A screenshot of a computer

AI-generated content may be incorrect.  
  
  
A screenshot of a computer

AI-generated content may be incorrect.  
  
  
A screenshot of a computer

AI-generated content may be incorrect.  
  
  
A screenshot of a computer

AI-generated content may be incorrect.  
  
  
A screenshot of a computer

AI-generated content may be incorrect.  
  
  
A screenshot of a computer

AI-generated content may be incorrect.  
  
  
  
A screenshot of a computer

AI-generated content may be incorrect.  
  
  
A screenshot of a computer

AI-generated content may be incorrect.  
  
  
Export/Import  
  
  
A computer screen shot of a computer screen

AI-generated content may be incorrect.  
  
  
A computer screen shot of a computer

AI-generated content may be incorrect.  
  
  
A screenshot of a computer

AI-generated content may be incorrect.  
  
  
Below is the **clearest, simplest, never-forget explanation** of **Access Modifiers** and **Modules** in TypeScript, with easy examples.

**🎯 PART 1 — Access Modifiers in TypeScript**

Access modifiers control **who can access class properties or methods**.

There are **3 access modifiers**:

**1️⃣ public (default)**

✔ Accessible **everywhere**  
✔ Inside class  
✔ Outside class  
✔ In subclasses

You don’t need to write public explicitly—everything is public by default.

**✅ Example**

class Person {

public name: string = "Hamid"; // public (default)

}

let p = new Person();

console.log(p.name); // Accessible → OK

**2️⃣ private**

❌ NOT accessible outside the class  
✔ Only inside the same class

**❌ Wrong**

class Student {

private marks: number = 90;

}

let s = new Student();

console.log(s.marks); // ❌ ERROR: marks is private

**✅ Right way (use method)**

class Student {

private marks: number = 90;

getMarks() {

return this.marks; // allowed

}

}

let s = new Student();

console.log(s.getMarks()); // 90

**3️⃣ protected**

✔ Accessible inside the same class  
✔ ALSO accessible in **child classes**  
❌ NOT accessible outside class

**Example**

class Animal {

protected sound: string = "Roar";

}

class Lion extends Animal {

printSound() {

console.log(this.sound); // OK

}

}

let l = new Lion();

l.printSound(); // "Roar"

console.log(l.sound); // ❌ ERROR protected

**🎁 Super Simplest Memory Trick**

| **Modifier** | **Where Accessible?** |
| --- | --- |
| **public** | Everywhere |
| **private** | Only inside same class |
| **protected** | Class + Derived class |

**🎯 PART 2 — Modules in TypeScript**

Modules help you split your code into files and reuse code using:

* **export** (sending code out of file)
* **import** (bringing code into another file)

**🧩 Why Modules?**

Imagine a big project → 50 classes → everything in one file = confusion.

Modules allow:

✔ Clean structure  
✔ Reusable code  
✔ Clear separation

**📦 1. Export**

We export anything that we want to use in another file.

Example: student.ts

export class Student {

name: string;

constructor(name: string) {

this.name = name;

}

}

**📥 2. Import**

We import whatever we exported.

Example: app.ts

import { Student } from "./student";

let s = new Student("Hamid");

console.log(s.name);

**📌 Named Export**

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

return a + b;

}

Then import:

import { add } from "./math";

**📌 Default Export**

(Only **one** default export per file)

**student.ts**

export default class Student {

name: string = "Hamid";

}

**app.ts**

import Student from "./student"; // no {}

**🎁 Simple Memory Trick**

**✔ export → "send"**

**✔ import → "bring"**

**✔ default export → no {}**

**✔ named export → use {}**

**🧠 FINAL SUMMARY (Never Forget)**

**✔ ACCESS MODIFIERS**

* **public** → Everyone
* **private** → Only that class
* **protected** → Class + Child class

**✔ MODULES**

* **export** → send class/function to other files
* **import** → bring class/function from other files
* **default export** → imported without {}

If you want, I can explain:

✅ Interfaces  
✅ Abstract classes  
✅ Inheritance  
✅ Getters & setters

Just tell me!