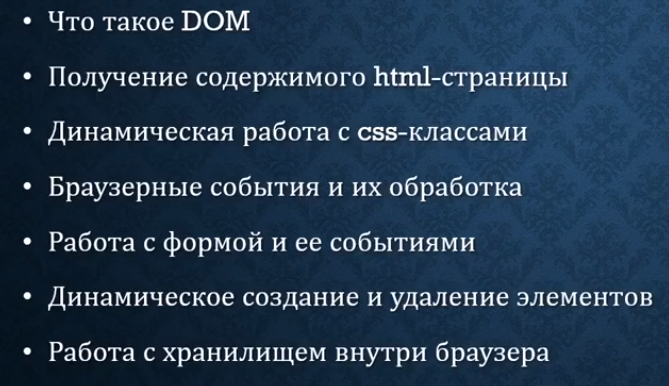
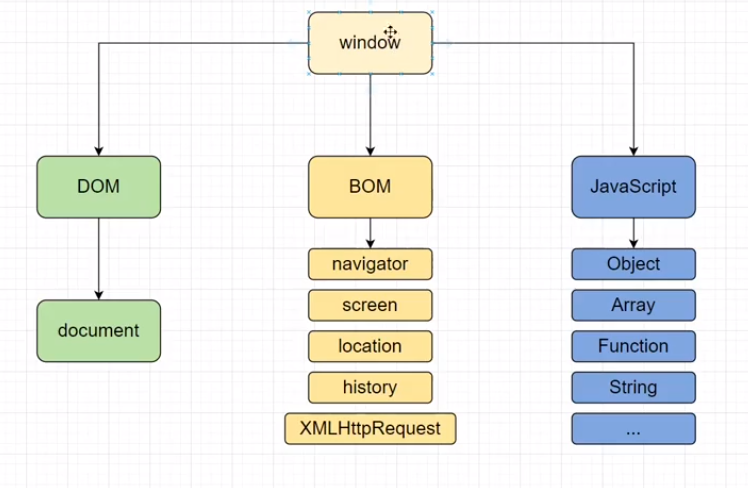
***JavaScript-курс по Stepik***

