**דברים חשובים שלמדתי בקורס שחשבתי שכדאי שאסכם**

**קומפילציה והרצה של אנגולר**

* Webpack : הוא כלי ש angular CLI משתמש בו.

הוא כלי אשר עושה build לקוד בצורה אוטומטית – build automation tool.

כשאנו עושים ng serve הwebpack מתחיל בקימפול שלו.

* + התהליך שלו : הוא לוקח את כל קבצי הjs שיש בתוכנה שלנו , ושם אותם ב-5 bundleים שונים ועושה להם minimize בשביל בצועים.

1. Pollyfills.bundle.js : מכיל קבצי js שמשלימים את הפער בין התמיכה שיש לדפדפן בגרסאת js מסוימת לבין מה שאנגולר צריך (במידה והדפדפן לא תומך בגרסא מספיק מתקדמת.)
2. **Main.bundle.js :** שהוא מכיל את כל קבצי הjs שהם הsource code של האפליקציה שלנו (כל הקומפוננטות , מודלים **, ואפילו כל הtemplates**.) מכניס אותם לבאנדל ועושה מיני פיקציה.
3. Styles.bundle.js – כל קבצי הcss
4. vendor – מכיל את כל הקבצים של ספריות צד שלישי.
   * עוד משו מגניב בwebpack זה שברגע שנשנה את אחד הקבצים הוא אוטומטית יקמפל מחדש וישנה את הbundleים בהתאם. ויעשה refresh אוטומטי לדפדפן.

* כאשר אנו נכנסים לאפליקציית הangular שלנו אנו ניגשים לעמוד index.html.
* **בזמן הרצה** הweb pack מכניס הפניה לbundleים שהוא יוצר לקובץ **index.html** ולכן בזכות זה – הselectorים של הcomponents שלנו מובנים ובכללי כל הframework של אנגולר עובד.
  + **לדוגמא אם נעשה view page source על הדפדפן אנו נראה את הindex html עם הפניות לbundleים של web pack.**

**Decorators**

* בתוך component יש decorator (@component()) שהוא הופך את הts class ל-component.
* הdecorator הזה מקבל אובייקט ובו מוגדר meta data לגבי אותו component:
  + Selector = זה השם של הcustom html tag שהקומפוננט תשתמש בו.
  + Template = הhtml שהcomponent ירנדר.
  + ועוד..
* בקובץ Index html ניתן לראות custom html tag בשם app-root
* זה למעשה הselector של ה-app. Component
* ואז ב-app component אנו נכניס selectorים של componentים שאנחנו יוצרים.
* Injectable decorator אומר שלקלאס הספציפי עליו הוא חל – ניתן להזריק תלויות.
* הcomponent decorator יש מימוש של inject able decorator בפנים לכן אין צורך להשתמש בdecorator הזV בcomponent למרות שכן מזריקים לו תלויות.
* לכן יש שימוש בinjectable decorator רק בserviceים (שיש בהם צורך להכניס תלויות )

Services,directives – בauthors list – וauthers.Service

וך

**Displaying and handling events**

**Property binding**

* תזכורת קלה – html זה שפת Markup בניגוד לDOM שזה אובייקטים ששמורים בזיכרון שמייצגים את הhtml .
  + בhtml לטקסט בתוף הtag קוראים attribute וב-DOM זה property.
* String interpolation מאחורי הקלעים עושה property bindning
* נשתמש בstring interpolation (השימוש ב –{{ field }}) בtagים שקשורים לטקסט.
* אך אם נרצה להביא מידע מהקלאס לattributes כמו נגיד <img src=>
  + נשתמש בproperty binding. כך : <img [src]="field\_name">
  + פשוט מוסיפים סוגריים מרובעים.
* יש לזכור ש property binding זה רק one way data binding מהcomponent class ל-**DOM ולא ל-HTML** ולא הפוך
* יש כמה attrubites של html שאין properties מקבילים אליהם ב-DOM (וגם הפוך)

מה שיכול ליצור בעיה בשימוש בproperty binding כי הוא מקשר ל-DOM.

* אז מה שעושים במקרה כזה זה : < <td [attr.colspan]="field\_name"

ככה אנחנו עושים property binding ל-html attribute ולא לDOM propery.

