Полная лекция: JavaScript, DOM и обмен с backend

## Введение в JavaScript

JavaScript (JS) — это язык программирования, который выполняется в браузере. Он позволяет веб-страницам реагировать на действия пользователя, изменять содержимое, обрабатывать события.  
JS — единственный язык, который браузеры понимают «из коробки» на стороне клиента. На сервере могут использоваться Python, PHP, Node.js, Java и другие языки.

## Подключение JavaScript к HTML

JavaScript можно подключить:  
1. Встроенно (inline): <script>alert('Привет');</script>  
2. Внутри <head> или <body> страницы  
3. Через внешний файл: <script src="script.js"></script>  
Для обучения подойдёт встроенный, но в реальных проектах лучше использовать внешний файл.

## Переменные: let, const, var

- let — переменная, которую можно переопределять (менять значение).  
- const — переменная, которую нельзя переопределить (константа).  
- var — устаревший способ, имеет функциональную область видимости, не блочную. Лучше не использовать.  
  
Пример:  
let name = "Иван";  
const year = 2025;  
var legacy = true;  
  
Сейчас рекомендуется использовать только let и const.

## Типы данных в JS

- Числа: 10, -5, 3.14  
- Строки: "текст", 'ещё текст'  
- Логические (boolean): true, false  
- Null и undefined — специальные значения (пустота и отсутствие значения)  
  
Примеры:  
let age = 30;  
let name = "Анна";  
let active = true;

## Операторы сравнения

- > — больше  
- < — меньше  
- >= — больше или равно  
- <= — меньше или равно  
- == — сравнение по значению (может преобразовать типы)  
- === — строгое сравнение по значению и типу  
- != — не равно (может преобразовать типы)  
- !== — строго не равно  
  
Пример:  
5 == "5" // true (сравнение значений)  
5 === "5" // false (разные типы)  
true !== false // true

## Логические операторы

- && — И: true, если оба условия true  
- || — ИЛИ: true, если хотя бы одно условие true  
- ! — НЕ: инвертирует значение  
  
Пример:  
if (age > 18 && active) { ... }

## Ввод и вывод

Для вывода используют:  
- alert("Привет!") — модальное окно для пользователя  
- console.log("текст") — вывод в консоль (F12 → Console)  
- document.write("прямо на страницу") — редко используется  
  
Рекомендуется использовать console.log для отладки.

## Условия (if/else)

Пример:  
if (age >= 18) {  
 console.log("Совершеннолетний");  
} else {  
 console.log("Несовершеннолетний");  
}  
Можно использовать else if для дополнительных проверок.

## Циклы

Пример цикла for:  
for (let i = 1; i <= 5; i++) {  
 console.log(i);  
}  
Также есть:  
- while (условие) { ... }  
- do { ... } while (условие);  
Обычно for используется, если известно количество повторений.

## Функции

Функция — именованный блок кода, который можно вызывать много раз:  
function greet(name) {  
 return "Привет, " + name;  
}  
console.log(greet("Катя"));  
  
Функции можно записывать стрелочным способом:  
const greet = (name) => "Привет, " + name;

## DOM: работа с HTML из JS

DOM — это структура HTML в виде дерева объектов. Через JS можно:  
- Найти элемент: document.getElementById("id"), document.querySelector(".class")  
- Изменить содержимое: element.innerText = "новый текст"  
- Изменить стиль: element.style.color = "red"

## События и обработка действий пользователя

Можно назначить обработчик события (например, нажатие):  
document.getElementById("btn").addEventListener("click", function() {  
 alert("Клик!");  
});  
События бывают: click, input, change, submit и другие.

## Инструменты разработчика

Открываются клавишей F12 или ПКМ → Инспектировать:  
- Elements — структура HTML  
- Console — выполнение JS, просмотр ошибок  
- Sources — отладка JS (breakpoints)  
- Network — просмотр сетевых запросов (важно при обмене с сервером)

## Отладка JavaScript

- Ошибки видны в Console (например, "undefined is not defined")  
- Можно ставить точки останова в Sources → выполнение остановится  
- Можно отслеживать переменные и шаги выполнения (Step Over, Step Into)

## Обмен данными с сервером (введение)

Иногда JS-код на странице должен получать или отправлять данные на сервер.  
  
Основные протоколы:  
- HTTP (чаще всего используется)  
- HTTPS (защищённый вариант HTTP)  
- WebSocket (двустороннее общение в реальном времени)  
  
Чаще всего используется HTTP/HTTPS. Запросы отправляются с помощью:  
- fetch (встроенный способ)  
- axios (библиотека)  
  
Пример с fetch:  
fetch("https://api.example.com/user")  
 .then(response => response.json())  
 .then(data => console.log(data));  
  
Данные обычно передаются в формате JSON. Инструменты разработчика → вкладка Network покажет все запросы и ответы от сервера.  
Это основа клиент-серверного взаимодействия. Мы подробно разберём её в следующем уроке.

## Домашнее задание

- Написать JS-код, который:  
 • выводит в консоль ваше имя  
 • считает сумму двух чисел  
 • проверяет совершеннолетие  
 • выводит чётные числа от 1 до 10  
- По желанию: создать HTML с кнопкой, JS при нажатии должен показать alert с текстом из input  
- Дополнительно: открыть DevTools → Network и посмотреть, какие запросы делает любой сайт