**Конспекты по учебе**

**Сеть** — это совокупность устройств и систем, которые подключены друг к другу с помощью кабелей таким образом, что они могут взаимодействовать между собой с целью совместного использования информации и ресурсов.

В зависимости от территориальной распространенности выделяют несколько типов сетей: персональная, локальная, кампусная, городская, глобальная (при перечислении видов сетей мы расположили их в порядке возрастания территории распространения). Чаще всего при разделении сетей на группы останавливаются на выделении **локальной** и **глобальной** сетей.

**Локальная** сеть есть даже у вас дома. Все устройства, которые подключены к роутеру (телефоны, планшеты, компьютер), образуют вместе локальную сеть.

Пример **глобальной** сети вы можете наблюдать, отправляя поисковый запрос, например, в Яндексе. Подключение к глобальной сети осуществляется через Интернет-провайдера.

**Устройства** являются узлами сети, которые обмениваются друг с другом данными. Ниже приведены примеры устройств:

**Конечные (End devices или hosts):**

- Компьютер
- IP телефон
- Сетевой принтер

**Промежуточные (Intermediary devices):**

- Маршрутизатор
- Коммутатор

**Среда передачи данных (Media)** – это собственно сама сеть, которая соединяет устройства. Например, провод (витая пара) – это среда передачи данных. Используются среды трёх типов: оптические, медные и беспроводные.

Под **сервисами** подразумеваются службы, работающие на конечных и промежуточных устройствах, а также услуги, которые они предоставляют. Примерами запущенных сервисов на конечном устройстве могут быть веб-сервисы, почтовые сервисы, файловые сервисы и другие.
На промежуточных устройствах тоже есть свои сервисы, например, на маршрутизаторе могут работать протоколы динамической маршрутизации, Cisco Discovery Protocol и множество других служб.

**Устройства** и **среда** — это физические элементы или оборудование сети, которые часто являются видимой частью сети: ноутбуки, ПК, коммутатор, маршрутизатор, кабели для соединения устройств.

Те устройства, которые образуют интерфейс между пользователем и коммуникационной сетью, которая предоставляет связь, называются **оконечными устройствами** или **узлами**. К ним относятся, например, ноутбуки, ПК, сетевые принтеры, терминальное оборудование, смартфоны.

Другие устройства, которые служат для соединения узлов, называются **промежуточными устройствами**, к ним относят, например, коммутаторы и маршрутизаторы.

**Коммутатор** — это устройство доступа к сети, например, для объединения нескольких компьютеров в локальную сеть для обмена данными между ними.

**Маршрутизатор** — устройство сетевого взаимодействия, которое осуществляет поиск оптимального пути передачи информации от одного устройства сети до другого. Например, от сервера какого-либо сайта до вашего персонального компьютера.

Все компьютерные узлы, которые имеются в современных сетях, могут принимать две разные роли: **клиент** и **сервер**. Эта роль определяется программным обеспечением.

**Серверы** — это узлы, на которых установлено ПО, позволяющее им предоставлять другим сетевым узлам какую-либо информацию. Например, доступ к электронной почте, веб-страницам.

**Клиенты** — это компьютерные узлы с установленным ПО, которое позволяет запрашивать и отображать полученную с сервера информацию. Например, веб-браузер или приложение электронной почты.

Модель **Open Systems Interconnection** (OSI) — это скелет, фундамент и база всех сетевых сущностей. Модель определяет сетевые протоколы, распределяя их на 7 логических уровней.

- **Физический** — самый низкий — определяет процесс прохождения сигналов через среду передачи между сетевыми устройствами (узлами сети).
- **Канальный** — уровень передачи данных.
- **Сетевой** — отвечает за передачу данных и управляет маршрутизацией сообщений — передача через несколько каналов связи по одной или нескольким сетям, что обычно требует включения в пакет сетевого адреса.
- **Транспортный уровень** — управляет сквозной передачей сообщений между оконечными узлами сети, обеспечивая надежность и экономическую эффективность передачи данных независимо от пользователя.
- **Сеансовый** — обеспечивает обслуживание двух «связанных» на уровне представления данных объектов сети и управляет ведением диалога между ними путем синхронизации, заключающейся в установке служебных меток внутри длинных сообщений.
- **Уровень представления** — обеспечивает совокупность служебных операций, которые можно выбрать на прикладном уровне для интерпретации передаваемых и получаемых данных.
- **Прикладной** — обеспечивает непосредственную поддержку прикладных процессов и программ конечного пользователя, а также управление взаимодействием этих программ с различными объектами сети.

Функционирование компьютерной сети тесно связно с **интерфейсом** и **протоколами**.

**Интерфейс** — это соглашение между взаимодействиями среди уровней одной системы, определяющее структуру данных и способ (алгоритм) обмена данными между соседними уровнями OSI-модели.

**Протокол** — это совокупность правил, регламентирующих формат и процедуры взаимодействия процессов одноименных уровней на основе обмена сообщениями.

## Протоколы прикладного уровня

**FTP**

File Transfer Protocol, протокол передачи файлов

FTP даёт возможность абоненту обмениваться двоичными и текстовыми файлами с любым компьютером сети. Установив связь с удалённым компьютером, пользователь может скопировать файл с удалённого компьютера на свой или скопировать файл со своего компьютера на удаленный.

Например, при работе с файлами сайта для быстрого обмена между ПК и директориями сайта.

**HTTP**

Hypertext Transfer Protocol, протокол работы с гипертекстовыми документами

Протокол HTTP используется при пересылке веб-страниц между компьютерами, подключенными к одной сети. Также сейчас имеется расширение протокола HTTP — **HTTPS** (безопасный протокол передачи гипертекста), который поддерживает шифрование посредством криптографических протоколов SSL и TLS.