**הוספת bootstrap לפרויקט**

* עושים Npm install bootstrap –save
* הולכים ל-styles.css וכותבים : Import "~bootstrap/dist/css/bootstrap.css"
  + ה~ = מעביר אותנו לnode\_modules במקרה הזה.

**Class binding**

* במידה ונרצה לתת קלאס מסוים במקרה זה Pull right בהתאם לערך שיש לנו בקומפוננט נוכל להשתמש בclass binding כך :
  + [class.pull-right]="isPulled"
* Style binding אותו סגנון בדיוק – שים לב שניתן גם להשתמש בתנאיים נגיד :
  + (isPulled) ? blue : red style.backgroundColor]=]

Event bindning

* פשוט משתמשים ב – (click=method()).
* אם רוצים להעביר את הevent object : (click=method($event))
  + שהוא מכיל כל מיני מאפיינים על הevent כמו איפה העכבר היה וכאלה.
* Event bubling = אם יהיה לי div ובתוכו כפתור , ובשניהם יהיה event handler (מחכים לevent)
* גם אם אני אלחץ רק על ה-button ה-div גם יקפיץ event וזה לא להיט , בשביל לעצור את זה אפשר לקרוא לפונקציה – stoppropgatation() משו כזה על ה$event בפונקציה של הbutton click

Event filtering

* בעזרת אנגולר ניתן לעשות לדוגמא : <input (onkeyup.enter)="onEnter()"
  + ואז תרוץ המטודה רק כשיילחצו אנטר אחרי keyup

Template variables

* בשביל להעביר קלט מהUi לclass אפשר להשתמש בtemplate variables כך :

<input **#email** (onkeyup.enter)="onEnter(**email.value**)"

וזה יעביר לנו את הערך.

* אבל תכלס תמיד עדיף להשתמש בtwo way binding עם [(ngmodel)] , לא לשכוח לעשות Import בapp.module ל – formModule.

Pipe ,two-way binding וcustom pipes : ניתן לראות : title-casing. Component

**Building reusable components**

**Input properties to components**

* כמו שיש properties ל-dom של html tags , אם אנחנו יוצרים component ואנו רוצים שנוכל להשתמש בה מספר פ]עמים במקומות שונים בקוד שלנו כדי שגם לנו יהיה properties שנוכל לתת להם ערך מבחוץ.
* כמו לדוגמא : <favourites [isFavaourite]='true' – ככה הכוכב איך שנטען אותו יהיה מלא ,
* ואם נרצה לטעון את הקומפוננט הזאת שוב ממקום אחר בקוד כשהוא מתחיל בתור כוכב ריק גם נוכל לעשות זאת.
* הInput גורם לcomponents שלנו להיות יותר reusable.
* בנוסף כדאי שניתן alias לinput properties שלנו כדי שלא יהיו באותו השם של הfield ואז במידה ונשנה את שם הfield הקוד יישבר..

**Output properties**

* שזה תכלס האפשרות להקפיץ eventים לcomponent שאתה יצרת.
* סתכל על favourite component , וapp-component,html – ותראה את הevent שקורא כל פעם שמשתנה כוכב. (כותב לconsole).
* \*שים לב לעשות Importים מהמקומות הנכונים - הauto import לפעמים עושה אימפורטים הזויים ששוברים תקומפילציה.
* שים לב שניתן גם להעביר eventdata במטוד - סתכל על isfavourite component ועל appcomponent.ts + html
* וגם פה ניתן להשתמש בaliasים כמו ב- input properties
* שים לב שבמידה ויוצרים directive ונותנים לו output , הevent שנוצר יכול להיות ממומש על כל tag שמשתמש באותו ה-directive.

**View encapsulation**

* שים לב שכל עיצוב שנכניס לקובץ css של component מסוים יעבוד רק על אותו component , סתכל על favourite component איך הוא צובע בצהוב את ה glyphicon class אבל המעטפה לא נצבעת.
* זה עובד בעזרת אימולציה של אנגולר למימוש של shadow dom (כי לא רוב הדפדפנים תומכים בshadow dom)
  + זה מוגדר באופן דיפולטי בmetadata של הcomponent תחת הproperty : encapsulation : emulated
  + הוא עושה את האימולציה הזאת בעזרת הגדרת attribute יעודי לכל component

**Ng-content**

* מאפשר לי להזריק markup לקומפננטה שיצרתי מחוץ לקומפוננט בעזרת class names שאני מגדיר לדגומא סתכל על component – panel ועל – app.component.html

**Ng-container**

* במידה ואנחנו רוצים להכניס רק טקסט לng-content שלנו , במקום שנשתמש באיזה div פשוט נשתמש ב-ngcontainer ראה app-component.html , וסתכל על הheading .
* אם תעשה לזה Inspect תראה שהוא מקבל רק את הטקסט , וככה הקוד פחות noisy..

