Основы клиент-серверного взаимодействия .

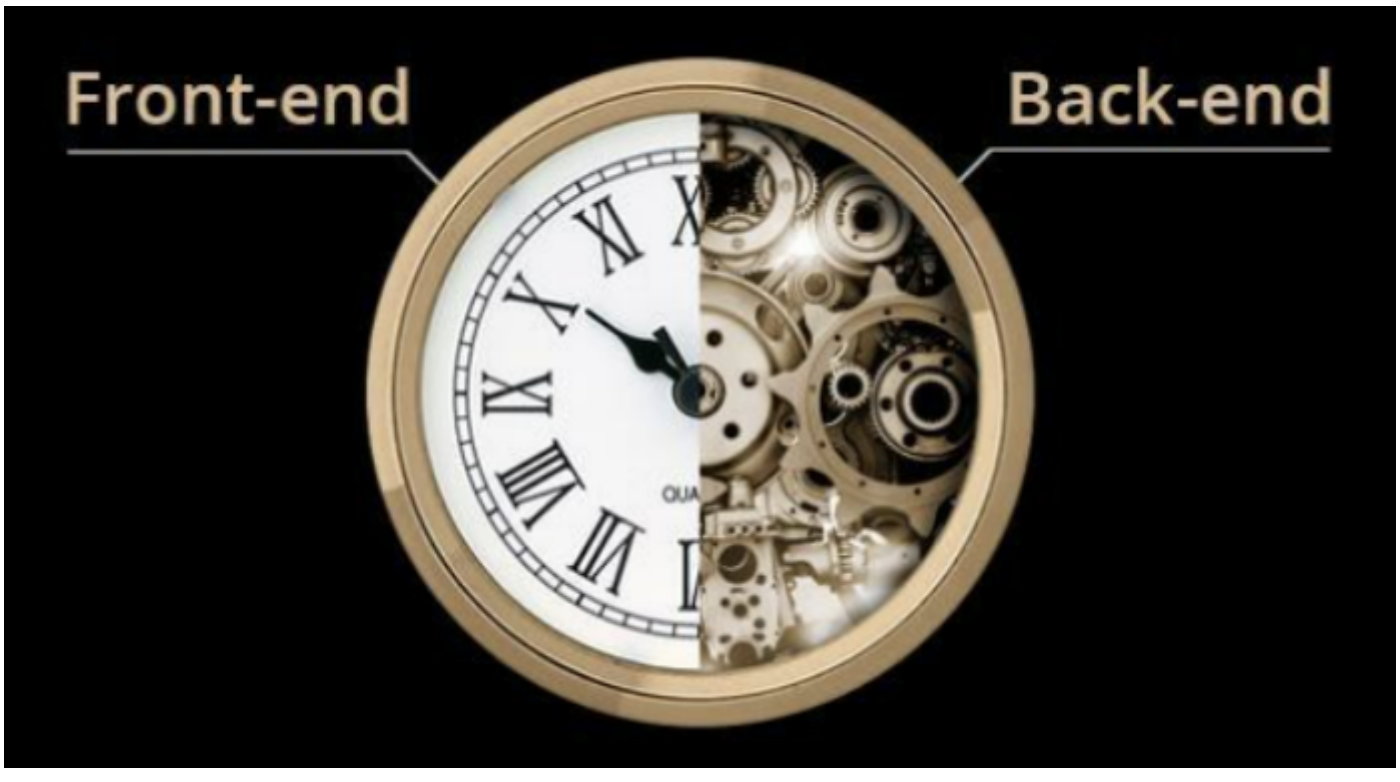
Frontend и Backend.

Frontend - система скриптов , которая работает на стороне пользователя (в браузере). **То что видит пользователь заходя на сайт. Используются технологии HTML , CSS и JavaScript.**

К frontend относятся:

1. создание дизайн-макета сайта
2. верстка сайтов и шаблонов для CMS
3. привязка к пользовательскому интерфейсу специальных скриптов , отвечающих за визуализацию и web-анимацию.

К **Backend** относится работа самой программы : как работает ее код , как взаимодействует приложение с сервером , работа с базами данных , и множество других действий, скрытых от глаз пользователя.



