\_au-fa-input-red-theme.scss:**

1. $border-color: red;
2. :host-context(.au-fa-input-red-theme){
3. border-color: $border-color;
4. &.input-focus{
5. -webkit-box-shadow: 0px 0px 5px $border-color;
6. box-shadow: 0px 0px 5px $border-color ;
7. }
8. }

**3. Styling Angular Components - Best Practices**

24. Setting Up A Library Module, Confirming AOT is Supported

შემდეგ უკვე (სასურველია წითელი სტილის კლასები მოვაშოროთ) აღნიშნულ ბიბლიოთეკას ვუქმნით ცალკე მოდულს. ამ მოდულს ვაიმპორტებთ **app**-ის დონეზე.

ხოლო გარეთ, სადაც **src**ფოლდერიცაა, ვქმნით **index.ts** ფაილს და შიგნით ვწერთ შემდეგ ექსპორტს:

1. export { AuInputModule } from './src/app/lib/au-input.module';

**3. Styling Angular Components - Best Practices**

25. Angular Component Testing Overview

ტესტირებისთვის მოსამზადებლად, მაგალითად იმეილის და პაროლის ელემენტებს დავამატოთ აიდები და თავად ინფუთებს კი კლასები, **like this**:

1. <app-input-lib id="password-field" icon="fas fa-lock">
2. <input class="normal-input" type="email" name="email" placeholder="Password" class="test-class">
3. </app-input-lib>

**აქ ტესტირებაზეა თემები, რომლებიც თუ დამჭირდება, ვუყურებ. ახლა არ მიყურებია.**

**ერთი თემა გამოვტოვე კონსპექთში, თუმცა პრაქტიკაში მაქვს გაკეთებული (ng-temp-cont-content)**