**Курс " Javascript Fullstack Developer" и "Javascript Frontend"**

# **Кожевников Сергей 2024 год**

# **Задание 5**

1) Написать аналоги Utility Types, придумать собственный тип.  
2) Типизировать задачи из домашнего задания JavaScript. Использовать настройки strict и noImplicitAny. Избегать утверждения типов (операторы as и non-null assertion).  
3) Изучить структуру tsconfig: <https://www.typescriptlang.org/tsconfig>.

Напишите аналоги Utility Types

Придумайте собственный тип

Ознакомьтесь со шпаргалками по TS (https://www.typescriptlang.org/cheatsheets)

# **Решение**

type TObject = {

    name:string,

    type\_obj:string,

    stereotype:string[],

    notes:string,

    status:string,

    version:number,

    author:string,

    };

type Requirement = TObject & {

    CL\_STS\_REQ?:number,

    }

type Artifact = TObject & {

    CL\_STS\_ART?:number,

    }

const req01: Requirement = {

    name:'TestReq01',

    type\_obj:'Requirement',

    stereotype:['EPC'],

    notes:'Тестовое требование',

    status:'Proposed',

    version:1.1,

    author:'Sergey Kozhevnikov',

    CL\_STS\_REQ:1,

    };

    console.log([req01.name](http://req01.name/))

const art01: Artifact = {

    name:'TestArt01',

    type\_obj:'Artifact',

    stereotype:['EPC'],

    notes:'Тестовый артифакт ',

    status:'Proposed',

    version:1.0,

    author:'Sergey Kozhevnikov',

    CL\_STS\_ART:110,

    };

    console.log([art01.name](http://art01.name/))

//class

//abstract

class A\_TObject{

    name:string;

    notes!: string;

    status!:string;

    version!:number;

    author!:string;

    constructor(name: string){

        this.name=name;

    };

getInfo() :void {

    console.log('Object 1 = '+ this.name)

    }

};

class cl\_TObject extends A\_TObject {

    private type\_obj!:string;

    private stereotype!: string[];

constructor(name: string,type\_obj:string,stereotype:string[]){

        super(name);

        this.type\_obj=type\_obj;

        this.stereotype=stereotype;

    }

getInfo(): void {

        console.log('Object 2 = '+ this.name +','+ this.type\_obj +','+ this.stereotype)

    }

}

let \_req02 = new A\_TObject('TestReq10100');

\_req02.getInfo();

let \_req01 = new cl\_TObject('TestReq10000','Requirement',['ЕПС']);

\_req01.getInfo();

Для запуска ts, нужно включить на компе политику

Set-ExecutionPolicy RemoteSigned

* Открываем терминал PowerShell от админа.
* Пишем и запускаем: Set-ExecutionPolicy RemoteSigned
* На вопрос отвечаем: A (Да для всех)
* Глобально установите компилятор TypeScript на вашей машине следующей командой:

npm i -g typescript

* Убедитесь, что установка прошла успешно (если да - то команда вернет номер версии).

tsc –v

Создаейте файл конфигурации TS:

tsc —init

После этого, в корневой директории, у вас должен появиться файл tsconfig.json .

Вот пример некоторых опций, о которых стоит знать:

{  
 "compilerOptions": {  
 ...  
 /\* Модули \*/  
 "target": "es2016", // Измените на "ES2015", чтобы скомпилировать в ES6  
 "rootDir": "./src", // Откуда компилировать  
 "outDir": "./public", // Куда компилировать (обычно папка, которая разворачивается на сервере)  
 /\* Поддержка JavaScript \*/  
 "allowJs": true, // Позволяет компилировать JS-файлы  
 "checkJs": true, // Проверяет типы в JavaScript-файлах и сообщает об ошибках  
  
 /\* Emit \*/  
 "sourceMap": true, // Создать source map файлы для готовых файлов JavaScript (хороошо подходит для дебаггинга)  
 "removeComments": true, // Игнорировать комментарии

/\* Type Checking \*/

"strict": true,                                      /\* Enable all strict type-checking options. \*/

"noImplicitAny": true,                            /\* Enable error reporting for expressions and declarations with an implied 'any' type. \*/

"strictNullChecks": true,                         /\* When type checking, take into account 'null' and 'undefined'. \*/

"strictFunctionTypes": true,                      /\* When assigning functions, check to ensure parameters and the return values are subtype-compatible. \*/

"strictBindCallApply": true,                      /\* Check that the arguments for 'bind', 'call', and 'apply' methods match the original function. \*/

"strictPropertyInitialization": true,             /\* Check for class properties that are declared but not set in the constructor. \*/

// "noImplicitThis": true,                           /\* Enable error reporting when 'this' is given the type 'any'. \*/

// "useUnknownInCatchVariables": true,               /\* Default catch clause variables as 'unknown' instead of 'any'. \*/

"alwaysStrict": true,                             /\* Ensure 'use strict' is always emitted. \*/

},  
  
 "include": ["src"] // Компилировать только файлы из папки src  
}

Скомпилировать все и наблюдать за изменениями:

tsc -w

**Как компилировать TypeScript**

создайте TypeScript-файл (например, index.ts).

Напишите какой-нибудь код на JavaScript или TypeScript:

Скомпилируйте это в JavaScript следующей командой:

tsc index

TSC скомпилирует код в JavaScript-файл index.js:

Если вы хотите изменить название файла:

tsc index.ts --outfile file-name.js

Если вы хотите, чтобы TSC компилировал ваш код автоматически, при изменениях, добавьте флаг “watch” (-w):

tsc index.ts -w

После выполнения, запускаем команду

PS D:\JSProject> tsc -init

Created a new tsconfig.json with:

TS

target: es2016

module: commonjs

strict: true

esModuleInterop: true

skipLibCheck: true

forceConsistentCasingInFileNames: true

You can learn more at https://aka.ms/tsconfig

PS D:\JSProject>