**סיכום על הפרק building reusable components ב-like component וapp.**

**ללמוד על change detection mechanisimשרץ ברקע וגורם לtemplates להיות בסנכרון עם הקבצי ts**

**Directives**

* Directiveים אחראים על שינוי הDOM.
  + Attributes directives : משנים properties ב-DOM כמו ngclass , ngstyle כל אלה .
  + Structure directives : אחראים על יצירה \ הסתרה של **אובייקטים בDOM ולא הproperties שלהם , הסוג הזה של directive יתחיל עם \* , לדוגמא :\*ngfor \ \*ngif**

**\*ng if**

* תסתכל על הcourses component – תראה שימוש בngif וng-template.
* שים לב שבמידה וקיימים קורסים יוצר הtemplate עם הtemplate variable – courseList ואם אין קורסים יוצג הtemplate עם הvariable noCourses.
* וכל התנאי הזה ייקרא רק במידה וה-viewmode יהיה courses בהתאם לswitchcase (שהוא משתנה בevent של לחיצה.)
* במילים אחרות במידה והngiif הוא truthy אז הוא יציג את הtemplate במידה והוא falsy הוא לא יציג.
* Turthy ו-falsey זה הערך של הדיקה בוליאנית לדוגמא : if(true) זה truthy בניגוד ל-if(false) שזה falsy.
* **שים לב שה ngif מתייחסים ל –variables ללא ה# בניגוד להצהרה שלהם.**
* שים לב שניתן לעשות property binding על hidden attribute ולהשווות אותו לביטוי בלויאני ואז יהיה כמעט אותו אפקט כמו ngif.
  + ההבדל בין השניים הוא שאם נשתמש בhidden attribute האלמנט שמוסתר עדיין יהיה ב-DOM בניגוד לngif שבו האלמנט לא יעלה בכלל ל-DOM , ולכן באפליקציות שיש בהם DOM גדול , עדיף להשתמש בngif בשביל ביצועים טובים יותר.
* שים לב שחובה להשתמש ב - \* כאשר נשתמש בngif – מה שהכוכבית עושה זה - היא יוצרת ng-templateים של הtagים אשר תלויים בתנאי.

ומוסיפה ng-ifים תואמים לתנאי ההצגה שלהם.

**Ngswitch**

* דוגמא ביחד עם ngif בcourse component

**ngFor**

* יש משתנים שניתן להשתמש בהם ב-ngfor כמו לדוגמא Index ו- even שיודעים להגיד לנו בהתאם לערכים שאנחנו עושים להם iteration – ראה table.component

**Change detection**

* כל פעם שיהיה בקשת ajax \ event \ timer ירוץ מאחורי הקלעים change detection אשר יזהה אם יש חוסר סנכרון בין הערכים בtemplate לבין הערכים בcomponent.ts ויעדכן את השינו – לדוגמא מחיקת שורה ב - table compenent

**ngFor and track by**

* נגיד ואנחנו נקבל את הטבלה שלנו משרת מסוים לפי לחיצת כפתורכמו שעשיתי בtable component.
* במידה ואני לא משתמש בtrack by method ליד ה-ngfor הDom objects שהngfor מייצר יווצר כל פעם מחדש כשאני יילחץ על הכפתור – למרות שהערכים תמיד יהיו זהים (ניתן לבדוק זאת לפי זה שב page source בדפדפן יש צבע סגול על הערכים שנוצרים כל פעם שנלחץ כפתור).
  + **הסיבה שהוא נוצר כל פעם מחדש היא שהngfor** מזהה את האובייקטים לפי ה-identity שלהם – שזה למעשה ה-כתובת שלהם בזכרון.
  + ובגלל שכל פעם שאנחנו לוחצים על הכפתור אנחנו יוצרים את המערך מחדש – אז הכתובת של המערך בזכרון כל פעם תיהיה שונה ולכן הngfor לא מודע בכלל שתוכן האלמנטים זהה והוא יוצר אותם מחדש כל פעם.
* לכן ניתן להשתמש ב-trackby לפי ערך של הdata.name במקרה של ה-component שלי , ואז הוא לא ירנדר את האלמנט מחדש כי הוא ייראה שהוא כבר קיים.
* **שים לב לא להשתמש בזה סתם !! רק אם באמת יש בעיות ביצועים באתר שלנו נשתמש בtrackby כי זה סתם מבלגן תקוד - preoptimization is the roots of all evuil**

