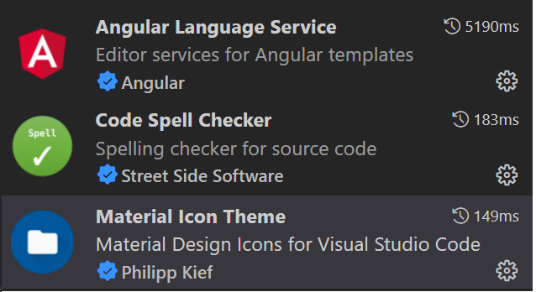
# **Angular**

שלושה תוספים חשובים ל-VS Code:



**Directory Structure**

סידור הקבצים והתיקיות:

src

app

components

employees-area

employee-list

.css, .html, .ts, .spec.ts

add-employee

.css, .html, .ts, .spec.ts

employees.module.ts

products-area

products.module.ts

models

services

...

**מוסכמת שמות של קבצים לפי נושאים שונים:**

name-of-file.type.extension

Examples:

cool.component.css

cool.component.html

cool.component.ts

very-cool.component.css

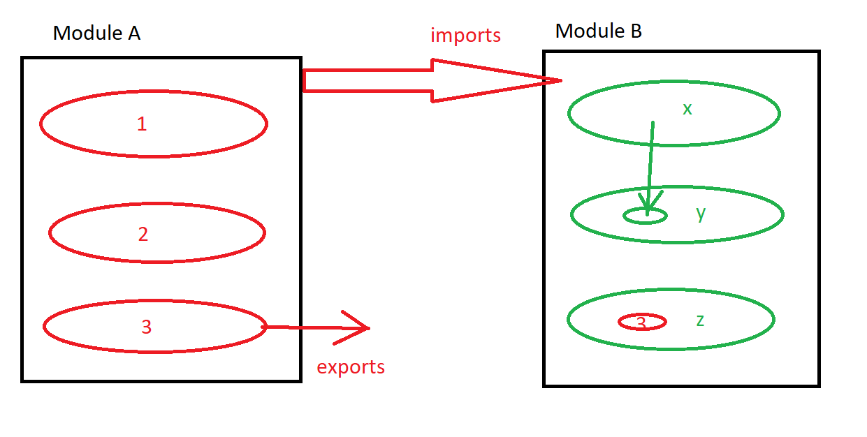
very-cool.component.html

very-cool.component.ts

gift-card.module.ts

**Module**

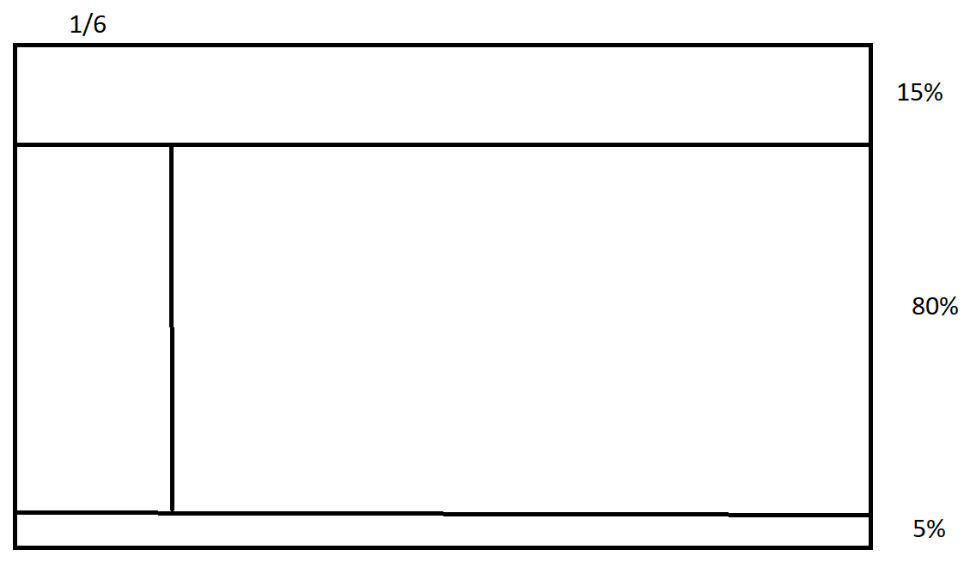
אזור נפרד בפרויקט המכיל Components, Directives, Pipes ועוד:



**Layout**

הסידור הכללי של האתר. נבנה ע"י קומפוננטה המגדירה סידור זה. לרוב זו הקומפוננטה הראשונה שמוצגת ב-index.html.

דוגמה:



**Routing**

מנגנון המאפשר להציג Component כתלות בכתובת ה-URL הנוכחית.

נבנה בתוך app-routing.module.ts

**Route Parameter**

אפשרות לשלוח מידע לכתובת ה-Route, לדוגמה קוד מוצר, סוג קטגוריה וכדומה.

**Service**

מחלקה לביצוע פעילות לוגית. נהוג להגדיר בתיקייה בשם services בתוך app.

**DI**

Dependency Injection – זהו Design Pattern לפיו ה-Framework (אנגולר) שולחת לקומפוננטה שלנו אובייקט Service הדרוש לפעילות שלה. אנו מבקשים את האובייקט ב-constructor ואנגולר שיוצרת אובייקט מה-Component – שולחת לנו את אובייקט ה-Service שביקשנו. אם אנו בונים את מחלקת ה-Service יש לנו צורך לומר לאנגולר היכן ליצור את האובייקט הזה (באיזה Scope). ישנם חמישה Scopes:

1. רמת הקומפוננטה
2. רמת מודול
3. רמת כל האתר
4. רמת האתר + רמה נפרדת לכל Lazy Loading Module
5. רמת Workspace (מספר פרויקטים הנמצאים בתוך סביבה אחת)

**ספרייה המכילה REST API מקומי של המידע האמיתי של חברת Northwind:**

npm i -g northwind-back-end

**Model**

מחלקת מידע טהור.

מחלקה המייצגת מידע שלרוב מגיע מהשרת, מכיל רק נתונים, ללא פונקציות.

**Promise**

אובייקט המאפשר להריץ בתוכו קוד אסינכרוני ולדווח הצלחה אחת או כישלון אחד.

**Observable**

אובייקט המאפשר להריץ בתוכו קוד אסינכרוני ולדווח הצלחות לאורך זמן, או כישלון אחד.