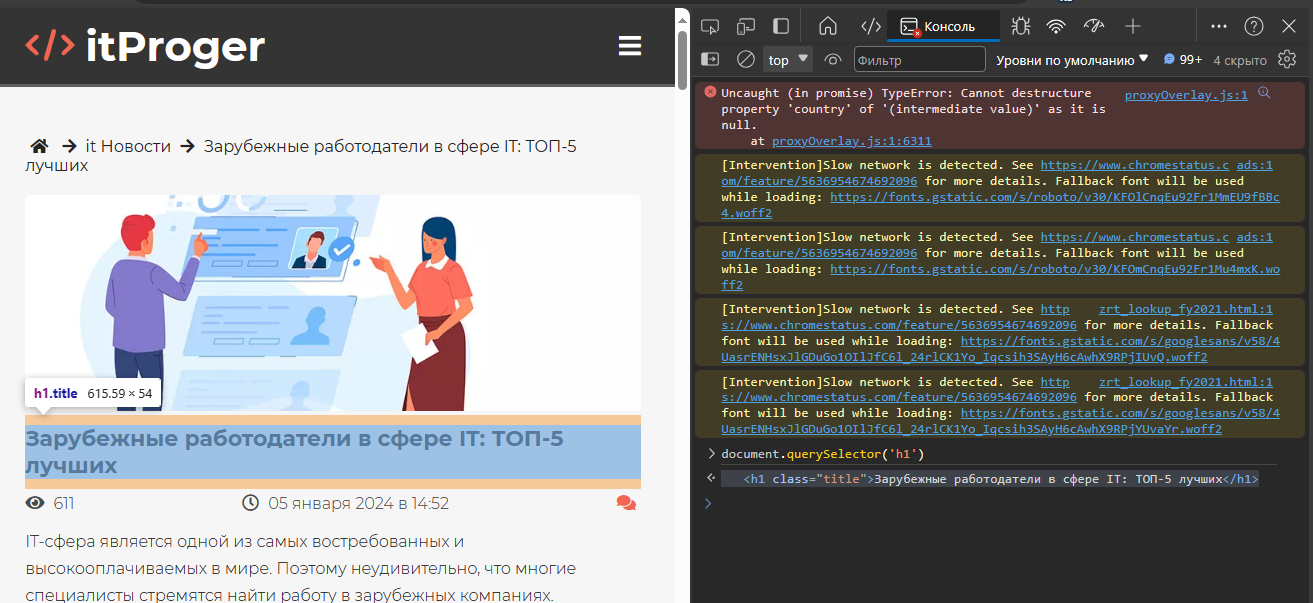
***Модуль DOM API 6***

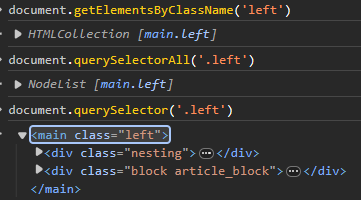
***document.querySelector(‘…’)***

***Выбор элементов на странице.***

***querySelectorAll(‘’) – список (коллекция) элементов на странице***

******

******

document.getElementById('everything-header') – выборка элемента по ID.

Если использовать getElements, то при некорректном названии мы получаем пустую коллекцию, если же в querySelector вводим несуществующий объект, то результат – null.

GetElementsbyTagName

**const** title = document.querySelector('h1') *//Берем элемент по тегу*

**const** btn = document.getElementById('btn') *//Берем по ID*

**const** subtitle = document.getElementsByClassName('title gray') *//Коллекция по ClassName*

**const** sections = document.getElementsByTagName('section') *//Коллекция по тегу*

console.log(title);

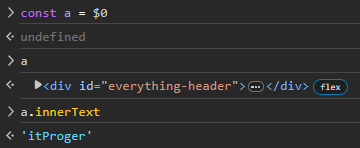
console.log(btn);

console.log(subtitle);

console.log(sections);

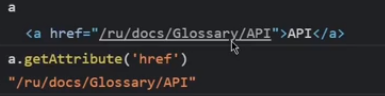
Другой способ:

\*\*\*const subtitle document.querySelector('.title.gray')\*\*\*

Обращение к элементам страницы

В консоли, прописав $0 (и предварительно выбрав нужный элемент html-страницы) мы можем обратиться к тому или иному элементу.

a.innerHTML – весь текст с учётом еще и html тегов

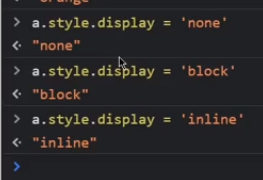
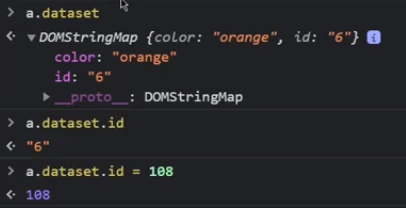


Свойство getAttribute берёт значение атрибута, найдённого в данном объекте (если не найден, то – null)

Разные способы получения атрибута – через функцию GetAttribute и через точку.

Если атрибут состоит из нескольких слов, то используем как правило camelCase – т.е. написание каждого нового слова с прописной буквы.

Обращение к for – принадлежащему к html. (Слово for само по себе зарезервировано).





