

Уральский государственный экономический университет

ОСНОВЫ WEB ПРОГРАММИРОВАНИЯ

02.03.03 -Математическое обеспечение и администрирование информационных систем,
направленность (профиль) -разработка и администрирование информационных систем

<http://vikchas.ru>

Лекция 6-2. «Современные тенденции в Web- программировании»

Часовских Виктор Петрович
доктор технических наук, профессор кафедры
ШИиКМ, ФГБОУ ВО «Уральский государственный
экономический университет

Екатеринбург 2026

Базы данных

Современные Web-приложения используют разные типы баз данных.

Реляционные базы данных

Примеры:

- ADABAS
- PostgreSQL;
- MySQL;
- MariaDB;
- MS SQL Server.

Преимущества:

- строгая структура данных;
- SQL;
- транзакции;
- надежность;
- связи между таблицами.

NoSQL базы данных

Примеры:

- ADABAS
- MongoDB;
- Redis;
- Cassandra;
- DynamoDB.

Преимущества:

- гибкая структура данных;
- горизонтальное масштабирование;
- удобство для некоторых типов задач;
- высокая скорость в определенных сценариях.

NewSQL и облачные БД

Современные проекты всё чаще используют:

- PlanetScale;
- CockroachDB;
- Google Spanner;
- Supabase;
- Neon;
- Firebase.

Тенденция — переход к управляемым облачным базам данных, где часть администрирования берет на себя провайдер.

Облачные технологии

Облака стали стандартом для размещения Web-приложений.

Популярные провайдеры:

- Amazon Web Services;
- Microsoft Azure;
- Google Cloud Platform;
- Cloudflare;
- Yandex Cloud;
- Selectel;
- VK Cloud.

Облачные технологии дают:

- масштабируемость;
- отказоустойчивость;
- быструю доставку контента;
- управляемые базы данных;
- хранение файлов;
- мониторинг;
- автоматическое резервное копирование.

Serverless

Serverless — модель, при которой разработчик не управляет серверами напрямую.

Примеры:

- AWS Lambda;
- Cloudflare Workers;
- Vercel Functions;
- Netlify Functions;
- Yandex Cloud Functions.

Преимущества:

- оплата за фактическое использование;
- автоматическое масштабирование;
- быстрая разработка;
- меньше администрирования.

Недостатки:

- холодный старт;
- ограничения по времени выполнения;
- зависимость от платформы;
- сложность отладки.

Микросервисы и монолиты

Монолит

Монолит — приложение, где все основные функции находятся в одной кодовой базе.

Преимущества:

- проще старт проекта;
- легче разрабатывать небольшой команде;
- проще тестировать и развертывать на ранних этапах.

Недостатки:

- сложнее масштабировать отдельные части;
- рост связности;
- увеличение времени сборки и деплоя.

Микросервисы

Микросервисы — архитектура, где приложение состоит из набора независимых сервисов.

Преимущества:

- независимое масштабирование;
- независимые команды разработки;
- гибкость выбора технологий;
- отказоустойчивость.

Недостатки:

- сложность инфраструктуры;
- сетевые задержки;
- сложность мониторинга;
- проблемы согласованности данных.

Современная тенденция: не начинать проект сразу с микросервисов без необходимости. Часто разумнее стартовать с модульного монолита.

DevOps и CI/CD

Современный Web-разработчик должен понимать основы DevOps.

DevOps включает:

- автоматизацию сборки;
- автоматическое тестирование;
- автоматический деплой;
- мониторинг;
- контейнеризацию;
- управление инфраструктурой.

CI/CD

CI/CD — это практики непрерывной интеграции и доставки.

CI — Continuous Integration:

- автоматическая проверка кода;
- запуск тестов;
- линтинг;
- сборка проекта.

CD — Continuous Delivery / Deployment:

- автоматическая доставка приложения на сервер;
- автоматический деплой после успешных проверок.

Инструменты:

- GitHub Actions;
- GitLab CI/CD;
- Jenkins;
- TeamCity;
- CircleCI.

Контейнеризация и Kubernetes

Docker

Docker позволяет упаковать приложение со всеми зависимостями в контейнер.

Преимущества:

- одинаковая среда разработки и продакшена;
- простота развертывания;
- изоляция;
- удобство масштабирования.

Kubernetes

Kubernetes — система оркестрации контейнеров.

