TypeScript - Data Modifiers

Data Modifiers - Access scope of entities.

In object-oriented programming, the concept of 'Encapsulation' is used to make class members public or private i.e. a class can control the visibility of its data members. This is done using access modifiers.

There are three types of access modifiers in TypeScript:

public

private

protected

public

By default, all members of a class in TypeScript are public. All the public members can be accessed anywhere without any restrictions.

JavaScript
```javascript
class Employee {
    public empCode: string;
    empName: string;
}
```

```
let emp = new Employee();
emp.empCode = 123;
emp.empName = "Swati";
```

- In the above example, `empCode` and `empName` are declared as public. So, they can be accessible outside of the class using an object of the class.
- Please notice that there is not any modifier applied before `empName`, as TypeScript treats properties and methods as public by default if no modifier is applied to them.

private

The private access modifier ensures that class members are visible only to that class and are not accessible outside the containing class.

```JavaScript
class Employee {
    private empCode: number;
    empName: string;
}

let emp = new Employee();
emp.empCode = 123; // Compiler Error
emp.empName = "Swati";//OK
```

In the above example, we have marked the member `empCode` as private. Hence, when we create an object `emp` and try to access the `emp.empCode` member, it will give an error.

protected

The protected access modifier is similar to the private access modifier, except that protected members can be accessed using their deriving classes.

```
Unset
class Employee {

    public empName: string;

    protected empCode: number;


    constructor(name: string, code: number){

        this.empName = name;

        this.empCode = code;

    }

}
```

```
class SalesEmployee extends Employee{

    private department: string;


    constructor(name: string, code: number, department: string) {

        super(name, code);

        this.department = department;

    }

}


let emp = new SalesEmployee("John Smith", 123, "Sales");

emp.empCode; //Compiler Error
```

In the above example, we have a class `Employee` with two members, public `empName` and protected property `empCode`. We create a subclass `SalesEmployee` that extends from the parent class `Employee`. If we try to access the protected member from outside the class, as `emp.empCode`, we get the following compilation error:

```
error TS2445: Property 'empCode' is protected and only accessible within class 'Employee' and its subclasses.
```

In addition to the access modifiers, TypeScript provides two more keywords: readOnly and static.