Cookies (или "куки") — это небольшие фрагменты данных, которые веб-сервер отправляет на клиентскую машину (веб-браузер) и которые хранятся на этом устройстве. Веб-приложения используют cookies для различных целей, включая аутентификацию, хранение пользовательских предпочтений, трекинг сессий и многое другое. Давайте подробно рассмотрим, что такое cookies, как они работают и как их использовать во фронтенд-разработке.

Основные концепции

Структура куки: Cookie состоит из имени, значения и ряда опциональных атрибутов:

Name: имя куки.

Value: значение куки.

Domain: домен, для которого будет использоваться куки.

Path: путь на сервере, в котором куки являются действительными.

Expires: дата и время, когда куки истечет.

Max-Age: время (в секундах), по истечении которого куки станет недействительным.

Secure: если установлен, куки передаются только по HTTPS.

HttpOnly: если установлен, куки не доступны через JavaScript (защита от XSS-атак).

Типы cookies:

Сессионные куки: Эти куки существуют только в течение текущей сессии и исчезают, когда пользователь закрывает браузер.

Постоянные куки: Эти куки остаются на устройстве пользователя в течение заданного времени, даже после закрытия браузера.

Как работают куки

Когда вы отправляете HTTP-запрос к серверу, сервер может отправить соответствующую куку в заголовке ответа. В дальнейшем браузер будет автоматически включать эти куки в каждом запросе к тому же домену, если сохраняются соответствующие условия (домен, путь, срок действия и т.д.). Это позволяет серверу "помнить" о пользователе или хранить состояния между запросами.

Работа с куками во фронтенде

Установка куки

В JavaScript вы можете устанавливать куки с помощью свойства document.cookie. Вот пример:

javascript

// Установка куки

document.cookie = "username=JohnDoe; expires=Fri, 31 Dec 2025 23:59:59 GMT; path=/";

Здесь мы устанавливаем куку с именем username, значением JohnDoe, сроком действия до конца 2025 года и для всего домена.

Получение куки

Для получения значения куки, вы можете использовать следующий код:

javascript

function getCookie(name) {

const value = `; ${document.cookie}`;

const parts = value.split(`; ${name}=`);

if (parts.length === 2) return parts.pop().split(';').shift();

}

// Использование функции

const username = getCookie('username');

console.log(username); // 'JohnDoe' (если кука существует)

Удаление куки

Чтобы удалить куку, вы можете установить ее срок действия в прошлом:

javascript

document.cookie = "username=; expires=Thu, 01 Jan 1970 00:00:00 GMT; path=/";

Безопасность куков

HttpOnly: Установка этого флага защитит куки от доступа через JavaScript, что уменьшает риск атак XSS.

Secure: Этот флаг гарантирует, что куки передаются только через HTTPS, защищая их от перехвата.

SameSite: Этот атрибут помогает предотвратить атаки CSRF, ограничивая отправку куки только для запросов с того же сайта.

Заключение

Cookies — это мощный инструмент для хранения информации о пользователе на стороне клиента. Однако важно помнить о безопасности и использовать куки разумно, чтобы защитить данные пользователей. Конечно, существует также альтернатива в виде локального хранилища (Local Storage) и сессионного хранилища (Session Storage), которые могут быть более удобными для определённых задач, но куки остаются стандартным способом аутентификации и управления сессиями в веб-приложениях.

1

/

1

**куки это текстовый документ?**

Куки (или cookies) не являются текстовыми документами в традиционном смысле, как, например, .txt или .doc файлы. Вместо этого это небольшие фрагменты данных, которые содержат текстовую информацию. Вот несколько важных моментов, которые помогут лучше понять, что такое куки и как они устроены:

Что такое куки?

Формат данных: Куки представляют собой текстовые строки, которые содержат пары "имя-значение". Например, кука может выглядеть так: username=JohnDoe. Эта строка состоит из имени куки (username) и ее значения (JohnDoe).

Хранение: Куки хранятся в браузере пользователя в виде простой текстовой информации. Браузеры управляют куками и могут отображать их пользователям в своих настройках.

Не файл: Хотя куки содержат текстовые данные, они не "файлы" в обычном понимании. Куки хранятся как часть HTTP-заголовков и передаются между браузером и сервером с помощью протокола HTTP.

Как работают куки?

Когда сервер отправляет куки в ответ на HTTP-запрос, браузер хранит эти куки для данного домена и пути, и автоматически отправляет их обратно на сервер при каждом последующем запросе к этому домену, если условия (домен, путь, срок действия) соблюдены.

Пример куки

Пример куки может выглядеть так:

text

Set-Cookie: sessionId=abc123; Expires=Wed, 21 Oct 2025 07:28:00 GMT; Path=/; Secure; HttpOnly

Это значит:

Кука с именем sessionId и значением abc123.