HTTPS-протокол необходимо использовать, например, на веб-сайтах, где вводится и передаётся **конфиденциальная информация** (личные данные, детали доступа, реквизиты платёжных карт), а также на любых веб-сайтах, где используется авторизация, взаимодействие с платёжными системами, почтовыми сервисами. Использование этого протокола позволит предотвратить получение и использование данных третьими лицами.

**Выбор HTTPS-протокола при продвижении веб-сайта влияет на:**

- Доверие пользователей.
- Доверие со стороны поисковых систем.

Использование HTTPS, по официальным данным, является одним из факторов ранжирования сайтов в поиске. Поэтому значимость веб-сайта будет расти со временем, если перейти с HTTP на HTTPS.

**TELNET**

Протокол удалённого доступа

TELNET даёт возможность абоненту работать на любой ЭВМ, находящейся с ним в одной сети, как на своей собственной — то есть запускать программы, менять режим работы и так далее.

На практике возможности ограничиваются тем уровнем доступа, который задан администратором удалённой машины.

### Протокол транспортного уровня

**TCP**

Transmission Control Protocol, протокол контроля передачи информации

Когда осуществляется передача от компьютера к компьютеру через Интернет, TCP работает на верхнем уровне между двумя конечными системами, например, браузером и веб-сервером. TCP осуществляет надёжную передачу потока байтов от одного процесса к другому.

### Протокол межсетевого уровня

**IP**

Internet Protocol Address

Протокол IP отвечает за приём форматированных сегментов TCP, инкапсуляцию их в пакеты, присвоение им соответствующих адресов и их доставку к узлу назначения.

### Протокол канального уровня

**Ethernet**

**Ethernet** — протокол, который определяет формат кадров и управление доступом к среде на канальном уровне модели OSI.

Давайте представим, как бы действовали эти протоколы, если бы участвовали в отправке почтовой посылки.

➜ Сначала включается **FTP-протокол**. Он даёт возможность установить связь с получателем.

➜ Дальше — **HTTP/HTTPS-протокол**, его можно представить в виде пути, по которому будет доставляться посылка.

➜ **Протокол TCP** включается в момент пересылки письма. Он контролирует его доставку до пункта назначения.

➜ Дальше в действие вступает **IP-протокол**, который присваивает письму соответствующий адрес.

Ethernet в данном случае отвечает за связь между отправителем и получателем через канал передачи данных и перемещает его по среде передачи.

Вот так это будет выглядеть в схематическом виде после того, как будет выполнена работа FTP-протокола:

**Web-технологии** — комплекс технических, коммуникационных, программных методов решения задач организации совместной деятельности пользователей с применением сети Интернет.

### Интернет-сервисы

**WWW** — Всемирная паутина.

Не стоит путать WWW и Интернет. **Интернет** — это глобальная сеть компьютерных сетей, благодаря которой можно обмениваться информацией между компьютерами, а **WWW** — это один из сервисов Интернет.

### Работа в Интернете

- Поисковые системы.
- Просмотр страниц в браузере.

### Информационные ресурсы Интернета

- Веб-страницы, Интернет-магазины, Интернет-порталы, веб-приложения.
- URL и протоколы передачи данных, адресация.

- Создание сайтов.
- Языки веб-программирования.

**Веб-страницы** — это файлы в формате «неформатированный текст» (plain text, текст в ASCII-кодах), распознаваемые любой операционной системой.

Для посетителя это контент, содержащий текст, картинки, меню, ссылки и так далее, отформатированные определенным образом. Для сервера это код, написанный с использованием языков веб-разработки.

**Веб-сайт** — это совокупность логически связанных между собой веб-страниц, или просто одна страница. Доступ в Интернет осуществляется с помощью специальной программы — **браузера**.

**Благодаря браузеру пользователь может:**

- Просматривать веб-страницы (например, сейчас вы находитесь на обучающей платформе, доступ к которой получили благодаря браузеру).
- Скачивать файлы (изображения, игры, музыку, фильмы, книги, документы и так далее).
- Хранить информацию: это и история посещения сайтов, пароли к аккаунтам, если вы их сохраняете, закладки, которые позволяют не потерять понравившиеся веб-сайты.
- Обмениваться данными с другими пользователями: отправка электронных писем, переписка в чатах, выкладывание фото и видео.
- Покупать товары онлайн: через Интернет-магазины и специализированные сайты объявлений.
- Учиться онлайн: всевозможные дистанционные программы, онлайн-курсы, как, например, тот, который вы проходите.

Современные браузеры имеют дополнительные функции, например, защиту от вредоносных программ, блокирующую опасные сайты или предупреждающую об угрозе.

Есть возможность расширения функциональности браузера за счёт установки **дополнений (плагинов).** Они позволяют изменить внешний вид браузера по вашему вкусу, добавить различные информеры (погода, новости, общение), облегчить какие-либо специализированные действия.

Браузер обрабатывает запрос пользователя, отправляет его на сервер, по этим запросам получает данные с сервера, которые, как переводчик, преобразует в видимую пользователю страницу сайта.

Но что именно происходит, когда мы вводим адрес сайта в браузер и нажимаем *Enter*? Как он выдаёт нам именно тот сайт, который мы хотим получить?

У каждого сайта есть определенный адрес — IP, под которым он хранится на веб-сервере. Фактически **IP-адреса обеспечивают связь между устройствами от источника до назначения и обратно** в любом сетевом взаимодействии.

IP-адреса могут быть присвоены физическим портам и виртуальным интерфейсам на всех устройствах. Виртуальный интерфейс означает, что с данным устройством не связано дополнительное физическое оборудование.

