**Лабораторная работа №6**

**Основы JavaScript. Синтаксис языка JavaScript. Функции, массивы, строки, сортировки. ES6 классы. Реализация функционала в консоли.**

Сдать до 14.11

Цель. *Познакомиться с основами JavaScript. Познакомиться с синтаксисом классов JavaScript. Перенести данные приложения из HTML в JavaScript****(согласно индивидуальному заданию)****. Научиться запускать и делать Debug JavaScript-кода в консоли браузера.  Научиться организовывать функциональность в объектном подходе. Реализовать основные методы для работы приложения в классе коллекции (модели MVC).*

**Задание**

**1. Перенести данные из HTML (если были созданы) в JavaScript.**

В JavaScript файле создать массивы для хранения списка информации ***согласно индивидуальным вариантам***. Заполнить массив *ObjInf* объектами *(минимум 20).*

*Если массивов несколько (например, предметы, студенты, преподаватели), то распределите разработку проекта между студентами в группе.*

Каждый объект *ObjInf* должен иметь следующую структуру:

**id: string**,уникальный для каждого объекта информации, обязательное поле

**description:  string**,длина < 200 символов, обязательное поле

**createdAt: Date**, обязательное поле

**author: string**, обязательное поле, не должно быть пустым

**photoLink: string**, не обязательное поле,(ссылка на фотографию)

**\*другие поля по инд. заданию**(смотреть поля таблиц БД в инд. задании)

**Пример:**

**varObjInf**= [

 {

**id**: '1',

**descriprion**: 'НовостиФПМИ',

**createdAt**: newDate('2021-03-5T23:00:00'),

**author**: 'Иванов Иван',

**photoLink: 'http://fpmi.bsu.by/ImgFpmi/Cache/banner\_61403.jpg'**

...

  },

  {

**id**: '2',

...

  }

];

**2. Создать класс, который должен содержать следующие методы для работы с массивом ObjInf:**

**getObjs(skip?: number, top?: number, filterConfig?:Object):Array<Object>** - получить массив объектов с сортировкой по Id (если фильтр **filterConfig** не указан) и пагинацией. Предусмотреть возможность фильтрации.

**skip** - number - параметр для пагинации, с какого элемента массива считывать информацию, опциональный параметр, default: 0.

**top** - number - параметр для пагинации, сколько информации получить, опциональный параметр, default: 10.

**filterConfig -** Object - параметр для применения фильтров. Структура объекта согласно индивидуальному варианту. Реализовать фильтры и сортировку по всем полям сруктуры **ObjInf.**

Пример:

* getObjs(0, 10)  должен отсортировать массив по дате создания и вернуть первые 10 элементов массива.
* getObjs(10, 10)  должен отсортировать массив по дате создания и вернуть 10 элементов массива, начиная с 11-ого.
* getObjs(0, 10, {author: ‘Mr. Snow’})  должен выбрать те данные из массива, где автором является ‘MrSnow’.

**getObj(id: string): Object** - получить информацию из массива Objs с определенным id.

**validateObj**(Obj:Object): **boolean** - проверить объект Obj на валидность: присутствие всех обязательных полей в нужном (!) формате (см. структуру объекта Obj выше).

**addObj**(Obj:Object): **boolean** - добавить новый Obj в массив Objs, предварительно проверив его на валидность, вернуть true если добавлен успешно, иначе false.

**editObj**(id:string, Obj: Object): **boolean**–изменить информацию в массиве Objs по id. Объект Obj может не содержать все обязательные поля, а иметь лишь те поля, которые необходимо изменить (можно менять все поля, кроме id, author, createdAt). Перед изменением требуется проверить его валидность. Вернуть true или false.

*Пример:*

*editObj('1', { descriprion: 'Новая информация}) - изменит descriprion в ObjInf с id === '1'.*

**removeObj**(id: string): boolean - удалить по id из массива Objs.   
  
  
P.S Сигнатура методов указана примерно. Реализовывать на свое усмотрение. В каждом методе желательно проверить параметры на корректность.

**3. Правильно организовать приватные поля и методы.**

Все вспомогательные методы (внутренняя логика, валидация зависящая от списка индивидуального объектов т.д.) и поля (*ObjIn*) должны быть оформлены в соответствии с private methods notation ( ***\_privateField*** ).

**4. Дополнить класс недостающими методами.**

Добавить :

* ***constructor***(Objs: Array<Object>) - конструктор, создает коллекцию на основе массива информации.
* ***addAll***(Objs: Array<Object>): Array<Object> - добавляет всю информацию из массива в коллекцию.
* другие методы на ваше усмотрение. Например: ***clear***() - очистить коллекцию

**5. Проверить работу всех методов.**

Требуется проверить работу всех методов в консоли. Выполнить их с различными параметрами: валидными и не валидными.

**6. Файл/ы и работу** продемонстрировать в аудитории преподавателю и записать на cсылку на github или https://docs.google.com/… или и.т.д.