<https://www.youtube.com/watch?v=kajJYsslE_M&list=PLeZsZBkrVMP-eYdS4rNJBe44rXAA5YSO4&index=4>

סיכום סרטון:

App.component.ts:

**Selector** איך אנחנו אחכ ניגשים למעשה לקומפוננטה.

Selector:'app-root',

בעצם בדף html של הקומפוננטה זו שהיא הראשית, השורש ניקרא לקומפוננטה לפי ה-selector אז ב-html נרשום כך בתגית body:

<app- root></app- root>

**tempelateUrl** מסביר לנו איך זה יראה בview כלומר מנתב אותנו לקובץ נגיד html שמוביל אותנו איך זה יראה בטכלס במסך מה שיהיה רשום בhtml ואם יש לנו משהו קטן להראות אז נרשום כך:

tempelate:'<h1>hellow world</h1>',

כלומר יראה לנו במסך את המילים:hellow world-זה כאשר יש דבר אחד קטן אך מלא דברים כבר נשלח אותו לקובץ. למשל html אז נרשום כך:

tempelateUrl :'./app.componenet.html',

עכשיו במחלקה:

Export class AppComponenet{

Titele='hellow world';

}

יש את המשתנה title ואז אנחנו מייבאים אותו כלומר ב-html רשום לי:

<h1>

{{title}}

</h1>

למעשה מכניס לנו את המשתנה הזה את מה שהוא מכיל בתוך ה-h1 ולכן בסופו של דבר יראה לנו על המסך את hellow world

**styleUrls** זה קובץ של css שממנו אפשר לעצב ולקשט את המסך הניראה אז בעצם נירשום כך:

styleUrls:['./app.component.css']

ואז בדף css עצמו נרשום ונעצב למשל:

פקודות בשורת הפקודה באנגולר:

ng g componenet new-cmp יצירת קומפוננטה חדשה למשל: שם הקומפוננטה- ng g componenet hellocomponenet

ng serve מאפשר לנו ליראות את הפרויקט שלנו ב-localhost

תגיות:

<br></br> תגית לרדת שורה.

<ul></ul> תגית לרשימה לא ממוספרת.

<li></li> תגית לאיבר ברשימה.

H1:{color:blue;}

אז זה סטייל פרטי לקומפוננטה הזו אז במקרה שלנו מה שהיה בh1 ב-html בעצם יצבע לי בכחול כלומר ה-title שבתוכו- hellow world כך בעצם במסך המילים הללו יראו.

בעצם עשינו כאן העברת מידע מהקומפוננטה לview ל- tempelate שלו -למסך הניראה.

עכשיו ניצור קומפוננטה חדשה ראו למטה במסגרת, לאחר שיוצרים קומפוננטה חדשה

כמובן שנוצר לי בכל קומפוננטה שניצור את דף html,ts,css וכו...

עכשיו יש אפשרות ליקרוא לקומפוננטה הזו מהקומפוננטה הראשית-קריאה של קומפוננטה פנימית מקומפוננטה חיצונית איך?

פשוט נרשום ב-html של הראשית (app-root)כך:

<app-hellowcomponent></ app-hellowcomponent>

ועכשיו אם נעשה משהו בcss של הקומפוננטה החדשה לדוג:

h3{color:red;}

וב-html של הקומפוננטה החדשה נרשום כך:

h3>hello-wcomponent works</h3> >כאן לא הכנסנו למשתנה כמו ה-title

אז עכשיו כשנריץ המסך יראה לנו כך:

hellow world-מהקומפוננטה הראשית

hello-wcomponent works -מהקופוננטה החדשה דרך הקומפוננטה הראשית.

שימו לב שאם רשמנוh1 בhtml אז גם בcss צריך שיהיה h1 ואם רשמנוh3 ב-html אז בcss יהיה h3.

<https://www.youtube.com/watch?v=reC73GBByEc&list=PLeZsZBkrVMP-eYdS4rNJBe44rXAA5YSO4&index=5>

סיכום סרטון:

העברת מידע מה- tempelateל-view

יצרנו כפתור(בראשית) ב- tempelate שזה ב-html:

<button(click)="onClick()">Click Me</button>

אז תעשה לי את הפונקציה הזו

באירוע של לחיצה

ואז ב-class ב- ts יצרנו את הפונקציהonClick:

onClick(){

console.log('Button Clicked');

}

ברגע שנילחץ ב-view על הכפתור הניראה אז תופעל פונקצייה זו ויודפס ב-console המילים:

Button Clicked

כמובן שאפשר להעביר נתונים מה-view ל-class מה-templent ל-component (class, ts)איך?