Frontend

Основы HTML

HTML (Hyper Text Markup Language - язык гипертекстовой разметки) не является языком программирования , это язык разметки.

Используется , чтобы сообщать вашему браузеру , как отображать веб страницы , которые вы посещаете .

Разметка	Результат
  - HTML - CSS - JavaScript - PHP 	1. HTML 2. CSS 3. JavaScript 4. PHP

тэг ol используется для нумерации используется парно с тэгом li

Структура документа HTML

```
<html>
<head>
    <title>Заголовок</title>
    <meta charset="UTF-8">
    <link rel="icon" href="favicon.ico">
</head>
<body>
    Тело документа
</body>
</html>
```

Основы CSS

CSS(Cascading Style Sheets - каскадные таблицы стилей) - формальный язык описания внешнего вида документа.Он позволяет применять стили выборочно к элементам , т.е. **позволяет оформлять внешний вид веб-страниц**, написанных с помощью языков разметки HTML.

Основы CSS

Варинаты написания CSS

Inline CSS

```
<p style="color: blue;">This is a paragraph.</p>
```

Internal CSS

```
<head>
  <style type = text/css>
    body {background-color: blue;}
    p { color: yellow;}
  </style>
</head>
```

External CSS

```
<head>
  <link rel="stylesheet" type="text/css" href="style.css">
</head>
```

Inline CSS внутри кода HTML с помощью тэга p и атрибута style

Internal CSS

с помощью тега style с атрибутом text, с написанием внутри стилей .(применяется для объядинения объектов стилями) .

External CSS

используется для того что указывать и групировать стили с ссылками на стиль.

JavaScript

Один из языков программирования котрый позвоялет нам писать веб приложения.

Мультипарадигменный **язык программирования , позволяющий создавать скрипты, которые встраиваются в HTML** - страницы и выполняются в браузере посетителя страницы , а не на сервере .

Все современные браузеры поддерживают язык **JavaScript.**

Позволяет оформить сайт отедльными действиями с добавлением анимации.

Что такое девелоперская консоль ?

Консоль разработчика помогает :

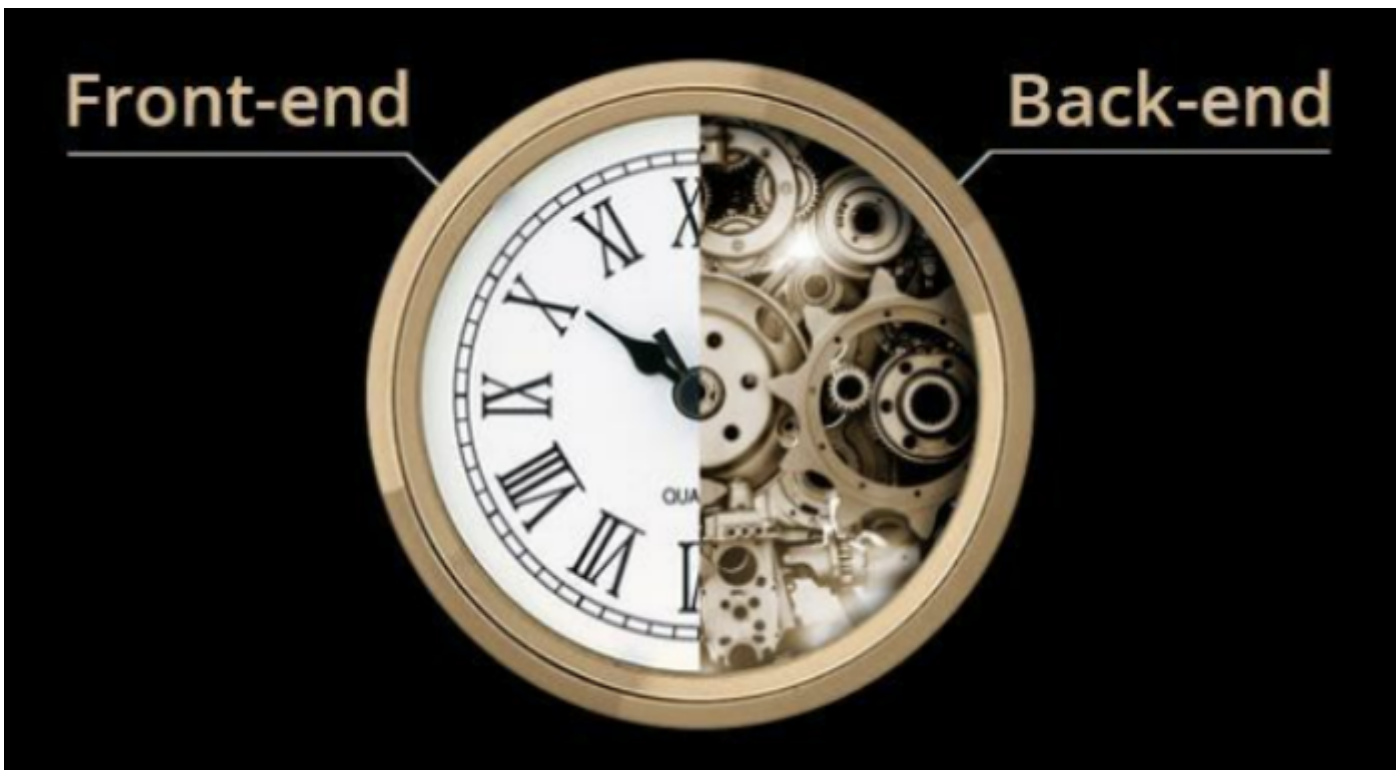
1. **читать ошибки в удобном виде ;**
2. **получать массу полезной информации о выполнении скриптов;**
3. **запускать команды JavaScript и пр.**