Использование атрибутов, стилей;

Обращение к css, изменение цвета текста

Обращение к спец. атр. data-… через getAttribute или через dataset. …

1. \*\*Map\*\*: Функция map() применяет заданную функцию к каждому элементу массива и возвращает новый массив с результатами. Это позволяет изменять каждый элемент массива без изменения исходного массива.

const numbers = [1, 2, 3, 4, 5];

const squaredNumbers = numbers.map(num => num \* num);

console.log(squaredNumbers); // [1, 4, 9, 16, 25]

2. \*\*Filter\*\*: Функция filter() создает новый массив из элементов исходного массива, которые удовлетворяют условию, заданному в виде функции.

const words = ["apple", "banana", "grape", "orange", "peach"];

const result = words.filter(word => word.length > 5);

console.log(result); // ["banana", "orange"]

3. \*\*Reduce\*\*: Функция reduce() применяет функцию к аккумулятору и каждому значению из массива (слева направо), чтобы свести массив к одному значению.

const array = [1, 2, 3, 4, 5];

const sum = array.reduce((acc, val) => acc + val, 0);

console.log(sum); // 15

4. \*\*Foreach\*\*: Метод forEach() выполняет указанную функцию один раз для каждого элемента в массиве.

const numbers = [1, 2, 3, 4, 5];

numbers.forEach(num => {

console.log(num \* 2);

});

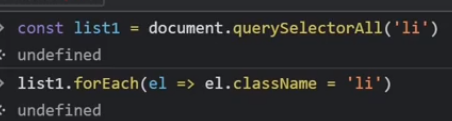
// Вывод:

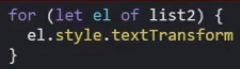
// 2

// 4

// 6

// 8

 // 10

Через querySelector мы получаем NodeList – которому доступны для использования forEach, reduce и другие функции.

Использование цикла for (в современном варианте (let el *of*list2) может работать даже в старых браузерах.

a.classList.add('newMyClass')

a.classList 🡺

DOMTokenList(3) ['lololoshka', 'newMyClass', 'someClass', value: 'lololoshka newMyClass someClass']

* a.classList.contains('fe')
* a.classList.remove('fe')

a.classList.toggle('some') //Своеобразный рычаг выкл/вкл.

* true

a.classList

DOMTokenList(4) ['lololoshka', 'newMyClass', 'someClass', 'some', value: 'lololoshka newMyClass someClass some']

a.classList.toggle('some')

* false

a.classList

DOMTokenList(3) ['lololoshka', 'newMyClass', 'someClass', value: 'lololoshka newMyClass someClass']

**Обращение к q.firstElementChild and q.lastElementChild**

q.children[1]

<h1 class=​"title">​Зарубежные работодатели в сфере IT: ТОП-5 лучших​</h1>​

q.children[1].innerText = 'Русские работодатели в сфере IT: ТОП-4 лучших'

'Русские работодатели в сфере IT: ТОП-4 лучших'

* q.children

1. *HTMLCollection(4) [div.main-image, h1.title, div.features, div.txt]*
   1. **0**: div.main-image
   2. **1**: h1.title
   3. **2**: div.features
   4. **3**: div.txt
   5. length: 4
   6. [[Prototype]]: HTMLCollection

**a.nextElementSibling.nextElementSibling** – получение следующего элемента (+2)

**q.previousElementSibling –** получение предыдущего элемента (-1)

**function fn1()** {

console.log('Hello World!')

}

<span class="mw-page-title-main" **onclick="fn1()"**>Карибский кризис</span>

Добавление обработчика клика на определённый блок.

a.onmouseenter = fn1

ƒ fn1() {

console.log('Hello World!')

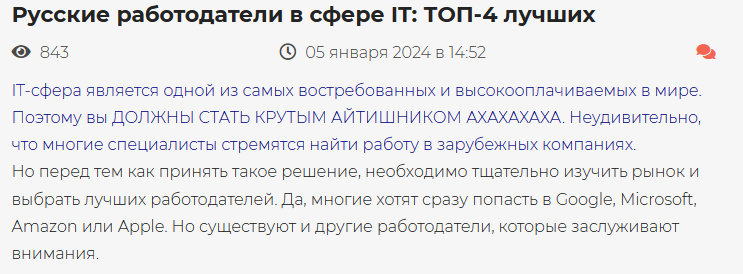
}

Назначение функции для определённого события.