פשוט בתוך הסוגריים של הonClick:

<button (click)="onClick('hello from angular 4')">Click Me</button>

ואז ב-classבסוגריים של הפונקציה ניתפוס val ונדפיס ב-console

onClick(val){

console.log('Button Clicked');

console.log(val);

}

העברת נתונים על ה-event איך?

לירשום$eventב-html

<button (click)="onClick($event)">Click Me</button>

ואותו דבר ניתפוס את זה ע"י ה-val ואז נראה ב-console בעת הלחיצה על הכפתור את הנתונים על ה-event מה-view ל-component.

<https://www.youtube.com/watch?v=UuP2l4SM7js&list=PLeZsZBkrVMP-eYdS4rNJBe44rXAA5YSO4&index=6>

הסבר סרטון:

Class&&Style Binding

נכתוב ב-html

<button class="btn btn-primary"

[class.active]="isActive" אם הוא שווה לטרו

[style.backgroundColor]=" isActive?'blue':'gray'">Sumbit</button>

אם isActive הוא true אז הכפתור יהיה blue אחרת יהיה gray

זה כפתור של בוסטרף שכאשר עוברים עליו עם העכבר הוא משנה צבע לצבע חלש.

Binding של class שיאופשר בתנאי מסוים אז נרשום לפניכן ב-class :

isActive=true;

ה-style הישתנה בהתאם למשתנה isActive בהתאם לתנאי מסוים. כנ"ל גם class נתנו לו איזושהי תכונה בהתאם לתנאי מסוים.

<https://www.youtube.com/watch?v=SQGokZYa01A&list=PLeZsZBkrVMP-eYdS4rNJBe44rXAA5YSO4&index=7>

הסבר סרטון:

Two way data binding

זה אומר שיש שתי כיוונים של הזרימה של המידע, כיוון אחד מה-component ל-view(html), כיוון שני -מעבר נתונים -מעבר מה-view(templent,html) חזרה ל-component(class, ts)-כמו בסרטון הקודם שעשינו בלחיצה על כפתור.

דוגמא כיוון אחד מה-component ל-view(html) :(כמו הדוגמא שעשינו עם ה-title בסרטונים קודמים).

ב-component נרשום:

Person={

name:'Dani',

age:25

}

ב-html נרשום:

<h2>גודל הכתב שבפנים.

Person:<br>

Name:{{person.name}}<br> מביא את תוכן שבמשתנה שרשמנו ב- component .

Age:{{person.age}}<br> כנ"ל.

</h2>

דוגמא לכיוון שני מה-view ל-component:

נרשום ב-html:

<input type="text"[(ngModel)]="person.name"><br>

כמין תיבת טקסט שאפשר לרשום בתוכה input זה בעצם קלט, בהתחלה ניראה שרשום Dani כי זה בא בעצם מה- component אבל אם נרשום אנחנו בתוכה ב-view במסך הניראה נרשום למשל Moshe אז ניראה על המסך שה-name השתנה.

דוגמת הרצה בהתחלה:

Dani

Person:

Name: Dani

Age:25

דוגמת הרצה בעת הכנסה ל-input שם אחר:

Moshe

Person:

Name: Moshe

Age:25

דוגמא נוספת לכיוון שני מה-view ל-component:((כמו שעשינו בסרטונים קודמים)

ב-:html

<button (click)="onClick(person)">Click Me</button>

ב- component:

onClick(person){

console.log(person);

}

ברגע שנלחץ על הכפתור אז יודפס לי ב-console עם השם Dani אבל כשנשנה ב-input אז יודפס לי ב-console עם השם ששיננו.

<https://www.youtube.com/watch?v=SQGokZYa01A&list=PLeZsZBkrVMP-eYdS4rNJBe44rXAA5YSO4&index=7>

הסבר סרטון:

Directive -למעשה זה כל מני הוראות

ונתחיל עם הוראת ngFor -זה בעצם איזשהי לולאה שרצה על איזשהו מערך מסויים ומדפיסה את האיברים, עוברת על כל האיברים.