**IP-адреса могут быть**:

- **Статическими** — назначается конкретному устройству и не изменяется. Статические IP-адреса обычно имеют все веб-сайты. Постоянный адрес служит гарантией того, что пользователь получит доступ к тому же серверу, что и ранее.
- **Динамическими** — адрес, который меняется при каждом подключении к сети, обычно «выдаётся» пользователям. Чаще всего набор IP-адресов у провайдера ограничен, поэтому, когда новое устройство подключается к сети, ему выдаётся любой свободный адрес. После его отключения этот адрес может быть присвоен другому устройству. Стоит отметить, что такие адреса считаются более безопасными по сравнению со статическими: затрудняется отслеживание компьютера и других устройств, подключенных к сети.

Таким образом, чтобы получить с сервера необходимую информацию, которую потом браузер преобразует в сайт, нужно знать его IP-адрес. Для

решения этой проблемы есть **DNS** (Domain Name System) — система доменных имён.

**DNS** — это огромная таблица, в которой хранятся имена сайтов и соответствующие им IP. Когда вы вводите адрес сайта, браузер обращается к DNS-серверу, просматривает таблицу данных, находит совпадение и получает IP. Таким образом, DNS — это **технология, которая помогает браузеру найти правильный сайт по доменному имени.**

**Подведём итоги**

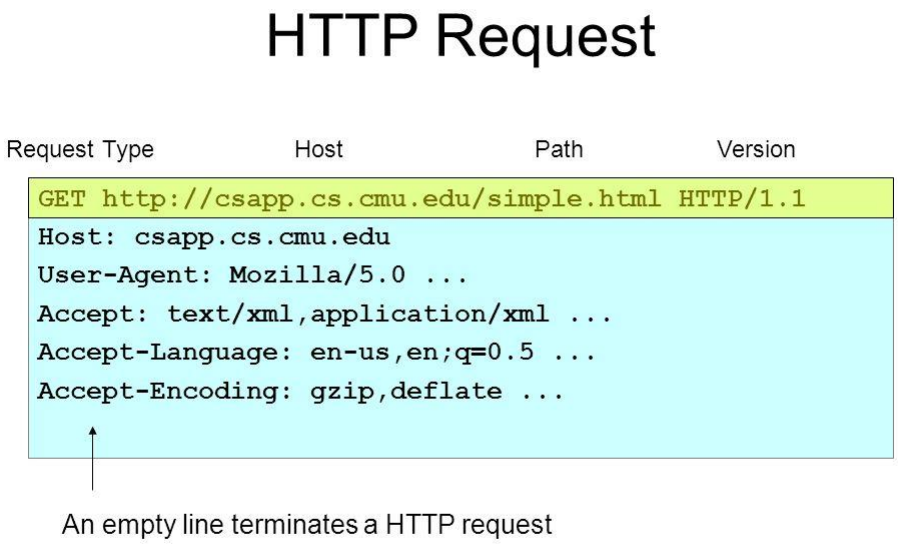
- Мы вводим адрес сайта в поисковую строку браузера.
- Браузер обращается к DNS-серверу, где находится IP-сайта.
- По IP информация (компоненты сайта), которая находится на веб-сервере, собирается (как пазл), и мы получаем определённую картинку.

Здесь включается в работу протокол TCP/IP, который можно назвать «службой доставки». Передаются по этой службе доставки HTTP-пакеты: либо HTTP-запрос, либо HTTP-ответ.

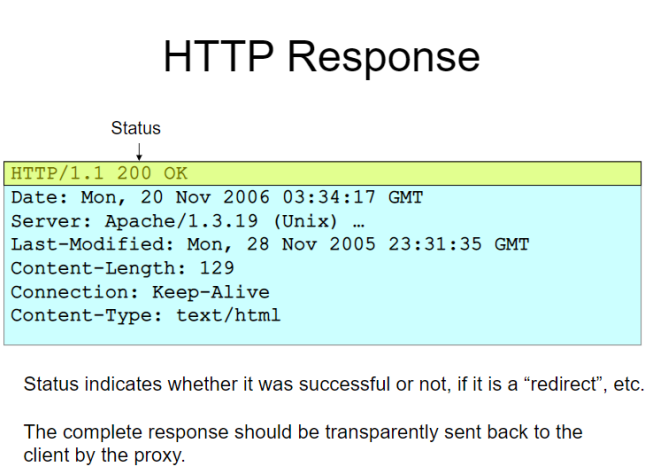
Рассмотрим подробнее **HTTP-запрос**. Это тот пакет, который отправляет браузер серверу, чтобы сервер дал ответ.

Какие данные несут в себе заголовки HTTP-запроса?

- **Host** — здесь указывается имя сайта, которое нужно отдать серверу.
- **User-Agent:** версия браузера.
- Заголовок **Accept,** в котором браузер может сказать серверу, какие именно данные он ожидает увидеть. В данном случае браузер ждёт от сервера XML-страницу.
- Ещё браузер может дать дополнительную информацию о себе, например: ожидает язык en-us, понимает архивирование данных и так далее.




Ответ :




- Первым, что увидит браузер, будет такая строчка: **HTTP/1.1** — версия протокола HTTP.
- **200** — это статус HTTP-ответа. Всё хорошо, сервер возвращает данные. Один из самых известных статусов, с которым вы знакомы — 404 — «Страница не найдена».
- Дальше идёт дата, имя сервера, последние внесенные изменения.
- Важное поле **Content-Type** — здесь говорится о том, что за данные получены в ответе. В нашем случае — текст HTML. В зависимости от того, какой тип данных будет в ответе, будет изменяться значение поля Content-Type: это могут быть и картинки, видео, графика, звуки.

