Раздел
Хранение данных в браузере
Навигация по уроку
Чтение из document.cookie
Запись в document.cookie
path
domain
expires, max-age
secure
samesite
httpOnly
Приложение: Функции для работы с куки
Приложение: Сторонние куки
Приложение: GDPR
Итого
Комментарии
Поделиться
Редактировать на GitHub

Куки, document.cookie

Куки – это небольшие строки данных, которые хранятся непосредственно в браузере. Они являются частью HTTP-протокола, определённого в спецификации RFC 6265.

Куки обычно устанавливаются веб-сервером при помощи заголовка `Set-Cookie`. Затем браузер будет автоматически добавлять их в (почти) каждый запрос на тот же домен при помощи заголовка `Cookie`.

Один из наиболее частых случаев использования куки – это аутентификация:

1. При входе на сайт сервер отсылает в ответ HTTP-заголовок `Set-Cookie` для того, чтобы установить куки со специальным уникальным идентификатором сессии («session identifier»).
2. Во время следующего запроса к этому же домену браузер посылает на сервер HTTP-заголовок `Cookie`.
3. Таким образом, сервер понимает, кто сделал запрос.

Мы также можем получить доступ к куки непосредственно из браузера, используя свойство `document.cookie`.

Куки имеют множество особенностей и тонкостей в использовании, и в этой главе мы подробно с ними разберёмся.

Чтение из document.cookie

Хранит ли ваш браузер какие-то куки с этого сайта? Посмотрим:

```
// На javascript.info мы используем сервис Google Analy
// поэтому какие-то куки должны быть
alert( document.cookie ); // cookie1=value1; cookie2=va
```

Значение `document.cookie` состоит из пар `ключ=значение`, разделённых `;`. Каждая пара представляет собой отдельное куки.

Чтобы найти определённое куки, достаточно разбить строку из `document.cookie` по `;`, и затем найти нужный ключ. Для этого мы можем использовать как регулярные выражения, так и функции для обработки массивов.

Оставим эту задачу читателю для самостоятельного выполнения. Кроме того, в конце этой главы вы найдёте полезные функции для работы с куки.

Запись в document.cookie

Мы можем писать в `document.cookie`. Но это не просто данные, а аксессор (геттер/сеттер). Присваивание обрабатывается особым образом.

Запись в `document.cookie` обновит только упомянутые в ней куки, но при этом не затронет все остальные.

Например, этот вызов установит куки с именем `user` и значением `John`:

```
document.cookie = "user=John"; // обновляем только куки
alert(document.cookie); // показываем все куки
```

Если вы запустите этот код, то, скорее всего, увидите множество куки. Это происходит, потому что операция `document.cookie=` перезапишет не все куки, а лишь куки с вышеупомянутым именем `user`.

Технически, и имя и значение куки могут состоять из любых символов, для правильного форматирования следует использовать встроенную функцию

Раздел

Хранение данных в браузере

Навигация по уроку

Чтение из document.cookie

Запись в document.cookie

path

domain

expires, max-age

secure

samesite

httpOnly

Приложение: Функции для работы с куки

Приложение: Сторонние куки

Приложение: GDPR

Итого

Комментарии

Поделиться

Редактировать на GitHub

`encodeURIComponent`:

```
// специальные символы (пробелы), требуется кодирование
let name = "my name";
let value = "John Smith"

// кодирует в my%20name=John%20Smith
document.cookie = encodeURIComponent(name) + '=' + enco

alert(document.cookie); // ...; my%20name=John%20Smith
```

Ограничения

Существует несколько ограничений:

- После `encodeURIComponent` пара `name=value` не должна занимать более 4Кб. Таким образом, мы не можем хранить в куки большие данные.
- Общее количество куки на один домен ограничивается примерно 20+. Точное ограничение зависит от конкретного браузера.

У куки есть ряд настроек, многие из которых важны и должны быть установлены.

Эти настройки указываются после пары `ключ=значение` и отделены друг от друга разделителем `;`, вот так:

```
document.cookie = "user=John; path=/; expires=Tue, 19 J
```

path

- `path=/mypath`

URL-префикс пути, куки будут доступны для страниц под этим путём. Должен быть абсолютным. По умолчанию используется текущий путь.

