Какво е Angular? Е фреймуърк за разработвана на фронт едн апликации на базата на тайп скрип, разработена от Аngular екипата на Google.

Базира се на компоненти и добре интегрирани библиотеки който покриват различни функции като рутиране, форми, комуникация клиент сървър.

Това което го отличава, че има скалируемост и актуализацията става максимално лесно

Какво Angula CLI? Is a command-line interface tool, който включва пълни набор от нужните ни инструменти за работа с Angular

Router – какво е и router outlet? **<router-outlet>** е директива в Angular, която се използва за динамично заместване на съдържанието в шаблона на компонентите в зависимост от текущия маршрут (route) в приложението. Това е един от ключовите елементи на Angular Router, който управлява навигацията и маршрутизацията в приложението.

--------------------------------------------------------------------------------------------------------------------------------------

Router – зарежда подходящото състояние на страницата когато пътя се смени (+ зарежда всички скриптове само веднъж, UI реагира бързо

Routing – Позволява навигация без презераждане на страницата

--------------------------------------------------------------------------------------------------------------------------------------

Видове директиви? Директивите са класове, които добавят допълнително поведение към елементи в едно приложение. Всеки един клас, който дава инструкции на Angular как да рендерира конкретен темплейт е директива.

Componentsdirectives - with template те са най често срещаните

Attribute directives – променя външния вид или поведението на елемнт, компонент или друга директива. Structural directives – променят DOM, чрез добавяне или премахване на DOM елементи с (\*ngIf and \*ngFor)

--------------------------------------------------------------------------------------------------------------------------------------

Как може да предаваме данни между компонентите? @Input() decorator, from parent to child

and from child to parent with @Output() decorators eventEmitter

--------------------------------------------------------------------------------------------------------------------------------------

Каква е разлиакта м/у constructor и ngOnInit

Конструктора е част от компонент класа изиква се когато angular създава и инициализира компонент. Използва се за обща инициализация, за DI когато компонента е инстанциран

ngOnInit is a lifecycle hook Това е метод, който Angular извиква, след като компонентът е инициализиран и свойствата му са зададени. Това е добро място за изпълнение на задачи, които включват извличане на данни, инициализации въз основа на входни свойства или всякакви други действия, които трябва да се извършат, след като компонентът е готов за стартиране.

Какво са Components? – са основните градивни елементи който съставят приложението. Компонентът включва TypeScript клас с @Component() декоратор, HTML и CCS.

HTML инструктира Angular как да изобрази компонента.

Основната им идея е да има сингъл респонсъбилити.

В него дефинираме апп логиката. Всеки компонент има жизнен цикъл който започва с инстанциране на компонент класа и рендериране на вюто.

Този цикъл приключва когато Angular унищожи инстанцията на компонента и махне рендерираня темплейт от DOM

Lifecycle hook ngOnChanges() – когато входаната или изходна стойност се променят

, ngOnInit() – след първата промяна на ngOnChanges(), ngOnDestroy() 8 на брой

Directives Lifecycle hook – те имат сходен жизнен цикъл, като те биват създавани, актуализирани и унищожавани в хода на изпълнение

Pipe – какво е? Вид функция която трансформира входните стойноти в изходни.

Полезни са защото се декларират само по веднъж

-----------------------------------------------------------------------------------------------------------------------------------

Subscribe – какво е и как се ползва? Функция, която определя как да се получат или генерират стойности или съобщения, които да бъдат публикувани.

Kогато се събскрайбнем към обзървъбъла се създава събскрипшън обек с три callbacks, next, error I complete

--------------------------------------------------------------------------------------------------------------------------------------

Observable – произвежада множество стойности, който пушва към свойте събскрайбъри. Използва се за асинхронна обработка на събития в Angular.

Със събскрайб метода се абонираме за него и може да го прекъснем по всяко време.

Observable are **hot** and **cold or shared/ not shared**, by default not shared.

--------------------------------------------------------------------------------------------------------------------------------------

Observer – е обект който е подаден на subscribe метода, като обекта дефинира callbacks за събскрайбърите, has methods next(),error() and complete()

-------------------------------------------------------------------------------------------------------------------------------------

Template- е кода който определя как да се рендерира изгледа на компоненнта

--------------------------------------------------------------------------------------------------------------------------------------

Template-driven-forms – при тях ползваме html фоми и инпут елементи от вюто.

The "source of truth" is the template. Двупосочното свързване с ngModel поддържа компонентния модел синхронизиран с въвежданите от потребителя входни данни.

Reactive forms – реактивните форми се създават чрез код в компонента

The "source of truth", the form model които е създаден в компонент класа.

Валидациите са чрез функции и тук темплейта не изполва ngModel

NgModule class – описва как отделните части на апп се фитват заедно, и казва как да се стартира приложението. Всяко приложение има поне един такъв, който се ползва за стартиране на приложението, обикновенно се наименова с appModule

--------------------------------------------------------------------------------------------------------------------------------------

Types of Data Binding:

-From data-source to view

-From view to data-source

-Two-way(FormsModule се изиква)

--------------------------------------------------------------------------------------------------------------------------------------

SOLID principles-

-single responsibility

-open/closed

-liskov substitution -

-interface segregation

-dependency inversion

- висок клас модулу да не зависят от нисък калс модулу А от абстракции

Coupling/Decamping - когато дадени елементи са зависими еднии от друг , не е добре да има couplig

Inversion of Control – когато пушваме депендънси в клас отвън(конструктора)

Използваме Services - защото компонентите ня трябва да фечват данни , те се грижат основно за презентирането на данните. Използваме ги за споделяне на информация м/у класове.

FRP - function reactive programing

RXJs -Reactive Extensions for JavaScript (operators map, filter, reduce)

Lazy Loading – помагани да заредим страницата на парче

Subjects -са специален вид обзървабъли

Behavior Subject - Съхранява най-новите стойности, излъчвани на своите потребители. Когато имаме нов събскрайбър то получва най актуалните стойности от Behavior Subject.

ReplaySubject е като BehaviorSubject само, че може да изпраща стари стойности на нови абонати

AsyncSubject е вариант, при който само последната стойност на изпълнението на Observable се изпраща на неговите наблюдатели

NgRx е библиотека за управление на състоянието в приложения, изградени с Angular. Тя се базира на концепцията на Redux - популярна архитектура за управление на състоянието във уеб приложения.

Основните елементи на NgRx са: store, actions, reducers, selectors & effects