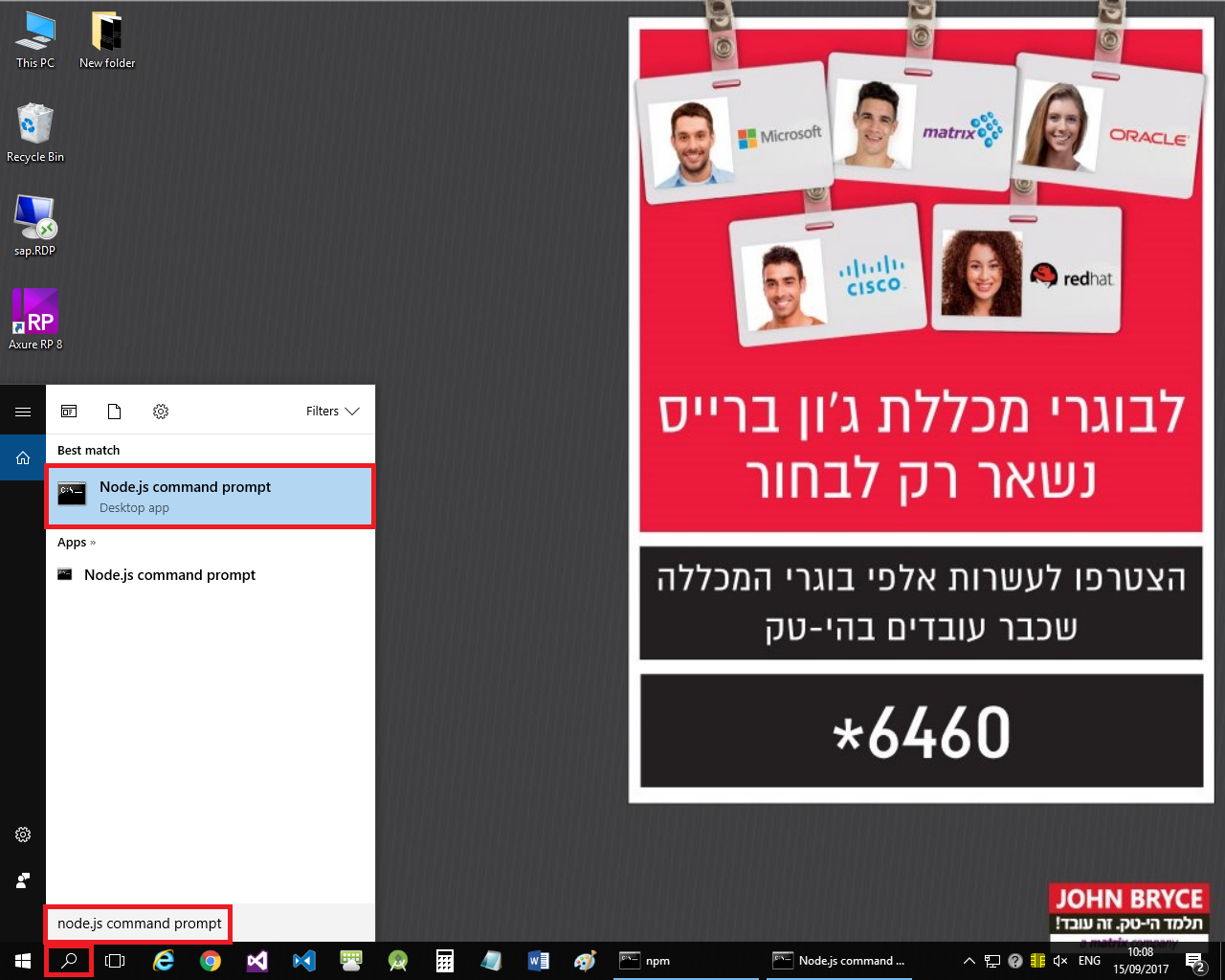
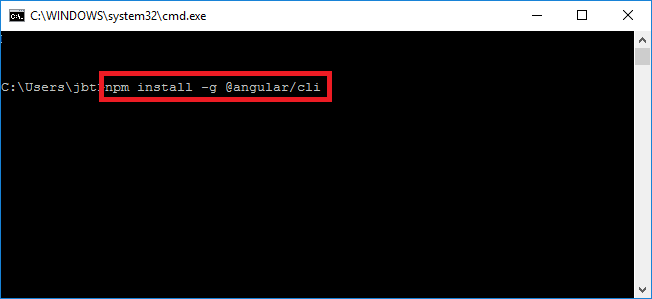
יצירת פרוייקט אנגולר בסיסי

