

# **Основы WEB-программирования и Современные WEB-технологии - ЛЕКЦИИ**

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

<https://vikchas.ru>

## **Лекция 2. Тема «Термины и определения для разработчиков и пользователей сайта. С# и Visual Studio»**

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

Екатеринбург 2025

# Термины и определения для разработчиков и пользователей сайта

## Общие термины

- **Сайт**— совокупность веб-страниц, объединенных общей темой, дизайном и доменным именем
- **Домен**— уникальное имя сайта в интернете (например, [example.com](https://example.com))
- **URL**— унифицированный указатель ресурса, адрес страницы в интернете
- **Хостинг**— услуга по размещению сайта на сервере с постоянным доступом в интернет

## Термины для пользователей

- **Личный кабинет**— раздел сайта, где пользователь может управлять своей учетной записью
- **Регистрация**— процесс создания учетной записи на сайте
- **Авторизация**— процесс входа пользователя в систему с использованием логина и пароля
- **Контент**— информационное наполнение сайта (тексты, изображения, видео и др.)
- **Навигация**— система перемещения по разделам сайта
- **Корзина**— виртуальное хранилище выбранных товаров в интернет-магазине
- **Куки (cookies)**— небольшие файлы, сохраняемые в браузере пользователя для хранения данных

## Термины для разработчиков

- **HTML**— язык гипертекстовой разметки, основа для создания веб-страниц
- **CSS**— каскадные таблицы стилей, используемые для оформления веб-страниц

- **JavaScript**— язык программирования для создания динамического контента на сайте
- **Backend**— серверная часть сайта, не видимая пользователю, обрабатывающая данные
- **Frontend**— клиентская часть сайта, видимая пользователю через браузер
- **CMS**— система управления контентом, позволяющая управлять сайтом без знания программирования
- **API**— интерфейс программирования приложений, позволяющий разным программам взаимодействовать
- **Верстка**— процесс создания HTML-страниц сайта на основе дизайн-макета
- **SEO**— поисковая оптимизация, комплекс мер для повышения позиций сайта в поисковых системах
- **Адаптивный дизайн**— подход к веб-дизайну, обеспечивающий правильное отображение сайта на разных устройствах

## Термины и определения WEB-программирования

### Основные технологии и языки

- **HTML (HyperText Markup Language)**— язык разметки гипертекста, стандартный язык для создания структуры веб-страниц
- **CSS (Cascading Style Sheets)**— каскадные таблицы стилей, язык для описания внешнего вида веб-документов
- **JavaScript**— скриптовый язык программирования для динамического управления контентом на стороне клиента
- **TypeScript**— надмножество JavaScript, добавляющее статическую типизацию
- **PHP**— скриптовый язык программирования, используемый преимущественно на стороне сервера
- **Python**— высокоуровневый язык программирования, часто применяемый в веб-разработке (Django, Flask)
- **Ruby**— динамический, объектно-ориентированный язык, используемый в веб-фреймворке Ruby on Rails

## Архитектура и подходы

- **Client-Side**— клиентская часть, код выполняется в браузере пользователя
- **Server-Side**— серверная часть, код выполняется на сервере
- **Frontend**— разработка пользовательского интерфейса и клиентской логики
- **Backend**— разработка серверной логики и работа с базами данных
- **Full-Stack**— разработка как клиентской, так и серверной частей приложения
- **MVC (Model-View-Controller)**— архитектурный паттерн, разделяющий приложение на три компонента
- **REST (REpresentational State Transfer)**— архитектурный стиль взаимодействия компонентов распределённого приложения
- **PWA (Progressive Web Applications)**— подход к созданию веб-приложений, которые работают как нативные

## Инструменты и технологии разработки

- **Framework**— программная платформа, определяющая структуру приложения (React, Angular, Vue.js, Laravel)
- **Library**— набор предопределённых функций и классов (jQuery, Lodash)
- **API (Application Programming Interface)**— набор протоколов и инструментов для создания приложений
- **DOM (Document Object Model)**— объектная модель документа, представляющая HTML-страницу в виде дерева узлов
- **AJAX (Asynchronous JavaScript and XML)**— технология асинхронного обмена данными с сервером
- **JSON (JavaScript Object Notation)**— текстовый формат обмена данными
- **WebSocket**— протокол связи, обеспечивающий полнодуплексный канал связи между клиентом и сервером
- **NPM (Node Package Manager)**— менеджер пакетов для JavaScript
- **Git**— система контроля версий для отслеживания изменений в исходном коде

## Базы данных и хранение данных

- **SQL**— язык структурированных запросов для работы с реляционными базами данных
- **NoSQL**— подход к построению баз данных, отличный от реляционной модели
- **MongoDB**— документо-ориентированная NoSQL база данных
- **MySQL/PostgreSQL**— популярные реляционные СУБД
- **LocalStorage**— механизм хранения данных в браузере пользователя
- **Cookies**— небольшие файлы данных, хранящиеся в браузере пользователя
- **Session Storage**— временное хранилище данных в браузере, существующее в рамках сессии