Кука истечет 21 октября 2025 года.

Она доступна для всего домена (Path=/).

Она будет передаваться только по безопасному соединению (Secure).

Она не доступна для JavaScript (HttpOnly).

Резюме

Таким образом, куки — это текстовые данные, которые используются для хранения информации о пользователе, но они не являются файлами в привычном понимании. Они работают на уровне протокола HTTP и хранятся в браузере, а не на файловой системе пользователя.

**как куки используют аутентификацию**

Куки часто используются для управления аутентификацией пользователей в веб-приложениях. Этот процесс позволяет веб-серверам "узнать" пользователей, которые уже вошли в систему, и поддерживать их сеансы без необходимости повторной аутентификации. Рассмотрим более подробно, как это работает.

Процесс аутентификации с использованием куки

Вход пользователя:

Когда пользователь вводит свои учетные данные (например, логин и пароль) и отправляет форму входа, приложение проверяет эти данные на сервере.

Если учетные данные верны, сервер создает уникальный идентификатор сессии (например, токен) и отправляет его обратно клиенту в виде куки.

Установка куки:

Сервер устанавливает куки с токеном аутентификации в ответе. Это может выглядеть так:

text

Set-Cookie: sessionId=abc123; Expires=Wed, 21 Oct 2025 07:28:00 GMT; Path=/; Secure; HttpOnly

Здесь sessionId — это имя куки, а abc123 — его значение.

Хранение куки:

Браузер пользователя сохраняет эту куку и будет автоматически отправлять ее с каждым последующим запросом к тому же домену.

Проверка аутентификации:

Каждый раз, когда пользователь делает запрос к защищенному ресурсу (например, страницу профиля, которая требует входа в систему), браузер отправляет куку с идентификатором сессии обратно на сервер.

Сервер проверяет значение куки, сравнивая его с сохраненными сессиями в базе данных. Если идентификатор сессии совпадает, сервер признает пользователя аутентифицированным.

Подтверждение доступа:

Если аутентификация успешна, сервер отправляет запрашиваемые данные обратно пользователю.

Если идентификатор сессии недействителен или истек, сервер может отправить ответ с статусом 401 (Unauthorized) и перенаправить пользователя на страницу входа.

Ротация и истечение токенов

Истечение срока действия:

Куки могут быть настроены на автоматическое истечение через определенное время (например, через 30 минут). Это повышает безопасность, так как уменьшает риск использования сторонними лицами.

Ротация токенов:

Некоторые системы реализуют ротацию токенов: они обновляют значение токена с каждой новой аутентификацией, что помогает предотвратить атаки путем повторного использования старых токенов.

Безопасность

Для повышения безопасности при использовании куки для аутентификации необходимо учитывать некоторые практики:

Secure и HttpOnly:

Устанавливайте флаги Secure и HttpOnly для куки, чтобы защитить их от перехвата через уязвимости (например, XSS или MITM-атаки).

SameSite:

Используйте атрибут SameSite, чтобы ограничить отправку куки только для запросов с того же сайта, что уменьшает риск CSRF-атак.

Обновление токенов:

Периодически обновляйте токены и позволяйте пользователю явно выйти из системы.

Заключение

Куки являются важным инструментом для управления аутентификацией пользователей в веб-приложениях. Они позволяют сохранять состояние входа в систему и упрощают взаимодействие пользователя с приложением, однако разработчики должны учитывать безопасность и применять лучшие практики при их использовании.

**а как куки работает с хранением пользовательских предпочтений**

Куки также широко используются для хранения пользовательских предпочтений на веб-сайтах, что позволяет улучшить пользовательский опыт. Они помогают запоминать настройки и предпочтения, чтобы пользователь не вынужденно вводил их каждый раз при посещении сайта. Вот как это работает на практике:

Как куки хранят пользовательские предпочтения

Сбор предпочтений:

Когда пользователь взаимодействует с веб-сайтом, он может задавать различные предпочтения: язык интерфейса, тема оформления, настройки уведомлений и т. д.

Например, когда пользователь выбирает темную тему оформления сайта, это значение может быть сохранено в куках.

Установка куки:

После того, как пользователь сделал выбор, веб-приложение создает куки с необходимыми данными. Например:

text

Set-Cookie: theme=dark; Expires=Wed, 21 Oct 2025 07:28:00 GMT; Path=/; Secure; HttpOnly

Здесь theme — это имя куки, а dark — значение, которое указывает на выбранную тему.

Хранение куки:

Браузер пользователя сохраняет эту куку. Она будет автоматически отправляться на сервер при последующих запросах к этому же домену.

Применение предпочтений:

При загрузке страницы, веб-приложение проверяет наличие куки с пользовательскими предпочтениями.

