- **Модуль:** Typescript. Необходимый уровень
- **Урок:** Модификаторы свойств: readonly (Property Modifiers)

TypeScript Handbook
from
Иван Петриченко
udemy

НЕИЗМЕНЯЕМЫЕ СУЩНОСТИ

TS позволяет на уровне синтаксиса сказать, что **свойства объекта, массивы или кортежи** являются неизменяемыми. Любая операция, направленная на это, будет воспринята как ошибка. Для этого используется модификатор `readonly`

```ts
interface User {
    readonly login: string;
}

const user: User = {
    login: "first!"
};

user.login = 'Error!'  // Error
```

```ts
const basicPorts: readonly number[] = [3000, 3001, 5555];
basicPorts[0] = 5; // Error
```

При использовании readonly на массиве, он и его содержимое становятся неизменяемыми. Такие методы как **pop(), push() и тп** работать не будут. В кортежах тоже самое:

```ts
const basicPorts: readonly [number, ...string[]] = [3000, '3001', '5555'];
basicPorts[0] = 5; // Error
basicPorts.push(566); // Error
```

Нужно помнить! Такие ограничения существуют только на этапе разработки, внутри TS. После компиляции эти свойства и массивы будут редактируемые. Но вот на этапе **разработки** они позволяют вам избежать ошибок, особенно, когда вы работаете в команде

- **Модуль:** Typescript. Необходимый уровень
- **Урок:** Модификаторы свойств: readonly (Property Modifiers)

TypeScript Handbook
from
Иван Петриченко
udemy

АЛЬТЕРНАТИВНЫЙ СИНТАКСИС

Существуют альтернативные варианты записи при помощи дженериков. Эта тема будущих уроков, но добавим её и сюда. Такой синтаксис встречается реже, так как оператор удобнее. Но в некоторых ситуациях они так же применимы:

```ts
const userFreeze: Readonly<User> = {
    login: "first!",
    password: "qwerty",
    age: 50,
};

userFreeze.age = 4484;
userFreeze.password = "dffd";
```

Все свойства в объекте стали неизменяемыми за счет применения Readonly на интерфейсе User

Для массивов другая конструкция:

```ts
const basicPorts: ReadonlyArray<number> = [3000, 3001, 5555];
basicPorts[0] = 5; // Error
basicPorts.push(566); // Error
```