**Ngclass**

* הוא attribute directive – תסתכל על favourite component על ה-heart.
* תשווה את זה לclass binding שיש לכוכב.
* זה יותר נוח
* הngclass מקבל אובייקט שבו יש key value pairs.
  + הkey הוא שם של css class – **שים לב לשים גרשיים סביבו** והvalue הוא ערך בוליאני שלפיו לelement יהיה באותו ה-class.

**Ngstyle**

* בדיוק אבל בדיוק אותו הגיון כמו ngclass רק ל-styleים.

**Safe traversal Operator – או במילים אחרות סימן שאלה(null able operator) בstring interpolation**

* תסתכל על הקומפוננט authors list
* במידה ותלך לauthor.service ותחזיר במערך Null (יש שם comment)
* **תופיע שגיאה בדפדפן כי הוא לא יכול להציג author.name כאשר author הוא Null**
* **אך אם נוסיף ? ליד author כמו שעשיתי בauthor list.html לא תיזרק שגיאה ופשוט יופיע לנו Null על הדפדפן.**
* **אנו נשתמש בזה כי לפעמים יש מצבים שאנחנו מביאים מידע מהשרת ולאיזה מאית שנייה האובייקטים המורכבים שאנחנו מייבאים הם Null – ולכן פשוט נוסיף את ה- Safe traversal Operator כדי למנוע זריקת שגיאה בruntime ואז כבר האובייקט לא יהיה null.**

**Custom directives**

* תסתכל על input-format.directive.
  + **Hostlistner – מאפשר לי לעשות event handeling על הdom object עליו חל הdirective.**
  + **ElementRef הוא service שנותן לי את גישה ל-Dom Object.**
  + **בשביל לתת אפשרות להכניס Input ל – directive פשוט מכניסים בInput alias את שם הdirective.**