אז נרשום ב- ts:

ניצור לדוגמא מערך של קורסים:

Courses=[

{language:'C#',year:2015},

{language:'java',year:2014},

{language:'C++',year:2013}

]

נרשום ב-html :

{{courses}}- אם נרשום זאת ונריץ אז הוא ייתן לנו את האובייקטים:

[object.Object], [object.Object], [object.Object]-בעצם נתן לנו מערך של שלושה אובייקטים.

אנחנו רוצים כמובן לעשות באמצעות ה-directive של ה-ngFor ולעבור על כל אחד מהשדות של המערך ולהדפיס.

אם נרשום:

<ul>-רשימה לא ממוספרת

<li>abc</li>

<li>def</li>

<li>hij</li>

</ul>

בעצם ידפיס לי את המחרוזות הללו ברשימה מנוקדת:

**.**abc

**.**def

**.**hig

עכשיו כדי לעשות זאת באמצעות ngFor שזה יותר יעיל במקום לעבור את כל השורות לעבור פעם אחת ולעשות לולאה של פעם אחת אז נרשום כך:

<ul>

<li \*ngFor="let course of courses">{{course. language }}</li>

</ul>

אז בעצם הוא ידפיס את השפות שבמערך שעשינו ב- ts:

.C#

.java

.C++

כנ"ל לגבי השנה:

<ul>

<li \*ngFor="let course of courses"> language :{{course. language }} year: {{course. year }} </li>

</ul>

עברנו על מערך קורסים- courses והדפסנו בכל קורס course את השפה language והשנה year שלו

אם נשכלל את זה קצת אז נוסיף img

לדוג:

ב- ts נוסיף:

Courses=[

{language:'C#',year:2015,img:'……….'},-בגרשיים נרשום כתובת התמונה- קישור לתמונה

{language:'java',year:2014,img:'……….'},

{language:'C++',year:2013,img:'……….'}

]

ב-html נוסיף שבכל li יהיה לו img:

<ul>

<li \*ngFor="let course of courses">

<img src={{cource.imge}} style="width: 125px; height:125px"> -אפשר להגדיר את הסטייל גם פה ב-inLine וגם בקובץ css

language :{{course. language }} year: {{course. year }} </li>

</ul>

חשוב לרשום את ה-ngFor\* עם כוכבית לפני ואות F גדולה כי אחרת זה לא יעבוד.

<https://www.youtube.com/watch?v=Zz05zaKdkQE&list=PLeZsZBkrVMP-eYdS4rNJBe44rXAA5YSO4&index=9>

הסבר סרטון:

הוראת ngIf- מאפשר לבצע עבודה עם כל מני תנאים אם משהו מתקיים או לא מתקיים אז תעשה דברים מסוימים.

נרשום ב-html

<p \* ngIf="usValid">hello this is a p element</p>

תנאי מסוים.

עכשיו נרשוםב-js משתנה בשם isValid

isValid=true

אז לא יהיה פה שום שינוי הוא ידפיס על המסך הניראה את hello this is a p element

אך אם נרשום isValid=false אז הוא לא ידפיס כלום נמחק לנו ה-P וזה מכיוון שה-ngIf הגדרנו אותו ל-false אז לפי המשתנה הזה isValidככה ה- ngIfיודע האם להציג את הטקסט שיש ב-p או לא להציג .

נגדיר משהו שיקרה כאשר ה-if יתקיים או משהו שיקרה כאשר לא מתקיים else

<div \* ngIf="isValid; then content1;else content2">< / div >

<ng-template #content1> then content 1 is done…</ ng-template>

<ng-template #content1> else content 2 is done…</ ng-template>

אם isValid=trueאז יוצג לי then content 1 is done… וה-p הקודם שעשינו

אם isValid=falseאז ה content1נמחק (כנ"ל ה-p כפי שראינו קודם)-הוא לא מתקיים ויוצג לי else content 2 is done… שזה כן מתקיים.

<https://www.youtube.com/watch?v=qlxH2j27dMM&list=PLeZsZBkrVMP-eYdS4rNJBe44rXAA5YSO4&index=10>

הסבר סרטון:

סנקוריזציה ותקשורת בין שני קומפוננטות

יש לנו input – אומר קלט קשר בין האבא parent component לילד child component

ויש לנו outout -קשר הפוך, קשר בין הילד לאבא.