В каждом браузере есть свой инструмент разработчика - это называют **консолью**.

Консоль в браузере - это инструмент , с помощью которого можно:

1. **посмотреть наполнение страницы , выводимой браузером;**
2. **найти существующие ошибки (что чаще всего и делают тестировщики);**
3. **исправить эти ошибки просто и быстро (что делают девелоперы)**
4. **замерить различные показатели и манипулировать страницей.**

Backend



Как работает интернет ?

Интернет - глобальная сеть соединенных между собой компьютеров , если оставить за скобками все дополнительные устройства (роутеры или свитчи).

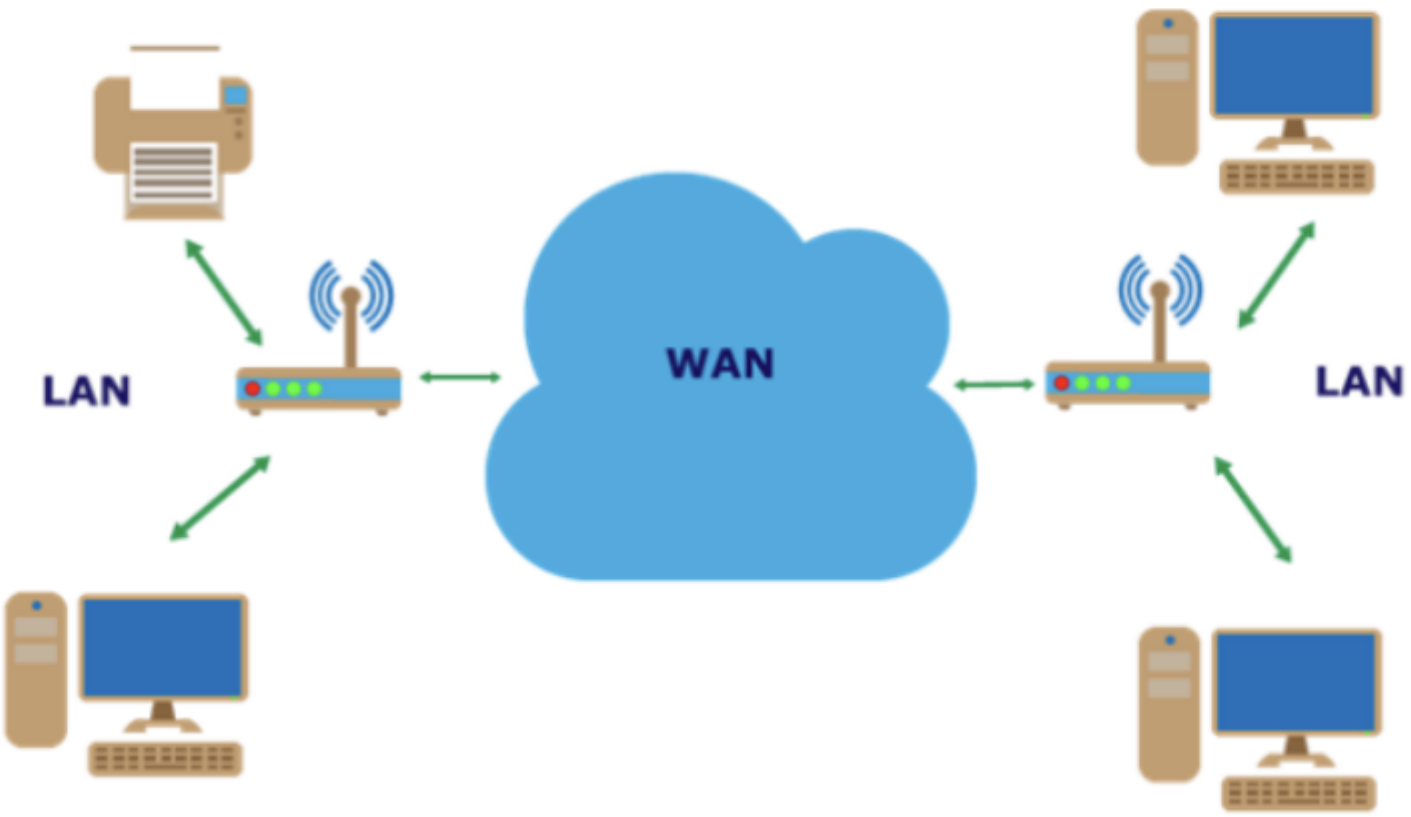
В этой сети нет определенного центра , все компьютеры равноправны между собой.

Как работает интернет ?

Компьютеры соединяются между собой напрямую с помощью локальной сети (LAN) или через глобальное соединение (WAN).

LAN (Local Area Network) - это локальная сеть. Проще говоря , это компьютеры , которые соединены между собой на не очень большом расстоянии.

WAN (Wide Area Network) - это глобальная компьютерная сеть интернета.



Клиент серверное взаимодействие.

Клиент

Под клиентом понимают обычный браузер (их может быть много)

Клиент - процесс , который запрашивает обслуживание от сервера.

Процесс не является клиентом по каким то параметрам своей структуры , он является клиентом только по отоношению к серверу.

Путь взаимодействий :Клиент-Сервер-База данных.

Клиентом для сервера может выступать и другой сервер.

