Интернет:

1. Сеть, в которой использует стек протоколов TCP/IP.
2. Службы интернет – сервер + протокол. Протокол, описывающий доступ к этому серверу. Специфический порт: из диапазона хорошо известных портов. Хост интернета.
3. Набор организаций, обеспечивающий работу Internet. ISOC, IETF – выпускают документы, ICANN – регистрируют типы данных MIME, ISO.
4. Документация.

Интернет-ресурсы - *сущности*, которые имеют адрес в сети Интернет.

Программирование в Интернет – один или несколько связанных ресурсов.

Веб-программирование – частный случай программирование в Интернет (разработка службы www).

Асимметричность – сообщения, которые идут от клиента к серверу отличаются от сообщений, которые идут от сервера к клиенту (Request, Response).

Полудуплексный канал – по которому в каждый момент времени сообщения могут перемещаться только в одну сторону.

TCP – дуплексный.

Заголовок – пара ключ значение, которые содержат служебную инфу о запросе или ответе.

2 типа заголовков: стандартные (описаны в HTTP) и кастомные.

Знать по 2 заголовка каждого типа.

DOM – модель представления браузером HTML- страниц (парсинг => объекты, API доступа к этой древовидной структуре).

Document – объект браузера.

Чем характеризуется браузер? – DOM, BOM (windows, localStorage, navigator, history), XMLHttpRequest, CSS (версии), JavaScript, HTML5 API (canvas, work, audio, video, geolocation, filesystem API’s), какие протоколы поддерживает.

Объекты сервера: Request, Response.

Session – сеансовый уровень. Помнит состояние. TCP тоже помнит состояние (потому что должен помнить номер пакета, чтобы расположить в нужной последовательности). UDP не помнит нифига а зочем ему это надо.

Программный объект – имеет id (128 бит) – sessionID, соответствует серии запросов, идентифицирует серию запросов. Запросы в рамках одной сессии (сеанса).

Как определяется конец сеанса? – Session Timeout – определяет максимальный интервал времени между запросами в рамках одной сессии (20-30 мин).

Как определяется начало сеанса? – используется механизм Cookie.

Cookie -фрагмент данных ключ-значение, который м.б. сохранен на стороне клиента по указанию сервера. Set-Cookie: имя-значение, path (шаблон URI). Браузер в качестве ключа берет шаблон URI. Теперь браузер обязан по URI из path вставить заголовок Cookie. Внутри стоит sessionID.

Как сервер определяет: если нет куки – значит начало сессии; если есть куки в них есть sessionID, но он просрочен, то начинается новая сессия, новый Set-Cookie. Сервер может учить куки если надо.

WebAPI = REST API

Контекст – хранит информацию и самом web-приложении, считывается с config, переменных окружения. В разных системах может называться по-разному: ASP – HttpApplication, Java – ServletContext, Nodejs – process. Резидентный объект. В нем есть хранилище (сохраняется пока сервер запущен).

У сессии тоже есть хранилище, в request тоже.

Итого три хранилища: Request (инфа в рамках одного запроса, middleware может считывать и записывать инфу из/в Request), Session (в объекте session, для хранения инфы между запросами), Context (сохранение инфы между сессиями).

В любой новой платформе для разработки web-приложений нужно разобраться как добраться к request, response, session, context.

**Персистентность –** система персистентна, если она может сохранять своё состояние. Например, на другом носителе: сохранение на диск инфы из контекста. Иногда не только контекст, но и сессия. Сохранение перед перезагрузкой и выключением.

Filter = Middleware. Это конвейер обработки request и response. Фильтры создаются на этапе инициализации приложения.

Listener – специальный серверный объект, предназначенный для обработки событий на стороне сервера. В nodejs находятся в process. Загружаются вместе с сервером, в контекст. Какие события? – старт сервера, завершение работы, образование и уничтожение сесии. С помощью listener можно создавать собственные резидентные объекты. Или listener на все запросы.

Кеширование – перемещение данных из менее быстрой памяти с более быструю память.

Кеширование выполняется на стороне сервера. Ресурс может обрабатываться долго. Кешируем response. Не во всех системах допустим кеш.

