Современные веб-браузеры поддерживают несколько способов хранения данных из веб-сайтов на компьютере пользователя - с разрешения пользователя - чтобы потом получать их, когда это необходимо. Это позволяет долгосрочно хранить данные, сохранять сайты или документы для использования без подключения к сети, сохранять пользовательские настройки для вашего сайта и многое другое.

Ранее, мы говорили о разнице между статическими и динамическими сайтами. Большинство современных веб-сайтов являются динамическими - они хранят данные на сервере, используя какую-то базу данных (серверное хранилище), а затем запускают код на стороне сервера чтобы извлечь необходимые данные, вставить их в шаблоны статических страниц и передать полученный HTML-код клиенту для отображения в браузере пользователя.

Хранилище на стороне клиента работает по схожим принципам, но используется по-другому. Оно состоит из API-интерфейсов JavaScript, которые позволяют вам хранить данные на клиенте (то есть на компьютере пользователя), а затем извлекать их при необходимости. Это имеет много разных применений, таких как:

* Персонализация настроек сайта (например, отображение выбранных пользователем виджетов, цветовой схемы или размера шрифта).
* Сохранение предыдущей активности на сайте (например, сохранение содержимого корзины покупок из предыдущего сеанса, запоминание, был ли пользователь ранее авторизован в системе).
* Сохранение данных и ресурсов локально, так что сайт будет быстрее (и, возможно, экономичнее) загружаться или использоваться без подключения к сети.
* Сохранение созданных веб-приложением документов локально для использования в автономном режиме.

Часто, хранилища на сторонах клиента и сервера используются совместно. К примеру, вы должны загрузить из базы данных пакет музыкальных файлов для веб-игры, или музыкальный плеер хранит их в базе данных на стороне клиента, и воспроизводит по мере необходимости.

Пользователь должен будет загрузить музыкальные файлы только один раз - при последующих посещениях они будут извлечены из локальной базы данных.

**Старый подход: Cookie** (HTTP cookie, web cookie, internet cookie, browser cookie)

**Куки** (англ. cookie, букв. - «печенье») - небольшой фрагмент данных, отправленный веб-сервером и хранимый на компьютере пользователя. Веб-клиент (обычно веб-браузер) всякий раз при попытке открыть страницу соответствующего сайта пересылает этот фрагмент данных веб-серверу в составе HTTP-запроса. Применяется для сохранения данных на стороне пользователя, на практике обычно используется для:

* аутентификации пользователя;
* хранения персональных предпочтений и настроек пользователя;
* отслеживания состояния сеанса доступа пользователя;
* сбора статистики о пользователях.

Поддержки браузерами cookie (прием, сохранение и последующая пересылка серверу сохранённых cookie) требуют многие сайты с ограничениями доступа, большинство интернет-магазинов. Настройка оформления и поведения многих веб-сайтов по индивидуальным предпочтениям пользователя тоже основана на cookie.

Cookie легко перехватить и подменить (например, для получения доступа к учетной записи), если пользователь использует нешифрованное соединение с сервером. В группе риска пользователи, выходящие в интернет при помощи публичных точек доступа Wi-Fi и не использующие при этом таких механизмов, как SSL и TLS. Шифрование позволяет также решить и другие проблемы, связанные с безопасностью передаваемых данных.

Большинство современных браузеров позволяет пользователям выбрать - принимать cookie или нет, но их отключение делает невозможной работу с некоторыми сайтами. Кроме того, по законам некоторых стран (например, согласно постановлению Евросоюза от 2016 года, см. общий регламент по защите данных) сайты должны в обязательном порядке запрашивать согласие пользователя перед установкой cookie.

**Назначение**

Cookie используются веб-серверами для идентификации пользователей и хранения данных о них.

К примеру, если вход на сайт осуществляется при помощи cookie, то, после ввода пользователем своих данных на странице входа, cookie позволяют серверу запомнить, что пользователь уже идентифицирован и ему разрешен доступ к соответствующим услугам и операциям.

Многие сайты также используют cookie для сохранения настроек пользователя. Эти настройки могут использоваться для персонализации, которая включает в себя выбор оформления и функциональности. Например, Википедия позволяет авторизованным пользователям выбрать дизайн сайта. Поисковая система Google позволяет пользователям (в том числе и не зарегистрированным в ней) выбрать количество результатов поиска, отображаемых на одной странице.

