МЕТОДЫ КЛАССОВ В TS

Методы работают **по тем же правилам**, что и обычные функции. К ним можно применять перегрузки, обобщения, типизацию возвращаемого значения и аргументов. Например, метод проверки размера посылки:

```ts
checkBoxSize(transport: number): string;
checkBoxSize(transport: number[]): string;
checkBoxSize(transport: number | number[]): string {
    if (typeof transport === "number") {
        return transport >= this.width ? "Ok" : "Not ok";
    } else {
        return transport.some((t) => t >= this.width) ? "Ok" : "Not ok";
    }
}
```

Для методов класса доступно так же ключевое слово **get/set**, превращающее их в **свойства-аксессоры**. Это позволяет инкапсулировать нужные свойства, делать проверки на моменте установки значения и возвращать наружу интерфейс работы с классом:

```ts
    get content() {
        return this._content;
    }

    set content(value) {
        this._content = `Date: ${new Date().toTimeString()}, Content: ${value}`;
    }

const firstBox = new Box(250);

console.log((firstBox.content = "Test"));
console.log(firstBox.content);
```

Свойство **_content** “скрыто” внутри класса. Мы изучим способы создать приватное свойство дальше по курсу. Но пока его можно спокойно получить через **firstBox._content** , так что символ “_” - это лишь **словесное указание** другим разработчикам.

👉 Если в паре с get не существует set, то это свойство автоматически становится **readonly**

👉 Тип аргумента **value** внутри set **устанавливается автоматически** на основании того типа, который возвращает get. Можно поменять при необходимости

👉 get/set свойства **не могут быть асинхронными**. Если нужна асинхронность - используйте обычный метод. Пример выше можно переписать, если дата приходит асинхронно (например после запроса на сервер):

```ts
async content(value: string) {
    const date = await new Date().toTimeString();
    this._content = `Date: ${date}, Content: ${value}`;
    console.log(this._content);
    return this._content;
}
```