При взаимодействии клиента и сервера инициатором диалога с сервером, как правило, является клиент. Сервер сам не инициирует совместную работу. Это не исключает , однако того , что сервер может извещать клиентов о каких то зарегестрированных им событиях .

Если возникает ошибка или не выполняется какое то действие которое должно выполнятся по ожиданию не обходимо найти источник почему не выполняется данное действие.**(Тестировщик в первую очередь проверяет клиент.)**

Веб сервер

Веб сервер - это сервер , который может получать входящие веб-запросы, а также знает , как их следует обрабатывать и отвечать на них.

Некоторые запросы являются статическими (html-файлы,изображения и т.д.),другие - динамическими. В случае динамических запросов веб-сервер будет знать , куда направлять обработку запроса.

Веб-сервер предназначен для обслуживания содержимого HTTP.

Его задача - принять запрос и понять , что с ним делать : определить, к какому файлу перешел запрос, отправить его по нужному адресу, обработать этот файл и выдать ответ пользователю.

Дополнительные задачи веб-сервера:

1. реализация журнала ошибок и обращений(логи);
2. какую папку с файлами надо обрабатывать , по каким правилам;
3. аутентификация и авторизация (может определить, кто обращается к файлам и выдает им доступ к ним).

Идентификация
— это процедура распознавания субъекта по его идентификатору (проще говоря, это определение имени, логина или номера)

Аутентификация
– это процедура проверки подлинности (пользователя проверяют с помощью пароля, письмо проверяют по электронной подписи и т.д.)