## Развертывание и инфраструктура

- **Хостинг**— услуга по размещению сайта на сервере
- **Веб-сервер**— программное обеспечение, обрабатывающее HTTP-запросы (Apache, Nginx)
- **CI/CD (Continuous Integration/Continuous Delivery)**— практика непрерывной интеграции и доставки
- **Docker**— платформа для автоматизации развертывания приложений в контейнерах
- **CDN (Content Delivery Network)**— географически распределённая сеть для доставки контента пользователям

# Термины и определения современных ВЕБ-технологий

## Современные фронтенд-технологии

- **React**— JavaScript-библиотека для создания пользовательских интерфейсов с компонентным подходом
- **Vue.js**— прогрессивный JavaScript-фреймворк для создания пользовательских интерфейсов

- **Angular**— платформа для разработки веб и мобильных приложений на TypeScript
- **Svelte**— компилируемый фреймворк, преобразующий компоненты в оптимизированный JavaScript
- **WebAssembly (WASM)**— бинарный формат инструкций для виртуальной машины, позволяющий запускать код на околонулевой скорости
- **CSS Grid**— двумерная система компоновки для CSS
- **Flexbox**— одномерная система компоновки для CSS
- **CSS Variables**— пользовательские свойства CSS, позволяющие хранить и повторно использовать значения
- **CSS Modules**— техника модульного подхода к CSS, где стили ограничены областью видимости
- **Tailwind CSS**— utility-first CSS-фреймворк для быстрого создания пользовательского интерфейса

### Современные бэкенд-технологии

- **Node.js**— среда выполнения JavaScript на сервере
- **Deno**— безопасная среда выполнения JavaScript и TypeScript, созданная разработчиком Node.js
- **GraphQL**— язык запросов и манипулирования данными для API
- **Microservices**— архитектурный стиль, структурирующий приложение как набор слабо связанных сервисов
- **Serverless**— модель облачных вычислений, где провайдер динамически управляет выделением ресурсов
- **gRPC**— высокопроизводительный фреймворк RPC (удаленного вызова процедур)
- **Next.js**— React-фреймворк с серверным рендерингом и статической генерацией
- **Nuxt.js**— фреймворк для Vue.js, упрощающий создание универсальных приложений
- **NestJS**— фреймворк для построения эффективных и масштабируемых Node.js-приложений

## Современные инструменты разработки

- **Webpack**— сборщик модулей JavaScript
- **Vite**— инструмент сборки нового поколения для быстрой разработки
- **ESLint**— инструмент для статического анализа JavaScript-кода
- **Prettier**— инструмент для автоматического форматирования кода
- **TypeScript**— типизированное надмножество JavaScript
- **Storybook**— инструмент для разработки компонентов пользовательского интерфейса в изоляции
- **Jest**— фреймворк для тестирования JavaScript
- **Cypress**— инструмент для end-to-end тестирования веб-приложений
- **Babel**— транспилятор JavaScript, преобразующий новый синтаксис в поддерживаемый браузерами
- **PWA (Progressive Web Apps)**— веб-приложения, использующие современные веб-возможности для обеспечения опыта, подобного приложениям

## Современные подходы к архитектуре

- **JAMstack**— архитектурный подход, основанный на JavaScript, API и статической разметке
- **Isomorphic/Universal JavaScript**— JavaScript-код, выполняющийся как на клиенте, так и на сервере
- **Headless CMS**— система управления контентом, предоставляющая только API без фронтенд-интерфейса
- **Single Page Application (SPA)**— веб-приложение, работающее в рамках одной веб-страницы
- **Server-Side Rendering (SSR)**— рендеринг веб-страниц на сервере перед отправкой клиенту
- **Static Site Generation (SSG)**— генерация HTML-страниц во время сборки
- **API-first Development**— подход, при котором API разрабатывается до
- **Современные технологии хранения и обработки данных**
- **Redis**— хранилище данных типа “ключ-значение” в оперативной памяти

- **MongoDB Atlas**— облачная база данных как сервис для MongoDB
- **Firebase**— платформа разработки мобильных и веб-приложений от Google
- **Supabase**— открытая альтернатива Firebase с PostgreSQL в основе
- **Prisma**— ORM нового поколения для Node.js и TypeScript
- **Elasticsearch**— распределенная поисковая и аналитическая система
- **Apache Kafka**— платформа для потоковой обработки данных