Cookie также используются для отслеживания действий пользователей на сайте. Как правило, это делается с целью сбора статистики, а рекламные компании на основе такой статистики формируют анонимные профили пользователей для более точного нацеливания рекламы

**Типы cookie**

* **Сессионные cookie**: Сессионные cookie, также известные как временные cookie, существуют только во временной памяти, пока пользователь находится на странице веб-сайта. Браузеры обычно удаляют сессионные cookie после того, как пользователь закрывает окно браузера. В отличие от других типов cookie, сессионные cookie не имеют истечения срока действия, и поэтому браузеры понимают их как сессионные.
* **Постоянные cookie**: Вместо того, чтобы удаляться после закрытия браузера, как это делают сессионные cookie, постоянные cookie-файлы удаляются в определенную дату или через определённый промежуток времени. Это означает, что информация о cookie будет передаваться на сервер каждый раз, когда пользователь посещает веб-сайт, которому эти cookie принадлежат. По этой причине постоянные cookie иногда называются следящие cookie, поскольку они могут использоваться рекламодателями для записи о предпочтениях пользователя в течение длительного периода времени. Однако, они также могут использоваться и в «мирных» целях, например, чтобы избежать повторного ввода данных при каждом посещении сайта.
* **Сторонние cookie**: Обычно атрибут домена cookie совпадает с доменом, который отображается в адресной строке веб-браузера. Это называется первый файл cookie. Однако сторонний файл cookie принадлежит домену, отличному от того, который указан в адресной строке. Этот тип файлов cookie обычно появляется, когда веб-страницы содержат контент с внешних веб-сайтов, например, рекламные баннеры. Это открывает возможности для отслеживания истории посещений пользователя и часто используется рекламодателями для предоставления релевантной рекламы каждому пользователю. В качестве примера предположим, что пользователь посещает www.example.org. Этот веб-сайт содержит рекламу от ad.foxytracking.com, которая при загрузке устанавливает файл cookie, принадлежащий домену рекламы (ad.foxytracking.com). Затем пользователь посещает другой веб-сайт www.foo.com, который также содержит рекламу от ad.foxytracking.com и устанавливает файл cookie, принадлежащий этому домену (ad.foxytracking.com). В конце концов, оба этих cookie будут отправлены рекламодателю при загрузке их рекламы или посещении их веб-сайта. Затем рекламодатель может использовать эти cookie для создания истории просмотров пользователя на всех веб-сайтах, на которых размещена реклама этого рекламодателя. По состоянию на 2014 год некоторые веб-сайты устанавливали cookie для чтения более чем на 100 сторонних доменах. В среднем на одном веб-сайте было установлено 10 файлов cookie, при этом максимальное количество файлов cookie (как для сторонних, так и для третьих сторон) может превышать 800. Большинство современных веб-браузеров содержит настройки конфиденциальности, которые могут блокировать сторонние cookie.
* **Супер-cookie**: Супер-cookie - это cookie-файл с источником домена верхнего уровня (например, .ru) или общедоступным суффиксом (например, .co.uk). Обычные cookie, напротив, имеют происхождение от конкретного доменного имени, например example.com. Супер-cookie могут быть потенциальной проблемой безопасности и поэтому часто блокируются веб-браузерами. Если браузер разблокирует вредоносный веб-сайт, злоумышленник может установить супер-cookie и потенциально нарушить или выдать себя за законные запросы пользователей на другой веб-сайт, который использует тот же домен верхнего уровня или общедоступный суффикс, что и вредоносный веб-сайт. Например, супер-cookie с происхождением .com может злонамеренно повлиять на запрос к example.com, даже если файл cookie не был создан с сайта example.com. Это может быть использовано для подделки логинов или изменения информации пользователя. Публичный список суффиксов помогает снизить риск, который представляют супер-cookie. Публичный список суффиксов - это инициатива кросс-вендоров, целью которого является предоставление точного и актуального списка суффиксов доменных имен. В старых версиях браузеров может отсутствовать актуальный список, и поэтому они будут уязвимы для супер-cookie из определённых доменов. Термин «supercookie» (супер-cookie) иногда используется для отслеживания технологий, которые не используют файлы cookie HTTP. В августе 2011 года на веб-сайтах Microsoft были обнаружены два таких механизма «супер-cookie»: синхронизация файлов cookie, которая порождает cookie MUID (уникальный идентификатор машины), и cookie ETag. Из-за внимания средств массовой информации Microsoft позже отключила этот код.
* **Зомби-cookie**: Поскольку cookie можно очень легко удалить из браузера, программисты ищут способы идентифицировать пользователей даже после полной очистки истории браузера. Одним из таких решений являются зомби-cookie (или evercookie, или persistent cookie) - неудаляемые или трудно удаляемые cookie, которые можно восстановить в браузере с помощью JavaScript. Это возможно потому, что для хранения куки сайт одновременно использует все доступные хранилища браузера (HTTP ETag, Session Storage, Local Storage, Indexed DB), в том числе и хранилища приложений, таких как Flash Player (Local Shared Objects), Microsoft Silverlight (Isolated Storage) и Java (Java persistence API). Когда программа обнаруживает отсутствие в браузере cookie-файла, информация о котором присутствует в других хранилищах, она тут же восстанавливает его на место и тем самым идентифицирует пользователя для сайта.