Авторизация
– это предоставление доступа к какому-либо ресурсу (например, к электронной почте).

Модель взаимодействия клиент-сервер:

Если мы посмотрим на архитектуру с позиции сайта, то первый уровень можно считать браузером, с помощью которого посетитель заходит на сайт. Второй уровень – это ПО сервера, а третий уровень – это база данных.

Когда тестировщик имеет опыт работы с кодом и базой данных , то он начинает проверять информацию на сервере и в Базе данных.

Балансировщик

Вспомогательная программа которая помогает распределять нагрузку.

Для бесперебойной работы веб приложений важно , чтобы взаимодействие между клиентом и серверами было надежным.

Для этого система имеет несколько серверов, между клиентом и серверами установлен балансировщик , который обеспечивает качественное взаимодействие между ними.

Предназначены для уровновешивания нагрузки

.

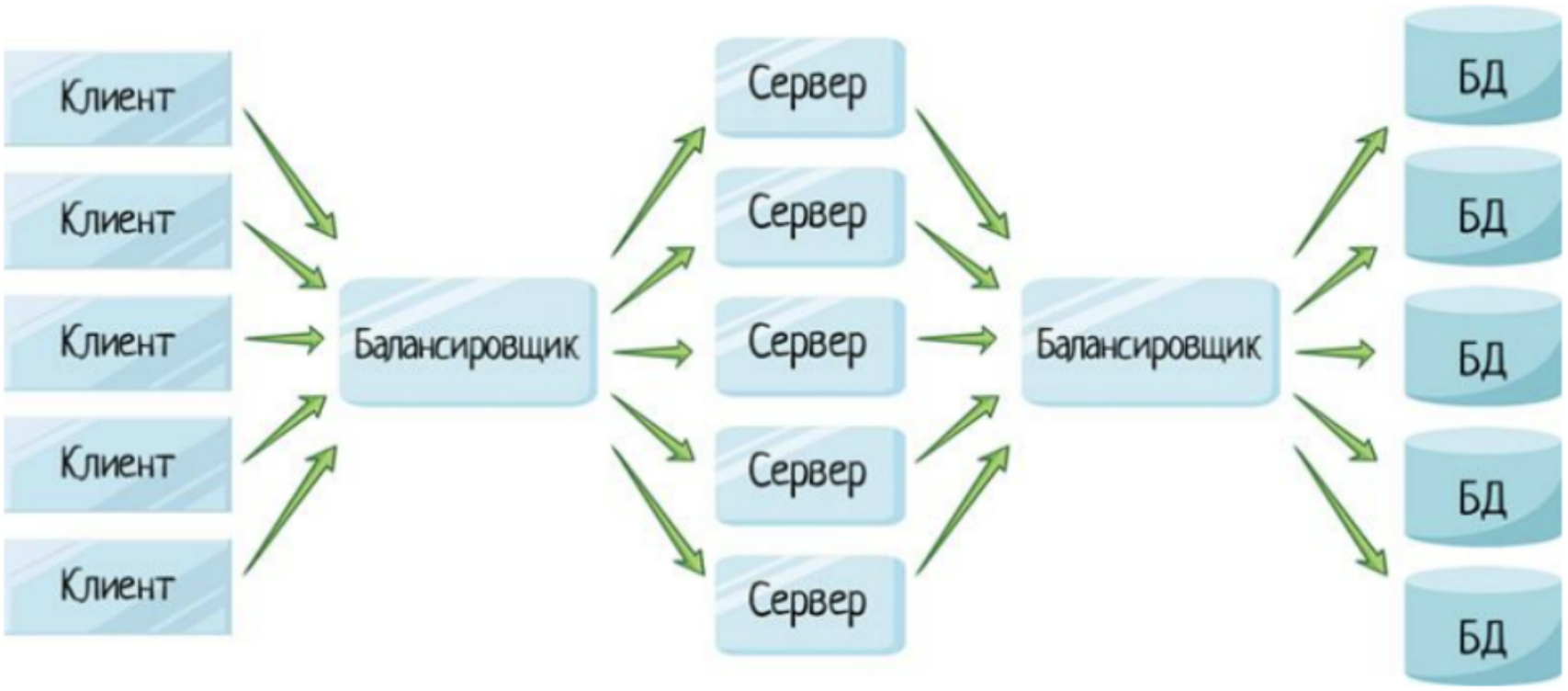
Если одни из серверов перестанет работать , то балансировщик определит это и будет работать с другими серверами , а пользователь даже не заметит , что в системе произошла проблема.