**תרגיל מסכם directives ב-zippy component – (תסתכל גם על app.html)**

**Template Driven Forms**

* FormControl class
  + ניתן להצמיד לכל Input field בformים שלנו Inststance של –FormControl.
  + וכך בעזרת הInstance הזה נוכל לבדוק מה הערך בנמצא בInput field , האם עשו focus על הInput field , האם הערך שונה בעבר ועוד מלא אפשרויות validation.
* FormGroup class
  + מייצגת קבוצה של -control ים ב-form.
  + יש לאובייקט הזה את אותם הproperties (dirty, ועד מלא אפשרויות validation).
  + הformgroup יהיה valid רק אם כל הcontrollים שלו יהיו valid..
* יצירת controls
  + ישנם שני דרכים ליצור את אובייקטי הcontrol האלו.
    - Directives – template driven forms
      * טוב ל- formים פשוטים
      * Validation פשוט.
      * יותר קל לכתוב אותם.
    - Code - reactive forms
      * יתרון – יותר שליטה על לוגיקה של validation
      * אפשר לעשות על זה testים.
* יצירת template driven forms
  + נשתמש בdirective ngModel ללא two way binding syntax
    - והוא ייצור מאחורי הקלעים control object בשביל אותו Input
  + נצטרך לתת שם ל-input field כדי שהקונטרול יידע לזהות את אותו ה-input field.
  + לדגומא תסתכל על inputform.html
  + ה-ngModel משתמש הproperties של form control ומעתיק אותם אליו לצורך גישה מהירה יותר לproperties שלו – זה למה אני לא עושה firstname.control.valid אלא ישר ניגש למאפיין.
* יצירת validations
  + תסתכל על input forms.
  + שים לב שיש צורך בשימוש של template variables בשביל לשמור את האובייקט של הngModel למשתנה שנוכל לגשת אליו בmarkup הבא.
* שימוש בכמה validations והצגת שגיאת validation בהתאם לבעיה.
  + תסתכל על input forms.
  + שימוש ב-error object.
  + שים לב שבעת שינוי לInput field עשיתי שהוא גם ידפיס את הfirstNAme כדי שאוכל לראות את שמות ה-properties של אובייקטי השגיאות.
  + שים לב שיש מאפיין נוסף לכל שגיאה כמו לדוגמא ב-minLength – requiredLength
  + שימוש במאפיינים האלה מונעים מאיתנו לעשות hardcode ב-html שעלול להשתנות בעתיד.
* Styling invalid input fields
  + תסתכל אם אתה עושה Inspect על הinput field בזמן שהוא Invalid - אנגולר מוסיף class - ng-invalid , וגם לעוד כל מיני דברים כמו ng-touched (אם שיניתי שם משו) ועוד..
  + ניתן להיעזר בהוספת class הזה לעיצוב הInput fields לדגומא תסתכל על styles.css וה- input-form,
* Cleaner templates – שים לב שהtemplates שלך לא דורשים גלגול ימינה בגלל כל הattributes – לכן עדיף להפריד בשורות למרות שזה אותו אלמנט.
* ngForm
  + אם נשים את הinputים שלנו תחת form – בצורה אוטומטית מאחורי הקלעים לאותו ה-form יהיה directive בשם ngForm – שהוא מייצג אובייקט של formGroup (כמו שמוסבר למעלה)
  + שים לב של-directive הזה מוגדר גם output (שזה בעצם אומר שכל מי שיש לו את הדיירקטיב הזה יש event שתואם את הדיירקטיב) לדוגמא במקרה זה הoutput הוא ngSubmit – אז תכלס זה פשוט event שכל tag שמשתמש בdirective ng model יכול לממש.
    - הסבר חוזר על outputs אם לא הבנתי למעלה ויצירת directive לדגומא :favourite component.
  + ניתן לראות אותו בInput-form.component.html אחרי שנלחץ על submit
* ngModelGroup directive
  + הcontrol instaceים שאני אכניס לתוך div שמשתמש בdirective ngModelGroup ייגרום לכך שב-value property שב- FormGroupיהיה אובייקט JSON שמכיל את הערכים של הinputים בפנים.
  + לדגומא תסתכל על – input-form.component : תלחץ submit ותסתכל על ה-value של האובקייט ngForm – ותראה שם אובייקט בשם contact (כי זה השם שהבאתי לngModelGroup directive) והוא יכיל את הvalues של הInputים.
  + אפשר (לא חובה בכלל) נשתמש בזה אם נגיד בform מסוים יש לנו קבוצות של Inputים שקשורים אחד לשני , נגיד mail ופלאפון ושם אפשר ליצור לו modelGroup בשם contact info.
  + זה יכול לתרום במידה ואנו רוצים לשלוח בpost לאיזה webapi jsonים ולעשות עליהם איזה logic בצד השרת.
* תכלס ההבדל בין ngModel ל-ngform זה שלngform יש את ה-ngSubmit event.
* שים לב-disable שלה-submit button.
* Validating check boxes

שים לב שזה עובד אותו דבר בcheckbox

* + ותראה טריק נחמד ה-p בסוף תמיד יראה לי את ה-values שיש ב-ngForm
  + **זה בעזרת הpipe json.**
* Drop down lists
  + שים לב שגם פה זה עובד תכלס אותו הדבר..
* ngValue
  + במידה והייתי שם attribute של ngValue על ה-option ככה: ngValue] = method]
  + הערך היה כל האובייקט ולא רק id או string..
* סיכום לפרק בcreateCourse component – שים לב לngsubmit שאני מדפיס אובייקט שמכיל את כל הערכים.

