Цель задачи

Практика проверки и преобразования различных видов данных.

Что нужно сделать

Создайте панель управления студентами, на которой будет располагаться:

* таблица со студентами с фильтрами и сортировкой,
* форма добавления нового студента.

Студенты должны храниться в массиве. Каждый студент — объект, содержащий следующие данные:

* имя,
* фамилия,
* отчество,
* дата рождения (объект Date),
* год начала обучения,
* факультет (строка).

Для добавления студентов на странице должна выводиться форма с полями, соответствующими данным студента. Форма должна проходить валидацию по следующим правилам:

* все поля обязательны для заполнения после применения к значению метода trim();
* дата рождения находится в диапазоне от 01.01.1900 до текущей даты;
* год начала обучения находится в диапазоне от 2000-го до текущего года.

Валидация должна происходить после нажатия на кнопку «Добавить студента», расположенную под полями для ввода. Если валидация прошла успешно, то все поля очищаются, а новый студент добавляется в таблицу. В противном случае над кнопкой нужно вывести сообщения с описанием ошибок для пользователя. Тексты сообщений остаются на ваше усмотрение, но они должны дать достаточно информации, чтобы понять, какие ошибки в каких полях были допущены.

Данные из массива должны выводиться в табличном виде. Каждая строка таблицы содержит информацию об одном студенте. Колонки таблицы:

* Ф. И. О. студента.
* Факультет.
* Дата рождения и возраст в формате «31.12.2000 (20 лет)». Возраст должен быть вычислен из даты рождения.
* Годы обучения и номер курса в формате «2019-2023 (2 курс)». Считается, что все студенты учатся четыре года, то есть диапазон с годами обучения выводится как {год начала обучения}-{+4 года}. Если сентябрь года окончания обучения уже прошёл, в скобках вместо указания курса должно выводиться «закончил».

Первая строка таблицы — заголовочная, в ней указываются заголовки колонок (Ф. И. О., факультет, дата рождения и возраст, годы обучения). При нажатии на ячейку заголовочной строки должна происходить сортировка по соответствующим полям студентов:

* Ф. И. О. сортирует по соединённой строке из фамилии, имени и отчества по алфавиту по возрастанию.
* Факультет — по факультету по алфавиту по возрастанию.
* Дата рождения и возраст — по дате рождения по возрастанию.
* Годы обучения — по году начала обучения.

Перед таблицей также нужно вывести фильтры, состоящие из полей:

* Ф. И. О. для поиска подстроки в фамилии, имени или отчестве.
* Факультет для поиска подстроки в названии факультета.
* Год начала обучения (точное совпадение).
* Год окончания обучения (точное совпадение).

При любых изменениях в полях для фильтрации содержимое таблицы должно измениться в соответствии с указанными фильтрами. Если указано несколько фильтров, то все они применяются к массиву студентов по очереди.

Рекомендации по выполнению

* В задаче нет чётких требований к внешнему виду программы, но рекомендуем использовать готовые библиотеки для стилей вроде Bootstrap, чтобы не верстать все элементы самостоятельно.
* Для ввода даты рождения студента можно воспользоваться элементом input с атрибутом type="date". Для хранения даты можно использовать специальный объект Date. Подробнее — в материалах:
  + [Объект Date](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Date)
  + [<input type="date">](https://developer.mozilla.org/ru/docs/Web/HTML/Element/Input/date)  
    Чтобы получить из <input type="date"> объект даты Date, можно воспользоваться свойством DOM-элемента этого поля valueAsDate.
* В этой задаче важно отделить данные от представления. Это значит, что сначала нужно отдельно сформировать массив со студентами, а только потом из получившегося массива создать DOM-дерево. При этом таблицу можно «перерисовывать» каждый раз, когда набор выводимых элементов изменился. Такой подход позволит сильно упростить код, так как за отрисовку таблицы будет отвечать простая функция, принимающая массив студентов, в то время как логика фильтрации, сортировки и добавления элемента будет работать только с этим массивом без взаимодействия с DOM.
* Можно составить собственный план работы или использовать предложенную ниже схему:

**Этап 1.** В HTML-файле создайте вёрстку элементов, которые будут статичны (неизменны).

**Этап 2.** Создайте массив объектов студентов. Добавьте в него объекты студентов, например пять.

**Этап 3.** Создайте функцию вывода одного студента в таблицу по аналогии с тем, как вы делали вывод одного дела в модуле 8. Функция должна вернуть HTML-элемент с информацией о пользователе. У функции должен быть один аргумент — объект студента.

**Этап 4.** Создайте функцию отрисовки всех студентов. Аргументом функции будет массив студентов. Функция должна использовать ранее созданную функцию создания одной записи для студента. Цикл поможет создать список студентов. Каждый раз при изменении списка студентов вы будете вызывать эту функцию для отрисовки таблицы.

**Этап 5.** К форме добавления студента добавьте слушателя события отправки формы, в котором будет проверка введённых данных. Если проверка пройдёт успешно, добавляйте объект с данными студентов в массив студентов и запустите функцию отрисовки таблицы студентов, созданную на этапе 4.

**Этап 5.** Создайте функцию сортировки массива студентов и добавьте события кликов на соответствующие колонки. [Пример](https://youtu.be/YinzpmFSP44) наглядно демонстрирует, как это можно сделать.

**Этап 6.** Создайте функцию фильтрации массива студентов и добавьте события для элементов формы. [Пример](https://youtu.be/SH5SizdvtR0) наглядно демонстрирует, как это можно сделать.

Проверка результата

Убедитесь, что результат выполнения задачи соответствует всем требованиям в описании. Реализованы сортировка, фильтрация и добавление элемента в массив к перерисовке таблицы. Таблица всегда содержит актуальные данные.