Она позволяет:

- управлять множеством контейнеров;
- автоматически перезапускать сервисы;
- балансировать нагрузку;
- масштабировать приложения;
- управлять конфигурациями и секретами.

Важно: Kubernetes нужен не каждому проекту. Для небольших приложений он может быть избыточным.

Безопасность Web-приложений

Безопасность — одна из ключевых тем современной разработки.

Основные угрозы:

- XSS;
- CSRF;
- SQL-инъекции;
- утечки данных;
- неправильная авторизация;
- небезопасное хранение паролей;
- атаки на зависимости;
- DDoS.

Современные практики безопасности

- использовать HTTPS;
- хранить пароли только в виде хэшей;
- применять bcrypt, Argon2;
- валидировать данные на сервере;
- использовать prepared statements;
- настраивать CORS;
- использовать Content Security Policy;
- обновлять зависимости;

- применять двухфакторную аутентификацию;
- ограничивать права доступа;
- вести аудит действий.

Производительность Web-приложений

Пользователи ожидают быстрой загрузки сайтов. Производительность влияет на:

- удобство пользователя;
- конверсию;
- SEO;
- удержание аудитории.

Метрики:

- LCP — Largest Contentful Paint;
- FID / INP — отзывчивость интерфейса;
- CLS — стабильность макета;
- TTFB — время до первого байта.

Методы оптимизации:

- сжатие изображений;
- lazy loading;
- code splitting;
- кэширование;
- CDN;
- минимизация JavaScript;
- оптимизация CSS;
- SSR/SSG;
- использование современных форматов изображений: WebP, AVIF.

Progressive Web Apps

PWA — Progressive Web Application.

Это Web-приложение, которое ведет себя похоже на мобильное приложение.

Возможности PWA:

- установка на устройство;
- работа офлайн;
- push-уведомления;
- кэширование ресурсов;
- адаптивный интерфейс.

Основные технологии:

- Service Worker;
- Web App Manifest;
- HTTPS;
- Cache API.

PWA особенно полезны там, где важно быстро доставить приложение без публикации в App Store или Google Play.

Адаптивная и доступная разработка

Адаптивность

Современный сайт должен хорошо работать на разных устройствах:

- смартфонах;
- планшетах;
- ноутбуках;
- больших мониторах.

Подходы:

- responsive design;
- media queries;
- flexible layout;
- CSS Grid;
- Flexbox;
- mobile-first.

Доступность

Доступность означает, что сайтом могут пользоваться люди с разными возможностями.

Принципы:

- семантическая HTML-разметка;
- достаточный контраст;
- навигация с клавиатуры;
- alt-тексты для изображений;
- правильные заголовки;
- ARIA-атрибуты при необходимости.

Доступность — это не только этика, но и качество продукта.

Искусственный интеллект в Web-разработке

ИИ активно влияет на Web-программирование.

ИИ как инструмент разработчика

Примеры:

- генерация кода;
- объяснение ошибок;
- автодополнение;
- написание тестов;
- помощь в документации;
- ревью кода.

Инструменты:

- GitHub Copilot;
- ChatGPT;
- Codeium;
- Tabnine;

- JetBrains AI.

ИИ как часть Web-приложений

Многие современные сайты добавляют:

- чат-ботов;
- персонализацию;
- рекомендации;
- анализ текста;
- генерацию изображений;
- голосовые интерфейсы;
- поиск на естественном языке.

Тенденция: ИИ становится обычной частью пользовательского опыта.

Low-code и No-code

Low-code и No-code платформы позволяют создавать приложения с минимальным количеством кода.

Примеры:

- Webflow;
- Bubble;
- Tilda;
- Retool;
- Appsmith;
- Directus.

Преимущества:

- быстрая разработка;
- снижение стоимости прототипирования;
- возможность участия не-программистов.

Недостатки:

- ограничения платформы;

- сложность масштабирования;
- зависимость от поставщика;
- проблемы с гибкостью.

Для студентов важно понимать: No-code не заменяет программистов полностью, но меняет рынок и требования к скорости разработки.

Headless CMS и Jamstack

Headless CMS

Headless CMS — система управления контентом без жестко привязанного frontend.

Примеры:

- Strapi;
- Directus;
- Contentful;
- Sanity;
- Ghost.

Контент хранится в CMS, а сайт получает его через API.

Jamstack

Jamstack — архитектурный подход, основанный на:

- JavaScript;
- API;
- Markup.