**Router and Navigation**

* כדי להשתמש ב-Routing חובה לעשות Import ל -router Module
  + בmodule הזה יש directiveים ו-serviceים שנוכל להשתמש בהם כדי לממש navigation באתר שלנו.
* ישנם מספר שלבים ביצירת routeים

1. לקנפג את הrouteים **– המיפוי בין נתיב לבין -component**
2. להוסיף router outlet **–הוספה של איפה יעלה הcomponent ב-markup**
3. להוסיף לינקים **– לחיצה על כפתורים שתוביל ל-path מסוים שיוביל ל-component.**

* בשביל לקנפג routeים ניתן לראות ב-פרוייקט hello-world (נמצא תחת תקיית downloads או dev\_project) תכנס ל-app.module
  + שים לב שעשיתי import ל –routerModule ובdecorator של ה-appModule – בimports הוספתי routerModule.forRoot() שם מכניסים מערך שמקבל אובייקטים (שלכל אחד מהם יש נתיב וcomponent) – שם מתבצע המיפוי בין הpath ל-component
  + שים לב שלסדר של האובייקטים במערך יש חשיבות רבה – לדוגמא אם הייתי שם את /followers ראשון , בחיים לא הייתי יכול להגיע ל-component של profile.
* לאחר מכן חייבים להוסיף את ה-router outlet – ניתן לראות אצלי ב-app.html.
* שים לב שהcomponent שתואמת ל-route תכנס tag אחד אחרי הrouter outlet ולא בתוכו

(יכול לבדוק עם inspect).

* **Router links**
* ב-routerModule קיים directive בשם – routerlink.
  + ולו ניתן ערך של הpathים שיצרנו להם מיפוי ל-component.
  + **לא לתת נתיב עם href!**
* הסיבה שלא ניתן נתיב עם href אלא אם routerLink היא :
  + אם נשתמש ב-href כל פעם שניגש לנתיב כל האפליקציה שלנו תרד מחדש , אפשר לראות זאת ב-network tab אם נעשה פילטר ל-js scripts – נראה שכל פעם שאנחנו נלחץ על לינק מסוים (שמשתמש בhref) יורדים לנו הסקריפטים הדיפולטים של אנגולר שהwebpack יוצר באופן אוטומטי – (שהם מייצגים את כל הקוד של האפליקציה שלנו.
* בניגוד לאם נשתמש ב-routerLink **ככה ירד לנו רק התוכן של אותה ה-component.**
* **דוגמא פשוט לשימוש בrouter link - navbar.component.html**
* שים לב שיש פעמים בהם נרצה להוביל לנתיב דינאמי – **כמו לדוגמא ב-followers.component**
  + **שים לב שאני עושה property binding על ה-routerLink ואני נותן לו מערך שהתא הראשון שלו זה הנתיב.**
  + **התא השני של המערך של הפארמטר שיבוא בנוסף לנתיב**
  + **במקרה הזה יש אובייקט follower לאותה component (הוא לא נמצא כfield בTS כנראה שזה משו שקשור לservice שמביא אותו בצורה מסוימת שאני לא מכיר.)**
* **routerLinkActive directive**
  + directive שמוגדר בrouterModule.
  + תסתכל על navbar.component.html
  + מה שקורה זה שאנחנו נותנים לdirective הזה classes של css (במקרה הזה bootstrap) במידה והroute שכרגע פעיל זה זה , אז הוא יגדיר את ה-classes האלה על אותו li.
* **Getting the Route parameters**
  + תסתכל על github.profile.component
  + שים לב שעושה Import ל-activated route
  + יש אובייקט שנקרא paramMap שהוא observable של route parameters
  + לאובייקט param יש key property שם נמצא השם של הנתיב
  + ו-עוד מאפיין בשם param שבו יש את הid וה-value שלו.
  + ניתן להשתמש בפונקציה get כדי לקבל את הערך  **(נותנים לו את שם הparameter).**
* **מה זה observable**
  + בפשוטת זהו stream של 0 או יותר eventים.
  + בעזרת הפונקציה subscribe נעשה callback ברגע שevent מסוים יעשה emit.
  + לדגומא אם יש לי כפתור שאני עוטף אותו ב-observable **כל פעם שאלחץ על הכפתור ייכנס event ל-stream ובעזרת ה-subscribe method אני יוכל לגשת למידע שהevent מקפיץ.**
* **כעת נסביר על Lifecycle של component**
  + איך ש-component נטען יש את הon init
  + וכשעוברים לcomponent אחר / יוצאים מהאתר יש on destroy
  + ניתן לכתוב קוד שיקרה בדיוק בזמן האלה בעזרת מימוש הinterfaceים onInit ו-onDestroy