Cache Manager – приходит запрос, запускает обработку, проходит обработку, запоминает response в хранилище. Запоминает по ключу (самый простой вариант – URL).

Когда запоминаем, мы говорим сколько времени должно храниться. Например, 5 сек. После кеш самоочищается. Это время абсолютное.

Время скользящее – когда отсчитывается время между запросами. Если в течение 5 сек приходит – то сдвигается ещё на 5 сек.

Составные ключи: URI + параметры и их значения, может ещё заголовок и его значение.

Порядок создания объектов сервера:

Контекст => listener => filter. Затем уже сессия и д.р.

SessionTimeout – скользящее время.

Параметры куки: path, expire date.

DOM описан в W3C (там описано все, что касается веба: XML, HTML, JavaScript и т.п.).

Connection: Keep-Alive -> Keep-Alive: --Timeout—

Конвейерная обработка – в рамках постоянного соединения, параллельный запуск нескольких запросов в рамках одного соединения (одного канала).

Пул соединений: сервер сразу должен установить min кол-во соединений. Можно: сделать через Singleton, записать в контекст или листенер. Может расти до max, затем выстраивается в очередь. Ограничивает кол-во соединений.

Веб-сервисы: предназначены для работы другого приложения: REST, RPC.

REST: запрещает кеширование на стороне сервера, потому что балансировщик по какой-то схеме распределяет запросы на несколько экземпляров веб-сервиса. Разрешается сохранение состояния на стороне клиента (Cookie, иногда прямо в URI).

1. Last-Modified. Дата, когда был последний раз изменён этот файл. Посылается сервером. Клиент отправляет If-Modified-Since: сервер сверяет дату изменения файла с это датой в last-modified. Если не изменилось, то 304 Not modified – бери из кеша. Если изменилось, то отправляет снова.
2. Etag. Содержит хеш файла. Говорит клиенту следить за хешем. Клиент: If-None-Math: 304 Not modified.
3. Expired: дата, когда иссекает срок свежести. Если срок не истек – то из кеша, если нет – то делает запрос.
4. Cache-Control. Max-age = время жизни кеша.

AJAX – методология разработки, которая основана на возможности XMLHttpRequest выдавать асинхронные запросы. Следствие: не перегружается страница. XMLHttpRequest не относится к BOM, DOM, но в рамках браузера.

Методология – принципы, правила написания кода.

Метод open (URI + метод, bool async, sync)

Send метод отправляет заявку.

Ответ: в свойство onChange записать ссылку на callback функцию, которая будет вызываться браузером в зависимости от 4 состояний. На 4 стадии можно обрабатывать сам ответ. Можно использовать для progress-bar.

Аутентификация – проверка подлинности идентификатора пользователя (секрет: пароль, отпечаток, фэйс).

Авторизация – процедура проверки прав аутентифицированного пользователя. На сеансовом уровне OSI/ISO.

Шифры ключи – хранить в сессии.

Два способа auth: Basic, Digest.

Authorization, WWW-Authenticate.

Proxy – если аутентификация на прокси серверах.

401 Unathorized.

Общая схема аутентификации.

Basic: Authorization: информация о пользователе. Логин:пароль в base64. Безопасно только в зашифрованном канале. Иначе снифферы и т.п.

Realm – обычно область действия аутентификации.

Браузер: если код 401 + WWW-Authenticate, то браузер вызывает alert, где даёт возможность ввести логин и пароль. Это встроенная реакция браузера на Basic, Digest. Сам браузер будет отправлять запрос с Authorization.

Digest: WWW-Authenticate: Digest realm + параметры для хеширования.

Forms-authentication: нет стандарта. Передаются логин и пароль в виде параметров. Если все в порядке, то 302 Found, Set-Cookie: token = base64. Теперь все запросы идут с Cookie.

TLS: HTTPs – обеспечивает 2 вещи:

1. Возможность аутентификации с помощью сертификатов.
2. Зашифровать канал.

HTTPs между TCP (сокеты) и HTTP.

TLS (Transport Layer Security) – надстройка над TCP – security sockets.

Процедура рукопожатия. Используется сертификат: электронный документ, разработанный ITU-D. Стандарт X509.

Сертификат содержит: версию, издателя, серийный номер, получателя (имя субъекта), перечень доменов, на которые распространяется сертификат, открытый ключ.