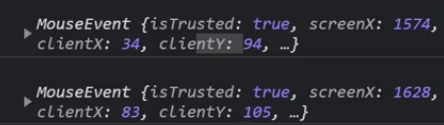
(Добавление событий вручную, напрямую через html-код, является не очень хорошей практикой (позже, при работе с фреймворками, можно будет решить эту проблему).

a.AddEventListener( ‘click’, fn1) -добавление обработчика событий



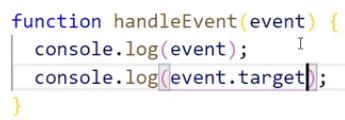


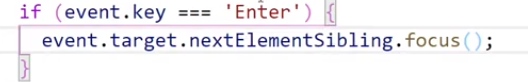
a.removeEventListener(***‘click’***, fn1) – снятие обработчика событий



btn.addEventListener(***‘keypress’***, fn2)

***Target*** – то место, куда мы нажали или где работали (target = inputField, target = btn, etc.)





**const** elements = document.querySelectorAll('.accordion')

elements.forEach(el **=>** el.addEventListener('click', fn1))

console.log(elements);

**function** fn1(){

    console.log('Hello');

    this.nextElementSibling.classList.toggle('show') //Обращение к стилю .panel.show

} //добавление данного класса к html-объекту

a.removeEventListener(***‘click’***, fn1) – снятие

**const** elements = document.querySelectorAll('button')

elements.forEach(el **=>** el.addEventListener('click', fn1))

console.log(elements);

**function** fn1(event) {

**let** counter = Number(event.target.dataset.clicked)

    counter += 1

    event.target.dataset.clicked = String(counter)

*// counter += 1*

*// event.target.setAttribute('data-clicked', counter)*

*// console.log(event.target.dataset.clicked);*

*// console.log(counter);*

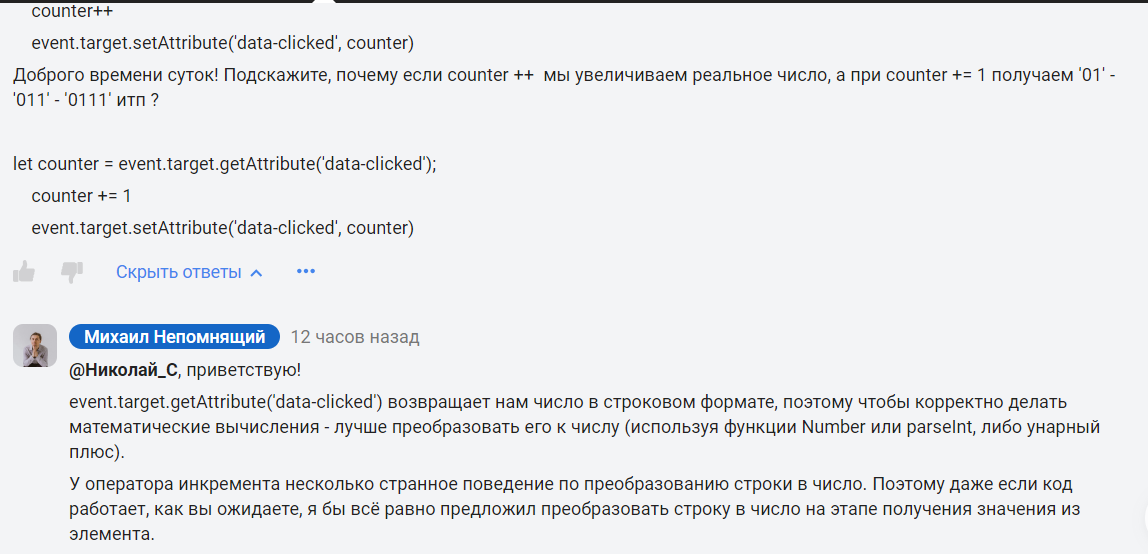
*// console.log(event.target);*

*// event.target.dataset.clicked++*

    console.log(event.target.dataset.clicked);

}

Обращение через:

1. event.target.dataset.clicked ( где clicked – инкремент)
2. setAttribute() // getAttribute ()
3. let counter = Number (…)

input.value.trim()– проверка строки, с предварительно удалёнными пробелами (value – значение строки, trim() – удаление лишних пробелов).

