- **Модуль:** Typescript. Generics and type manipulations
- **Урок:** Template literal types

TypeScript Handbook from Иван Петриченко
udemy

TEMPLATE LITERAL TYPES

При создании новых типов есть механизм, похожий на стандартную интерполяцию строк в JS. Он позволяет формировать **литеральные типы по тем же синтаксическим правилам***(косые кавычки, обрамление переменных в ${})*:

```ts
type MyAnimation = "fade";

type MyNewAnimation = `${MyAnimation}In`; // "fadeIn"
```

При использовании **union type** будет формироваться так же тот же тип, но с модификациями:

```ts
type MyAnimation = "fade" | "swipe";

type MyNewAnimation = `${MyAnimation}In`; // "fadeIn" | "swipeIn"
```

```ts
type MyAnimation = "fade" | "swipe";
type Direction = "in" | "out";

type MyNewAnimation = `${MyAnimation}${Direction}`; // "fadein" | "fadeout" | "swipein" | "swipeout"
```

Для удобных модификаций были добавлены специальные дженерики по работе со строками. Передача в них **не строки** ведет к ошибке

Uppercase<StringType>
Lowercase<StringType>
Capitalize<StringType>
Uncapitalize<StringType>

```ts
type MyAnimation = "fade" | "swipe";
type Direction = "in" | "out";

type MyNewAnimation = `${MyAnimation}${Capitalize<Direction>}`; // "fadeIn" | "fadeOut" | "swipeIn" | "swipeOut"
```

Этот механизм можно использовать внутри **Mapped types** для изменения названий свойств:

```ts
interface Currencies {
    usa: "usd";
    china: "cny";
    ukraine: "uah";
    kz: "tenge";
}

type CreateCustomCurr<T> = {
    [P in keyof T as `custom${Capitalize<string & P>}`]: string;
};

type CustomCurrencies = CreateCustomCurr<Currencies>;
```