Знакомство с TypeScript

Contents

Task 01. Basic Types

1. Реализуйте функцию **getAllBooks()**, которая возвращает коллекцию книжек. Объявите эту коллекцию внутри функции, используя let.

```
[
      { title: 'Refactoring JavaScript', author: 'Evan Burchard', available: true},
      { title: 'JavaScript Testing', author: 'Liang Yuxian Eugene', available: false },
      { title: 'CSS Secrets', author: 'Lea Verou', available: true },
      { title: 'Mastering JavaScript Object-Oriented Programming', author: 'Andrea Chiarelli',
      available: true }
]
```

2. Реализуйте функцию **logFirstAvailable()**, которая принимает массив книг в качестве параметра и выводит в консоль:
   a. количество книг в массиве
   b. название первой доступной книги

   Используйте:
   c. следующие типы данных number, string, void.
   d. for-of для обхода коллекции
   e. бектикс (`) для вывода строчных значений
3. Запустите функцию **logFirstAvailable()**

Task 02. Enum

1. Объявите **enum Category** для хранения следующих категорий книг:
   a. JavaScript
   b. CSS
   c. HTML
   d. TypeScript
   e. Angular
2. Добавьте категорию к объектам в функции **getAllBooks()**
3. Реализуйте функцию **getBookTitlesByCategory()**, которая на вход будет получать категорию и возвращать массив наименований книг, которые принадлежат указанной категории. Используйте тип Array<string> и объявленный enum.
4. Реализуйте функцию **logBookTitles(),** которая принимает массив строк и выводит его в консоль. Используйте типы: string[] и void.

Task 03. Defining an Interface

1. Объявите интерфейс **Book**, который включает следующие поля:
    a. id - число
    b. title - строка
    c. author - строка
    d. available - булеан
    e. category – категория
2. Внесите изменения в функцию **getAllBooks()**, укажите тип возвращаемого значения, используя объявленный выше интерфейс **Book**. Удалите временно id у книжки и увидите, что появится ошибка.
3. Создайте функцию **printBook()**, которая на вход принимает книгу и выводит в консоль фразу **book.title + by + book.author**. Для типа параметра используйте интерфейс **Book**.
4. Объявите переменную **myBook** и присвойте ей следующий объект

```
{
  id: 5,
  title: 'Colors, Backgrounds, and Gradients',
  author: 'Eric A. Meyer',
  available: true,
  category: Category.CSS,
  year: 2015,
  copies: 3
}
```

5. Вызовите функцию **printBook()** и передайте ей **myBook.** Никаких ошибок при этом не должно появляться.
6. Добавьте в интерфейс **Book** свойство **pages: number**. Вы получите ошибку в функции **getAllBooks().** Чтобы ошибка не возникала сделайте свойство не обязательным.
7. Укажите явно для переменной **myBook** тип **Book**. Вы снова получите ошибку. Удалите свойства **year, copies**. Добавьте свойство **pages: 200**.
8. Добавьте в интерфейс **Book** необязательное свойство **markDamaged**, которое является методом. Метод принимает на вход строчный параметр **reason** и ничего не возвращает. Добавьте этот метод в объект **myBook**. Метод должен выводить строчку **`Damaged: ${reason}`**, используя стрелочную функцию. Вызовите этот метод и передайте строку **‘missing back cover’**

Task 04. Interfaces for Class Types

1. Объявите интерфейс **Librarian**, который содержит следующие свойства
   - Строчные свойства **name, email**, **department**
   - Функция **assistCustomer**, которая принимает строчный параметр **custName** и ничего не возвращает.
2. Создайте класс **UniversityLibrarian**, который реализует интерфейс **Librarian** и реализуйте все необходимые свойства. Метод **assistCustomer** должен выводить в консоль строчку **`${this.name} is assisting ${custName}`.**
3. Объявите переменную **favoriteLibrarian** используя интерфейс **Librarian** и проинициализируйте ее с помощью объекта, созданного классом **UniversityLibrarian()**. Никаких ошибок при этом не должно возникать. Проинициализируйте свойство **name** и вызовите метод **assistCustomer()**.

Task 05. Creating and Using Classes

1. Создайте класс **ReferenceItem**, который содержит:
    a. Строчное свойство **title**
    b. Числовое свойство **year**
    c. Конструктор с двумя параметрами: строчный параметр **newTitle**, числовой параметр **newYear**, который в консоль выводит строчку **'Creating a new ReferenceItem...'** и инициализирует поля.
    d. Метод **printItem()** без параметров, который ничего не возвращает. Этот метод должен использовать template string literal и выводить строчку **«title was published in year»** в консоль.
2. Объявите переменную **ref** и проинициализируйте ее объектом **ReferenceItem**. Передайте значения параметров в конструктор. Вызовите метод **printItem()**.
3. Закомментируйте конструктор, свойства **title** и **year** и реализуйте создание свойств через параметры конструктора **(title- public, year - private)**.
4. Создайте приватное свойство **_publisher: string**.
    a. Добавьте геттер **publisher**, который преобразовывает свойство **_publisher** в верхний регистр и возвращает его.
    b. Добавьте сеттер **publisher**, который принимает строчный параметр **newPublisher** и устанавливает значение свойства **_publisher** в значение этого параметра.
    c. Проинициализируйте свойство **ref.publisher** каким-либо сточным значением и выведите его в консоль. Результат должен быть в верхнем регистре.
5. Создайте статичное строчное свойство **department** и проинициализируйте его каким-либо значением по умолчанию. Внесите изменения в метод **printItem()** – добавьте вывод в консоль этого статического свойства.

Task 06. Extending Classes

1. Создайте класс **Encyclopedia** как наследника класса **ReferenceItem**. Добавьте одно дополнительное числовое публичное свойство **edition**. Используйте параметры конструктора.
2. Объявите переменную **refBook** и создайте объект **Encyclopedia**. Вызовите метод **printItem()**;
3. Переопределите метод **printItem().** Пусть он делает то, что делал и дополнительно выводит строчку в консоль «**Edition: edition (year)**». Вы получите ошибку, что свойство **year** недоступно. Чтобы оно было доступно измените модификатор доступа в классе **ReferenceItem** на **protected.**

Task 07. Export and Import

1. Создайте файл **enums.ts**, перенесите в него **enum Category**. Добавьте экспорт в конце файла.
2. Создайте файл **intefaces.ts** и перенесите в него интерфейсы:
    a. **Book, Librarian**
    b. Добавьте импорт **Category**
    c. Добавьте экспорт интерфейсов **Book, Librarian** в конце файла.
3. Создайте новый файл **classes.ts** и перенесите в него классы. **UniversityLibrarian, ReferenceItem**.
    a. Добавьте импорт интерфейсов как целого модуля с именем **Interfaces**
    b. Измените описание класса **UniversityLibrarian**, чтобы он реализовывал интерфейс **Interfaces.Librarian**
    c. Добавьте экспорт в конце файла и экспортируйте оба класса.
4. Внесите изменения в файл **app.ts**
    a. Добавьте импорт **Category**, интерфейсов **Book, Librarian**, классов **UniversityLibrarian, ReferenceItem**.

Task 08. Re-Export

1. Создайте папку **classes** и переместите в нее файл **encyclopedia.ts**
2. Разнесите классы **UniversityLibrarian** и **ReferenceItem** по разным файлам и тоже переместите в папку **classes**.
3. Удалите файл **classes.ts**
4. Создайте файл **classes/index.ts** и добавьте в него реэкспорт классов **Encyclopedia**, **ReferenceItem**, **UniversityLibrarian**.
5. Исправьте импорты в файле **app.ts**