Handshake:

Цель: клиент должен убедиться, что сервер тот, за кого себя выдаёт + получить сеансовый ключ для шифрования (симметричный ключ).

ClientHello – запуск процедуры рукопожатия. Устанавливается на сеансовом уровне. Сеанс – серия запросов и ответов, между которым время не больше, чем SessionTimeout. Информация о сеансе хранится в сессии.

Клиент отправляет список шифронаборов (совокупность, в которой описывается, как вырабатывать ключ, какое шифрования, хеширование, сжатие). Сервер отправляет свой шифронабор и сертификат. Клиент проверяет сертификат, берет публичный ключ, генерирует сеансовый ключ, шифрует его с помощью публичного ключа и отправляет серверу. Сервер получает, с помощью приватного ключа расшифровывает и начинает передачу.

Алгоритм Дифии-Хаффмана.

Service-Provider – зарегитрировался в identity и теперь он может проверять токены -> Trust.

Сесии в ASP.NET Core:

Клиент 🡪 Сервер: есть ли Cookie с Session и есть ли такая сессия. Если нет, то создает объект сессии. У него есть свой ID.

Сервер 🡪 Клиент: сервер отправляет клиенту Cookie, в котором указывается новый ID для клиента. Пара ClientId, ServerId записывается в специальную таблицу.

Сеанс оканчивается по SessionTimeout.

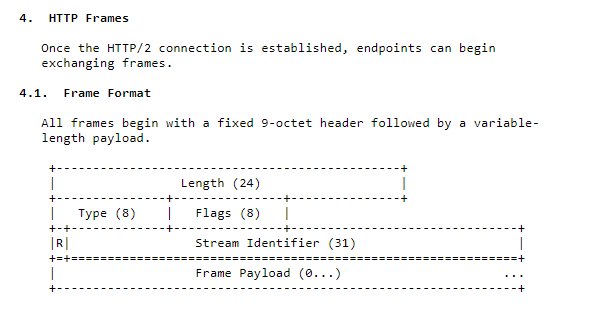
По истечению сессии, куки не меняется, только создается новая сессия с новым серверным ID. Старый объект сессии уничтожается. Таблица обновляется, ставит новый ServerId.

HTTP/2

Поддержка только HTTPs. Без шифрования браузеры не поддерживают. Обратная совместимость с HTTP/1.1

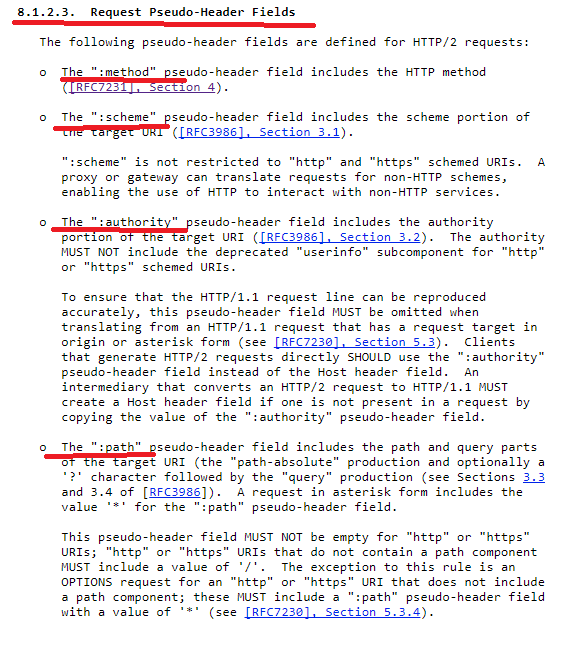
Бинарный формат: информация находится в закодированном виде.

RFC 7540. Сервер может не поддерживать HTTP/2. Обмен фреймами. Типы фреймов: data, headers, priority, rst-stream…



Внутри одного соединения может быть несколько потоков.

Псевдозаголовки – имеют специальное назначение.



Мультиплексирование: в рамках одного соединения много каналов.

Сжатие заголовков (HPACK).

push-сервер – сервер может отсылать данные клиенту (браузеру) для записи в кэш.