**התקנת אנגולר גלובלית על המחשב ע"י שימוש ב- nodeJS cli**

**שלב 1** – נפתח את הcli



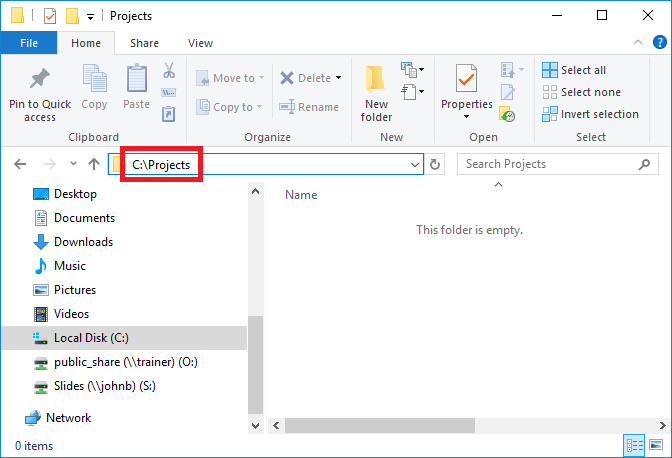
**שלב 2** – נוריד באמצעות הcli את האפשרות לעבוד עם angular, (פעולה זו צריכה להתבצע פעם אחת בלבד, ולאחריה המחשב שלנו כבר יכיר תמיד את ספריית אנגולר, ונוכל לבצע דרך הcli פקודות עם הקידומת ng שמסמלת פקודות הקשורות לangular)



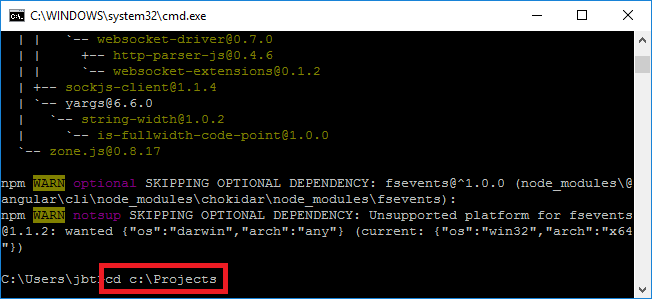
**יצירת פרוייקט אנגולר חדש**

יצירת פרוייקט חדש תתבצע באמצעות פקודת ng new בעזרת ה- CLI.

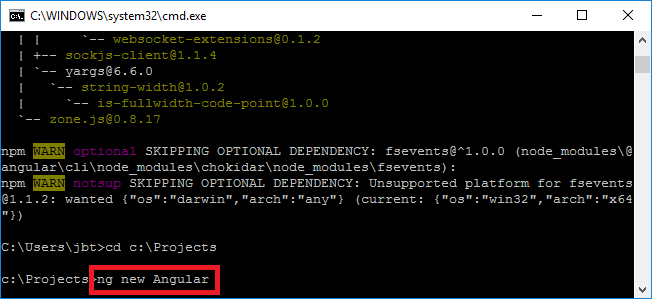
את הפרוייקט החדש שניצור, נרצה למקם בספריה הבאה:



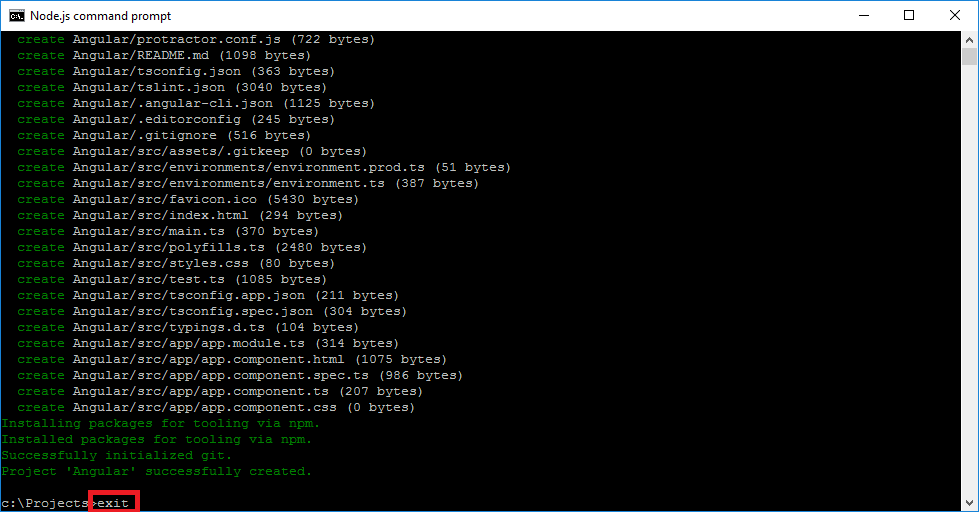
**שלב 1** – נעביר את מיקום הcli למקום בו אנו רוצים ליצור את הפרוייקט ע"י פקודת cd והמסלול היחסי אליו אנו רוצים לעבור



**שלב 2** – נשתמש בפקודה new שיוצרת פרוייקט אנגולר חדש, ולאחריה נרשום את שם הפרוייקט שנרצה לתת לפרוייקט שייווצר



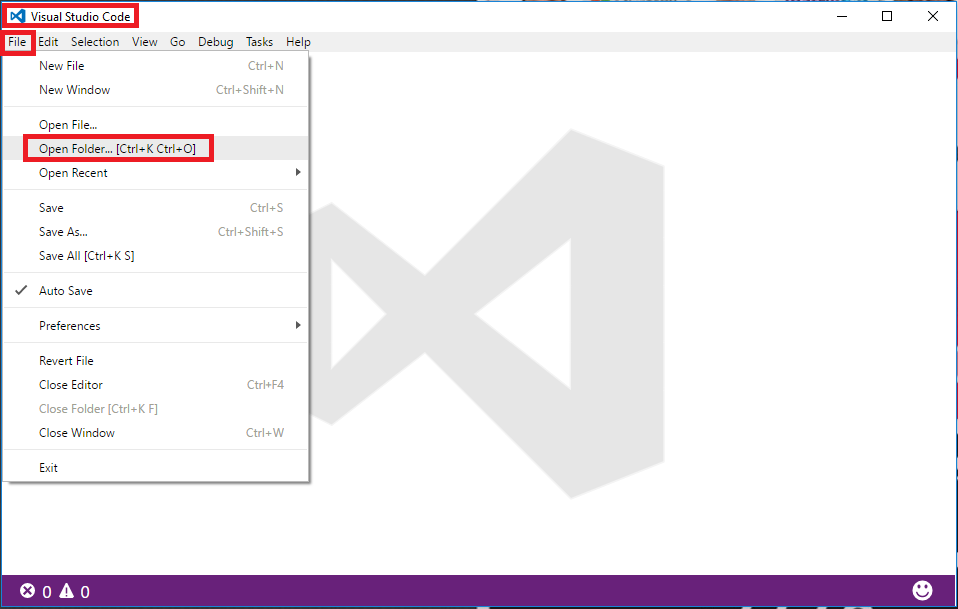
**שלב 3** – נלחץ enter ונחכה עד שהפרוייקט יבנה בשלמות (בניית הפרוייקט הסתיימה רק כאשר נקבל בcli את שורת הנתיב המסמלת שהפקודה הקודמת הסתיימה, ואפשר לעבור לפקודה הבאה)