### Современные протоколы и коммуникации

- **HTTP/3**— третья основная версия протокола HTTP, использующая QUIC
- **WebSockets**— протокол для постоянного соединения между клиентом и сервером
- **WebRTC**— технология, позволяющая веб-браузерам обмениваться медиаконтентом напрямую
- **Server-Sent Events (SSE)**— технология отправки уведомлений от сервера к браузеру
- **Push API**— API для получения push-уведомлений от серверов

## С# и Visual Studio в WEB-программировании и современных WEB-технологиях

### С# в веб-разработке

#### Основные платформы и фреймворки

- **[ASP.NET Core](#)**— кроссплатформенный, высокопроизводительный фреймворк с открытым исходным кодом для создания современных веб-приложений
- **[ASP.NET MVC](#)**— реализация паттерна Model-View-Controller для создания веб-приложений
- **[ASP.NET Web API](#)**— фреймворк для создания HTTP-сервисов, доступных широкому кругу клиентов
- **Blazor**— фреймворк для создания интерактивных веб-приложений с использованием С# вместо JavaScript:



- **Blazor Server**— выполняет C# код на сервере с передачей UI-обновлений через SignalR
- **Blazor WebAssembly**— запускает .NET код непосредственно в браузере через WebAssembly

## Преимущества C# в веб-разработке

- **Статическая типизация**— выявление ошибок на этапе компиляции
- **Высокая производительность**— особенно заметна в [ASP.NET Core](#)
- **Единый язык**— использование одного языка для бэкенда и фронтенда (с Blazor)
- **Корпоративный стандарт**— широко используется в корпоративной разработке
- **Экосистема .NET**— доступ к обширной библиотеке пакетов NuGet

## Visual Studio для веб-разработки

### Возможности для веб-разработчиков

- **Интегрированная среда разработки**— полный набор инструментов для веб-разработки
- **IntelliSense**— расширенное автодополнение кода для HTML, CSS, JavaScript, TypeScript и C#
- **Отладка на стороне клиента и сервера**— интегрированный отладчик для серверного кода и JavaScript
- **Публикация в облако**— упрощенное развертывание в Azure и другие сервисы
- **Интеграция с Git и Azure DevOps**— инструменты для управления исходным кодом
- **Встроенные шаблоны проектов**— готовые шаблоны для различных типов веб-приложений

## Visual Studio Code

- Легковесная, кроссплатформенная альтернатива полному Visual Studio
- Популярный редактор для фронтенд и бэкенд разработки
- Богатая экосистема расширений для веб-разработки

## Позиционирование в современных веб-технологиях

### Микросервисная архитектура

- **.NET Microservices**— создание масштабируемых микросервисов на C# и [ASP.NET Core](#)
- **Docker поддержка**— отличная интеграция с контейнерами и оркестраторами
- **gRPC**— первоклассная поддержка высокопроизводительных RPC

### Cloud-native разработка

- **Azure App Service**— оптимизированное развертывание .NET приложений
- **Azure Functions**— serverless разработка на C#
- **Azure SignalR Service**— управляемый сервис для добавления функциональности реального времени

### Современные подходы

- **Minimal API**— упрощенный синтаксис для создания REST API с минимальным кодом
- **GraphQL для .NET**— библиотеки Hot Chocolate и GraphQL .NET для создания GraphQL API
- **.NET MAUI**— создание кроссплатформенных приложений, включая десктопные и мобильные

### Интеграция с фронтенд-экосистемой

- **SPA интеграция**— готовые шаблоны для работы с React, Angular и Vue.js
- **TypeScript поддержка**— расширенная поддержка в Visual Studio
- **JavaScript/TypeScript отладка**— встроенные инструменты отладки

### Актуальность в современной веб-разработке

C# и Visual Studio сохраняют сильные позиции в веб-разработке, особенно в:

- Корпоративной разработке
- Создании высокопроизводительных бэкенд-систем
- Разработке для экосистемы Microsoft (Azure, SharePoint, Dynamics)
- Веб-приложениях, требующих интеграции с Windows-сервисами
- Проектах, где требуется единая технологическая платформа для фронтенда и бэкенда

Технологии Microsoft активно развиваются в сторону открытого исходного кода, кроссплатформенности и соответствия современным тенденциям веб-разработки, что позволяет им оставаться конкурентоспособными в современном веб-ландшафте.