Если куки установлено с `path=/admin`, то оно будет доступно на страницах `/admin` и `/admin/something`, но не на страницах `/home` или `/adminpage`.

Как правило, указывают в качестве пути корень `path=/`, чтобы наше куки было доступно на всех страницах сайта.

domain

- `domain=site.com`

Домен, на котором доступны наши куки. На практике, однако, есть ограничения – мы не можем указать здесь какой угодно домен.

По умолчанию куки доступно лишь тому домену, который его установил. Так что куки, которые были установлены сайтом `site.com`, не будут доступны на сайте `other.com`.

...Но что более интересно, мы не сможем получить эти куки на поддомене `forum.site.com`!

```
// на site.com
document.cookie = "user=John"

// на forum.site.com
alert(document.cookie); // нет user
```

Нет способа сделать куки доступным на другом домене 2-го уровня, так что `other.com` никогда не получит куки, установленное сайтом `site.com`.

Раздел

Хранение данных в браузере

Навигация по уроку

Чтение из document.cookie

Запись в document.cookie

path

domain

expires, max-age

secure

samesite

httpOnly

Приложение: Функции для работы с куки

Приложение: Сторонние куки

Приложение: GDPR

Итого

Комментарии

Поделиться

Редактировать на GitHub

Это ограничение безопасности, чтобы мы могли хранить в куки конфиденциальные данные, предназначенные только для одного сайта.

...Однако, если мы всё же хотим дать поддоменам типа `forum.site.com` доступ к куки, это можно сделать. Достаточно при установке куки на сайте `site.com` в качестве значения опции `domain` указать корневой домен: `domain=site.com`:

```
// находясь на странице site.com
// сделаем куки доступным для всех поддоменов *.site.co
document.cookie = "user=John; domain=site.com"

// позже

// на forum.site.com
alert(document.cookie); // есть куки user=John
```

По историческим причинам установка `domain=.site.com` (с точкой перед `site.com`) также работает и разрешает доступ к куки для поддоменов. Это старая запись, но можно использовать и её, если нужно, чтобы поддерживались очень старые браузеры.

Таким образом, опция `domain` позволяет нам разрешить доступ к куки для поддоменов.

expires, max-age

По умолчанию, если куки не имеют ни одного из этих параметров, то они удалятся при закрытии браузера. Такие куки называются сессионными («session cookies»).

Чтобы помочь куки «пережить» закрытие браузера, мы можем установить значение опций `expires` или `max-age`.

- **`expires=Tue, 19 Jan 2038 03:14:07 GMT`**

Дата истечения срока действия куки, когда браузер удалит его автоматически.

Дата должна быть точно в этом формате, во временной зоне GMT. Мы можем использовать `date.toUTCString`, чтобы получить правильную дату. Например, мы можем установить срок действия куки на 1 день.

```
// +1 день от текущей даты
let date = new Date(Date.now() + 86400e3);
date = date.toUTCString();
document.cookie = "user=John; expires=" + date;
```

Если мы установим в `expires` прошедшую дату, то куки будет удалено.

- **`max-age=3600`**

Альтернатива `expires`, определяет срок действия куки в секундах с текущего момента.

Если задан ноль или отрицательное значение, то куки будет удалено:

```
// куки будет удалено через 1 час
document.cookie = "user=John; max-age=3600";

// удалим куки (срок действия истекает прямо сейчас)
document.cookie = "user=John; max-age=0";
```

secure

- **`secure`**

Куки следует передавать только по HTTPS-протоколу.

Раздел

Хранение данных в браузере

Навигация по уроку

Чтение из document.cookie

Запись в document.cookie

path

domain

expires, max-age

secure

samesite

httpOnly

Приложение: Функции для работы с куки

Приложение: Сторонние куки

Приложение: GDPR

Итого

Комментарии

Поделиться

Редактировать на GitHub

По умолчанию куки, установленные сайтом `http://site.com`, также будут доступны на сайте `https://site.com` и наоборот.

То есть, куки, по умолчанию, опираются на доменное имя, они не обращают внимания на протоколы.

