

ПЕРЕЧИСЛЕНИЯ В TYPESCRIPT

В разработке бывают ситуации, когда что-то ограничено перечислением нескольких вариантов. И программа выбирает один из них. Например:

- Варианты движения объекта: *вверх, вниз, влево, вправо*
- Список доступных валют: *€, \$, ₪*
- Список доступных анимаций: *fadeIn, fadeDown, swipeLeft, swipeRight*

И мы хотим предоставить ей и разработчикам только ограниченный выбор. Для этого используются **перечисления (Enum)**
Эта структура существует **только внутри TS**. Она позволяет избегать опечаток и применения сторонних вариантов:

```
1 enum Directions {
2   TOP,
3   RIGHT,
4   LEFT,
5   BOTTOM,
6 }
```

Стандартный enum (числовой)
Автоматическая нумерация значений

```
1 enum TimingFunc {
2   EASE = "ease",
3   EASE_IN = "ease-in",
4   LINEAR = "linear",
5 }
```

Строковый enum

```
1 enum TimingFunc {
2   EASE = 1,
3   EASE_IN = 2,
4   LINEAR = 10
5 }
```

Числовой enum
с установленными значениями

```
1 function frame(elem: string, dir: Directions, tFunc: TimingFunc): void {
2   if (dir === Directions.RIGHT) {
3     console.log(tFunc);
4   }
5 }
6
7 frame("id", Directions.RIGHT, TimingFunc.LINEAR);
```

Применение перечисления в функции
Enum ограничивает варианты и не дает опечататься

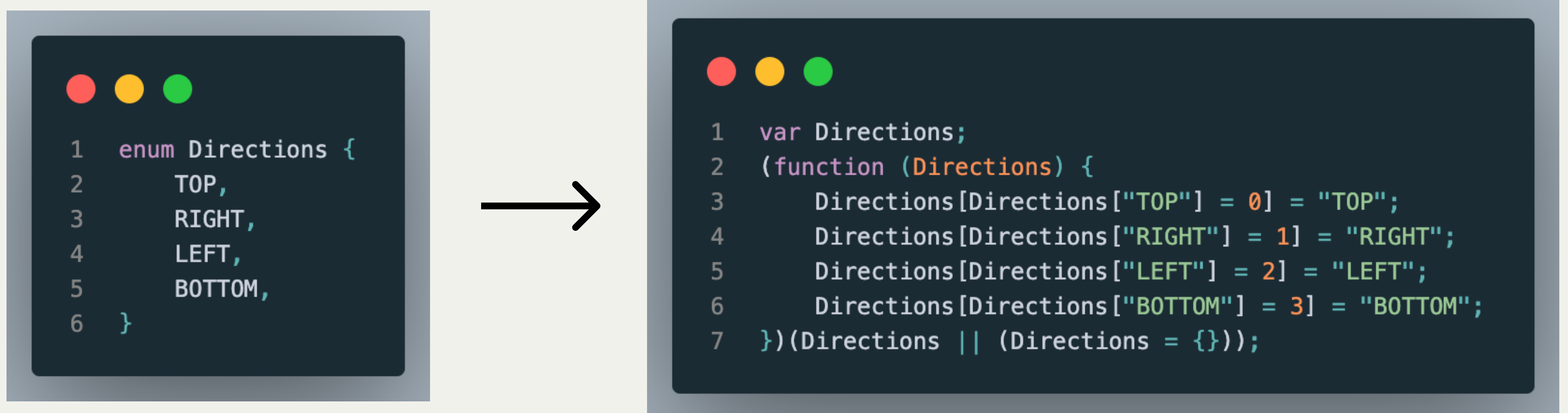
Существует вариант гетерогенного enum'а - это **комбинация строковых и числовых значений**. Но в реальности использовать такое **не стоит**.
Лучше задуматься об рефакторинге кода в таком случае.

В строковых перечислениях нельзя заниматься вычислениями для получения новых значений. В числовых - можно:

```
1 enum TimingFunc {
2   EASE = 1,
3   EASE_IN = 2,
4   LINEAR = EASE * 2,
5 }
```


ПЕРЕЧИСЛЕНИЯ ПОСЛЕ КОМПИЛЯЦИИ

После преобразования .ts файлов в .js перечисления превращаются в функции с вычислением и записью значений:



Есть вариант немного ускорить их работу в конечном коде, если вычисления внутри enum занимают какое-то время. Для этого используются константные перечисления:

```
1 const enum TimingFunc {  
2     EASE = "ease",  
3     EASE_IN = "ease-in",  
4     LINEAR = "linear",  
5 }
```

После компиляции они не формируют функции. Компилятор сделал все вычисления, нашел все ссылки на это перечисление и заменил их значениями в конечном коде. Остаются лишь комментарии где это произошло

Этот прием имеет свои подводные камни и может приводить к багам. Так что почти всегда вы будете встречать стандартный вариант. Лучше пожертвовать несколькими миллисекундами, но не получать ошибок