Обзор платформ: операционных систем и устройств

**Операционная система** (ОС) — это программный комплекс, который управляет ресурсами компьютера, предоставляет интерфейс пользователю и обеспечивает запуск других программ.

На своём настольном компьютере или ноутбуке вы скорее всего пользуетесь операционной системой Windows или MacOS.

Эти ОС имеют графический интерфейс, состоящий из экранных окон, форм, полей ввода, кнопок и других уже привычных вам элементов управления.

Вам также известно, что некоторые важные функции встроены в саму ОС: например, рабочий стол, панель задач и проводник для управления папками и файлами.

Но большую часть времени за компьютером вы используете не голую ОС, а установленные прикладные программы: браузер, мессенджер, текстовый редактор или даже какие-то игры.

Многие прикладные программы создаются для конкретных ОС, а сами операционные системы — для конкретного аппаратного обеспечения («железа»).

Совокупность ОС и железа называется **платформой**.

В вашем смартфоне совершенно другой процессор и сенсорный экран, который хоть и встречается в ноутбуках, но достаточно редко. Отличается не только железо, но и операционная система: скорее всего смартфон работает под управлением Android или iOS.

**Кросс-платформенные приложения**

Приложение созданное для одной платформы нельзя запустить на другой. Разработчикам популярных приложений приходится выпускать и поддерживать сразу несколько версий для разных платформ.

Это затратно, поэтому неоднократно предпринимались попытки как-то компенсировать различия платформ.

Так появилась технология **эмуляции**, когда одна операционная система пытается для запускаемой программы сымитировать другую ОС.

И технология **виртуализации**, когда в одной операционной системе запускается как бы другой виртуальный компьютер с отдельной операционной системой.

Обе технологии предъявляют повышенные требования к железу ПК и не тривиальны в настройке и использовании.

На сегодняшний день наиболее простым способом создания приложения для всех платформ — является разработка веб-приложения.

Действительно, браузеры существуют везде. Они поставляются вместе с ОС (как Edge или Safari) или могут быть скачаны и установлены отдельно. У пользователей даже есть выбор из нескольких популярных браузеров: Google Chrome, Mozilla Firefox, Opera, Яндекс.Браузер и других.

Поэтому веб-сайты и веб-приложения можно считать **кросс**-**платформенными**. Они работают на любой платформе, где есть браузер. Но при этом создатели веб-приложения всё же должны придумать, как оно будет адаптироваться под экраны разного размера. И эту **адаптивность** реализовать.

Раньше недостатком считалось то, что для работы веб-приложения требуется постоянное подключение к интернету. Но сейчас, во-первых, интернет есть практически в любой точке мира, а, во-вторых, веб-приложения могут быть построены таким образом, чтобы часть функций работала автономно (локально, без интернета).

Отметим, что кросс-платформенными бывают не только веб-приложения. В своё время язык программирования Java (не путать с JavaScript) получил бурное развитие именно за счёт виртуальной машины JVM, которая могла выполнять Java-программы и существовала для разных платформ (включая кнопочные мобильные телефоны, на тот момент ещё даже не смартфоны).

**Кросс-браузерность**

К сожалению, не все браузеры идентично отображают один и тот же сайт. Различия связаны в основном с тем, что разработчики каждого браузера пытаются внедрять в него новые возможности, но их реализация не всегда делается по единому стандарту (новшества внедряются быстрее, чем утверждаются стандарты).

Требуются дополнительные усилия от веб-разработчика для того, чтобы сайт не рассыпался, вёл себя и выглядел идентично при просмотре не только в разных браузерах, но и в нескольких версиях одного и того же браузера — тогда сайт станет **кросс**-**браузерным**.

Обычно во внимание принимаются несколько последних версий каждого популярного браузера.

* Мировую статистику использования браузеров можно посмотреть тут [https://www.w3schools.com/browsers](https://www.w3schools.com/browsers/)
* Распределение по конкретным версиям браузеров можно посмотреть здесь <https://gs.statcounter.com/browser-market-share> или на любимом многими веб-разработчиками сайте <https://caniuse.com/usage-table>
* Статистику Рунета по браузерам можно посмотреть тут <https://radar.yandex.ru/browsers>

# Возможности современного браузера

Браузер для многих пользователей стал основным инструментом на ПК, в том числе и для рабочих задач.

Современные браузеры имеют средства синхронизации, что позволяет на разных устройствах иметь единый набор закладок, паролей, общую историю посещений.

Кроме этого для браузеров можно устанавливать расширения, добавляющие новые функции (согласитесь, это похоже на то, как прикладные приложения расширяют возможности ОС и компьютера в целом).

А для веб-разработчиков браузеры предоставляют уже намного больше возможностей, чем отображение статических веб-страницы, сгенерированных сервером.

### JavaScript

В первую очередь, к страницам можно подключать программы на языке JavaScript (их часто называют — скрипты).