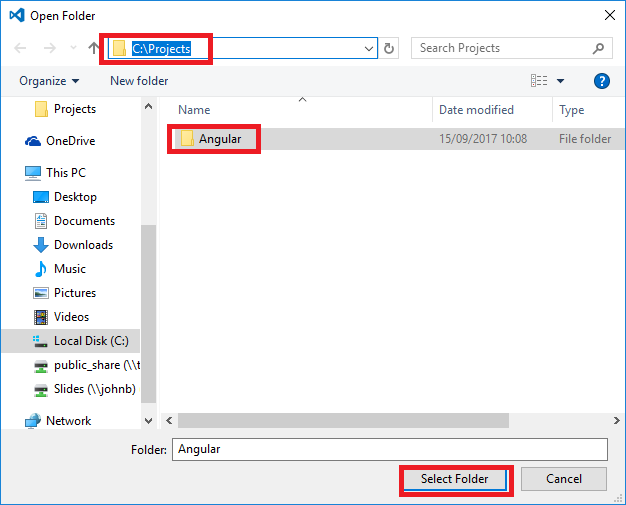
ה- CLI יוצר פרוייקט חדש עם קומפוננטת AppComponent כברירת מחדל ותמיכה בקבצי וספריות אנגולר.

**פתיחת הפרוייקט בvisual studio code**

**שלב 1** – ניכנס לfile ונבחר באופציה open folder



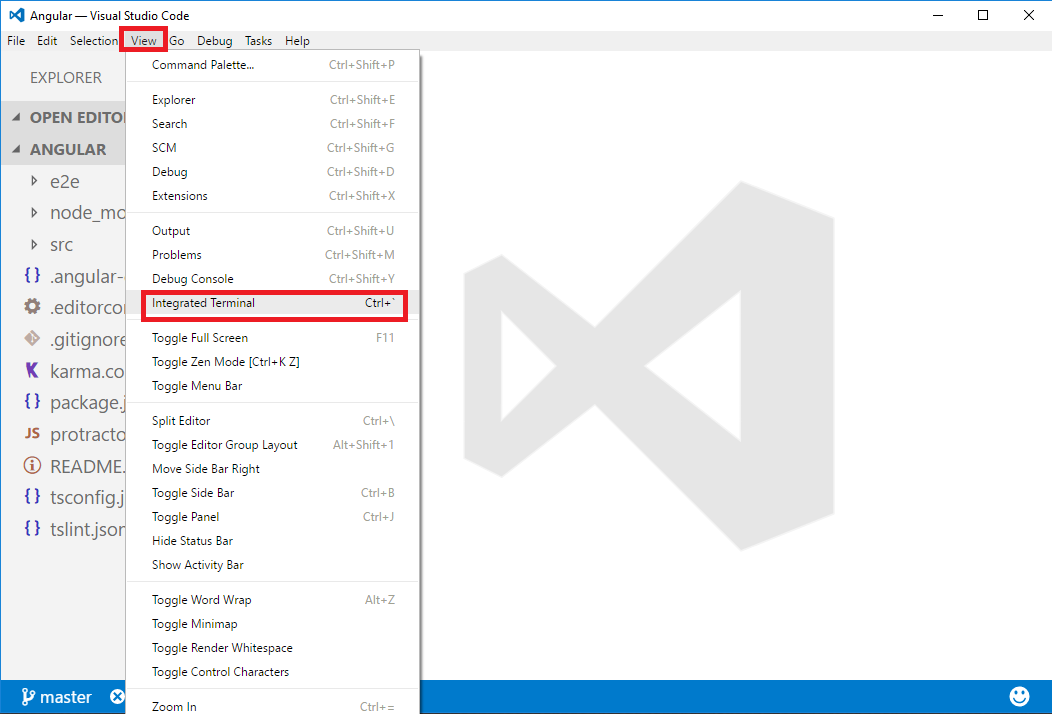
**שלב 2** – בחלון שנפתח, נבחר את תיקיית הפרוייקט שיצרנו עבור אנגולר, ונלחץ "select folder"



**הרצת הפרוייקט**

במהלך העבודה עם visual studio code, נוכל להריץ את כל פקודות הcli באחד משני הדרכים:

1. שימוש בnode js cli - ששימש אותנו להתקנת האנגולר בצורה גלובלית
2. שימוש בcli המובנה של visual studio code הניתן לגישה בצורה הבאה:

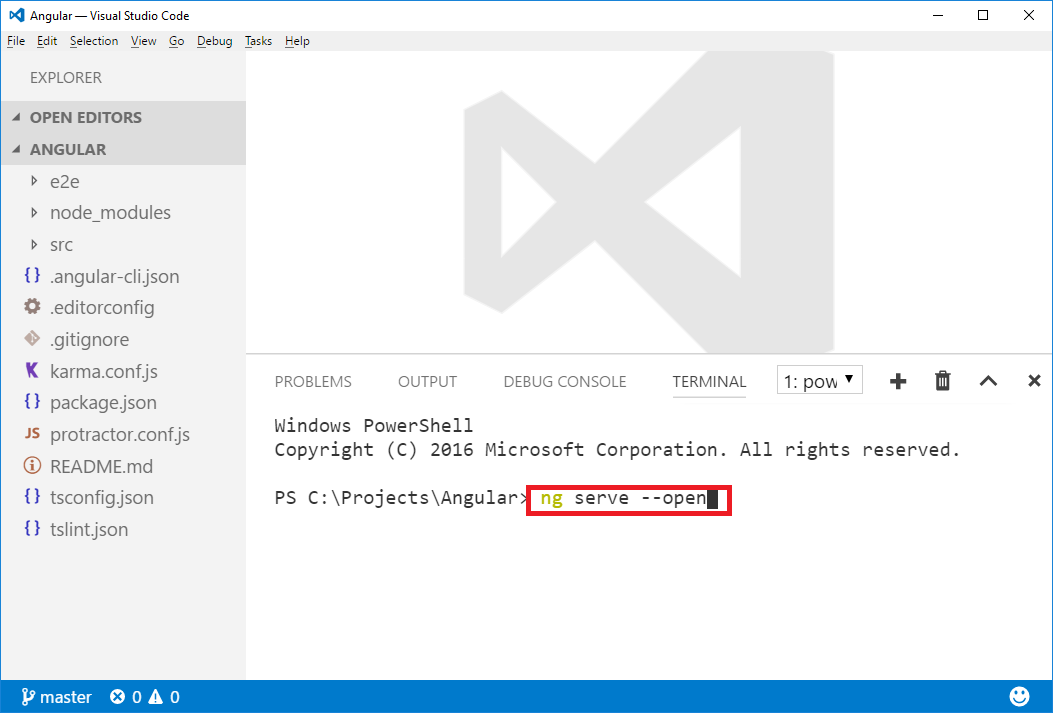


בשני הדרכים, עלינו לדאוג שהנתיב של הcli פונה אל התיקיה של פרוייקט האנגולר הנוכחי.

בכדי להריץ את הפרוייקט נשתמש בפקודהserve ng - המורה לשרת לבנות את האפליקציה, ע"י המרת קבצי המקור של typescript, לקבצים מתאימים של js.

ההוספה של --open נותנת הוראה לאפליקציה המורצת להיפתח בדפדפן

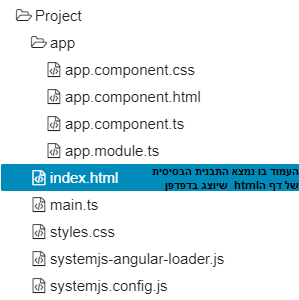
פרוייקט אנגולר שמורץ על ידי הcli ירוץ בצורה דיפולטיבית על הלינק: http: // localhost: 4200 /.



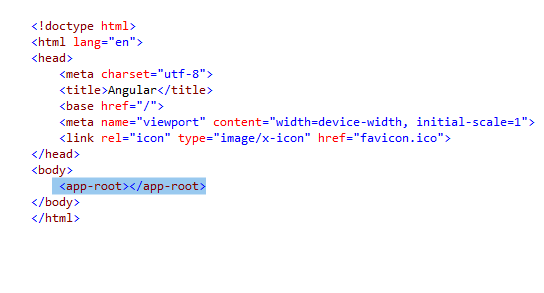
**יצירת פרוייקט בסיסי**

כאשר נפתח את התיקייה בvs code נראה שהיא מורכבת מעשרות קבצים ותיקיות.