Тестировщику необходимо переодически проверять работу серверов и балансировщика ,чтобы понимать что все работает исправно.

Многоуровневая модель

Суть многоуровневой архитектуры в том что запрос клиента обрабатывается сразу несколькими серверами.

Многоуровневые приложениями могут быть такие как : Facebook ,Vk.



СУБД

Это данные обычно в виде взаимосвязанных таблиц.

СУБД (система управления базами данных) - **ПО, с помощью которого можно создавать базы данных(БД) и проводить над ними различные операции:**

1. обновлять
2. удалять
3. выбирать
4. редактировать
5. и т.д.

СУБД гарантирует сохранность , целостность ,безопасное хранение данных и позволяет выдавать доступ к администрированию БД.

- **Для выполнения операций используется механизм запросов.**
- **Результатом выполнения запросов является либо отобранное по определенным критериям множество записей , либо изменения в таблицах.**
- **Запросы к базе формируются на специально созданном для этого языке , который так и называется “язык структурированных запросов”(SQL- Structured Query Language)**

SQL

SQL-структурирвоанный язык запросов

```
SELECT * FROM table_name;

UPDATE table_name SET column1 = value1, column2 = value2, WHERE id=2;

DELETE FROM table_name WHERE id=10;
```

```
SELECT - извлекает данные из базы данных
UPDATE - обновляет данные в базе данных
DELETE - удаляет данные из базы данных
INSERT INTO - вставляет новые данные в базу данных
CREATE DATABASE - создает новую базу данных
ALTER DATABASE - изменяет базу данных
CREATE TABLE - создает новую таблицу
ALTER TABLE - изменяет таблицу
DROP TABLE - удаляет таблицу
```

GET	POST	PUT	PATCH	DELETE
find	create	replace	update	destroy
SELECT	INSERT	UPDATE	UPDATE	DELETE

URL | HTTP / HTTPs

URL (Uniform Resource Locator) - **это адрес , который выдан уникальному ресурсу в интернете.** В теории каждый корректный URL ведет на уникальный ресурс.

- Напрмер, ресурсам могут быть HTML - страница , CSS - файл, изображение.
- На практике существуют некоторые исключения , когда , например , URL ведет на ресурс , который больше не существует или который был перемещен.
- Ресурс, доступный по URL, и сам URL обрабатывается веб-сервером.

URL состоит из различных частей, некоторые из которых являются обязательными, а некоторые - факультативными. Рассмотрим наиболее важные части на примере:

http://www.example.com:80/patin....

http:// - protocol (протокол), часть URL ,которая указывает браузеру , какой протокол использовать.

Обычно это HTTP-протокол или его безопасная версия - HTTPS.

Интернет требует эти 2 протокола, но браузеры часто могут использовать и другие протоколы, например mailto: (чтобы открыть почтовый клиент) или ftp: для запуска передачи файлов , так что не стоит удивляться , если вдруг увидите другие протоклы.

Для передачи веб-страниц используется **HyperTextTransferProtocol (HTTP)** - протокол передачи гипертекста. Текст веб-страниц назван гипертекстом из за того , что веб-страницы связаны между собой ссылками , образуя тем самым гиперпространства.

HTTPS (Hyper Text Transfer Protocol Secure) - расширение протокола HTTP с поддержкой шифрования в целях повышения безопасности.Данные в протоколе HTTPS передаются поверх криптографических протоколов SSL или TLS.

TLS (transport layer security - Протокол защиты транспортного уровня), как и его предшественник **SSL(более старая версия TLS) (secure sockets layer - слой защищенных сокетов)** - это криптографические протоколы, обеспечивающие защищенную передачу данных между узлами в сети Интернет.