## **С# и Visual Studio в WEB-программировании: позиция в конкурентной среде**

### **Позиционирование С# и Visual Studio на рынке веб-технологий**

#### **Сильные стороны в конкурентной среде**

- **Корпоративная экосистема**— доминирующее положение в корпоративном секторе благодаря интеграции с другими продуктами Microsoft
- **Единая технологическая платформа**— полный стек от фронтенда до бэкенда и базы данных
- **Производительность**—[ASP.NET](#)Core регулярно занимает топовые позиции в тестах производительности веб-фреймворков
- **Долгосрочная поддержка**— Microsoft обеспечивает длительный срок поддержки (LTS) для своих технологий
- **Безопасность**— регулярные обновления безопасности и зрелый стек технологий

#### **Конкуренты и сравнение**

##### **1. Java (Spring) vs С# ([ASP.NET](#)Core)**

- Похожие области применения (корпоративная разработка)
- Java имеет более широкое распространение в определенных секторах (банковский, телеком)
- [ASP.NET](#)Core часто выигрывает в производительности и имеет более современный API

##### **2. Node.js vs С#**

- Node.js лидирует в стартапах и проектах с одновременным использованием JavaScript на фронтенде
- С# предлагает более строгую типизацию и лучшую производительность для сложной бизнес-логики

- Node.js обеспечивает более низкий порог входа для фронтенд-разработчиков

### 3. PHP vs C#

- PHP доминирует в сегменте малого и среднего бизнеса, особенно с CMS-решениями
- C# обычно используется в более сложных проектах с высокими требованиями к архитектуре
- PHP имеет преимущество в более низкой стоимости хостинга и большом количестве доступных специалистов

### 4. Python (Django/Flask) vs C#

- Python популярен в научных проектах, ML/AI и стартапах
- C# предлагает лучшую производительность и более строгую типизацию
- Python имеет более низкий порог входа и богатую экосистему для научных вычислений

## Эволюция и адаптация к конкуренции

### Стратегические изменения Microsoft

- **Открытый исходный код**— перевод .NET Core и многих компонентов в опенсорс для расширения сообщества
- **Кроссплатформенность**— отход от Windows-only подхода к поддержке Linux и macOS
- **Интеграция с популярными инструментами**— поддержка Docker, Kubernetes, Git, npm
- **Поддержка облачных технологий**— тесная интеграция с Azure и поддержка других облачных провайдеров
- **Модернизация подхода к разработке**— внедрение Minimal API, Blazor и других современных подходов

### Позиционирование Visual Studio в среде IDE

- **Конкуренция с JetBrains (Rider, WebStorm, IntelliJ IDEA)**
  - Rider набирает популярность как кроссплатформенная среда для .NET
  - Visual Studio сохраняет преимущество в глубокой интеграции с экосистемой Microsoft

- **VS Code как ответ на легкие редакторы**
  - Microsoft успешно конкурирует с Sublime Text, Atom и другими с помощью VS Code
  - VS Code стал наиболее популярным редактором среди веб-разработчиков

## Эксплуатационные аспекты

### Преимущества в эксплуатации

- **Зрелые инструменты мониторинга**— Application Insights, интеграция с Azure Monitor
- **Высокая надежность**— стабильность работы в продакшн-среде
- **Масштабируемость**— эффективное горизонтальное масштабирование [ASP.NET](#) Core приложений
- **Безопасность и соответствие нормативам**— регулярные обновления безопасности и поддержка соответствия требованиям
- **DevOps-интеграция**— тесная связь с Azure DevOps и поддержка CI/CD пайплайнов

### Эксплуатационные недостатки и проблемы

- **Стоимость лицензирования**— полная версия Visual Studio Enterprise требует значительных затрат
- **Требования к ресурсам**— более высокие системные требования по сравнению с некоторыми конкурентами
- **Зависимость от экосистемы Microsoft**— может быть затруднительно интегрироваться с не-Microsoft инструментами
- **Более высокие требования к хостингу**— по сравнению с PHP/Node.js решениями

## Будущие перспективы в конкурентной среде

### Направления развития

- **Artificial Intelligence**— интеграция с GitHub Copilot и другими AI-инструментами для разработки
- **Blazor как альтернатива JS-фреймворкам**— попытка конкурировать с React, Angular и Vue
- **Serverless и Cloud-native**— усиление позиций в современных архитектурных подходах

- **WebAssembly**— C# через Blazor становится одним из ключевых языков для WASM
- **Контейнеризация и микросервисная архитектура**— улучшение инструментов для разработки и оркестрации

### **Рыночные тенденции**

- Рост интереса к Blazor как альтернативе JavaScript-фреймворкам
- Увеличение использования .NET в стартапах благодаря открытому исходному коду
- Расширение географического присутствия за пределами традиционных рынков Microsoft
- Усиление конкуренции с облачными провайдерами через интеграцию с Azure

C# и Visual Studio сохраняют значительную долю рынка веб-разработки благодаря адаптации к современным требованиям и тенденциям. Microsoft успешно трансформировала свой подход от закрытой экосистемы к более открытой и совместимой платформе, что позволяет этим технологиям оставаться конкурентоспособными даже в условиях быстро меняющегося ландшафта веб-технологий.

В корпоративном секторе C# и Visual Studio продолжают доминировать, а с развитием технологий вроде Blazor и .NET 6/7/8 они также стремятся расширить свое присутствие в сегментах, традиционно занятых другими технологиями.