נתרכז כעת בתיקיית src, בה נמצאים הקבצים החשובים לנו לצורך הבנת מבנה הפרוייקט:

****

נכנס לתוך הקובץ index.html ונראה כי תוכנו הוא:



מהי התגית <app-root></app-root> ?

זו אינה תגית של html5 , אלא מופע של קומפוננטת אנגולר.

בשלב זה נכיר את המושג **Component:**

**Component**

**רכיב נפרד. מכיל: קוד בשפת TypeScript, תבנית HTML, עיצוב CSS. מכיל גם selector שמאפשר לשתול את הרכיב בתוך רכיב אחר או בתוך קובץ ה-index.html.**

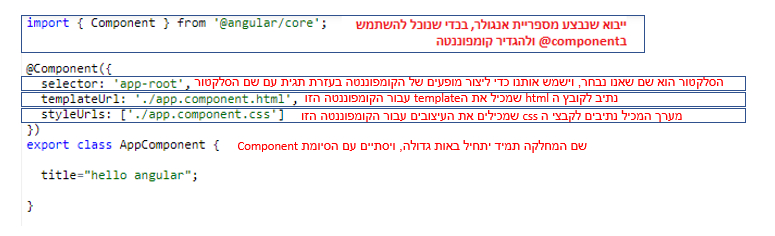
Components הם אבני הבניין הבסיסיים של יישומי Angular. הם מציגים נתונים על המסך, קולטים את ה input מהמשתמש, ומפעילים את הפונקציות המתאימות בעת הפעלת אירועים שונים (כגון לחיצה על כפתור).וניצור אותם תחת תיקיית app

בצורה דיפולטיבית, כאשר אנו יוצרים פרוייקט אנגולר פשוט, נוצרת לנו בתוך תיקיית app קומפוננטה בשם AppComponent המורכבת משלושה קבצים:

1. **app.component.ts**- קוד של מחלקה המייצג את ה- component, נכתב ב- TypScript.

2. **app.component.html** - תבנית הרכיב, שנכתבה ב- HTML, וכוללת תגיות שיוצגו בדפדפן.

3. **app.component.css**- סגנונות CSS הפרטיים של ה component.

בקובץ app.component.ts נראה את התוכן:

נשים לב, לשני תוספות חשובות, שיש להקפיד להוסיף ביצירת הclass עבור הcomponent:

* **@Component**

**פקודה מעל Class שמגדירה את ה-Class כ-Component.**

* **export**

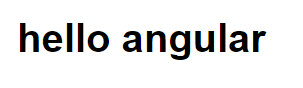
**פקודה הנכתבת לפני class ומאפשרת להשתמש ב-class בקבצי TypeScript אחרים.**

בקובץ app.component.html נראה את התוכן:



הסוגריים המסולסלים הכפולים הם תחביר אינטרפולציה של אנגולר.

באמצעות אינטרפולציה זו אנו מציגים את ערך המאפיין title של AppComponent בתוך תג h1 של HTML, והתוצאה שנראה בדפדפן היא:



**יצירת קומפוננטה חדשה:**

יצירת קומפוננטה חדשה מתבצעת באמצעות Angular CLI, בתחביר הבא:



(המילה cat היא שם הקומפוננטה שברצונינו ליצור, וכמובן שיש לשנות אותה בכל פעם שיוצרים קומפוננטה – לשם הרצוי של הקומפוננטה שתיווצר)

אפשר גם להשתמש בפקודה הבאה, שזהה לחלוטין לפקודה הקודמת, ורק מהווה קיצור בכתיבה:



**כדי להציג את CatComponent, עלינו לבצע שני שלבים:**

1. להוסיף אלמנט <app-cat> לקובץ app.component.html כדי ליצור מופע של הקומפוננטה.



1. להוסיף את הclass שיצרנו עבור הקומפוננטה ל- declarations (מערך של כל הרכיבים שמשוייכים למודול שאנו בונים ונכתב בתוך @NgModule - זה מה שקובע שרכיב משויך למודול).

