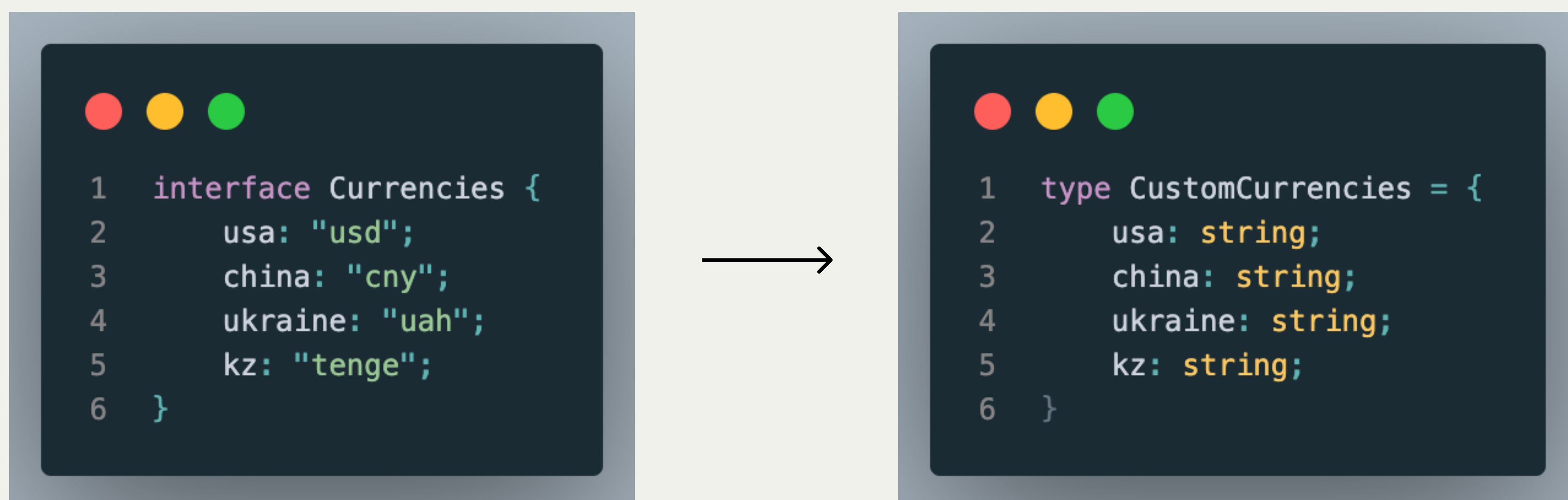


## MAPPED TYPES

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



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:

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

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

```
1 type Keys = "name" | "age" | "role";
2
3 type User = {
4   [K in Keys]: string;
5 };
6
7 const alex: User = {
8   name: "Alex",
9   age: "25",
10  role: "admin",
11 };
```

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

```
1 type CreateCustomCurr<T> = {  
2   [P in keyof T]: string;  
3 };  
4  
5 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**:

```
1 type CreateCustomCurr<T> = {  
2   readonly [P in keyof T]: string;  
3 };
```

```
1 type CreateCustomCurr<T> = {  
2   [P in keyof T]?: string;  
3 };
```

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

```
1 type CreateCustomCurr<T> = {  
2   -readonly [P in keyof T]-?: string;  
3 };
```