Скрипты могут не только менять страницу на лету (добавляя или модифицируя элементы страницы), но также отправлять новые запросы на сервер, добавлять новые данные и элементы управления без перезагрузки страницы.

При этом скрипты хранятся в виде текстовых файлов с командами (такие файлы удобно редактировать), а их компиляция в машинный код происходит сразу во время выполнения специальным движком JavaScript.

Например, в браузере Chrome используется JavaScript-движок V8. Он удался настолько производительным, что его стали применять даже отдельно от браузера. И это вылилось в целую технологию NodeJS для выполнения JS-скриптов на серверах.

Так что веб-разработчик, начав свою карьеру с верстки страниц и изучив далее JavaScript, может в итоге дойти до программирования бэкенда или даже нейронных сетей, не меняя знакомый язык программирования.

### Встроенные браузерные API

Напомним, что API — это набор готовых методов, через которые программа может взаимодействовать с внешними системами.

Для скриптов, подключенных к веб-странице, браузер предоставляет множество готовых API, через которые современные веб-приложения могут делать почти всё то же самое, что и полноценные «настольные» программы или мобильные приложения.

Например, браузер предоставляет API геолокации, благодаря чему сайт или веб-приложения могут получать местонахождение пользователя, помогая таким образом легко настроить доставку или рассчитать маршрут. Также браузер предоставляет API устройств, чтобы скрипты имели доступ даже к аппаратному обеспечению (например, к камере смартфона, чтобы сделать селфи для аватарки, или к сканеру штрих-кодов, чтобы в формате веб-приложения работала целая складская система).

### Безопасность

При этом браузер устроен так, что для доступа ко многим встроенным API сайт должен запросить и получить от пользователя разрешение. Вы, наверняка, замечали, как при первом обращении к гео-локации какого-то нового сайта или интернет-магазина (которым вы пользуетесь впервые) в браузере всплывает окошко с запросом на доступ к данным.

Такое явное для пользователя ограничение доступа делает даже вредоносные веб-сайты менее опасными, чем локальные программы с аналогичными вредоносными функциями. Потому что браузер явно предупреждает пользователя обо всех потенциально опасных намерениях пользователя. И уже сам пользователь решает, стоит ли конкретному сайту доверять.

С одной стороны, регулярные запросы надоедают, но, с другой, именно это воспитывает в пользователе «цифровую гигиену» и осознанное отношение к информационной безопасности.

Что выбрать для разработки веб-приложения: фреймворк или CMS? Чем они отличаются и какие плюсы и минусы есть у фреймворков и CMS?

CMS ускоряет разработку простого шаблонного сайта. У такого сайта сразу готова админка, её не надо писать отдельно, в отличие от разработки на фреймворке. Однако скорость создания сайта достигается за счет шаблонности, ограниченности или излишней универсальности CMS.

При желании расширить функционал, добавить что-то необычное, возможно, придётся изрядно потрудиться. В случае с фреймворком таких проблем не наблюдается. Безусловно, на разработку уходит больше времени, однако любая идея осуществима без каких-либо проблем.

Ведя разработку с помощью фреймворка, можно самостоятельно построить взаимодействие пользователя с функциями и данными и решить, как это будет выводиться в браузер. В CMS всё это решено за пользователя. С одной стороны, это неплохо, ведь если у разработчика недостаточно опыта, то с фреймворком или чистым языком можно наворотить такого, что в итоге CMS окажется очень хорошим вариантом.

Одним из самых популярных CMS является Wordpress. С ее помощью можно создавать различные сайты и управлять ими без знаний и навыков программирования.

:info:CMS (Content Management System) — это система управления контентом на сайте.

Всё, что формирует сайт, создается и редактируется в CMS. Ее также называют движком ресурса. Сегодня в сети существует множество различных программ или систем управления, но WordPress является самой популярной. Более 30 % всех сайтов в интернете работают на WordPress. А если говорить исключительно о площадках, работающих на CMS, то на WP приходится более 60 %, что делает ее однозначным лидером рынка.

Использование CMS Wordpress имеет массу преимуществ:

* Позволяет создавать, редактировать записи, страницы, рубрики, метки.
* Дает возможность добавить блок комментариев на странице.
* Имеет удобный редактор, который может понять начинающий специалист без опыта в создании сайтов.

Но Wordpress не единственная CMS на рынке. Есть также и другие решение, такие как:

* 1С-Битрикс — это CMS, система управления сайтом. Она предназначена для создания информационных проектов, интернет-магазинов. В ней происходят все работы с сайтом — от загрузки изображений до подключения модулей. Система платная и была разработана Сотбит — российско-белорусской IT-компанией
* Drupal — система управления содержимым (CMS), используемая также как каркас для веб-приложений (CMF), написанная на языке PHP и использующая в качестве хранилища данных реляционную базу данных. На Drupal работает более миллиона сайтов — от личных блогов до сайтов компаний, политических партий и государственных организаций. Что еще примечательно, Drupal является открытым и свободным продуктом, построенном на фреймворке Symfony.

