1. מה זה TypeScript ולמה להשתמש בו?
2. התקנה והגדרת סביבת עבודה
3. הגדרת טיפוסים בסיסיים
4. מחלקות ואובייקטים
5. Interfaces ו-Generics-
6. מודולים וייבוא קבצים
7. קומפילציה והרצה
8. תרגיל כיתתי- ניהול מסעדה

**חלק 1: מה זה TypeScript ולמה להשתמש בו?**

**TypeScript** הוא שפת **superset** של ,JavaScript כלומר היא **מרחיבה את JS** ומוסיפה **טיפוסים (Types)** כדי להפחית שגיאות בקוד.  
הוא **מתקמפל** ל-JavaScript רגיל, ולכן עובד עם **Node.js, דפדפנים, ופרויקטי Web**

**למה להשתמש ב-TypeScript?**

* זיהוי שגיאות בזמן קומפילציה (במקום לגלות אותן בזמן ריצה)
* השלמה אוטומטית טובה יותר ב- VS Code
* קוד קריא ומסודר יותר
* תומך ב-OOP: מחלקות, ממשקים , Generics
* מתאים במיוחד לפרויקטים גדולים

**חלק 2: התקנה והגדרת סביבת עבודה (תזכורת)**

1. **התקנת TypeScript באופן גלובלי**

npm install -g typescript

1. **בדיקת גרסת TypeScript**

tsc -v

1. **יצירת תיקייה לפרויקט** (אפשר פשוט ליצור תקיה ולהריץ בה את INIT)

mkdir my-ts-project

cd my-ts-project

npm init -y

1. **התקנת ts-node**

npm install --save-dev ts-node

1. **יצירת קובץ tsconfig.json**

npx tsc --init

זה ייצור קובץ הגדרות לקומפילציה של TypeScript.

**חלק 3: הגדרת טיפוסים בסיסיים**

1. **משתנים וטיפוסים**

let message: string = "Hello TypeScript!";

let age: number = 25;

let isActive: boolean = true;

1. **מערכים**

let numbers: number[] = [1, 2, 3, 4];

let names: string[] = ["Alice", "Bob"];

1. **טיפוסי any ו-unknown**

let anything: any = "Hello"; // יכול להיות כל דבר

let unknownValue: unknown; // דורש בדיקה לפני שימוש

1. **טיפוס tuple**

let person: [string, number] = ["Alice", 30];

**חלק 4: מחלקות ואובייקטים**

**יצירת מחלקה ב-TypeScript**

class Person {

name: string;

age: number;

constructor(name: string, age: number) {

this.name = name;

this.age = age;

}

getString(): void {

console.log(`Hello, my name is ${this.name} and I'm ${this.age} years old.`);

}

}

const john = new Person("John", 25);

john. getString ();

קוד זה **יחסום שגיאות** אם תנסי להעביר טיפוסים שגויים.

הערות:

1. מספיק להגדיר בקריאה לבנאי את המשתנים והן הופכים למשתני מחלקה
2. כדי לגשת למשתני מחלקה – משתמשים במילה this.

**חלק 5: Interfaces ו-Generics**

1. **שימוש ב-Interface**

interface User {

id: number;

name: string;

email?: string; // שדה אופציונלי

}

const user: User = { id: 1, name: "Alice" };

console.log(user.name);

1. **שימוש ב-Generics**

function getFirstElement<T>(arr: T[]): T {

return arr[0];

}

const numbers : number[] = [1, 2, 3];

console.log(getFirstElement(numbers)); // 1

מאפשר פונקציות **גמישות** שמתאימות לכל טיפוס.

**חלק 6: מודולים וייבוא קבצים**

**יצירת מודול והייבוא שלו**

**math.ts**

export function add(a: number, b: number): number {

return a + b;

}

**index.ts**

import { add } from "./math";

console.log(add(2, 3)); // 5

אם נרצה ליצא משתנה ולהשתמש בו בקבץ אחר בתוכנית שלנו, נוכל גם אז להשתמש במילה export

// **src/data.ts**

export const students: string[] = [];

// **src/addStudent.ts**

import { students } from './data';

students.push("Sara");

**// src/main.ts**

import { students } from './data';

console.log(students); // יכיל גם את Sara

**חלק 7: קומפילציה והרצה**

**המרת TypeScript ל-JavaScript**

tsc index.ts

זה ייצור קובץ **index.js** שאפשר להריץ עם Node.js.

**הרצה עם ts-node (בלי קומפילציה)**

npx ts-node index.ts

**חלק 8: דוגמא לכיתה- הזמנות במסעדה**

1. **מבנה המחלקות:**

restaurant-system/

│── src/

│ ├── models/

│ │ ├── Table.ts

│ │ ├── Order.ts

│ │ ├── Customer.ts

│ ├── services/

│ │ ├── Restaurant.ts

│ ├── app.ts

│── tsconfig.json

│── package.json

1. **המחלקות:**
2. מחלקת שולחן
   1. משתני מחלקה:
      1. מספר שולחן- tableNumber
      2. מספר כסאות- seats
      3. האם תפוס - isOccupied
   2. פונקציות
      1. תפיסת שולחן - occupyTable
      2. שחרור שולחן – freeTable
3. מחלקת לקוח Customer
   1. משתני מחלקה
      1. Id
      2. Name
      3. TableNumber
   2. פונקציות
      1. placeOrder(order)
4. מחלקת הזמנה – order
   1. משתני מחלקה
      1. TableNumber
      2. Items: {name: string; price: number}[]
      3. calculateTotal
         * return this.items.reduce((total, item) => total + item.price, 0);
      4. printOrder()
5. מחלקת מסעדה לניהול הזמנות
   1. Imports
   2. addTable
   3. addCustomer

const table = this.tables.find(t => t.tableNumber === customer.tableNumber);

if (table && !table.isOccupied) {

table.occupyTable();

this.customers.push(customer);

* 1. placeOrder(Order, customerId)
  2. closeOrder(customerId)

1. מחלקת APP
   1. שימוש בכל הפונקציות הרלוונטיות