УНИВЕРСИТЕТ СИНЕРГИЯ

ДОМАШНЕЕ ЗАДАНИЕ

Тема 3. Продвинутые возможности TypeScript

Цель работы: Сформировать навык подключения и настройки TypeScript в проект, используя полученные знания из уроков

План работы:

1. Создать новый TS-проект со структурой

   src/

   --**LegacyModule**/

   -----index.js

   -----index.d.ts

   --**Posts**/

   -----index.ts - реэкспортирует всё из других файлов в папке Posts

   -----posts.interface.ts - описан интерфейс поста

   -----posts.service.ts - берет данные с внешнего API

   --index.ts - главный файл (код будет приведен ниже)

   tsconfig.json
2. Инициировать в папке проекта NPM. Установить через NPM модуль работы с http-запросами Axios
3. Установить через NPM библиотеку Lodash и декларации для данной библиотеки из DefinitelyTyped
4. Написать функцию getPosts для получения постов методом GET с публичного API https://jsonplaceholder.typicode.com/posts через Axios в файле posts.service.ts
5. Описать интерфейс поста IPost в файле posts.interface.ts.

6. Написать в LegacyModule/index.js функцию capitalizeTitles и декларацию к ней в d.ts-файле, которая принимает только наш массив постов типа IPost[], заменяет заголовок (title) каждого поста на заглавные буквы, а затем возвращает измененный массив того же типа обратно (см. вызов этой функции в сниппете кода ниже)
7. Команда для компиляции приложения и запуска через терминал NodeJS **tsc -p ./tsconfig.json && node ./build/index.js** (включает в себя 2 последовательно выполняемые команды, разделенные оператором &&; первая - компиляция, вторая - запуск скомпилированного проекта через nodejs)

src/index.ts

```
import { filter } from 'lodash';
import { IPost, getPosts } from './Posts';
import capitalizeTitles from './LegacyModule';


(async function filterPosts() {
   const userId: number = 1;
   const posts = await getPosts(); // каст типов
   console.log('posts?.data.length', posts?.data.length); // Выведем в консоли длину полученного массива - должна быть 100

   const filteredPosts: IPost[] = filter(posts?.data, (p: IPost) => { return p.userId === userId })
   // console.log('filteredPosts', filteredPosts)
   console.log('filteredPosts.length', filteredPosts.length) // выведем в консоли длину отфильтрованного массива, должна быть 10

   const test = capitalizeTitles(filteredPosts);
   console.log('test', test) // filteredPosts, где все заголовки заглавными буквами
})()
```

tsconfig.json (в корне проекта)

```
{
 "include": ["./src"],
 "compilerOptions": {
   "target": "es2015",                          /* Set the JavaScript
language version for emitted JavaScript and include compatible library
declarations. */
   "module": "commonjs",                        /* Specify what module
code is generated. */
   "rootDir": "./src",                          /* Specify the root
folder within your source files. */
   "allowJs": true,                             /* Allow JavaScript
files to be a part of your program. Use the 'checkJS' option to get
errors from these files. */
   "outDir": "./build",                         /* Specify an output
folder for all emitted files. */
 "removeComments": true,                      /* Disable emitting
comments. */
   "noEmitOnError": true,                       /* Disable emitting
files if any type checking errors are reported. */
   "esModuleInterop": true,                     /* Emit additional
JavaScript to ease support for importing CommonJS modules. This
enables 'allowSyntheticDefaultImports' for type compatibility. */
   "forceConsistentCasingInFileNames": true,  /* Ensure that casing is
correct in imports. */
   "strict": true,                              /* Enable all strict
type-checking options. */
   "noImplicitAny": true,                       /* Enable error
reporting for expressions and declarations with an implied 'any' type.
*/
   "skipLibCheck": true                         /* Skip type checking
all .d.ts files. */
 }
}
```

Рекомендации для выполнения:

1. Файл Posts/index.ts только реэкспортирует все содержимое двух других файлов из папки Posts/
2. Для описания интерфейса IPost этого можно либо перейти по ссылке https://jsonplaceholder.typicode.com/posts, либо посмотреть структуру объекта поста в console.log и вручную расписать поля объекта
3. Для асинхронной функции getPosts в файле Posts/posts.service.ts можно использовать либо await axios.get(), либо await axios(), см документацию. Импорт библиотеки Axios в документации производится с помощью функции require (CommonJS), мы же будем использовать TS-синтаксис import … from …
4. Особенностью Axios является то, что response содержит служебные поля, а непосредственно данные от API содержатся в поле response.data, что отражено в коде файла index.ts, представленного в документе выше
5. LegacyModule будет экспортировать функцию через module.exports = capitalizeTitles;
6. Файл декларации index.d.ts должен импортировать интерфейс IPost с целью его переиспользования в типизации функции capitalizeTitles

Вспомогательные материалы к домашнему заданию:

https://axios-http.com/docs/intro - установка
https://axios-http.com/docs/example - пример get-запроса

Критерии оценивания:

К1— Создана структура проекта, скопированы файлы src/index.ts и tsconfig.json, инициирован NPM и установлены axios, lodash и @types/lodash (1 балл).
К2 — Написан индексный файл, сервис и интерфейс в папке Posts, сделаны экспорты и реэкспорт (2 балла).
К3 — Написан legacy модуль и файл декларации к нему (1 балл).
К4 — Нет ошибок TS-компилятора при сборке и в IDE, а так же выводятся ожидаемые результаты без ошибок в терминале в рантайме (при выполнении через NodeJS) (1 балл)

Итого: 5 баллов
Минимальное количество баллов, чтобы преподаватель смог зачесть вашу работу — 4