Если строка не пустая, то значение будет истинно.

**const** inputs = document.querySelectorAll('input')

inputs.forEach(el **=>** el.addEventListener('keypress', fn))

**const** form = document.querySelector('form')

form.addEventListener('submit', formSubmit)

**function** fn(event) {

    if(event.key === "Enter")

    {

        event.preventDefault(); *//По умолчанию форма отправляется,*

*//И чтобы этого не допустить, отменяем действие по умолчанию*

        event.target.nextElementSibling.focus();

    }

}

**function** formSubmit(event){

    event.preventDefault();

    if(validate()==true){

*// submit AJAX*

        alert('Access!');

        form.reset(); *// - очистка полей формы*

    } else {

        alert('Fix empty fields');

    }

}

**function** validate() {

**let** isValid = true;

    inputs.forEach(input **=>** {

        if(!input.value.trim()){

            input.style.borderColor = 'red';

            isValid = false

        } else {

            input.style.borderColor = 'black';

        }

    })

    return isValid

}

Задание – проверка логина и пароля от пользователя

**const** registeredUsers = [

    ['admin', 'KoI18'],

    ['manager', 'SuperMe108'],

    ['editor', '12345'],

];

**const** input1 = document.querySelector('input')

**const** input2 = input1.nextElementSibling;

**const** form = document.querySelector('form')

input1.addEventListener('keypress', enterFunction)

input2.addEventListener('keypress', enterFunction)

form.addEventListener('submit', submitFunction)

**function** enterFunction(event){

    if(event.key == 'Enter') {

        event.preventDefault();

        event.target.nextElementSibling.focus();

    }

}

**function** submitFunction(event){

    event.preventDefault();

    if(!input1.value.trim() || !input2.value.trim()) {

        alert("All fields are required");

    } else if((input1.value == registeredUsers[0][0] && input2.value == registeredUsers[0][1])

        || (input1.value == registeredUsers[1][0] && input2.value == registeredUsers[1][1])

        || (input1.value == registeredUsers[2][0] && input2.value == registeredUsers[2][1])) {

            alert('Access granted');

            form.reset();

            location.reload(); *// Перезапуск страницы*

    }

    else {

        alert("Incorrect login or password")

        form.reset();

    }

}

Понятие всплытия

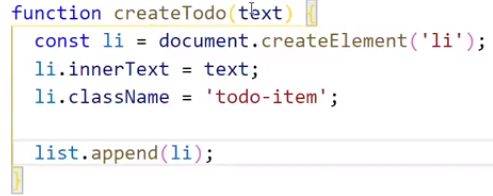
*bubbles: true – параметр у event*

event.StopPropagation(); – отмена всплытия

event.currentTarget – показывает, где именно сработала та функция

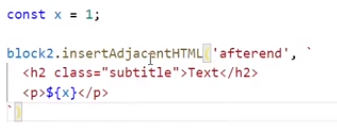
event.target – просто показывает объект, на который мы кликнули

(С учётом всплытия)\*

Мы можем искать какую-то сущность внутри всего документа, однако если мы ищем внутри уже найденного объекта, то мы не можем использовать:

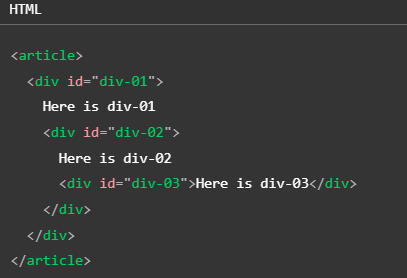
* Block2 = block1.getElementById(‘…’);
* Block2 = block1.getElementByTagName(‘…’);



Получение геометрических параметров; (положение, ширина, высота).

