Typescript

И не Javascript

Кормышев Егор ИСиП-301

22 февраля 2024 г.

Вступление

Что такое TypeScript?

Typescript - Это язык программирования, изначально созданный для упрощения жизни javascript-программистов

Он добавляет к javascript статическую типизацию:

Javascript

1 + "яблоко" = "1яблоко"

Typescript

1 + "яблоко" = Error

И добавляет более понятные ошибки

Например:

Код

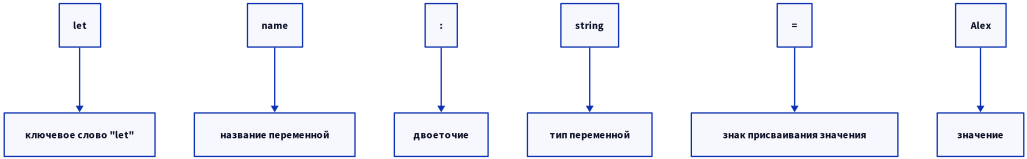
```
function add(a:number, b:number): number {
  return a + b;
}

add(1, "яблоко");
```

Ошибка

```
sample.ts:5:8 - error TS2345: Argument of type 'string' is not assignable to
parameter of type 'number'. (тип "число"нельзя привести к типу "строка")
```

Типизация переменных

```
let name:string = "Alex";
```



Базовые типы

- `string` - строка - "qwerty 'hello world'
- `number` - число (целое и дробное) - 3, 1.5
- `boolean` - логическое - true, false
- `null` - ничего (пустая переменная) - null
- `undefined` - неопределенное значение (значение может не существовать) - undefined
- `object` - объект - , name:string, age:number
- `void` - функция ничего не возвращает

Типизация объектов

Пример объекта в js

```
let user = {
  name: "Alex",
  age: 25,
  isAdmin: false
}
```

Типизированная версия

```
let user: {
  name: string,
  age: number,
  isAdmin:boolean
} = {
  name: "Alex",
  age: 25,
  isAdmin: false
}
```

Типизация функций

Чтобы типизировать функцию, нужно типизировать ее входные и выходные параметры

Пример

```
function isEven(num:number): boolean {
  if (num % 2 == 0) {
  return true
} else {
  return false
}
}
```

В этом примере `number` - тип входного параметра `num`, а `boolean` - тип возвращаемого значения

Типизация массивов

Есть 2 способа типизировать массив:
Наиболее распространенный:

```
let numbers: number[] = [1,2,3]
```

(Написать тип и поставить [] после него)

И неиспользуемый:

```
let numbers: Array<number> = [1,2,3]
```

(Написать тип Array и указать нужный тип как его generic <> после него)

Мультитипы

В Typescript одно значение или параметр может иметь несколько типов.
Для того чтобы указать это используется знак "побитовое или"(|)

Пример

```
let stringOrNumber: string | number = 1;
```

или

```
let stringOrNumber: string | number = "some string";
```

При данной типизации оба значения верны

Кастомные (Свои) типы

Для объявления типов используется следующий синтаксис:

```
type myBooleanString = string | boolean
```

Теперь более сложный пример: свой тип объекта (правильная типизация объектов)

Пример

```
type User = {
  name: string,
  age: number,
  isAdmin: boolean
}
```

Таким образом, теперь для типизации объекта достаточно сделать следующее

```
let myUser: User = {...}
```

Generics. Обобщенные типы

Обобщенный тип означает что на его месте может быть любой тип
Такой тип обозначается <T> (Вместо T может быть любая другая буква, но принято "T"т.к. type)

Пример написания Generic-функции

```
function gettype<T>(v: T): void {
  // оператор typeof возвращает тип переменной
  console.log(typeof v)
}

gettype(1)
gettype(true)
```

Результатом выполнения будет:

```
number
boolean
```

Тест

1. К какому типу принадлежат следующие значения: 4.56, 37, 23?
2. Напишите 2 способа типизации массива логических значений
3. Какой тип имеет функция в которой нет `return`?
4. Какой символ используется для обозначения мультитипа (несколько типов на одно значение)?
5. как "отключить"типизацию?

Задачи I

1. Типизировать функцию pow которая принимает число и возвращает квадрат этого числа
2. Дан объект post:

```
let post = {
  id: 1,
  title: "Первый пост",
  content: "Какой-то пост о чем-то",
  stats: {
    likes: 3,
    reposts: 2
  }
}
```

   Написать тип Post соответсвующий объекту

Задачи II

3. Написать функию `isThree` которая принимает массив любого типа <T> и проверяет кол-во элементов в нем.
   Если их кол-во 3 или более - вернуть 3 элемент, иначе вернуть `null`

Ответы на тест

1. number
2. `boolean[]` и `Array<boolean>`
3. `void`
4. `|`
5. поставить `any`

Ответы на задачи I

Решение задачи 1

```
function pow(a: number): number {
  return Math.pow(a, 2)
}
```

Ответы на задачи II

Решение задачи 2

```
type Post {
  id: number,
  title: string,
  content: string,
  stats: {
    likes: number,
    reposts number
  }
}

let myPost: Post = {...}
```

Ответы на задачи III

Решение задачи 3

```
function isThree<T>(arr: T[]): T | null {
  return arr.length >= 3 ? arr[2] : null
}
```

Спасибо за внимание!