**Работа cookie**

Как и любой другой HTTP-заголовок, cookie должны передаваться в браузер до того, как будут переданы какие-либо другие данные, включая пустые строки и пробельные символы (это ограничение HTTP-протокола).

Установка cookie: Запрашивая страницу, браузер отправляет веб-серверу короткий текст с HTTP-запросом. Например, для доступа к странице http://www.example.org/index.html браузер отправляет на сервер www.example.org следующий запрос:

* GET /index.html HTTP/1.1
* Host: www.example.org

Сервер отвечает, отправляя запрашиваемую страницу вместе с текстом, содержащим HTTP-ответ. Там может содержаться указание браузеру сохранить cookie:

* HTTP/1.1 200 OK
* Content-type: text/html
* Set-Cookie: name=value

Строка Set-cookie отправляется лишь тогда, когда сервер желает, чтобы браузер сохранил cookie. В этом случае, если cookie поддерживаются браузером и их приём включен, браузер запоминает строку name=value (имя = значение) и отправляет её обратно серверу с каждым последующим запросом. Например, при запросе следующей страницы http://www.example.org/spec.html браузер пошлёт серверу www.examle.org следующий запрос:

* GET /spec.html HTTP/1.1
* Host: www.example.org
* Cookie: name=value
* Accept: */*

Этот запрос отличается от первого запроса тем, что содержит строку, которую сервер отправил браузеру ранее. Таким образом, сервер узна́ет, что этот запрос связан с предыдущим. Сервер отвечает, отправляя запрашиваемую страницу и, возможно, добавив новые cookie.

Значение cookie может быть изменено сервером путем отправления новых строк Set-Cookie: name=new\_value. После этого браузер заменяет старое cookie с тем же name на новую строку.

Cookie также могут устанавливаться программами на языках типа JavaScript, встроенными в текст страниц, или аналогичными скриптами, работающими в браузере. В JavaScript для этого используется свойство cookie объекта document - document.cookie. Например, document.cookie="temperature=20" создаст cookie под именем «temperature» и значением 20.

**Аутентификация**

Cookie могут использоваться сервером для опознания ранее аутентифицированных пользователей. Это происходит следующим образом:

* Пользователь вводит имя пользователя и пароль в текстовых полях страницы входа и отправляет их на сервер.
* Сервер получает имя пользователя и пароль, проверяет их и, при их правильности, отправляет страницу успешного входа, прикрепив cookie с неким идентификатором сессии. Эта cookie может быть действительна не только для текущей сессии браузера, но может быть настроена и на длительное хранение.
* Каждый раз, когда пользователь запрашивает страницу с сервера, браузер автоматически отправляет cookie с идентификатором сессии серверу. Сервер проверяет идентификатор по своей базе идентификаторов и, при наличии в базе такого идентификатора, «узнаёт» пользователя.

Этот метод широко используется на многих сайтах, например на Yahoo!, в Википедии и в Facebook'е.

Многие браузеры (в частности Opera, FireFox) путём редактирования свойств cookie могут управлять поведением веб-сайтов. Изменив срок истечения непостоянных (сессионных) cookie, можно, например, получить формально-неограниченную сессию после авторизации на каком-либо сайте. Возможность редактирования cookie стандартными средствами отсутствует в Internet Explorer. Но, воспользовавшись иными механизмами, например, JavaScript, пользователь может изменить cookie-файл. Более того, существует возможность заменить сессионные cookie постоянными (с указанием срока годности).

Однако серверное программное обеспечение может отслеживать такие попытки. Для этого сервер выдаёт cookie на определенный срок и записывает дату окончания cookie у себя или, в зашифрованном виде, в самих cookie, каждый раз, когда пользователь обращается к серверу. Если cookie, присланный браузером, имеет дату годности, отличную от той, что хранится на сервере или содержатся в cookie, значит, имеет место попытка подмены даты годности cookie. Сервер может отреагировать, например, запросив у пользователя повторную авторизацию.

**Тестирование файлов cookie**

Тестирование файлов cookie - это процесс проверки того, работают ли файлы cookie должным образом. При тестировании файлов cookie тестировщикам необходимо проверить статус файла cookie, срок действия файла cookie, доступность файла cookie, ограничения безопасности и т. д.

