JavaScript
IT

T Y P E S C R I P T

`npm i -g typescript` - глобально установить пакет typescript

`tsc --init` - создать конфиг для typescript

`tsc` - скомпилировать typescript файлы в javascript

Primitives

Примитивам тип указывается через двоеточие после названия переменной

```
let a: number = 1;
let b: string = "abc";
let c: boolean = false;
```

Functions

В функциях типы указываются аргументам и после круглых скобок для результата данной функции

```
const getFullName = (name: string, surname: string): string => {
  return `${name} ${surname}`;
}
```

Arrays

`const skills: string[] = ['Dev', 'DevOps']` - применение типа массива

`const skills: Array<string> = ['Dev', 'DevOps']` - применение дженерика

Tuples

`const skills: [number, string] = [1, 'Dev']` - создание кортежа

`const arr: [number, string, ...boolean[]] = [1, 'hello', true, false, false, false]` - спред типов

Readonly

`const skills: readonly [number, string] = [1, 'Dev']` - запрещает модифицировать сложные типы данных

`const skills: ReadonlyArray<string> = ['Dev', 'DevOps']` - применение дженерика

Enum

По умолчанию нумерация enum начинается с 0, но её можно изменить

```
enum StatusCode {
  SUCCESS = 1
  IN_PROCESS // номер 2
  FAILED // номер 3
}
```

Union

`const skills: string | boolean` - вертикальная черта значит "или"

`const arr: string[] | number[] = [1, 2, 'hello']` - массив только из чисел или только из строк

Типы можно группировать с помощью скобок:
`const arr: (string | number)[] = [1, 2, 'hello']` - массив из строк и чисел

Сужение типов:

```
function logId(id: string | number | boolean) {
if (typeof id === 'string') {
 ...
} else if (typeof id === 'number') { - тайпскрипт уже знает что строка не
попадает в это условие
 ...
} else {
 ...
 }
}
```

Narrowing - Оператор сужения

Есть ли данное свойство в объекте

```
    if ('role' in person) {
        additionalInformation = person.role;
    } else {
```

```
        additionalInformation = person.occupation;
    }
```

Return type

Мы определяем что функция вернёт person is Admin, то есть значение будет с типом данного интерфейса

```
export function isAdmin(person: Admin | User): person is Admin {
    return person.type === 'admin';
}
```

Type Guard

Функция тайпгард должна возвращать булевое значение. Тип возвращаемого значение должен быть
`x is string` - имя переменной и возвращаемый тип

```
function isString(x: string | number): x is string {
        return typeof id === 'string'
}
```

```
function logId (id: string | number) {
        if (typeof id === 'string') {
        console.log(id)
        } else if (typeof id === 'number') {

        }
}
```