**Получение элементов со страницы**

1.методы

Const box = Document.getElementById(‘записываем название индентификатора с html’);

-По правилам id может находиться только 1 на странице в html

Можем получать через название тэгов

Тэгов на странице может быть сотни

Пример у нас есть 5 одинаковых кнопок:

Const btns = document.getElementsByTagName(‘написать название тэга который я хочу найти’);

МЫ получили не 1 объект а массив тоесть [button, button, button, button, button]

Чтобы получить каждую отдельную кнопку которая нас интересует:

Const btns = document.getElementsByTagName(‘button’)[1]; выбираем кнопку по порядку в массиве которую хотим получить

А когда хочу ее использовать

Console.log(btns[1]); один конкретный элемент

Можем получать через классы

Const circles = document.getElementsByClassName(‘circle’);пишем название класса не ставим .(точку) это ошибка, этот метод и так знает что ищется класс

Console.log(‘circle’); и получаем html коллекцию массив [div.circle, div.circle, div.circle]

**Обращать внимание на название элемента в конце стоит s это значит что элементы находит а не 1 элемент!!!**

**Более современные методы**

Const hearts = document.querySelectorAll(‘.headt’);

Тут уже мы ставим точку

Console.log(hearts);

получаем html коллекцию

Внутрь помещаем css селектор абсолютно любой это может быть:

1. Уникальный идентификатор через #
2. Просто класс
3. Вложенность классов
4. Псевдоклассы
5. Атрибуды комбинации всего этого

**Самый большой плюс этого метода это forEach**

**Hearts.forEach(item => {**

**Console.log(item);**

**});**

**И мы получаем все дивы(или что там будет) с этим классом**

**Item это каждый элемент который будет находиться в hearts**

**Const oneHeart = document.querySelector(‘.heart’);**

**Console.log(oneHeart);**

**Позволяет получить 1 элемент со страницы и тот элемент который попал первым под сочетание этого селектора на странице тот и приходит в переменную oneHeart**

**Это может быть первый div span h1 и тд**

**Удобнее работать с уникальный css селектором**

**Действия с элементами на странице**

Обращаемся к предыдущим элементам

Box.style.backgroundColor = ‘blue’; тут мы меняем цвет писать стили в ‘’

Box.style.width = ‘500px’;

И у нас в браузере появляется синий прямоугольник с шириной 500px

Btns[1].style.borderRadius = ‘100%’;

И мы видим что к нашей кнопки применился радиус

**Самая частая ошибка**

circles.style.backgroundColor = ‘red’;

не чего не поменяется потому что будет ошибка, все потому что пытаюсь обратиться не к какомуто объекту а к псевдомассивумассиву объектов, а он совершенно не знает об объекте style

circles[0].style.backgroundColor = ‘red’; нужно четко указать элемент на котором будет это происходить

и тогда наш кружек покрасится в красный цвет

**Назначение нескольких стилей**

CssText

Box.style.cssText = ‘background-color: blue; width: 500px’;

И мы упрощаем наш код не нужно несколько раз обращаться то к цвету то к размеру и тд

Этот метод на практике позволяет быстро сформировать инлайн стили, при чем сюда мы так же можем подставлять какие то переменные например:

Box.style.cssText = `background-color: blue; width: ${num}px`;

Это полезно если мы определенные параметры расчитываем динамически например ширину какого то модального окна в зависимости от устройства с которого зашел пользователь

**Над несколькими элементами произвести одни и теже действия**

Перебирающие методы

Hearts.forEach(item => {

Item.style.backgroundColor = ‘blue’;

});

И наш бэкгроунд сердечек поменялся он стал синим

**Основные методы для работы с элементами страницы**

Метод document.createElement();

Const div = document.createElement(‘div’); внутрь помещаем тэг который создаем

Пока что он существует только в js

Модифицируем css классы элементов можем добавить можем удалить

**В реальных проэктах при работе со стилями буду работать с css классами**

**Можем через точку писать действия с нашим классом это удаление, добавление, переключение, проверка содержания или даже определение количества применения классов к элементу**

**div.classList.add(‘black’);**

**где: div(наш блок).classList. указать действие которое будет производиться тоесть это метод на свойстве classList add это добавить**

**И так нам нужно наш див бовавить в конец body для этого делаем**

**Document.body.append(div);**

**В результате наш див в конце боду**

**Мы также можем вставить в любом месте родителя**

**Например у нас есть родитель wrapper и в нем 3 элемента и мы хотим вставить в конец делаем:**

**Document.querySelector(‘.wrapper’).append(div);**

**И внутри этого родителя в конце мы получаем этот див**

**Вставить в начало:**

**Wrapper.prepend(div);**

**И мы получаем в начале наш див**

**Методы before and after**

**Hearts[0].before(div);**

**Можем удалять элементы со странице**

**Circles[0].remove();**

**Метод чтоб один элемент заменить другим:**

**Hearts[0].replaceWith(circles[0]);**

**Сначало пишем какой элемент мы хотим заменить а потом на какой**

**Редактирование элемента будем вписывать простой текст**

**1 вариант:**

**div.innerHTML = “Hello World”;**

**и в нашем черном обекте появляется текст**

**Можно еще в объект вставлять структуру HTML**

**div.innerHTML = “<h1>Hello World</h1>”;**

**2 вариант:**

**div.textContent = “Hello”;**

**он работает только с текстом**

**Комбинация всех методов которые описаны выше**

**Мы вставляем кусочек HTML перед или после определенных тегов**

**Сначало мы указываем элемент над которым мы будем производить действия**

**Этот метод принимает 2 аргумента**

**1.beforebegin – перед элементом**

**Afterbegin – в начало нашего элемента (перед началом)**

**Beforeend – в конец нашего элемента (перед концом)**

**Afterend – вставляет после элемента**

**2 тот HTML который мы хотим вставить**

**div.insertAdjacentHTML(‘ ’, ‘<h2>Hello</h2>’);**

Таким же образом мы можем создавать текстовые узлы это элементы без оболочки тэга

Const text = document.createTextNode(‘Тут был я’); во внутрь помещаем тот текст который хотим поместить

И такой текст появится у нас на странице **Этот метод используется редко**