АО «АЛМАТИНСКИЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»
https://atu.edu.kz/

Факультет «Информационные технологии»
Кафедра «Компьютерная инженерия»

Название курса: «Разработка клиентских Web-приложений»

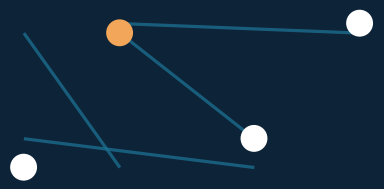


Зиятбекова Гулзат Зиятбеккызы – PhD, ассоциированный профессор
g.ziyatbekova@atu.edu.kz

ОСНОВЫ ВЕБ-ТЕХНОЛОГИЙ

ЛЕКЦИЯ 1

Введение в клиентские веб-приложения и архитектуру веб-технологий



ПЛАН ЛЕКЦИИ

Что мы рассмотрим сегодня?

1. Понятие веб-приложения
2. История развития веб-приложений
3. Понятие клиентского веб-приложения
4. Архитектура клиент–сервер
5. Основные компоненты веб-приложения
6. Роль браузера
7. Веб-сервер и серверные программы
8. Протоколы HTTP и HTTPS
9. Технологии HTML5, CSS3 и JavaScript
10. Особенности современных клиентских веб-приложений

1. ОСНОВНОЕ ПОНЯТИЕ

Что такое веб-приложение?

Веб-приложение (Web Application) — программная система, которая выполняется в веб-браузере через интернет или компьютерную сеть.

Для запуска не требуется устанавливать специальную программу на компьютер — достаточно открыть браузер и перейти по нужному адресу.

Отличие от веб-сайта

Веб-сайт — в основном направлен на предоставление информации

Веб-приложение — активно взаимодействует с пользователем

Например: пользователь вводит информацию, нажимает кнопку, ищет данные, выбирает товар, ведёт учёт или входит в личный кабинет.

К веб-приложениям относятся

- Интернет-магазины
- Системы онлайн-банкинга
- Электронное образование
- Социальные сети
- Системы работы с документами
- Системы бронирования
- Корпоративные информационные системы

Веб-сайт и веб-приложение: сравнение

Веб-сайт

Цель

предоставление информации, презентация

Взаимодействие

низкое — в основном чтение/просмотр

Содержимое

в основном статично, редко меняется

Примеры

сайт-визитка, новостной портал, блог

Веб-приложение

Цель

выполнение конкретной задачи, расчёты, обработка

Взаимодействие

высокое — ввод, обработка, хранение данных

Содержимое

динамическое, адаптируется под пользователя

Примеры

интернет-магазин, банковская система, обучающая платформа

Этапы развития веб-приложений

1



Статичные веб-страницы

Сервер отправлял браузеру готовый HTML-документ напрямую. Содержимое было одинаковым для всех пользователей.

2



Серверная динамика

В зависимости от запроса на сервере формировалось разное содержимое (напр., технологии PHP, ASP, CGI).

3



Развитие JavaScript

Появилась возможность изменять часть страницы без перезагрузки (технология AJAX).

4



Современные приложения

Клиентские приложения предлагают интерфейс и функционал, похожие на настольные программы.

Клиентское веб-приложение и его основа

Клиентское веб-приложение (Client-side Web Application) — веб-приложение, в котором основной интерфейс и часть логики выполняются в браузере пользователя.

Три основные технологии клиентской части



HTML5

Формирует структуру документа

Теги, семантические элементы: <header>, <section>, <footer>, <article>



CSS3

Управляет внешним видом

Цвета, расположение, Flexbox, Grid, анимации и переходные эффекты



JavaScript

Реализует логику и интерактивность

Манипуляция DOM, обработка событий, работа с данными, ES6+

Эти три технологии составляют технологическую основу веба.

Архитектура клиент–сервер

Клиент — браузер пользователя

Запрашивает веб-страницу, принимает код HTML/CSS/JS и отображает его на экране.

Сервер — среда, обрабатывающая запросы

Принимает запрос, готовит нужную информацию и отправляет её в ответ.

Рабочий процесс



Например: пользователь переходит в раздел «Ноутбуки» интернет-магазина → браузер отправляет запрос серверу → сервер готовит информацию → браузер обрабатывает полученный код HTML, CSS, JS и показывает интерфейс.

5. КОМПОНЕНТЫ

Основные компоненты веб-приложения



Frontend

Клиентская часть, которую видит пользователь — обеспечивает интерфейс и взаимодействие.