Давайте рассмотрим более подробно, как работают сайты.

Прямо сейчас вы находитесь на странице сайта `skillfactory.ru` и видите текст с картинками. Чтобы создать такую страничку и красиво отформатировать, используются HTML и CSS.

**HTML** — это язык гипертекстовой разметки, с помощью которого создаётся структура веб-страниц.

**CSS** отвечает за стили, такие как фон, цвета, макеты, интервал и анимация.

В то время как CSS определяет внешний вид веб-страницы, HTML формирует её структуру (скелет) посредством заголовков, списков и других подобных элементов. Браузер (Google Chrome, Яндекс.Браузер, Safari, Firefox и так далее) преобразовывает HTML-разметку и отображает страницы в том виде, в котором мы привыкли: не просто текст разметки, а картинки, списки и текст с красивым шрифтом.

Каждый сайт состоит из набора таких страниц. Их может быть 1–2, а могут быть сотни. Когда вы заходите на какую-то страничку в Интернете, браузер отправляет запрос серверу этого сайта, и тот ему отвечает. Теперь давайте разберёмся, а чем он именно отвечает.

Например, вы открываете страницу `https://skillfactory.ru/courses/`. В ссылке этой страницы `https://` — это протокол, `skillfactory.ru` — доменное имя, которое преобразуется в IP-адрес с помощью DNS, и `/courses` — это адрес конкретной страницы, которую браузер хочет получить. Когда сервер `skillfactory.ru` получит этот запрос, он просто вернёт HTML-страничку, которая лежит у него в папке `courses`. Браузер получит её и отобразит. На рисунке ниже отображена эта схема.

Кроме отправки запросов и отображения веб-страниц, браузер также отвечает за:

1. **Скачивание различных файлов:** документы, картинки, фильмы, музыка, программы и так далее.

2. **Хранение данных:** браузер может хранить пароли для доступа к сайтам, использовать закладки для важных ресурсов, сохранять историю посещений.
3. Современные браузеры имеют дополнительные функции, блокирующие опасные сайты или предупреждающие об угрозе, например, **защита от вредоносных программ.**
4. Есть возможность расширения функционала за счёт установки дополнений (плагинов). Они позволяют изменить внешний вид браузера по вашему вкусу, добавить различные информеры (погода, новости, общение), облегчить какие-либо специализированные действия.

**С помощью браузера вы можете**:

- **Обмениваться информацией:** здесь и электронная почта, и всевозможные чаты, форумы, общение в соцсетях.
- **Покупать товары онлайн:** через Интернет-магазины и специализированные сайты объявлений.
- **Обучаться онлайн**: всевозможные дистанционные программы, онлайн-курсы, как тот, который вы проходите, например.

Давайте убедимся, что всё, что вы видите сейчас, создано с помощью HTML.

Для этого нужно выполнить следующие действия:

Откройте *«Инструменты разработчика» (DevTools)*. В большинстве браузеров это можно сделать, нажав клавишу *F12.*

Перейдите на вкладку «*Элементы» («Elements»)*.

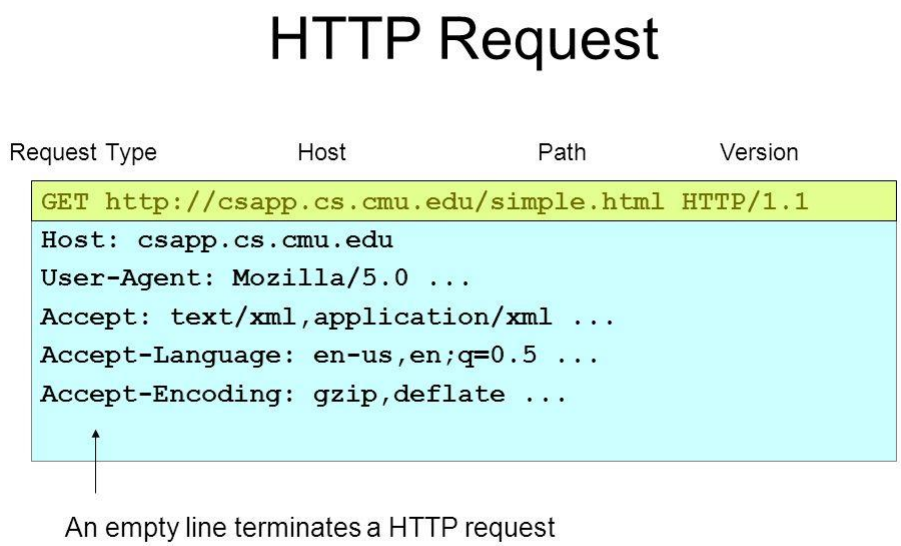
Мы уже рассматривали этот протокол. Он используется при пересылке web-страниц. Запросы, которые браузер отправляет на сервер, называются **HTTP-запросами.**

