**1. Доменное имя**

— это, простыми словами, «название» сайта. домен сайта — это его уникальный «адрес». Если у вашего сайта не будет домена, пользователи просто не найдут к нему дорогу и не увидят содержимое.

**2. Веб-страница**

— это простой документ, отображаемый браузером . Такие документы пишутся на языке HTML.

**3. Веб-сайт**

— это набор связанных веб-страниц (плюс связанные с ними ресурсы), которые имеют общее уникальное доменное имя. Каждая веб-страница данного веб-сайта содержит явные ссылки — в виде кликабельной части текста, — которые позволяют пользователю переходить с одной страницы веб-сайта на другую. Чтобы получить доступ к веб-сайту, введите его доменное имя в адресной строке браузера, и браузер отобразит главную веб-страницу или домашнюю страницу веб-сайта.

**4. URI**

— URI обозначает Uniform Resource Identifier и по сути является последовательностью символов, которая идентифицирует какой-то ресурс ( это символьная строка, позволяющая идентифицировать какой-либо ресурс: документ, изображение, файл, службу, ящик электронной почты и т. д). URI может содержать URL и URN.

**5. Веб-сервер**

— это компьютер, на котором размещен один или несколько веб-сайтов . «Хостинг» означает, что все веб-страницы и их вспомогательные файлы доступны на этом компьютере.

**6. HTTP**

— HTTP (англ. *Hypertext Transfer Protocol - гипертекстовый транспортный протокол*). Как следует из названия, HTTP указывает, как передавать гипертекст (т.е. связанные веб-документы) между двумя компьютерами.

Протокол представляет собой набор правил для связи между двумя компьютерами. HTTP является текстовым протоколом без сохранения состояния.

**7. Статический веб-сервер**

— состоит из компьютера ("железо") с сервером HTTP (ПО). Мы называем это «статикой», потому что сервер посылает размещённые файлы в браузер «как есть».

**8. Динамический веб-сервер**

состоит из статического веб-сервера и дополнительного программного обеспечения, чаще всего *сервера приложения* и *базы данных*. Мы называем его «динамическим», потому что сервер приложений изменяет исходные файлы перед отправкой в ваш браузер по HTTP.

**9. Хостинг файлов**

— Прежде всего, веб-сервер должен содержать файлы веб-сайта, а именно все HTML-документы и связанные с ними ресурсы, включая изображения, CSS-стили, JavaScript-файлы, шрифты и видео.

Технически, вы можете разместить все эти файлы на своём компьютере, но гораздо удобнее хранить их на выделенном веб-сервере, который:

· всегда запущен и работает

· всегда подключён к Интернету

· имеет неизменный IP адрес (не все [провайдеры](https://developer.mozilla.org/ru/docs/Glossary/ISP) предоставляют статический IP-адрес для домашнего подключения)

· обслуживается третьей, сторонней компанией

По всем этим причинам поиск хорошего хостинг-провайдера является ключевой частью создания вашего сайта. Рассмотрите многочисленные предложения компаний и выберите то, что соответствует вашим потребностям и бюджету (предложения варьируются от бесплатных до тысяч долларов в месяц). Вы можете найти подробности в [этой статье.](https://developer.mozilla.org/en-US/docs/Learn/Common_questions/How_much_does_it_cost#hosting)

Как только вы решили проблему с хостингом, вам понадобится только загрузить свои файлы на ваш веб-сервер.

**10. Жизненный цикл запроса на уровне HTTP-протокола.**

— запрос на ресурс формирует клиент, запрос – это сформированный URI, по протоколу запрос отправляется на сервер, HTTP-запрос – это текст, состоящий из стартовой строки запроса, заголовков и тела запроса, сервер отправляет ответ по тому же протоколу, HTTP-ответ включает в себя код ответа и сформированную HTML-страницу

**11. Что такое GIT**

— система контроля версий, которая позволяет хранить и отслеживать внесённые в файлы изменения. С Git над одним проектом могут работать несколько разработчиков.

**12. Зачем используют XAMPP**

XAMPP — кроссплатформенная сборка веб-сервера, содержащая Apache, MySQL, интерпретатор скриптов PHP, язык программирования Perl и большое количество дополнительных библиотек, позволяющих запустить полноценный веб-сервер

**13. Какую логику можно реализовать, программируя серверную часть кода.**

**14. JSON: расшифровка, использование**

— (англ. JavaScript Object Notation) — текстовый формат обмена данными, основанный на JavaScript. Несмотря на то, что он очень похож на буквенный синтаксис объекта JavaScript, его можно использовать независимо от JavaScript, и многие среды программирования имеют возможность читать (анализировать) и генерировать JSON.

