**Конспект «Коллекции и свойства элементов»**

**Метод querySelectorAll**

Метод querySelectorAll находит все элементы на странице, которые соответствуют указанному селектору, и возвращает **коллекцию** — набор этих элементов.

// Найдёт все абзацы на странице

let elements = document.querySelectorAll('p');

**Коллекция**

Коллекцию можно сохранить в переменной. Самый простой способ узнать, какие элементы содержит коллекция, — вывести её в консоль:

// Выведет коллекцию в консоль

console.log(elements);

В консоли коллекция выглядит как список, в котором элементы перечислены через запятую. Весь список обёрнут в квадратные скобки, а у элементов указан только их тег и, например, класс. Чтобы элементы отобразились так же, как в разметке, коллекцию нужно развернуть, кликнув на стрелку-треугольник слева.

[p.card\_\_text, p, p]

<p class="card\_\_text">Готовим мороженое!</p>

<p>Санкт-Петербург</p>

<p>mail@htmlacademy.ru</p>

К элементу коллекции можно обращаться по индексу. **Индекс** — это порядковый номер элемента в коллекции. Отсчёт начинается с нуля, поэтому у первого элемента индекс 0, а у второго — 1. Индексы пишут в квадратных скобках после имени коллекции:

console.log(elements[0]); // Выведет первый элемент коллекции

console.log(elements[1]); // Выведет второй элемент коллекции

**Data-атрибуты**

В HTML можно создавать свои собственные атрибуты. Имена таких атрибутов начинаются с префикса data-, после которого идёт любое выбранное разработчиком слово.

<div data-cat-name="Кекс">

Чтобы получить значение data-атрибута в JavaScript, используют свойство dataset, после которого указывают имя атрибута без префикса data-:

элемент.dataset.имяАтрибутаБезПрефикса

Если имя атрибута состояло из нескольких слов и в нём были дефисы, то в JavaScript его записывают в «верблюжьем» стиле (по-английски *camelCase*): дефисы убирают, а каждое слово, кроме первого, пишут с большой буквы.

let element = document.querySelector('div');

console.log(element.dataset.catName); // Выведет: Кекс

**Цикл for of**

Цикл — это конструкция, которая позволяет выполнить код несколько раз. Цикл for of выполнит код из фигурных скобок столько раз, сколько элементов содержится в коллекции, указанной в круглых скобках. Каждое такое повторение называется *итерацией*.

for (переменная of коллекция) {

// Код, который нужно выполнить несколько раз

}

При создании цикла в круглых скобках также нужно указать переменную. Обычно для этого объявляют новую переменную и используют её только внутри цикла. На каждой итерации JavaScript будет автоматически записывать в эту переменную очередной элемент коллекции.

let elements = document.querySelectorAll('p'); // Находим все абзацы

for (let element of elements) { // Создаём цикл и переменную

console.log(element); // Выводим элементы в консоль

}

Цикл for of завершится, когда в коллекции закончатся элементы. После этого JavaScript перейдёт к инструкциям, которые идут после цикла.

**Обработчик событий oninput**

Обработчик событий oninput (в переводе с английского это означает «при вводе») позволяет выполнять инструкции из фигурных скобок каждый раз, когда меняется значение в поле ввода. Изменением считается и добавление, и удаление символов.

// Найдём поле ввода

let textarea = document.querySelector('textarea');

// Добавим обработчик событий

textarea.oninput = function () {

// Выведем данные из поля ввода

console.log(textarea.value);

};

**Свойство length**

Узнать длину строки можно с помощью свойства length (по-английски «длина»). Значение этого свойства равно числу символов в строке. Символами считаются не только буквы и цифры, но также пробелы и переносы строки.

let text = 'Я люблю JavaScript';

console.log(text.length); // Выведет: 18

let textarea = document.querySelector('textarea');

console.log(textarea.value); // Выведет: Кекс

console.log(textarea.value.length); // Выведет: 4

**Оператор сравнения >**

Оператор сравнения > («больше») сравнивает два числа и возвращает булево значение: true, если левое число больше правого, и false во всех остальных случаях:

console.log(3 > 2); // Вернёт: true

console.log(1 > 2); // Вернёт: false

console.log(2 > 2); // Вернёт: false

**Свойство disabled**

Блокировать и разблокировать кнопку в JavaScript можно, присваивая булевы значения свойству disabled (по-английски значит «отключён») этой кнопки. Если присвоено значение true, то кнопка заблокирована, а если false — разблокирована.

let button = document.querySelector('button');

// Блокирует кнопку

button.disabled = true;

// Разблокирует кнопку

button.disabled = false;