- **Модуль:** Typescript. Generics and type manipulations
- **Урок:** Mapped types, +/- операторы

TypeScript Handbook
from
Иван Петриченко
udemy

MAPPED TYPES

TS позволяет формировать объектные типы путем **перебора и модификации исходного типа.** Этот механизм называется **Mapped types (сопоставление типов).** К примеру, мы можем сформировать:

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

→

```ts
type CustomCurrencies = {
    usa: string;
    china: string;
    ukraine: string;
    kz: string;
}
```

Создание вручную такого типа ведет к нескольким проблемам:

! Если в списке будут все страны мира, то вы потратите огромное количество времени на изменение данных **вручную**

! При этом копия будет занимать огромное количество места в коде

! Но главное то, что у вас не будет **зависимости** одного типа от другого. Если вы удалите свойство в целевом интерфейсе, то это никак не повлияет на копию

Для оптимизации работы воспользуемся **Mapped types, синтаксис** которых:

```ts
type СопоставимыйТип = {
    [ПроизвольныйИдентификатор in Множество]: ПроизвольныйТипДанных;
};
```

Сформированный тип должен быть обязательно задан через **type!**
В самом простом варианте его можно применять к обычным литералам:

```ts
type Keys = "name" | "age" | "role";

type User = {
    [K in Keys]: string;
};

const alex: User = {
    name: "Alex",
    age: "25",
    role: "admin",
};
```

Но в практике чаще всего **mapped types комбинируется с дженериками.** Для создания нового типа на базе интерфейса валют создадим тип:

```ts
type CreateCustomCurr<T> = {
    [P in keyof T]: string;
};

type CustomCurrencies = CreateCustomCurr<Currencies>;
```

Где P - это свойства, которые берутся из ключей приходящего в дженерик типа. **keyof T** - получение этих ключей. Вместо string может быть **какой угодно тип**, необходимый вам.
Таким образом мы **установили связь** между типами и удаление одного из свойств в **Currencies** приведет к изменению типа **CustomCurrencies**

МОДИФИКАЦИИ И ОПЕРАТОРЫ +/-

Во время формирования нового типа к свойствам можно добавлять модификаторы **readonly и/или optional:**

```ts
type CreateCustomCurr<T> = {
    readonly [P in keyof T]: string;
};
```

```ts
type CreateCustomCurr<T> = {
    [P in keyof T]?: string;
};
```

Так же существуют операторы “+” и “-”, которые добавляют или убирают эти модификаторы из **исходного** типа. Оператор “+” аналогичен записи выше, когда идет простое добавление

```ts
type CreateCustomCurr<T> = {
    -readonly [P in keyof T]-?: string;
};
```