С этой настройкой, если куки будет установлено на сайте `https://site.com`, то оно не будет доступно на том же сайте с протоколом HTTP, как `http://site.com`. Таким образом, если в куки хранится конфиденциальная информация, которую не следует передавать по незашифрованному протоколу HTTP, то нужно установить этот флаг.

```
// предполагается, что сейчас мы на https://
// установим опцию secure для куки (куки доступно тольк
document.cookie = "user=John; secure";
```

samesite

Это ещё одна настройка безопасности, применяется для защиты от так называемой XSRF-атаки (межсайтовая подделка запроса).

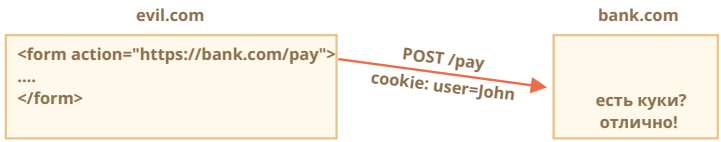
Чтобы понять, как настройка работает и где может быть полезной, посмотрим на XSRF-атаки.

Атака XSRF

Представьте, вы авторизовались на сайте `bank.com`. То есть: у вас есть куки для аутентификации с этого сайта. Ваш браузер отправляет его на сайт `bank.com` с каждым запросом, чтобы сервер этого сайта узнавал вас и выполнял все конфиденциальные финансовые операции.

Теперь, просматривая веб-страницу в другом окне, вы случайно переходите на сайт `evil.com`, который автоматически отправляет форму `<form action="https://bank.com/pay">` на сайт `bank.com` с заполненными полями, которые инициируют транзакцию на счёт хакера.

Браузер посылает куки при каждом посещении `bank.com`, даже если форма была отправлена с `evil.com`. Таким образом, банк узнает вас и выполнит платёж.



Такая атака называется межсайтовая подделка запроса (или Cross-Site Request Forgery, XSRF).

Конечно же, в реальной жизни банки защищены от такой атаки. Во всех сгенерированных сайтом `bank.com` формах есть специальное поле, так называемый «токен защиты от xsrf», который вредоносная страница не может ни сгенерировать, ни каким-либо образом извлечь из удалённой страницы (она может отправить форму туда, но не может получить данные обратно). И сайт `bank.com` при получении формы проверяет его наличие.

Но такая защита требует усилий на её реализацию: нам нужно убедиться, что в каждой форме есть поле с токеном, также мы должны проверить все запросы.

Настройка samesite

Параметр куки `samesite` предоставляет ещё один способ защиты от таких атак, который (теоретически) не должен требовать «токенов защиты xsrf».

У него есть два возможных значения:

- **`samesite=strict` (или, что то же самое, `samesite` без значения)**

Куки с `samesite=strict` никогда не отправятся, если пользователь пришёл не с этого же сайта.

Раздел

Хранение данных в браузере

Навигация по уроку

Чтение из document.cookie

Запись в document.cookie

path

domain

expires, max-age

secure

samesite

httpOnly

Приложение: Функции для работы с куки

Приложение: Сторонние куки

Приложение: GDPR

Итого

Комментарии

Поделиться

Редактировать на GitHub

Другими словами, если пользователь переходит по ссылке из почты, отправляет форму с `evil.com` или выполняет любую другую операцию, происходящую с другого домена, то куки не отправляется.

Если куки имеют настройку `samesite`, то атака XSRF не имеет шансов на успех, потому что отправка с сайта `evil.com` происходит без куки. Таким образом, сайт `bank.com` не распознает пользователя и не произведёт платёж.

Защита довольно надёжная. Куки с настройкой `samesite` будет отправлено только в том случае, если операции происходят с сайта `bank.com`, например отправка формы сделана со страницы на `bank.com`.

Хотя есть небольшие неудобства.

Когда пользователь перейдёт по ссылке на `bank.com`, например из своих заметок, он будет удивлён, что сайт `bank.com` не узнал его. Действительно, куки с `samesite=strict` в этом случае не отправляется.

Мы могли бы обойти это ограничение, используя два куки: одно куки для «общего узнавания», только для того, чтобы поздороваться: «Привет, Джон», и другое куки для операций изменения данных с `samesite=strict`. Тогда пользователь, пришедший на сайт, увидит приветствие, но платежи нужно инициировать с сайта банка, чтобы отправилось второе куки.