עכשיו נסביר על input

נחזור לhello. Component (app hello) לקומפוננטה הפנימית שיצרנו בסרטונים קודמים וזה בעצם הילד

יש לנו app-root שזה הקומפוננטה הראשית האבא וקראנו לילד דרך הselector-(app hello)

יש לנו input text בתוך האבא וכשנכניס משהו בטקסט נרצה שזה יתקשר עם הילד ויציג את השינוי על המסך

נרשום - ts של הילד hello. Component. ts לפני הבנאי:@Input() parentInput;-זה קלט ה-input אפשר לרשום ב מקום שורה זו גם [parentInput inputs:[ איפה שהסלקטור וכו.... אך אנו נשתמש ב@...

ולפניכן נרשום בהכי למעלה ב-import את Input:import{….,Input,…..}…. זה מאפשר בעצם לקבל inputim בלי לייבא לא נוכל לרשום את מה שלפני הבנאי זה יעשה שגיאה.

עכשיו נרצה להציג את הנתונים שאנו מקבלים ב-input על המסך אז נרשום ב-html של הילד

hello. Component.html:

<h2>

This is Child Component<br></br>

Value entered in parent component:{{parentInput}}

</h2>

עכשיו ניראה איך שולחים input מהאבא לילד אז נלך לאבא ונעשה את זה על ידי atirbute:

ב- app. Component.htmlנרשום בסלקטור של הילד שייבאנו בתוך האבא:

<p>This is Parent Component</p>

<input type="text"><br>

<app-hello [parentInput]="abc"></app-hello>

זה השם של ה-input שלנוזז

עכשיו כשנריץ אז מה שיודפס לי זה:

This is Parent Component

This is Child Component

Value entered in parent component: abc

כאן נרצה שזה לא יהיה דווקאabc נרצה לשנות זאת על ידי מה שנכניס בתוך ה-input כלומר ה- input שבאבא מה שיירשם בתוכו ילך לילד וירשם בו.

נרשום ב-html של האבא:

<p>This is Parent Component</p>

<input type="text" #myInput (keyup)="0"><br> יצרנו שם משתנה ל-input

<app-hello [parentInput]="myInput.value"></app-hello> וכאן קארנו לתוכן שבinputעל ידי ה- value בעצם השם משתנה myInput זה גישה לinput

מאפשר לנו לעקוב על הטקסט שהוקלד במקלדת זה יוצר את ה-event שבאירוע של הלחיצה על ה-input זה יוצג לי בילד

בעצם עשינו כאן העברת נתונים מהאבא לילד.

<https://www.youtube.com/watch?v=Z_eW6Ht6s0g&list=PLeZsZBkrVMP-eYdS4rNJBe44rXAA5YSO4&index=11>

הסבר סרטון:

Outputs- תקשורת מכיוון הילד לאבא.

בילד אין לנו איזשהו סלקטור כמו שעשינו בכיוון מהאבא לילד שבאבא היה לנו את הסלקטור של הילד-<app-hello ></app-hello> עכשיו כדי לעשות מכיוון של הילד לאבא נצטרך לעשות איזשהו event בילד.

נרשום ב- ts של הילד:

@Output() childChanged=new EventEmitter<string>();

כנ"ל לא לשכוח לעשות import

import{ Output,Input, EventEmitter }…..

לא לשכוח לעשות import למעלה להוסיף Output

import{ Output,Input,…..}…..

ברגע שיש לנו את ה-event הזה אנחנו יכולים להעביר נתונים מהילד לאבא

עכשיו נלך ל-html וניצור event בעת לחיצה על כפתור ולשייך אותו לevent החדש של ה- EventEmitter ולשלוח נתונים לפלט.

נלך ל-html של הילד ונרשום:

<input type="text" #childInput (keyup)="onChange(childInput.value)">

ברגע שנכניס נתונים ב-input של הילד אז זה יועבר לי לאבא ויוצג על המסך.

קראנו לinput בשם משתנה #childInput לצורך הגישה אליו.