**Запросы содержат следующие элементы**:

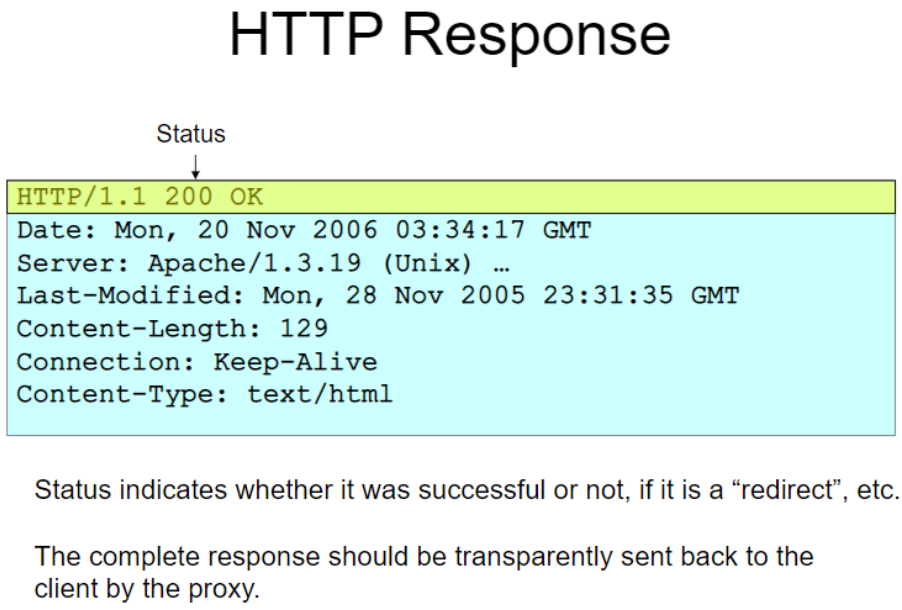
- **HTTP-метод:** обычно глагол (GET, POST) или существительное (OPTIONS или HEAD), определяющее операцию, которую клиент хочет выполнить. Обычно клиент хочет получить ресурс

(используя GET) или передать что-то на сервер (например, загрузить фото), тогда используется POST.
    - Адрес сервера.
    - Версию HTTP-протокола.
    - Заголовки (опционально), предоставляющие дополнительную информацию для сервера.
    - Или тело, например, для метода POST.

Пример запроса на картинке ниже:




А что тогда происходит в ответе? Давайте тоже разбираться! Здесь уже происходит кое-что поинтереснее.



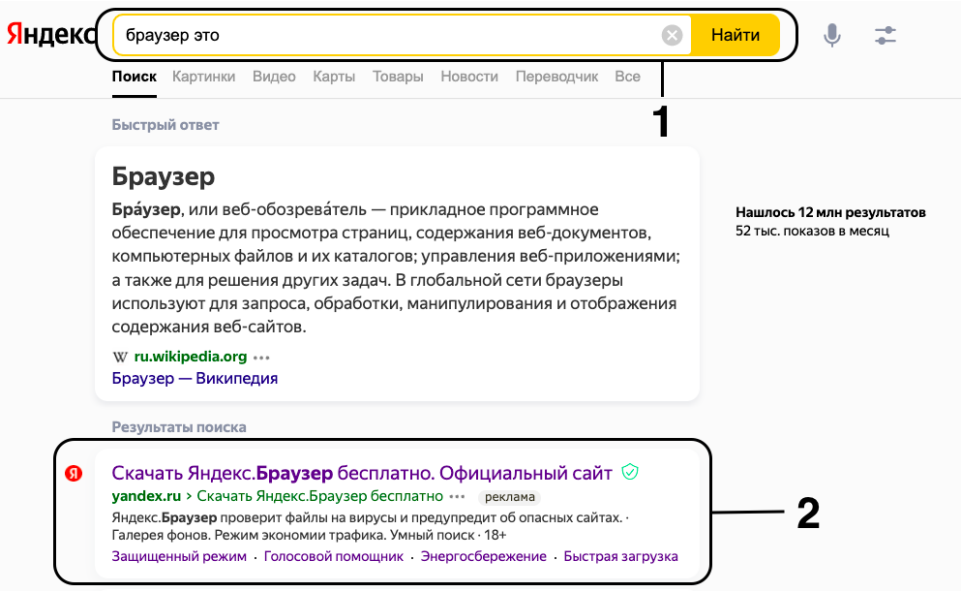

- Первым, что увидит браузер, будет такая строчка: **HTTP/1.1** — версия протокола HTTP.
- **200** — это статус HTTP-ответа. Всё хорошо, сервер возвращает данные.
- Один из самых известных статусов, с которым вы знакомы — 404 — «Страница не найдена».
- Дальше идёт дата, имя сервера, последние внесенные изменения.
- Важное поле **Content-Type** — здесь говорится о том, что за данные получены в ответе. В нашем случае — текст HTML. В зависимости от того, какой тип данных будет в ответе, будет изменяться значение поля Content-Type: это могут быть и картинки, видео, графика, звуки.
- Самое важное для нас в этом ответе — **тело.** Это незакодированный HTML-код.

Прежде чем перейти к созданию веб-сайтов, необходимо узнать их анатомию. Если вы решили заняться профессиональным спортом, то нужно понимать анатомию и физиологию человека, чтобы преуспеть в этом деле. Так и здесь — у веб-сайта есть определенная структура, которая задаётся с помощью написания его разметки в HTML.

**Функциональная часть**

Одна из самых важных частей сайта  — это **функциональная часть.** То, ради чего сайт был создан. Если сайт не будет нести в себе функциональности, тогда зачем он нужен?

Функциональность может быть как основная, так и косвенная. Давайте посмотрим на примере поисковой страницы Яндекса:




Здесь **1** — это основной функционал сайта, поисковая строка, которая позволяет по необходимому запросу получить возможные ответы.

А **2** — это дополнительный функционал, благодаря которому можно реализовать рекламу компании или её продукта.

**Описательная часть**

Вторая часть, которую можно выделить в анатомии веб-страницы — это **описательная часть или контент**: текст, сочетание текста и картинок, видео, таблицы, которые содержат в себе информацию о том, что за веб-сайт открыт в браузере.

Здесь важно учитывать, что такую часть должна иметь каждая страница сайта, так как чаще всего пользователи попадают не на главную страницу, а сразу внутрь сайта.