- **`samesite=lax`**

Это более мягкий вариант, который также защищает от XSRF и при этом не портит впечатление от использования сайта.

Режим Lax так же, как и `strict`, запрещает браузеру отправлять куки, когда запрос происходит не с сайта, но добавляет одно исключение.

Куки с `samesite=lax` отправляется, если два этих условия верны:

1. Используются безопасные HTTP-методы (например, GET, но не POST).

   Полный список безопасных HTTP-методов можно посмотреть в спецификации RFC7231. По сути, безопасными считаются методы, которые обычно используются для чтения, но не для записи данных. Они не должны выполнять никаких операций на изменение данных. Переход по ссылке является всегда GET-методом, то есть безопасным.

2. Операция осуществляет навигацию верхнего уровня (изменяет URL в адресной строке браузера).

   Обычно это так, но если навигация выполняется в `<iframe>`, то это не верхний уровень. Кроме того, JavaScript-методы для сетевых запросов не выполняют никакой навигации, поэтому они не подходят.

Таким образом, режим `samesite=lax`, позволяет самой распространённой операции «переход по ссылке» передавать куки. Например, открытие сайта из заметок удовлетворяет этим условиям.

Но что-то более сложное, например, сетевой запрос с другого сайта или отправка формы, теряет куки.

Если это вам походит, то добавление `samesite=lax`, скорее всего, не испортит впечатление пользователей от работы с сайтом и добавит защиту.

В целом, `samesite` отличная настройка, но у неё есть важный недостаток:

- `samesite` игнорируется (не поддерживается) старыми браузерами, выпущенными до 2017 года и ранее.

Так что, если мы будем полагаться исключительно на `samesite`, то старые браузеры будут уязвимы.

Но мы, безусловно, можем использовать `samesite` вместе с другими методами защиты, такими как XSRF-токены, чтобы добавить дополнительный слой защиты, а затем, в будущем, когда старые браузеры полностью исчезнут, мы, вероятно, сможем полностью удалить XSRF-токены.

httpOnly

Эта настройка не имеет ничего общего с JavaScript, но мы должны упомянуть её для полноты изложения.

Раздел

Хранение данных в браузере

Навигация по уроку

Чтение из document.cookie

Запись в document.cookie

path

domain

expires, max-age

secure

samesite

httpOnly

Приложение: Функции для работы с куки

Приложение: Сторонние куки

Приложение: GDPR

Итого

Комментарии

Поделиться

Редактировать на GitHub

Веб-сервер использует заголовок `Set-Cookie` для установки куки. И он может установить настройку `httpOnly`.

Эта настройка запрещает любой доступ к куки из JavaScript. Мы не можем видеть такое куки или манипулировать им с помощью `document.cookie`.

Эта настройка используется в качестве меры предосторожности от определённых атак, когда хакер внедряет свой собственный JavaScript-код в страницу и ждёт, когда пользователь посетит её. Это вообще не должно быть возможным, хакер не должен быть в состоянии внедрить свой код на ваш сайт, но могут быть ошибки, которые позволят хакеру сделать это.

Обычно, если такое происходит, и пользователь заходит на страницу с JavaScript-кодом хакера, то этот код выполняется и получает доступ к `document.cookie`, и тем самым к куки пользователя, которые содержат аутентификационную информацию. Это плохо.

Но если куки имеет настройку `httpOnly`, то `document.cookie` не видит его, поэтому такое куки защищено.

Приложение: Функции для работы с куки

Вот небольшой набор функций для работы с куки, более удобных, чем ручная модификация `document.cookie`.

Для этого существует множество библиотек, так что они, скорее, в демонстрационных целях. Но при этом полностью рабочие.

getCookie(name)

Самый короткий способ получить доступ к куки – это использовать регулярные выражения.

Функция `getCookie(name)` возвращает куки с указанным `name`:

```
// возвращает куки с указанным name,
// или undefined, если ничего не найдено
function getCookie(name) {
  let matches = document.cookie.match(new RegExp(
    "(?:^|; )" + name.replace(/([\.$?*|{}\(\)\[\]\\\/\+
  ));
  return matches ? decodeURIComponent(matches[1]) : und
}
```

Здесь `new RegExp` генерируется динамически, чтобы находить `; name=<value>`.