נתנו איזשהו event שיעשה לנו תפונקציה onChange ויחזיר לנו childInput.value

שם משתנה שקראנו לinput

הערך של ה-input

עכשיו נעשה את הפונקציה onChange ב ts של הילד לפני הבנאי נרשום:

onChange(value:string)

{

this.childChanged.emit(value);

}

עכשיו ב- html של האבא נרשום:

<p>This is Parent Component</p><br>

Value entered in child component:{{childValue}}<br><Br>

<input type="text" #myInput (keyup)="0"><br>

<app-hello [parentInput]="myInput.value"

childChanged)=" childValue=$event"></app-hello>)

**השלבים שבעצם יצרנו:**

Hello.component.ts

קודם כל יצרנו משתנה Output שהוא למעשה event החוצה קראנו לו childChanged שקוראים לוEventEmitter שהוא למעשה event

Hello.component.html

קראנו שם ל-input בשם:#childInput ואז שייכנו ל-event של (keup) כשלוחצים על מקש, ברגע שלוחצים על מקש אז זה ילך ל-onChange

Hello.component.ts

onChange מקבל את ה-value את הערך של ה-input

ועושה-emit ל-event שעשינו ב-Output childchanged@ ומעבירה את ה-value

app.component.html-אבא

הוספנו את הפלט של ה-childValue}}}} ואת ה- childValueקיבלנו באמצעות האירוע childchanged-שזה קסטר זה אירוע שאנחנו יצרנו זה לא אירוע שקיים כבר במערכת כמו click או ( keyup)

פשוט זה אירוע שיצרנוב-child בילד וקראנו לו מה-parent מהאבא כדי לתקשר ולקבל את ה נתונים מה-child ל-parent ועשינו childValue זה מה שידפיס על המסך באבא יכלנו לעשות זאת בפונקציה חיצונית אבל עשינו כאן קיצור דרך ועשינוevent$ שזה בעצם דולר emit מה שקבלנו ה-value שקבלנו ב-event

(childChanged)=" childValue=$event

מקביל למה שכתבנוב-ts בילד

onChange(value:string)

{

this.childChanged.emit(value);

}

עשינו תקשורת בין קומפוננטות מלמעלה למטה ומלמטה למעלה.

<https://www.youtube.com/watch?v=MDe6LtX2IBo&list=PLeZsZBkrVMP-eYdS4rNJBe44rXAA5YSO4&index=12>

הסבר סרטון:

Routing

למעשה כשיש לנו אפליקציה אחת אנחנו רוצים לפעמים שיהיה לנו עוד נתיבים למשל נתיב שלmembers , נתיב של products ונתיב של dicshionary כל מיני נתיבים שהם צריכים להיות במסגרת האתר אבל הם לא צריכים לצאת מהאתר כל הנתיבים צריכים להיות נגישים.

איך יוצרים את ה- Routing?

אוקי ניצור שלושה קומפוננטות על ידי הפקודה במסגרת למטה בשמות: products, members dicshionary

וברגע שיצרנו את הקומפוננטות הוא יצר לי גם שלושה ספריות וקומפוננטות שמכילות את קבצים:ts,html,css,spec.ts

עכשיו נרצה לייבא את השלוש קומפוננטות ב-app.modules.ts שזה למעשה כל היבואיים שיש לנו בפרויקט שמתרכזים בקובץ הזה:

import{ ProductsComponent} from './products/products.component';

@Ngmodule({

Declarations:[

AppComponenet,-זה הקומפוננטה הראשית אב.

HelloComponent,

ProductsComponent,

אלה הקומפוננטות החדשות שייבאנו ועשינו להן import למעלה לכל אחת מהן בספריה המתאימה( למעלה יצרנו לדוגמא רק import ל ProductsComponent

MembersComponent,

DicitionaryComponent

],

עכשיו למעשה אנו רוצים ליצור את ה-routing על מנת לייצר את ה-roting אנו צריכים לייבא עוד משהו ונרשום זאת לפני הקומפוננטות בimport למעלה:

import{ RouterModule} from '@angular/router';-זה הכרחי עם רוצים לעבוד עם ה- routing

לשלב הבא נוסיף עוד כמה דברים לimport

@Ngmodule({

Declarations:[

AppComponenet,

HelloComponent,

ProductsComponent,

MembersComponent,

DicitionaryComponent

],

Imports:[

…….

…….

RouterModule.forRoot([-יש לה מתודה forRoot שהיא הכי עמוקה, שורש

כל אחד מה-routing הוא למעשה אובייקט

([

],