Например, на Википедии описательная часть — это фрагмент скриншота под номером **3**. Здесь вы сразу понимаете, что вы попали на сайт «свободной энциклопедии, которую редактировать может каждый».

**Навигация**
Третья часть в анатомии сайта — **система навигации**. Этот механизм необходим, чтобы пользователь понимал, в какой части он находится, что ещё есть на сайте, как попасть в другие места сайта.

Система навигации может иметь вид ссылки, меню, поиска, карты сайта и так далее.

Посмотрим на тот же самый сайт Википедии выше. Здесь навигацией является, например, **строка поиска** и **левое меню** (фрагменты под номером **4**).

**Дизайн**
И, конечно же, есть та часть, которая позволяет сделать сайт красочнее и удобнее для пользователей — это **дизайн.**

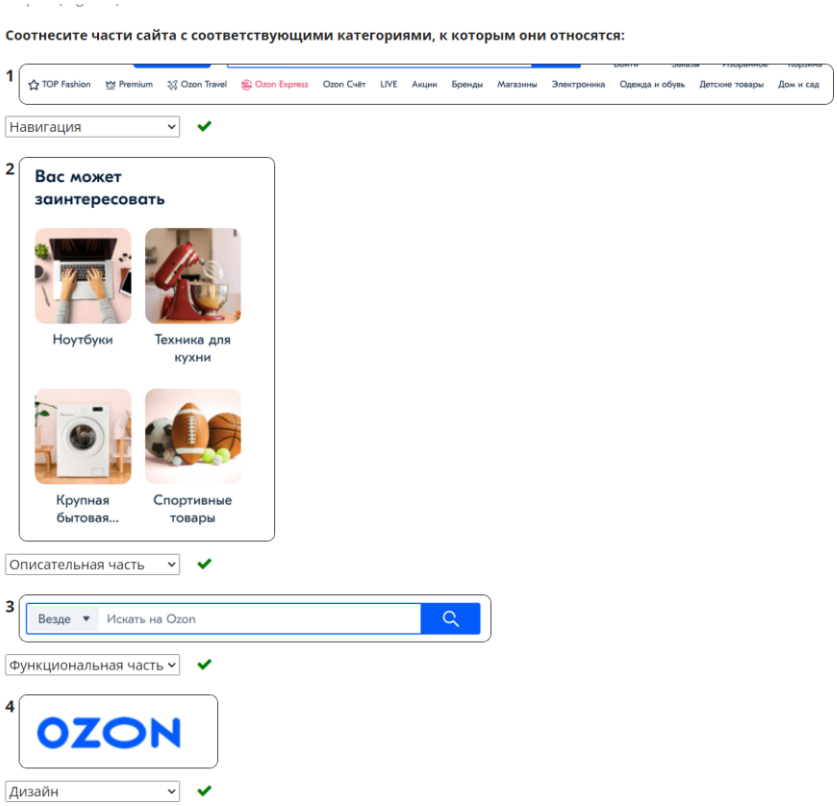
Естественно, функциональная, описательная части и навигация также являются элементами дизайна. И есть **определенные критерии**, которые лучше соблюдать при их реализации:

- **Функциональная часть** должна занимать главенствующее положение на странице. Она должна быть на виду, на неё должно отводиться столько места, сколько ей нужно.
- **Описательная часть** также должна быть на виду, но не перекрывать функциональную. Например, табличка с именем художника и названием картины никогда не будет больше самой картины.
- **Система навигации** — определенно важная, но не основная часть на сайте. Поэтому, с точки зрения дизайна, она не должна бросаться в глаза. Но если у пользователя возникнет вопрос, например, «как найти список статей/фильмов?», то ему нужно сразу понять, с помощью чего можно это сделать.

Если говорить о дизайне веб-сайта в целом, то он должен преследовать **две цели**:

**Структурировать информацию на сайте для лучшего восприятия**
**Создать необходимый образ**

Например, если это сайт *event*-агентства, то он должен быть ярким и красочным. И, наоборот, если это сайт серьезной фирмы, например, страховой компании, то он должен создавать ощущение надежности.




**В структуре сайта можно выделить 4 элемента:**

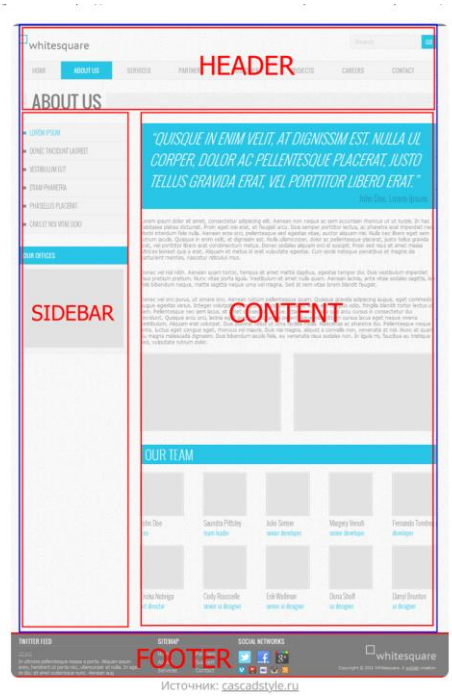
**HEADER** или шапка сайта — верхняя область на сайте, предназначенная для облегчения навигации по странице, первый элемент, который привлекает внимание пользователя. Здесь указывается информация, которая поможет пользователю понять, где он находится и что есть на сайте (например: меню, регистрация, вход в личный кабинет, контакты и т.д.)

**CONTENT** — это информация (статьи, аудио, видео, изображение и т. д.) — все то, ради чего посетитель приходит на сайт. Контент — основа любого

интернет ресурса и от его качества зависит посещаемость и заработок на сайте.

