TypeScript

Типы, Интерфейсы, Классы

TypeScript

TypeScript — язык программирования, представленный Microsoft в 2012 году и позиционируемый как средство разработки веб-приложений, расширяющее возможности JavaScript

TypeScript является обратно совместимым с JavaScript и компилируется в последний

Объявление типов

TypeScript обеспечивает объявления типов для статической проверки их согласования.
Это не является обязательным и может быть проигнорировано, чтобы использовать обычную динамическую типизацию JavaScript

let a: number;

function sum(numbers: number[]): number;

sum([‘a’, ‘b’]); // Ошибка, т.к. ожидается массив с числами

Базовые типы

- boolean
- number
- string
- array
- tuple
- enum
- any
- void
- null
- undefined
- never
- object

Boolean

Логический тип данных. Может принимать значения true и false.

let isDone: boolean = false;

Number

В js все числа представлены как float. Для всех чисел используется тип number.

```
let decimal: number = 6;
let hex: number = 0xf00d;
let binary: number = 0b1010;
let octal: number = 0o744;
```

String

String - любое строковое значение. Строки оборачиваются в одинарные или двойные кавычки.

```
let color: string = "blue";
color = 'red';

let fullName: string = `Bob Bobbington`;
let age: number = 37;
let sentence: string = `Hello, my name is ${ fullName }. I'll be ${ age + 1 } years old next month.`;
```

Array

Синтаксис:

let list: number[] = [1, 2, 3];
let list: Array<number> = [1, 2, 3];

Массив может содержать любые типы, например:
let a: string[];
let b: boolean[];
let c: any[];

Также можно использовать пользовательские типы

Tuple (кортеж)

Tuple по факту является массивом, но ограниченным по размеру и типам элементов

```
// Declare a tuple type
let x: [string, number];
// Initialize it
x = ["hello", 10]; // OK
// Initialize it incorrectly
x = [10, "hello"]; // Error
```

Enum (перечисление)

Enum используется для облегчения именования и ограничения возможных значений.

```
enum Color {Red, Green, Blue}
let c: Color = Color.Green;
```

Так, например, если мы создадим функцию которая принимает на вход тип Color, то не сможем передавать чтолибо другое.

```
const setColor = (color: Color) => {}

setColor(‘red’) // Error
```

Any

Any - все что угодно. Используется там, где заранее нельзя предугадать какой тип данных придет, либо он не имеет значения.

```
const log = (logObject: any) => {
    console.log(`Log: ${logObject}`);
}
```

Void

Void - противоположность any. Обозначает отсутствие какого-либо типа.

Используется в основном в функциях которые ничего не возвращают.

```
const click = (): void => {}
```

Null and Undefined

Используются для описания стандартных null и undefined в js. Не могут принимать других значений.

```
// Not much else we can assign to these variables!
let u: undefined = undefined;
let n: null = null;
```

Object

Object - любой не примитивный тип.

Все, кроме number, string, boolean, symbol, null, or undefined

Пользовательские типы

```
type Person = {
    name: string;
    email: string;
    id: number;
};
```

Интерфейсы

Интерфейс определяет какие поля и методы должен содержать объект.

```
interface Person {
    firstName: string;
    lastName: string;
}
function greeter(person: Person) {
    return "Hello, " + person.firstName + " " + person.lastName;
}
let user = { firstName: "Jane", lastName: "User" };
document.body.textContent = greeter(user)
```

Классы

Синтаксис класса в TypeScript:

```
class Student {
    fullName: string;
    constructor(public firstName: string, public lastName: string) {
        this.fullName = firstName + " " + lastName;
    }
    getFullName() {
        return this.fullName;
    }
}

const student = new Student(‘Ivan’, ‘Ivanov’);
```