**15. Аутентификация**

Аутентифика́ция ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *authentication* < [греч.](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B5%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) [αὐθεντικός](https://ru.wiktionary.org/wiki/%CE%B1%E1%BD%90%CE%B8%CE%B5%CE%BD%CF%84%CE%B9%CE%BA%CF%8C%CF%82) [authentikos] «реальный, подлинный» < αὐτός [autos] «сам; он самый») — процедура проверки подлинности

**16. Авторизация**

Авториза́ция ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *authorization* «разрешение; уполномочивание») — предоставление определённому лицу или группе лиц прав на выполнение определённых действий; а также процесс проверки (подтверждения) данных прав при попытке выполнения этих действий.[[1]](https://ru.wikipedia.org/wiki/%D0%90%D0%B2%D1%82%D0%BE%D1%80%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F#cite_note-1)[[2]](https://ru.wikipedia.org/wiki/%D0%90%D0%B2%D1%82%D0%BE%D1%80%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F#cite_note-2)[[3]](https://ru.wikipedia.org/wiki/%D0%90%D0%B2%D1%82%D0%BE%D1%80%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F#cite_note-3) Часто можно услышать выражение, что какой-то человек «авторизован» для выполнения данной операции — это значит, что он имеет на неё право.

**17. Статический сайт**

*Статический сайт* — это тот, который возвращает тот же жёсткий кодированный контент с сервера всякий раз, когда запрашивается конкретный ресурс. Например, если у вас есть страница о товаре в /static/myproduct1.html, эта же страница будет возвращена каждому пользователю. Если вы добавите ещё один подобный товар на свой сайт, вам нужно будет добавить ещё одну страницу (например, myproduct2.html) и так далее. Это может стать действительно неэффективным — что происходит, когда вы попадаете на тысячи страниц товаров? Вы повторяли бы много кода на каждой странице (основной шаблон страницы, структуру и т. д.), И если бы вы захотели изменить что-либо в структуре страницы — например, добавить новый раздел «связанные товары» — тогда вам придётся менять каждую страницу отдельно.

**18. Динамический сайт**

*Динамический сайт* — это тот, который может генерировать и возвращать контент на основе конкретного URL-адреса запроса и данных (а не всегда возвращать один и тот же жёсткий код для определённого URL-адреса). Используя пример сайта товара, сервер будет хранить «данные» товара в базе данных, а не отдельные HTML-файлы. При получении GET-запроса для товара сервер определяет идентификатор товара, извлекает данные из базы данных и затем создаёт HTML-страницу для ответа, вставляя данные в HTML-шаблон.

**19. Жизненный цикл запроса к статическому сайту**

**20. Жизненный цикл запроса к динамическому сайту**

**21. Дополнительные задачи, выполняемые веб-сервером**

**22. Технология AJAX: в чем суть технологии**

AJAX – это аббревиатура от «Asynchronous JavaScript and XML», которая дословно переводится как «асинхронный JavaScript и XML».

AJAX – это термин, который описывает как можно используя существующие технологии получать данные с сервера в фоновом режиме и использовать их для обновления страницы (без перезагрузки). Основная цель AJAX – это сделать сайты и веб-приложения более удобными, быстрыми и отзывчивыми.

Основные преимущества использования AJAX:

снижение трафика, уменьшение нагрузки на сервер, увеличение быстродействия и отзывчивости, повышение интерактивности

**23. Методы HTTP-запроса**

HTTP определяет множество методов запроса, которые указывают, какое желаемое действие выполнится для данного ресурса. Несмотря на то, что их названия могут быть существительными, эти методы запроса иногда называются HTTP глаголами. Каждый реализует свою семантику, но каждая группа команд разделяет общие свойства: так, методы могут быть безопасными, идемпотентными или кешируемыми.

GET метод запрашивает представление ресурса. Запросы с использованием этого метода могут только извлекать данные.

HEAD запрашивает ресурс так же, как и метод GET, но без тела ответа.

POST используется для отправки сущностей к определённому ресурсу. Часто вызывает изменение состояния или какие-то побочные эффекты на сервере.

PUT заменяет все текущие представления ресурса данными запроса.

DELETE удаляет указанный ресурс.

CONNECT устанавливает "туннель" к серверу, определённому по ресурсу.

OPTIONS используется для описания параметров соединения с ресурсом.

