**Конспект «Условия и создание элементов»**

**Как подключить на страницу несколько скриптов**

Чтобы подключить на страницу ещё один файл с кодом на JavaScript, снова используем тег script:

<body>

<script src="themes.js"></script>

<script src="likes.js"></script>

</body>

Браузер обрабатывает инструкции последовательно: сначала в первом файле, потом во втором — как будто они все находятся в одном месте. Программы часто разбивают на несколько файлов, обычно одному файлу соответствует одна задача: например, управление темами или подписка на рассылку.

**Числа**

Переменной можно присвоить значение — число. Числа не нужно оборачивать в кавычки:

let number = 0;

Чтобы увеличить или уменьшить число в JavaScript, можно использовать разные записи:

// Полная запись

number = number + 2; // Значение переменной: 2

number = number - 2; // Значение переменной: 0

// Краткая запись

number += 2; // Значение переменной: 2

number -= 2; // Значение переменной: 0

// Увеличение числа на 1

number++; // Значение переменной: 1

// Уменьшение числа на 1

number--; // Значение переменной: 0

**Метод classList.contains**

Метод classList.contains проверяет, есть ли у элемента класс:

элемент.classList.contains('класс');

Метод вернёт true (истина), если класс у элемента есть, и false (ложь), если класса нет. Значения true и false называют булевыми. Таких значений всего два.

**Условная конструкция**

Условная конструкция позволяет выполнять действия в зависимости от *условия*. Условие — это инструкция, которая возвращает true или false. Выглядит условная конструкция так:

if (условие) {

// Инструкции, которые выполнятся, если условие истинно

} else {

// Инструкции, которые выполнятся, если условие ложно

}

if

Условие записывают в *круглых* скобках после слова if (переводится с английского как «если»). После этого внутри *фигурных* скобок пишут инструкции, которые выполнятся, если условие *истинно*. Условие считается истинным, если инструкция внутри круглых скобок возвращает true.

else

Конструкция else (переводится как «иначе») говорит JavaScript, что делать, если условие ложно. Внутри фигурных скобок после else пишут инструкции, которые должны выполниться, если условие вернёт false.

Использование условных конструкций в скрипте ещё называют ветвлением, а код внутри фигурных скобок — веткой.

**Метод append**

элемент-родитель.append(добавляемый-элемент);

Метод append добавляет указанный в скобках элемент в конец элемента-родителя. При этом содержимое элемента-родителя не затирается. Добавлять с помощью этого метода можно и элементы, и простые строки.

**Метод createElement**

document.createElement('имя тега');

Чтобы создать новый элемент на странице, к которой подключён скрипт, нужно использовать слово document. Внутри скобок в кавычках указывают элемент, который нужно создать. Например:

// Создаём новый элемент <div> и записываем его в переменную

let newElement = document.createElement('div');

Новый элемент будет доступен из скрипта, но в разметке не появится, пока мы не скажем JavaScript, где разместить новый элемент. Для этого можно использовать метод append:

// Находим элемент-родитель

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

// Добавляем новый элемент на страницу

parent.append(newElement);

Элементы, созданные с помощью метода createElement, можно изменять так же, как и любые другие.

**Очищаем поле ввода**

Чтобы пользователь по ошибке не отправил форму несколько раз, после отправки поле ввода лучше очистить. Для этого в его свойство value записывают пустую строку. Вот так:

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

input.value = '';

# Конспект «JS в браузере». Раздел 2

Свойство DOM-элемент.children возвращает коллекцию дочерних, то есть вложенных, DOM-элементов.

Создание элемента и добавление его в DOM-дерево:

var list = document.querySelector('.cards');

// Создаём новый элемент

var card = **document.createElement('li')**;

card.classList.add('card');

// После вызова этого метода новый элемент отрисуется на странице

**list.appendChild(card)**;

Вот что произойдёт с разметкой после вызова appendChild:

*<!-- Исходное состояние разметки -->*

<ul class="cards">

<li class="card">Существующий элемент</li>

</ul>

*<!-- Состояние после вызова appendChild -->*

<ul class="cards">

<li class="card">Существующий элемент</li>

<li class="card">Добавленный элемент</li>

</ul>

Работа с текстовым содержимым элемента:

// HTML

<p>Я — <em>текстовый элемент</em>.</p>

// JS

var p = document.querySelector('p');

console.log(**p.textContent**);

// Выведет: Я — текстовый элемент.

**p.textContent** = 'Теперь у меня новое содержимое.';

console.log(p.textContent);

// Выведет: Теперь у меня новое содержимое.

// В HTML содержание тега изменится

<p>Теперь у меня новое содержимое.</p>

Работа с изображениями:

// Создание изображения

var picture = document.createElement('img');

// Добавляем адрес картинки

picture.src = 'images/picture.jpg';

// Добавляет альтернативный текст

picture.alt = 'Непотопляемая селфи-палка';

// Добавляет изображение в конец родительского элемента

element.appendChild(picture);