TLS и SSL используют асиметричное шифрование для аунтентификации для сохранения целостности сообщений.

Техническое отличие : **HTTP обычно использует порт соединения 80, тогда как HTTPS - порт 443**. При необходимости системный администратор может указать другой порт , но они всегда будут разными для HTTP и HTTPS протоколов.

Функции HTTP - протокла

1. передает содержимое веб-страниц;
2. передает ответ сервера на запрос (например -всё ОК, страница не найдена , нет прав и тд.)
3. передает “заголовки” технической информации ,например , кодировку или cookies;
4. передает данные со стороны клиента , в том числе загружает файлами и многое другое.

JSON | XML

JSON (JavaScript Object Notation) - это текстовый формат обмена структурированных данными , основанный на JavaScript.

Его легко переобразовать в структуру данных для большинства языков программирования : числа,строки,логические переменные,массивы.

JSON Example:

```
{
"employees":[
  { "firstName":"John", "lastName":"Doe" },
  { "firstName":"Anna", "lastName":"Smith" },
  { "firstName":"Peter", "lastName":"Jones" }
]}
```

Каждый раз, когда мы вводим пароль на Facebook или в другом веб-приложении, сервер воспринимает это как следующий текст:

```
{
  "login": "Netoologe@gmail.com",
  "password": "net190!93theBestF0RStu41ent$",
  "admin": true
}
```

XML (eXtensible Markup Language - расширяемый язык разметки) - это язык описания данных. Он используется для хранения и передачи информации в удобном для человека и компьтерном виде.

XML Example:

```
<employees>
  <employee>
    <firstName>John</firstName> <lastName>Doe</lastName>
  </employee>
  <employee>
    <firstName>Anna</firstName> <lastName>Smith</lastName>
  </employee>
  <employee>
    <firstName>Peter</firstName> <lastName>Jones</lastName>
  </employee>
</employees>
```

Особенности:наличие парных тегов как и в html , сложность в читаемости кода из за большого количества информации.

Наиболее частое распространенное использование JSON|XML - пересылка данных от сервера к браузеру.

XML - это язык разметки , который используется для добавления дополнительной информации в обычный текст , а **JSON - это способ предоставления структурированных данных** в удобном формате.

Разница между JSON и XML

XML

```
<empinfo>
    <employees>
        <employee>
            <name>James Kirk</name>
            <age>40</age>
        </employee>
        <employee>
            <name>Jean-Luc Picard</name>
            <age>45</age>
        </employee>
        <employee>
            <name>Wesley Crusher</name>
            <age>27</age>
        </employee>
    </employees>
</empinfo>
```