Если кука с темой оформления существует и имеет значение dark, то сайт будет загружаться с темной темой.

Это позволяет пользователю избежать повторного выбора предпочтений при каждом посещении сайта.

Примеры применения пользовательских предпочтений через куки

Язык интерфейса:

Если пользователь выбирает язык (например, русский или английский), это значение можно сохранить в куках. При следующем посещении сайта браузер будет отправлять куку с языковыми настройками, чтобы интерфейс отображался на правильном языке.

Настройки отображения:

Предпочтения по отображению контента (например, количество отображаемых элементов на странице, размер шрифта) можно сохранять в куках для индивидуального пользовательского опыта.

Список желаемого и корзина:

Некоторые сайты используют куки, чтобы запомнить товары в корзине или в списке желаемого, даже если пользователь не авторизован. Это позволяет браузеру запоминать выбор пользователя, когда он возвращается на сайт.

Преимущества хранения предпочтений с использованием куки

Удобство для пользователя: Пользователи получают настраиваемый опыт без необходимости повторного ввода своих предпочтений на каждом посещении.

Снижение нагрузки на сервер: Хранение предпочтений на стороне клиента (в куках) уменьшает необходимое количество запросов на сервер для получения и сохранения настроек.

Безопасность и конфиденциальность

Учет безопасности:

При использовании куков для хранения предпочтений важно учитывать безопасность. Например, не храните личные данные или конфиденциальные предпочтения (такие как пароли) в куках.

Согласие пользователя:

В некоторых регионах, согласно законодательству (таким как GDPR в Европе), необходимо запрашивать согласие пользователя на хранение куков. Пользователи должны иметь возможность видеть, какие куки используются и с какой целью.

Управление куками:

Разработчики должны предоставить пользователям возможность управлять куками, позволяя им в любой момент менять или удалять свои предпочтения.

Заключение

Куки являются эффективным способом хранения пользовательских предпочтений на веб-сайтах, что улучшает общий пользовательский опыт и делает взаимодействие более персонализированным. Тем не менее, необходимо помнить о безопасности и конфиденциальности, предоставляя пользователям контроль над своими данными.

**куку настраиваются на backend?**

Да, куки настраиваются и управляются на стороне сервера (backend). Когда веб-приложение генерирует ответ на запрос пользователя, оно может указать, какие куки необходимо установить в браузере клиента. Давай рассмотрим процесс настройки куков на backend подробнее.

Как настраиваются куки на сервере

Создание куки:

Когда сервер обрабатывает запрос, например, при входе пользователя в систему или изменении его настроек, сервер может создать куку с помощью соответствующих заголовков ответа HTTP.

Например, в веб-приложении на Node.js с использованием Express, вы можете установить куки следующим образом:

javascript

res.cookie('username', 'JohnDoe', { maxAge: 900000, httpOnly: true, secure: true });

В этом примере:

username — имя куки,

JohnDoe — значение куки,

{ maxAge: 900000, httpOnly: true, secure: true } — опции куки, где указываются:

maxAge — время жизни куки (в миллисекундах),

httpOnly — если true, куки недоступны из JavaScript (повышает безопасность),

secure — если true, куки будут передаваться только по HTTPS.

Отправка куки в ответе:

Сервер отправляет куки пользователю в заголовках ответа. Это происходит автоматически, когда вы используете метод вроде res.cookie() в Express, но можно также установить заголовок вручную:

javascript

res.setHeader('Set-Cookie', 'username=JohnDoe; Max-Age=900; HttpOnly; Secure');

Получение куки:

Когда пользователь повторно отправляет запрос (например, при переходе на другую страницу), браузер автоматически прикрепляет куки, связанные с данным доменом, к запросу. Сервер может получить их для проверки аутентификации или применения пользовательских предпочтений.

Это можно сделать, например, в Express, считывая куки из req.cookies.

Изменение и удаление куки:

Сервер также может изменять или удалять куки. Для изменения достаточно установить куку с тем же именем. Для удаления можно установить куку с "истекшим" временем:

javascript

res.clearCookie('username'); // Удаляет куку

Резюме

Куки настраиваются и управляются на стороне сервера (backend). Сервер может создавать, изменять и очищать куки, отправляя соответствующие заголовки в ответах на HTTP-запросы. Клиент (браузер) затем хранит эти куки и отправляет их обратно на сервер при последующих запросах, позволяя серверу восстанавливать состояние сеанса, хранить пользовательские предпочтения и выполнять другие задачи, связанные с аутентификацией и персонализацией.

**вся информация находится в одном куки?**

Нет, не вся информация находится в одном куки. Обычно для хранения информации используют несколько различных куков, и это делается по нескольким причинам:

Причины использования нескольких куков

Организация данных:

Разделяя данные на разные куки, вы можете лучше организовать информацию. Например, вы можете иметь один куки для аутентификационных данных (например, идентификатор сессии) и другой для пользовательских настроек (например, тема оформления).

