TypeScript

תכנות מונחה עצמים

מחלקות מאפיינים ומתודות

- מה ההבדל בתחביר בהקשר של מחלקות בין JavaScript ו-TypeScript?
- כיצד משתמשים במודיפיקטורים public ו-private ?
- כיצד משתמשים ב-static?
- כיצד משתמשים ב-protected?
- כיצד משתמשים ב-readonly?

Modifier	תאור
public	מודיפיקטור ברירת המחדל. כאשר חבר מחלקה (מאפיין או מתודה) מוגדר כ-public או שלא מוגדר מודיפיקטור בכלל, ניתן לגשת אליו מכל מקום בקוד.
private	כאשר חבר מחלקה מוגדר כ-private, ניתן לגשת אליו רק מתוך המחלקה שבה הוא מוגדר. אין גישה אליו מחוץ למחלקה.
protected	כאשר חבר מחלקה מוגדר כ-protected, ניתן לגשת אליו מתוך המחלקה שבה הוא מוגדר וגם מתוך מחלקות שיורשות ממנה. אין גישה אליו מחוץ למחלקה או מהאובייקטים עצמם.
readonly	כאשר חבר מחלקה מוגדר כ-readonly, ניתן להגדירו או לאתחלו רק פעם אחת, בדרך כלל בעת יצירת האובייקט (בבנאי או בעת ההגדרה).
static	כאשר חבר מחלקה מוגדר כ-static, הוא משתייך למחלקה עצמה ולא לאובייקטים שנוצרים ממנה. ניתן לגשת אליו ישירות דרך שם המחלקה.

מחלקות מאפיינים ומתודות

```
class Person {
    public name: string;
    public age: number;
    private readonly id: number;
    private static count: number = 0;

    constructor(name: string, age: number) {
        this.name = name;
        this.age = age;

        Person.count++;
        this.id = Person.count;

        this.printInfo();
    }

    public printInfo() {
        const info: string = `ID: ${this.id}\nName: ${this.name}\nAge: ${this.age}`;
        console.log(info);
    }
}

const jhon = new Person('John', 25);
```

- מה הם הכמסה והפשטה?
- כיצד נוכל לממש עקרונות אלו באמצעות שימוש במודיפיקטורים?
- האם יש הבדל בתחביר בין בנאי של מחלקה ב-TypeScript לעומת זה של JavaScript?
- כיצד נוכל לדרוש פרמטרים בבנאי ולבצע השמה למאפיינים מבלי ליצור את המאפיינים בראש המחלקה?
- מה יקרה אם לא נשלח את הפרמטרים הנדרשים לבנאי של מחלקה?

מחלקות מאפיינים ומתודות - תרגול

- צרו את מחלקת **Calculator**.
- תנו למחלקה את המאפיינים הפרטיים – **number1**,**number2**,**operator**,**sum**.
- תנו למחלקה את המתודות (ציבוריות **public**) – **calc**, **printInfo**.
- הפונקציה **calc** תערוך את החישוב ותבצע השמה ל-**sum**.
- בנאי המחלקה ידע לדרוש את הפרמטרים - **number1**,**number2**,**operator** ולבצע השמה למאפיינים הפרטיים, לאחר מכן יפעיל את המתודה **calc** של המופע.
- הפונקציה **printInfo** תדפיס את התרגיל והתוצאה בטקסט מסודר.
- צרו 3 מופעים של המחלקה באמצעות שליחת ערכים שונים.
- הפעילו את המתודה **printInfo** של המופעים השונים על מנת לחשב ולהדפיס לקונסול את התוצאה.

interface

- מהו Interface (ממשק, חוזה מימוש)?
- מה Interface יכול להכיל?
- מי יכול לממש interface?
- כיצד נוכל להשתמש ב-Interface כ-type?
- איך מטמיעים וממשים interface במחלקה?
- מה זה בא לפתור לנו?

```
interface IPoint {
    x: number;
    y: number;
    print: () => void;
}

class Point implements IPoint {
    constructor(public x: number, public y: number) { }

    public print(): void {
        console.log(`x: ${this.x}, y: ${this.y}`);
    }
}

const point = new Point(10, 20);
point.print();
```

שיעורי בית

- צרו **inteface** בשם **IVehicle** ותנו לו את המאפיינים – **brand, year, wheels**.
- צרו את מחלקת **Vehicle** וממשו בה את ה-**interface** שיצרתם.
- בנאי המחלקה ידע לדרוש את הפרמטרים - **brand, year, wheels** ולבצע השמה למאפיינים.
- הוסיפו ל-**interface** את המתודות – **drive**, **getInfo** וממשו במחלקה.
- הפונקציה **getInfo** תדע **להחזיר** את פרטי המופע כ-**string**.

הפונקציה **drive** דורשת:

1. **number** כפרמטר בשם **mph** המסמל את מהירות הנסיעה (קמ"ש),
2. **string** כפרמטר בשם **direction** המסמל את כיוון הנסיעה (קדימה, אחורה)
3. מדפיסה הודעה על נסיעה במהירות המבוקשת לכיוון הנסיעה המבוקש (כולל שימוש במתודה **getInfo**).

- צרו מופע של המחלקה ונסו להפעיל את המתודה **drive** תוך שליחת ערכים שונים בכל פעם.

סיכום שיעור

תודה על ההקשבה

אני וצוות המכללה כאן עבורכם