Обратите внимание, значение куки кодируется, поэтому `getCookie` использует встроенную функцию `decodeURIComponent` для декодирования.

setCookie(name, value, options)

Устанавливает куки с именем `name` и значением `value`, с настройкой `path=/` по умолчанию (можно изменить, чтобы добавить другие значения по умолчанию):

```
function setCookie(name, value, options = {}) {

  options = {
    path: '/',
    // при необходимости добавьте другие значения по ум
    ...options
  };

  if (options.expires instanceof Date) {
    options.expires = options.expires.toUTCString();
  }

  let updatedCookie = encodeURIComponent(name) + "=" +

```

Раздел

Хранение данных в браузере

Навигация по уроку

Чтение из document.cookie

Запись в document.cookie

path

domain

expires, max-age

secure

samesite

httpOnly

Приложение: Функции для работы с куки

Приложение: Сторонние куки

Приложение: GDPR

Итого

Комментарии

Поделиться

Редактировать на GitHub

```
  for (let optionKey in options) {
    updatedCookie += "; " + optionKey;
    let optionValue = options[optionKey];
    if (optionValue !== true) {
      updatedCookie += "=" + optionValue;
    }
  }

  document.cookie = updatedCookie;
}

// Пример использования:
setCookie('user', 'John', {secure: true, 'max-age': 360
```

deleteCookie(name)

Чтобы удалить куки, мы можем установить отрицательную дату истечения срока действия:

```
function deleteCookie(name) {
  setCookie(name, "", {
    'max-age': -1
  })
}
```

Операции обновления или удаления куки должны использовать те же путь и домен

Обратите внимание: когда мы обновляем или удаляем куки, нам следует использовать только такие же настройки пути и домена, как при установке куки.

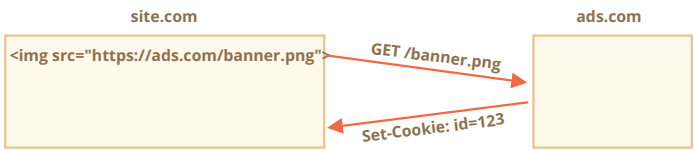
Всё вместе: cookie.js.

Приложение: Сторонние куки

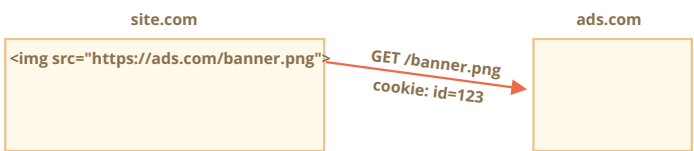
Куки называются сторонними, если они размещены с домена, отличающегося от страницы, которую посещает пользователь.

Например:

1. Страница `site.com` загружает баннер с другого сайта: `<img src="https://ads.com/banner.png">`.
2. Вместе с баннером удалённый сервер `ads.com` может установить заголовок `Set-Cookie` с куки, например, `id=1234`. Такие куки создаются с домена `ads.com` и будут видны только на сайте `ads.com`:



3. В следующий раз при доступе к `ads.com` удалённый сервер получит куки `id` и распознает пользователя:



Раздел

Хранение данных в браузере

Навигация по уроку

Чтение из document.cookie

Запись в document.cookie

path

domain

expires, max-age

secure

samesite

httpOnly

Приложение: Функции для работы с куки

Приложение: Сторонние куки

Приложение: GDPR

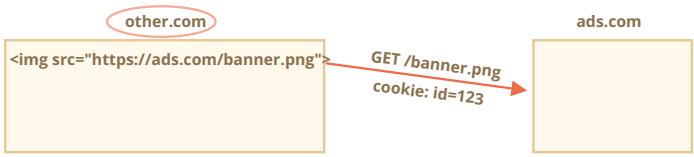
Итого

Комментарии

Поделиться

Редактировать на GitHub

4. Что ещё более важно, когда пользователь переходит с `site.com` на другой сайт `other.com`, на котором тоже есть баннер, то `ads.com` получит куки, так как они принадлежат `ads.com`, таким образом `ads.com` распознает пользователя и может отслеживать его перемещения между сайтами:



Сторонние куки в силу своей специфики обычно используются для целей отслеживания посещаемых пользователем страниц и показа рекламы. Они привязаны к исходному домену, поэтому ads.com может отслеживать одного и того же пользователя на разных сайтах, если оттуда идёт обращение к нему.