Размер куки:

Каждый куки имеет ограничение на размер (обычно около 4096 байт), а также ограничение на количество куков для одного домена (обычно до 20-50 куков). Поэтому если у вас много данных, может потребоваться использовать несколько куков.

Безопасность:

Хранение разных типов информации в отдельных куках позволяет применять разные уровни безопасности и политики. Например, вы можете установить флаг HttpOnly только для куки с чувствительными данными, такими как токены аутентификации.

Удобство обновления:

Если вам нужно обновить только одну часть информации, например, изменить тему оформления, вы можете изменить только соответствующий куки, не затрагивая другие куки и их данные.

Пример использования нескольких куков

Вот пример возможного использования нескольких куков для хранения разных типов информации:

Аутентификация:

sessionId=abc123; HttpOnly; Secure — кука, хранящая идентификатор сессии пользователя.

Пользовательские предпочтения:

theme=dark; Max-Age=3600 — кука, хранящая выбор темы оформления.

language=en; Max-Age=3600 — кука, определяющая язык интерфейса.

Корзина покупок:

cartItems=item1,item2,item3; Max-Age=3600 — кука, хранящая список товаров в корзине.

Как это работает на сервере

На стороне сервера вы можете устанавливать и управлять несколькими куками. Например, в Node.js с использованием Express это может выглядеть так:

javascript

res.cookie('sessionId', 'abc123', { httpOnly: true, secure: true });

res.cookie('theme', 'dark', { maxAge: 3600000 });

res.cookie('language', 'en', { maxAge: 3600000 });

res.cookie('cartItems', 'item1,item2,item3', { maxAge: 3600000 });

При этом каждый куки будет отправляться обратно на сервер при последующих запросах, в зависимости от домена. Сервер может прочитать их и соответственно обработать данные.

Заключение

Использование нескольких куков позволяет лучше организовать данные, улучшить безопасность, соответствовать ограничениям на размер и количество куков, а также упростить обновление отдельных частей информации. Это делает управление пользовательскими данными более эффективным и безопасным.

**Если к примеру пользователь устанавливает темную тему как это сохраняется в куки, на бекенд не идет эта информация?**

Существует несколько способов сохранения пользовательских предпочтений, таких как выбор темы (например, темной) при помощи куков. В зависимости от реализации, информация о выбранной теме может либо отправляться на сервер (бекенд), либо храниться только на стороне клиента (фронтенд).

Сценарии сохранения темы через куки

Сохранение темы только на клиенте (фронтенд):

В этом подходе, когда пользователь выбирает тему, соответствующее значение просто сохраняется в куках непосредственно с клиентской стороны без отправки информации на сервер. Вот как это может выглядеть в JavaScript:

javascript

function setTheme(theme) {

document.cookie = `theme=${theme}; max-age=3600; path=/`; // Сохранение темы в куки на час

applyTheme(theme); // Функция, которая применяет тему к интерфейсу

}

// Пример использования

setTheme('dark');

В этом случае, когда пользователь выбирает "темную тему", браузер устанавливает куку с именем theme и значением dark, и эта информация не отправляется на сервер.

Сохранение темы на сервере (бекенд):

В альтернативном подходе, информация о выбранной теме может быть отправлена на сервер для сохранения. Это может быть полезно, если вы хотите, чтобы пользовательские настройки сохранялись между устройствами или после выхода из системы.

Например, если вы используете AJAX для отправки запросов на сервер, это может выглядеть так:

javascript

function saveTheme(theme) {

// Отправляем запрос на сервер

fetch('/api/saveTheme', {

method: 'POST',

headers: { 'Content-Type': 'application/json' },

body: JSON.stringify({ theme: theme })

})

.then(response => {

if (response.ok) {

document.cookie = `theme=${theme}; max-age=3600; path=/`;

applyTheme(theme);

}

});

}

// Пример использования

saveTheme('dark');

В этом случае вы отправляете AJAX-запрос на сервер с данными о выбранной теме. Сервер может сохранить ее в базе данных или в куках, а затем при следующем посещении сайта пользователю будет возвращено соответствующее значение.

Выводы

Сохранение только на клиенте: Это быстрее и не требует обращения к серверу, но не позволяет сохранять настройки между разными устройствами или сессиями (если куки не будут сохранены).

Сохранение на сервере: Это лучший подход для приложений, где требуется синхронизация настроек пользователя между разными устройствами, однако он требует дополнительной логики на сервере, а также обработку запросов.

В зависимости от архитектуры вашего приложения и требований к удобству пользователя, вы можете выбрать один из этих подходов или даже комбинировать их.