- **Модуль:** Typescript. Классы
- **Урок:** Модификаторы видимости свойств

TypeScript Handbook
from
Иван Петриченко
udemy

МОДИФИКАТОРЫ ВИДИМОСТИ

TS предоставляет модификаторы для **свойств и методов класса.** Они позволяют указать разработчикам на их принадлежность и “открытость”. Они **будут удалены** из кода после компиляции, так что вводят ограничения только в TS

По умолчанию любые свойства и методы являются **публичными**, то есть доступными снаружи экземпляра класса. Так же это можно указать модификатором `public`:

```
class User {
    public email: string;
    public name: string;
}
```

Обычно их не указывают, за исключением определенного стиля проекта. Или **сокращенной записи свойств**, где TS сам создаст все нужные данные и присвоит их:

```
class User {
    public email: string;
    public name: string;

    constructor(email: string, name: string) {
      this.email = email;
      this.name = name;
    }
}
```

То же самое:

```
class User {
    constructor(public email: string, public name: string) {}
}
```

Если мы хотим запретить разработчикам работать с методом или свойством “**снаружи**”, то мы можем использовать модификатор `private`:

```
class Player {
    private login: string;
    private password: string;
    public server: string;
}

const player = new Player();
player.login = '1qaz'; // Error
player.server = 'first' // Ok
```

Теперь его можно использовать **только внутри самого класса.** Как один из вариантов работы со скрытым свойством “снаружи” - это использование **get/set**:

```
class Player {
    private login: string;
    private _password: string;
    public server: string;

    get password() {
        return this._password;
    }

    set password(newPass: string) {
        // Validation
        this._password = newPass;
    }
}

const player = new Player();
player.password = '1qaz'
```

При работе с **наследованием**, private-свойства/методы **не будут видны у потомков.** Если вы хотите сделать так, чтобы снаружи свойство не было доступно, но **появлялось у всех потомков**, необходимо использовать модификатор `protected`:

```
class Player {
    private login: string;
    private _password: string;
    public server: string;
    protected consent: boolean;
}

class CompetitivePlayer extends Player {
    rank: number;

    isConsented() {
        this.consent ? 'Yes' : 'No'; // Ok
        this.login // Error, Property 'login' is private
                   // and only accessible within class 'Player'
    }
}
```