*Динамическое редактирование html-кода страницы*

Использование метода closest():

const el = document.getElementById("div-03");

// the closest ancestor with the id of "div-02"

console.log(el.closest("#div-02")); // <div id="div-02">

// the closest ancestor which is a div in a div

console.log(el.closest("div div")); // <div id="div-03">

// the closest ancestor which is a div and has a parent article

console.log(el.closest("article > div")); // <div id="div-01">

// the closest ancestor which is not a div

console.log(el.closest(":not(div)")); // <article>

Работа с открытием/закрытием модального окна через: escape, cross click, outside click;

**const** btn = document.querySelector('button');

**const** modal = document.getElementById('myModal');

btn.addEventListener('click', openModalWidnow)

**function** openModalWidnow() {

    modal.classList.add('open'); *//Добавляем класс open - display: block;*

    addEvents();

}

**function** addEvents() {

    modal.querySelector('.close').addEventListener('click', closeModal);

    modal.addEventListener('click', outsideClick)

    document.addEventListener('keydown', handleClose)

}

**function** handleClose(event){

    if(event.key === "Escape") {

        closeModal();

    }

}

**function** outsideClick(event){

**const** isClick = !!event.target.closest('modal-content');

*// ! - true, если объект = null; !! - false, если объект - null*

    if(isClick == false)

    {

        closeModal();

    }

}

**function** closeModal(){

    modal.classList.remove('open');

    detachModalEvents();

}

**function** detachModalEvents(){

    modal.querySelector('.close').removeEventListener('click', closeModal);

    modal.removeEventListener('click', outsideClick);

    document.removeEventListener('keydown', handleClose); *//Очищение памяти -*

*// Удаление closeModal и handleClose - после того, как мы прикрепили эти функции*

*// к соотв.объектам; Чтобы события далее не обрабатывались*

}

index.html

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <link rel="stylesheet" type="text/css" href="main.css">

    <title>Form</title>

</head>

<body>

*<!-- Trigger/Open The Modal -->*

<button id="myBtn">Open Modal</button>

*<!-- The Modal -->*

<div id="myModal" class="modal">

*<!-- Modal content -->*

  <div class="modal-content">

    <span class="close">&times;</span>

    <p>Some text in the Modal..</p>

  </div>

</div>

    <script src="app.js"></script>

</body>

</html>

Main.css

@import url('https://fonts.googleapis.com/css2?family=Kalam:wght@300&family=Merriweather:ital,wght@1,300&display=swap');

*/\* The Modal (background) \*/*

.modal {

  display: none; */\* Hidden by default \*/*

  position: fixed; */\* Stay in place \*/*

  z-index: 1; */\* Sit on top \*/*

  left: 0;

  top: 0;

  width: 100%; */\* Full width \*/*

  height: 100%; */\* Full height \*/*

  overflow: auto; */\* Enable scroll if needed \*/*

  background-color: rgb(0,0,0); */\* Fallback color \*/*

  background-color: rgba(0,0,0,0.4); */\* Black w/ opacity \*/*

}

*/\* Modal Content/Box \*/*

.modal-content {

  background-color: cornflowerblue;

  border-radius: 10px;

  font-size: 20px;

*/\* border: 10px solid yellow; \*/*

  color: aliceblue;

  margin: 15% auto; */\* 15% from the top and centered \*/*

  padding: 20px;

  border: 1px solid #888;

  width: 80%; */\* Could be more or less, depending on screen size \*/*

}

*/\* The Close Button \*/*

.close {

  color: #aaa;

  float: right;

  font-size: 28px;

  font-weight: bold;

}

.open {

  display: block;