**Пример чек-листа** для Cookie testing:

* Файлы cookie корректно создаются на диске;
* Поведение при отказе включить куки:
  + Отображается ли соответствующее сообщение для Пользователей, чтобы включить файлы cookie для доступа к сайту?
  + Есть ли обходной путь для доступа к сайту для браузеров с отключенными файлами cookie.
* Работоспособность после удаления файлов cookie;
* Работоспособность после повреждения (путем редактирования) файлов cookie и невозможность входа в систему после редактирования данных для входа;
* Личные или конфиденциальные данные, хранящиеся в файле cookie, находятся в зашифрованном формате;
* Кросс-браузерное тестирование файлов cookie;
* Не должно быть чрезмерного использования файлов cookie.

**Недостатки куки**

Концепция хранения на стороне клиента существует уже давно. С первых дней Интернета, использовали cookies для хранения информации, чтобы персонализировать пользовательский опыт на веб-сайтах. Это самая ранняя форма хранилища на стороне клиента, обычно используемая в Интернете.

Из-за этого возраста существует ряд проблем - как технических, так и с точки зрения пользовательского опыта - связанных с файлами cookie. Эти проблемы настолько значительны, что при первом посещении сайта людям, живущим в Европе, показываются сообщения, информирующие их о том, будут ли они использовать файлы cookie для хранения данных о них. Это связано с частью законодательства Европейского Союза, известного как EU Cookie directive.

Они устарели, у них множество проблем с безопасностью, и они не способны хранить сложные данные. При этом существуют лучшие, более современные, способы хранения более широкого спектра данных на компьютере пользователя.

Единственным преимуществом файлов cookie является то, что они поддерживаются очень старыми браузерами, поэтому, если ваш проект требует, чтобы вы поддерживали устаревшие браузеры (например, Internet Explorer 8 или более ранние версии), файлы cookie могут по-прежнему быть полезными, но для большинства проектов вы не нужно больше прибегать к ним.

Почему по-прежнему создаются новые сайты с использованием файлов cookie? Это происходит главным образом из-за привычек разработчиков, использования старых библиотек, которые всё ещё используют куки-файлы, и наличия множества веб-сайтов, предоставляющих устаревшие справочные и учебные материалы для обучения хранению данных.

**Новый подход: Web Storage и IndexedDB**

Современные браузеры имеют гораздо более простые и эффективные API для хранения данных на стороне клиента, чем при использовании файлов cookie.

* [**The Web Storage API**](https://developer.mozilla.org/ru/docs/Web/API/Web_Storage_API) обеспечивает очень простой синтаксис для хранения и извлечения данных, состоящих из пар 'ключ' : 'значение'. Это полезно, когда вам просто нужно сохранить некоторые простые данные, такие как имя пользователя, вошли ли они в систему, какой цвет использовать для фона экрана и т. д.
* [**The IndexedDB API**](https://developer.mozilla.org/ru/docs/Web/API/IndexedDB_API) обеспечивает браузер полной базой данных для хранения сложных данных. Это может быть использовано для хранения полных наборов записей клиентов и даже до сложных типов данных, таких как аудио или видео файлы.

Интернет-хранилище упрощенно можно рассматривать как усовершенствование куки. Тем не менее, оно отличается от куки в некоторых ключевых направлениях:

* Размер хранилища: интернет-хранилище поддерживает гораздо больше места на диске в сравнении с куки, которому доступно всего 4 Кбайта, что примерно в 1000 раз меньше чем у веб-хранилища (5 Мбайт на домен в Mozilla Firefox, Google Chrome, и Opera, и 10 Мбайт в Internet Explorer);
* Интерфейс на стороне клиента: в отличие от куки, которые могут быть доступны как на сервере, так и на стороне клиента, веб-хранилище попадает исключительно под компетенцию сценариев (скриптов) на стороне клиента. Данные интернет-хранилища не передаются на сервер при каждом запросе HTTP, и веб-сервер не может напрямую записать в интернет-хранилище;
* Локальное хранилище и Сессионное хранилище: интернет-хранилище предлагает две различных области: локальное хранилище и сессионное хранилище, которые различаются по своим объемам и времени жизни. Данные размещаются в отдельное для каждого домена локальное хранилище (оно доступно для всех скриптов из домена, который первоначально добавил данные) и сохраняются после закрытия браузера. Сессия сохраняется по принципу одна страница - одно окно и ограничивается жизнью данного окна, то есть для каждого открытого окна создается новая сессия, которая прекращает свое существование при закрытии окна и не зависит от домена, открывшего её. Сохранение сессии предназначено для предоставления отдельных экземпляров одного и того же веб-приложения для работы в разных окнах, не мешая друг другу. В случае с куки подобное становится крайне затруднительно или даже невозможно;
* Интерфейс и модель данных: интернет-хранилище в настоящее время предоставляет программный интерфейс лучше, чем куки. Интерфейс представляет собой ассоциативный массив модели данных, где ключи и значения являются строками. Дополнительный API для доступа к структурированным данным на основе SQL находится на рассмотрении рабочей группы W3C.