Примеры: HTML, CSS, JavaScript, React, Vue, Angular



Backend

Программная часть на стороне сервера — обрабатывает данные, реализует бизнес-логику.

Примеры: Node.js, Python (Django), Java (Spring), PHP



Database

База данных — хранит информацию о пользователях, заказах, контенте.

Примеры: MySQL, PostgreSQL, MongoDB, Redis



API

Интерфейс обмена данными между клиентом и сервером — стандартизирует запрос и ответ.

Примеры: REST API, GraphQL, WebSocket

Frontend принимает действие пользователя → передаёт его Backend через API → Backend обрабатывает данные и связывается с Database.

Роль браузера

Процесс формирования страницы (рендеринг)



HTML → DOM

Браузер считывает HTML-документ и формирует его структуру DOM (Document Object Model).



CSS → CSSOM

Обрабатывает файлы CSS, определяет расположение и внешний вид элементов.



Выполнение JavaScript

Выполняет код JS, динамически изменяет интерфейс и обрабатывает события.

Популярные браузеры



Google Chrome



Mozilla Firefox



Microsoft Edge



Apple Safari

Веб-сервер и серверные программы

Функции веб-сервера

- Приём и обработка HTTP-запросов, поступающих от клиента
- Хранение и передача запрошенных файлов (HTML, CSS, JS, изображения)
- Связь с базой данных для формирования динамического содержимого
- Аутентификация пользователей и обеспечение безопасности

Широко используемые серверные программы

- **Apache HTTP Server** — веб-сервер с открытым кодом, широко распространён
- **Nginx** — быстрый веб-сервер, устойчивый к высокой нагрузке
- **Microsoft IIS** — веб-сервер для среды Windows Server
- **Node.js** — серверная среда (runtime) на основе JavaScript

На заметку: при выборе веб-сервера учитываются скорость, безопасность, отказоустойчивость под нагрузкой и совместимость с операционной системой.

Протоколы HTTP и HTTPS

HTTP (HyperText Transfer Protocol) — протокол для обмена информацией между клиентом и сервером.

Основные методы HTTP-запроса

GET	POST	PUT	PATCH	DELETE
получение информации	отправка новых данных	полное обновление данных	изменение части данных	удаление данных



HTTPS — защищённая версия

Зашифрованная версия протокола HTTP. Сертификат SSL/TLS защищает данные при передаче, поэтому пароли и платёжные данные остаются в безопасности.

Часто встречающиеся коды ответа

200	успешно выполнено	301	адрес изменён
404	страница не найдена	500	ошибка сервера

Объяснение на примере строительства дома



HTML

Структура дома

Располагает комнаты и стены — формирует структуру страницы



CSS

Дизайн дома

Красит и украшает — управляет внешним видом страницы



JavaScript

Инженерные системы

Как свет, двери, сигнализация — запускает функциональные механизмы страницы

Например: *HTML создаёт кнопку, CSS меняет её цвет и размер, а JavaScript выполняет определённое действие при нажатии кнопки.*

Особенности современных веб-приложений

SPA (Single Page Application)

Страница не перезагружается полностью, обновляется только нужная часть — быстрый и удобный опыт работы.

PWA (Progressive Web App)

Веб-приложение устанавливается как мобильное приложение, может работать и без интернета.

Адаптивный дизайн

Интерфейс автоматически подстраивается под размер разных устройств (телефон, планшет, компьютер).

Компонентная архитектура

Интерфейс строится из переиспользуемых небольших частей (компонентов).

Фреймворки и библиотеки

Такие инструменты, как React, Angular, Vue, Svelte, ускоряют и структурируют разработку.

Офлайн-работа (Service Worker)

Фоновые скрипты кэшируют данные, поддерживая основные функции даже без интернета.

САМОПРОВЕРКА

Контрольные вопросы



Что такое веб-приложение?



В чём отличие веб-сайта от веб-приложения?



Что такое клиентское веб-приложение?



Как работает архитектура клиент–сервер?



Что такое Frontend и Backend?



Каковы основные функции браузера?



Для чего используется HTTP?



В чём отличие запросов GET и POST?



Какую роль выполняют HTML5, CSS3 и JavaScript?

ЗАКЛЮЧЕНИЕ

Основные выводы



Веб-приложение — программная система, выполняемая в браузере и активно взаимодействующая с пользователем.



Архитектура клиент–сервер: браузер отправляет запрос, сервер обрабатывает его и возвращает ответ.



HTML5, CSS3 и JavaScript — три основные технологии клиентской части.



Современные приложения опираются на принципы SPA, PWA и компонентной архитектуры.