Лабораторная работа 5

Документная объектная модель (DOM)

Цель: получить навыки использования методов для работы с массивами

- Лабораторная работа сдается преподавателю лично студентом.
- Сдача лабораторной работы заключается в том, что студент отвечает на вопросы преподавателя по заданиям, а также показывает и комментирует результаты самостоятельно выполненных заданий.
- Всё что написано в заданиях «изучите, узнайте и прочее» нужно изучить и для себя зафиксировать, так как преподаватель может спросить об этом при сдаче лабораторной работы.

(!) В качестве онлайн-справочников можно пользоваться http://javascript.ru/ или https://learn.javascript.ru/

(!) В качестве редактора кода можно воспользоваться Visual Studio или Code Studio (Создать файл, выбрать нужный тип .html или .js) или ресурсом https://jsfiddle.net/

(!) Для вывода результатов воспользуйтесь `console.log()`

Для выполнения заданий ниже вам понадобятся знания по созданию элементов формы.

Форма — это компонент веб-страницы с элементами управления, такими как текстовые поля, кнопки, флажки, диапазон или поле выбора цвета. Пользователь может взаимодействовать с такой формой, предоставляя данные, которые затем могут быть отправлены на сервер для дальнейшей обработки (например, возвращая результаты поиска или вычислений).
Написание формы состоит из нескольких шагов, которые могут выполняться в любом порядке: написание пользовательского интерфейса, реализация обработки на стороне сервера и настройка пользовательского интерфейса для связи с сервером.
В текущей лабораторной работе форма понадобится в рамках написания пользовательского интерфейса без связи с сервером.

Тег `<form>` устанавливает форму на веб-странице. Область применения форм не ограничена отправкой данных на сервер, с помощью клиентских скриптов можно получить доступ к любому элементу формы, изменять его и применять по своему усмотрению. Форм на странице может быть несколько, но формы не должны вкладываться одна в другую.

Тег `<input>` является одним из разносторонних элементов формы и позволяет создавать разные элементы интерфейса и обеспечить взаимодействие с пользователем. Главным образом `<input>` предназначен для создания текстовых полей, различных кнопок, переключателей и флажков. Хотя элемент `<input>` не требуется помещать внутрь контейнера `<form>`, определяющего форму, но если введенные пользователем данные должны быть отправлены на сервер, где их обрабатывает серверная программа, то указывать `<form>` обязательно. Аналогично, в случае обработки данных с помощью клиентских приложений, например, скриптов на языке JavaScript.
Основной атрибут тега `<input>`, определяющий вид элемента — `type`, который позволяет задавать следующие элементы формы:

- текстовое поле (`text`),
- поле с паролем (`password`),
- переключатель (`radio`),
- флажок (`checkbox`),

- скрытое поле (`hidden`),
- кнопка (`button`),
- кнопка для отправки формы (`submit`),
- кнопка для очистки формы (`reset`),
- поле для отправки файла (`file`),
- кнопка с изображением (`image`).

Для каждого элемента существует свой список атрибутов, которые определяют его вид и характеристики.
Кроме того, в HTML5 добавлено ещё более десятка новых элементов.
Для данного элемента доступна возможность добавления обработчика событий, описание которых можно посмотреть здесь.
Для знакомства работы с формами из кода javascript познакомьтесь со следующей статьей.

Также перед выполнением заданий ниже познакомьтесь с материалом как назначать обработчики событий на элементы веб-страницы: https://learn.javascript.ru/introduction-browser-events

Задание 1. Создайте страницу с калькулятором, у которого реализованы следующие операции: сумма, умножение, деление, а также перевод числа из 10-ой в 2-ую систему счисления и наоборот. Предусмотрите, чтобы калькулятор позволял записать выражение (со скобочками) и вычислить его значение (для этого можно воспользоваться конструкцией `new Function`).
Интерфейс калькулятора (кнопки и поле для выражений/вычислений) должен создаваться не в html-коде, а с помощью js-кода.

Задание 2. Создайте отдельную страницу с двумя текстовыми полями и кнопкой «Добавить». Первое текстовое поле предназначено для ввода *авторов книги*, второе – *названия книги*.

На странице есть список `<ul id="books">`, который изначально пустой. По нажатию кнопки «Добавить», книга попадает в список с `id="books"`, в котором отдельный пункт `<li>` описывает продукт «авторы книги» - «название книги».

Также рядом с каждым продуктом (внутри пункта) появляется кнопка «Удалить», позволяющая удалить книгу из списка. Также на странице есть текстовое поле и кнопка для поиска книги по автору или названию. Реализуйте нечёткий поиск то есть возможность искать как по точному названию/авторам, так и части.
Добавьте возможность менять тему страницы через CSS-стили (фон, рамки, оформление кнопок, списков и текстовых полей). Предусмотрите не менее трёх возможных тем страницы (например, тёмную/светлую и цветную).

Задание 3 Пусть есть массив строк, представляющих теги для статей блога.
Например:

```
const tags = ["JavaScript", "CSS", "HTML", "React", "JavaScript", "Node.js", "CSS"];
```

Выполните:

a) Удалить дубликаты из массива.
b) Создать все возможные пары тегов (например: ["JavaScript", "CSS"], ["JavaScript", "HTML"]).
c) Отсортировать пары по алфавиту.
d) Вывести результат в виде сетки на странице, где каждая ячейка содержит одну пару тегов.
e) Добавьте возможность добавлять новый тег через форму на странице, после чего сетка должна обновляться.

Задание 4. Создайте игру "Найди пару", где пользователь должен находить совпадающие карточки. Игровое поле состоит из сетки карточек, каждая из которых имеет лицевую сторону (изображение) и обратную сторону (закрытая карточка).
Реализуйте следующие функции:

a) Генерация игрового поля с заданным количеством карточек (например, 4x4).
b) Переворот карточек при клике (показ лицевой стороны). Если две перевернутые карточки совпадают, они остаются открытыми; если нет — они снова переворачиваются.
c) Подсчет количества ходов и времени игры.
d) Кнопка "Перезапустить игру", которая обновляет игровое поле.

Задание 5. Создайте страницу, на которой генерируются случайные цитаты. Цитаты хранятся в массиве объектов, где каждый объект содержит: текст цитаты, автора цитаты.
Реализуйте следующие функции:
a) Кнопка "Сгенерировать цитату", которая выбирает случайную цитату из массива и отображает ее на странице.
b) Возможность сохранять понравившиеся цитаты в отдельный список ("Избранное").
c) Возможность удалять цитаты из списка "Избранное".
d) При нажатии на цитату из списка «Избранное» она выделяется случайным цветом.