**Why route parameters are observable ?**

* עכשיו השאלה היא למה ה-paramMap מחזיר את הפארמטרים של הcomponent בObservable ולא בצורה רגילה ?
* בוא נראה אפשרות ובה הוא לא observable :
  + תסתכל על profile.Component.ts על שימוש בsnapshot property.
  + הוא אינו observable ואכן כאשר אני עובר מן הfollowers component ל –profile של אחד המשתמשים מודפס לי הid שלו – משמע הגישה ל-route parameters מוצלחת.
  + **למה זה עובד ?**
    - בגלל שכל פעם שאני עובר מה-followers ל-profile אני מגיע ל-ngOnInit ושם אני ניגש למאפיין של הsnapshot ומדפיס אותו.
  + **מתי זה לא עובד?**
    - **כאשר אני לוחץ על הכפתור next , אני בכלל לא ניגש ל – ngOnInit ולכן זה לא מדפיס**
  + **כאשר אני משתמש בobservable** 
    - **נכון , גם במקרה הזה הngOnInit לא מופעל , אבל בגלל שobservable מתפקד כ-stream של מידע ,שכל הזמן רץ ברקע וברגע שיוחלף הפרמטר (שזה תכלס הevent של אותו observable) ה-subscribe method יקלוט את זה וירוץ.**

**Routes with multiple parameters**

* תסתכל על זה שהוספתי עוד פארמטר של login ל-followers , פשוט מוסיפים /:param\_name וככה עושים עם כמה פארטמרטים.

**Query parameters**

* הם פרמטרים **אופציונאלים** שאנחנו מוסיפים אחרי ?
* לדוגמא אם אלך ל-facebook ואוסיף ? והרבה lachtongים זה עדיין יוביל אותי ל-facebook הרגיל.
* אבל עדיין אפשר להיעזר בparameterים האלה כי ניתן לגשת אליהם מה-.ts ולעשות איתם Logic מסוים.
* תסתכל בnvabar component.htmll איך אני מוסיף שני query parameters ל-followers - ככה שכל פעם שאני לוחץ על הfollowers ב-navbar זה מוביל אותי ל-followers עם פרמטרים אופציונאלים האלה.
* ותסתכל ב-followers.ts איך אני ניגש ומדפיס אותם – כמעט אותו דבר כמו פרמטרים רגילים. (גם פה יש אפשרות לעשות עם snapshot).

**Subscribing to multiple observables**

* כעת יש לנו subscribe לparamMap observable שבאמצעותו ניגש ל-id parameter.
  + **תכלס מה שבד"כ נעשה אחרי שנשיג את ה-id זה : נשתמש בservice שלנו כדי לגשת לweb api שייתן לנו profile עם id שתואם ל-required parameter שלנו.**
  + **מה שאומר שאנחנו נקרא לפונקציה של הservice בתוך הsubscribe שמקבל את ה-parameter.(אחרת לא נוכל לשלוף את הprofile בהתאם ל-id)**
* אבל אם יש לנו גם query parameter שאנחנו צריכים להתחשב בו(לדוגמא Page) כשאנו ניגשים ל-web api **איך נעשה זאת ?**
* **איפה נשים את הsubscribe של ה-service כך שהוא יוכל להשתמש בשני הפארמטרים ?**
* בשביל זה נאחד את הobserverables שלנו (paramMap ו -queryParamMap) ל- observable אחד.
  + ומשם נוכל לגשת לשני הפארמטרים שלנו ולקרוא לservice בפנים.
* אפשר לראות דוגמא ב-followers.ts אבל שים לב שאין לו באמת שימוש בid והservice לא באמת דורש את שני הפארמטרים – זה סתם בשביל לראות איך מחברים בין שני observerables.

**Programmatic navigation**

* תסתכל על github profile component .ts ו-html
* בליחצה על submit אני עובר עמוד – המימוש הוא בקובץ ts ולא בhtml כמו מקודם עם הdirective של ה-routerLink.
* וגם אני מוסיף quryPrameters.
* במידה והייתי רוצה להוסיף required parameters אז פשוט בתא הבא של המערך שבו אני מגדיר את ה-path.

**תרגול של הsection הזה ב-router-nav-proj תחת dev\_projects**