**FOOTER** или подвал сайта — нижняя область на сайте, предназначена для логического завершения страницы, облегчения навигации и размещения дополнительных страниц. Там обычно также дублируется навигация по сайту в виде списка с названием разделов, контакты, ссылки на другие ресурсы компании и т.д.

**SIDEBAR** — это закрепленная боковая панель ресурса, область навигации или вспомогательной информации, графически отделенная от основной области контента. Сайдбар может находиться как с одной стороны сайта, например, слева, так и с обеих сторон. Здесь можно расположить навигационное меню, информационные блоки (например, популярные публикации), функциональные элементы (например, форма поиска, Корзина), объявления с рекламой, предложения товаров и услуг, дополнительные виджеты.



Источник: cascadstyle.ru

Понимая, из чего состоит сайт, можно переходить к тому, как это всё создаётся. И начнем мы с основы сайта — **HTML**.

Мы привыкли считать, что языки, которые используются для создания приложений — это языки приложения. Но работает ли это с HTML? Ведь, по сути, он участвует в создании веб-сайта.

**Характеристики языка программирования:**

➜ Формальная знаковая система.

➜ Наличие лексических, синтаксических и семантических правил, задающих внешний вид программы и действия, которые будут выполнены в результате её исполнения одним из устройств вычислительной машины.

Применим это определение к HTML. Это формальная знаковая система? Да. HTML имеет свои лексические и синтаксические правила.

Но! Задача обычного языка программирования — в обработке данных, а **задача HTML — отображение данных, на нём нельзя произвести вычислений.** Поэтому он не является языком программирования.

**HTML** — это язык гипертекстовой разметки. Для того чтобы работать с HTML, создаётся текстовый документ (HTML-документ).

Давайте теперь разбираться с тем, из чего он состоит.

Каждый HTML-документ имеет свою структуру, которую можно отлично запомнить, сравнив её с телом человека. Посмотрите на картинку:


Каждый элемент обозначается в исходном документе начальным (открывающим) и конечным (закрывающим) тегом (за редким исключением).

**Тег** — это базовый элемент языка разметки, основа HTML-документа.

Закрывающий тег образуется путем добавления слэша `/` перед именем тега: `<имя тега>…</имя тега>`.

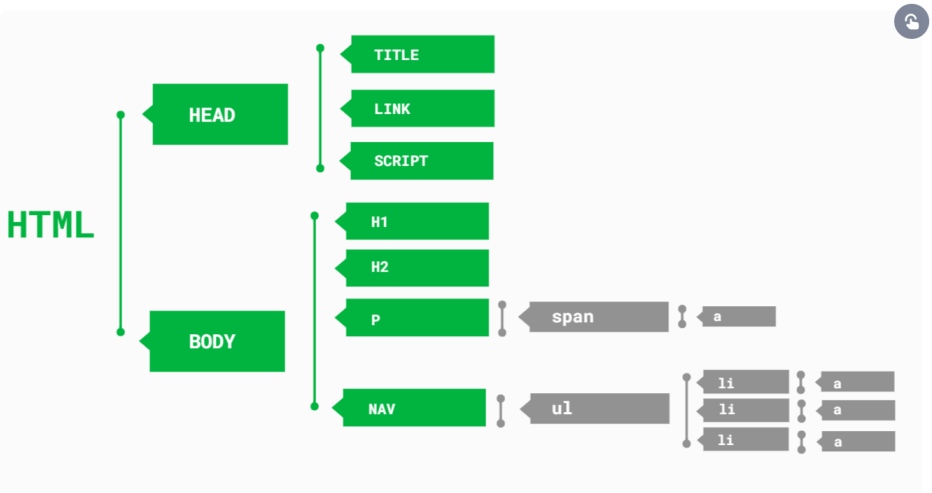
Между начальным и закрывающим тегами находится содержимое тега — контент.

Элементы, находящиеся внутри тега `<html>`, образуют дерево документа, так называемую **объектную модель документа**, **DOM** (Document Object Model). При этом элемент `<html>` является корневым элементом.

Далее мы можем видеть два ответвления в дереве — это элементы `<head>` и `<body>`.

- В <head> мы выделяем информацию, которая будет потом доступна при поиске сайта, задаём заголовок документа, а также подключаем CSS-файлы и скрипты.
- В <body> добавляются элементы, которые пользователь будет видеть на сайте: меню, текст, картинки и так далее.

Для каждого элемента указываем дальше уточнение, если что-то может быть добавлено. Например, мы можем выделить список <ul>, состоящий из нескольких элементов <li>, каждый из которых будет содержать ссылку <a>. Можно предположить по данной структуре, что это может быть, например, навигация по сайту.




Когда браузер запрашивает страницу и получает в ответе от сервера её исходный HTML-код, он не может сразу его запустить. Ему необходимо сначала его разобрать. В процессе анализа и разбора HTML-кода браузер строит на основе него **DOM-дерево**, то есть объектную модель HTML-кода страницы.

«Потрогать» DOM мы с вами напрямую не можем, но если откроем код любой страницы в браузере, то увидим, что страница состоит из **объектов**.

**DOM** — это объектная модель документа, которую браузер создаёт в памяти компьютера на основании HTML-кода, полученного им от сервера. В соответствии с объектной моделью документа, каждый HTML-тег является объектом. Вложенные теги являются «детьми» родительского элемента. Текст, который находится внутри тега, также является объектом.

### Супер-функция DOM. Автоисправление

Сейчас мы с вами вносили исправления в HTML-код сами. Но когда браузер сталкивается с некорректным HTML-кодом, DOM умеет его автоматически исправлять.