}

button {

  margin: 0 auto;

  display: block;

  margin-top: 10%;

  height: 5rem;

  width: 20rem;

  font-size: 30px;

  background-color: rosybrown;

  color: beige;

  font-weight: 600;

  font-family: 'Merriweather', serif;

  border-radius: 10px;

  border-width: 4px;

  border-color: wheat

}

.close:hover,

.close:focus {

  color: black;

  text-decoration: none;

  cursor: pointer;

}

Local Storage and Session Storage

* localStorage.clear()
* undefined
* localStorage.setItem('todos','123')
* undefined
* localStorage.getItem('todos')
* '123'

Local Storage and Session Storage

**function** clearLocalStorage() {

**const** text = this.innerText;

**let** tasks = JSON.parse(localStorage.getItem('tasks'));

    tasks.pop(text);

    localStorage.setItem('tasks', JSON.stringify(tasks));

}

- Удаление элемента из local storage. С мутацией

**const** text = this.innerText;

**const** tasks = JSON.parse(localStorage.getItem('tasks'))

**const** tasks2 = tasks.filter(el **=>** el !== text); *//Получаем новый массив*

    localStorage.setItem('tasks', JSON.stringify(tasks2));

- Удаление элемента из local storage. Без мутаций, с использованием фильтрации.

Порядок удаления объекта (li из нашего списка):

**function** removeTodo() {

    this.removeEventListener('click', removeTodo); *//Удаляем событие click*

    this.remove(); *//Удаляем сам элемент li*

}

history.forward() history.back() – передвижение по истории браузера

location.reload('...') - загрузка страницы

location.search.split(‘&’) – разделение элементов по ключам

***Проект Спойлер***

**app.js**

**const** button = document.getElementById('myBtn');

**const** element = document.getElementById('spoiler');

button.addEventListener('click', openCloseSpoiler);

**function** openCloseSpoiler() {

**let** choice = element.classList.toggle('closed');

    console.log(element.classList);

    if (choice === false) {

*// console.log("u there");*

        document.addEventListener('keydown', closeWindow);

    }

*//Когда элемент скрыт,то есть в класс-листе присутствует*

*// closed, тогда его значение = true*

}

**function** closeWindow(event) {

    if(event.key === 'Escape') {

        element.classList.toggle('closed');

    }

    document.removeEventListener('keydown', closeWindow);

}

**HTML - разметка**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

*<!-- указание режима совместимости для Internet Explorer -->*

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

*<!-- определение параметров просмотра страницы на устройствах с*

*различными размерами экранов -->*

    <link rel="stylesheet" type="text/css" href="main.css">

    <title>Shop</title>

    <style>

        :root {

            color: #222;

            font-family: Arial;

        }

        button {

            padding: 0.5rem 1rem;

            background-color: burlywood;

            border: none;

            border-radius: 5px;

        }

        button:hover {

            background-color: cadetblue;

        }

        div {

            margin-top: 4%;

            text-align: center;

            padding: 1rem;

            border: 4px solid rebeccapurple;

            border-radius: 10px;

            background-color: gainsboro;

        }

        .closed {

            display: none;

        }

    </style>

</head>

<body>

  <button id="myBtn">Toggle spoiler</button>

  <div id="spoiler" class="closed">

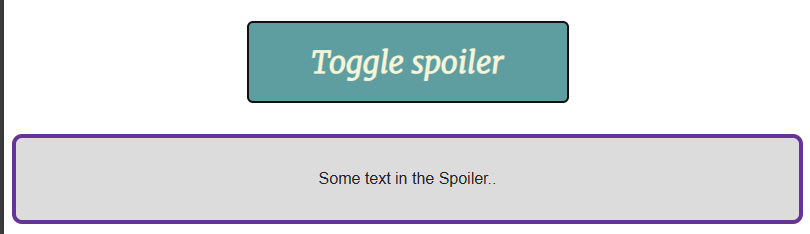
      <p>Some text in the Spoiler..</p>

  </div>

      <script src="app.js"></script>

</body>

</html>



***Код преподавателя***