- **Модуль:** Typescript. Базовые знания
- **Урок:** Примитивные литеральные типы (Literal types)

TypeScript Handbook
from
Иван Петриченко
udemy

ПРИМИТИВНЫЕ ЛИТЕРАЛЬНЫЕ ТИПЫ

Примитивы – это простые типы данных, строки, числа, булевы значения, символы и тд.
Литералы – это конкретные значения этих типов.
Примитивные литеральные типы - это типы на основании конкретных значений примитивов:

```
1   let msg: "Hello" = "Hello";
2   const salary: 5000 = 5000;
3   const truthy: true = true;
```

Теперь значение в переменной может быть только указанного типа.
Частое применение - это аргументы функций:

```
1   function startServer(protocol: "http" | "https", port: 3000 | 3001): "Server started" {
```

Теперь аргументы и результат работы функции могут иметь только несколько значений, что убережет вас от ошибок + включаются подсказки во время вызова функции

- **Модуль:** Typescript. Базовые знания
- **Урок:** Примитивные литеральные типы (Literal types)

TypeScript Handbook
from
Иван Петриченко
udemy

ИСПОЛЬЗОВАНИЕ ЛИТЕРАЛЬНЫХ ТИПОВ

В плане логики кода всегда ищите сущности, которые могут иметь **ограниченное количество значений:** методы запросов на сервер, порты, свойства браузера, анимаций и тп.

Но! В TS существует специальная структура - **Enum (перечисление)** Она мощнее и служит для тех же целей. Часто предпочтение отдается именно ей. Будет дальше по курсу.

Пример аннотирования функции с ограниченным кол-вом значений:

```ts
function createAnimation(
    id: string | number,
    animName: string,
    timingFunc: "ease" | "ease-out" | "ease-in" = "ease",
    duration: number,
    iterCount: "infinite" | number
): void {
```

Обратите внимание на аргумент iterCount. Тут не перечисление,
а выбор из общего или литерального типа. Enum тут может и не подойти

Для аргументов, которые приходят динамически в функцию, можно комбинировать аннотации, условия и константы. Так мы будем дополнительно активировать проверку и на уровне runtime кода:

```ts
const port3000: number = 3000;
const port3001: number = 3001;

function startServer(protocol: "http" | "https", port: 3000 | 3001): "Server started" {
    if (port === port3000 || port === port3001) {
        console.log(`Server started on ${protocol}://server:${port}`);
    } else {
        console.error("Invalid port");
    }

    return "Server started";
}
```