Особенности:

- статическая генерация страниц;
- использование внешних API;
- CDN;
- высокая скорость;
- безопасность;

- масштабируемость.

Jamstack популярен для блогов, документации, корпоративных сайтов и интернет-магазинов.

Real-time Web

Многие современные приложения требуют обновления данных в реальном времени.

Примеры:

- чаты;
- онлайн-игры;
- биржевые графики;
- совместное редактирование документов;
- уведомления;
- трекинг доставки.

Технологии:

- WebSocket;
- Server-Sent Events;
- WebRTC;
- MQTT;
- [Socket.IO](https://socket.io/).

WebAssembly

WebAssembly, или WASM, позволяет запускать в браузере код, написанный на других языках, например:

- C;
- C++;
- Rust;
- Go.

Преимущества:

- высокая производительность;

- возможность переносить существующие библиотеки в Web;
- выполнение сложных вычислений в браузере.

Применения:

- графические редакторы;
- игры;
- CAD-системы;
- обработка видео и аудио;
- криптография;
- научные вычисления.

Современные CSS-тенденции

CSS активно развивается.

Актуальные возможности:

- CSS Grid;
- Flexbox;
- CSS variables;
- container queries;
- cascade layers;
- nesting;
- новые псевдоклассы;
- prefers-color-scheme;
- анимации без JavaScript.

Популярные подходы:

- Utility-first CSS;
- Tailwind CSS;
- CSS Modules;
- CSS-in-JS;
- design systems.

Дизайн-системы

Дизайн-система — набор правил, компонентов и принципов для создания интерфейсов.

Она включает:

- палитру цветов;
- типографику;
- сетку;
- UI-компоненты;
- правила использования;
- документацию;
- accessibility-рекомендации.

Преимущества:

- единый стиль продукта;
- ускорение разработки;
- меньше ошибок;
- проще поддерживать большие интерфейсы.

Примеры:

- Material Design;
- Ant Design;
- Carbon Design System;
- Fluent Design.

Тестирование Web-приложений

Современная разработка невозможна без тестирования.

Виды тестов:

- unit-тесты;
- integration-тесты;

- end-to-end тесты;
- snapshot-тесты;
- нагрузочные тесты;
- security-тесты.

Инструменты:

- Jest;
- Vitest;
- Mocha;
- Playwright;
- Cypress;
- Testing Library;
- Selenium;
- k6.

Тенденция — автоматизировать тестирование как можно раньше и запускать его в CI/CD.

Наблюдаемость и мониторинг

После запуска приложения важно понимать, что происходит в продакшене.

Наблюдаемость включает:

- логи;
- метрики;
- трассировку;
- алерты;
- анализ ошибок;
- мониторинг пользовательского опыта.

Инструменты:

- Sentry;
- Grafana;
- Prometheus;

- OpenTelemetry;
- Datadog;
- New Relic;
- ELK Stack.

Какие навыки нужны современному Web-разработчику

Студентам важно развивать не только знание конкретных технологий, но и фундаментальные навыки.

Технические навыки

- HTML, CSS, JavaScript;
- TypeScript;
- основы HTTP;
- работа с API;
- Git;
- базы данных;
- безопасность;
- тестирование;
- основы DevOps;
- понимание архитектуры приложений.

Soft skills

- умение работать в команде;
- коммуникация;
- чтение документации;
- критическое мышление;
- умение задавать вопросы;
- навык самостоятельного обучения;
- ответственность за качество продукта.

Главные тенденции кратко

Современное Web-программирование развивается в сторону:

- TypeScript;
- компонентной архитектуры;
- SSR и гибридного рендеринга;
- облачных платформ;
- serverless;
- автоматизации CI/CD;
- безопасности;
- высокой производительности;
- ИИ-инструментов;
- PWA;
- WebAssembly;
- headless-архитектуры;
- микросервисов там, где они действительно нужны;
- доступности интерфейсов.

Заключение

Web-программирование сегодня — это не только создание страниц в браузере. Это комплексная инженерная область, включающая frontend, backend, базы данных, безопасность, DevOps, облака, тестирование и пользовательский опыт.

Технологии быстро меняются, но фундаментальные принципы остаются важными:

- понимание работы Web;
- чистая архитектура;
- безопасность;
- удобство пользователя;
- производительность;
- способность постоянно учиться.

Главная задача будущего Web-разработчика — не просто знать модные инструменты, а уметь выбирать подходящие технологии для конкретной задачи.