WebDAV – на стороне находится удаленный файловый сервер. Он представляет данные в виде файлов, папок и т.д.

WebDAV притворяется, что является HTTP протоколом. Унаследованные HTTP-методы.

Не все файловые серверы поддерживают все методы WebDAV. Например версионность.

GET – скачать файл.

PUT – загрузить файл на сервер.

DELETE – удалить серверный объект.

RFC 3253 – набор методов.

RFC 3648: ORDERPATH.

RFC 3744: ACL

RFC 4437: UPDATEREDIRECTREF

RFC 4791: MKCALENDAR

RFC 5618: PROPFIND – получить свойства серверного объекта в XML-формате.

PROPPATCH – изменить свойства серверного объекта.

MKCOL – создать папку на сервере.

COPY – копировать на сервере.

MOVE – переместить на сервере.

LOCK – заблокировать серверный объект.

UNLOCK – разблокировать серверный объект.

REST:

Представление – это URI, управление ресурсом с помощью глагола (GET, POST, PUT…). Альтернатива REST: RPC.

Нет официального стандарта.

RESTful – описание web-службы, поддерживающей REST в полном объёме.

Использует стандарты HTTP, URI/URL, XML, JSON.

Два основных типа ресурса: коллекция и элемент коллекции. Удаленный ресурс, удаленная коллекция.

Обязательные ограничения:

* Модель клиент-сервер
* Отсутствие состояния на стороне сервера (можно на стороне клиента или в другом сервисе)
* Кеширование на стороне клиента, сервер явно управляет кешированием
* Единообразие интерфейсов (идентификация ресурсов, манипуляция ресурсами через представления, самодостаточные сообщения, HATEOAS)
* Для клиента сервер должен быть представляться конечным
* Код по требованию: допускается выгрузка на клиент апплетов или сценариев для расширения его функциональности.

Кеширование на стороне клиента:

Cache-Control, Etag, Last-Modified.

HTTP 2.0 – Push-Server позволяет отправить некоторые данные на клиент для кеширования.

HATEOAS: Hypermedia As the Engine of Application State.

Для чего? Позволяет получить инфу и настроить клиент. Это дает возможность на стороне сервера менять ссылки. Принцип формата ответов, в которых идут дополнительные ссылки на ресурсы этого сервера.

Проблема CORS: для обхода можно загружать из любого домена данные в виде JSON.

Каким образом задается формат? Обычно в accept, но иногда в URI.

RPC – JSON RPC

Программный объект, у которого есть набор методов. У методов есть имя, набор параметров, возвращаемое значение. Передача по HTTP.

Что нужно определить перед тем, как писать RPC-интерфейс?

* Формат
* Заголовки HTTP для описания данных, Content-Type.
* Свои собственные заголовки.

JSON-RPC предшествовал XML-RPC.

Транспорт: HTTP/HTTPs или TCP Socket.

JSON RFC 4627

Объект запроса:

1. jsonrpc: версия
2. method: имя удаленной процедуры
3. params: параметры метода.
4. id: необязателен, id запроса.

Ответ:

1. jsonrpc: версия
2. result
3. error
4. id, такой же, как в request.

Ошибка:

1. Code: принято отрицательные
2. Message
3. Data: данные об ошибке

Очередь выгребается триггером. База данных для запросов + база данных, сохраняющая запросы на изменение.

Пакет запросов: массив запросов, сервер пачкой и отвечает.

HTTP протокол пролазит через маршрутизаторы.

IndexedDb

Сохранение на клиенте: Cookie, localStorage, URI.

СУБД, встроенная в web-browser, NoSQL, система индекcированных хранилищ, формат данных JSON, JavaScript API, асинхронная работа, транзакционная модель, поддерживается курсор, использует события DOM об уведомлении javascript-приложения (error/success).

Статику нужно держать отдельно.

Nginx: forward proxy, reverse proxy, кеширование балансировка нагрузки.

Зеркалирование: отправить запрос в два места.

WebRTC – набор технологий, предназначенные для потоковой передачи данных. Позволяет выстраивать P2P связь между двумя браузерами. Содержит протоколы RTP, RTSP.

Два браузера: передать видеопоток (с камеры).

Условие: белые ip-адреса. RTSP – прикладной уровень.