Например, в начале документа всегда должен быть тег `<html>`. Даже если его нет в документе — он будет в дереве DOM, браузер его создаст. То же самое касается и тега `<body>`.

Или если HTML-файл состоит из единственного слова «Привет», браузер обернёт его в теги `<html>` и `<body>`, добавит необходимый тег `<head>`.

Давайте посмотрим, как это работает. Попробуйте создать HTML-документ в *WebStorm*, напишите любую фразу, лучше на английском, так как русскоязычный текст браузер может не распознать. И посмотрите, что будет.

Сегодня практически любая компания имеет свой сайт. Когда мы хотим узнать какую-либо информацию о компании, мы заходим в браузер, пишем запрос в поисковике и ищем сайт этой компании. Конечно, сайты не создаются сами, это мы прекрасно понимаем.

Но вот как они создаются? Способов, с помощью которых сегодня можно создать свой сайт, множество.

Сайт состоит из двух частей: **пользовательской** и **серверной**.

На странице веб-сайта в браузере вы видите текст, кнопки, панели, изображения, видео или другие объекты. Эта пользовательская часть сайта создаётся благодаря работе **frontend-разработчика**: визуализация, интерактивность, понятность интерфейса — всё, над чем работает такой специалист.

Но чтобы создать свой сайт, не обязательно владеть навыками frontend-разработчика: HTML, CSS и JavaScript.

Есть множество инструментов с готовыми решениями, которые позволяют создать сайт человеку, абсолютно не знакомому с анатомией веб-страниц. Их называют **конструкторами сайтов** (website builders).

Давайте рассмотрим несколько примеров таких инструментов.

### Bubble

Bubble служит для создания веб-приложений без использования кода. Инструмент воспроизводит все основные опции веб-программирования в понятном визуальном интерфейсе.

### Tilda

Tilda позволяет создавать впечатляющие, красивые и легкие в управлении сайты, Интернет-магазины, лендинги и спецпроекты без использования программирования.

### Wordpress

WordPress — это одна из наиболее популярных CMS-систем управления контентом на сайте (создание и публикация записей, размещение виджетов, изменение дизайна, расположение и отображение различных элементов и так далее).

В прошлом юните мы узнали про инструменты с готовыми решениями, но если вы хотите создать свой уникальный сайт, вам необходимо изучить инструменты frontend-разработки — **HTML, CSS и JavaScript.**

Давайте разбираться, что это за инструменты и зачем они нужны.

Начнём с основы — HTML.

**HTML** (HyperText Markup Language, язык гипертекстовой разметки) — это **не язык программирования.** В первую очередь, это логическая разметка страницы: HTML описывает структуру страницы, а не её поведение.

Чтобы было проще понять, можно представить, что разметка страницы аналогична реферату, который нужно отредактировать в соответствии с необходимыми требованиями, задать структуру. HTML как раз задаёт нужную структуру странице.

Ниже представлен пример того, как будет выглядеть заголовок и подзаголовок на странице.

Чтобы сайты были такими, какими мы их привыкли видеть, разработчики используют нижеприведенные инструменты. Они делают работу программистов более творческой и расширяют простор возможностей для реализации своей уникальной задумки.

**CSS** (Cascading Style Sheets, каскадные таблицы стилей) — язык таблиц стилей, который позволяет прикреплять стиль, например, шрифты и цвет.

**Особенности CSS:**

- описывает, как элементы HTML должны отображаться на экране компьютера, смартфона или на других носителях;
- экономит много времени при работе: может контролировать стили сразу нескольких веб-страниц;
- внешние таблицы стилей (файлы, содержащие весь CSS-код) хранятся в файлах CSS.

**JavaScript** (JS) — логический язык программирования, который можно использовать для изменения содержимого веб-сайта, заставить сайт вести себя по-разному **в ответ на действия пользователя**. Общее использование: окна подтверждения, призывы к действию и добавление новых идентификаторов к существующей информации.

В прошлых двух юнитах мы познакомились с различными инструментами frontend-разработки, но это всё только внешняя оболочка сайта.

Внутренняя часть сайта — это работа **backend-разработчиков**. Они отвечают за логику, работоспособность и правильное функционирование сайта. Управлять этой частью сайта может только администратор сайта через специальный интерфейс или через непосредственную работу с кодом сайта.

Когда пользователь совершает запрос, он передаётся на сервер. Так вот, backend отвечает за правильное выполнение процесса обработки запроса, фильтрации и отправки ответа обратно.

**Для backend-разработки используются такие языки программирования, как PHP, Python, JavaScript, Java, Kotlin, Swift, Golang.**

Сегодня **Python** всё чаще используют для веба, и он уже «наступает на пятки» Java, обогнав PHP в рейтинге.

Синтаксис языка очень простой, его часто используют даже для обучения детей.

Python используется и в веб-разработке, и для создания приложений. Например, он используется в YouTube, Instagram, Facebook, Pinterest, Google, Netflix.

В «вебе» использование Python упрощает процесс обработки адресов, обращение к базам данных и создание HTML, отображающихся на пользовательских страницах.

Для веба лучшими фреймворками считаются Django и Flask.

- **Flask** минималистичен, для реализации задач необходимо подключение внешних модулей, за счёт чего возможно разработать своё уникальное решение.
- **Django** имеет огромное количество встроенных пакетов и готовых решений, о нём часто говорят «всё включено».

Для разработки сайтов есть помощник — **CMS** (Content Management Software или System, управляющее содержимым программное обеспечение). Это написанная на скриптовом языке PHP специализированная программа, обеспечивающая быстрое создание сайта и управление его содержимым.

Устанавливается такое ПО на виртуальный хостинг или локальный сервер.