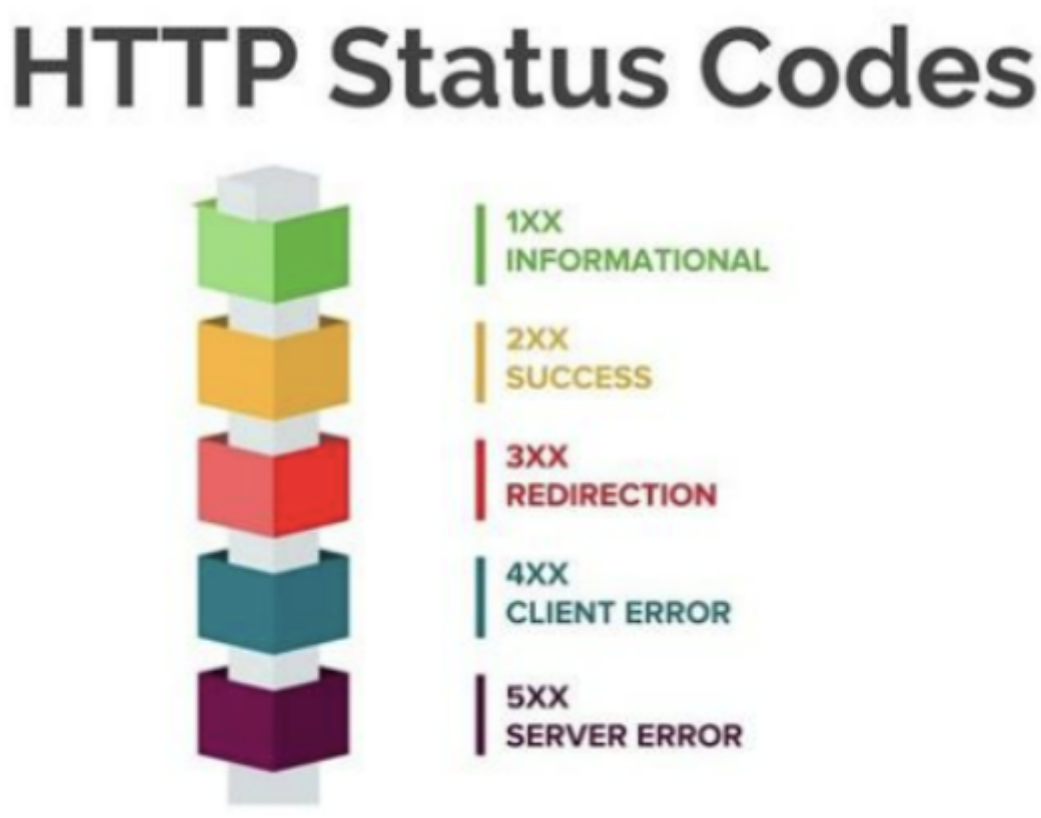
JSON

```
{  "empinfo" :
    {
        "employees" :  [
        {
            "name" : "James Kirk",
            "age" : 40,
        },
        {
            "name" : "Jean-Luc Picard",
            "age" : 45,
        },
        {
            "name" : "Wesley Crusher",
            "age" : 27,
        }
                    ]
    }
}
```

- в XML используются теги , и передается строками
- в JSON мы используем поле и значение поля

Коды ответов клиента и сервера

Код состояния протокола HTTP - это числовое значение состояния протокола, определяемое результатом обработки запроса.



1XX- информационный статус . Код, отвечающий за передачу данных.Такие коды временны и показывают , что запрос принят и обрабатывается.

2XX- успешные .Код успешной обработки запроса. Сервис получил и обработал запрос.

3XX - статус о перемещении.Сервер сигнализирует, что для выполнения запроса нужно предпринять дополнительные действия, например , перейти на другой адрес.

4XX-клиентская ошибка (404 not found).Ошибка на стороне клиента.Возможно пользователь что то сделал неправильно, и поэтому запрос не может быть успешно обработан.

5XX-ошибка сервера.По какой то внутренней причине сервер не может выполнить пользовательский запрос.

Типы запросов

Метод	Описание
GET	Получение данных
POST	Отправка данных
PUT	Обновление данных
PATCH	Частичное обновление данных
DELETE	Удаление
HEAD	Возвращает заголовки
CONNECT	Туннелирование
OPTIONS	Возвращает поддерживаемые сервером методы
TRACE	Отладка (возвращает данные запроса)

GET - запрашивает информацию из указанного источника и не влияет на его содержимое. Запрос доступен для кэширования данных и добавления в закладки. Длина запроса ограничена (макс.длина URL - 2048)

POST позволяет передавать большие объемы данных в бинарном виде , те без искажений и изменений передаваемых данных.

PUT загружает содержимое запроса на указанный в запросе URL .Если по заданному URL ресурсу нет , то сервер создает его, возвращая статус 201(Created).

PATCH частичная замена данных.

DELETE удаляет указанный ресурс.

OPTIONS используется для описания параметров коммуникации между клиентом и сервером.

HEAD аналогичен методу GET , однако в ответе сервера содержится только заголовок,без тела.Обычно применяется для того , чтобы проверить существует ли ресурс по указанному адресу , а также не изменился ли он с момента последнего обращения.

CONNECT проеобразует соединение запроса в прозрачный TCP/IP - тунель.

Logs

Зачем QA-инженеру понимать Логи ?

Анализируя данные сервера или браузера QA-инженер всегда использует запротоколированную информацию от пользователья или от сервера , которая называется Логи .

Логи - это история операций , в которую заносятся данные о действиях пользователей и программы.

Данные в логах решаюшие для тестироващика при орпеделении характера бага.