**Со́кет** (англ. socket - разъем) - название программного интерфейса для обеспечения обмена данными между процессами. Процессы при таком обмене могут исполняться как на одной ЭВМ, так и на различных ЭВМ, связанных между собой сетью. Сокет - абстрактный объект, представляющий конечную точку соединения.

Следует различать клиентские и серверные сокеты. Клиентские сокеты грубо можно сравнить с конечными аппаратами телефонной сети, а серверные - с коммутаторами. Клиентское приложение (например, браузер) использует только клиентские сокеты, а серверное (например, веб-сервер, которому браузер посылает запросы) - как клиентские, так и серверные сокеты.

**Принципы сокетов**

Для взаимодействия между машинами с помощью стека протоколов TCP/IP используются адреса и порты. Адрес представляет собой 32-битную структуру для протокола IPv4, 128-битную для IPv6. Номер порта - целое число в диапазоне от 0 до 65535 (для протокола TCP).

Эта пара определяет сокет («гнездо», соответствующее адресу и порту).

В процессе обмена, как правило, используется два сокета - сокет отправителя и сокет получателя. Например, при обращении к серверу на HTTP-порт сокет будет выглядеть так: 194.106.118.30:80, а ответ будет поступать на mmm.nnn.ppp.qqq:xxxxx.

Каждый процесс может создать «слушающий» сокет (серверный сокет) и привязать его к какому-нибудь порту операционной системы (в UNIX непривилегированные процессы не могут использовать порты меньше 1024).

Слушающий процесс обычно находится в цикле ожидания, то есть просыпается при появлении нового соединения. При этом сохраняется возможность проверить наличие соединений на данный момент, установить тайм-аут для операции и т. д.

Каждый сокет имеет свой адрес. ОС семейства UNIX могут поддерживать много типов адресов, но обязательными являются INET-адрес и UNIX-адрес. Если привязать сокет к UNIX-адресу, то будет создан специальный файл (файл сокета) по заданному пути, через который смогут сообщаться любые локальные процессы путём чтения/записи из него (см. сокет домена Unix). Сокеты типа INET доступны из сети и требуют выделения номера порта.

Обычно клиент явно «подсоединяется» к слушателю, после чего любое чтение или запись через его файловый дескриптор будут передавать данные между ним и сервером.

**WebSocket**

WebSocket - протокол связи поверх TCP-соединения, предназначенный для обмена сообщениями между браузером и веб-сервером в режиме реального времени.

В настоящее время в W3C осуществляется стандартизация API Web Sockets. Черновой вариант стандарта этого протокола утверждён IETF.

WebSocket разработан для воплощения в веб-браузерах и веб-серверах, но он может быть использован для любого клиентского или серверного приложения. Протокол WebSocket - это независимый протокол, основанный на протоколе TCP. Он делает возможным более тесное взаимодействие между браузером и веб-сайтом, способствуя распространению интерактивного содержимого и созданию приложений реального времени.

**Разница Socket и WebSocket**

Socket и WebSocket - это разные понятия в принципе. При работе по протоколу WebSocket вы будете использовать обычные сокеты для соединения. Так же как и при работе с другими протоколами будут использованы сокеты (и для работы с http, с ftp и др.).

Например, рассмотрим строку вида - ws://127.0.0.1:15000. В ней ws - это именно указание на то, что при обмене данными будет использован протокол WebSocket. 127.0.0.1 - ip адрес компьютера, 15000 - порт, на который производится подключение. Так вот 127.0.0.1:15000 - эта пара, если можно так выразится, и является сокетом.

Протокол WebSocket создавался для того, чтобы можно было поддерживать длительные неразрывные соединения между браузером (который является клиентом) и веб-сайтом (который является сервером).

Протокол WebSocket не похож на HTTP. Единственное, чем он напоминает HTTP - только одним самым первым запросом на подключение (так называемым рукопожатием/handshake). Это было сделано, потому что изначально протокол рассчитан на работу в браузере и необходимо было определение возможности поддержки его. После того, как соединение установлено, ничего похожего на протокол HTTP в протоколе WebSocket даже близко нет. Именно отсутствие каких-либо наворотов в протоколе WebSocket и дает ему возможность быстрой работы.

**Разница WebSocket и Socket.IO**

Говоря о веб-сокетах, мы имеем ввиду протокол веб-коммуникации, представляющий полнодуплексный канал коммуникации поверх простого TCP-соединения. Проще говоря, эта технология позволяет установить связь между клиентом и сервером с минимальными затратами, позволяя создавать приложения, использующие все преимущества живого общения. Например, представьте, что вы создаете чат: вам необходимо получать и отправлять данные как можно быстрее, верно? С этим прекрасно справляются веб-сокеты! Вы можете открыть TCP-соединение и держать его открытым сколько потребуется. Веб-сокеты используются в следующих случаях:

* Чаты;
* Многопользовательские игры;
* Совместное редактирование;
* Социальные (новостные) ленты;
* Приложения, работающие на основе местоположения.

Socket.IO - библиотека JavaScript, основанная (написанная поверх) на веб-сокетах… и других технологиях. Она использует веб-сокеты, когда они доступны, или такие технологии, как Flash Socket, AJAX Long Polling, AJAX Multipart Stream, когда веб-сокеты недоступны.

**AJAX**

Ajax ( Asynchronous Javascript and XML - «асинхронный JavaScript и XML») - подход к построению интерактивных пользовательских интерфейсов веб-приложений, заключающийся в «фоновом» обмене данными браузера с веб-сервером. В результате при обновлении данных веб-страница не перезагружается полностью, и веб-приложения становятся быстрее и удобнее. По-русски иногда произносится транслитом как «аякс». У аббревиатуры AJAX нет устоявшегося аналога на кириллице.

В классической модели веб-приложения:

* Пользователь заходит на веб-страницу и нажимает на какой-нибудь ее элемент;
* Браузер формирует и отправляет запрос серверу;
* В ответ сервер генерирует совершенно новую веб-страницу и отправляет ее браузеру и т. д. , после чего браузер полностью перезагружает всю страницу.

При использовании AJAX:

* Пользователь заходит на веб-страницу и нажимает на какой-нибудь ее элемент;
* JavaScript определяет, какая информация необходима для обновления страницы;
* Браузер отправляет соответствующий запрос на сервер;
* Сервер возвращает только ту часть документа, на которую пришел запрос;
* Скрипт вносит изменения с учетом полученной информации (без полной перезагрузки страницы).