TRACE выполняет вызов возвращаемого тестового сообщения с ресурса.

PATCH используется для частичного изменения ресурса.

**24. Что такое CRUD**

CRUD (сокр. от англ. create, read, update, delete — «создать, прочесть, обновить, удалить») — акроним, обозначающий четыре базовые функции, используемые при работе с хранилищами данных:

создание;

чтение;

редактирование;

удаление.

**25. Состав HTTP-сообщения**

Обмен сообщениями идёт по обыкновенной схеме «запрос-ответ». Каждое HTTP-сообщение состоит из трёх частей, которые передаются в указанном порядке:

· Стартовая строка (Starting line) — определяет тип сообщения;

· Заголовки (Headers) — характеризуют тело сообщения, параметры передачи и прочие сведения;

· Тело сообщения ( Message Body) — непосредственно данные сообщения. Обязательно должно отделяться от заголовков пустой строкой.

**26. На какие группы делятся HTTP-заголовки**

Заголовки HTTP (HTTP Headers) — это строки в HTTP-сообщении, содержащие разделённую двоеточием пару имя-значение. Формат заголовков соответствует общему формату заголовков текстовых сетевых сообщений ARPA (RFC 822). Заголовки должны отделяться от тела сообщения хотя бы одной пустой строкой.

Все заголовки разделяются на четыре основных группы:

1. General Headers (Основные заголовки) — должны включаться в любое сообщение клиента и сервера.

2. Request Headers (Заголовки запроса) — используются только в запросах клиента.

3. Response Headers (Заголовки ответа) — только для ответов от сервера.

4. Entity Headers (Заголовки сущности) — сопровождают каждую сущность сообщения.

Именно в таком порядке рекомендуется посылать заголовки получателю.

**27. Что такое интерфейс**

(от англ. interface — взаимодействие) — это устройство или система для взаимодействия между не связанными друг с другом объектами.

Интерфейс — это набор инструментов, который позволяет пользователю взаимодействовать с программой. В более широком смысле термин обозначает любые инструменты для соприкосновения между разными системами и сущностями. Часто говорят о графическом интерфейсе — это внешний вид сайта, программы или приложения.

Бывают: Командная строка, Графический интерфейс, Голосовой интерфейс

**28. GUI**

GUI (Graphical User Interface) или ГИП (графический интерфейс пользователя) — программная оболочка, которая предоставляет пользователю удобный интерфейс для работы с операционной системой. Она визуализирует многие компоненты в виде графических объектов, например, кнопки, меню, стрелки и т. д.

Графический интерфейс разделяют на 3 основных типа:

Простой. В качестве объектов визуализации используются стандартные шаблоны, встроенные в GUI.

Двумерный. Здесь объектами выступают библиотеки сторонних систем графических оболочек. В некоторых случаях используют нестандартные объекты.

Трехмерный. Как следует из названия, объекты представлены для пользователя в виде трехмерной проекции.

К преимуществам графического интерфейса относится наличие более дружелюбной (с англ. Friendly-user) системы управления ОС по сравнению со стандартной консолью. Также возможность разбивать настройки по группам для удобства администрирования.

К недостаткам GUI относят повышенное потребление системных ресурсов, особенно оперативной памяти. Это связано с тем, что все графические объекты, используемые в графическом интерфейсе загружены в оперативную память на постоянной основе.

**29. API**

API (англ. Application Programming Interface — программный интерфейс приложения) — это набор способов и правил, по которым различные программы общаются между собой и обмениваются данными.

Все эти взаимодействия происходят с помощью функций, классов, методов, структур, а иногда констант одной программы, к которой обращаются другие.

**30. Rest API**

REST – это Representational State Transfer, т. е. передача репрезентативного состояния. REST определяет набор функций, таких как GET, PUT, DELETE и т. д., которые клиенты могут использовать для доступа к данным сервера. Клиенты и серверы обмениваются данными по протоколу HTTP.

Главной особенностью [REST API](https://docs.aws.amazon.com/apigateway/latest/developerguide/http-api-vs-rest?pg=wianapi&cta=restapi) является то, что такая передача выполняется без сохранения состояния. Без сохранения состояния означает, что серверы не сохраняют клиентские данные между запросами. Клиентские запросы к серверу аналогичны URL-адресам, которые вы вводите в браузере для посещения веб-сайта. Ответ от сервера представляет собой простые данные без типичного графического отображения веб-страницы.

REST API имеет четыре главных преимущества: Интеграция, Инновации, Расширение, Простота обслуживания