Естественно, некоторым пользователям не нравится, когда их отслеживают, поэтому браузеры позволяют отключать такие куки.

Кроме того, некоторые современные браузеры используют специальные политики для таких куки:

- Safari вообще не разрешает сторонние куки.
- У Firefox есть «чёрный список» сторонних доменов, чьи сторонние куки он блокирует.

> **На заметку:**
>
> Если мы загружаем скрипт со стороннего домена, например `<script src="https://google-analytics.com/analytics.js">`, и этот скрипт использует `document.cookie`, чтобы установить куки, то такое куки не является сторонним.
>
> Если скрипт устанавливает куки, то нет разницы откуда был загружен скрипт – куки принадлежит домену текущей веб-страницы.

Приложение: GDPR

Эта тема вообще не связана с JavaScript, но следует её иметь в виду при установке куки.

В Европе существует законодательство под названием GDPR, которое устанавливает для сайтов ряд правил, обеспечивающих конфиденциальность пользователей. И одним из таких правил является требование явного разрешения от пользователя на использование отслеживающих куки.

Обратите внимание, это относится только к куки, используемым для отслеживания/идентификации/авторизации.

То есть, если мы установим куки, которые просто сохраняют некоторую информацию, но не отслеживают и не идентифицируют пользователя, то мы свободны от этого правила.

Но если мы собираемся установить куки с информацией об аутентификации или с идентификатором отслеживания, то пользователь должен явно разрешить это.

Есть два основных варианта как сайты следуют GDPR. Вы наверняка уже видели их в сети:

1. Если сайт хочет установить куки для отслеживания только для авторизованных пользователей.

   То в регистрационной форме должен быть установлен флажок «принять политику конфиденциальности» (которая определяет, как используются куки), пользователь должен установить его, и только тогда сайт сможет использовать авторизационные куки.

Раздел

Хранение данных в браузере

Навигация по уроку

Чтение из document.cookie

Запись в document.cookie

path

domain

expires, max-age

secure

samesite

httpOnly

Приложение: Функции для работы с куки

Приложение: Сторонние куки

Приложение: GDPR

Итого

Комментарии

Поделиться

Редактировать на GitHub

2. Если сайт хочет установить куки для отслеживания всем пользователям.

   Чтобы сделать это законно, сайт показывает модально окно для пользователей, которые зашли в первый раз, и требует от них согласие на использование куки. Затем сайт может установить такие куки и показать пользователю содержимое страницы. Хотя это создаёт неудобства для новых посетителей – никому не нравится наблюдать модальные окна вместо контента. Но GDPR в данной ситуации требует явного согласия пользователя.

GDPR касается не только куки, но и других вопросов, связанных с конфиденциальностью, которые выходят за рамки материала этой главы.

Итого

`document.cookie` предоставляет доступ к куки

- операция записи изменяет только то куки, которое было указано.
- имя и значение куки должны быть закодированы.
- одно куки вмещает до 4kb данных, разрешается более 20 куки на сайт (зависит от браузера).

Настройки куки:

- `path=/`, по умолчанию устанавливается текущий путь, делает куки видимым только по указанному пути и ниже.
- `domain=site.com`, по умолчанию куки видно только на текущем домене, если явно указан домен, то куки видно и на поддоменах.
- `expires` или `max-age` устанавливает дату истечения срока действия, без них куки умрёт при закрытии браузера.
- `secure` делает куки доступным только при использовании HTTPS.
- `samesite` запрещает браузеру отправлять куки с запросами, поступающими извне, помогает предотвратить XSRF-атаки.

Дополнительно:

- Сторонние куки могут быть запрещены браузером, например Safari делает это по умолчанию.
- Установка отслеживающих куки пользователям из стран ЕС требует их явного согласия на это в соответствии с законодательством GDPR.

Комментарии

перед тем как писать...