Как итог, можно понять, что CMS хорошо подходит для простых задач, а также для начинающих свою карьеру программистов, в то время как фреймворк требует наличия опыта и общего понимания структуры сайтов при решении сложных разноплановых задач.

Веб-стек

**Веб-стек** — это совокупность платформы и установленных на ней программ для запуска веб-приложений.

Классическим примером является LAMP-стек, который в соответствии с буквами из аббревиатуры состоит из:

* **Linux** — операционная система, как основа платформы
* **Apache** — веб-сервер, который получает входящие запросы по HTTP протоколу и возвращает ответы сам, либо передавая обработку другим приложениям
* **MySQL** — СУБД для хранения данных веб-приложением
* **PHP** — интерпретатор языка PHP, который, получая запросы от Apache, обрабатывает их с помощью веб-приложения, написанного на языке PHP

Далее обсудим каждый компонент стека.

**Linux**

Это открытая свободная операционная система, ядро которой используется для многих дистрибутивов (сборок ОС) и для разных платформ (например, мобильная ОС Android тоже использует ядро Linux).

Отметим, что СУБД и интерпретатор PHP можно запускать и на других ОС, не только на Linux, но и, например, на Windows. Однако, на веб-серверах господствует именно Linux (примерно с тем же перевесом, как Windows лидирует на десктопах).

**Дистрибутивы Linux**

В отличии от Windows и MacOS множество разных производителей делают свои сборки Linux.

Популярные дистрибутивы, о которых вы возможно слышали: Ubuntu, Debian, Mint, CentOS, Arch Linux. Их десятки. Есть и отечественные: ROSA Linux, ALT Linux, Astra Linux, RED OS (некоторые из них имеют в том числе сертификацию ФСТЭК).

Ряд дистрибутивов имеют несколько версий, включая серверные версии, у которых вообще нет графического интерфейса, управлять ими можно через **интерфейс командной строки**.

**Apache**

**Apache** — это программа веб-сервер. Она принимает запросы по HTTP, выясняет, что с ним надо делать (например, в зависимости от того, какой адрес фигурирует в запросе) и либо сама обрабатывает запрос (например, если запросили картинку, то отдаёт файл с сервера), либо передаёт запрос какой-то ещё программе на сервере (например, приложению на языке PHP), а потом ждёт обработки и возвращает ответ.

Кромe Apache существует очень популярный веб-сервер **Nginx** — это открытая и свободная разработка россиянина Игоря Сысоева. В некоторых задачах Nginx быстрее Apache, но не поддерживает обработку распространенных конфигурационных файлов .htaccess, из-за чего часто на одном сервере используют вместе и Nginx, и Apache.

Обратите внимание, что веб-сервером часто называют не только программу, но и весь компьютер, на котором эта программа работает.

**MySQL**

О базах данных и СУБД мы с вами подробнее поговорим на следующем занятии.

А пока отметим, что почти все современные веб-приложения хранят данные в БД, потому что хранить их там надёжнее, чем в отдельных файлах. И выбирать нужные данные из базы намного быстрее.

**PHP**

Самый популярный язык программирования для бэкенда в вебе. Используется во всем известных продуктах: ВКонтакте, Facebook, Wikipedia.

Большинство сайтов в вебе (по данным W3Techs за 2022 год более 75%) построены с использованием PHP.

Язык имеет достаточно низкий порог вхождения.

Неоднократно подвергался критике за низкую производительность, но непрерывно развивается.

**Критика PHP**

Особенность PHP в том, что написанная на нём программа запускается по входящему запросу и после формирования ответа — прекращает работу.

Из-за этого на PHP неудобно создавать программы-сервисы, которые должны работать постоянно и непрерывно выполнять действия.

Упомянутый ранее Python предусматривает иной подход: приложение постоянно запущенно, один экземпляр приложения обрабатывает все запросы. Предварительные или промежуточные результаты вычислений между запросами Python может хранить в оперативной памяти, для PHP об этом необходимо заботиться отдельно разработчику.

**Хостинг**

Теперь, когда вы знаете обо всех компонентах веб-стека, мы можем воспринимать хостинг — как услугу по предоставлению веб-стека для исполнения ваших веб-приложений.

Хостинг может быть предоставлен в виде физического настроенного сервера или в виде VPS — виртуального сервера (несколько таких работают на одном физическом, и вам не надо беспокоиться о железе — это забота поставщика услуг).

А самый доступный вариант — виртуальный хостинг. Когда на одном настроенном веб-сервере просто выделяется дисковое пространство и ресурсы от общей СУБД и общего веб-сервера.