Публикация на тему

# TypeScript

*Основы TypeScript. Использование принципов Объектно-ориентированного программирования в разработке приложений на TypeScript.*

**Анотация**

-

-

**Автор**

Михалькевич Александр Викторович

## Публикация

**Наименование** TypeScript

**Автор** А.В.Михалькевич

**Специальность** Основы TypeScript. Использование принципов Объектно-ориентированного программирования в разработке приложений на TypeScript.,

**Анотация** -

**Anotation in English** -

**Ключевые слова** JavaScript, TypeScript, ООП, разработка web-приложения, web

**Количество символов** 4882

## Содержание

Введение
1 Зачем нужен TypeScript
2 Установка TypeScript
3 Базовые типы
4 Пользовательские типы
Заключение
Список использованных источников
Приложения

## Введение

# 1 Зачем нужен TypeScript

Любая программа написанная на TypeScript компилируется в JavaScript код, и выполняется как JavaScript.

Тогда возникает вопрос: зачем вообще нужен этот "TypeScript", если всё тоже самое можно написать на JavaScript?

На такой вопрос можно отвеить так: основное преимущество TypeScript - его типизация. Строгая типизация TypeScript позволяет писать код в разных клиентских фрэймворках единым синтаксисом. Vue, React, Angular, да и в самом Node.js мы можем использовать один синтаксис TypeScript.

# 2 Установка TypeScript

В Ubuntu установить TypeScript можно одним из следующих способов:

- с помощью npm, как глобальный пакет node:

```
npm install -g typescript
```

- с помощью apt-get:

```
sudo apt install node-typescript
```

Убедиться в том что компилятор TypeScript установлен, можно с помощью команды

```
tsc -v
```

# 3 Базовые типы

При объявлении переменной есть возможность добавить двоеточие и аннотацию типа, чтобы указать тип этой переменной. Пример

```
let name: string;
let age: number;
```

В TypeScript имеются следующие встроенные аннотации типов (core types):

string
boolean
number
symbol
any
unknow
never
void

И еще два специальных значения:

null

undefined

Следующий фрагмент кода объявляет функцию, возвращающую либо `string` либо `null`

```
function getName(): string | null { ... }
```

Для объявления функции не возвращающей значение, можем использовать `void`:

```
function logErr(): void {...}
```

Вот еще пример с объявлением переменной и пустым дефолтным значением:

```
let name: string = null;
```

Типы в аргументах функции:

```
 function calc(state: string, income: number, dependts: number): number {...}
```

---

В TypeScript, вы работаете с такими типами, как `string` или `number` и другие.

**Important**: `string` и `number` (etc.), **НЕ** `String`, `Number` etc.

**Базовые типы в TypeScript всегда в нижнем регистре!**

Давайте посмотрим на такой JavaScript объект:

```
const product = {
  id: 'abc1',
  price: 12.99,
  tags: ['great-offer', 'hot-and-new'],
  details: {
    title: 'Red Carpet',
    description: 'A great carpet - almost brand-new!'
  }
}
```

Такой объект может быть объявлен следующим типом:

```
{
  id: string;
  price: number;
  tags: string[];
  details: {
    title: string;
    description: string;
  }
}
```

Пример функции в TypeScript:

```
function add(n1: number, n2: number, showResult: boolean, phrase: string) {
  // if (typeof n1 !== 'number' || typeof n2 !== 'number') {
  //   throw new Error('Incorrect input!');
```

```
  // }
  const result = n1 + n2;
  if (showResult) {
    console.log(phrase + result);
  } else {
    return result;
  }
}

let number1: number;
number1 = 5;
const number2 = 2.8;
const printResult = true;
let resultPhrase = 'Result is: ';

add(number1, number2, printResult, resultPhrase);
```

# 4 Пользовательские типы

TypeScript позволяет создать пользовательский тип разными способами:

- с помощью ключевого слова `type`
- объявлением класса, `class`
- объявлением интерфейса, `interface`
- с помощью ключевого слова `enum`

# Заключение

# Список использованных источников

# Приложения